FAT16 Structure Information - Written by Jack Dobiash

Updated : June 17th, 1999

Looking for FAT32 Info? Go here.
Looking for Informaton on how to Read and Write to your Hard Drive? Go here.

I've written this page for anyone who wishes to write software that can do low-level reading and writing of a hard drive, and needs to know what the underlying structure of a FAT16 Drive is, in order to interpret the information properly. Basically I've searched all over the web, and have compiled this information in one spot. Hopefully it can be of use to someone. I don't guarantee that all of this information is correct or complete, but so far is seems to have been working for me.

A lot of the number references I've made in this document are in Hexadecimal. Any that are have an 'h' after them. Also, just in case my terminology may be different from yours; a 'WORD' is 2 Bytes and a 'DOUBLE WORD' is 4 Bytes.

Master Boot Record

The Master Boot Record is the same for pretty much all Operating Systems. It is located on the first Sector of the Hard Drive, at Cylinder 0, Head 0, Sector 1. It is the first piece of code that your computer runs after it has checked all of your hardware (POST) and turned control of loading software over the hard drive. It also contains the partition table, which defines the different sections of your hard drive. Basically if anything happens to this little 512 byte section, your hard drive is brain dead. Kinda scary, eh? :)

Offset	Description	Size
000h	Executable Code (Boots Computer)	446 Bytes
1BEh	1st Partition Entry (See Next Table)	16 Bytes
1CEh	2nd Partition Entry	16 Bytes
1DEh	3rd Partition Entry	16 Bytes
1EEh	4th Partition Entry	16 Bytes
1FEh	Executable Marker (55h AAh)	2 Bytes

Partition Entry (Part of MBR)

Offset	Description	Size
00h	Current State of Partition (00h=Inactive, 80h=Active)	1 Byte
01h	Beginning of Partition - Head	1 Byte
02h	Beginning of Partition - Cylinder/Sector (See Below)	1 Word

04h	Type of Partition (See List Below)	1 Byte
05h	End of Partition - Head	1 Byte
06h	End of Partition - Cylinder/Sector	1 Word
08h	Number of Sectors Between the MBR and the First Sector in the Partition	1 Double Word
0Ch	Number of Sectors in the Partition	1 Double Word

Cylinder/Sector Encoding

I guess back in the days of 10MB hard drives and 8086's, code was at a premium. So they did everything they could to preserve space. Unfortunately now we have to live with it, but luckily they created new ways of translating the system so the 1024 Cylinder Limit (2^10) isn't too big of a problem, for newer computers, at least. Older ones usually need some sort of Disk Overlay program to make them see the whole hard drive.

Anyway, to get the Sector out of this, you need to apply an AND mask ($3F) to it. To get the Cylinder, you take the high byte and OR it with the low byte that has been AND masked with ($C0) and then Shifted Left Two. It's not very easy to explain, so I'll just show you how I did it with two routines I made (In Pascal) for Encoding and Decoding the Cylinder/Sector. Hopefully even if you don't know Pascal you'll be able to read it.

```
Function CylSecEncode(Cylinder, Sector : Word) : Word;
Begin
   CylSecEncode := (Lo(Cylinder) shl 8) or (Hi(Cylinder) shl 6) or Sector;
End;

Procedure CylSecDecode(Var Cylinder, Sector : Word; CylSec : Word);
Begin
   Cylinder := Hi(CylSec) or ((Lo(CylSec) and $C0) shl 2);
   Sector := (CylSec and $3F);
End;
```

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Cylinder Bits 7 to 0								Cylinder Bits 9+8		Sector Bits 5 to 0					

Partition Type Listing

There are more than just these shown, but I've only included that ones relevant to MS Operating Systems.

Value	Description
00h	Unknown or Nothing
01h	12-bit FAT
04h	16-bit FAT (Partition Smaller than 32MB)

05h	Extended MS-DOS Partition
06h	16-bit FAT (Partition Larger than 32MB)
0Bh	32-bit FAT (Partition Up to 2048GB)
0Ch	Same as 0BH, but uses LBA_1 13h Extensions
0Eh	Same as 06H, but uses LBA_1 13h Extensions
0Fh	Same as 05H, but uses LBA_1 13h Extensions

Reading Multiple Partitions

Since FAT16 is limited to 2GB per partition, drives that use it tend to have multiple partitions. The first partition is the Primary Partition, and everything else is stored in the Extended Partition. It's a little tricky when it comes to reading those extra partitions though (not a lot, just a little). The first record in the partition table shows where the Primary partition is (how big it is, where it starts, and where it ends). The second entry in the partition table shows where the Entire Extended Partition is (which may include more than just one partition). To read any more partitions, you go to the where it says the Extended Partition starts, and read the first sector. It acts just like the MBR. It'll have blank where the code is supposed to be, and in the partition table it will have for it's first entry the next Partition in the Drive, and if there are anymore, there will be another Extended partition, just like before. However, all references to Sector Numbers are made using the that new MBR point as the reference, making it a virtual drive. Just incase this doesn't make much sense (and by the way I explain things I can understand if it doesn't), let me show you how a drive with three partitions is setup.

MBR of Whole Drive

Entry #1 - Points to Partition #1
Entry #2 - Points to the Entire Extended Partition

You would read the first sector of that Extended Partition, and see another MBR Structure.

MBR of Extended Partition

Entry #1 - Points to Partition #2
Entry #2 - Points to Rest of Extended Partition after Partition #2

Now, all references to Sector Numbers (most specifically the entry at Offset 08h) in those Entries wouldn't be referenced from the start of the drive, but from the start of the Extended Partition. However, the CHS (Cylinder, Head, Sector) numbers would still be right.

Once again, you would read the first sector of that Extended Partition, and see the next MBR.

MBR of Rest of Extended Partition

Entry #1 - Points to Partition #3
No Entry #2, since this was the Last Partition

If there were another partition, the pattern would continue just like before, until the last one was reached.

FAT16 Boot Record

This information is located in the first sector of every partition.

Offset	Description	Size
00h	Jump Code + NOP	3 Bytes
03h	OEM Name	8 Bytes
0Bh	Bytes Per Sector	1 Word
0Dh	Sectors Per Cluster	1 Byte
0Eh	Reserved Sectors	1 Word
10h	Number of Copies of FAT	1 Byte
11h	Maximum Root Directory Entries	1 Word
13h	Number of Sectors in Partition Smaller than 32MB	1 Word
15h	Media Descriptor (F8h for Hard Disks)	1 Byte
16h	Sectors Per FAT	1 Word
18h	Sectors Per Track	1 Word
1Ah	Number of Heads	1 Word
1Ch	Number of Hidden Sectors in Partition	1 Double Word
20h	Number of Sectors in Partition	1 Double Word
24h	Logical Drive Number of Partition	1 Word
26h	Extended Signature (29h)	1 Byte
27h	Serial Number of Partition	1 Double Word
2Bh	Volume Name of Partition	11 Bytes
36h	FAT Name (FAT16)	8 Bytes
3Eh	Executable Code	448 Bytes
1FEh	Executable Marker (55h AAh)	2 Bytes

FAT16 Drive Layout

Offset	Description
Start of Partition	Boot Sector
Start + # of Reserved Sectors	Fat Tables
Start + # of Reserved + (# of Sectors Per FAT * 2)	Root Directory Entry

Start + # of Reserved + (# of Sectors Per FAT * 2) + ((Maximum Root Directory Entries * 32) / Bytes per Sector)	Data Area (Starts with Cluster #2)

Cluster Meaning (FAT Table Entries)

A Cluster is a Group of Sectors on the Hard Drive that have information in them. A 16K Cluster has 32 Sectors in it (512*32=16384). Each Cluster is given a spot in the FAT Table. When you look at an Entry in the FAT, the number there tells you whether or not that cluster has data in it, and if so, if it is the end of the data or there is another cluster after it. All Data on a Partition starts with Cluster #2 (Right after Root Directory). If the FAT Entry is 0, then there is no data in that cluster. If the FAT Entry is FFFFh, then it is the last entry in the chain.

FAT Code Range	Meaning
0000h	Available Cluster
0002h-FFEFh	Used, Next Cluster in File
FFF0h-FFF6h	Reserved Cluster
FFF7h	BAD Cluster
FFF8h-FFFF	Used, Last Cluster in File

Directory Table

Another aspect when looking at a File System at Low Level is the Directory Table. The Directory Table is what stores all of the File and Directory Entries. Someone else has already written a good resource for this information on the net, so go here to look at it. The link doesn't work anymore, but luckily I saved the page a while back, so i'll just post it on my site.

Footnotes

1 - LBA = Logical Block Addressing - Uses the Int 13h Extensions built into newer BIOS's to access data above the 8GB barrier, or to access strickly in LBA mode, instead of CHS (Cylinder, Head, Sector).

Home Reference Point Software FAT32 Structure Information FAT16 Structure Information Disk Access Information
About Me Links Dobiash?