9 AMSDOS

AMSDOS is a disc operating system used with all the CPC range of computers, of course, in the case of the 464 the DDI-1 has to be fitted. AMSDOS enables programs to be access disc files in a similar manner to cassette files, indeed existing programs which currently use the cassette should be able to use disc files with little or no modification. The main source of incompatability will be filenames, AMSDOS filenames must conform to CP/M standards but cassette filenames are far less restricted.

AMSDOS has been designed to complement CP/M, not to compete with it. They share the same file structure and can read and write each other's files.

AMSDOS resides in the same ROM as the CP/M BIOS.

9.1 Features.

AMSDOS provides the following facilities:

> Switching the cassette input and output streams to and from disc. So that all the facilities available on the cassette become available on disc.
>
> Displaying the disc directory.
>
> Erasing disc files.
>
> Renaming disc files.
>
> Selecting the default drive and user.

Whenever AMSDOS creates a new file it is always given a name with a type part of **.$$$** regardless of the given name. When the file is closed any previous version of the file is renamed with a **.BAK** type part and the new version is renamed from **.$$$** to its proper name. Any existing **.BAK** version is deleted. This gives an automatic one level file back-up.

For example, if the disc contains the files **FRED.BAS** and **FRED.BAK** and the user opens a file called **FRED.BAS** then AMSDOS will create a new file called **FRED.$$$** When the file is closed the existing **FRED.BAK** is deleted, **FRED.BAS** is renamed to **FRED.BAK** and **FRED.$$$** is then renamed to **FRED.BAS**.

All AMSDOS facilities are implemented either by intercepting the cassette firmware calls or by external commands.

The intercepted firmware calls are:

- CAS IN OPEN
- CAS IN CHAR
- CAS IN DIRECT
- CAS RETURN
- CAS TEST EOF
- CAS IN CLOSE
- CAS IN ABANDON
- CAS OUT OPEN
- CAS OUT CHAR
- CAS OUT DIRECT
- CAS OUT CLOSE
- CAS OUT ABANDON
- CAS OUT CATALOG

The remaining cassette firmware calls are not intercepted and remain unaffected.

Full descriptions of both the tape and disc versions of these routines are given in section 15.

The AMSDOS external commands are:

A	Selects default drive A:
B	Selects default drive B:
CPM	Cold boot CP/M
DIR	Display disc directory
DISC	Redirect cassette routines to disc
DISC.IN	Redirect cassette input routines to disc
DISC.OUT	Redirect cassette output routines to disc
DRIVE	Select default drive
ERA	Erase files
REN	Rename files
TAPE	Redirect cassette routines to cassette
TAPE.IN	Redirect cassette input routines to cassette
TAPE.OUT	Redirect cassette output routines to cassette
USER	Select default user

From BASIC all these commands must be preceded by a '|'.

Some of these commands require parameters.

Full descriptions of these external commands are given in section 20.

9.2 Filenames

AMSDOS filenames are upwards compatible with CP/M filenames. But in addition the user number may also be specified and non-significant spaces are permitted before and after the name and any embedded punctuation.

Examples:

`ANAME`	Default user, drive, and type
`10:WOMBAT.TXT`	Default drive, and user number 10
`2A:WOMBAT.TXT`	User 2, on Drive A:
`*.*`	Default drive, user, and all files
`5B : POSSUM . $$$`	A name with non-significant spaces
`a:aard?ark`	Lowercase, AMSDOS will convert to uppercase

If given, the user number must be in the range 0..15, the drive letter must be A or B. If either the user or the drive is given they must be followed by a colon.

The following characters may be used in the name and type parts:

```
a-z A-Z 0-9 ! " # $ & ' + - @ ^ ' } {
```

Any other characters will cause the commands to fail with the message:

```
BAD COMMAND
```

The characters '**?**' and '*****' are wildcards, that is, when placed within a filename or type it will be interpreted as 'any valid character'. For example if the filename '**G??E??.B***' was used in the **|DIR** command then the files '**GAME1.BAS**' '**GAME1.BAK**' '**GAME29.BAS**' and '**GREET.BAS**', and any matching combinations, would be displayed in the directory.

When parsing a filename, AMSDOS shifts lower case letters into uppercase and removes bit 7.

If the user or drive is omitted then the current default values are assumed. These defaults may be set by the user.

If the type part is omitted then a default type is assumed. This depends on the context in which the name is being used, but usually a default type part of three spaces is assumed.

9.3 File Headers

Cassette files are subdivided into 2K blocks, each of which is preceded by header. CP/M files do not have headers. AMSDOS files may, or may not have a header depending on the contents of the file. This will not cause problems for programs written in BASIC but it is an important difference between cassette and disc files.

Unprotected ASCII files do no have header. All other AMSDOS files have a single header in the first 128 bytes of the file, the header record. These headers are detected by checksumming the first 67 bytes of the record. If the checksum is as expected then a header is present, if not then there is no header. Thus it is possible, though unlikely, that a file without a header could be mistaken for one with a header.

The format of the header record is as follows:

Bytes	0…63	Cassette/Disc header (see below)
	64…66	Length of the file in bytes, excluding the header record. 24 bit number, least significant byte in lowest address
	67…68	Sixteen bit checksum, sum of bytes 0..66
	69…127	Undefined

The use that the cassette manager makes of the header is described in section 8.4. AMSDOS uses the header as follows:

	Bytes	
Filename	0	User number, #00..#0F
	1…8	Name part, padded with spaces
	9…11	Type part, padded with spaces
	12…15	#00
Block number	16	Not used, set to 0
Last block	17	Not used, set to 0
File type	18	As per cassette
Data length	19…20	As per cassette
Data location	21…22	As per cassette
First block	23	Set to #FF, only used for output files
Logical length	24…25	As per cassette
Entry address	26…27	As per cassette
Unallocated	28…63	As per cassette

When a file without a header is opened for input a fake header is constructed in store as follows:

	Bytes	
Filename	0	User number, #00..#FF
	1..8	Name part, padded with spaces
	9..11	Type part, padded with spaces
	12..15	#00
File type	18	#16, unprotected ASCII version 1
Data location	19..20	Address of 2K buffer
First block	23	#FF

All other fields are set to zero.

9.4 Disc Organisation

AMSDOS and the CP/M 2.2 BIOS support three different disc formats: SYSTEM format, DATA ONLY format, and IBM format. The CP/M Plus BIOS supports the SYSTEM and DATA formats but not IBM format.

The BIOS automatically detects the format of a disc. Under CP/M this occurs for drive A at a warm boot and for drive B the first time it is accessed. Under AMSDOS this occurs each time a disc with no open files is accessed. To permit this automatic detection each format has unique sector numbers as detailed below.

3 inch discs are double sided, but only one side may be accessed at a time depending on which way round the user inserts the disc. There my be different formats on the two sides.

Common To All Formats

Single sided (the two sides of a 3 inch disc are treated separately).

512 byte physical sector size.

40 track numbered 0 to 39.

1024 byte CP/M block size.

64 directory entries.

System Format

9 sectors per track numbered #41 to #49.

2 reserved tracks.

2 to 1 sector interleave.

The system format is the main format supported, CP/M can only be loaded (Cold Boot) from a system format disc. CP/M 2.2 also requires a system format disc to warm boot. The reserved tracks are used as follows:

Track 0 sector	#41:	boot sector.
Track 0 sectors	#42:	configuration sector
Track 0 sectors	#43..#47:	unused
Track 0 sectors	#48..#49:	and
Track 1 sectors	#41..#49:	CCP and BIOS

CP/M Plus only uses Track 0 sector #41 as a boot sector
Track 0 sector #42...#49 and Track 1 are unused.

Note: Another format called 'VENDOR' format is a special version of system format which does not contain any software on the two reserved tracks. It is intended for use in software distribution.

Data Only Format

9 sectors per track numbered #C1 to #C9.

0 reserved tracks.

2 to 1 sector interleave.

This format is not recommended for use with CP/M 2.2 since it is not possible to ‘ warm boot’ from it. However, because there is a little more disc space available it is useful for AMSDOS or CP/M Plus.

IBM Format

8 sectors per track numbered 1 to 8

1 reserved track

no sector interleave

This format is logically the same as the single-sided format used by CP/M on the IBM PC. It is intended for specialist use and is not otherwise recommended as it is not possible to ‘ warm boot’ from it.

9.5 Boot Sector

In order that non-CP/M systems may be implemented at a later date the BIOS initialization is performed, in part, by a boot program which is read from the disc before attempting to load CP/M. In the non-CP/M case the boot program would not jump to the warm boot routine but go on its own way, using the BIOS and firmware routines as desired.

The boot program is in the boot sector which is the first sector (sector #41) on track 0.

During a cold boot the BIOS is initialized into a minimum state before loading and executing the boot program. This state is as follows.

> All routines in the ROM copy of the BIOS jumpblock all the routines in the extended jumpblock are available.
>
> Alternate and IY register saving is enabled.
>
> Interrupts are indirected via the BIOS and run on the BIOS's stack.
>
> Disc messages are enabled.
>
> The initial command buffer is empty.
>
> The IOBYTE at #0003 is initialized to #81 (**`LST:=LPT:`**, **`PUN:=TTY:`**, **`RDR:=TTY:`**, **`CON:=CRT:`**).
>
> The current drive at #0004 is initialized to #00.
>
> The serial interface is not initialized.
>
> The CCP and BDOS are not in store.
>
> The CP/M jumps at #0000 and #0005 are not initialized.

The boot sector is read and loaded into store at #0100; the stack pointer is initialized to a value immediately below the BIOS's data (#AD33 is normal) area and the boot program is entered at #0100. The boot program may use store from #0100 upwards until it reaches the stack.

To run CP/M the boot program must, at least, jump to the warm boot entry in the ROM jumpblock.

The boot program for CP/M 2.2 loads and obeys the configuration sector and then warm boots CP/M.

The boot program for CP/M Plus searches for, loads and executes a file with the type part .EMS.

The boot program has the following interface:

Entry:

SP=highest address available+1 (a good place for the stack)
BC=address of ROM copy of BIOS jumpblock (BOOT)

Exit:
To run CP/M the program should jump to the WBOOT entry in the above jumpblock

The ROM copy of the BIOS jumpblock should not be used at any time (indeed, only the boot program knows where it is).

9.6 AMSDOS Messages

AMSDOS uses the CP/M 2.2 BIOS in order the access the disc. Thus the BIOS messages will be displayed in the event of a disc error. This section explains the meaning of the AMSDOS messages.

In the following <drive> means A or B, <filename> means an AMSDOS filename.

```
Bad command
```

There is a syntax error in a command or filename.

```
<filename> already exists
```

The user is trying to rename a file to a name which is already in use.

```
<filename> not found
```

The user is trying to open for input, erase or rename a file that does not exist.

```
Drive <drive>: directory full
```

There is no more free directory entries (64 directory entries per disc).

```
Drive <drive>: disc full
```

There are no more free disc blocks.

```
Drive <drive>: disc changed, closing <filename>
```

The user has changed the disc while files were still open on it.

```
<filename> is read only
```

The user is trying to erase or rename a file which is marked read-only. May also be caused by closing a file when existing version of the file is read-only.

9.7 BIOS Facilities Available to AMSDOS

AMSDOS uses the CP/M BIOS 2.2 to access the disc. In order that a program running under AMSDOS may access the disc directly nine of the BIOS extended jumpblock routines are available. The routines are accessed as external commands. An example of using these commands is given in section 10.6.

NOTE: The BIOS extended jumpblock itself is not available, indeed it does not exist in the AMSDOS environment.

The BIOS routines available and their command names are as follows:

`SET MESSAGE`	Ctrl A	(#01)
`SETUP DISC`	Ctrl B	(#02)
`SELECT FORMAT`	Ctrl C	(#03)
`READ SECTOR`	Ctrl D	(#04)
`WRITE SECTOR`	Ctrl E	(#05)
`FORMAT TRACK`	Ctrl F	(#06)
`MOVE TRACK`	Ctrl G	(#07)
`GET DR STATUS`	Ctrl H	(#08)
`SET RETRY COUNT`	Ctrl I	(#09)

These routines are described in section 19.

The word at #BE40 contains the address of the disc parameter header vector. Disc parameter headers and extended disc parameter blocks may be patched as required (see section 9.8).

Only the BIOS facilities mentioned here may be used from a program running under AMSDOS.

9.8 Store requirements

When initialized AMSDOS reserves #500 bytes of memory from the memory pool and the kernel reserves another 4 for its external command chaining information.

When loading a machine code program from disc into store using the AMSDOS routine CAS IN DIRECT it is important that AMSDOS's variables are not overwritten. This presents a problem since in general it is not possible to discover where these variables are! This is because variables for expansion ROMs are allocated dynamically. Note that this problem does not arise when loading from the cassette since the cassette manager's variables are in the firmware variable area.

AMSDOS reserves store from the top of the memory pool so the simplest solution is to always load machine code programs into the bottom of store. The program can then relocate itself to a higher address if required.

Alternatively the machine code program could be loaded in two stages: first load and run a small loader in the bottom of store. The action of MC BOOT PROGRAM will have shut down all RSXs and extension ROMs. The loader program should now initialize AMSDOS using KL INIT BACK thus forcing AMSDOS variables to be wherever you so wish. The loader can now load the machine code program using the AMSDOS routines CAS IN OPEN, CAS IN DIRECT, and CAS IN CLOSE together with MC START PROGRAM.

In order to initialize AMSDOS using KL INIT BACK, AMSDOS's ROM number is required. To determine AMSDOS's ROM number look at any of the intercepted cassette jumpblock entries with the DISC routines selected. Each entry is a far call, the address part of which points at a three byte far address, the third part of the far address is the ROM number. Obviously this should be done before AMSDOS is shut done.

Existing machine code programs, developed on cassette systems without any expansion ROMs, frequently only use store to #ABFF in order to avoid BASICs variables. These can easily be modified to use AMSDOS. Write some machine code to initialize AMSDOS using KL INIT BACK. AMSDOS will reserve RAM down to #ABFC, almost the same as used by BASIC.

9.9 Extended Disc Parameter Blocks

In order to facilitate reading and writing ‘ foreign’ discs of differing formats, all the parameters concerning a drive are kept in RAM in an extended CP/M disc parameter block (XPB). The knowledgeable user may patch an XPB.

There are two XPBs, one per drive.

XPB structure:

bytes	0..14:	standard CP/M 2.2 DPB (see below).
byte	15:	first sector number.
	16:	number of sectors per track.
	17:	gap length (read/write).
	18:	gap length (format).
	19:	filler byte for formatting.
	20:	$\log_2$(sector size)-7, 'N' for ?PD765A.
	21:	sector size/128
	22:	reserved: current track (set by BIOS).
	23:	reserved: #00 => not aligned, #FF => aligned (set by BIOS)
	24:	#00 => auto-select format, #FF => don't auto-select format.

The XPB for a drive may be found by accessing the Disc Parameter Header (DPH) vector. The first word of the DPH is the address of the XPB for drive A, the second word is the address of the XPB for drive B. The address of the DPH is stored at location #BE40.

The values stored in the standard CP/M 2.2 DPB (Disc Parameter Block) are often derived from the data block allocation size, BLS, which is the number of bytes in a block and which may be 1024, 2048, 4096, 8192 or 16384. The value of BLS is not stored in the DPB but it may be deduced from the values stored there. CP/M plus has a slightly different DPB. The CP/M 2.2 DPB is laid out as follows:

bytes	0..1	(SPT):	Total number of 128 byte records on each track.
	2	(BSH):	$\log_2$BLS-7.
	3	(BLM):	BLS/128-1.
	4	(EXM):	If DSM<256 then BLS/1024-1 else BLS/2048-1.
	5..6	(DSM):	Total size of disc in blocks excluding any reserved tracks.
	7..8	(DRM):	Total number of directory entries -1.
	9..10	(AL0/1):	Bit significant representation of number of directory blocks (#0080 => 1, #00C0 => 2 etc).
	11..12	(CKS):	Length of checksum vector. Normally DRM/4+1 but if checksumming is not required then 0.
	13..14	(OFF):	Number of reserved tracks. This is also the track on which the directory starts.

The XPBs for the different formats are initialized as follows:

System format

36	records per track
3	block shift
7	block mask
0	extent mask
170	number of blocks-1
63	number of directory entries-1
#00C0	2 directory blocks
16	size of checksum vector
2	reserved track
#41	first sector number
9	sectors per track
42	gap length (read/write)
82	gap length (format)
#E9	filler byte
2	$\log_2$(sector size)-7
4	records per track
0	current track
0	not aligned
0	do auto select format

Data only format

36	records per track
3	block shift
7	block mask
0	extent mask
179	number of blocks-1
63	number of directory entries-1
#00C0	2 directory blocks
16	size of checksum vector
0	reserved track
#C1	first sector number
9	sectors per track
42	gap length (read/write)
82	gap length (format)
#E9	filler byte
2	$\log_2$(sector size)-7
4	records per track
0	current track
0	not aligned
0	do auto select format

IBM Format

32	records per track
3	block shift
7	block mask
0	extent mask
155	number of blocks-1
63	number of directory entries-1
#00C0	2 directory blocks
16	size of checksum vector
1	reserved track
#01	first sector number
8	sectors per track
42	gap length (read/write)
80	gap length (format)
#E9	filler byte
2	$\log_2$(sector size)-7
4	records per track
0	current track
0	not aligned
0	do auto select format