A.D.U. contains 15 commands, several of which are direct replacements for those on the 'Acorn' ADFS Utilities Disc. A.D.U. will work with any version of the BBC computer. It will work with Winchester Discs (not applicable to the 'Compact') and floppy or 3.5 inch discs (not on Model 'B' unless 1770 floppy disc controller fitted).

A.D.U. cannot be used with the model 'B' which has an 8271FDC, because the 8271 cannot work in Double Density. However the 'Electron' does have a 1770 fitted.

When the B+ was released with its 1770 FDC replacing the 8271, the ADFS could be used for both floppy discs and Winchesters. This continues with the Master series though the 'Compact's ADFS does not have the Winchester controlling routines

The ADFS recognizes 3 layouts of floppy/3.5 inch disc, these are 40,80 and 160 tracks ( or more commonly called small, medium or large). The latter convention is used in ADFS Utilities.

The ADFS recognizes 4 drives in all. With no Winchesters in the system the floppy/3.5inch drives are called 0 or A and 1 or B. When a Winchester is present the Winchester becomes drive 0 or A, drive 1 or B is reserved for a second Winchester, and the disc drives become 4 or E and 5 or F. Drive 'D' is never used in ADFS on Model Bs or Masters.

A.D.U. works with the Solidisk ADFS , though owners of Issue.1. boards (i.e. those that cannot be driven by an 'Acorn' ADFS) must use their own disc formatting routine because the FDC appears at a different place in memory on the two boards. ADFS Utilities allows numbers or letters to be used but insists on correct notation when clarity is required (e.g. in a non-Winchester system, 4/E and 5/F also refer to the disc drives, but the recognized numbering of 0/A and 1/B should be used.

A freshly formatted ADFS disc contains 2 fixed areas occupying the first 7 sectors of the disc. The first 2 sectors are the 'free space map'. This is a list of all the free space on the disc and where it is. It is essential to the operation of the disc, and is protected by a check sum in each sector. If this check sum is incorrect, a 'Bad F.S Map' error will be given. ADFS Utilities Disc Editor routine (DISCEDIT) will generate the correct check-sum if it is used to alter the free space map which isn't recommended unless you really have to. The next 5 sector are the 'root' (or base) directory.

An ADFS directory may hold up to 47 objects. An object may be either a file or another directory name. If it is a directory, then that directory may hold a further 47 objects, which could again be a mix of any files or directories. Although there are limits on the nesting of directories, imposed by the ADFS it is more practical to work to a lower limit and never nest directories to the extent where the path-name is longer than may be entered into the computer (no more than 250 characters, allowing for *DIR). ADFS Utilities assumes that no path-names exceed 120 characters.

The ADFS holds a copy of the 'free space map' and the current directory in memory. If this is corrupted a 'Bad sum' error will be given (and the system must be re-set). Consequently you should never change an ADFS disc without using *MOUNT to tell the ADFS that it has been changed. ADFS Utilities accesses this area for some of it's operations, so some routines may produce strange effects if the ADFS is confused about the current directory. As a rule the problem will be detected and the error message 'ADFS workspace corrupt' printed, though perfect protection cannot be guaranteed. If the message is displayed then the computer should be re-set either with CTRL/BREAK or *PWRBRK as soon as convenient.

Ex DFS users should read the next 2 paragraphs carefully; they explain the concept of 'Libraries' and the *COPY command, both of which are different in DFS.

The Library is a directory, usually within the 'root', which contains 'machine code' programs. For an example of this philosophy in use, examine your 'Acorn Utilities' disc with *CATALL. Whenever a *<name> or *RUN<name> cannot be found in the current directory, the ADFS will check the Library before issuing an error. This gives you up to 47 more machine code programs available from ANY directory (and is one of the reasons why ADFS access always appears slow). The current directory may be examined by using *CAT and the current Library may be examined by using *LCAT. Similarly *LEX does for the Library what *EX does for the current directory.

The syntax of the *COPY command is totally different in ADFS to the DFS. In DFS you had to provide 3 parameters, the source drive, the target drive, and the file(s) to copy. The ADFS uses only 2 parameters. The first specifies the file(s) to copy and second specifies the directory in which they are to be placed. The ADFS *COPY will not copy directory names, or their contents (use *DIRCOPY instead). Some examples will illustrate the idea:

```
*COPY $.game $.Basic
```
copy a file called <game> into directory $.Basic

```
*COPY text* :4.$
```
copy every file called text(anything else) into the root directory of drive 4.

```
*COPY :0.$.* :1.$
```
copy all the files in the root directory of drive 0 into drive 1.
N.B. This does not necessarily constitute a back-up of a disc. If there are any sub-directories on the disc, in drive 0 they will not be copied. Use instead *DIRCOPY or *BACKUP.

*DIRCOPY should be used to add the copied files to those already on the target disc and *BACKUP will produce an exact copy of the disc in drive 0 on the disc in drive 1 (so overwriting anything previously on the disc in drive 1).

Files may be deleted by using *DELETE or *REMOVE for single files, or *DESTROY for 'ambiguous filespecs' eg:-

*DESTROY * would delete every file in the current directory. A directory is protected by the 'L' (locked) attribute and may not be deleted until:-
i) The 'L' attribute is removed and
ii) It is empty.

Emptying redundant directories is quite a chore, and *DIRDESTROY will soon make it's worth apparent.

ADFS Utilities should be placed in a socket of lower priority than the ADFS. This is not due to any failing of A.D.U. or the ADFS but to avoid a problem with the 'Acorn' Utility Programs, which use *AD. to re-select the ADFS, which with ADFS Utilities in a senior socket, will be taken as *ADU.

All the commands may be abbreviated, and any name clashes may be resolved by using the letter 'Z' to prefix the command. Upper and Lower case may be used. For those who insist on American spellings, the Disc Editor (DISCEDIT) may also be started with DISKEDIT,

If any A.D.U. command name clashes with a filename you should *RUN<name> or */<name> to execute the disc file.

If you want to disable ADFS Utilities totally, you may use *KILLADU, after which no *commands will be taken by A.D.U.

*HELP will produce 'Killed ADFS Utilities'

A.D.U. may be brought back at any time by typing *HELP A.D.U. which will resurrect A.D.U. and display the usual HELP information.

The ADFS Utilities HELP text displayed by typing *HELP A.D.U. is shown below

```
ADFS Utilities 1.00
ADU
BACKUP <source> <target>
CATALL (<dir>)
DFSADFS <DFS drive> <ADFS dir>
DIRALL (<dir>)
DIRCOPY <$.afsp> <$.dir>
DIRDESTROY <dir>
DISCEDIT
DRIVE (<drive>)
FORMAT <drive> (<size>)
KILLADU
MENU (<dir>)
PWRBRK
VERIFY <drive>
VFORMAT <drive> (<size>)
```

Although the parameters are fully explained in the sections for each command, some brief notes will help people who want to dive straight into using ADFS Utilities. Any parameter in () brackets may be omitted.

| | |
|---|---|
| <source> | An ADFS drive specification (0145ABEF), which will contain the disc providing data for the transfer. |
| <target> | An ADFS drive specification (0145ABEF), which will contain the disc receiving data from the transfer. May be the same as <source>, in which case prompts will be provided to change discs. |
| <dir> | An ADFS directory path-name, may also include a drive specifier (e.g. :4.$.BASIC). |
| <DFS drive> | The number of the drive (in DFS nomenclature) which will hold the DFS disc whose data is to be transferred to ADFS |
| <ADFS dir> | An ADFS directory specification for the ADFS disc which will receive the data from the DFS disc. |
| <$.afsp> | An ADFS filespec/path-name. It must use $ (i.e. be referenced from the 'root'), and may be a directory or file. It may contain the wild characters '*' and '?'[1]. |
| <$.dir> | An ADFS directory specification which must be referenced from the 'root' directory by inclusion of the '$' character. |
| <drive> | An ADFS drive specification (0145ABEF). |
| <size> | The size of disc to be created. Use 'S' (small), 'M' (medium) or 'L' (large). |

In the instructions that follow, please remember that reference to floppy discs also refers to 3.5 inch discs and that in either case the presence of a 1770 (or 1772) FDC is assumed. If you are using a Model 'B' still fitted with an 8271 and a Winchester disc under ADFS you should seriously consider upgrading to a 1770 based system which will allow you to use ADFS on floppy discs.

ADFS Utilities uses 4 areas of memory as workspace. It uses various stack locations for temporary storage, some zero-page locations allocated to 'ECONET' (between &0090 and &009F). The major routines also use parts of pages 'B' and 'C'. On a Model 'B'-'B+' this will result in loss of function key definitions and defined characters between 128 and 159 (assuming an exploded font). On a Master series machine these 2 stages will be re-allocated for the 'ECONET'- so on a Master series machine there will be no interference to function key definitions or character definitions. *FX18 can be used to clear the function key area, and *FX20 to implode the font.

Some routines also use the main memory for workspace. The approximate use of memory is explained with each routine.

---

[1] Personal note:- I think that '?' should probably be '#'.

## **ADU*

| | |
|---|---|
| Purpose: | Selects ADFS Utilities as the current language |
| Memory used: | As *MENU |
| Syntax: | *ADU |

This command first tells the Operating System to select ADFS Utilities as the current language. Once it has established itself, it will start the MENU routine. If the MENU is left, by pressing ESCAPE, you will see a '*' prompt and any commands entered, will be passed to the O.S. for distribution.

The language entry (as invoked by *ADU) is included to allow the system to start up in the MENU routine. By placing ADFS Utilities so that it is the senior language in the machine, it will be entered when the machine is switched on, whereupon the MENU routine will try to read in the 'root' directory from Drive 0.

## **BACKUP*

| | |
|---|---|
| Purpose: | Copy ALL data from one ADFS disc to another |
| Memory used: | Pages B & C and all user memory (including Tube) |
| Syntax: | *BACKUP <source Drive No.><Target Drive No.> |

Examples:

```
*BACKUP 0 1
*BACKUP :0 :0
*BACKUP 0 4
```

This command copies all data from one ADFS disc (the source) onto another (the target).It will not copy beyond the highest used sector on the source disc, that means that back-ups between discs of varying sizes are possible, as are Winchester to/from floppy disc back-ups

The same drive may be used for the source and target discs and prompts will be printed when a disc needs changing.

This routine will copy between discs of different sizes and will allow you to create back-ups of Winchesters and multiple floppy discs. The routine can also re-load a Winchester from floppies. Whenever the routine needs another source or target disc, it will prompt 'Insert NEW.disc'.

The routine will examine both discs involved and work out how many 'source' and 'target' discs will be used. It will display it's results and then ask for confirmation to begin the transfer. When prompted, type in YES and press RETURN to proceed or any other character to abort.

When making back-ups from a large disc or Winchester onto smaller multiple discs, the multiple discs will not themselves be usable except for recopying back to a disc or Winchester of the original size

The routine uses all the available memory in the main processor and all available memory in a 2nd processor if one is available. When using 2nd processor memory, the language in use, will be overwritten, so this routine will re-select the language when it finishes, which causes it to be copied into the 2nd processor again

If the routine completes successfully, it will re-select the 'root' directory of the current drive. If an error occurs during the routine, the current drive will be 'Unset' and should be selected with a *MOUNT, *DRIVE or *DIR command

Most DFSs also have a command *BACKUP. To avoid name clashes, this routine will only execute if ADFS is the current filing system. On other occasions ADFS Utilities will not claim the command, so allowing it to pass on to the DFS

Several errors are possible:

Each message is given below, with an explanation of it's cause.
Parameter(s) missing:-
*BACKUP must be followed by 2 drive numbers, optionally prefixed with a colon(:). This error is generated if one or both drive numbers were not given

Can't change Winchester:-
You have entered *BACKUP 0 0 or *BACKUP 1 1 on a system with an active Winchester disc. This is ridiculous because it is impossible to to take out the Winchester and put in another

Bad F.S Map:-
This error could be given for one of 2 reasons. The first possibility is that an ADFS disc was not MOUNTed before this command was started. The second possibility is far more serious because it suggests that the 'Free space Map' check-sum of the 'source' or 'target' disc is wrong. It can be corrected by using DISCEDIT (read in, edit, and save the sector - DISCEDIT will recalculate the checksum for you), but there is still some reason why it went wrong in the first place. The only expectable cause would be if you had mistakenly specified a non-standard 'source' disc (eg The second and subsequent discs of a Winchester back-up) which does not have an F.S Map or directory on the disc.

F.S Map full:-
This error is very unlikely since it can only occur when a disc with a full FS Map, whose last sector is allocated to an object, is being copied to a disc which is different in size

## *CATALL

| | |
|---|---|
| Purpose: | Display all objects residing in or below the current directory, together with information about the data in the file and it's addresses |
| Memory Used: | Page 'B' and 6 pages above OSHWM |
| Syntax: | *CATALL (<dirspec>) |

Examples:
*CATALL
*CATALL $.Basic

This command is best considered as a combination of the Basic programs EXALL and CATALL on the Acorn Utilities disc.

Starting from the current drive and root directory (unless a directory was specified with the command) it will list all objects in the directory, entering any sub-directories it encounters. Each new sub-directory will be indented on the screen by 3 spaces, giving an effect similar to a BASIC program viewed in LISTO 7

For each file the routine will display it's name, attributes, type(explained below), load and execution addresses, and it's length and start sector on the disc.

The routine can recognize 6 different types of file:
BASIC (as saved by BASIC itself), VIEW documents, Data files, ROM images, Machine code and VIEWSHEET spreadsheets.

However there are several ways the routine can be caught out; here are just 3 examples

BASIC programs not saved by BASIC, e.g. saved by the BASIC editor (they will appear as machine code)

VIEW files copied from DFS onto ADFS. The ADFS allows a 4 byte 'load' address (unlike the DFS) which is tested to separate VIEW and VIEWSHEET files. On files copied from DFS, this address will not be set correctly and the file will be mistaken as a VIEWSHEET spreadsheet.

Sabotaged load/execute addresses, e.g. by the utility recently published in one magazine to allow date stamping of files.

The routine can generate three errors:

Not Found/Bad Name
The parameter entered as an optional directory name, is not a directory

Only in ADFS
Select the ADFS and then start again

Disc Error
A disc error was found by ADFS whilst trying to read a directory

## *DFSADFS

| | |
|---|---|
| Purpose: | Transfer all files from DFS to ADFS in one pass |
| Memory used: | Pages 'B' & 'C' and main memory up to screen start |
| Syntax: | *DFSADFS <DFS disc drive No.><ADFS dir> |

Examples:

```
*DFSADFS 1 :0.$.disc10
*DFSADFS 0 :0.$
```

This command provides a faster alternative to CopyFile for transferring files from DFS to ADFS. It must have 2 parameters, a drive number for the DFS and a directory for the ADFS. If they are on the same drive, prompts will be issued to change discs. You should be in ADFS when you issue the command and will be left in ADFS upon completion. Every file on the DFS disc is transferred, with any ADFS sub-directories required being created within the directory specified in the command line. In contrast CopyFile has to be used once per DFS directory, so several passes are often needed, sometimes copying just one file

Unless you are using the same disc drive for DFS and ADFS no user interaction is required once the transfer has started. Progress messages will be displayed on the screen to show what is happening, and a message will also be displayed when the routine completes.

Before beginning to transfer each file the routine will check whether the name already exists on the ADFS disc. If it is a directory, the routine will stop, with the error message 'File exists as a Directory'. In other cases one of the 3 prompts below will be displayed

a) 'File exists. Overwrite (Y/N)'
b) 'Locked file exists. Overwrite (Y/N)'
c) 'E protected file exists. Overwrite (Y/N)'

The possible responses to the above prompts are identical. Pressing ESCAPE will stop the routine with the error message 'Aborted'. Pressing 'N' will cause the routine to move on to the next file to be transferred without overwriting the file already existing on the ADFS disc. Pressing 'Y' will cause the file On the ADFS, to be 'Unlocked' and then deleted, after which it will be re-written. No other key-strokes are accepted
This routine can generate several errors:

Parameters required
Both parameters are compulsory

DFS drive should be 0-3
The first character tested, which should have been DFS drive number was outside the range 0-3

ADFS directory not found
The directory name entered can not be reached from the current directory. If you are uncertain, specify a full pathname from '$'

DFS disc error

The routine was unable to read the catalogue on the DFS disc. Ensure that the DFS disc is in the correct drive and that the correct disc is inserted if one drive is being used.

No files
The DFS disc catalogue is empty, so there is nothing to be transferred

DIR name used for a file
The ADFS already has a file of the same name as the directory that this routine wanted to create. These will be single letter names like DFS directories

File exists as a directory
The filename which the routine was trying to transfer, is already used as a directory name on the ADFS disc. There are several ways around this problem.

i) Destroy the directory using *DESTROY
ii) Rename the directory (you have to remove the 'L' attribute first with *ACCESS name R). The directory does not need to be emptied to be RENAMEd. Remember to put back 'L' attribute with &ACCESS name LR.
iii) Create a new ADFS directory for the entire transfer to go into.

Unable to open ADFS file, Aborted
Various possible reasons, the most likely being a full directory.

Unable to open DFS file, Aborted
Very unlikely, the only reason being an attempt to copy a DFS disc where the track and sector numbers have been manipulated.

## *DIRALL

| | |
|---|---|
| Purpose: | Displays directory structure |
| Memory used: | Page 'B' and 6 pages above OSHWM |
| Syntax: | *DIRALL (<dir>) |

Examples:

```
*DIRALL
*DIRALL :1.$
*DIRALL :E.Basic
```

This command behaves like *CATALL with the sole exception that only directories are displayed. For other details please refer to CATALL

## *DIRCOPY

| | |
|---|---|
| Purpose: | Copy directory/file structure |
| Memory used: | Pages 'B' & 'C' and main memory up to screen start |
| Syntax: | *DIRCOPY <$.afsp><$.dir> |

Examples:

```
*DIRCOPY :0.$ *1.$
*DIRCOPY $.Basic :1.$.back.Basic
```

This routine is similar to DIRCOPY on the 'Acorn' ADFS Utilities disc. It copies an ADFS structure from one place to another, taking with it, all files and sub-directories. Care should be taken not to set up a recursive loop (e.g. *DIRCOPY :4.$ :4.$.Basic). Incidentally, the example just given CAN be achieved simply by using the ADFS *COPY command.
*COPY :4.$.* :4.$.BASIC (which acts only on files)

The first parameter must be a 'filespec' or a 'directory name', but it must be prefixed by a full 'path-name', i.e. It must contain '$'. The second parameter must be a directory name and again, must be a full 'path-name'. Once started, no user intervention is needed and progress messages will be displayed while the routine works and when it finishes.

If you want to see the object names as they are transferred, you should enter the command *OPT1,2 before the *DIRCOPY command. The display of filenames can be turned off with *OPT1,0.
Several errors can be generated by this routine.

$ must be used
The 2 pathnames required must contain a '$', i.e. they must be full path referenced from the 'root' directory. The drive specification is optional unless needed to differentiate between two discs.

Two parameters required
Two 'path-names' must be given

Access violation
The 'path-name' is the 'filespec' of an 'E' protected file

## *DIRDESTROY

| | |
|---|---|
| Purpose: | Remove part of the directory structure |
| Memory used: | Pages 'B' & 'C' |
| Syntax: | *DIRDESTROY <directory> |

Examples:
*DIRDESTROY :5.$
*DIRDESTROY basic.games1

This routine should be used with extreme care and forethought. It removes the directory name given from the disc, taking with it, any sub-directories, and files within the directories or any of it's sub-directories. *DIRDESTROY $ will clear every file and directory on the disc except the root directory '$' which cannot be deleted.

The routine will check that the directory given exists, and then ask you whether it should begin. Type 'YES' and press RETURN to confirm your intention. The next question allows you a measure of second thought about your action. Each group of deletions is performed by the ADFS command *DESTROY which requires the response 'YES' to perform the deletion. *DIRDESTROY can either let you type in 'YES' and press RETURN, so affording the opportunity to abort the routine, or provide the 'YES' itself. If you answer 'Y' to the second question 'Prompt Individually', the routine will wait for you to answer 'YES' and press RETURN confirm each group of deletions.
Several errors can be generated

ADFS workspace is corrupt
The memory used by ADFS has been overwritten. The system should be CTRL/BREAKed as soon as possible. *PWRBRK may even be necessary.

Aborted!
The answer to the 'Begin?' was not 'YES' followed by the RETURN key, or the answer to ' Prompt Individually' was ESCAPE, or the answer to a 'DESTROY?' prompt was not 'YES' followed by the RETURN key.

Not a directory
The 'path-name' given does not lead to a directory

Wildcard
You used a 'wildcard' in the directory which is to be deleted. This error will occur at the end of the routine; all other deletions will be successful. To illustrate the error, the following are valid commands:-

```
*DIRDESTROY $.A*.B*.games
*DIRDESTROY A.B
*DIRDESTROY $.A.C.
```

but this command will cause a 'wildcard' error when the routine attempts to delete the directory:-

```
*DIRDESTROY A*.B*.G*
```

## *DISCEDIT/*DISKEDIT

| | |
|---|---|
| Purpose: | Disc sector editor for ADFS discs |
| Memory used: | Pages 'B' & 'C' |
| Syntax: | *DISCEDIT or *DISKEDIT |

This routine is a disc sector editor for use on an ADFS disc. Unlike some other sector editors, it can circulate the checksum required for the 'free space map' sectors, so avoiding a 'Bad FS Map' after fiddling with it. Be warned though, that inserting invalid bytes in the 'FS map' will also cause a 'Bad FS Map' (e.g. fouling up things such that 'used sectors'+'free sectors' does not equal the number of sectors on the disc).

The first action of this routine is to read in the first sector on the disc, and identify the size of the disc. This number is then displayed in Hexadecimal in the form 'Sector current sector/maximum sector' above the sector data. The contents of sector zero will then be displayed.

Movement around the disc is made by using the 'arrow' keys. Pressing the upwards key, causes the last sector on the disc to be read. The downwards arrow causes the first sector to be read.

The effect of the left and right 'arrow' keys varies with the SHIFT and CTRL keys. With neither pressed, the lefthand arrow reads in the sector before the current one (i.e. sector-1), and the righthand arrow reads in the sector after the current one (i.e. sector+1).

If the SHIFT key is held down when a left or right arrow is pressed, the change is &10 sectors and the CTRL key causes a &100 change.

It is worth remembering that DISCEDIT can be started from MENU, in which case it will read sector zero first to get the disc size, and then read in the first sector of the object under the cursor when 'K' was pressed in the MENU. This approach is usually more convenient if you want to edit data in a known file.

Once the sector to be edited has been reached press the COPY key. The prompt message at the bottom of the screen will change and the contents of byte 0 in both the Hex and Character tables will be highlighted.

At the bottom of the screen you will see 'HEX mode'. Data may be inserted into the sector by either typing in the component characters of a hexadecimal byte (e.g. pressing 'E' and 'A' for &EA). This is the initial mode on entry to the editor. By pressing the TAB key you can switch into ASCII mode, which allows the ASCII code of the character you type to be inserted into the byte. You can switch back to Hex mode by pressing TAB again.

In Hex or ASCII mode, the value entered will be placed in the highlighted byte. The highlighting will then move along to the next byte. You may also move the highlighting around the sector by using the arrow keys.

When you have finished altering the sector press the COPY key. A prompt will be displayed asking whether you want to save the amended sector onto the disc. Press 'Y' to save the sector. If you have been editing sectors 0 or 1 and choose to save it, a new 'check-sum' byte will be calculated and inserted. This is done to prevent a 'Bad FS Map' error when ADFS next tries to use the 'free space map'.

You can then move around to another sector and edit it, or press ESCAPE to leave the routine. If any errors occur whilst trying to read or write a sector, the error number will be displayed at the bottom of the screen with the prompt 'Re-try (Y/N)'. If you press 'Y' the routine will repeat the operation which failed, otherwise the data which was read will be displayed and the current sector number updated.

## *DRIVE

| | |
|---|---|
| Purpose: | Increase compatibility with programs written for DFS |
| Memory used: | Nil |
| Syntax: | *DRIVE (<drive>) |

ADFS 1.30 (B and B+) and 1.50 (Master128) do not have a *DRIVE command. Consequently, any programs written for DFS need every occurrence of DRIVE changed to MOUNT to work on ADFS.

This routine gets around that problem. The parameter entered with this command is given to the ADFS as a *MOUNT command, so DFS programs can run unaltered.

To avoid clashes with the DFS this routine will only work whilst ADFS is active, under any other system it will not process the command, allowing other *DRIVE commands to function normally. The other commands which behave in this manner are *BACKUP and *VERIFY, all the other ADFS only commands in this ROM will generate an error if ADFS is not active.

## *FORMAT

| | |
|---|---|
| Purpose: | Format a floppy/3.5 inch disc for ADFS use. |
| Memory used: | Page 'B', main memory |
| Syntax: | *FORMAT <drive> (<size>) <drive>-(0,1,4,5,A,B,E,F) (size>- (S,M,L) |

Example:
*FORMAT 0
*FORMAT 1S

This routine may be used in place of AFORM for formatting a floppy/3.5 inch disc. It can format a disc for ADFS use with 40, 80 or 160 tracks of storage (subject to the physical limitations of the disc drive). It can only be used when a 1770 or 1772 FDC is present, and cannot format Winchesters (which are usually formatted by sending a command to the interface for the Winchester)

With a 40 track drive you can only use the 'S' (small) size, which formats one side of the disc. If you have double sided 40 track drives, you might as well format the second side under DFS (drives 2 & 3) and have the benefit of a dual standard disc.

With an 80 track drive, you can use the 'M' (medium) size, which uses one side of the disc. If the drive is double-sided you can format the second side for DFS or use the 'L' (large) size.

So, to summarise the possible sizes:-

| | | | | |
|---|---|---|---|---|
| S | Small | 40 track | &280 sectors | 160k bytes |
| M | Medium | 80 tracks | &500 sectors | 320k bytes |
| L | Large | 160 tracks | &A00 sectors | 640k bytes |

A drive number must be specified in the format command. Normally it will be 0/A or 1/B, but if a Winchester is present, will be 4/E or 5/F. The size can be specified with S, M or L, though it will default to 'L'.

The disc to be formatted should be placed in the drive before answering the confirmation prompt.

The routine will ask you to confirm that you wish to format a disc, type in 'YES' and press RETURN to proceed. It will then test the disc to see whether it is already formatted in double density.

If the disc has been formatted before, a warning will be displayed. Press 'Y' to format the disc or 'N' to abort.

The routine will attempt to format the first track. If the disc is write-protected, an error message will be displayed and the routine will finish.

The disc will then be formatted. A count of the track being formatted will be displayed. For 'small' discs it will count up to&27, on 'medium' or 'large' discs, it will count to &4F. Formatting 'large' discs will appear slower than 'medium' because it is formatting both sides of the disc.

If you want to verify the disc, you may either follow the *FORMAT with *VERIFY or use *VFORMAT which combines the 2 commands.

If the routine completes successfully, it will select the 'root' directory of the newly formatted disc. If an error occurs during the routine, the current drive 'Unset' and should be selected with a *MOUNT, *DRIVE or *DIR command

If you are in screen modes 0, 1 or 2 on a machine without active 'shadow' screen memory, the screen may be overwritten by data for the track being formatted. This will not cause the FORMAT to fail even if the track number display appears in the overwritten section.

Several errors can be produced by the routine:-

ONLY IN ADFS
Select the ADFS with *ADFS or for a blank disc *FADFS

Illegal character (Use 0145ABEF SML)
A character other than those above was read. The first parameter must be a drive identifier, and the second (optional) parameter is the size of the disc.

WINCHESTER present - use 4/E 5/F
There is a Winchester disc in the system and you used one of the Winchester identifiers 0/A 1/B instead of one of the floppy/3.5inch disc identifiers 4/E 5/F

No WINCHESTER - use 0/A 1/B
You used 4/E 5/F as a drive identifier in a system without a Winchester drive present

Write Protected !
The disc is protected by a write-protect label. The electronics in the disc drive will not allow the disc to be overwritten whilst the label is in place

Note that this routine will not work with 'Solidisk' issue No.1 boards. Use Solidisk's own formatting routine.

## *KILLADU

| | |
|---|---|
| Purpose: | Tell ADU to 'kill' itself (i.e. not accept any commands) |
| Memory used: | Nil |
| Syntax: | *KILLADU |

This command allows you to disable ADU. Once 'killed', it will not accept any '*' commands, or trap errors which occur. The only thing it will still do is respond to *HELP

*HELP<return> will display 'Killed ADFS Utilities' as a reminder

*HELP ADU<return> will resurrect ADU and display the normal HELP information

The killing is done by storing &FF in ADU's workspace byte in page &D. This mechanism is also used by DFS and ADFS. The location to use is &DF0 + ROM number. Placing &FF in this location will put ADU into the 'killed' state and placing zero in it, will restore it.

In most cases ADU will stay 'killed' over a BREAK or CRTL/BREAK though *FX200,2 BREAK, will definitely bring it back, as will the official method - *HELP ADU

## *MENU

Purpose: Provide easy management and use of ADFS discs
Memory used: Page 'C' and 'B' and main memory for some routines
Syntax: *MENU (<dirspec>)

Examples:
*MENU
*MENU $.b*.games

This command enters a menu routine which allows you to browse through the directory structure of an ADFS disc, starting from either the current directory, or the one specified in the command line. If the message 'ADFS workspace is corrupt' is displayed, you should CTRL/BRK the computer as soon as possible. The message indicates that the ADFS's workspace has been overwritten and is unusable in it's present form. If you cannot recover by using CTRL/BRK use *PWRBRK.

If an error occurs whilst in the MENU, it will be intercepted for house-keeping, and current disc will be dismounted. However, this routine will re-MOUNT the disc if asked to access it again.

Once the directory has been found, the data screen is displayed. This consists of 3 sections. The first gives the drive number and current directory name and title.

The next section contains the objects within this directory, or the message 'EMPTY' if it is an empty directory. Each name is suffixed by a letter giving the file 'type'

The section below gives details of the 'paged' ROMs in the system.

After the socket number are some characters giving information about the socket. Not all will be displayed for every socket.

| | |
|---|---|
| S | ROM has a service entry (it handles '*' commands) |
| L | ROM has a language entry |
| T | ROM has a relocation address for a co-processor |
| R | ROM is in write-enabled ROM |
| U | Socket not recognised ('Unplugged') by the O.S. |

After these characters is the name of the ROM in that socket.

Finally, one of the ROMs may have the message '<- Data (D) files' alongside it's title. Several word-processors save files which this routine recognises as data. All data files will be passed to the marked ROM. Those recognised are VIEW, EDIT, WORDWISE (PLUS) and Interword. If none of them are found, data files will be *DUMPed.

For each object name displayed, this routine will attempt to decide what type it is. The types, and the default actions when the object is selected are shown below

| | |
|---|---|
| D | Directory*DIR name |
| E | E protected file*RUN name |
| D | DataWordprocessor else *DUMP |
| B | Basic program*BASIC. CHAIN"name" |
| V | VIEW document*WORD> LOAD name |
| R | ROM imageLoader routine |
| M | Machine code*RUN name |
| S | VIEWSHEET spreadsheet*SHEET. LOAD sheet |

The first object name displayed is highlighted. To select this object press the COPY, SPACE or RETURN keys. Once selected, this routine will try to treat it appropriately actin upon the table above. This may be overridden, which is described in a moment.

The highlighted block may be moved around the objects displayed by using the arrow keys, or the traditional games keys 'z' 'x' '/' ':' and the routine may be left by pressing the ESCAPE key. You should not use BREAK to leave this routine because an orderly exit is essential to allow it to clear some of it's actions.

To override the standard action of a selected file, hold down the SHIFT key and then press COPY, SPACE or RETURN. You can now press a single key to tell the routine what to do with the file. To remind you of the options, they are displayed in turn at the bottom of the screen whilst the routine is waiting for a key to be pressed. Any unrecognised key presses will return you to the selection stage where the highlighted block may be moved around the screen. You need not wait until the option you want is being displayed, to press the key.

The options are:-

| | | |
|---|---|---|
| 0 | Command file | *EXEC name |
| 1 | Basic program | *BASIC. CHAIN "name" |
| 2 | VIEWSHEET spreadsheet | *SHEET. LOAD name |
| 3 | VIEW document | *WORD. LOAD name |
| 4 | Data | *DUMP name |
| 5 | Machine code | *RUN name |
| 6 | ROM image | Internal loader |
| 7 | EDIT text file | *EDIT f1 name |
| 8 | Basic (BASIC program) | *BE. LOAD name |
| 9 | BASIC program | *BASIC. LOAD "name" |
| G | Autoboot a !BOOT file | *DIR name. *EXEC name |
| V | Read a non-VIEW text file | *WORD. NEW. READ name |
| I | Load an INTERWORD file | *IWORD. 2 name |
| W | Load a WORDWISE file | *WORDW.. 2 name |

Certain commands will only work if the ROM required is present, for example option '8' will not work if the Basic Editor is not present or option 'W' will not work if WORDWISE(+) is not present.

The internal ROM loader routine will ask you which socket to load the file to. It will then first load the data at &4000 and then page in the desired bank and copy the code up. This approach should work with the majority of 'paged' RAM systems. Users of DFS 2.0j and earlier (who should consider an upgrade to 2.2 ) will be relieved to know that this routine (in common with other 'paged' RAM routines) does not use the *SRLOAD/*SRSAVE commands which are not in DFS 2.0j.

On the Model B the screen will be overwritten by the data, but the screen will be re-written on completion of the command.

Whenever selecting a file involves leaving the routine, the screen mode selected on entry, will be re-selected.

In addition their are quite a few other commands which can be used whilst in the object selection stage. They are listed below; first in alphabetical order and then explained by category.

| | |
|---|---|
| A | Select drive A (0) |
| B | Select drive B (1) |
| C | Compact disc until all free space is together |
| D | DISMOUNT and *BYE |
| E | Select drive E (4) |
| F | Select drive F (5) |
| I | Insert a ROM (cause it to be 'seen' by the O.S) |
| K | DISCEDIT the first sector of the current object |
| S | Save a ROM image into the current directory |
| U | Unplug a RON (cause it to be hidden from the O.S) |
| W | Delete the current object |
| X | Move right |
| Z | Move left |
| 0 | Select drive 0 (A) |
| 1 | Select drive 1 (B) |
| 4 | Select drive 4 (E) |
| 5 | Select drive 5 (F) |
| - | Change foreground colour ('ink') |
| = | Change background colour ('paper') |
| : | Move up |
| / | Move down |
| $ | Select the 'root' directory ($) on this disc |
| ? | Display information about this object |
| * | Enter a O.S command |
| @ | Display information about the current directory |
| TAB TAB | Re-set computer to state after being switched on. |
| Up arrow | Move up |
| Down arrow | Move down |
| Left arrow | Move left |
| right arrow | Move right |
| COPY | Select current object |
| SPACE | Select current object |
| RETURN | Select current object |
| ^ | Select the parent directory of the current directory |
| DELETE | Wipe a 'paged' RAM bank (fill with zeros) |

Commands related structure/use of disc

0,1,4,5,A,B,E,F
Mount a new disc and read in it's 'root' directory. Should also be used after changing discs in the drive being used to force DFS to read in the new directory from the disc instead of using the copy it is holding in the computer's memory

$
Return to root directory of current drive

?

Display addresses and attributes for this file (as *INFO). If the object is a directory, the contents of the directory will also be displayed.

@
Display information about all the objects in the current directory (as in *EX)

C
Compact disc. This routine will display the amount of free space on the disc and the current 'free-space map'. If the 'free-space map' consists of more than one entry, the disc will be compacted, using &4000 -&7FFF as workspace. On a Model B this will cause overwriting of the screen. At the end of the compaction, the summary of the free space will be up-dated and the disc compacted again until all free space is together

At the end of each compaction the routine will check to see whether the ESCAPE key has been pressed and will stop if it has. A message will be displayed to inform you that the routine has been aborted

D
Dismount current drive. This command tells the ADFS that the current disc is no longer available. I also issues a *BYE command which will clear the head on a Winchester disc from the disc surface in readiness for the drive to be switched off. Since there is no longer a directory to use, this command also terminates the routine.

W
Delete the currently selected object. To confirm your intention, a message will be displayed, asking you to press 'Y' to confirm that you want to delete the file. This command will cause the routine to stop if the deletion fails, typically by trying to take out a 'locked' object like a directory.

Commands relating to 'paged' ROMs
(Loading ROM images is performed by selecting the object if it has a type code 'R' or override option 6, otherwise):-

DELETE
Wipe a RAMbank. This routine attempts to fill the bank number entered with zeros. If it is successful, it also removes the bank from O.S's list of ROMS to prevent any calls being offered to a image that is no longer there. For safety the bank should be 'unplugged' first.

I
'Insert' a ROM image. This command informs the O.S that a ROM is present in the bank number given. If used on a Master series machine. a *INSERT command is also generated. It should be used to initialise a bank when first loaded or re-loaded.

Not all ROMs will function correctly if initialised without pressing the BREAK key (eg. Filing systems) and some will not work unless they have detected a 'power on' BREAK. The latter condition can be simulated by using TAB TAB which is explained below.

S
Save the selected bank's contents to disc in the current directory. You will be prompted for the bank number and then the filename to use. To achieve speed, and independence of the O.S, this routine first copies down to &4000 and then saves it from there.

This will cause the screen to be overwritten on a Model B, but it will be re-written when the command is completed

U
'Unplug' a ROM image. This command prevents a ROM from receiving calls from the O.S. It will not be sufficient to completely disable certain types of ROM which also intercept the main system vectors(eg. Filing Systems). If used on a Master series machine, a *UNPLUG command will also be generated.

Other commands

TAB TAB
Simulate a power-on reset. After priming the reset this routine will check if a 'Tube' is present. If not, it will reset the machine itself, and you will see the prompt which is displayed whenever your computer is switched on
BBC Computer 32k &c.&c.

On tube systems, a complete reset relies on a hardware signal from the BREAK key to reset the 'Tube' processor. When a 'Tube' is detected, a message 'press BREAK' will be displayed. When you press the BREAK key, the machine will reset as if it had just been switched 'on'.

*
After typing '*', you may type in a line of text which will be passed to the O.S. for interpretation. Before allowing the command to be executed, this routine will reselect the original screen mode (and do a few other pieces of tidying up) in case the command does not return (e.g. *BASIC). If the call does not return, the prompt 'press a key' will be displayed and the main screen will be re-written. If 'paged' mode is required, use CTRL/N after typing '*'.

ESCAPE
Leave the MENU routine. This is the official way to exit. Before allowing you out, the original screen mode will be reselected and several other pieces of tidying up done to allow a clean exit. Do NOT press BREAK to exit the MENU routine (unless using TAB TAB)

K
This command passes you to DISCEDIT, which then positions itself on the first sector of the currently selected object. When you press ESCAPE to leave DISCEDIT you will return to the MENU routine.

-
Change the foreground colour (the 'ink'). Flashing colours are not allowed, nor is the current background colour (which prevents you from obscuring the text).

=
Change the background colour (the 'paper'). Flashing colours are not allowed, nor is the current foreground colour (which prevents you from obscuring the text).

Any errors which occur within MENU are trapped, and used to DISMOUNT the disc. However, errors which occur while MENU is starting up, are not trapped.

Not a directory
The directory name you gave MENU is in fact a file, not a directory-thus it is impossible for MENU to use it as a directory.

ADFS workspace is corrupt
The memory used by ADFS for holding the current directory has been overwritten and is unusable. The only realistic solution is CTRL/BREAK as soon as possible (or if things get serious, *PWRBRK)

Various other errors can occur e.g. 'Locked' (using the 'W' command on a locked object),'DIR not empty' (trying to delete a directory which contains some objects) &c,&c. For a full list of the possible errors, consult the ADFS documentation. Once the screen display has been written, any error will be trapped.

## *PWRBRK

| | |
|---|---|
| Purpose: | Reset the computer as for a power-on reset |
| Memory used: | Nil |
| Syntax: | *PWRBRK |

This command provides access to the TAB TAB routine in *MENU from other environments and is often useful for removing any lingering effects after using games programs.

## *VERIFY

| | |
|---|---|
| Purpose: | Test an ADFS disc by reading every sector |
| Memory used: | Page 'B' and 16 pages from OSHWM |
| Syntax: | *VERIFY <drive> |

Example:
*VERIFY B

This routine will verify, or test, a disc by reading every sector on the disc, into the computer. It will work on any size of disc (including Winchesters) because it decides how many sectors to read from information stored in the 'free-space' map of the disc.

If *VERIFY is entered whilst not in ADFS this routine will ignore the command and allow other ROMs to respond (typically a DFS)

Whilst verification is in progress, the message 'reading &xx sector from &yyyyyy' will be displayed and up-dated. The first value xx, is the number of sectors being read in. This will usually be 16 (&10) but may alter if there is not an exact multiple of 16 sectors on the disc (typically on Winchesters). The second value is the first sector being read

The value of 16 (&10) is used because it is the number of sectors per track on an ADFS floppy/3.5 inch, thus the verification is a track at a time. This concept is meaningless with a Winchester which will typically have several hundred sectors on a track and ADFS does not need to know the layout of the platters in the disc pack. In cases like this there will typically be some strange number of sectors on the disc (eg. &13C9B for a 20Mb Winchester). Using &13C9b as an example, the last 2 reads will &10 sectors from &13C80 followed by &0C from &1C90.

If the sector read fails, the message 'Disc error' will be displayed. The value &yyyyyy may be used to note where it is on the disc (within the range of sectors which the routine was attempting to read).

Once verification has been successfully accomplished, a message will be displayed.

If the routine completes successfully, it will reselect the 'root' directory of the current drive. If an error occurs during the routine the current drive will be 'Unset' and should be selected with a *MOUNT *drive or *DIR command.

The routine can generate 2 errors

Parameter required
The drive specification was not given , use one of 0145ABEF

Disc error
An attempt to read a sector failed, indicating an unreliable disc.

## *VFORMAT

| | |
|---|---|
| Purpose: | FORMAT and VERIFY a disc |
| Memory used: | Page 'B' and main memory |
| Syntax: | as FORMAT |

This command allows you to FORMAT and the VERIFY a disc. It is simply *FORMAT followed by *VERIFY. For details refer to the respective command descriptions