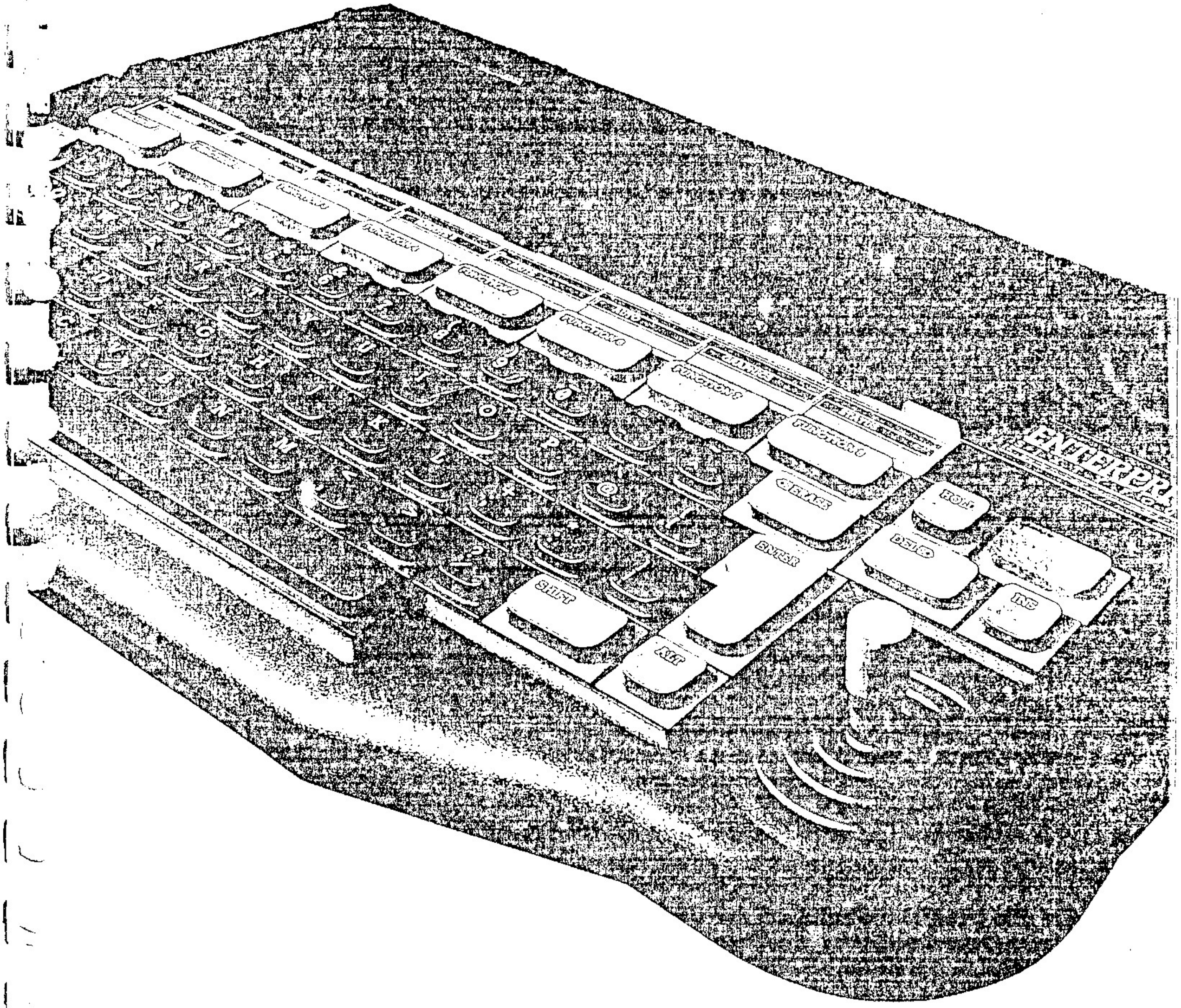
ENTERPRISE COMPUTERS



IS-DOS 1.0

The ENTERPRISE IS-DOS Manual

Version 1.0

Please read this manual carefully before using IS-DOS. This manual should be read in conjunction with your EXDOS manual.

Zaterdag
Kogelgieterij 33

EXDOS

NOM =17
VOLL =19

CONTENTS

PART 1 - USING IS-DOS

INTRODUCTION

IS-DOS is a disk-based disk operating system which is a powerful addition to your Enterprise computer. As well as providing useful utilities such as DISKCOPY, BACKUP and UNDELete, it allows the Enterprise to run the majority of application programs designed to operate under version 2.2 of the CP/M-80 operating system. The batch processing facilities of EXDOS are also enhanced, and EXDOS commands can be issued from IS-DOS. Like EXDOS, IS-DOS uses the MS-DOS file structure, and files fom both systems may be mixed on a single disk.

Any CP/M-80 program to be run on the Enterprise must be on an MS-DOS format disk; details are given later in this manual on how to configure programs to run under IS-DOS. It should be noted that certain programs 'modify' CP/M during installation; it is not possible to run these programs under IS-DOS. Certain programs have been tested by Enterprise, and a list of these appears in an appendix at the rear of this manual. We will update this list periodically by producing an 'applications note' which will be available on request. Obviously it is not possible for us to test all of the many CP/M programs available. We would ask users to supply us with any information they have on the operation (or non-operation) of specific packages that have not been tested by Enterprise.

(CP/M is a registered trademark of Digital Research and MS-DOS is a registered trademark of Microsoft.)

IMPORTANT -- Make sure your master IS-DOS disk is write protected; refer to your EXDOS manual for information on write protecting disks. You are advised to make a copy of your IS-DOS disk as soon as possible (see page 5).

STARTING IS-DOS

Your system should be set up as described in the EXDOS manual. We will assume you have just powered up the computer or performed a 'cold reset' (two quick presses on the RESET button). Press a key to enter BASIC. Insert the IS-DOS disk into drive A. Then, from the BASIC screen, type:

:ISDOS <ENTER>

IS-DOS will now load. The status line at the top of the screen will display 'IS-DOS', and you will be given the prompt:

A>

Any of the IS-DOS commands, as listed in the Operator's Reference Section of this manual, may now be used.

Once in IS-DOS, the screen will default to 80-column mode. This will be difficult to read if you are using a TV receiver. To obtain a 40-column screen, type:

MODE 40 <ENTER>

Now try typing:

DIR <ENTER>

to get the directory of the disk contents.

You will note that a number of files are present, including:

```
IS-DOS   SYS
BACKUP   COM
CHKDSK   COM
DISKCOPY COM
UNDEL    COM
XDIR     COM
```

The first of these, ISDOS.SYS, is the system file containing the IS-DOS operating system; this is the file that had to be loaded for IS-DOS to operate. The files with the suffix 'COM' are command files containing utility programs. We shall now briefly describe their use.

There are three kinds of command which IS-DOS recognizes: internal, external and transient.

To execute internal commands, IS-DOS does not need to load any additional program data from the system disk (or any other disk). The command ATTR is one example; you could remove the system disk from the drive, insert a different one, then type the following (which will convert all files on the disk into 'read only' files):

ATTR a: R <ENTER>

External commands are those which are passed, for implementation, to EXOS system extensions such as EXDOS. The commands described in the EXDOS manual come into this category. Any of them can be entered on the IS-DOS screen. Again IS-DOS has no need to load further program data.

A transient command, however, is treated as an instruction to load and execute a program contained in a disk file. On receiving the command, IS-DOS will search for a file with a name consisting of the command word plus the extension 'COM' or 'BAT'. The file is loaded and executed if found.

The Programmer's Reference Section explains to the machine code programmer how to configure his own programs as transient command files. The utilities supplied on the system disk, which we have listed, are also examples of transient command programs, and are labelled as such in the Operator's Reference Section.

Your first use of these utilities should be to make a backup copy of the system disk itself. Replace this disk in drive A (if you have removed it). If you have a multi-drive system, insert a new disk (or one containing no files that you want to preserve) into drive B. Then type:

BACKUP a: b:/F <ENTER>

This will execute the BACKUP.COM program. Notice that the command is the same as the file name -- without the suffix, but followed by an appropriate set of parameters. On receiving the command, IS-DOS searches for the file in the current directory of the logged-on drive. (It is possible to make it search other directories too; for this, see the PATH command in the Operator's Reference Section.)

If you are using a single-drive system, the computer will prompt you each time you need to change disks.

Note that if you wish to back up onto an Apricot disk, you should first format the disk using an Apricot.

* * *

You can exit from IS-DOS by typing (for example) BASIC <ENTER>, or WP <ENTER>. To return to IS-DOS later, you will have to re-load the system from disk as before. You may load it from a drive other than the default drive, by specifying (eg)

:ISDOS b: <ENTER>

If you type the command from the EXDOS screen, the initial colon should be omitted. For full information on the options when loading IS-DOS, see the description of the ISDOS command in the EXDOS manual.

USING BATCH FILES

We have seen that when a transient command is given to IS-DOS, a disk file with the corresponding name, plus the extension 'COM' or 'BAT', is searched for. If a COM file is not found, but a file with a BAT extension is present, the latter will be executed as a batch file.

A batch file consists of a series of DOS commands which are implemented as though they were typed one after the other at the keyboard. The EXDOS manual explains how batch files are produced.

When the computer is first powered up, or following a cold reset, an EXDOS batch file with the name EXDOS.INI is executed if found. This can be used to set up the machine configuration for the whole session.

Every time IS-DOS is entered (eg from BASIC), a file named AUTOEXEC.BAT is searched for, and executed if found. AUTOEXEC.BAT is therefore likely to be executed several times in one session.

The IS-DOS system disk contains an EXDOS.INI file (which prompts the user to enter the current date and time). It does not contain an AUTOEXEC.BAT file. You may find it useful to create one yourself -- for example, it could contain the 'MODE 40' command, to set up the 40-column screen automatically upon loading of IS-DOS.

IS-DOS batch files may contain commands for executing COM programs. In addition -- unlike EXDOS batch files -- they may have parameters passed to them.

As a simple example of this, you could create a batch file called FVID.BAT, containing the following single command:

```
VAR 90 %1
```

Then you would execute the file by typing its name followed by the desired parameter, eg:

```
FVID ON
```

This is equivalent to typing VAR 90 ON, and selects IS-DOS's fast video driver (see the VAR command in the Operator's Reference Section). FVID ON is more convenient, since you don't need to remember that 90 is the relevant EXOS variable number. On execution of the batch file, the parameter is automatically substituted for '%1'. Typing FVID by itself would be equivalent to VAR 90. The current status of the fast video driver (0 = 'ON', 255 = 'OFF') would then be displayed.

The number of parameters passed to a batch file can be anything up to 9. Suppose you have a file named MISC.BAT, and you execute it by typing its name followed by three parameters:

MISC 90 ON FRED

Wherever '%1' occurs in the batch file, the number 90 (the first parameter on the list) will be substituted. ON is substituted for %2, and FRED for '%3'.

For '%0', the name of the batch file itself is substituted, including any drive and path specification given in the command line which executes the file.

If the character following the '%' is not a number (0-9), it is taken as a literal character -- ie no special meaning is attached to it, and nothing will be substituted for it.

One IS-DOS batch file may contain instructions to execute other batch files -- which in turn may contain similar instructions. Normally, this 'nesting' of batch files can be continued to a maximum depth of 10, though this figure may be reduced if large numbers of parameters are passed. When the execution of one file ends, the file which invoked it is automatically resumed.

HELP FILES

The HELP command displays information about the system. If you type the word on its own, it lists the system extensions that are present. Alternatively, you may obtain information on some facility in particular, by typing (eg) HELP WP or HELP EXDOS.

Under IS-DOS, the HELP command may also be used to display text files that you have created yourself. Suppose you have a disk in drive B, containing these programs:

1ST-PROG.COM
TEST.COM
PATZER.COM

To go with each of these, you can create a text file giving information on the program to the user. First, you must make a sub-directory called HELP, in which the three files will be placed. The HELP sub-directory must be contained in the root directory of the disk. So type:

MKDIR B:\HELP <ENTER>

Now go into the Word Processor, and type the text for one of your 'help' files. For example: --

Patzer is a program that plays five-minute chess. It has the unique feature of allowing the user to type his moves in English descriptive notation.

Then press function key 3 ("PRINT"), and enter the path and filename for the help file:

B:\HELP\PATZER.HLP <ENTER>

You can now similarly create the text files 1ST-PROG.HLP and TEST.HLP. Help files are created in just the same way as batch files. Their names must always be followed by the 'HLP' extension.

Now type:

HELP B: <ENTER>

This specifies the drive in which help files will subsequently be searched for. Afterwards, you can type (eg)

HELP PATZER <ENTER>

-- to see the text of the corresponding file printed on the screen.

The IS-DOS disk contains help files relating to all standard IS-DOS commands. So if this disk is in drive A, and you now want information on (say) the MODE command, type:

HELP A: <ENTER>
HELP MODE <ENTER>

See also the explanation of the HELP command in the Operator's Reference Section.

PART 2 - OPERATOR'S REFERENCE SECTION

NOTATION

The following notation is used in the detailed description of commands: --

Words in upper case

These are keywords and must be entered as shown, in any mixture of upper or lower case.

Items in lower case

These are parameters which must be supplied to the command at this point in the command line.

Items in square brackets ('[' and ']')

These are optional items. The brackets themselves should not be included in the command line.

Items separated by a vertical bar ('|')

This indicates that one, and not more than one, of the items must be given. The vertical bar itself should not be included in the command line.

* * *

The following is a list of items which can appear on a command line:

d:

This indicates that a drive name is required (A:, B: etc). If no drive is specified, the default of the current logged-on drive is assumed.

Valid filename designators are A:...Z:, although only drives A:, B:, C: and D: would be supported by the standard disk interface hardware. Drive E: is the standard RAM disk.

path

This indicates that a directory path is required, each directory name being separated from the next by a backslash '\'. A backslash at the start of the path indicates that the path starts at the root directory, otherwise it starts at the current directory.

Two consecutive dots '..' signify the immediate parent directory in the path. A single dot '.' signifies the current directory in the path, and therefore has no value in a path specification. In both cases backslashes must be used to indicate that the dots refer to a directory.

For compatibility with keyboard or character set configurations which do not support the '\' character, a single inverted comma (') may be used in its place when specifying a path.

The syntax of directory names follows that for filenames given below.

filename

This indicates that the name of a file is required. An ambiguous filename is one that contains '*' or '?' characters and may match more than one file on disk, whilst one that does not contain these is an unambiguous filename.

A filename has the following syntax:

    filespec.suf

-- where 'filespec' is a sequence of up to eight characters, and 'suf' is an optional suffix of up to three characters. If the suffix is given, it must be separated from the filespec by a single dot '.'.

The following characters cannot be used in the filespec or suffix: --

    control codes and SPACE (in range 00h-20h)

    : ; . , = + \ < > ! / " [ ] # !

Normal alpha characters in filenames are turned into upper case by IS-DOS, and therefore lower and upper case characters have the same meaning.

Frequently a filename follows a path, in which case the two must be separated by a backslash.

volname

This indicates that a volume name should be given. A volume name is a sequence of up to eleven characters, which can include the characters not valid for filenames, with the exception of control codes and '/' (although leading spaces will be deleted).

device

A named input/output port, such as LPT:.

GENERAL RULES

The parameters of a command must be separated from the command itself by at least one space or tab character (these two characters generally being treated as equivalent by the command interpreter). Where two filenames are required as parameters, they too must be separated by a space or tab. Options given after a filename (after a '/' character) need not be separated in this way.

Although this is not a command as such, the currently logged-on drive can be changed by giving the instruction:

```
d:
```

-- which causes drive d: to become the currently logged-on drive, even if it does not actually exist.

In the command examples, underlined text is an example response to a command. Comments that are not part of the command are given in parentheses ('(' and ')').

DETAILED DESCRIPTION OF COMMANDS

ASSIGN

Format

ASSIGN [d: [d:]]

Purpose

Sets up a logical to physical translation for two drive names.

Use

The first drive name is assigned to the second, neither of which need be drives which actually exist. If only one drive name is given, then this drive is re-assigned to itself. If no drives are given, all previous ASSIGN commands are cancelled.

Example

```
ASSIGN C: A:

-- This causes all access to drive C:  to go instead to
drive A:.
```

ATDIR

Format

ATDIR [d:] [path] [/H] [[-][+]H]

Purpose

Changes the attributes of directories, to make them hidden or not hidden.

Use

At least one of the first two parameters must be given, to specify the directories whose attributes are to be changed.

If the H or +H option is given, then the selected directories are marked as hidden, and will not be affected by other directory commands or shown by a DIR command unless a /H option is given with those commands. The -H option marks the selected directories as not hidden. Note that -H will not have any effect unless the /H option is also given.

Unlike files, directories cannot be made read only.

Examples

```
ATDIR BOOT H
ATDIR SOURCE? H
ATDIR BOOT/H -H
```

ATTR

Format

ATTR [d:] [path] [filename] [/H] [[-]|[+]H] [[-]|[+]R]

Purpose

Changes the attributes of files to make them hidden/not hidden and read only/not read only.

Use

At least one of the first three parameters must be given, to specify the files whose attributes are to be changed.

The /H option permits hidden files to have their attributes changed by the command.

If the H or +H option is given, then the selected files are marked as hidden, and will not be affected by most commands or be shown by the DIR command unless a /H option is given with those commands. The -H option marks the selected files as not hidden. Note that -H will not have any effect unless the /H option is also given.

If the R or +R option is given, then the selected files are marked as read only. The -R option marks the selected files as not read only (read/write).

Examples

```
ATTR FRED R
ATTR B:\BOOT\*.COM H
ATTR \SOURCE/H -H -R
```

BACKUP (transient)

Format

[BACKUP] source [dest] [/D] [/M] [/F] [/X] [/T] [/V]

Purpose

To make a backup copy of a disk or a sub-set of files on a disk.

Use

source consists of [d:] [path] [filename]
dest consists of [d:] [path]

If only one of the above is given, it is assumed to be the source, and the destination is taken to be the default drive. If neither source nor destination is specified, the program will prompt for both drives.

The drives specified in the command are the logical drive names. If ASSIGN has been used with the effect that both source and destination refer to the same physical drive, then an error will be given. Operations of MAPDISK continue as normal, with prompts issued by the system if the backup involves two disks utilizing the same actual drive.

If no filename is given for the source, or if an ambiguous filename is specified which includes a match with a directory name, then all descendant sub-directories will be included within the backup operation. (This is an advantage of BACKUP over the COPY command provided by EXDOS.)

It is not possible to specify names of files and directories to be created on the destination disk, as these are given the same names as on the source disk. Following the backup operation, the destination disk will have the same sub-directory structure as the source disk, starting from the paths specified (although specifying a path on the destination disk will cause an error if the /F option is given -- see below).

Files are copied by BACKUP irrespective of their file attributes (system, hidden or read only).

If the /D option is given, the backup operation is confined to files in the directory which the source path specifies -- ie files in descendant directories are not copied.

The /F option has the effect of deleting all files on the destination disk, irrespective of any file attributes, and formatting the disk if it is not already formatted as a valid DOS disk. This option is designed to facilitate creating backups of complete disks. The default operation preserves existing files on the destination disk.

If a file on the destination disk has a name matching that of a file being copied into the same directory, then the original file will be deleted if it does not have its read only attribute set. If the backup operation attempts to create a new directory with a name matching that of an existing file, then an error will be generated, and the backup process will continue without copying that directory or its descendant sub-directories.

The /M option is selected when the backup is used as a formal archive operation. Files backed-up when the /M option is set are marked to indicate that they have been archived. The next time that a backup of this disk takes place, the same files will not be copied if they have remained unchanged since the last backup /M operation. IS-DOS always removes the archive marking if a file is written to.

The /X option is used to exclude all warning prompts. If the destination disk contains data and the /F option was specified, a prompt is issued to prevent accidental overwriting of the wrong disk, unless suppressed by the /X option.

When backing up from one file to another, the attributes are not affected, and the destination file is given the same attributes as the old. Similarly, the destination file is given the same date and time values as the source -- unless the /T option is specified, when the new file is given the current date and time instead.

The /V option is used to verify that the data was written correctly.

BACKUP has the advantage over DISKCOPY (see later) that it will be quicker if the source disk is only partly full, and that it consolidates all files into contiguous blocks on the destination disk; it also does not promulgate the disk errors of the source disk, and can be used to backup onto media of a different format (number of tracks etc).

Note that this is a transient command, and must be loaded from disk.

Examples

```
BACKUP
BACKUP A: B:/X
BACKUP A:\DATA B:\DATABACK/D/M/P
```

BUFFERS

Format

BUFFERS [number]

Purpose

To display or change the number of disk buffers in the system.

Use

If the number is not given, then the number of disk buffers currently in the system will be displayed. Otherwise the number of buffers will be increased to the number specified. The number of buffers cannot be decreased.

The default number of buffers in the system is 3.

Examples

```
BUFFERS
3
BUFFERS 5
```

CD - see EXDOS manual (synonym for CHDIR)

CHDIR - see EXDOS manual

CHKDSK (transient)

Format

CHKDSK [d:] [/F]

Purpose

Checks the integrity of the disk.

Use

If d: is not specified, the default drive is checked.

CHKDSK searches the disk for errors in the disk data structures. It reports any errors found, and attempts to find means of correcting them, ensuring that the disk is in a valid state.

If the /F option is given, the corrections will be written to disk. Otherwise CHKDSK will report the corrections that could be made, but will not actually carry them out.

A prompt inquires whether lost disk space which CHKDSK has freed for use is to be converted into files.

Note that when a disk file is written to, but not closed by a command or program, it is possible for disk space to be 'lost'. It is therefore recommended that CHKDSK is run on disks regularly.

Note that this is a transient command, and must be loaded from disk.

Examples

CHKDSK
CHKDSK B:/F

CLS - see EXDOS manual

COPY - see EXDOS manual

DATE - see EXDOS manual

DEL - see EXDOS manual (synonym for ERASE)

DIR - see EXDOS manual

DISKCOPY (transient)

Format

DISKCOPY [d: [d:]] [/X]

Purpose

To copy an entire disk, without any file processing.

Use

The entire disk specified by the first drive is copied to the disk specified by the second. If only one drive is given, then that drive is copied on to the current drive. If neither drive is specified, the program will prompt for both drives.

If the destination disk is unformatted, then it will be automatically formatted to the same format as the source disk.

If the destination disk is already formatted, then a warning message is issued, giving the volume name of the destination disk if it is available. When the user has pressed a key to indicate that the data can be overwritten, the disk copying operation will proceed. If the destination disk has a format incompatible with the source disk, then it will be automatically re-formatted.

The /X option suppresses the warning prompt, and will overwrite data on the destination disk without pausing for confirmation.

The result of the diskcopy operation will be two disks which are identical, with the exception that the hidden volume identity used by IS-DOS for disk checking will be different. The volume name of each disk will be the same, and even disk faults (such as lost data areas) will have been copied on to the destination disk.

Note that this is a transient command, and must be loaded from disk.

Examples

```
DISKCOPY
DISKCOPY A: B:/X
```

ECHO - see EXDOS manual

ERA - see EXDOS manual (synonym for ERASE)

ERASE - see EXDOS manual

EXDOS - see EXDOS manual

FORMAT - see EXDOS manual

HELP

Format

HELP d: | <name>

Purpose

Provides help text for a command.

Use

If <name> is a HELP command recognized by an EXOS system extension, then the standard help text for that extension is printed in the normal way. If <name> is not recognized by an extension, then IS-DOS looks for a text file called <name>.HLP in a directory called \HELP. If this is found, it is printed on the screen, otherwise an error is given. The drive used for looking for the .HLP file is initially the drive from which IS-DOS was loaded.

If d: is given instead of <name>, subsequent HELP commands will search for .HLP files in the \HELP directory in the specified drive.

Examples

```
HELP
HELP BASIC
HELP PATH
HELP C:
```

ISDOS - see EXDOS manual

LOAD - see EXDOS manual

MAPDISK

Format

MAPDISK d: d:

Purpose

To allow more than one disk to be used with one drive.

Use

Accesses to the first drive will cause the disk to be accessed in the actual drive specified by the second drive name, which must exist. If a disk change is required, then IS-DOS will issue a prompt. This allows a single drive system to behave like a multi-drive system.

If the computer is able to detect that it is only a single drive system, then on start-up, IS-DOS will automatically perform an operation equivalent to MAPDISK B: A:.

Example

MAPDISK B: C:

MD - see EXDOS manual (synonym for MKDIR)

MKDIR - see EXDOS manual

MODE

Format

MODE [[80] | [40] | [Logical-device:=EXOS-device]]

Purpose

Changes the screen mode or I/O devices.

Use

If no parameters are given, then the current MODE settings are displayed.

If a number is given, then this must be 40 or 80, and the screen is changed to 40 or 80 column mode as appropriate.

If a Logical device is given, this must be LST:, AUX:, RDR: or PUN:. If no EXOS-device is given, then the specified logical-device is assigned with NULL:, a special EXOS device which does not output anywhere and returns 'end of file' for input.

Logical devices are:

LST:

-- the output device used for standard printer output. LST: defaults to the EXOS device PRINTER:.

RDR:

-- Special purpose auxiliary input device, which has no standard function but may be used by programs. By default, RDR: uses the device assigned to AUX: (see below).

PUN:

-- special purpose auxiliary output device, which has no standard function but may be used by programs. By default, PUN: uses the device assigned to AUX: (see below).

AUX:

-- accumulation of RDR: and PUN: devices. If AUX: is assigned to an EXOS device, this device is automatically assigned to both RDR: and PUN:. By default, AUX: uses the EXOS device SERIAL:.

Note that PUN: and LST: both create an EXOS channel for output when the MODE command is done, and RDR: and AUX: both open one for I/O. The channel is closed when another MODE command specifying the logical device is given.

Examples

```
MODE
MODE LST:
MODE 40
MODE LST:=SERIAL:
MODE PUN:=TAPE:MYFILE
MODE AUX:
```

MOVE - see EXDOS manual

MVDIR - see EXDOS manual

PATH

Format

PATH [item [;item]...]

Purpose

Displays or sets the transient command search path.

Use

item consists of [d:] [path]

The 'search path' is a list of directory paths which are searched, in the specified order, when the name of a transient command is given to IS-DOS for execution and is not found in the current directory. The search continues until the corresponding command file is found or the list is exhausted.

If no parameters are given with the PATH command, the list constituting the current 'search path' will be printed, with the items separated by semi-colons. If the PATH command includes parameters, it specifies the search path which is subsequently to be used.

If no drive name is specified for an item, the logged-on drive (at the moment when the transient command is given) will be assumed. Similarly, if an item includes no directory path, the current directory of the specified drive will be searched.

If no PATH command has been given, the default action when searching for command files is to search only the current directory of the current drive. Hence, the command

PATH ;

can be used to cancel a previous PATH command.

Other valid separators, apart from a semi-colon, are space, comma, / and =.

Examples

```
PATH
E:\; A:\; A:\UTIL; B:\

PATH ;
PATH A:\, A:\COM, B:
```

The effect of the last example is that when a transient command is subsequently given, the (then) current directory of the logged-on drive will be searched first, followed by the root directory of drive A:, then the sub-directory A:\COM, then the current directory for drive B:.

PAUSE - see EXDOS manual

PRINTER

Format

PRINTER [ON | OFF]

Purpose

To switch the printer on or off.

Use

If neither ON or OFF is given, the command toggles the printer between the two states.

When the printer is on, all screen output is echoed to the printer -- ie logical device LST: -- and the message PRINTER ON appears on the status line. Under default conditions, LST: utilizes the EXOS device PRINTER:. It may use other devices as specified by the MODE command.

The PRINTER command is also programmed into Function Key 8, so that at the A> prompt the key will toggle the orinter on/off. This may also work in some programs (those written for CP/M).

EXAMPLES

PRINTER
PRINTER ON

RAMDISK - see EXDOS manual

RD - see EXDOS manual (synonym for RMDIR)

REM - see EXDOS manual

REN - see EXDOS manual (synonym for RENAME)

RENAME - see EXDOS manual

RMDIR - see EXDOS manual

RNDIR - see EXDOS manual

SAVE

Format

SAVE [number] [K | P] [d:] [path] filename

Purpose

Saves part of the TPA.

Use

The number specifies the amount of TPA (Transient Program Area) memory to save. If the K is given, then the number specifies the number of Kbytes to save. If the P is given, then number specifies the number of 256-byte pages to save (as in the CP/M SAVE command). Otherwise, number is the actual number of bytes to save.

The filename specifies the destination file.

Note that the TPA memory may be corrupted by certain commands which require the memory, such as COPY.

Examples

```
SAVE 1 TEST
SAVE 10K B:\UTIL\NEW.COM
```

TIME - see EXDOS manual

TYPE - see EXDOS manual

UNDEL (transient)

Format

UNDEL [d:] [path] [filename]

Purpose

Undeletes a previously deleted file.

Use

The files in the default or specified drive and path are undeleted if possible, the filename defaulting to *.*.

Files can only be undeleted if they have been deleted using IS-DOS, and if no disk allocation has taken place since the file was originally deleted.

Each file and sub-directory reference within the directory specified by path will be undeleted if the file or sub-directory name matches the filename specified, and if undeletion is possible. UNDEL can therefore be used to restore a sub-directory removed with the RMDIR or RD commands; to restore the contents of that sub-directory, a further UNDEL command is required with the correct path specification.

Note that UNDEL is a transient command, and therefore must be loaded from disk.

Examples

```
UNDEL B:FRED.MAC
UNDEL A:\BOOT
```

VAR - see EXDOS manual

NOTE: Appendix F of the EXDOS manual lists some EXOS system variables which can be displayed or changed by giving the VAR command to EXDOS. When operating IS-DOS, you may, in addition, wish to use the following: --

VAR 69 [ON | OFF]

This determines the 'echo' state of IS-DOS batch files. If ON is specified (within a batch file or prior to executing it), each line in the file will be printed when implemented. The default is OFF.

VAR 83 [ON | OFF]

When OFF (the default) is given, some transient programs may overwrite part of IS-DOS in memory -- making it necessary to re-load IS-DOS from disk once the program terminates. If ON is specified, IS-DOS cannot be overwritten.

VAR 90 [ON | OFF]

Specifying ON enables IS-DOS's internal video driver to be used for screen output in place of the normal EXOS video channel. Screen operations will then be faster, though more memory will be used. The default is OFF. See the Programmer's Reference Section (4.48).

VAR 91 [ON | OFF]

If ON is specified (the default is OFF), the current directory will be printed as the standard DOS prompt -- for example, the prompt might be A:\UTIL> instead of A>.

This allows slightly less flexibility in the editing of lines previously entered. If you give a command in response to the A:\UTIL> prompt, then move the cursor back to the same line and re-press ENTER, IS-DOS will this time treat 'UTIL' as a command to execute a transient program file in the root directory of drive A.

A full list of EXOS variables which can be controlled by IS-DOS is given in part 6 of the Programmer's Reference Section.

VER

Format

VER

Purpose

Prints the IS-DOS version number.

VOL - see EXDOS manual

XDIR (transient)

Format

XDIR [d:] [path] [filename] [/H]

Purpose

Displays the names of all the files in a directory tree.

Use

The drive, path and ambiguous filename specify which files are to be listed. If the /H option is given, hidden files will also be listed.

This command is similar to DIR, but it displays not only the contents of one directory, but also the files within all descendant sub-directories. Thus, the command

XDIR A:\

will display a list of all the files and directories on drive A: (excluding hidden files).

This directory listing includes the attributes of files, but not their date and time values. The listing uses indentation of files to indicate their depth in the directory tree.

Note that this is a transient command, and must therefore be loaded from disk.

Examples

```
XDIR
XDIR B:\DATA
```

PART 3 - PROGRAMMER'S REFERENCE SECTION

1. INTRODUCTION

This section describes the environment which IS-DOS provides for transient programs on the Enterprise computer. It is intended as a guide for writing new programs to run under IS-DOS and also to assist in converting existing CP/M programs.

A transient program interfaces with the operating system primarily by making function calls. Under IS-DOS there are four distinct types of call which can be made. These are as follows:

1. EXOS calls using 'RST 30h'.
2. Direct calls to the EXDOS ROM (FISH or DISKIO).
3. IS-DOS function calls using 'CALL 5'.
4. CP/M compatible 'BIOS' calls.

IS-DOS itself is an EXOS application program, and a transient program is run effectively as as subroutine of IS-DOS. This means that a transient program has full use of the EXOS function calls as described in the separate EXOS technical documentation. However if the transient program wants eventually to return control to IS-DOS, care must be taken with how EXOS is used.

The second type of function call (direct calls to FISH or DISKIO in the EXDOS ROM) is mainly intended for specialist programs such as the utilities provided with IS-DOS. The details of FISH and DISKIO function calls are described in the separate EXDOS technical documentation and so are not included here.

The bulk of the Programmer's Reference Section describes in some detail the function calls of the third type -- IS-DOS function calls. These are designed to be compatible with CP/M 2.2 and MSX-DOS 1.0, with a few additional functions most of which are specific to the Enterprise/EXOS environment.

The fourth type of function call (CP/M compatible BIOS calls) is not described in detail, since it will normally only be used by converted CP/M programs. Certain points relating to the use of these calls are mentioned in section 2.4.

2. TRANSIENT PROGRAM ENVIRONMENT

2.1. ENTRY FROM IS-DOS

A transient program will be loaded at address 0100h and CALLed by IS-DOS with the stack pointer set to a small stack within IS-DOS. This stack is sufficient for making IS-DOS or EXOS calls, and for allowing EXOS interrupts. If more stack is required (as it normally will be) then the transient program must set up its own stack in the TPA (Transient Program Area).

The contents of the Z-80 registers when a transient program is entered are undefined. The first 256 bytes of RAM will have been set up with various parameters and code as described in section 2.3.

Interrupts are enabled when a transient program is entered and should generally be left enabled. IS-DOS function calls will generally re-enable interrupts if the transient program has disabled them.

2.2 RETURN TO IS-DOS

A transient program can terminate itself in any of the following four ways:

1. Returning, with the original stack pointer.
2. Jump to location 0000h.
3. IS-DOS function call 00h.
4. IS-DOS function call 80h.

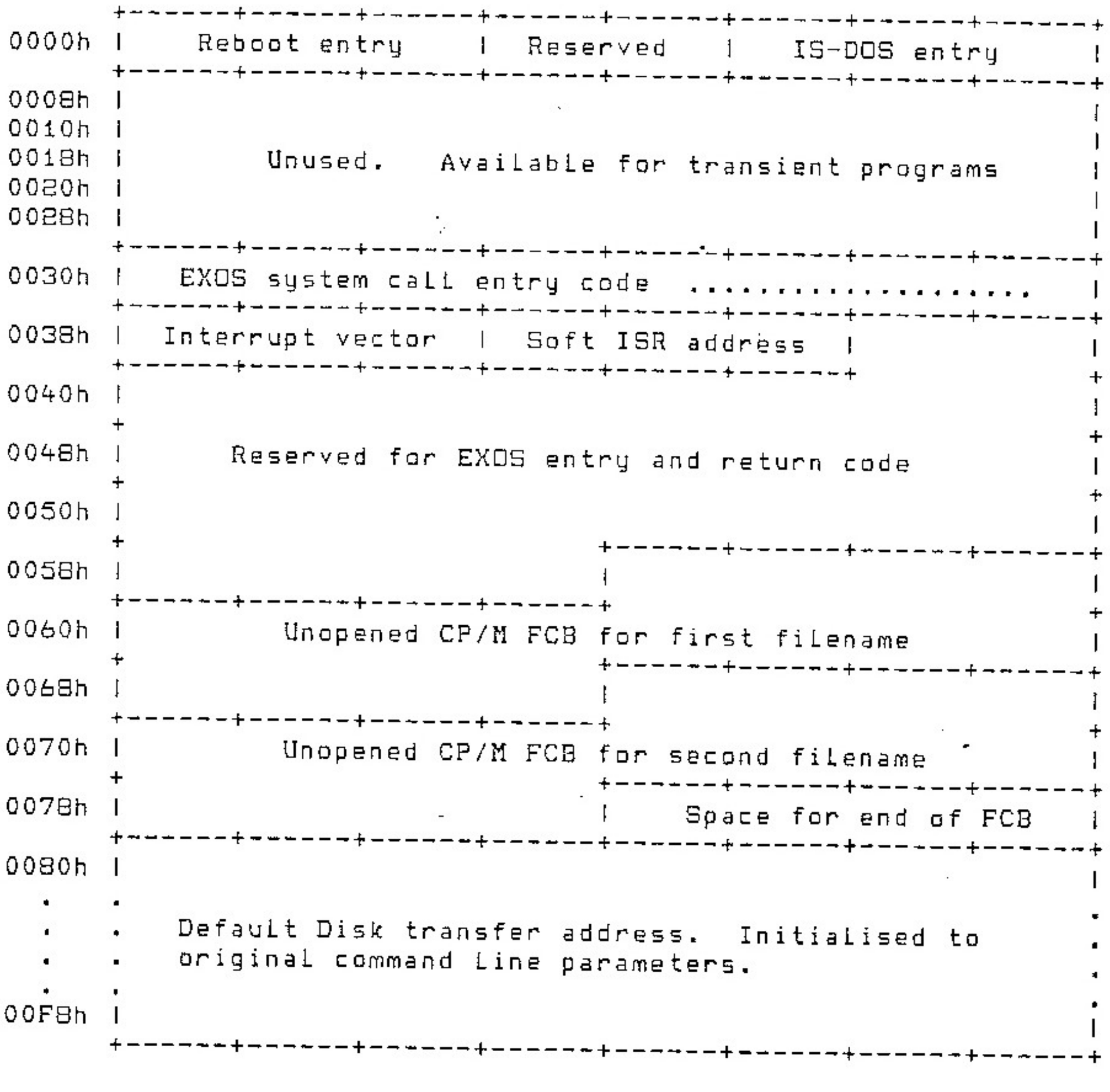
The first three of these methods are identical as far as IS-DOS is concerned, and are compatible with CP/M and MSX-DOS. The fourth method (function call 80h) is an addition which allows the program to return an error code to IS-DOS. The first three methods always return error code zero.

A transient program can also be terminated by the user typing "Ctrl-C" at the console (depending on the type of I/O being done), or by the user selecting "Abort" as the response to an "Abort/Retry/Ignore" message during a disk operation. In both of these cases an error code will be returned to IS-DOS.

A transient program can define an 'abort routine'. This will be called when the program is terminated by CtrL-C, by an abort after a disk error or by a function call 80h. The method of defining this routine, and its uses, are explained in section 4.44.

2.3 PAGE ZERO USAGE

On entry, various parameter areas are set up for the transient program in the first 256 bytes of RAM. The layout of this area is as below and is compatible with CP/M, apart from the area reserved for EXOS entry code.



At address 0000h is a jump instruction which can be used for terminating the transient program. The destination of this jump can also be used to locate the BIOS jump vector (see section 2.4). The low byte of this jump address will always be 03h for CP/M compatibility.

The two reserved bytes at addresses 0004h and 0005h are the IOBYTE and current drive/user in CP/M. IS-DOS keeps the current drive byte up to date. However the user number and IOBYTE are not supported since I/O redirection is handled separately and there is no concept of user numbers.

At address 0005h is a jump instruction to the start off the resident part of IS-DOS which is used for making IS-DOS calls. In addition the address of this jump defines the top of the TPA which the program may use. The size of the TPA depends on the amount of RAM in the system and also on whether the CLI (Command Line Interpreter) has been protected to force it to remain in memory. The maximum size TPA is 56k (0100h...E105h). The low byte of the destination of this jump will always be 06h for CP/M compatibility, and the six bytes immediately preceding it will contain the CP/M version number and a serial number.

The restart locations from 0008h up to 002Fh are available for use by transient programs. It is anticipated that debuggers will use address 0028h for breakpoints so this should normally be kept free.

The whole area from 0030h to 005Bh is reserved for EXOS entry and return code, and for interrupts -- and must not be modified by transient programs, apart from setting the software interrupt vector at address 003Dh. Note particularly that most CP/M debuggers (such as ZSID and DDT) use address 3Bh as a breakpoint entry. These programs will have to be modified and it is recommended that address 28h be used instead.

The two FCBs set up at addresses 005Ch and 006Ch are valid unopened FCBs containing the first two command line parameters interpreted as filenames. If both filenames are to be used, then the second one must be copied to a separate FCB elsewhere in memory, because it will be overwritten when the first one is opened. See section 3 for the format of FCBs.

The whole of the command line, with the initial command removed, is stored in the default disk transfer area at address 0080h, with a length byte first and a terminating null (neither the null nor the length byte are included in the length). This string will have been upper-cased and will include any leading spaces typed to ensure CP/M compatibility. Note that the null is not defined to be present in CP/M but is provided by IS-DOS as an added convenience.

2.4 BIOS JUMP TABLE

The jump at address 0000h will always jump to an address whose low byte is 03h. At this address will be another jump instruction which is the second entry in a seventeen entry jump table. This corresponds exactly to the BIOS jump table in CP/M 2.2.

The first eight entries in the table are for rebooting and for character I/O. These routines are implemented with the same specification as CP/M. The remaining jumps are low-level disk related functions in CP/M, and have no equivalent in IS-DOS since its filing system is totally different. These routines simply return without doing anything apart from corrupting the main registers.

IS-DOS switches to an internal stack while executing a BIOS call and so only a small amount of space (8 bytes) is required on the user's stack.

Note that although the jump table is always on a 256-byte page boundary, it is not the 'correct' distance above the top of the TPA (as defined by the contents of address 0006h) to correspond with CP/M 2.2. This should not matter to well-behaved CP/M programs, but it is rumoured that some programs rely on the size of the BDOS in CP/M 2.2. These programs will need modification.

The entries in the BIOS jump vector are as below:

```
xx00h - JMP   WBOOT     ;Warm boot
xx03h - JMP   WBOOT     ;Warm boot
xx06h - JMP   CONST     ;Console status
xx09h - JMP   CONIN     ;Console input
xx0Ch - JMP   CONOUT    ;Console output
xx0Fh - JMP   LIST      ;List output
xx12h - JMP   PUNCH     ;Punch (auxilliary) output
xx15h - JMP   READER    ;Reader (auxilliary) input
xx18h - JMP   RETURN    ;Home in CP/M
xx1Bh - JMP   RETURN    ;Select disk in CP/M
xx1Eh - JMP   RETURN    ;Set track in CP/M
xx21h - JMP   RETURN    ;Set sector in CP/M
xx24h - JMP   RETURN    ;Set DMA address in CP/M
xx27h - JMP   RETURN    ;Read sector in CP/M
xx2Ah - JMP   RETURN    ;Write sector in CP/M
xx2Dh - JMP   LSTST     ;Always returns A=0 (not ready)
xx30h - JMP   RETURN    ;Sector translate in CP/M
```

2.5 CHARACTER INPUT and OUTPUT

An IS-DOS transient program has two major ways in which it can do "console" input and output. These are by making EXOS calls to the default channel (channel number 255) or by making CP/M compatible IS-DOS or BIOS calls. Generally a program should use one or the other of these methods and not try to mix them since this can have unpredictable results.

The default EXOS channel will normally be an editor channel, although if printer echo is enabled it will be a channel to a special internal device which directs its output to the editor and to the printer. This is the channel which the IS-DOS CLI uses for its input and output. All the supplied transient programs (DISKCOPY, BACKUP, CHKDSK, UNDEL and XDIR) also use this channel.

IS-DOS console input and output functions use the EXOS keyboard and video channels directly without going through the editor channel. This is necessary because the editor is a line orientated device and CP/M input and output is character orientated. When the first character output function call is made, the screen is cleared to ensure that the old text from the editor will not interfere. The output produced by the program is displayed on the screen but will not be in the editor's buffer; so when the program terminates, the text, though visible, cannot be edited in the normal way or scrolled back after going off the screen.

When characters are written using IS-DOS character output function calls, they are first translated from VT52 control codes and escape sequences to EXOS video driver ones. This allows CP/M programs which are installed for VT52 terminals to be used directly without re-installation. IS-DOS also contains a built in VT52 driver which bypasses the normal EXOS video driver and accesses screen memory directly. This is intended for use where speed of screen access is particularly important -- eg for word processors. These programs can be installed to select the fast video driver automatically when they start up. Details of the IS-DOS function call to do this are in section 4.48.

The advantage of using EXOS calls is that the facilities of the EXOS screen editor are available -- and provide a user interface to the program which will feel like the IS-DOS command interpreter and other Enterprise programs. The disadvantages are that it is not CP/M compatible and is rather slower at output; generally it is necessary for a program to buffer output and block write it a line at a time to the default channel to improve the speed.

The advantages of using IS-DOS function calls are compatibility with CP/M and some speed improvement. However, the environment will not 'feel' like the IS-DOS CLI or other Enterprise programs such as IS-BASIC. Also the screen will be cleared when the program is entered, thus breaking the continuous flow of commands on the screen.

3. FILE CONTROL BLOCKS (FCBs)

ALL IS-DOS function calls which refer to disk files must be passed a pointer to a file control block (FCB) in register DE. These FCBs are identical to CP/M FCBs except that some of the reserved fields are used differently, reflecting the different filing system which IS-DOS uses.

A basic FCB is 33 bytes long. This type of FCB can be used for file management operations (delete, rename etc) and also for sequential reading and writing. The random read and write functions use an extra 3 bytes on the end of the FCB to store a random record number. The MSX-DOS compatible block read and write functions also use these additional three (or in some cases four) bytes -- see the function descriptions for details.

The layout of an FCB is given below. A general description of each of the fields is included here. The individual function descriptions later on give details of how the field are used for each function where this is not obvious.

00h Drive number 1...26. 0 => default drive. Must be set up in all FCBs used, never modified by IS-DOS function calls.

01h...08h Filename, left justified with trailing blanks. Can contain '?' characters if ambiguous filename is allowed (see function descriptions). When doing comparisons, case will be ignored, and so will bit-7. When creating new files, name will be uppercased but bit-7 will be left unchanged.

09h...0Bh Filename extension. Identical to filename. Note that bit-7 of the filename extension characters are not interpreted as flags as they are in CP/M.

0Ch Extent number (low byte). Must be set (usually to zero) by the transient program before open or create. It is used and updated by sequential read and write, and also set by random read and write. This is compatible with CP/M and MSX-DOS.

0Dh Not used. Zeroed by "open" and "create".

0Eh Extent number (high byte) for CP/M functions. Zeroed by open and create. For sequential read and write it is used and updated as an extension to the extent number, to allow larger files to be accessed. Although this is different from CP/M it does not interfere with CP/Ms use of FCBs and is the same as MSX-DOS.
Record size (low byte) for MSX-DOS block functions. Must be set to the required record size before using the block read or write functions.

15 0Fh Record count for CP/M functions. Set up by open and create and modified when necessary by sequential and random reads and writes. This is the same as CP/M and MSX-DOS.
Record size (high byte) for MSX-DOS block functions. Must be set to the required record size before using the block read and write functions.

16 - 17

10h...13h File size in bytes, lowest byte first. File size is exact, not rounded up to 128 bytes. This field is set up by open and create and updated when the file is extended by write operations. Should not be modified by the transient program, as it is written back to disk by a close function call. This is the same as MSX-DOS but different from CP/M which stores allocation information here.

18 - 22

14h...17h Volume-id. This is a four-byte number identifying the particular disk which this FCB is accessing. It is set up by open and create, and is checked on read, write and close calls. Should not be modified by the program. Note that this is different from MSX-DOS which stores the date and time of last update here, and from CP/M which stores allocation information.

23 31

18h...1Fh Internal information. These bytes contain information to enable the file to be located on the disk. Should not be modified at all by the transient program. The internal information kept here is similar but not identical to that kept by MSX-DOS and totally different from CP/M.

32

20h Current record within extent (0...127). Must be set (normally to zero) by the transient program before first sequential read or write. Used and modified by sequential read and write. Also set up by random read and write. This is compatible with CP/M and MSX-DOS.

33 36

21h...24h Random record number, low byte first. This field is optional; it is only required if random or block reads or writes are used. It must be set up before doing these operations, and is updated by block read and write but not by random read or write. Also set up by the "set random record" function.
For the block operations, which are in MSX-DOS but not in CP/M, all four bytes are used if the record size is less than 64 bytes, and only the first three bytes are used if the record size is 64 bytes or more. For random read and write, only the first three bytes are used (implied record size is 128 bytes). This is compatible with CP/M and with MSX-DOS.

4. IS-DOS FUNCTION CALLS

IS-DOS function calls are made by putting the function code in register C, with any other parameters in the other main registers, and then executing a 'CALL 5' instruction. The operation will be performed and results will be returned in various registers depending on the function.

Generally all main registers (AF, BC, DE, HL) are corrupted by IS-DOS calls, but the index registers (IX, IY) and alternate register set (AF', BC', DE', HL') are always preserved. Only a small amount (8 bytes) is needed on the transient program's stack because IS-DOS switches to an internal stack when it is called.

For compatibility, all functions which have a CP/M counterpart return with registers A=L and B=H. Frequently register A will be a sucess code, with zero indicating sucess and 01h or FFh indicating failure. All functions return with the flags set by an 'OR A' instruction, even when register A is undefined on return.

Below is a list of all the IS-DOS function calls. Any functions not in this table will simply return with registers A, B and HL set to zero. 'CPM' indicates that the function is at least broadly CP/M compatible, and 'MSX' indicates MSX-DOS compatibility.

00h	Re-boot IS-DOS	CPM	MSX
01h	Console input	CPM	MSX
02h	Console output	CPM	MSX
03h	Auxiliary input	CPM	MSX
04h	Auxiliary output	CPM	MSX
05h	List output	CPM	MSX
06h	Direct console I/O	CPM	MSX
07h	Direct console input		MSX
08h	Console input without echo		MSX
09h	String output	CPM	MSX
0Ah	Buffered Line input	CPM	MSX
0Bh	Console status	CPM	MSX
0Ch	Get CP/M version number	CPM	MSX

13	0Dh	Disk reset (flush buffers)	CPM	MSX
14	0Eh	Select disk	CPM	MSX
15	0Fh	Open file	CPM	MSX
16	10h	Close file	CPM	MSX
17	11h	Search for first	CPM	MSX
18	12h	Search for next	CPM	MSX
19	13h	Delete file	CPM	MSX
20	14h	Sequential read	CPM	MSX
21	15h	Sequential write	CPM	MSX
22	16h	Create file	CPM	MSX
23	17h	Rename file	CPM	MSX
24	18h	Get login vector	CPM	MSX
25	19h	Get default drive	CPM	MSX
26	1Ah	Set disk transfer address	CPM	MSX
27	1Bh	Get allocation information		MSX
28	1Ch	Unused (write protect in CP/M)		
29	1Dh	Unused (get R/O vector in CP/M)		
30	1Eh	Unused (file attributes in CP/M)		
32	20h	Unused (get/set user code in CP/M)		
33	21h	Random read	CPM	MSX
34	22h	Random write	CPM	MSX
35	23h	Get file size	CPM	MSX
36	24h	Set random record	CPM	MSX
37	25h	Unused (reset drive in CP/M)		
38	26h	Block write		MSX
39	27h	Block read		MSX
40	28h	Random write with zero fill	CPM	MSX
41	29h	Unused (not used in CP/M)		
42	2Ah	Get date		MSX
43	2Bh	Set date		MSX
44	2Ch	Get time		MSX
45	2Dh	Set time		MSX
46	2Eh	Set/reset verify flag		MSX
47	2Fh	Absolute disk read		MSX
48	30h	Absolute disk write		MSX
128	80h	Terminate with error code		
129	81h	Define abort routine		
130	82h	Allocate segment		
131	83h	Free segment		
132	84h	Get uncased command line		
133	85h	Select fast video driver		
134	86h	Translate logical/physical drive		
135	87h	Get FISH stack pointer		
136	88h	Get IS-DOS version number		

4.1 FUNCTION 00h -- REBOOT IS-DOS

Parameters: None
Results: Does not return

This function call is equivalent to a 'JMP 0000h'. It will return control to the IS-DOS command interpreter, which will be reloaded from disk if necessary. A zero error code will be returned to IS-DOS unless the user's abort routine has been called (see section 4.44).

4.2 FUNCTION 01h -- CONSOLE INPUT

Parameters: None
Results: L=A = Character from keyboard

A character will be read from the keyboard and echoed to the screen. If no character is ready then this function will wait for one. The characters 'Ctrl-S', 'Ctrl-P' and 'Ctrl-C' will be intercepted to provide the normal CP/M control functions (pause, printer on/off and reboot) and will not be returned to the transient program. 'Ctrl-C' will give a prompt to the user before rebooting to allow him to change his mind.

4.3 FUNCTION 02h -- CONSOLE OUTPUT

Parameters: E = Character to be output
Results: None

The character passed in register E is written to the screen. Tabs are expanded to columns of eight characters. If the printer is enabled then the character is also written to the printer. A console status check is done to test for 'Ctrl-P', 'Ctrl-S' and 'Ctrl-C'. Certain control codes and escape sequences can be used to control the screen. These are described in section 5 and are based on the VT-52 standard.

4.4 FUNCTION 03h -- AUXILIARY INPUT

Parameters: None
Results: L=A = Input character

A character is read from the auxiliary input device. If no character is ready then it will wait for one. This device must be set up by use of the 'MODE' command before this function call will do anything.

4.5 FUNCTION 04h -- AUXILIARY OUTPUT

Parameters: E = Character to be output
Results: None

The character passed in register E will be written to the auxiliary output device. This device must be set up by use of the 'MODE' command before this function will do anything.

4.6 FUNCTION 05h -- LIST OUTPUT

Parameters: E = Character to be output
Results: None

The character passed in register E will be sent to the printer. The printer channel is opened by default to the 'PRINTER:' device but can be altered by the 'MODE' command. This allows a serial printer to be used, and can also be used to send printer output to a file. The same channel is used for console output which is echoed to the printer, and in this case TABs will be expanded to eight columns.

4.7 FUNCTION 06h -- DIRECT CONSOLE I/O

Parameters: E = 00h...FEh - character for output
= FFh - requests input
Results: A=L = input - 00h - no character ready
else input character
undefined for output

If E=FFh on entry then the keyboard will be examined for a character and 00h returned if no character is ready. If a character is ready then it will be returned in register A without being echoed and with no check for control characters.

If E<>FFh on entry then the character in register E will be printed directly to the console. TABs are not expanded and printer echo is not performed.

4.8 FUNCTION 07h -- DIRECT CONSOLE INPUT

Parameters: None
Results: L=A = Input character

This is the same as the input part of function 06h except that if no character is ready then it will wait for one. Like function 06h no echo or control charcater checks are done. This is provided for MSX-DOS compatibility; it is not available in CP/M, which uses this function number for 'get I/O byte'.

4.9 FUNCTION 08h -- CONSOLE INPUT NO ECHO

Parameters: None
Results: L=A = Input character

This function is identical to function 01h except that the input character will not be echoed to the screen. The same control character checks will be done. This is provided for MSX-DOS compatibility; it is not present in CP/M, which uses this function number for 'set I/O byte'.

4.10 FUNCTION 09h -- STRING OUTPUT

Parameters: DE = Address of string
Results: None

The characters of the string pointed to by register DE will be output using the normal console output routine (function call 02h). The string is terminated by '$' (ASCII 24h).

4.11 FUNCTION 0Ah -- BUFFERED LINE INPUT

Parameters: DE = Address of an input buffer
Results: None

DE must point to a buffer to be used for input. The first byte of this buffer must contain the number of characters which the buffer can hold (0...255). Characters are read from the keyboard and put in the buffer starting at (DE+2) until ENTER is typed. The number of characters entered, which does not include the CR itself, will be stored at (DE+1). If there is room in the buffer then the CR will be stored after the last character (this is like MSX-DOS and is different from, but compatible with, CP/M).

If the buffer becomes full then the console bell is rung for each character which cannot be stored. Simple editing is provided using the joystick and delete/erase/insert keys. Characters are echoed to the screen as they are typed, but if printer echo is enabled they are only echoed to the printer when ENTER is pressed. This ensures that what is printed is the final entered line not including any editing which may have been done.

4.12 FUNCTION 0Bh -- CONSOLE STATUS

Parameters: None
Results: L=A = 00h if no character ready
= FFh if a character is ready

A flag is returned in register A to indicate whether a character is ready. Checks for 'Ctrl-C', 'Ctrl-S' and 'Ctrl-P' will be done as for function 2.

4.13 FUNCTION 0Ch -- GET CP/M VERSION NUMBER

Parameters: None
Results: L=A = 22h
H=B = 00h

This function simply returns the CP/M version number which is being emulated. This is always version 2.2 in current systems. The IS-DOS and EXDOS version numbers can be found using function call 88h (see section 4.50).

4.14 FUNCTION 0Dh -- DISK RESET

Parameters: None
Results: None

Any data which is waiting in internal buffers is written out to disk. It is not necessary to call this function in order to allow a disk change as is the case with CP/M. The disk transfer address is also set back to its default value of 80h by this function.

4.15 FUNCTION 0Eh -- SELECT DISK

Parameters: E = Drive number. 0=A: 1=B: etc.
Results: L=A = Number of drives (1...26)

This function simply selects the specified drive as the default drive. The default drive is stored internally as an EXOS variable called 'DEF_UNIT'; see the EXDOS specification for details. The current drive is also stored at address 0004h for CP/M compatibility.

The number of drives is returned in register A for compatibility with MSX-DOS, although this feature is not present in CP/M. It is calculated by counting the number of set bits in the login vector (see function call 18h, section 4.25). Note that the system does not in general know how many drives are connected, only the number of drives for which unit handlers exist. This means that the system will always think it has at least five drives (four floppy drives and a single RAM-disk).

4.16 FUNCTION 0Fh -- OPEN FILE

```
Parameters:     DE = Pointer to unopened FCB
Results:      L=A = 0FFh if file not found
                  =   0  if file is found
```

The unopened FCB must contain a drive -- which may be zero to indicate the default drive -- and a filename and extension, which may be ambiguous. The current directory of the specified drive will be searched for a matching file, and if found it will be opened. Matching entries which are sub-directories or system files will be ignored, and if the filename is ambiguous then the first suitable matching entry will be opened.

The low byte of the extent number is not altered by this function, and a file will only be opened if it is big enough to contain the specified extent. Normally the transient program will set the extent number to zero before calling this function. The high byte of the extent number will be set to zero to ensure compatibility with CP/M.

The filename and extension in the FCB will be replaced by the actual name of the file opened from the directory entry. This will normally be the same as what was there before, but may be different if an ambiguous filename, or one containing lower case letters, was used.

The record count will be set to the number of 128-byte records in the specified extent, which is calculated from the file size. The file size field itself, the volume-id and the 8 reserved bytes will also be set up. The current record and random record fields will not be altered by this function; it is the program's responsibility to initialize them before using the read or write functions.

4.17 FUNCTION 10h -- CLOSE FILE

```
Parameters:     DE = Pointer to opened FCB
Results:      L=A = 0FFh if not succesful
                  =   0  if succesful
```

The FCB must have previously been opened with either an OPEN or a CREATE function call. If the file has only been read, then this function does nothing. If the file has been written to, then any buffered data will be written to disk and the directory entry updated appropriately. The file may still be accessed after a close, so the function can be regarded as doing an 'ensure' operation.

4.18 FUNCTION 11h -- SEARCH FOR FIRST

```
Parameters:  DE = Pointer to unopened FCB
Results:    L=A = 0FFh if file not found
                =   0  if file found.
```

This function searches the current directory of the drive specified in the FCB (default drive if FCB contains zero) for a file which matches the filename and extension in the FCB. The filename may be ambiguous (containing '?' characters), in which case the first match will be found. The low byte of the extent field will be used, and a file will only be found if it is big enough to contain this extent number. Normally the extent field will be set to zero by the program before calling this function. System file and sub-directory entries will not be found.

If a suitable match is found (A=0), then the directory entry will be copied to the DTA address, preceded by the drive number. This can be used directly as an FCB for an OPEN function call if desired. The extent number will be set to the low byte of the extent from the search FCB, and the record count will be initialized appropriately (as for OPEN). This means that the attributes byte in the directory entry will not be available because it will have been replaced by the extent number. The format of directory entries is described in the separate EXDOS specification.

If no match is found (A=0FFh), then the DTA will not be altered. In no case will the FCB pointed to by DE be modified at all. This function remembers sufficient information internally to allow it to continue the search with a 'SEARCH FOR NEXT' function.

In CP/M, if the drive number is set to '?' in this function, then all directory entries, allocated or free, will be matched. Also, if the extent field is set to '?', then any extent of a file will be matched. Both of these features are normally only used by special purpose CP/M programs which are generally specific to the CP/M filing system (such as 'STAT'). Neither feature is present in MSX-DOS 1.0 (although the documentation claims that they are). IS-DOS does not implement these features.

4.19 FUNCTION 12h -- SEARCH FOR NEXT

```
Parameters:      None
Results:     L=A = 0FFh if file not found
                 =   0  if file found.
```

This function must only be done after a 'SEARCH FOR FIRST' has been done. It continues the search to look for the next match with the filename. The results returned from this function are identical to 'SEARCH FOR FIRST', and all the same comments apply. The information used to continue the search is held internally within IS-DOS, and so the original FCB used in the 'SEARCH FOR FIRST' need not still exist.

4.20 FUNCTION 13h -- DELETE FILE

```
Parameters:      DE = Pointer to unopened FCB
Results:     L=A = 0FFh if no files deleted
                 =   0  if files deleted OK
```

ALL files in the current directory of the disk specified by the FCB, and which match the ambiguous filename in the FCB, are deleted. Sub-directories, system files and read-only files are not deleted. If any files at all are sucessfully deleted then this function returns with A=0. A return with A=FFh indicates that no files were deleted.

4.21 FUNCTION 14h -- SEQUENTIAL READ

```
Parameters:      DE = Pointer to opened FCB
Results:     L=A = 01h if error (end of file)
                 =   0  if read was succesful
```

This function reads the next sequential 128-byte record from the file into the current disk transfer address. The record is defined by the current extent (high and low bytes) and the current record. After succesfully reading the record, this function increments the current record and if it reaches 080h, sets it back to zero and increments the extent number. The record count field is also kept updated when necessary.

In CP/M and MSX-DOS, the error return (A=01h) is defined always to mean end of file. In IS-DOS there are various other errors which could result in this error return, but generally the transient program will not go far wrong if it assumes that this error means end of file.

Sequential read operations are done through a 2k buffer within IS-DOS in order to improve the speed of reading files. This is totally transparent to the transient program.

UnLike CP/M it is possible to have partially filled records, since the file size is not necessarily a multiple of 128 bytes. If this occurs then the partial record is padded out with zeroes when it is copied to the transient program's DTA address.

4.22 FUNCTION 15h -- SEQUENTIAL WRITE

```
Parameters:     DE = Pointer to opened FCB
Results:      L=A = 01h if error (disk full)
                  =  0  if write was succesful
```

This function writes the 128 bytes from the current disk transfer address to the file at the position defined by the current record and extent, which are then incremented appropriately. The record count byte is kept updated correctly if the file is extended or if the write moves into a new extent. The file size in the FCB is also updated if the file is extended.

In CP/M and MSX-DOS, the error return (A=01h) is defined always to mean 'disk full'. IS-DOS may return this error for various other reasons, but generally a program will not go too far wrong if it assumes that this error does mean disk full.

4.23 FUNCTION 16h -- CREATE FILE

```
Parameters:     DE = Pointer to unopened FCB
Results:      L=A = 0FFh if unsuccesful
                  =   0  if successful
```

This function creates a new file in the current directory of the specified drive and opens it ready for reading and writing. The drive, filename and low byte of the extent number must be set up in the FCB and the filename must not be ambiguous. Checks will be done to ensure that invalid filenames are not created.

If there is already a file of the required name, then the action depends on the value of the extent number byte. Normally this will be zero, and in this case the old file will be deleted and a new one created. However, if the extent number is non-zero then the existing file will be opened without creating a new file. This ensures compatibility with early versions of CP/M where each extent had to be explicitly created.

In all cases the resulting file will be opened with the required extent number exactly as if an 'OPEN' function call had been done.

4.24 FUNCTION 17h -- RENAME FILE

```
Parameters:     DE = Pointer to modified unopened FCB
Results:      L=A = 0FFh not if successful
                  =   0  if succesful
```

The modified unopened FCB has the normal drive and filename, and also a second filename starting at (DE+17). Every file in the current directory of the specified drive which matches the first filename is changed to the second filename -- with '?' characters in the second filename leaving the appropriate character unchanged. Checks are done to prevent duplicate or illegal filenames from being created. Entries for sub-directories and system files will not be renamed.

4.25 FUNCTION 18h -- GET LOGIN VECTOR

```
Parameters:          None
Results:       DE:HL = Login vector
```

This function returns a bit set in DE:HL for each drive which is available, bit-0 of L corresponding to drive 'A:'. In CP/M and MSX-DOS, register DE is undefined since these systems have a maximum of 16 drives.

Note that IS-DOS cannot tell whether a particular drive is actually connected; it can only tell whether there is a unit handler for that drive, after translating the logical drive number to a physical one. This means that normally it will always claim to have drives A, B, C, D and E corresponding to the four possible floppy drives and the RAM-disk. If any 'ASSIGN' or 'MAPDISK' commands have been done, this will affect which drives are available. If a bit for a particular drive is clear, then attempting to access that drive will produce an immediate error; whereas if the bit is set, then IS-DOS will try to access the drive and may then fail with an abort/retry message.

4.26 FUNCTION 19h -- GET DEFAULT DRIVE

```
Parameters:          None
Results:      L=A = Current default drive (0=A: etc)
```

This function just returns the current drive number.

4.27 FUNCTION 1Ah -- SET DISK TRANSFER ADDRESS

Parameters: DE = Required Disk Transfer Address
Results: None

This function simply records the address passed in DE as the disk transfer address. This address will be used for all subsequent read and write calls, and also for the search calls to store the directory entry. The address is set back to 80h by a 'DISK RESET' call (see section 4.14).

4.28 FUNCTION 1Bh -- GET ALLOCATION INFORMATION

Parameters: E = Drive number (1...26, 0=default)
Results: A = Sectors per cluster
BC = Sector size (always 512)
DE = Total clusters on disk
HL = Free clusters on disk

This function returns various information about the disk in the specified drive and is compatible with MSX-DOS. It is not compatible with CP/M, which uses this function number to return the address of an allocation vector.

4.29 FUNCTION 21h -- RANDOM READ

Parameters: 32, DE = Pointer to opened FCB
Results: L=A = 01h if error (end of file)
= 0 if read was succesful

This function reads a 128-byte record from the file to the current disk transfer address. The file position is defined by the three-byte random record number in the FCB (bytes 21h...23h). Unlike CP/M (but like MSX-DOS) all three bytes of the random record number are used. A partial record at the end of the file will be padded with zeroes before being copied to the user's DTA.

The random record number is not altered, so successive calls to to this function will read the same record unless the transient program alters the random record number. A side effect is that the current record and extent are set up to refer to the same record as the random record number. This means that sequential reads (or writes) can follow a random read, and will start from the same record. The record count byte is also set up correctly for the extent.

4.30 FUNCTION 22h -- RANDOM WRITE

```
Parameters: 34  DE = Pointer to opened FCB
Results:      L=A = 01h if error (disk full)
                  =  0  if no error
```

This function writes a 128-byte record from the current disk transfer address to the file, at the record position specified by the three-byte random record number (bytes 21h...23h). All three bytes of the random record number are used. If the record position is beyond the current end of file, then un-initialized disk space will be allocated to fill the gap.

The random record number field will not be changed, but the current record and extent fields will be set up to refer to the same record. The record count byte will be adjusted as necessary if the file is being extended or if the write goes into a new extent.

4.31 FUNCTION 23h -- GET FILE SIZE

```
Parameters: 35  DE = Pointer to unopened FCB
Results:      L=A = 0FFh if file not found
                  =   0  if file found OK
```

This function searches for the first match with the filename in the FCB, exactly the same as 'OPEN FILE' (see section 4.16). The size of the located file is rounded up to the nearest 128 bytes and the number of records determined. The three-byte random record field of the FCB is set to the number of records, so it is the number of the first record which doesn't exist. The fourth byte of the random record number is not altered.

4.32 FUNCTION 24h -- SET RANDOM RECORD

```
Parameters: 36  DE = Pointer to FCB
Results:      None
```

This function simply sets the three-byte random record field in the FCB to the record determined by the current record and extent number. The fourth byte of the random record number is not altered. No check is done as to whether the record actually exists in the file.

4.33 FUNCTION 26h -- BLOCK WRITE

Parameters: DE = Pointer to opened FCB
HL = Number of records to write
Results: A = 01h if error
= 0 if no error

This function is the preferred method of writing data to files for programs which do not have to be CP/M compatible. It is compatible with MSX-DOS, but in CP/M there is no corresponding function. Data can be written in large blocks which can be much faster than normal CP/M writes.

Data is written from the current disk transfer address to the position in the file defined by the random record number. The record size is determined by the record size field in the FCB (bytes 0Eh and 0Fh), which must be set by the user after opening the file and before calling this function. If the record size is less than 64 bytes, then all four bytes of the random record number are used; otherwise only the first three are used.

The number of records to be written is specified by HL, and together with the record size this determines the amount of data to be written. An error will be returned if the write attempts to go above the top of memory (0FFFFh), thus limiting the maximum size of a transfer.

After writing the data, the random record field is adjusted to the next record number in the file (ie HL is added on to it). The current record and extent fields are not used or altered. The file size field is updated if the file has been extended.

The record size can be any value from 1...0FFFFh. Small record sizes are no less efficient than large record sizes, so if desired the record size can be set to one and the record count then becomes a byte count. It is desirable to read as much as possible with one function call, since one large transfer will be quicker than several small ones.

The MSX-DOS version of this function has a special case if the number of records specified is zero. This is used to allow the file size to be explicitly changed, allocating or freeing disk space as required. This facility is not supported by IS-DOS.

4.34 FUNCTION 27h -- BLOCK READ

```
Parameters:   DE = Pointer to opened FCB
              HL = Number of records to read
Results:       A = 01h if error
                 =  0  if no error
              HL = Number of records actually read
```

This function is the complement of the 'BLOCK WRITE' function described above (see section 4.33), and most of the same comments apply as regards its use. Again, if large blocks are read, it will be much faster than the normal CP/M operation.

The number of records actually read is returned in HL. This may be smaller than the number of records requested if the end of the file was encountered. In this case any partial record will be padded out with zeroes before being copied to the user's DTA. The random record field is adjusted to the first record not read, ie the value returned in HL is added on to it.

4.35 FUNCTION 28h -- RANDOM WRITE WITH ZERO FILL

```
Parameters:    DE = Pointer to opened FCB
Results:     L=A = 01h if error
                 = 00h if no error
```

This function is identical to 'RANDOM WRITE' (see section 4.30), except that if the file has to be extended, any extra allocated disk clusters will be filled with zeroes before writing the data.

4.36 FUNCTION 2Ah -- GET DATE

```
Parameters:  None
Results:     HL = Year 1980...2079
              D = Month (1=Jan...12=Dec)
              E = Date (1...31)
              A = Day of week (0=Sun...6=Sat)
```

This function simply returns the current value of EXOS's internal calendar in an MSX-DOS compatible format. Note that the values returned are binary rather than BCD as is the case for the EXOS date and time function calls.

4.37 FUNCTION 2Bh -- SET DATE

```
Parameters: 43  HL = Year 1980...2079
                D  = Month (1=Jan...12=Dec)
                E  = Date (1...31)
Results:        A  = 00h if date was valid
                   = FFh if date was invalid
```

The supplied date is checked for validity; if it is valid, it is stored as the new EXOS date. The validity checks include full checking for the number of days in each month and leap years. If the date is invalid then the current date will be left unaltered.

4.38 FUNCTION 2Ch -- GET TIME

```
Parameters:     None
Results:  44    H = Hours (0...23)
                L = Minutes (0...59)
                D = Seconds (0...59)
                E = Centiseconds (always zero)
```

This function returns the current value of EXOS's internal clock in an MSX-DOS compatible format. The clock is updated by a 1Hz interrupt, and at midnight the date is incremented. Note that the centiseconds value returned is always zero because the resolution of EXOS's clock is only one second.

4.39 FUNCTION 2Dh -- SET TIME

```
Parameters:  45 H = Hours (0...23)
             L  = Minutes (0...59)
             D  = Seconds (0...59)
             E  = Centiseconds (ignored)
Results:     A  = 00h if time was valid
                = FFh if time was invalid
```

This function sets EXOS's internal system clock to the specified time value. If the time is invalid, then register A will be returned as 0FFh to indicate an error, and the current time will be left unaltered. The centiseconds value is included for MSX-DOS compatibility, but is ignored by IS-DOS since the resolution of EXOS's internal clock is only one second.

4.40 FUNCTION 2Eh -- SET/RESET VERIFY FLAG

```
Parameters: 46  E =  0 to disable verify
                 <> 0 to ensble verify
Results:        None
```

This function simply enables or disables automatic verification of all writes. It defaults to off when IS-DOS is started up.

4.41 FUNCTION 2Eh -- ABSOLUTE SECTOR READ

```
Parameters:  DE = Sector number
              L = Drive number (1=A: etc)
              H = Number of sectors to read
Results:      A = EXOS error code (0=> no error)
             DE = Next sector number
              H = Number of sectors read
```

This function reads sectors directly from the disk without interpreting them as files. The disk must be a valid DOS disk in order for the sector number to be translated into a physical position on the disk. Details of disk layout etc can be found in the separate EXDOS documentation.

4.42 FUNCTION 2Eh -- ABSOLUTE SECTOR WRITE

```
Parameters:  DE = Sector number
              L = Drive number (1=A: etc)
              H = Number of sectors to write
Results:      A = EXOS error code
             DE = Next sector number
              H = Number of sectors written
```

This function writes sectors directly to the disk without interpreting them as files. The disk must be a valid DOS disk in order for the sector number to be translated into a physical position on the disk. Details of disk layout etc can be found in the separate EXDOS documentation.

4.43 FUNCTION 80h -- TERMINATE WITH ERROR CODE

```
Parameters:  B = Error code for terminetion
Results:     Does not return
```

This function terminates the transient program and returns the specified error code to IS-DOS as the result of the program. A value of zero means success and any other value is an error. For all error codes in the range 20h...FFh, IS-DOS will display an error message, while for values 01h...1Fh no message will be displayed. Many useful error codes with associated messages are defined by EXOS and EXDOS (see separate documentation).

Note that if a user abort routine has been defined then this function will call the abort routine as if it had been called from the user program, and it must terminate with a normal 'JP 0' exit. The error code is remembered internally and will be returned to IS-DOS. See section 4.44 for more about the user abort routine.

4.44 FUNCTION 81h -- DEFINE ABORT ROUTINE

Parameters: DE = Address of abort routine
Results: None

Register DE contains the address of a routine which will be called by IS-DOS when the program is terminated. The routine will only be called if termination results from the user pressing 'Ctrl-C', an abort response from a disk error, or the program itself making a function call 80h (see section 4.43). It is not called for normal CP/M type exits (function call 00h, return or 'JP 0').

When the abort routine is called, the error code which is going to be returned to IS-DOS will be in register A, and all other main registers will be undefined. The abort routine itself can make any desired IS-DOS calls to tidy things up, but care should be taken to avoid recursion. When the abort routine has finished, it should exit with a 'JMP 0000h', which will finally transfer control to IS-DOS.

Most programs will not require an abort routine. It is provided particularly for EXOS specific programs which may have opened channels, since these must close them again before returning to IS-DOS.

4.45 FUNCTION 82h -- ALLOCATE SEGMENT

Parameters: None
Results: A = EXOS error code
C = Segment number

This is identical in function to the EXOS allocate segment call, and will allocate a user segment. However, IS-DOS remembers the segment numbers of any segments allocated with this function, and is thus able to free them when the program terminates. This relieves the program from the responsibility of freeing any allocated segments and more importantly ensures that segments will be freed even if the program is terminated by a warm reset. A maximum of 32 segments can be allocated by this function.

4.46 FUNCTION 83h -- FREE SEGMENT

Parameters: C = Segment number to free
Results: A = EXOS error code

This function is identical to the equivalent EXOS function call but it also deletes the specified segment number from IS-DOS's internal list of allocated segments. It is not necessary for a program to free segments, since this will be done automatically when IS-DOS re-boots, but it may be useful in some circumstances.

4.47 FUNCTION 84h -- GET UNCASED COMMAND LINE

Parameters: 122 None
Results: None

The command line passed to a transient program at 80h will have been uppercased by IS-DOS to ensure CP/M compatibility. However some programs may want to have their paramaters as typed by the user, for example a 'FIND' utility may want to search for an exact match to a lower-case string. This function call simply copies the original command line to the buffer at 80h, without upper-casing. The resulting command line will be exactly the same as it was when the program was entered, except for any upper/lower case differences.

4.48 FUNCTION 85h -- SELECT FAST VIDEO DRIVER

Parameters: 123 None
Results: A = EXOS error code

This function call selects IS-DOS's internal video driver to be used for screen output in place of the normal EXOS video channel. It is provided for programs such as WORDSTAR which can be installed to make this function-call and then gain a speed advantage for their screen operations. The fast video driver uses the same VT-52 control codes as the normal video driver. More details are given in section 5.

4.49 FUNCTION 86h -- GET FISH STACK POINTER

Parameters: 124 None
Results: DE = Address of stack in page-2

This returns the address of IS-DOS internal stack in the EXOS system segment (segment 0FFh) in page-2. It is intended for programs which do direct FISH or DISKIO calls since the stack must be in the system segment in page-2 when calling these routines.

4.50 FUNCTION 87h -- GET IS-DOS VERSION NUMBER

Parameters: 125 None
Results: A = CP/M version number
B = EXDOS version number
C = IS-DOS version number

This function simply returns the three version numbers as indicated above. Each is returned as a two-digit BCD number, so the CP/M version number is always 22h. This function can be used by a program to determine whether it is running under IS-DOS or CP/M, since CP/M will not return 22h in register A.

5. SCREEN OUTPUT

Below is a list of all control codes and escape sequences which may be used when doing console output by IS-DOS calls or BIOS calls. When the fast video driver is being used, these codes are handled directly. For the normal EXOS video driver, the codes are translated internally by IS-DOS. This ensures that the control sequences are identical, so that the same program can work with either driver. The only difference is that some of the colour pairs are different, and the fast video cursor flashes.

The screen is 24 lines of 80 characters. When a printing character is displayed, the cursor is moved to the next position, or, at the end of a line, to the start of the next line. If a character is written in the bottom right position, then the screen will be scrolled to allow the cursor to be positioned at the start of the next line. The letters in escape sequences must be in the correct case; the spaces are inserted for readability, and are not part of the sequence. Numbers (indicated by <n> or <m>) are included in the sequence as a single byte usually with an offset of 20h added.

Ctrl-G 07h = Bell.

Ctrl-H 08h = Cursor left, wraps around to previous line, stop at top left of screen.

Ctrl-I 09h = Tab, overwrites with spaces up to next 8th column, wraps around to start of next line, scrolls at bottom right of screen.

Ctrl-J 0Ah = Line feed, scrolls if at bottom of screen.

Ctrl-K 0Bh = Cursor home.

Ctrl-L 0Ch = Clear screen and home cursor.

Ctrl-M 0Dh = Carriage return.

Ctrl-Y 19h = Erase to end of line, don't move cursor.

Ctrl-[1Bh = ESC - see below for escape sequences.

Ctrl-\ 1Ch = Cursor right, wrap around to next line, stop at bottom right of screen.

Ctrl-] 1Dh = Cursor Left, wrap around to previous line, stop at top left of screen.

Ctrl-^ 1Eh = Cursor up, stop at top of screen.

Ctrl-_ 1Fh = Cursor down, stop at bottom of screen.

7Fh = Delete character and move cursor left, wrap around to previous line, stop at top of screen.

Esc A = Cursor up, stops at top of screen.

Esc B = Cursor down, stops at bottom of screen.

Esc C → = Cursor right, stops at end of line.

Esc D ← = Cursor left, stops at start of line.

Esc E = Clear screen and home cursor.

Esc d <n><m>= Scroll down lines <n> to <m> inclusive, leaving new line <m> blank. Lines are numbered starting with top line as 20h.

Esc E = Clear screen and home cursor.

Esc H = Cursor home.

Esc I = Cursor up, scrolls if at top of screen.

Esc i <n> = Select ink colour <n>.

	NORMAL VIDEO	FAST VIDEO
n=0	green on black	
n=2	red on black	
n=4	blue on white	black on green
n=6	black on red	black on red

Esc J = Erase to end of screen, don't move cursor.

Esc j = Clear screen and home cursor.

Esc K = Erase to end of line, don't move cursor.

Esc L = Insert a line above cursor line, scroll rest of screen down. Leave cursor at start of new blank line.

Esc l = Erase entire line, don't move cursor.

Esc M = Delete cursor line, scrolling rest of screen up. Leave cursor at start of next line.

Esc d <n><m>= Scroll up lines <n> to <m> inclusive, leaving new line <m> blank. Lines are numbered starting with top line as 20h.

Esc x 4 = Select block cursor.

Esc x 5 = Cursor off.

Esc 7 or Esc y 25

Esc Y <n><m>= Position cursor at row <n> column <m>. Top left of screen is n=m=20h (ASCII space).

Esc y 4 = Select underscore cursor.

Esc y 5 = Cursor on.

6. EXOS SYSTEM VARIABLES

The following is a list of the EXOS system variables which can be controlled by IS-DOS.

Number	Name	Function
64	ROM_EXDOS	Segment number of the EXDOS ROM
65	IS_P0	\
66	IS_P1	\ Disk transfer segments for
67	IS_P2	/ paging mode 2
68	IS_P3	/
69	ECHO	ECHO flag for IS_DOS batch files
70	VERIFY	EXDOS verify flag (0 = on, FFh = off)
71	DEF_UNIT	Default unit (drive) number (1...26)
72	BOOT_DRV	Drive for booting IS-DOS
73	STEP_RATE	Step rate for UNITH (0 = 6ms, 1 = 12ms, 2 = 20 ms, 3 = 30 ms)
74	DSK_CHK	Zero to enable full disk change checking for already open files
75	IN_ERROR	\ Input and output channels for
76	OUT_ERROR	/ EXDOS default error handling
77	IN_CLI	\ Input and output channels for
78	OUT_CLI	/ EXDOS CLI commands
79	DT_FORM	EXDOS CLI date and time format
80	ER_MESS	Explain error disable flag
81	AB_ERR	Disk error code for .ABORT error
82	DSK_ERR	Error from last IS-DOS command
83	CLI_PROT	Protection flag for IS-DOS CLI
84	RND_0	\
85	RND_1	\ 32 bit random number maintained
86	RND_2	/ by DISKIO for volume ids
87	RND_3	/
88	CLI_EN	Non-zero to disable EXDOS CLI

Number	Name	Function
89	DCH_DIS	Non-zero to make EXDOS believe 'disk not changed' responses from unit handlers
90	FAST_VID	Zero to enable IS-DOS fast video by default
91	CD_PROMPT	Zero to enable printing of current directory as the IS-DOS prompt
92 . . 99		\ \ Reserved for future use by IS-DOS / /

APPENDIX 1 - SUMMARY OF COMMANDS

The following is a list of all the commands available and a summary of their syntax and purpose.

ASSIGN [d: [d:]]

Sets up a logical to physical translation for two drive names.

ATDIR [d:] [path] [/H] [[-]|[+]H]

Changes the attributes of directories to make them hidden/not hidden.

ATTR [d:] [path] [filename] [/H] [[-]|[+]H] [[-]|[+]R]

Changes the attributes of files to make them hidden/not hidden and read only/not read only.

BACKUP [source] [dest] [/D] [/M] [/F] [/X] [/T] [/V] (transient)

Makes a backup copy of a disk or files.

BUFFERS [number]

Displays or changes the number of disk buffers in the system.

CD [d:] [path]

Displays or changes the current directory.

CHDIR [d:] [path]

Displays or changes the current directory.

CHKDSK [d:] [/F] (transient)

Checks the integrity of the disk.

CLS

CLears the screen.

COPY source [/A] [/H] [dest [/A] [/T]] [/V]

Copies data from a file or device to another file or device.

DATE [date]

Displays or sets the current date.

DEL [d:] [path] [filename] [/H]

Deletes all files which match the path/filename.

DIR [d:] [path] [filename] [/H] [/W]

Displays the names of files on disk.

DISKCOPY [d: [d:]] [/X] (transient)

Copies one disk to another.

ECHO [text]

Prints text (comments, instructions etc) in a batch file.

ERA [d:] [path] [filename] [/H]

Deletes all files which match the path/filename.

ERASE [d:] [path] [filename] [/H]

Deletes all files which match the path/filename.

EXDOS [device:filename | channel]

Initializes the EXDOS system and runs simple batch files.

FORMAT [d:] [volname] [/1] [/H] [/8]

Formats a disk.

HELP d: | <name>

Provides help text for a command.

ISDOS [d:] [/command]

Boots IS-DOS from disk.

LOAD device:filename | channel

Loads an EXOS module via EXOS.

MAPDISK d: d:

Allows more than one disk to be used with one drive.

MD [d:] path

Creates a new sub-directory.

MKDIR [d:] path

Creates a new sub-directory.

MODE [[80] | [40] | [Logical-device:=EXOS-device]]

Sets the screen mode or assigns an EXOS device to an IS-DOS logical device.

MOVE [d:] [path] [filename] [/H] [d:] [path]

Moves files from one sub-directory to another on a disk.

MVDIR [d:] [path] [/H] [d:] [path]

Moves sub-directories from one directory to another on a disk.

PATH [item [;item]...]

Displays or sets the transient command search path.

PAUSE [text]

Suspends execution of a batch file until a key is pressed.

PRINTER [ON | OFF]

Switches the printer on or off.

RAMDISK [number] [/D]

Creates or destroys the RAM disk.

RD [d:] path [/H]

Removes one or more sub-directories.

REM [characters]

Allows comments to be put in batch files.

REN [d:] [path] [filename] [/H] [d:] filename

Renames one or more files.

RENAME [d:] [path] [filename] [/H] [d:] filename

Renames one or more files.

RMDIR [d:] path [/H]

Removes one or more sub-directories.

RNDIR [d:] path [/H] [d:] filename

Renames one or more sub-directories.

SAVE [number] [K | P] [d:] [path] filename

Saves part of the TPA.

TIME [time]

Displays or sets the current time.

TYPE [[d:] [path] filename [/H]] | [channel] | [device]

Displays data from a file or device.

UNDEL [d:] [path] [filename] (transient)

Undeletes previously deleted files.

VAR number [[number] | [ON] | [OFF]]

Displays or sets the value of an IS-DOS variable.

VER

Prints the IS-DOS version number.

VOL [d:] [filename]

Displays or changes the volume name on a disk.

XDIR [d:] [path] [filename] [/H] (transient)

Displays the names of all the files in a directory tree.

APPENDIX 2 - ERROR MESSAGES

For details of the error messages issued by EXDOS, see Appendix G of the EXDOS manual. The following additional messages are specific to IS-DOS: --

*** Invalid PATH string

An invalid specification for a drive and path was given as a parameter in a PATH command.

*** Unrecognised command

A transient command was given to IS-DOS, but no corresponding file (with suffix 'COM' or 'BAT') was found.

*** Batch files nested too deeply

A batch file may contain instructions to execute other batch files, which in turn may contain similar instructions. Normally, this 'nesting' of batch files can be continued to a maximum depth of 10, though this figure may be reduced if large numbers of parameters are passed. An attempt to exceed the maximum produces the above error message.

*** Program terminated

Normally, a press on the STOP key during execution of a transient program delivers the prompt 'Terminate program (Y/N)?'. If N is pressed, execution is resumed; if Y is pressed, the program ends and the above message is printed. Then the standard DOS prompt (A>) is given.

*** File for HELP not found

A HELP command was entered, but the corresponding text file (containing a 'HLP' suffix and residing in the \HELP sub-directory) was not found.