

EMS SD/MMC/FRAM *Application Note*

Mengenal SD Card & FAT16

Oleh: Tim IE

Secure Digital (SD) atau MultiMedia Card (MMC) seringkali digunakan sebagai sarana penyimpan data pada Personal Digital Assistant (PDA), kamera digital, dan telepon seluler (ponsel). Beberapa perintah dasar untuk SD Card juga dapat digunakan untuk MMC sehingga kita dapat menggunakan SD atau MMC. Format data pada SD maupun MMC umumnya menggunakan format FAT. FAT12 digunakan untuk kapasitas 16 MB ke bawah. FAT16 digunakan untuk kapasitas 32 MB hingga 2 GB. FAT32 digunakan untuk kapasitas di atas 2 GB (SDHC).

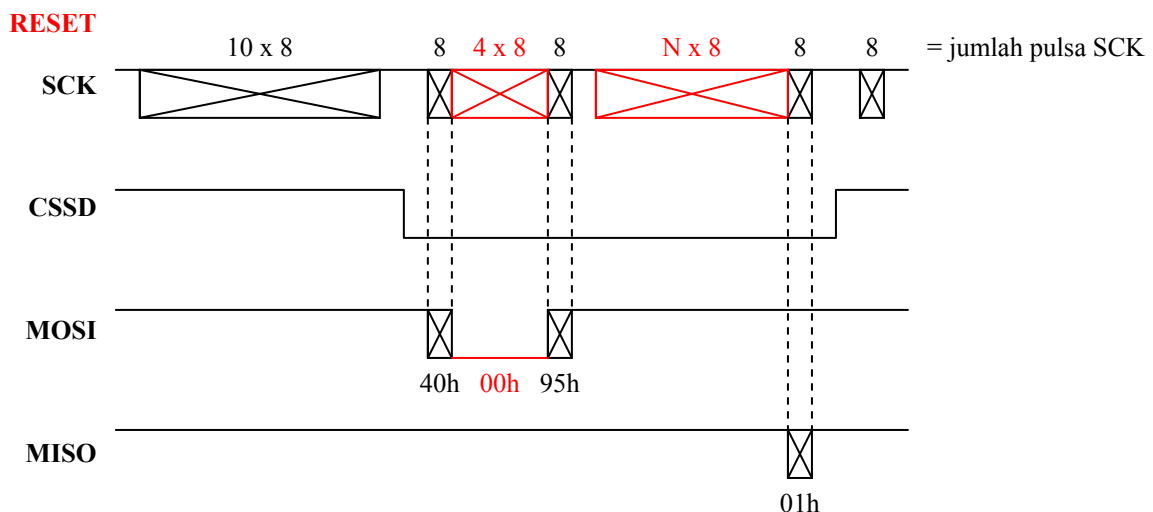
Ada 2 bagian yang akan dijelaskan secara singkat pada AN ini, yaitu:

1. Protokol komunikasi SD card secara SPI
2. Format data FAT16

Perlu diketahui bahwa penjelasan berikut ini merupakan penjelasan dasar. Penjelasan ini tidak mencakup keseluruhan protokol atau format data namun dapat digunakan sebagai bahan untuk membuat aplikasi berbasis SD Card dengan FAT16 secara sederhana.

SD card terbagi atas sektor-sektor dan tiap satu sektornya berisi 512 byte (sumber: http://document.sharpsma.com/files/LH79520_AN_multimediacar.pdf). Secara default, proses baca atau tulis selalu melibatkan satu sektor (512 byte). Secara default juga, nilai byte CRC bersifat don't care sehingga nilainya sembarang / dapat diabaikan (tidak dihitung).

Salah satu sumber mengenai protokol komunikasi SPI untuk SD/MMC adalah dari situs internet <http://www.ulrichradig.de> milik Ulrich Radig, tepatnya pada dokumen http://www.ulrichradig.de/site/arm_projekts/mmc_sd/doc/MMCSDTiming.pdf. Berdasar dari sumber ini dan juga percobaan, maka timing diagram untuk perintah RESET, INIT, READ, dan WRITE adalah sebagai berikut:

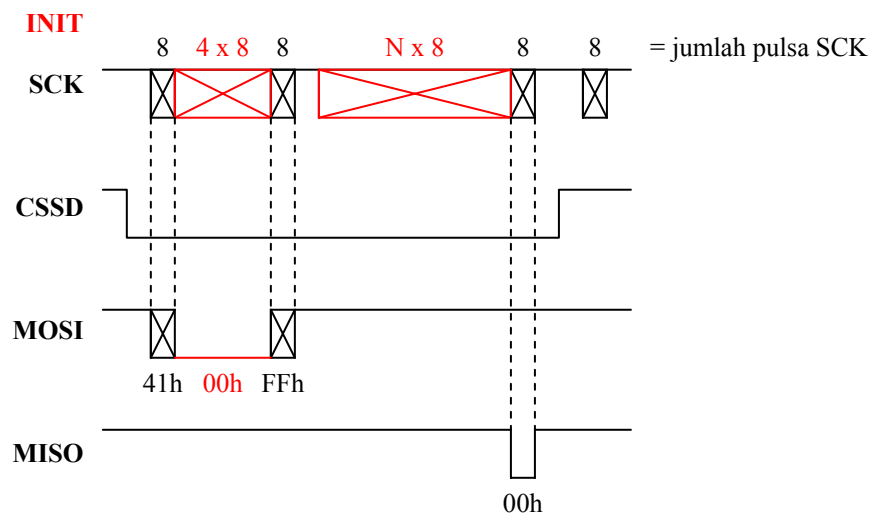


Gambar 1
Timing Diagram Perintah RESET

1. Pada dokumen asli milik Ulrich Radig, RESET diawali dengan pengiriman lebih dari 74 pulsa clock SCK. Oleh karena itu pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, pulsa yang dikirimkan adalah 10 byte data FFh atau 80 pulsa clock. Semua ini dilakukan saat kondisi CSSD, MOSI, dan MISO berlogika High.
2. Langkah berikutnya adalah mengubah logika CSSD menjadi Low untuk mulai mengakses SD card. Pada dokumen Ulrich Radig memang tidak disebutkan berapa lama jarak antara perubahan logika CSSD

dengan pengiriman clock berikutnya. Namun pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, diletakkan jeda selama 5 μ s sebelum mulai mengirimkan clock berikutnya.

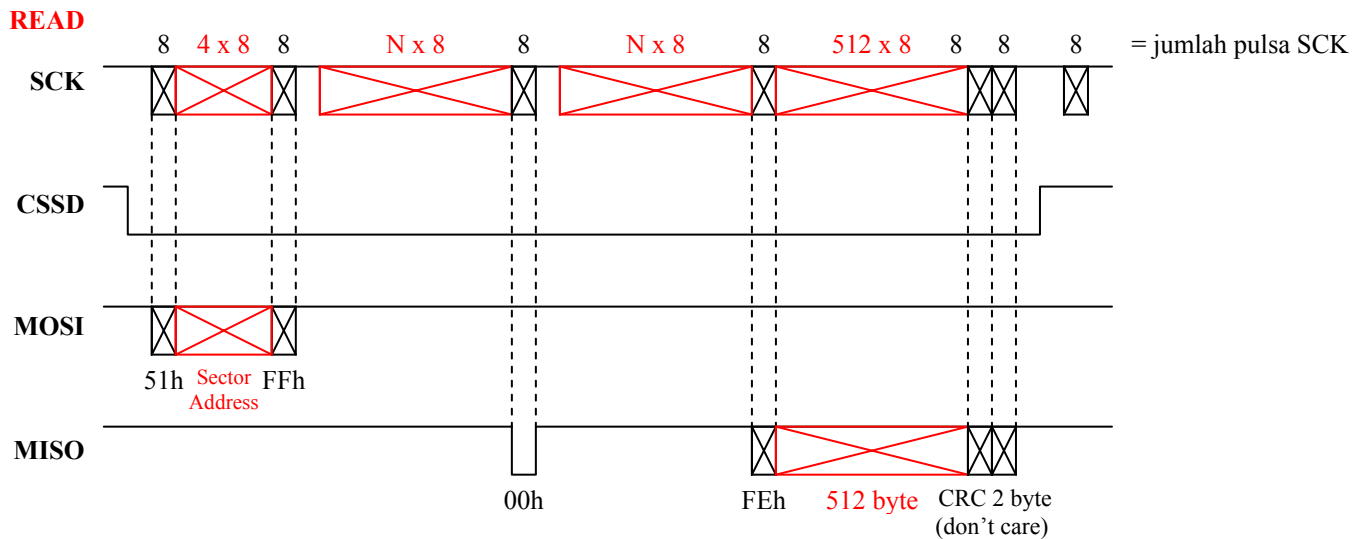
3. Data pertama yang dikirim adalah 40h (CMD0), diikuti dengan pengiriman 4 byte bernilai 00h, dan diakhiri dengan pengiriman CRC bernilai 95h.
4. Selama CSSD masih berlogika Low, dilakukan pemeriksaan terhadap MISO untuk mengetahui apakah perintah ini sudah diterima oleh SD card. Hal ini dilakukan dengan membaca data dari MISO hingga bernilai 01h. Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, pemeriksaan ini dibatasi hanya 9 kali pembacaan.
5. Setelah menerima 01h, diletakkan jeda selama 5 μ s sebelum mengembalikan CSSD ke logika High (pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51).
6. Setelah CSSD berlogika High, dilakukan pengiriman 1 byte data FFh atau 8 pulsa clock. Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, hal ini tidak dilakukan. Sebagai gantinya, sebelum perintah INIT, READ, atau WRITE, ditambahkan pengiriman 1 byte FFh atau 8 pulsa clock.



Gambar 2
Timing Diagram Perintah INIT

1. Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, ditambahkan pengiriman 1 byte FFh atau 8 pulsa clock sebelum mengirimkan perintah ini.
2. Pengiriman perintah ini diawali dengan mengubah logika CSSD menjadi Low untuk mulai mengakses SD card. Pada dokumen Ulrich Radig memang tidak disebutkan berapa lama jarak antara perubahan logika CSSD dengan pengiriman clock berikutnya. Namun pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, diletakkan jeda selama 5 μ s sebelum mulai mengirimkan clock berikutnya.
3. Data pertama yang dikirim adalah 41h (CMD1), diikuti dengan pengiriman 4 byte bernilai 00h, dan diakhiri dengan pengiriman CRC bernilai FFh.
4. Selama CSSD masih berlogika Low, dilakukan pemeriksaan terhadap MISO untuk mengetahui apakah perintah ini sudah diterima oleh SD card. Hal ini dilakukan dengan membaca data dari MISO hingga bernilai 00h. Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, pemeriksaan ini dibatasi hanya 9 kali pembacaan.
5. Setelah menerima 00h, diletakkan jeda selama 5 μ s sebelum mengembalikan CSSD ke logika High (pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51).
6. Setelah CSSD berlogika High, dilakukan pengiriman 1 byte data FFh atau 8 pulsa clock. Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, hal ini tidak dilakukan. Sebagai gantinya, sebelum perintah INIT, READ, atau WRITE, ditambahkan pengiriman 1 byte FFh atau 8 pulsa clock.

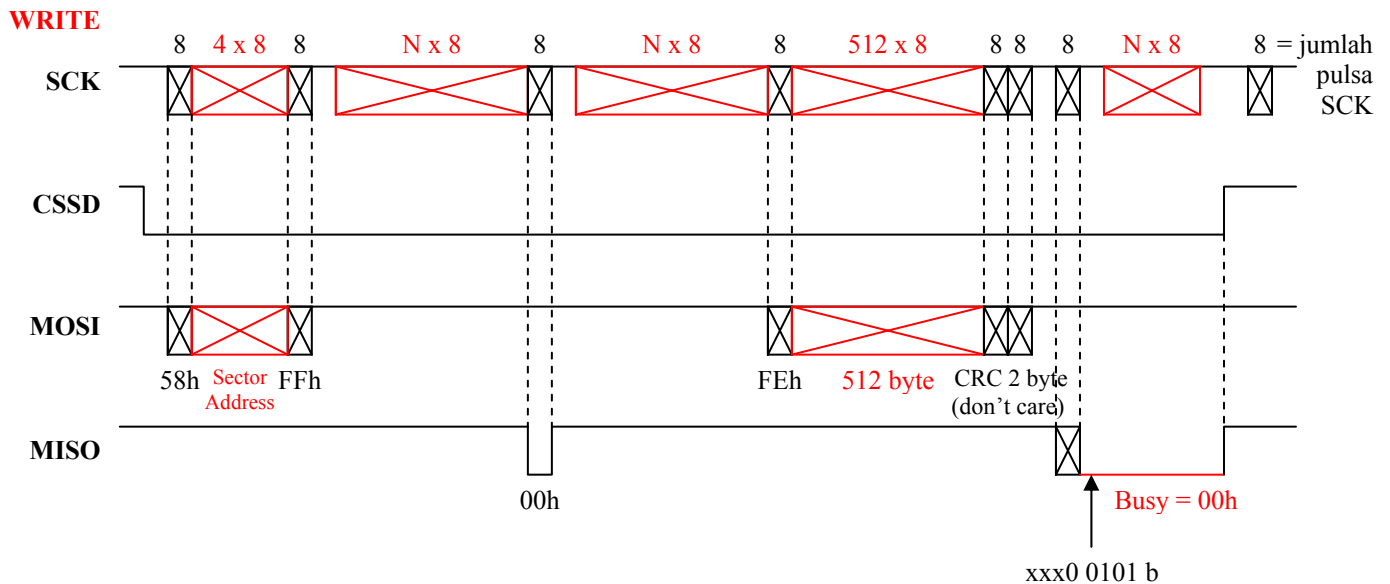
Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, jika SD card tidak mengirimkan 00h, maka proses pengiriman perintah INIT akan diulangi dari awal. Proses pengulangan ini akan dibatasi sebanyak 250 kali.



Gambar 3
Timing Diagram Perintah READ

1. Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, ditambahkan pengiriman 1 byte FFh atau 8 pulsa clock sebelum mengirimkan perintah ini.
2. Pengiriman perintah ini diawali dengan mengubah logika CSSD menjadi Low untuk mulai mengakses SD card. Pada dokumen Ulrich Radig memang tidak disebutkan berapa lama jarak antara perubahan logika CSSD dengan pengiriman clock berikutnya. Namun pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, diletakkan jeda selama 5 μ s sebelum mulai mengirimkan clock berikutnya.
3. Data pertama yang dikirim adalah 51h, diikuti dengan pengiriman 4 byte alamat sektor SD card yang dimulai dari MSB, dan diakhiri dengan pengiriman CRC bernilai FFh.
4. Selama CSSD masih berlogika Low, dilakukan pemeriksaan terhadap MISO untuk mengetahui apakah perintah ini sudah diterima oleh SD card. Hal ini dilakukan dengan membaca data dari MISO hingga bernilai 00h. Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, pemeriksaan ini dibatasi hanya 9 kali pembacaan.
5. Setelah menerima 00h, dilakukan pemeriksaan terhadap MISO lagi untuk menunggu Start Byte yang bernilai FEh. Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, pemeriksaan ini dibatasi hingga 9 x 250 kali pembacaan.
6. Penerimaan Start Byte diikuti dengan penerimaan 512 byte data dan 2 byte CRC.
7. Setelah menerima 2 byte CRC, diletakkan jeda selama 5 μ s sebelum mengembalikan CSSD ke logika High (pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51).
8. Setelah CSSD berlogika High, dilakukan pengiriman 1 byte data FFh atau 8 pulsa clock. Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, hal ini tidak dilakukan. Sebagai gantinya, sebelum perintah INIT, READ, atau WRITE, ditambahkan pengiriman 1 byte FFh atau 8 pulsa clock.

Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, jika SD card tidak mengirimkan 00h, maka proses pengiriman perintah READ bagian pertama (hingga penerimaan 00h) akan diulangi dari awal. Proses pengulangan ini akan dibatasi sebanyak 250 kali.



Gambar 4
Timing Diagram Perintah WRITE

1. Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, ditambahkan pengiriman 1 byte FFh atau 8 pulsa clock sebelum mengirimkan perintah ini.
2. Pengiriman perintah ini diawali dengan mengubah logika CSSD menjadi Low untuk mulai mengakses SD card. Pada dokumen Ulrich Radig memang tidak disebutkan berapa lama jarak antara perubahan logika CSSD dengan pengiriman clock berikutnya. Namun pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, diletakkan jeda selama 5 μ s sebelum mulai mengirimkan clock berikutnya.
3. Data pertama yang dikirim adalah 58h, diikuti dengan pengiriman 4 byte alamat sektor SD card yang dimulai dari MSB, dan diakhiri dengan pengiriman CRC bernilai FFh.
4. Selama CSSD masih berlogika Low, dilakukan pemeriksaan terhadap MISO untuk mengetahui apakah perintah ini sudah diterima oleh SD card. Hal ini dilakukan dengan membaca data dari MISO hingga bernilai 00h. Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, pemeriksaan ini dibatasi hanya 9 kali pembacaan.
5. Setelah menerima 00h, dilakukan pengiriman beberapa pulsa clock terlebih dahulu. Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51 jumlah clock yang dikirimkan adalah 72 pulsa clock atau 9 byte FFh.
6. Lalu dilanjutkan dengan pengiriman Start Byte (FEh), diikuti dengan 512 byte data yang akan dituliskan, dan diakhiri dengan pengiriman 2 byte CRC (dapat menggunakan nilai FFh semua).
7. Meski pada dokumen Ulrich Radig dinyatakan bahwa pembacaan MISO berikutnya akan menghasilkan nilai xxx00101b (x = don't care), namun pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, pemeriksaan dibatasi hingga 9 x 250 kali pembacaan.
8. Pada dokumen Ulrich Radig dinyatakan bahwa setelah pengiriman 1 byte xxx00101b, MISO akan bernilai 00h (Busy). Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, program akan memeriksa hingga MISO bernilai 00h. Pemeriksaan ini dibatasi hingga 250 kali pembacaan. Lalu program akan memeriksa hingga MISO tidak bernilai 00h lagi. Pemeriksaan ini juga dibatasi hingga 250 kali pembacaan.
9. Setelah MISO tidak bernilai 00h lagi, CSSD dikembalikan ke logika High.
10. Setelah CSSD berlogika High, dilakukan pengiriman 1 byte data FFh atau 8 pulsa clock. Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, hal ini tidak dilakukan. Sebagai gantinya, sebelum perintah INIT, READ, atau WRITE, ditambahkan pengiriman 1 byte FFh atau 8 pulsa clock.

Pada AN SD Card & FRAM – AVR dan AN SD Card & FRAM – MCS-51, jika SD card tidak mengirimkan 00h, maka proses pengiriman perintah WRITE bagian pertama (hingga penerimaan 00h) akan diulangi dari awal. Proses pengulangan ini akan dibatasi sebanyak 250 kali.

Format FAT16 menyerupai FAT12. Perbedaan yang paling jelas diantara keduanya adalah bahwa FAT16 menggunakan tabel FAT sebanyak 16 bit per alamat sedangkan FAT12 menggunakan 12 bit. Hal ini membuat FAT12 sedikit lebih sulit untuk diproses karena variabel pada bahasa pemrograman umumnya menggunakan kelipatan 8 bit. Jadi, untuk penulisan ataupun pembacaan tabel, kita dapat menggunakan variabel word atau integer sepanjang 16 bit. FAT32 jauh lebih kompleks daripada FAT12 / FAT16 sehingga juga sulit diaplikasikan.

Dari dokumen pada http://document.sharpsma.com/files/LH79520_AN_multimediacar.pdf, http://www.maverick-os.dk/FileSystemFormats/FAT16_FileSystem.html, dan dokumen Microsoft mengenai FAT32 yang didapat dari <http://www.microsoft.com/whdc/system/platform/firmware/fatgen.mspx>, diperoleh gambaran lebih jelas mengenai alokasi alamat dalam SD card dengan format FAT16.

Sektor 0 (512 byte pertama) berisi Master Boot Record (MBR) yang terdiri dari:

Alamat		Ukuran	Keterangan
Heksadesimal	Desimal		
000h	0	446 byte	Executable Code
1BEh	446	16 byte	1 st Partition Entry
1CEh	462	16 byte	2 nd Partition Entry
1DEh	478	16 byte	3 rd Partition Entry
1EEh	494	16 byte	4 th Partition Entry
1FEh	510	2 byte	Executable Marker (55h dan AAh)

Masing-masing Partition Entry terdiri dari:

Alamat		Ukuran	Keterangan
Heksadesimal	Desimal		
00h	0	1 byte	Current State of Partition (00h = inactive, 80h = active)
01h	1	1 byte	Beginning of Partition – Head
02h	2	2 byte	Beginning of Partition – Cylinder/Sector
04h	4	1 byte	Type of Partition
05h	5	1 byte	End of Partition – Head
06h	6	2 byte	End of Partition – Cylinder/Sector
08h	8	4 byte	Number of Sectors Between MBR and First Sector in Partition
0Ch	12	4 byte	Number of Sectors in Partition

Dimana nilai tipe partisi adalah salah satu dari nilai-nilai berikut:

Nilai		Keterangan
Heksadesimal	Desimal	
00h	0	Unknown or Nothing
01h	1	12 bit FAT
04h	4	16 bit FAT (Partition smaller than 32 MB)
05h	5	Extended MS-DOS Partition
06h	6	16 bit FAT (Partition larger than 32 MB)
0Bh	11	32 bit FAT (Partition up to 2048 GB)
0Ch	12	Same as 0Bh, but uses LBA1 13h Extensions
0Eh	14	Same as 06h, but uses LBA1 13h Extensions
0Fh	15	Same as 05h, but uses LBA1 13h Extensions

Pembacaan MBR yang sah dapat ditentukan dari 2 byte terakhir yaitu 55h (pada alamat 1FEh/510) dan AAh (pada alamat 1FFh/511).

Dengan asumsi bahwa sebuah SD card tidak dipartisi lagi, maka hanya akan ada 1 partisi saja (1st Partition). Tiap partisi FAT16 terdiri dari:

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

Keterangan partisi pertama berlokasi di sektor 0 (MBR) alamat 446 hingga 461. Namun sebagai dasar, cukup alamat 454 hingga 457 (1st Partition Entry alamat 8 hingga 11) saja yang akan digunakan untuk proses berikutnya. Keempat nilai ini digunakan untuk mencari Boot Sector dari partisi pertama dengan rumus:

$$\text{Alamat Boot Sector} = ([457] * 1000000h) + ([456] * 10000h) + ([455] * 100h) + [454]$$

Lalu dilakukan pembacaan lagi terhadap SD card dengan alamat sektor Boot Sector yang telah dihitung tersebut. Susunan Boot Sector adalah sebagai berikut:

Alamat		Ukuran	Keterangan
Heksadesimal	Desimal		
0000h	0	3 byte	Code to jump to the bootstrap code
0003h	3	8 byte	Oem ID – nama sistem operasi yang melakukan proses format
000Bh	11	2 byte	Bytes per Sector – biasanya ada 512 byte tiap sector
000Dh	13	1 byte	Sectors per Cluster
000Eh	14	2 byte	Reserved Sectors from the start of the volume
0010h	16	1 byte	Number of FAT copies – biasanya 2 kopi FAT digunakan untuk mencegah hilangnya data
0011h	17	2 byte	Number of possible root entries – disarankan 512 entri
0013h	19	2 byte	Small number of sectors – digunakan jika kapasitas kurang dari 32 MB
0015h	21	1 byte	Media Descriptor
0016h	22	2 byte	Sectors per FAT
0018h	24	2 byte	Sectors per Track
001Ah	26	2 byte	Number of Heads
001Ch	28	4 byte	Hidden Sectors
0020h	32	4 byte	Large number of sectors – digunakan jika kapasitas lebih besar dari 32 MB
0024h	36	1 byte	Drive Number – digunakan oleh beberapa <i>bootstrap code</i> , fx. MS-DOS
0025h	37	1 byte	Reserved – digunakan Windows NT untuk menentukan apakah integritas media penyimpanan ini harus diperiksa
0026h	38	1 byte	Extended Boot Signature – mengindikasikan bahwa 3 <i>field</i> berikutnya tersedia
0027h	39	4 byte	Volume Serial Number
002Bh	43	11 byte	Volume Label – seharusnya sama dengan yang ada di <i>root directory</i>
0036h	54	8 byte	File System Type – berisi karakter 'FAT16'
003Eh	62	448 byte	Bootstrap code
01FEh	510	2 byte	Boot sector signature - 55h dan AAh

Pembacaan Boot Sector yang sah dapat ditentukan dari 2 byte terakhir yaitu 55h (pada alamat 1FEh/510) dan AAh (pada alamat 1FFh/511).

Dari Boot Sector dan MBR, didapat beberapa parameter:

$$\text{FAT Region Start} = \text{Reserved Region} + \text{Reserved Sectors}$$

FAT Region Start adalah alamat awal dari tabel FAT. Reserved Region adalah alamat sektor pertama dalam partisi. Dalam kasus yang sederhana, Reserved Region sama dengan alamat Boot Sector.

$$\text{Root Directory Region Start} = \text{FAT Region} + (\text{Number of FAT copies} \times \text{Sectors per FAT})$$

Root Directory Region Start adalah alamat awal Root Directory.

$$\text{Data Region Start} = \text{Root Directory Region} + ((\text{Root Entries Count} \times 32) / \text{Bytes per Sector})$$

Data Region Start adalah alamat awal Data Region.

$$\text{First Sector of Cluster N} = \text{Data Region} + ((N - 2) \times \text{Sector per Cluster})$$

First Sector of Cluster N adalah alamat sektor pertama untuk file dengan cluster awal N.

Untuk mengetahui tipe FAT yang digunakan, dilakukan perhitungan sebagai berikut:

Total Sectors = nilai total Small Number of Sectors

Atau

Total Sectors = nilai total Large Number of Sectors

Clusters = Total Sectors – (Data Region – Boot Sector) / Sector per Cluster)

Jika Clusters < 4085 berarti FAT bertipe FAT12. Jika Clusters < 65525 berarti FAT bertipe FAT16. Di atas itu, FAT yang digunakan adalah FAT32.

Untuk mengetahui kapasitas total dari media penyimpanan dalam satuan Byte, dilakukan perhitungan sebagai berikut:

Capacity = (Total Sectors x Bytes per Sector)

Di dalam **FAT Region** terdapat tabel FAT dengan nilai-nilai yang mungkin adalah:

Nilai	Keterangan
0000h	Cluster bebas
0001h – 0002h	Tidak boleh digunakan
0003h – FFEFh	Nomor cluster berikutnya
FFF7h	Ada bad sector dalam cluster
FFF8h – FFFFh	Akhir file

Tabel FAT didahului dengan nilai FFxxh dimana xx adalah Media Descriptor (Boot Sector alamat 21). Nilai-nilai yang mungkin adalah:

Nilai	Kapasitas	Bentuk Fisik
F0h	2,88 MB	3,5 inci, 2 sisi, 36 sektor
F0h	1,44 MB	3,5 inci, 2 sisi, 18 sektor
F8h	?	<i>Fixed disk</i>
F9h	720 KB	3,5 inci, 2 sisi, 9 sektor
F9h	1,2 MB	5,25 inci, 2 sisi, 15 sektor
FAh	?	?
FBh	?	?
FCh	180 KB	5,25 inci, 1 sisi, 9 sektor
FDh	360 KB	5,25 inci, 2 sisi, 9 sektor
FEh	160 KB	5,25 inci, 1 sisi, 8 sektor
FFh	320 KB	5,25 inci, 2 sisi, 8 sektor

Untuk SD card dapat menggunakan nilai F8h. Tabel FAT biasanya tidak hanya satu, melainkan 2 atau sesuai dengan jumlah yang tertera pada parameter Number of FAT copies. Hal ini digunakan sebagai pengaman jika salah satu FAT rusak.

Setiap entri pada **Root Directory** terdiri dari 32 byte yang memiliki format sebagai berikut:

Alamat		Ukuran	Keterangan
Heksadesimal	Desimal		
00h	0	8 byte	File Name – Nama file
08h	8	3 byte	File Name Extension – Nama ekstensi file
0Bh	11	1 byte	Attribute Byte – Byte atribut
0Ch	12	1 byte	Reserved for Windows NT – bernilai 00h
0Dh	13	1 byte	Creation Milisecond Stamp – Waktu pembuatan file (seperseratus detik), bernilai 0 hingga 199
0Eh	14	2 byte	Creation Time – Waktu pembuatan file
10h	16	2 byte	Creation Date – Tanggal pembuatan file
12h	18	2 byte	Last Access Date – Tanggal akses terakhir
14h	20	2 byte	Reserved for FAT32 – bernilai 0000h untuk FAT16
16h	22	2 byte	Last Write Time – Waktu penulisan file terakhir
18h	24	2 byte	Last Write Date – Tanggal penulisan file terakhir
1Ah	26	2 byte	Starting Cluster – Cluster Awal
1Ch	28	4 byte	File Size – Ukuran file dalam satuan byte

Nama file sepanjang maksimal 8 byte. Untuk nama yang kurang dari 8 karakter, ruang sisanya harus diisi dengan karakter spasi (20h). Nama ekstensi file sepanjang maksimal 3 byte. Untuk nama yang kurang dari 3 karakter, ruang sisanya harus diisi dengan karakter spasi (20h). Karakter yang boleh digunakan adalah huruf alfabet kapital dan angka 0 hingga 9.

Byte pertama pada nama file memiliki aturan sebagai berikut:

1. Tidak boleh bernilai 20h.
2. Nilai 00h diartikan sebagai: “hentikan pencarian, tidak ada entri lagi dalam direktori ini”.
3. Nilai 05h harus diganti dengan karakter ASCII E5h. karakter ini digunakan di Jepang.
4. Nilai E5h diartikan sebagai: “ruang entri ini masih bebas”.

Karakter berikut ini tidak boleh digunakan untuk nama file dan nama ekstensi:

1. Semua karakter dibawah 20h kecuali 05h.
2. Semua karakter berikut ini: 22h ("), 2Ah (*), 2Bh (+), 2Ch (,), 2Eh (.), 2Fh (/), 3Ah (:), 3Bh (;), 3Ch (<), 3Dh (=), 3Eh (>), 3Fh (?), 5Bh (I), 5Ch (\), 5Dh (J), dan 7Ch (I).

Byte atribut memiliki format sebagai berikut:

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

R = Read Only; atribut ini digunakan untuk mencegah agar program tidak menghapus atau menumpangi entri ini secara otomatis.

H = Hidden; atribut ini digunakan untuk memberitahu sistem bahwa file ini harus disembunyikan saat menampilkan daftar direktori secara normal.

S = System; atribut ini digunakan untuk menandai bahwa file atau direktori ini penting untuk sistem dan tidak seharusnya diubah.

V = Volume Name; jika atribut ini bernilai 1, entri ini tidak menunjuk ke apapun juga. Jadi, Cluster Awal harus menunjuk ke cluster 0. Informasi yang dapat digunakan hanyalah nama file dan nama ekstensi yang membentuk volume label sepanjang 11 byte/karakter (tanpa ada titik yang memisahkan antara 8 byte pertama dengan 3 byte berikutnya). Hanya boleh ada 1 entri dengan V bernilai 1 dalam satu volume. Entri ini harus diletakkan di root directory dan sebaiknya di awal. Jika tidak, MS-DOS mungkin akan kesulitan menampilkan volume label yang tepat jika ada nama file yang panjang. Nama volume ini harus sama dengan volume label yang berada di boot sector.

D = Direktori; atribut ini bernilai 1 jika entri tidak menunjuk ke file namun ke direktori lain, sebuah sub-direktori. Sub-direktori diletakkan di cluster yang ditunjuk oleh nilai Cluster Awal. Format tabel sub-direktori sama dengan tabel root directory.

A = Archive; atribut ini digunakan oleh utilitas backup. Atribut ini diberi nilai 1 saat file dibuat atau diganti. Utilitas backup dapat menggunakan atribut ini untuk menentukan file mana yang berubah sejak backup terakhir.

Format waktu adalah sebagai berikut:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Jam (0 – 23)					Menit (0 – 59)					Detik (0 – 29)					

Nilai Detik dihitung dalam satuan 2 detik, sehingga nilai 29 berarti 58 detik.

Format tanggal adalah sebagai berikut:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Tahun dimulai dari 1980 (0 – 127 → 1980 – 2107)							Bulan (1 – 12)				Tanggal (1 – 31)				

Nilai Cluster Awal menunjuk ke nomor cluster awal dari sebuah data. Jika entri berupa direktori, maka Cluster Awal menunjuk ke cluster yang mengandung awal sub-direktori. Jika entri berupa file, maka Cluster Awal menunjuk ke cluster yang mengandung awal file.

Ukuran file adalah ukuran total file dalam satuan byte. Ukurannya tidak boleh melebihi 4 GB untuk satu file. Untuk entri selain berupa file, nilainya harus 0.

Isi dari file dituliskan pada alamat **First Sector of Cluster N** dengan Cluster N adalah nilai Cluster Awal. Jika ukuran file melebihi 1 cluster, maka isi selanjutnya dapat dituliskan pada cluster lain. Tabel FAT pun harus disesuaikan dengan perubahan ini agar sistem dapat mengetahui isi file secara berurutan.

Jika ingin melakukan penambahan isi file, maka parameter yang harus dicari/dihitung tidak hanya alamat Cluster Awal dan First Sector namun juga Ukuran File, Bytes per Sector, dan Sector per Cluster. Dari ukuran file dan Bytes per Sector didapat jumlah sektor yang ditempati file. Dari nilai tersebut dan Sector per Cluster, didapat jumlah cluster yang ditempati file tersebut.

Daftar sumber yang dapat digunakan sebagai referensi lebih lanjut:

- http://www.ulrichradig.de/site/arm_projekts/mmc_sd/doc/MMCSDTimming.pdf
- <http://www.it.lth.se/ssoa/Project1/FAT12Description.pdf>
- <http://downloads.amilda.org/MODs/SDCard/ProdManualSDCardv1.9.pdf>
- http://document.sharpsma.com/files/LH79520_AN_multimediacar.pdf
- http://www.sdcard.org/sd_memorycard/Simplified%20Physical%20Layer%20Specification.PDF
- <http://www.sandisk.com/Assets/File/OEM/Manuals/ProdManRS-MMCv1.3.pdf>
- http://www.sandisk.com/Assets/File/OEM/Manuals/SD_Physical_specs_v101.pdf
- <http://www.microsoft.com/whdc/system/platform/firmware/fatgen.mspx>

Selamat berinovasi!