# CPC464

# DDI-1 FIRMWARE

**SOFT 158A:** *The Complete CPC 464*
*DISC Operating System ROM Specification*

Amsoft

AMSTRAD
CPC 464

CPC464

# DDI-1 FIRMWARE

**SOFT 158A:** *The Complete CPC 464*

*DISC Operating System ROM Specification*

# SOFT158A
# DDI-1 Firmware

***An appendix to SOFT158, describing the CPC464 disc interface ROM routines and explanations***

Paul Overell, Locomotive Software

**Published by AMSOFT, a division of**
Amstrad Consumer Electronics plc
Brentwood House
169 Kings Road
Brentwood
Essex

First edition 1984

Amsoft

Scanned by The King
for
WWW.CPCWIKI.COM

# DDI-1 FIRMWARE

**SOFT 158A:** *The Complete CPC 464 DISC Operating System ROM Specification*

# AMSTRAD CPC 464

# SOFT158A

# DDI-1 Firmware

***An appendix to SOFT158, describing the CPC464 disc interface ROM routines and explanations***

Paul Overell, Locomotive Software

AMSDOS BIOS BOOTGEN.COM CAS CATALOG(DISC) CAS IN ABANDON (DISC) CAS IN CHAR (DISC) CAS IN CLOSE (DISC) CAS IN DIRECT (DISC) CAS IN OPEN (DISC) CAS OUT ABANDON (DISC) CAS OUT CHAR (DISC) CAS OUT CLOSE (DISC) CAS OUT DIRECT (DISC) CAS OUT OPEN (DISC) CAS RETURN (DISC) CAS TEST EOF (DISC) CHKDISC.COM CLOAD.COM COPYDISC.COM CP/M CSAVE.COM DO IN DO IN STATUS DO OUT DO OUT STATUS D1 IN D1 IN STATUS D1 OUT D1 OUT STATUS DISCCHK.COM DISCCOPY.COM ENTER FIRMWARE FILECOPY.COM FORMAT.COM FORMAT TRACK GET DR STATUS IDBYTE MOVCPM.COM MOVE TRACK READ SECTOR RSX SELECT FORMAT SET CMND BUFFER SET MESSAGE SET REG SAVE SET RETRY COUNT SET SID SETUP.COM SETUP DISC SYSGEN.COM WRITE SECTOR AMSDOS BIOS BOOTGEN.COM CAS CATALOG(DISC) CAS IN ABANDON (DISC) CAS IN CHAR (DISC) CAS IN CLOSE (DISC) CAS IN DIRECT (DISC) CAS IN OPEN (DISC) CAS OUT ABANDON (DISC) CAS OUT CHAR (DISC) CAS OUT CLOSE (DISC) CAS OUT DIRECT (DISC) CAS OUT OPEN (DISC) CAS RETURN (DISC) CAS TEST EOF (DISC) CHKDISC.COM CLOAD.COM COPYDISC.COM CP/M CSAVE.COM DO IN DO IN STATUS DO OUT DO OUT STATUS D1 IN D1 IN STATUS D1 OUT D1 OUT STATUS DISCCHK.COM DISCCOPY.COM ENTER FIRMWARE FILECOPY.COM FORMAT.COM FORMAT TRACK GET DR STATUS IDBYTE MOVCPM.COM MOVE TRACK READ SECTOR RS X SELECT FORMAT SET CMND BUFFER SET MESSAGE SET REG SAVE SET RETRY COUNT SET SID SETUP.COM SETUP DISC SYSGEN.COM WRITE SECTOR AMSDOS BIOS BOOTGEN.COM CAS CATALOG(DISC) CAS IN ABANDON (DISC) CAS IN CHAR (DISC) CAS IN CLOSE (DISC) CAS IN DIRECT (DISC) CAS IN OPEN (DISC) CAS OUT ABANDON (DISC) CAS OUT CHAR (DISC) CAS OUT CLOSE (DISC) CAS OUT DIRECT (DISC) CAS OUT OPEN (DISC) CAS RETURN (DISC) CAS TEST EOF (DISC) CHKDISC.COM CLOAD.COM COPYDISC.COM CP/M CSAVE.COM DO IN DO IN STATUS DO OUT DO OUT STATUS D1 IN D1 IN STATUS D1 OUT D1 OUT STATUS DISCCHK.COM DISCCOPY.COM ENTER FIRMWARE FILECOPY.COM FORMAT.COM FORMAT TRACK GET DR STATUS IDBYTE MOVCPM.COM MOVE TRACK READ SECTOR RSX SELECT FORMAT SET CMND BUFFER SET MESSAGE SET REG SAVE SET RETRY COUNT SET SID SETUP.COM SETUP DISC SYSGEN.COM WRITE SECTOR AMSDOS BIOS BOOTGEN.COM CAS CATALOG(DISC) CAS IN ABANDON (DISC) CAS IN CHAR (DISC) CAS IN CLOSE (DISC) CAS IN DIRECT (DISC) CAS IN OPEN (DISC) CAS OUT ABANDON (DISC) CAS OUT CHAR (DISC) CAS OUT CLOSE (DISC) CAS OUT DIRECT (DISC) CAS OUT OPEN (DISC) CAS RETURN (DISC) CAS TEST EOF (DISC) CHKDISC.COM CLOAD.COM COPYDISC.COM CP/M CSAVE.COM DO IN DO IN STATUS DO OUT DO OUT STATUS D1 IN D1 IN STATUS D1 OUT D1 OUT STATUS DISCCHK.COM DISCCOPY.COM ENTER FIRMWARE FILECOPY.COM FORMAT.COM FORMAT TRACK GET DR STATUS IDBYTE MOVCPM.COM MOVE TRACK READ SECTOR RSX SELECT ET

**Published by AMSOFT, a division of**
Amstrad Consumer Electronics plc
Brentwood House
169 Kings Road
Brentwood
Essex

First edition 1984

BLANK PAGE

# Appendix XIII
# The Disc System.

# Chapter 1 - Overview

The disc system hardware consists of a DDI-1 interface and one or two FD-1 floppy disc drives. The DDI-1 interface plugs into the edge connector at the rear of the CPC464 labelled 'FLOPPY DISC'. The disc drive(s) are connected to the interface via a single ribbon cable. Each disc drive takes a single 3" floppy disc. Either side of the disc may be used, depending on which way up the disc is inserted into the drive. The DDI-1 interface contains a 16K expansion ROM, 8K of which contains the disc driving software, the remainder being used by DR LOGO.

Two disc operating systems are provided: AMSDOS, which enables BASIC programs to use disc files in much the same way as cassette files; and CP/M 2.2, the industry standard operating system. Both AMSDOS and CP/M use the same file structure and may read and write each other's files.

CP/M 2.2 is invoked from BASIC by typing `|CPM`. Part of CP/M (the CCP and BDOS) is loaded from the disc in drive A: The CP/M BIOS resides in the DDI-1 ROM.

AMSDOS is enabled whenever BASIC is used with the DDI-1 interface connected. This intercepts most of the cassette firmware routines and redirects them to disc. Thus existing BASIC programs which use cassette files can use disc files with little or no modification. AMSDOS also provides a number of external commands for erasing and renaming files and redirecting the cassette firmware routines.

Provided with the disc system are a number of utility programs for formatting and copying discs and for changing various system parameters. These all run under CP/M.

## 1.1 CP/M 2.2

CP/M 2.2 is the industry standard operating system for 8080/8085/Z80 based computers. With CP/M 2.2 literally hundreds of application programs are available.

To invoke CP/M from BASIC use the `|CPM` command.

The CPC464 implementation of CP/M offers a TPA of 39.5K, 180K per side disc capacity, the IOBYTE, support for a two channel serial interface and access to the low level device driving routines.

The TPA (Transient Program Area) is the area of RAM into which an application program is loaded and run. This area always starts at #0100. The end of the area is different on different CP/M systems. The address of the first byte after the TPA is held in the word at #0006. On the CPC464 the TPA normally extends to #9F05. See chapter 2.2.

The discs have a capacity of 180K bytes per side. Two tracks are reserved for the CP/M system, and 2K is used for the directory. This leaves 169K per side for files. Two other disc formats are available for specialist use, see chapter 2.11. It is possible to use other non-AMSTRAD disc formats by patching the relevant eXtended disc Parameter Block (XPB) see chapter 2.15.

The IOBYTE is an optional CP/M feature for redirecting character I/O. It is fully implemented on the CPC464. See the CP/M Operating System Manual for details on the use of the IOBYTE.

The BIOS will drive a two channel asynchronous serial interface, if fitted. This interface is not supplied with the DDI-1. See chapter 5.2.

Many of the low-level device driving routines are available to an application program via an extended BIOS jumpblock. Facilities include reading and writing sectors to disc, formatting a track, specifying various system parameters and initializing the serial interface (if fitted). See chapter 2.14.

Existing application programs written for CP/M 2.2 should run on the CPC464 without any difficulty. The only potential problems are:

Programs which use the Z80 alternate or IY registers.

> Provided that the BIOS is set up to save these registers then an application program may use the alternate and IY registers (see chapter 2.17). The system as supplied saves these registers. The utility SETUP is used to enable or disable this facility (see chapter 4.10). Programs which have been designed to run on the Intel 8080 or 8085 will run regardless of this facility.

Programs which use the restart instructions and/or locations.

> CP/M application programs tend not to use the restart instructions precisely because they are often used for interrupts. On the CPC464 only RST6 is available to application programs. RSTs 1..5 are used by the firmware, they may be used for other purposes if special steps are taken, see chapter 2.16. RST7 is reserved for interrupts.

Programs which require a TPA bigger than 39.5K.

> Will require modification before they can be run on the CPC464.

The effect of control codes sent to the screen is as described in the CPC464 Firmware Manual Appendix VII.

The keyboard may be re-configured as required. The utility SETUP is used to define the required keyboard layout, that is which key should return what. The information is stored on disc in the configuration sector and is used to initialize the keyboard during cold boot. The system as supplied does not re-configure the keyboard, the layout is as given in the Firmware Manual Appendices I..IV.

# 1.2 AMSDOS

AMSDOS stands for AMStrad Disc Operating System. AMSDOS has been designed to enable BASIC programs to use disc files in exactly the same manner as cassette files. Whenever the DDI-1 interface is connected all BASIC file operations are redirected to the disc instead of the cassette. AMSDOS uses the same file structure as CP/M so files can be freely interchanged between the two systems.

The following BASIC commands will all work with the disc:

```
LOAD, RUN, SAVE, CHAIN, MERGE, CHAIN MERGE, OPENIN, OPENOUT,
CLOSEIN, CLOSEOUT, CAT, EOF, INPUT #9, LINE INPUT #9, WRITE
#9, LIST #9.
```

The main point to watch out for is that filenames must conform to CP/M conventions.

The cassette can still be used: `|TAPE` switches all file operations back to the cassette, `|DISC` switches back to disc. `|TAPE.IN` switches just the input file operations to tape, `|DISC.IN` switches just the input file operations to disc, similarly `|TAPE.OUT` and `|DISC.OUT` switch just the output operations.

The `CAT` command will display the disc directory. It is sorted into alphabetical order, shows the size of each file, together with the remaining free space on the disc.

Files can be renamed and erased by the external commands `|REN` and `|ERA`. For example to erase a disc file `TEXT.DOC`

```
FILENAME$="TEXT.DOC"
|ERA,@FILENAME$
```

To rename the file `RED.BAS` to `GREEN.BAS`

```
OLDNAME$="RED.BAS"
NEWNAME$="GREEN.BAS"
|REN,@NEWNAME$,@OLDNAME$
```

This somewhat convoluted syntax is used because strings must be passed to external commands by address.

If the drive letter is omitted from a filename then the current default drive is assumed, initially drive `A:`. For two drive systems `|B` will select drive `B:` as the default and `|A` will reselect drive `A:`.

Just part of the directory may be displayed, if required, by using the `|DIR` command. This takes an optional parameter of a filename which may contain wild cards. Only those files which match the filename are displayed. This is very similar to the CP/M `DIR` command. For example, to display just the files ending in `.BAS`

```
FILENAME$="*.BAS"
|DIR,@FILENAME$
```

User numbers are a means of logically dividing the disc into sixteen different 'users' numbered 0..15 each with its own directory. This is a CP/M feature also supported by AMSDOS. The default user is initially set to 0. The default user may be changed by the |USER command. For example the command |USER, 10 will change the default user to 10. Alternatively when specifying a filename the user number may be prefixed to the drive part of the name. For example to rename a file from user 5 to user 11:

```
OLDNAME$="5:OLDNAME"
NEWNAME$="11:NEWNAME"
|REN,@NEWNAME,@OLDNAME
```

The USER command has a counterpart in CP/M, however, it is not possible to specify a user as part of a filename except under AMSDOS. The user number can be incorporated whenever a drive name is used. For example to display the directory of all files in user 7 on drive B: which start with the letter Z and have a type part of .HEX

```
FILENAME$="7B:Z*.HEX"
|DIR,@FILENAME$
```

AMSDOS works by intercepting most of the cassette firmware entries, so a machine code program loaded via BASIC can also use AMSDOS. AMSDOS requires an area of RAM for its own puposes, a machine code program must take care not to overwrite this area while AMSDOS is using it. See chapter 3.9 for more details.

AMSDOS uses the CP/M BIOS to access the discs. Some of the extended BIOS routines are also available to AMSDOS. See chapters 3.8 and 2.14.

# 1.3 Utilities.

The disc system is supplied with a number of utility programs, they all run under CP/M 2.2. Some of these are supplied by Digital Reasearch and are standard CP/M utilities, the others have been specially developed for the CPC464 by AMSTRAD and should not be used on other CP/M systems. Also supplied is the programming language DR LOGO, see the AMSTRAD DDI-1 User Instruction Manual and 'A Guide to LOGO' SOFT 160.

AMSDOS.COM (AMSTRAD)

Returns to AMSDOS and BASIC from CP/M. The inverse operation of |CPM.

ASM.COM (Digital Research)

8080 assembler. Although the processor used in the CPC464 is a Zilog Z80A this assembler may still be used because the Z80 supports all the 8080 opcodes. The converse is not true.

BOOTGEN.COM (AMSTRAD)

Copies the boot and configuration sectors from one disc to another.

CLOAD.COM (AMSTRAD)

Copies a file from cassette to disc.

CSAVE.COM (AMSTRAD)

Copies a file from disc to cassette.

CHKDISC.COM (AMSTRAD)

Compares two discs using two drives.

COPYDISC.COM (AMSTRAD)

Copies one disc to another using two drives.

DDT.COM (Digital Research - patched by AMSTRAD to use RST6)

Dynamic Debugging Tool, 8080 debugger.

DISCCHK.COM (AMSTRAD)

Compares two discs using one drive.

DISCCOPY.COM (AMSTRAD)

Copies one disc to another using one drive.

DUMP.COM (Digital Reasearch)

Displays a file on the screen in hex.

ED.COM (Digital Research)

Text editor.

FILECOPY.COM (AMSTRAD)

Copies files from one disc to another using one drive.

FORMAT.COM (AMSTRAD)

Formats a disc.

LOAD.COM (Digital Research)

Reads a file in Intel HEX format and produces a .COM file.

MOVCPM.COM (Digital Research - patched by AMSTRAD for page boundaries)

Constructs a CP/M system of any given size. Used to make room for RSXs.

PIP.COM (Digital Research)

Peripheral Interchange Program, copies files on disc and between the other peripherals.

SETUP.COM (AMSTRAD)

Changes the parameters in the configuration sector.

STAT.COM (Digital Research)

Gives details on files, discs, users and the IOBYTE. Can also change the IOBYTE.

**SUBMIT.COM** (Digital Research)

Takes CP/M commands from a file instead of from the keyboard.

**SYSGEN.COM** (AMSTRAD)

Writes the CP/M system onto the system tracks.

**XSUB.COM** (Digital Research)

Used with **SUBMIT.COM** to provide buffered input for programs.

# Chapter 2 - CP/M 2.2 and the BIOS

This chapter describes the CP/M 2.2 implementation on the CPC464, with particular reference on how to use the extensive facilities available in the firmware jumpblock and in the BIOS extended jumpblock. This is not a tutorial, the reader should have some knowledge of CP/M and its structure (see 'A Guide to CP/M' SOFT 159).

## 2.1 Structure of CP/M

CP/M consists of three software modules: Console Command Processor (CCP), Basic Disc Operating System (BDOS) and the Basic Input Output System (BIOS). (Note the term 'Basic' in this context has nothing to do with the BASIC language.) The CCP and BDOS are the machine independent parts of CP/M as supplied by Digital Reserach. The BIOS is the machine dependent part which deals with all the low-level device driving, in particular the disc driving. The BIOS resides in the expansion ROM in the DDI-1 interface. The CCP and BDOS reside on the first two tracks of a system format disc and are loaded into RAM by the BIOS when a 'Warm boot' is performed.

The BIOS ROM starts at #C000 and ends at #FFFF, that is it overlays the screen RAM. The ROM also contains AMSDOS and some parts of DR LOGO.

The RAM from #AD33 to #BFFF is always reserved. It contains the firmware jumpblock and the firmware and BIOS data areas.

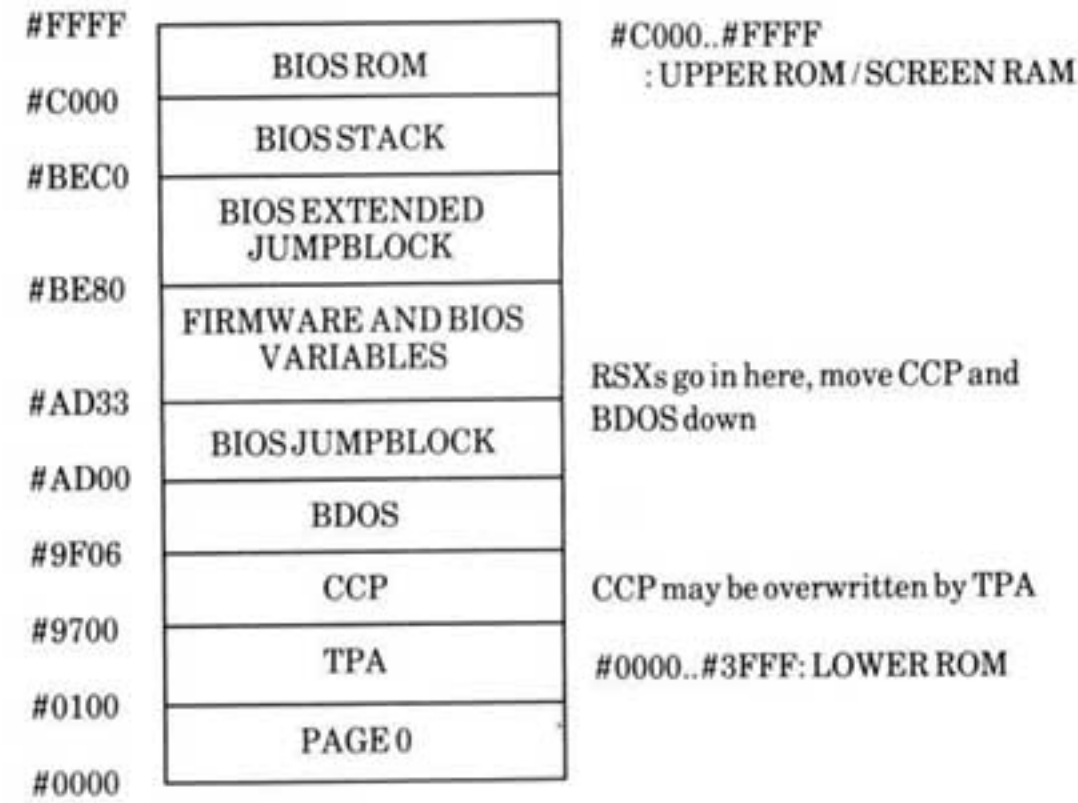
The rest of store is for use by CP/M and Resident System eXtensions (RSXs). In the system as supplied there is no room for any RSXs, however, the utility `MOVCPM.COM` can be used to create a smaller CP/M system with space for RSXs as required.

The BDOS communicates with the BIOS via the BIOS jumpblock. This is positioned immediately after the BDOS.

The 16 bytes from #0040 to #004F are reserved for use by DR LOGO, they are not used by the BIOS.

## 2.2 Store Map

This store map has CP/M positioned as high as possible.




## 2.3 Cold Boot

The term 'cold boot' means invoking CP/M for the first time after the machine has been switched on or reset.

CP/M is invoked in one of two ways: either automatically when the machine is switched on, or by issuing the |CPM external command.

The method used is determined by an option link on the disc interface board. This sets the address of the BIOS ROM to either 0 or 7. If the ROM address is set to 0 then the BIOS cold boot will be invoked automatically when the machine is switched on. If the ROM address is set to 7 then CP/M may be invoked by issuing the external command `|CPM`. The DDI-1 is supplied with the ROM number set to 7. See chapter 5.1 for details on how to change the ROM number.

Once the BIOS cold boot is invoked the firmware is reset by calling MC START PROGRAM, thus any RAM allocated to background ROMs and RSXs will be lost and all ticker chains reset.

The BIOS cold boot routine first initializes the BIOS into a 'minimum' working state, loads a boot program from the single boot-sector on the disc into store, starting at #0100, and enters it. If all the bytes in the boot sector have the same value an error message is displayed. Otherwise it is assumed that the sector contains a valid boot program.

The 'minimum' working state in which the boot program runs is sufficient to run CP/M but does not do any of the following: display a sign-on message, initialize the serial interface, set any keyboard translations.

The standard boot program loads the configuration sector from the disc and performs some further initialization. This 'two stage' initialization is done so that, if required, the boot program could bring up a totally different operating system. See chapter 2.12.

Once the BIOS is initialized a warm boot is performed to load the CCP and BDOS.

## 2.4 Warm Boot

The term 'warm boot' means reloading CP/M after a transient program has run or after the user has typed Control-C in order to change a disc.

A warm boot is invoked immediately after initialization and thereafter by a jump to location zero, or by calling the BDOS System Reset function. The primary function of the warm boot is to load the BDOS and CCP into memory and then to enter the CCP. The memory location used to start the load is calculated from the jump instruction at the start of the CCP. #035C is subtracted from the JMPs destination address giving the origin of the CCP. Since the position of the CCP is recalculated at every warm boot differing sized CP/Ms can be warm booted from each other.

If all the bytes in the first sector of the CCP have the same value an error message is displayed. Otherwise it is assumed that the system tracks contain a valid CCP and BDOS.

The warm boot routine initializes the jumps at #0000 and #0005 and also copies the BIOS jumpblock into RAM immediately after the BDOS (see chapter 2.2).

## 2.5 Resident System eXtensions

Resident System eXtensions (RSXs) allow the user to keep programs, such as special character I/O drivers, permanently in memory. They must be loaded into memory below the BIOS variable area (which starts at location #AD33) but, above the end of the RAM copy of the BIOS's jumpblock. In the CP/M system as supplied there is no space for RSXs. If RSXs are required the user must reduce the size of the TPA by the amount of space required for all RSXs. Note that the `MOVCPM.COM` transient utility only allows the user to relocate CP/M onto 256 byte (page) boundaries.

Let Srsx be the size of RSX required in pages (1 page = 256 bytes) then the RSX area will start at address #AD33 - (256 * Srsx) and end at #AD32.

The BIOS does not initialize any background ROMs. AMSDOS must not be initialized under CP/M.

Under version 1 of the firmware ROM the following restrictions apply:

> Any background ROMs required must be explicitly initialized by calling KL INIT BAK. Using KL ROM WALK would cause the disc expansion ROM to be re-initialized which would lock up the system. The memory pool for the ROM's dynamic variables must lie in the RSX area, that is between the end of the BIOS jumpblock and below the BIOS's variables.
>
> If the ROM address is set to `Ø` and BASIC is invoked AMSDOS will not be available.

# 2.6 Using MOVCPM.COM

In order to insert an RSX into store it is necessary to move the CP/M system down store. The CP/M utility `MOVCPM.COM` will create an in-store image of a CP/M system of the required size. The utility `SYSGEN.COM` is then used to write this image onto disc, see section 4.11.

The `MOVCPM` utility requires the size of CP/M system to be specified. Historically this size meant the total amount of RAM available to the original CP/M system with BIOS size of 1.5K. Nowadays it is merely a magic number to feed `MOVCPM` with! The `MOVCPM` utility supplied requires this size to be expressed in pages (1 page = 256 bytes). It is calculated as follows:

> Let Scpm be the 'size of CP/M' in pages as required by the `MOVCPM` utility
> Let Srsx be the size in pages of RSX required.
>
> Then Scpm = 179 - Srsx.
>
> The BDOS will start at address Bstart = (Scpm - 20) * 256
>
> The BDOS entry point will be at address Bentry = Bstart + 6
>
> The TPA size = Bentry - #0100
>
> To run `MOVCPM` type:
>
> ```
> MOVCPM nnn *
> ```
>
> where nnn is the size required and must be in the range 64..179.
> The asterix must be present and separated from the size by a space.
>
> The CP/M system supplied has been created from:
>
> ```
> MOVCPM 179 *
> ```

# 2.7 IOBYTE

CP/M supports four logical character I/O devices called CONSOLE, READER, PUNCH and LIST. These four logical devices are mapped onto different physical devices. At any one time the current mapping of logical to physical devices is defined in a location called the IOBYTE. The STAT transient utility program can be used to change the IOBYTE.

CONSOLE may be assigned to:

| | |
|---|---|
| TTY: | (special I/O device 0) |
| CRT: | (keyboard and screen) |
| BAT: | (input from READER and output to LIST) |
| UC1: | (special I/O device 1) |

READER may be assigned to:

| | |
|---|---|
| TTY: | (special I/O device 0) |
| PTR: | (end of file input) |
| UR1: | (special I/O device 1) |
| UR2: | (keyboard) |

PUNCH may be assigned to:

| | |
|---|---|
| TTY: | (special I/O device 0) |
| PTP: | (null output) |
| UP1: | (special I/O device 1) |
| UP2: | (screen) |

LIST may be assigned to:

| | |
|---|---|
| TTY: | (special I/O device 0) |
| CRT: | (screen) |
| LPT: | (Centronics port) |
| UL1: | (special I/O device 1) |

The Keyboard input device uses the firmware routine KM WAIT CHAR to read characters from the keyboard.

The screen output device uses the firmware routine TXT OUTPUT to display text and process control characters. Refer to the firmware documentation for details of the control characters supported.

The end of file input device always returns the end of file character #1A. The null output device discards all characters sent to it.

The two special I/O devices are initially setup to drive channels A and B of a Z80 SIO. Each device has four driver routines which are all indirected via the extended jumpblock (see chapter 2.14). The user may patch these jumps in order to use an RSX device driver.

# 2.8 BIOS User Interaction

In general the user may freely type-ahead on the keyboard, provided the BIOS is not accessing the discs. While accessing the discs interrupts are disabled and characters can be lost.

Some characters are treated specially by the BIOS (see also chapter 2.14 on retries). When performing character I/O with the Centronics or serial interface ports the BIOS polls the keyboard for the following control characters (any other characters are lost).

**CONTROL-C**

To stop the system from locking-up the user may press Control-C to perform a warm boot under the following conditions:-

a) Whilst waiting to output a character to the serial interface.
b) Whilst waiting to input a character from the serial interface.
c) Whilst waiting to send a character to the Centronics printer port.

**CONTROL-S**

To suspend CP/M when sending or receiving characters from the serial interface the user may press Control-S. Once suspended the user must press any key to continue operation of the BIOS. The control-S and the character typed are both lost.

**CONTROL-Z**

Whilst waiting to input a character from the serial interface, the user may press Control-Z. This will result in a Control-Z (eof) being passed back to the calling program.

# 2.9 Initial Command Buffer

When CP/M is invoked, the keyboard input device reads characters from a special buffer called the Initial Command Buffer. The `SETUP` utility program allows the user to initialize the initial command buffer, specifying a number of commands to be automatically executed when CP/M is cold booted.

The BIOS routine CONST does not reflect the status of the initial command buffer. Thus programs which poll for Control-C will not drain the initial command buffer. In particular the BIOS user interaction, chapter 2.8, will not fetch characters from the initial command buffer.

# 2.10 BIOS messages

All BIOS messages are sent to the screen via TXT OUTPUT. They are not indirected via the IOBYTE.

BIOS messages are followed by the question `Retry, Ignore or Cancel?`. The system then discards any outstanding characters, turns on the cursor and waits for the user to type `R`, `I` or `C`, anything else will cause a bleep.

Typing R for retry causes the BIOS to repeat the operation.

Typing I for ignore causes the BIOS to continue as if the problem had not occured.

Typing C for cancel causes the BIOS to abandon the operation. This will often result in a BDOS error message.

After the user has typed R, I or C the cursor is turned off.

The AMSTRAD BIOS messages are as follows:-

`Drive ‹n›: disc missing`

Where ‹n› is the drive identifier, either A or B. This message is produced when the BIOS attempts to access a drive that does not have a disc inserted or a non-existent drive.

`Failed to load boot sector`

This message is produced during a cold boot when the boot sector is not read correctly or if all the bytes in the boot sector have the same value.

`Failed to load CP/M`

This message is produced during a warm boot when a sector of the CCP or BDOS is not read correctly or if all the bytes in the first CCP sector have the same value.

`Drive ‹n›: disc is write protected`

Where ‹n› is the drive identifier, either A or B. This message is produced when the BIOS attempts to write to a disc that is write-protected. If the user wishes to write on this disc then the user should remove the disc, write enable it, re-insert it into the drive and then type R for retry.

`Drive ‹n›: read fail`

Where ‹n› is the drive identifier, either A or B. This message is produced when a hardware error has been reported whilst reading from the disc. It may also be caused by trying to read from a disc with the wrong format, for example: trying to warm boot from a DATA ONLY format disc.

`Drive ‹n›: write fail`

Where ‹n› is the drive identifier, either A or B. This message is produced when a hardware error has been reported whilst writing to the disc.

In the event of a read or write fail the user is recommended to remove and re-insert the disc then type 'R'. This may help in case the disc was badly positioned or may shift any fluff or what-not adhering to the head. If this does not clear the problem then the data will have to be recovered in some other way. It is strongly recommended that the user takes regular back-up copies of discs.

# 2.11 Disc Organization

The BIOS supports three different disc formats: SYSTEM format, DATA ONLY format and 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.

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 may 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 tracks 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, since CP/M can only be loaded (cold and warm boot) from a system format disc. The reserved tracks are used as follows:

Track 0 sector #41: boot sector.

Track 0 sector #42: configuration sector.

Track 0 sectors #43..#47: unused.

Track 0 sectors #48..#49 and track 1 sectors #41..#49: CCP and BDOS.

Note: '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..#C9.

0 reserved tracks.

2 to 1 sector interleave.

This format is intended for future enhancement, it is not recommended for use with CP/M since it is not possible to 'warm boot' from it. However, if only AMSDOS is to be used then there is a little more disc space available.

### IBM Format

8 sectors per track numbered 1..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.

## 2.12 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 its own way, using the BIOS and firmware routines as desired.

The boot sector is sector #41 on track 0.

During a cold boot the BIOS is initialized into a minimum state, i.e:

All the routines in the ROM copy of the BIOS jumpblock and all 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 BIOS RAM jumpblock is 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 #AD33, a value immediately below the BIOS's data area and the boot program is entered at #0100. The boot program may use store from #0100 to #AD32 inclusive.

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

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:

program should jump to the WBOOT entry in the above jumpblock

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

## 2.13 Configuration Sector.

The Boot program supplied loads the configuration sector from disc. This contains various system parameters which are used to initialize the BIOS. These parameters may be changed by using the utility `SETUP.COM` (see chapter 4.10).

The configuration sector is sector #42 on track 0.

Format:

byte 0 = #35, byte 1 = #12 Disc signature.

This signature allows the boot program to check that the configuration sector has been `SETUP`.

bytes 2..3 Drive motor on delay

Time to wait between turning the drive motor on and starting a read/write operation. In units of 1/50 sec. A value of zero will lock up the system.

bytes 4..5 Drive motor off delay

Time to wait after a read/write operation before turning the drive motor off. In units of 1/50 sec. A value of zero will keep the motor on permanently.

byte 6 Step rate.

Step rate for moving tracks. Must be an even number in the range 2..32 milliseconds.

byte 7 IOBYTE.

Initial value of the CP/M IOBYTE.

byte 8 BIOS disc message enable/disable.

#00 ◊ enable BIOS disc messages.
#FF ◊ disable BIOS disc messages.

byte 9 Saving alternate and IY register enable/disable.

#00 ◊ enable saving alternate and IY registers (slow mode).
#FF ◊ disable saving alternate and IY registers (fast mode).

Bytes 10..21 are used to initialize the serial interface, if fitted.

byte 10 : SIO channel A write register 4 (mode register).

byte 11 : SIO channel A write register 5 (Tx register).

byte 12 : SIO channel A write register 3 (Rx register).

byte 13 : SIO channel B write register 4 (mode register).

byte 14 : SIO channel B write register 5 (Tx register).

byte 15 : SIO channel B write register 3 (Rx register).

bytes 16..17 : 8253 timer 0 (SIO channel A Tx baud rate).

bytes 18..19 : 8253 timer 1 (SIO channel A Rx baud rate).

bytes 20..21 : 8253 timer 2 (SIO channel B Tx and Rx baud rate).

byte 22 Keep command buffer.

#00 ◊ the initial command buffer will be flushed when a key is pressed.
#FF ◊ the initial command buffer will not be flushed when a key is pressed.

bytes 23..99 reserved.

These bytes are currently unused. All set to #00.

The following fields are all of variable length and live in bytes 100....

Sign-on string.

String of characters that is output to the screen. The string is terminated by a dollar ($).

Printer power-up string.

This string of bytes is sent to the CP/M list device, that is it is indirected via the IOBYTE as set from byte 7.

byte 0 length of string, 0..255.
byte 1... string.

Keyboard NORMAL translations.

This table redefines the codes produced by keys with neither SHIFT nor CONTROL pressed. Each entry in the table is two bytes long. The first byte is the key code, the second the key number that will produce that code. See Appendix I for the key numbering.

Table format:

byte 0 number of entries, 0..255.
bytes 1... entries.

Entry format:

byte 0 key code.
byte 1 key number.

Keyboard SHIFT translations.

This table redefines the codes produced by keys with SHIFT pressed. Each entry in the table is two bytes long. The first byte is the key code, the second the key number that will produce that code. See Appendix I for the key numbering.

Table format:

byte 0 number of entries, 0..255.
bytes 1... entries.

Entry format:

byte 0 key code.
byte 1 key number.

Keyboard CONTROL translations.

This table redefines the codes produced by keys with CONTROL pressed. Each entry in the table is two bytes long. The first byte is the key code, the second the key number that will produce that code. See Appendix I for the key numbering.

Table format:

byte 0 number of entries, 0..255.
bytes 1... entries.

Entry format:

byte 0 key code.
byte 1 key number.

Keyboard expansions.

This table defines the keyboard expansion strings (see section 3.7).

Table format:

byte 0 number of entries, 0..255.
bytes 1... entries.

Entry format:

| | |
|---|---|
| byte 0 | expansion token. |
| byte 1 | length of expansion string. |
| byte 2... | expansion string. |

The remainder of the configuration sector is set to #00.

# 2.14 BIOS Jumpblocks

The AMSTRAD BIOS has two jumpblocks: the standard CP/M jumpblock and an extended jumpblock. The extended jumpblock starts at #BE80 and contains jumps for additional facilities such as physical sector reading and writing. When CP/M is warm-booted the standard BIOS jumpblock is copied into RAM immediately after the BDOS. The address of the warm boot entry in the RAM jumpblock is held in the word at #0001.

The entry and exit conditions of the standard BIOS jumpblock are a superset of those defined in the CP/M alteration guide as listed below. All routines in the standard BIOS jumpblock optionally preserve the Z80 alternate and IY registers as specified by the SET REG SAVE routine.

READ
WRITE

These two routines return some hardware error status in the accumulator when a read or write has failed. The definitions of the status bits are listed below.

| | | |
|---|---|---|
| bit 7 | end of cylinder | - this bit should always be set |
| bit 6 | not used | - always 1 |
| bit 5 | data error | - CRC error on data or ID field |
| bit 4 | overrun error | |
| bit 3 | drive not ready | - no disc in drive |
| bit 2 | no data | - can't find the sector |
| bit 1 | not writable | - drive write protected |
| bit 0 | address mark missing | |

In addition to the above status byte, the HL register pair contains the address of a buffer of all the bytes received during the results phase of the $\mu$PD765A disc controller, the first byte is a count of the number of bytes that follow. (See manufacturer's documentation.)

SELDSK

If bit 0 of E = 0, indicating that this is the first time that the disc is being accessed, then the BIOS will attempt to determine the format of the disc by reading an ID header from the current track. If successful then the relevant extended DPB is altered to correspond to the detected format.

The extended jumpblock starts at #BE80 and has the following format:

```
#BE80   JMP   SET MESSAGE       ;enable/disable disc error messages
#BE83   JMP   SETUP DISC        ;initialize drive parameters
#BE86   JMP   SELECT FORMAT     ;select one of the standard formats
#BE89   JMP   READ SECTOR       ;read a sector from disc
#BE8C   JMP   WRITE SECTOR      ;write a sector to disc
#BE8F   JMP   FORMAT TRACK      ;format an entire track
#BE92   JMP   MOVE TRACK        ;move to specified track
#BE95   JMP   GET DR STATUS     ;get drive status (μPD765A reg 3)
#BE98   JMP   SET RETRY COUNT   ;set number of error retries
#BE9B   JMP   ENTER FIRMWARE    ;invoke a firmware routine
#BE9E   JMP   SET REG SAVE      ;set/clear alternate register saving
#BEA1   JMP   SET SIO           ;initialize the serial interface
#BEA4   JMP   SET CMND BUFFER   ;initialize the command buffer
```

The following 8 routines are the indirection jumps for the special character I/O devices. The BIOS uses these jumps to access the devices. Thus the user may patch these jumps to an RSX driver if required. They all default to the serial interface driver. Device 0 is channel A of the serial interface, device 1 is channel B.

```
#BEA7   JMP   D0 IN STATUS      ;input status of device 0
#BEAA   JMP   D0 IN             ;input from device 0
#BEAD   JMP   D0 OUT STATUS     ;output status of device 0
#BEB0   JMP   D0 OUT            ;output to device 0
#BEB3   JMP   D1 IN STATUS      ;input status of device 1
#BEB6   JMP   D1 IN             ;input from device 1
#BEB9   JMP   D1 OUT STATUS     ;output status of device 1
#BEBC   JMP   D1 OUT            ;output to device 1
```

The action performed by each entry in the extended jumpblock is detailed below, along with the entry and exit conditions of each.

These routines will, in turn, call firmware routines so they must themselves be treated like firmware routines, see chapter 2.17.

# 0 SET MESSAGE #BE80

Enable/disable the disc error messages.

## Action:

When disc error messages are enabled and an error occurs the BIOS will display error messages on the screen and interact with the user. When disabled no messages are displayed.

## Entry conditions:

A = #00◊ enable disc error messages
A = #FF◊ disable disc error messages

## Exit conditions:

A = previous state

All flags HL corrupt
All other registers preserved.

## Notes:

The power-up state is ENABLED.

## Related entries:

**SET RETRY COUNT**

# 1 SETUP DISC #BE83

Reset various disc parameters.

## Action:

Sets the values for the motor on, motor off, write current off and head settle times. Sends a SPECIFY command to the floppy disc controller.

## Entry conditions:

HL = address of parameter block

Format of the parameter block:

| | |
|---|---|
| bytes 0, 1 | motor on timeout in 20 millisecond units. |
| bytes 2, 3 | motor off timeout in 20 millisecond units. |
| byte 4 | write current off time in 10 microsecond units. |
| byte 5 | head settle time in 1 millisecond units. |
| byte 6 | step rate time in 1 millisecond units. |
| byte 7 | head unload delay (as per $\mu$PD765A SPECIFY command). |
| byte 8 | bits 7..1: head load delay, bit 0: non-DMA mode (as per $\mu$PD765A SPECIFY command). |

## Exit conditions:

AF BC DE HL corrupt
All other registers preserved.

## Notes:

The values given are used for both drives. When using two differing drives use the slowest of the two times.

The power-up values are:

| | |
|---|---|
| motor on timeout | 50 |
| motor off timeout | 250 |
| write current off time | 175 |
| head settle time | 15 |
| step rate time | 12 |
| head load time | 1 |
| head unload time | 1 |
| non-DMA mode | 1 |

A motor on time of zero will lock up the system. A motor off time of zero will never turn the motor off.

The standard boot program calls this routine to reset some of the disc parameters as specified in the configuration sector, that is, motor on and off timeouts and the step rate.

# 2 SELECT FORMAT #BE86

Select an AMSTRAD format, regardless of the actual disc format.

## Action:

This routine initializes the extended disc parameter block for the given format. Normally the BIOS automatically detects the format of a disc when SELDSK is called by looking at the sector numbers, but for programs such as formatters it is necessary to pre-set the format.

## Entry conditions:

A = first sector number of required format

#41 ◊ system format
#C1 ◊ data only format
#01 ◊ IBM format

E = drive

#00 ◊ A:
#01 ◊ B:

## Exit conditions:

AF BC DE HL corrupt
All other registers preserved.

## Notes:

Bytes 0..21 of the extended disc parameter block are completely reset, all previous values are lost. Bytes 22..24 (track, align flag, auto-select flag) are not affected. See chapter 2.15.

To set a non-AMSTRAD format the user may patch the extended disc parameter block directly.

# 3 READ SECTOR #BE89

Read a physical sector from disc.

## Action:

Read the specified sector into store.

## Entry conditions:

HL = address of sector buffer
E = drive number
    #00 ◊ A:
    #01 ◊ B:
D = track number
C = sector number

## Exit conditions:

if carry true

    sector read OK
    A = 0
    HL preserved

if carry false

    failed to read sector correctly
    A = error status byte as defined above
    HL = address of error status buffer

always

    other flags corrupt
    all other registers preserved

## Related entries:

**WRITE SECTOR**

# 4 WRITE SECTOR #BE8C

Write a physical sector to disc.

## Action:

Write the required sector from store.

## Entry conditions:

HL = address of sector buffer
E = drive number
    #00 ◊ A:
    #01 ◊ B:
D = track number
C = sector number

## Exit conditions:

if carry true

    sector written OK
    A = 0
    HL preserved

if carry false

    failed to write sector correctly
    A = error status byte as defined above
    HL = address of error status buffer

always

    other flags corrupt
    all other registers preserved

## Notes:

Sector buffer may be under ROM.

## Related entries:

**READ SECTOR**

# 5 FORMAT TRACK #BE8F

Format an entire track.

## Action:

Format a track.

## Entry conditions:

HL = address of header information buffer
E = drive number
#00 ◊ A:
#01 ◊ B:
D = track number

Format of header information:

sector entry for first sector
sector entry for second sector
...
sector entry for last sector

sector entry format:

byte 0 : track number
byte 1 : head number
byte 2 : sector number
byte 3 : $\log_2$ (sector size) - 7

## Exit conditions:

if carry true

track formatted OK
A = 0
HL preserved

if carry false

failed to format track correctly
A = error status byte as defined above
HL = address of error status buffer

always

other flags corrupt
all other registers preserved

## Notes:

The extended DPB must be preset for the required format (see SELECT FORMAT).

## Related entries:

**SELECT FORMAT**

# 6 MOVE TRACK #BE92

Unverified move to specified track.

## Action:

Move head to specified track.

## Entry conditions:

E = drive number
    #00 ◊ A:
    #01 ◊ B:
D = track number

## Exit conditions:

if carry true

    moved to track OK
    A = 0
    HL preserved

if carry false

    failed to move track correctly
    A = error status byte as defined above
    HL = address of error status buffer

always

    other flags corrupt
    all other registers preserved

## Notes:

This routine is intended as a diagnostic aid and need not normally be used because the read/write/format routines all seek to the correct track automatically.

# 7 GET DR STATUS #BE95

Return status for specified drive.

## Action:

This routine returns status register 3 of the floppy disc controller as defined below for the specified drive.

| | | |
|---|---|---|
| bit 7 | undefined | |
| bit 6 | write protect | - The write protect line is true. |
| bit 5 | drive ready | - The ready line is true. |
| bit 4 | track zero | - The track zero line is true. |
| bit 3 | undefined | |
| bit 2 | head address | - always zero. |
| bit 1 | unit select 1 | - unit select 1, always zero. |
| bit 0 | unit select 0 | - currently selected drive. |

## Entry conditions:

A = drive number

#00 ◊ A:
#01 ◊ B:

## Exit conditions:

if carry true

A = Drive status byte as defined above
HL preserved

If carry false

HL = address of error status buffer, second byte = Drive status byte as defined above
A corrupt

always

other flags corrupt
all other registers preserved

## Notes

This routine returns carry to indicate which set of exit conditions have occurred. No other meaning should be attached to the state of carry.

# 8 SET RETRY COUNT #BE98

Set the number of retries for reading/writing/formatting.

## Action:

Sets the number of times an operation is retried in the event of an error.

## Entry conditions:

A = new value for retry count

## Exit conditions:

A = old value of retry count
All flags HL corrupt

## Notes:

The pattern of retries is as follows. Each 'Try' counts one. The retry pattern is repeated until either the operation succeeds or the number of tries has reached the retry count:

Try
Try
Move in one track and back again
Try
Move out one track and back again
Try
Move to inner track and back again
Try
Try
Move in one track and back again
Try
Move out one track and back again
Try
Move to outer track and back again
Repeat

The default value is 16, i.e. twice around the above loop.

# 9 ENTER FIRMWARE #BE9B

Call a firmware routine.

## Action:

Calls a firmware routine. The application program must set up all registers as required by the firmware routine to be called, its address being passed as an in-line parameter.

The given routine is run on the BIOS's stack.

The following routines must NEVER be called using ENTER FIRMWARE:

ENTER FIRMWARE.
SET REG SAVE.
All the standard BIOS routines.

See chapter 2.17 for when and why to use this routine.

## Entry conditions:

AF BC DE HL IX as required by routine to call
Return address points to address of routine to call

## Exit conditions:

AF BC DE HL IX as per routine called
all other registers preserved

## Related entries:

**SET REG SAVE**

Example:

To read characters from the keyboard using KM WAIT KEY echoing them to the screen using TXT OUTPUT.

```
LOOP:   CALL    ENTERFIRMWARE
        DW      KMWAITKEY           ;in-line parameter

        CALL    ENTERFIRMWARE
        DW      TXTOUTPUT           ;in-line parameter

        JP      LOOP
```

# 10 SET REG SAVE #BE9E

Set/clear saving of the alternate and IY registers.

## Action:

Enable or disable the saving of the alternate and IY registers. See chapter 2.17 and the ENTER FIRMWARE routine.

This routine must NEVER be called by using the ENTER FIRMWARE routine.

## Entry conditions:

A = #00 ◊ save alternate and IY registers
A = #FF ◊ don't save alternate and IY registers

## Exit conditions:

A = previous value
All flags HL corrupt
all other registers preserved

## Related entries:

ENTER FIRMWARE

# 11 SET SIO #BEA1

Reset and initialize the serial interface.

## Action:

Reset and initialize both channels of the SIO. Initialize the 8253 used as a baud rate generator.

## Entry conditions:

HL = address of parameter block

parameter block format:

| | |
|---|---|
| byte 0 | SIO channel A write register 4 (mode register) |
| byte 1 | SIO channel A write register 5 (tx register) |
| byte 2 | SIO channel A write register 3 (rx register) |
| byte 3 | SIO channel B write register 4 (mode register) |
| byte 4 | SIO channel B write register 5 (tx register) |
| byte 5 | SIO channel B write register 3 (rx register) |
| byte 6, 7 | 8253 timer 0 (SIO channel A tx baud rate) |
| byte 8, 9 | 8253 timer 1 (SIO channel A rx baud rate) |
| byte 10, 11 | 8253 timer 2 (SIO channel B tx and rx baud rate) |

## Exit conditions:

AF BC DE HL corrupt
all other registers preserved

## Notes:

The SIO and 8253 are not included on the DDI-1 interface.

Refer to the Zilog SIO and Intel 8253 documentation.

# 12 SET CMND BUFFER #BEA4

Initialize the initial command buffer.

## Action:

The buffer supplied is copied into the BIOS's initial command buffer. When the BIOS routine CONIN is called it fetches characters from the command buffer until it is empty, then characters are fetched from the console.

Optionally, while characters are being read from the command buffer, hitting a key will empty the command buffer. The character typed will be the next one returned. All subsequent characters will be fetched from the keyboard.

## Entry conditions

A = #00 ◊ the command buffer will be junked if a key is pressed
A = #FF ◊ the command buffer is kept even if a key is pressed
HL = address of buffer.

Buffer format:

byte 0 count of characters 0..128
byte 1..128 characters

## Exit conditions:

AF BC DE HL corrupt
all other registers preserved

## Notes:

Although this routine is primarily for use during cold boot, it may be called at any time to refill the buffer.

CONST returns the status of the keyboard, not of the initial command buffer. Thus a program which polls for Control-C will not drain the command buffer.

# 13 D0 IN STATUS #BEA7

Test if special character I/O device 0 has a character available.

## Action:

Test if there is a character available.

## Entry conditions:

No conditions

## Exit conditions:

if A = #FF

a character is available for input

if A = #00

no character is available for input

always

All flags BC DE HL corrupt

## Notes:

The default device 0 is channel A of the serial interface.

This jump may be patched to a RSX driver if required.

## Related entries:

**D0 IN**
**D0 OUT STATUS**
**D0 OUT**

# 14 D0 IN #BEAA

Fetch a character.

## Action:

Input a character from special character I/O device 0. If none is available then wait until one appears.

## Entry conditions:

No conditions

## Exit conditions:

A = character input
All flags BC DE HL corrupt
all other registers preserved

## Notes:

The default device 0 is channel A of the serial interface.

This jump may be patched to a RSX driver if required.

## Related entries:

**D0 IN STATUS**
**D0 OUT STATUS**
**D0 OUT**

# 15 D0 OUT STATUS #BEAD

Test if the output is ready.

## Action:

Test if special character I/O device 0 is ready to output a character.

## Entry conditions

No conditions

## Exit conditions:

if A = #FF

output device is ready

if A = #00

output device is busy

always

All flags BC DE HL corrupt
all other registers preserved

## Notes:

The default device 0 is channel A of the serial interface.

This jump may be patched to a RSX driver if required.

## Related entries:

**D0 IN STATUS**
**D0 IN**
**D0 OUT**

# 16 D0 OUT #BEB0

Output a character.

## Action:

Output a character to special character I/O device 0. If device is busy then wait until it becomes ready.

## Entry conditions:

C = character to output

## Exit conditions:

AF BC DE HL corrupt
all other registers preserved

## Notes:

The default device 0 is channel A of the serial interface.

This jump may be patched to a RSX driver if required.

## Related entries:

D0 IN STATUS
D0 IN
D0 OUT STATUS

# 17 D1 IN STATUS #BEB3

Test if special character I/O device 1 has a character available.

## Action:

Test if there is a character available.

## Entry conditions:

No conditions

## Exit conditions:

if A = #FF

a character is available for input

if A = #00

no character is available for input

always

All flags BC DE HL corrupt

## Notes:

The default device 1 is channel B of the serial interface.

This jump may be patched to a RSX driver if required.

## Related entries:

D1 IN
D1 OUT STATUS
D1 OUT

# 18 D1 IN #BEB6

Fetch a character.

## Action:

Input a character from special character I/O device 1. If none is available then wait until one appears.

## Entry conditions:

No conditions

## Exit conditions:

A = character input
All flags BC DE HL corrupt
all other registers preserved

## Notes:

The default device 1 is channel B of the serial interface.

This jump may be patched to a RSX driver if required.

## Related entries:

D1 IN STATUS
D1 OUT STATUS
D1 OUT

# 19 D1 OUT STATUS #BEB9

Test if the output is ready.

## Action:

Test if special character I/O device 1 is ready to output a character.

## Entry conditions:

No conditions

## Exit conditions:

if A = #FF

output device is ready

if A = #00

output device is busy

always

All flags BC DE HL corrupt
all other registers preserved

## Notes:

The default device 1 is channel B of the serial interface.

This jump may be patched to a RSX driver if required.

## Related entries:

D1 IN STATUS
D1 IN
D1 OUT

# 20 D1 OUT #BEBC

Output a character.

## Action:

Output a character to special character I/O device 1. If device is busy then wait until it becomes ready.

## Entry conditions:

C = character to output

## Exit conditions:

AF BC DE HL corrupt
all other registers preserved

## Notes:

The default device 1 is channel B of the serial interface.

This jump may be patched to a RSX driver if required.

## Related entries:

**D1 IN STATUS**
**D1 IN**
**D1 OUT STATUS**

# 2.15 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 DPB. |
| byte 15 | : first sector number. |
| byte 16 | : number of sectors per track. |
| byte 17 | : gap length (read/write). |
| byte 18 | : gap length (format). |
| byte 19 | : filler byte for formatting. |
| byte 20 | : $\log_2$ (sector size) - 7, 'N' for $\mu$PD765A. |
| byte 21 | : sector size / 128 (must be a power of 2). |
| byte 22 | : reserved: current track (set by BIOS). |
| byte 23 | : reserved: #00 ◊ not aligned, #FF ◊ aligned (set by BIOS). |
| byte 24 | : #00 ◊ auto-select format, #FF ◊ don't auto-select format. |

To find the XPB for a particular drive:

Either call the BIOS SELDSK routine. This returns the address of the DPH. The word at offset 10 in the DPH is the address of the DPB.

Or, the word at location #BE40 contains the address of the DPH vector. The first DPH is for drive A, the second for drive B.

Or use BDOS function 31.

When calling SELDSK if bit 0 of E = 0 and byte 24 of the XPB = #00 then the BIOS will attempt to determine the format of the disc and patch the XPB accordingly.

The XPBs for the different formats are initialized as follows:

**System Format**

| | |
|---|---|
| 36 | SPT, records per track |
| 3 | BSH, block shift |
| 7 | BLM, block mask |
| 0 | EXM, extent mask |
| 170 | DSM, number of blocks - 1 |
| 63 | DRM, number of directory entries - 1 |
| #C0 | AL0, 2 directory blocks |
| #00 | AL1 |
| 16 | CKS, size of checksum vector |
| 2 | OFF, reserved tracks |
| | |
| #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 sector |
| 0 | current track |
| 0 | not aligned |
| 0 | do auto select format |

**Data Only Format**

| | |
|---|---|
| 36 | SPT, records per track |
| 3 | BSH, block shift |
| 7 | BLM, block mask |
| 0 | EXM, extent mask |
| 179 | DSM, number of blocks - 1 |
| 63 | DRM, number of directory entries - 1 |
| #C0 | AL0, 2 directory blocks |
| #00 | AL1 |
| 16 | CKS, size of checksum vector |
| 0 | OFF, reserved tracks |
| | |
| #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 sector |
| 0 | current track |
| 0 | not aligned |
| 0 | do auto select format |

**IBM Format**

| | |
|---|---|
| 32 | SPT, records per track |
| 3 | BSH, block shift |
| 7 | BLM, block mask |
| 0 | EXM, extent mask |
| 155 | DSM, number of blocks - 1 |
| 63 | DRM, number of directory entries - 1 |
| #C0 | AL0, 2 directory blocks |
| #00 | AL1 |
| 16 | CKS, size of checksum vector |
| 1 | OFF, reserved tracks |
| #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 sector |
| 0 | current track |
| 0 | not aligned |
| 0 | do auto select format |

# 2.16 The Restart Instructions and Locations

The BIOS uses the restart instructions and their locations as follows:

| | |
|---|---|
| RST 0 | : used by CP/M |
| RST 1..RST 5 | : used for firmware calls etc. |
| RST 6 | : available to the user - used by DDT. |
| RST 7 | : used for interrupts. |

An application program may use RST6 without any further ado.

An application program may not use RST7 or its location since this is used for interrupts - unless, that is, the program wishes to intercept interrupts.

If an application program wishes to use the restart instructions 1..5 and/or their locations for its own purposes then it must intercept all entries into the BIOS and restore the restart locations.

Initially copy the required restart location(s) into RAM. On entry to a BIOS routine, and again on exit, swap the restart location(s) with the RAM version.

If an application program does not call the BIOS directly then only the BDOS entry at #0005 need be intercepted.

The restart location(s) must also be restored if any firmware or extended BIOS jumpblock routines are called.

The DDT program supplied has been modified to use RST6 instead of the usual RST7.

# 2.17 Using the Alternate Register Set and the ENTER FIRMWARE Routine

The firmware uses BC' and carry' for its own purposes and corrupts the other alternate registers and IY. If an application program wishes to use these registers then the BIOS will automatically save the users alternate and IY registers on entry into the BIOS and restore them on exit. While inside the BIOS these registers are set as required by the firmware. If the application programs call firmware or extended BIOS routines then they must be called via the ENTER FIRMWARE routine which performs the necessary register saving.

The routine SET REG SAVE sets or clears the register saving option.

If register saving is enabled then an application program:

May use the alternate and IY registers.

May keep the stack under a ROM.

Must use the ENTER FIRMWARE routine for calling firmware and extended BIOS routines.

(It is not recommended, but exceptionally a firmware routine can be called directly if both the user and the routine called preserve BC' and carry' and the stack is not under a ROM. Routines which change the screen mode or the ROM enable state will change BC'.)

If register saving is disabled then an application program:

Must not use the alternate registers.

If the stack is under a ROM then must use the ENTER FIRMWARE routine.

If the stack is not under a ROM then may call the firmware directly.

The BIOS will corrupt IY.

It is strongly recommended that the BIOS is run with the alternate register saving enabled and that application programs follow the procedure outlined above. If an application program does disable the register saving it should re-enable it on exit. Otherwise certain standard application programs may not work.

SET REG SAVE must not be called via ENTER FIRMWARE.

See the firmware documentation for more details on the use of the alternate registers.

Note: the SETUP utility describes register saving enabled as 'slow' and disabled as 'fast'; these terms are misleading. The difference in speed is negligible.

BLANK PAGE

# Chapter 3 - AMSDOS

AMSDOS is a disc operating system for the AMSTRAD CPC464 fitted with the DDI-1 floppy disc interface. AMSDOS enables BASIC programs to 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 in that, for AMSDOS, filenames must conform to CP/M standards whereas 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.

## 3.1 Features

AMSDOS provides the following facilities:

Switching the cassette input and output streams (#9) to and from disc. Thus all the facilities available on 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 user issues the command `OPENOUT "FRED.BAS"` then AMSDOS will create a new file called `FRED.$$$`. When the command `CLOSEOUT` is issued the existing `FRED.BAK` is deleted, `FRED.BAS` is renamed to `FRED.BAK` and `FRED.$$$` is 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 CATALOG

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

The AMSDOS external commands are:

| | |
|---|---|
| A | Select default drive A: |
| B | Select 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 a file |
| 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 '|', and some require parameters.

# 3.2 Filenames

AMSDOS filenames are upwards compatible with CP/M 2.2 filenames in that 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:ANOTHER.BAS` | default drive, explicit user number |
| `2A:WOMBAT.TXT` | all parts specified |
| `*.*` | default user, default drive, all files |
| `5B : POSSUM . $$$` | a name with non-significant spaces |
| `a:aard?ark` | lowercase, AMSDOS will convert to upper case. |

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:

letters digits `! " # $ & ' + - @ ↑ ` } { ~`

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

```
Bad command
```

The characters '?' and '*' are wildcards (see the CP/M Operating System Manual). The name part may be up to 8 characters long. The type part may be up to 3 characters long.

When parsing a filename AMSDOS will shift lower case letters into upper case and remove bit 7.

If the user or drive is omitted then the current default values are assumed. These are user settable.

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 the default type part of three spaces is assumed.

# 3.3 File Headers

Cassette files are subdivided into 2K blocks each of which is preceded by a 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 is an important difference between cassette and disc files.

Unprotected ASCII files do not have headers. 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 unlikely, but possible, 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 header (see below) |
| bytes 64..66 | Length of the file in bytes, excluding the header record. 24 bit number, least significant byte in lowest address |
| bytes 67..68 | Sixteen bit checksum, sum of bytes 0..66 |
| bytes 69..127 | Undefined |

The cassette header fields are used by AMSDOS as follows (see the main firmware documentation for the cassette use of the header):

filename

| | |
|---|---|
| byte 0 | user number, #00..#0F |
| bytes 1..8 | name part, padded with spaces |
| bytes 9..11 | type part, padded with spaces |
| bytes 12..15 | #00 |

The filename in the header is the filename used to create the file. If the actual name or user is subsequently changed this entry is not updated

| | | |
|---|---|---|
| block number | byte 16 | not used, set to 0 |
| last block | byte 17 | not used, set to 0 |
| file type | byte 18 | as per cassette |
| data length | bytes 19..20 | as per cassette |
| data location | bytes 19..20 | as per cassette |
| first block | byte 23 | set to #FF, only used for output files |
| logical length | bytes 24..25 | as per cassette |
| entry address | bytes 26..27 | as per cassette |
| unallocated | bytes 28..63 | as per cassette |

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

| | | |
|---|---|---|
| filename | byte 0 | user number, #00..#0F |
| | bytes 1..8 | name part, padded with spaces |
| | bytes 9..11 | type part, padded with spaces |
| | bytes 12..15 | 0 |
| file type | byte 18 | #16, unprotected ASCII version 1 |
| data location | bytes 19..20 | address of 2K buffer |
| first block | byte 23 | #FF |

All other fields are set to zero

## 3.4 Changing Discs

Under AMSDOS a disc may be changed, or removed, whenever the drive is not being accessed and neither the input nor output files are open on that drive. Unlike CP/M there is no need to 'log in' a disc.

Changing a disc while it is still being written to may corrupt the data on the disc.

If a disc is changed while there are still files open on it then, as soon as AMSDOS detects this, all the open files on the drive will be abandoned and an error message produced. Any data yet to be written will be lost and the latest directory entry will not be written to disc. However, AMSDOS can only detect this change when it reads the directory, which it does every 16K of the file and whenever a file is opened or closed. Thus, potentially, 16K of data could be corrupted by changing a disc while there are still files open on it.

Note that if there is any doubt as to whether or not there are files open on a drive issuing any of the BASIC commands `CAT`, `RUN`, `LOAD`, `CHAIN`, `MERGE` or `CHAIN MERGE` will abandon both the input and output files.

## 3.5 Intercepted Cassette Firmware Routines

When AMSDOS is initialized it copies the relevant cassette jumpblock entries into its own data area. When `DISC` is selected the cassette jumpblock entries are overwritten by AMSDOS entries, when `TAPE` is selected the original cassette entries are restored.

Initially the disc routines are selected.

In order to intercept the jumpblock entries the following procedure should be observed: copy the three bytes from the required jumpblock entry into your own data area - do not make any assumption as to what these three bytes are. Replace the jumpblock entry with your own JMP, RST or whatever. When you receive control restore the jumpblock entry, and CALL it. When you receive control once again save the jumpblock entry and replace it with your own. This procedure will work no matter what the jumpblock entry contains.

Note: when intercepting AMSDOS routines the above procedure must be followed, merely executing a copy of the jumpblock entry will not work, it must be restored to its original place in the jumpblock.

So far as it is possible the AMSDOS routines all have the same interface as their cassette counterparts, although in some cases the interpretation of the return parameters is different. Errors which are detected by both the cassette and disc routines are returned carry false, zero false. Errors which are only detected by the disc routines are returned carry false, zero true. This latter case corresponds to the cassette routine BREAK condition. In both cases register A contains an error number.

When a routine fails (carry false) it returns a six bit error number in the A register. Bit 6 is zero and bit 7 is one if the error has already been reported to the user. The error numbers are as follows:

| | |
|---|---|
| #0E | the file is not open as expected. |
| #0F | hard end of file. |
| #1A | soft end of file. |
| #20 | bad command, usually caused by an incorrect filename. |
| #21 | file already exists. |
| #22 | file doesn't exist. |
| #23 | directory is full. |
| #24 | disc is full. |
| #25 | disc has been changed with files open on it. |
| #26 | file is read-only. |

Errors detected by the floppy disc controller are reported as a bit significant value between #40..#7F, i.e. bit 6 is always one and bit 7 is zero. The other bits are returned as follows:

| | | |
|---|---|---|
| bit 5 | data error | - CRC error on data or ID field. |
| bit 4 | overrun error. | |
| bit 3 | drive not ready | - there is no disc in the drive. |
| bit 2 | no data | - can't find the sector. |
| bit 1 | not writable | - disc is write protected. |
| bit 0 | address mark missing. | |

On the following pages are the interfaces to the intercepted routines:

# 125 CAS IN OPEN (DISC) #BC77

Open a file for input.

## Action:

Set up the read stream for reading a file and read the header if there is one, otherwise create a fake header in store.

## Entry conditions:

B contains the length of the filename.
HL contains the address of the filename.
DE contains the address of a 2K buffer to use.

## Exit conditions:

If the file was opened OK:

Carry true.
Zero false.
HL contains the address of a buffer containing the file header.
DE contains the data location (from the header).
BC contains the logical file length (from the header).
A contains the file type (from the header).

If the stream is already open:

Carry false.
Zero false.
A = error number, #0E.
BC, DE and HL corrupt.

If the open failed for any other reason:

Carry false.
Zero true.
A = error number.
BC, DE and HL corrupt.

Always:

IX and other flags corrupt.
All other registers preserved.

## Notes:

The 2K buffer (2048 bytes) supplied is used to store the contents of the file when it is read from disc. It will remain in use until the file is closed by calling either CAS IN CLOSE or CAS IN ABANDON. The buffer may lie anywhere in memory, even underneath a ROM.

The filename must conform to the AMSDOS conventions with no wild cards. The filename may lie anywhere in RAM, even underneath a ROM.

If the type part of the filename is omitted AMSDOS will attempt to open, in turn, a file with the following type parts '.  ', '.BAS', '.BIN', If none of these exist then the open will fail.

When the file is opened the first record of the file is read immediately. If this record contains a header then it is copied into store, otherwise a fake header is constructed in store. The address of the area where the header is stored is passed back to the user so that information can be extracted from it. This area will lie in the central 32K of RAM. The user is not allowed to write to the header, only to read from it. AMSDOS uses some fields in the header for its own purposes and so these may differ from those read from the disc. The file type, logical length, entry point and all user fields will remain unchanged.

# 126 CAS IN CLOSE (DISC) #BC7A

Close the input file properly.

## Action:

Mark the read stream as closed.

## Entry conditions:

No conditions.

## Exit conditions:

If the stream was closed OK:

Carry true.
Zero false.
A corrupt.

If the stream is not open:

Carry false.
Zero false.
A = error number, #0E.

If the close failed for any other reason:

Carry false.
Zero true.
A = error number.

## Always:

BC, DE, HL and other flags corrupt.
All other registers preserved.

## Notes:

This routine should be called to close a file after reading from it using either CAS IN CHAR or CAS IN DIRECT.

The user may reclaim the buffer passed to CAS IN OPEN after calling this routine.

The drive motor is turned off immediately after the input file is closed. This is done so that a loaded program which takes over the machine is not left with the motor running indefinitely.

# 127 CAS IN ABANDON (DISC) #BC7D

Close the input file immediately.

## Action:

Abandon reading from the read stream and close it.

## Entry conditions:

No conditions.

## Exit conditions:

AF, BC, DE and HL corrupt.
All other registers preserved.

## Notes:

This routine is intended for use after an error or in similar circumstances.

The user may reclaim the buffer passed to CAS IN OPEN after calling this routine.

# 128 CAS IN CHAR (DISC) #BC80

Read a character from the input file.

## Action:

Read a character from the input stream.

## Entry conditions:

No conditions.

## Exit conditions:

If the character was read OK:

Carry true.
Zero false.
A contains the character read from the file.

If the end of the file was found, or stream not open as expected:

Carry false.
Zero false.
A = error number, #0E, #0F, #1A.

If failed for any other reason:

Carry false.
Zero true.
A = error number.

Always:

IX and other flags corrupt.
All other registers preserved.

## Notes:

Once the first character has been read from a file the rest of the file may only be read character by character (using CAS IN CHAR). It is not possible to switch to direct reading (by CAS IN DIRECT).

The CP/M end of file character is treated as end of file (carry false, zero false), however, it is possible to continue reading characters until the hard end of file. A is set to #1A for the CP/M end of file and #0F for hard end of file.

This action of spotting #1A is not performed by the equivalent cassette routine. It is necessary in order to deal with CP/M files generated by other programs. Thus special action will be required if the file is known to contain binary data, viz soft eof must be ignored. The version 1.0 cassette routine CAS IN CHAR never returns the value #1A when carry is false.

# 129 CAS IN DIRECT (DISC) #BC83

Read the input file into store.

## Action:

Read the input file directly into store in one go rather than one character at a time.

## Entry conditions:

HL contains the address to put the file (anywhere in RAM).

## Exit conditions:

If the file was read OK:

Carry true.
Zero false.
HL contains the entry address (from the header).
A corrupt.

If the stream is not open as expected:

Carry false.
Zero false.
A = error number, #0E.
HL corrupt.

If the read failed for any other reason:

Carry false.
Zero true.
A = error number.
HL corrupt.

Always:

BC, DE, IX and other flags corrupt.
All other registers preserved.

## Notes:

The read stream must be newly opened (by CAS IN OPEN). If the stream has been used for character access (by calling CAS IN CHAR or CAS TEST EOF) then it is not possible to directly read the file. Neither is it possible to directly read from the file more than once. (Any attempt to do so will corrupt the copy of the file read.)

If the file has a header then the number of bytes read is that recorded in the 24 bit file length field (bytes 64..66 of the disc file header). If there is no header the file is read until hard end of file.

The CP/M end of file character, #1A, is not treated as end of file.

# 130 CAS RETURN (DISC) #BC86

Put the last character read back.

## Action:

Put the last character read by CAS IN CHAR back into the read buffer. The character will be re-read next time CAS IN CHAR is called.

## Entry conditions:

No conditions.

## Exit conditions:

All registers and flags preserved.

## Notes:

It is only possible to use this routine to return the last character that has been read by CAS IN CHAR. At least one character must have been read since:

the stream was opened
or the last character was returned
or the last test for the end of file was made.

# 131 CAS TEST EOF (DISC) #BC89

Have we reached the end of the input file yet?

## Action:

Test if the end of the input file has been reached.

## Entry conditions:

No conditions.

## Exit conditions:

If the end of the file was not found:

Carry true.
Zero false.
A corrupt.

If the end of the file was found or stream not open as expected:

Carry false.
Zero false.
A = error number. #0E, #0F, #1A.

If failed for any other reason:

Carry false.
Zero true.
A = error number.

Always:

IX and other flags corrupt.
All other registers preserved.

## Notes:

This routine will report end of file if either there are no more characters in the file or if the next character to be read is the CP/M end of file character, #1A.

Calling this routine puts the stream into character input mode. It is not possible to use direct reading after calling this routine.

It is not possible to call CAS RETURN after this routine has been called. A character must be read first.

# 132 CAS OUT OPEN (DISC) #BC8C

Open a file for output.

## Action:

Set up the write stream for output.

## Entry conditions:

B contains the length of the filename.
HL contains the address of the filename.
DE contains the address of a 2K buffer to use.

## Exit conditions:

If the file was opened OK:

Carry true.
Zero false.
HL contains the address of a buffer containing the header.
A corrupt.

If the stream is open already:

Carry false.
Zero false.
A = error number, #0E.
HL corrupt.

If the open failed for any other reason:

Carry false.
Zero true.
A = error number.
HL corrupt.

Always:

BC, DE, IX and other flags corrupt.
All other registers preserved.

## Notes:

When characters are output to the file using CAS OUT CHAR the 2K buffer supplied is used by AMSDOS to buffer the output. It will remain in use until the file is closed by calling either CAS OUT CLOSE or CAS OUT ABANDON. The buffer may reside anywhere in memory - even underneath a ROM.

The filename passed must conform to AMSDOS conventions with no wild cards. It is copied into the write stream header. The filename may lie anywhere in RAM - even underneath a ROM.

The file is opened with a type part of '**.$$$**' regardless of the type part supplied. Any existing file with the same name and a type part of '**.$$$**' is deleted. The file is renamed to its supplied name when CAS OUT CLOSE is called.

When the stream is opened a header is set up. Many of the fields in the header are set by AMSDOS but the remainder are available for use by the user. The address of this header is passed to the user so that information can be stored in it. The user may write to the file type, logical length, entry point and all user fields. The user is not allowed to write to any other field in the header. The user settable fields are all zeroized initially, with the exception of the file type which is set to unprotected ASCII (version 1). For character output, writing to the header type field should be done before calling CAS OUT CHAR and should not be changed thereafter.

Unless the file is type unprotected ASCII the header will be written to the start of the file when it is closed (CAS OUT CLOSE).

# 133 CAS OUT CLOSE (DISC) #BC8F

Close the output file properly.

## Action:

Mark the write stream as closed and give it its correct name.

## Entry conditions:

No conditions.

## Exit conditions:

If the stream was closed OK:

Carry true.
Zero false.
A corrupt.

If the stream is not open:

Carry false.
Zero false.
A = error number, #0E.

If the close failed for any other reason:

Carry false.
Zero true.
A = error number.

Always:

BC, DE, HL, IX and other flags corrupt.
All other registers preserved.

## Notes:

It is necessary to call this routine after using CAS OUT CHAR or CAS OUT DIRECT to ensure that all the data is written to the disc, to write the header to the start of the file and to give the file its true name.

If no data has been written to the file then it is abandoned and nothing is written to disc. This is for compatability with cassette routines.

When the file was opened it was given the type part of '.$$$'. This routine will rename the file to its true name and rename any existing version to have a '.BAK' type part. This ensures that any previous version of the file is automatically kept as a backup. Any existing '.BAK' version is deleted. If, when the file was opened, the caller did not specify a type part then AMSDOS will use the type part '.BAS' for BASIC files, '.BIN' for binary files and '.  ' for any other, as specified by the file type field in the header.

If the actual length of the file is not a multiple of 128 bytes (a CP/M record) then a CP/M end of file character, #1A, is added to the file. This additional character is not recorded in the length of the file.

If writing is to be abandoned then CAS OUT ABANDON should be called as this does not write any more data to disc.

The user may reclaim the buffer passed to CAS OUT OPEN after calling this routine.

# 134 CAS OUT ABANDON (DISC) #BC92

Close the output file immediately.

## Action:

Abandon the output file and mark the write stream closed. Any unwritten data is discarded and not written to disc.

## Entry conditions:

No conditions.

## Exit conditions:

AF, BC, DE and HL corrupt.
All other registers preserved.

## Notes:

This routine is intended for use after an error or in similar circumstances.

If more than one 16K physical extent has already been written to disc then the file will appear in the disc directory with a type part of '**.$$$**'. Otherwise the file will disappear. This is because each 16K of a file requires a directory entry. A directory entry is not written to disc until the 16K has been written or the file is closed (CAS OUT CLOSE).

# 135 CAS OUT CHAR (DISC) #BC95

Write a character to the output file.

## Action:

Add a character to the buffer for the write stream. If the buffer is already full then it is written to disc before the new character is inserted.

## Entry conditions:

A contains the character to write.

## Exit conditions:

If the character was written OK:

Carry true.
Zero false.
A corrupt.

If the stream is not open as expected:

Carry false.
Zero false.
A = error number, #0E.

If failed for any other reason:

Carry false.
Zero true.
A = error number.

Always:

IX and other flags corrupt.
All other registers preserved.

## Notes:

It is necessary to call CAS OUT CLOSE after sending all the characters to the file to ensure that the file is correctly written to disc.

Once this routine has been called it is not possible to switch to directly writing the file (CAS OUT DIRECT).

# 136 CAS OUT DIRECT (DISC) #BC98

Write the output file directly from store.

## Action:

Write the contents of store directly out to the output file.

## Entry conditions:

HL contains the address of the data to write (to go into the header).
DE contains the length of the data to write (to go into the header).
BC contains the entry address (to go into the header).
A contains the file type (to go into the header).

## Exit conditions:

If the file was written OK:

Carry true.
Zero false.
A corrupt.

If the stream is not open as expected:

Carry false.
Zero false.
A = error number, #0E.

If failed for any other reason:

Carry false.
Zero true.
A = error number.

Always:

BC, DE, HL, IX and other flags corrupt.
All other registers preserved.

## Notes:

After writing the file it must be closed using CAS OUT CLOSE to ensure that the file is written to disc.

It is not possible to change the method for writing files from character output (using CAS OUT CHAR) to direct output (using CAS OUT DIRECT) or vice versa once the method has been chosen. Nor is it possible to directly write a file in two or more parts by calling CAS OUT DIRECT more than once - this will write corrupt data.

# 137 CAS CATALOG (DISC) #BC9B

Display the disc directory.

## Action:

Display the disc directory for the current drive and current user. The directory is sorted into alphabetical order and displayed in as many columns as will fit in the current text window (stream #0). The size in Kbytes is shown along side each file. The total amount of free space on the disc is also shown.

## Entry conditions:

DE contains the address of a 2K buffer to use.

## Exit conditions:

If the cataloging went OK:

Carry true.
Zero false.
A corrupt.

If failed for any reason:

Carry false.
Zero true.
A = error number.

Always:

BC, DE, HL, IX and other flags corrupt.
All other registers preserved.

## Notes:

Files marked SYS are not shown.

Files marked R/O are shown with a '*' after the file name.

Unlike the cassette version of this routine, the disc/cassette input stream is not required. (Note: BASIC abandons both the input and output streams when CATaloging.)

# 3.6 External Commands

## |CPM

### Action:

This command shuts down BASIC and AMSDOS and cold boots CP/M.

### Parameters:

None.

### Notes:

MC START PROGRAM is used so all ticker chains etc are lost.

The CP/M utility `AMSDOS.COM` performs the inverse function and restores AMSDOS and BASIC.

# |DISC.IN

## Action:

This command redirects the tape input firmware entries to their disc counterparts.

## Parameters:

None

## Notes:

The redirected firmware entries are:

**CAS IN OPEN**
**CAS IN CLOSE**
**CAS IN ABANDON**
**CAS IN CHAR**
**CAS IN DIRECT**
**CAS RETURN**
**CAS TEST EOF**
**CAS CATALOG**

# |DISC.OUT

## Action:

This command redirects the tape output firmware entries to their disc counterparts.

## Parameters:

None.

## Notes:

The redirected firmware entries are:

**CAS OUT OPEN**
**CAS OUT CLOSE**
**CAS OUT ABANDON**
**CAS OUT CHAR**
**CAS OUT DIRECT**

# |DISC

## Action:

This command redirects both the tape input and output firmware entries to their disc counterparts.

## Parameters:

None

## Notes:

The redirected firmware entries are:

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

`|DISC` is equivalent to the two commands `|DISC.IN |DISC.OUT`.

# \|TAPE.IN

## Action:

This command restores the tape input firmware entries to the state they were before AMSDOS was initialized.

## Parameters:

None.

## Notes:

The restored firmware entries are:

```
CAS IN OPEN
CAS IN CLOSE
CAS IN ABANDON
CAS IN CHAR
CAS IN DIRECT
CAS RETURN
CAS TEST EOF
CAS CATALOG
```

# \|TAPE.OUT

## Action:

This command restores the tape output firmware entries to the state they were before AMSDOS was initialized.

## Parameters:

None.

## Notes:

The restored firmware entries are:

```
CAS OUT OPEN
CAS OUT CLOSE
CAS OUT ABANDON
CAS OUT CHAR
CAS OUT DIRECT
```

# |TAPE

## Action:

This command restores the tape firmware entries to the state they were before AMSDOS was initialized.

## Parameters:

None.

## Notes:

The restored firmware entries are:

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

`|TAPE` is equivalent to the two commands `|TAPE.IN |TAPE.OUT`

# |A

## Action:

Set the default drive to drive A.

## Parameters:

None.

## Notes:

This command is equivalent to the |DRIVE command with 'A' as a parameter.

This command will fail if AMSDOS is unable to determine the format of the disc in drive A. In which case the default drive will not be changed.

When AMSDOS is initialized the default drive is set to drive A.

# |B

## Action:

Set the default drive to drive B.

## Parameters:

None.

## Notes:

This command is equivalent to the |DRIVE command with 'B' as a parameter.

The command will fail if AMSDOS is unable to determine the format of the disc in drive B. In which case the default drive is not changed.

When AMSDOS is initialized the default drive is set to drive A.

# |DRIVE

## Action:

Set the current default drive.

## Parameters:

One string parameter.

## Notes:

The string parameter must be a single letter in the range 'A'..'P' or 'a'..'p'. Drives 'C'..'P' are for future enhancement.

The command will fail if AMSDOS is unable to determine the format of the disc in the requested drive. In which case the default drive will remain unchanged.

When AMSDOS is initialized the default drive is set to drive A.

In BASIC string parameters must be passed by address, e.g.

```
10 A$ = "A"
20 |DRIVE, @A$
```

# |USER

## Action:

Set the default user number.

## Parameters:

One integer parameter.

## Notes:

The user number must be in the range 0..15. Any other parameter will cause an error and the default user will remain unchanged.

When AMSDOS is initialized the default user number is set to 0.

# |DIR

## Action:

Display the disc directory.

## Parameters:

One optional string parameter.

## Notes:

The parameter is a filename, possibly containing wild cards, only those files which match this filename are displayed. If the parameter is omitted then '*.*' is assumed.

The total amount of free space on the disc is shown in Kbytes.

The directory is displayed in as many columns as will fit in the current text window (stream #0).

Files marked SYS are not shown.

Files without an extent zero are not shown.

Unlike CAS CATALOG (DISC) the directory is neither sorted nor are the sizes shown. The output is similar to that of the CP/M DIR command.

In BASIC string parameters must be passed by address, e.g.

```
10 F$="*.BAS"
20 |DIR,@F$
```

# |ERA

## Action:

Erase files.

## Parameters:

One string parameter.

## Notes:

The string parameter is a filename, possibly containing wild cards. All files which match this filename are erased.

A file which matches the filename but is marked R/O will not be erased. In this event a message is displayed for each 16K (extent) of the file.

If none of the files on the disc match the filename then an error message is displayed.

In BASIC string parameters must be passed by address, e.g.

```
10 F$="FILE.BAS"
20 |ERA,@F$
```

# I REN

## Action:

Rename a file.

## Parameters:

Two string parameters.

## Notes:

The first string parameter is the new name for the file. A file of this name must not already exist. The second parameter is the name of the file to be renamed.

Neither name may contain wild cards.

Both files must be on the same drive.

The files may be in different users.

If the file to be renamed is marked R/O then an error message is displayed and the file is not renamed.

The renamed file will have attributes R/W DIR regardless of the original file's attributes.

If the file to be renamed does not exist then an error message is displayed.

In BASIC string parameters must be passed by address, e.g.

```
10 OLDNAME$ = "OLDNAME.BAS"
20 NEWNAME$ = "NEWNAME.BAS"
30 IREN,@NEWNAME$, @OLDNAME$
```

# 3.7 AMSDOS Messages

AMSDOS uses the CP/M BIOS in order to access the disc. Thus 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 has already been used.

‹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 are no more free directory entries (there are 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 the existing version of the file is read-only.

# 3.8 BIOS Facilities Available To AMSDOS

AMSDOS uses the CP/M BIOS 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. To find the address of the required routine call KL FIND COMMAND. The command names are single control characters (Ctrl A..Ctrl I) as these cannot be typed in from BASIC.

**NOTE ** the BIOS extended jumpblock itself is not available, indeed it does not exist in an 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) |

See chapter 2.14 for the interfaces to these routines.

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 Chapter 2.15 for details.)

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

# 3.9 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 the AMSDOS variables to be wherever you so wish. The loader can now load the machine code program using the AMSDOS routines CAS OPEN IN, 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 byte of the far address is the ROM number. This must obviously be done before AMSDOS is shut down.

Existing machine code programs, devoloped on cassette systems without any expansion ROMs, frequently only use store to #ABFF in order to avoid BASIC's variables. These can be easily 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.

BLANK PAGE

# Chapter 4 - Utility Programs

Supplied with the disc system are a number of utility programs. Some of these are supplied by Digital Research and are fully documented in SOFT159 - A Guide to CP/M. The others are specific to the CPC464 and are documented here.

The CPC464 specific utilities are:

| | |
|---|---|
| AMSDOS.COM | Return to AMSDOS and BASIC |
| CLOAD.COM | Copy a file from cassette to disc. |
| CSAVE.COM | Copy a file from disc to cassette. |
| CHKDISC.COM | Compare two discs using two drives. |
| DISCCHK.COM | Compare two discs using one drive. |
| COPYDISC.COM | Copy one disc to another using two drives. |
| DISCCOPY.COM | Copy one disc to another using one drive. |
| FILECOPY.COM | Copy files. |
| FORMAT.COM | Format a disc in drive A:. |
| SETUP.COM | Change the data in the configuration sector. |
| BOOTGEN.COM | Write the boot and configuration sectors to disc. |
| SYSGEN.COM | Write the CCP and BDOS to the system tracks. |

# 4.1 AMSDOS.COM

## Introduction

AMSDOS has been written to allow the user to exit from the CP/M environment, back to BASIC, under the AMSDOS environment. The AMSDOS program, in effect, performs the opposite of ICPM, which is used to exit from AMSDOS to CP/M.

## Using AMSDOS.COM

The program is invoked by typing:

```
AMSDOS
```

No command parameters are required, any parameters entered will be ignored.

The program searches all foreground ROMs for the command BASIC. When the ROM, containing BASIC, has been found, the screen will clear, and the BASIC sign-on message will be printed. The user is now in BASIC. The AMSDOS environment has been re-established, such that BASIC may use the disc system.

In the unlikely event that no foreground ROM containing BASIC can be found the program will display the message:

```
BASIC not found.
```

and perform a warm boot.

# 4.2 CLOAD.COM

## Introduction

CLOAD is a CP/M transient program that reads a cassette file and writes it to a disc file.

CLOAD does not create the AMSDOS header record when writing files to disc. This may cause some compatibility problems when using AMSDOS to access a disc program file created by CLOAD.

## Starting CLOAD

The CLOAD program is held in a file called CLOAD.COM and may be invoked with the command:

```
CLOAD ·cassette file·,·disc file·
```

CLOAD takes two command parameters which are entered after the CLOAD command, preceded by at least one space character. The command parameters may be separated by spaces or commas.

The first parameter, ‹cassette file›, is the name of the cassette file to read, enclosed in double quotes ("). If ‹disc file› is not specified then the trailing quote may be omitted. If ‹cassette file› is not specified then CLOAD reads the first file encountered on the cassette.

If the first character of ‹cassette file› is '!', then it is removed from the filename and has the effect of suppressing the messages generated by the cassette reading software.

If ‹cassette file› is more than 16 characters long then the following message is output and CLOAD is abandoned:-

```
Invalid cassette filename
```

The second parameter, ‹disc file›, is the name of the CP/M file to write. If the cassette filename is also a valid CP/M filename, then the user need not specify ‹disc file›, in which case the CP/M file is given the same name as the cassette file. The user cannot specify ‹disc file› without first specifying ‹cassette file›.

If ‹disc file› is not a valid CP/M filename or if ‹disc file› is not specified and the cassette filename is not a valid CP/M filename then the following message is output and CLOAD is abandoned:-

```
Invalid CP/M filename
```

## Protected Files

If the cassette file that is to be read is protected then the following message is output and CLOAD is abandoned:-

```
Cannot read protected cassette files
```

## Naming Of Disc Files

Whilst CLOAD is transfering data from cassette to disc a temporary file is created with .$$$ as the type. On successful completion of the transfer, if the specified file already exists then it is renamed with .BAK as the type. The temporary file is then given the specified filename.

## Other Messages

```
Program error: Cassette stream in use
```

This message is output if CLOAD fails to open the cassette file because the current stream is in use. This message should not normally be produced.

```
Program error: Cassette stream not in use
```

This message is output if CLOAD fails to close the cassette file because the current stream is not in use. This message should not normally be produced.

```
**Break**
```

This message is output if the **[ESC]** key is pressed whilst reading from the cassette.

```
Disc directory full
```

This message is output if the directory of the disc is full.

```
Failed to rename temporary file
```

This message is output if CLOAD fails to rename the temporary file it created to the specified disc filename. This message should not normally be produced.

```
Disc or directory full
```

This message is output if CLOAD fails to write a record to disc due to either the disc or the directory being full.

```
Failed to close CP/M file correctly
```

This message is output if an error is produced when CLOAD closes the CP/M file.

# 4.3 CSAVE.COM

## Introduction

CSAVE is a CP/M transient program that reads a disc file and writes it to a cassette file.

CSAVE does not use the header record that AMSDOS creates when writing files to disc. This may cause some compatibility problems when using CSAVE to access a disc program file created by AMSDOS.

When CSAVE opens the cassette file it sets the file type to be an ASCII file version 1.

## Starting CSAVE

The CSAVE program is held in a file called CSAVE.COM and may be invoked with the command:

```
CSAVE ·disc file·,·cassette file·,·write speed·
```

CSAVE takes three command parameters which are entered after the CSAVE command, preceeded by at least one space character. The command parameters may be separated by spaces or commas. To specify a particular parameter all preceding parameters must also be specified.

‹disc file› is the name of the CP/M file to read. The user must enter this parameter. The filename may not contain any wildcard characters (* or ?). If ‹disc file› is not a valid CP/M filename then the following message is output and CSAVE is abandoned:-

```
Invalid CP/M filename
```

‹cassette file›, if present, specifies the name of the cassette file to write, enclosed in double quotes ("). If no more parameters are specified after ‹cassette file› then the trailing quote may be omitted. If ‹cassette file› is not present then CSAVE uses the CP/M filename.

If the first character of ‹cassette file› is '!', then it is removed from the filename and has the effect of suppressing the messages generated by the cassette reading software.

If ‹cassette file› is more than 16 characters long then the following message is output and CSAVE is abandoned:-

```
Invalid cassette filename
```

‹write speed›, if present, selects one of two speeds at which data is written to the cassette, zero specifies a nominal 1000 bits per second and one specifies a nominal 2000 bits per second. If no ‹write speed› is given then the slower speed, zero, is assumed.

## Other Messages

```
Program error: Cassette stream in use
```

This message is output if CSAVE fails to open the cassette file because the current stream is in use. This message should not normally be produced.

```
Program error: Cassette stream not in use
```

This message is output if CSAVE fails to close the cassette file because the current stream is not in use. This message should not normally be produced.

```
Invalid speed setting
(you may only specify 0 or 1)
```

This message is output if the user specifies an illegal speed setting.

```
**Break**
```

This message is output if the **[ESC]** key is pressed whilst reading from the cassette.

```
CP/M file does not exist
```

This message is output if the CP/M file to be read does not exist.

# 4.4 CHKDISC.COM

## Introduction

CHKDISC is a CP/M transient program which checks that the disc in drive B is an exact copy of the disc in drive A. The program supports the three AMSTRAD CP/M disc formats.

CHKDISC is for use on two drive systems, for one drive systems use DISCCHK.

CHKDISC requires a minimum TPA size of 39K.

## Starting CHKDISC

The CHKDISC program is held in a file called CHKDISC.COM and may be invoked with the command CHKDISC whereupon it displays a sign on message and then outputs the following prompt:-

```
Please insert source disc into drive A
and destination disc into drive B
Then press any key
```

The source disc is the original disc and the destination disc is the supposed copy.

## Disc Formats

After the user has inserted the discs into drives A and B CHKDISC establishes the format of the source disc. This format is compared with the format of the destination disc. If the two are not the same, or the destination disc has not been formatted, then CHKDISC displays the following error message and the program is abandoned:-

```
Source and destination discs have
different formats
```

Otherwise checking commences with the message:

```
Copy checking started
```

and finishes with the message:

```
Copy checking complete
```

## Copy Checking

The two discs are compared starting at track zero. Each track from the source disc is then compared with the data on the destination disc. If the data is found not to match then the following error message is displayed:-

```
Failed to verify destination disc
correctly: track n sector m
```

where n and m are the track and sector numbers of the bad sector respectively.

## Multiple Checking

When checking is complete the following prompt is issued:-

```
Do you want to check another disc (Y/N):_
```

To check another disc the user should enter Y. To exit CHKDISC and return to CP/M the user should enter N, which results in the following prompt being issued:-

```
Please insert a CP/M system disc into drive A
then press any key:_
```

The user should insert a CP/M system disc into drive A and then press a key. CHKDISC then initiates a warm boot to return to CP/M.

## Other Messages

```
You must insert the source disc into drive A
```

This message is output if there is no disc in drive A when the user has been prompted to insert one. After this message is output the user is again prompted to insert the source disc into drive A.

```
You must insert a CP/M system disc
into drive A
```

This message is output if there is no disc in drive A when the user has been prompted to insert a CP/M system disc. After this message is output the user is again prompted to insert a CP/M system disc into drive A.

```
You must insert the destination disc
into drive B
```

This message is output if there is no disc in drive B when the user has been prompted to insert one. After this message is output the user is again prompted to insert the destination disc into drive B.

```
WARNING: Failed to compare disc correctly
```

This message is output if CHKDISC is abandoned whilst copying.

```
The source disc has an unknown format
```

This message is output if CHKDISC does not recognise the format of the source disc. After this message is output CHKDISC is abandoned.

```
Reading track: n
```

Where n is the current track number. This message is output when CHKDISC is reading a track from the source disc.

```
Checking track: n
```

Where n is the current track number. This message is output when CHKDISC is comparing a track of the destination disc.

```
Failed to read source disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if CHKDISC fails to read a sector from the source disc correctly. After this message is output CHKDISC is abandoned.

```
Failed to read destination disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if CHKDISC fails to read a sector from the destination disc correctly. After this message is output CHKDISC is abandoned.

```
Failed to verify destination disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if the data read back from the destination disc does not match that of the source disc.

```
↑C...aborted
```

This message is output if the user types Control-C when CHKDISC is waiting for user input. After this message is output CHKDISC is abandoned.

```
Illegal message number: Program aborted
```

This message indicates an error in the execution of the CHKDISC program. It should not normally be produced.

```
Insufficient space in TPA
```

This message is output if the size of the Transient Program Area is too small to contain the buffers used by CHKDISC. The user should warm boot CP/M with a 39K or greater CP/M system disc and invoke CHKDISC again.

# 4.5 DISCCHK.COM

## Introduction

DISCCHK is a CP/M transient program which checks that one disc is an exact copy of another, using a single disc drive. The program supports the three AMSTRAD CP/M disc formats.

DISCCHK is for use on a one drive system, for two drive systems use CHKDISC.

DISCCHK checks eight tracks at a time and requires a minimum TPA size of 39K.

## Starting DISCCHK

The DISCCHK program is held in a file called DISCCHK.COM and may be invoked with the command DISCCHK whereupon it displays a sign on message and then outputs the following prompt:-

```
Please insert source disc into drive A
then press any key
```

As checking begins and ends the following messages are displayed:

```
Copy checking started
Copy checking complete
```

## Disc Formats

After the user has inserted the source disc into drive A DISCCHK establishes its format. This format is compared with the format of the destination disc. If the two are not the same, or the destination disc has not been formatted, then DISCCHK displays the following error message and the program is abandoned:-

```
Source and destination discs have
different formats
```

## Copy Checking

The two discs are compared eight tracks at a time, starting at track zero. Each block of eight tracks is read from the source disc then compared with the data on the destination disc. If the data is found not to match then the following error message is displayed:-

```
Failed to verify destination disc correctly:
track n sector m
```

where n and m are the track and sector numbers of the bad sector respectively.

## Changing Discs

DISCCHK uses only drive A for checking. Prior to reading the source disc the following prompt is displayed:-

```
Please insert source disc into drive A
then press any key:_
```

Prior to checking the destination disc the following prompt is displayed:-

```
Please insert destination disc into drive A
then press any key:_
```

The user must ensure that the correct disc is inserted into drive A.

## Multiple Checking

When copy checking is complete the following prompt is issued:-

```
Do you want to check another disc (Y/N):_
```

To copy check another disc the user should enter Y. To exit DISCCHK and return to CP/M the user should enter N, which results in the following prompt being issued:-

```
Please insert a CP/M system disc into drive A
then press any key:_
```

The user should insert a CP/M system disc into drive A and then press a key. DISCCHK then initiates a warm boot to return to CP/M.

## Other Messages

```
You must insert the source disc into drive A
```

This message is output if there is no disc in drive A when the user has been prompted to insert one. After this message is output the user is again prompted to insert the source disc into drive A.

```
You must insert a CP/M system disc into drive A
```

This message is output if there is no disc in drive A when the user has been prompted to insert a CP/M system disc. After this message is output the user is again prompted to insert a CP/M system disc into drive A.

```
You must insert the destination disc into
drive A
```

This message is output if there is no disc in drive A when the user has been prompted to insert one. After this message is output the user is again prompted to insert the destination disc into drive A.

```
WARNING: Failed to compare disc correctly
```

This message is output if **DISCCHK** is abandoned whilst copying.

```
The source disc has an unknown format
```

This message is output if **DISCCHK** does not recognise the format of the source disc. After this message is output **DISCCHK** is abandoned.

```
Source and destination discs have
different formats
```

This message is output if the format of the source disc is different to that of the destination disc. After this message is output **DISCCHK** is abandoned.

```
Reading track: n
```

Where n is the current track number. This message is output when **DISCCHK** is reading a track from the source disc.

```
Checking track: n
```

Where n is the current track number. This message is output when **DISCCHK** is comparing a track of the destination disc.

```
Failed to read source disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if **DISCCHK** fails to read a sector from the source disc correctly. After this message is output **DISCCHK** is abandoned.

```
Failed to read destination disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if **DISCCHK** fails to read a sector from the destination disc correctly. After this message is output **DISCCHK** is abandoned.

```
Failed to verify destination disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if the data read back from the destination disc does not match that of the source disc.

```
↑C...aborted
```

This message is output if the user types Control-C when **DISCCHK** is waiting for user input. After this message is output **DISCCHK** is abandoned.

```
Illegal message number: Program aborted
```

This message indicates an error in the execution of the DISCCHK program. It should not normally be produced.

```
Insufficient space in TPA
```

This message is output if the size of the Transient Program Area is too small to contain the buffers used by DISCCHK. The user should warm boot CP/M with a 39K or greater CP/M system disc and invoke DISCCHK again.

# 4.6 COPYDISC.COM

## Introduction

COPYDISC is a CP/M transient program that copies the data from a disc in drive A onto a disc in drive B. The program supports the three AMSTRAD CP/M disc formats.

COPYDISC is for use on a two drive system. For a one drive system use DISCCOPY.

COPYDISC requires a minimum TPA size of 39K.

## Starting COPYDISC

The COPYDISC program is held in a file called COPYDISC.COM and may be invoked with the command COPYDISC whereupon it displays a sign on message and then outputs the following prompt:-

```
Please insert source disc into drive A
and destination disc into drive B
then press any key
```

As copying begins and ends the following messages are displayed:

```
Copying started

Copying complete
```

## Disc Formats

After the user has inserted the discs into drives A and B COPYDISC establishes the format of the source disc. This format is compared with the format of the destination disc. If the two are not the same, or the destination disc has not been formatted, then COPYDISC will format each track of the destination disc as it copies.

## Copying

The source disc is copied starting at track zero. Each track written to the destination disc is verified by reading it back and comparing it with the original data.

## Multiple Copies

When copying is complete the following prompt is issued:-

```
Do you want to copy another disc (Y/N):_
```

To copy another disc the user should enter `Y`. To exit `COPYDISC` and return to CP/M the user should enter `N`, which results in the following prompt being issued:-

```
Please insert a CP/M system disc into drive A
then press any key:_
```

The user should insert a CP/M system disc into drive A and then press a key. `COPYDISC` then initiates a warm boot to return to CP/M.

## Error Messages

```
You must insert the source disc into drive A
```

This message is output if there is no disc in drive A when the user has been prompted to insert one. After this message is output the user is again prompted to insert the source disc into drive A.

```
You must insert a CP/M system disc
into drive A
```

This message is output if there is no disc in drive A when the user has been prompted to insert a CP/M system disc. After this message is output the user is again prompted to insert a CP/M system disc into drive A.

```
You must insert the destination disc
into drive B
```

This message is output if there is no disc in drive B when the user has been prompted to insert one. After this message is output the user is again prompted to insert the destination disc into drive B.

```
The destination disc in drive B
must be write-enabled
```

This message is output if the disc in drive B is write-protected. After this message is output the user is again prompted to insert the destination disc into drive B.

```
Formatting whilst copying
```

This message is output if the destination disc is unformatted or does not have the same format as the source disc.

```
WARNING: Failed to copy disc correctly
The destination disc should not be used until
it is successfully copied onto.
```

This message is output if COPYDISC is abandoned whilst copying.

```
The source disc has an unknown format
```

This message is output if COPYDISC does not recognise the format of the source disc. After this message is output COPYDISC is abandoned.

```
Reading track: n
```

Where n is the current track number. This message is output when COPYDISC is reading a track from the source disc.

```
Writing track: n
```

Where n is the current track number. This message is output when COPYDISC is writing and verifying a track of the destination disc.

```
Failed to read source disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if COPYDISC fails to read a sector from the source disc correctly. After this message is output COPYDISC is abandoned.

```
Failed to write destination disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if COPYDISC fails to write a sector to the destination disc correctly. After this message is output COPYDISC is abandoned.

```
Failed to read destination disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if COPYDISC fails to read a sector from the destination disc correctly. After this message is output COPYDISC is abandoned.

```
Failed to verify destination disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if the data read back from the destination disc does not match that of the source disc.

```
Failed to format destination disc correctly:
track n
```

Where n is the current track number. This message is output if COPYDISC fails to format a track of the destination disc correctly. After this message is output COPYDISC is abandoned.

```
↑C...aborted
```

This message is output if the user types Control-C when COPYDISC is waiting for user input. After this message is output COPYDISC is abandoned.

```
Illegal message number: Program aborted
```

This message indicates an error in the execution of the COPYDISC program. It should not normally be produced.

```
Insufficient space in TPA
```

This message is output if the size of the Transient Program Area is too small to contain the buffers used by COPYDISC. The user should warm boot CP/M with a 39K or greater CP/M system disc and invoke COPYDISC again.

# 4.7 DISCCOPY.COM

## Introduction

DISCCOPY is a CP/M transient program that copies the data from one disc onto another, using a single disc drive. The program supports the three AMSTRAD CP/M disc formats.

DISCCOPY is for use on a one drive system. For two drive systems use COPYDISC.

DISCCOPY copies eight tracks at a time and requires a minimum TPA size of 39K.

## Starting DISCCOPY

The DISCCOPY program is held in a file called DISCCOPY.COM and may be invoked with the command DISCCOPY whereupon it displays a sign on message and then outputs the following prompt:-

```
Please insert source disc into drive A
then press any key
```

As copying begins and ends the following messages are displayed:

```
Copying started

Copying complete
```

## Disc Formats

After the user has inserted the source disc into drive A DISCCOPY establishes its format. This format is compared with the format of the destination disc. If the two are not the same, or the destination disc has not been formatted, then DISCCOPY will format each track of the destination disc as it copies.

## Copying

The source disc is copied eight tracks at a time, starting at track zero. Each block of eight tracks is read from the source disc with contiguous reads. It is then written to the destination disc, and verified, one track at a time. This keeps the number of disc interchanges required to a minimum.

## Changing Discs

DISCCOPY uses drive A only for copying. Prior to reading the source disc the following prompt is displayed:-

```
Please insert source disc into drive A
then press any key
```

Prior to writing the destination disc the following prompt is displayed:-

```
Please insert destination disc into drive A
then press any key
```

The user must ensure that the correct disc is inserted into drive A.

## Multiple Copies

When copying is complete the following prompt is issued:-

```
Do you want to copy another disc (Y/N):_
```

To copy another disc the user should enter `Y`. To exit `DISCCOPY` and return to CP/M the user should enter `N`, which results in the following prompt being issued:-

```
Please insert a CP/M system disc into drive A
then press any key:_
```

The user should insert a CP/M system disc into drive A and then press a key. `DISCCOPY` then initiates a warm boot to return to CP/M.

## Other Messages

```
You must insert the source disc into driveA
```

This message is output if there is no disc in drive A when the user has been prompted to insert one. After this message is output the user is again prompted to insert the source disc into drive A.

```
You must insert a CP/M system disc
into drive A
```

This message is output if there is no disc in drive A when the user has been prompted to insert a CP/M system disc. After this message is output the user is again prompted to insert a CP/M system disc into drive A.

```
You must insert the destination disc
into drive A
```

This message is output if there is no disc in drive A when the user has been prompted to insert one. After this message is output the user is again prompted to insert the destination disc into drive A.

```
The destination disc in drive A
must be write-enabled
```

This message is output if the disc in drive A is write-protected. After this message is output the user is again prompted to insert the destination disc into drive A.

```
Formatting whilst copying
```

This message is output if the destination disc is unformatted or does not have the same format as the source disc.

```
WARNING: Failed to copy disc correctly
The destination disc should not be used until
it is successfully copied onto
```

This message is output if DISCCOPY is abandoned whilst copying.

```
The source disc has an unknown format
```

This message is output if DISCCOPY does not recognise the format of the source disc. After this message is output DISCCOPY is abandoned.

```
Reading track: n
```

Where n is the current track number. This message is output when DISCCOPY is reading a track from the source disc.

```
Writing track: n
```

Where n is the current track number. This message is output when DISCCOPY is writing and verifying a track of the destination disc.

```
Failed to read source disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if DISCCOPY fails to read a sector from the source disc correctly. After this message is output DISCCOPY is abandoned.

```
Failed to write destination disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if DISCCOPY fails to write a sector to the destination disc correctly. After this message is output DISCCOPY is abandoned.

```
Failed to read destination disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if DISCCOPY fails to read a sector from the destination disc correctly. After this message is output DISCCOPY is abandoned.

```
Failed to verify destination disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if the data read back from the destination disc does not match that of the source disc.

```
Failed to format destination disc correctly:
track n
```

Where n is the current track number. This message is output if DISCCOPY fails to format a track of the destination disc correctly. After this message is output DISCCOPY is abandoned.

```
↑C...aborted
```

This message is output if the user types Control-C when DISCCOPY is waiting for user input. After this message is output DISCCOPY is abandoned.

```
Illegal message number: Program aborted
```

This message indicates an error in the execution of the DISCCOPY program. It should not normally be produced.

```
Insufficient space in TPA
```

This message is output if the size of the Transient Program Area is too small to contain the buffers used by DISCCOPY. The user should warm boot CP/M with a 39K or greater CP/M system disc and invoke DISCCOPY again.

# 4.8 FILECOPY.COM

## Introduction

The FILECOPY program has been designed to copy disc files from one disc to another whilst optionally transferring files between different user areas. All copying is carried out using drive A. If a two drive system is being used, then PIP is recommended for copying files. The three AMSTRAD CP/M disc formats are supported, however the disc to be written to must be formatted. The source and destination discs may be the same.

The file to be copied is read into a buffer before being written. The amount of buffer memory used is dynamically chosen to suit the size of TPA available.

Any source files that have a read/only status (R/O) will be copied to the destination file as read/write (R/W). Source files hidden from the directory with SYS attributes cannot be copied.

The program may be abandoned, whenever user input is required, by typing Control-C. Any file on the destination disc that has the same name as that being copied will be overwritten. The user should ensure that valuable files are not lost.

## Starting FILECOPY

FILECOPY is invoked with the following command:

```
FILECOPY ‹filename› ‹/users›
```

FILECOPY can be given one or two command parameters, which are entered after the FILECOPY command, preceded by one or more space characters. The second command parameter must be separated from the first, by at least one space character. If no parameters are given, the message:

```
No SOURCE file present on input line
```

will be displayed, before FILECOPY is abandoned.

The first parameter, ‹filename›, is the name of the disc file to copy, and must conform to CP/M conventions. The wildcard characters '?' and '*' may be used in ‹filename›, in which case all matching files will be copied.

The second parameter, ‹/users›, is optional. If the parameter is used the first character must be a '/', otherwise the parameter will be ignored by FILECOPY. The parameter ‹/users› allows the selection of the user area to read the file from, and the user area to write the file to. It consists of one or two options, which may be specified in any order. No space characters are permitted after the '/' character, any options after a space will be ignored by FILECOPY.

The user area to read from, that is, the SOURCE user, may be chosen by using 'S', followed by the user number.

The user area to write to, that is, the DESTINATION user, may be chosen by using 'D', followed by the user number.

The user number for both options must be a decimal number in the range 0..15, otherwise the message:

```
Invalid user number in options
```

will be given and FILECOPY will be abandoned. Numbers less than ten may include an optional leading zero.

If FILECOPY cannot recognise any option, or if a number is longer than two digits, the message:

```
Syntax error in options
```

will be given and FILECOPY will be abandoned.

When the 'S' option is used the message:

```
Copying will be from user n
```

where n is a decimal number in the range 0..15, will be printed as confirmation. When the 'D' option is used the message:

```
Copying will be to user n
```

will be displayed. If both the 'S' and 'D' options are used the message:

```
Copying will be from user n to user n
```

will be given.

If the second parameter is omitted, or ignored by FILECOPY, the current user number is assumed for both reading and writing of files: no confirmation is displayed.

## Copying a single file

When ‹filename› does not contain any wildcard characters a single file is copied. The message:

```
Please insert SOURCE disc into drive A
then press any key:_
```

is given as a prompt. The user should then insert the disc that holds the file to be copied, and then press any of the alphanumeric keys. FILECOPY will then look for the source file on the disc. If the file cannot be found the message:

```
No SOURCE file present on disc
```

will be given, and FILECOPY will prompt for a CP/M system disc to be inserted, before abandoning. When the source file has been found, the message:

```
Copying started....
```

is printed and the file is then read into a buffer until either the end of the file is reached, or the buffer becomes full. The prompt:

```
Please insert DESTINATION disc into drive A
then press any key:_
```

is then displayed. The user should then insert the disc to write the file to, and press a key. The buffer is then written to the disc. If the file cannot be copied in one go, the source disc will again be asked for. The copy process will repeat until the destination file is complete. Note that a large file will require many disc changes. Upon completion of the copy, the message:

```
‹filename› Copied.
```

is printed as confirmation. ‹filename› will be the name of the file actually copied. When all files have been copied the message:

```
Copying complete
Please insert a CP/M system disc into drive A
then press any key:_
```

will be given. The user should then insert a disc with CP/M on it and press a key. FILECOPY will then finish.

## Copying Multiple Files

When ‹filename› contains wildcard characters FILECOPY will allow any matching files to be copied, in a continuous fashion. After prompting for the source disc the message:

```
Ambiguous filename:
Confirm individual files (Y/N)?
```

is displayed. If the user wishes to copy only some of the matching files then 'Y' should be typed. In this case FILECOPY will search for all matching file names, one at a time, displaying each on the screen, with the message:

```
‹filename› Copy (Y/N)?
```

where ‹filename› is the name of a matching file. If the user types 'Y' then the filename is stored in a buffer, ready for copying, and the search for matching files continues. If confirmation of individual files is not selected then all matching filenames are read into a buffer, without being listed. If FILECOPY cannot find any matching filenames on the source disc, then the message:

```
No SOURCE file present on disc
```

will be printed and FILECOPY will be abandoned. If 'N' has been entered as confirmation for each file name listed, then the message:

```
No files to copy
```

will be given and FILECOPY will also be abandoned.

Once all matching filenames have been obtained each file is copied as if it were a single file.

## Examples Of Use

```
FILECOPY STAT.COM
```
Copy the single file STAT.COM

```
FILECOPY HELLO.* /S4
```
Copy all files matching HELLO.* from user 4 to current user.

```
FILECOPY WHAT?.COM /D3S1
```
Copy all files matching WHAT?.COM from user 1 to user 3

```
FILECOPY PIP.COM /S12D01
```
Copy PIP.COM from user 12 to user 1.

## Error Messages

During the operation of FILECOPY the following error messages may occur.

```
↑C...aborted
```

The user has typed Control-C whilst FILECOPY was waiting for a key to be pressed. The program is abandoned.

```
Failed to open SOURCE file correctly
```

This message is given when FILECOPY cannot open the source file. This may be due to a read fail, or the insertion of an incorrect source disc. The program is abandoned.

```
Failed to close DESTINATION disc correctly
```

This message is generated when the destination file cannot be closed. This may be due to the disc directory being full. The program is abandoned.

```
DESTINATION disc directory full
```

This message is generated when there is no room to create a new file in the directory of the destination disc. The program is abandoned.

```
DESTINATION disc full
```

This message is generated when the there is no more storage space on the destination disc. The program is abandoned.

```
The DESTINATION disc has an unknown format
```

This message is printed when the destination disc does not have any of the three AMSTRAD formats, if the disc is unformatted, or if the disc is corrupt. The destination disc prompt is repeated.

```
The SOURCE disc has an unknown format
```

This message is printed when the source disc does not have any of the three AMSTRAD formats, if the disc is unformatted, or if the disc is corrupt. The SOURCE disc prompt is repeated.

```
SOURCE disc missing
```

A key has been pressed in response to the source disc prompt without a disc in the drive. The prompt is repeated.

```
DESTINATION disc missing
```

A key has been pressed in response to the destination disc prompt without a disc in the drive. The prompt is repeated.

```
DESTINATION disc is write protected
```

This message is displayed when `FILECOPY` is unable to write to the destination disc because the disc is write protected. The user should re-insert the same destination disc with the write protect tabs disabled. The destination disc prompt is repeated.

```
Incorrect DESTINATION disc
```

The user has inserted a destination disc different to that originally used to copy the file to. This message is only displayed when more than one buffer of data is required to copy a file. The program is abandoned.

```
Failed to read SOURCE disc correctly
```

This message is output if the source file is incomplete, has zero length, or if a read fail has occurred. The program is abandoned.

```
Failed to write DESTINATION disc correctly
```

This message is displayed if a write fail occurs whilst copying to the destination disc. The program is abandoned.

```
WARNING: DESTINATION file ‹filename› is
incomplete
```

This message is output if FILECOPY cannot successfully write the destination file ‹filename›.

## Sign-off Messages

```
FILECOPY V2.1 abandoned
```

This message is displayed whenever the normal operation of FILECOPY has been affected. The last file to be copied may be incomplete.

```
FILECOPY V2.1 finished
```

This message is given after FILECOPY has successfully copied all files, as requested.

# 4.9 FORMAT.COM

## Introduction

FORMAT is a CP/M transient program that formats discs for AMSTRAD CP/M. The program supports the three AMSTRAD CP/M disc formats.

## Starting FORMAT

The FORMAT program is held in a file called FORMAT.COM and may be invoked with the command:

FORMAT ·optional format parameter·

whereupon it displays a sign on message and then validates the format parameter, if one is specified.

## FORMAT Parameter

When FORMAT is invoked the user may select which format is to be used, by specifying a format parameter. The parameter may be one of S, V, D or I as detailed below and is entered after the FORMAT command, the two being separated by at least one space character.

S - SYSTEM format
V - VENDOR format
D - DATA-ONLY format
I - IBM PC format

The S and V options create discs with the same physical format. The difference is that the V option does not initialize the system tracks. This allows CP/M packages to be distributed on CP/M discs that do not contain the CP/M operating system and thus do not break the Digital Research licensing agreements.

If no parameter is specified then the S option is assumed. If the user enters an illegal parameter (i.e. not S, V, D or I) then the following message is output and the FORMAT
program is abandoned:

```
Bad format option
(you may only enter S, V, D, or I)
```

## System Discs

When formatting system discs the two reserved system tracks must be set up correctly. To do this FORMAT reads these two tracks from the disc that is in drive A when FORMAT is invoked. If this disc is not a system disc then the following message is output and FORMAT is abandoned.

```
The disc in drive A is not a system disc
```

## Formatting

Once the format parameter has been succesfully validated FORMAT outputs the following prompt:-

```
Please insert the disc to be formatted into
drive A then press any key
```

As formatting begins and ends the following messages are displayed:

```
Formatting started

Formatting complete
```

Tracks are formatted sequentially, starting at track zero. After each track is formatted, all sectors on the track are read back to ensure they have been formatted correctly. The data read from each sector is ignored.

## Multiple Formats

When formatting is complete the following prompt is issued:-

```
Do you want to format another disc (Y/N):_
```

To format another disc the user should enter Y. To exit FORMAT and return to CP/M the user should enter N, which results in the following prompt being issued:-

```
Please insert a CP/M system disc into drive A
then press any key:_
```

The user should insert a CP/M system disc into drive A and then press a key. FORMAT then initiates a warm boot to return to CP/M.

## Other Messages

```
You must insert a CP/M system disc
into drive A
```

This message is output if there is no disc in drive A when the user has been prompted to insert a CP/M system disc. After this message is output the user is again prompted to insert a CP/M system disc into drive A.

```
You must insert the disc to be formatted
into drive A
```

This message is output if there is no disc in drive A when the user has been prompted to insert one. After this message is output the user is again prompted to insert the destination disc into drive A.

```
The disc to be formatted in drive A
must be write-enabled
```

This message is output if the disc in drive A is write-protected. After this message is output the user is again prompted to insert the destination disc into drive A.

```
WARNING: Failed to format destination disc
correctly
The destination disc should not be used until
it is successfully formatted
```

This message is output if FORMAT is abandoned whilst formatting.

```
The disc in drive A is not a CP/M system disc
```

This message is output when FORMAT attempts to read the two reserved system tracks from a non CP/M system disc. After this message is output FORMAT is abandoned. The user should insert a CP/M system disc into drive A and invoke FORMAT again.

```
Formatting track: n
```

Where n is the current track number. This message is output when FORMAT is formatting a track from the source disc.

```
Failed to read source disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if FORMAT fails to read a sector from the reserved system track correctly. After this message is output FORMAT is abandoned.

```
Failed to read destination disc correctly:
track n sector m
```

Where n and m are the track and sector numbers of the bad sector respectively. This message is output if FORMAT fails to read a sector from the destination disc correctly. After this message is output FORMAT is abandoned.

```
Failed to format destination disc correctly:
track n
```

Where n is the current track number. This message is output if FORMAT fails to format a track of the destination disc correctly. After this message is output FORMAT is abandoned.

```
↑C...aborted
```

This message is output if the user types Control-C when FORMAT is waiting for user input. After this message is output FORMAT is abandoned.

```
Illegal message number: Program aborted
```

This message indicates an error in the execution of the FORMAT program. It should not normally be produced.

# 4.10 SETUP.COM

## Introduction

SETUP is a CP/M transient program that allows the user to setup the parameters within the configuration sector of an AMSTRAD CP/M system disc.

## Configuration Sector

AMSTRAD CP/M allows the user to set the initial values of certain parameters when CP/M is invoked. These parameters are held in a special sector on the system track called the configuration sector. Refer to chapter 2.13 for details of the format of the configuration sector.

SETUP is a program that allows the user to set the values of the parameters held in the configuration sector of a disc in drive A. SETUP allows the user to change the following configuration parameters:-

1) Initial command buffer.
2) The sign-on string.
3) Printer power-up string.
4) Keyboard translations.
5) Keyboard expansion strings.
6) IOBYTE setting.
7) Saving alternate and IY registers enable/disable.
8) BIOS messages enable/disable.
9) Initial command buffer clearing.
10) Motor on delay.
11) Motor off delay.
12) Drive stepping rate
13) Z80 SIO channel A baud rate, data bits, parity and stop bits.
14) Z80 SIO channel B baud rate, data bits, parity and stop bits.

## Starting SETUP

The SETUP program is held in a file called SETUP.COM and may be invoked with the command SETUP whereupon the program steps through all the parameters in the configuration sector allowing the user to change only the ones required.

If the configuration sector is invalid then SETUP displays the following message:-

```
Invalid config sector:
Do you want to use default settings (Y/N):_
```

If the user enters N in response to the above prompt then SETUP is abandoned. Otherwise the following default parameters are used:-

1) Initial command buffer is empty.
2) The sign-on string is empty.
3) Printer power-up string is empty.
4) No keyboard translations are set.
5) No keyboard expansion strings are set.
6) The default IOBYTE is set for CON:=CRT:, RDR:=TTY:, PUN:=TTY: and LST:=LPT:.
7) Alternate and IY register saving enabled.
8) BIOS messages are enabled.
9) The initial command buffer will be cleared on keyboard input.
10) The motor on delay is set to 50.
11) The motor off delay is set to 250.
12) The drive stepping rate is set to 12 milliseconds
13) Z80 SIO channel A is set for 9600 baud, 8 data bits, no parity and 1 stop bit.
14) Z80 SIO channel B is set for 9600 baud, 8 data bits, no parity and 1 stop bit.

The following sections of this document detail how SETUP allows the user to change each of these parameters.

## Entering Control Codes

A number of SETUP options prompt for text input. Most control codes may be entered simply by pressing the relevant control keys. However, a number of control codes have special meanings and are not entered into the text buffer. These control codes are detailed below. Refer to CP/M BDOS function 10 (read console buffer) documentation for further details on the actions of these control codes.

| | |
|---|---|
| Control-C | warm boots when at the beginning of a line |
| Control-E | causes physical end of line |
| Control-H | backspace one character position |
| Control-J | (line feed) terminates input line |
| Control-M | (return) terminates input line |
| Control-R | retypes the current line after new line |
| Control-U | removes current line |
| Control-X | same as control-U |

The above control codes (and indeed all others) may be entered into the text buffer as up-arrow followed by the ASCII character that represents the required control code, e.g. ↑ C is translated into Control-C. This translation is performed by setting the three most significant bits of the ASCII character code to zero. Thus, control codes may be represented by one of three ASCII characters, e.g. Control-C may be represented by ↑ #, ↑ C or ↑ c.

Two up-arrows next to each other are translated into one.

## Initial Command Buffer

This buffer allows the user to specify up to 128 characters that will be input to CP/M automatically when CP/M is invoked. If the initial command buffer is empty then SETUP displays the following prompt:

```
**Initial command buffer empty
Is this correct (Y/N):_
```

Otherwise SETUP displays the current contents of the initial command buffer as follows:-

```
Initial command buffer:
"command buffer"
Is this correct (Y/N):_
```

Where "command buffer" is the current contents of the initial command buffer. If the user enters Y in response to the above then the initial command buffer remains unaltered and SETUP moves on to the next parameter. If the user enters N then the following prompt is displayed:-

```
Enter new initial command buffer:_
```

and the user may then enter the required initial command buffer of up to 128 characters in length. When the new buffer has been entered, it is displayed and the user asked if this is correct, if not the process is repeated until it is.

## Sign-on string

This is a string of characters that is output to the screen when CP/M is invoked. SETUP prompts the user in the same manner as for the Initial command buffer. The sign-on string may be up to 253 characters long.

## Printer Power-up String

This is a string of characters that is output to the BIOS list device when CP/M is invoked. Note that the actual physical device the string is sent to, depends on the default setting of the IOBYTE. SETUP prompts the user in the same manner as for the Initial command buffer. The printer power-up string string may be up to 253 characters long.

## Keyboard Translation Table

This is a table that redefines the codes generated when keys are pressed. Each entry in the table has four parameters as detailed below:-

‹key number›, ‹normal›, ‹shift›, ‹control›

The ‹key number› is a number in the range 0 to 79, specifying which key to redefine.

The ‹normal›, ‹shift› and ‹control› parameters, if present, are numbers in the range 0 to 255. These parameters redefine the value to be generated when the key is pressed as follows:-

‹normal› the value when neither Shift nor Control is pressed.

‹shift› the value when shift, but not Control, is also pressed.

‹control› the value when Control is also pressed.

The values generated by keys are interpreted as follows:-

| | |
|---|---|
| 0 - 31 | Control characters which usually have special effects. |
| 32 - 127 | Ordinary ASCII characters |
| 128 - 159 | The special characters which are expanded according to the expansion strings that are setup. |
| 160 - 223 | Ordinary characters - not all CP/M programs will be able to cope with characters in this range. |
| 224 - 254 | Reserved for special use by the firmware. |
| 255 | Ignored. |

If no keyboard translations are set then SETUP displays the following prompt:

```
No keyboard translations set
Is this correct (Y/N):_
```

Otherwise SETUP displays the current keyboard translation settings as follows:-

```
Keyboard translations:

Key Code    Normal      Shift      Control

   69          97          65          1
   54          98          66          -
...etc


Is this correct (Y/N):_
```

Where dash means the translation remains unchanged. If the user enters Y in response to the above prompt then the keyboard translations remain unaltered and SETUP moves on to the next parameter. If the user enters N then the following prompt is displayed:-

```
Enter required command from:-

A-Add key translation to table
D-Delete key translation from table
C-Clear translation table
F-Finish translations

Command:_
```

The user should then enter the appropriate command letter followed by a suitable number of parameters. Spaces or commas must be used to separate each parameter. The parameters required by each command are as follows:-

```
A, key number , normal , shift , control
D, key number
C
F
```

Note that the C and F commands require no parameters.

When the F command has been entered the translation table is displayed and the user asked if this is correct, if not, the process is repeated until it is.

## Keyboard Expansion Table

This is a table that redefines the 32 expansion characters. Each entry in the table contains the expansion character and its associated expansion string.

The normal default keyboard expansions 0 to 12, assigned to keys on the numeric keypad, are already set up. The keyboard expansion table allows the user to add extra expansions or to modify the default expansions.

If no extra or modified keyboard expansions are set SETUP displays the following prompt:

```
No keyboard expansions set
Is this correct (Y/N):_
```

Otherwise SETUP displays the extra or modified keyboard expansion settings as follows:-

```
Keyboard expansion strings:

 ExpansionToken                Expansionstring
        0                          DIR↑M
        1                          STAT↑M
etc...

Is this correct (Y/N):_
```

If the user enters Y in response to the above prompt then the keyboard expansions remain unaltered and SETUP moves on to the next parameter. If the user enters N then the following prompt is displayed:-

```
Enter required command from:-

A-Add keyboard expansion to table
D-Delete keyboard expansion from table
C-Clear expansion table
F-Finish keyboard expansions

Command:_
```

The user should then enter the appropriate command letter followed by a suitable number of parameters. Spaces or commas must be used to separate each parameter. The parameters required by each command are as follows:-

```
A,·key number·,·expansion string·
D,·key number·
C
F
```

Note that the C and F commands require no parameters. The A command requires an expansion string of up to 30 characters in length. When the F command has been entered the expansion table is displayed and the user asked if this is correct, if not, the process is repeated until it is.

## Default IOBYTE Setting

The default IOBYTE setting is the value loaded in to the CP/M IOBYTE at location #0003 in RAM when CP/M is invoked. SETUP displays the current default settings of the IOBYTE as follows:-

```
Default IO byte settings are:

CON: is assigned to CON-DEV
RDR: is assigned to RDR-DEV
PUN: is assigned to PUN-DEV
LST: is assigned to LST-DEV

Is this correct (Y/N):_
```

Where:

```
CON-DEV is the default device assigned to CON:
RDR-DEV is the default device assigned to RDR:
PUN-DEV is the default device assigned to PUN:
LST-DEV is the default device assigned to LST:
```

If the user enters Y in response to the above prompt then the default IOBYTE settings remain unaltered and SETUP moves on to the next parameter. If the user enters N then the following prompt is displayed:-

```
Enter required IO byte setting:_
```

The user may then assign one of the four CP/M logical devices to one of its associated physical devices.

‹logical device› ‹physical devive›

Where ‹logical device› is one of CON: RDR: PUN: LST: and ‹physical device› is one of CRT: TTY: BAT: UC1: PTR: UR1: UR2: UP1: UP2: LPT: UL1:. The two devices must be separated by at least one space, comma or equals sign. For example CON:=CRT: may be used to assign the logical device CON: to the physical device CRT:. Detailed below are all the valid assignments of logical to phyisical devices:-

CON: may be assigned to: TTY: CRT: BAT: UC1:
RDR: may be assigned to: TTY: PTR: UR1: UR2:
PUN: may be assigned to: TTY: PTR: UP1: UP2:
LST: may be assigned to: TTY: CRT: LPT: UL1:

Refer to chapter 2.7 for further details on the above assignments.

When a new default IOBYTE setting has been entered it is displayed and the user asked if this is correct, if not the process is repeated until it is.

## Alternate and IY Register Saving enable/disable (MODE).

The MODE setting defines whether or not the BIOS saves the alternate and IY registers. SLOW mode means that the BIOS does save these registers and is the recommended mode. FAST means that the BIOS does not save these registers. The terms SLOW and FAST are misnomers. SETUP prompts as follows:

```
Default mode setting is ‹mode›
Is this correct (Y/N):_
```

Where ‹mode› is the default mode, either fast or slow. If the user enters Y in response to the above prompt then the default MODE setting remains unaltered and SETUP moves on to the next parameter. If the user enters N then the default MODE is changed to the other state and the user asked if this is correct, if not the process is repeated until it is.

It is recommended that the user does not select FAST mode unless there are some special requirements. Any application program which requires the alternate or IY registers will not work in FAST mode, e.g. DR LOGO.

## BIOS Message Enable/Disable

The value of this parameter defines whether BIOS error messages are enabled or disabled when CP/M is invoked. SETUP displays the current default state as follows:-

```
Default: BIOS messages ‹state›

Is this correct (Y/N):_
```

Where ‹state› is the default message state, either enabled or disabled. If the user enters Y in response to the above prompt then the default message setting remains unaltered and SETUP moves on to the next parameter. If the user enters N then the default state is swapped to the other state and the user asked if this is correct, if not the process is repeated until it is.

## Initial command buffer clear/preserve

The value of this parameter defines whether the initial command buffer is cleared when a key is pressed on the keyboard. SETUP displays the current default state as follows:-

```
Default: ‹state› initial command buffer on
keyboard input

Is this correct (Y/N):_
```

Where ‹state› is either preserve or clear. If the user enters Y in response to the above prompt then the default setting remains unaltered and SETUP moves on to the next parameter. If the user enters N then the default state is swapped to the other state and the user asked if this is correct, if not the process is repeated until it is.

## Drive Motor On Delay

The motor on delay specifies the length of time the BIOS must wait between turning on the drive motors and accessing the disc. The time is specified in 1/50 of a second periods. SETUP displays the current setting of the motor on delay as follows:-

```
Default motor on delay is ‹nn› 1/50
second units

Is this correct (Y/N):_
```

Where ‹nn› is the current motor on delay measured in 1/50 of a second periods. If the user enters Y in response to the above prompt then the motor on delay remains unaltered and SETUP moves on to the next parameter. If the user enters N then the following prompt is displayed:-

```
Enter new motor on delay in 1/50
second units:_
```

The user may then enter the required value of the motor on delay. When the new value has been entered it is displayed and the user asked if this is correct, if not the process is repeated until it is.

The recommended value for the DDI-1/FD-1 is 1 second, i.e. 50. A larger value will unnecessarily slow down the disc system, a smaller value will cause spurious disc errors.

## Drive Motor Off Delay

The motor off delay specifies the length of time the BIOS must wait between the last disc access and turning off the drive motors. The time is specified in 1/50 of a second periods. SETUP displays the current setting of the motor off delay as follows:-

```
Default motor off delay is ‹nn› 1/50
second units
Is this correct (Y/N):_
```

Where ‹nn› is the current motor off delay measured in 1/50 of a second periods. If the user enters Y in response to the above prompt then the motor off delay remains unaltered and SETUP moves on to the next parameter. If the user enters N then the following prompt is displayed:-

```
Enter new motor off delay in 1/50 second
units:_
```

The user may then enter the required value of the motor off delay. When the new value has been entered it is displayed and the user asked if this is correct, if not the process is repeated until it is.

The recommended value is 5 seconds, i.e. 250. The reason for leaving the motor on after a disc operation is to avoid the motor start-up delay in the event of another disc operation occurring shortly afterwards. However, leaving the motor on for long periods of time causes unnecessary drive wear. The time of 5 seconds is a compromise, the user may freely increase this if required.

## Stepping Rate

The stepping rate specifies the speed at which the disc drive head may be stepped across the disc. The time is specified in milliseconds. SETUP displays the current setting of the stepping rate as follows:-

```
Default stepping rate is ‹nn› milliseconds
Is this correct (Y/N):_
```

Where ‹nn› is the current stepping rate measured in milliseconds. If the user enters Y in response to the above prompt then the stepping rate remains unaltered and SETUP moves on to the next parameter. If the user enters N then the following prompt is displayed:-

```
Enter new stepping rate in milliseconds:_
```

The user may then enter the required value of the stepping rate. When the new value has been entered it is displayed and the user asked if this is correct, if not the process is repeated until it is.

The step rate should be 12 milliseconds for the DDI-1/FD-1 drives. Other drives may require a different step rate.

## Serial Interface Channel A Configuration

A number of parameters may be specified for the configuration of channel A of the serial interface. These parameters are:-

Tx baudrate - the default value of the transmit baud rate.

Rx baudrate - the default value of the receive baud rate.

Data bits - the default number of data bits

Parity - the default parity setting

Stop bits - the default number of stop bits

`SETUP` displays the default settings of the Z80 SIO CHANNEL A as follows:-

```
Z80 SIO Channel A: ‹AA› tx baudrate,
‹BB› rx baudrate, ‹CC› data bits
‹DD› parity, ‹EE› stop bits
Is this correct (Y/N):_
```

Where:
- ‹AA› is the default transmit baud rate
- ‹BB› is the default receive baud rate
- ‹CC› is the default number of data bits 5, 6, 7 or 8
- ‹DD› is the default parity setting Even, Odd or None
- ‹EE› is the default number of stop bits 1, 1.5 or 2

The following baud rates are supported:-

| | | | | | |
|---|---|---|---|---|---|
| 19200 | 9600 | 4800 | 3600 | 2400 | 2000 |
| 1800 | 1200 | 600 | 300 | 200 | 150 |
| 110 | 75 | 50 | | | |

If the user enters `Y` in response to the above prompt then the default configuration of channel A remains unaltered and `SETUP` moves on to the next parameter. If the user enters `N` then the following prompt is displayed:-

```
Enter Channel A parameters:_
```

The user may then enter the required configuration parameters in the order defined above. If any of the parameters are invalid or out of range then a suitable error message is output and the user must enter all the parameters again. When the new values have been successfully entered they are displayed and the user asked if this is correct, if not the process is repeated until it is.

## Serial Interface Channel B Configuration

A number of parameters may be specified for the configuration of channel B of the serial interface expansion board. These parameters are:-

| | |
|---|---|
| Baud rate | - the default value of the transmit and receive baud rate. |
| Data bits | - the default number of data bits |
| Parity | - the default parity setting |
| Stop bits | - the default number of stop bits |

SETUP displays the default settings of the Z80 SIO CHANNEL B as follows:-

```
Z80 SIO Channel B: ‹AA› baudrate, ‹BB› data bits
‹CC› parity, ‹DD› stop bits
Is this correct (Y/N):_
```

Where: ‹AA› is the default Transmit and receive baud rate
‹BB› is the default number of data bits 5, 6, 7 or 8
‹CC› is the default parity setting Even, Odd or None
‹DD› is the default number of stop bits 1, 1.5 or 2

The following baud rates are supported:-

| | | | | | |
|---|---|---|---|---|---|
| 19200 | 9600 | 4800 | 3600 | 2400 | 2000 |
| 1800 | 1200 | 600 | 300 | 200 | 150 |
| 110 | 75 | 50 | | | |

If the user enters Y in response to the above prompt then the default configuration of channel B remains unaltered and all parameter setting is complete. If the user enters N then the following prompt is displayed:-

```
Enter Channel B parameters:_
```

The user may then enter the required configuration parameters in the order defined above. If any of the parameters are invalid or out of range then a suitable error message is output and the user must enter all the parameters again. When the new values have been successfully entered they are displayed and the user asked if this is correct, if not the process is repeated until it is.

## Updating the system disc

Once the user has setup all the parameters SETUP then issues the following prompt:-

```
Do you want to update your system disc (Y/N):_
```

If the user enters N in response to the above prompt then SETUP is abandoned, without altering the system disc. If the user enters Y then the disc in drive A is updated with the new parameter settings. The following prompt is then output:-

```
Do you want to restart CP/M (Y/N):_
```

If the user enters N to the above prompt then SETUP exits via a warm boot. If the user enters Y then SETUP initiates a complete power-up initialization of CP/M so that the newly configured parameters come into effect.

## SETUP Messages

SETUP contains the following messages:-

```
Failed to read configuration sector correctly
```

This message is output if SETUP fails to read the configuration sector correctly. After this message is output SETUP is abandoned.

```
Failed to write configuration sector correctly
```

This message is output if SETUP fails to write the configuration sector correctly. After this message is output SETUP is abandoned.

```
Failed to verify configuration sector correctly
```

This message is output if the data read back from the configuration sector is not the same as the data written to it. After this message is output SETUP is abandoned.

```
Sector buffer overflow
```

This message is output if there is insufficient space in the configuration sector to hold all the buffers, strings and tables. After this message is output SETUP is abandoned, without attempting to update the configuration sector.

```
The disc in drive A is not a system disc
```

This message is output when SETUP attempts to read the configuration sector from a non-system disc. After this message is output SETUP is abandoned.

```
Bad control character
```

This message is output if the user input a string which contained an up-arrow as the last character.

```
Bad command option (Enter A, D, C or F)
```

The message is output if the user entered an invalid command option when setting up the keyboard translation or expansion buffers.

```
Invalid number input
```

This message is output if SETUP is expecting a valid number to be input and one was not found or was not terminated by a delimiter (space, comma or end-of-line).

```
Keyboard translation table empty
```

This message is output if the user attempts to delete an entry in the keyboard translation table when no translations are set.

```
Keyboard translation table full
```

This message is output if the user attempts to add an entry in the keyboard translation table when the translation table is full. Note that space has been reserved for more than 80 entries, thus this message should never be produced.

```
Key numbers must be in the range Ø to 79
```

This message is output if the user specifies a key number which is not in the range 0 to 79.

```
Key codes must be in the range Ø to 255
```

This message is output if the user specifies a key translation code which is not in the range 0 to 255.

```
Keyboard expansion buffer empty
```

This message is output if the user attempts to delete an entry in the keyboard expansion buffer when no expansions are set.

```
Keyboard expansion buffer full
```

This message is output if the keyboard expansion buffer overflows.

```
Key tokens must be in the range Ø to 31
```

This message is output if the user specifies a keyboard expansion token that is not in the range 0 to 31.

```
Invalid command
(you may only specify CON:, RDR:, PUN: or LST:)
```

This message is output if the user specifies an illegal IOBYTE logical device.

```
CON: may only be assigned to
TTY:, CRT:, BAT:, or UC1:
```

This message is output if the user attempts to assign CON: to an illegal phsyical device.

```
RDR: may only be assigned to
TTY:, PTR:, UR1:, or UR2:
```

This message is output if the user attempts to assign `RDR:` to an illegal phsyical device.

```
PUN: may only be assigned to
TTY:, PTP:, UP1:, or UP2:
```

This message is output if the user attempts to assign `PUN:` to an illegal phsyical device.

```
CON: may only be assigned to
TTY:, CRT:, LPT:, or UL1:
```

This message is output if the user attempts to assign `LST:` to an illegal physical device.

```
Step rate must be in the range 1 to 32
```

This message is output if the user specifies a step rate that is not in the range 1 to 32.

```
Invalid baudrate (you may only specify
19200, 9600, 4800, 3600, 2400, 2000, 1800,
1200, 600, 300, 200, 150, 110, 75 or 50)
```

This message is output if the user specifies an illegal baud rate for one of the two SIO ports.

```
Invalid data bits
(you may only specify 5, 6, 7 or 8)
```

This message is output if the user specifies an illegal number of data bits for one of the two SIO ports.

```
Invalid parity
(you may only specify ODD, EVEN or NONE)
```

This message is output if the user specifies an illegal parity setting for one of the two SIO ports.

```
Invalid stop bits
(you may only specify 1, 1.5 or 2)
```

This message is output if the user specifies an illegal number of stop bits for one of the two SIO ports.

```
↑C...aborted
```

This message is output if the user types Control-C when SETUP is waiting for user input. After this message is output SETUP is abandoned.

```
Illegal message number: Program aborted
```

This message indicates an error in the execution of the SETUP program. It should not normally be produced.

# 4.11 BOOTGEN.COM and SYSGEN.COM

## Introduction.

The utilities BOOTGEN and SYSGEN are provided to permit reconfiguration of the system tracks on System discs, or the creation of system tracks on Vendor discs. System track reconfiguration may be caused by the need to distribute a non-standard sized CP/M system (made by MOVCPM) or a non-standard configuration (made by SETUP).

## Conversion of Vendor discs to System discs

Vendor discs are CP/M system discs with blank system tracks. They are intended for the distribution of applications software and require the addition of system track information before they can be used. It is feasible, using the FILECOPY utility, to transfer the contents of a Vendor disc to a suitable System disc thereby avoiding the necessity for ever using SYSGEN or BOOTGEN in this context. It is prudent NEVER to convert the original Vendor disc, but to convert a copy of it (having used DISCCOPY or COPYDISC).

The conversion is achieved by invoking each of the utilities once in turn.

BOOTGEN should be invoked first by simply typing BOOTGEN.

The utility will first ask for the source disc, from which it will read the bootstrap and configuration information - just type **[ENTER]** to use the System disc already in the drive, otherwise insert the required alternative system disc and type **[ENTER]**.

When the information has been read, BOOTGEN will then ask for the destination disc, you should now insert the vendor disc which you wish to convert.

You will need to insert a proper system disc when asked for a CP/M disc at the end of the program, since the disc you have written to only contains half of the required CP/M information.

In order to generate the second half of the CP/M information SYSGEN must be executed in exactly the same way, once it has been invoked by typing SYSGEN.

When you exit from this second utility, the vendor disc will now be correctly configured as a system disc.

## Distribution of Non-standard version of CP/M

The system track information is held in two different blocks on the reserved tracks of a system disc. One section contains the boot sector (i.e. what is executed on start-up) and the configuration sector (this contains information such as sign-on message, key translations and disc drive hardware parameters).

The boot sector is common to all system discs i.e. they do not vary if a different size CP/M is constructed. The configuration sector is peculiar to a particular application, and is set up independently. These sectors need not, therefore, be updated when altering the CP/M size on an existing disc.

The second section of system data contains the CCP and the BDOS, which do need to be reconfigured for a different size CP/M. The CCP and BDOS for a different size CP/M is created by the utility **MOVCPM** and is left in memory when that program has finished executing (see chapter 2.6). **SYSGEN** can regenerate system tracks either directly from this memory image, or from a file copy of it. Such a file copy of the memory is made by invoking the '**SAVE 34 ...**' as prompted by **MOVCPM**.

The **SYSGEN** program makes certain checks to ensure that the information about to be written to the system tracks is suitable.

To reconfigure the system tracks **SYSGEN** should be invoked in one of the two following formats:

```
SYSGEN *
SYSGEN ‹filename›
```

**SYSGEN** * generates system tracks from the memory image following a **MOVCPM** ‹nnn› *. Where ‹nnn› is the size of the CP/M system, see chapter 2.6. Note that the * parameter for **MOVCPM** is mandatory.

**SYSGEN** ‹filename› generates the tracks from the relevant records in the file specified. ‹filename› cannot contain any wildcards.

## Distribution of non-standard Configurations

The **SETUP** utility provides a method of customizing the Configuration sector. It is possible to transfer a configuration sector from one disc to another by running the **SETUP** utility making no changes and saving the result to the required destination disc. It is somewhat simpler to run the **BOOTGEN** utility which will copy the bootstrap sector (the same on all discs) and the required configuration sector. The utility will prompt for source and destination discs.

## Summary

| | |
|---|---|
| **BOOTGEN** | copies Boot and configuration sectors. |
| **SYSGEN** | copies CCP and BDOS. |
| **SYSGEN** * | saves CCP and BDOS from TPA image. |
| **SYSGEN** ‹filename› | writes CCP and BDOS from **SAVE**d file of TPA image. |

## BOOTGEN and SYSGEN Messages

```
Please insert SOURCE disc into drive A
and press any key
Please insert DESTINATION disc into drive A
and press any key
```

Generated when the program requires access to a disc to either read a file or to read or write system tracks.

```
File not found
```

Generated when the SOURCE file (generally produced following a **MOVCPM**) that has been specified in the invocation line cannot be found on the currently selected user number of the inserted disc.

```
Loading
```

Indicates that the SOURCE file specified on the invocation line has been found on the disc and a load of the relevant part is being attempted.

```
‹filename› has incorrect format
```

The SOURCE file specified does not have sufficient records for it to contain the required information.

```
Ambiguous SOURCE filename
```

The filename specified contains wildcard characters and cannot therefore be correctly opened.

```
↑C...aborted
```

The user has pressed Control-C during execution of the utility and its operation has therefore been aborted.

```
MOVCPM memory image not present in TPA
```

The user is attempting to generate system tracks from data that has not either been created from **MOVCPM** or taken from existing system tracks.

```
SOURCE disc is not system format
DESTINATION disc is not system format
```

The user is trying to read/write the system tracks on a disc with format other than System and which therefore does not possess system tracks.

```
SOURCE disc has unknown format
DESTINATION disc has unknown format
```

The user is trying to read/write to a disc which is not of a standard AMSTRAD format.

```
SOURCE disc missing
DESTINATION disc missing
```

The program is trying to perform a disc operation when there is no disc present in the drive.

```
DESTINATION disc is write protected
```

Produced when the program is unable to write to a disc because it is write protected.

```
SOURCE disc error
DESTINATION disc error
```

A read/write operation has failed due to reasons other than those described above. This message should not appear during normal program execution.

```
Do you wish to reconfigure
another disc (Y/N)?:_
```

Appears when the DESTINATION disc system tracks have been correctly written. Note that any additional discs will use the same system track information as the first. In order to use different information the program will have to be re-executed.

```
Please insert a CP/M disc into drive A
then press any key:_
```

When the program is exited on completion or when aborted, this message is displayed indicating that a disc from which CP/M can be warm booted should be inserted.

BLANK PAGE

# Chapter 5 - Hardware

This chapter describes the disc interface in the DDI-1 with some remarks on using other drives and gives details of the recommended hardware configuration for the serial interface.

Note that the serial interface is not supplied with the DDI-1.

## 5.1 Disc Interface

### Floppy Disc Controller

The controller uses a NEC type μPD765A Floppy Disc Controller IC to connect to the disc drives. Only two disc drives are supported, since the US1 line from the μPD765A is ignored. This results in the two disc drives being accessed as drives 0 and 1 and again as 2 and 3. The controller supports both single and double sided and single and double density mini-floppy disc drives. Note that the clock frequency supplied to the μPD765A CLK pin is 4.00 MHz rather than the 8 MHz used with larger disc drives.

The full facilities described in the NEC data sheet for the μPD765A are available, with the exception of interrupts and DMA which are not supported.

The disc interface uses the Z80 I/O ports as follows:

| Port | Output | Input |
|---|---|---|
| #FA7E | Motor Control | ** not used ** |
| #FB7E | ** not used ** | μPD765A Status Register |
| #FB7F | μPD765A Data Register | μPD765A Data Register |

### Expansion ROM

The disc expansion ROM is on the interface board. The ROM is normally number #07, but may be set to number #00 by cutting option trace LK1. This board will ground the expansion bus EXP signal (pin 48) if the ROM number is #07 in order that this address can be avoided by other expansion peripherals. A 200 nanosecond 27128 type EPROM or ROM is normally used, and may be fitted in a DIL socket on the board.

Option traces LK2 and LK3 are manufacturing options for write pre-compensation. They should not need alteration by the user.

## Motor Control

Writing to this channel starts and stops the disc drive motors. Writing #00 will stop the motors, #01 will start the motors. On power-up and other system resets the motors are not stopped.

## Connector Type

The connector within this unit is a straight 34 way PCB mounting flat cable connector.

## Electrical Levels

All electrical levels on the controller are TTL compatable. Signals originating in the drives are terminated by 680 ohm resistors to +5v at the controller and received with gates with input hysteresis. The maximum permissible cable length is 0.75 metre.

## Pin Arrangement

All odd-numbered pins are ground. All signals are active low.

| Pin No. | Signal |
|---|---|
| 2 | +5v |
| 4 | +5v |
| 6 | +5v |
| 8 | Index |
| 10 | Drive Select 0 |
| 12 | Drive Select 1 |
| 14 | +5v |
| 16 | Motor On |
| 18 | Direction Select |
| 20 | Step |
| 22 | Write Data |
| 24 | Write Gate |
| 26 | Track 0 |
| 28 | Write Protect |
| 30 | Read Data |
| 32 | Side 1 Select |
| 34 | Ready |

## Using Other Disc Drives

It is possible to use other disc drives with the DDI-1, in particular 5 $^{1}/_{4}$ inch drives. Some hardware knowledge will be required. The following gives some advice and information which should assist in using a different drive.

If drive A: the 5V power should be supplied to pins 2,4,6 and 14 of the $5^1/_4$ inch drive after ensuring that any existing connections to the drive circuitry have been removed.

If Drive B: no terminating resistor should be installed.

The drive must have a 'ready' signal on pin 34.

The drive will require its own power supply.

The extra cabling should be as short as possible and should consist of a cable-mounting male connector (to mate with the female socket connector on the cable from the interface), all 34 conductors and, normally, a 34 way double-sided card edge connector (to mate with the $5^1/_4$ inch drive).

The step rate, motor on and off timeout may have to be changed, see the SETUP utility.

A drive such as the Shugart 201 is not suitable since it does not have a ready signal, but one such as the Chinon F 051-MD is suitable.

## 5.2 Serial Interface

The BIOS supports a two channel asynchronous serial interface. However, such an interface is not part of the DDI-1. This describes the 'Recommended Hardware Configuration' for the serial interface such that, if fitted, it can be driven by the BIOS.

The interface consists of a Zilog Z80A SIO/0 or Z80A DART together with an Intel 8253 programmable interval timer. The 8253 is used as a baud rate generator as follows:

Timer 0 generates the transmit clock for channel A of the SIO.
Timer 1 generates the receive clock for channel A of the SIO.
Timer 2 generates both the transmit and receive clocks for channel B of the SIO.

It is assumed that the CLK inputs to all three channels of this device are driven by a 2.0 (± 0.1%) MHz clock signal derived from the 4.0 MHz CPU clock. The GATE inputs to all three channels are tied permanently high.

The I/O ports are used as follows:

| Port | Output | Input |
|---|---|---|
| #FADC | SIO channel A data | SIO channel A data |
| #FADD | SIO channel A control | SIO channel A control |
| #FADE | SIO channel B data | SIO channel B data |
| #FADF | SIO channel B control | SIO channel B control |
| #FBDC | 8253 load counter 0 | 8253 read counter 0 |
| #FBDD | 8253 load counter 1 | 8253 read counter 1 |
| #FBDE | 8253 load counter 2 | 8253 read counter 2 |
| #FBDF | 8253 write mode word | ** not used ** |

BLANK PAGE

# Index to Disc Firmware Supplement

*AMSOFT and AMSTRAD welcome you to the world of the CPC464 computer - where value for money, perfomance and quality combine to provide a complete service to the personal computer user.*

# AMSOFT

169 Kings Road,
Brentwood,Essex
CM14 4EF

AMSOFT is pleased to hear if you have a program for the CPC464 system that you feel will be of interest to other owners -contact the Software Development Manager and submit a brief summary of your proposal, accompanied by a working example of the program - which will naturally be treated in confidence.