SOLIDISK ADVANCED DISC FILING SYSTEM

1. INTRODUCTION TO THE STL ADFS

The STL ADFS has many improvements over the STL DDFS, especially in the field of file handling (up to ten channels may be open at one time, and each file may be up to 512 MB) and Disc space management (both of the sides of the ADFS disc are treated as a single logical surface). The latter feature is probably the most important, because - as frequent observation shows - most 80 track double-sided discs are hardly ever used on side two.

However, in addition to formatting over 160 tracks, the STL implementation allows you the options of formatting your disc on single sides - with 80 or even 40 tracks. This allows two double-sided drives to be used as four logical drives (as with the original DFS) as well as the use of single-sided 80 or 40 track drives with the ADFS.

File names can now be up to ten characters long. Also, if you try to extend a file, the ADFS will automatically relocate the file elsewhere, so you will never get the message 'Can't Extend'.

The Directory system now has a Tree structure, allowing you to organise the disc logically, i.e. into Topics and Sub-Topics, etc.

Up to 47 entries can be made to each Directory, and they can be files or Sub-Directories. Each Sub-Directory can then accept up to 47 entries, and so on. The total number of entries is only limited by the Disc size. Also you can copy files from one Directory to another Directory, even if they are on the same Disc. Lastly, the ADFS will control both Winchester and Floppy disc drives.

In addition, the STL ADFS has many more advanced features and commands, then the Acorn ADFS (as implemented on the Electron Plus 3 and the Acorn Winchester Disc System). The STL ADFS is designed to work with the STL DFS 2.1 - with the latter providing both complete DFS (and DDFS) facilities and various utilities which are common to both. Some of the added commands are *BACKUP, *DZAP, *FORMAT, *OPEN, *PASSWORD, *VERIFY etc.

This manual should therefore be read in conjunction with that entitled "Solidisk Disk Filing System" (which has a pink cover) - particularly regarding the installation of Floppy Disc Interfaces and of ROM chips. However, when installing both the ADFS and the DFS 2.1 ROMs, the DFS chip should be to the right - in a higher priority socket.

Floppy discs and hard discs that are formatted on the Acorn ADFS, may be used directly on BBC Micro and Electron computers equipped with the Solidisk ADFS hardware and software, and vice-versa. The only exception is when the disc is protected by the STL password option - which can be set on any directory.

2: IMPORTANT CONCEPTS IN THE STL ADFS

2.1 Glossary of new terms

FILE: A file is an assembly of bytes stored in sequence on the disc. Files may be regarded as the leaves of a tree, with each leaf being connected to a branch or Directory by its entry in the Directory.

NAMES: Filenames are similar to those in the old DFS, but names in the ADFS can be up to ten characters long, and can consist of any Alphabetical character, except a space, and any other characters except the # (hash character), the * (asterisk), the $ (dollar), : (colon), . (period), , (comma) or the & (ampersand) character. You are allowed to name both 'files' and directories. If one of these characters were used, then many disc operations would not work properly. E.g. If you have a file called 'FRED' and a file called 'FRED2', and you typed LOAD "FRED*", it would load any file with 'FRED' as the first four characters.

DIRECTORY: A directory is a defined structure, stored on five consecutive sectors on the disc, and containing entries for files or sub-directories. A directory is very similar to a branch in that it can have leaves and other branches connected to itself. It always contains the Disc Address of its PARENT, which is the branch from whence it came.

ROOT DIRECTORY: If you try to go from a Directory to its Parent, then from this Directory to its own Parent, and so on, you will find a Directory which has a Parent, which points to itself. This is called the ROOT directory. It has a fixed position on the disc. (Sector 000002) and has a fixed title - '$' (dollar) - which is why you are not allowed to use the '$' character for a directory name.

PASSWORD: The STL ADFS has a feature that allows you to protect your Directories with an eight letter password. This password is then stored in the Directory in a scrambled form, so access to unauthorised users is made as difficult as possible.

TITLES: Titles are similar to the 'Disc Title' in the old DFS, but can be up to nineteen characters long. Titles can be given to any Directory on the ADFS disc, with the command *TITLE <Title>. Such titles are not used by the system, but are for reference by the user of the system.

DISC ADDRESS: The Disc Address (DA) is a three-byte pointer, that is used by the ADFS to find the records. It is expressed in units of sectors, each of which can contain up to 256 bytes of recorded data. The Disc Address has a range of &1FFFFF sectors, or 512 MBytes. A secondary Disc Address - known as the Logical Unit Number (LUN, with a value of zero to seven) - extends this range even further - to 4 Gigabytes. It is for identifying multiple logical drives within one physical drive.

FREE SPACE MAP: The free space (FS) map, is displayed by *MAP. It occupies the first two sectors of the disc, and consists of a list of Disc Addresses and the sizes of all the free spaces on the Disc.

SASI: (Shugart Associates Standard Interface). All of the SASI hard disc controller boards are made in such a way that they are 'Pin compatible' i.e. one board can be used on several Winchester systems, but some of these boards are capable of more functions than others. The Solidisk XD20-40 controller board is fitted with the Western Digital WD1002-SHD interface chip.

ALTERNATE TRACKS. The XD20-40 Winchester system has six hundred and twelve tracks, but only six hundred are normally used. The remaining tracks are reserved as alternate tracks so, if in the formatting process a normal track is found to be faulty, then the XD20-40 will automatically write an 'illegal access code' on this track, and set a pointer to one of the free reserved tracks. In practice though, you will not be aware of the existence of the reserved tracks.

2.2 THE DRIVE NUMBERS - HARD DISC AND FLOPPY DISC

In the STL DFS, Drive numbers are from 0 to 3 for Floppy discs, and 4 for the Silicon disc, but in the ADFS, Drive numbers from 0 to 7 (or A to H), are possible for either mechanical or Silicon discs.

Normally, the hardware only allows you to use four numbers for mechanical drives, and these are 0, 1, 4 and 5 (or A, B, E and F). The ADFS will check for the presence of the hard disc, by reading location &FC40. If positive, then the Hard disc is given priority over the Floppy discs. The drives are then numbered as follows:-

```
Winchester connected to socket J2 = Drive 0 or A
Winchester connected to socket J3 = Drive 1 or B
Floppy disc configured as drive 0 = Drive 4 or E
Floppy disc configured as drive 1 = Drive 5 or F
```

If the STL Winchster controller card, (the XD20-40) is not connected to your system, then the ADFS will check to see if the WD1770 floppy disc controller (FDC) is present. If it is, the Floppy drives will default to numbers 0 to 3 (or A to D). If the 1770 is not connected, then the filing system will default to the system present in your machine - e.g. to a DFS using an 8271 (FDC) or to cassette tape.

2.3 OBJECTS AND PATHS

The ADFS contains entries for files ('Leaves') and for sub-directories, ('sub-branches') - both of which are referred to as 'Objects'. An object is therefore a record describing a unique entry in the directory. When an Object is stored on to the disc, certain information is also stored with it. This 'Parameter' block will contain the name, the access code, the load address, the execution address, the length of the file, the disc address, and the sequence number. These are called the 'Object Specification' or 'Object spec'.

When a program, or data, is saved on to the disc, they are stored in such a way that they are treated like a leaf on a tree. i.e. each file is connected to a sub-directory, and each directory can be connected to another sub-directory and so on. So when you wish to load a file, or open a file, or to perform a data manipulation, then the ADFS needs to know its 'Object specification'. The ADFS will follow all indications that you supply, to find the Object, and the route it takes is called a path.

With the ADFS - as on a real tree - you could take a pen, and mark a continuous line, from the branch where you are, to another branch, then another branch, and so on, until you reach the 'leaf' or object that you require. Hence the line has to be continuous, passing only over the branches that you specify. Along the path, when the ADFS finds a . (period), it means that it must take a new 'branch', the name to the left of the . (period) is the 'parent' of the name to the right of the . (period), and so on. So this shows that the system is organised in descending (or hierarchical) order.

The path will and at the last 'object' specified, and this is usually the file itself except in the special case of *DIR and *LIB which refer to directories. If, when you specify a path, you include an object in the middle of the path, then you will get the 'Bad path' error message, but if you include the name of a directory that is not found, then you will get the message 'Not found'.

Paths are usually indicated as <*Obspec*>, and the two Asterisks ('*') are to indicate that you are allowed to use Wildcards.

2.4 WILDCARDS

Wildcards in the ADFS are identical to those used in the original DFS, i.e. the # (hash character), represents any one character, whereas the * (asterisk) represents any number of characters. Therefore 'FRED##' could mean the files called 'FRED1' or 'FRED99', but not 'FRED100' - because of non-matching - while '*D*' would mean any file with the letter 'D' present in the file name.

Thus, if you wanted to chain a program called 'DESIGN', in a sub-directory called 'SPRITE', in a directory called 'WORK', on drive '0', you might have to type in:

```
CHAIN":0.$.WORK.SPRITE.DESIGN"
```

But, using wildcards, you might be able to type:

```
CHAIN":0.$.W*.SP*.DES*"
```

In the above example, the ADFS will find, in each instance, the first Directory that contains the specified character. Hence you must take care to supply enough characters to avoid ambiguity in each instance.

Some of the ADFS commands - such as *DESTROY or *COPY - can operate on a collection of objects in one go - e.g.

```
*DESTROY @.*
```

where '@' signifies the Currently Selected Directory and '*' is the multiple wildcard character.

whereupon the computer will display the objects in the CSD and ask:-

'Are you sure (Y/N) ?'

If you answer 'YES' [or 'Y'], any directory that is empty and any file that is not open, and is not locked, will be deleted.

If you enter 'NO' (or anything except 'YES') then they will not be harmed.

2.5 DIRECTORIES AND THE CURRENTLY SELECTED DIRECTORY

The STL DFS 2.1 has a single, large directory that contains only file objects. This directory is divided into pages, each holding up to 31 files. Each page also has a pointer to the next page, or catalogue. So if you have a file called 'FRED251', then the DFS 2.1 will have to search from the first file to the last file, and check to see if the file is found.

This is how the DFS 2.1 directory is organised:

CATALOGUE 1: (pointer to cat' 2)

```
FRED001
FRED002
   -
FRED031
```

CATALOGUE 2: (pointer to cat' 3)

```
????? (cat 1 protected)
FRED032
FRED033
   -
FRED061
```

CATALOGUE 3: (pointer to cat' 4)

```
????? (cat 2 protected)
```

```
FRED062
   -
etc....
```

Multiple directories, linked in a hierarchy, are the powerhouse behind the ADFS. They are treated as branches of a tree, i.e. they may contain other smaller branches or leaves, so that a Directory will consist principally of a list of entries, of different objects.

The is an example of an ADFS Directory - named 'WORK'.

```
 !BOOT        (WR)      Commands     (WR)
 DEMO         (WR)      Header       (WR)
 Part1        (WR)      Part2        (WR)
 Solicomms    (DLR)     Solimon      (DLR)
 Spricode     (WR)      Sprite       (DLR)
 STLToolkit   (DLR)     z80          (WR)
```

So if you wish to use the program called 'DESIGN', then you could enter something like this:

```
 CHAIN"$.WORK.SPRITE.DESIGN"
```

To the beginner, it might appear difficult to access a file quickly, but in reality things are much simpler. The ADFS lends itself to logical organisation of your files. You could re-group them all into a few topics, such as WORDPROCESSOR, DATABASE, SPREADSHEET, GRAPHICS, GAMES, PROGRAMMING, etc. These topics could then be sub-divided into smaller topics, and so on. So if you need to work on the 'SPRITE.DESIGN' program, then you could start by selecting SPRITE as your Currently Selected Directory.

When the ADFS is first entered, the ROOT directory ('$') is selected by default.

You may specify any directory as the Currently Specified Directory (CSD) by typing *DIR <*Object specification*> - where this particular 'Object specification' is known as a 'path', and must end in a directory, not in a file.

Issuing *DIR alone means the CSD becomes the ROOT directory, by default.

The pathname usually starts from - i.e. is relative to - the Currently Selected Directory. This method is called 'Relative Object Referencing'. For example, if the CSD is called 'ACCOUNTS', you could issue:

```
 *DIR 1985.MAY
```

However, you can also reference it to the ROOT directory. For example:

```
 *DIR $.ACCOUNTS.1985.MAY
```

Without the 'Relative Object Referencing', most of the programs which refer to files that they wish to open, would not work. The Currently Selected Directory concept will let you run a program that was written for the original DFS, on the ADFS, without changing it.

If you need to go frequently to the program called 'DESIGN', it is possible to build up a !BOOT file in the ROOT directory, to set the CSD for you, and then CHAIN the program, i.e.

```
 *BUILD !BOOT

 0001 *DIR WORK.SPRITE
 0002 CHAIN "DESIGN"
 0003 <Escape>
```

The ADFS will keep track of the path leading to the CSD, and will let you go back up any number of levels or directories, by using the '^'

(circumflex) key. This is on the top row of dark keys, fourth from the right.

Example:
If you have previously set the Currently Selected Directory with:

```
*DIR :4.$.BADDEBTS.FRIEND.FRED.JANUARY
```

And then need to go back to the directory called FRIEND, you can type:

```
*DIR ^^^
```

Of course, if you want to go back to the directory $ - the 'root' directory - you can just type *DIR.

2.5.1 DIRECTORY STRUCTURE

A directory is an object containing a list of up to 47 entries, and each entry represents, and describes another object - a sub-directory or a file. Each entry will contain 26 bytes, and these are used as follows:

LOCATION 00	10 bytes	NAME and ACCESS STATUS
LOCATION 0A	4 bytes	LOAD Address (in memory)
LOCATION 0E	4 bytes	EXECUTION Address (in memory)
LOCATION 12	4 bytes	LENGTH (in bytes)
LOCATION 16	4 bytes	DISC Address (in sectors, on the disc)
LOCATION 19	4 bytes	SEQUENCE number
TOTAL:	26 bytes * 47 entries = 1222 bytes	

The remaining 58 bytes are used to store directory name, Title, Password and other object information. The directory is normally loaded into memory from &1200 to &16FF, with the first entry found at location &1205.

2.5.2 DIRECTORY PASSWORD

If you enter:

```
*CDIR MYWORK (<Password>)
```

The directory name will be MYWORK, while the password for this directory is called (<Password>). If a directory is protected by a password, then you can only access it (and hence any of its contents) by entering it with the correct password, i.e.

```
*DIR MYWORK (<Password>)

*DIR MYWORK      will result in the message 'Password Required'
```

A directory password is stored within the directory concerned, and protects a user's domain from being accessible by other users who are sharing the same disc.

In contrast, the Write Protect Password is stored in the FS (Free Space) map, and prevents unauthorised persons from copying from, or writing into, any portion of the disc. Operations such as *DELETE, *RENAME, *COPY, *SAVE etc... will result in the message 'Write protected' - although you can still load and run any program.

2.6 LIBRARIES, THE CURRENTLY SELECTED LIBRARY

The library is a special directory which is intended to contain utility program. It acts as an extension for the CSD, since the CSD can only contain up to 47 objects, and this number may be insufficient.

If the ROOT directory has a sub-directory whose name begins with LIB - such as LIB10/30 - then this directory will be used as the Currently Selected Library. Otherwise, you may set any directory to be the Currently Selected Library by *LIB <*Obspec*>.

For example:

```
*LIB :1.$.ALLMYROMS
```

The Currently Selected Library is very useful for storing disc images of Sideways ROMs, or overlay code for a large program. If you *RUN a program which is not on the Currently Selected Directory, the ADFS will try to find it on the Currently Selected Library. For example, if you have Sideways RAM, and you need to use the PRINTER BUFFER prgoram, then you can type *PRINTER, instead of *$.LIB10/30.PRINTER. Most of the time you can leave the Currently Selected Library "unset".

2.7 CHANGING THE CURRENT FILING SYSTEM

The ADFS may be selected as the current filing system by the command *ADFS, or by holding the CTRL key and the A key, while the BREAK key is pressed, in which case the Currently Selected Directory is reloaded.

For example:

```
*TAPE
LOAD "MYPROGRAM"
*ADFS
SAVE "MYPROGRAM"
```

The STL ADFS will keep track of the path, and of the Disc Address of the CSD all the time. It will even keep track of them while the ADFS is not the current filing system, so that when you re-enter the ADFS, the same CSD will be re-selected. This feature is of particular value when transferring objects between filing systems - e.g. between the DFS and the ADFS.

You can also use the command *FADFS, which will then select the ADFS without reloading the Currently Selected Directory. This command is very useful if you need to change the disc or drive number when changing between filing systems. For example:

```
*TAPE
LOAD ""
*FADFS
*MOUNT 5
SAVE "MYPROG2"
```

If you need to switch from the ADFS to the original DFS, you could either use the command *DISC (or *DDFS, for the STL double density mode), or hold down the CTRL and D keys, and then press the BREAK key.

Other Filing systems may be selected by the usual commands - e.g. *NET, *TAPE3, TAPE12, *TELESOFT or *ROM.

3. SUMMARY OF STL ADFS COMMANDS

The STL ADFS contains the following commands:

Command	Minimum Abbreviation
ACCESS <List Spec> (L) (W) (R) (E)	A.
ADFS	
BACK	BAC.
BACKUP (<Drive>) (<Drive>)	BACKU.
BYE	BY.
CDIR <Object Spec>	CD.
CLOSE	CL.
COMPACT	CO.
COPY <List Spec> <*Object Spec*>	COP.
DELETE <Object Spec>	DE.
DESTROY <List Spec>	DES.
DIR <Object Spec>	DIR
DISMOUNT (<Drive>)	DISM.
EX <*Object Spec*>	EX
FADFS	
FORM160 (<Drive>)	
FORM80 (<Drive>)	
FORM40 (<Drive>)	
FREE	FR.
INFO <List Spec>	I.
LCAT	LC.
LEX	LE.
LIB <*Object Spec*>	LIB
MAP	MA.
MOUNT (<Drive>)	MOU.
OPEN <No. of files>	OP.
PASSWORD <Password>	PASS.
REMOVE <Object Spec>	RE.
RENAME <Object Spec> <Object Spec>	REN.
TITLE <Title>	TI.
VERIFY (<Drive>)	V.

Full details of the above are given in the next section.

In addition, the STL ADFS is designed to make use of a number of utilities contained in the STL DFS 2.1. These are listed in section 8 and set out in detail in section 9.

4. STL ADFS COMMANDS IN DETAIL

*ACCESS <listspec> (E) (L) (W) (R)

Purpose

To prevent an object from being accidentally overwritten or deleted, an object is said to have certain 'attributes' which control the way it can be accessed. For instance:

E - Execute only. The E attribute is used to protect files containing machine code programs that the author wants left untouched. If the E attribute is set, then the file cannot be *LOADed, also all OSFILE calls except call 6 (delete file) are prevented - as in the display of object information by the *INFO and *EX commands. The only commands which can affect a file with the E attribute set are:

*RUN <filename, *<filename>, *DELETE, *REMOVE, *DESTROY, *ACCESS

When the E attribute is set, *ACCESS can only be used to set or unset the L attribute and the R and W attributes are prevented.

L - Lock. If the L attribute is set, the object cannot be deleted or overwritten. This applies to both files and directories.

R - Read access. This must be set for reading or loading to be allowed.

W - Write access. This must be set for writing or updating to be allowed.

Examples.

ACCESS $.TOOLKIT. E

This command would set the Execute only attribute, to all the files in the directory called TOOLKIT, so they can only be accessed with *ACCESS, *DELETE, *DESTROY or *REMOVE

*ACCESS HELP L

This command will Lock the file called HELP in the currently selected directory.

*ACCESS *.* R

This command will allow all the files in the currently selected directory to be read or loaded.

ACCESS LETTER LR

This command will Lock and set the read access attribute to all the files beginning with LETTER in the currently selected directory.

Description

Sets the attribute string of a list of objects to that given.

Notes

The last attribute - D - is only set if the object is a directory. The D attribute cannot be changed or set.

It is not necessary to specify attributes when an object or a file is first created. The filing system does this for you by setting the default attributes. These are:-

For a file	- WR (Read and Write access)
For a directory	- DLR (Directory, Locked and Read access)

If a file is Locked, then certain commands capable of writing to the file, or its entry in the directory, will not affect the file and the message: 'Locked' will be produced. These commands are:

*SAVE, *DELETE, *DESTROY, *RENAME

If a file is Locked, it can still be erased if a FORMAT command is issued, but the only way to remove the L attribute is by using the *ACCESS command without the L attribute being set.

If the R attribute is not set then a file cannot be read, and any attempt to use the commands: *LOAD or *COPY would result in the message:

Access violation

being printed. Also, if the R attribute is missing and an attempt to use the BASIC command 'OPENIN' or the 'OSFIND' command in an assembly listing, or if the W attribute is not set on a file and an attempt to open the file for update, then the same message 'Access violation' will occur.

*ADFS

Purpose

To enter the ADFS from another filing system.

Associated commands

*DISMOUNT, *FADFS, *MOUNT

Note

The same effect may be had by pressing BREAK while holding down CTRL and 'A' or CTRL and the right arrow key.

Alternatively, pressing BREAK while holding down just 'A' will enter the ADFS without resetting the Currently Selected Directory to '$' (the 'root').

*BACK

This command will return to the previously selected directory (PSD). The directory selected before the last *DIR or *BACK command becomes current, and the PSD is set to the old CSD. Thus if you repeatedly typed *BACK, you could switch between the two frequently used directories.

Example

*DIR $.LETTER	Selets directory LETTER as the Currently Selected Directory
*DIR $.WORK	Selects WORK as the CSD, and LETTER as the PSD
*BACK	LETTER is now the 'CSD' and WORK is the 'PSD'
*BACK	WORK is now the 'CSD' and LETTER is the 'PSD'

Description

*BACK will make the previously selected directory into the currently selected directory

Associated commands

*DIR, *CDIR

*BACKUP (<Drive>) (<Drive>)

Purpose

This command is for backing up from one drive to another - including the ability to handle files longer than a single disc.

Examples

*BACKUP 0 1 would back up all the data from drive 0 to drive 1 - as might be used in a system having only floppy disc drives.

*BACKUP 0 4 would back up all the data from drive 0 (a Winchester) to drive 4 (a floppy drive)

Note

This command assumes that all the floppy discs used have been formatted with 160 tracks over both sides as one logical drive - of 640K capacity.

Unlike the corresponding DFS command, *BACKUP is intelligent - in that it only backs up as far as the highest occupied address on the source disc.

It will recognise when a Winchester is the source drive and calculate how many floppies (of 640K each) will be required to hold all the data. Before carrying out the backup operation, it prompts you for confirmation with 'Are you sure (Y/N)'.

When backing up a Winchester, because of the number of floppies that may be involved, you should label them carefully.

To restore the data from the backup discs to the original drive, you simply issue the *BACKUP command again with the drive numbers in the reverse order.

*BYE

Purpose

This is for use when ending a session of using the Winchester disc unit. This command MUST be used before moving the Winchester Disc. It will close all open files, copy all the RAM buffers to disc, and - most importantly - move the READ / WRITe head to the Shipping Zone.

Associated commands

*CLOSE

*CAT (<*Object Specification*>)

Purpose

This command will display a catalogue of the specified directory on the currently selected output unit (Screen, Printer). This consists of a directory 'header' and a list of the objects in the directory. If the specified directory is not found, then the error 'Not Found' is returned. If the <*Object specification*> is not supplied, then the currently selected directory is catalogued.

The objects in the directory are listed in alphanumerical order, with their attributes and their sequence numbers.

Examples

```
 *CAT


WORK              (08)
Drive: 0          Option 00 (Off)
CSD: Memo         CSL: "unset"

MEMO1         WR (04)    MEMO2        LWR (05)
PAPERS       LWR (07)    SUPPLIES      WR (08)
```

The title of the currently selected directory is WORK and the number in brackets following the directory title is the Master Sequence Number (MSN) for this directory. When a new object is added to the directory, or when an object is modified and then stored back on disc, then the sequence number of the new object is set equal to the MSN, unless an object exists with the same number as the MSN, in which case the MSN is incremented by one, and the new object is given the new value of the MSN.

The MSN is a decimal number, which goes from 00 to 99, and then starts again at 00.

Description

Displays the catalogue of a directory.

Associated commands

*ACCESS, *DIR, *EX, *LEX, *INFO, *OPT 4, (<option number>), *TITLE

*CDIR <*Object Specification*>

Purpose

To create a new directory. A new, empty directory is created with the specified name. The name is allocated as the directory title (as the default) and the Master Sequence Number is set to 00.

Example

*CDIR ACCOUNTS

which creates a new directory called ACCOUNTS in the Currently Selected Directory.

*CAT ACCOUNTS

```
ACCOUNTS          (00)
Drive: 0          Option 00 (Off)
CSD: $            CSL:  Library1
```

This shows that ACCOUNTS is an empty directory, and the currently selected directory is the root directory of drive 0.

Description

Creates a new directory.

Associated commands

*CAT, *., *DIR, *EX, *LEX, *TITLE

*CLOSE

Purpose

To close all open files and ensure that all disc buffers in RAM are placed back onto the disc. *CLOSE is equivalent to CLOSE #0 in BASIC.

Example

*CLOSE

Description

Closes all open files.

*COMPACT (<Start-Page> <Length-in-Pages>)

Purpose

To compact the information on the disc drive so that free space may be gathered into larger contiguous blocks. This improves speed for accessing the drive and the error messages: 'Compaction required' or 'Map full' are avoided. The area of RAM used to temporarily hold disc information, while the compaction is taking place, is the current screen memory unless specified otherwise.

<Start-Page> and <Length-in-Pages> are optional, and both are Hexadecimal numbers. <Start-Page> is the start page and <Length-in-Pages> is the length in pages of the area of memory to be used by the command.

There must be RAM in the area specified for the command to work correctly.

This command will corrupt the RAM contents, so if there is information that you wish to keep then SAVE it first.

Examples

*COMPACT 30 40

which will use memory between &3000 and &7000 inclusively.

Description

This command will compact the drive, and gather up the free space.

Associated commands

*FREE, *MAP

Notes

The command causes each object on the disc to be examined, and if there is sufficient free space just before, the object is copied into the free space - using the specified area of memory as a buffer. Thus objects tend to move towards sector zero on the disc and free space tends to move towards the end of the disc - thus gathering together in larger blocks.

If neither <Start-Page> nor <Length-in-Pages> are issued, i.e. you just type in *COMPACT, then the Current screen memory up to &7FFF will be used.

As a further example, consider the following:

*MAP is a command which lists the free space on the disc. Hence, if you type *MAP then the output could be:

```
 Address  :   Length
 000420   :   0000A0
 000A4D   :   000060
```

Then if you typed *COMPACT and then *MAP the result would be:

```
 Address  :   Length
 000E54   :   000004
 000CD4   :   00FFD2
```

The information has now been shifted to reduce the number of free spaces on the disc, with the result that the first free space is now at a higher address.

*COPY <list specification> <*object specification*>

Purpose

To copy a list of files into another directory. All the files referred to by the <List specification> are copied into the directory specified by <*Object specification*>. Memory from the start of the BASIC user RAM area up to the start of screen memory is used, so you will find that a *COPY is more efficient in Mode 7 than it is in Mode 0.

Any data or programs in memory are lost.

Example

```
*COPY @.* $.WORK
```

will copy all the files in the current directory to the directory called WORK.

```
*COPY $.WORK.MEMO* $.WORK2
```

will copy all files in directory WORK which start with MEMO. into the directory called WORK2.

```
*COPY $.WORK.* @
```

will result in all the files in the directory called WORK being copied to the currently selected directory.

Description

For copying multiple objects within the currently selected filing system - here the ADFS.

Notes

For copying between two different filing systems, see the utilities *MVADFS and *MVDFS.

For backing up the contents of whole directories or drives - e.g. from a Winchester to floppies - see the command *BACKUP.

*DELETE <Object specification>

Purpose

To delete a single object from the disc. The space that was occupied by the object then becomes available for other information. Once an object has been deleted you cannot retrieve it, unless you use *DZAP, or *RECOVER.

Examples

```
*DELETE MEMO45
```

would delete the file called MEMO45 from the currently selected directory.

```
*DELETE $.WORK.MEMO50
```

would delete the file or object called MEMO50 from the directory WORK.

Description

Deletion of a single object.

Notes

The currently selected directory, the current library or open files cannot be deleted.

If you try to delete a file that has the L attribute ('Locked') then the message 'Locked' would be printed and the file would be left intact.

A directory or file with the L attribute can only be deleted by resetting the L attribute with the *ACCESS command, and if the directory is empty.

Associated commands

*DESTROY, *REMOVE

*DESTROY <List specification>

Purpose

This command will remove a number of objects from the disc in a single operation. A list of the objects which are to be removed is displayed, followed by the message 'To be destroyed. Are you sure?'. If you want to delete the objects or files then type in 'Y' but if you do not, then type 'N' or any other key.

Example

DESTROY WORK deletes all objects in the currently selected directory that begin with 'WORK'.

Description

Deletion of multiple objects.

Associated commands

*ACCESS, *DELETE

Note

The operation will not be performed if the object is 'Locked' - you will get an error message "Locked".

*DIR (<*object specification*>)

Purpose

This command will make the Directory specified in <*Object specification*> the currently selected directory. If the <*Object specification*> is not specified, then the ROOT directory is selected.

When the system is first activated, or after the CTRL and BREAK keys are held down together, then the Currently Selected Directory is set to the ROOT directory.

Example

*DIR WORK	will select the directory called WORK as the CSD.
*DIR	will select the ROOT directory as the CSD.
*DIR :1.$.WORK	will select drive 1 as the current drive and WORK as the CSD. The '$' (dollar) sign, signifying the 'root' directory, may be omitted - as it is understood.
*DIR ^	will select the parent of the CSD as the new CSD.

Associated commands

*CDIR, *LIB.

Note

Unlike under the DFS, a directory must be created explicitly (with *CDIR) before it can be set as the Currently Selected Directory (CSD).

*DISMOUNT (<Drive number>)

Purpose

To ensure that all open files have been closed, and all the Buffer pages are empty. It is intended for use before changing dismountable media - either floppy discs or Winchester drives with interchangeable discs.

Examples

*DISMOUNT	will close all open files on the Currently Selected Drive
*DISMOUNT 1	will close all open files on drive 1.

Notes

*DISMOUNT only closes the files on drive specified (or on the currently selected drive, if none is specified)

After a drive has been *DISMOUNTed, both the Currently Selected Directory and the Currently Selected Library are "unset". If the next disc command requires a drive and a directory to be specified explicitly (e.g. as part of an object specification), then "No directory" will be returned. If however the command has a default drive and directory (i.e. drive :0. directory 'root') - as does *CAT, then it will work accordingly.

The CSD and CSL may be set using the *DIR and *LIB commands.

Associated commands:

*BYE, *MOUNT

*EX (<*Object specification*>)

Purpose

This command will display information about the length and location of objects in the directory named, or the Currently Selected Directory. The information is displayed (in Hexadecimal) in a set order across the screen, i.e.

Object Name	Attributes	Sequence Number	Load Address in Sectors	Execution Address in Sectors	Length in bytes	Start Sector

Example

*EX WORK could result in:

```
WORK                    (19)
Drive: 0                Option 03 (exec)
CSD: 0                  CSL: 0

!BOOT       WR  (04)      00000000   FFFFFFFF   00000065
MEMOS       DLR (02)                            0000004D
MEMO        WR  (07)      00000000   00000000   000002B8
```

If the *EX command is issued without the <*Object specification*> then the Currently Selected Directory is examined, and if the object is a directory then only the start sector is displayed.

If the E attribute is set for a file or object then only the attribute string and the generation number is shown. For example if the file called !BOOT had the E attribute set then, after an *EX command had been issued, the screen would show:-

```
!BOOT         E (04)
```

Description

Displays information about directory contents.

Associated commands

*CAT, *INFO

*EXEC (<*object specification*>)

The *EXEC command reads from the specified file, one byte at a time, and places it into the keyboard buffer.

Example

*EXEC !BOOT

will take the contents of the file called !BOOT, read them one character at a time, as if it was being typed in at the keyboard, and act on any commands.

Associated commands

*BUILD, *WORD (Both of which are utilities)

*FADFS

Purpose

The *FADFS command will start up the Advanced Disc Filing System (ADFS). Unlike *ADFS, this command will set the currently selected directory and the currently selected library to the "unset" state.

Example

If you type in *FADFS and then type in *CAT then the screen might look something like:

```
$                       (19)
Drive : 0               Option 00 (Off)
CSD: "Unset"            CSL: "Unset"
```

Associated commands

*ADFS, *DISMOUNT, *MOUNT

Note

The same effect may be had by pressing BREAK while holding down CTRL and 'F'.

Unlike the Acorn implementation, when the next disc command is issued, the STL ADFS automatically loads drive :0 directory 'root' as the Currently Selected Directory and Currently Selected Library - by default. These defaults can be overridden with *MOUNT <drive> and *DIR <*object specification*>.

*FREE

Purpose

This command is issued to display the amount of free space that is left on the disc - measured in disc sectors (hexadecimal), and in bytes (decimal).

Example

If you typed in *FREE then the screen might show:

```
00AF83 Sectors - 11502336 Bytes Free
00827D Sectors =  8551680 Bytes Used
```

Description

The *FREE command will show the amount of free space left on the disc.

Associated commands:

*MAP

*HELP (<Keyword>)

This command will display useful information about the ROMs that are inside your system. If no Keyword is given, a list of all the ROMs is returned. If a Keyword is given, more information about that ROM is returned.

For instance, information about the Advanced Disc Filing System can be shown if you type:

```
*HELP ADFS
```

The screen will look like:

```
STL ADFS 2.1
  ACCESS        <List Spec> (L)(W)(R)(E)
  ADFS
  BACK
  BYE
  CDIR          <Ob Spec>
  CLOSE
  COMPACT       <SP> <LP>
  COPY          <List Spec>
  DELETE        <Ob Spec>
  DESTROY       <List Spec>
  DIR           <Ob Spec>
  DISMOUNT      (<Drive>)
  EX            <*Ob Spec*>
  FADFS
  FORM160       (<Drive>)
  FORM80        (<Drive>)
  FORM40        (<Drive>)
  FREE
  INFO          <List Spec>
  LCAT
  LEX
  LIB
  MAP
  MOUNT         (<Drive>)
  PASSWORD      <Password>
  REMOVE        <Ob Spec>
  RENAME        <Ob Spec><Ob Spec>
  TITLE         <Title>
  VERIFY        (<Drive>)

OS 1.20
```

Note

Certain commands are not included in the list because they are not ADFS commands. Some originate from the Machine Operating System and include *CAT, *EXEC, *LOAD, *OPT, SAVE, *SPOOL and some are utilities which are common to both the STL DFS and the STL ADFS. They will be shown in response to *HELP UTILS and are summarised in Section 8 and detailed in Section 9.

*INFO <List Specification>

Purpose

This command will display information about a list of objects. It consists of the Object name, Attribute string, Sequence Number, Load address, Execution address, Length of the file and the Disc Sector address.

Example

INFO MEMO

This will display information about all the objects in the currently selected directory that begins with MEMO.

INFO WORK.

This will display information about all the objects in the Directory called WORK.

Description

The *INFO command will return detailed information about a set of objects.

Associated Commands:

*CAT, *EX, *LEX

*LCAT

The *LCAT command will catalogue the current library, and the result is in the same format as the *CAT command.

Example

*LCAT might result in:

```
$                    (47)
Drive:0              Option 03  (Exec)
CSD: WORK2           CSL: $

!BOOT      LWR (17)  MENU       LWR(19)
```

This shows that the currently selected library is the ROOT directory, and the currently selected directory is called WORK2.

Associated commands

*CAT, *LIB

*LEX

Purpose

The *LEX command will examine the currently selected library, and returns the same result as the *EX command on other directories.

Example

```
*LEX

$                   (47)
Drive:0             Option 00  (Off)
CSD: WORK2          CSL: $

!BOOT      LWR (17)  00000000  FFFFFFFF  00000047   000056
MENU       LWR (19)  FFFF1900  FFFF8023  00000C75   000208
MEMO1      LR  (22)  00000000  FFFFFFFF  00000013   0001AB
```

In the above example, the currently selected directory is called WORK2, and the currently selected library is the ROOT directory.

Associated commands

*EX, *LIB

*LIB (<*Object specification*>)

Purpose

This command will set the library to the specified drive number, and the specified directory.

Example

If the command *LIB $.WORK is entered, then it will select the directory called WORK in the ROOT as the current directory, and after this, if you type:

*<filename>

it will first search the CSD, and if not found, then in the directory called WORK for the named file. If it is found, it will be loaded into memory and executed, just as if you had typed *RUN WORK.<filename>.

Description

Sets the directory that is to hold the library.

Associated commands

*LCAT, *LEX, *RUN

Notes

When the ADFS is first entered, the library directory is set to $, Unless there is a directory with a filename beginning with $.'LIB', in which case this latter directory would be allocated as the library. A directory will not be retained as the currently selected library following a HARD BREAK from ADFS, unless its name begins with the filename $.LIB.

The Library directory can only be deleted from the disc by allocating another directory as the library. E.g. if you type *LIB then the ROOT would be set as the library directory, and the 'old' library directory can then be deleted.

The library is usually used to contain machine code utility programs, which you may want to *RUN whichever is the Currently Selected Directory.

The library also provides a means of having more objects "on tap" than can be held in a single directory (i.e. 47 under the ADFS).

*LOAD <*Object Specification*> (<Reload Address>)

Purpose

This command reads the named file from the disc into memory, either at the address specified in the *LOAD command, or at the reload address with which it was *SAVEd.

Examples

*LOAD MENU

will read the file called MENU and load it into memory starting at the reload address with which it was *SAVEd.

*LOAD MENU 2000 will read the file called MENU and load it into address &2000 upwards.

*LOAD Newdrives FFFE3000 will load into (screen) shadow RAM on the B+.

*LOAD TOOLKIT FF0A8000 will load into sideways RAM. [bank 8 ?] on the B+.

Description

Loads a file into memory.

Associated commands:

*EX, *INFO, *SAVE

Notes

You should not try to *LOAD programs into memory below the default setting of PAGE because the ADFS uses this area to keep track of the disc, and to manage the drive.

Under the STL ADFS, the value of PAGE varies with the number of file channels *OPENed:

- from the default of &1900 for one file channel open
to &1D00 for five file channels open
and to &2200 for ten file channels open.

Since only one open file channel is required in many cases, this gives maximum compatibility for programs written for the original DFS.

In the Acorn implementation, PAGE is set at &1D00. This is higher than for the DFS and the STL ADFS for 1 to 5 channels open but lower than the latter for 6 to 10 channels open. However, this is only achieved by swapping the contents of channels temporarily out to disc - which greatly slows any operations which use more than 5 channels.

*MAP

Purpose

This command will display a map of the free space that is available for use on the disc. The format is a list of numbers that are paired in the form:

<Sector Address> : <Length in Sectors>

If there is a large number of entries in the free space map, and the disc is becoming fragmented, then certain commands, such as SAVE, *CDIR, *COPY, etc might cause the error message 'Compaction required' to be displayed. If you do get this message, then you are advised to *COMPACT the disc. You are allowed up to eighty entries in the free space list, but if the *MAP command gives more than sixty-five, then you are advised to *COMPACT the disc.

Example

```
*MAP

Address  :  Length
0000A4   :  000001
000207   :  000001
000216   :  0007EA
```

Description

Displays the available free space map.

Associated commands

*FREE

*MOUNT (<Drive number>)

Purpose

This command is for use with dismountable media - either floppy discs or Winchester drives with interchangeable discs.

The command initialises a Winchester drive (by forcing the controller to do a 'hard reset') and reads the Free Space map into RAM. You should issue a *MOUNT command after a disc error has occurred.

Example

```
*MOUNT 1
```

will initialise drive number 1.

Description

Initialises a drive.

*MVADFS

Purpose

To move objects from the ADFS to the DFS.

*MVDFS

Purpose

To move objects from the DFS to the ADFS.

*OPEN <Number of file channels>

Purpose

To set the number of open file channels - each of which raises PAGE by &100. The minimum is 1 and the maximum is 10.

Notes

Under the STL ADFS, the value of PAGE varies with the number of file channels *OPENed:

- from the default of &1900 for one file channel open
  to &1D00 for five file channels open
  and to &2200 for ten file channels open.

Since only one open file channel is required in many cases, this gives maximum compatibility for programs written for the original DFS.

In the Acorn implementation, PAGE is set at &1D00. This is higher than for the DFS and the STL ADFS for 1 to 5 channels open but lower than the latter for 6 to 10 channels open. However, this is only achieved by swapping the contents of channels temporarily out to disc - which greatly slows any operations which use more than 5 channels.

This command must be followed by CTRL-BREAK, since PAGE and the ADFS private workspace must be reset.

All file operations require at least 1 channel open, but few require more than 2. One instance where more may be needed is when maintaining multiple indexes up to date, while amending the records in certain database programs - such as VIEWSTORE. With up to 10 open channels being possible, and 1 being needed for the datafile, up to 9 indexes may be maintained up to date while amending records.

*OPT 1 (number)

Purpose

This command enables or disables the system which controls the displaying of information (the same as the *INFO command).

*OPT 1,0 will stop the information from being displayed.

*OPT 1,1 will give short messages.

*OPT 1,2 will display extended information about files.

Note

There must be a comma or a space between the *OPT 1 and its argument (number).

*OPT 4 (number)

Purpose

This command will set the auto start option. There are four possible options - 0, 1, 2 and 3. Each of them initiates a different action when you hold the SHIFT key while pressing BREAK. The computer will either do nothing or automatically *LOAD, *RUN or *EXEC a file called !BOOT which is in the Currently Selected Directory.

Examples

```
 *OPT 4,0       will turn the autostart option off
 *OPT 4,1       will *LOAD a file called !BOOT
 *OPT 4,2       will *RUN  a file called !BOOT
 *OPT 4,3       will *EXEC a file called !BOOT
```

Description

This command sets the start-up option for the directories.

Note

*OPT 4 is one of the MOS commands and is not shown in the *HELP screen on the ADFS. It is passed onto the ADFS by the OSFSC call.

If the option is set to 1, 2 or 3, and if you press SHIFT-BREAK, then the DFS will search for a file called !BOOT, but if the file called !BOOT is not present, then the message 'Not found' will be produced.

The Acorn implementation only allows a single !BOOT file, in the ROOT directory. The STL ADFS however, allows !BOOT files in any directory - including the ROOT directory.

*PASSWORD <Password>

Purpose

To limit access to the objects in any directory for any purpose other than *LOADing and *RUNning files, to only those knowing the password. The main purpose is to afford privacy to multiple users of a Winchester disc.

Example

If the directory 'MINE' is protected by a password, and you type *DIR MINE, "Password required" is returned. The correct response is *PASSWORD <Password>, otherwise access is limited to *LOADing and *RUNning files.

Notes

Password protection may be applied to the Currently Selected Directory, by running the program called 'PROTECT', which is on the Utility disc (Number 9). It may only be removed by running 'PROTECT' again - and using the <Password> !

This facility can also be used from within a program or an *EXEC file.

*REMOVE <Object specification>

Purpose

This command will delete a single object or file from the disc. The *REMOVE command works exactly the same way as the *DELETE command, but if the file does not exist, then the message 'Not found' will not be produced. This can be an advantage when calling it from a program - for example, to avoid upsetting a screen format.

Example

```
*REMOVE MEMO45
```

will remove the file called MEMO45 from the currently selected directory, but if the file is not present, no error message is given.

Description

The *REMOVE command is used for object or file deletion without error reporting.

Associated commands

*DELETE, *DESTROY

*RENAME <Old object specification> <New object specification>

Purpose

This command will change the object name, and move it to another directory if need be.

Example

```
*RENAME MEMO45 MEMO46
```

would rename the file called MEMO45 in the currently selected directory, and change its name to MEMO46 in the same directory.

```
*RENAME $.WORK.MEMO32 $.BACKUP.OLDDEMO
```

would remove the entry of 'MEMO32' from the directory called WORK, and add a new entry to the directory called BACKUP, and with a new name 'OLDMEMO'.

If the <New name for file> already exists, or the source file is locked, then an error message 'Bad rename' will be produced.

Note

The operation consists of removing a directory entry, and changing its name if necessary, then adding the entry to the same, or a new directory.

Do not use Wildcards in a *RENAME command.

When renaming a directory, the root name (i.e. '$') may not be used as either the old or the new name.

A directory cannot be renamed so as to refer to itself. You must first *COPY it and then *DELETE the original.

*RUN <*Object specification*> (<Optional parameters>)

The *RUN command is used to run machine code programs. It loads a file into memory and then jumps to its execution address, unless the execution address is &FFFFFFFF, in which case the file is *EXECed (as a text file) into the KEYBOARD buffer.

Example

*RUN SHIFTER

will load the file called SHIFTER from the currently selected directory, into memory at its reload address, and execute it at its execution address, with which it was *SAVEd.

Description

Loads and runs a machine code file, or *EXECs a file if the execution address is set to &FFFFFFFF

Notes

*RUN is a MOS command - not part of the ADFS

The *RUN command will not work with BASIC programs

Also typing *<filename> or *<*object specification>, or */<*object specification> is equal to *RUN <*object specification>.

If the (<optional parameters>) are entered with the command, then the program will load into that address.

*SAVE <Object specification> <Start address> <Finish address> (<Execution address>) (<Reload address>)
or
*SAVE <Object specification> <Start address + Length> (<Execution address> (<Reload address>))

Purpose

This command should not be confused with the BASIC command SAVE, because they are quite different. This command takes a copy of the specified memory inside the computer, and saves it to the disc (or other filing system). The main purpose of this command is to save machine code routines or graphic screens.

Examples

```
          File    Start    Finish   Execute  Reload
          Name    Address  Address  Address  Address
          ----------------------------------------

    *SAVE PROGRAM FFFF2000 FFFF4000 FFFF2003 FFFF2000

          File    Start    Finish   Execute  Reload
          Name    Address  Address  Address  Address
          ----------------------------------------

    *LOAD PROGRAM FFFF2000 +FFFF2000 FFFF2003
```

Notes

The Execute and Reload addresses may be omitted - in which case they are taken to be the same as the Start address.

As an enhancement on the Acorn implementation, the STL version of *SAVE can be made to save to disc any Sideways ROM - and not only that of the currently selected filing system. This is done by including the ROM socket number (in hexadecimal) just before the 4-digit start address (normally 8000). Thus:

*SAVE <filename> FF088000 +2000 D9CD will save the sideways ROM in socket number 8 onto the disc.

This is designed to work only with floppy drives, as the file could be corrupted - which would be too dangerous for a Winchester.

*SPOOL (<Object specification>)

Purpose

This command will open a file to the disc with the name specified, and then sends all the text that is 'Printed' (on the screen) to the file.

If no name is specified then the last *SPOOL file is closed.

Example

To produce a text file for a file called MENU type:

```
LOAD "MENU"

*SPOOL MENU2

LIST

*SPOOL
```

This will cause the tokenised BASIC program called MENU to be converted into an ASCII file called MENU2.

Description

This command will spool all subsequent output to the screen, to the named file. The file is closed by issuing the command *SPOOL on its own.

Notes

SPOOL (i.e. ASCII text) files are much less specific to a given type of computer than are e.g. tokenised BASIC programs. This can therefore be one step in transferring such programs between machines of different types.

*TITLE <Title>

Purpose

This command sets the title of the Currently Selected Directory. The specified title may be up to nineteen characters long, and all characters to the right of the command (leading spaces are ignored) up to the RETURN character or double quotes (inverted commas) are copied into the title field of the currently selected directory.

The title is distinct from the directory name and is not used by the ADFS, but is stored for reference by the user. However, the directory name will be used as the directory title by default until you change it with a new *TITLE command.

Example:

*CAT might produce:

```
$                    (22)
Drive:0              Option 00 (Off)
CDS: $               CSL: $

!BOOT       WRL(14)  MENU        WRL(14)
```

If you then type *TITLE NEWMENU and then type *CAT again, the screen would show:

```
NEWMENU              (22)
Drive:0              Option 00 (Off)
CDS: $               CSL: $

!BOOT       WRL(14)  MENU        WRL(14)
```

5. TECHNICAL INFORMATION ON THE STL ADFS

5.1 FILE HANDLING USING 'BASIC'.

This is documented in the User Guide pg 395 and the Advanced Disk User Guide pg. 269.

5.2 FILE HANDLING USING ASSEMBLY LANGUAGE

Disc drives are usually considered as an extended part of the computer memory, and the Disc Filing System as providing a software interface between the Disc drive hardware and the program, through seven vectors called 'Filing System Tasks'. They are:

OSFILE (&FFDD), OSFIND (&FFCE), OSARGS (&FFDA), OSBGET (&FFD7), OSBPUT (&FFD4), OSGBPB (&FFD1) and OSWORD (&FFF1).

All but the last are well-documented elsewhere:

- The User Guide, pg. 451.
- The Advanced User Guide, pg. 335.
- The Advanced Disk User Guide, pg. 121

and so will be covered only briefly below.

The last - OSWORD - is not documented elsewhere, and is therefore covered in more detail below.

When you move data between the computer and the disc, the ADFS normally issues the request (also called the Reason Code) in the A register, and then jumps to the required subroutine.

Although the way these tasks are performed is quite different between one filing system and another, the result is always the same, e.g. the data is saved.

When a file is accessed it is presumed that it is in the Currently Selected Directory, unless the Correct pathname is used, i.e.

*LOAD :4.UTILS.SILEX 6000 will load the file called SILEX, from the directory called UTILS, starting at address &6000.

A simple program to use this Load function is:

```
10 DIM Q% 100, FCB 16, Name 20:OSFILE=&FFDD
20 &Name=":4.UTILS.SILEX"
30 !FCB=Name
40 !(FCB+2)=&FFFF6000
50 X%=FCB MOD 256:Y%=FCB DIV 256:A%=&FF
60 CALL OSFILE
```

5.2.1 OSFILE (&FFDD)

X (Load Address) and Y (High Address) will point to the FCB (File Control Block). A holds the function to be performed:

A=0	*SAVE the specified file
A=1	Write/update the directory entry of the specified file
A=2	Write the load Address
A=3	Write the Execution Address
A=4	Write the File Access Attributes
A=5	Read the file information from its directory entry. On return, A=1 (file object) or A=2 (directory object), or A=0 if not found.
A=6	*DELETE the specified file
A=7	Add a new entry to the directory. i.e. dummy *SAVE
A=&FF	*LOAD the specified file

Note. The 'Reason code 7' was not implemented on the original DFS, but on the ADFS, it lets you initialise a large datafile, without having to fill the file with 00's.

5.2.2 OSFIND (&FFCE)

OSFIND is used to open or close a file. Note that the ORARGS &FF calls and the OSFSC6 call will also close the file.

A=0, Y=channel	: CLOSE #channel
A=0, Y=0	CLOSE #0

A=&40, &80 or &C0: a file is to be opened. X (low address) and Y (high address) point to the filename. On return, A contains the channel number (32 to 41 with the ADFS), if A=0 then the file could not be found, or its access is forbidden.

5.2.3 OSARGS (&FFDA)

If Y=channel number (From 32 to 41):

A=0	(Pointer) = PTR # channel
A=1	PTR # channel = (Pointer)
A=2	(Pointer) = EXT # channel
A=&FF	Close # channel

If Y=0,

A=0	On return, A=4 for the DFS and A=8 for the ADFS
A=1	On return, the pointer contains the address of the * command
A=&FF	Close All files

5.2.4 OSBGET (&FFD7)

When this routine is called, Y is equal to the channel number, and on exit, A contains the Byte that was read, and PTR#channel will then be incremented automatically. The carry flag is set if the end of the file was reached.

5.2.5 OSBPUT (&FFD4)

On entry, the Y register, contains the channel number, and the Accumulator contains the Byte to be written.

5.2.6 OSGBPB (&FFD1)

On entry, X and Y point to a file control block in memory. A defines the information to be transferred to or from the open file.

5.2.7 OSWORD (&FFF1)

There are several important hardware features which can be of interest to a programmer, such as OSWORD &70 for use with Telesoft, OSWORD &7F with the original DFS and OSWORD &72 with the ADFS.

6. THE ADFS ERROR MESSAGES

The STL ADFS error messages are given below, preceded by the corresponding error codes. These are not shown on the screen, but may be incorporated in error-trapping routines in your own programs.

&92 Aborted
The response to the question 'Destroy' has been other than 'YES' (or 'yes' etc), following a *DESTROY command.

&BD Access violation
An attempt has been made to read or load a file with the R attribute not set, or to write to a file with the W attribute not set.

&C2 Already open
An attempt has been made to delete (or overwrite - by saving a new version of) a file which is open. This message also occurs if an attempt is made to open a file that is already open (unless both "opens" are for input only, using e.g. OPENIN in BASIC II).

&C4 Already exists
An attempt has been made to create an object with the same name as an existing object. The logic protects - and hence the message can occur with - *CDIR and *RENAME, but not *SAVE or SAVE in BASIC.

&AA Bad checksum
The computer memory has become corrupted, which prevents the ADFS from being able to read or write to a file, or to close it. The computer must be reset using CTRL-BREAK.

&FE Bad command
The last command was not recognised by the ADFS, nor by any other Service ROM, nor was it found (e.g. as a machine code utility) in the currently selected directory or current library of the currently selected filing system.

&A9 Bad FS Map
A bad Free Space map means that either the computer memory or sectors 0 and 1 of the current disc are corrupted. The computer must be reset using CTRL-BREAK.

&CC Bad name
An illegal filename was used - e.g. one including a : (colon) or $ (dollar), when not referring to a root directory, or other special characters, such as ^ (circumflex) or @ (commercial 'at') or even a "null" object - as implied by .. (two periods) adjacent to each other.

&CB Bad opt
An invalid argument has been supplied with a *OPT command.

&94 Bad parms
Invalid address parameters were supplied when specifying the memory to be used by *COMPACT.

&B0 Bad rename
A directory cannot be renamed in this way - e.g. so that the new object specification embodies the old object.

&A8 Broken directory
The directory has been corrupted. This should be a very rare occurrence - particularly with a Winchester disc. One course of action is to reformat the disc and restore the contents using your backups!

&96 Can't delete CSD
You are not allowed to delete the currently selected directory. You must set another directory as the currently selected directory before deleting the old one.

&97 Can't delete library
    You are not allowed to delete the current library. You must set another directory as the current library before deleting the old one.

&DE Channel
    A sequential file operation has been attempted with an invalid file handle - e.g. the file has not been opened.

&98 Compaction required
    Writing to the disc has been attempted when the free space is too fragmented - e.g. more than 80 free spaces or none large enough.

&B3 Dir full
    You can only have 47 objects in a single ADFS directory. You may be able to put some machine code files into the library, or you can create another directory alongside or below the present one.

&B4 Dir not empty
    You are not allowed to delete a directory until you have deleted all the objects in it.

&C8 Disc changed
    The disc has been changed and you should issue *MOUNT <drive> to read in the free space map.

&C7 Disc error
    During the last disc operation, the controller found a fault on the disc.

&C6 Disc full
    There is insufficient space for the operation requested. These include the creation of directories (with *CDIR) and of files (with *SAVE and SAVE in BASIC), the opening of files for writing (which have a default size of 256 sectors = 64Kbytes) and extending existing files.

&C9 Disc protected
    The (floppy) disc has a write-protect tab in place.

&CD Drive not ready
    The drive is not yet up to speed (e.g. soon after starting). If this persists, the drive may be faulty.

&11 Escape
    The Escape key has been pressed.

&DF EOF
    Two attempts have been made to read beyond the End of a File. The failure of the first attempt is shown by the contents of the C flag following an OSBGET or OSGBPB command

&C3 Locked
    You are not allowed to delete, rename or overwrite an object which is locked.

&99 Map full
    The free space map is full - e.g. it contains 80 entries. The disc should be *COMPACTed in order to allow further information to be saved to it.

&A7 No directory

&D6 Not found
    The object referred to has not been found.

&C1 Not open for update
    An attempt has been made to write to a random access file that is only open for reading. You should issue OPENUP in place of OPENOUT (or the equivalent in assembler).

&B7 Outside file
    An attempt has been made to set the pointer of a file (which is only open for reading) to a value beyond the end of the file.

&95 Too many defects
    Too many media defects were found during formatting. For example, a Winchester disc may expect to have 600 tracks, and have only 12 available as alternates for any that are defective.

&C0 Too many open files
    An attempt has been made to open more files than there are channels open. In the STL ADFS, you may have from one to ten channels open, set by the *OPEN command.

&FD Wildcards
    A wildcard character - '*' (asterisk) or '#' (hash) has been found where a full, unambiguous object specification is required - e.g. in a *CDIR, *SAVE, *DELETE or *REMOVE command.

&93 Won't
    An attempt has been made to *RUN a file, the load address of which is &FFFFFFFF. While files that are *RUN are normally machine code programs, this address is reserved for text files (often sequences of commands), which are *EXECed - i.e. read in as if being typed at the keyboard.

7. FEATURES OF THE STL DFS 2.1

The new STL DFS 2.1 is capable of replacing any version of the Acorn DFS, including the DNFS, with the DFS 1.2. When selected, the 2.0 ROM will identify itself by one of the following messages:

```
DFS 2.1 (1770)          or      DFS 2.1 (8271)
BASIC                           BASIC
>                               >
```

The DFS 2.1 also has a very extensive list of commands available, to provide users with the outstanding features detailed below and the most friendly error-trapping and correction mechanism. It also has a neat way of avoiding clashes with commands of the same name in any other ROM.

The STL DFS 2.1 ROM has a built-in mini word processor (*WORD) but if you require a full-blown version, we can supply it on a separate disk. This contains the WP word processor program (around 5K of machine code) and SILEX, the spelling checker, and comes with a 70-page manual - at a nominal cost of £3.00 inclusive of VAT, postage and packing. The WP disk is normally available in both 40- and 80-track formats. Please specify when ordering.

7.1 STL DFS COMMAND IDENTIFY LETTER

All Solidisk DFS commands may be made unique to avoid name clashing with other ROMs in your BBC, by prefixing the command with an identity letter. The Solidisk identity letter is lower case 'z'.

For example, if Computer Concepts Disc Doctor ROM is present in a higher priority (i.e. higher numbered) socket, and you type in *RECOVER, it will be passed to Disc Doctor. If however, you type in *zRECOVER, it will always be passed to the Solidisk DFS ROM.

7.2 UNLIMITED FILENAMES

When the number of filenames on the first (or current) catalogue reaches 31, a new catalogue will be created automatically.

Issuing *CAT will print on the screen:

```
320  Total Sectors              Disc1-JAN     (40)
Drive:0                         Option:3 (EXEC)
Directory:0.$                   Library:0.$
-------------------------------------------
   !BOOT                        MENU
   etc...                       .etc...

Press any key to continue-
```

Press the space bar and you should see:

```
Catalogue 2
-------------------------------------------

   !BOOT                        FRED
   etc...                       .etc...
```

This process is completely transparent to the user.

The Operating System and Disc Filing System (*) commands - LOAD, SAVE, RENAME, INFO, SPOOL, DUMP, TYPE, LIST, OPENIN, OPENOUT, OPENUP, BGET, BPUT, etc.. will work exactly as usual.

7.2.2. STORED CATALOGUES AND CURRENT CATALOGUE

When more than one catalogue is present on the disc, the last one (as returned by *CAT) is the current catalogue. All other catalogues are stored.

7.2.3 DELETE FILE

*DELETE <fsp> and *WIPE <fsp> work as usual on any catalogue. If the file is on the current catalogue, they actually remove it, but if it is on a stored (other than current) catalogue, they simply mark it 'File Deleted' by changing its directory letter to &FF (the ASCII code for Backspace and Delete).

You may use *DZAP to restore a file that has been 'deleted' in this way, but remember that, if the same name is being used in another catalogue, you should rename that file first.

7.2.4 *COMPACT

*COMPACT works as usual on the current catalogue, but will not operate on stored catalogues.

If you need to tidy the entire disc when several catalogues are present, one way of doing it is to *COPY *.* (all files) to another disc.

7.3 RUNNING PROTECTED DISCS

The 2.0 DDFS ROM will allow many protected discs, made for operation on the 8271 Floppy Disc Controller, to run on the 1770 FDC.

The way that this is done is by including in the 2.0 DDFS ROM code that allows a 1770 to emulate an 8271 in certain respects. The discs now supplied by Acornsoft, Micro Power and Island Logic are compatible with the Solidisk DDFS so the emulation is only intended to enable the running of other games discs. Most disc-based business software - including Clares Database - will run perfectly with a 1770 FDC.

Please check with your dealer to ensure that you have the latest versions of any discs produced by the above firms.

7.3.1 ELITE

ELITE runs perfectly without any special attention, unless you have an early copy of the game.

Early versions of ELITE can be made to run with the 2.0 DDFS ROM by holding the SHIFT key while switching on the computer. If the computer hangs up at this stage, try *MASKOFF then, then restart by SHIFT-BREAK. If this is still not effective, then try *MASKOFF then *SSTEP followed by SHIFT-BREAK. This time the program will more than likely run. However, if it still fails, then enter *MASKOFF and press SHIFT-BREAK again until it runs.

7.3.2 MINI OFFICE

To run MINI OFFICE, it is necessary to type in *ENABLE 80 (RETURN), followed by *MASKOFF (RETURN). Once this has been done, press SHIFT-BREAK and the program should run. If any problems are encountered, type in *FX 200,3 (RETURN), press the BREAK key and repeat the above procedure.

7.3.3 CASTLE QUEST

CASTLE QUEST requires you to enter *ENABLE 80 (if you are using an 80-track disc drive), followed by *MASKOFF (RETURN) - exactly as MINI OFFICE does.

7.3.4 OTHER PROTECTED DISCS

Most leading software producers are now well aware of the increasing number of DDFS users, and do make a real effort to make their products compatible.

7.4 THE STL DFS 2.0 AND SECOND PROCESSORS

The Solidisk DFS 2.0 ROM is capable of speeding up operations on the Acorn 6502 and Z80 Second Processors, for which disc accessing becomes even more important as the bottleneck. Even compared with the Acorn 1.2 DFS, the Solidisk 2.0 ROM can often double the speed of your programs.

However, along with such great enjoyment, there are some minor problems.

With the 6502 Second Processor, Robocom Bitstick is not wholly compatible with the STL DFS 2.1 in its present form. The reason is that the DFS 2.1 and Bitstick compete for some zero page locations.

To boot up with the Z80 Second Processor, proceed as follows:

Switch on the BBC and the Z80. Insert a system disc (or the Utilities disc) and do a CTRL-BREAK. If the system refuses to boot, press BREAK. Type *. then repeat CTRL-BREAK. It should now boot the system disc in just 2 seconds.

To format discs for use with the Z80:

Do not use the PREPARE or FORMAT programs since, although they will run if you have the 8271 FDC, they are slow.

It is quicker in the long run to prepare a master disc, formatted and holding several of the most frequently used files, such as PIP.COM, STAT.COM and BBCBASIC.COM, and to copy it in BBC mode.

The following procedure is valid for use with the 8271, but works even better with the 1770:

1) Switch off the Z80. Insert a blank disc in Drive B (i.e. 1/3) and format it for 80-tracks, single density on both sides (by *ENABLE S and *F80 1 then *ENABLE S and *F 80 3).

2) Place a system disc (or Utilities No 1 disc) in Drive A and backup side 0 (Drive A) to side 1 (Drive B) (by *ENABLE and *BACKUP 0 1). The disc in Drive B now contains CP/M system and directory tracks.

3) Switch on the Z80 and press CTRL-BREAK to boot CP/M.

4) Type in B: (RETURN) then ERA *.* and to the question "Erase all?", answer 'Y'. The disc in Drive B is now formatted but completely empty of files.

5) Type in PIP B:=A:PIP.COM
   PIP B:=A:STAT.COM
   PIP B:=A:BBCBASIC.COM

   and any other programs that you may wish to be copied onto every working disc.

If you REN (rename) BBCBASIC.COM as BOOT.COM, every time you boot up CP/M you will be in BBC BASIC (Z80).

The disc in drive B is now your MASTER disc and can be duplicated without even switching on the Z80.

To make a copy as a working disc, repeat steps 1 and 2 above, using the MASTER disc in place of the Utilities No 1. It will save you a lot of time!

Solidisk plan to produce a new BIOS for CP/M, allowing the use of double density (MFM) recording - giving 640K per disc in place of 400K. The new BIOS disc will also contain other programs, such as a disassembler for the Z80 and a disc sector editor and should cost about £5.00.

8. A SUMMARY OF THE STL DFS UTILITIES COMMANDS

The STL ADFS is designed to make use of a number of utilities contained in the STL DFS 2.1:

```
BUILD <fsp>
DCOPY <src drv> <dest drv>
DDFS
DISC
DISK
DOWNLOAD <fsp>
DSTEP
DUMP <fsp>
DZAP <trk> <sctr>
ENABLE 40 80 D S V
LIST <fsp>
LOADTAPE <fsp>
MASKOFF
MVADFS <afsp> <afsp>
MVDFS <afsp> <afsp>
MZAP <add>
RECOVER <trk> <sctr> <scrts> <add>
RESTORE <trk> <sctr> <scrts> <add>
RTRACK <drv> <trk>
SPEED
SSTEP
TAPEDISC <fsp>
TAPESAVE <fsp>
TYPE <fsp>
WORD <fsp>
WTRACK
```

Full details of these are given in the next section.

9. THE STL DFS UTILITIES IN DETAIL

*BUILD <file specification>, *DUMP <file specification>, *LIST <file specification> and *TYPE <file specification>.

These utility commands are available both to the STL DFS and the STL ADFS. In the Acorn ADFS implementation, they are provided as disc-based utilities.

*DCOPY <source drive> <destination drive>

This command allows the user to make backup copies of disks with non-standard (non-Acorn) formats. *DCOPY works on one track at a time, copying first the format and then the data.

In the case of the 8271 Floppy Disc Controller (FDC) (as fitted by Acorn), up to 10 sectors per track are possible, whereas with the 1770 FDC (as used by STL), up to 16 sectors per track are catered for.

However, there are a few limitations when using the 1770 to copy certain disks. This is because the 1770 cannot format any sector with an identification number (ID) over &F6. To get round this problem, a mask is used with any sector or track number greater than &F0. This mask has the value of &EF and is inserted automatically during *DCOPY when the 1770 is used.

At the beginning of the *DCOPY routine, the user is asked whether the sector lengths should be normalised or not. This is because some discs use a false sector length, which will cause the 1770 to crash out with a Cyclic Redundancy Check (CRC) error. This may be avoided by answering Y for yes to the question about normalising the sector lengths. This will then set all sectors encountered to a length of 256 bytes (the standard on the BBC Micro).

*DDFS, *DISC and *DISK

These are for entering the original disc filing system - the DFS. This has a single file catalogue, which can hold only 31 files - although these may be distinguished or grouped into directories having single character names.

*DISC (or *DISK, in America) enters it in the single density, FM recording mode, which has 10 sectors per track, and hence a 40-track disc surface provides 100K and an 80-track disc surface provides 200K.

This single density mode is the Acorn standard, and is the only one possible when using the 8271 Floppy Disc Controller.

*DDFS enters it in a "double density" MFM recording mode, which has 16 sectors per track. Hence a 40-track disk surface provides 160K and an 80-track disc surface provides 320K.

Both the single density and double density modes are possible when using e.g. the 1770 Floppy Disc Controller.

*DSTEP and *SSTEP

These two commands are used to tell the computer whether to single- or double-step the heads of any disc drives. DSTEP will have to be used if you are using an 80-track drive to read 40-track discs (assuming that you do not have, or use, a 40/80 track switch on the drive), DSTEP only has to be issued once at the start of each session.

Some protected discs will require one of these commands to be issued before running and some will even require it to be issued twice, if they hang up on SHIFT-BREAK. The correct sequence depends upon the size of the drive and on the disc, and will have to be found by trial and error.

*DZAP (<track>) (<sector>)

This is a utility to enable the contents of a disc to be examined and, if necessary, to be altered. If just *DZAP is typed, then both the track and sector will default to zero. If any others are required, then just type in the track, followed by a space and then the sector number - both in hexadecimal.

A display of the required sector will be given on the screen in rows, hexadecimal on the left, and ASCII on the right.

You can toggle the cursor between hex and ASCII by pressing the TAB key, and the current setting is shown at the top right of the screen.

This information may now be altered in either hex or ASCII. The cursor may be moved around the display by means of the arrow and shifted-arrow keys, data is entered at the cursor position, which is incremented after each change.

The sector and track position is changed by means of the CTRL-arrow keys. The current position is displayed at the top of the screen.

If a disc error is encountered, the error type is displayed below the status line. If the data was able to be recovered from the disc, then it may be modified and restored to the disc.

The current sector may be saved by pressing ESCAPE, whereupon a flashing prompt "SAVE Y/N ?" is displayed. If you require that sector to be saved to the disc, type "Y". Any other character will exit from DZAP without saving the sector.

*ENABLE can be used with various arguments, as follows:

*ENABLE 40 will make the machine single step on both 40- and 80-track discs, and is useful when using one 40- and one 80-track drive on the same computer.

*ENABLE 80 is similar to DSTEP, in that it informs the machine that 80-track drives are being used, and it should double step on 40-track discs.

*ENABLE D: Similar to entering *DDFS. It enables Double Density (MFM) recording before formatting (either with *F40 or *F80).

*ENABLE S: Similar to *DISC or *DISK. It enables Single Density (FM) recording before formatting

*ENABLE V: Enables Verify before formatting or backup - to be carried out immediately afterwards.

Suppose you want to format and verify a disc to 40-track, single density in an 80-track (non-switchable) drive, you should enter:

*ENABLE 80 (RETURN)
*ENABLE S (RETURN)
*ENABLE V (RETURN)
*F40 (RETURN)

*MVADFS <source *object specification*><destination *object specification*>
and
*MVDFS <source *object specification*><destination *object specification*>

These utility commands are for moving objects - i.e. directories and files - between the original DFS (in both the standard single density and the STL DDFS double density mode) and the ADFS.

*MVADFS is for moving FROM the ADFS
and *MVDFS is for moving FROM the DFS.

*MZAP <address>

The MZAP command is another utility, similar to DZAP in operation. However, it operates on the memory of the BBC Micro, rather than on a disc.

The start address of MZAP defaults to 0000, but another address may be specified if required (in hex), e.g. *MZAP 1900 for a start address of &1900.

Cursor movement is again controlled by the arrow and unshifted arrow keys, and data entered at the current cursor position.

Again the cursor may be toggled between hex and ASCII displays, with the status being indicated at the top right of the screen.

*RECOVER <track> <sector> <sectors> <address>
and *RESTORE <track> <sector> <sectors> <address>

These commands are used to recover data from - and then to restore it to - the disc. The information needed to direct these commands to the appropriate location on the disc is the track number, followed by the first sector required and the number of sectors. Finally, to define an area in memory where the data may be looked at and worked on, a single start address must be given. All of these numbers must be in hexadecimal and separated by spaces.

For example, to recover 5 sectors starting at sector 2 of track 0 to address 1900, you enter:

```
*RECOVER 0 2 5 1900
```

Data so recovered may be worked on with MZAP (which see) or may be saved as a separate file or restored to the disc.

The *RESTORE command uses the same information as *RECOVER but will transfer the data in the opposite direction - from the computer memory to disc.

These two commands are useful for recovering and/or repairing programs and other data from corrupted discs.

*RESTORE requires *ENABLE to be issued beforehand.

For example, to restore 5 sectors starting at sector 2 of track 0 from address 1900, you enter:

```
*ENABLE
*RESTORE 0 2 5 1900
```

*RTRACK <drive> <track> and *WTRACK

The *RTRACK command allows the user to recover all the sectors from a chosen track on the disc.

The ID fields are transferred to memory address &1800 upwards and the data fields to &4000 upwards. If the 1770 FDC is being used, the track information - exactly as read from the disc - is placed at &5000 upwards. Once the data is in memory, it may be examined and altered with the *MZAP command (which see).

Using the *WTRACK command will restore the ID fields to the disc, followed by the data field - exactly as seen in the memory from &4000 - by just loading the drive head and writing to the same physical track. *ENABLE must be issued before the *WTRACK command.

Using these two commands may sometimes enable damaged tracks to be completely recovered. They work automatically.

Another use is the alteration of ID or data fields.

Example

Assume that the drive number is 4.

First do *VERIFY to find the corrupted track - e.g. it stops at track 5. Then do *RTRACK 4 5

The *RTRACK command should correct any CRC error - which at least makes the data readable. If you like, you can also use *MZAP to correct the data, before writing it back to the track.

Finally, type *ENABLE (RETURN), *WTRACK (no arguments).

*SPEED

The SPEED command has been included to allow the changing of the track stepping time of the disc drive, independently of the keyboard switch or link settings. Permanent speed settings can be obtained by soldering 1 or 2 jumpers at the (former) keyboard switch location, at the lower right hand corner.

There are four possible speeds available and the exact result depends upon the type of floppy disc controller chip in use. The speeds are as follows:

FDC	SPEED	STEP	SETTLE	LOAD	LINKS
1770	0	30 ms	0 ms	0 ms	3, 4
1770	1	20	0	0	4
1770	2	12	0	0	3
1770	3	6	0	0	NONE
8271	0	24	20	64	3, 4
8271	1	6	50	32	4
8271	2	6	16	0	3
8271	3	4	16	0	NONE

*TAPEDISC (<file specification>)

The TAPEDISC utility allows most tape-based programs to be transferred to disc. This includes programs that are "Locked".

The program may be run be entering *TAPEDISC <fsp>. <fsp> may be the name of a specific file or just RETURN.

If a filename is specified, the program will look for and transfer just that one file. If however only the RETURN key was pressed, then the program will stay in a continuous loop and transfer as many files as it can find, either until the end of the tape or until the disc is full. To exit at any time from this mode, it will be necessary to press the BREAK key.

Once transferred to disc, many tape programs will run correctly. However, any containing portions of machine code will have to have these relocated to the correct start address - usually &0E00.

This can be done by using *DOWNLOAD <filename> (which see), down to &0E00. The program must then be started by a call to the original execution address, which will be displayed during the tape to disc transfer. Thus if the start address is &0E46, then the program must be started by typing in CALL&0E46.

Many tape programs call the next program on the tape with CHAIN"" or LOAD"". For them to run on disc, the "" must be replaced by appropriate filenames. A further point to remember is that [under the DFS], disc filenames can only be up to 7 characters long, whereas tape filenames may be up to 10 characters long. This will often lead to the filenames being truncated on transfer to disc, and these may need to be renamed in order to distinguish them and to ensure that they are called in the correct order.

Occasionally, files are found on tape which have no name. In this instance, the file sent to disc will be called "No-name", and once more may be renamed at any time.

*LOADTAPE <file specification>, *TAPESAVE <file specification> and *DOWNLOAD <file specification>

*LOADTAPE and *TAPESAVE are almost identical to *TAPEDISC, but will save everything on the tape to disc, regardless of the status of the tape block flag. These commands are used to recover certain protected tapes.

The *LOADTAPE command is issued first - to recover the data from the tape. When enough of the tape has been loaded, press ESCAPE to exit from the load routine. Once this has been done, use the TAPESAVE command to save the recovered file to disc.

*DOWNLOAD can be used to load BASIC or machine code programs which were originally written for tape. This it does by loading them at the current value of PAGE - e.g. &1900 - and then shifting them down in memory to the value of PAGE that they were written for - e.g. &0E00. For example:

If you type: *DOWNLOAD GAME1 (RETURN)
The computer will reply: "To &- "
Answer with an appropriate address (but not below 200 hex), such as 0E00.

If it is a BASIC program, you may run it by entering:

```
PAGE=&E00 (RETURN>
RUN (RETURN)
```

If it is in machine code, you will have to CALL the execution address, which can be found by entering: *INFO GAME1.

*WORD <file specification>

This is a simple word processor. It is not "fully featured" but is better than *BUILD!

To create a new document, simply enter *WORD <filename>

To edit an old document (e.g. FRED), you should enter: *WORD FRED. There must be a space between *WORD and FRED.

This causes the word processor to load the old document (FRED) into memory, and no alteration is made to the copy on your disc.

The last step in any word processing is to save the edited version to disc. A filename must be specified, otherwise an error will result. If you want to keep the old file (FRED), you should save it to a new file (e.g. JIM) thus:

!S JIM (RETURN)

If however you no longer want the earlier version, you can use the same name (FRED) again. The contents of the old file FRED will then be overwritten by those of the newly edited version.

In between editing the word processor and saving the document, you may enter any amount of text or editing commands (up to the memory limit). Whereas loading and saving operate on the whole document however, the editing commands work only on the CURRENT PARAGRAPH. A paragraph is defined as being preceded and ended by RETURN and may be up to 253 characters in length.

*WORD works in all screen modes and provides automatic word-wrap at the ends of lines, but it does not justify the right-hand edge.

You will see below that all *WORD commands are prefixed by the exclamation mark (or "pling") '!'. Only the first letter of the command is significant, so you do not have to type the command in full. You do not even need to end with a full stop (or period) - as when abbreviating commands in other programs. Some commands also take a number as argument. In this case, the number must be typed immediately after the first letter of the command.

The screen displays two dashed lines - usually with some text between them. This is the currently edited paragraph - which you can think of as a window.

*WORD works just like the BASIC line editor - with an entry line at the bottom and the current (and some previous or subsequent) paragraph(s) above.

f0 UP:

This will move the current paragraph "window" up one.

f1 DOWN:

This will move the current paragraph "window" down one.

!Q (QUICK):

This will move the current paragraph to the end of the text and the lower dashed line will then disappear completely. This will happen when you first create a new document or when you are adding more text to the end of an existing document.

In this mode, text is automatically inserted into/added to the document without you having to enter !I (INSERT). See below.

!D (DELETE):

This will delete the current paragraph. You may delete (the last) 1 to 9 paragraphs (including the current one) by adding a number immediately after !D. For example:

```
!D5 (RETURN) will delete the last five paragraphs.
```

!I (INSERT):

This will insert an extra paragraph just above the current one. The newly-inserted paragraph then becomes the current paragraph.

You may insert 1 to 9 paragraph(s) by adding a number immediately after !I. For example:

!I5 (RETURN) will insert 5 lines (which may each be expanded into paragraphs).

!* (SYSTEM):

This enables System (*) commands to be issued, such as *CAT, *EXEC and *MZAP etc.

With the exception of f0 and f1, the function keys may be programmed by you for the insertion of repeated words or phrases. For example:

```
*KEY 5 This phrase is needed repeatedly.
```

*WORD * RESTART WORD PROCESSOR

If an error occurs while you are using a System (*) command, you will lose no text if you proceed as follows:

1) Press the BREAK key
2) Enter *WORD * (RETURN), taking care to include the space before the second *.

LEAVING THE WORD PROCESSOR:

After saving the document e.g. with !S JIM (RETURN), press the BREAK key to leave the word processor.

PRINTING

To print the document, ensure that you are in BASIC (by typing *B. if necessary) and then enter:

```
VDU 2,12:*TYPE JIM (RETURN)
```

10. TECHNICAL INFORMATION ON THE STL DFS 2.1

10.1 FILING SYSTEM COMMANDS

This is of particular importance to those wishing to write their own disc software. Even if you are only interested in using existing programs, a quick read-through will still be beneficial.

The filing system is part of the Machine Operating System (MOS) ROM, which is fitted as standard in every machine. It deals with the storage of data and programs - e.g. on tape, disc and network systems.

The MOS ROM itself controls the tape filing system, but leaves the disc and network filing systems to additional software in a DFS or DNFS ROM.

A convention is then required so that the MOS can work with a DFS - whether from Acorn or someone else, such as Solidisk. This convention is now explained in some detail.

1) Unknown commands.
2) Seven file tasks - OSFILE (&FFDD), OSARGS (&FFDA), OSBGET (&FFD7), OSBPUT (&FFD4), OSGBPB (&FFD1), OSFIND (&FFCE) and OSFSC (&FF2A).
3) Unknown OSWORDs.

Let's start with the easy points of unknown commands.

10.1.1 UNKNOWN COMMANDS

All system commands going through OSCLI (entry point at &FFF7) will be either serviced by the MOS or, if unknown to the MOS, offered to other ROMs.

On the first pass, the unknown command is offered with service call 4. The Solidisk DDFS will check it against the list that is printed on *HELP UTILS. If nothing matches the command word, it will return the call to the MOS, which then passes it on to the other ROMs.

If unknown to the ROMs, it will be passed on to the current filing system.

10.1.2 OSFILE, OSBGET, OSBPUT, OSGBPB, OSARGS, OSFIND and OSFCV:

Some information about the use of these filing system tasks may be found in:

- the User Guide, pages 452 to 455.
- the Advanced User Guide, Chapter 16, pages 333 to 345. (Beware of the error on pg 335; the OSFILE entry point is &FFDD, not as printed!)
- the Advanced Disk User Guide, pages 121 to 263

The Solidisk Software package - specifically the Utilities disc - contains example programs showing the practical use of these filing system functions.

One important point about OSGBPB 8: This function returns a specified number of filenames in a specified directory - as used in many 'MENU' programs. The STL DFS 2.0 implementation however, will work with multiple catalogues.

Example:

```
 10 REM Program to read n filenames from disc directory $
 20 *DIR $
 30 HIMEM=&2000: fcb=&2000: OSGBPB=&FFD1: n=1000
 40 ?fcb=0: REM Set up file control block, directory
 50 !(fcb+1)=&2100: REM Data storage
 60 !(fcb+5)=n: REM choose n as large as you like
 70 !(fcb+5)=0: REM start from the beginning
 80
 90 X%=0: Y%=&2000: A%=8: CALL OSGBPB
100 *MZAP 2100: Inspect result
```

10.1.3 UNKNOWN OSWORDS

Three OSWORDs are dealt with by the Solidisk DFS.

OSWORD 7D

This OSWORD supplies the 'Cycle Number' of the requested disk.

To use this OSWORD, you must designate some free memory, into which the DFS can return the result.

For example:

```
 10 HIMEM = &2000:REM Make some memory free above &2000
 20 A%=&7D:X%=0:Y%=&20:CALL &FFF1: REM This last calls OSWORD 7D [with
parameter block at &2000.
 30 PRINT ?&2000:END
```

This program will read and then print the Cycle Number - i.e. the number of times that the disc has been written to since being formatted. The number is in hex/decimal, and only goes up to 99 before starting again at 0.

The Cycle Number is often used as a way of detecting whether someone has been tampering with the disc. For example, it is used by *WIPE *.* to detect whether you have removed the disc during the questions about deleting each (unlocked) file.

OSWORD 7E

This OSWORD returns the disc size - in sectors, as a hexadecimal number. For example:

```
 10 HIMEM=&2000: REM Make some memory free
 20 A%=&7E:X%=0:Y%=&20:CALL &FFF1:REM This last calls OSWORD 7E with
parameter block at &2000
 30 PRINT "Disc size in bytes = &";?&2002;?&2001;?&2000
```

The Solidisk DFS replies with three bytes - low, mid, high.
The same number is then shown in the top left corner of the disc directory [header] - see after *CAT.

OSWORD 7F

This is the most complicated command that the Solidisk DFS has to deal with.

Firstly, if the 1770 Floppy Disk Controller (FDC) is being used, the Solidisk DFS will translate the command code and results to match the response of the 8271 FDC (which is the standard for the BBC machine).

If the 8271 is being used, 40-track discs are checked for, and double-stepping performed as required.

The general format for OSWORD 7F is as follows:

Parameter Block:	Location	Contents
0		Drive number (0-3 or FF if same)
1		Data Address low
2		Data Address high
3		High order (FF or 00 if I/O only)
4		High order (as above)
5		Number of details (0-3)
6		Command code (see table below)
7	First	detail if any (usually track number)
8	Second	detail if any (usually sector number)
9	Third	detail if any (usually sector size + number of sectors involved)

NEXT LOCATION Result

Note that the location for the result is not fixed. E.g. if 3 details are supplied, the next location will be Parameter Block +10, but if no details is given (as for Read Status), the next location will be Parameter Block +8.

If you wish to use OSWORD 7F in your programs, here is an example:

```
 10 HIMEM=&2000: REM Make some memory free
 20 INPUT "Read from Track= "track
 30 INPUT "and from Sector= "sector
 40 INPUT "How many sectors (1 to 10)= "n: REM No more than 10 sectors
 50 REM Build parameter block
 60 block=&2000:?block=&FF: REM same drive
 70 block!1=&FFFF2100: REM Data will be sent to &2100
 80 block?6=&53: REM Read command
 90 block?7=track: REM Starting from
100 block?8=sector
110 size=&20: REM 256 bytes/sector
120 block?9=n + size
130 A%=&7F:X%=0:Y%=&20:CALL &FFF1: REM Do OSWORD 7F
140 IF block?10: GOTO 130: REM Retry if result is bad
150 *MZAP 2100: REM Inspect data read
```

You can also use OSWORD 7F on a double density disc (when you can specify up to 16 sectors)

Remember

1) Deal with one track at a time.
2) Check the result. If it is not zero, repeat the last command (i.e. retry).

OSWORD 7F COMMAND TABLE

Command	Code	Details (e.g. other parameters)
Seek	&69	Track number
Read Status	&6C	None
Write Spec Reg	&5A	Reg No, data
Read Spec Reg	&5D	Reg No
Read Sectors	&53	Track, Sector, No of sectors to be read
Read deleted	&5B	Track, Sector, No of sectors to be read
Read ID	&5B	Number of IDs
Verify sectors	&5F	Track, Sector, No of sectors to be verified
Format	&4B	Track, Gap3-6, Size/No of sector, Gap5-6, Gap 1-6

NOTES

Seek = This moves the drive head to the specified track.

Status = This is the FDC status. The bit pattern is as follows:

D7	D6	D5	D4	D3	D2	D1	D0
0	RDY1	WRTFAULT	INDEX	WRTPROT	RDY0	TRACK0	0

When working with the WD1770, the STL DFS will return &44 for 'Drive Ready' as the WD1770 does not require a 'drive up to speed' signal.

Special Registers:

The 8271 has 14 special registers, but the WD1770 has none. However, when working with the WD1770, the STL DFS maintains 4 pseudo special registers (the current track registers) - to emulate the 8271.

Read Data and Read Deleted Data.

Each sector on the disc is composed of an ID field and a data field.
The ID field contains 6 bytes - as displayed when using *RTRACK.
The data field can contain 128 or 256 bytes.
The sector can also be marked deleted - as shown by the data mark &C8.
The STL DFS will read both (non-deleted) and deleted data.
The deleted data flag (&20) will be returned to the result byte.

Format:

Although the STL DFS has a built-in formatter, optimised for both 8271 (2 sector skew) and WD1770 (1 sector skew), you may want to run special disc copy programs, with their own formatter. Such programs will only work if you have the 8271 chip. The WD1770 cannot format tracks or sectors greater than &F5. The sector mask (&EF) will then be used automatically to permit formatting, but the resulting disc will not work satisfactorily.

To use OSWORD 7F efficiently, you will need the complete specifications on the Intel 8271 FDC. This is available from Intel, 3065 Bowers Ave., Santa Clara, California, USA. Tel: 408) 987 8080.

A new standard - OSWORD 72 - has been introduced for the Acorn ADFS - for use with double density (MFM) floppies (as on the Electron) and with the Winchester drives.