# J10 ADFS Technical Information

A disc formatted for use with the Advanced Disc Filing System will be either:
- single-sided
in which case it will contain either 40 Or 80 tracks, each divided into 16
.
sectors containing 256 bytes.
- double-sided
in which case it will contain 2 x 80 = 160 tracks, each divided into 16 sectors containing 256 bytes. (The two surfaces are considered to be a single entity).
A given sector may therefore be identified by means of either its track and sector number or by means of its 'absolute' sector number (ie 16*<track number> + <sector number on track> . Absolute sector numbers are used throughout this section.
Files stored by the Advanced Disc Filing System are merely sequences of bytes which always begin at the start of a sector and extend for the number of (complete) sectors necessary to accommodate the data contained on the file (ie there may be a number of 'unused' bytes at the end of the last sector allocated to the file). The last 'data' byte in the file is derived from the file length stored in the catalogue entry for the file ( see below).
Unlike the Disc Filing System ( in which the areas of free space are derived from the catalogue entries for each file), the Advanced Disc Filing System maintains a map of the free space (and other information) in sectors 0 and 1 of each disc. Sectors 2 to 6 inclusive always contain the information relating to the root directory and the remaining sectors contain either information relating to files and subordinate directories or the actual content of files.
Note. Since track 0 is the outermost track on both 40- and 80-track discs, it is possible to access both the Free Space Map and the information relating to the root directory of both types of disc in either type of drive. The same is true for any files or directories which can be guaranteed to exist solely on track 0 (sector numbers 0 - 15), although this facility should be used with caution.
A reference to an absolute sector number is always represented by a 3-byte value.

**The Space Map .**
The Free Space Map is stored in sectors 0 and 1 on each drive. The format being:
**Sector 0**
**Bytes Content**
o- 2 Start sector of first free space
3 - 5 Start sector of second free space
6- 8 Start sector of third free space
.
. Repeated for 82 free space entries
. Reserved
246
247 Reserved
248 Reserved
249 Reserved
250 Reserved
251 Reserved
252 LSB of total number of sectors on disc
253 .
254 MSB of total number of sectors on disc
255 Checksum on free space map, sector 0
Sector 1
Bytes Content
o- 2 Length of first free space (in sectors)
3 - 5 Length of second free space
6 - 8 Length of third free space
. Repeated for 82 free space entries
246. Reserved
247 Reserved
248 Reserved
249 Reserved
250 Reserved
251 - 252 Disc identifier
253 Boot option number ( as set by *OPT4 )
254 Pointer to end of free space list
255 Checksum on free space map, sector 1

**Directory information**

A directory consists of five contiguous sectors on the disc - the root directory is always located on track 0 (sectors 2 - 6). The information relating to the current directory is always resident in the area of RAM reserved for use by the MOS/Filing Systems.

Note that whereas the Disc Filing System uses 18-bit addresses, all addresses held by the Advanced Disc Filing System are 32 bits (4 bytes) in length.

The format for each directory is shown below .

**Bytes Content**

0 Directory Master Sequence Number (in binary coded decimal)
1 - 4 Fixed string identifying the sector as the start of a directory
5 - 14 Name and access string for first directory entry (see note)
15- 18 Load address for first directory entry
19- 22 Execution address for first directory entry
23- 26 Length of first directory entry in bytes
27 - 29 Start sector for first directory entry
30 Sequence number for first directory entry (see note)
31 - 40 Name and access string for second directory entry
41 - 44 Load address for second directory entry
45 - 48 Execution address for second directory entry
49 - 52 Length of second directory entry in bytes
53 - 55 Start sector for second directory entry
56 Sequence number for second directory entry
.
. Repeated for 47 directory entries
1227 0
1228 - 1237 Directory name/ access string
1238 - 1240 Start sector of parent directory
1241 - 1259 Directory title
1260- 1273 Reserved
1274 Directory Master Sequence Number ( in binary coded decimal )
1275 - 1278 Fixed string identifying a directory
1279 0

**Notes**
A directory can take a maximum of47 entries - if there are less than 47, this is indicated by the entry after the last file having a name string starting with &00* Directory entries are held in alphabetical order.
me and directory attributes are stored in the top bit of the first four bytes of the name/access string string:
1st byte bit 7 set/clear indicates R attribute set/not set
2nd byte bit 7 set/clear indicates W attribute set/not set
3rd byte bit 7 set/clear indicates L attribute set/not set
4th byte bit 7 set indicates that the entry is a directory
. bit 7 clear indicates that the entry is a file

Ifa string is shorter than the space reserved for it, the string ends with a &0D and the remaining bytes are not significant.
Each directory has a two-byte master sequence number , held in binary coded decimal format. This value is set to zero when the directory is created.
The master sequence number is incremented every time a change is made to the content of the directory catalogue and the new value is assigned to the sequence number associated with the new/changed directory entry. The sequence number for each directory entry (as displayed by * CAT ) may therefore be used to assess the 'age' of each file.

**J.11 Using DFS and ADFS from assembly language**

The routines described below are implemented by both the Disc Filing System and the Advanced Disc Filing System unless indicated to the contrary .

**OSFIND Open or close a file for byte access**

call address &FFCE

Indirected via &21C (FINDV)

On entry : A defines the action to be taken.

X and/or Y contain values depending upon the action specified.

**Actions specified by A :**

A = 0 (&00) indicates that a file is to be closed.

Y may contain either the <file handle> of the file to be closed or zero, which indicates that all currently open sequential files associated with the current Filing System are to be closed.

A < file handle > is allocated by the Filing System when a file is opened.

A = 64 (&40) indicates that a file is to be opened for input.

X and Y point to the location in memory (X=LSB, Y=MSB) containing the first character of the file name (which must be terminated by a carriage return character) .

The named file must exist.

A = 128 (&80) indicates that a file is to be opened for output. .

X and Y point to the file name as described above.

Ifthenamed file exis . ts, it will be opened and
its file pointer 8et to the start of the file; If
the file does not exist, a new file is created
with a default length depending upon the
Filing System:

DFS . 16K (&4000) bytes

.
ADFS: 64K (&10000) bytes
A = 192 (&C0) indicates that a file is to be opened for input
and output (random access).
X and Y point to the file name as described
above .
The named file must exist.
Onexit : X and Y are preserved
A is preserved ifzero on entry, otherwise A
contains the file handle allocated to the file.
A value of zero indicates that the Filing
System was unable to open the file.
C, N, D and V are undefined.
The interrupt state is preserved but may be
enabled during the operation.

**OSGBPB Read or write a group of bytes**

Call address &FFDl
Indirected via &21A (GBPBV)
On entry : A defines the action to be taken
X = <LSB ofparameter block address>
Y = <MSB of parameter block address >
Parameter block size : 13

**Parameter block format :** XY = <file handle>
XY+l = <LSB of pointer to data in
memory>
XY+2 .
XY+3 .
XY +4 = <MSB of pointer to data in
. memory>
XY+5 = <LSB of number of bytes to
transfer>
XY+6 .
XY+7 .
XY+8 =<MSB of number of bytes to
transfer>
XY+9 =<LSB of sequential pointer
value>
XY+l0 .
XY+ll .
XY + 12= <MSB of sequential pointer
value>
Not aU sections of the parameter block are
used by all actions-

**Actions specified by A :**

A = 1 (&01) Write bytes to file
The number of bytes specified in XY +5 to
XY+8, starting at the address specified in
XY + 1 to XY +4 are written to the file using
the sequential pointer value specified in
XY+9 to XY+12.

A = 2 (&02) Append bytes to file
The number of bytes specified in XY+5 to
XY+8, starting at the address specified in
XY+l to XY+4 are appended to the
specified file (ie the bytes are written to the
file starting at the current value of the file
pointer, which is incremented for each byte
transferred) .

A = 3 (&03) Read bytes from a specified position in a file
The number of bytes specified in XY + 5 to
XY + 8 are read from the file starting at the
pointer value specified in XY+9 to XY+12.

Bytes read are placed .
In memory locations
starting at the location contained in XY + 1
to XY+4.
A = 4 (&04) Read bytes from the current position in the
file
The number of bytes specified in XY + 5 to
XY +8 are read from the file starting at the
current value of the file pointer (ie the value
in XY+9 to XY+12 is ignored). Bytes read
are placed in memory locations starting at
the address specified in XY+1 to XY+4.
A- 5 (&05) Read title, option and drive
The current title , option and drive number
are returned in the area of memory specified
in XY+1 to XY+4.
Under DFS, the title is that relating to the
~~~ disc in the current drive; under ADFS, the
title is that of the current directory .
The format of the returned data is:
length of title (1 byte)
. title string in ASCII (length as specified)
start option (1 byte)
drive number (1 byte)
The contents of XY and XY+5 to XY+13
are ignored.
A = 6 (&06) Read current drive and directory name
The current drive and the name (rather
than the title) of the current directory are
returned in the area of memory specified in
XY+1 to XY+4
The format of the returned data is :
. length of drive number (1 byte)
drive number in ASCII (see note)
length of directory name (1 byte)
directory name in ASCII (length as
specified)
~~~

The drive number will always be one byte in
length for DFS and ADFS - this call may
return a string of more than one byte for
other Filing Systems
The contents of XY and XY+5 to XY+13
are ignored.
A = 7 (&07) Read current library drive and name
As for A = 6 but with relation to the current
library .
A = 8 (&08) Read file names from the current directory
The content ofXY+5 to XY+8 is treated as
the number of filenames to transfer.
, XY+9
to XY+13 contain a pointer to the first
name to be transferred (i.e. if it is zero, the
search will begin with the first file) .
The disc Master Sequence Number is
returned in XY and the file names are
returned in locations specified by XY + 1 to
XY+4. The format ofthe returned data is:
length of first file name (1 byte)
first file name in ASCII (length as
specified)
length of second file name (1 byte)
second file name in ASCII (length as
specified)
.
. Repeated as specified by XY+5 to XY+8
.
**On exit :** A, X and Y are preserved.
N, V and Z are undefined
C = 0 indicates that the transfer was .
completed
C = 1 indicates that the transfer was
incomplete for some reason.

The values in the parameter block are updated to reflect the position after the transfer (i.e. XY+1 to XY+4 contain the address of the byte after the last byte transferred to or from the file and XY+5 to XY + 8 contain the number of bytes remaining to be transferred - zero ifC = 0). The interrupt state is preserved, but may be enabled during the call.

**OSBPUT Write a byte to an open file**
Call address &FFD4
Indirected via &218 (BPUTV)
On entry : Y = <file handle>
A = <byte to be written>
. The byte is written to the specified file using the current value of the file pointer . The pointer is incremented for each byte written.
On exit : A, X and Y are preserved
C, N, V and Z are undefined
The interrupt state is preserved but may be enabled during the call.

**OSBGET Read a single byte from an open file**
Call address &&FFD7
Indirected via &216 (BGETV)
On entry : Y = <file handle>
The byte at the current file pointer position is returned in A. The file pointer is incremented for each byte read.

On exit : A = <byte read>
X and Y are preserved
N, V and Z are undefined
C = 0 indicates that the transfer was
completed
C = 1 indicates that end of file was
encountered and that the value in
A should be discarded.
The interrupt status is preserved but may
be enabled during the call.

**0SARGS Read filing system information**
Call address &FFD A
Indirected via &214 (ARGSV)
On entry : A specifies the action to be taken.
X points to a four-byte area in page zero
Y contains either a file handle or zero
**Actions specified by A :**
A = 0 (&00) Y = 0 Return Filing System
number in A.
Y = <file handle> Return sequential
pointer for file in
locations specified by
X.
A = 1 (&01) Y = 0 Return address of
remainder of the last
command line in
locations specified by
X.
Y = <file handle> Write sequential
pointer for file from
locations specified by
X.

A = 2 (&02) Y = <file handle> Return length of file
(in bytes) in locations
specified by X.
A = 255 (&FF) Y = 0 Ensure any buffered
data has been written
to aU files.
Y = <file handle> Ensure any buffered
data has been written
to the specified file.
**Onexit** : X and Y are preserved
A is preserved except when A=0 and Y=0
on entry , in which case it contains the Filing
System number (see section G.1).
The interrupt status is preserved although
interrupts may be enabled during the call.
Note
Addresses returned in the four-byte block
specified by the content of X ALW AYS point
to the IO processor and should therefore be
read using OSWORD 5.

**OSFILE Load or save a complete file**
Call address &FFDD
Indirected via &212 (FILEV)
On entry : A specifies the action to be performed
X - - <LSB of parameter block
address>
Y - - <MSB of parameter block
address>
Parameter block size : 18

**Parameter block format :** XY = <LSB of address of file name>
XY + 1 = <MSB of address of file name>
XY+2 = <LSB of load address for file>
XY+3 .
XY+4 .
XY+5 = <MSB ofload address for file>
XY + 6 - - <LSB of execution address for file>
XY+7 .
XY+8 .
XY + 9 = <MSB of execution address for file>
XY+10 LSB ofeither <start address> or <length>
XY +11 .
XY+12 .
XY+13 =MSB of either <start address> or <length>
XY+14 = LSB of either <end address>or
XY+15 <file attributes>
.
XY+16 .
XY+17 =MSB ofeither <end address or <file attributes>
The file name pointed to by XY and XY + 1 must terminate with a carriage return.

**Actions specified by A :**

A = 0 (&00) Save a block of memory.
On entry, XY+10 to XY+13 contain the start address ofthe data in memory. XY+14
,
. to XY + 17 contain the end address.
On exit, XY+10 to XY+13 are replaced by the length ofthe file and XY+14 to XY+17 are replaced by the file attributes assigned by the Filing System (see below). .
A = 1 (&01) Write catalogue information for the named file.

The load address, execution address and me
attributes from the parameter block are
written to the named me' s catalogue entry .
A = 2 (&02) Write load address (only) for the named me.
A = 3 (&03) Write execution address (only) for the
named file.
A = 4 (&04) Write attributes (only) for the named file.
A = 5 (&05) Read catalogue information for the named
file.
The load address , execution address , length
and file attributes from the named file's
catalogue entry are read into the parameter
block.
On exit, A contains the file type:
0 indicates Not found
1 indicates File found
2 indicates Directory found (ADFS
only)
&FF indicates a protected file (ie E
attribute set) (ADFS only)
A = 6 (&06) Delete the named file.
The information in the named file's
catalogue entry is transferred to the
parameter block and then deleted from the
catalogue .
A = 7 (&07) Create an empty file.
The size of the empty file is determined by
the start address and end address entries in
the parameter block but no data is
transferred. It is usually convenient to set
the start address bytes to zero and use the
end address bytes to define the length of the
file.

A =255 ( &FF) Load the named file into memory at a
location determined by the content of
parameter block byte XY + 6:
If XY+6 is zero, the file is loaded into
memory at the address specified in XY+2 to
XY+5.
If XY+6 is non-zero, the file is loaded into
memory using the files own load address
(see section J.10),
On exit : A is undefined (except for OSFILE 5)
X and Y are preserved
C, N, V and Z are undefined
The interrupt status is preserved but may
be enabled during the call.

**Note**

Although four bytes (XY+14 to XY+17) are allocated for the attributes associated with an object, only the least significant four bits of XY +14 have any meaning under DFS or ADFS:

**bit Meaning when set**

0 R attribute set ( ADFS only)
1 W attribute set (ADFS only)
2 E attribute set ( ADFS only)
3 L attribute set

**OSWORD**

Three OSW ORD calls are recognised by the Disc Filing System and four by the Advanced Disc Filing System. In each case, the call number is supplied in A and X and Y must point to a control block in memory, details of which are given in each description.

Call address &FFF1
Indirected via &20C (WORDV)

**Actions specified by A :**
A = 112 (&70) Read Master Sequence number and status byte (ADFS).
XY points to a two-byte block in memory .
On return, XY contains the Master Sequence number for the current directory (in binary-coded decimal (BCD) format).
XY+1 contains a status byte; bits set have the following significance :

| bit | Meaning when set |
|---|---|
| 0 | File ensuring in progress (IRQ pending) |
| 1 | Bad free space map |
| 2 | *OPT1 setting |
| 3 | undefined |
| 4 | undefined |
| 5 | Winchester disc controller present |
| 6 | Tube in use by ADFS |
| 7 | Tube present |

A = 113 (&71) Read free space (ADFS)
XY points to a four-byte block in memory .
On return, the 32-bit value of the available free space (equivalent to the value output by *FREE) is placed in this block.
A = 114 (&72) General read/write function (ADFS)
XY points to a 15-byte parameter block with the following format :
XY zero
XY+1 <LSB of pointer to data in memory >
XY+2 .
XY+3 . .
XY+4 <MSB of pointer to data in memory>
XY+5 <&08> to read; <&0A> to write
XY+6 bits 5-7 <drive> (see below)
bits 0-4 <5 high order bits of absolute sector number >

XY+7 <8 middle order bits of absolute sector number >
XY + 8 <8 low order bits of absolute sector number>
XY+9 <sector count for read operations>
XY +10 unused
XY + 11 <LSB of data length for write operations >
XY+12 .
XY+13 .
XY+14 <MSB of data length for write operations >
Bits 5-7 of XY +6 are ORed with the current drive number to give the drive number to be accessed. The absolute sector number is a 21-bit value, high order bits first.
On exit, XY contains 0 if the operation was completed successfully; any other value indicates a disc error, typically.

**Error code Meaning**

72 (&48) Cyclic Redundancy check error
80 (&50) Sector not found
96 (&00) Bad command
97 (&61) Bad address
99 (&63) Volume error
101 (&65) Bad drive

A. = 115 (&73) Read last error information (ADFS)
If this call is made immediately after a disc error of some kind (including a data error in sequential filing) , error information is returned in a 5-byte control block with the following format :
.
XY <8 low order bits of absolute sector number>
XY + 1<8 middle-order bits of absolute sector number >

XY+2 bits 5-7 <drive number>
bits 0-4 <5 high order bits of absolute
sector number>
XY + 3 Disc error number (see below)
. XY + 4 Channel number of file where error
occurred
. Only one of the contents of XY+3 and
XY +4 will be valid for a given type of error.
Where a disc error number is appropriate,
the top bit is set if the absolute sector
number for the operation was valid. The
channel number (where appropriate) is
given in hexadecimal.
A = 125 (&7D) Read Master Sequence number (DFS)
XY points to a single byte in memory. On
exit, the specified byte contains the Master
Sequence number (in binary-coded decimal
(BCD) format) for the current drive.
A = 126 (&7E) Read disc size (DFS)
XY points to three-byte block in memory.
On exit, the block contains the total number
of bytes associated with the current drive
(LSB first) :
40-track : &19000
80-track : &32000
A = 127 (&7F) General read/write function (DFS)
XY points to a 10-byte parameter block with
following format:
XY <drive number>
XY +1 <LSB of pointer to data in
memory >
XY+2 . .
XY+3 .
XY +4 <MSB of pointer to data in
memory>

XY+5 3 (see below)
XY+6 <&53> to read; <&4B> to write
XY+7 <track number>
XY + 8 <sector number>
XY+9 bits 5-7 <size ofsector in bytes>
bits 0-4 <number of sectors> (see
below)
XY+10 result (see below)
XY 3 contains the number of parameters associated with the 'command' specified in XY+6 - this value will always be 3 for general read/write operations.
The sector size in bits 5-7 of XY +9 is a coded value which denotes the number of bytes in each sector of the disc:

**bit 7 bit 6 bit 5**

0 0 0 128 bytes/sector
0 0 1 256 bytes/sector
0 1 0 512 bytes/sector
etc.
All disc sectors contain 256 bytes.
Bits 0-4 of XY +9 contain the number of sectors to be read/written by the call.
On exit from a read or write operation, XY + 10 will contain zero if the operation was successful or a disc error number , typically.

**Error code Meaning**

12 (&0C) Cyclic redundancy error ( ID )
14 (&0E) Cyclic redundancy error (data) .
20 (&14) Track 0 not found
22 (&16) Write fault
24 (&18) Sector not found