3.4 MSX-DOS Boot Procedure

1) Boot Procedure

When all the buffers for the disk system are successfully allocated, the disk ROM checks the contents of address 0FEDAH to see if a ROM cartridge has set the hook (H.STKE) to gain control of the disk system. If the contents is not a 'RET' instruction (0C9H), the disk ROM sets up environments for disk BASIC and jumps to this hook.

The disk ROM next checks if there is an existing cartridge which has a TEXT entry in the cartridge header. If such a cartridge is found, the disk ROM sets up environments for disk BASIC and executes the BASIC program from the cartridge.

Next, the first sector of a first track (logical sector number 0) is read and transferred to 0C000H to 0C0FFH. If this read routine fails because of a drive not ready, a read error, or if the first byte of the boot sector is not 0EBH nor 0E9H, disk BASIC starts up.

Next, address 0C01EH is called with the carry flag set. This routine is provided so as to make game or other application programs take control of the disk system. The standard boot sector (provided) will just execute a 'RET' instruction if the carry flag is reset.

The ROM program next does a non-destructive memory check. If a 64K-byte RAM is not available, the program transfers control to disk BASIC.

Next the environments for MSXDOS are set up, and the routine jumps to 0C01EH with the carry flag set. Our standard boot sector loads MSXDOS.SYS at 100H and jumps to it. If MSXDOS.SYS not present, disk BASIC is invoked.

MSXDOS.SYS loads COMMAND.COM at 100H and jumps to it. If COMMAND.COM is not present, the routine prompts the user to insert a diskette with COMMAND.COM in it.

2) AUTOEXEC.BAT

When MSXDOS is first booted, it searches for a file named AUTOEXEC.BAT and executes it as a batch file.

3) AUTOEXEC.BAS

When MSX disk BASIC is first invoked, it looks for a file named AUTOEXEC.BAS and executes it as a BASIC program.

3.5 MSX-DOS and MSX Disk BASIC Disk Driver

The following values must be defined and declared as PUBLIC by the person or organization doing the interfacing.

MYSIZE
: Byte size of the work area used by the driver.

SECLEN
: The maximum sector size for the media supported by the driver.

DEFDPB
: The base address of the DPB (which consists of 18 bytes) for the media having the largest value for FATSIZ*SECSIZ.

The following subroutines must be provided and declared as PUBLIC by the person or organization doing the interfacing.

INIHRD	Initialize hardware
DRIVES	Return number of drives in system
INIENV	Initialize work area
DSKIO	Read/Write to disk
DSKCHG	Get disk change status
GETDPB	Get drive parameter block
CHOICE	Return character string for disk formatting
DSKFMT	Format disk
OEMSTATEMENT	(Entry point for use in system expansion)

The following is a detailed description the above routines.

INIHRD

Inputs:
: None

Outputs:
: None

Registers:
: AF, BC, DE, HL, IX, IY may be affected.

This routine initializes the hardware as soon as the control passes to the cartridge. Note that no work area is assigned when this routine is initiated.

DRIVES

Inputs:
[F] = The zero flag is reset in case one physical drive must act as two logical drives.

Outputs:
[L] = Number of drives connected

Registers:
F, HL, IX, IY may be affected.

Before any other processing can be done, the number of drives connected to the cartridge must be counted. If more than one drive is detected, or if the zero flag passed from the calling routine is set, the number of drives is returned (unmodified).

If only one drive has been detected and the zero flag passed is reset, a '2' must be returned as the number of drives, and the DSKIO and DSKFMT routines must logically support two drives. Use the PROMPT routine (described below) when switching drives.

When this routine is entered, the work area for the driver is already allocated.

INIENV

Inputs:
None

Outputs:
None

Registers:
AF, BC, DE, HL, IX, IY may be affected.

This entry initializes the work area (environment).

```
==========================================================
=       INIHRD, DRIVES and INIENV are called only        =
=       once during initialization, in the above         =
=       order.                                           =
==========================================================
```

DSKIO

Inputs:

[F]	=	Carry flag reset for read, set for write
[A]	=	Drive number (starts at 0)
[B]	=	Number of sectors to read/write
[C]	=	Media descriptor
[DE]	=	Logical sector number (starts at 0)
[HL]	=	Transfer address

Outputs:

If successful, carry flag cleared.
Otherwise, carry flag set, error code is placed in [A], number of remaining sectors in [B].

Registers:

AF, BC, DE, HL, IX, IY may be affected.

The drive number and media descriptor come from the drive parameter block. The number of sectors may range from 1 to 255. The logical sector numbers start at zero and is incremented in ones, so the I/O system must map these the logical sector numbers into tracks and sectors. The logical sector 0 corresponds to track 0, sector 1.

The error codes are defined as follows:

0	Write protected
2	Not ready
4	Data (CRC) error
6	Seek error
8	Record not found
10	Write fault
12	Other errors

DSKCHG

Inputs:

```
[A]  =  Drive number
[B]  =  0
[C]  =  Media descriptor
[HL] =  Base address of DPB
```

Outputs:

```
If successful:
    Carry flag reset,
    [B]  = Disk change status
              1      Disk unchanged
              0      Unknown
             -1      Disk changed

ELSE:
    Carry flag set,
    Error code in [A] (same as DSKIO above)
```

[NOTE]

If the disk has been changed or may have been changed (Unknown), read the boot sector or the first byte of the FAT of the currently inserted disk and transfer a new DPB as with the GETDPB call described below.

Registers:

AF, BC, DE, HL, IX, IY may be affected.

GETDPB

Inputs:

[A]	=	Drive number
[B]	=	First byte of FAT
[C]	=	Media descriptor
[HL]	=	Base address of DPB

Outputs:

[HL+1] .. [HL+18] = DPB for the specified drive

The Drive Descriptor Block (DPB) is defined as follows:

MEDIA	Byte	Media type
SECSIZ	Word	Sector size (Must be 2^n)
DIRMSK	Byte	(SECSIZ/32)-1
DIRSHFT	Byte	Number of one bits in DIRMSK
CLUSMSK	Byte	(Sectors per cluster)-1
CLUSSHFT	Byte	(Number of one bits in CLUSMSK)+1
FIRFAT	Word	Logical sector number of first FAT
FATCNT	Byte	Number of FATs
MAXENT	Byte	Number of directory entries (Max=254)
FIRREC	Word	Logical sector number of where the data area starts
MAXCLUS	Word	(Number of clusters on drive [not including reserved sectors, FAT sectors, or directory sectors])+1
FATSIZ	Byte	Number of sectors used
FIRDIR	Word	FAT logical sector number of start of directory

Note that the logical sector number always begins at zero.

CHOICE

Returns in [HL] the pointer to the character string (terminated by a zero) that is used as a user prompt in menu form by the main code. The simplest form of the routine be as follows.

```
CHOISE: LD      HL,CHOMSG
        RET
;
CHOMSG: DEFB    '1 - Single sided, 8 sectors',CR,LF
        DEFB    '2 - Single sided, 9 sectors',CR,LF
        DEFB    '3 - Double sided, 8 sectors',CR,LF
        DEFB    '4 - Double sided, 9 sectors',CR,LF
        DEFB    0
```

If there is no choice (i.e., only one format is supported), return with 0 in [HL] register.

All registers except SP may be affected.

DSKFMT

Formats a disk, both physically and logically. The input parameters are as follows.

[A] Choice specified by the user (1 to 9). Meaningless unless there is a choice.
[D] Drive number, beginning at zero
[HL] Beginning address of the work area which can be used by the format process.
[BC] Length of the work area described above.

All registers except SP may be affected.

This routine formats all of the disk's tracks physically, writing the boot sector, and clearing FATs and directory entries.

'Clearing FATs' means:

Writing the media descriptor byte at the first byte, writing 0FFH at the second and the third byte, and filling the remainder with 0's

'Clearing directory entries' means:

Filling all bytes with 0's

If the format ends successfully, return with carry flag reset, otherwise return with carry flag set. The error codes are defined as follows:

0	Write protected
2	Not ready
4	Data (CRC) error
6	Seek error
8	Record not found
10	Write fault
12	Bad parameter
14	Insufficient memory
16	Other errors

[NOTE]

No prompting messages should be generated by this routine.

OEMSTATEMENT

Statement for system expansion for use by OEMs. After disk BASIC scans its own expanded statements, control is passed to this entry. The calling sequence is identical to using a general-purpose expansion statement handler. If your ROM does not have expansion statements, set the carry flag and do a Z80 'RET' instruction.

```
****************************************
*                                      *
*    Some useful external routines     *
*                                      *
****************************************
```

PROMPT

Prints a message as follows and waits for the user to enter a key from the keyboard.

```
'Insert diskette for drive X:
 and strike a key when ready'
```

The 'X' is the drive name of the current target drive of your cartridge.

SETINT

This routine saves a previously set interrupt hook to a location specific to your cartridge, and sets the new interrupt hook. The address of the interrupt routine should be passed via the [HL] register. See DSKDRV.Z80 for details.

PRVINT

This routine jumps to the interrupt hook that you might have overwritten. Requires no argument. See DSKDRV.Z80 for details.

GETSLOT

Gets the slot address (i.e., where I am) in [A].
Preserves DE, IX, IY

GETWRK

Gets the base of the work area in [IX] and [HL].
Preserves DE, IY

DIV16

[BC]=[BC]/[DE], remainder in [HL].
Preserves DE, IX, IY

ENASLT

Enables a slot at an address specified by [A] and [HL], respectively. Destroys all registers.

XFER

Moves [BC] bytes from [HL] to [DE] (i.e., LDIR)
Preserves AF, IX, IY
BC is set to 0, HL, and DE pointing to the next location of source and destination, respectively.

Use this routine when a read/write operation is requested to 4000H..7FFFH, and your hardware does not have any special mechanism to transfer directly to these areas.

```
****************************************
*                                      *
*          External variables          *
*                                      *
****************************************
```

$SECBUF

Pointer to a temporary storage which is at least SECLEN byte long. Prepared for use combined with the XFER subroutine described above, but can be used TEMPORARILY for any purpose.

RAMAD0, RAMAD1, RAMAD2, RAMAD3

Slot address of RAM (if present) at 0000H..3FFFH, 4000H..7FFFH, 8000H..BFFFH, C000H.FFFFH respectively.

RAWFLG

Read-After-Write flag. When this byte contains non-0 value, the disk driver should do a read-after-write check. However, it is completely up to the driver whether to do the check or not.

How to determine media types

a) Read the boot sector (track 0, sector 1) of the target drive.

b) Check if the first byte is either 0E9H or 0EBH (the JMP instruction on the 8086)

c) If step b) fails, the disk is a version prior to MS-DOS 2.0; therefore, use the first byte of FAT passed from the caller and make sure it is between 0F8H and 0FFH.

   If step c) is successful, use this as a media descriptor.
   If step c) fails, then this disk cannot be read.

d) If step b) succeeds, read bytes # 0B to # 1D. This is the DPB for MS-DOS, Version 2.0 and above. The DPB for MSXDOS can be obtained as follows.

Contents of MS-DOS boot sector

+00	0E9H,XX,XX or 0EBH,XX,XX	
+03	ASCII string of OEM name	
+0B	Bytes per sector	(low)
+0C		(high)
+0D	Sectors per cluster	
+0E	Number of reserved sectors	(low)
+0F		(high)
+10	Number of FATs	
+11	Number of directory entries	(low)
+12		(high)
+13	Total number of sectors in the media	(low)
+14		(high)
+15	Media descriptor	
+16	Number of sectors per FAT	(low)
+17		(high)
+18	Sectors per track	(low)
+19		(high)
+1A	Number of heads	(low)
+1B		(high)
+1C	Number of hidden sectors	(low)
+1D		(high)

MS-DOS Disk formats

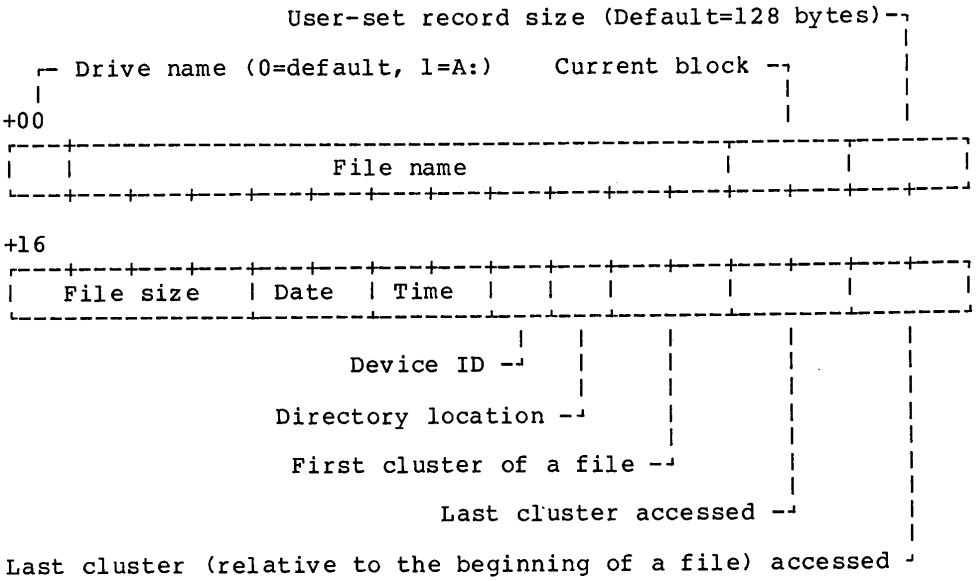
For 3, 3.5, and 5 inch disks (IBM PC format)

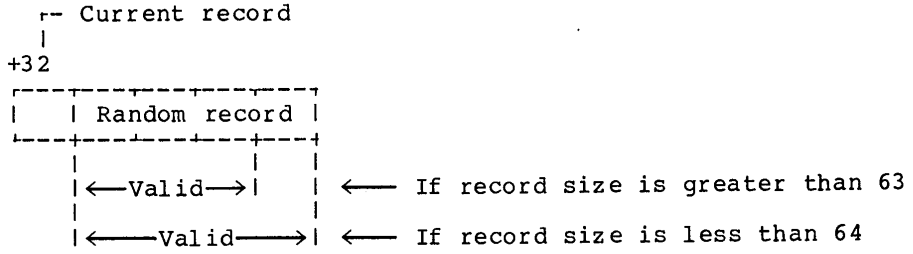
First digit : track number 8=80, 4=40
Second digit: Sector count 8 or 9
Third digit : Head count 1 or 2

	891	892	881	882	491	492	481	482
Root directory entry	112	112	112	112	64	112	64	112
Media descriptor byte (FATID)	0F8H	0F9H	0FAH	0FBH	0FCH	0FDH	0FEH	0FFH
Sectors per FAT	2	3	1	2	2	2	1	1
Sectors/track	9	9	8	8	9	9	8	8
No. of sides	1	2	1	2	1	2	1	2
Tracks/side	80	80	80	80	40	40	40	40
Bytes/sector	512	512	512	512	512	512	512	512
No. of FATs	2	2	2	2	2	2	2	2
Sectors/cluster	2	2	2	2	1	2	1	2

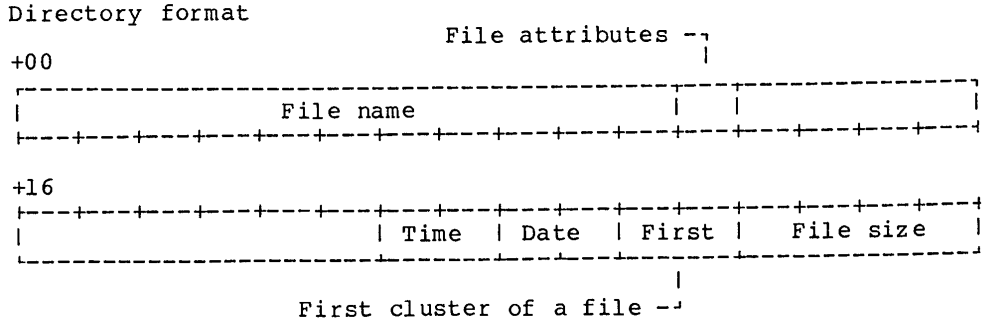
3.6 MSX-DOS System Calls

1) File Control Block (FCB) and Directory Entry

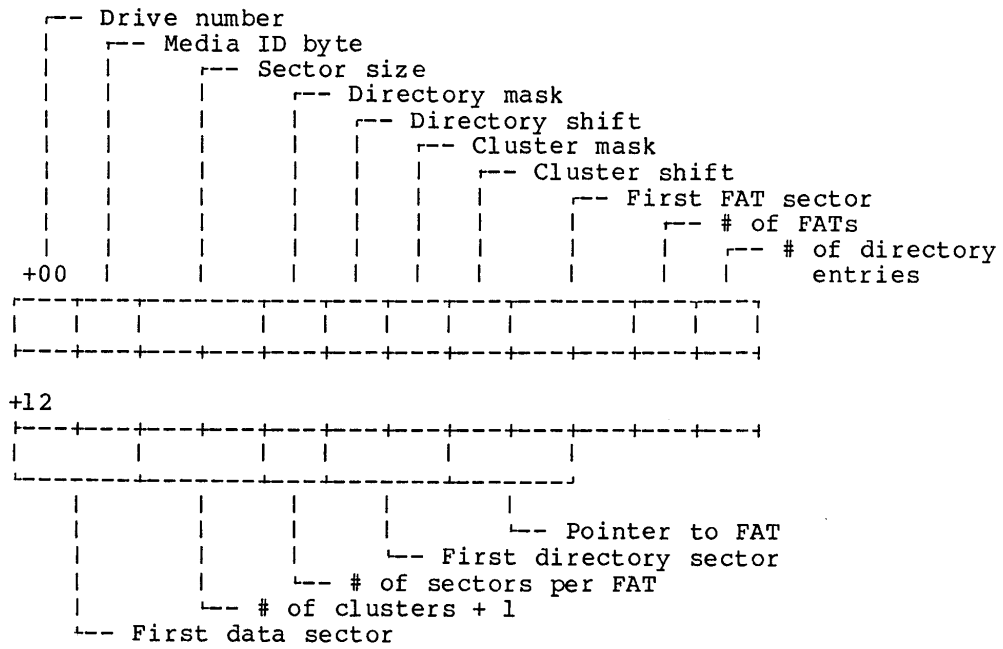




Directory format



2) Drive Parameter Block (DBP)



3) File Allocation Table (FAT)