FAT Boot Sector

Introduction

This paper describes the FAT boot sector. I will try to unify the boot sectors of the FAT32, FAT16, and FAT12 file systems.

Structure

Boot sector is always the very first sector in the partition. Validity check is performed by comparing the 16 bit word at offset `1FE` to `AA55`. For FAT systems, this sector always contains the Bios Parameter Block (BPB) at offset `0B`. The structure of the boot sector is below.

Offset in Boot Sector	*Length in Bytes*	*Mnemonic*
03	8	OEM_Identifier
0B	2	BytesPerSector
0D	1	SectorsPerCluster
0E	2	ReservedSectors
10	1	NumberOfFATs
11	2	RootEntries
13	2	NumberOfSectors
15	1	MediaDescriptor
16	2	SectorsPerFAT
18	2	SectorsPerHead
1A	2	HeadsPerCylinder
1C	4	HiddenSectors
20	4	BigNumberOfSectors
24	4	BigSectorsPerFAT
28	2	ExtFlags
2A	2	FSVersion
2C	4	RootDirectoryStart
30	2	FSInfoSector
32	2	BackupBootSector
34	12d	Reserved

`OEM_Identifier` is the eight-byte ASCII string that identifies the system that formatted the disk. All eight characters are meaningful. Spaces or zeroes are appended if the name is less than eight characters long. As any other OEM name, this string is nice to display near to the other disk information, but it is absolutely useless for any other purpose.

`BytesPerSector` is how many bytes long the physical sector is. All disks I have ever seen contained 512 in this field. The gossips are, DOS supports disks with different sector sizes. Other gossips are, early versions of DOS silently set this field to 512 and ignored whatever the original value was. Anyway, as of today, you will not loose much if you compare this value to 512 and refuse to work with the disks that have a different sector size.

`SectorsPerCluster` is how many sectors are in one logical cluster. What is cluster was described earlier. This value should not be zero.

`ReservedSectors` are reserved, starting from the LBA sector `(0)` relative to current partition. LBA `(ReservedSectors)` is the beginning of the first FAT. For FAT12 and FAT16 this value is usually 1. For FAT32 it is 20h. At least one sector must always be reserved.

`NumberOfFATs` is the number of File Allocation Tables. This value is usually two. FATs are consecutive on the disk: the second copy of FAT goes right after the first copy. At least one copy of FAT should be present.

`RootEntries` contains the number of the entries in the root directory if root directory is fixed. It is zero if the root directory is not fixed. FAT32 disks should contain zero in this field, indicating that the root directory can be arbitrarily long. Otherwise, this field usually contains 512. Each directory entry takes up 32 bytes. To avoid wasting space, `RootEntries*32` should be divisible by `BytesPerSector`.

`NumberOfSectors` is the total number of sectors on the disk. If the number of sectors is greater than 65535, then this field is set to zero and the dword at `BigNumberOfSectors` contains the actual number of sectors. By `NumberOfSectors` I will refer to that of `NumberOfSectors` and `BigNumberOfSectors`, which is used. Note that this field should contain the same or lesser value as the corresponding field in the partition table. If the values are not equal, the lesser of them should be used. `NumberOfSectors` should be large enough to contain at least the reserved sectors, all FAT copies, and the root directory, if any. Disk layout is described below.

`MediaDescriptor` describes the device:

Value	*DOS version*	*Meaning*
FF	1.1	5 1/4 floppy, 320KB
FE	1.0	5 1/4 floppy, 160KB
FD	2.0	5 1/4 floppy, 360KB
FC	2.0	5 1/4 floppy, 180KB
F9	3.0	5 1/4 floppy, 1.2MB
F9	3.2	3 1/2 floppy, 720KB
F8	2.0	Any Hard Drive
F0	3.3	3 1/2 floppy, 1.44MB

As of today, no other values are defined.

`SectorsPerFAT` contains the number of sectors in one FAT. This field is zero for FAT32 drives, and `BigSectorsPerFAT` contains the actual value. Note that the Microsoft FAT32 boot loader will not work with the FAT32 drives if `SectorsPerFAT` is not zero, and FAT12 and FAT16 loaders will not work with the drives if `SectorsPerFAT` is zero. As with `NumberOfSectors`, by `SectorsPerFAT` I will mean the appropriate value. It goes without saying, FAT should be long enough to contain the information about all clusters on the disk.

`SectorsPerHead` is the number of sectors grouped under one head. `HeadsPerCylinder` is also what you think it is. If this partition is a CHS partition, these values must be the same as those returned by BIOS. If they are not the same, the disk was misconfigured and the partition should not be used. Note that the Microsoft boot loader alteres the BIOS Diskette Parameters table by setting the `SectorsPerTrack` field of this structure to `SectorsPerHead` read from the boot disk. The values in these fields do not matter for LBA partitions.

`HiddenSectors` is the number of sectors between the beginning of this partition and the partition table. This field should be the same as "number of sectors preceding the partition" in the partition table. Note that it is not necessarily the physical LBA address of the first sector because there exist secondary partitions. If `HiddenSectors` is not the same as in the partition table, boot sector was corrupted and the partition should not be used. Also note that the high word contains garbage for old versions of DOS.

`ExtFlags`, and all fields described below, are defined only for FAT32 disks. They are defined differently for FAT12 and FAT16 (that issue is discussed below). If the left-most bit of `ExtFlags` value is set then only the *active* copy of FAT is changed. If the bit is cleared then FATs will be kept in synchronization. Disk analyzing programs should set this bit only if some copies of the FAT contain defective sectors. Low four bits define which copy should be active. As you see, only the first sixteen copies of the FAT may be selected active, so the disk is usable if and only if among the first sixteen copies at least one is usable. For this flag, FAT numbers start from zero. Sanity check insures that the active FAT is less than `NumberOfFATs`.

`FSVersion` is the version of the file system. The high byte is the major version, the low byte is the minor version. Both are set to zero on my Windows 95 OSR2. I do not think that this value should be checked. However, the Microsoft boot loader does check it *in certain cases*, and it complains if it is not zero.

`RootDirectoryStart` contains the number of the first cluster for the root directory. Yes, finally the root directory became stored like any other directory, in the cluster chain. This also implies that it may grow as needed. The value in this field should be at least two. It is two on my system.

`FSInfoSector` is the sector number for the file system information sector. This sector is new to FAT32. Its structure is below:

Offset in Sector	*Size*	*Meaning*
`00`	`4`	`Signature, should be 41615252h (?)`
`1E4`	`4`	`Signature, should be 61417272h`
`1E8`	`4`	`Number of free clusters on the drive, or -1 if unknown`

1EC	4	Number of the most recently allocated cluster
1F0	12d	Reserved
1FE	2	Signature AA55

All the other bytes are set to zero. As you see, these values are introduced to improve performance. Make sure that `FSInfoSector` is at least one and it lies within the reserved disk area. Also make sure that this value is not the same as `BackupBootSector`. If it does not satisfy these conditions, do not use this sector, but the file system should still be usable. Do not use the information in this sector if its signature is incorrect. Note that only the second signature is documented by Microsoft. Normally, the number of free clusters is checked only by special disk analysis programs. `FSInfoSector` is usually one.

`BackupBootSector` is the sector number for the backup copy of the boot sector. This copy can be used if the main copy was corrupted. It is also nice to compare the two copies on startup. If they do not match, a warning should be issued. They may not be in tact because of corruption or a boot virus. If this field contains zero or the number greater than or equal to `ReservedSectors` or the same value as `FSInfoSector`, the backup sector should not be used.

> My experience says, there are two complete copies of the boot information. The first copy starts from the very first sector in the partition and is three sectors long. The first sector is being described now. The second sector is usually the FS Information Sector, as described above. The third sector is some extra code for the boot loader. The third sector also contains the `AA55` signature at `1FE`. Then, starting from `BackupBootSector`, goes another triple. The first and the third sectors of this triple are identical to those of the main one. The second sector also contains the FS Information Sector, but it is not kept in synchronization with the main FS Information Sector. It is not documented by Microsoft. Its "Number of Free Clusters" is set to `-1` and "The Most Recently Allocated Cluster" is two, which suggests that it was touched only by the formatting program. All the other reserved sectors contain zeroes.

Notes

Note that some of the values in the boot sector have a different meaning in FAT12/16 systems as compared to the FAT32 system. These fields form the Extended BIOS Parameter Block, or EBPB. EBPB is the same for FAT12/16 and FAT32, but it starts from the different offsets in the boot sector. For the sake of completeness I will describe it here:

Offset for FAT12/16	*Offset for FAT32*	*Length in Bytes*	*Meaning*
24	40	1	BIOS drive number
25	41	1	Reserved
26	42	1	Extended Boot Record signature = 29h
27	43	4	Serial Number
2B	47	11	Volume label
36	52	8	System Identifier (FAT12, FAT16, or FAT32)

In my humble opinion, you do not need these fields even for FAT12 or FAT16. However, there is one compatibility issue. If you have detected that FAT12 or FAT16 is used, and the extended boot record signature is not `29h`, you should ignore all the values in the BPB starting from the offset `1E`. This is not a typo: `HiddenSectors` is really split by half, and if the extended boot record signature is incorrect then only the lower word of `HiddenSectors` is valid.

Sanity check might also insure that System Identifier is correct for the detected file system. It should be `"FAT12"`, `"FAT16"`, or `"FAT32"` according to the file system. Strings are padded with spaces to fit in eight bytes.

Finally, the presense of the EBPB in FAT32 is not documented by Microsoft.

Now it is high time to explain how clusters are mapped to sectors. First, consider the layout of the FAT disk:

LBA Location	*Length in Sectors*	*Description*
0	ReservedSectors	Boot Sector(s), File System Info Sector
ReservedSectors	NumberOfFATs*SectorsPerFAT	File Allocation Tables
RootStart *	(RootEntries*32)/BytesPerSector	Root Directory, if any
ClustersStart	NumberOfClusters*SectorsPerCluster	Data Clusters

* Note: `RootStart` does not make sense for FAT32 partitions. However, the formulae below are still valid for FAT32.

Some of the values in the table are in the BPB. Let us calculate the rest of them:

```
RootStart=ReservedSectors+NumberOfFATs*SectorsPerFAT
ClustersStart=RootStart+(RootEntries*32) div BytesPerSector
* Note: if (RootEntries*32) mod BytesPerSector then ClustersStart=ClustersStart+1
NumberOfClusters=2+(NumberOfSectors-ClustersStart) div SectorsPerCluster
```

To convert cluster address to LBA address use the formula:

```
LBA=ClustersStart+(Cluster-2)*SectorsPerCluster
```

Now we are ready to detect which file system is used.

- If `NumberOfClusters<4087` then FAT12 is used.
- If `4087<=NumberOfClusters<65,527` then FAT16 is used.
- If `65,527<=NumberOfClusters<268,435,457` then FAT32 is used.
- Otherwise, none of the above is used.

I fairly warn you: I have not yet found two sources that agree on these values. So, be careful if the number of clusters is close to the border value. One might have noticed that the maximum `NumberOfClusters` for FAT32 looks odd. Since only 28 of 32 bits are currently used, the FAT32 partition can have no more than 268,435,456 clusters. Looking at `NumberOfClusters` is the **only** recommended by Microsoft way of detecting the FAT entry size.

Since the type of the file system depends on the cluster size, which can be set arbitrarily by the formatting program, let me introduce the Microsoft recommendations. Note that these are only guidelines. The partition size is **not** used directly to determine the FAT type. It is first converted to `NumberOfClusters` as described above.

Partition Size	*Type of FAT*
10MB or less	FAT12
10MB through 512MB (?)	FAT16
512MB (?) through 2TB	FAT32

The following table contains the ranges for the partition sizes. They do overlap.

Min Partition Size	*Max Partition Size*	*Type of FAT*
1.5KB	510.75MB	FAT12
2.0435MB	8190.75MB	FAT16
~32MB	32TB	FAT32

Author: Alex Verstak 3/10/1998