The MultiMediaCard

System Specification

Version 3.31

MMCA Technical Committee

Revision History

Version	Date	Section/ Page	Changes compared to previous issue
1.0	09-09-96	all	Initial version of system specification
1.1	27-09-96	Chap. 4,6,10	Complete revision
1.2	15-11-96	all	
1.4	25-2-98	all	Complete revision. Added new chapters
2.0	1/99	all	Several changes in SPI mode (chapter 7). Smaller changes in chapters 2, 4, 5. All references to “MMC” changed to “MultiMediaCard”.
2.1	5/99	4, 5, 7, 9, 10, 11	Several new features introduced: • Data interchange format • Password locking • Application specific commands Several clarifications and minor corrections for MultiMediaCard mode (chapter 4) and SPI mode (chapter 7). Backward compatible change of card dimensional tolerances (chapter 9).
2.11	6/99	4	Corrections in the status/command cross reference table (chapter 4)
2.2	1/00	7, 9	ESD tolerance, SPI status bits, Definition of card top side.
3.0	1/01	All	This is a major upgrade of the spec that contains new mechanical, electrical as well as protocol changes. See appendix for details.
3.1	6/01	5	The CSD version identification fields in 3.0 version are not compatible with old versions. 3.1 fixes the issues, replaces and obsoletes 3.0
3.2	1/02	4, 5, 6, 9	Definition of Low Voltage MultiMediaCard completed. Some details added to the mechanical form factor of the card. Erase and Group write protect classes defined as mandatory for Read/Write cards and 2 CSD bits where allocated for application specific use.
3.3	3/03	2, 3, 8	Introduced the reduced size MMC. The high voltage range is now mandatory for all cards, as well as OCR register and CMD1 implementation. Corrected some typos in the wording.
3.31	5/03	A	Corrected connector order in figure 72

You acknowledge that the attached standard (the “Standard”) is provided to you on an “AS IS” basis. MULTIMEDIACARD ASSOCIATION (“MMCA”) MAKES NO EXPRESS, IMPLIED OR STATUTORY WARRANTIES AND EXPRESSLY DISCLAIMS THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS. MMCA SHALL NOT BE LIABLE FOR (I) TECHNICAL OR EDITORIAL ERRORS OR OMISSIONS CONTAINED WITHIN THE STANDARD, OR (II) ANY INCIDENTAL, SPECIAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS OR LOSS OF USE) RESULTING FROM THE FURNISHING, PERFORMANCE OR USE OF THE STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Table Of Contents

Table Of Contents

List Of Figures

1 General Description

The MultiMediaCard is an universal low cost data storage and communication media. It is designed to cover a wide area of applications as electronic toys, organizers, PDAs, cameras, smart phones, digital recorders, MP3 players, pagers, etc. Targeted features are high mobility and high performance at a low cost price. It might also be expressed in terms of low power consumption and high data throughput at the memory card interface.

The MultiMediaCard communication is based on an advanced 7-pin serial bus designed to operate in a low voltage range. The communication protocol is defined as a part of this standard and referred to as MultiMediaCard mode. For compatibility to existing controllers the cards may offer, in addition to the MultiMediaCard mode, an alternate communication protocol which is based on the SPI standard.

To provide for the forecasted migration of CMOS power (V_{DD}) requirements and for compatibility and integrity of MultiMediaCard systems, two types of MultiMediaCard(s) are defined in this standard specification, which differ only in the valid range of system V_{DD}. These two card types are referred to as High Voltage MultiMediaCard and Low Voltage MultiMediaCard.

The purpose of the system specification is the definition of the MultiMediaCard, its environment and handling. It gives guidelines for a system designer. The system specification also defines a tool box (a set of macro functions and algorithms) which helps reducing the design-in costs.

The document is split up into several portions. Chapter 3 gives a general overview of the system components: card, bus and host.

The common MultiMediaCard characteristics are described in Chapter 4. As this description defines an overall set of card properties, it is recommend to use the, vendor specific, product documentation in parallel. The card registers are described in Chapter 5.

Chapter 6 defines the MultiMediaCard bus as an universal communication interface, and the electrical parameters of the interface. The SPI mode is described in Chapter 7.

All error protection techniques employed in this standard are described in Chapter 8.

Chapter 9 describes the physical and mechanical properties of the cards and the minimal requirements to the card slots or cartridges.

The standard compliance criteria for the cards and hosts are described in Chapter 10. For achieving high data interchangeability, three basic file formats are defined in Chapter 11 as valid file formats for the MultiMediaCard

Some application specific remarks are attached to this document (Appendix A). The application notes contain useful hints for the circuit and system designers, helping simplifying the design process. Finally, the major changes between the previous and the current version are listed in Appendix B.

As used in this document, “shall” or “will” denotes a mandatory provision of the standard. “Should” denotes a provision that is recommended but not mandatory. “May” denotes a feature whose presence does not preclude compliance, that may or may not be present at the option of the implementor.

2 System Features

- **Targeted for portable and stationary applications**
- **System Voltage (V_{DD}) Range:**

	High Voltage MultiMediaCard	Low Voltage MultiMediaCard
Communication	**2.7 - 3.6**	**1.65 - 1.95, 2.7 - 3.6**[1]
Memory Access	**2.7 - 3.6**	**1.65 - 1.95, 2.7 - 3.6**

1) V_{DD} range: 1.95V - 2.7V is not supported.

- **Designed for read-only, read/write and I/O cards**
- **Variable clock rate 0 - 20 MHZ**
- **Maximum data rate with up to 10 cards**
- **Password protection of data**
- **Basic file formats for high data interchangeability**
- **Application specific commands**
- **Correction of memory field errors**
- **Optional data cache to enhance the access performance**
- **Built-in write protection features (permanent and temporary)**
- **Comfortable erase mechanism**
- **Protocol dependent attributes of the communication channel:**

MultiMediaCard Mode	SPI Mode
Three-wire serial data bus (clock, command, data)	Three-wire serial data bus (clock, dataIn, dataOut) + card specific CS signal.
Up to 64k cards addressable by the bus protocol	Card selection via a hardware CS signal
Up to 30 cards stackable on a single physical bus	Card stacks require a “per card” CS signal.
Easy identification and assignment of session address to individual cards in a card stack	Not available. Card selection via a hardware CS signal
Error-protected data transfer	Optional. A non-protected data transfer mode is available.
Sequential and Single/Multiple block Read/Write commands	Single/Multiple block Read/Write commands

- **Two form factors: Normal size(24mm x 32mm x 1.4mm) and Reduced size (24mm x 18mm x 1.4mm)**

3 MultiMediaCard System Concept

The main design goal of the MultiMediaCard system is to provide a very low cost mass storage product, implemented as a 'card' with a simple controlling unit, and a compact, easy-to-implement interface. These requirements lead to a reduction of the functionality of each card to an absolute minimum.

Nevertheless, since the complete MultiMediaCard system has to have the functionality to work with a low cost card stack and execute tasks (at least for the high end applications) such as error correction and standard bus connectivity, the system concept is described next. It is based on modularity and the capability of reusing hardware over a large variety of cards.

Figure 1 shows four typical architectures of possible MultiMediaCard systems.

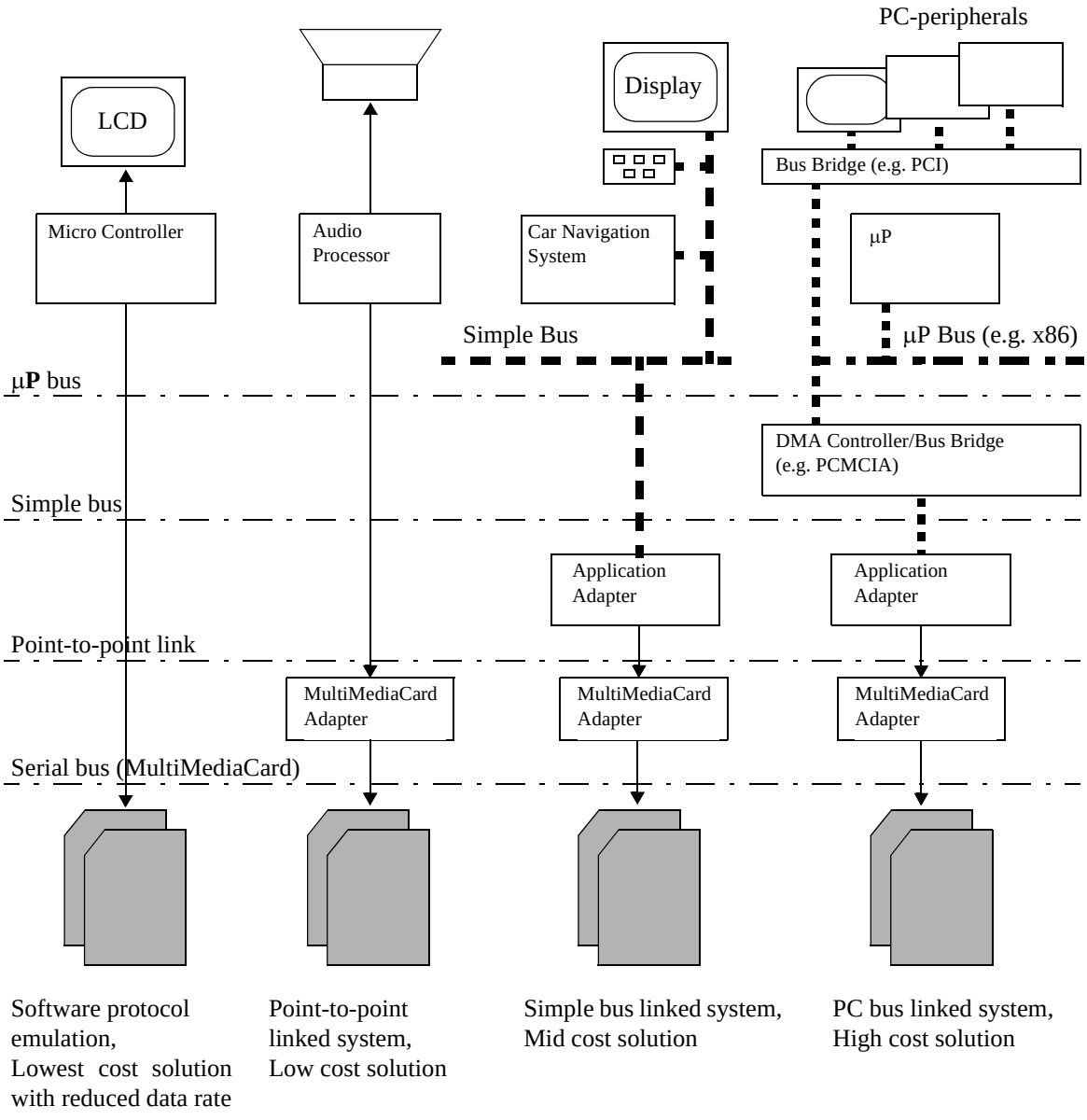


Figure 1: Topology Of MultiMediaCard Systems

Four typical types of MultiMediaCard systems can be derived from this diagram:

- Software emulation: reduced data rate (typical 100-300 kbit per second, restricted by the host)
- Point to point linkage: full data rate (with additional hardware)
- Simple bus: full data rate, part of a set of addressable units
- PC bus: full data rate, addressable, extended functionality (like DMA capabilities)

In the first variant, the MultiMediaCard bus protocol is emulated in software using three port pins of a microcontroller. This solution requires no additional hardware and is the cheapest system in the list. The other applications extend the features and requirements, step by step, towards a sophisticated PC solution. The various systems, although different in their feature set, have a basic common functionality, as can be seen in Figure 2. This diagram shows a system partitioned into hierarchical layers of abstract (’virtual’) components. It describes a logical classification of functions which cover a wide variety of implementations (see also Figure 1). It does not imply any specific design nor specify rules for implementing parts in hardware or software.

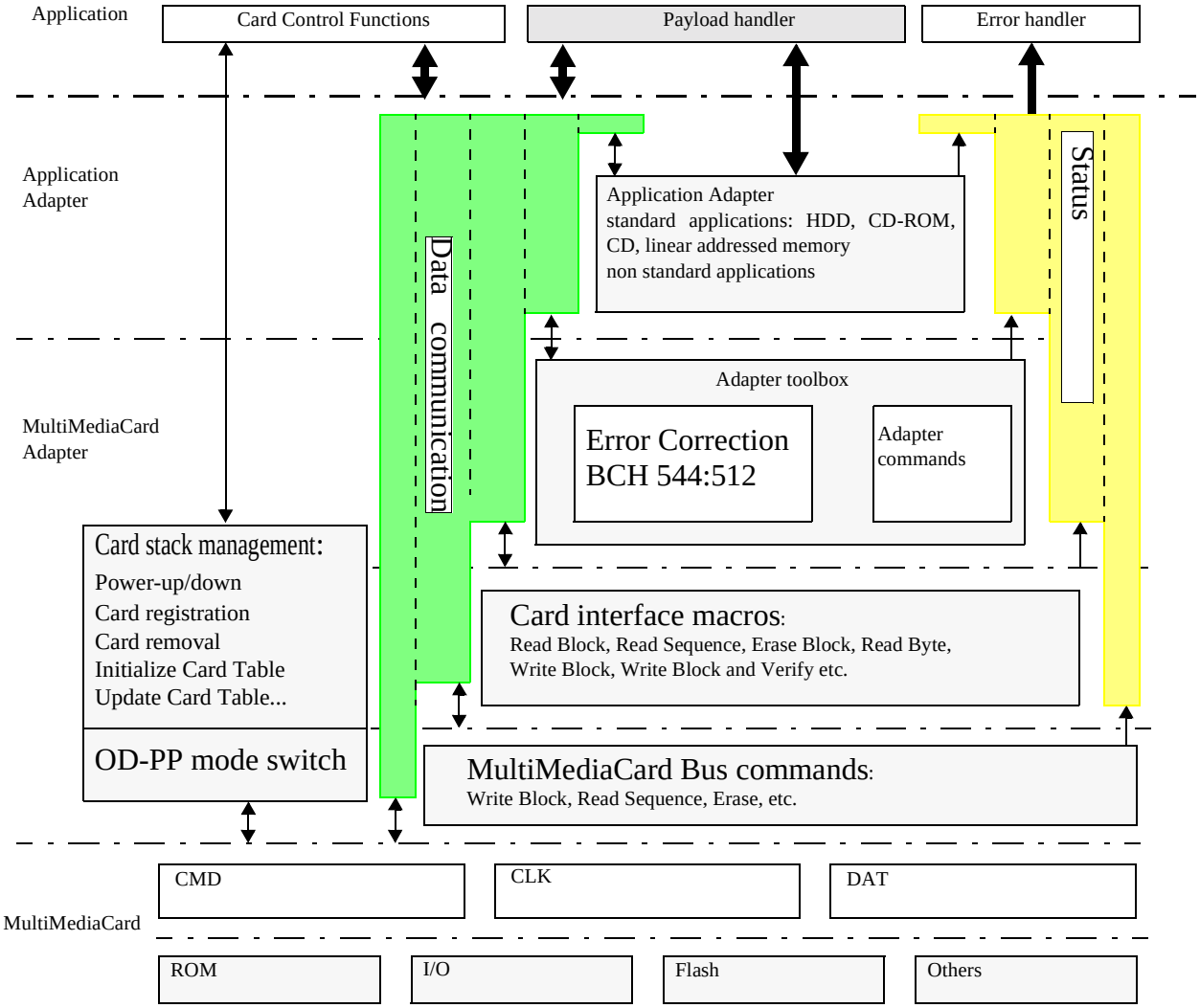


Figure 2: MultiMediaCard System Overview

Figure 3 is a specific design example based on the abstract layer model described in Figure 2

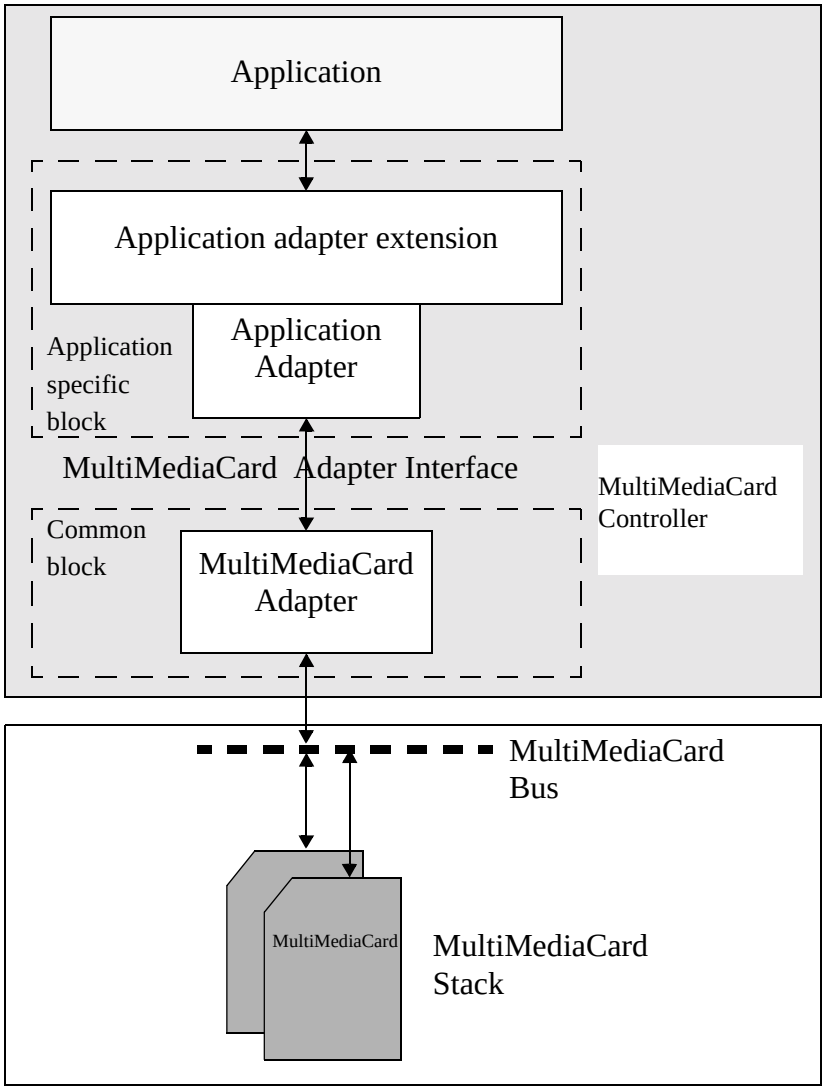


Figure 3: MultiMediaCard System Example

This MultiMediaCard system contains at least two components:

- The set of cards, called the MultiMediaCard stack
- The MultiMediaCard controller

The MultiMediaCard controller is divided into two major blocks. In some implementations (like this example) the controller may implement the whole application, while in others it may be divided into several physical components which (apart from the application itself) can be identified as:

- Application specific block (e.g. a microprocessor or an adapter to a standard bus like USB or ATA) = Application adapter
  - Performs application oriented tasks (e.g. display controlling or input decoding for hand-held applications).
  - Typically connected as a bus slave for a standard bus

- The common block = MultiMediaCard adapter
  - Contains all card specific functions (like initialization and error correction)
  - Serves as a bus master for the MultiMediaCard bus
  - Implements the standard interface to the card stack.

3.1 Card Concept

The basic MultiMediaCard concept is based on transferring data via a minimal number of signals. The communication signals are:

- **CLK**: with each cycle of this signal an one bit transfer on the command and data lines is done. The frequency may vary between zero and the maximum clock frequency.
- **CMD**: is a bidirectional command channel used for card initialization and data transfer commands. The CMD signal has two operation modes: open-drain for initialization mode and push-pull for fast command transfer. Commands are sent from the MultiMediaCard bus master to the card and responses from the cards to the host.
- **DAT**: is a bidirectional data channel. The DAT signal operates in push-pull mode. Only one card or the host is driving this signal at a time.

MultiMediaCards can be grouped into several card classes which differ in the functions they provide (given by the subset of MultiMediaCard system commands):

- Read Only Memory (ROM) cards. These cards are manufactured with a fixed data content. They are typically used as a distribution media for software, audio, video etc.
- Read/Write (RW) cards (Flash, One Time Programmable - OTP, Multiple Time Programmable - MTP). These cards are typically sold as blank (empty) media and are used for mass data storage, end user recording of video, audio or digital images.
- I/O cards. These cards are intended for communication (e.g. modems) and typically will have an additional interface link.

All cards are connected directly to the signals of the MultiMediaCard bus. The following table defines the card contacts:

Pin No.	Name	Type[1]	Description
1	RSV	NC	Reserved for future use
2	CMD	I/O/PP/OD	Command/Response
3	V_{SS1}	S	Supply voltage ground
4	V_{DD}	S	Supply voltage
5	CLK	I	Clock
6	V_{SS2}	S	Supply voltage ground
7	DAT[2]	I/O/PP	Data

Table 1: MultiMediaCard pad definition

1)S: power supply; I: input; O: output; PP: push-pull; OD: open-drain; NC: Not connected (or logical high)
2)The DAT line for read-only cards is output only

The card stack initialization uses only the CMD channel and is, therefore, compatible for all cards.

Each card has a set of information registers (see also Chapter 5):

Name	Width	Description	Implementation
CID	128	Card IDentification number, a card individual number for identification.	Mandatory
RCA	16	Relative Card Address, is the card system address, dynamically assigned by the host during initialization.	Mandatory
DSR	16	Driver Stage Register, to configure the card's output drivers.	Optional
CSD	128	Card Specific Data, information about the card operation conditions.	Mandatory
OCR	32	Operation Conditions Register. Used by a special broadcast command to identify the voltage type of the card.	Mandatory

Table 2: MultiMediaCard registers

The host may reset the cards by switching the power supply off and back on. Each card shall have its own power-on detection circuitry which puts the card into a defined state after the power-on. No explicit reset signal is necessary. The cards can also be reset by a special command.

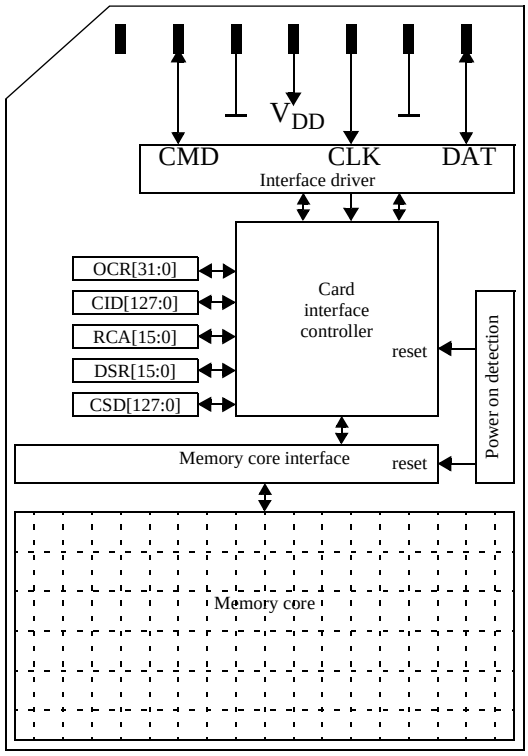


Figure 4: MultiMediaCard Architecture

3.1.1 Form Factors

The MultiMediaCard has two possible form factors. The normal size form factor is 24mm x 32mm x 1.4mm. The reduced size form factor is 24mm x 18mm x 1.4mm. To use a reduced size MMC in a normal size MMC socket, a special adaptor has to be used. Figure 5 shows the two form factors. The mechanical and electrical

interface is identical in both form factors.

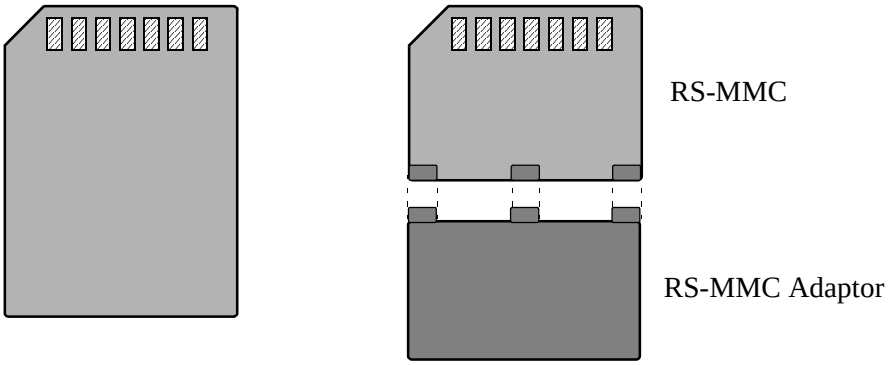


Figure 5: MultiMediaCard Form Factors (Bottom View)

3.2 Bus Concept

The MultiMediaCard bus is designed to connect either solid-state mass-storage memory or I/O-devices in a card format to multimedia applications. The bus implementation allows the coverage of application fields from low-cost systems to systems with a fast data transfer rate. It is a single master bus with a variable number of slaves. The MultiMediaCard bus master is the bus controller and each slave is either a single mass storage card (with possibly different technologies such as ROM, OTP, Flash etc.) or an I/O-card with its own controlling unit (on card) to perform the data transfer.

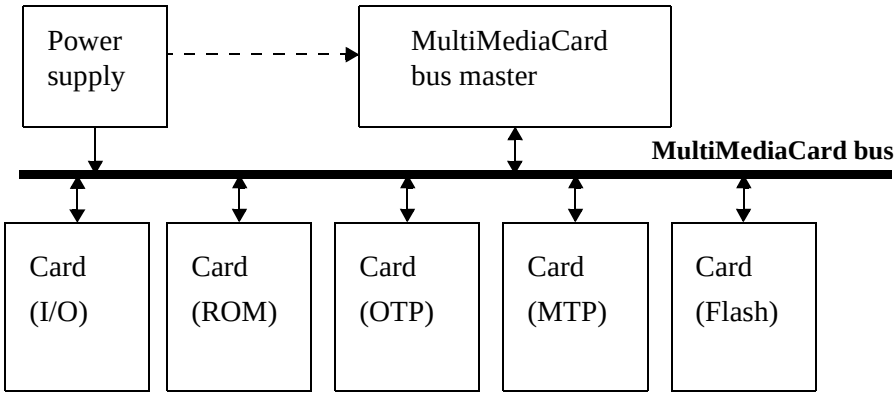


Figure 6: MultiMediaCard Bus System

The MultiMediaCard bus also includes power connections to supply the cards.

The bus communication uses a special protocol (MultiMediaCard bus protocol) which is applicable for all devices. Therefore, the payload data transfer between the host and the cards can be bidirectional.

3.2.1 Bus Lines

The MultiMediaCard bus architecture requires all cards to be connected to the same set of lines. No card has an individual connection to the host or other devices, which reduces the connection costs of the MultiMediaCard system.

The bus lines can be divided into three groups:

- Power supply: V_{SS1} and V_{SS2},V_{DD} - used to supply the cards.
- Data transfer: CMD, DAT - used for bidirectional communication.
- Clock: CLK - used to synchronize data transfer across the bus.

The bus line definitions and the corresponding pad numbers are described in Chapter 3.1.

3.2.2 Bus Protocol

After a power-on reset, the host must initialize the cards by a special message-based MultiMediaCard bus protocol. Each message is represented by one of the following tokens:

- command: a command is a token which starts an operation. A command is sent from the host either to a single card (addressed command) or to all connected cards (broadcast command). A command is transferred serially on the CMD line.
- response: a response is a token which is sent from an addressed card, or (synchronously) from all connected cards, to the host as an answer to a previously received command. A response is transferred serially on the CMD line.
- data: data can be transferred from the card to the host or vice versa. Data is transferred via the data line.

Card addressing is implemented using a session address assigned during the initialization phase, by the bus controller to all currently connected cards. Individual cards are identified by their CID number. This method requires that every card will have an unique CID number. To ensure uniqueness of CIDs the CID register contains 24 bits (MID and OID fields - see Chapter 5) which are defined by the MMCA. Every card manufacturers is required to apply for an unique MID (and optionally OID) number.

The structure of commands, responses and data blocks is described in Chapter 4.

MultiMediaCard bus data transfers are composed of these tokens. One data transfer is a *bus operation*. There are different types of operations. Addressed operations always contain a command and a response token. In addition, some operations have a data token, the others transfer their information directly within the command or response structure. In this case no data token is present in an operation. The bits on the DAT and the CMD lines are transferred synchronous to the host clock.

Two types of data transfer commands are defined:

- Sequential commands: These commands initiate a continuous data stream, they are terminated only when a stop command follows on the CMD line. This mode reduces the command overhead to an absolute minimum.
- Block-oriented commands: These commands send a data block succeeded by CRC bits. Both read and write operations allow either single or multiple block transmission. A multiple block transmission is terminated when a stop command follows on the CMD line similarly to the sequential read.

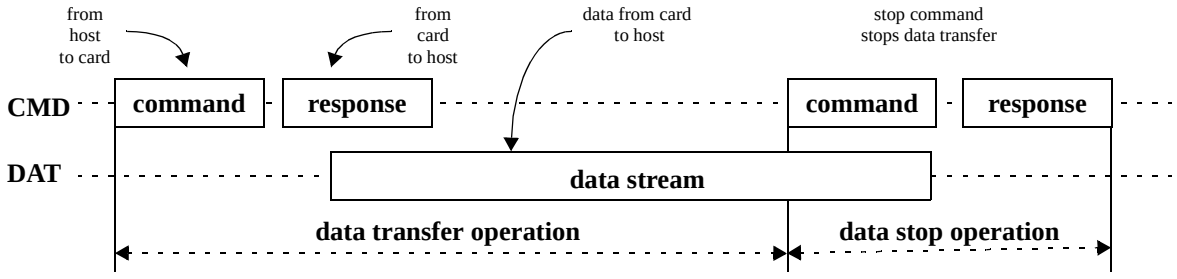


Figure 7: Sequential Read Operation

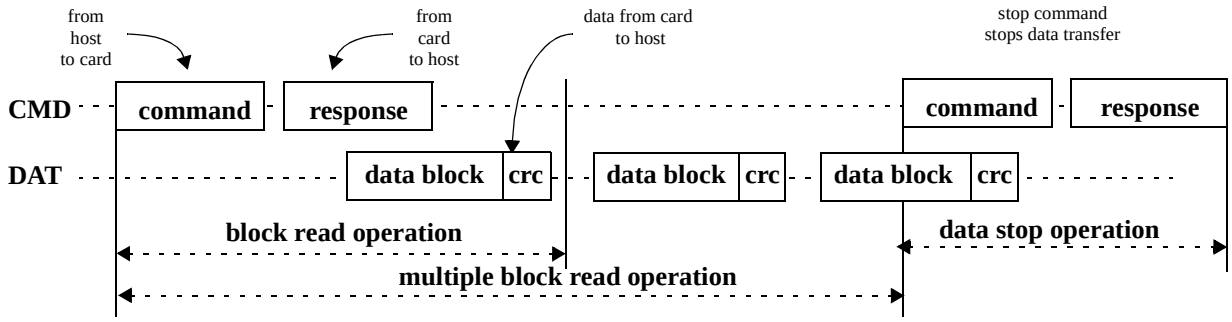


Figure 8: (Multiple) Block Read Operation

.

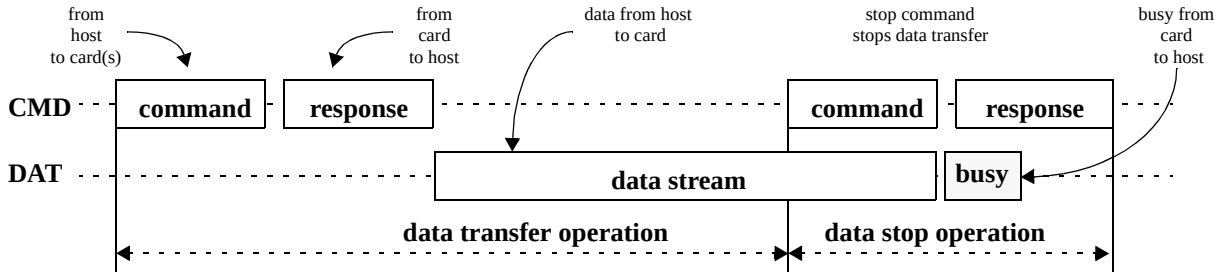


Figure 9: Sequential Write Operation

The block write operation uses a simple busy signalling of the write operation duration on the data (DAT) line (see Figure 10).

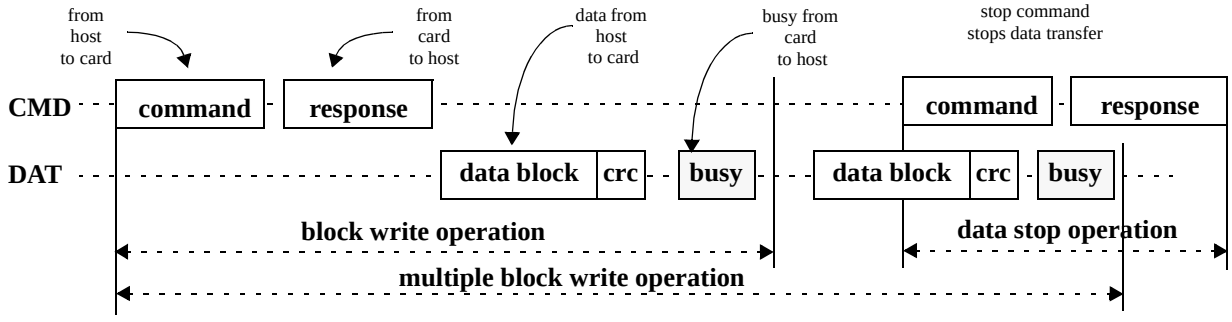


Figure 10: (Multiple) Block Write Operation

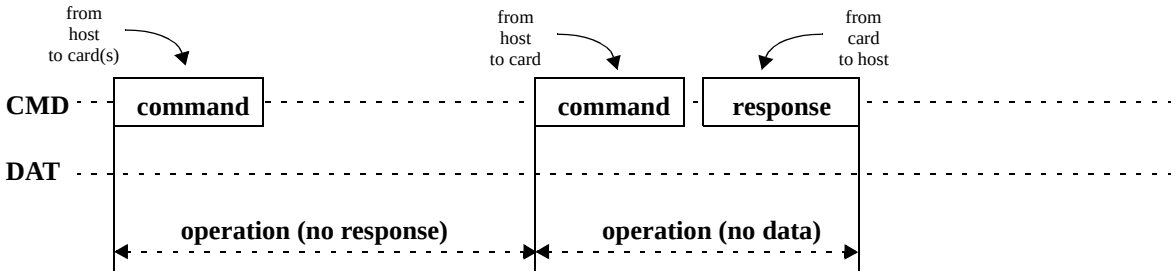


Figure 11: “no response” And “no data” Operations

Command tokens have the following coding scheme:

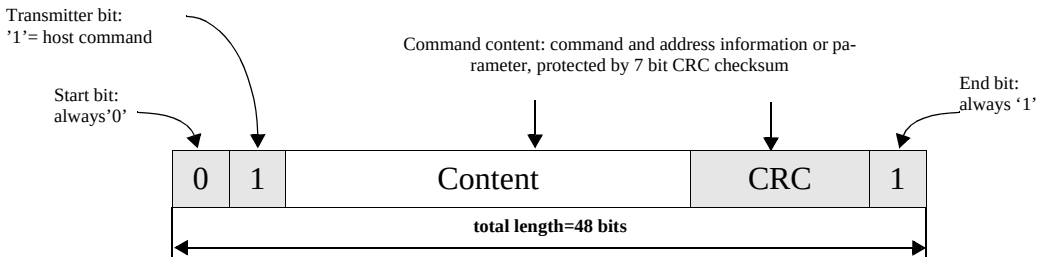


Figure 12: Command Token Format

Each command token is preceded by a start bit (‘0’) and succeeded by an end bit (‘1’). The total length is 48 bits. Each token is protected by CRC bits so that transmission errors can be detected and the operation may be repeated.

Response tokens have five coding schemes depending on their content. The token length is either 48 or 136 bits. The detailed commands and response definition is given in Chapter 4.7 and Chapter 4.9.

Due to the fact that there is no predefined end in sequential data transfer, no CRC protection is included in this case. The CRC protection algorithm for block data is a 16 bit CCITT polynomial. All used CRC types are

described in Chapter 8.

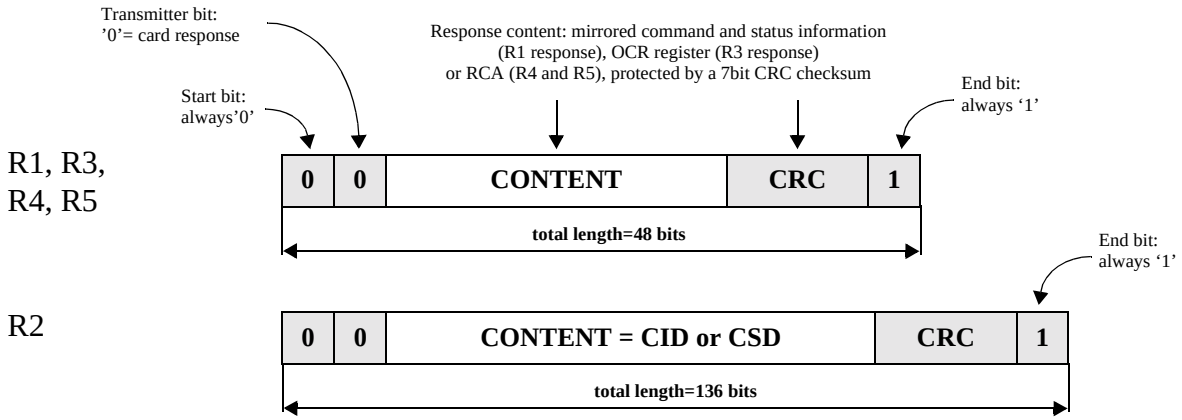


Figure 13: Response Token Format

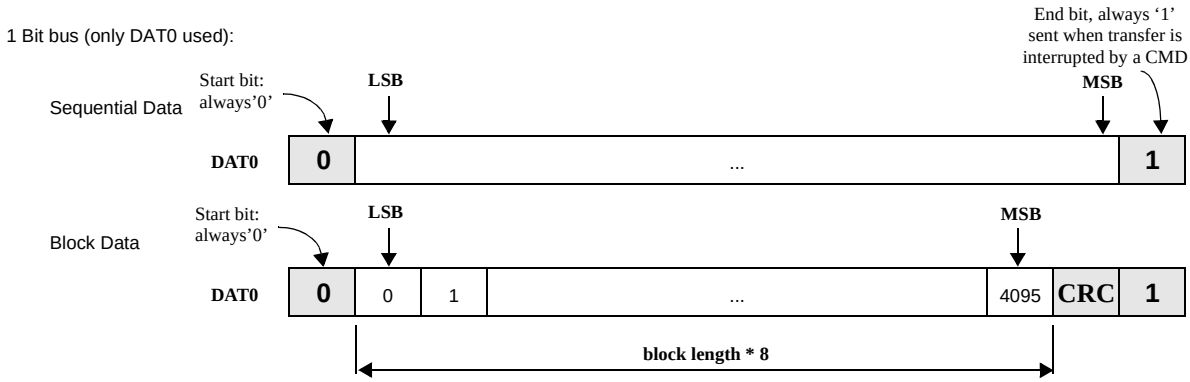


Figure 14: Data Packet Format

3.3 Controller Concept

The MultiMediaCard is defined as a low cost mass storage product. The shared functions have to be implemented in the MultiMediaCard system. The unit which contains these functions is called the MultiMediaCard controller. The following points are basic requirements for the controller:

- Protocol translation from standard MultiMediaCard bus to application bus
- Data buffering to enable minimal data access latency
- MultiMediaCard stack management to relieve the application processor
- Macros for common complex command sequences

The MultiMediaCard controller is the link between the application and the MultiMediaCard bus with its cards. It translates the protocol of the standard MultiMediaCard bus to the application bus. It is divided into two major parts:

- The application adapter: the application oriented part

- The MultiMediaCard adapter: the MultiMediaCard oriented part

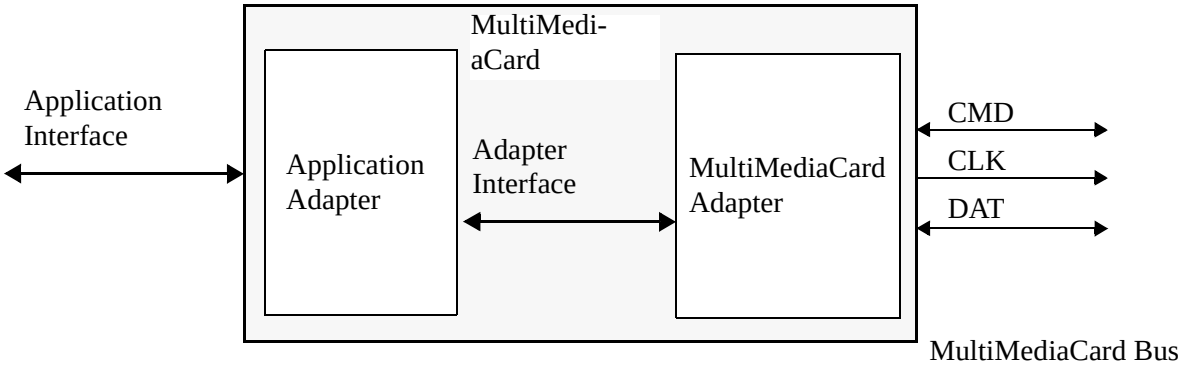


Figure 15: MultiMediaCard Controller Scheme

The application adapter consists at least of a bus slave and a bridge into the MultiMediaCard system. It can be extended to become a master on the application bus and support functions like DMA or serve application specific needs. Higher integration will combine the MultiMediaCard controller with the application.

Independently of the type and requirements of the application the MultiMediaCard bus requires a host. This host may be the MultiMediaCard adapter. On the MultiMediaCard bus side it is the only bus master and controls all activity on that bus. On the other side it is a slave to the application adapter or to the application, respectively. No application specific functions shall be supported here except those that are common to most MultiMediaCard systems. The adapter includes all card stack management functions. It supports all MultiMediaCard bus commands and provides additionally a set of macro commands. The adapter includes error correction capability for non error-free cards. The used error correction codes are defined in Chapter 8.1.

Because the application specific needs and the chosen application interface are out of the scope of this specification, the MultiMediaCard controller defines an internal adapter interface. The two parts communicate across this interface. The adapter interface is directly accessible in low cost (point to point link) systems where the MultiMediaCard controller is reduced to an MultiMediaCard adapter.

3.3.1 Application Adapter Requirements

The application adapter enhances the MultiMediaCard system in the way that it becomes plug&play in every standard bus environment. Each environment will need its unique application adapter. For some bus systems standard off the shelf application adapters exist and can interface with the MultiMediaCard adapter. To reduce the bill of material it is recommended to integrate an existing application adapter with the MultiMediaCard adapter module to form an MultiMediaCard controller.

The application adapter extension is a functional enhancement of the application adapter from a bus slave to a bus master on the standard application bus. For instance, an extended application adapter can be triggered to perform bidirectional DMA transfers.

3.3.2 MultiMediaCard Adapter Architecture

The architecture and the functional units described below are not implementation requirements, but general recommendations on the implementation of a MultiMediaCard adapter. The adapter is divided into two major parts:

- The controller: macro unit, stack management and power management
- The data path: Adapter interface, ECC unit, read cache, write buffer, CRC unit and MultiMediaCard bus interface

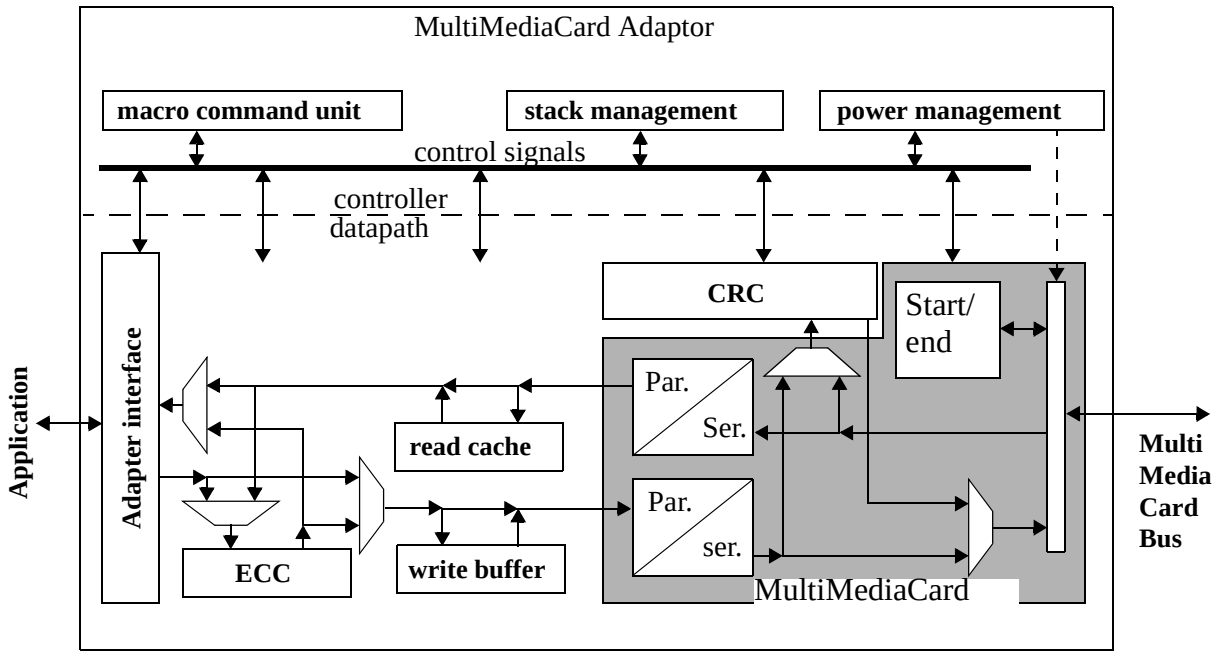


Figure 16: MultiMediaCard Adaptor Architecture

The data path units should be implemented in hardware to guarantee the full capabilities of the MultiMediaCard system. The controller part of the adapter can be implemented in hardware or software depending on the application architecture.

The width of the data path should be a byte; the units which are handling data should work on bytes or blocks of bytes. This requirement is derived from the MultiMediaCard bus protocol, which is organized in data blocks. Blocks are multiples of bytes. Thus, the smallest unit of a data access or control unit is a byte.

Commands for the MultiMediaCard bus follow a strict protocol. Each command is encapsulated in a syntactical frame. Each frame contains some special control information like start/end bits and CRC protection. Some commands include stuffing bits to enable simple interpreters to use a fixed frame length. This transport management information should be generated in the MultiMediaCard adapter. These functions are combined in the MultiMediaCard bus interface of the adapter.

The response delays of the MultiMediaCard system may vary; they depend on the type of cards. So the adapter interface must handle asynchronous mode via handshake signals(STB,ACK) or the host has to poll the state (busy/not busy) if no handshake signals are required (synchronous mode). This interface may be a general unit supporting most application protocols or can be tailored to one application.

It is recommended to equip the MultiMediaCard adapter with data buffers for write an read operation. It will, in most cases, improve the system level performance on the application side.The MultiMediaCard bus transports its data in a serial protocol with a data rate up to 20 Mbit. This is slower than a typical applications CPU bus. Enabling the CPU to off load the data to the buffers will free up CPU time for system level tasks, while the MMC adapter handles the data transfer to the card.

The access time for random access read operations from a card may be improved by caching a block of data in the read cache. After reading a complete block into the MultiMediaCard adapter cache, repeated accesses to

that block can be done very fast. Especially read-modify-write operations can be executed in a very efficient way on a block buffer with the help of the SRAM swapper.

4 MultiMediaCard Functional Description

4.1 General

All communication between host and cards is controlled by the host (master). The host sends commands of two types: broadcast and addressed (point-to-point) commands.

- Broadcast commands

Broadcast commands are intended for all cards. Some of these commands require a response.

- Addressed (point-to-point) commands

The addressed commands are sent to the addressed card and cause a response from this card.

A general overview of the command flow is shown in Figure 17 for the card identification mode and in Figure 19 for the data transfer mode. The commands are listed in the command tables (Table 6 - Table 14). The dependencies between current state, received command and following state are listed in Table 15. In the following sections, the different card operation modes will be described first. Thereafter, the restrictions for controlling the clock signal are defined. All MultiMediaCard commands together with the corresponding responses, state transitions, error conditions and timings are presented in the succeeding sections.

Three operation modes are defined for the MultiMediaCard system (host and cards):

- Card identification mode
  The host will be in card identification mode after reset and while it is looking for new cards on the bus. Cards will be in this mode after reset until the SET_RCA command (CMD3) is received.

- Interrupt mode
  Host and all cards will enter and exit interrupt mode simultaneously. In the interrupt mode there is no data transfer. The only message allowed is an interrupt service request from one of the cards or the host.

- Data transfer mode
  Cards will enter data transfer mode once an RCA is assigned to them. The host will enter data transfer mode after identifying all the cards on the bus.

The following table shows the dependencies between bus modes, operation modes and card states. Each state in the MultiMediaCard state diagram (see Figure 17 and Figure 19) is associated with one bus mode and one operation mode:

Card state	Operation mode	Bus mode
Inactive State	inactive	open-drain
Idle State	card identification mode	
Ready State		
Identification State		

Table 3: Bus Modes Overview

Card state	Operation mode	Bus mode
Stand-by State	data transfer mode	push-pull
Transfer State		
Sending-data State		
Receive-data State		
Programming State		
Disconnect State		
Wait-IRQ State	interrupt mode	open-drain

Table 3: Bus Modes Overview

4.2 Card Identification Mode

While in card identification mode the host resets all cards, validates operation voltage range, identifies cards and sets a Relative Card Address (RCA) to each card on the bus. All data communication in the Card Identification Mode uses the command line (CMD) only.

4.2.1 Card Reset

After power-on by the host, all cards (including the cards having been in *Inactive State)* are in MultiMediaCard mode (as opposed to SPI mode) and in *Idle State*.

Command GO_IDLE_STATE (CMD0) is the software reset command and sets all cards into *Idle State.* It is also used to switch the card into SPI mode (refer to Chapter 7 for details).

After power-on or CMD0, all cards' output bus drivers are in an high-impedance state and the cards are initialized with a default relative card address („0x0001“) and with a default driver stage register setting (lowest speed, highest driving current capability). The host clocks the bus at the identification clock rate f_{OD} (see Chapter 6.7).

CMD0 is valid in all states with the exception of the *Inactive* State. While in *Inactive* state the card will not accept CMD0 unless it is used to switch the card into SPI mode.

4.2.2 Operating Voltage Range Validation

Each type of MultiMediaCard (Either High voltage or Low Voltage) shall be able to establish communication with the host, as well as perform the actual card function (e.g. accessing memory), using any operating voltage within the voltage range specified in this standard for the given card type (See Chapter 6.5).

The SEND_OP_COND (CMD1) command is designed to provide MultiMediaCard hosts with a mechanism to identify and reject cards which do not match the V_{DD} range desired by the host. This is accomplished by the host sending the required V_{DD} voltage window as the operand of this command (See Chapter 5.1). Cards which can not perform data transfer in the specified range must discard themselves from further bus operations and go into *Inactive State*. All other cards will respond concurrently sending back their V_{DD} range. The wired-and result of their responses will show all voltage ranges supported by the remaining active cards. For this, the levels in the OCR register shall be defined accordingly (see Chapter 5.1).

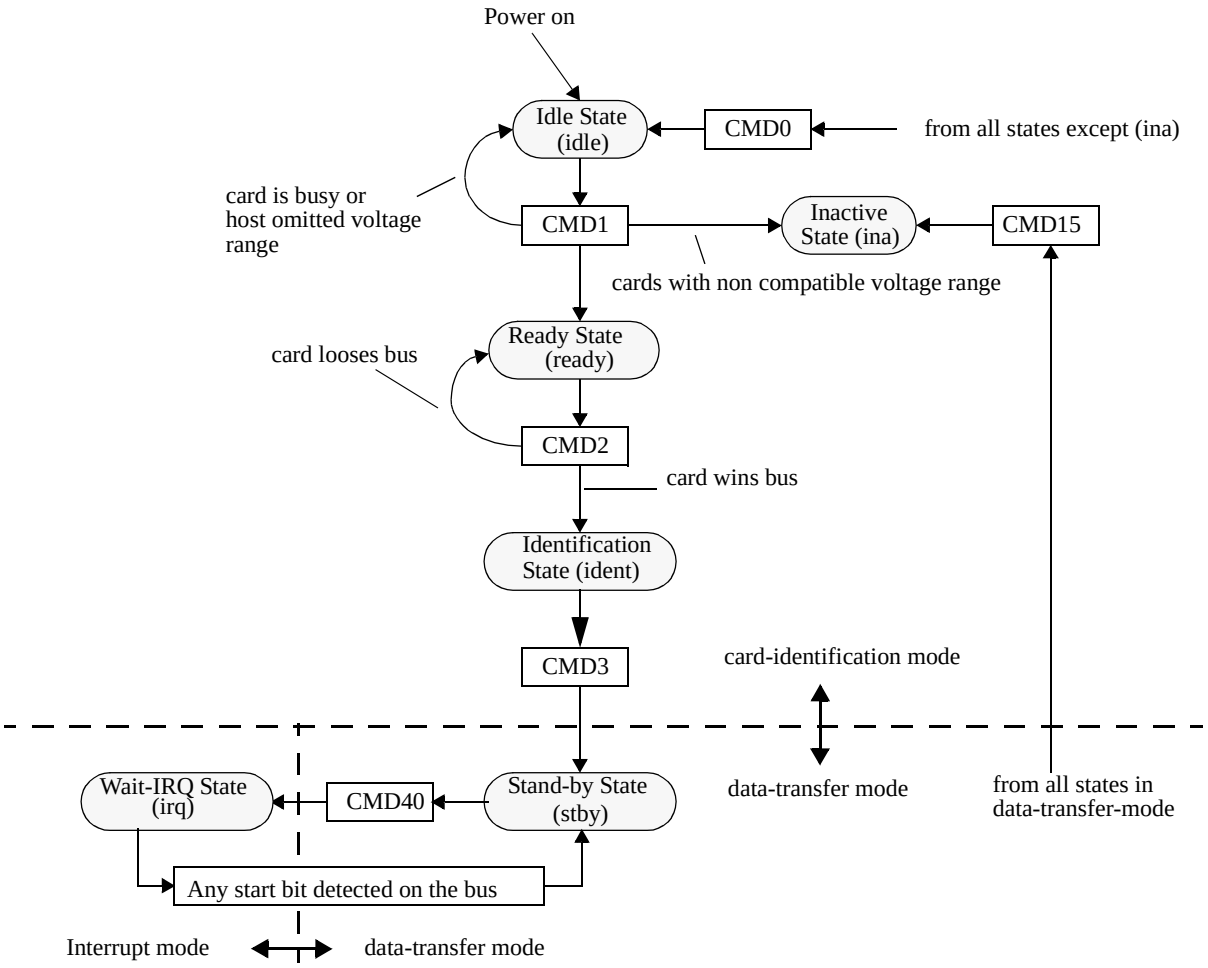


Figure 17: MultiMediaCard State Diagram (Card Identification Mode)

By omitting the voltage range in the command, the host can query the card stack and determine the common voltage range before sending out-of-range cards into the *Inactive State*. This bus query should be used if the host is able to select a common voltage range or if a notification to the application of non usable cards in the stack is desired. Afterwards, the host must choose a voltage for operation and reissue CMD1 with this condition, sending incompatible cards into the *Inactive State*.

The busy bit in the CMD1 response can be used by a card to tell the host that it is still working on its power-up/reset procedure (e.g. downloading the register information from memory field) and is not ready yet for communication. In this case the host must repeat CMD1 until the busy bit is cleared.

During the initialization procedure, the host is not allowed to change the operating voltage range. Such changes shall be ignored by the card. If there is a real change in the operating conditions, the host must reset the card stack (using CMD0) and restart the initialization procedure. However, for accessing also the cards being already in *Inactive State*, a hard reset must be done by switching the power supply off and on.

The command GO_INACTIVE_STATE (CMD15) can be used to send an addressed card into the *Inactive State*. This command is used when the host explicitly wants to deactivate a card (e.g. host is changing V_{DD} into a range which is known to be not supported by this card).

The command CMD1 shall be implemented by cards which do not support the whole range of V_{DD} defined by this standard. All other cards may simply decode and ignore this command.

If the host intends to operate the Low Voltage MultiMediaCards in the 1.65V to 1.95V range, it is recommended that the host first validate the operating voltage in the 2.7V to 3.6V range, then power the card down fully, and finally power the card back up to the 1.65V to 1.95V range for operation. Using the 2.7V to 3.6V range initially, which is common to High and Low Voltage MultiMediaCards, will allow reliable screening of host & card voltage incompatibilities. High voltage cards may not function properly if VDD < 2.0V is used to establish communication. Low voltage cards may fail if 1.95 to 2.7V is used.

4.2.3 Card Identification Process

The host starts the card identification process in open-drain mode with the identification clock rate f_{OD} (see Chapter 6.7). The open drain driver stages on the CMD line allow parallel card operation during card identification.

After the bus is activated the host will request the cards to send their valid operation conditions (CMD1). The response to CMD1 is the 'wired and' operation on the condition restrictions of all cards in the system. Incompatible cards are sent into *Inactive State*. The host then issues the broadcast command ALL_SEND_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards (i.e. those which are in *Ready State*) simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bitstream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop sending their CID immediately and must wait for the next identification cycle (remaining in the *Ready State*). Since CID numbers are unique for each card, there should be only one card which successfully sends its full CID-number to the host. This card then goes into *Identification State*. Thereafter, the host issues CMD3 (SET_RELATIVE_ADDR) to assign to this card a relative card address (RCA), which is shorter than CID and which will be used to address the card in the future data transfer mode (typically with a higher clock rate than f_{OD}). Once the RCA is received the card state changes to the *Stand-by State*, and the card does not react to further identification cycles. Furthermore, the card switches its output drivers from open-drain to push-pull.

The host repeats the identification process, i.e. the cycles with CMD2 and CMD3 as long as it receives a response (CID) to its identification command (CMD2). If no more card responds to this command, all cards have been identified. The time-out condition to recognize completion of the identification process is the absence of a start bit for more than N_{ID} clock cycles after sending CMD2 (see timing values in Chapter 4.12).

4.3 Interrupt Mode

The interrupt mode on the MultiMediaCard system enables the master (MultiMediaCard host) to grant the transmission allowance to all slaves (cards) simultaneously. This mode reduces the polling load for the host and hence, the power consumption of the system, while maintaining adequate responsiveness of the host to a card request for service. Supporting MultiMediaCard interrupt mode is an option, both for the host and the cards.

The system behavior during the interrupt mode is described in the state diagram in Figure 18.

- The host must ensure that all cards (including those cards which do not support interrupt mode) are in the *Stand-by* State before issuing the GO_IRQ_STATE (CMD40) command. While waiting for an interrupt response from a card, the host must keep the clock signal active. Clock rate may be changed according to the required response time.
- The host sets the cards into interrupt mode using GO_IRQ_STATE (CMD40) command.
- All cards in the Wait-IRQ-State are waiting for an internal interrupt trigger event. Once the event occurs,

the card starts to send its response to the host. This response is sent in the open drain mode.

- While waiting for the internal interrupt event, the cards are also waiting for a start bit on the command line. Upon reception of a start bit (generated by either another card or the host) the card will abort interrupt mode and switch to the *stand-by* state.
- Since the interrupt request message (response to command 40) is being sent in open drain mode, the host will receive a single valid response even when multiple cards are responding simultaneously (this can happen in the rare case of an interrupt event occurring in multiple cards synchronizing their start bit and preventing them from detecting each other's start bit).
- Regardless of winning or losing bus control during CMD40 response, all cards switch to the *stand-by* state (as opposed to CMD2).
- After the interrupt response was received by the host, the host returns to the standard data communication procedure.

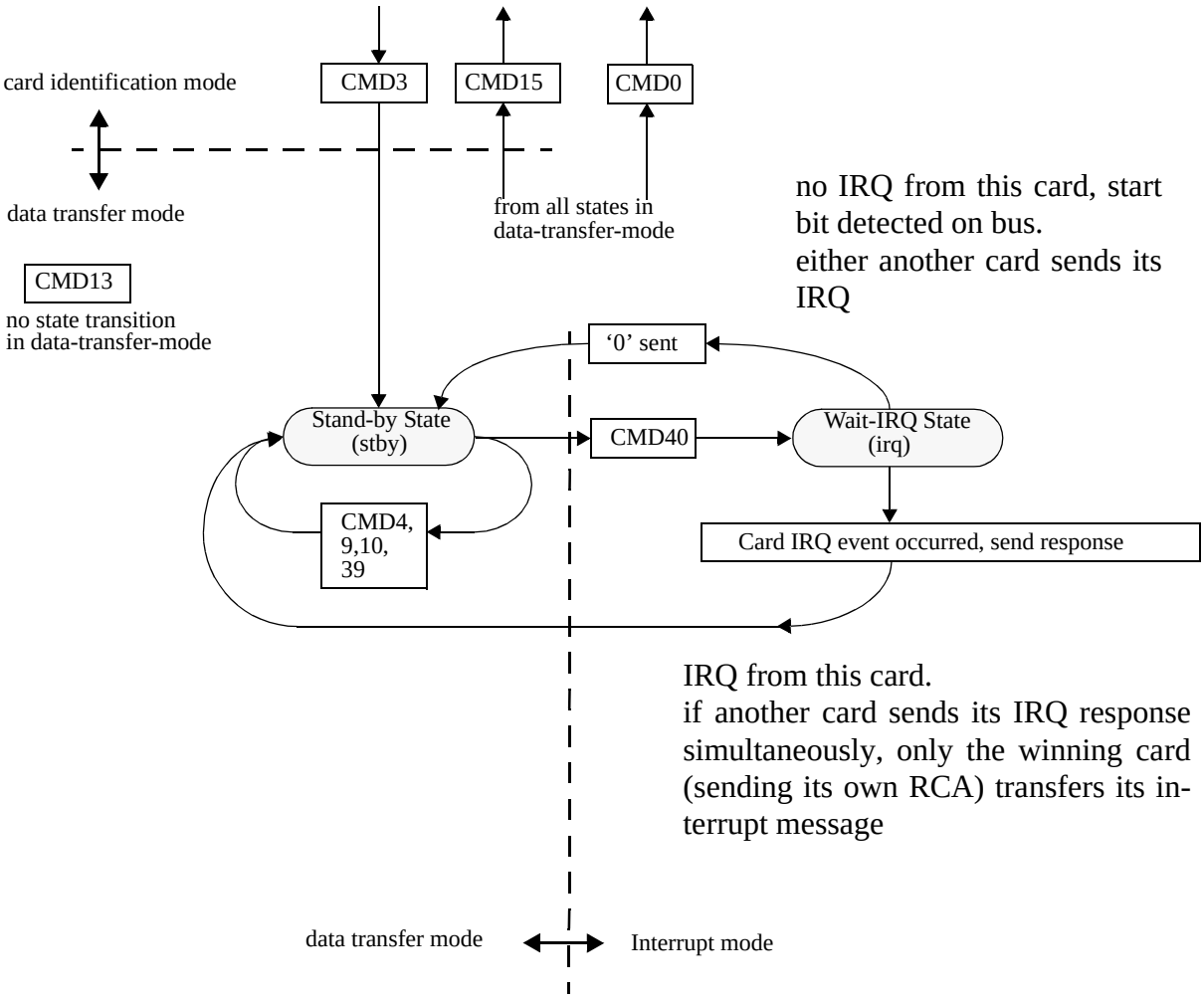


Figure 18: MultiMediaCard State Transition Diagram, Interrupt Mode.

- If the host wants to terminate the interrupt mode before an interrupt response is received, it can generate the CMD40 response by himself (with card bit = 0) using the reserved RCA address 0x000; This will bring all cards from Wait-IRQ-State back into the Stand-by-State. Now the host can resume the standard communication procedure.

4.4 Data Transfer Mode

When all cards are in *Stand-by State*, communication over the CMD and DAT lines will be performed in push-pull mode. Until the contents of all CSD registers is known by the host, the f_{PP} clock rate must remain at f_{OD} because some cards may have operating frequency restrictions (see Chapter 6.7). The host issues SEND_CSD (CMD9) to obtain the Card Specific Data (CSD register), e.g. block length, card storage capacity, maximum clock rate, etc.

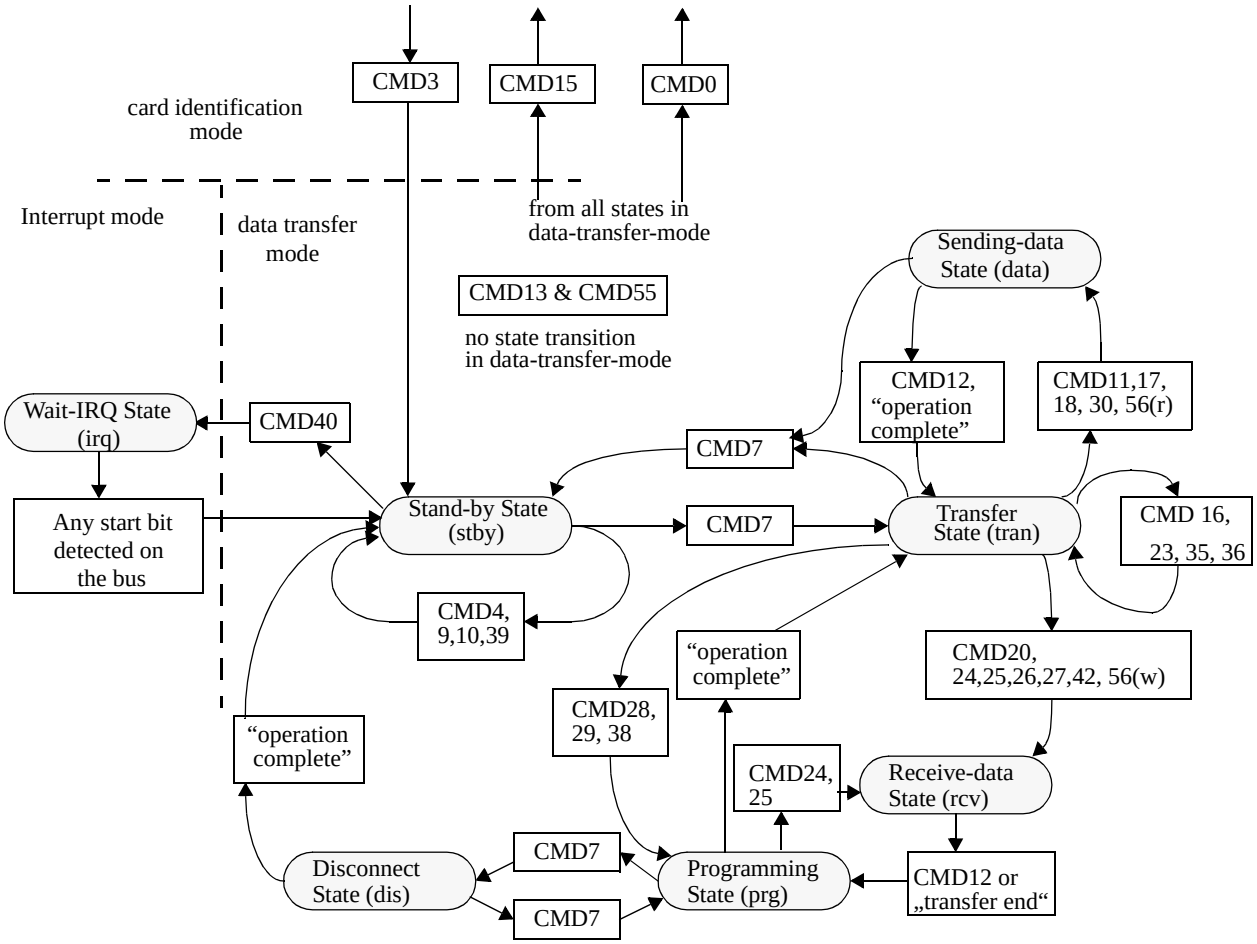


Figure 19: MultiMediaCard State Diagram (Data Transfer Mode)

The broadcast command SET_DSR (CMD4) configures the driver stages of all identified cards. It programs their DSR registers corresponding to the application bus layout (length) and the number of cards on the bus and the data transfer frequency. The clock rate is also switched from f_{OD} to f_{PP} at that point.

CMD7 is used to select one card and put it into the *Transfer State*. Only one card can be in the *Transfer State* at a given time. If a previously selected card is in the *Transfer State* its connection with the host is released and it will move back to the *Stand-by State*. When CMD7 is issued with the reserved relative card address "0x0000", all cards are put back to *Stand-by State*. This may be used before identifying new cards without resetting other already registered cards. Cards which already have an RCA do not respond to identification commands (CMD1, CMD2, CMD3, see Chapter 4.2.3) in this state.

All data communication in the Data Transfer Mode is point-to point between the host and the selected card (using addressed commands). All addressed commands get acknowledged by a response on the CMD line.

The relationship between the various data transfer modes is summarized below (see Figure 19):

- All data read commands can be aborted any time by the stop command (CMD12). The data transfer will terminate and the card will return to the *Transfer State.* The read commands are: stream read (CMD11), block read (CMD17), multiple block read (CMD18) and send write protect (CMD30).
- All data write commands can be aborted any time by the stop command (CMD12). The write commands must be stopped prior to deselecting the card by CMD7. The write commands are: stream write (CMD20), block write (CMD24 and CMD25), write CID (CMD26), and write CSD (CMD27).
- If a stream write operation is stopped prior to reaching the block boundary and partial blocks are allowed (as defined in the CSD), the part of the last block will be packed as a partial block and programmed. If partial blocks are not allowed the data will be discarded.
- As soon as the data transfer is completed, the card will exit the data write state and move either to the *Programming State* (transfer is successful) or *Transfer State* (transfer failed).
- If a block write operation is stopped and the block length and CRC of the last block are valid, the data will be programmed.
- If data transfer in stream write mode is stopped, not byte aligned, the bits of the incomplete byte are ignored and not programmed.
- The card may provide buffering for stream and block write. This means that the next block can be sent to the card while the previous is being programmed.
  If all write buffers are full, and as long as the card is in *Programming State* (see MultiMediaCard state diagram Figure 19), the DAT line will be kept low.
- There is no buffering option for write CSD, write CID, write protection and erase. This means that while the card is busy servicing any one of these commands, no other data transfer commands will be accepted. DAT line will be kept low as long as the card is busy and in the *Programming State*.
- Parameter set commands are *not* allowed while card is programming.
  Parameter set commands are: set block length (CMD16), and erase group selection (CMD35-36).
- Read commands are *not* allowed while card is programming.
- Moving another card from *Stand-by* to *Transfer State* (using CMD7) will not terminate a programming operation. The card will switch to the *Disconnect State* and will release the DAT line.
- A card can be reselected while in the *Disconnect State*, using CMD7. In this case the card will move to the *Programming State* and reactivate the busy indication.
- Resetting a card (using CMD0 or CMD15) will terminate any pending or active programming operation. This may destroy the data contents on the card. It is up to the host's responsibility to prevent this.

In the following format definitions, all upper case flags and parameters are defined in the CSD (Chapter 5.3), and the other status flags in the Card Status (Chapter 4.10).

4.4.1 Data Read

The DAT bus line level is high when no data is transmitted. A transmitted data block consists of a start bit (LOW), followed by a continuous data stream. The data stream contains the payload data (and error correction bits if an off-card ECC is used). The data stream ends with an end bit (HIGH) (see Figure 26-Figure 28). The data transmission is synchronous to the clock signal.

The payload for block oriented data transfer is protected by a CRC check sum (see Chapter 8.2).

• Stream Read

There is a stream oriented data transfer controlled by READ_DAT_UNTIL_STOP (CMD11). This command instructs the card to send its payload, starting at a specified address, until the host sends a STOP_TRANSMISSION command (CMD12). The stop command has an execution delay due to the serial command transmission. The data transfer stops after the end bit of the stop command.

If the end of the memory range is reached while sending data and no stop command has been sent yet by the host, the contents of the further transferred payload is undefined.

In order for the card to sustain data transfer in stream mode, the time it takes to transmit the data (defined by the bus clock rate) must be lower then the time it takes to read it out of the main memory field (defined by the card in the CSD register). Therefore, the maximum clock frequency for stream read operation is given by the following formula:

$$\text{Max Read Frequency} = \min\left(\text{TRAN\_SPEED}, \frac{8 \times 2^{\text{READ\_BL\_LEN}} - 100 \cdot \text{NSAC}}{\text{TAAC} \times \text{R2W\_FACTOR}}\right)$$

All the parameters are being defined in Chapter 5. If the host attempts to use a higher frequency, the card will not be able to sustain data transfer. If this happens, the card will set the UNDERRUN error bit in the status register, abort the transmission and wait in the data state for a stop command.

Since the timing constrains in the CSD register are typical (not maximum) values (refer to Chapter 4.6.2) using the above calculated frequency may still yield and occasional UNDERRUN error. In order to ensure that the card will not get into an UNDERRUN situation, the maximum read latency (defined as 10x the typical - refer to Chapter 4.6.2) should be used:

$$\text{No Underrun Read Frequency} = \min\left(\text{TRAN\_SPEED}, \frac{8 \times 2^{\text{READ\_BL\_LEN}} - 1000 \cdot \text{NSAC}}{10 \cdot \text{TAAC} \times \text{R2W\_FACTOR}}\right)$$

In general, the probability of an UNDERRUN error will decrease as the frequency decreases. The host application can control the trade-off between transfer speed (higher frequency) and error handling (lower frequency) by selecting the appropriate stream read frequency.

- **Block Read**

Block read is similar to stream read, except the basic unit of data transfer is a block whose maximum size is defined in the CSD (READ_BL_LEN). If READ_BL_PARTIAL is set, smaller blocks whose starting and ending address are entirely contained within one physical block (as defined by READ_BL_LEN) may also be transmitted. Unlike stream read, a CRC is appended to the end of each block ensuring data transfer integrity. CMD17 (READ_SINGLE_BLOCK) initiates a block read and after completing the transfer, the card returns to the *Transfer State*.

CMD18 (READ_MULTIPLE_BLOCK) starts a transfer of several consecutive blocks. Two types of multiple block read transactions are defined (the host can use either one at any time):

- Open-ended Multiple block read

The number of blocks for the read multiple block operation is not defined. The card will continuously transfer data blocks until a stop transmission command is received.

- Multiple block read with pre-defined block count

The card will transfer the requested number of data blocks, terminate the transaction and return to *transfer* state. Stop command is not required at the end of this type of multiple block read, unless terminated with an error. In order to start a multiple block read with pre-defined block count the host must use the SET_BLOCK_COUNT command (CMD23) immediately preceding the READ_MULTIPLE_BLOCK (CMD18) command. Otherwise the card will start an open-ended multiple block read which can be stopped using the STOP_TRANSMISION command.

The host can abort reading at any time, within a multiple block operation, regardless of the its type. Transaction abort is done by sending the stop transmission command.

If the card detects an error (e.g. out of range, address misalignment, internal error, etc.) during a multiple block read operation (both types) it will stop data transmission and remain in the *Data State*. The host must than abort the operation by sending the stop transmission command. The read error is reported in the response to the stop transmission command.

If the host sends a stop transmission command after the card transmits the last block of a multiple block operation with a pre-defined number of blocks, it will be responded to as an illegal command, since the card is no longer in *data* state.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed, the card shall detect a block misalignment error condition at the beginning of the first misaligned block (ADDRESS_ERROR error bit will is set in the status register).

4.4.2 Data Write

The data transfer format of write operation is similar to the data read. For block oriented write data transfer, the CRC check bits are added to each data block. The card performs a CRC parity check (see Chapter 8.2) for each received data block prior to the write operation. By this mechanism, writing of erroneously transferred data can be prevented.

- **Stream Write**

Stream write (CMD20) starts the data transfer from the host to the card beginning from the starting address until the host issues a stop command. If partial blocks are allowed (if CSD parameter WRITE_BL_PARTIAL is set) the data stream can start and stop at any address within the card address space, otherwise it shall start and stop only at block boundaries. Since the amount of data to be transferred is not determined in advance, CRC can not be used. If the end of the memory range is reached while sending data and no stop command has been sent by the host, all further transferred data is discarded.

In order for the card to sustain data transfer in stream mode, the time it takes to receive the data (defined by the bus clock rate) must be lower then the time it takes to program it into the main memory field (defined by the card in the CSD register). Therefore, the maximum clock frequency for stream write operation is given by the following formula:

$$\text{Max Write Frequency} = \min\left(\text{TRAN\_SPEED}, \frac{8 \times 2^{\text{WRITE\_BL\_LEN}} - 100 \cdot \text{NSAC}}{\text{TAAC} \times \text{R2W\_FACTOR}}\right)$$

All the parameters are defined in Chapter 5. If the host attempts to use a higher frequency, the card may not be able to process the data and will stop programming, set the OVERRUN error bit in the status register, and while ignoring all further data transfer, wait (in the *Receive-data-State*) for a stop command. The write operation shall also be aborted if the host tries to write over a write protected area. In this case, however, the card shall set the WP_VIOLATION bit.

Since the timing constrains in the CSD register are typical (not maximum) values (refer to Chapter 4.6.2) using the above calculated frequency may still yield and occasional OVERRUN error. In order to ensure that the card will not get into an OVERRUN situation, the maximum write latency (defined as 10x the typical -

refer to Chapter 4.6.2) should be used:

$$\text{Error-Free Write Frequency} = \min\left(\text{TRAN\_SPEED}, \frac{8 \times 2^{\text{WRITE\_BL\_LEN}} - 1000 \cdot \text{NSAC}}{10 \cdot \text{TAAC} \times \text{R2W\_FACTOR}}\right)$$

In general, the probability of an OVERRUN error will decrease as the frequency decreases. The host application can control the trade-off between transfer speed (higher frequency) and error handling (lower frequency) by selecting the appropriate stream write frequency.

- **Block Write**

During block write (CMD24 - 27) one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. A card supporting block write shall always be able to accept a block of data defined by WRITE_BL_LEN. If the CRC fails, the card shall indicate the failure on the DAT line (see below); the transferred data will be discarded and not written, and all further transmitted blocks (in multiple block write mode) will be ignored.

CMD25 (WRITE_MULTIPLE_BLOCK) starts a transfer of several consecutive blocks. Two types of multiple block write transactions, identical to the multiple block read, are defined (the host can use either one at any time):

- Open-ended Multiple block write

The number of blocks for the write multiple block operation is not defined. The card will continuously accept and program data blocks until a stop transmission command is received.

- Multiple block write with pre-defined block count

The card will accept the requested number of data blocks, terminate the transaction and return to *transfer* state. Stop command is not required at the end of this type of multiple block write, unless terminated with an error. In order to start a multiple block write with pre-defined block count the host must use the SET_BLOCK_COUNT command (CMD23) immediately preceding the WRITE_MULTIPLE_BLOCK (CMD25) command. Otherwise the card will start an open-ended multiple block write which can be stopped using the STOP_TRANSMISION command.

The host can abort writing at any time, within a multiple block operation, regardless of the its type. Transaction abort is done by sending the stop transmission command. If a multiple block write with pre-defined block count is aborted, the data in the remaining blocks is not defined.

If the card detects an error (e.g. write protect violation, out of range, address misalignment, internal error, etc.) during a multiple block write operation (both types) it will ignore any further incoming data blocks and remain in the *Receive State*. The host must than abort the operation by sending the stop transmission command. The write error is reported in the response to the stop transmission command.

If the host sends a stop transmission command after the card received the last data block of a multiple block write with a pre-defined number of blocks, it will be responded to as an illegal command, since the card is no longer in *rcv* state.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card shall detect the block misalignment error before the beginning of the first misaligned block. (ADDRESS_ERROR error bit is set in the status register).

Programming of the CID and CSD registers does not require a previous block length setting. The transferred

data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card will report an error and not change any register contents.

Some cards may require long and unpredictable times to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DAT line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) at any time, and the card will respond with its status. The status bit READY_FOR_DATA indicates whether the card can accept new data or whether the write process is still in progress). The host may deselect the card by issuing CMD7 (to select a different card) which will displace the card into the *Disconnect State* and release the DAT line without interrupting the write operation. When reselecting the card, it will reactivate busy indication by pulling DAT to low if programming is still in progress and the write buffer is unavailable.

4.4.3 Erase

MultiMediaCards, in addition to the implicit erase executed by the card as part of the write operation, provides a host explicit erase function. The erasable unit of the MultiMediaCard is the “Erase Group”; Erase group is measured in write blocks which are the basic writable units of the card. The size of the Erase Group is a card specific parameter and defined in the CSD.

The host can erase a contiguous range of Erase Groups. Starting the erase process is a three steps sequence. First the host defines the start address of the range using the ERASE_GROUP_START (CMD35) command, next it defines the last address of the range using the ERASE_GROUP_END (CMD36) command and finally it starts the erase process by issuing the ERASE (CMD38) command. The address field in the erase commands is an Erase Group address in byte units. The card will ignore all LSB’s below the Erase Group size, effectively rounding the address down to the Erase Group boundary.

If an erase command is received out of sequence, the card shall set the ERASE_SEQ_ERROR bit in the status register and reset the whole sequence.

If an out of sequence (neither of the erase commands, except SEND_STATUS) command received, the card shall set the ERASE_RESET status bit in the status register, reset the erase sequence and execute the last command.

If the erase range includes write protected blocks, they shall be left intact and only the non protected blocks shall be erased. The WP_ERASE_SKIP status bit in the status register shall be set.

As described above for block write, the card will indicate that an erase is in progress by holding DAT low. The actual erase time may be quite long, and the host may issue CMD7 to deselect the card.

4.4.4 Write Protect Management

In order to allow the host to protect data against erase or write, the MultiMediaCard shall support two levels of write protect commands:

- The entire card may be write protected by setting the permanent or temporary write protect bits in the CSD.
- Specific segments of the cards may be write protected. The segment size is defined in units of WP_GRP_SIZE erase groups as specified in the CSD. The SET_WRITE_PROT command sets the write protection of the addressed write-protect group, and the CLR_WRITE_PROT command clears the write protection of the addressed write-protect group.

The SEND_WRITE_PROT command is similar to a single block read command. The card shall send a data block containing 32 write protection bits (representing 32 write protect groups starting at the specified

address) followed by 16 CRC bits. The address field in the write protect commands is a group address in byte units. The card will ignore all LSB's below the group size.

4.4.5 Card Lock/Unlock Operation

The password protection feature enables the host to lock a card while providing a password, which later will be used for unlocking the card. The password and its size is kept in an 128 bit PWD and 8 bit PWD_LEN registers, respectively. These registers are non-volatile so that a power cycle will not erase them.

Locked cards respond to (and execute) all commands in the "basic" command class (class 0) and "lock card" command class. Thus the host is allowed to reset, initialize, select, query for status, etc., but not to access data on the card. If the password was previously set (the value of PWD_LEN is not '0') the card will be locked automatically after power on.

Similar to the existing CSD and CID register write commands the lock/unlock command is available in "transfer state" only. This means that it does not include an address argument and the card has to be selected before using it.

The card lock/unlock command has the structure and bus transaction type of a regular single block write command. The transferred data block includes all the required information of the command (password setting mode, PWD itself, card lock/unlock etc.). The following table describes the structure of the command data block.

Byte #	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved				ERASE	LOCK_ UNLOCK	CLR_ PWD	SET_ PWD
1	PWD_LEN							
2	Password data							
...								
PWD_LEN + 1								

Table 4: Lock Card Data Structure

- **ERASE:** '1' Defines Forced Erase Operation (all other bits shall be '0') and only the cmd byte is sent.
- **LOCK/UNLOCK**: '1' = Locks the card. '0' = Unlock the card (note that it is valid to set this bit together with SET_PWD but it is not allowed to set it together with CLR_PWD).
- **CLR_PWD**: '1' = Clears PWD.
- **SET_PWD**: '1' = Set new password to PWD
- **PWD_LEN**: Defines the following password length (in bytes). Valid password length are 1 to 16 bytes.
- **PWD:** The password (new or currently used depending on the command).

The data block size shall be defined by the host before it sends the card lock/unlock command. This will allow different password sizes.

The following paragraphs define the various lock/unlock command sequences:

- **Setting the Password**
  - Select a card (CMD7), if not previously selected already
  - Define the block length (CMD16), given by the 8bit card lock/unlock mode, the 8 bits password size (in bytes), and the number of bytes of the new password. In case that a password *replacement* is

done, then the block size shall consider that both passwords, the old and the new one, are sent with the command.
- Send Card Lock/Unlock command with the appropriate data block size on the data line including 16 bit CRC. The data block shall indicate the mode (SET_PWD), the length (PWD_LEN) and the password itself. In case that a password *replacement* is done, then the length value (PWD_LEN) shall include both passwords, the old and the new one, and the PWD field shall include the old password (currently used) followed by the new password.
- In case that the sent old password is not correct (not equal in size and content) then LOCK_UNLOCK_FAILED error bit will be set in the status register and the old password does not change. In case that PWD matches the sent old password then the given new password and its size will be saved in the PWD and PWD_LEN fields, respectively.

Note that the password length register (PWD_LEN) indicates if a password is currently set. When it equals '0' there is no password set. If the value of PWD_LEN is not equal to zero the card will lock itself after power up. It is possible to lock the card immediately in the current power session by setting the LOCK/UNLOCK bit (while setting the password) or sending additional command for card lock.

- **Reset the Password:**
  - Select a card (CMD7), if not previously selected already
  - Define the block length (CMD16), given by the 8 bit card lock/unlock mode, the 8 bit password size (in bytes), and the number of bytes of the currently used password.
  - Send the card lock/unlock command with the appropriate data block size on the data line including 16 bit CRC. The data block shall indicate the mode CLR_PWD, the length (PWD_LEN) and the password (PWD) itself (LOCK/UNLOCK bit is don't care). If the PWD and PWD_LEN content match the sent password and its size, then the content of the PWD register is cleared and PWD_LEN is set to 0. If the password is not correct then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

- **Locking a card:**
  - Select a card (CMD7), if not previously selected already
  - Define the block length (CMD16), given by the 8 bit card lock/unlock mode, the 8 bit password size (in bytes), and the number of bytes of the currently used password.
  - Send the card lock/unlock command with the appropriate data block size on the data line including 16 bit CRC. The data block shall indicate the mode LOCK, the length (PWD_LEN) and the password (PWD) itself.

If the PWD content equals to the sent password then the card will be locked and the card-locked status bit will be set in the status register. If the password is not correct then LOCK_UNLOCK_FAILED error bit will be set in the status register.

Note that it is possible to set the password and to lock the card in the same sequence. In such case the host shall perform all the required steps for setting the password (as described above) including the bit LOCK set while the new password command is sent.

If the password was previously set (PWD_LEN is not '0'), then the card will be locked automatically after power on reset.

An attempt to lock a locked card or to lock a card that does not have a password will fail and the LOCK_UNLOCK_FAILED error bit will be set in the status register.

- **Unlocking the card:**
  - Select a card (CMD7), if not previously selected already.
  - Define the block length (CMD16), given by the 8 bit card lock/unlock mode, the 8 bit password size (in bytes), and the number of bytes of the currently used password.

- Send the card lock/unlock command with the appropriate data block size on the data line including 16 bit CRC. The data block shall indicate the mode UNLOCK, the length (PWD_LEN) and the password (PWD) itself.

If the PWD content equals to the sent password then the card will be unlocked and the card-locked status bit will be cleared in the status register. If the password is not correct then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

Note that the unlocking is done only for the current power session. As long as the PWD is not cleared the card will be locked automatically on the next power up. The only way to unlock the card is by clearing the password.

An attempt to unlock an unlocked card will fail and LOCK_UNLOCK_FAILED error bit will be set in the status register.

- **Forcing Erase:**

In case that the user forgot the password (the PWD content) it is possible to erase all the card data content along with the PWD content. This operation is called *Forced Erase*.

- Select a card (CMD7), if not previously selected already.
- Define the block length (CMD16) to 1 byte (8bit card lock/unlock command). Send the card lock/unlock command with the appropriate data block of one byte on the data line including 16 bit CRC. The data block shall indicate the mode ERASE (the ERASE bit shall be the only bit set).

If the ERASE bit is not the only bit in the data field then the LOCK_UNLOCK_FAILED error bit will be set in the status register and the erase request is rejected.

If the command was accepted then ALL THE CARD CONTENT WILL BE ERASED including the PWD and PWD_LEN register content and the locked card will get unlocked. In addition, if the card is temporary write protected it will be unprotected (write enabled), the temporary-write-protect bit in the CSD and all Write-Protect-Groups will be cleared.

An attempt to force erase on an unlocked card will fail and LOCK_UNLOCK_FAILED error bit will be set in the status register.

If a force erase command is issued on a permanently-write-protect media the command will fail (card stays locked) and the LOCK_UNLOCK_FAILED error bit will be set in the status register.

4.4.6 Application specific commands

The MultiMediaCard system is designed to provide a standard interface for a variety applications types. In this environment it is anticipated that there will be a need for specific customers/applications features. To enable a common way of implementing these features, two types of generic commands are defined in the standard:

- **Application Specific Command – APP_CMD (CMD55)**

This command, when received by the card, will cause the card to interpret the following command as an application specific command, ACMD. The ACMD has the same structure as of regular MultiMediaCard standard commands and it may have the same CMD number. The card will recognize it as ACMD by the fact that it appears after APP_CMD.

The only effect of the APP_CMD is that if the command index of the, immediately, following command has an ACMD overloading, the non standard version will used. If, as an example, a card has a definition for ACMD13 but not for ACMD7 then, if received immediately after APP_CMD command, Command 13 will be interpreted as the non standard ACMD13 but, command 7 as the standard CMD7.

In order to use one of the manufacturer specific ACMD's the host will:

- Send APP_CMD. The response will have the APP_CMD bit (new status bit) set signaling to the host that ACMD is now expected.
- Send the required ACMD. The response will have the APP_CMD bit set, indicating that the accepted command was interpreted as ACMD. If a non-ACMD is sent then it will be respected by the card as normal MultiMediaCard command and the APP_CMD bit in the Card Status stays clear.

If a non valid command is sent (neither ACMD nor CMD) then it will be handled as a standard MultiMediaCard illegal command error.

From the MultiMediaCard protocol point of view the ACMD numbers will be defined by the manufacturers without any restrictions.

- **General Command - GEN_CMD (CMD56)**

The bus transaction of the GEN_CMD is the same as the single block read or write commands (CMD24 or CMD17). The difference is that the argument denotes the direction of the data transfer (rather than the address) and the data block is not a memory payload data but has a vendor specific format and meaning.

The card shall be selected ('*tran_state*') before sending CMD56. The data block size is the BLOCK_LEN that was defined with CMD16. The response to CMD56 will be R1b (card status + busy indication).

4.5 Clock Control

The MultiMediaCard bus clock signal can be used by the host to turn the cards into energy saving mode or to control the data flow (to avoid under-run or over-run conditions) on the bus. The host is allowed to lower the clock frequency or shut it down.

There are a few restrictions the host must follow:

- The bus frequency can be changed at any time (under the restrictions of maximum data transfer frequency, defined by the cards, and the identification frequency defined by the specification document).
- It is an obvious requirement that the clock must be running for the card to output data or response tokens. After the last MultiMediaCard bus transaction, the host is required, to provide **8 (eight)** clock cycles for the card to complete the operation before shutting down the clock. Following is a list of the various bus transactions:
- A command with no response. 8 clocks after the host command end bit.
- A command with response. 8 clocks after the card response end bit.
- A read data transaction. 8 clocks after the end bit of the last data block.
- A write data transaction. 8 clocks after the CRC status token.
- The host is allowed to shut down the clock of a "busy" card. The card will complete the programming operation regardless of the host clock. However, the host must provide a clock edge for the card to turn off its busy signal. Without a clock edge the card (unless previously disconnected by a deselect command - CMD7) will force the DAT line down, forever.

4.6 Error Conditions

4.6.1 CRC and Illegal Command

All commands are protected by CRC (cyclic redundancy check) bits. If the addressed card's CRC check fails, the card does not respond and the command is not executed. The card does not change its state, and COM_CRC_ERROR bit is set in the status register.

Similarly, if an illegal command has been received, a card shall not change its state, shall not response and shall set the ILLEGAL_COMMAND error bit in the status register. Only the non-erroneous state branches are shown in the state diagrams (see Figure 17 to Figure 19). Table 15 contains a complete state transition description.

There are different kinds of illegal commands:

- Commands which belong to classes not supported by the card (e.g. write commands in read only cards).
- Commands not allowed in the current state (e.g. CMD2 in Transfer State).
- Commands which are not defined (e.g. CMD6).

4.6.2 Read, Write and Erase Time-out Conditions

The times after which a time-out condition for read/write/erase operations occurs are (card independent) **10 times longer** than the typical access/program times for these operations given below. A card shall complete the command within this time period, or give up and return an error message. If the host does not get a response within the defined time-out it should assume the card is not going to respond anymore and try to recover (e.g. reset the card, power cycle, reject, etc.). The typical access and program times are defined as follows:

- **Read**

The read access time is defined as the sum of the two times given by the CSD parameters TAAC and NSAC (see Chapter 4.12). These card parameters define the typical delay between the end bit of the read command and the start bit of the data block. This number is card dependent and should be used by the host to calculate throughput and the maximal frequency for stream read.

- **Write**

The R2W_FACTOR field in the CSD is used to calculate the typical block program time obtained by multiplying the read access time by this factor. It applies to all write/erase commands (e.g. SET(CLEAR)_WRITE_PROTECT, PROGRAM_CSD(CID) and the block write commands). It should be used by the host to calculate throughput and the maximal frequency for stream write.

- **Erase**

The duration of an erase command will be (order of magnitude) the number of write blocks to be erased multiplied by the block write delay.

4.6.3 Read ahead in Stream and multiple block read operation

In stream, or multiple block, read operations, in order to avoid data under-run condition or improve read performance, the card may fetch data from the memory array, ahead of the host. In this case, when the host is reading the last addresses of the memory, the card attempts to fetch data beyond the last physical memory address and generates an OUT_OF_RANGE error.

Therefore, even if the host times the stop transmission command to stop the card immediately after the last byte of data was read, The card may already have generated the error, and it will show in the response to the stop transmission command. The host should ignore this error.

4.7 Commands

4.7.1 Command Types

There are four kinds of commands defined to control the MultiMediaCard:

- broadcast commands (bc), no response
- broadcast commands with response (bcr)
  response from all cards simultaneously
- addressed (point-to-point) commands (ac)
  no data transfer on DAT
- addressed (point-to-point) data transfer commands (adtc)
  data transfer on DAT

All commands and responses are sent over the CMD line of the MultiMediaCard bus. The command transmission always starts with the left bit of the bitstring corresponding to the command codeword.

4.7.2 Command Format

All commands have a fixed code length of 48 bits, needing a transmission time of 2.4 microSec @ 20 MHz

Bit position	**47**	**46**	**[45:40]**	**[39:8]**	**[7:1]**	**0**
Width (bits)	**1**	**1**	**6**	**32**	**7**	**1**
Value	**‘0’**	**‘1’**	**x**	**x**	**x**	**‘1’**
Description	**start bit**	**transmission bit**	**command index**	**argument**	**CRC7**	**end bit**

A command always starts with a start bit (always ‘0’), followed by the bit indicating the direction of transmission (host = ‘1’). The next 6 bits indicate the index of the command, this value being interpreted as a binary coded number (between 0 and 63). Some commands need an argument (e.g. an address), which is coded by 32 bits. A value denoted by ‘x’ in the table above indicates this variable is dependent on the command. All commands are protected by a CRC (see Chapter 8.2 for the definition of CRC7). Every command codeword is terminated by the end bit (always ‘1’). All commands and their arguments are listed in Table 6-Table 14.

4.7.3 Command Classes

The command set of the MultiMediaCard system is divided into several classes (See Table 5). Each class supports a subset of card functions.

Class 0 is mandatory and shall be supported by all cards. The other classes are either mandatory only for specific card types or optional (refer to chapter 10 for detailed description of supported command classes as a function of card type). By using different classes, several configurations can be chosen (e.g. a block writable card or a stream readable card). The supported Card Command Classes (CCC) are coded as a parameter in the

card specific data (CSD) register of each card, providing the host with information on how to access the card.

Card Command Class (CCC)	class description	Supported commands																			
		0	1	2	3	4	7	9	10	11	12	13	15	16	17	18	20	23	24	25	26	27
class 0	basic	+	+	+	+	+	+	+	+		+	+	+									
class 1	stream read									+												
class 2	block read													+	+	+		+				
class 3	stream write																+					
class 4	block write													+				+	+	+	+	+
class 5	erase																					
class 6	write protection																					
class 7	lock card																					
Class 8	application specific																					
class 9	I/O mode																					
class 10-11	reserved																					

Card Command Class (CCC)	class description	Supported commands										
		28	29	30	35	36	38	39	40	42	55	56
class 0	basic											
class 1	stream read											
class 2	block read											
class 3	stream write											
class 4	block write											
class 5	erase				+	+	+					
class 6	write protection	+	+	+								
class 7	lock card									+		
class 8	application specific										+	+
class 9	I/O mode							+	+			
class 10-11	reserved											

Table 5: Card command classes (CCCs)

4.7.4 Detailed Command Description

The following tables define in detail all MultiMediaCard bus commands. The responses R1-R5 are defined in

Chapter 4.9. The registers CID, CSD and DSR are described in Chapter 5.

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	asks all cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	asks all cards to send their CID numbers on the CMD line
CMD3	ac	[31:16] RCA [15:0] stuff bits	R1	SET_RELATIVE_ ADDR	assigns relative address to the card
CMD4	bc	[31:16] DSR [15:0] stuff bits	-	SET_DSR	programs the DSR of all cards
CMD5	reserved				
CMD6	reserved				
CMD7	ac	[31:16] RCA [15:0] stuff bits	R1b (only from the select ed card)	SELECT/DESELECT_ CARD	command toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases the card is selected by its own relative address and gets deselected by any other address; address 0 deselects all.
CMD8	reserved				
CMD9	ac	[31:16] RCA [15:0] stuff bits	R2	SEND_CSD	addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:16] RCA [15:0] stuff bits	R2	SEND_CID	addressed card sends its card identification (CID) on CMD the line.
CMD11	adtc	[31:0] data address[1]	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	R1 or R1b	STOP_ TRANSMISSION	Terminates a read/write stream/multiple block operation. When CMD12 is used to terminate a read transaction the card will respond with R1. When it is used to stop a write transaction the card will respond with R1b.
CMD13	ac	[31:16] RCA [15:0] stuff bits	R1	SEND_STATUS	addressed card sends its status register.
CMD14	reserved				
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 communication breakdowns.

Table 6: Basic commands & read stream commands (class 0 and class 1)

1)The addressing capability @ 8 bit address resolution is 2^{32} = 4 Gbyte

CMD INDEX	type	argument	resp	abbreviation	command description
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_ BLOCK	reads a block of the size selected by the SET_BLOCKLEN command.[1]
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_ BLOCK	continuously transfers data blocks from card to host until interrupted by a stop command or the requested number of data block transmitted
CMD19	reserved				

Table 7: Block oriented read commands (class 2)

1)The data transferred must not cross a physical block boundary unless READ_BLK_MISALIGN is set in the CSD

CMD INDEX	type	argument	resp	abbreviation	command description
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_ UNTIL_STOP	writes data stream from the host, starting at the given address, until a STOP_TRANSMISSION follows.
CMD21 ... CMD22	reserved				

Table 8: Stream write commands (class 3)

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.
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	writes a block of the size selected by the SET_BLOCKLEN command.[1]
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	continuously writes blocks of data until a STOP_TRANSMISSION follows or the requested number of block received.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	programming of the card identification register. This command shall be issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	programming of the programmable bits of the CSD.

Table 9: Block Oriented Write Commands (Class 4)

1)The data transferred must not cross a physical block boundary unless WRITE_BLK_MISALIGN is set in the CSD

CMD INDEX	type	argument	resp	abbreviation	command description
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	if the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	if the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	if the card provides write protection features, this command asks the card to send the status of the write protection bits. [1]
CMD31	reserved				

Table 10: Block oriented write protection commands (class 6)

1)32 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits are transferred in a payload format via the data line. The last (least significant) bit of the protection bits corresponds to the first addressed group. If the addresses of the last groups are outside the valid range, then the corresponding write protection bits shall be set to zero.

CMD INDEX	type	argument	resp	abbreviation	command description
CMD32 .... CMD34	Reserved. These command indexes cannot be used in order to maintain backwards compatibility with older versions of the MultiMediaCards				
CMD35	ac	[31:0] data address	R1	ERASE_GROUP_START	sets the address of the first erase group within a range to be selected for erase
CMD36	ac	[31:0] data address	R1	ERASE_GROUP_END	sets the address of the last erase group within a continuous range to be selected for erase
CMD37	Reserved. This command index cannot be used in order to maintain backwards compatibility with older versions of the MultiMediaCards				
CMD38	ac	[31:0] stuff bits	R1b	ERASE	erases all previously selected write blocks

Table 11: Erase Commands (Class 5)

CMD INDEX	type	argument	resp	abbreviation	command description
CMD39	ac	[31:16] RCA [15:15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	used to write and read 8 bit (register) data fields. The command addresses a card and a register and provides the data for writing if the write flag is set. The R4 response contains data read from the addressed register. This command accesses application dependent registers which are not defined in the MultiMediaCard standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode
CMD41	reserved				

Table 12: I/O Mode Commands (Class 9)

CMD INDEX	type	argument	resp	abbreviation	command description
CMD42	adtc	[31:0] stuff bits.	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/ unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43 ... CMD54	reserved				

Table 13: Lock Card (Class 7)

CMD INDEX	type	argument	resp	abbreviation	command description
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather than a standard command
CMD56	adtc	[31:1] stuff bits. [0]: RD/WR[1]	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block shall be set by the SET_BLOCK_LEN command.
CMD57 ... CMD59	reserved				
CMD60 -63	reserved for manufacturer				

Table 14: Application Specific Commands (Class 8)

1) RD/WR: “1” the host gets a block of data from the card.
“0” the host sends block of data to the card.

All future reserved commands shall have a codeword length of 48 bits, as well as their responses (if there are any).

4.8 Card State Transition Table

Table 15 defines the card state transitions in dependency of the received command.

	current state										
	idle	**ready**	**ident**	**stby**	**tran**	**data**	**rcv**	**prg**	**dis**	**ina**	**irq**
command	**changes to**										
class independent											
CRC error	-	-	-	-	-	-	-	-	-	-	stby
command not supported	-	-	-	-	-	-	-	-	-	-	stby
class 0											
CMD0	idle	idle	idle	idle	idle	idle	idle	idle	idle	-	stby
CMD1, card V_{DD} range compatible	ready	-	-	-	-	-	-	-	-	-	stby
CMD1, card is busy	idle	-	-	-	-	-	-	-	-	-	stby
CMD1, card V_{DD} range not compatible	ina	-	-	-	-	-	-	-	-	-	stby
CMD2, card wins bus	-	ident	-	-	-	-	-	-	-	-	stby
CMD2, card loses bus	-	ready	-	-	-	-	-	-	-	-	stby
CMD3	-	-	stby	-	-	-	-	-	-	-	stby
CMD4	-	-	-	stby	-	-	-	-	-	-	stby
CMD7, card is addressed	-	-	-	tran	-	-	-	-	prg	-	stby
CMD7, card is not addressed	-	-	-	-	stby	stby	-	dis	-	-	stby
CMD9	-	-	-	stby	-	-	-	-	-	-	stby
CMD10	-	-	-	stby	-	-	-	-	-	-	stby
CMD12	-	-	-	-	-	tran	prg	-	-	-	stby
CMD13	-	-	-	stby	tran	data	rcv	prg	dis	-	stby
CMD15	-	-	-	ina	ina	ina	ina	ina	ina	-	stby
class 1											
CMD11	-	-	-	-	data	-	-	-	-	-	stby
class 2											
CMD16	-	-	-	-	tran	-	-	-	-	-	stby
CMD17	-	-	-	-	data	-	-	-	-	-	stby
CMD18	-	-	-	-	data	-	-	-	-	-	stby
CMD23	-	-	-	-	tran	-	-	-	-	-	stby
class 3											
CMD20	-	-	-	-	rcv	-	-	-	-	-	stby
class 4											
CMD16	see class 2										
CMD23	see class 2										

Table 15: Card State Transition Table

	current state										
	idle	**ready**	**ident**	**stby**	**tran**	**data**	**rcv**	**prg**	**dis**	**ina**	**irq**
CMD24	-	-	-	-	rcv	-	-	rcv	-	-	stby
CMD25	-	-	-	-	rcv	-	-	rcv	-	-	stby
CMD26	-	-	-	-	rcv	-	-	-	-	-	stby
CMD27	-	-	-	-	rcv	-	-	-	-	-	stby
class 6											
CMD28	-	-	-	-	prg	-	-	-	-	-	stby
CMD29	-	-	-	-	prg	-	-	-	-	-	stby
CMD30	-	-	-	-	data	-	-	-	-	-	stby
class 5											
CMD35	-	-	-	-	tran	-	-	-	-	-	stby
CMD36	-	-	-	-	tran	-	-	-	-	-	stby
CMD38	-	-	-	-	prg	-	-	-	-	-	stby
class 7											
CMD42	-	-	-	-	rcv	-	-	-	-	-	stby
class 8											
CMD55	-	-	-	stby	tran	data	rcv	prg	dis	-	irq
CMD56; RD/WR = 0	-	-	-	-	rcv	-	-	-	-	-	stby
CMD56; RD/WR = 1	-	-	-	-	data	-	-	-	-	-	stby
class 9											
CMD39	-	-	-	stby	-	-	-	-	-	-	stby
CMD40	-	-	-	irq	-	-	-	-	-	-	stby
class 10 - 11											
CMD41; CMD43...CMD54, CMD57-CMD59	reserved										
CMD60...CMD63	reserved for manufacturer										

Table 15: Card State Transition Table

4.9 Responses

All responses are sent via the command line CMD. The response transmission always starts with the left bit of the bitstring corresponding to the response codeword. The code length depends on the response type.

A response always starts with a start bit (always ‘0’), followed by the bit indicating the direction of transmission (card = ‘0’). A value denoted by ‘x’ in the tables below indicates a variable entry. All responses except for the type R3 (see below) are protected by a CRC (see Chapter 8.2 for the definition of CRC7). Every command codeword is terminated by the end bit (always ‘1’).

There are five types of responses. Their formats are defined as follows:

- **R1** (normal response command): code length 48 bit. The bits 45:40 indicate the index of the command to be responded to, this value being interpreted as a binary coded number (between 0 and 63). The status of

the card is coded in 32 bits. The card status is described in Chapter 4.10

Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width (bits)	1	1	6	32	7	1
Value	'0'	'0'	x	x	x	'1'
Description	start bit	transmission bit	command index	card status	CRC7	end bit

- **R1b** is identical to R1 with an optional busy signal transmitted on the data line. The card may become busy after receiving these commands based on its state prior to the command reception. Refer to Chapter 4.12.3 for detailed description and timing diagrams.
- **R2** (CID, CSD register): code length 136 bits. The contents of the CID register are sent as a response to the commands CMD2 and CMD10. The contents of the CSD register are sent as a response to CMD9. Only the bits [127...1] of the CID and CSD are transferred, the reserved bit [0] of these registers is replaced by the end bit of the response.

Bit position	135	134	[133:128]	[127:1]	0
Width (bits)	1	1	6	127	1
Value	'0'	'0'	'111111'	x	'1'
Description	start bit	transmission bit	reserved	CID or CSD register incl. internal CRC7	end bit

- **R3** (OCR register): code length 48 bits. The contents of the OCR register is sent as a response to CMD1. The **level coding** is as follows: restricted voltage windows=LOW, card busy=LOW.

Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width (bits)	1	1	6	32	7	1
Value	'0'	'0'	'111111'	x	'1111111'	'1'
Description	start bit	transmission bit	reserved	OCR register	reserved	end bit

- **R4** (Fast I/O): code length 48 bits. The argument field contains the RCA of the addressed card, the register

address to be read out or written to, and its contents.

Bit position	47	46	[45:40]	[39:8] Argument field			[7:1]	0
Width (bits)	1	1	6	16	8	8	7	1
Value	'0'	'0'	'100111'	x	x	x	x	'1'
Description	start bit	transmission bit	CMD39	RCA [31:16]	register addr. [15:8]	read register contents [7:0]	CRC 7	end bit

- **R5** (Interrupt request): code length 48 bits. If the response is generated by the host, the RCA field in the argument shall be 0x0.

Bit position	47	46	[45:40]	[39:8] Argument field		[7:1]	0
Width (bits)	1	1	6	16	16	7	1
Value	'0'	'0'	'101000'	x	x	x	'1'
Description	start bit	trans-mission bit	CMD40	RCA [31:16] of winning card or of the host	[15:0] Not defined. May be used for IRQ data	CRC7	end bit

4.10 Card Status

The response format R1 contains a 32-bit field named *card status*. This field is intended to transmit the card's status information (which may be stored in a local status register) to the host. If not specified otherwise, the status entries are always related to the previous issued command. The semantics of this register is according to the CSD entry SPEC_VERS (see Chapter 5.3), indicating the version of the response formats (possibly used for later extensions).

Table 16 defines the different entries of the status. The type and clear condition fields in the table are abbreviated as follows:

- **Type:**
- E: Error bit.
- S: Status bit.
- R: Detected and set for the actual command response.
- X: Detected and set during command execution. The host must poll the card by issuing the status command in order to read these bits.[1]

- **Clear Condition:**
- A: According to the card current state.
- B: Always related to the previous command. Reception of a valid command will clear it (with a delay of

1. When an error bit is defined as "ERX", it means the bit could be set either for the actual command response (ER), or during command execution (EX). Therefore, if the host wants to properly detect this bit it will need to check in both the actual command response and after a SEND_STATUS command (CMD13).

one command).
- C: Clear by read

.

Bits	Identifier	Type	Value	Description	Clear Condition
31	OUT_OF_RANGE	E R	’0’= no error ’1’= error	The command’s argument was out of the allowed range for this card.	C
30	ADDRESS_ERROR	E R X	’0’= no error ’1’= error	A misaligned address which did not match the block length was used in the command.	C
29	BLOCK_LEN_ERROR	E R	’0’= no error ’1’= error	The transferred block length is not allowed for this card, or the number of transferred bytes does not match the block length.	C
28	ERASE_SEQ_ERROR	E R	’0’= no error ’1’= error	An error in the sequence of erase commands occurred.	C
27	ERASE_PARAM	E X	’0’= no error ’1’= error	An invalid selection of erase groups for erase occurred.	C
26	WP_VIOLATION	E R X	’0’= not protected ’1’= protected	Attempt to program a write protected block.	C
25	CARD_IS_LOCKED	S X	‘0’ = card unlocked ‘1’ = card locked	When set, signals that the card is locked by the host	A
24	LOCK_UNLOCK_FAILED	E R X	‘0’ = no error ‘1’ = error	Set when a sequence or password error has been detected in lock/unlock card command or if there was an attempt to access a locked card	C
23	COM_CRC_ERROR	E R	’0’= no error ’1’= error	The CRC check of the previous command failed.	B
22	ILLEGAL_COMMAND	E R	’0’= no error ’1’= error	Command not legal for the card state	B
21	CARD_ECC_FAILED	E X	’0’= success ’1’= failure	Card internal ECC was applied but failed to correct the data.	C
20	CC_ERROR	E R X	’0’= no error ’1’= error	Internal card controller error	C
19	ERROR	E R X	’0’= no error ’1’= error	A general or an unknown error occurred during the operation.	C
18	UNDERRUN	E X	’0’= no error ’1’= error	The card could not sustain data transfer in stream read mode	C
17	OVERRUN	E X	’0’= no error ’1’= error	The card could not sustain data programming in stream write mode	C

Table 16: Card Status

Bits	Identifier	Type	Value	Description	Clear Condition
16	CID/CSD_OVERWRITE	E R X	’0’= no error ’1’= error	can be either one of the following errors: - The CID register has been already written and can not be overwritten - The read only section of the CSD does not match the card content. - An attempt to reverse the copy (set as original) or permanent WP (unprotected) bits was made.	C
15	WP_ERASE_SKIP	S X	’0’= not protected ’1’= protected	Only partial address space was erased due to existing write protected blocks.	C
14	CARD_ECC_DISABLED	S X	’0’= enabled ’1’= disabled	The command has been executed without using the internal ECC.	A
13	ERASE_RESET	S R	’0’= cleared ’1’= set	An erase sequence was cleared before executing because an out of erase sequence command was received	C
12:9	CURRENT_STATE	S X	0 = idle 1 = ready 2 = ident 3 = stby 4 = tran 5 = data 6 = rcv 7 = prg 8 = dis 9-15 = reserved	The state of the card when receiving the command. If the command execution causes a state change, it will be visible to the host in the response to the next command. The four bits are interpreted as a binary coded number between 0 and 15.	B
8	READY_FOR_DATA	S X	’0’= not ready ’1’= ready	corresponds to buffer empty signalling on the bus	A
7:6					
5	APP_CMD	S R	‘0’ = Disabled ‘1’ = Enabled	The card will expect ACMD, or indication that the command has been interpreted as ACMD	C
4	reserved				
3, 2	reserved for application specific commands				
1, 0	reserved for manufacturer test mode				

Table 16: Card Status

The following table defines for each command responded by a R1 response the affected bits in the status field. An ‘x’ means the error/status bit may be affected by the respective command.

CMD #	Response Format 1 Status bit #																				
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12:9	8	5
3								x	x	x		x	x							x		
7								x	x	x		x	x						x	x		
11	x	x					x	x	x	x	x	x	x	x				x	x	x		
12								x	x	x	x	x	x	x	x				x	x		
13	x	x			x	x		x	x	x	x	x	x	x	x		x			x	x	
16			x				x	x	x	x		x	x						x	x		
17	x	x					x	x	x	x	x	x	x					x	x	x		
18	x	x					x	x	x	x	x	x	x					x	x	x		
20	x	x				x	x	x	x	x		x	x		x			x	x	x	x	
24	x	x				x	x	x	x	x		x	x					x	x	x	x	
23							x	x	x	x		x	x					x	x	x	x	
25	x	x				x	x	x	x	x		x	x					x	x	x	x	
26							x	x	x	x		x	x			x		x	x	x	x	
27							x	x	x	x		x	x					x	x	x	x	
28	x						x	x	x	x		x	x					x	x	x	x	
29	x						x	x	x	x		x	x					x	x	x	x	
30	x						x	x	x	x		x	x					x	x	x		
35	x			x	x		x	x	x	x		x	x						x	x		
36	x			x	x		x	x	x	x		x	x						x	x		
38				x			x	x	x	x		x	x				x		x	x	x	
42							x	x	x	x		x	x					x	x	x	x	
55							x	x	x	x		x	x						x	x		x
56							x	x	x	x		x	x					x	x	x	x	x

Table 17: Card Status Field/Command - Cross Reference

4.11 Memory Array Partitioning

The basic unit of data transfer to/from the MultiMediaCard is one byte. All data transfer operations which require a block size always define block lengths as integer multiples of bytes. Some special functions need other partition granularity.

For block oriented commands, the following definition is used:

- **Block**: is the unit which is related to the block oriented read and write commands. Its size is the number of bytes which will be transferred when one block command is sent by the host. The size of a block is either programmable or fixed. The information about allowed block sizes and the programmability is stored in the CSD.

For R/W cards, special erase and write protect commands are defined:

- The granularity of the erasable units is the **Erase Group:** The smallest number of consecutive write blocks which can be addressed for erase. The size of the Erase Group is card specific and stored in the CSD.
- The granularity of the Write Protected units is the **WP-Group:** The minimal unit which may be individually write protected. Its size is defined in units of erase groups. The size of a WP-group is card specific and

stored in the CSD.

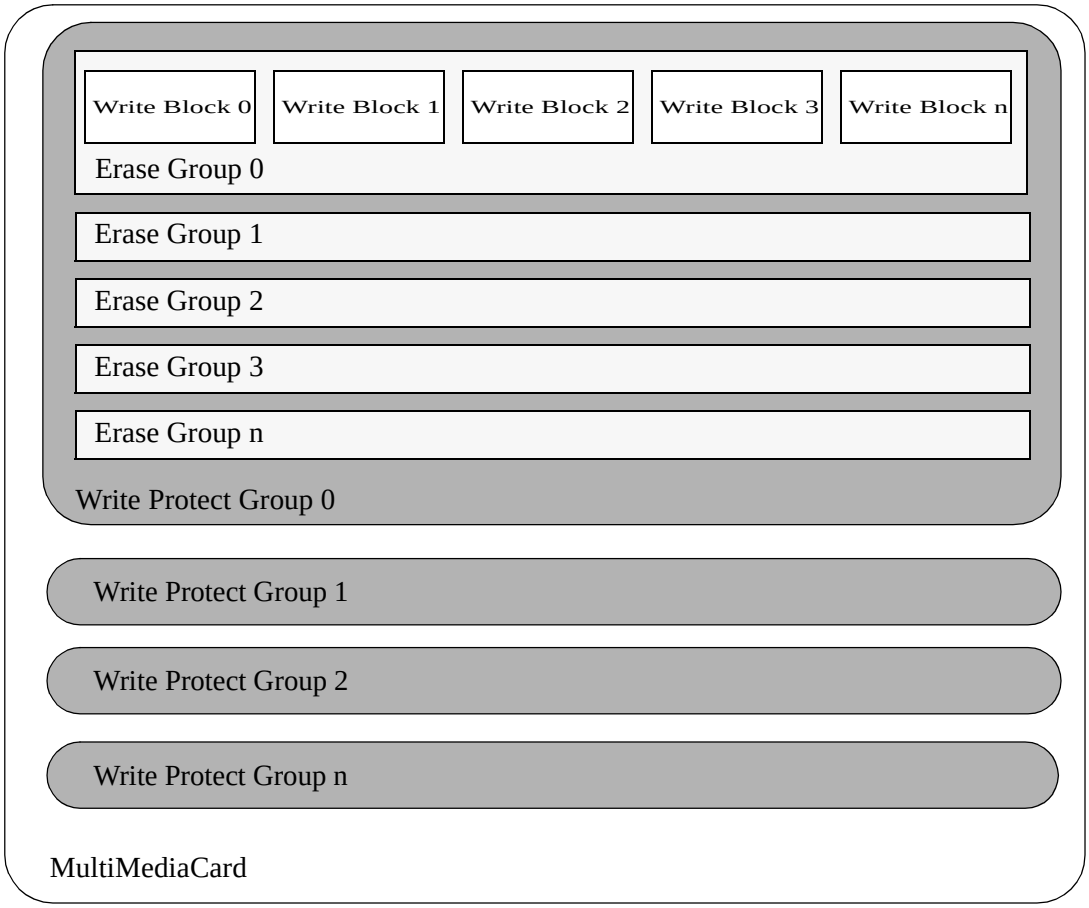


Figure 20: Memory Array Partitioning

4.12 Timings

All timing diagrams use the following schematics and abbreviations:

S	Start bit (= ‘0’)
T	Transmitter bit (Host = ‘1’, Card = ‘0’)
P	One-cycle pull-up (= ‘1’)
E	End bit (=1)
Z	High impedance state (-> = ‘1’)
D	Data bits
*	Repetition
CRC	Cyclic redundancy check bits (7 bits)
	Card active
	Host active

Table 18: Timing Diagram Symbols

The difference between the P-bit and Z-bit is that a P-bit is actively driven to HIGH by the card respectively host output driver, while Z-bit is driven to (respectively kept) HIGH by the pull-up resistors R_{CMD} respectively R_{DAT}. Actively-driven P-bits are less sensitive to noise.

All timing values are defined in Table 19.

4.12.1 Command and Response

Both host command and card response are clocked out with the rising edge of the host clock.

- **Card identification and card operation conditions timing**

The card identification (CMD2) and card operation conditions (CMD1) timing are processed in the open-drain mode. The card response to the host command starts after exactly N_{ID} clock cycles.

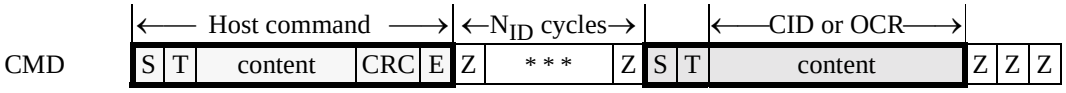


Figure 21: Identification Timing (Card Identification Mode)

- **Assign a card relative address**

The SET_RCA (CMD 3) is also processed in the open-drain mode. The minimum delay between the host command and card response is N_{CR} clock cycles.

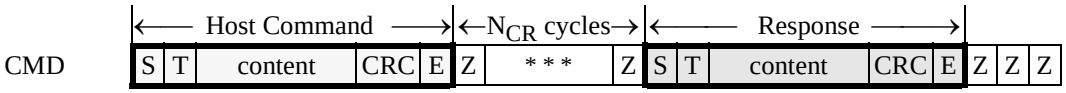


Figure 22: SET_RCA Timing (Card Identification Mode)

- **Data transfer mode.**

After a card receives its RCA it will switch to data transfer mode. In this mode the CMD line is driven with push-pull drivers. The command is followed by a period of two Z bits (allowing time for direction switching on the bus) and than by P bits pushed up by the responding card. This timing diagram is relevant for all responded host commands except CMD1,2,3:

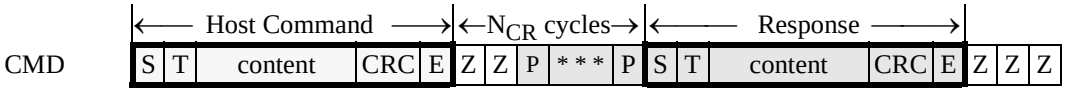


Figure 23: Command Response Timing (Data Transfer Mode)

- **Last Card Response - Next Host Command Timing**

After receiving the last card response, the host can start the next command transmission after at least N_{RC}

clock cycles. This timing is relevant for any host command.

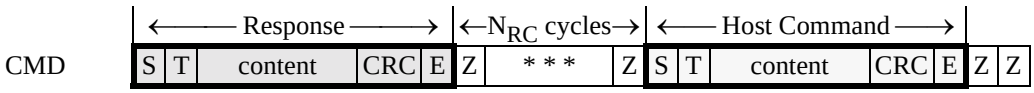


Figure 24: Timing Response End To Next Command Start (Data Transfer Mode)

- **Last Host Command - Next Host Command Timing**

After the last command has been sent, the host can continue sending the next command after at least N_{CC} clock periods.

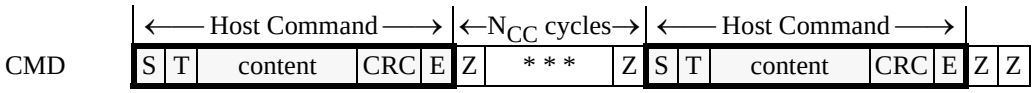


Figure 25: Timing Of Command Sequences (All Modes)

When the ALL_SEND_CID command is not responded by any card, the host waits NCR + 136 (the length of the R2 response) + NCC clock periods, at least, from the end of the command, before issuing a new command.

4.12.2 Data Read

- **Single Block Read**

The host selects one card for data read operation by CMD7, and sets the valid block length for block oriented data transfer by CMD16. The basic bus timing for a read operation is given in Figure 26. The sequence starts with a single block read command (CMD17) which specifies the start address in the argument field. The response is sent on the CMD line as usual.

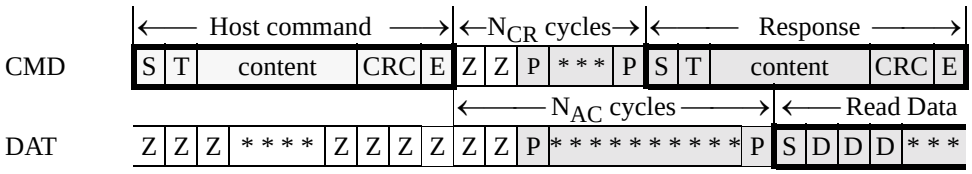


Figure 26: Single Block Read Timing

Data transmission from the card starts after the access time delay N_{AC} beginning from the end bit of the read command. After the last data bit, the CRC check bits are suffixed to allow the host to check for transmission errors.

- **Multiple Block Read**

In multiple block read mode, the card sends a continuous flow of data blocks following the initial host read command. The data flow is terminated by a stop transmission command (CMD12). Figure 27 describes the timing of the data blocks and Figure 28 the response to a stop command. The data transmission stops two clock cycles after the end bit of the stop command.

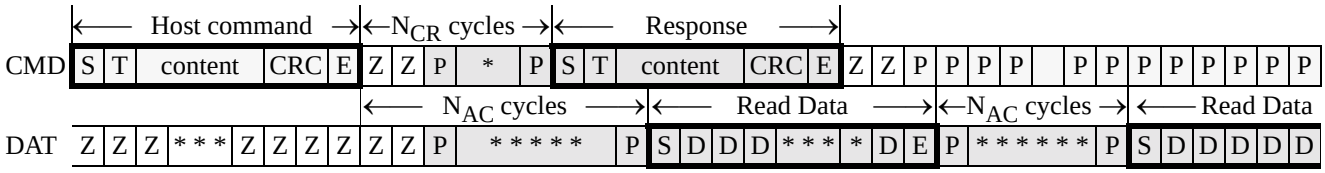


Figure 27: Multiple Block Read Timing

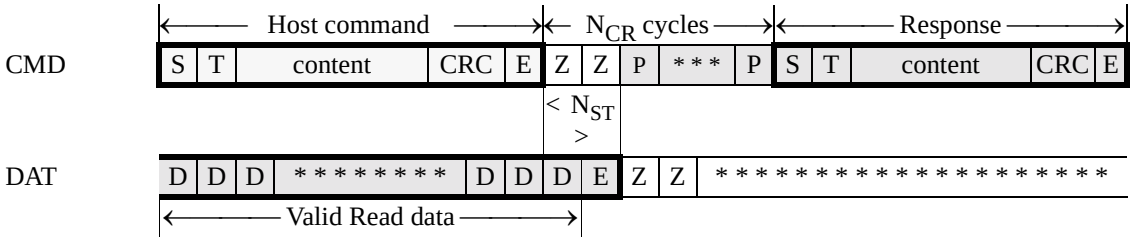


Figure 28: Stop Command Timing (CMD12, Data Transfer Mode)

- **Stream Read**

The data transfer starts N_{AC} clock cycles after the end bit of the host command. The bus transaction is identical to that of a read block command (see Figure 26). As the data transfer is not block oriented, the data stream does not include the CRC checksum. Consequently the host can not check for data validity. The data stream is terminated by a stop command. The corresponding bus transaction is identical to the stop command for the multiple read block (see Figure 28).

4.12.3 Data Write

- **Single Block Write**

The host selects one card for data write operation by CMD7.

The host sets the valid block length for block oriented data transfer (a stream write mode is also available) by CMD16.

The basic bus timing for a write operation is given in Figure 29. The sequence starts with a single block write command (CMD24) which determines (in the argument field) the start address. It is responded by the card on the CMD line as usual. The data transfer from the host starts N_{WR} clock cycles after the card response was received.

The data is suffixed with CRC check bits to allow the card to check it for transmission errors. The card sends back the CRC check result as a CRC status token on the data line. In the case of transmission error the card sends a negative CRC status (‘101’). In the case of non erroneous transmission the card sends a positive CRC status (‘010’) and starts the data programming procedure

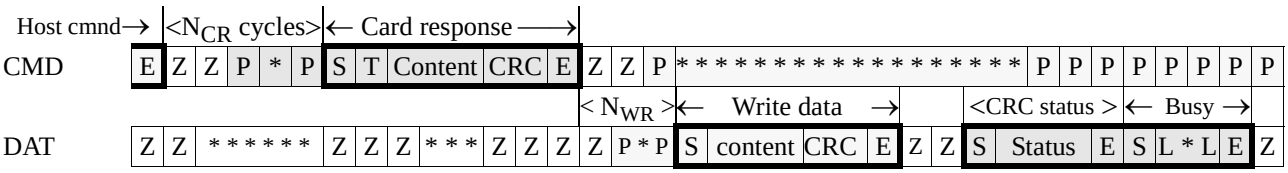


Figure 29: Block Write Command Timing

If the card does not have a free data receive buffer, the card indicates this condition by pulling down the data line to LOW. The card stops pulling down the data line as soon as at least one receive buffer for the defined data transfer block length becomes free. This signalling does not give any information about the data write status which must be polled by the host.

- **Multiple Block Write**

In multiple block write mode, the card expects continuous flow of data blocks following the initial host write command. The data flow is terminated by a stop transmission command (CMD12). Figure 30 describes the timing of the data blocks with and without card busy signal.

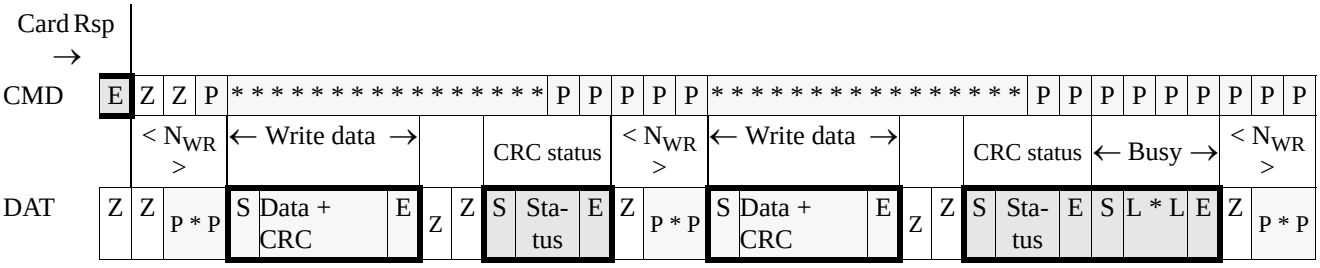


Figure 30: Multiple Block Write Timing

The stop transmission command works similar as in the read mode. Figure 31 to Figure 34 describe the timing of the stop command in different card states.

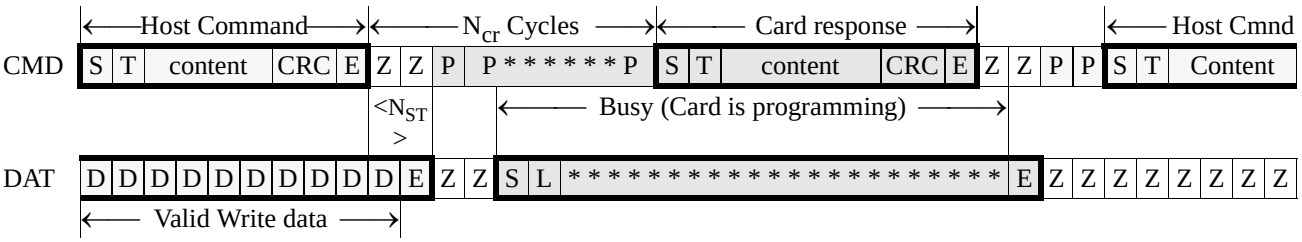
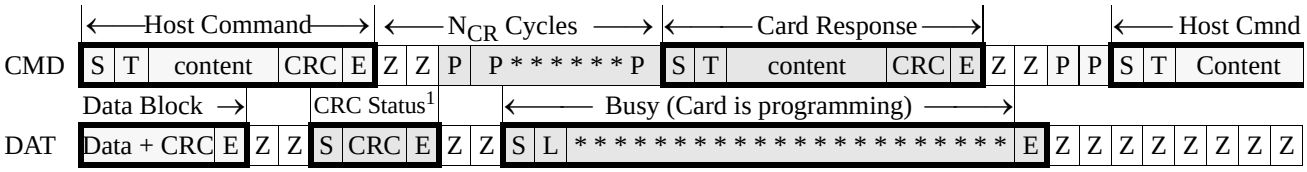


Figure 31: Stop Transmission During Data Transfer From The Host

The card will treat a data block as successfully received and ready for programming only if the CRC data of the block was validated and the CRC status token sent back to the host. Figure 32 is an example of an interrupted (by a host stop command) attempt to transmit the CRC status block. The sequence is identical to all other stop transmission examples. The end bit of the host command is followed, on the data line, with one

more data bit, end bit and two Z clock for switching the bus direction. The received data block, in this case is considered incomplete and will not be programmed.



1) The card CRC status response is interrupted by the host.

Figure 32: Stop Transmission During CRC Status Transfer From The Card

All previous examples dealt with the scenario of the host stopping the data transmission during an active data transfer. The following two diagrams describe a scenario of receiving the stop transmission between data blocks. In the first example the card is busy programming the last block while in the second the card is idle. However, there are still unprogrammed data blocks in the input buffers. These blocks are being programmed as soon as the stop transmission command is received and the card activates the busy signal.

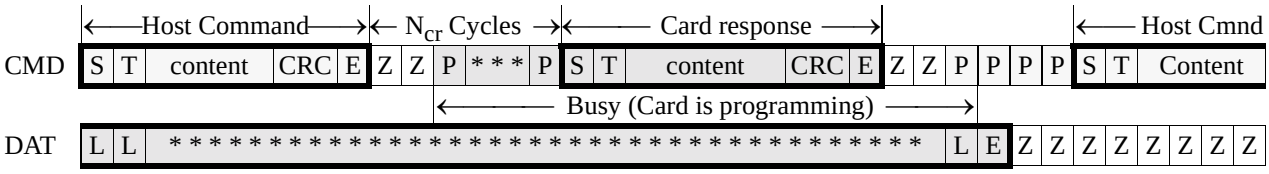


Figure 33: Stop Transmission After Last Data Block. Card Is Busy Programming.

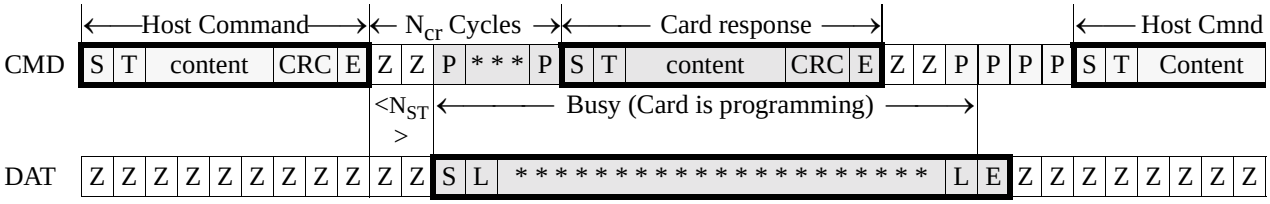


Figure 34: Stop Transmission After Last Data Block. Card Becomes Busy.

- **Stream Write**

The data transfer starts N_{WR} clock cycles after the card response to the sequential write command was received. The bus transaction is identical to that of a write block command (see Figure 29). As the data transfer is not block oriented, the data stream does not include the CRC checksum. Consequently the host can not receive any CRC status information from the card. The data stream is terminated by a stop command. The bus transaction is identical to the write block option when a data block is interrupted by the stop command (see Figure 31).

- **Erase, Set and Clear Write Protect Timing**

The host must first select the erase groups to be erased using the erase start and end command (CMD35,

CMD36). The erase command (CMD38), once issued, will erase all selected erase groups. Similarly, set and clear write protect commands start a programming operation as well. The card will signal “busy” (by pulling the DAT line low) for the duration of the erase or programming operation. The bus transaction timings are identical to the variation of the stop transmission described in Figure 34.

- **Reselecting a busy card**

When a busy card which is currently in the dis state is reselected it will reinstate its busy signaling on the data line. The timing diagram for this command / response / busy transaction is given in Figure 34.

4.12.4 Timing Values

Table 19 defines all timing values.

	Min	Max	Unit
N_{CR}	2	64	clock cycles
N_{ID}	5	5	clock cycles
N_{AC}	2	10 * (TAAC *F_{OP} + 100 * NSAC)[1]	clock cycles
N_{RC}	8	-	clock cycles
N_{CC}	8	-	clock cycles
N_{WR}	2	-	clock cycles
N_{ST}	2	2	clock cycles

Table 19: Timing values

1) F_{OP} is the MMC clock frequency the host is using for the read operation.
Following is a calculation example:
CSD value for TAAC is 0x26; This is equal to 1.5mSec;
CSD value for NSAC is 0;
The host frequency F_{OP} is 10MHRZ
N_{AC} = 10 * (1.5e-3*10e6+0) = 150,000 clock cycles

5 Card Registers

Within the card interface five registers are defined: OCR, CID, CSD, RCA and DSR. These can be accessed only by corresponding commands (see Chapter 4.7). The OCR, CID and CSD registers carry the card/content specific information, while the RCA and DSR registers are configuration registers storing actual configuration parameters.

5.1 OCR Register

The 32-bit operation conditions register stores the V_{DD} voltage profile of the card. In addition, this register includes a status information bit. This status bit is set if the card power up procedure has been finished. The OCR register shall be implemented by all cards.

OCR bit	VDD voltage window	High Voltage MultimediaCard	Low Voltage MultiMediaCard
[6:0]	Reserved	0000000b	0000000b
[7]	1.65 - 1.95	0b	1b
[14:8]	2.0-2.6	0000000b	0000000b
[23:15]	2.7-3.6	111111111b	111111111b
[30:24]	Reserved	0000000b	0000000b
[31]	card power up status bit (busy)[1]		

Table 20: OCR Register Definition

1)This bit is set to LOW if the card has not finished the power up routine

The supported voltage range is coded as shown in Table 20, for High Voltage and Low Voltage MultiMediaCards. As long as the card is busy, the corresponding bit (31) is set to LOW, the ‘wired-and’ operation, described in Section 4.2.2 yields LOW, if at least one card is still busy.

5.2 CID Register

The Card IDentification (CID) register is 128 bits wide. It contains the card identification information used during the card identification phase (MultiMediaCard protocol). Every individual flash or I/O card shall have an unique identification number. Every type of MultiMediaCard ROM cards (defined by content) shall have an unique identification number.

The structure of the CID register is defined in the following paragraphs:

Name	Field	Width	CID-slice
Manufacturer ID	MID	8	[127:120]
OEM/Application ID	OID	16	[119:104]
Product name	PNM	48	[103:56]
Product revision	PRV	8	[55:48]
Product serial number	PSN	32	[47:16]

Table 21: The CID fields

Name	Field	Width	CID-slice
Manufacturing date	MDT	8	[15:8]
CRC7 checksum	CRC	7	[7:1]
not used, always ’1’	-	1	[0:0]

Table 21: The CID fields

- **MID**

An 8 bit binary number that identifies the card manufacturer. The MID number is controlled, defined and allocated to a MultiMediaCard manufacturer by the MMCA. This procedure is established to ensure uniqueness of the CID register.

- **OID**

A 16 bit binary number that identifies the card OEM and/or the card contents (when used as a distribution media either on ROM or FLASH cards). The OID number is controlled, defined and allocated to a MultiMediaCard manufacturer by the MMCA. This procedure is established to ensure uniqueness of the CID register.

- **PNM**

The product name is a string, 6 ASCII characters long.

- **PRV**

The product revision is composed of two Binary Coded Decimal (BCD) digits, four bits each, representing an “n.m” revision number. The “n” is the most significant nibble and “m” is the least significant nibble.

As an example, the PRV binary value field for product revision “6.2” will be: 0110 0010

- **PSN**

A 32 bits unsigned binary integer.

- **MDT**

The manufacturing date is composed of two hexadecimal digits, four bits each, representing a two digits date code m/y;

The “m” field, most significant nibble, is the month code. 1 = January.

The “y” field, least significant nibble, is the year code. 0 = 1997.

As an example, the binary value of the MDT field for production date “April 2000” will be: 0100 0011

- **CRC**

CRC7 checksum (7 bits). This is the checksum of the CID contents computed according to Chapter 8.

5.3 CSD Register

The Card-Specific Data register provides information on how to access the card contents. The CSD defines the data format, error correction type, maximum data access time, data transfer speed, whether the DSR register can be used etc. The programmable part of the register (entries marked by W or E, see below) can be changed by CMD27. The type of the entries in the table below is coded as follows: R = readable, W = writable

once, E = erasable (multiple writable).

Name	Field	Width	Cell Type	CSD-slice
CSD structure	CSD_STRUCTURE	2	R	[127:126]
System specification version	SPEC_VERS	4	R	[125:122]
reserved	-	2	R	[121:120]
data read access-time-1	TAAC	8	R	[119:112]
data read access-time-2 in CLK cycles (NSAC*100)	NSAC	8	R	[111:104]
max. data transfer rate	TRAN_SPEED	8	R	[103:96]
card command classes	CCC	12	R	[95:84]
max. read data block length	READ_BL_LEN	4	R	[83:80]
partial blocks for read allowed	READ_BL_PARTIAL	1	R	[79:79]
write block misalignment	WRITE_BLK_MISALIGN	1	R	[78:78]
read block misalignment	READ_BLK_MISALIGN	1	R	[77:77]
DSR implemented	DSR_IMP	1	R	[76:76]
reserved	-	2	R	[75:74]
device size	C_SIZE	12	R	[73:62]
max. read current @V_{DD} min	VDD_R_CURR_MIN	3	R	[61:59]
max. read current @V_{DD} max	VDD_R_CURR_MAX	3	R	[58:56]
max. write current @V_{DD} min	VDD_W_CURR_MIN	3	R	[55:53]
max. write current @V_{DD} max	VDD_W_CURR_MAX	3	R	[52:50]
device size multiplier	C_SIZE_MULT	3	R	[49:47]
erase group size	ERASE_GRP_SIZE	5	R	[46:42]
erase group size multiplier	ERASE_GRP_MULT	5	R	[41:37]
write protect group size	WP_GRP_SIZE	5	R	[36:32]
write protect group enable	WP_GRP_ENABLE	1	R	[31:31]
manufacturer default ECC	DEFAULT_ECC	2	R	[30:29]
write speed factor	R2W_FACTOR	3	R	[28:26]
max. write data block length	WRITE_BL_LEN	4	R	[25:22]
partial blocks for write allowed	WRITE_BL_PARTIAL	1	R	[21:21]
Reserved	-	4	R	[20:17]
Content protection application	CONTENT_PROT_APP	1	R	[16:16]
File format group	FILE_FORMAT_GRP	1	R/W	[15:15]
copy flag (OTP)	COPY	1	R/W	[14:14]
permanent write protection	PERM_WRITE_PROTECT	1	R/W	[13:13]
temporary write protection	TMP_WRITE_PROTECT	1	R/W/E	[12:12]
File format	FILE_FORMAT	2	R/W	[11:10]
ECC code	ECC	2	R/W/E	[9:8]
CRC	CRC	7	R/W/E	[7:1]
not used, always ’1’	-	1	-	[0:0]

Table 22: The CSD fields

The following sections describe the CSD fields and the relevant data types. If not explicitly defined otherwise, all bit strings are interpreted as binary coded numbers starting with the left bit first.

- **CSD_STRUCTURE**

describes the version of the CSD structure.

CSD_STRUCTURE	CSD structure version	Valid for System Specification Version
0	CSD version No. 1.0	Version 1.0 - 1.2
1	CSD version No. 1.1	Version 1.4 - 2.2
2	CSD version No. 1.2	Version 3.1 - 3.2
3	Reserved	Reserved

Table 23: CSD register structure

- **SPEC_VERS**

Defines the MultiMediaCard System Specification version supported by the card.

SPEC_VERS	System Specification Version Number
0	Version 1.0-1.2
1	Version 1.4
2	Version 2.0 - 2.2
3	Version 3.1 - 3.2 - 3.3
4 - 15	Reserved

Table 24: System Specification Version

- **TAAC**

Defines the asynchronous part of the data access time.

TAAC bit position	code
2:0	time unit 0=1ns, 1=10ns, 2=100ns, 3=1µs, 4=10µs, 5=100µs, 6=1ms, 7=10ms
6:3	time value 0=reserved, 1=1.0, 2=1.2, 3=1.3, 4=1.5, 5=2.0, 6=2.5, 7=3.0, 8=3.5, 9=4.0, A=4.5, B=5.0, C=5.5, D=6.0, E=7.0, F=8.0
7	reserved

Table 25: TAAC access time definition

- **NSAC**

Defines the typical case for the clock dependent factor of the data access time. The unit for NSAC is 100 clock cycles. Therefore, the maximal value for the clock dependent part of the data access time is 25.5k clock cycles.

The total access time N_{AC} as expressed in Table 19 is calculated based on TAAC and NSAC. It has to be com-

puted by the host for the actual clock rate. The read access time should be interpreted as a typical delay for the first data bit of a data block or stream.

- **TRAN_SPEED**

The following table defines the maximum data transfer rate TRAN_SPEED:

TRAN_SPEED bit	code
2:0	transfer rate unit 0=100kbit/s, 1=1Mbit/s, 2=10Mbit/s, 3=100Mbit/s, 4... 7=reserved
6:3	time value 0=reserved, 1=1.0, 2=1.2, 3=1.3, 4=1.5, 5=2.0, 6=2.5, 7=3.0, 8=3.5, 9=4.0, A=4.5, B=5.0, C=5.5, D=6.0, E=7.0, F=8.0
7	reserved

Table 26: Maximum data transfer rate definition

- **CCC**

The MultiMediaCard command set is divided into subsets (command classes). The card command class register CCC defines which command classes are supported by this card. A value of '1' in a CCC bit means that the corresponding command class is supported. For command class definition refer to Table 5.

CCC bit	Supported card command class
0	class 0
1	class 1
.....	
11	class 11

Table 27: Supported card command classes

- **READ_BL_LEN**

The data block length is computed as $2^{READ\_BL\_LEN}$. The block length might therefore be in the range 1, 2,4...2048 bytes (see Chapter 4.11 for details):

READ_BL_LEN	Block length	Remark
0	2^0 = 1 Byte	
1	2^1 = 2 Bytes	
.....		
11	2^{11} = 2048 Bytes	
12-15	reserved	

Table 28: Data block length

- **READ_BL_PARTIAL**

Defines whether partial block sizes can be used in block read commands.

READ_BL_PARTIAL=0 means that only the READ_BL_LEN block size can be used for block oriented data transfers.

READ_BL_PARTIAL=1 means that smaller blocks can be used as well. The minimum block size will be equal to minimum addressable unit (one byte)

- **WRITE_BLK_MISALIGN**

Defines if the data block to be written by one command can be spread over more than one physical block of the memory device. The size of the memory block is defined in WRITE_BL_LEN.

WRITE_BLK_MISALIGN=0 signals that crossing physical block boundaries is invalid.

WRITE_BLK_MISALIGN=1 signals that crossing physical block boundaries is allowed.

- **READ_BLK_MISALIGN**

Defines if the data block to be read by one command can be spread over more than one physical block of the memory device. The size of the memory block is defined in READ_BL_LEN.

READ_BLK_MISALIGN=0 signals that crossing physical block boundaries is invalid.

READ_BLK_MISALIGN=1 signals that crossing physical block boundaries is allowed.

- **DSR_IMP**

Defines if the configurable driver stage is integrated on the card. If set, a driver stage register (DSR) must be implemented also (see Chapter 5.5).

DSR_IMP	DSR type
0	no DSR implemented
1	DSR implemented

Table 29: DSR implementation code table

- **C_SIZE**

This parameter is used to compute the card capacity. The memory capacity of the card is computed from the entries C_SIZE, C_SIZE_MULT and READ_BL_LEN as follows:

memory capacity = BLOCKNR * BLOCK_LEN

where

BLOCKNR = (C_SIZE+1) * MULT

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

BLOCK_LEN = $2^{READ\_BL\_LEN}$, (READ_BL_LEN < 12)

Therefore, the maximal capacity which can be coded is 4096*512*2048 = 4 GBytes. Example: A 4 MByte card with BLOCK_LEN = 512 can be coded by C_SIZE_MULT = 0 and C_SIZE = 2047.

- **VDD_R_CURR_MIN, VDD_W_CURR_MIN**

The maximum values for read and write currents at the minimal power supply V_{DD} are coded as follows:

VDD_R_CURR_MIN **VDD_W_CURR_MIN**	**code for current consumption @ V_{DD}**
2:0	0=0.5mA; 1=1mA; 2=5mA; 3=10mA; 4=25mA; 5=35mA; 6=60mA; 7=100mA

Table 30: $V_{DD,min}$ current consumption

- **VDD_R_CURR_MAX, VDD_W_CURR_MAX**

The maximum values for read and write currents at the maximal power supply V_{DD} are coded as follows:

VDD_R_CURR_MAX **VDD_W_CURR_MAX**	**code for current consumption @ V_{DD}**
2:0	0=1mA; 1=5mA; 2=10mA; 3=25mA; 4=35mA; 5=45mA; 6=80mA; 7=200mA

Table 31: $V_{DD,max}$ current consumption

- **C_SIZE_MULT**

This parameter is used for coding a factor MULT for computing the total device size (see 'C_SIZE'). The factor MULT is defined as $2^{C\_SIZE\_MULT+2}$.

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	$2^9 = 512$	

Table 32: Multiply factor for the device size

- **ERASE_GRP_SIZE**

The contents of this register is a 5 bit binary coded value, used to calculate the size of the erasable unit of the card. The size of the erase unit (also referred to as erase group) is determined by the ERASE_GRP_SIZE and the ERASE_GRP_MULT entries of the CSD, using the following equation:

size of erasable unit = (ERASE_GRP_SIZE + 1) * (ERASE_GRP_MULT + 1)

This size is given as minimum number of write blocks that can be erased in a single erase command.

- **ERASE_GRP_MULT**

A 5 bit binary coded value used for calculating the size of the erasable unit of the card. See ERASE_GRP_SIZE section for detailed description.

- **WP_GRP_SIZE**

The size of a write protected group. The contents of this register is a 5 bit binary coded value, defining the number of erase groups which can be write protected. The actual size is computed by increasing this number by one. A value of zero means 1 erase group, 31 means 32 erase groups.

- **WP_GRP_ENABLE**

A value of ‘0’ means no group write protection possible.

- **DEFAULT_ECC**

Set by the card manufacturer. It defines the ECC code which is recommended for use. The field definition is the same as for the ECC field described later.

- **R2W_FACTOR**

Defines the typical block program time as a multiple of the read access time. The following table defines the field format.

R2W_FACTOR	Multiples of read access time
0	1
1	2 (write half as fast as read)
2	4
3	8
4	16
5	32
6	64
7	128

Table 33: R2W_FACTOR

- **WRITE_BL_LEN**

Block length for write operations. See READ_BL_LEN for field coding.

- **WRITE_BL_PARTIAL**

Defines whether partial block sizes can be used in block write commands.

WRITE_BL_PARTIAL=’0’ means that only the WRITE_BL_LEN block size can be used for block oriented data write.

WRITE_BL_PARTIAL=’1’ means that smaller blocks can be used as well. The minimum block size is one byte.

- **FILE_FORMAT_GRP**

Indicates the selected group of file formats. This field is read-only for ROM. The usage of this field is shown in Table 34 (see FILE_FORMAT).

- **COPY**

Defines if the contents is original (= ‘0’) or has been copied (=’1’). The COPY bit for OTP and MTP devices, sold to end consumers, is set to ‘1’ which identifies the card contents as a copy. The COPY bit is an one time programmable bit.

- **PERM_WRITE_PROTECT**

Permanently protects the whole card content against overwriting or erasing (all write and erase commands for

this card are permanently disabled). The default value is ‘0’, i.e. not permanently write protected.

- **TMP_WRITE_PROTECT**

Temporarily protects the whole card content from being overwritten or erased (all write and erase commands for this card are temporarily disabled). This bit can be set and reset. The default value is ‘0’, i.e. not write protected.

- **CONTENT_PROT_APP**

This field in the CSD indicates whether the content protection application is supported. MultiMediaCards which implement the content protection application will have this bit set to ‘1’;

- **FILE_FORMAT**

Indicates the file format on the card. This field is read-only for ROM. The following formats are defined:

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

Table 34: File formats

A more detailed description is given in Chapter 11.

- **ECC**

Defines the ECC code that was used for storing data on the card. This field is used by the host (or application) to decode the user data. The following table defines the field format.:

ECC	ECC type	Maximum number of correctable bits per block
0	none (default)	none
1	BCH (542,512)	3
2-3	reserved	-

Table 35: ECC type

- **CRC**

The CRC field carries the check sum for the CSD contents. It is computed according to Chapter 8.2. The checksum has to be recalculated by the host for any CSD modification. The default corresponds to the initial CSD contents.

The following table lists the correspondence between the CSD entries and the command classes. A '+' entry indicates that the CSD field affects the commands of the related command class.

	Command classes									
CSD Field	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
CSD_STRUCTURE	+	+	+	+	+	+	+	+	+	+
SPEC_VERS	+	+	+	+	+	+	+	+	+	+
TAAC		+	+	+	+	+	+	+	+	
NSAC		+	+	+	+	+	+	+	+	
TRAN_SPEED		+	+	+	+					
CCC	+	+	+	+	+	+	+	+	+	+
READ_BL_LEN			+							
READ_BL_PARTIAL			+							
WRITE_BLK_MISALIGN					+					
READ_BLK_MISALIGN			+							
DSR_IMP	+	+	+	+	+	+	+	+	+	+
C_SIZE_MANT		+	+	+	+	+	+	+	+	
C_SIZE_EXP		+	+	+	+	+	+	+	+	
VDD_R_CURR_MIN		+	+							
VDD_R_CURR_MAX		+	+							
VDD_W_CURR_MIN				+	+	+	+	+	+	
VDD_W_CURR_MAX				+	+	+	+	+	+	
ERASE_GRP_SIZE						+	+	+	+	
WP_GRP_SIZE							+	+	+	
WP_GRP_ENABLE							+	+	+	
DEFAULT_ECC		+	+	+	+	+	+	+	+	
R2W_FACTOR				+	+	+	+	+	+	
WRITE_BL_LEN				+	+	+	+	+	+	
WRITE_BL_PARTIAL				+	+	+	+	+	+	
FILE_FORMAT_GRP										
COPY	+	+	+	+	+	+	+	+	+	+
PERM_WRITE_PROTECT	+	+	+	+	+	+	+	+	+	+
TMP_WRITE_PROTECT	+	+	+	+	+	+	+	+	+	+
FILE_FORMAT										
ECC		+	+	+	+	+	+	+	+	
CRC	+	+	+	+	+	+	+	+	+	+

Table 36: Cross reference of CSD fields vs. command classes

5.4 RCA Register

The writable 16-bit relative card address register carries the card address assigned by the host during the card identification. This address is used for the addressed host-card communication after the card identification procedure. The default value of the RCA register is 0x0001. The value 0x0000 is reserved to set all cards into

the *Stand-by State* with CMD7.

5.5 DSR Register

The 16-bit driver stage register is described in detail in Chapter 6.4. It can be optionally used to improve the bus performance for extended operating conditions (depending on parameters like bus length, transfer rate or number of cards). The CSD register carries the information about the DSR register usage. The default value of the DSR register is 0x404.

6 The MultiMediaCard Bus

The MultiMediaCard bus has three communication lines and three supply lines:

- CMD: Command is a bidirectional signal. The host and card drivers are operating in two modes, open drain and push pull.
- DAT: Data is a bidirectional signal. Host and card drivers are operating in push pull mode
- CLK: Clock is a host to card signal. CLK operates in push pull mode
- V_{DD}: V_{DD} is the power supply line for all cards.
- V_{SS1}, V_{SS2} are two ground lines.

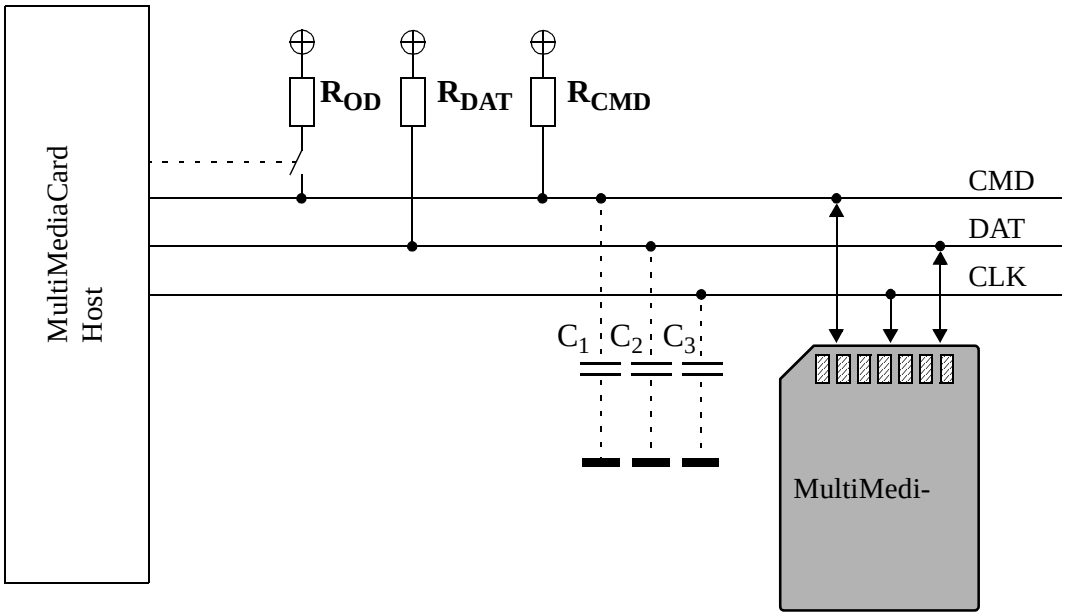


Figure 35: Bus Circuitry Diagram

The R_{OD} is switched on and off by the host synchronously to the open-drain and push-pull mode transitions. The host does not have to have open drain drivers, but must recognize this mode to switch on the R_{OD}. R_{DAT} and R_{CMD} are pull-up resistors protecting the CMD and the DAT line against bus floating when no card is inserted or when all card drivers are in a high-impedance mode.

A constant current source can replace the R_{OD} by achieving a better performance (constant slopes for the signal rising and falling edges). If the host does not allow the switchable R_{OD} implementation, a fixed R_{CMD} can be used (the minimum value is defined in the Chapter 6.5). Consequently the maximum operating frequency in the open drain mode has to be reduced if the used R_{CMD} value is higher than the minimal one given in Chapter 6.5.

6.1 Hot Insertion and Removal

To guarantee the proper sequence of card pin connection during hot insertion, the use of either a special hot-insertion capable card connector or an auto-detect loop on the host side (or some similar mechanism) is mandatory (see Chapter 9).

No card shall be damaged by inserting or removing a card into the MultiMediaCard bus even when the power (V_{DD}) is up. Data transfer operations are protected by CRC codes, therefore any bit changes induced by card

insertion and removal can be detected by the MultiMediaCard bus master.

The inserted card must be properly reset also when CLK carries a clock frequency f_{PP}. Each card shall have power protection to prevent card (and host) damage. Data transfer failures induced by removal/insertion are detected by the bus master. They must be corrected by the application, which may repeat the issued command.

6.2 Power Protection

Cards shall be inserted/removed into/from the bus without damage. If one of the supply pins (V_{DD} or V_{SS}) is not connected properly, then the current is drawn through a data line to supply the card.

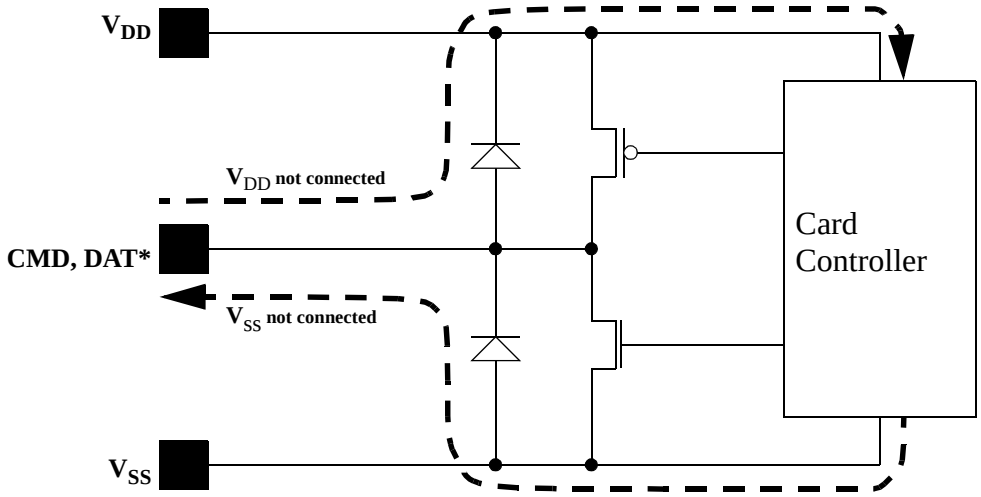


Figure 36: Improper Power Supply

Every card's output also shall be able to withstand shortcuts to either supply.

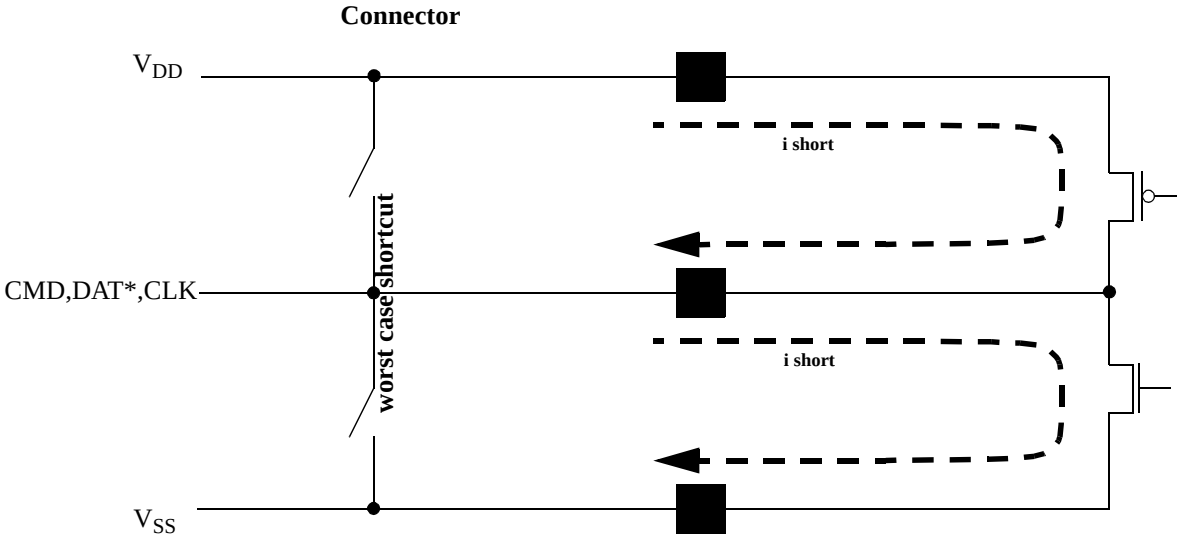


Figure 37: Shortcut Protection

If hot insertion feature is implemented in the host, than the host has to withstand a shortcut between V_{DD} and

V_{SS} without damage.

6.3 Power Up

The power up of the MultiMediaCard bus is handled locally in each card and in the bus master.

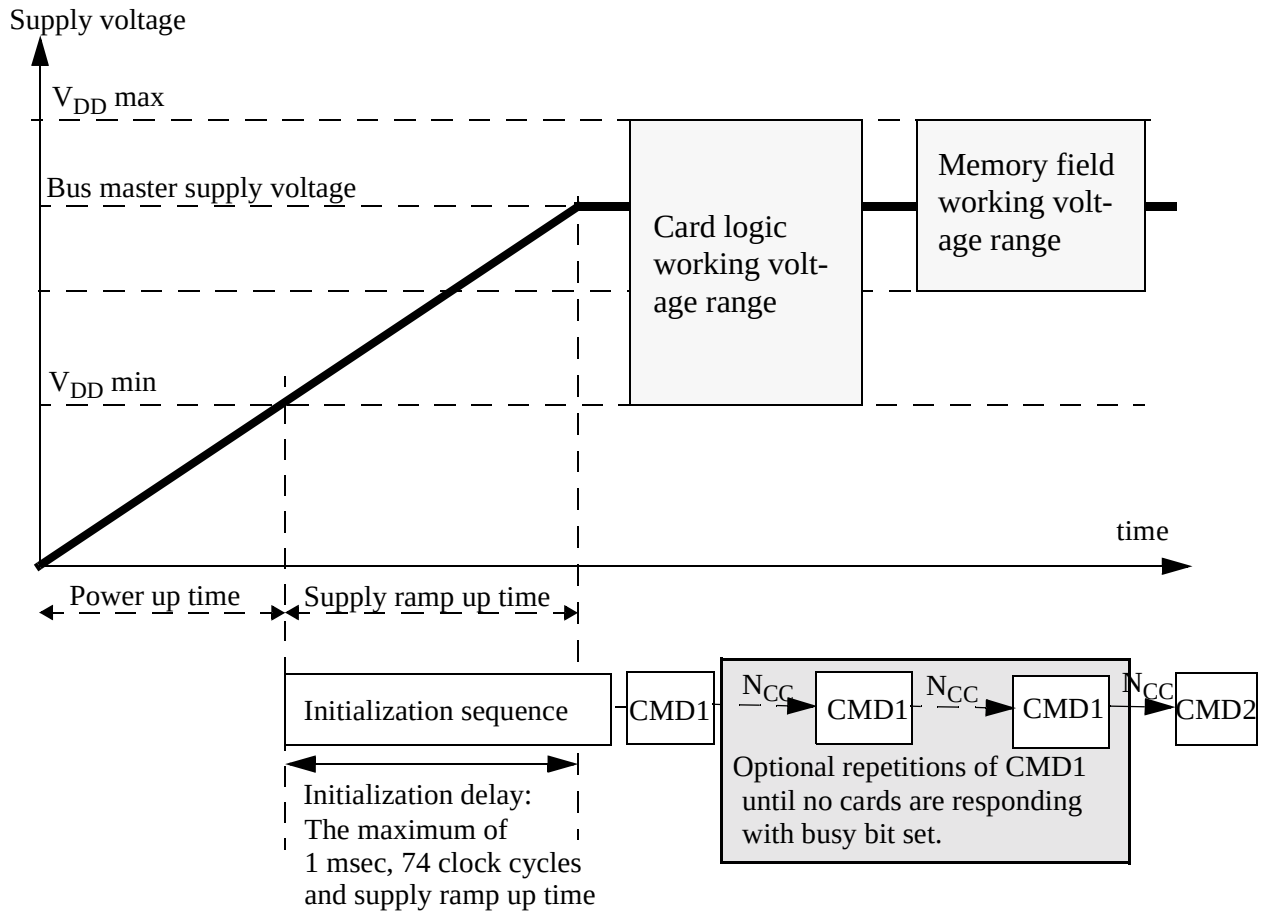


Figure 38: Power-up Diagram

- After power up (including hot insertion, i.e. inserting a card when the bus is operating) the card enters the *idle state*. During this state the card ignores all bus transactions until CMD1 is received.
- CMD1 is a special synchronization command used to negotiate the operation voltage range and to poll the cards until they are out of their power-up sequence. Besides the operation voltage profile of the cards, the response to CMD1 contains a busy flag, indicating that the card is still working on its power-up procedure and is not ready for identification. This bit informs the host that at least one card is not ready. The host has to wait (and continue to poll the cards) until this bit is cleared.
- Getting individual cards, as well as the whole MultiMediaCard system, out of *idle state* is up to the responsibility of the bus master. Since the power up time and the supply ramp up time depend on application parameters such as the maximum number of cards, the bus length and the power supply unit, the host must ensure that the power is built up to the operating level (the same level which will be specified in CMD1) before CMD1 is transmitted.
- After power up the host starts the clock and sends the initializing sequence on the CMD line. This

sequence is a contiguous stream of logical ‘1’s. The sequence length is the maximum of 1msec, 74 clocks or the supply-ramp-up-time; The additional 10 clocks (over the 64 clocks after what the card should be ready for communication) is provided to eliminate power-up synchronization problems.

- Every bus master has to implement CMD1. The CMD1 implementation is mandatory for all MultiMediaCards.

6.4 Programmable Card Output Driver

The bus capacitance of each line of the MultiMediaCard bus is the sum of the bus master capacitance, the bus capacitance itself and the capacitance of each inserted card. The sum of host and bus capacitance are fixed for one application, but may vary between different applications.The card load may vary in one application with each of the inserted cards.

In the following, programmable card output drivers for the push pull mode are described as an optional method for ensuring the defined maximum clock rate independently of the topology and of the number of inserted cards.

Both data and command driver stages in the push-pull mode have programmable peak current driving capabilities and programmable rise and fall times. The driver stage register (DSR) consists of two 8-bit latches. The contents of the latches is calculated from the required transfer speed of the interface and the bus load.

The CMD and DAT bus drivers consist of a predriver stage and a complementary driver transistor (Figure 39). The predriver stage output rise and fall time is set with the DSR1 register and determines the speed of the driver stage. The complementary driver transistor size is set with the DSR2 register and determines the current driving capabilities of the driver stage and also influences the peak current consumption of the bus driver. The proper combination of both allows the optimum bus performance.

Table 37 defines the DSR register contents:

DSR1	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
$t_{switch\text{-}on\ max}$	reserved				5ns	20ns	100ns	500ns
$t_{switch\text{-}on\ min}$					2ns	10ns	50ns	200ns

DSR2	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
$i_{peak\ min}$	reserved				100mA	20mA	5mA	1mA
$i_{peak\ max}$					200mA	50mA	10mA	2mA
$t_{rise\ typ}$					5ns	20ns	100ns	500ns

Table 37: DSR register contents

The time in DSR1 specifies the switch-on time of the output driver transistors. At the external interface, it is measurable as a delay time between the clock and driver stage output signal (e.g. for testing).

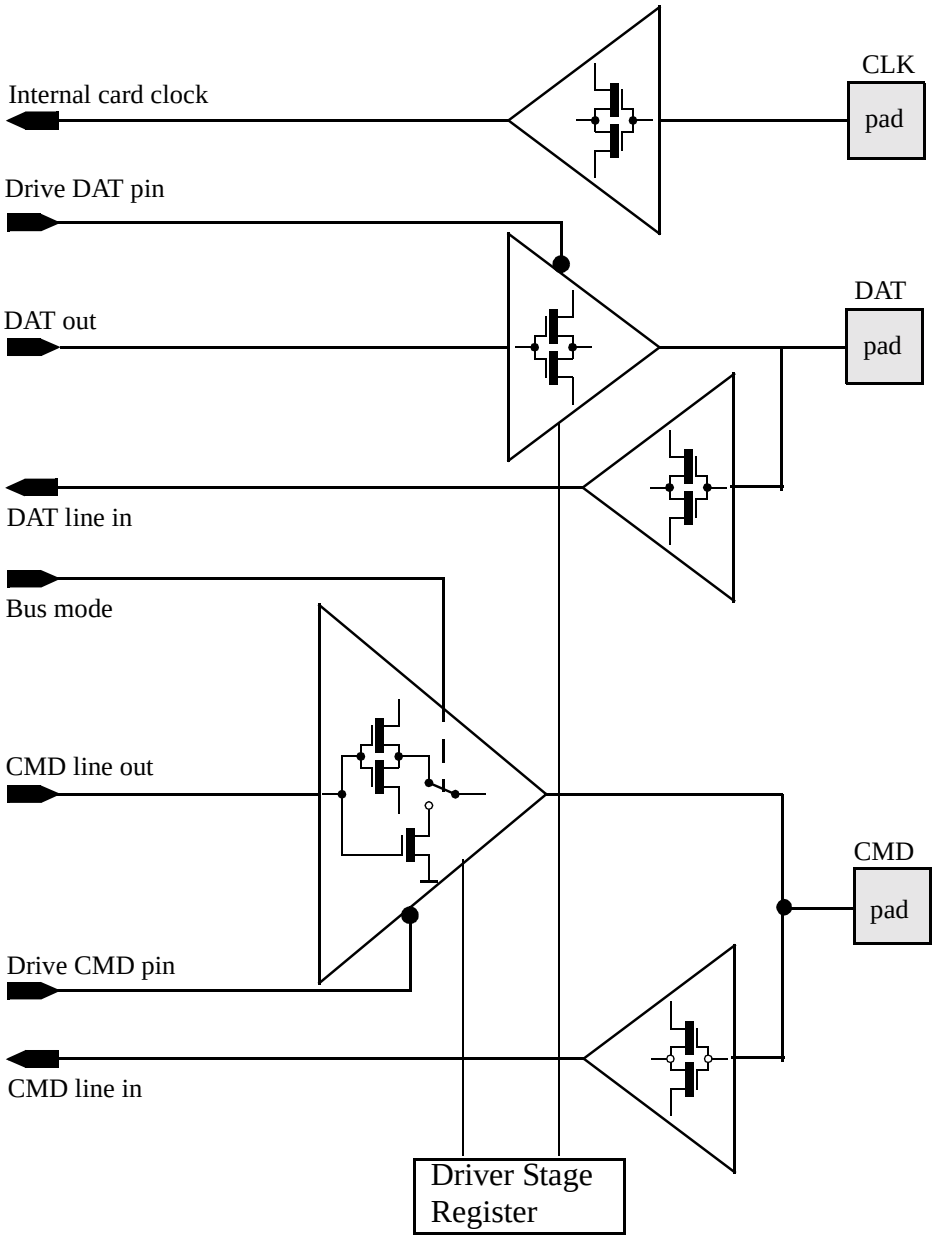


Figure 39: MultiMediaCard Bus Driver

All data is valid for the specified operating range (voltage, temperature). Any combination of DSR1 and DSR2 bits may be programmed. DSR1 has to be programmed for the required clock frequency, where

$f_{clock} = (2\ t_{switch\text{-}on\ max})^{-1}$.

The DSR2 register must be programmed with the required driver size. Hints for the proper driver stage selection are part of future application notes (see Appendix).

6.5 Bus Operating Conditions

- **General**

Parameter	Symbol	Min	Max.	Unit	Remark
Peak voltage on all lines		-0.5	3.6	V	
All Inputs					
Input Leakage Current		-10	10	µA	
All Outputs					
Output Leakage Current		-10	10	µA	

- **Power Supply Voltage - High Voltage MultiMediaCard**

Parameter	Symbol	Min	Max.	Unit	Remark
Supply voltage	V_{DD}	2.7	3.6	V	
Supply voltage differentials (V_{SS1}, V_{SS2})		-0.5	0.5	V	

- **Power Supply Voltage - Low Voltage MultiMediaCard**

Parameter	Symbol	Min	Max.	Unit	Remark
Supply voltage (low voltage range)	V_{DDL}	1.65	1.95	V	1.95V - 2.7V is not supported
Supply voltage (high voltage range)	V_{DDH}	2.7	3.6	V	
Supply voltage differentials (V_{SS1}, V_{SS2})		-0.5	0.5	V	

The current consumption of any card during the power-up procedure must not exceed 10 mA.

- **Bus Signal Line Load**

The total capacitance C_L of each line of the MultiMediaCard bus is the sum of the bus master capacitance C_{HOST}, the bus capacitance C_{BUS} itself and the capacitance C_{CARD} of each card connected to this line:

$C_L = C_{HOST} + C_{BUS} + N*C_{CARD}$

where N is the number of connected cards. Requiring the sum of the host and bus capacitances not to exceed 30 pF for up to 10 cards, and 40 pF for up to 30 cards, the following values must not be exceeded:

Parameter	Symbol	Min	Max.	Unit	Remark
Pull-up resistance for CMD	R_{CMD}	4.7	100	KOhm	to prevent bus floating
Pull-up resistance for DAT	R_{DAT}	50	100	KOhm	to prevent bus floating
Bus signal line capacitance	C_L		250	pF	f_{PP} <= 5 MHz, 30 cards
Bus signal line capacitance	C_L		100	pF	f_{PP} <= 20 MHz, 10 cards
Single card capacitance	C_{CARD}		7	pF	
Maximum signal line inductance			16	nH	f_{PP} <= 20 MHz

6.6 Bus Signal Levels

As the bus can be supplied with a variable supply voltage, all signal levels are related to the supply voltage.

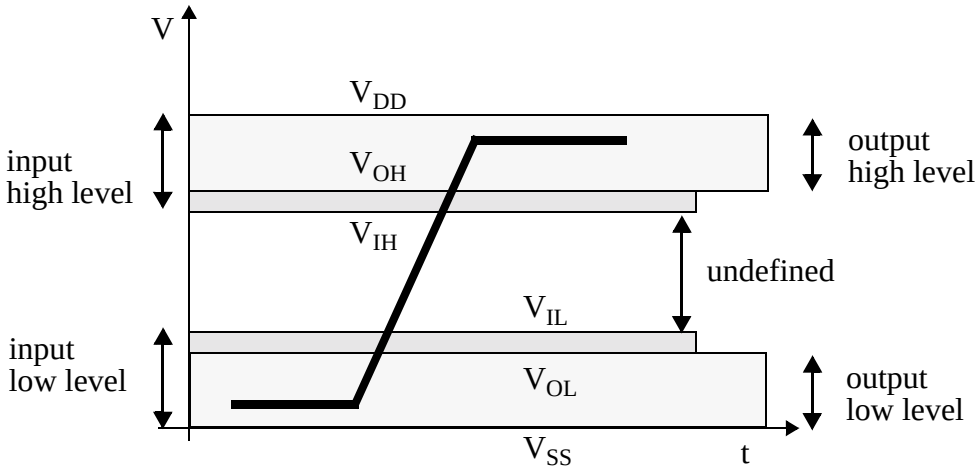


Figure 40: Bus Signal Levels

6.6.1 Open-Drain Mode Bus Signal Level

Parameter	Symbol	Min	Max.	Unit	Conditions
Output HIGH voltage	V_{OH}	V_{DD}-0.2		V	I_{OH} = -100 µA
Output LOW voltage	V_{OL}		0.3	V	I_{OL} = 2 mA

The input levels are identical with the push-pull mode bus signal levels.

6.6.2 Push-Pull Mode Bus Signal Level - High Voltage MultiMediaCard

To meet the requirements of the JEDEC specification JESD8-1A, the card input and output voltages shall be within the following specified ranges for any V_{DD} of the allowed voltage range:

Parameter	Symbol	Min	Max.	Unit	Conditions
Output HIGH voltage	V_{OH}	0.75*V_{DD}		V	I_{OH}=-100 µA @V_{DD} min
Output LOW voltage	V_{OL}		0.125*V_{DD}	V	I_{OL}=100 µA @V_{DD} min
Input HIGH voltage	V_{IH}	0.625*V_{DD}	V_{DD} + 0.3	V	
Input LOW voltage	V_{IL}	VSS-0.3	0.25*V_{DD}	V	

6.6.3 Push-Pull Mode Bus Signal Level - Low voltage MultiMediaCard

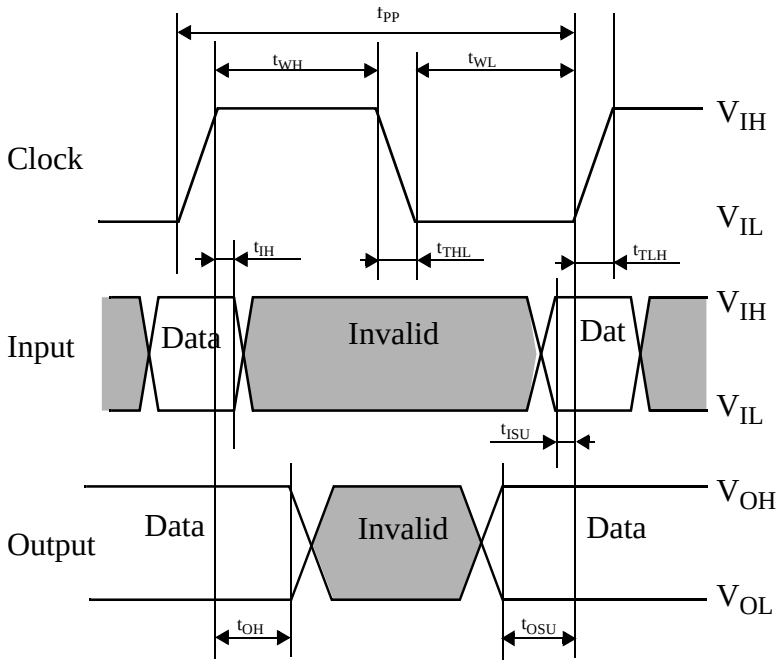
The definition of the I/O signal levels for the Low Voltage MultiMediaCard changes as a function of $V_{DD.}$

- 2.7V - 3.6V: Identical to the High Voltage MultiMediaCard (refer to Chapter 6.6.2 above).

- 1.95 - 2.7V: Undefined. The card is not operating at this voltage range.
- 1.65 - 1.95V: Compatible with EIA/JEDEC Standard “EIA/JESD8-7 Wide Range” as defined in the following table.

Parameter	Symbol	Min	Max.	Unit	Conditions
Output HIGH voltage	V_{OH}	V_{DD} - 0.2V		V	I_{OH}=-100 µA @V_{DD} min
Output LOW voltage	V_{OL}		0.2V	V	I_{OL}=100 µA @V_{DD} min
Input HIGH voltage	V_{IH}	0.7 * V_{DD}	V_{DD} + 0.3	V	
Input LOW voltage	V_{IL}	VSS-0.3	0.3 * V_{DD}	V	

6.7 Bus Timing



Data must always be sampled on the rising edge of the clock.

Figure 41: Timing Diagram: Data Input/Output

Parameter	Symbol	Min	Max.	Unit	Remark
Clock CLK (All values are referred to min (V_{IH}) and max (V_{IL}),					
Clock frequency Data Transfer Mode (PP) 10 cards	f_{PP}	0	20	MHz	C_L <= 100 pF (10 cards)
Clock frequency Data Transfer Mode (PP) 30 cards	f_{PP}	0	5	MHz	C_L <= 250 pF (30 cards)

Parameter	Symbol	Min	Max.	Unit	Remark
Clock frequency Identification Mode (OD)	f_{OD}	0	400	kHz	
Clock low time	t_{WL}	10		ns	C_L <= 100 pF (10 cards)
Clock high time	t_{WH}	10		ns	C_L <= 100 pF (10 cards)
Clock rise time	t_{TLH}		10	ns	C_L <= 100 pF (10 cards)
Clock fall time	t_{THL}		10	ns	C_L <= 100 pF (10 cards)
Clock low time	t_{WL}	50		ns	C_L <= 250 pF (30 cards)
Clock high time	t_{WH}	50		ns	C_L <= 250 pF (30 cards)
Clock rise time	t_{TLH}		50	ns	C_L <= 250 pF (30 cards)
Clock fall time	t_{THL}		50	ns	C_L <= 250 pF (30 cards)
Inputs CMD, DAT (referenced to CLK)					
Input set-up time	t_{ISU}	3		ns	
Input hold time	t_{IH}	3		ns	
Outputs CMD, DAT (referenced to CLK)					
Output set-up time	t_{OSU}	5		ns	
Output hold time	t_{OH}	5		ns	

7 SPI Mode

7.1 Introduction

The SPI mode consists of a secondary, optional communication protocol which is offered by Flash-based MultiMediaCards. This mode is a subset of the MultiMediaCard protocol, designed to communicate with a SPI channel, commonly found in Motorola's (and lately a few other vendors') microcontrollers. The interface is selected during the first reset command after power up (CMD0) and cannot be changed once the part is powered on.

The SPI standard defines the physical link only, and not the complete data transfer protocol. The MultiMediaCard SPI implementation uses a subset of the MultiMediaCard protocol and command set. It is intended to be used by systems which require a small number of cards (typically one) and have lower data transfer rates (compared to MultiMediaCard protocol based systems). From the application point of view, the advantage of the SPI mode is the capability of using an off-the-shelf host, hence reducing the design-in effort to minimum. The disadvantage is the loss of performance of the SPI mode versus MultiMediaCard mode (lower data transfer rate, fewer cards, hardware CS per card, etc.).

7.2 SPI Interface Concept

The Serial Peripheral Interface (SPI) is a general purpose synchronous serial interface originally found on certain Motorola microcontrollers. A virtually identical interface can now be found on certain TI and SGS Thomson microcontrollers as well.

The MultiMediaCard SPI interface is compatible with SPI hosts available on the market. As in any other SPI device, the MultiMediaCard SPI channel consists of the following four signals:

CS: Host to card Chip Select signal.

CLK: Host to card clock signal

DataIn: Host to card data signal.

DataOut: Card to host data signal.

Another SPI common characteristic is byte transfers, which is implemented in the card as well. All data tokens are multiples of bytes (8 bit) and always byte aligned to the CS signal.

7.3 SPI Bus Topology

The card identification and addressing methods are replaced by a hardware Chip Select (CS) signal. There are no broadcast commands. For every command, a card (slave) is selected by asserting (active low) the CS signal (see Figure 42).

The CS signal must be continuously active for the duration of the SPI transaction (command, response and data). The only exception occurs during card programming, when the host can de-assert the CS signal without affecting the programming process.

The bidirectional CMD and DAT lines are replaced by unidirectional *dataIn* and *dataOut* signals. This eliminates the ability of executing commands while data is being read or written and, therefore, makes the sequential operations obsolete. Only single and multiple block read/write operations are supported in SPI channel.

The MultiMediaCard pin assignment in SPI mode (compared to MultiMediaCard mode) is given in Table 38.

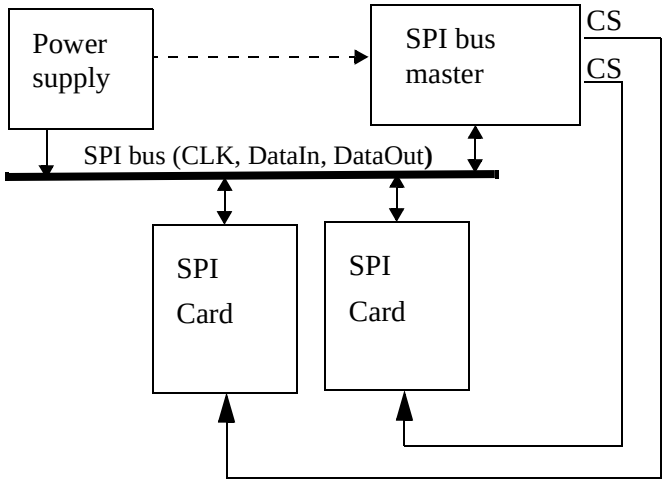


Figure 42: MultiMediaCard Bus System

Pin #	MultiMediaCard Mode			SPI Mode		
	Name	**Type[1]**	**Description**	**Name**	**Type**	**Description**
1	RSV	NC	Reserved for future use	CS	I	Chip Select (neg true)
2	CMD	I/O/PP/OD	Command/Response	DI	I/PP	Data In
3	V_{SS1}	S	Supply voltage ground	VSS	S	Supply voltage ground
4	V_{DD}	S	Supply voltage	VDD	S	Supply voltage
5	CLK	I	Clock	SCLK	I	Clock
6	V_{SS2}	S	Supply voltage ground	VSS2	S	Supply voltage ground
7	DAT	I/O/PP	Data	DO	O/PP	Data Out

Table 38: SPI Interface Pin Configuration

1) S: power supply; I: input; O: output; PP: push-pull; OD: open-drain; NC: Not connected (or logical high)

7.4 MultiMediaCard Registers in SPI Mode

The register usage in SPI mode is summarized in Table 39 (refer to Chapter 3 for more information). Most of them are inaccessible.

Name	Available in SPI mode	Width [Bytes]	Description
CID	Yes	16	Card identification data (serial number, manufacturer ID, etc.)
RCA	No		
DSR	No		
CSD	Yes	16	Card-specific data, information about the card operation conditions.
OCR	Yes	32	Operation condition register.

Table 39: MultiMediaCard registers in SPI mode

7.5 SPI Bus Protocol

While the MultiMediaCard channel is based on command and data bit streams which are initiated by a start bit and terminated by a stop bit, the SPI channel is byte oriented. Every command or data block is built of 8-bit bytes and is byte aligned to the CS signal (i.e. the length is a multiple of 8 clock cycles).

Similar to the MultiMediaCard protocol, the SPI messages consist of command, response and data-block tokens (see Chapter 3 for a detailed description). All communication between host and cards is controlled by the host (master). The host starts every bus transaction by asserting the CS signal low.

The response behavior in the SPI mode differs from the MultiMediaCard mode in the following three aspects:

- The selected card always responds to the command.
- Additional (8, 16 & 40 bit) response structures are used
- When the card encounters a data retrieval problem, it will respond with an error response (which replaces the expected data block) rather than by a time-out, as in the MultiMediaCard mode.

Only single and multiple block read/write operations are supported in SPI mode (sequential mode is not supported). In addition to the command response, every data block sent to the card during write operations will be responded to with a special data response token. A data block may be as big as one card write block and as small as a single byte. Partial block read/write operations are enabled by card options specified in the CSD register.

7.5.1 Mode Selection

The MultiMediaCard wakes up in the MultiMediaCard mode. It will enter SPI mode if the CS signal is asserted (negative) during the reception of the reset command (CMD0). Selecting SPI mode is not restricted to *Idle* state (the state the card enters after power up) only. Every time the card receives CMD0, including while in *Inactive* state, CS signal is sampled.

If the card recognizes that the MultiMediaCard mode is required (CS signal is high), it will not respond to the command and remain in the MultiMediaCard mode. If SPI mode is required (CS signal is low), the card will switch to SPI and respond with the SPI mode R1 response.

The only way to return to the MultiMediaCard mode is by a power cycle (turn the power off an on). In SPI mode, the MultiMediaCard protocol state machine is not observed. All the MultiMediaCard commands supported in SPI mode are always available.

7.5.2 Bus Transfer Protection

Every MultiMediaCard token transferred on the bus is protected by CRC bits. In SPI mode, the

MultiMediaCard offers a non-protected mode which enables systems built with reliable data links to exclude the hardware or firmware required for implementing the CRC generation and verification functions.

In the non-protected mode, the CRC bits of the command, response and data tokens are still required in the tokens. However, they are defined as ‘don’t care’ for the transmitter and ignored by the receiver.

The SPI interface is initialized in the non-protected mode. However, the RESET command (CMD0), which is used to switch the card to SPI mode, is received by the card while in MultiMediaCard mode and, therefore, must have a valid CRC field.

Since CMD0 has no arguments, the content of all the fields, including the CRC field, are constants and need not be calculated in run time. A valid reset command is:

0x40, 0x0, 0x0, 0x0, 0x0, 0x95

The host can turn the CRC option on and off using the CRC_ON_OFF command (CMD59).

7.5.3 Data Read

The SPI mode supports single and multiple block read operations. The main difference between SPI and MultiMediaCard modes is that the data and the response are both transmitted to the host on the DataOut signal (refer to Figure 43 and Figure 44). Therefore the card response to the STOP_COMMAND may cut-short and replace the last data block.

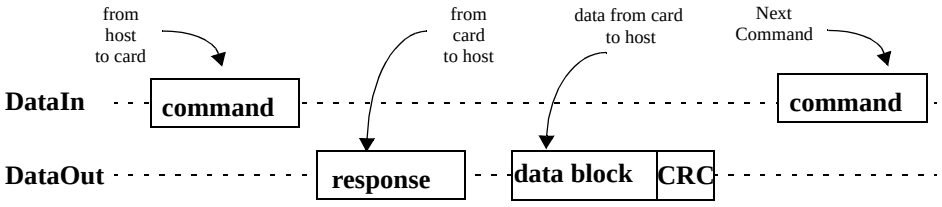


Figure 43: SPI Single Block Read Operation

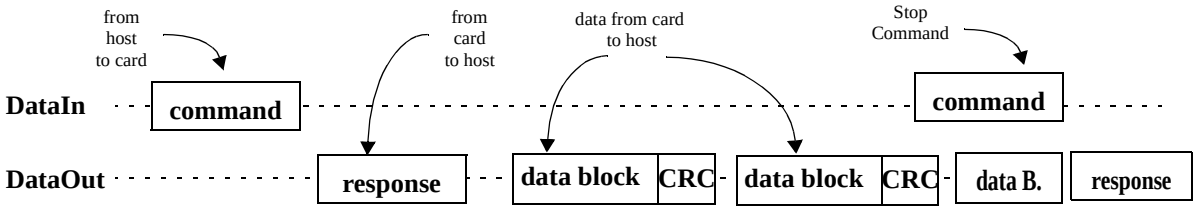


Figure 44: SPI Multiple Block Read Operation

The basic unit of data transfer is a block whose maximum size is defined in the CSD (READ_BL_LEN).If READ_BL_PARTIAL is set, smaller blocks whose starting and ending address are entirely contained within

one physical block (as defined by READ_BL_LEN) may also be transmitted. A CRC is appended to the end of each block ensuring data transfer integrity. CMD17 (READ_SINGLE_BLOCK) initiates a single block read. CMD18 (READ_MULTIPLE_BLOCK) starts a transfer of several consecutive blocks. Two types of multiple block read transactions are defined (the host can use either one at any time):

- Open-ended Multiple block read

The number of blocks for the read multiple block operation is not defined. The card will continuously transfer data blocks until a stop transmission command is received.

- Multiple block read with pre-defined block count

The card will transfer the requested number of data blocks and terminate the transaction. Stop command is not required at the end of this type of multiple block read, unless terminated with an error. In order to start a multiple block read with pre-defined block count the host must use the SET_BLOCK_COUNT command (CMD23) immediately preceding the READ_MULTIPLE_BLOCK (CMD18) command. Otherwise the card will start an open-ended multiple block read which can be stopped using the STOP_TRANSMISION command.

The host can abort reading at any time, within a multiple block operation, regardless of the its type. Transaction abort is done by sending the stop transmission command.

In case of a data retrieval error (e.g. out of range, address misalignment, internal error, etc.), the card will not transmit any data. Instead (as opposed to MultiMediaCard mode where the card times out), a special data error token will be sent to the host. Figure 45 shows a single block read operation which terminates with an error token rather than a data block.

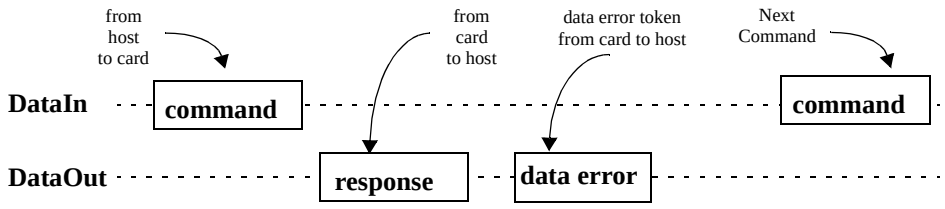


Figure 45: SPI Read Operation - Data Error

Multiple block read operation can be terminated the same way, the error token replacing a data block anywhere in the sequence. The host must than abort the operation by sending the stop transmission command.

If the host sends a stop transmission command after the card transmitted the last block of a multiple block read with a pre-defined number of blocks, it will be responded to as an illegal command.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed, the card shall detect a block misalignment error condition at the beginning of the first misaligned block (ADDRESS_ERROR error bit will is set in the data error token).

7.5.4 Data Write

The SPI mode supports single block and Multiple block write commands. Upon reception of a valid write command (CMD24 or CMD25), the card will respond with a response token and will wait for a data block to be sent from the host. CRC suffix, block length and start address restrictions are (with the exception of the

CSD parameter WRITE_BL_PARTIAL controlling the partial block write option) identical to the read operation (see Figure 46). If a CRC error is detected it will be reported in the data-response token and the data block will not be programmed.

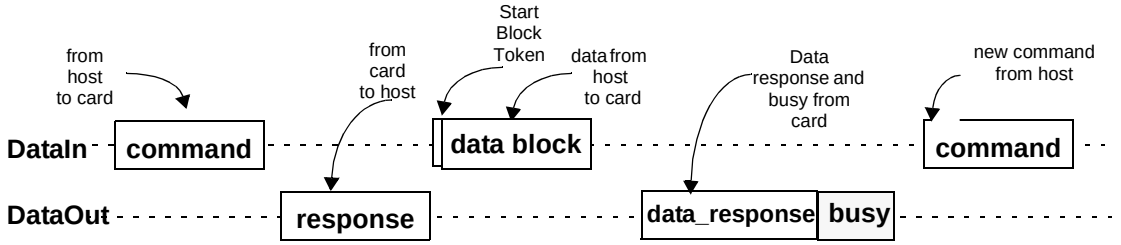


Figure 46: SPI Single Block Write Operation

Every data block has a prefix of ‘Start Block’ token (one byte).

After a data block has been received, the card will respond with a data-response token. If the data block has been received without errors, it will be programmed. As long as the card is busy programming, a continuous stream of busy tokens will be sent to the host (effectively holding the DataOut line low).

In Multiple Block write operation the stop transmission will be done by sending ‘Stop Tran’ token instead of ‘Start Block’ token at the beginning of the next block.

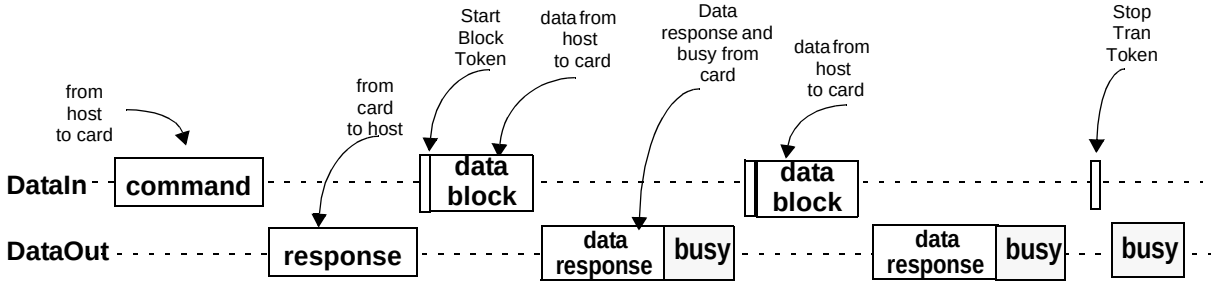


Figure 47: SPI Multiple Block Write Operation

Two types of multiple block write transactions, identical to the multiple block read, are defined (the host can use either one at any time):

- Open-ended Multiple block write

The number of blocks for the write multiple block operation is not defined. The card will continuously accept and program data blocks until a ‘Stop Tran’ token is received.

- Multiple block write with pre-defined block count

The card will accept the requested number of data blocks and terminate the transaction. ‘Stop tran’ token is not required at the end of this type of multiple block write, unless terminated with an error. In order to start a multiple block write with pre-defined block count the host must use the SET_BLOCK_COUNT command (CMD23) immediately preceding the WRITE_MULTIPLE_BLOCK (CMD25) command. Otherwise the card will start an open-ended multiple block write which can be stopped using the ‘Stop tran’ token.

The host can abort writing at any time, within a multiple block operation, regardless of the its type. Transaction abort is done by sending the ‘Stop tran’ token. If a multiple block write with pre-defined block count is aborted, the data in the remaining blocks is not defined.

If the card detects a CRC error or a programming error (e.g. write protect violation, out of range, address misalignment, internal error, etc.) during a multiple block write operation (both types) it will report the failure in the data-response token and ignore any further incoming data blocks. The host must than abort the operation by sending the ‘Stop Tran’ token.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card shall detect the block misalignment error before the beginning of the first misaligned block and respond with an error indication in the data-response token and ignore any further incoming data blocks. The host must than abort the operation by sending the ‘Stop Tran’ token.

Once the programming operation is completed (either successfully or with an error), the host must check the results of the programming (or the cause of the error if already reported in the data-response token) using the SEND_STATUS command (CMD13).

If the host sends a ‘Stop Trans’ token after the card received the last data block of a multiple block operation with pre-defined number of blocks, it will be interpreted as the beginning of an illegal command and responded accordingly.

While the card is busy, resetting the CS signal will not terminate the programming process. The card will release the DataOut line (tri-state) and continue with programming. If the card is reselected before the programming is finished, the DataOut line will be forced back to low and all commands will be rejected.

Resetting a card (using CMD0) will terminate any pending or active programming operations. This may destroy the data formats on the card. It is in the responsibility of the host to prevent it.

7.5.5 Erase & Write Protect Management

The erase and write protect management procedures in the SPI mode are identical to those of the MultiMediaCard mode. While the card is erasing or changing the write protection bits of the predefined erase groups list, it will be in a busy state and hold the DataOut line low. Figure 48 illustrates a ‘no data’ bus transaction with and without busy signalling.

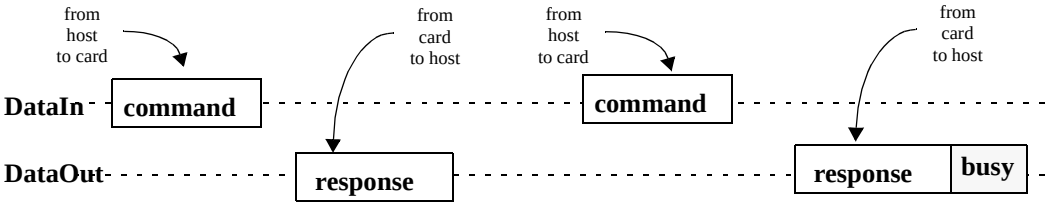


Figure 48: SPI ‘No data’ Operations

7.5.6 Read CID/CSD Registers

Unlike the MultiMediaCard protocol (where the register contents is sent as a command response), reading the contents of the CSD and CID registers in SPI mode is a simple read-block transaction. The card will respond with a standard response token (see Figure 45) followed by a data block of 16 bytes suffixed with a 16 bit CRC.

The data time out for the CSD command cannot be set to the card TAAC since this value is stored in the CSD.

Therefore, the standard response time-out value (N_{CR}) is used for read latency of the CSD register.

7.5.7 Reset Sequence

The MultiMediaCard requires a defined reset sequence. After power on reset or CMD0 (software reset) the card enters an idle state. At this state the only legal host commands are CMD1 (SEND_OP_COND) and CMD58 (READ_OCR).

The host must poll the card (by repeatedly sending CMD1) until the 'in-idle-state' bit in the card response indicates (by being set to 0) that the card has completed its initialization processes and is ready for the next command.

In SPI mode, as opposed to MultiMediaCard mode, CMD1 has no operands and does not return the contents of the OCR register. Instead, the host may use CMD58 (available in SPI mode only) to read the OCR register. Furthermore, it is in the responsibility of the host to refrain from accessing cards that do not support its voltage range.

The usage of CMD58 is not restricted to the initializing phase only, but can be issued at any time. The host must poll the card (by repeatedly sending CMD1) until the 'in-idle-state' bit in the card response indicates (by being set to 0) that the card has completed its initialization processes and is ready for the next command.

7.5.8 Clock Control

The SPI bus clock signal can be used by the SPI host to set the cards to energy saving mode or to control the data flow (to avoid under-run or over-run conditions) on the bus. The host is allowed to change the clock frequency or shut it down.

There are a few restrictions the SPI host must follow:

- The bus frequency can be changed at any time (under the restrictions of maximum data transfer frequency, defined by the MultiMediaCards
- It is an obvious requirement that the clock must be running for the MultiMediaCard to output data or response tokens. After the last SPI bus transaction, the host is required, to provide 8 (eight) clock cycles for the card to complete the operation before shutting down the clock. throughout this 8 clocks period the state of the CS signal is irrelevant. it can be asserted or de-asserted.
  Following is a list of the various SPI bus transactions:
- A command / response sequence. 8 clocks after the card response end bit. The CS signal can be asserted or de-asserted during these 8 clocks.
- A read data transaction. 8 clocks after the end bit of the last data block.
- A write data transaction. 8 clocks after the CRC status token.
- The host is allowed to shut down the clock of a "busy" card. The MultiMediaCard will complete the programming operation regardless of the host clock. However, the host must provide a clock edge for the card to turn off its busy signal. Without a clock edge the MultiMediaCard (unless previously disconnected by de-asserting the CS signal) will force the dataOut line down, permanently.

7.5.9 Error Conditions

• CRC and Illegal Command

All commands are (optionally) protected by CRC (cyclic redundancy check) bits. If the addressed MultiMediaCard's CRC check fails, the COM_CRC_ERROR bit will be set in the card's response. Similarly, if an illegal command has been received the ILLEGAL_COMMAND bit will be set in the card's response.

There are different kinds of illegal commands:

- Commands which belong to classes not supported by the MultiMediaCard (e.g. interrupt and I/O commands).
- Commands not allowed in SPI mode (e.g. CMD20 - write stream)
- Commands which are not defined (e.g. CMD6).

- **Read, Write and Erase Time-out Conditions**

The time period after which a time-out condition for read/write/erase operations occurs are (card independent) 10 times longer than the typical access/program times for these operations given below. A card shall complete the command within this time period, or give up and return an error message. If the host does not get a response within the defined time-out it should assume the card is not going to respond any more and try to recover (e.g. reset the card, power cycle, reject, etc.).

The typical access and program times are defined as follows:

- Read

The read access time is defined as the sum of the two times given by the CSD parameters TAAC and NSAC. These card parameters define the typical delay between the end bit of the read command and the start bit of the data block. This number is card dependent.

- Write

The R2W_FACTOR field in the CSD is used to calculate the typical block program time obtained by multiplying the read access time by this factor. It applies to all write/erase commands (e.g. SET(CLEAR)_WRITE_PROTECT, PROGRAM_CSD(CID) and the block write commands).

- Erase

The duration of an erase command will be (order of magnitude) the number of write blocks to be erased multiplied by the block write delay.

- **Read ahead in Multiple Block read operation**

In Multiple Block read operations, in order to improve read performance, the card may fetch data from the memory array, ahead of the host. In this case, when the host is reading the last addresses of the memory, the card attempts to fetch data beyond the last physical memory address and generates an OUT_OF_RANGE error.

Therefore, even if the host times the stop transmission command to stop the card immediately after the last byte of data was read, the card may already have generated the error, and it will show in the response to the stop transmission command. The host should ignore this error.

7.5.10 Memory Array Partitioning

Same as for MultiMediaCard mode.

7.5.11 Card Lock/unlock

Usage of card lock and unlock commands in SPI mode is identical to MultiMediaCard mode. In both cases, the command response is of type R1b. After the busy signal clears, the host should obtain the result of the operation by issuing a GET_STATUS command. Please refer to Chapter 4.4.5 for details.

7.5.12 Application Specific commands

Identical to MultiMediaCard mode with the exception of the APP_CMD status bit (refer to Chapter 4.4.6), which is not available in SPI.

7.6 SPI Mode Transaction Packets

7.6.1 Command Tokens

- **Command Format**

All the MultiMediaCard commands are 6 bytes long. The command transmission always starts with the left bit of the bitstring corresponding to the command codeword. All commands are protected by a CRC (see Chapter 8.2). The commands and arguments are listed in Table 41.

Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width (bits)	1	1	6	32	7	1
Value	'0'	'1'	x	x	x	'1'
Description	start bit	transmission bit	command index	argument	CRC7	end bit

- **Command Classes**

As in MultiMediaCard mode, the SPI commands are divided into several classes (See Table 40). Each class supports a set of card functions. A MultiMediaCard will support the same set of optional command classes in both communication modes (there is only one command class table in the CSD register). The available command classes, and the supported command for a specific class, however, are different in the MultiMediaCard and the SPI communication mode.

Card CMD Class (CCC)	Class Description	Supported commands																							
		0	1	9	10	12	13	16	17	18	23	24	25	27	28	29	30	35	36	38	42	55	56	58	59
class 0	Basic	+	+	+	+		+																	+	+
class 1	Not supported in SPI																								
class 2	Block read					+		+	+	+	+														
class 3	Not supported in SPI																								
class 4	Block write							+			+	+	+	+											
class 5	Erase																	+	+	+					
class 6	Write-protection														+	+	+								
class 7	Lock Card																				+				
class 8	Application specific																					+	+		
class 9	Not supported in SPI																								
class 10-11	Reserved																								

Table 40: Command classes in SPI mode

- **Detailed Command Description**

The following table provides a detailed description of the SPI mode commands. The responses are defined in Chapter 7.6.2. Table 41 lists all MultiMediaCard commands. A "yes" in the SPI mode column indicates that

the command is supported in SPI mode. With these restrictions, the command class description in the CSD is still valid. If a command does not require an argument, the value of this field should be set to zero. The reserved commands are also reserved in MultiMediaCard mode.

The binary code of a command is defined by the mnemonic symbol. As an example, the content of the **command index** field is (binary) '000000' for CMD0 and '100111' for CMD39.

CMD INDEX	SPI Mode	Argument	Resp	Abbreviation	Command Description
CMD0	Yes	None	R1	GO_IDLE_STATE	resets the MultiMedia card
CMD1	Yes	None	R1	SEND_OP_ COND	activates the card's initialization process
CMD2	No				
CMD3	No				
CMD4	No				
CMD5	reserved				
CMD6	reserved				
CMD7	No				
CMD8	reserved				
CMD9	Yes	None	R1	SEND_CSD	asks the selected card to send its card-specific data (CSD)
CMD10	Yes	None	R1	SEND_CID	asks the selected card to send its card identification (CID)
CMD11	No				
CMD12	Yes	None	R1	STOP_ TRANSMISSION	Stop transmission on multiple block read
CMD13	Yes	None	R2	SEND_STATUS	asks the selected card to send its status register
CMD14	reserved				
CMD15	No				
CMD16	Yes	[31:0] block length	R1	SET_BLOCKLEN	selects a block length (in bytes) for all following block commands (read and write)[1]
CMD17	Yes	[31:0] data address	R1	READ_SINGLE_ BLOCK	reads a block of the size selected by the SET_BLOCKLEN command[2]
CMD18	Yes	[31:0] data address	R1	READ_MULTIPLE _BLOCK	continuously transfers data blocks from card to host until interrupted by a stop command or the requested number of data blocks transmitted
CMD19	reserved				
CMD20	No				
CMD21 ... CMD22	reserved				

Table 41: Commands and arguments

CMD INDEX	SPI Mode	Argument	Resp	Abbreviation	Command Description
CMD23	Yes	[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 exceeding multiple block read or write command.
CMD24	Yes	[31:0] data address	R1	WRITE_BLOCK	writes a block of the size selected by the SET_BLOCKLEN command. [3]
CMD25	Yes	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	continuously writes blocks of data until a “Stop Tran’ Token or the requested number of blocks received.
CMD26	No				
CMD27	Yes	None	R1	PROGRAM_CSD	programming of the programmable bits of the CSD
CMD28	Yes	[31:0] data address	R1b[4]	SET_WRITE_PROT	if the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29	Yes	[31:0] data address	R1b	CLR_WRITE_PROT	if the card has write protection features, this command clears the write protection bit of the addressed group
CMD30	Yes	[31:0] write protect data address	R1	SEND_WRITE_PROT	if the card has write protection features, this command asks the card to send the status of the write protection bits [5]
CMD31	reserved				
CMD32 ... CMD34	Reserved. These command indexes cannot be used in order to maintain backwards compatibility with older versions of the MultiMediaCards				
CMD35	Yes	[31:0] data address	R1	TAG_ERASE_GROUP_START	sets the address of the first erase group within a range to be selected for erase
CMD36	Yes	[31:0] data address	R1	TAG_ERASE_GROUP_END	sets the address of the last erase group within a continuous range to be selected for erase
CMD37	Reserved. This command index cannot be used in order to maintain backwards compatibility with older versions of the MultiMediaCards				
CMD38	Yes	[31:0] stuff bits	R1b	ERASE	erases all previously selected erase groups
CMD39	No				
CMD40	No				
CMD41	reserved				
CMD42	Yes	[31:0] stuff bits.	R1b	LOCK_UNLOCK	used to Set/Reset the Password or lock/unlock the card. The structure of the data block is described in Chapter 4.4.5. The size of the Data Block is defined by the SET_BLOCK_LEN command.

Table 41: Commands and arguments

CMD INDEX	SPI Mode	Argument	Resp	Abbreviation	Command Description
CMD43 ... CMD54	reserved				
CMD55	Yes	[31:0] stuff bits	R1	APP_CMD	defines to the card that the next command is an application specific command rather than a standard command
CMD56	Yes	[31:1] stuff bits. [0]: RD/WR_[6]	R1b	GEN_CMD	used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block is defined by the SET_BLOCK_LEN command.
CMD57	Reserved				
CMD58	Yes	None	R3	READ_OCR	reads the OCR register of a card
CMD59	Yes	[31:1] stuff bits [0:0] CRC option	R1	CRC_ON_OFF	turns the CRC option on or off. A ‘1’ in the CRC option bit will turn the option on, a ‘0’ will turn it off
CMD60 -63	No				

Table 41: Commands and arguments

1)The default block length is as specified in the CSD.
2)The data transferred must not cross a physical block boundary unless READ_BLK_MISALIGN is set in the CSD.
3)The data transferred must not cross a physical block boundary unless WRITE_BLK_MISALIGN is set in the CSD.
4)R1b: R1 response with an optional trailing busy signal.
5)32 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits are transferred in a payload format via the data line. The last (least significant) bit of the protection bits corresponds to the first addressed group. If the addresses of the last groups are outside the valid range, then the corresponding write protection bits are set to zero.
6) **RD/WR_**: “1” the host receives a data block from the card.
“0” the host sends a data block to the card.

7.6.2 Responses

There are several types of response tokens. As in the MultiMediaCard mode, all are transmitted MSB first:

- **Format R1**

This response token is sent by the card after every command, with the exception of SEND_STATUS commands. It is one byte long, and the MSB is always set to zero. The other bits are error indications, an error being signaled by a ‘1’. The structure of the R1 format is given in Figure 49. The meaning of the flags is defined as follows:

- **In idle state**: The card is in idle state and running the initializing process.
- **Erase reset:** An erase sequence was cleared before executing because an out of erase sequence command was received.
- **Illegal command**: An illegal command code was detected.

- **Communication CRC error**: The CRC check of the last command failed.
- **Erase sequence error**: An error occurred in the sequence of erase commands.
- **Address error**:A misaligned address, which did not match the block length, was used in the command.
- **Parameter error**: The command's argument (e.g. address, block length) was out of the allowed range for this card.

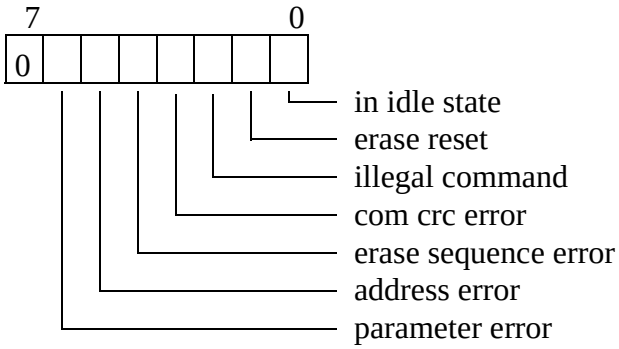


Figure 49: R1 Response Format

- **Format R1b**

This response token is identical to the R1 format with the optional addition of the busy signal.

- **Busy**

The busy signal token can be any number of bytes. A zero value indicates card is busy. A non-zero value indicates the card is ready for the next command.

- **Format R2**

This response token is two bytes long and sent as a response to the SEND_STATUS command. The format is given in Figure 50.

The first byte is identical to the response R1. The content of the second byte is described in the following:

- **out of range | csd_overwrite:** This status bit has two functions. It is set if the command argument was out of its valid range or if the host is trying to change the ROM section or reverse the copy bit (set as original) or permanent WP bit (un-protect) of the CSD register.
- **Erase param**: An invalid selection of erase groups, for erase.
- **Write protect violation**: The command tried to write a write-protected block.
- **Card ECC failed**: Card internal ECC was applied but failed to correct the data.
- **CC error**: Internal card controller error.
- **Error**: A general or an unknown error occurred during the operation.
- **Write protect erase skip | lock/unlock command failed**: This status bit has two functions. It is set when the host attempts to erase a write-protected write block or if a sequence or password error occurred during a card lock/unlock operation.
- **Card is locked**: Set when the card is locked by the user. Reset when it is unlocked.

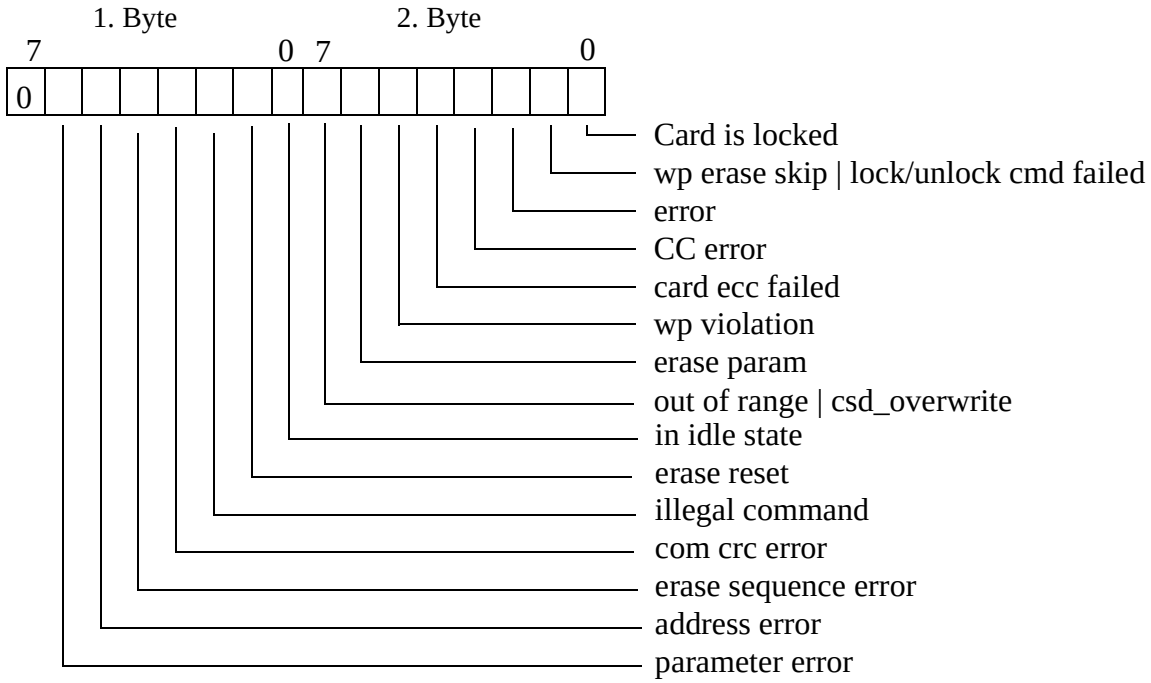


Figure 50: R2 Response Format

- **Format R3**

This response token is sent by the card when a READ_OCR command is received. The response length is 5 bytes (see Figure 51). The structure of the first (MSB) byte is identical to response type R1. The other four bytes contain the OCR register.

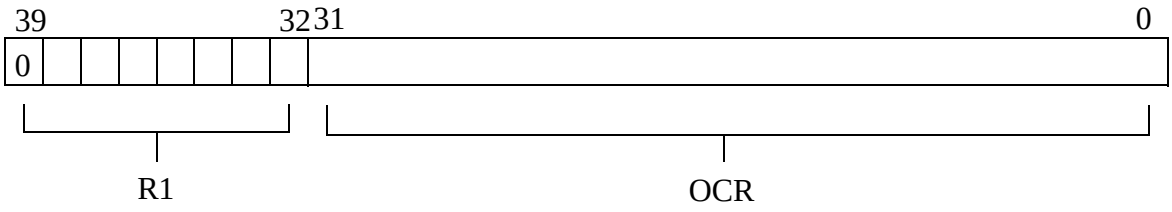


Figure 51: R3 Response Format

- **Data Response**

Every data block written to the card will be acknowledged by a data response token. It is one byte long and has the following format:

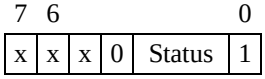


Figure 52: Data Response Format

The meaning of the status bits is defined as follows:

‘010’ - Data accepted.

‘101’ - Data rejected due to a CRC error.

’110’ - Data Rejected due to a Write Error

In case of any error (CRC or Write Error) during Write Multiple Block operation, the host shall abort the operation using the “Stop Tran” Token. In case of Write Error (response ‘110’) the host should send CMD13 (SEND_STATUS) in order to get the cause of the write problem.

7.6.3 Data Tokens

Read and write commands have data transfers associated with them. Data is being transmitted or received via data tokens. All data bytes are transmitted MSB first.

Data tokens are 4 to (N + 3) bytes long (Where N is the data block length set using the SET_BLOCK_LENGTH Command) and have the following format:

- First byte:

Token Type	Transaction Type	7	Bit Position						0
Start Block	Single Block Read	1	1	1	1	1	1	1	0
Start Block	Multiple Block Read	1	1	1	1	1	1	1	0
Start Block	Single Block Write	1	1	1	1	1	1	1	0
Start Block	Multiple Block Write	1	1	1	1	1	1	0	0
Stop Tran	Multiple Block Write	1	1	1	1	1	1	0	1

Figure 53: Start Data Block Token Format

- Bytes 2 - (N + 1): User data
- Last two bytes: 16 bit CRC.

7.6.4 Data Error Token

If a read operation fails and the card cannot provide the required data, it will send a data error token instead. This token is one byte long and has the following format:

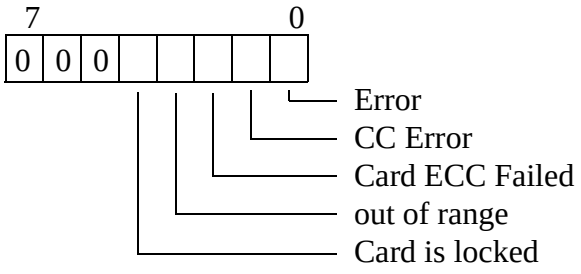


Figure 54: Data Error Token

The 4 least significant bits (LSB) are the same error bits as in the response format R2.

7.6.5 Clearing Status Bits

As described in the previous paragraphs, in SPI mode, status bits are reported to the host in three different formats: response R1, response R2 and data error token (the same bits may exist in multiple response types—e.g Card ECC failed)

As in the MultiMediaCard mode, error bits are cleared when read by the host, regardless of the response format. State indicators are either cleared by reading or in accordance with the card state.

All Error/Status bits defined in MultiMediaCard mode, with the exception of underrun and overrun, have the same meaning and usage in SPI mode.

The following table summarizes the set and clear conditions for the various status bits:

.

Identifier	Included in resp	Type[1]	Value	Description	Clear Condition[2]
Out of range	R2 DataErr	E R X	'0'= no error '1'= error	The command argument was out of the allowed range for this card.	C
Address error	R1 R2	E R X	'0'= no error '1'= error	An address which did not match the block length was used in the command.	C
Erase sequence error	R1 R2	E R	'0'= no error '1'= error	An error in the sequence of erase commands occurred.	C
Erase param	R2	E X	'0'= no error '1'= error	An error in the parameters of the erase command sequence.	C
Parameter error	R1 R2	E R X	'0'= no error '1'= error	An error in the parameters of the command.	C
WP violation	R2	E R X	'0'= not protected '1'= protected	Attempt to program a write protected block.	C
Com CRC error	R1 R2	E R	'0'= no error '1'= error	The CRC check of the previous command failed.	C
Illegal command	R1 R2	E R	'0'= no error '1'= error	Command not legal for the card state.	C
Card ECC failed	R2 DataEr	E X	'0'= success '1'= failure	Card internal ECC was applied but failed to correct the data.	C
CC error	R2 dataEr	E R X	'0'= no error '1'= error	Internal card controller error	C
Error	R2 dataEr	E R X	'0'= no error '1'= error	A general or an unknown error occurred during the operation.	C
WP erase skip	R2	S X	'0'= not protected '1'= protected	Only partial address space was erased due to existing write protected blocks.	C
Lock/Unlock cmd failed	R2	E X	'0'= no error '1'= error	Sequence or password error during card lock/unlock operation.	C

Table 42: SPI mode status bits

Identifier	Included in resp	Type[1]	Value	Description	Clear Condition[2]
Card is locked	R2 dataEr	S X	‘0’ = card is not locked ‘1’ = card is locked	Card is locked by a user password	A
Erase reset	R1 R2	S R	’0’= cleared ’1’= set	An erase sequence was cleared before executing because an out of erase sequence command was received.	C
In Idle state	R1 R2	S R	0 = Card is ready 1 = Card is in idle state	The card enters the idle state after power up or reset command. It will exit this state and become ready upon completion of its initialization procedures.	A
CSD overwrite	R2	E X	’0’= no error ’1’= error	The host is trying to change the ROM section, or is trying to reverse the copy bit (set as original) or permanent WP bit (un-protect) of the CSD register.	C

Table 42: SPI mode status bits

1) Type:

E: Error bit.

S: State bit.

R: Detected and set for the actual command response.

X: Detected and set during command execution. The host must poll the card by issuing the status command in order to read these bits.

When an error bit is defined as “ERX”, it means the bit could be set either for the actual command response (ER), or during command execution (EX). Therefore, if the host wants to properly detect this bit it will need to check in both the actual command response and after a SEND_STATUS command (CMD13).

2) Clear Condition:

A: According to the card current state.

C: Clear by read

7.7 Card Registers

In SPI mode, only the OCR, CSD and CID registers are accessible. Their format is identical to the format in the MultiMediaCard mode. However, a few fields are irrelevant in SPI mode.

7.8 SPI Bus Timing Diagrams

All timing diagrams use the following schematics and abbreviations:

H	Signal is high (logical ‘1’
L	Signal is low (logical ‘0’)
X	Don’t care (Undefined Value)
Z	High impedance state (-> = 1)
*	Repeater
Busy	Busy Token
Command	Command token
Response	Response token
Data block	Data token

All timing values are defined in Table 43. The host must keep the clock running for at least N_{CR} clock cycles after receiving the card response. This restriction applies to both command and data response tokens.

7.8.1 Command / Response

- **Host Command to Card Response - Card is ready**

The following timing diagram describes the basic command response (no data) SPI transaction.

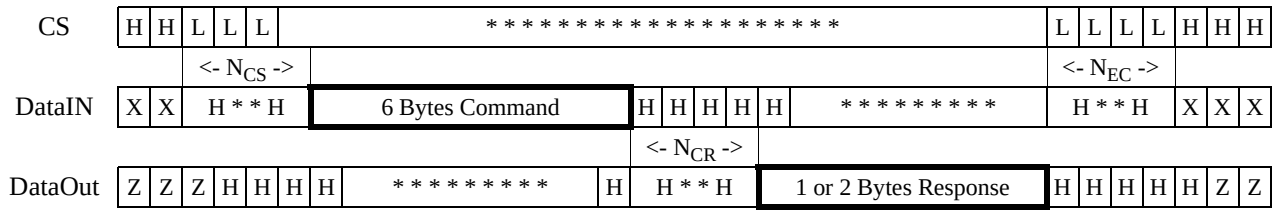


Figure 55: SPI Command/Response Transaction, Card Is Ready

- **Host Command to Card Response - card is busy**

The following timing diagram describes the command response transaction for commands when the card response is of type R1b (e.g. SET_WRITE_PROT and ERASE). When the card is signaling busy, the host may deselect it (by raising the CS) at any time. The card will release the DataOut line one clock after the CS going high. To check if the card is still busy, it needs to be reselected by asserting (set to low) the CS signal.

The card will resume busy signal (pulling DataOut low) one clock after the falling edge of CS.

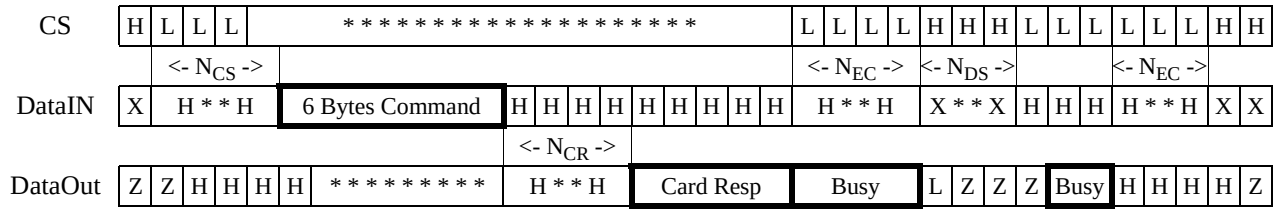


Figure 56: SPI Command/Response Transaction, Card Is Busy

- **Card Response to Host Command**

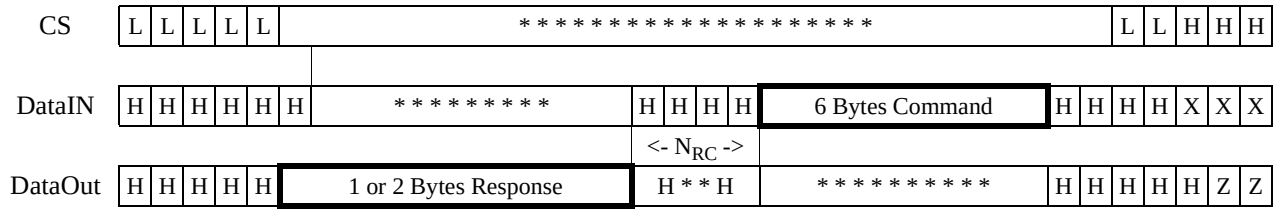


Figure 57: SPI Card Response To The Next Host Command

7.8.2 Data read

- **Single Block Read**

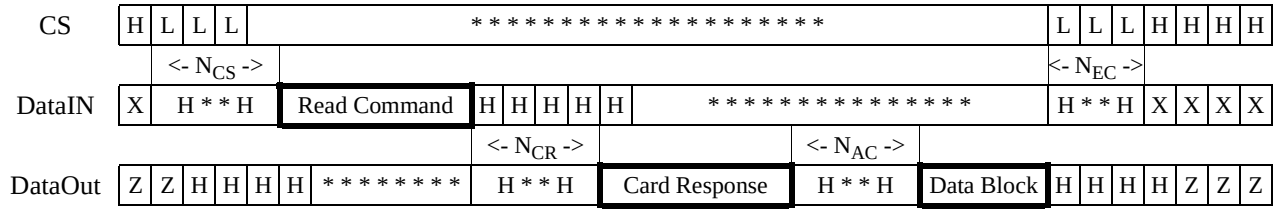


Figure 58: SPI Single Block Read

- **Multiple Block Read - Stop Transmission is sent between blocks**

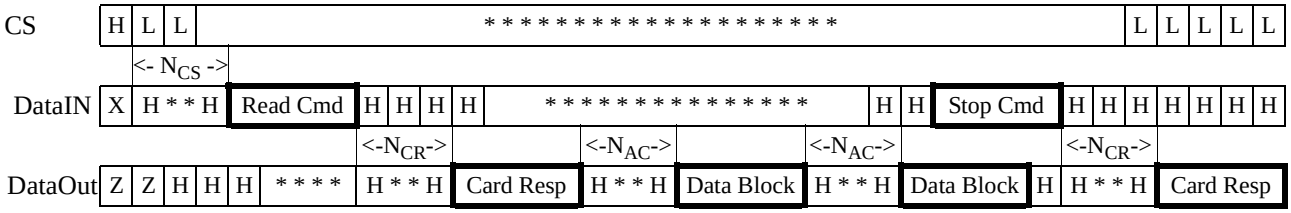


Figure 59: SPI Multiple Block Read, Stop Transmission Does Not Overlap Data

The timing for de-asserting the CS signal after the last card response is identical to a standard command/ response transaction as described in Figure 55;

- **Multiple Block Read - Stop Transmission is sent within a block**

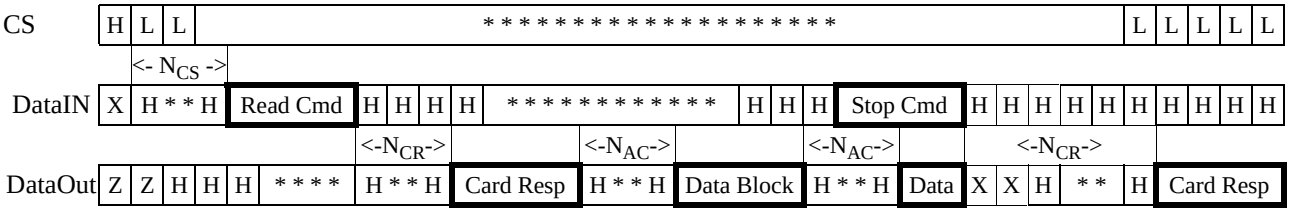


Figure 60: SPI Multiple Block Read, Stop Transmission Overlaps Data

In an Open-ended (or host aborted) multiple block read transaction the stop transmission command may be sent asynchronously to the data transmitted out of the card and may overlap the data block. In this case the card will stop sending the data and transmit the response token as well. The delay between command and response is standard N_{CR} Clocks. The first byte, however, is not guaranteed to be all set to '1'. The card is allowed up to two clocks to stop data transmission.

The timing for de-asserting the CS signal after the last card response is identical to a standard command/ response transaction as described in Figure 55;

- **Reading the CSD register**

The following timing diagram describes the SEND_CSD command bus transaction. The time-out values between the response and the data block is N_{CX} (Since the N_{AC} is still unknown).

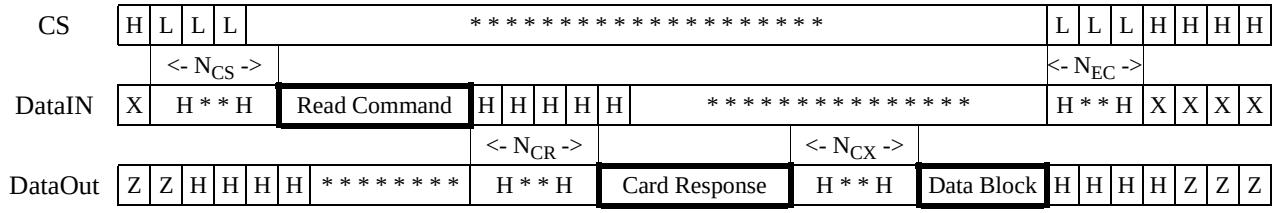


Figure 61: SPI Read CSD Register

7.8.3 Data write

- **Single Block Write**

The host may deselect a card (by raising the CS) at any time during the card busy period (refer to the given timing diagram). The card will release the DataOut line one clock after the CS going high. To check if the card is still busy it needs to be reselected by asserting (set to low) the CS signal. The card will resume busy signal

(pulling DataOut low) one clock cycle after the falling edge of CS.

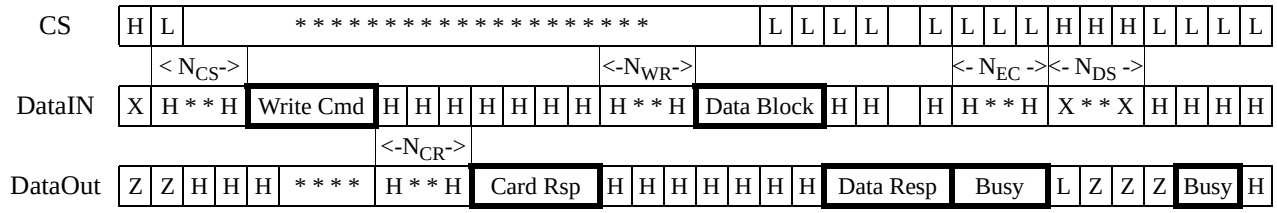


Figure 62: SPI Single Block Write

- **Multiple Block Write**

The timing behaviour of the multiple block write transaction starting from the command up to the first data block is identical to the single block write. Figure 63 describes the timing between the data blocks of a multiple block write transaction. Timing of the ‘Stop Tran’ token is identical to a standard data block. After the “Stop Tran” token is received ny the card, the data on the DataOut line is undefined for one byte (N_{BR}), after which a Busy token may appear. The host may deselect and reselect the card during every busy period between the data blocks. Timing for toggling the CS signal is identical to the Single block write transaction.

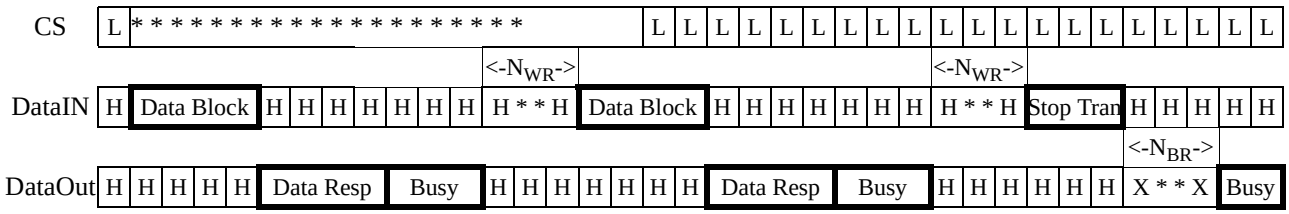


Figure 63: SPI Multiple Block Write

7.8.4 Timing Values

	Min	Max	Unit
N_{CS}	0	-	8 clock cycles
N_{CR}	1	8	8 clock cycles
N_{CX}	0	8	8 clock cycles
N_{RC}	1	-	8 clock cycles
N_{AC}	1	(10/8) * (TAAC * F_{OP} + 100 * NSAC)[1]	8 clock cycles
N_{WR}	1	-	8 clock cycles
N_{EC}	0	-	8 clock cycles
N_{DS}	0	-	8 clock cycles
N_{BR}	1	1	8 clock cycles

Table 43: Timing values

1) F_{OP} is the MMC clock frequency the host is using for the read operation.

7.9 SPI Electrical Interface

Identical to MultiMediaCard mode with the exception of the programmable card output drivers option, which is not supported in SPI mode.

7.10 SPI Bus Operating Conditions

Identical to MultiMediaCard mode.

7.11 Bus Timing

Identical to MultiMediaCard mode. The timing of the CS signal is the same as any other card input.

8 Error protection

8.1 Error Correction Codes (ECC)

In order to detect data defects on the cards the host may include error correction codes in the payload data. For error free devices this feature is not required. With the error correction implemented off card, an optimal hardware sharing can be achieved. On the other hand the variety of codes in a system must be restricted or one will need a programmable ECC controller, which is beyond the intention of a MultiMediaCard adapter.

If a MultiMediaCard requires an external error correction (external means outside of the card), then an ECC algorithm has to be implemented in the MultiMediaCard host. The DEFAULT_ECC field in the CSD register defines the recommended ECC algorithm for the card.

The shortened BCH (542,512) code was chosen for matching the requirement of having high efficiency at lowest costs. The following table gives a brief overview of this code:

Parameter	Value
Code type	shortened BCH (542,512) code
Payload block length	512 bit
Redundancy	5.5%
Number of correctable errors in a block	3
Codec complexity (error correction in HW)	encoding + decoding: 5k gates
Decoding latency (HW @ 20MHz)	< 30 microSec
Codec gatecount (error detection in HW, error correction in SW-only if block erroneous)	encoding + error detection: ~ 1k gates error correction: ~ 20 SW instructions/each bit of the erroneous block
Codec complexity (SW only)	encoding: ~ 6 instructions/bit error detection: ~ 8 instructions/bit error correction: ~ 20 instructions/each bit of erroneous block

As the ECC blocks are not necessarily byte-aligned, bit stuffing is used to align the ECC blocks to byte boundaries. For the BCH(542,512) code, there are two stuff bits added at the end of the 542-bits block, leading to a redundancy of 5.9%.

8.2 Cyclic Redundancy Codes (CRC)

The CRC is intended for protecting MultiMediaCard commands, responses and data transfer against transmission errors on the MultiMediaCard bus. One CRC is generated for every command and checked for every response on the CMD line. For data blocks one CRC per transferred block is generated. The CRC is generated and checked as described in the following.

- **CRC7**

The CRC7 check is used for all commands, for all responses except type R3, and for the CSD and CID registers. The CRC7 is a 7-bit value and is computed as follows:

generator polynomial: $G(x) = x^7 + x^3 + 1$.

$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + ... + (\text{last bit}) * x^0$

CRC[6...0] = Remainder [(M(x) * x^7) / G(x)]

All CRC registers are initialized to zero. The first bit is the most left bit of the corresponding bit string (of the command, response, CID or CSD). The degree *n* of the polynomial is the number of CRC protected bits decreased by one. The number of bits to be protected is 40 for commands and responses (n = 39), and 120 for the CSD and CID (n = 119).

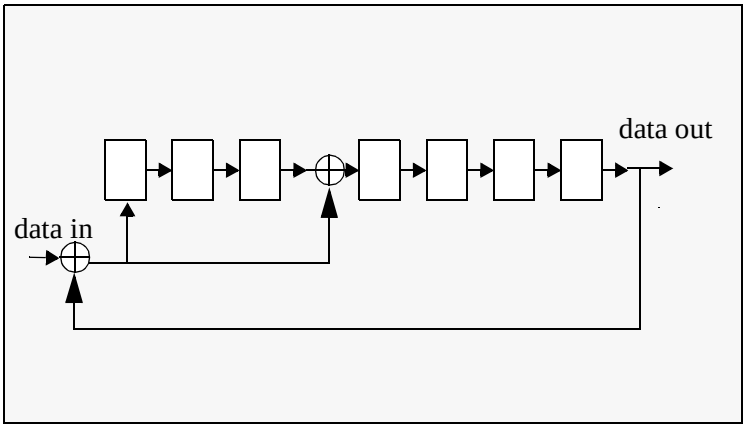


Figure 64: CRC7 Generator/Checker

- **CRC16**

The CRC16 is used for payload protection in block transfer mode. The CRC check sum is a 16-bit value and is computed as follows:

Generator polynomial $G(x) = x^{16} + x^{12} + x^5 + 1$

$M(x) = (\text{first bit}) \times x^n + (\text{second bit}) \times x^{n-1} + \ldots + (\text{last bit}) \times x^0$

$CRC[15\ldots0] = \text{Remainder}[(M(x) \cdot x^{16}) / G(x)]$

All CRC registers are initialized to zero. The first bit is the first data bit of the corresponding block. The degree *n* of the polynomial denotes the number of bits of the data block decreased by one (e.g. n = 4095 for a block length of 512 bytes). The generator polynomial $G(x)$ is a standard CCITT polynomial. The code has a

minimal distance d=4 and is used for a payload length of up to 2048 Bytes (n <= 16383).

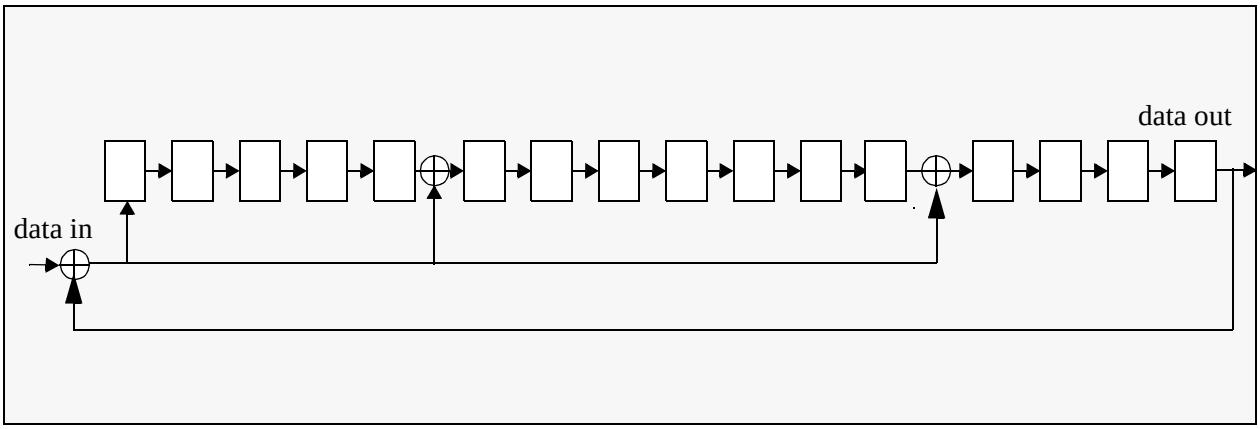


Figure 65: CRC16 Generator/Checker

9 MultiMediaCard Mechanical Specification

This chapter describes the mechanical and electromechanical features of the MultiMediaCard, as well as the minimum requirements of the MultiMediaCard connector. All technical drafts follow DIN ISO standard.

The functions of the card package are:

- **Protecting the chip(s)**
- **Easy handling for the end user**
- **Reliable electrical interconnection**
- **Bearing textual information and image**
- **Appealing appearance**

The functions of the connector are:

- **Attaching and fixing the card**
- **Electrical interconnection of the card to the system board**
- **Optional: switch on/off power supply**
- **Protection against card inverse insertion**

9.1 Card Package

Every card package shall have the characteristics described in the following sections.

9.1.1 External Signal Contacts (ESC)

Number of ESC	7
Distance from front edge	1 mm
ESC grid	2.5 mm
Contact dimensions	1.7mm x 3.5mm
Location of Via holes (on the contacts)	Not allowed on a 0.5mm stripe centered along the center line of the contact pad (refer to Figure 68)
Electrical resistance	30 mOhm (worst case: 100 mOhm)
Micro interrupts	< 0.1 μSec

9.1.2 Design and Format

Card Package Dimensions	Normal Size: 24mm x 32mm; (min. 23.9mm x 31.9mm; max.24.1mm x 32.1mm) other dimensions Figure 66 Testing according to MIL STD 883, Meth 2016
	Reduced Size: 24mm x 18mm; (min. 23.9mm x 17.9mm; max.24.1mm x 18.1mm) other dimensions Figure 67 Testing according to MIL STD 883, Meth 2016
Thickness	1.4mm +- 0.1mm
Restrictions on usage of package material	Some area of the external surface of the card edge may not contain conductive materials (refer to Figure 69)
Label or printable area	Whole card, except contact area
Surface	Plain (except contact area)
Edges	Smooth edges, see Figure 66
Inverse insertion	Protection on right corner (top view), see Figure 66
Position of ESC contacts	Along middle of shorter edge; -0.625mm Offset

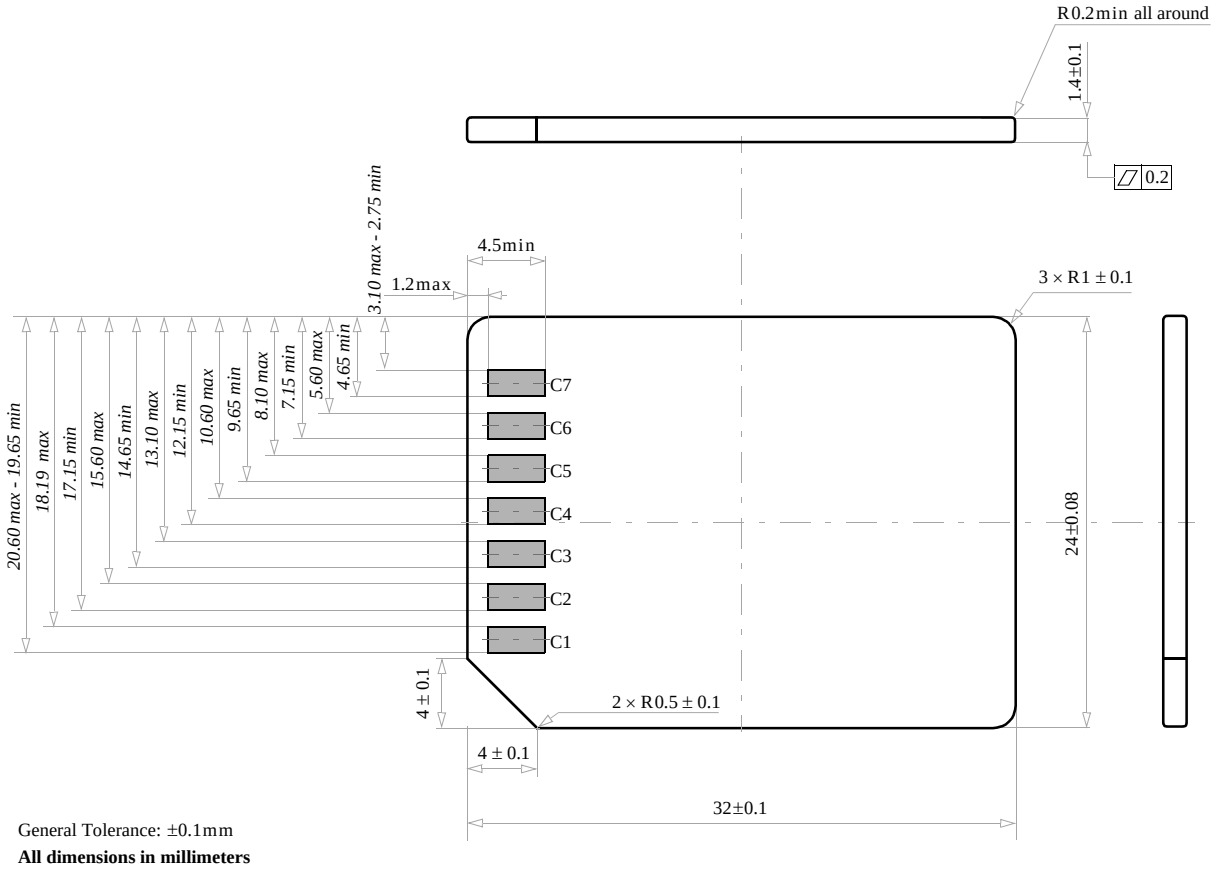


Figure 66: Dimensions Of A Normal Size MultiMediaCard (Bottom View, DIN)

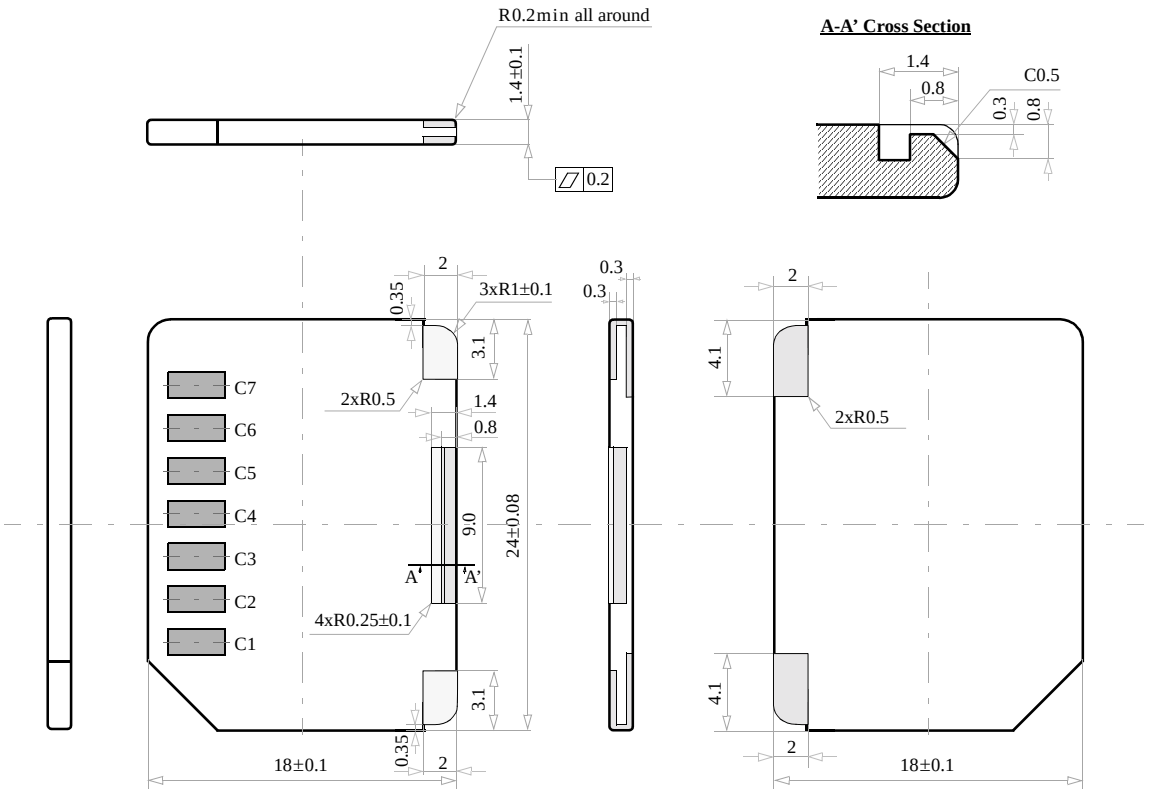


Figure 67: Dimensions Of A Reduced Size MultiMediaCard (DIN)

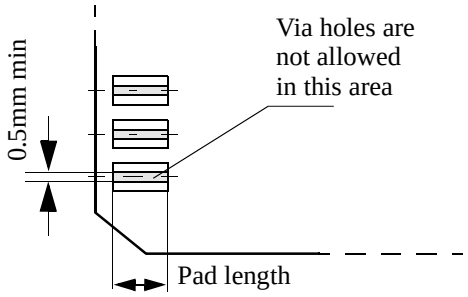


Figure 68: Location Of Pads Via Holes

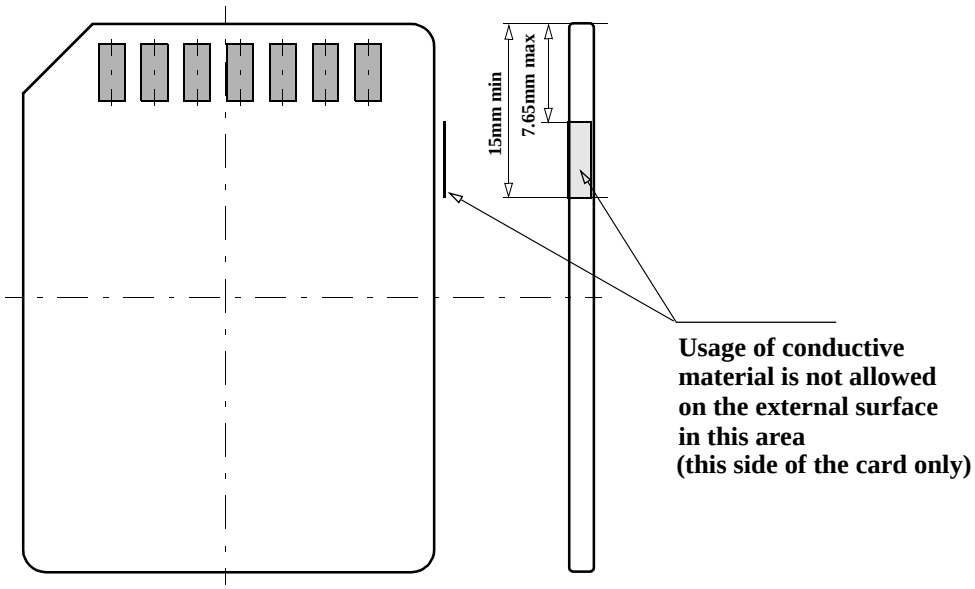


Figure 69: Conductive Material Usage Restrictions

9.1.3 Reliability and Durability

temperature	operation: -25°C / 85°C storage: -40°C (168h) / 85°C (500h) junction temperature: max. 95°C
moisture and corrosion	operation: 25°C / 95% rel. humidity stress: 40°C / 93% rel. hum./500h salt water spray: 3% NaCl/35C; 24h acc. MIL STD 883 Method 1009
ESD protection	Contact Pads: +/- 4kV, Human body model according to ANSI EOS/ESD-S5.1-1998 Non Contact Pads area: +/-8kV (coupling plane discharge) +/-15kV (air discharge) Human body model according to IEC61000-4-2
durability	10.000 mating cycles; test t.b.d.
bending	t.b.d.
torque	t.b.d.
drop test	1.5m free fall
UV light exposure	UV: 200nm, 15Ws/cm² according to ISO 7816-1
visual inspection shape and form	no warpage; no mold skin; complete form; no cavities surface smoothness sigma-0.1 mm/cm^2 within contour; no cracks; no pollution (fat, oil dust, etc.)

9.1.4 Quality Assurance

The product traceability shall be ensured by an individual card identification number.

9.2 System: Card and Connector

The description of the connector (especially of a multi-card connector) is out of the scope of this document. However, minimal requirements to the connector comprise the ability to guarantee hot insertion and removal of the card, and to prevent inverse insertion. An example for such a connector is described in Appendix A.3.

9.2.1 Card Hot Insertion

To guarantee a reliable initialization during hot insertion, some measures shall be taken on the host side. For instance, a special hot-insertion capable card connector may be used to guarantee the proper sequence of card pin connection. As another method, a switch could ensure that the power is switched on only after all card pads are contacted. Of course, any other similar mechanism is allowed. A possible connector realization is described in Appendix A.3.

9.2.2 Inverse Insertion

Inverse insertion is prevented by the reclining corners of the card and the connector.

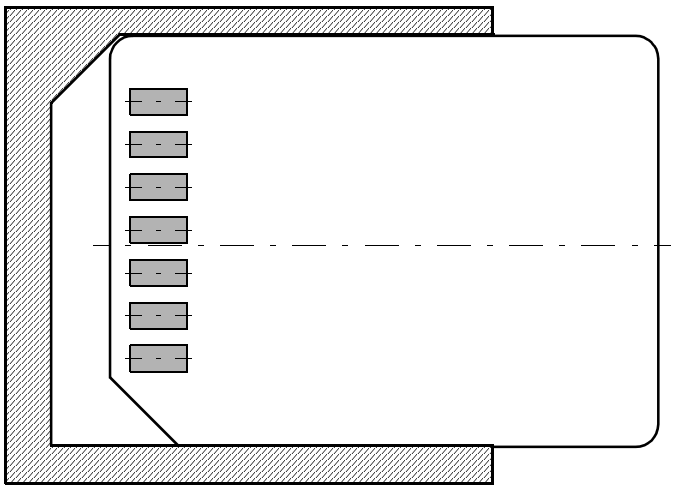


Figure 70: Inverse Insertion

9.2.3 Card Orientation

For the benefit of unified terminology when discussing the three dimensional orientation of a card (e.g. for connector definition), the non contact-pads side, is defined as the TOP side of the card.

10 MultiMediaCard Standard Compliance

The MultiMediaCard standard provides all the necessary information required for media exchangeability and compatibility.

- Generic card access and communication protocol (Chapter 4, Chapter 5).
- Electrical interface parameters, such as: power supply, peak and average current consumption and data transfer frequency (Chapter 6).
- The description of the optional SPI mode (Chapter 7).
- Data integrity and error handling (Chapter 8).
- Mechanical interface parameters, such as: connector type and dimensions and the card form factor (Chapter 9).
- Basic file formats for achieving high data interchangeability

However, due to the wide spectrum of targeted MultiMediaCard applications—from a full blown PC based application down to the very-low-cost market segments—it is not always cost effective nor useful to implement every MultiMediaCard standard feature in a specific MultiMediaCard system. Therefore, many of the parameters are configurable and can be tailored per implementation.

A card is compliant with the standard as long as all of its configuration parameters are within the valid range. A MultiMediaCard host is compliant as long as it supports at least one MultiMediaCard class as defined below. Card classes have been introduced in Chapter 3.1: Read Only Memory (ROM) cards, Read/Write (RW) cards and I/O cards. Every provider of MultiMediaCard system components is required to clearly specify (in its product manual) all the MultiMediaCard specific restrictions of the device.

MultiMediaCards (slaves) provide their configuration data in the Card Specific Data (CSD) register (refer to Chapter 5.3). The MultiMediaCard protocol includes all the necessary commands for querying this information and verifying the system concept configuration. MultiMediaCard hosts (masters) are required (as part of the system boot-up process) to verify host-to-card compatibility with each of the cards connected to the bus. The I/O card class characteristics and compliance requirements will be refined in coming revisions.

The following table summarizes the requirements from a MultiMediaCard host for each card class (CCC = card command class, see Chapter 4.7). The meaning of the entries is as follows:

- *Mandatory*: any MultiMediaCard host supporting the specified card class must implement this function.
- *Optional*: this function is an added option. The host is compliant to the specified card class without having implemented this function.
- *Not required*: this function has no use for the specified card class.

Function	ROM card class	R/W card class	I/O card class
0-20 MHz transfer rate	sub-range allowed	sub-range allowed	sub-range allowed
2-3.6 volts power supply	sub-range allowed	sub-range allowed	sub-range allowed
CCC 0 basic	mandatory	mandatory	mandatory
CCC 1 and 2 sequential and block read	one of the two mandatory, the other optional	one of the two mandatory, the other optional	optional
CCC 3 and 4 sequential and block write	not required	one of the two mandatory, the other optional	optional
CCC 5 erase	not required	mandatory	not required

Function	ROM card class	R/W card class	I/O card class
CCC 6 write protection functions	not required	mandatory	not required
CCC 7 lock card commands	optional	optional	optional
CCC 8 application specific commands	optional	optional	optional
CCC 9 interrupt and fast read/write	not required	optional	mandatory
DSR	optional	optional	optional
SPI Mode	optional	optional	optional
ECC generation and verification	optional	optional	not required

Comments on the optional functions:

- The interrupt command is intended for reducing the overhead on the host side required during polling for some events.
- The setting of the DSR allows the host to configure the MultiMediaCard bus in a very flexible, application dependent manner
- The external ECC in the host allows the usage of extremely low-cost cards.

11 File Formats for the MultiMediaCard

In general, MultiMediaCard data are structured by means of a file system. Since the definition of a MultiMediaCard file system is not part of the system specification, the user or content provider is free to choose any appropriate file system for the application. However, for achieving high data interchangeability, some basic conventions on how to identify the file system structure by the host may be desirable. For this reason, the basic mechanism for indicating the file system type has been introduced in Chapter 5.

Three basic types have been defined as valid file formats for the MultiMediaCard. The description of these formats will be given in the following sections.

11.1 Hard Disk-like File System with Partition Table

Similar to hard disks in PCs, the first data block of the memory consists of a partition table. Thus, using the same notation as for hard disks, i.e. partitioning the memory field into logical sectors of 512 bytes each, the first sector is reserved for this partition table. The data in this sector is structured as follows:

Byte position	Length (bytes)	Entry description	Value / Range
0x0	446	consistency check routine	
0x1be	16	partition table entry	(see below)
0x1ce	16	partition table entry	(see below)
0x1de	16	partition table entry	(see below)
0x1ee	16	partition table entry	(see below)
0x1fe	1	signature	’0x55’
0x1ff	1	signature	’0xaa’

Table 44: Partition table for hard disk-like file system

Every partition entry consists of the following fields:

Byte position	Length (bytes)	Entry description	Value / Range
0x0	1	boot descriptor	0x00 (non-bootable device), 0x80 (bootable device)
0x1	3	first partition sector	address of first sector
0x4	1	file system descriptor	0 = empty 1 = DOS 12-bit FAT < 16 MB 4 = DOS 16-bit FAT < 32 MB 5 = extended DOS 6 = DOS 16 bit FAT >= 32 MB 0x10-0xff = free for other file systems*
0x5	3	last partition sector	address of last sector
0x8	4	first sector position relative to beginning of device	number of first sector (linear address)
0xc	4	Number of sectors in partition	between 1 and maximal number of sectors on device

Table 45: Partition entry description

The descriptors marked by an asterisk are not used in DOS systems. Every DOS partition is based on a 12-bit, 16-bit FAT or VFAT respectively. All sector numbers are stored in Little-Endian format (least significant byte first). The start and end address of the partition are given in terms of heads, tracks and sectors, and can therefore be ignored for the MultiMediaCard, since the position of the partition can be determined by the last two entries.

The recommended default configuration in the boot sector is described in the following table:

Byte position	Length (bytes)	Entry description	Value / Range
0x0	3	Jump command	0xeb 0xXX 0x90
0x3	8	OEM name	XXX
0xb	2	Bytes / sector	512
0xd	1	Sectors / cluster	XXX (range: 1 - 64)
0xe	2	Reserved sectors (Number of reserved sectors at the beginning of the media including the boot sector)	1
0x10	1	Number of FAT's	2
0x11	2	Number of root directory entries	512
0x13	2	Number of sectors on media	XXX (depends on card capacity, if the media has more than 65535 sectors, this field is zero and the 'number of total sectors' is set)
0x15	1	Media descriptor	0xf8 (hard disk)
0x16	2	Sectors / FAT	XXX
0x18	2	Sectors / track	32 (no meaning)
0x1a	2	Number of heads	2 (no meaning)
0x1c	4	Number of hidden sectors	0
0x20	4	Number of total sectors	XXX (depends on capacity)
0x24	1	Drive number	0
0x25	1	Reserved	0
0x26	1	Extended boot signature	0x29
0x27	4	Volume ID or serial number	XXX
0x2b	11	Volume label	XXX (ASCII characters padded with blanks if less than 11 characters)
0x36	8	File system type	XXX (ASCII characters identifying the file system type FAT12 or FAT16)
0x3e	448	Load program code	XXX
0x1fe	1	Signature	0x55
0x1ff	1	Signature	0xaa

Table 46: Boot sector configuration

All 'X' entries are denoting card dependent or non-fixed values. The number of sectors per track and the number of heads are meaningless for the MultiMediaCard and can be ignored.

11.2 DOS FAT File System without Partition Table

For simple file systems, the partition table can be omitted by using only a boot block for a DOS FAT file system[1]. In this case, exactly the same boot block as recommended for the FAT in the previous section can be used.

11.3 Universal File System for the MultiMediaCard

T.B.D.

1. Note: this would not work with common software drivers on a PC, since a partition table is always required for hard disks

12 Abbreviations and terms

Block	a number of bytes, basic data transfer unit
Broadcast	a command sent to all cards on the MultiMediaCard bus
CID	Card IDentification number register
CLK	clock signal
CMD	command line or MultiMediaCard bus command (if extended CMDXX)
CRC	Cyclic Redundancy Check
CSD	Card Specific Data register
DAT	data line
DSR	Driver Stage Register
Flash	a type of multiple time programmable non volatile memory
Group	a number of write blocks, composite erase and write protect unit
LOW, HIGH	binary interface states with defined assignment to a voltage level
NSAC	defines the worst case for the clock rate dependent factor of the data access time
MSB, LSB	the Most Significant Bit or Least Significant Bit
OCR	Operation Conditions Register
open-drain	a logical interface operation mode. An external resistor or current source is used to pull the interface level to HIGH, the internal transistor pushes it to LOW
payload	net data
push-pull	a logical interface operation mode, a complementary pair of transistors is used to push the interface level to HIGH or LOW
RCA	Relative Card Address register
ROM	Read Only Memory
stuff bit	filling bits to ensure fixed length frames for commands and responses
SPI	Serial Peripheral Interface
TAAC	defines the time dependent factor of the data access time
three-state driver	a driver stage which has three output driver states: HIGH, LOW and high impedance (which means that the interface does not have any influence on the interface level)
token	code word representing a command
V_{DD}	+ power supply
V_{SS}	power supply ground

Appendix A: Application notes

A.1 Power Supply Decoupling

The $V_{SS1,}$ V_{SS2} and V_{DD} lines supply the card with operating voltage. For this, decoupling capacitors for buffering current peak are used. These capacitors are placed on the bus side corresponding to Figure 71.

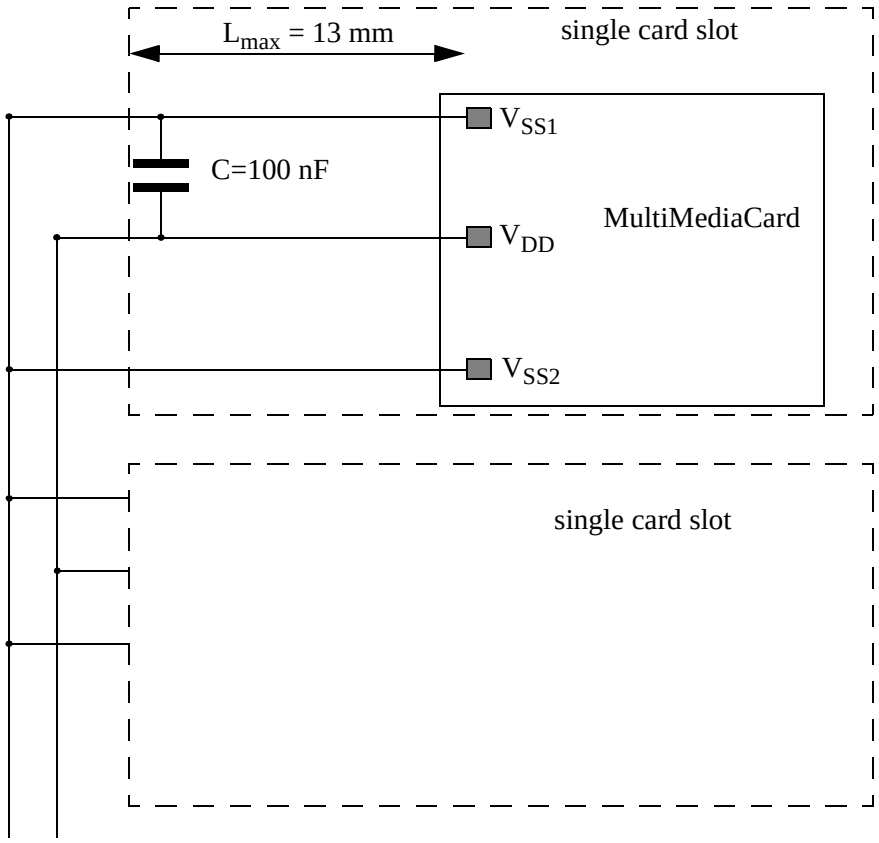


Figure 71: Power Supply Decoupling

The host controller includes a central buffer capacitor for V_{DD}. Its value is 1 µF/slot.

A.2 Payload Block Length and ECC Types Handling

There are two entries in the CSD register concerning the payload block length:

- block length type and
- external ECC.

The block length entry depends on the card memory field architecture. There are fixed values in 2-exponent steps defined for the block length size in the range 1 Byte - 2 kByte. Alternatively, the device allows application of any block length in the range between 1 Byte and the maximum block size.

The other CSD entry having an influence on the block length is the selected external ECC type. If there is an

external ECC code option selected, this entry generally does not have to match with the block length entry in the CSD. If these entries do not match, however, there is an additional caching at the host side required. To avoid that, using cards allowing the usage of any block length within the allowed range for applications with an external ECC is strongly recommended.

A.3 Connector

The connector described in this chapter serves as an example and is subject to further changes.

A.3.1 General

The connector housing which accommodates the card is formed of plastic. Inside are 7 contact springs for contacting the pads of the inserted card. Testing procedures are performed according to DIN IEC 68.

A.3.2 Card Insertion and Removal

Insertion of the MultiMediaCard is only possible when the contact area of the card and the contact area of the connector are in the correct position to each other. This is ensured by the reclining corners of the card and the connector, respectively.

To guarantee a reliable initialization during hot insertion, some measures must be taken on the host side. One possible solution is shown in Figure 72. It is based on the idea of a defined sequence for card contact connection during the card insertion process. The card contacts are contacted in two steps:

1. ground V_{SS1} (pin 3) and supply voltage V_{DD} (pin 4)

2. others (CLK, CMD, DAT, V_{SS2} and R_{SV})

Pins 3 and 4 should make first contact when inserting and release last when extracting.

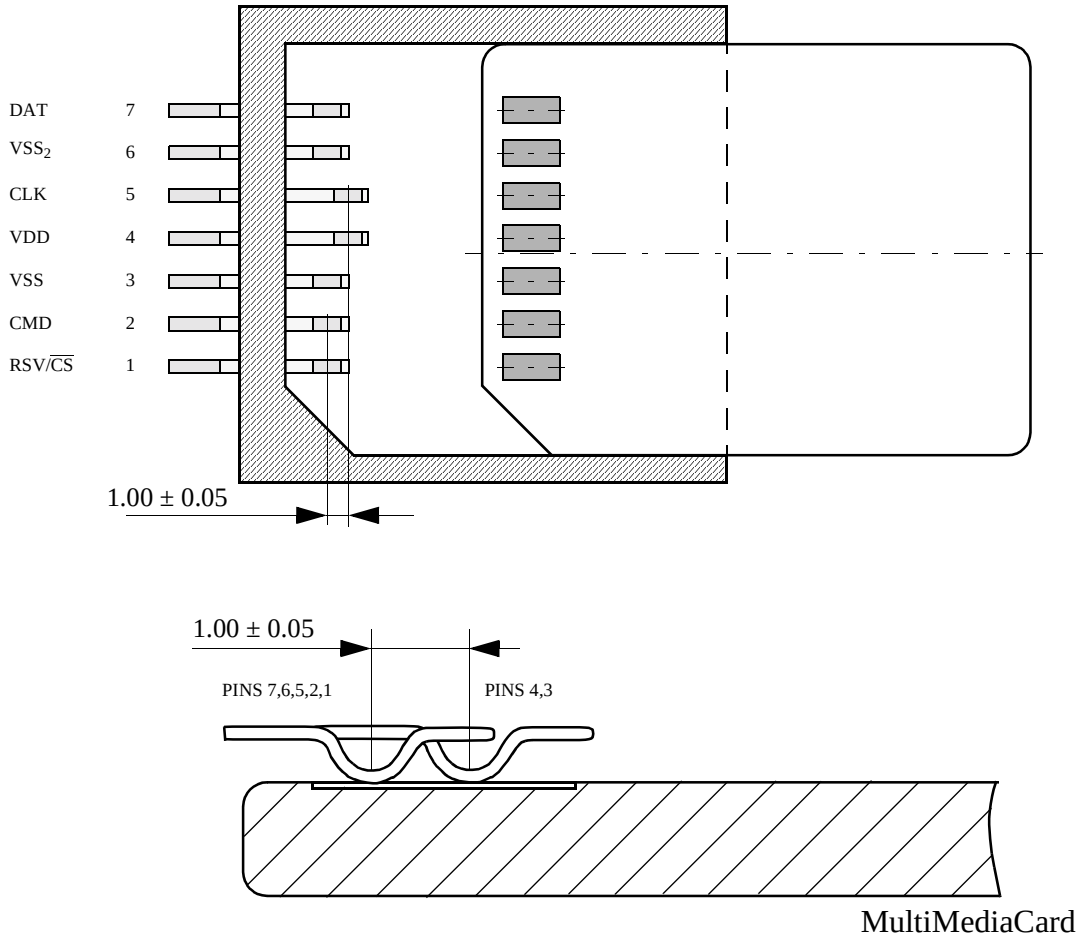


Figure 72: Modified MultiMediaCard Connector For Hot Insertion

A.3.3 Characteristics

The features described in the following must be considered when designing a MultiMediaCard connector. The given values are typical examples.

- **Mechanical Characteristics**

-	Max. number of mating operations	> 10000	
-	Contact force	0.2.....0.6 N	
-	Total pulling force	min.2 N	DIN IEC 512 part 7
-	Total insertion force	max. 40 N	DIN IEC 512 part 7
-	Vibration and High Frequency		
	- Mechanical frequency range	10.....2000 Hz	DIN IEC 512 part 2 and 4

- Acceleration 2 g
- Shock:
- Acceleration 5 g

- **Electrical Characteristics**

DIN IEC 512

- Contact resistance 100 mOhm
- Current carrying capacity at 25°C 0.5 A
- Insulation resistance > 1000 MOhm, > MOhm after test
- Operating voltage 3.3 V
- Testing voltage 500 V
- Operating current 100 mA max.

- **Climatic Characteristics**

DIN IEC 512 part 6-9

- Operating temperature -25°C.....90°C
- Storage temperature -40°C.....90°C
- Humidity 95% max. non condensing

A.4 Description of method for storing passwords on the card

In order to improve compatibility and inter-operability of the card between different applications it is required that different host application will use identical algorithms and data formats. Following is a recommended way of storing passwords in the 128-bit password block on the card. It is provided as application note only.

This method is applicable only if the password consists of text, possibly entered by the user. The application may opt to use another method if inter-operability between devices is not important, or if the application chooses to use, for example, a random bit pattern as the password.

- Get the password (from the user, from a local storage on the device, or something else). The password can be of any length, and in any character set.
- Normalize the password into UTF-8 encoded Unicode character set. This guarantees inter-operability with all locales, character sets and country-specific versions. In UTF-8, the first 128 characters are mapped directly to US-ASCII, and therefore a device using only US-ASCII for the password can easily conform to this specification.
- Run the normalized password through SHA-1 secure hash algorithm. This uses the whole key space available for password storage, and makes it possible to use also longer passwords than 128 bits. As an additional bonus, it is not possible to reverse-engineer the password from the card, since it is not possible to derive the password from its hash.
- Use the first 128 bits of this hash as the card password. (SHA-1 produces a 160-bit hash. The last 32 bits are not used.)

Following is an example (note that the exact values need to be double-checked before using this as implementation reference):

The password is “foobar”. First, it is converted to UTF-8. As all of the characters are US-ASCII, the resulting bit string (in hex) is

```
66 6F 6F 62 61 72
```

After running this string through SHA-1, it becomes

```
88 43 d7 f9 24 16 21 1d e9 eb b9 63 ff 4c e2 81 25 93 28 78
```

Of which the first 128 bits are

```
88 43 d7 f9 24 16 21 1d e9 eb b9 63 ff 4c
```

Which is then used as the password for the card.

UTF-8 is specified in *UTF-8, a transformation format of Unicode and ISO 10646*, RFC 2044, October 1996. ftp://ftp.nordu.net/rfc/rfc2044.txt

SHA-1 is specified in *Secure Hash Standard*, Federal Information Processing Standards Publication (FIPS PUB) 180-1, April 1995. http://www.itl.nist.gov/fipspubs/fip180-1.htm

A.5 MultiMediaCard Macro Commands

This section defines the way complex MultiMediaCard bus operations (e.g. card stack initialization, erase, read, etc.) may be executed using predefined command sequences. Executing these sequences is the responsibility of the MultiMediaCard bus master. Nevertheless, it may be used for host compatibility test purposes.

Mnemonic	Description
CIM_UPDATE_ACQ	Starts an identification cycle of a card stack. The card management information in the controller will be updated. New cards will be initialized; old cards keep their configuration. At the end all active cards are in Stand-by state.
CIM_SINGLE_CARD_ACQ	Starts an identification cycle of a single card.
CIM_INIT_STACK	Sends all cards to idle-state and starts a new identification cycle using the acquisition sequence.
CIM_CHECK_STACK	Tries to read the CSD of every active card in the stack. As a result, the card management information in the controller is updated.
CIM_SETUP_CARD	Select a card by writing the RCA and reads its CSD.
CIM_STREAM_READ	Sets the start address and reads a continuous stream of data from the card.
CIM_READ_BLOCK	Sets the block length and the starting address and reads a data block from the card.
CIM_READ_MBLOCK	Sets the block length and the starting address and reads (continuously) data blocks from the card. Data transfer is terminated by a stop command.
CIM_WRITE_BLOCK	Sets the block length and the starting address and writes a data block from the card.
CIM_WRITE_MBLOCK	Sets the block length and the starting address and writes (continuously) data blocks to the card. Data transfer is terminated by a stop command.
CIM_ERASE_GROUP	Erases a range of erase groups on the card.

Table 47: Macro commands

The MultiMediaCard command sequences are described in the following paragraphs. Figure 73 provides a legend for the symbols used in the sequence flow charts.

The status polling by CMD13 can explicitly be done any time after a response to the previous command has been received.

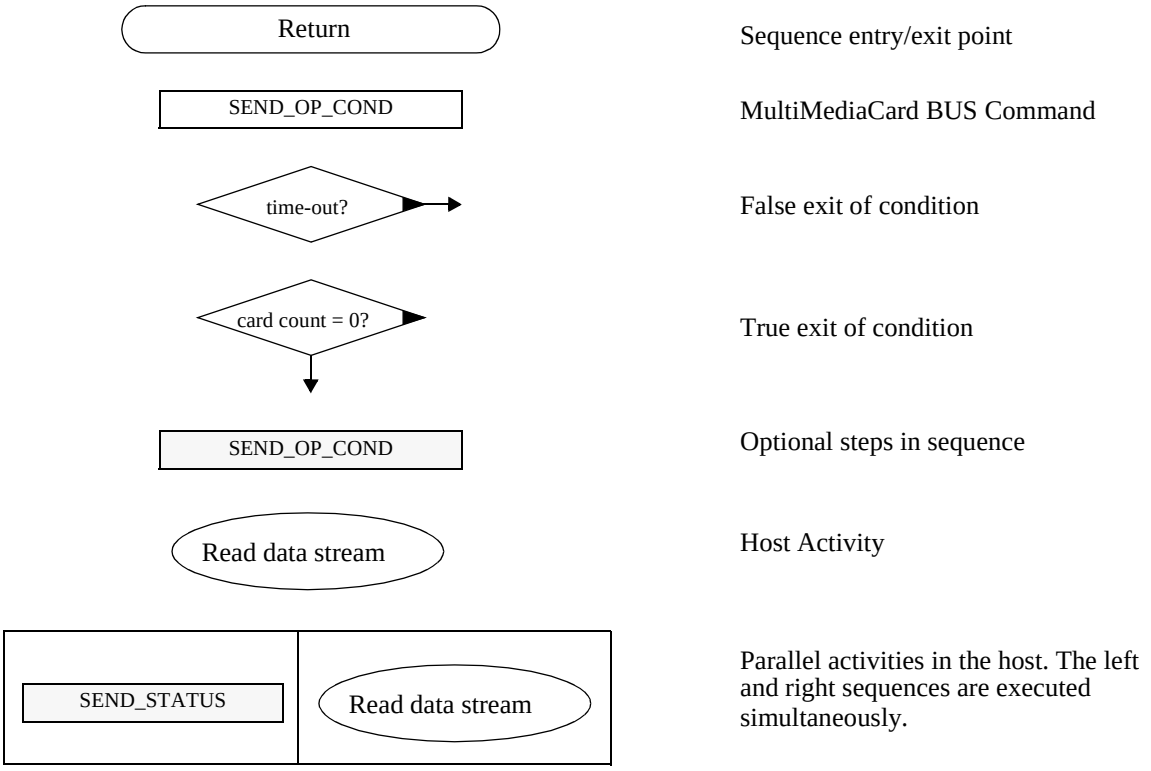


Figure 73: Legend For Command Sequences' Flow Charts

- CIM_UPDATE_ACQ

The update acquisition procedure (CIM_UPDATE_ACQ) of a card stack is shown in Figure 74. This update of card management information starts with the SEND_OP_COND command (CMD1), with no voltage range specified. This will enable the host to get the common voltage range for all cards. Next, the host will repeat sending its required voltage range until all cards are not busy. Cards supporting the required voltage range will respond and move to the *Ready State*. All other cards will move to the *Inactive State.*

With ALL_SEND_CID (CMD2) the host starts to scan all active cards in the stack. All not yet configured cards will respond to this command simultaneously. Because of the open drain bus operation mode only one card can successfully respond its complete card identification number (CID). The host assigns this card with a relative card address (RCA) using SET_RELATIVE_ADDR command (CMD3).

The sequence CMD2, CMD3 is repeated until all cards are identified. Optionally, the host can compare the number of already acquired cards with the number of slots in the actual application and finish the acquisition or set an error condition. Identified cards do not respond to CMD2 again. Since the host does not know the number of cards in the stack a time-out condition occurs after the last card had been identified. This triggers the host to end this routine and go into data transfer mode. The bus drivers go from open-drain into push-pull mode.

With no cards in the stack the acquisition procedure stops. Otherwise, the host will start another loop with SEND_CSD (CMD9). CMD9 reads the CSD register and can adapt its bus controller to the cards' configuration. The frequency remains at 400 kHz max. since a host must scan first the CSD of all cards before it can choose a common denominator.

After the last card has been configured, SET_DSR (CMD4) completes the identification procedure. The host

sends the driver stage register (DSR) information and configures all cards' driver stages if applicable.

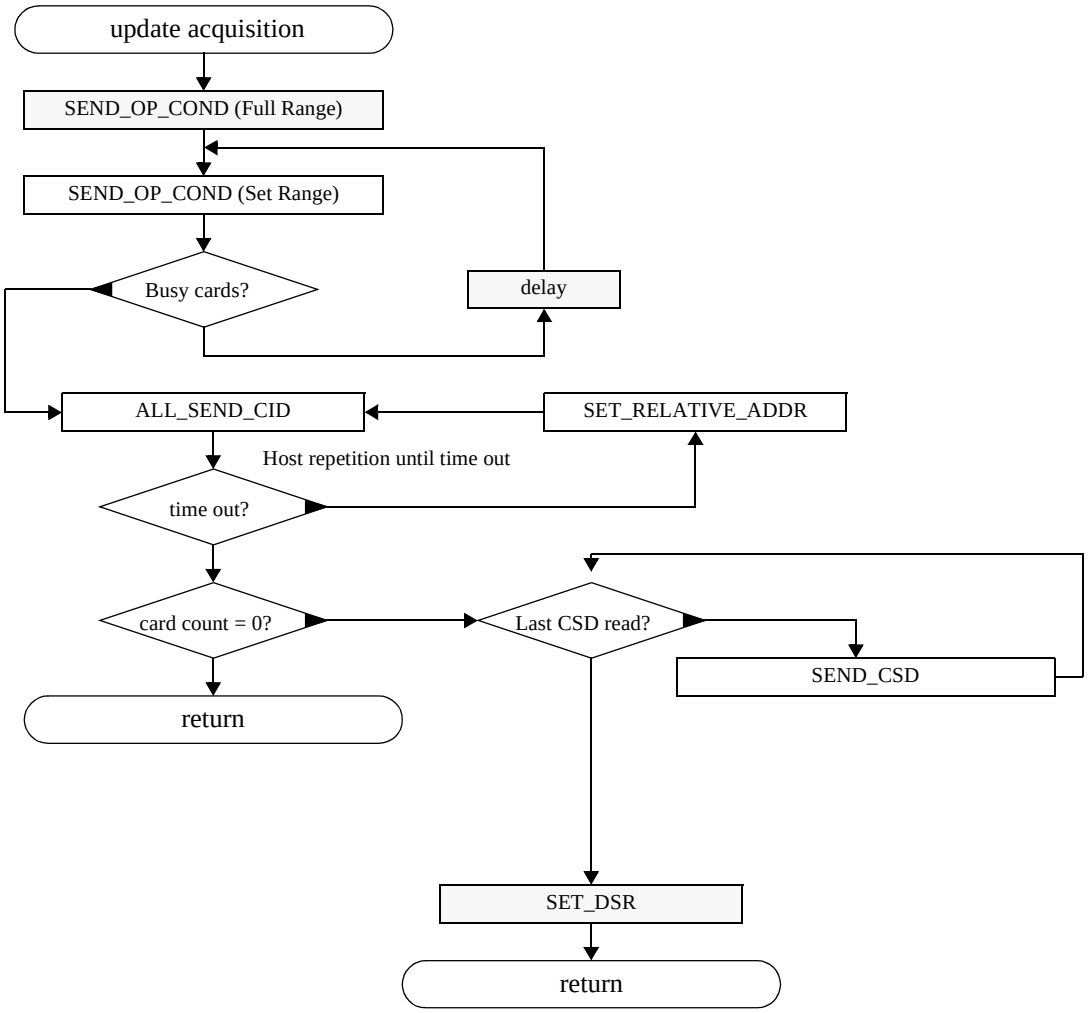


Figure 74: CIM_UPDATE_ACQ

- CIM_SINGLE_CARD_ACQ

The concept of a single card acquisition is similar to the acquisition of a card stack. The difference is that the host knows that there is a single card in the system and, therefore, does not have to implement the identification loop. In this case only one ALL_SEND_CID is required.

Similarly, a single SEND_CSD is sufficient.

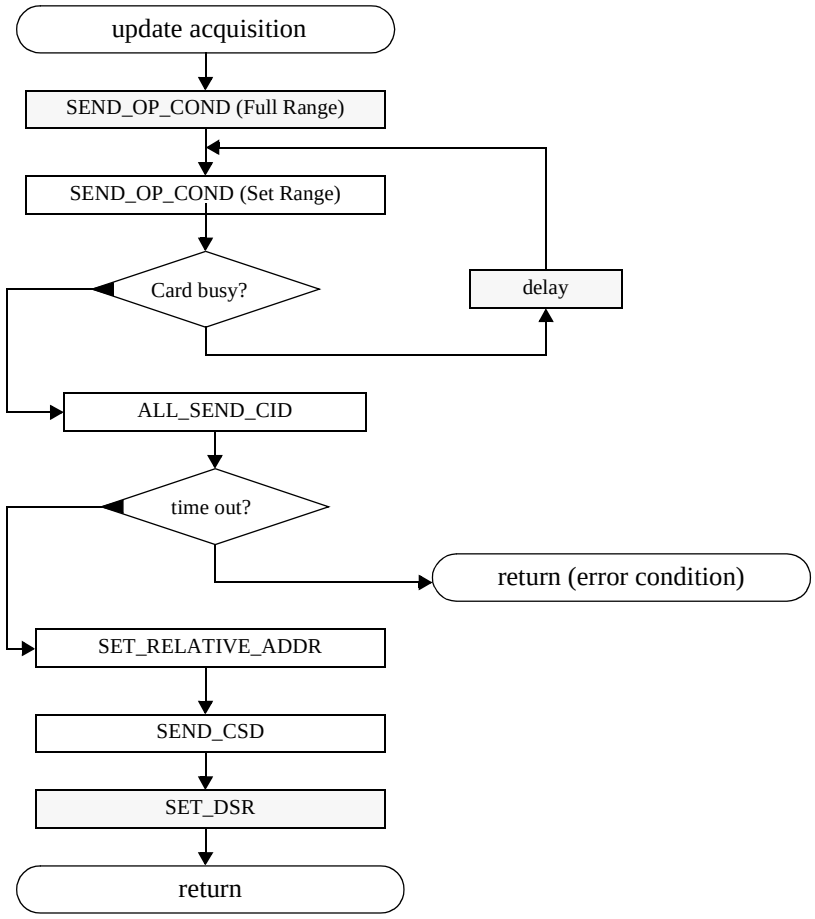


Figure 75: CIM_SINGLE_CARD_ACQ

- CIM_INIT_STACK

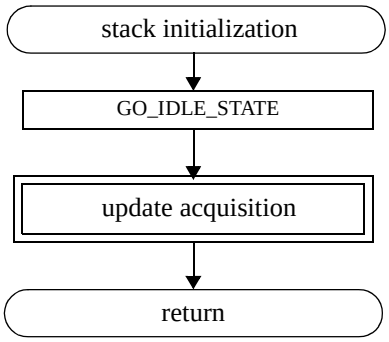


Figure 76: CIM_INIT_STACK

The identification procedure CIM_INIT_STACK of a card stack application is shown above.

The identification starts with the reset command GO_IDLE_STATE (CMD0). This clears all previous settings of all cards (except the inactive cards).

The next steps are identical to the update acquisition procedure.

- CIM_CHECK_STACK

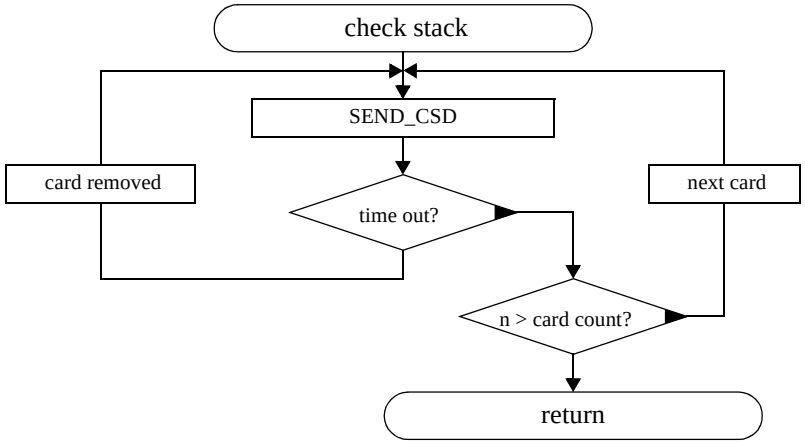


Figure 77: CIM_CHECK_STACK

The check stack procedure (CIM_CHECK_STACK) confirms the entries in the adapter card stack management table. The procedure starts by reading the CSD information with SEND_CSD (CMD9) command of the first card entry. If the card replies and it was not the last card, the loop continues with the next card. If a time out occurred, then the card has been removed (note: new cards cannot be detected this way).

- CIM_SETUP_CARD

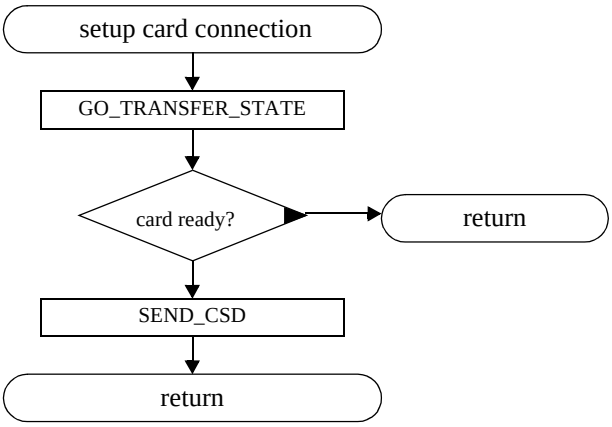


Figure 78: CIM_SETUP_CARD

The setup card connection procedure (CIM_SETUP_CARD) links the bus master with a single card. The argument required for this command is the RCA of the chosen card. The adapter cross checks with the internal stack management information if a card still exists in that slot. A single card is selected with GO_TRANSFER_STATE (CMD7) command by its RCA. The response indicates whether the card is ready or not. If the card confirms the connection, the adapter will read the card specific data with SEND_CSD (CMD9). The information within the response is used to configure the data path and controller options.

- CIM_STREAM_READ

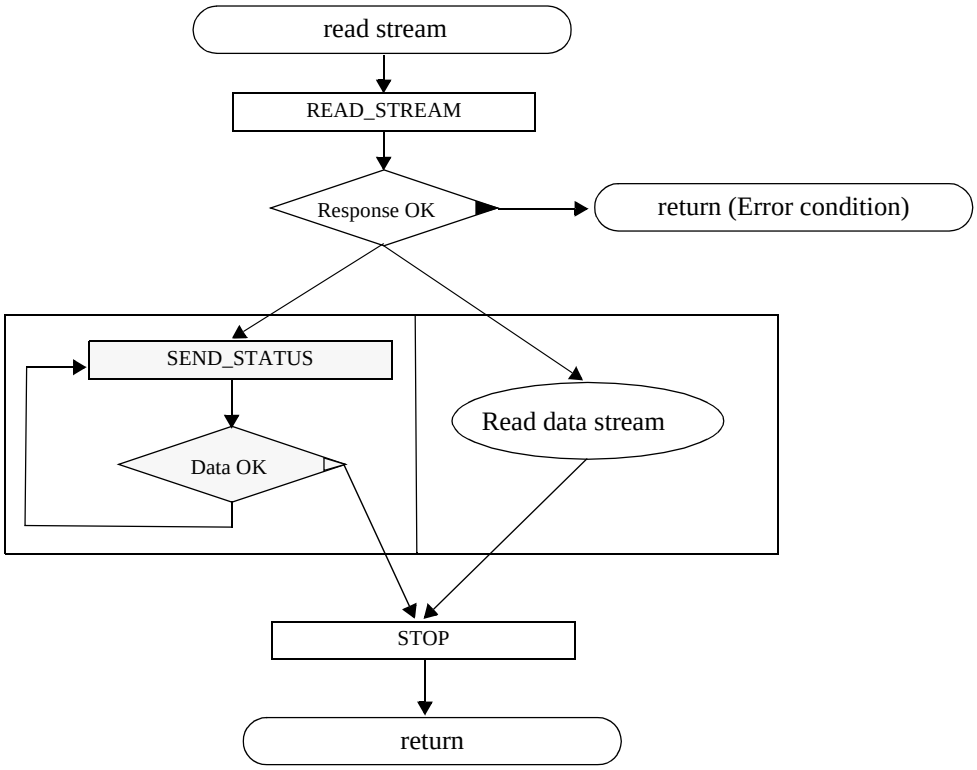


Figure 79: CIM_STREAM_READ

The sequence of stream read starts with the STREAM_READ (CMD11) command. If the card accepts the command it will send the data out on the DAT line and the host will read it. While reading the data line the host may send SEND_STATUS (CMD13) commands to the card to poll any new status information the card may have (e.g. UNDERRUN).

When the host has read all the data it needs or the card is reporting an error, the host will stop data transmission using the STOP (CMD12) command.

- CIM_READ_BLOCK

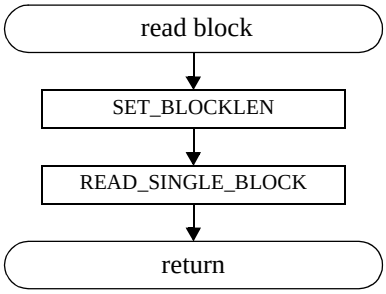


Figure 80: CIM_READ_BLOCK

The read block procedure (CIM_READ_BLOCK) reads a data block from a card. The arguments required for this command are the block length (4 bytes) and the starting address of the block (4 bytes). This operation also

includes a data portion (in this case, the read block). The procedure starts by setting the required block length with the SET_BLOCKLEN (CMD16) command. If the card accepts this setting, the data block is transferred via command READ_SINGLE_BLOCK (CMD17), starting at the given address.

- CIM_READ_MBLOCK

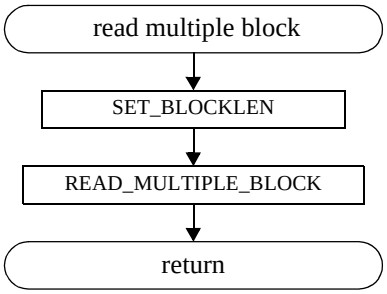


Figure 81: CIM_READ_MBLOCK

The read multiple block procedure (CIM_READ_BLOCK) sequentially reads blocks of data from a card. The arguments required for this command are the block length (4 bytes) and the starting address of the first block (4 bytes). This operation also includes a data portion (in this case, the read blocks). The procedure starts by setting the required block length with the SET_BLOCKLEN (CMD16) command. If the card accepts this setting, the data blocks are transferred via command READ_MULTIPLE_BLOCK (CMD18), starting at the given address.

- CIM_WRITE_BLOCK

This command sequence is similar to multiple block write except that there is no repeat loop for write data block.

- CIM_WRITE_MBLOCK

The sequence of write multiple block starts with an optional SET_BLOCK_LEN command. If there is no change in block length this command can be omitted. If the card accepts the two starting commands the host will begin sending data blocks on the data line.

After each data block the host will check the card response on the DAT line. If the CRC is OK, the card is not busy and the host will send the next block if there are more data blocks.

While sending data blocks, the host may query the card status register (using the SEND_STATUS conned) to poll any new status information the card may have (e.g. WP_VIOLATION, MISALIGMENT, etc.)

The sequence must be terminated with a STOP command.

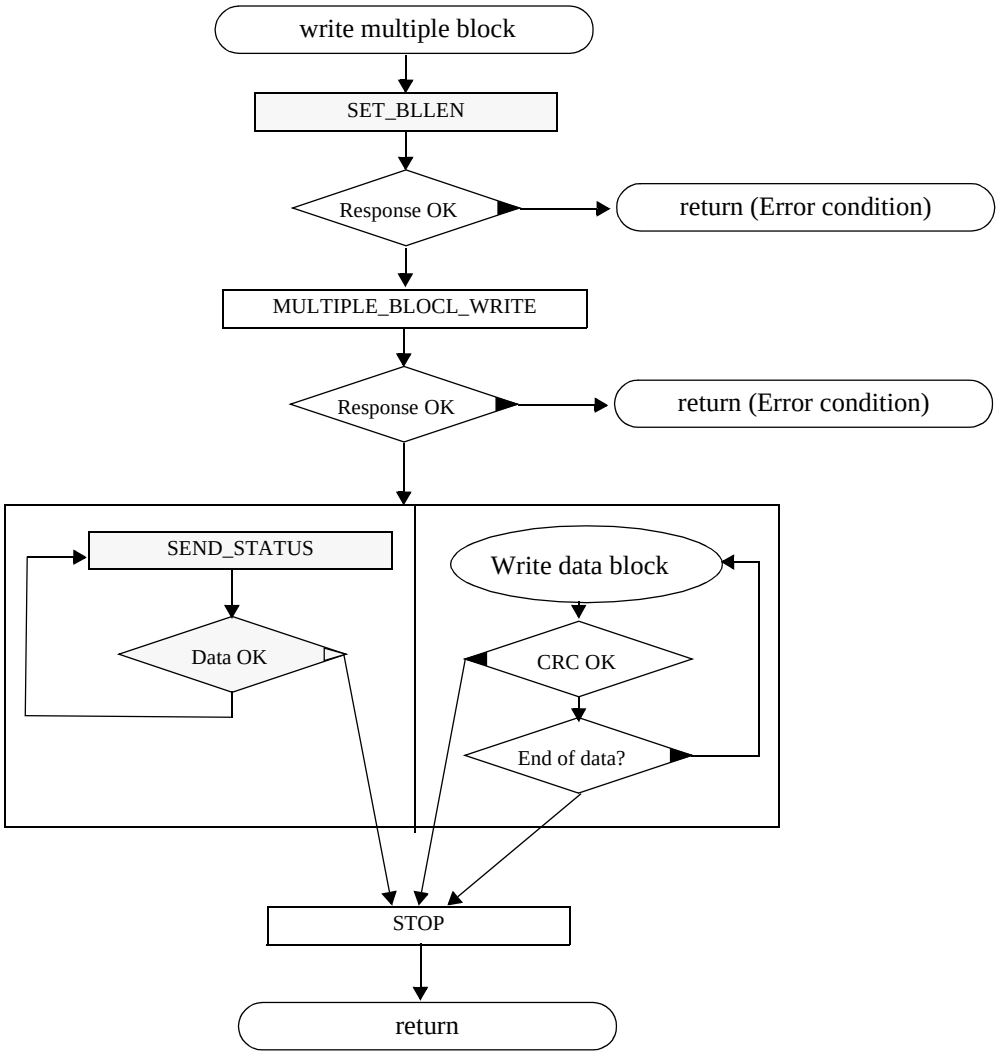


Figure 82: CIM_WRITE_MBLOCK

- CIM_ERASE_GROUP

The erase group procedure starts with ERASE_START (CMD35) and ERASE_END (CMD336 commands. Once the erase groups are selected the host will send an ERASE (CMD38) command. It is recommended that the host terminates the sequence with a SEND_STATUS (CMD13) to poll any additional status information the card may have (e.g. ERASE_WP_SKIP, etc.).

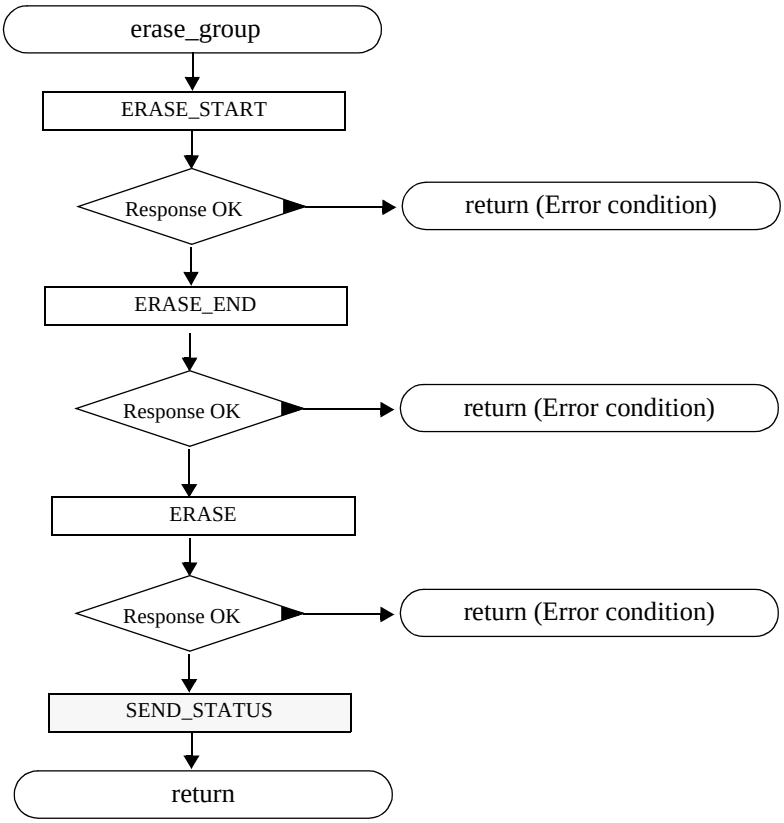


Figure 83: CIM_ERASE_GROUP

Appendix B: Changes Between System Specification Versions

B.1 Changes from version 1.4 to 2.0

The following major changes have been introduced in version 2.0, among other minor changes and corrections:

- **CID structure**

A new CID structure has been introduced (see Chapter 5). Not backward compatible to Vers. 1.4.

- **System specification version number**

The former CSD field MMC_PROT has been replaced by the field SPEC_VERS, reflecting the actual system specification version number (see Chapter 5). Backward compatible to Vers. 1.4.

- **Write protection class**

The former three write protection classes have been combined to one class. The other two classes are reserved (see Chapter 4). Not backward compatible to Vers. 1.4.

- **Response time-out in SPI mode**

The maximum value for response time-out (N_{CR}) in SPI mode has been increased to 8 (was 2 before, see Chapter 7). Not backward compatible to Vers. 1.4.

- **New command in SPI mode**

A new command (CMD58) has been introduced in SPI mode allowing the user to read the OCR register (see Chapter 4).

B.2 Changes from version 2.0 to 2.11

The following major changes have been introduced in version 2.11, among other minor changes and corrections:

- **New password lock command**

The password protection feature enables the host to lock a card while providing a password (see Section 4.4.5)

- **Application specific commands**

To enable a common way of implementing these features, two types of generic commands have been defined (see Section 4.4.6)

- **Equations for maximum clock frequency**

The equations for the maximum clock frequency in stream operations have been corrected (see Chapter 4).

- **Time-out for SEND_CSD command**

The time-out value of the SEND_CSD command in SPI mode has been set to a default value (was not properly defined before).

- **Minimum clock cycles for CS in SPI mode**

The minimum clock cycles of CS after the last bit of an SPI transaction has been defined.

- **Data interchange format**

A basic mechanism for indicating the file system type has been introduced in Chapter 5 and explained in more

detail in Chapter 11.

- **Change of card dimensional tolerances**

A backward-compatible definition of minimum contact areas and their distance to the upper and left card edge has been introduced (see Chapter 9).

B.3 Changes from version 2.11 to 2.2

- **Card level ESD tolerance definition**

The ESD tolerance definition of the cards was enhanced to include a card level tolerance definition. Previous versions of the standard defined the ESD requirements on the card pads only (see Chapter).

- **Card status in SPI mode**

CSD-overwrite and **card-is-locked** status bits where added to the card status register in SPI mode. In previous versions they where defined for MMC mode only (see Chapter 7.6.2).

- **Card Orientation**

A definition of the top side of the card was added to the standard (see Chapter 9.2.3).

B.4 Version 3.0

Version 3.0 (dated May 2001) was released with errors in the CSD_STRUCTURE and SPEC_VER fields of the CSD register. As a results it introduced ambiguities in the process of identifying the protocol version of an MMC card. version 3.1 (dated June 2001) is identical to 3.0 with the exception of fixing these errors. MMCA obsoletes version 3.0 of the MMCA standard specification and encourages designers not to use it.

B.5 Changes from version 2.2 to 3.1

- **Mechanical Specification:**
  - The allowed contact force of the MMC connector was increased (See Chapter A.3.3)
  - The allowed tolerance of the distance from the external edges of pads #1 and #7 to the adjacent edge of the card was decreased (see Chapter 9.1)
- **Electrical Specification:**
  - Definition of data valid and non valid timing relative to the bus clock was made clearer (See Figure 41; Section 6.7)
  - A second, Low-Voltage MultiMediaCard was introduced This change is not completely backwards compatible (see Section 4.2.2 & Section 6.6).
- **Card Registers:**
  - The format of the R2W_FACTOR field in the CSD was changed (See Section 5.3)
  - The format of the fields describing the size of card erase group was changed (See Section 5.3)
- **MMC protocol:**
  - A definition of read ahead error condition was added (See Section 4.6.3)
  - Timing of STOP_TRANSMISSION command (CMD12) was made clearer (See Figure 28, Section

4.12.2; Figure 31 and Figure 34, Section 4.12.3)

- A second type, using pre-defined number of blocks, of multiple block read/write transactions was added. (See Section 4.4.1 & Section 4.4.2)
- Erase function was changed. This is a non Backwards compatible change. (See Section 4.4.3), This change is applicable for SPI mode as well.
- The definition for Force erase operation on locked cards was completed (See Section 4.4.5). This change is applicable for SPI mode as well.

- **SPI protocol:**
  - MultiMediacard can be switched to SPI mode from *Inactive* state as well (See Section 4.2.1& Section 7.5.1)
  - A definition of the level of control the SPI host has on the SPI bus clock was added (See Section 7.5.8)
  - The definition of the error conditions in SPI mode was enhanced (See Section 7.5.9)
  - A typo in the timing diagram of read CSD/CID registers transaction was fixed (See Figure 61 & Section 7.8.4)
  - Added Support for Multiple block read and write (See Section 7.5.3 & Section 7.5.4)
- **Application notes:**
  - A description of a recommended way to use card lock password was added (See Section A.4)

B.6 Changes from version 3.1 to 3.2

Revision 3.2 of the MMCA standard spec includes only minor changes adding to the clarity and consistency of the sepc and conform to the actual design of the 3.1 compatible cards in the market. There are no functional additions nor changes to the spec. Therefore, the definition of the version codes of rev 3.1 (as defined in the CSD register) where extended to include version 3.2 as well.

There is one exception to this statement, the definition of the Low Voltage MultiMediaCard card (in version 3.1 it was defined as TBD). Low Voltage MultiMediaCard card must be compliant to version 3.2 (or higher) of the MMCA standard.

- **Mechanical Specification:**
  - In order to ensure connector / card compatibility, a couple of mechanical implementation requirements where added (refer to Section 9.1.1& Section 9.1.2).
- **Electrical Specification:**
  - The definition of the OCR register and the I/O signal levels for the Low Voltage MultiMediaCard was completed (refer to Section 5.1 and Section 6.6 respectively).
- **Card Registers:**
  - Revision 3.2 was added to the CSD_STRUCTURE and SPEC_VERS fields of the CSD register. Encoding of the new version is using the same codes of rev 3.1.
  - OCR bits 7:0 redefined; OCR bit 7 changed to represent 1.65 - 1.95V range, bits 6..0 changed back to reserved.

- **MMC and SPI protocol:**
  - The definition of the maximum read latency in the timing diagrams (see Table 19) was changed to match the text (see Section 4.6.2)
  - The relationship between the availability of erase and group write protect command classes and the MultiMediaCard type (e.g. Read/Write, ROM, I/O) was made less ambiguous. They where defined as mandatory for Read/Write cards.
  - Bits 16 of the CSD was allocated for content protection application (Refer to Section 5.3)
  - A clearer definition of the response type to CMD12, in MMC mode, is given (Refer to Table 6).

B.7 Changes from version 3.2 to 3.3

The major change in version 3.3 is the introduction of the Reduced Size MMC, the definition is mainly mechanical (See Figure 67 & Section 3.1.1 & Section 9.1.2).

The stream read and write stream formulas were split to allow the calculation of error-free stream clock frequencies. See ’Stream Read’ on page 35 and ’Stream Write’ on page 37.

The high voltage range, 2.7-3.6 is made mandatory for all cards, no partial range support is allowed anymore. Every card must implement the OCR register and CMD1 so the host can correctly identify low voltage cards.

Other changes are minor typos:

- Define open-drain type for interrupt mode in Table 3
- Define ‘wired-and’ connectivity for the OCR register, see Section 5.1
- Rephrase the paragraph after Figure 25 for clearer understanding