Specifications

FAT16 File System

Used: On machines with small harddrives running MS-DOS, Windows 95/98

Introduction

This is the 16-bit version of the FAT file system. The 16-bit part describes the way units are allocated on the drive. The FAT16 file system uses a 16-bit number to identify each allocation unit (called cluster), and this gives it a total of 65.536 clusters. The size of each cluster is defined in the boot sector of the volume (volume = partition). The File System ID number usually associated with FAT16 volumes are 04h and 06h. The first is used on volumes with less than 65536 sectors (typical this is on drives less than 32 Mb in size), and the latter one is used on volumes with more than 65536 sectors. There is also another variant which is used with the LBA address mode, that variant has a File System ID of 0Eh. **Note:** I do not know if the LBA variant is different from the CHS type. So far I don't se why anything should be changed to support LBA addresses.

Basic Structure

The FAT16 file system structure contains the following regions:

FAT16 File System Structure

Region
Reserved Region (incl. Boot Sector)
File Allocation Table (FAT)
Root Directory
Data Region

The first sector (boot sector) contain information which is used to calculate the sizes and locations of the other regions. The boot sector also contain code to boot the operating system installed on the volume. The data region is split up into logical blocks called clusters. Each of these clusters has an accompanying entry in the FAT region. The cluster specific entry can either contain a value of the next cluster which contain data from the file, or a so called End-of-file value which means that there are no more clusters which contain data from the file. The root directory and its sub-directories contain filename, dates, attribute flags and starting cluster information about the filesystem objects.

Boot Sector

The first sector in the reserved region is the boot sector. Though this sector is typical 512 bytes in can be longer depending on the media. The boot sector typical start with a 3 byte jump instruction to where the bootstrap code is stored, followed by an 8 byte long string set by the creating operating system. This is followed by the BIOS Parameter Block, and then by an Extended BIOS Parameter Block. Finally the boot sector contain boot code and a signature.

Structure of the FAT16 Boot sector

Part	Offset	Size	Description
Code	0000h	3 bytes	Code to jump to the bootstrap code.
OS Name	0003h	8 bytes	Oem ID - Name of the formatting OS
	000Bh	2 bytes	Bytes per Sector
	000Dh	1 bytes	Sectors per Cluster - Usual there is 512 bytes per sector.
	000Eh	2 bytes	Reserved sectors from the start of the volume.
	0010h	1 bytes	Number of FAT copies - Usual 2 copies are used to prevent data loss.
	0011h	2 bytes	Number of possible root entries - 512 entries are recommended.
BIOS Para- meter Block	0013h	2 bytes	Small number of sectors - Used when volume size is less than 32 Mb.
	0015h	1 bytes	Media Descriptor
	0016h	2 bytes	Sectors per FAT
	0018h	2 bytes	Sectors per Track
	001Ah	2 bytes	Number of Heads
	001Ch	4 bytes	Hidden Sectors
	0020h	4 bytes	Large number of sectors - Used when volume size is greater than 32 Mb.
	0024h	1 bytes	Drive Number - Used by some bootstrap code, fx. MS-DOS.
	0025h	1 bytes	Reserved - Is used by Windows NT to decide if it shall check disk integrity.
Ext. BIOS Para- meter Block	0026h	1 bytes	Extended Boot Signature - Indicates that the next three fields are available.
	0027h	4 bytes	Volume Serial Number
	002Bh	11 bytes	Volume Label - Should be the same as in the root directory.
	0036h	8 bytes	File System Type - The string should be 'FAT16 '
Code	003Eh	448 bytes	Bootstrap code - May schrink in the future.
Sig.	01FEh	2	Boot sector signature - This is the AA55h signature.

BIOS Parameter Block

The BIOS Parameter Block contains basic information about the overall structure of the FAT file system. That is information such as sector and cluster size, location information of FAT copies, the root directory size etc..

Bytes Per Sector
This value is the number of bytes in each physical sector. The allowed values are: 512, 1024, 2048 or 4096 bytes. A lot of code outthere are assuming 512 bytes per sectors, so any other values can give compatibility problems.

Sectors Per Cluster
This is the number of sectors per cluster. The allowed values are: 1, 2, 4, 8, 16, 32 or 128. But de-facto is that most combinations of 'BytesPerCluster' * 'SectorsPerCluster' which gives a total value over 32 Kb per cluster, are not supported on many system.

Reserved Sectors
Since the reserved region always contain the boot sector a zero value in this field is not allowed. The usual setting of this value is 1. The value is used to calculate the location for the first sector containing the FAT.

Number of FAT copies
This is the number of FAT copies in the file system. The recommended value is 2 (and then have two FAT copies), but other values are validm though they may not be supported on some system. The usage of two copies are to prevent data loss if one or part of one FAT copy is corrupted.

Root Entries Count
This value contain the number of entries in the root directory. Its recommended that the number of entries is an even multiple of the BytesPerSector values. The recommended value for FAT16 volumes is 512 entries (compatibility reasons).

Small Number of Sectors
This field states the total number of sectors in the volume. That includes the number of sectors occupied by the four regions which the FAT16 file system consist of. For FAT16 volumes that use less than 65536 sectors this field is used. The File System Id in the MBR is then 04h. For FAT16 volumes that use more the 65535 sectors the Large Number of Sectors field is used and this one should be set to 0h.

Media Descriptor
These are the possible media descriptors values in the FAT boot sector. **Note:** This is the same value which must be in the low byte in the first entry of the FAT.

Media Descriptors

Hex Value	Capacity	Physical Format
F0	2.88 MB	3.5-inch, 2-sided, 36-sector
F0	1.44 MB	3.5-inch, 2-sided, 18-sector
F8	?	Fixed disk
F9	720 KB	3.5-inch, 2-sided, 9-sector
F9	1.2 MB	5.25-inch, 2-sided, 15-sector
FA	?	?
FB	?	?
FC	180 KB	5.25-inch, 1-sided, 9-sector
FD	360 KB	5.25-inch, 2-sided, 9-sector
FE	160 KB	5.25-inch, 1-sided, 8-sector
FF	320 KB	5.25-inch, 2-sided, 8-sector

Sectors Per FAT
This is the number of sectors occupied by one copy of the FAT.

Sectors Per Track
This value is used when the volume is on a media which have a geometry, that is when the LBA number is broken down into a Cylinder-Head-Sector address. This field represents the multiple of the max. Head and Sector value used when the volume was formatted. The field itself is used to check if the LBA to CHS translation has changed, since the formatting. And for calculating the correct Cylinder, Head and Sector values for the translation algorithm.

Number of Heads
This value is used when the volume is on a media which have a geometry, that is when the LBA number is broken down into a Cylinder-Head-Sector address. This field represents the Head value used when the volume was formatted. The field itself is used to check if the LBA to CHS translation has changed, since the formatting. And for calculating the correct Cylinder, Head and Sector values for the translation algorithm.

Hidden Sectors
When the volume is on a media that is partitioned, this value contains the number of sectors preceeding the first sector of the volume.

Large Number of Sectors
This field states the total number of sectors in the volume. That includes the number of sectors occupied by the four regions which the FAT16 file system consist of. For FAT16 volumes that use more than 65535 sectors this field is used. The File System Id in the MBR is then 06h. For FAT16 volumes that use less than 65536 sectors the Small Number of Sectors field is used and this one should be set to 0h.

Extended BIOS Parameter Block
The Extended BIOS Parameter Block contains information that is only used in the FAT16 file system.

Drive Number

This is the int 13h drive number of the drive. The value 00h is used for the first floppy drive and the value 80h is used for the first harddrive. MS-DOS's bootstrap uses this value to find the correct drive.

Reserved
Reserved byte. It was original used to store the cylinder on which the boot sector is located. But Windows NT uses this byte to store two flags. The lowest bit would indicates that a check disk should be run at boot time, and the second lowest flag indicates that a surface scan should be run.

Extended Boot Signature
If this byte contain a value of 29h it indicates that the following three fields are available.

Volume Serial Number
This value is a 32 bit random number, which, combined with the volume label, makes is possible to track removable media and check if the correct one is inserted.

Volume Label
This 11 byte long string should match the volume label entry in the root directory. If no such entry is available this field should contain the string 'NO NAME ' (11 bytes long string). When updating the volume label, both, this field and the entry in the root directory should be updated.

File System Type
This 8 byte long string should be used for informational display only. Thats because its sometime incorrectly set. The field should contain the string 'FAT16 ' (8 bytes long string).

Bootstrap Code
The bootstrap code is different between operating system and versions which are intended to be loaded of this FAT16 volume. The responsability of the bootstrap code is to continue the boot sequence. If ex. MS-DOS is installed the bootstrap code will locate the file IO.SYS in the file system, load part of it into memory and then jump to a specific entrypoint in IO.SYS. What the bootstrap code does is vary between operating system.

Boot Sector Signature
The word at offset 1FEh should contain the signature AA55h. This will indicate to the BIOS that the sector is executable. The signature is also used by other applications when validating that the correct sector has been loaded. **Note:** No matter what the Bytes Per Sector value is this signature should always be at offset 1FEh.

File Allocation Table
The FAT structure contain linked lists of files in the file system. Any file or directory entry in a (sub)directory list contain a cluster number for the first chunk of the file/directory. This cluster number also has an associated entry in the FAT. At this entry in the FAT a single word value either points to the next cluster/chunk or it contain an End-of-file value. These are the valid values:

Valid FAT16 values

Value	Description
0000h	Free cluster
0001h - 0002h	Not allowed
0003h - FFEFh	Number of the next cluster
FFF7h	One or more bad sectors in cluster
FFF8h - FFFFh	End-of-file

Each FAT copy start with a value of FFxxh for the first cluster, where xx is equal to the Media Descriptor field in the BIOS Parameter Block. The FAT entry for the second cluster is set to the End-of-file value (FFFFh). The two highest bits of this value may be used for dirty volume flags in the following way.

FAT Entry Value for 2nd cluster

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
C	E	These bits must be set														0000h

C = If clear then the volume may be dirty because it was not cleanly dismounted. If the bit is set then the volume is clean.

E = If clear a read/write error was encountered during the last time the volume was mounted. If the bit is set then no read/write error was encountered.

It should be noted that the last entry in the FAT should be calculated by taking the number of clusters in the data area, and not relying on the Sectors Per FAT field in the BIOS Parameter Block. This is because the number of entries in the FAT doesn't have to be an even multiple of the bytes per sector value.

The maximum size of each FAT copy is 128 Kb (2 * 65536).

Directory Entry Structure
Each of the entries in a directory list is 32 byte long. The only directory which is in a fixed location is the root directory. This is also the only directory which may contain an entry with the Volume Label attribute set. The size of the root directory is defined in the BIOS Parameter Block.

Sub-directories are created by allocating a cluster which then are cleared so it doesn't contain any directory entries. Two default entries are then created: The '.' entry point to the directory itself, and the '..' entry points to the parent directory. If the contents of a sub-directory grows beyond what can be in the cluster a new cluster is allocated in the same way as for files.

This is the format of the directory entries:

Structure of the Directory Entries

Offset	Size	Description
00h	8 bytes	Filename
08h	3 bytes	Filename Extension
0Bh	1 bytes	Attribute Byte
0Ch	1 bytes	Reserved for Windows NT
0Dh	1 bytes	Creation - Millisecond stamp (actual 100th of a second)
0Eh	2 bytes	Creation Time
10h	2 bytes	Creation Date
12h	2 bytes	Last Access Date
14h	2 bytes	Reserved for FAT32
16h	2 bytes	Last Write Time
18h	2 bytes	Last Write Date
1Ah	2 bytes	Starting cluster
1Ch	4 bytes	File size in bytes

Filename & Extension

The filename is 8 byte long. Shorter names must be trailing padded with space bytes (ASCII: 20h). The extension is 3 byte long and shorter names also have to be trailing padded. The characters allowed in the filename and extension are basicly the uppercase letters of the english alphabet, plus the digits 0 to 9.

The first byte in the filename is treated special. The following rules apply:

1. A value of 00h is interpertated as 'stop the search, there is no more entries in this directory'.
2. A value of 05h is should be replaced with the ASCII character E5h. The character is used in Japan.
3. Must not contain the value 20h.
4. A value of E5h is interpertated as 'the entry is free'.
5. Any of the values mentioned below are allowed.

The following characters are not allowed in the filename or its extension:

1. Any character below 20h, except the 05h character.
2. Any character in the following list: 22h ("), 2Ah (*), 2Bh (+), 2Ch (,), 2Eh (.), 2Fh (/), 3Ah (:), 3Bh (;), 3Ch (<), 3Dh (=), 3Eh (>), 3Fh (?), 5Bh ([), 5Ch (\), 5Dh (]), 7Ch (|).

For compatibility reasons it is recommended to limit the characters to:

1. Any uppercase characters from A to Z.
2. Any digit from 0 to 1.
3. Any of the following characters: #, $, %, &, ', (,), -, @

Attribute Byte

The attribute byte defines a set of flags which can be set for directories, volume name, hidden files, system files, etc. These are the flags:

Flags in the Attribute byte

7	6	5	4	3	2	1	0	
Reserved		A	D	V	S	H	R	0000h

Read Only

This flag is used to prevent programs from not automatically overwriting or deleting this entry.

Hidden

This flag indicates to the system that the file should be hidden when doing normal directory listings. But in a lot of programs this can be overwritten by the user.

System

This flag indicates that the file/directory is important for the system, and shouldn't be manipulated without any concern.

Volume Name

When this flag is set, the directory entry is not pointing to a file, but to nothing. Thus the the Starting cluster must point the cluster 0. The only information used from this entry is the filename (8 bytes) plus the filename extension (3 bytes). These bytes form an 11 byte long volume label (without any .) There may be only **one** valid entry in the entire volume with this flag set. This entry must be in the root directory and preferably among the first entries, if not, then MS-DOS can have trouble displaying the right volume label if there are long file names present. This volume name should be the same as the one in the boot sector.

Directory

This flag is set, when an entry in the directory table is not pointing to the beginning of a file, but to another directory table. A sub-directory. The sub-directory is placed in the cluster, which the Starting Cluster field points to. The format of this sub-directory table is identical to the root directory table.

Achieve Flag
This flag is used by backup utilities. The flag is set when ever a file is created, renamed or changed. Backup utilities can then use this flag to determine which files that has been modified since the last backup.

Reserved for Windows NT
This byte is used by Windows NT. It set the value to 0 when the file is created and then never look at it again. For what purpose it uses it is unknown.

Creation Time - Millisecond
Due to size limitations this field (1 byte) only contains the millisecond stamp in counts of 10 milliseconds. Therefore valid values are between 0 and 199 inclusive.

Creation Time & Date
The 16 bit time field contain the time of day when this entry was created. The 16 bit date field contain the date when the entry was created. These two values never change. Both the time field and the date field are in a special format.

Last Access Date
This 16 bit field contain the date when the entry was last read or written to. In case of writes this field of cause contain the same value as the Last Write Date field. The date field is the same special format as the other dates.

Reserved for FAT32
This word is reserved for the FAT32 File System. When used in that file system it will contain the high word of the starting cluster. In the FAT16 File System this word should be set to 0.

Last Write Time
This 16 bit field is set to the time when the last write occured. When the entry is create this field and the Creation Time field should contain the same values. In case the entry is a directory entry this field should change when the contents of the sub-directory changes.

The field is in the special format described below:

Time Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Hours (0-23)							Minutes (0-59)					Seconds (0-29)				0000h

Note: Seconds are counted with 2 seconds interval, so a value of 29 in this field gives 58 seconds.

Last Write Date
This 16 bit field is set to the date when the last write occured. When the entry is create this field and the Creation Date field should contain the same values. In case the entry is a directory entry this field should change when the contents of the sub-directory changes.

The field is in the special format described below:

Date Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Years from 1980 (0-127 -> 1980-2107)							Month of year (1-12)					Day of month (1-31)				0000h

First Cluster
This 16-bit field points to the starting cluster number of entrys data. If the entry is a directory this entry point to the cluster which contain the beginning of the sub-directory. If the entry is a file then this entry point to the cluster holding the first chunk of data from the file.

File Size
This 32-bit field count the total file size in bytes. For this reason the file system driver must not allow more than 4 Gb to be allocated to a file. For other entries than files then file size field should be set to 0.

Calculation Algorithms

How to calculate the starting location of each region. The VolumeStart variable contain a LBA address of the first sector in the volume. On drives which are not partitioned the VolumeStart value is 0. On drives which are partitioned the VolumeStart variable contain the sector number at which the partition start.

How to calculate regions and their size (in sectors)

What to calculate	How to calculate it
ReservedRegion start =	VolumeStart
FATRegion start =	ReservedRegion + ReservedSectors
RootDirectoryRegion start =	FATRegion + (NumberOfFATs * SectorsPerFAT)
DataRegion start =	RootDirectoryRegion + ((RootEntiesCount * 32) / BytesPerSector)
ReservedRegion size =	ReservedSectors
FATRegion size =	NumberOfFATs * SectorsPerFAT
RootDirectoryRegion size =	(RootEntiesCount * 32) / BytesPerSector (Remember to round up, if there is a remainder)
DataRegion size =	TotalNumberOfSectors - (ReservedRegion_Size + FATRegion_Size + RootDirectoryRegion_Size)

Previous calculated values which are used again, are marked with this color.

How to calculate FAT related values

What to calculate	How to calculate it
Location of n'th FAT copy	ReservedRegion + (N * SectorsPerFAT)
Number of FAT entries	DataRegion_Size / SectorsPerCluster (Remember to round down if there is a remainder)
Which FAT sector contain the Nth cluster entry I need ?	Location of FAT copy + ((N * 2) / BytesPerSector)

Previous calculated values which are used again, are marked with this color.

How to calculate other values

What to calculate	How to calculate it
First sector of cluster N =	DataRegion + ((N - 2) * SectorsPerCluster)

Previous calculated values which are used again, are marked with this color.

Special Notes

When creating a FAT16 volume special care should be taken to ensure best compatibility. Following these rules ensure the best compatibility:

- A FAT16 partition may not have less than 4085 clusters or more than 65524 clusters.

Conclusion

The FAT family of file systems are relative simple file systems. The complexity can be enhanced by adding support for long file names, using the VFAT Long File Names. Also have a look at the 32 bit version of the FAT file system.