PRES
PO Box 319, Lightwater,
SURREY GU18 5PW

# AFM
# Advanced File Manager
# User Guide

# CONTENTS

# INTRODUCTION

The PRES Advanced File Manager is designed as a 'Front End' for filing systems - a program which cushions the user from the intricacies of the Disc or Advanced Disc Filing Systems (DFS and ADFS).

It addresses four distinct area :

*Menu* - allows the user to browse through a disc with the facility to load and use a file in an appropriate manner.

*Delete* - allows one or more files in a directory to be marked and then deleted in one go.

*Copy* - allows files to be copied within or between DFS and ADFS. This is the largest component of **AFM**, and is extremely sophisticated, while still retaining a straightforward user interface. It is capable of generating multiple copies on different discs, and of splitting and recombining files larger than the disc being used.

**AFM** works in the Electron, and all models of the BBC Computer. It uses only documented OS calls for disc access, so should work with non-Acorn Filing Systems. In particular it will accept up to 62 filenames from a DFS disc, providing compatibility with dual-catalogue DFSes which access both catalogues together. Care has been taken to write **AFM** legally avoiding any features which may vary between vendors' Filing Systems. **AFM** assumes that the calls OSARGS, OSFIND, OSFILE, OSGBPB and OSWORD &70 (ADFS) behave as documented by Acorn.

In the Menu, Delete and Rename modes the user's memory from PAGE (or more properly OSHWM - the Operating System High Water Mark) to HIMEM is unaltered; other than the possible effects of a change of screen mode.

In the Copy mode it can adopt all available memory for file buffering. This includes RAM cartridges (including the PRES ABR or AP7 providing 2 x 16K in two banks, and the PRES AQR providing 16 x 16K in one bank), sideways RAM, Tube memory, unused Shadow screen memory and the 12K bank of paged RAM in the B+, and free memory on the Slogger Master RAM board for the Electron. Fuller details of the memory available and how it is claimed are in the Copy Mode section.

**AFM** may be configured as the Language ROM with which your computer starts when it is switched on in most machines. However, the procedure varies with each machine, and is detailed below:

*Electron (without PRES AP2 ROM):*

The Plus 1 ROM will look for a default language in the sequence 13, 12, 7 to 0, then 11 and 10. The effect of this is that a ROM in any cartridge bank (or on a PRES AP6 or AP7) will override BASIC.

*Electron (with PRES AP2 ROM):*

**AFM** may be placed in any bank. When the computer is switched on you will start in BASIC. Use *LANG to set the bank number of **AFM**. After that a CTRL-BREAK will select **AFM**.

*BBC Model B:*

**AFM** must be in a bank with a higher number than any other Language. In an unexpanded machine this means it should ideally go in the far right hand ROM socket (as viewed from the front).

*BBC Model B+:*

Again **AFM** must be higher than any other language. However BASIC appears in bank 14 initially. To demote BASIC move link S13 to its North position (towards the back of the computer). This link exchanges banks 14 and 15 with 0 and 1. Note: If you have a B + 128K the Ram banks in 0 and 1 will now appear in 14 and 15 - a good reason for using *SRLOAD with W to Z to specify the bank numbers!

*BBC Master 128 & Master Compact:*

**AFM** may go in any bank. Use *CONFIGURE LANG. to set the bank number in the non-volatile memory.

You can check that **AFM** is present by typing *HELP. If **AFM** is active, a message similar to:

```
Advanced File Manager 1.00
  *AFM
```

will be printed. This tells you that you have version 1.00 and that *AFM is the command to start up **AFM**.

**AFM** can be disabled by putting 255 in (Bank number + &DF0). Once disabled it will not appear on *HELP, or respond to *AFM. It can be restored by putting 0 in the same location. NB. Tube users must note that this location is in the Host processor, so is not immediately accessible using ? from BASIC in the Tube.

Within **AFM** certain keys have special uses:

| | |
|---|---|
| Arrow keys | - Move around the current window |
| Z, X, :, / | - Move around the current window |
| SPACE | - Leave this window |
| RETURN | - Make a file or alter a value |
| COPY | - Act upon marked files |

Other keystrokes are available at particular times, and are explained in the relevant sections.

# USING AFM

To enter **AFM** type `*AFM` and press RETURN. The screen will clear and the computer will draw the outlines of the five windows used by **AFM**.

On Master series machines, and models B or B+ with the Acorn GXR fitted, a flood fill will be executed to fill the outline of the screen. If you do not want to wait for the fill to finish you can press ESCAPE whilst the fill is in progress and it will be abandoned at that point. Be careful not to hold down the ESCAPE key for too long or it will be read twice, which will leave **AFM**.

**AFM** stores information about the screen mode which it usurped. It save the mode number, the palette (ie the VDU 19 settings), and whether it was a Shadow screen mode (HIMEM was &8000). This information is used to restore the screen and colours being used when you leave **AFM**.

To leave **AFM** press ESCAPE. In some cases more than one press will be needed. Generally ESCAPE works back towards the Menu screen and the exits from there.

If the message 'DFS/ADFS only' is displayed, then neither DFS or ADFS were active. Select either then type `*AFM` again.

Throughout these instructions the five windows will be referred to as windows 1 to 5, numbering from the top down.

In most cases **AFM** will initialise without the disc being accessed, however if nothing is being displayed, or a striped screen appears on an Electron, you should check that a disc is in the current drive.

Once the information has been read from the filing system **AFM** will set up the windows with their initial contents.

*Window 1* holds your position on the disc.

It displays 'Disc FS' or 'Advanced Disc FS' as appropriate, and the drive number and directory name of the current directory.

*Window 2* lists the objects (ie files and directories).

This window holds up to 47 (ADFS) or 62 (DFS) file/directory names read from the current directory. The entry in the top left corner will have a flashing cursor alongside it. Otherwise 'Empty Directory' will be displayed if no files or directories were found.

The flashing character varies according to the object it is next to. It will be a '!' next to 'empty directory', '>' next to a file and '*' next to a directory. Whilst the character is flashing you are working in this window and can move the flashing cursor around within the range of objects displayed.

*Window 3* lists the operating modes of **AFM**.

Initially 'Menu' will be highlighted, indicating that this is the current mode. The other modes; 'Delete', 'Rename' and 'Copy' are chosen by pressing SPACE until window 3 is selected, and then moving the highlighted block to the desired mode and pressing SPACE again.

*Window 4* lists the sideways ROM and RAM image.

This is a list of all currently active images. Initially the Language ROM from which you entered **AFM** will be highlighted. If **AFM** was entered as a language then BASIC will be highlighted. When moving the highlighted block about you will notice that it only stops on certain images; those which are 'Languages', ie distinct applications rather than transient utilities. This is used in conjunction with the Menu mode for executing files. Images which are 'Languages' have their bank number highlighted.

There is also a facility for leaving **AFM** and going to the image selected in this window.

*Window 5* shows the selected Rename string.

In Menu and Delete modes it will show '<Rename OFF>'. With Rename mode selected it holds the string used for the ambiguous renaming.

How to see which window is selected.

There are two things to look for. Firstly a flashing object, either the normal cursor, or '!', '>' or '*' flashing in window 2 or 4, or a field which is not usually highlighted. If you are in any doubt moving around the window with the arrow keys or ZX:/ will soon indicate which window is active.

## Menu Mode

The menu facility in **AFM** has been designed to be expandable, unlike other menu programs. This is achieved by using the currently selected SWR to dictate what to do to a file. Up to seven separate options are catered for.

This approach allows other products to add their own specific loading protocols to **AFM** transparently to the user. This is achieved by using a call allocated to PRES by Acorn for this purpose. We encourage users to make use of this facility to expand the menu capabilities of **AFM** - full details of the call, and how it should be responded to are given later. This means that **AFM** is not restricted by default settings, instead it can be developed to cater for new ROMs not even available yet, or to cover those not included in **AFM**'s own loading protocols.

In addition the in-built loading protocols may be replaced by this method. **AFM** will first attempt to find an external loader before reverting to its own loading protocols. This allows the user to replace **AFM**'s options with different ones customised for specific applications.

After using *AFM, the windows will contain the information mentioned previously, and window 2 will be selected. This is indicated by the flashing cursor.

## Starting a file

To ensure that you can re-create the next part of the instructions yourself you should arrange to have a BASIC program amongst the list of files displayed, and BASIC as the current SWR. You can either achieve this by using commands within **AFM**, or by exiting (press the ESCAPE key) and re-entering.

To start a file, move the flashing cursor until it is next to the file you want to use. Press RETURN. Window 4 will clear and the list of loading options exclusive to BASIC will appear. this consists of:

```
CHAIN <Highlighted>
LOAD
*TYPE
*DUMP
*LIST
*RUN
*EXEC
```

The highlighted block may be moved using : and / or the up and down arrow keys. Move it back to 'CHAIN', if it is not still there, and press RETURN. The screen mode you were in before entering **AFM** will be reselected, then BASIC will be entered and the CH."<filename>" command issued to load the program.

The * commands will only work if you have a program in your machine that will accept them. They are usually within the DFS, though can come from other sources, eg the PRES AP2 ROM in the Electron, or the Utils part of the Compact's Operating System.

The options displayed will vary with the SWR selected. **AFM** itself supports BASIC, VIEW, VIEWSHEET, VIEWSPELL, VIEWSTORE, WORDWISE (Plus), INTERWORD, THE BASIC EDITOR and EDIT. Other SWRs can be catered for using the expansion facilities already mentioned.

In each case, up to seven options will be displayed, and the desired one is chosen by moving the highlighted block up or down and pressing RETURN. Note that you are allowed to move upwards from the first item around to the last, and vice versa.

If you decide that you do not want to start a file, perhaps because the wrong SWR is selected you can press ESCAPE and revert to normal menu operation.

If the computer beeps and no options are displayed then **AFM** could not find a loader for the current SWR,

## Other Menu mode facilities

*In window 2*

Whilst within window 2 a number of other commands are possible.

RETURN whilst on an ADFS directory name; will cause that directory to be entered. ADFS and DFS users can also alter the parameters in window 1 directory, which is mentioned shortly.

$ will select the directory $.

^ (ADFS only) will move back to the parent of the current directory. This can be repeated until $ is reached.

0-5 will change drive and select directory $. The computer will beep if an illegal number for the current filing system is used. Illegal numbers are 4 & 5 in DFS, and 2 & 3 in ADFS.

- will alter the text colour

= will alter the background colour

You are prevented from having the same colour as text and background at the same time.

If **AFM** is being used as an image in sideways RAM the new settings will be retained and used on subsequent initialisation with `*AFM` instead of the default blue and white combination.

These extra commands are in addition to the standard use of the arrow keys and ZX:/ in this window. The COPY key has no special function, so behaves like the RETURN key in this window. The <SPACE BAR> will move into window 3 - the cursor in window 2 will stop flashing, and a cursor will appear alongside the highlighted word 'Menu' in window 3.

*In Window 3*
This window is used to select another mode instead of 'Menu'. Move the highlighted block along to the mode you want to select, and then press SPACE to leave the window.

If you have altered the mode setting to Menu, Delete or Rename then any files marked in the delete or rename modes will be unmarked, and the cursor will return to window 2 immediately.

If the mode is unchanged (even though the highlighted block may have been moved, but put back to where it started) then the cursor will move to window 4.

When copy mode is selected windows 3 and 4 take on different functions, which are explained in the copy mode section.

*In Window 4*
This window is used to select a SWR. Some banks may not contain an image or a Language ROM. Consequently not all SWRs will have something alongside them, and there will be some (eg DFS or ADFS) which cannot be highlighted because they are not a language. The bank number of those that may be selected is highlighted.

The arrow keys (or ZX:/) are used to move around. You will notice the highlighted block jumps sometimes, which is from one language to the next. When using the right or down arrows (or X and /) it will move downwards, otherwise it will move upwards.

The SWR selected will be cited by Menu mode as the file destination when RETURN or COPY is pressed in window 2 on a filename.

When RETURN or COPY is pressed in this window the currently selected SWR will be started up, exiting **AFM**. The original screen mode and palette will be restored. Note: **AFM** does not, and cannot, know the appropriate * command for every SWR, instead it uses an OSBYTE (or *FX) call. In most cases there will be no appreciable difference in the effect.

When SPACE is pressed to leave this window you will skip window 5, unless you are in rename mode, and go straight to window 1.

*In Window 5*
This window is only accessible in rename mode. Instead of selecting this window you will go straight on to window 1.

*In Window 1*
When Window 1 is selected the 'Filing System' will he highlighted. Pressing RETURN will toggle this field between DFS and ADFS.

By using the arrow keys or ZX:/ you can reach the other fields.

The 'Drive' field will cycle through the allowable drive numbers for the current filing system, which are 0-3 with DFS, and 0,1,4,5 on ADFS.

The 'Directory' field works slightly differently. When you press RETURN the current directory will disappear, and a $ will be displayed. This can be deleted if you do not want it. Type in the desired directory or directory/pathname and press RETURN.

In DFS you input will be rejected if it exceeds one character.

If you decide you have made a mistake pressing the ESCAPE key once will restore the parameters as they were set upon entry to the window. Do not hold down the ESCAPE key too long though since the second press will exit **AFM** as usually occurs when ESCAPE is pressed.

When you press SPACE to move onto window 2 the directory name will be checked. If you are in ADFS the directory name will be tested to ensure that it exists on the disc. If it does not the computer will bleep and you will remain in this window.

Once **AFM** is satisfied with the parameters it will move into window 2. If any parameters were altered it will read in the new object names, otherwise it will go directly into window 2.

## Delete Mode

The delete mode allows one or more files in the same directory to be marked, and then all deleted with a single keystroke. WARNING - Files which are Locked will also be deleted if they have been marked.

Delete mode is selected from menu or rename modes by moving into window 3, then shifting the highlighted block onto 'delete' and pressing SPACE. The list of objects in window 2 will be unmarked (if applicable) and then window 2 will be selected.

The majority of operation in delete mode is the same as in menu mode, so only additional or differing information is presented here.

*In Window 2*

Files are marked by pressing RETURN or DELETE. The name will be re-displayed in inverse text. A file may be unmarked by pressing RETURN or DELETE on it again. However, RETURN and DELETE behave differently on directories in ADFS. RETURN will attempt to enter the directory - which will be blocked if any other files or directories have been marked. The DELETE key marks a directory. The directory must already be empty if the deletion is to be successful. If you inadvertently attempt to delete a non-empty directory the error 'Dir Not Empty' will be given by ADFS and **AFM** will return to menu mode.

When the COPY key is pressed, all the marked files will be deleted. As each deletion is accomplished the filename will be removed from the screen. After the last one the screen display will be rewritten to close up any gaps that have been created.

While one or more files are marked you are not allowed to leave the current directory. The computer will bleep when you have file(s) marked and press $, ^, 0-5 or RETURN on a directory name. However you can still change the directory entirely using window 1.

Pressing ESCAPE in delete mode (other than in window 1 after altering parameters) will exit **AFM**.

There is no need to clear marked files if you decide to change to another mode, they will be cleared automatically as the new mode starts up.

A short cut exists for marking all the files in the directory - press TAB. All unmarked files will be marked, and marked files will remain marked.

Electron users must use CTRL-I for TAB. Hold down CTRL, press and release I, then release CTRL. Practice using CTRL-G in BASIC, which will produce a beep.

*In Window 3*

No differences to menu mode.

*In Window 4*

No differences to menu mode.

*In Window 5*
No differences to menu modes; i.e. this window is not accessible in delete mode either.

*In Window 1*
No differences to menu mode. Note though that this is the only way to change directory while some files are still marked. A change of directory in this window will automatically unmark them.

## Rename Mode

The rename mode of **AFM** allows one or more files to be renamed, with the renaming controlled by a special text string, referred to as the 'rename string'.

The approach used is very versatile and allows many varied operations to be accomplished easily.

The 'rename string' has a maximum length which is the same as a filename's maximum length in the current filing system. This is 7 in DFS and 10 in ADFS.

Within those 7 or 10 characters you can put three distinct things:

*ASCII Literals*
Characters which will be put in the new name, which may not be any reserved characters (eg not $ or . among others).

*
Which represents the whole of the original name

*CTRL 1 - CTRL 0*
Which represent the 1st, 2nd, 3rd ... characters of the original name. CTRL 8,9 are 0 are not allowed in DFS.

Master Compact owners must use the '0' key on the numeric keypad for CTRL 0. This is an effect of the introduction of the CODE key above '0' on the normal keypad.

Some examples may help at this point.

Assume that we are renaming three files, Game1, Game2 and BasicGame. The table below shows the effect of different rename strings.

A number in brackets, e.g. <3>, means CTRL 3. If Cannot! or Exists! is shown, an illegal name was produced upon expansion of the string, either because it was too long or too short, or because a file of that name exists in the directory already. **AFM** will work through the marked names from top left to bottom right, so when a string has been set up which will cause the same name several times upon expansion the first will succeed, and the rest will fail.

If CTRL n specifies a character higher than appears in the name being processed no error will result, and nothing will be used for that character.

The example assumes ADFS, with 10 character filenames.

| Rename String used | BasicGame | Game1 | Game2 |
|---|---|---|---|
| * | BasicGame | Game1 | Game2 |
| *Old | Cannot! | Game1Old | Game2Old |
| <1>,<2>,<3>,<4>,<5> | Basic | Game1 | Game2 |
| <0>,<9>,<8>,<7>,<6>,<5> | emaGc | 1 | 2 |
| <1>,<2><5>,Orig | BacOrig | Ga1Orig | Ga2Orig |

Probably the best way to get a feel for the way rename works is to experiment: copy a disc with a number of files onto a scratch disc, and use that to try out different combinations.

Once again the usage of the windows is very similar to delete and menu. In common with delete mode you are prevented from leaving a directory while files are marked with the $, ^, 0-5 or RETURN keys.

*In Window 1*
No difference to menu mode

*In Window 2*
As with delete mode, except that this time the COPY key begins renaming. After COPY has been pressed each rename will be attempted in turn. The highlighted name will be replaced with either the new name, or the Cannot! or Exists! error messages. Once renaming has been completed the screen will be rewritten to reorder the files.

Again, the TAB key may be used to mark all unmarked files in the directory.

*In Window 3*
No difference to menu mode

*In Window 4*
No differences to menu mode, however when you press SPACE you will go into window 5, and from there to window 1.

*In Window 5*
When Rename mode is selected you will see the default rename string, `*Old` appear here.

When you move into this window the 'Rename string' is highlighted.

If you want to change it press RETURN and type in the new string. The number of dots printed when RETURN is pressed show how many characters are allowed. If you start entering a string and decide to keep the old one press ESCAPE. When you have finished

entering the string, press RETURN. Providing that the string is satisfactory, the cursor will move directly to window 2 for marking/unmarking of files.

The * will appear in inverse, as will CTRL-1 to CTRL-0. This indicates that they will not literally appear in the resultant filename. Other characters will appear in normal text, showing that they will appear in the new name as entered. If an illegal character is typed, the computer will bleep and ignore it.

Pressing SPACE to leave the window will take you to window 1. Note though that entering a new Rename String will put you directly back to window 2.

# COPY MODE

**AFM**'s copy mode can copy any number of files between or within DFS and ADFS. Provision is made for splitting and recombining files larger than a disc (eg archiving a large database from Winchester onto floppy discs). It can also handle multiple source and target discs allowing, for example, one file to be copied to several discs.

All errors which occur during copy mode are trapped, and in most cases are recoverable.

It can cope with single disc drive copying (even under ADFS) and has an option allowing the user to turn on and off the prompting for source and target discs in a single drive.

To select copy mode move to window 3, position the highlighted block on top of copy, and press SPACE.

When SPACE is pressed, a number of things will change, as **AFM** initialises itself for copying files.

The information in window 1 will be examined, and the directory will be forced back to $, if the user is not already there, or has entered a path from $. If window 1 is changed then window 2 will be cleared, to indicate that the information which was there is longer valid and will shortly be refreshed.

Window 3 will show 'Advanced File Manager Copy Mode'.

Window 4 will show the Filing System, Drive and Directory for the target (i.e. destination) directory. It is changed in the same manner as window 1 in the other modes. There is an additional field 'Additional memory usage' which controls how **AFM** will treat the memory in the computer. The options are 'Automatic select', the default, in which all available memory will be claimed; 'No S/Ram or Tube', where the memory from OSHWM to HIMEM will be used and free screen memory (and also the 12K ram segment in the B+) and 'Selected by user' in which **AFM** indicates which areas of memory it can use and asks the user whether it can use them.

Window 5 is cleared.

At this point some information about the use of memory under copy mode will be useful:

## Memory usage

There are three distinct areas of memory used by **AFM**.

*a) Memory always claimed*
On all machines the memory between OSHWM (PAGE in BASIC) and HIMEM will be taken. On machines which have HIMEM at &8000 six pages (&600 bytes) will be used for workspace, otherwise only three pages (&300 bytes) will be used.

If you are using a Slogger Master RAM board, or any type of sideways memory, a 4K block (16 page &1000 bytes) will be allocated for data buffering. On machines with HIMEM at &8000 it will be taken immediately, and on other machines where memory

may be tighter it will be taken when the screen changes mode just before actual copying starts.

On a B+ the 12K segment of paged ram in bank 128 will be used.

On a 64K or Turbo Electron, 11.75K of ram will be used.

*b) Optional memory*
These are the types of memory which **AFM** can use, but will seek confirmation for with 'selected by user' set, or not use at all if 'no s/ram or tube' is set.

Any 8K or 16K RAM banks, including those in PRES ASR, ABR and AP7.

Any PRES AQR cartridges (256K)

61K of Tube memory

However it will not want to use RAM banks currently holding a ROM image, or AQR cartridges working as a ram-disc under the PRES Electron ADFS.

*c) Memory claimed when the screen mode changes*
When mode 7 (or 6 on the Electron) is selected, any memory released between the old and new values of HIMEM will be claimed. Of the memory released 4K (16 pages) will be taken for other uses if needed (see notes in section a). In addition some extra memory is available on certain machines:

On a B+ 19K of free shadow RAM will be used.

On a Master/Compact 19K of free shadow memory will be used.

On a 64K Electron 12K of free shadow memory will be used.

## Memory management

Apart from the special ages mentioned earlier all the memory claimed by **AFM** is used as a single area of scratch pad memory, regardless of its size.

This is used for holding details about the directories which have been entered, and files to be copied initially. Once copying has started the remainder plus memory gained by the screen mode change, will be used for buffering the file transfer.

Each directory visited will take <length of name>+2 bytes on its first visit. Each file marked will take 41 bytes, regardless of the file's own size.

When using copy mode the user need not worry about where actual memory is located. **AFM**'s memory access routines serve to connect all the various segments of memory it has found into one continuous expanse of memory.

There are two events which will cause **AFM** to immediately start copying the files which have been marked. The first is when the available memory falls below 256 bytes, and the

second is when the 65th or 257th directory storage is attempted. In practice neither of these are likely to prove serious. On systems with HIMEM at &8000 256 directories are maintained, and 64 on other systems.

When either of these errors occur **AFM** will give you the option of pressing ESCAPE (to return to menu mode), and if any files have been marked it will also let you press COPY to begin copying the files marked thus far. Both these errors are serious enough to prevent further effective browsing, but are trapped before they could jeopardise any file copying which had been set up.

## Using copy mode

When the files in window 4 are set to your satisfaction, press SPACE. **AFM** will check the directory specification, and the computer will bleep and remain in the same display if the directory was not found and must be re-entered.

If the memory usage was set at 'Selected by user' **AFM** will check the system for additional memory it can use and ask before taking each bank. The questions appear in window 4, and will be in the form:

```
Use 15K (or 8K) or RAM in <n> (Y/N) ?
Use 156K of AQR in <n> (Y/N) ?
Use 61K of tube memory (Y/N) ?
```

The only expected use of this facility is when a ram disc is being used, which is not running PRES ADFS 1.1 or 1.2, and **AFM** must be told to ignore it. **AFM** will avoid RAM banks with a working image in them.

The copying routines in **AFM** work slightly slower when using Sideways RAM. If you want a faster copy, rather than a large buffer memory, use this option to discard the RAM banks.

In normal use the memory setting is best left on 'Automatic select'.

Once the memory map of the system has been established and stored, **AFM** will write the target directory name in window 3.

It will then read in the directory on the target disc. If necessary a prompt for the target disc to be inserted will be issued. Put the disc in the drive and press RETURN. If you are using the same disc just press RETURN.

**AFM** initially decides automatically whether to issue prompts for discs to be changed. This can be changed using the 'C' command available once the directories have been read.

After the contents of the target directory have been read and displayed in window 4 it will read the source directory. This may require another prompt for a disc change. As before change disc (if appropriate) and press RETURN.

**AFM** has now reached a 'usable' state. To summarise; the current window usage is :

Window 1 - Source directory

Window 2 - Objects in Source directory

Window 3 - Target directory

Window 4 - Objects in Target directory

Window 5 - Status, and other messages.

Window 5 contains five pieces of useful information :

*The status of the Change Disc prompts*. This is initially set to **AFM**'s decision based upon the information entered for the Target directory. Press the C key to alter it.

*The status of Directory Following*. When ON the Target directory will attempt to follow the Source directory. This is explained in detail shortly. It may be altered by pressing F.

*The status of automatic Overwriting.* When ON and a file already exists with the name of the file being copied it will be overwritten without asking for the confirmation. The default state is OFF - so an attempt to overwrite a file will require confirmation by the user. Press the O (the letter O, not the number zero) to change it.

*The number of bytes free*, which will be updated as memory is used for new directories and files to be copied.

*The number of files* which are currently marked for copying. This will be 0 initially.

## Change Disc Formats

**AFM** will conclude that prompts are needed if:

The same drive is being used in the same filing system. This includes different sides of the same disc in DFS, but will exclude Winchester drives under ADFS and AQR Ram discs under PRES ADFS 1.1 or 1.2 (i.e. immovable discs).

The same drive is being used across two filing systems, however a Winchester drive will be checked and allowed for (providing copy mode was entered in ADFS).

If the default choice is wrong it can be altered by pressing C once both directories have been read.

## Directory Following

The concept of Directory Following is intended for ADFS discs, but works to some degree with DFS too.

The idea is that the Target directory will automatically go where the Source does, entering and creating directories as needed.

This means that a Target disc can begin blank, and will assume the hierarchy of the Source disc as the user moves through directories in the Source disc to reach the files to be copied.

Directory following can be used between ADFS and DFS, but a number of restrictions apply because of the limited structure of DFS discs.

These are the commands which, when executed with the cursor in the source window, cause an effect in the target directory:

$ Select $ on both directories. NB when copying between ADFS and DFS you should use ^ or @ in preference to $. This is because ^ and @ will maintain the relative position within the two directories.

0-5 Select new source disc, and $ in target.

^ Recurse a directory in ADFS, or select $ in DFS.

@ No effect if ADFS is the source filing system. Between DFS and DFS the target will go where the source does. Between DFS and ADFS entering a directory other than $ will cause a directory to be entered on ADFS (and created if necessary), and selecting $ will take the target directory back one level - ie to the parent of the directories. If that sounds complicated don't worry! It makes much more sense in action!

RETURN No effect if DFS is the source filing system (since directories are not indicated in the file window). In ADFS the name will be truncated to one character and used to select a DFS directory in the target. Use ^ to get back to $ on DFS without losing your place in the ADFS disc.
Between ADFS and ADFS the target will go where the source does, creating directories as necessary.

If an illegal operation would be caused - for example by entering a directory in the source window, which is a file in the target window - then the computer will bleep and directory following will be turned off automatically.

## Automatic Overwriting

When **AFM** discovers that a file it is copying already exists in the target directory it can behave in one of two ways depending on the setting of <O>ver.

If the file was marked for copying whilst <O>ver was OFF then **AFM** will issue a 'File Exists' message, giving you the opportunity to go ahead and overwrite it or leave it as it is.

If the file was marked whilst <O>ver was ON the message will not be issued and the file will be overwritten without any further intervention from the user. Note: although the effect is to overwrite the file, it is in fact deleted completely to overcome problems like a clash of attributes, or "Can't Extend" errors in DFS, and then rewritten.

## Browsing in copy mode

Once the directory contents have been read for the first time, a flashing cursor will appear alongside the first entry in window 2. From now, until the COPY key is pressed or a memory-full error occurs, you are browsing in the source and target directories.

The cursor will be in window 2 initially. The arrow keys and ZX:/ move around the list of files. RETURN marks a file for copying - a confirmatory message will appear in window 5, and COPY enters file copying. Since entering the file copying is irreversible you are asked to press RETURN afterwards. Any other key will retreat back into a browsing state again.

You can unmark a file by pressing DELETE on the highlighted name. All copy commands for that file will be cancelled.

The TAB key can be used to mark a number of files. If it is pressed while the cursor is against a file then all files in the present directory will be marked for copying. If it is pressed with the cursor against a directory (ADFS) the directory will be entered and then all files in that directory will be marked.

You can also use:

| | |
|---|---|
| SPACE | - to swap between windows 2 and 4 |
| @ | - to enter a new directory for whichever window you are in |
| 0-5 | - to select a new disc for the window you are in |
| - | - to change the foreground colour |
| = | - to change the background colour |
| $ | - to return to $ |
| ^ | - to go back a directory (ADFS) or select $ (DFS) |

If you want to alter the Target directory, select window 4 by pressing SPACE, so that the flashing cursor appears in window 4. You can then use all the above commands except RETURN. To return to window 2 press SPACE again.

## Starting File copying

To start copying files press COPY and then RETURN. If COPY simply makes the computer bleep, and nothing else happen, you have not got any files marked for copying (you can see how many files are currently marked on the status line in window 5). The screen will change to Mode 7 (Mode 6 on the Electron). **AFM** will then claim whatever memory has been freed by the mode change, and display the revised Bytes Free figure.

If disc change prompts are selected you will be prompted for the source and target discs as needed. If you have used several different source or target discs, **AFM** will ask you to check the drive each time it wants something from a disc which is not currently in the drive.

Consequently, **AFM** will simply read in as many files as possible, until memory is filled or the list of files to copy is exhausted, and then write them to the appropriate place. This will continue until all files have been copied or aborted.

## Errors

Sometimes things do not go strictly to plan; so **AFM** contains comprehensive error trapping to get around recoverable errors. Errors are detected in one of three phases; a) reading the file, b) checking whether the file already exists (providing that the <O>ver option was OFF when the file was marked), and c) writing the file. The actions offered for recovery vary slightly in each case.

In all cases the actual error reported will be displayed, followed by the options available for recovery.

*A Read error*

In theory there should be no read errors, since all input was validated whilst browsing. However there is still a possibility of disc errors, or the wrong disc being in the drive.

Three options are available for an error whilst attempting to read file :

A Aborts the file. **AFM** will forget about copying this file (abandoning any parts copied on previous passes), and go straight onto the next file to be copied.

S Stops copying. This aborts the current file, and terminates copying. The 'Copying finished' prompt will be displayed.

T Tries again. Allows you to try again, either to keep on attempting to read a disc with a faulty track, or to retry with the correct disc in the drive.

*A File Exists error*

Although technically a write error this has its own menu.

This error can be suppressed by setting the <O>ver option to ON before marking files for copying - which is especially useful if you are renewing an archive, for example, which involves overwriting several files with new files of the same name.

A Aborts this file. **AFM** will forget about copying this file, and go straight onto the next one.

D Delete file. This deletes the file - useful for disposing of files which cannot immediately be overwritten, eg those without an R attribute (ADFS), or Locked, or with the E attribute (ADFS).

E Extend the file. The data held in memory will be appended to the existing file. It is intended for reconstituting files which have been split across several discs (see also C in the write error section).

O Overwrite the file. The data held in memory will be written on top of the current contents of the file.

S Stop copying. This aborts the current file, and terminates copying.

T Try again. This allows you to retry, for example when the source disc was left in the drive by accident during a single drive transfer with disc change prompts.

*A Write error*

The most likely errors here are those concerned with the disc being unable to hold the data being put on it, eg Disc Full, Cat Full, etc etc.

A Aborts the file. **AFM** will not delete any part of the file copied so far - in case you are aborting it because you have realised that you want to keep the file which is already there. It will abandon the attempt to copy the file and proceed to the next file.

C Continue on another disc/directory. **AFM** will close the file on this disc, then ask for a new disc or directory, and continue from the point at which the error occurred using the new disc or directory. This is used to split large files across several discs.

D Delete file. The target file will be deleted. This is intended for use when you have answered 'O' to a 'file exists' error, which has then failed because of an attribute mismatch making it impossible to overwrite. Once the file has been deleted the copying will proceed.

R Restart on another disc/directory. **AFM** will delete the copied portion of the file, then ask for a new disc or directory. It will then start the file afresh. In some cases the start of the file may not be in memory at the time, in which case an explanatory message will be given and **AFM** will proceed with the next file to be written. The start of the file will be read in on the next read cycle.

S Stop copying. This aborts the current file, and terminates copying.

T Try again. This allows you to retry operations.

When C or R are selected a secondary prompt will be issued asking you to choose between:

N New disc, same path. **AFM** will expect a fresh disc, but will use the same directory.

P New Path. **AFM** will ask for a new path, which may be on the same disc or a different one.

In either case the path will be tested on ADFS. If it does not exists a message will be displayed asking whether you want to create the directory or enter a new one.

The creation facility allows you to maintain a file's hierarchical position on a fresh disc. It can create several directories in one go, so can cope with a specification involving several directories, none of which exist.

Once a satisfactory path has been established copying will continue, from the point at which the error occurred if option C was used, or from the start of the file if option R was chosen.

The new settings will be remembered for any further parts of the file, and will also stay in effect for any subsequent copies to the same directory, until a different target directory is required.

## Splitting and recombining files

**AFM** provides the facilities to split large files across several discs and recombine them.

To split a file set up a copy between the file and one of the discs which are to hold the file. If a 'File Exists' error is given answer 'O' to overwrite the file. Alternately the 'File Exists' error could be suppressed by setting <O>ver to OFF.

When a 'Disc Full' error occurs select option 'C', then insert a fresh disc and select option 'N'. If an error is given because the directory was not found press RETURN to create it. Copying will then proceed. Continue like this until the file has been copied.

To recombine the file set up a copy from each portion back to the original. This is achieved by changing the source drive for as many discs as are needed using '0'-'5'.

Start copying. **AFM** will prompt you to 'Check that the correct Source disc is in the drive'. This is **AFM**'s warning that it thinks the disc needs to be changed. Put back the first disc and press RETURN.

If a 'File Exists' error is given for the first portion answer 'O' to overwrite the file. Change source discs each time **AFM** asks you to check the drive, and answer 'E' to all subsequent 'File Exists' errors to extend the file.

On completion of this process the original file will have been recombined to its full size.

Note: When splitting files in ADFS the intermediate files will have attributes of WR, regardless of the file's attributes.

## Finishing Copying

When all files have been attempted the message 'Finished' will be displayed. If Tube memory was actually used you will then be asked to press BREAK to reset the Tube.

If Tube memory was not used (either because it was not present, not selected, or simply not loaded with data) you will be asked to press a key. You can either press ESCAPE and exit **AFM**, or press another key and return to menu mode, on the Target directory.

# EXPANDING THE MENU

This section is, of necessity, technical and must assume a familiarity with OSWORD calls.

The core of the menu expansion system is OSWORD call &B5, which has been allocated to PRES by Acorn for this purpose.

When RETURN is pressed on a file in menu mode, **AFM** will issue this call to see if someone else can provide loading options. If no-one responds it will try its own loading routines, after which it will give up and make the computer bleep to indicate that no loader is present. Note the sequence of events - it is intentionally arranged so that if the OSWORD call is responded to it can replace one of **AFM**'s own loaders.

If the call was responded to the list of options will be displayed in window 4 and **AFM** will wait for the user to choose an option.

When an option has been chosen **AFM** will re-issue the call to tell the code which responded before that it should now start the file using the option number given.

There may be up to seven options. Although there is no actual length restriction on each option you should aim for a maximum of about 20 characters. The option text is intended to describe the loader, so can include full syntax, eg LOAD <filename>. The actual content of each option is unregulated. However the total length of the parameter block cannot exceed 256 bytes.

First Call - AFM trying to find a loader

| Byte | Contents |
|---|---|
| 00 | FF - do not alter |
| 01 | FF - do not alter |
| 02 | 00 |
| 03 - n | Title string of the current SWR bank |
| n+1 | 00 |
| n+2 - m | Filename selected |
| m+1 | 00 |

On receiving the call you should check that this is the right OSWORD call number and then see whether you recognise the SWR Title String beginning in byte 3. If you do not you should pass on the block unaltered. Failure to do this could mask out other loaders, including **AFM**'s own routines.

If you recognise the title you should then check that byte 02 is zero. If it is you should replace the parameter block with your list of options, and change byte 02 to the number of options you are returning (1-7). If byte 02 contains 1-7 you are being asked to start the file (explained in a moment), and if it contains a value above 7 you should ignore the call (byte 02 values 8-&FF are reserved for future use by PRES).

The options list returned should look like:

Returning block for acknowledged First Call

| Byte | Contents |
|---|---|
| 00 | FF - do not alter |
| 01 | FF - do not alter |
| 02 | <1-7> - the number of options you are returning |
| 03 - n | Text describing first option |
| n+1 | 00 |
| n+2 - m | Text describing second option (if present) |
| | <repeated for up to 7 loaders> |
| x | 00 - End of last option |

When the OSWORD call returns to **AFM** it will check the value in byte 02. Providing that it is between 1 and 7 it will display that number of options from the parameter block, and wait for the user to choose one.

When the user has chosen, **AFM** will issue the call again:

Second Call - **AFM** telling the loader to execute the file

| Byte | Contents |
|---|---|
| 00 | FF - do not alter |
| 01 | FF - do note alter |
| 02 | <1-7> - The number of the option the user selected |
| 03 - n | Title string of the current SWR bank |
| n+1 | 00 |
| n+2 - m | Filename selected |
| m+1 | 00 |

Like the first call it should be isolated initially by the correct OSWORD call number, and then by the ROM title. However this time byte 02 contains a number between 1 and 7, which is the number of the option which the user chose.

You should start the file in a manner consistent with the option number chosen. **AFM** does not expect the call to return, and will already have restored the original screen mode and palette in preparation for you to start the file.

## How to handle the OSWORD call

This section concludes with an outline of the intended handling of this call:

Verify that it is OSWORD &B5 - exit if not
Check that byte 02 is 00-07 - exit if not
Attempt to match the ROM name - exit if you fail
If byte 02 was 00 return a list of options
If byte 02 was 01-07 execute the loader requested

# AFM'S ERROR HANDLING

As well as reporting errors from the filing system, **AFM** can generate a number of errors of its own, which are detailed in this section. The majority of these errors are things that should not occur, but have been trapped nonetheless.

Too long

A directory pathname has exceeded 128 characters and is too long to store.

Bad memstk

This is an internal error message, and cannot occur in normal use. It is symptomatic of corruption of the memory map workspace.

Not a file

Raised on the read cycle if it cannot find the file it wants to read. This will only occur if something odd has happened, since the file will have been validated when it was marked for copying.

No R attribute

Raised on the read cycle if the file does not have an R attribute (ADFS only) and is unreadable. Again this should not occur normally.

E protected

Raised on the write cycle when the file exists in the target directory and is E protected (ADFS only).

Directory Exists

Raised on the write cycle if a name which **AFM** wants to use for a file turns out to already belong to a directory. This is only likely to occur when a new target disc is being used which was not the one in use when browsing through the source and target directories.

File is locked

Raised on the write cycle if the target file is locked.

No W attribute

Raised on the write cycle if the target file exists, but does not have a W attribute (ADFS only).

File has vanished

Raised on the write cycle if **AFM** finds that a file which was there before isn't now, eg when it comes back to write the second or subsequent part of a file. Usually caused by something mundane like the wrong disc being in the drive.

Can't open file

Raised on the write or read cycles if **AFM** failed to open a file. There is no reason why this should occur in normal use.

Path blocked

Raised on the write cycle if, whilst **AFM** is trying to create a directory path (see the C and R options from the write error routine), it finds that a file already has the name of a directory it wants to use (ADFS only).

# AFM LOADER ROUTINES

This section details the loader routines provided by **AFM** in Menu mode.

## BASIC

| | | |
|---|---|---|
| 1 | `CHAIN` | Start Basic, then CHAIN the file |
| 2 | `LOAD` | Start Basic, then LOAD the file |
| 3 | `*TYPE` | Start Basic, then *TYPE the file |
| 4 | `*DUMP` | Start Basic, then *DUMP the file |
| 5 | `*LIST` | Start Basic, then *LIST the file |
| 6 | `*RUN` | Start Basic, then *RUN the file |
| 7 | `*EXEC` | Start Basic, then *EXEC the file |

## View

| | | |
|---|---|---|
| 1 | `LOAD` | Start View, then LOAD the file |
| 2 | `NEW, LOAD` | Start View, then NEW, and LOAD the file |
| 3 | `READ` | Start View. then READ the file |
| 4 | `NEW, READ` | Start View, then NEW, and READ the file |
| 5 | `READ 1` | Start View, then READ the file to marker 1 |
| 6 | `PRINT` | Start View, then VIEW the file |
| 7 | `PRINTER` | Start View, then load the file as a printer driver |

## ViewSheet

| | | |
|---|---|---|
| 1 | `LOAD` | Start ViewSheet, then LOAD the file |
| 2 | `LW` | Start ViewSheet, then load the file as window definitions |
| 3 | `PRINTER` | Start ViewSheet, then load the file as a printer driver |

## ViewSpell

| | | |
|---|---|---|
| 1 | `LOAD` | Start ViewSpell, then LOAD the file |
| 2 | `READ` | Start ViewSpell, then READ the file |

## ViewStore

| | | |
|---|---|---|
| 1 | `LOAD` | Start ViewStore, then LOAD the file |
| 2 | `LF` | Start ViewStore, then load the file as a format file |
| 3 | `U` | Start ViewStore, then execute the file as a utility |
| 4 | `PRINTER` | Start ViewStore, then load the file as a printer driver |

Note: Options 1-3 will move you back a directory, eg DIR ^ in ADFS, or DIR $ in DFS, to be in the right place for VIEWSTORE.

## WordWise (Plus) & Inter-Word

| | | |
|---|---|---|
| 1 | `LOAD` | Start WW or IW, then load the file using menu command 2 |
| 2 | `LOAD CURSOR` | Start WW or IW, then load to cursor using menu command 4 |

## The BASIC Editor

1 `LOAD` Start BE, then LOAD the file
2 `APPEND` Start BE, then APPEND the file

## Edit

1 `LOAD` Start Edit, then load the file using f2

# AN EXAMPLE OF AN EXPANSION OF THE AFM MENU USING OSWORD

This program, which may be freely used and amended, shows how to co-operate with the OSWORD call used in menu mode to provide expansion facilities.

As it stands it assembles an image to be loaded into sideways RAM, which will replace the built in loader for the BASIC Editor. The features are the same, though the option text differs slightly.

Note that this program will override **AFM**'s own code even if it is in a bank with a lower priority than **AFM**.

After assembly the program checks itself by both checking the length of the assembled code and performing a simple checksum of the code. If either is wrong you will be asked to check the listing again. When the checks are passed the code will be saved, using the filename 'AFMexp'.

If you alter the program, which you are encouraged to do for your own uses, remember to alter or remove the checking code!

```
 10 REM * AFM loader for The BASIC Editor
 20 DIM Code &200
 30 FOR A%=4 TO 6 STEP 2
 40 P%=&8000:O%=Code
 50 [
 60 OPT A%
 70 \
 80 \This listing assembles an example of an interface to the AFM
 90 \menu mode OSWORD call. It produces a sideways Ram image, which
100 \when loaded will replace AFM's own loader for the BASIC editor
110 \
120 \This code may be modified and freely incorporated into user's
130 \programs.
140 \
150 \Writeen by PRES, 1988
160 \
170 \
180 \First the standard language header
190 \
200 BRK:BRK:BRK                     \No language entry
210 JMP SVCE                        \Jump to the service call header
220 EQUB &82                        \Service code only
230 EQUB (c MOD 256)                \Offset to the copyright message
240 EQUB 0:EQUS "AFM Expansion"     \The Title string
250 .c EQUB 0:EQUS "(C) 1988"       \Copyright string must be present
260 EQUB 0
270 \
280 \
290 .SVCE                           \The Service call handler
300 \
310 CMP #8:BEQ SVCE1:RTS            \Only accept call 8 (unknown OSWORD)
320 \
330 .SVCE
340 \
350 LDA &EF:CMP #&B5:BEQ SVCE2:LDA #8:RTS
360                                 \Only accept OSWORD &B5
```

```
 370 \
 380 .SVCE4
 390 PLA:STA &92:PLA:STA &91:PLA:STA &90
 410                                 \Restore saved locations
 420 \
 430 PLA:TAY:LDX &F4:LDA #8:RTS      \and exit without claiming call
 440 \
 450 .SVCE2                          \It is OSWORD B5
 460 \
 470 TYA:PHA:LDA &90:PHA:LDA &91:PHA:LDA &92:PHA
 480                                 \Save Y, and workspace locations
 490 \
 500 LDA &F0:STA &90:LDA &F1:STA &91 \Set up &90/1 to point at OSWORD block
 510 \
 520 LDY #2:LDA (&90),Y:CMP #8:BCS SVCE4
 530                                 \Reject the call if byte 02 is too big
 540 \
 550 INY                             \To the start of the ROM name at 03
 560 \
 570 .SVCE3 LDA (&90),Y:CMP string-3,Y:BNE SVCE4
 580                                 \Compare, and reject if different
 590 \
 600 CMP #0:BEQ SVCE5:INY:BNE SVCE3  \Finish on 0, else go round again
 610 \
 620 .SVCE5
 630 LDY #2:LDA (&90),Y:BNE SVCE6    \If non-zero we have to start file,
 640                                 \If zero, we have to return option list
 650 \
 660 .SVCE7                          \Copy down the option list
 670 LDA table-2,Y:STA (&90),Y:INY:BNE SVCE7
 680 \
 690 .SVCE8                          \Restore workspace and claim call
 700 PLA:STA &92:PLA:STA &91:PLA:STA &90:PLA:TAY:LDX &F4:LDA #0:RTS
 710 \
 720 .bes                            \Command to start the ROM
 730 EQUS ("BE"):EQUB 0
 740 \
 750 .string                         \The ROM title to match
 760 EQUS ("The BASIC Editor"):EQUB 0
 770 \
 780 .table                          \The options table to return
 790 \
 800 EQUB 2                          \Two loaders available
 810 EQUS("*BE Load <file>"):EQUB 0  \Option 1
 820 EQUS("*BE Append <file>"):EQUB 0\Option 2
 830 \
 840 .SVCE6A                         \Option 2 handler (AP for Append)
 850 \
 860 LDY #ASC("A"):JSR INKBRD        \Put A in keyboard buffer
 870 LDY #ASC("P"):JSR INKBRD        \Put P in keyboard buffer
 880 JMP SVCE6C                      \Join the option 1 handler
 890 \
 900 .SVCE6                          \Option handler entry, A=option no.
 910 CMP #2:BEQ SVCE6A \Go to option 2 if it was chosen
 920 \
 930 \                               \which leaves option 1 (L for Load)
 940 LDY #ASC("L"):JSR INKBRD        \Put L in keyboard buffer
 950 \
 960 .SVCE6C                         \Option 2 handler rejoins me here
 970 LDY                             \Put a space in the keyboard buffer
 980 \
 990 LDY #20 \Start of filename. NB will vary with
1000                                 \length of ROM name!
1010 \
1020 .SVCE9                          \Put the filename into the kbrd buffer
1030 \
1040 LDA (&90),Y:BEQ SVCE10          \Finish when we read 0
1050 JSR INKBR2:INY:BNE SVCE9        \Send to keyboard and repeat
```

```
1060 \
1070 .SVCE10
1080 \
1090 LDY #13:JSR INKBRD              \Finish the command with a RETURN
1100 \
1110 LDX #0                         \Copy down the *BE command
1120 .SVCE12
1130 LDA bes,X:STA &110,X:INX:CMP #13:BNE SVCE12
1140                                \Copy up to the RETURN
1150 \
1160 LDX #(&110 MOD 256):LDY #(&110 DIV 256)
1170                                \Set X and to the command address
1180 \
1190 JMP &FFF7                      \and let OSCLI execute it.
1200 \
1210 .INKBRD                        \put Y register in keyboard buffer
1220 \
1230 LDA #138:LDX #0:JMP &FFF4      \using FX 138,0,n
1240 \
1250 .INKBR2                        \put A in kbrd buffer, save Y
1260 STY &92:TAY:JSR INKBRD:LDY &92:RTS
1270 ]
1280 NEXT
1290 REM * Check before saving it - CHANGE THESE IF YOU ALTER THE CODE
1300 IF O%-Code<>269 THEN PRINT "Code is wrong length:please check it":END
1310 Q%=0
1320 FOR F%=Code TO O%-1:Q%=Q%+?F%:NEXT
1330 IF Q%<>&7A2A THEN PRINT "Code has wrong checksum:please check it":END
1340 OSCLI("Save AFMexp "+STR$~(Code)+" "+STR$~O%+" FFFF8000 FFFF8000")
1350 PRINT "Code saved on disc"
```

To aid understanding of the program it is well commented, and has been written to be straightforward to assimilate.

The program assumes the presence of BASIC 2 or better. Users of BASIC 1 should replace the EQUB, EQUS and OSCLI commands with appropriate FN calls.