CONFIDENTIAL - INTERNAL CIRCULATION ONLY

Document PER-16
Title ..... IS-DOS - System Specification
Issue 2
Date ..... 29th April 1985

CONTENTS
==========

1. Introduction
2. Overall System Operation
  2.1 Loading and Initialisation
  2.2 Standard Channels
  2.3 Command line interpretation
  2.4 Running Transient Programs
  2.5 Batch File Processing
  2.6 System Relocation
  2.7 Redirection of Standard I/O
  2.8 Piping of Standard I/O
3. Transient Program Environment
  3.1 Entry From and Return to IS-DOS
  3.2 Page Zero Usage
  3.3 File Control Blocks (FCBs)
    3.3.1 Format of Normal FCBs
    3.3.2 Extended FCBs
    3.3.3 Indirect FCBs and Pathnames
  3.4 IS-DOS Function calls
4. IS-DOS Commands

1. INTRODUCTION

IS-DOS is an EXOS applications program which provides an MS-DOS like command environment for the user of the Enterprise Computer, and a CP/M and MSX-DOS compatible environment for running transient programs. Its disk functions are all implemented using the disk handling facilities provided by the EXDOS ROM, by going through the EXDOS CLI, the DISK DEVICE or the FISH (Filing System Handling) entry point.

The user's interface to IS-DOS is through the IS-DOS COMMAND LINE INTERPRETER (IS-DOS CLI). This reads and interprets commands from the user through an EXOS editor channel. The commands which the user types may be INTERNAL COMMANDS which are handled by the IS-DOS CLI itself (or in some case by the EXDOS CLI in ROM), SYSTEM COMMANDS which are passed around all system extensions for interpretation, or EXTERNAL COMMANDS which IS-DOS loads from disk.

There are various different types of external commands which IS-DOS handles, which are described later. One very important type is the TRANSIENT PROGRAM, which is simply a binary RAM image file with no EXOS module headers, normally contained in a file with a ".COM" filename extension. IS-DOS will load the file into RAM at 100h and set up a CP/M and MSX-DOS compatible environment for it. The loaded program can then make CP/M compatible calls (referred to as IS-DOS calls in the text) just as if it was running under true CP/M 2.2, but can also make use of certain extended disk handling facilities which EXDOS provides, and can also make EXOS calls in order to use the versatile device independant facilities of EXOS.

Thus transient programs are provided with a very flexible and powerful environment. Programs written specially for IS-DOS can make use of EXOS's channel handling facilities, or EXDOS's tree structured directories. As well as this 'though, the user can use traditional CP/M programs such as Wordstar, M80, L80 and so on, without any modification at all.

The IS-DOS CLI provides various advanced features based on MS-DOS 2.00. These include batch file handling with parameter substitution, and redirection and piping of standard I/O. At the same time it uses the built in EXOS editor for command input to allow correction of mis-typed commands and re-entry of earlier commands.

2. OVERALL SYSTEM OPERATION

2.1 LOADING and INITIALISATION

The code for IS-DOS is contained in two disk files called "IS-DOS.SYS" and "IS-CLI.COM". These files can be put on a disk by the FORMAT program, or can be copied onto a disk later with a "COPY" command. Unlike MS-DOS system files, "IS-DOS.SYS" does not have to be the first file on the disk and neither does it have to be stored on sequential sectors, it is a perfectly ordinary file. This is why it can be copied with the "COPY" command rather than needing a special "SYS" command.

IS-DOS can be loaded by an EXDOS "LOAD IS-DOS.SYS" command, which will normally be contained in the EXDOS initialisation file ("EXDOS.INI") so that IS-DOS starts up automatically at power up.

When IS-DOS is loaded, the first part to be executed is the cold boot program. This resets EXOS, allocates as much user RAM from EXOS as it can up to 64k, transfers the rest of IS-DOS to the top of this memory and then jumps to it. This code includes the resident part of IS-DOS which will remain in memory for as long as IS-DOS is running.

When it gets control from the boot program, IS-DOS first initialises its internal variables and data structures and opens its default channels. It then loads the file "IS-CLI.COM" from the same directory on the same disk as "IS-DOS.SYS" was loaded from. This file contains the IS-DOS CLI (Command Line Interpreter). IS-DOS looks on the disk which it booted from, for a file called "AUTOEXEC.BAT". If it is found then all the commands in it will be executed using the normal batch processing facilities (see section 2.5). If this file is not found, or when all the commands in it have been executed, the IS-DOS CLI will be ready to accept commands from the user.

It is possible to leave IS-DOS by transferring control to another applications program (such as the IS-BASIC ROM) and then starting IS-DOS up again. If this happens then the "AUTOEXEC.BAT" file will be executed again. Any commands which should only be executed once, and whose effect can outlast IS-DOS, should therefore be put in the "EXDOS.INI" file rather than the "AUTOEXEC.BAT" file.

2.2 STANDARD CHANNELS

When IS-DOS starts up it opens the following channels which remain open permanently unless a transient program closes them. If they are closed then IS-DOS will reopen them when it needs them. They can be redirected or captured, and indeed IS-DOS provides explicit facilities for redirection during command execution.

```
Channel 0   - "EDITOR:" channel
Channel ??? - "VIDEO:" channel for editor
Channel ??? - "KEYBOARD:" channel for editor
Channel ??? - "PRINTER:" channel
Channel ??? - "SOUND:" channel
```

The video channel is an 80 column page by default although the "MODE" command can be used to change this to 40 in case a TV is being used for display. The MODE command can also be used to set these channels to different destinations, for example setting the printer to a serial channel.

The editor channel is used by the IS-DOS CLI for displaying prompts, error messages and command output (such as directory listings), and also for reading in command lines. This corresponds to the "standard input" and "standard output" in MS-DOS and will generally be referred to as this in the documentation. This channel will also be set up as the EXOS default channel so that system commands (such as HELP) will use it for I/O.

Transient programs can also use this editor channel for terminal I/O, but will have to do so by making EXOS calls to the channel. It is recommended that programs written specially for IS-DOS use this technique whenever possible, rather than using the IS-DOS screen and keyboard calls which are provided for CP/M compatibility. A program should not mix the two types of calls for terminal I/O.

The IS-DOS character input and output function calls go directly to the screen and keyboard channels in order to be compatible with CP/M. Also the IS-DOS line input function call goes to the video channel and keyboard channels through a CP/M style line editor rather than the full EXOS editor, again for compatibility.

If a transient program makes any IS-DOS screen or keyboard function calls, then when the first one is made, IS-DOS will clear the editor channel (and therefore the screen). All IS-DOS screen and keyboard calls will then be passed to the video and keyboard channels. When the transient program terminates, IS-DOS will read the position of the video channel's cursor and put the editors cursor on the start of the next line. The screen will therefore appear correct although the editor will NOT know about the text displayed by the program and so it cannot be edited or scrolled back on after it has gone off the top of the screen.

2.3 COMMAND LINE INTERPRETATION

The IS-DOS CLI displays a prompt to indicate when it is waiting to read in a command. This prompt is the default drive letter followed by an angle bracket, for example "A>" or "C>" (like MS-DOS and CP/M). When IS-DOS reads a command line, it specifically ignores a prompt of this form at the start of the line. This ensures that the user can re-enter pervious commands simply by moving the cursor to them and pressing ENTER, or can type a command immediately after the prompt.

The general format of an IS-DOS command is a command name, followed by a series of parameters separated by any combination of the characters (space, tab, comma, semicolon and equal sign). Before executing the command, IS-DOS will scan the line for special parameters for redirection and piping, remove them from the command line and set up the required redirection functions (see sections 2.7 and 2.8).

IS-DOS will then examine the command name to determine how to execute it. If the command starts with a colon character then it will be passed around system extensions as a system command. Otherwise IS-DOS will compare it with its list of internal commands, and if it is one of these then IS-DOS will execute the command itself, or in some cases pass it to the EXDOS CLI for execution.

If the command is not a system or internal command then IS-DOS will look for a disk file with the same name as the command and a ".COM" extension, or if that is not found a ".BAT" extension. When looking for these files, IS-DOS first looks in the current directory of the default drive (for the .COM and .BAT file) and if not found there then looks in turn in each directory path specified by the last "PATH" command.

The user may optionally precede an external command name with a drive specifier (for example "B:" or "13:") in which case this drive will be used rather than the default drive for the first search, but if no found the path list will still be searched. Also optionally, the command may be preceded by a pathname (for example the following is a valid command "A:\UTIL\FORMAT"), in which case the specified directory rather than the current directory on the specified or default drive will be searched. In this case if the file is not found in the directory specified then the path list is NOT scanned.

The user may specify an explicit filename extension in the command in which case only this extension will be searched for, and if found it will be treated as a ".COM" file. Note that neither the command name, nor any component of the directory path, may be ambiguous.

If the user gives a command which is just a drive specifier then this drive will be selected as the default drive. If the command is just a drive specifier and a pathname, with no filename, then the specified pathname will be made the current directory for the drive, and if a drive was included then this will be made the default drive.

2.4 RUNNING TRANSIENT PROGRAMS

When IS-DOS finds a ".COM" file, it first passes the file (actually the channel number) to the EXOS "load module" function call. If the file has a module header of a type supported by EXOS or a system extension then it will be loaded appropriately. This ensures that any files which can be loaded by an IS-BASIC "LOAD" command (other than BASIC programs), can be loaded by IS-DOS simply by typing the filename.

If the file does not start with an EXOS module header, then it is a TRANSIENT PROGRAM. IS-DOS will load the whole file into RAM starting at address 100h, set up a CP/M compatible environment for it (see section 3) and call it at address 100h. The highest address which the transient program can use is specified by the destination of the IS-DOS entry jump, ie. the address specified at (0006h,0007h).

Normally IS-DOS sets its entry jump to just above the top of the IS-DOS CLI, but below the resident part of IS-DOS. This allows a transient program to overlay the IS-DOS CLI and thus use more memory. When the transient program terminates, IS-DOS test whether the CLI has been overwritten (with a checksum test) and if so re-loads it from disk.

There is an EXOS variable called "CLI_PROT" which is normally set to 0FFh (OFF) which selects the above action. However if it is set to zero (ON) then the IS-DOS entry jump will be put below the IS-DOS CLI, thus reducing the maximum amount of memory available to transient programs, but ensuring that the CLI will not have to be re-loaded which will save time. Note that whenever a ".BAT" file is entered, this option will be switched on to avoid having to re-load the CLI between batch commands, but can switched off by a suitable "SET" command in the batch file. The variable will be set back to its previous state when the batch file terminates.

The transient program will be passed the remainder of the command line which the user typed (with any redirection parameters removed) in a buffer at address 80h. The first two parameters will be decoded as filenames and stored in unopened FCBs at addresses 5Ch and 6Ch, with pointers to the two pathname strings (copied into internal IS-DOS buffers) at addresses 6Ah and 7Ah. See section 3 for details of IS-DOS FCBs and pathnames.

2.5 BATCH FILE PROCESSING

When the user types the name of a ".BAT" file, IS-DOS will read a series of commands from this file and pass them to the CLI for execution. Any normal CLI commands can be included in the batch file, including system, internal and external commands. Note however that another batch command must be the last command in the file because batch files cannot be nested, they can only be chained.

Parameter substitution is provided for batch files. Before starting execution of the batch file, IS-DOS will scan the original command line (after removal of re-direction parameters) and store the parameters internally. These parameters can then be referred to in commands within the batch file by the special symbols "%0"...."%9". Wherever these occur, apart from within a quoted string, the original text of the appropriate parameter will be substituted before executing the command.

There are certain control commands which are only allowed (or at least are only useful) within a batch file. These are included in the list in section 4, they allow for conditional or repeated execution of commands by using jump and loop constructs.

If an external command executed from within a batch file overwrites the CLI, then the CLI will have to be reloaded from disk before executing the next command from the file, which can slow down batch file operation considerably. To get around this problem, the "CLI_PROT" feature (described in section 2.4 above) is automatically enabled at the start of the batch file and restored to its original state afterwards. This ensures that the CLI will not have to be re-loaded between commands, at the expense of reducing the amount of RAM available to transient programs. If the additional space is needed then the batch file can set "CLI_PROT" off again.

If a command executed from within a batch file is aborted by the user, after a disk error or with the STOP key, then the user will be asked whether to continue batch file processing or return to reading commands from the editor.

The commands read from the batch file will be echoed to the screen before executing them with the usual IS-DOS prompt, unless the "ECHO" EXOS variable has been set off with a "SET" command in the batch file. If "ECHO" is off then only output produced by the individual commands, and error messages will be displayed. This EXOS variable is only used while processing batch files so its value is not restored after the batch file terminates.

The standard input and output of a batch file can be redirected or piped by including the appropriate parameters in the command line. If the standard output is re-directed or piped, then the prompts and command echos will be redirected (if "ECHO" is on) as well as the output from the various commands in the batch file.

The individual commands within a batch file can also use re-direction and piping, which will over-ride any re-direction or piping for the whole batch file for the duration of this one command. This re-direction only applies to the output of the command, not to the IS-DOS prompt and command echo.

2.6 SYSTEM RELOCATION

When IS-DOS starts up, as mentioned before, it obtains as much RAM as possible up to 64k, and uses this to emulate as large a CP/M system as possible. The RAM it obtains is "user RAM" as far as EXOS is concerned, and the top segment of it may be a shared segment. This restricts the amount of RAM available for EXOS, which limits the ability of transient programs to open EXOS channels, and may prevent system extensions from being loaded.

To avoid these problems, IS-DOS provides the ability for the user to relocate it down in memory thereby freeing more memory for EXOS, or to move it back up to give more memory back to transient programs. This relocation can be done by a "RELOCATE" command to the CLI, or by an IS-DOS function call. In either case the effect is permanent until the next relocate operation.

When IS-DOS is relocated it frees or allocates any necessary RAM segments and adjusts the "user boundary" in EXOS. A transient program should never interfere with IS-DOS's memory allocation from EXOS by trying to free its segments etc. However a transient program may allocate additional user segments for its own use, and may manipulate the user boundary in one of these if it gets a shared segment. However if it does this it MUST free the segments before returning to EXOS as otherwise they will be lost for the rest of this IS-DOS session.

2.7 REDIRECTION OF STANDARD I/O

Any IS-DOS command line can contain special parameters to redirect the standard input to or standard output from the command. The output which will be redirected is all that which would normally go to the screen, and the input is all the input which would normally be read from the keyboard.

The redirection parameters are as below, with DEST and SOURCE specifying where the redirected data is to be written to or read from.

>DEST	redirect output to "dest"
>>DEST	append output to "dest"
<SOURCE	get input from "source"

DEST and SOURCE can be either a channel number (eg. £0, £200, £174) or an EXOS device/filename string. If a channel number is specified then the channel must be already open and will not be closed when the redirection stops. If an EXOS device/filename string is specified then IS-DOS will open (for input or append) or create (for output) an EXOS channel using this string, using channel number ??? for input and ??? for output. This channel will be closed when the redirection finishes. Note, of course, that the device/filename string may specify a disk file.

For the append operation, if the specified channel supports random access (EXOS function 10) then the file pointer will be positioned at the start of the file, so that the output will be added to the end of the file.

The re-direction is done using the EXOS redirect and capture facility on the standard I/O channel (channel 0). This ensures that any system commands which use the default channel will be redirected. When redirection is in operation, IS-DOS will trap the appropriate IS-DOS keyboard or screen function calls and use the re-directed EXOS channel.

When an error occurs from the redirected channel, EXOS will automatically cancel the redirection of EXOS channel 0, and IS-DOS will stop its redirection of the IS-DOS function calls. This ensures particularly that if an input redirection has been done and there is insufficient data in the file, then input will revert to the editor.

2.8 PIPING OF STANDARD I/O

Piping is a way of using the standard output of one command as the standard input of another. IS-DOS does this by creating a temporary file on the root directory of the default drive with a name of the form:

%PIPEx.$$$

Piping is specified by giving both command names with their parameters on the same command line, separated by a vertical bar ("|") character. The first command will be executed with its standard output redirected to a file. When this command terminates the second command will be executed with its standard input being redirected to come from the same file. When the second command finishes the pipe file is deleted.

Several piping stages can be specified on one line, sending data through a whole series of programs (or filters as they are referred to in MS-DOS). The first command in a pipe can have its input redirected, and the last one can have its output redirected. The intermediate commands cannot have redirection because the piping is an implicit redirection operation.

3. TRANSIENT PROGRAM ENVIRONMENT

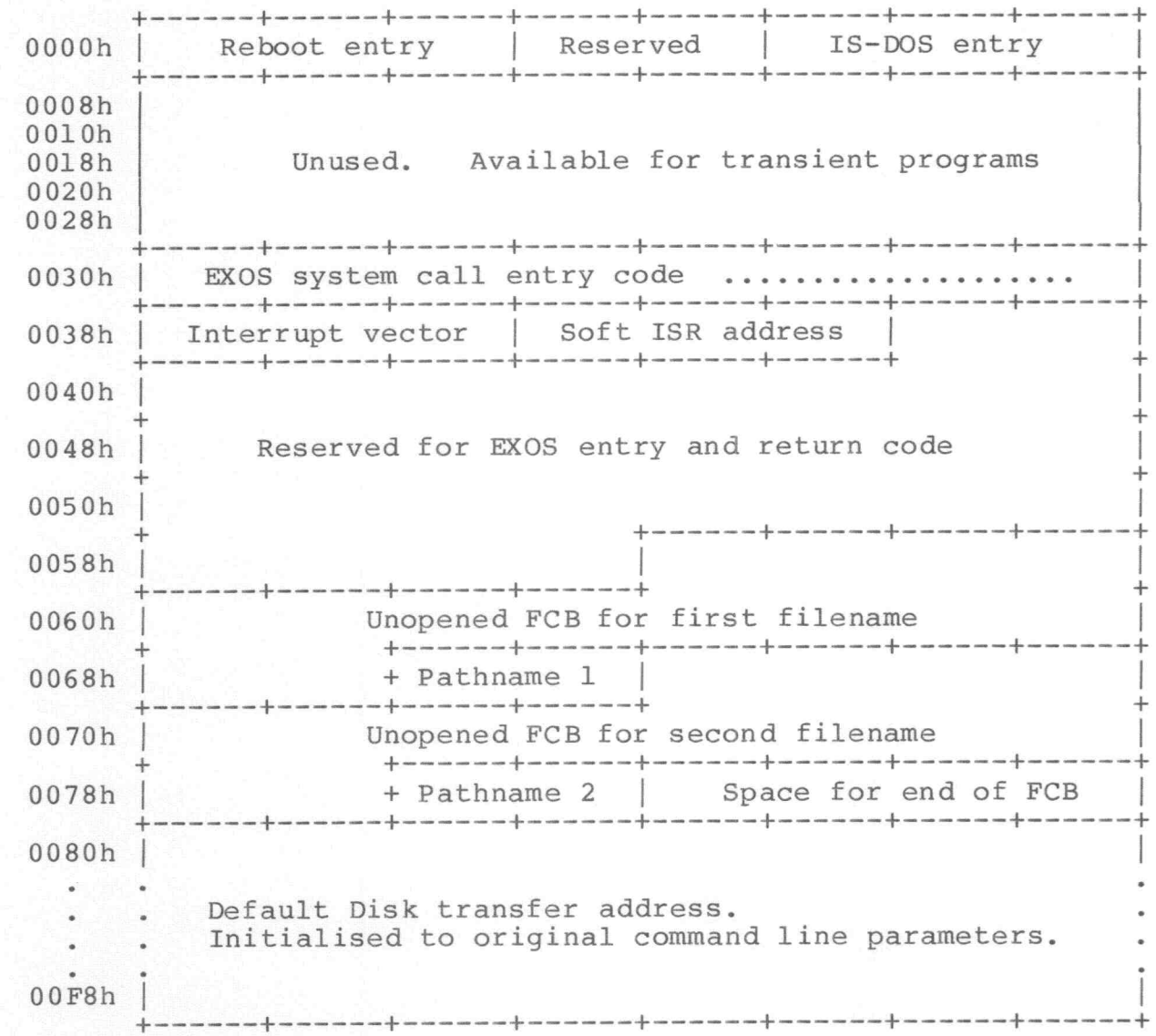
3.1 ENTRY FROM and RETURN TO IS-DOS

A transient program will be loaded at 100h and CALLed by IS-DOS with the stack pointer set to a small (16 byte) 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 then the transient program should set up its own stack.

The transient program can terminate in three ways which are all equivalent for IS-DOS. If it has preserved the original stack pointer then it can simply execute a "RET" instruction. Normally however a program will either jump to address 0000h or execute an IS-DOS function call with code zero. In either case control will get back to IS-DOS which will reload the CLI if it has been overwritten.

3.2 PAGE ZERO USAGE

On entry various parameter areas are set up for the transient program in the lower 256 bytes of RAM. The layout of this area is as below.



The transient program should not disturb the EXOS entry code, even if it makes no EXOS calls, as it is required for interrupt entry and will be needed after return to IS-DOS. The "reboot entry" and "IS-DOS entry" are simple "JMP" instructions to code within IS-DOS.

The "IS-DOS entry" jumps to the lowest address used by IS-DOS and so the destination of this jump can be used by the transient program to determine the highest memory address which it can use. Many programs put their stack here by starting with a "LD SP,(6)" instruction. Note that if the "CLI_PROT" EXOS variable is zero then this address will be below the IS-DOS CLI as well as below the resident part of IS-DOS.

The two reserved bytes at address 0004h and 0005h are the IOBYTE and current drive/user in CP/M. They are not supported by IS-DOS since I/O redirection is handled separately and the current drive is an EXOS variable. CP/M transient programs should not be accessing these bytes directly anyway.

The whole area from 0030h to 005Bh is reserved for EXOS entry and return code and must not be modified by transient programs, apart from setting the software interrupt vector at address 005Dh. Note particularly that most CP/M debuggers (such as ZSID and DDT) use address 38h as a breakpoint entry. These programs will have to be modified and it is recommended that address 28h should be used.

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.
"Pathname 1" and "pathname 2" are pointers to the two pathname strings which were specified on the command line corresponding to the two FCBs. If no pathname was given then these strings will be null. The text of the strings is stored in buffers within IS-DOS. They can be used in conjunction with indirect FCBs to access the specified files if directory paths were used.

The whole of the command line, with the initial command removed, is stored in the default disk transfer address at 0080h, with a length byte first and a terminating null (neither the null nor the length byte are included in the length). No processing at all is done on this string, it includes any leading and trailing blanks which were typed.

3.3 FILE CONTROL BLOCKS (FCBs)

Every file which the transient program wants to access using IS-DOS calls must have an FCB associated with it, the address of which is passed to IS-DOS whenever the file is accessed. The basic type of FCB is a "normal FCB" the format of which is described in the next section. Normal FCBs are compatible with CP/M and MSX-DOS FCBs except that the reserved fields are used differently.

There are two extra types of FCBs which can be used with IS-DOS, "extended FCBs" and "indirect FCBs". Extended FCBs have a 7 byte prefix identical to extended FCBs in MS-DOS and enable file attributes to be manipulated (see section 3.3.2). Indirect FCBs are associated with a pathname string and allow files to be accessed in any sub-directory (see section 3.3.3). Note that an extended FCB can also be indirect.

The FCBs set up by IS-DOS at 005Ch and 006Ch are normal FCBs. They can be made into extended FCBs by putting a prefix on the front (they will have to be moved first), and can be made into indirect FCBs by setting the drive number to 0FEh and using the pathname strings set up by IS-DOS.

When FCBs are passed to IS-DOS they will always be copied into a buffer within IS-DOS and converted into FISH format FCBs (see PER-5) which are always extended and indirect and have additional fields on the end for volume id. checking. After being used the relevant fields will then be copied back to the user's FCB.

Any IS-DOS function calls which take FCBs as parameters can take either normal or extended FCBs. In addition, functions which take unopened FCBs can always accept direct or indirect FCBs.

3.3.1 FORMAT OF NORMAL FCBs

00h	Drive number. 00h => default drive 01h...1Ah => drives A: to Z: 1Bh...FDh => not allowed FEh => indirect FCB (see 3.3.3) FFh => extended FCB (see 3.3.2)
01h...08h	Filename, left justified with trailing blanks. Will be uppercased, bit-7 ignored, and will be rejected if it has illegal characters. "?" can be included for an ambiguous filename.
09h...0Bh	Extension. Similar to filename.

0Ch...0Dh Block number. Set to zero when a file is opened and incremented when necessary in read and write operations. A block is 128 records (corresponding to a CP/M extent).

0Eh...0Fh Record size for block reads and writes. Ignored by CP/M style sequential and random reads and writes. Set to 128 by OPEN and can be changed to any size afterwards by user. Powers of two should be used as other values are inefficient. Small records are no less efficient than large records. A record size of 1 can be used.

10h...13h File size in bytes, lowest byte first. File size is exact, not rounded up to 128 bytes.

14h...15h Date of last change, see PER-5 for format.
16h...17h Time of last change, see PER-5 for format.

18h...1Fh Reserved for IS-DOS use.

20h Current record within block 0...127. Must be set by user (normally to 0) after opening and before reading or writing. Updated by all read and write operations.

21h...24h Random record number, low byte first. This field is optional if only sequential reads and writes are done then it is not required and will not be written to by IS-DOS. Must be set before doing a random or block read or write. If record size is 64 bytes or more then only the first three bytes are used, otherwise all four bytes are used.

3.3.2 EXTENDED FCBs

Extended FCBs are provided to allow creating or searching for files which have special attributes set. An extended FCB has a 7 byte prefix on the front of an otherwise normal FCB. The pointer to the FCB should point to the first byte of this prefix which is always 0FFh to distinguish it from the normal drive number. The drive number is then 7 bytes after this. The format of an extended FCB is:

00h FFh to indicate extended FCB.
01h...05h Five zeroes for future compatibility.
06h File attribute byte
07h...2Bh Normal FCB

Details of how the attribute byte can be used, and of the bit assignments within it, can be found in document PER-5.

3.3.3 INDIRECT FCBs and PATHNAMES

An indirect FCB is exactly the same as a normal unopened FCB but the first byte is set to 0FEh instead of a valid drive number. An indirect FCB can of course have a 7 byte extended FCB prefix on the front. Whenever an indirect FCB is passed to IS-DOS, a pathname string must also be passed. This string specifies the drive and directory path to be used to look for the file, either of which are optional (a null string is acceptable).

When an indirect FCB is opened, or used for a "search for first", the actual drive number will be put into the first byte in place of the 0FEh and the FCB can then be used exactly like a normal FCB. The pathname string is no longer required because the reserved part of the FCB contains the disk address of the directory which was used.

The format of pathname strings is exactly the same as for EXDOS, indeed the strings are simply passed through to EXDOS after copying to an internal buffer. Details of the format of these strings can be found in PER-5.

3.4 IS-DOS FUNCTION CALLS

IS-DOS function calls are made by putting the function code in register C and other parameters in other 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.

All main registers will be corrupted by IS-DOS calls, but the index registers and alternate register set will be preserved. Only a small amount (14 bytes) is needed on the transient program's stack because IS-DOS switches to an internal stack when it is called.

Most functions return an indication of success or failure in register A, for compatibility with CP/M and MSX-DOS. In the case of failure more information can be obtained by making a "get error code" function call which will return the EXOS error code (always non-zero if error) from the previous function call.

In the list of functions below, an asterisk ("*") indicates a CP/M compatible function and a plus ("+") indicates an MSX-DOS compatible function. Some of the functions which are marked as being compatible have been extended, but their basic operation is compatible. If the functions marked "No function" or those not mentioned below are used then they will simply return with HL=DE=A=0.

+	*	00h	Re-boot IS-DOS. Equivalent to "JMP 0".
+	*	01h	Console input
+	*	02h	Console output
+	*	03h	Auxilliary input
+	*	04h	Auxilliary output
+	*	05h	List output
+	*	06h	Direct console I/O
+		07h	Direct input with checks (get I/O byte in CP/M)
+		08h	Direct input no checks (set I/O byte in CP/M)
+	*	09h	String output
+	*	0Ah	Buffered input
+	*	0Bh	Console status
+	*	0Ch	Get version number
+	*	0Dh	Disk reset (flush buffers)
+	*	0Eh	Select Disk
+	*	0Fh	Open file
+	*	10h	Close file
+	*	11h	Search for first
+	*	12h	Search for next
+	*	13h	Delete file
+	*	14h	Sequential read
+	*	15h	Sequential write
+	*	16h	Create file
+	*	17h	Rename file
+	*	18h	Get login vector
+	*	19h	Get default drive
+	*	1Ah	Set Disk transfer address
		1Bh	Get allocation information
		1Ch	No function (set write protect vector in CP/M)
		1Dh	No function (get write protect vector in CP/M)
		1Eh	No function (set file attributes in CP/M)
		1Fh	No function (get disk parameter address in CP/M)
		20h	No function (set/get user code in CP/M)
+	*	21h	Random read
+	*	22h	Random write
+	*	23h	Get file size
+	*	24h	Set random record
		25h	No function (reset disk drive in CP/M)
+		26h	Random block write
+		27h	Random block read
+	*	28h	Random write with zero fill
		29h	No function
+		2Ah	Get date
+		2Bh	Set date
+		2Ch	Get time
+		2Dh	Set time


```
+    2Eh  Set/reset verify flag
+    2Fh  Absolute disk read
+    30h  Absolute disk write
     31h  \
      .    \  No function.  Reserved
      .    /
     3Fh  /
     40h  Get or set UPB
     41h  Relocate IS-DOS
     42h  Set error vector
     43h  Get/change file attributes
     44h  Set current directory path
     45h  Get current directory path
     46h  Make sub-directory
     47h  Remove sub-directory
     48h  Open searched file
     49h  Move file
```

4. IS-DOS COMMANDS

The overall operation of the IS-DOS CLI was described in section 2. Commands fall into three catagories, INTERNAL, SYSTEM or EXTERNAL commands. EXTERNAL commands are located on disk and the command name is the same as the filename. SYSTEM commands are any which begin with a colon (":"), and will be passed around system extensions for interpretation.

INTERNAL commands are those where IS-DOS specifically recognises and acts on the command name itself. Many of these commands are actually implemented by the EXDOS CLI but are classed as internal commands because teh colon does not have to be typed. Below is a list of commands, "*" indicates that the command is actually an EXDOS CLI command.

```
*    CD        - Changes the current directory
*    MD        - Makes a new sub-directory
*    RD        - Removes a sub-directory
     PATH      - Sets a new path list for commands

*    DIR       - Displays directory listing
*    DEL       - Deletes files
*    REN       - Renames files or sub-directories
*    ATTR      - Changes attributes of file
*    TYPE      - Types a file on console
*    COPY      - Copies files
*    MOVE      - Moves files to a new directory
*    EDIT      - Edits a text file using the EXOS editor
```



```
    CLS         - Clear screen
*   DATE        - Sets or displays date
*   TIME        - Sets or displays time
*   SET         - Sets value of EXOS variables
    MODE        - Allows manipulation of standard I/O channels
*   ASSIGN      - Assigns new name for a drive
*   MAPDISK     - Allows one drive to access two disks

    GOTO        - Goto a label in a batch file
    SKIP        - Jumps forward to a label in a batch file
    *label      - Specifies a label in a batch file
    FOR         - Allows loops is a batch file
    IF          - Conditional test in a batch file
    ECHO        - Displays a message from a batch file
    SHIFT       - Shifts parameters in a batch file
    PAUSE       - Waits for key response in a batch file
    EXIT        - EXits from a batch file

    RELOCATE    - Relocates IS-DOS to a different address
*   BUFFERS     - Creates additional permanent buffers
*   RAMUNIT     - Creates or removes the RAM disk
```

++++++++++ END OF DOCUMENT ++++++++++

