CHAPTER 9:
RAM DATA STRUCTURES

Accessing MOS's Data Structures

As MOS evolves, the method used to access the kernel's data structures has had to change. Prior to version 4.10, direct access to the SCB was achieved through a pointer obtained from Extended Services function 02H. Likewise, function 04H was used to obtain a pointer for direct TCB access.

Beginning with version 4.10, all access to MOS's data structures should be achieved through the use of Extended Services functions 26H, 27H, 28H, 29H and 2AH. This group of functions provides a means of access which will insulate the developer from any further changes in access methods.

This change has been made to accommodate the eventual relocation of MOS's kernel and kernel data structures into memory above the 1st megabyte. Once this migration is completed, the use of these new Extended Services functions will be the only way to access data within the SCB and TCB data structures.

For an example of the use of both the old and new data structure access techniques, see Chapter 13.

The System Memory Pool

The System Memory Pool (SMP) is a fixed section of memory set aside at MOS initialization time. It contains control blocks, I/O buffers (but not disk buffers), device drivers, screen save areas, and any other scratch space that might be required by the system.

The size of the SMP may be altered by the user as a CONFIG.SYS parameter (SMPSIZE=nnnK), and takes effect only at bootup time.

Memory is allocated as a linked list within the SMP. Each block allocated is a multiple of 16 bytes, and includes a control prefix having the following format:

Byte 0 - 'H' to indicate allocated from the SMP

Byte 1 - Type of block (Header Block ID value, e.g. 'T' = TCB)

Bytes 2-3 - Length of this block, in paragraphs

Bytes 4-5 - Segment address of next block (0 if end)

Bytes 6-7 - Segment address of previous block (0 if first)

Bytes 8-9 - Address of next block of same type

Bytes A-B - Address of previous block of same type

Bytes C-D - Address of associated TCB (if applicable)

Bytes E-F - Reserved

Bytes 10+ - Data

NOTE: A "segment address" is defined as bits 4-19 of an absolute memory address for which bits 0-3 and 20-31 are zero.

System Control Block

The System Control Block (SCB) is the root of all data managed by the operating system. It contains all system-wide parameters, as well as pointers to other control blocks.

Most of the information in the SCB is reserved. The following list contains fields that might be of interest to developers. See Chapter 13 for an example of the use of the offsets shown to access these data.

Offset	Description
0H	A word value that is a pointer to the first TCB in the linked list of TCB's.
13H	A word value that is a pointer to the current task's TCB.
15H	A word value that points to the TCB of the task currently being viewed at the main console.

Task Control Block

There is a Task Control Block (TCB) allocated from the SMP for every active task in the system. It contains information relative to that task as a whole. Note that the offset of the first item shown in the list below is 10h. This is due to the fact that the SMP header occupies the first 16 bytes.

An include file for this data structure is provided on the companion diskette. Refer to Chapter 13 for an example of its use.

Header Block ID value: 'T'

The TCB Fields

Field	Ofs	Size	Description
TCBTCBPN	0008	word	Pointer to next TCB
TCBTCBPP	000A	word	Pointer to previous TCB
TCBID	0010	word	Task id (task0 = 0, task1 = 1, etc.)
TCBNCADR	0012	word	Native context save address
TCBBEGAD	0014	word	Partition's start address
TCBENDAD	0016	word	Partition's end address
TCBPRI	0018	byte	Partition's priority
TCBSLICE	0019	byte	Partition's time slice value
TCBWAIT	001A	byte	Bit0 = 1 if waiting
TCBSTAT	001B	byte	Indicates what the task is waiting on
TCBPOLL	001C	dword	Address of polling routine
TCBERRCOD	0020	byte	Error code from last function call
TCBPRGNM	0021	11bt	Name of program currently running
TCBIBASE	002C	word	Base address of keyboard input buffer
TCBIBSIZ	002E	word	Keyboard buffer size - in bytes
TCBLOOP	0030	byte	DIS/NODIS flag. See note 1
TCBKSHFT	0031	byte	Current kybd shifts and toggles
TCBMODE	0034	byte	Current video mode
TCBPAGE	0035	byte	Current video page
TCBCOLS	0036	byte	Number of text columns per screen
TCBROWS	0037	byte	Number of text rows per screen
TCBSCLN	0038	word	Length of video buffers
TCBPGLN	003A	word	Video page length
TCBPGST	003C	word	Page start address in video ram
TCBCPOS	003E	4w	Cursor positions for 4 screen pages
TCBCTYP	004E	word	Current cursor type

The TCB Fields (cont)

Field	Ofs	Size	Description
TCBPAL	0050	byte	Current palette setting
TCBOMODE	0051	byte	Original video mode
TCBWINPO	0052	byte	Starting crt row number. 0 or 1
TCBVRAM	0053	byte	Vidram indicator. See note 2
TCBVIDH1	0054	word	Handle of video save area
TCBVIDP1	0056	word	Page count of video save area
TCBVIDW2	0058	word	Segment address of video save area
TCBTFBPF	005A	word	Pointer to first TFB
TCBCDBPF	005C	word	Pointer to first CDB
TCBCDBPC	005E	word	Pointer to CDB for current drive
TCBNDRIV	0060	byte	Number of drives
TCBCDRIV	0061	byte	Current drive (0=A, 1=B, etc.)
TCBDTA	0062	word	Disk Transfer Address
TCBVFLG	006A	byte	Verify flag. nz = on for this task
TCBBRK	006B	byte	Break flag. nz = on for this task
TCBNTRY	006C	word	Share/lock retry count
TCBTTRY	006E	word	Ticks between share/lock retries
TCBAUXBT	0070	byte	Remote printer flags. See note 3
TCBAUXCT	0071	byte	ETX/ACK delay count
TCBSPOOL	0074	word	Spooler segment address
TCBLPTXL	0076	3b	Printer redirection. See note 4
TCBLSCAN	0116	byte	Last scan code
TCBCONDD	05D0	dword	Far pointer to DDT's entry point
TCBDLOFS	05D4	word	Offset of logical screen
TCBDLSEG	05D6	word	Segment of logical screen
TCBDCOFS	05D8	word	Cursor's offset within page
TCBDCOL	05DA	byte	Screen cols/row
TCBPORT	05DB	word	Async port number, 0=none
TCBBAUD	05DD	dword	Physical baud rate
	05E1	19bt	Reserved area for ddt
TCBUNREG	07A6	dword	Far pntr to unregister calling chain

Additional reserved fields in the TCB follow those described above.

Note 1: If Bit 1 is 1, use DIS mode.

Note 2: If bit 0 = 1 Vidram is active. Bit 7 = 1, this is the main console.

Note 3: The remote printer flag is defined as:

Bit 0: 1 = LPT1 to terminal.
Bit 1: 1 = LPT2 " "
Bit 2: 1 = LPT3 " "
Bit 3: 1 = Escape to printer pending
Bit 4: 1 = Use XON/XOFF protocol
Bit 5: 1 = Use ETX/ACK protocol
Bit 6: 1 = Waiting for ACK or XON
Bit 7: 1 = Transparent printing on

Note 4: The remote printer redirection information contains three bytes. One byte corresponds to each logical LPT device. The bytes are interpreted as follows: If the byte is 0, no redirection. If the byte is in the range of 1 - 24, it is the COM port number the printer is redirected to. If the byte is 81 - 83, it's the LPT port this printer redirected to.

Current Directory Block

The Current Directory Blocks (CDB) are chained from each TCB, with one CDB per drive per task. It contains information about the drive's current directory for that task.

Header Block ID value: 'C'

The CDB Fields

Field	Ofs	Size	Description
CDBDRIVE	0010	byte	Drive number. 0 = A, 1 = B, etc.
CDBPATH	0012	64b	Directory name
CDBCLUS	0042	word	1st directory cluster. 0 = root

Total Block Length: 84 bytes.

Global File Block

Each file opened by any task has one Global File Block (GFB), a Task File Block (TFB) for each task for which the file is open, and a Record Lock Block (RLB) for each locked region of the file.

Header Block ID value: 'G'

The GFB Fields

Field	Ofs	Size	Description
GFBATTR	001A	byte	File attribute
GFBDRVR	001D	dword	Address of device driver
GFBCBOF	0021	word	First cluster
GFBTIME	0023	word	Time of last modification
GFBDATE	0025	word	Date of last modification
GFBSIZE	0027	dword	32-bit size of file in bytes
GFBNAME	0036	11b	Device name or file name and extension
GFBTFB	0041	word	Segment address of TFB list
GFBRLB	0043	word	Seg address of 1st RLB. 0 = none
GFBDEV	0045	byte	Non-zero if GFB refers to a char device
GFBBDB	0046	word	Address of block device block
GFBDLOC	0048	word	Sector of file's directory entry
GFBHLOC	004A	word	High word of file's directory entry
GFBDOFS	004C	word	Offset within sector of dir entry

Total Block Length: 54 bytes.

Task File Block

Header Block ID value: 'F'

The TFB Fields

Field	Ofs	Size	Description
TFBNGFB	0010	word	Segment address of next TFB
TFBPGFB	0012	word	Segment address of previous TFB
TFBGFB	0014	word	Segment address of TFB's GFB
TFBPSP	0016	word	Segment address of owner's PSP
TFBHDL	0018	word	File handle
TFBPOS	001D	dword	File read/write pointer
TFBIOCTL	0025	byte	IOCTL flag. See following list

Total Block Length: 40 bytes.

The bits of the IOCTL flag are defined as:

Bit 7: 1 = a device. 0 = file
Bit 6: 1 if EOF encountered on input
Bit 5: 1 if binary mode. 0 = if ASCII
Bit 4: Reserved
Bit 3: 1 = clock device
Bit 2: 1 = null device
Bit 1: 1 = standard output device
Bit 0: 1 = standard input device

Record Lock Block

Header Block ID value: 'R'

The RLB Fields

Field	Ofs	Size	Description
RLBPSP	0010	word	Segment address of owner's PSP
RLBGFB	0012	word	Segment address of GFB
RLBTFB	0014	word	Segment address of owner TFB
RLBBGN	0016	dword	File offset of start of locked region
RLBEND	001A	dword	Length of locked region
RLBHDL	001E	word	Owner's handle for file

Total Block Length: 30 bytes.

Block Device Block

A Block Device Block (BDB) is allocated for each unit of each block device driver which is present in MOS.

Header Block ID value: 'B'

The BDB Fields

Field	Ofs	Size	Description
BDBID	0010	byte	Logical drive. 0 = A, 1 = B, etc.
BDBUNIT	0011	byte	Unit number passed to driver
BDBSSIZ	0012	word	Sector size
BDBCMSK	0014	byte	Cluster mask. (Cluster number - 1)
BDBCSHL	0015	byte	Cluster shift count
BDBFAT	0016	word	Starting sector of first FAT
BDBFNUM	0018	byte	Number of FATs
BDBRNUM	0019	word	Number of root directory entries
BDBCLUS	001B	word	Sector number of cluster 2
BDBCNMX	001D	word	Number of clusters + 1
BDBFSIZ	001F	byte	Number of sectors in a FAT
BDBDIR	0020	word	Beginning directory sector number
BDBDRVR	0022	word	Device driver address
BDBMDIA	0026	byte	Media descriptor byte
BDBBIG	002C	byte	Volume size. N: 32 MB, Y: >32MB
BDBCSIZ	002E	byte	Number of sectors in one cluster
BDBCNUM	002F	word	Number of clusters on device
BDBCFREE	0031	word	Number of clusters free (-1 if invalid)
BDBROOT	0033	word	Root directory cluster number
BDBALIAS	0035	word	Pointer to alias string
BDBTASK	0037	word	TCB segment address of owner. 0 none

Total Block Length: 52 bytes.

File Control Block

The File Control Block (FCB) is supplied for compatibility purposes, and is not described here. It is supported as a means of communicating file I/O back and forth between MOS and the application. However, all file I/O is performed internally using the same logic and consistent data structures. Every file will have a handle, though handles for FCB files will of course not be communicated to the application. FCBs are not recommended for new applications, and are not supported for native mode programs.

Program Segment Prefix

The Program Segment Prefix (PSP) immediately precedes a program loaded into memory. It is provided for compatibility purposes. All "known" areas of the PSPs are maintained as a standard source of information for each task, but in general are not used for control purposes by MOS.

The PSP Fields

Field	Ofs	Size	Description
PSPTERM	0000	word	MOS program terminate address
PSPMEM	0002	word	End of program's memory block
PSPCALL	0005	byte	Long CALL to MOS
PSPTRMV	000A	dword	Terminate vector (INT 22H)
PSPCTRV	000E	dword	Ctrl-Break vector (INT 23H)
PSPCRTV	0012	dword	Critical Error vector (INT 24H)
PSPPRNT	0016	word	PSP segment of parent
PSPHTBL	0018	20b	File handle table
PSPENV	002C	word	Segment address of environment
PSPSP	002E	word	Stack pointer of parent process
PSPSS	0030	word	Stack segment of parent process
PSPHDLS	0032	word	Number of handles in use
PSPPNTR	0034	dword	Pointer to PSP handle table
PSPFARC	0050	3b	Far call to function dispatcher
PSPFCB1	005C	16b	FCB #1
PSPFCB2	006C	20b	FCB #2
PSPCNT	0080	byte	Number of chars on command line
PSPCMD	0081	127b	Application's command line

Total Block Length: 256 bytes

Native Context Area

The Native Context Area (NCA) contains I/O buffers and other information needed to support native mode applications under MOS. Its space is provided by the application itself, and does not come from the SMP.

This data structure is used by the memory management device driver for native mode applications. Its definition is:

Field	Ofs	Size	Description
ncavflag	0000	2w	Flag register info. For V86 tasks
ncavdi	0000	2w	DI register save area. V86 tasks
ncavsi	0000	2w	SI register save area. V86 tasks
ncavdx	0000	2w	DX register save area. V86 tasks
ncavax	0000	2w	AX register save area. V86 tasks
ncavbx	0000	2w	BX register save area. V86 tasks
ncavcx	0000	2w	CX register save area. V86 tasks
ncaves	0000	2w	ES register save area. V86 tasks
ncavds	0000	2w	DS register save area. V86 tasks
ncavbp	0000	2w	BP register save area. V86 tasks
ncanes	0000	2w	ES register save area. V86 tasks
ncands	0000	2w	DS register save area. V86 tasks
ncanfs	0000	2w	FS register save area. V86 tasks
ncangs	0000	2w	GS register save area. V86 tasks
ncanip	0000	2w	IP register save area. V86 tasks
ncancs	0000	2w	CS register save area. V86 tasks
ncanflag	0000	2w	Flags register save area. V86 tasks
ncansp	0000	2w	SP register save area. V86 tasks
ncanss	0000	2w	SS register save area. V86 tasks
scratch	0000	8w	Scratch area
reserved	0000	10w	Reserved area

CHAPTER 10:
DISK DATA STRUCTURES

Introduction

This section will describe the data structures MOS uses for disk/diskette support. These structures serve various functions, such as describing the media, defining logical drives, keeping track of allocated disk space, and loading other data.

Most of these structures are common to all types of media. Some pertain only to fixed disks. The first three are used by all media types. The remaining structure pertains to fixed disks only. The structures are:

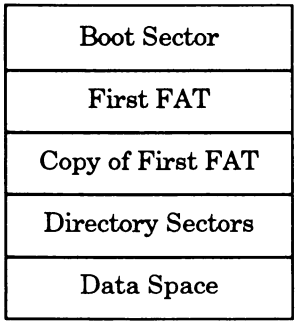
1. The File Allocation Table (FAT)
2. The Disk Directory
3. The Boot Sector
4. The Master Boot Record

In this discussion of MOS's disk structures, the term disk refers to both fixed disks and diskettes unless otherwise noted.

MOS works with logical disks or volumes. This helps MOS maintain its device independence. MOS's internal disk and diskette device drivers provide the translation from logical volumes to physical disks.

The basic unit of disk space is the sector. Normally, the size of a logical sector is the same as a physical sector. MOS works with logical sectors, which are referenced by numbers. For example, logical sector 0 is always the boot sector. The logical sector number does not always directly correspond to a physical sector number. Physical sectors are defined by a "three-dimensional" number. This number is made up of the head, cylinder and sector numbers used by the ROM-BIOS.

When the FORMAT command is executed on a logical drive, it checks the media for errors and builds the logical definition of the disk. The logical disk structure is as follows:



The boot sector is sector 0. The FATs and the directory can occupy a variable number of sectors.

Boot Sector

This 512 byte sector is always the first sector on the disk. Its purpose is two-fold. First, it contains the code necessary to load MOS into memory. Second, it contains the BIOS Parameter Block. (See the chapter on Device Drivers for information on the BPB).

File Allocation Table (FAT)

MOS uses the FAT to manage the allocation and de-allocation of disk storage. You can think of the FAT as a map that defines which areas of the disk are allocated and which are free.

MOS supports 12-bit and 16-bit FATs. If a disk has less than 20740 sectors, MOS uses 12-bit FAT entries. For larger volumes, MOS uses 16-bit FAT entries. 12-bit FAT entries are 1.5 bytes long. 16-bit entries are two bytes long.

Each FAT entry represents a disk allocation unit or cluster. A cluster is an allocation of X sectors, where X is an integral power of 2.

MOS reserves the first two FAT entries. The first byte indicates the disk format described in the following table. The remaining bytes are FFh.

Format Types Supported:

FF	2 sides. 8 sectors/track
FE	1 side. 8 sectors/track
FD	2 sides. 9 sectors/track
FC	1 side. 9 sectors/track
F9	2 sides. 15 sectors/track
F8	Fixed disk

The FAT entries listed in the previous table are provided only for compatibility with older systems. Note that 1.4M 3-1/2" floppy disks (Descriptor F0) are not represented in this table as they may ONLY be identified by the media descriptor byte in their BIOS Parameter Blocks (BPBs). See the chapter on device drivers for more information on BPBs.

Cluster number 2 (the third cluster) begins the disk's data area. Values in the FAT entries indicate whether the cluster is allocated, if it's a bad cluster, or if it's the end of the file. A zero means the cluster has not been allocated, it's free for use. FFF8-FFFF is an end-of-file indicator. An FFF7 indicates that the cluster is bad unless it's part of an allocation chain. Any other number is the number of the next cluster in the file.

To interpret 12-bit FAT entries, follow these steps. Get the file's starting cluster number from its directory entry. Multiply this value by 3 and divide by 2. Round the product down. It is now an offset into the FAT. Retrieve the word at the calculated offset. If the cluster number is even, keep bits 0-12. Otherwise keep bits 4-16. If the bits are FF8-FFF, you've reached end-of-file. Otherwise, it's the number of the next cluster.

16-bit FATs are somewhat easier. Use the starting cluster number from the directory entry. Multiply it by 2. Retrieve the word from the FAT using the calculated offset.

To convert a cluster number to a logical sector number, use the following formula:

```
((CN-2) * SPC) + DSN
```

Where CN = cluster number, SPC = sectors/cluster, DSN = first data sector number. To achieve the converse, reverse the formula:

```
((LSN-DSN) / SPC) + 2
```

Where LSN = logical sector number. All other values are the same as above.

The number of sectors the FAT occupies on the disk is fixed for floppy disks. For fixed disks, the FORMAT command uses a formula.

Floppy disks are set up as follows:

Disk	Heads	SPT	SPF	SPD	DIR Entries	SPC
5-1/4" 160K	1	8	1	4	64	1
5-1/4" 320K	2	8	1	7	112	2
5-1/4" 180K	1	9	2	4	64	1
5-1/4" 360K	2	9	2	7	112	2
5-1/4" 1.2M	2	15	7	14	224	1
3-1/2" 720K	2	9	3	7	112	2
3-1/2" 1.4M	2	18	9	14	224	1

SPT is sectors-per-track. SPC is sectors-per-cluster. SPF is sectors-per FAT. SPD is sectors-per-directory.

Hard disk FATs are set up as follows: Usually, 12-bit FATs use 8 sectors/cluster. 16-bit FATs use 4 sectors/cluster. These are the default values for disks < or = 32 megabytes.

A volume that is 10 meg or less will use 12-bit FATs and 8 sectors/cluster. Volumes greater than 10 meg use 4 sectors/cluster. For volumes larger than 32 meg, the cluster size varies depending upon the size of the desired volume.

The formula for sectors-per-FAT for fixed disks is:

(TS - RS - ((RDE * BPE) / BPS)) / (FC + ((BPS * SPC) / FES)).

where:

TS = total number of sectors on the disk
RS = the number of reserved sectors
RDE = the number of root directory entries, 512
BPE = the number of bytes/entry, 32
BPS = bytes/sector, usually 512
FC = the number of FATs, usually 2
SPC = sectors/cluster
FES = FAT entry size. (1.5 for 12-bit FATs, 2 for 16-bit FATs)

Directory Structure

The following table defines MOS's directory entry fields. Each entry is 32 bytes.

Byte	Description
0-7	ASCII filename. Byte 0 special
8-10	ASCII file extension
11	File attribute byte. See below
12-17	Reserved
18,19	File creation time
20,21	File creation date
22,23	Time last updated
24,25	Date last updated
26,27	Beginning cluster number
28-31	File size in bytes

The first byte of the filename has several interpretations. 00H - means this entry has never been used. 05H - the first character is actually an E5H. E5H - the file is deleted. 2EH is a period, for a subdirectory. If byte 2 of the filename is 2EH, the cluster field contains its parent's cluster number. If it's zero, the cluster field contains the current directory's cluster number.

The attribute field can have the following values:

01H - Read-only
02H - Hidden file
04H - Secured file
08H - Volume label
10H - Directory
20H - Archive bit

The archive bit is set when the file is changed. The EXPORT/IMPORT commands clear this bit when the file is backed up. If the hidden bit is set, the file or directory is skipped during normal directory displays.

The format of the time and date words for both the creation and update fields is:

TIME FIELD:

Bits 0-4: The number of 2 second increments. This is a binary number.

Bits 5-10: Minutes - binary number (0-59).

Bits 11-15: Hour - binary number(0-23).

DATE FIELD:

Bits 0-4: Day - binary (1-31).

Bits 5-8: Month - binary (1-12).

Bits 9-15: Year - binary (0-119) 0 = 1980.

The file size is stored low order word first.

Master Boot Record

This structure is specific to fixed disks. It serves two functions. First, it contains code which boots MOS from the hard disk. Possible errors the boot code could encounter are an invalid operating system, an invalid boot signature found when it loads a boot record from a logical partition, and an error from the BIOS during a read attempt.

Second, it allows division of a hard disk into multiple logical volumes. It indicates where on the hard disk each partition starts, how long it is, what format it's in, and whether or not the partition is bootable.

The partition table starts at offset 1BEH from the beginning of the Master Boot Record. The last 2 bytes of the sector are 55AAH, to identify it as a valid partition table. Each partition table entry is 16 bytes long. The table has room for up to 4 entries.

The entries of the table are defined as follows:

Byte 0: The Boot indicator. 00H - not bootable. 80H - bootable.

Byte 1: The beginning head number of the partition.

Byte 2: The beginning sector number of the partition.

Byte 3: The beginning cylinder number of the partition.

Byte 4: The system indicator. 00H = unknown operating system. 01H = MOS disk, 12-bit FAT. 04H = MOS disk, 16-bit FAT.

Byte 5: The ending head number of the partition.

Byte 6: The ending sector number of the partition.

Byte 7: The ending cylinder number of the partition.

Bytes 8 and 9: This word is the low order word of a double word which is the partition's beginning relative sector number. The value is relative to the beginning of the fixed disk. This is a physical sector count starting with head 0, cylinder 0, sector 1 and counting all the sectors on the disk until the beginning of the partition.

Bytes 10 and 11: This is the high-order word of the relative sector number.

Bytes 12 and 13: This word is the low-order word of the number of sectors in the partition.

Bytes 14 and 15: This is the high-order word of the number of sectors in the partition.

All of the above bytes are offset from the beginning of the partition table in the master boot record. So byte 0 of the first entry is at offset 1BEH from the beginning of the master boot record. The second partition entry begins a 1CEH from the beginning of the boot record, and so on for the remaining two partitions. The boot signature is a word at offset 1FEH from the beginning of the boot sector.

CHAPTER 11: PROGRAM FILE STRUCTURES

COM Files

COM files are executable modules with the following attributes:

1. The file ORGs at 100H.

2. All segment registers within the module have the same initial value, i.e. CS=DS=ES=SS. All other registers are initialized to 0, except CX and SP.

3. The file can be larger than 64K. However bytes 0FFFEH and 0FFFFH, offset from PSP:0 will not be set to 0. You can not terminate the program by ending the main routine with a RET instruction. Terminate the program using INT 21H, function 4CH.

4. The PSP is at CS:0 when the program is loaded.

5. No runtime relocation is performed on .COM files. They have no headers, i.e. they are direct memory images.

CHAPTER 11

EXE Files

EXE files have the following attributes:

1. Each file has a header as well as the actual load module itself.

2. The header contains the following information.

Offset	Description
0	2 bytes. Valid EXE file ID bytes. 4DH,5AH
2	Word. Image length mod 512
4	Word. File size. See below
6	Word. Relocation item count
8	Word. Size of header. In paragraphs
10	Word. Minimum paragraph count. See below
12	Word. Maximum paragraph count. See below
14	Word. SS displacement value. See below
16	Word. Program's initial SP value
18	Word. Checksum. See below
20	Word. Program's initial IP value
22	Word. CS displacement value. See below
24	Word. Byte count to 1st relocation value
26	Word. Overlay number. 0 for main part

The word value at offset 4 is the number of pages in the file including the header. A page is 512 bytes.

The value at offset 10 is the minimum number of paragraphs the program will need above its end when it is finally loaded. The value at offset 12 is the maximum paragraph count the program will need beyond its end once its loaded. A paragraph is 16 bytes.

Offset 14 is a word value giving the paragraph displacement of the stack segment within the load module.

Offset 18 is the negated sum of all the words in the file. Overflow is ignored.

Offset 22 is the word value giving the displacement, in paragraphs, of the code segment within the load module portion of the file.

Offset 24 is the byte displacement of the first relocation item within the file. This is used to locate the first item to be relocated.

3. The relocation table is a variable number of items which must have an address value added to them for the program to execute properly. Each table entry has two values. The first is a word offset and the second is a segment value. These point to an area within the load module that must be altered before the program gets control.

   The following steps accomplish this:

   a. A PSP is built right after the main portion of the program performing the load.

b. The header is read.

c. The size of the load module is calculated by subtracting the header size from the directory file size. The remainder (value at offset 2) is used to round the size down. Memory for the load module is allocated (load region). This region is where the load module is read into memory.

d. The relocation table items are read into a work area.

e. The relocation item's segment value is added to the load region value. This new value plus the relocation item's offset point to a word in the load module to which the load region value is added.

f. After all relocation table items are adjusted within the load module, SS:SP are initialized from the values within the header, with the load region value added to SS. The DS and ES register are set to the PSP segment address, and CS:IP are initialized from the header values with the load region value added to CS. Control is then passed to CS:IP.