
# Multimedia card (MMC)

**هدایت اله روستا**

**دانشگاه آزاد واحد جهرم**

Hedy_telecom@yahoo.com

## چکیده:

امروزه اکثر دستگاه های دیجیتال که به بازار ارائه می شوند معمولا شامل mmc هستند و دلایل استفاده از mmc این است که آنها از طریق سریال داده ها را انتقال می دهند و سرعت بالایی در حدود 20Mbit/s یا به عبارت دیگر چیزی حدود 2.5Mbyte/s دارند. همچنین یکی دیگر از دلایل استفاده از mmc این است که به راحتی می توان اطلاعات را بر روی آنها ذخیره و بازیابی کرد.

در این مقاله هدف، بیشتر شناسایی ساختار درونی و رجیسترهای mmc و چگونگی انتقال داده با استفاده از دستورات 48 بیتی بین mmc و Host (کنترلر) می باشد. که هرکدام از دستورات 48 بیتی کار خاصی را انجام می دهند.

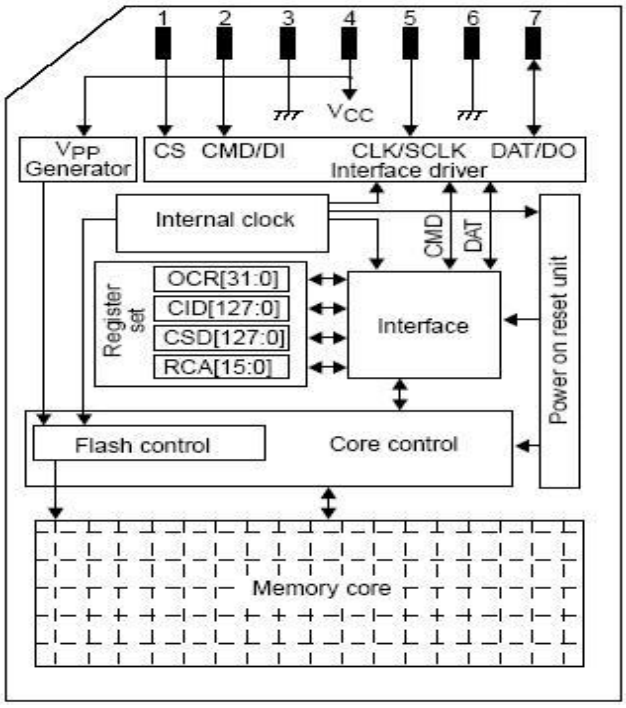
## کلمات کلیدی:

mmc (کارت حافظه چند رسانه ای)، Host (سخت افزاری که mmc را کنترل می کند) ، CMD (دستور 48 بیتی یا Command ) ، DAT (data یا داده) ، CLK (کلاک یا Clock ) ، CS (Chip Select یا انتخاب تراشه) ، Interface (واسط سخت افزاری) ، Protocol (پروتکل یا قانون) ، Pin (پین یا خط داده) ، رجیستر(ثبات) ، فیلد (قسمت، زمینه Field )

## 1- مقدمه:

شکل زیر ساختار درونی mmc را نشان می دهد که از بلوک های (قسمت های) جداگانه ای تشکیل شده است. همه این بلوک ها به وسیله یک اسیلاتور داخلی کلاک می خورند. چون mmc دارای اسیلاتور داخلی با فرکانس مشخص میباشد بنابراین برای انتقال داده توسط خطوط CMD (دستور) و DAT (داده) باید فرکانس اسیلاتور داخلی با فرکانس خط CLK (کلاک) که توسط Host (کنترلر) بوجود می آید سنکرون (همزمان) شود. این کار توسط واحد Interface driver انجام می شود.

Mmc ها به وسیله سه پین(خط داده) CMD ، DAT و CLK کنترل می شوند که هریک از این خطوط داده کار خاصی را در طول انتقال داده انجام می دهند.




همچنین mmc ها دارای چهار رجیستر داخلی هستند که از این رجیستر ها برای شناسایی و یک سری از خصوصیات mmc استفاده می شود. این رجیستر ها عبارتند از :

- **OCR** (Operation Condition Register) این رجیستر رنج ولتاژهایی را که توسط کارت پشتیبانی می شود نشان می دهد
- **CID** (Card Identification) رجیستر شناسایی کارت
- **CSD** (Card Specific Data) رجیستر تعیین داده (دانستن محتویات این رجیستر مهم است)
- **RCA** (Relative Card Address) رجیستر تخصیص آدرس برای انتقال داده بین Host و mmc دانستن محتویات این رجیستر ها حائز اهمیت است.

Mmc برای انتقال داده از دو پروتکل (Protocol) پشتیبانی می کنند یا به عبارت دیگر به دو روش(mode) می توان داده را بین mmc و Host تبادل کرد که عبارتند از:

- **Multimedia Mode یا (mmc mode)**
- **SPI mode (Serial Peripheral Interface mode)**

در هر یک از دو پرتکل بالا طریقه اتصال بین mmc و Host تفاوت دارد (به شکل زیر توجه کنید)

خوب، ممکن است بعضی از جملات برای شما تازگی داشته باشد و مفهوم آن را متوجه نشوید.

در این مقاله هریک از آنها راجداگانه بررسی کنیم تا با مفهوم آنها بیشتر آشنا شویم.

### 1-1- Multimedia mode (mmc mode) :

در این روش برای انتقال داده از کمترین خط داده استفاده شده است که عبارتند از :

**1-1-1 - CLK :** با هر دوره تناوب (سیکل) از سیگنال ورودی به این پین یک بیت داده بر روی خط DAT یا CMD جابجا می شود(منتقل می شود). فرکانس این خط می تواند بین صفر و ماکزیمم فرکانس ممکنه باشد(20Mbit/S ) و این فرکانس می تواند در حین انتقال داده تغییر کند(این یک مزیت است).

**1-1-2 - CMD :** این خط یک خط داده دوطرفه است(ورودی و خروجی) که برای فعال کردن mmc و دستور انتقال داده و پاسخ مورد استفاده قرار می گیرد. این خط داده به دو روش عمل میکند 1-Open drain mode برای فعال سازی mmc 2- Push pull modeبرای انتقال سریع داده(Command )

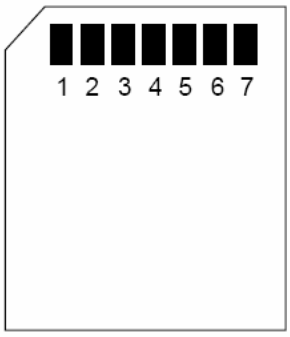
دستور همیشه توسط Host اما پاسخ (Response ) توسط mmc فرستاده می شود.

**1-1-3 - DAT :** این خط یک خط داده دو طرفه است و برای انتقال داده ای که باید بر روی حافظه ذخیره یا از حافظه خوانده شود مورد استفاده قرار می گیرد.

**1-1-4 - RSV :** این خط در این روش بدون استفاده است اما در SPI mode به عنوان خط انتخاب تراشه به کار می رود.

(A) MMC Mode

| Pin No. | Name | Type | Pin Function |
|---|---|---|---|
| 1 | RSV | NC | Reserved pin |
| 2 | CMD | I/O, PP, OD | Command/response |
| 3 | $V_{SS1}$ | S | GND |
| 4 | $V_{DD1}$ | S | $V_{CC}$ |
| 5 | CLK | I | Clock |
| 6 | $V_{SS2}$ | S | GND |
| 7 | DAT | I/O, PP | Data input/output |




(B) SPI Mode

| Pin No. | Name | Type | Pin Function |
|---|---|---|---|
| 1 | CS | I | Chip select |
| 2 | DI | I | Data input |
| 3 | $V_{SS}$ | S | GND |
| 4 | $V_{DD}$ | S | $V_{CC}$ |
| 5 | SCLK | I | Clock |
| 6 | $V_{SS2}$ | S | GND |
| 7 | DO | O, PP | Data output |

Note: S = Power supply
I = Input
O = Output
PP = Push pull
OD = Open drain
NC = Not connected

**Figure 2.2 MultiMediaCard™ Pin Assignments**

### 2-1- SPI mode :

پروتکل SPI توسط شرکت Motorola ارائـه شـد و توسـط اکثر میکروکنترلر های PIC ، 8051 و AVR و... پشتیبانی می شود.این مزیت باعث شده که بیشتر پروژه های امـروزی از این کارت حافظه (mmc ) استفاده کنند.

خطوط داده در این پروتکل عبارتند از :

**1-2-1 - CLK :** سیگنال کلاک برای سنکرون سازی انتقال داده است.

**1-2-2 - Data In :** این خـط داده ورودی mmc اسـت. بـر روی میکـرو کنترلـر خـط داده(پـین) MOSI ( Master Output Slave Input ) باید به این پین وصل شـود. زیـرا Host به عنوان Master و mmc بـه عنـوان Slave عمـل می کند.

**1-2-3 - Data Out :** این خط داده ورودی mmc است. . بر روی میکرو کنترلر خط داده(پین) MISO ( Master Input Slave Output ) باید به این پین وصل شود.

**1-2-4 - CS :** ( Chip Select) از این خط داده (پین) برای انتخاب mmc و فعال سازی SPI mode استفاده می شود.

در پروتکل mmc ، mmc به وسیله آدرس مخصوصـی کـه توسط Host(کنترلر) به آن اختصاص داده، شناسایی می شود اما در پروتکل SPI شناسایی کارت توسط خـط(پـین) CS انجام می گیرد که برای انتخاب هـر کـارت مـی تـوانیم بـه وسیله خط CS (وقتی که 0 می شود کارت فعال مـی شـود) mmc را انتخاب کنیم و باید توجه کنیم که در طـول انتقـال داده این خط باید فعال باشـد. همچنـین در پرتکـل SPI بـر خلاف پرتکل mmc ، که خطوط دو طرفه بود، خطوط یـک طرفه است یعنی یا ورودی اسـت یـا خروجـی و ایـن باعـث کاهش سرعت و کـاهش دسـتورات انتقـال داده در پرتکـل SPI است.

## 2-رجیسترهای mmc :

mmc ها دارای چهار رجیستر هستند که در جدول زیر مشاهده می کنید

چون این رجیسترها، رجیسترهایی هستند که اطلاعات کاملی در مورد mmc و چگونگی انتقال داده بین Host و mmc را به ما میدهند بنابراین دانستن اطلاعات داخلی این رجیسترها و برای انتقال داده مهم می باشند.

در جدول بالا رجیستر OCR برای مشخص کردن رنج ولتاژ هایی است که توسط این mmc پشتیبانی می شود. رجیسترهای CID و RCA برای شناسایی و آدرس دهی به mmc استفاده می شود ((از این دو رجیستر در مواقعی استفاده می شود که چند mmc را به خطوط داده (CLK ، CMD و DAT) وصل کنیم)).

از همه این رجیسترها مهمتر رجیستر CSD است که شامل اطلاعاتی راجع به خود mmc ، چگونگی انتقال و فرمت داده ، اندازه بلوک(سکتور) داده و ... می باشد.

### 2-1- OCR (Operation Condition Register):

همان طور که در جدول بالا مشاهده می کنید این رجیستر 32 بیتی است و رنج ولتاژ هایی که توسط mmc پشتیبانی می شود را نشان می دهد.

بیت 31 این رجیستر در هنگام فعال کردن mmc به کار می رود. وقتی Host دستور فعال شدن را برایmmc می فرستد منتظر پاسخی(Response ) از طرف mmc می ماند. Mmc نیز در پاسخ 32 بیت این رجیستر را برای Host می فرستد. Host با چک کردن آخرین بیت ارسالی (اگر 1 باشد نشان میدهد کارت فعال شده) متوجه فعال شدن یا مشغول بودن mmc می شود. اگر mmc مشغول باشد Host دوباره این روند را تکرار می کند.

### 2-2- CID (Card Identification) : این رجیستر

دارای طول داده 128 بیت (16 بایت)است که توسط کارخانه سازنده برنامه ریزی و برای شناسایی کارت استفاده می شود.

از این رجیستر بیشتر برای سرشماری کردن زمانی که چندین mmc به طور همزمان به Host متصل می شود.

### 2-3- RCA (Relative Card Address) : این

یک رجیستر 16 بیتی (2 بایتی) است که این رجیستر در هنگام سرشماری (فعال کردن) mmc توسط Host مقدار دهی می شود و مقدار این رجیستر. آدرس mmc را در هنگام تبادل اطلاعات با Host مشخص می کند. در حالت معمولی RCA مقدار 0x0001 را دارد.از مقدار 0x0000 برای تنظیم همه کارت ها در حالت Standby استفاده می شود.

| Name | Width | Type | Description |
|---|---|---|---|
| OCR | 32 | Programmed by the manufacturer. Read only for user | Supported voltage range, card power up status bit |
| CID | 128 | Programmed by the manufacturer. Read only for user | Card identification number, card individual number for identification. |
| RCA | 16 | Programmed during initialization, not readable | Relative card address, local system address of a card, dynamically assigned by the host during initialization. |
| CSD | 128 | Programmed by the manufacturer. Partially programmable by the user. | Card specific data, information about the card operation conditions. |

### 4-2- CSD (Card Specific Data) : این رجیستر یک رجیستر 128 بیتی است که چگونگی دسترسی به محتویات کارت را توصیف می کند. محتویات این رجیستر در جدول زیر نشان داده شده است.

| Name | Field | Width | CSD-slice | Value | Type |
|---|---|---|---|---|---|
| Write protect group size | WP_GRP_SIZE | 5 | [36:32] | 0x01 (16 kByte) | read only |
| Write protect group enable | WP_GRP_ENABLE | 1 | [31:31] | '1' | read only |
| Manufacturer default ECC | DEFAULT_ECC | 2 | [30:29] | 0 | read only |
| Write speed factor | R2W_FACTOR | 3 | [28:26] | 2 (4) | read only |
| Max. write data block length | WRITE_BLK_LEN | 4 | [25:22] | 9 (512 Bytes) | read only |
| Partial blocks for write allowed | WRITE_BLK_PARTIAL | 1 | [21:21] | '0' | read only |
| Reserved | — | 5 | [20:16] | 0 | read only |
| File format group | FILE_FORMAT_GRP | 1 | [15:15] | ×*4 | read/write |
| Copy flag (OTP) | COPY | 1 | [14:14] | × | read/write |
| Permanent write protection | PERM_WRITE_PROTECT | 1 | [13:13] | × | read/write |
| Temporary write protection | TMP_WRITE_PROTECT | 1 | [12:12] | × | read/write/erase |
| File format | FILE_FORMAT | 2 | [11:10] | × | read/write |
| ECC code | ECC | 2 | [9:8] | × | read/write/erase |
| 7-bit CRC | CRC7 | 7 | [7:1] | × | read/write/erase |
| Not used, always 1 | — | 1 | [0:0] | 1 | read only |

Notes: 1. This field is depended on the model. Refer to also C_SIZE
2. This field is depended on the model.
3. This field is depended on the model. Refer to also C_SIZE_MULT
4. × means user programmable

بعضی از بیت های رجیستر CSD قابلیت یک یا چند بار برنامه ریزی توسط کاربر را دارا می باشند و بیت های دیگر ثابت و فقط قابل خواندن هستند.
چون محتویات این رجیستر دارای اهمیت زیادی است بنابراین هر یک از فیلد آن را به طور جداگانه ای بررسی می کنیم.

- **NSAC :** این فیلد شامل 8 بیت است و تعداد کلاک های اولیه که باید توسط Host به mmc داده شود تا هر دو سنکرون شوند را مشخص می کند در جدول این مقدار برابر 0x01 است.

یعنی برای سنکرون شدن فرکانس اسیلاتور داخلی mmc و فرکانس کلاک Host ، باید 100 کلاک توسط Host زده شود تا mmc با Host سنکرون شود.

- **READ_BLK_LEN :** این فیلد یک فیلد 16 بیتی بسیار مهم است. در اینجا لازم است با دو مفهوم Block (بلوک ) و Group که در این مقاله با آن بسیار سروکار داریم آشنا شویم.

**Block(Sector) :** هر Block از چند بایت تشکیل شده است که می تواند بین 1 تا 2048 بایت متغییر باشد. به هر Block یک سکتور (Sector ) نیز گفته می شود.

**Group (Cluster) :** هر Group از چند بلوک (Block ) یا سکتور تشکیل شده است که این هم مانند بلوک می تواند متغییر باشد. به یک Group یک کلاستر (Cluster ) نیز گفته می شود.

همان طور که در جدول زیر مشاهده میکنید این فیلد شامل 16 بیت است که از 12 بیت آن برای مشخص کردن اندازه بلوک استفاده می شود. در حالت معمولی این فیلد مقدار 9 را دارد یعنی اندازه بلوک داده برابر 512 بایت است.

( علاوه بر اینکه ما اندازه یک بلوک را تعیین میکنیم حافظه نیز از بلوک های فیزیکی تشکیل شده است که بستگی به نوع FAT و اندازه حافظه دارد )

| READ_BLK_LEN | Block length | Remark |
|---|---|---|
| 0 | $2^0$ = 1 byte | |
| 1 | $2^1$ = 2 bytes | |
| ..... | ..... | |
| 11 | $2^{11}$ = 2048 bytes | |
| 12–15 | reserved | |

- **READ_BLK_PARTIAL :** این فیلد شامل یک بیت است . اگر READ_BLK_PARTIAL = 0 باشد یعنی اینکه فقط اندازه بلوکی که در

$$BLOCKNR = (C\_SIZE+1)*MULT$$

$$MULT = 2^{C\_SIZE\_MULT+2} \ (C\_SIZE\_MULT < 8)$$

$$BLOCK\_LEN = 2^{READ\_BLK\_LEN}, \ (READ\_BLK\_LEN < 12)$$

READ_BLK_LEN تعریف شده میتواند برای انتقال یک بلوک داده مورد استفاده قرار گیرد. حال اگر READ_BLK_PARTIAL = 1 باشد یعنی اینکه بلوک هایی با اندازه کوچکتر می تواند مورد استفاده قرار گیرد که کمترین اندازه بلوک می تواند یک بایت باشد.

- **READ_BLK_MISALIGN :** این فیلد مشخص می کند که وقتی یک بلوک داده به وسیله یک دستور از حافظه خوانده می شود، می تواند؟ بیشتر از یک بلوک فیزیکی از حافظه را بخواند.

- **DSR_IMP :** این فیلد مشخص می کند که آیا mmc قابل ترکیب بندی مجدد است یا خیر؟ مقدار این فیلد ثابت وبرابر صفر است یعنی قابل ترکیب بندی مجدد نیست.
- **C_SIZE :** از این فیلد برای محاسبه ظرفیت حافظه mmc استفاده می شود

بنابراین برای محاشبه ظرفیت mmc می توان از فرمول زیر استفاده کرد.

- **C_SIZE_MULT :** از این فیلد نیز برای محاسبه ظرفیت حافظه استفاده می شود و به عنوان یک فیلد کمکی و ضریبی برای فرمول فیلد C_SIZE مورد استفاده قرار می گیرد

| C_SIZE_MULT | MULT | Remark |
|---|---|---|
| 0 | $2^2$ = 4 | |
| 1 | $2^3$ = 8 | |
| 2 | $2^4$ = 16 | |
| 3 | $2^5$ = 32 | |
| 4 | $2^6$ = 64 | |
| 5 | $2^7$ = 128 | |
| 6 | $2^8$ = 256 | |
| 7 | $2^9$ = 512 | |

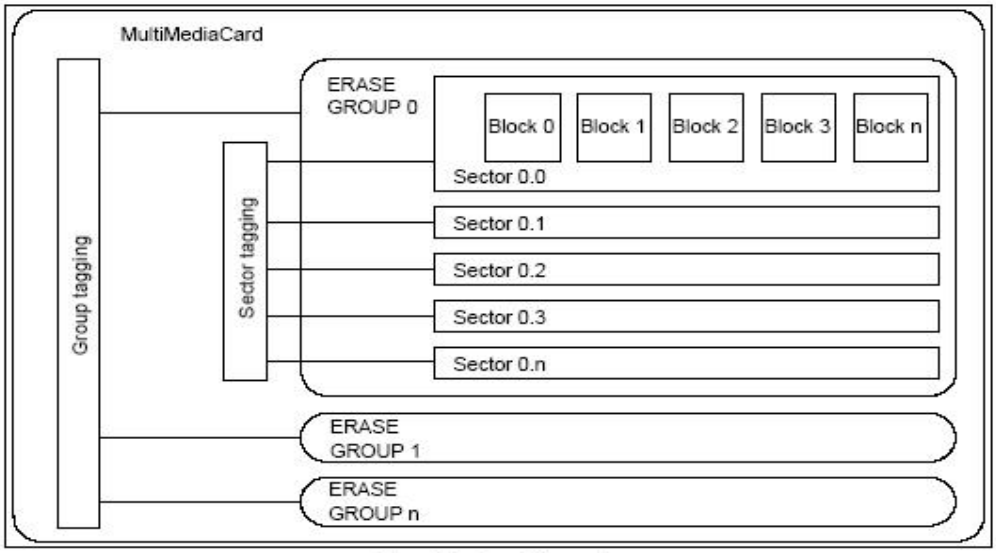
$2^{C\_SIZE\_MULT+2}$

| Model | C_SIZE | C_SIZE_MULT | READ_BLK_LEN | Card Capacity |
|---|---|---|---|---|
| HB28E016MM2 | 0x7A7 | 2 | 9 | 16 MByte |
| HB28D032MM2 | 0x7A7 | 3 | 9 | 32 MByte |
| HB28D064MM2 | 0x7A7 | 4 | 9 | 64 MByte |
| HB28B128MM2 | 0x7A7 | 5 | 9 | 128 MByte |

- **EERASE_GRP_SIZE :** محتویات این فیلد یک مقدار 5 بیتی است و اندازه یک Group یا کلاستر قابل پاک شدن را نشان می دهد.

مقدار این فیلد همیشه 0 است بنابراین با توجه به مقدار ERASE_GRP_MULT تعداد بایتی که به ازای

- **WP_GRP_ENABLE :** این مقدار همیشه یک است یعنی Write Protect Group فعال است
- **R2W_FACTOR :** این فیلد مدت زمان نوشتن




**Erase Tagging Hierarchy**

فرستادن هر دستور Erase ( CMD38 و CMD35 ) پاک می شود برابر 8kbyte است.

- **ERASE_GRP_MULT :** یک فیلد 5 بیتی کمکی است که برای محاسبه مقدار بایتی که به ازای فرستادن هر دستور Erase از حافظه پاک می شود مورد استفاده قرار می گیرد.

توجه: پاک شدن اطلاعات در mmc به صورت Sector یا Group است بدین صورت که Sector ها و Group ها دارای یک بیت هایی هستند که به آنها تگ(Tag) گفته می شود واز این تگ ها برای انتخاب Sector یا Group که در هنگام ارسال دستور Erase باید پاک شود استفاده می شود.

ما برای پاک کردن یک یا چند Sector ابتدا باید به وسیله دستورهای CMD33 , CMD32 این تگ ها را فعال کنیم سپس به وسیله دستور CMD38 این Sector ها را پاک کنیم.

- **WP_GRP_SIZE :** این فیلد اندازه یک Group را برای محافظت از نوشتن و پاک شدن (Write Protect ) رانشان می دهد مثلا اگر این مقدار برابر 0x01 باشد یعنی اندازه آن برابر16Kbyte است.

(Write) را به صورت ضریبی از خواندن (Read ) از حافظه نشان می دهد که در این فیلد مقدار حالت معمولی 2 است یعنی سرعت نوشتن در حافظه نصف سرعت خواندن از حافظه است.

- **FILE_FORMAT_GRP :** این فیلد فرمت Group که انتخاب شده را مشخص میکند و در حالت معمولی این مقدار برابر صفر است. از این فیلد در فیلد FILE_FORMAT استفاده می شود.
- **PERM_WRITE_PROTECT :** این فیلد تعیین می کند که آیا کارت به طور دائم Write Protect است یا نه؟ اگر مقدار این فیلد برابر یک با شدیعنی این کارت به هیچ کدام از دستورات Erase و Write پاسخ نخواهد داد. اما در حالت معمولی این مقدار همیشه صفر است.
- **TEMP_WRITE_PROTECT :** این فیلد برخلاف فیلد PERM_WRITE_PROTECT قابل برنامه ریزی است یعنی به وسیله این فیلد می توانیم کارت حافظه را به طور موقت Write Protect کنیم.
- **FILE_FORMAT :** این فیلد فرمت ذخیره شدن اطلاعات را بر روی حافظه نشان می دهد که

به طور معمولی ایـن فیلـد مقـدار صـفر دارد یعنـی فرمت ذخیره شدن اطلاعـات ماننـد فایـل سیسـتم است که بر روی هارد دیسک ها وجود دارد.

| FILE_FORMAT_GRP | FILE_FORMAT | Type |
|---|---|---|
| 0 | 0 | Hard disk-like file system with partition table |
| 0 | 1 | DOS FAT (floppy-like) with boot sector only (no partition table) |
| 0 | 2 | Universal File Format |
| 0 | 3 | Others/Unknown |
| 1 | 0, 1, 2, 3 | Reserved |

## 3- ساختار دستور ها (Commands ):

دستور همیشه توسط Host فرستاده می شود و هر دستور از طـرف Host بـا یـک پاسـخ (Response ) از طـرف mmc همراه است یعنی اینکه Host دستور یا درخواستی را به mmc می فرستد و منتظر دریافت پاسخ مـی شـود mmc نیز باید به محض دریافت دستور پاسخی متناسـب با دستور بـه Host بفرسـتد. Host بـا دریافـت پاسـخ و پردازش آن از وضعیت کارت مطلع می شود .

انتقال دستور همیشه با بیت پرارزش (MSB ) شروع می شود ودر آخر با 7 بیت CRC و یک بیت پایان تمام مـی شود. دستور ها در mmc همیشه دارای انـدازه ثابـت 48 بیت (6 بایت ) است. بیت شروع همیشه 0 و بیـت پایـان همیشه 1 است. بیت بعد از بیت شـروع مشـخص کننـده فرستنده است اگر این بیت 1 باشد نشـان مـی دهـد کـه Host فرستنده این دستور(Command ) است حال اگر این بیت0 بود آنگاه فرستنده آن mmc و این یک پاسـخ (Response ) اسـت فیلـد Command دارای 6 بیـت است و شامل کد دستور است فیلـد Argument بسـتگی به نوع دستور دارد و فیلد CRC بـرای مشـخص کـردن وجود یا عدم وجود خطا در ارسال دستور است.

جدول دستورات مربوط به هـر کـلاس را در شـکل زیـر مشاهده می کنید

باید توجه داشت که منظور از CMDx همان کد دسـتور 6 بیتی است که در فیلد Command باید قـرار دهـیم و هر کد دستور کار خاصی را انجام می دهد.

کد دستور CMD0 برابر 0 و کد دسـتور CMD1 برابـر 1 است وبه همین ترتیـب کـد دسـتور CMDx برابـر x است.

| 0 | 1 | bit5...bit0 | bit31...bit0 | bit6...bit0 | 1 |
|---|---|---|---|---|---|
| start bit | host | command | argument | CRC*1 | end bit |

Note: 1. (Cyclic Redundancy Check)

**Basic Commands** (class 0) **and Read Stream Command** (class 1)

| CMD index | Type | Argument | Resp | Abbreviation | Command description |
|---|---|---|---|---|---|
| CMD0 | bc | [31:0] stuff bits | — | GO_IDLE_STATE | resets all cards to Idle State |
| CMD1 | bcr | [31:0] OCR without busy | R3 | SEND_OP_COND | checks for cards not supporting the full range of 2.0 V to 3.6 V. After receiving CMD1 the card sends an R3 response (refer to Chapter "Responses"). |
| CMD2 | bcr | [31:0] stuff bits | R2 | ALL_SEND_CID | asks all cards in ready state to send their CID*1 numbers on CMD-line |
| CMD3 | ac | [31:16] RCA [15:0] stuff bits | R1 | SET_RELATIVE_ADDR | assigns relative address to the card in identification state. |
| CMD4 | bc | [31:16] DSR [15:0] stuff bits | — | SET_DSR | programs the DSR of all cards in stand-by state. |
| CMD7 | ac | [31:16] RCA [15:0] stuff bits | R1b (only the selected card) | SELECT/ DESELECT_CARD | command toggles a card between the standby and transfer states or between the programming and disconnect state. In both cases the card is selected by its own relative address while deselecting the prior selected card. Address 0 deselects all. |
| CMD9 | ac | [31:16] RCA [15:0] stuff bits | R2 | SEND_CSD | asks the addressed card to send its card-specific data (CSD)*2 on CMD-line. |
| CMD10 | ac | [31:16] RCA [15:0] stuff bits | R2 | SEND_CID | asks the addressed card to send its card identification (CID) on CMD- line. |
| CMD11 | adtc | [31:0] data address | R1 | READ_DAT_UNTIL_STOP | reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows. |
| CMD12 | ac | [31:0] stuff bits | R1b*3 | STOP_TRANSMISSION | forces the card to stop transmission |
| CMD13 | ac | [31:16] RCA [15:0] stuff bits | R1 | SEND_STATUS | Asks the addressed card to send its status register. |
| CMD15 | ac | [31:16] RCA [15:0] stuff bits | — | GO_INACTIVE_STATE | Sets the card to inactive state in order to protect the card stack against communications breakdowns. |

**Block-Oriented Read/Write Command** (class 2/4)

| CMD index | Type | Argument | Resp | Abbreviation | Command description |
|---|---|---|---|---|---|
| CMD23 | ac | [31:16] set to 0 [15:0] number of blocks | R1 | SET_BLOCK_COUNT | Defines the number of blocks which are going to be transferred in the immediately succeeding multiple block read or write command. |

**Block-Oriented Read Commands** (class 2)

| CMD index | Type | Argument | Resp | Abbreviation | Command description |
|---|---|---|---|---|---|
| CMD16 | ac | [31:0] block length | R1 | SET_BLOCKLEN | Selects a block length (in bytes) for all following block-oriented read commands and lock card command.*1 |
| CMD17 | adtc | [31:0] data address | R1 | READ_SINGLE_BLOCK | Reads a block of the size selected by the SET_BLOCKLEN command.*2 |
| CMD18 | adtc | [31:0] data address | R1 | READ_MULTIPLE_BLOCK | Continuously send blocks of data until interrupted by a stop. |

**Block-Oriented Write Commands** (class 4)

| CMD index | Type | Argument | Resp | Abbreviation | Command description |
|---|---|---|---|---|---|
| CMD24 | adtc | [31:0] data address | R1 | WRITE_BLOCK | Write a block. The write-block-length of the Hitachi MultiMediaCard is permanently assigned to the value 512 bytes.*1 |
| CMD25 | adtc | [31:0] data address | R1 | WRITE_MULTIPLE_BLOCK | Continuously writes blocks of data until a STOP_TRANSMISSION follows. |
| CMD26 | adtc | [31:0] stuff bits | R1 | PROGRAM_CID | Programming of the card identification register. This command is only done once per MultiMediaCard card. The card has some hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer. The CID register is programmed at the manufacture. |
| CMD27 | adtc | [31:0] stuff bits | R1 | PROGRAM_CSD | Programming of the programmable bits of the CSD. |

## 4- مد انتقال داده :

**4-1- Stream Read (خواندن به صورت پشت سرهم):** در این روش با استفاده از دستور READ_DAT_UNTIL_STOP ( CMD11) داده ها را از آدرسی که در Argument دستور قرار داده ایم شروع به خواندن می کنیم.

زمانی که این دستور را به mmc ارسال می کنیم بعد از دریافت پاسخ ، mmc شروع به ارسال داده می کند و این ارسال داده ادامه دارد تا اینکه دستور (CMD20 ) STOP_TRANSMISSTION را ارسال کنیم و با این دستور ارسال داده متوقف می شود. نکته ای که در این روش ارسال داده باید به آن توجه کرد این است که در پایان ارسال داده فیلد CRC وجود ندارد.

**4-2- Block Read (خواندن داده به صورت بلوکی):** در این روش با استفاده از دستور READ_SINGLE_BLOCK ( CMD17) می توانیم یک بلوک داده را از آدرس بلوکی که در Argument دستور قرار داده ایم ، بخوانیم واندازه یک بلوک داده بستگی به مقدار فیلد READ_BLK_LEN (در رجیستر CSD ) دارد.

حال اگر نیاز به ارسال چند بلوک داده داشتیم می توانیم با استفاده از دستور (CMD18 ) READ_MULTIPLE_BLOCK ، mmc شروع به ارسال چند بلوک داده می کند البته قبل از ارسال این دستور باید دستور (CMD23 ) SET_BLOCK_COUNT فرستاده شود چون این دستور تعداد بلوک هایی که باید خوانده شود را مشخص می کند. در این روش در پایان ارسال هر بلوک فیلد CRC16 (16 بیتی) وجود دارد.

## 5- Response (پاسخ) :

همان طور که دستور ها توسط Host از طریق خط(پین) CMD فرستاده می شوند پاسخ ها نیز از طریق همین خط دریافت می شوند. به ازای همه دستور هایی که فرستاده می شود ممکن است یکی از چهار نوع پاسخ زیر دریافت شود:

**5-1- R1 :** این پاسخ 48 بیتی است و وضعیت mmc را به ما گزارش میدهد. پس از دریافت این پاسخ و بررسی بیت های فیلد Status از وضعیت هر یک از قسمت های mmc مطلع می شویم.

**5-2- R1b :** این پاسخ شبیه R1 است فقط دارای یک بیت اضافی است که مشغول بودن خط (پین) DAT را گزارش میکند.

**5-3- R2 :** این پاسخ مخصوص رجیستر های CID و CSD و طول آن 128 بیت است. پاسخ R2 محتویات این دو رجیستر را برای ما می فرستد.

**5-4- R3 :** این پاسخ مخصوص رجیستر OCR و دارای طول 48 بیت است.

OCR در پاسخ به دستورCMD1 از طرف mmc به

| | | | | |
|---|---|---|---|---|
| 0 | 0 | bit5...bit0 | bit127...bit1 | 1 |
| start bit | card | reserved | CID or CSD register including internal CRC | end bit |

Host فرستاده می شود و رنج ولتاژ هایی که توسط mmc پشتیبانی می شود نشان می دهد معمولا رنج این ولتاژ ها بین 2.7 تا 3.6 ولت است.

اگر فیلد OCR در این پاسخ دارای مقدار 0x80FF8000 باشد نشان می دهد که کارت فعال شده است اما اگر این فیلد دارای مقدار 0x00FF8000 باشد(یعنی بیت 32 دارای مقدار 0 باشد) نشان می دهد که mmc آمادگی فعال شدن را ندارد (مشغول است).

در آخر برای روشن تر شدن مطلب یک سری ار سیگنال ها که شامل خواندن از mmc و نوشتن در mmc و... را به این مطالب اضافه خواهیم کرد.

| 0 | 0 | bit5...bit0 | bit31...bit0 | bit6...bit0 | 1 |
|---|---|---|---|---|---|
| start bit | card | reserved | OCR field | reserved | end bit |

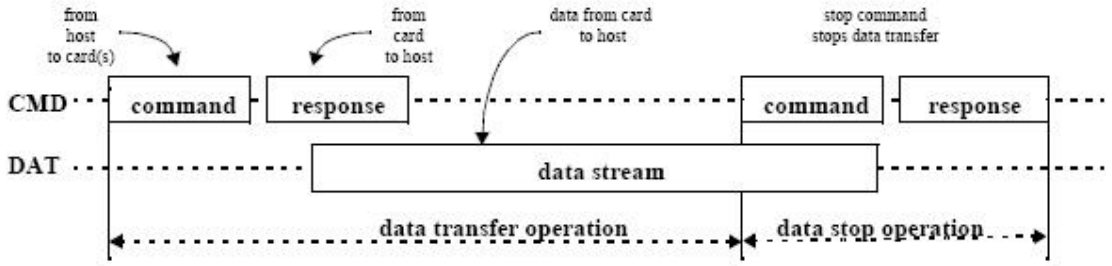



Figure 7: Sequential read operation

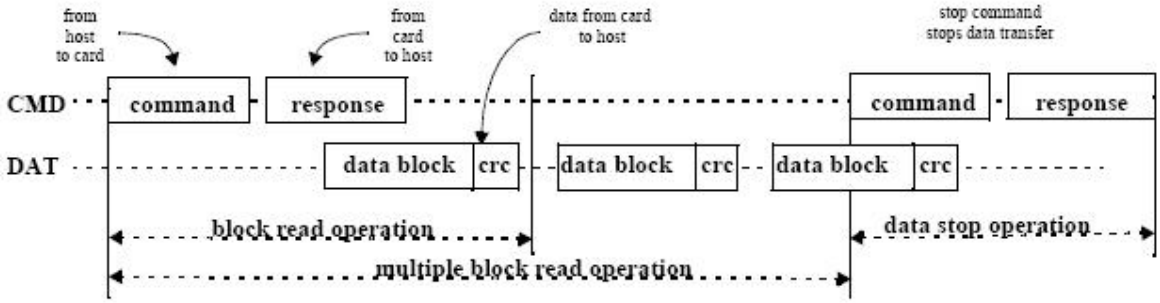



Figure 8: (Multiple) Block read operation

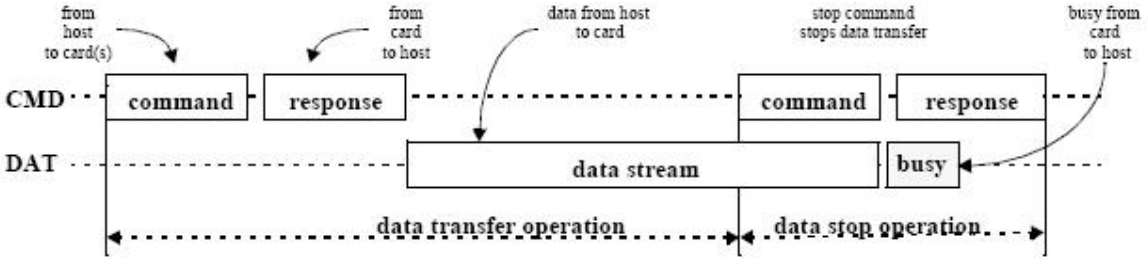



Figure 9: Sequential write operation

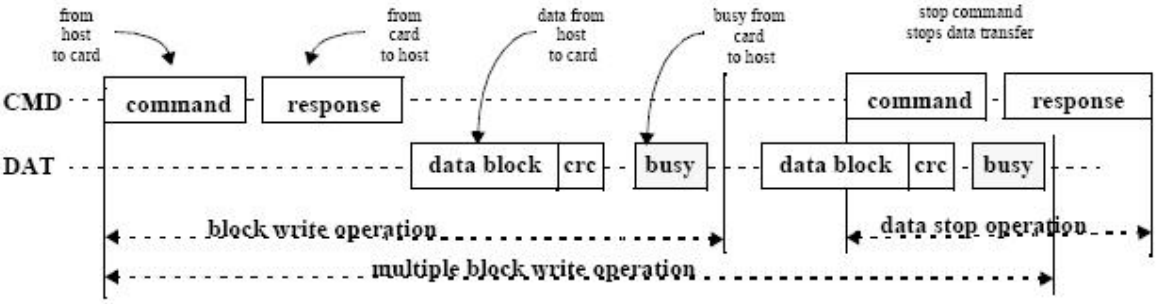



Figure 10: (Multiple) Block write operation

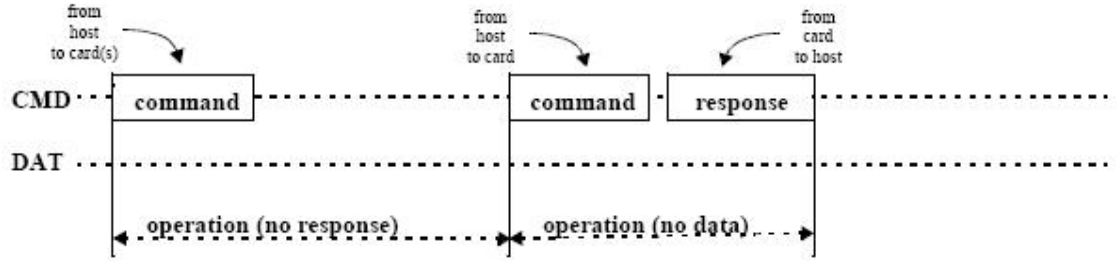



Figure 11: ?no response? and ?no data? operations

در ایـن مقالـه بیشـتر در مـورد پروتکـل mmc صـحبت کردیم بحث در مورد این پروتکل بسیار گسترده اسـت و ما فقط چکیده ای از آن را ارائه دادیم.

اما در یک مقاله دیگر که در همین کنفرانس ارائه خواهد یـک پـروژه عملـی بـه عنـوان (( ***ارتبـاط بـین میکروکنترلـر AVR وMMC***)) اسـت کـه در ایـن مقاله از پروتکل SPI استفاده شده است

## 6- منابع:

**Reference**:

1- http://www.Hitachi.com

2- http://www.hitachisemiconductor.com

3- http://www.MMCA.com/