JEDEC STANDARD

Embedded MultiMediaCard (eMMC) eMMC/Card Product Standard, High Capacity, including Reliable Write, Boot, and Sleep Modes

(MMCA, 4.3)

JESD84-A43

NOVEMBER 2007

JEDEC SOLID STATE TECHNOLOGY ASSOCIATION

JEDEC

MULTIMEDIACARD ASSOCIATION

NOTICE

JEDEC standards and publications contain material that has been prepared, reviewed, and approved through the JEDEC Board of Directors level and subsequently reviewed and approved by the JEDEC legal counsel.

JEDEC standards and publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for use by those other than JEDEC members, whether the standard is to be used either domestically or internationally.

JEDEC standards and publications are adopted without regard to whether or not their adoption may involve patents or articles, materials, or processes. By such action JEDEC does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the JEDEC standards or publications.

The information included in JEDEC standards and publications represents a sound approach to product specification and application, principally from the solid state device manufacturer viewpoint. Within the JEDEC organization there are procedures whereby a JEDEC standard or publication may be further processed and ultimately become an ANSI standard.

No claims to be in conformance with this standard may be made unless all requirements stated in the standard are met.

Inquiries, comments, and suggestions relative to the content of this JEDEC standard or publication should be addressed to JEDEC at the address below, or call (703) 907-7559 or www.jedec.org

Published by

©JEDEC Solid State Technology Association 2007

2500 Wilson Boulevard

Arlington, VA 22201-3834

PRICE: Please refer to the current

Catalog of JEDEC Engineering Standards and Publications online at

http://www.jedec.org/Catalog/catalog.cfm

Printed in the U.S.A.

Embedded MultiMediaCard (eMMC) eMMC/Card Product Standard, High Capacity, including Reliable Write, Boot, and Sleep Modes

CONTENTS

Page

Embedded MultiMediaCard (eMMC) eMMC/Card Product Standard, High Capacity, including Reliable Write, Boot, and Sleep Modes

CONTENTS*(continued)*

Page

Embedded MultiMediaCard (eMMC) eMMC/Card Product Standard, High Capacity, including Reliable Write, Boot, and Sleep Modes

CONTENTS*(continued)*

Page

Embedded MultiMediaCard (eMMC) eMMC/Card Product Standard, High Capacity, including Reliable Write, Boot, and Sleep Modes

CONTENTS*(continued)*

Page

Embedded MultiMediaCard (eMMC) eMMC/Card Product Standard, High Capacity, including Reliable Write, Boot, and Sleep Modes

CONTENTS*(continued)*

Page

Embedded MultiMediaCard (eMMC) eMMC/Card Product Standard, High Capacity, including Reliable Write, Boot, and Sleep Modes

CONTENTS*(continued)*

Page

Embedded MultiMediaCard (eMMC) eMMC/Card Product Standard, High Capacity, including Reliable Write, Boot, and Sleep Modes

CONTENTS*(continued)*

Page

Embedded MultiMediaCard (eMMC) eMMC/Card Product Standard, High Capacity, including Reliable Write, Boot, and Sleep Modes

CONTENTS*(continued)*

Embedded MultiMediaCard (eMMC) eMMC/Card Product Standard, High Capacity, including Reliable Write, Boot, and Sleep Modes

CONTENTS*(continued)*

Page

Foreword

This standard has been prepared by JEDEC and the MultiMediaCard Association, hereafter referred to as MMCA.

JEDEC has taken the basic MMCA specification and adopted it for embedded applications, calling it “eMMC.” In addition to the packaging differences, eMMC devices use a reduced-voltage interface.

The purpose of this specification is the definition of the MMC/eMMC Electrical Interface, its environment and handling. It provides guidelines for systems designers. The specification also defines a tool box (a set of macro functions and algorithms) that contributes to reducing design-in costs.

The SPI mode is obsolete in this version.

Introduction

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

MMC/eMMC communication is based on an advanced 10-signal bus. The communication protocol is defined as a part of this standard and referred to as the MultiMediaCard mode.

To provide for the forecasted migration of CMOS power (V_{DD}) requirements and for compatibility and integrity of MultiMediaCard systems, two types of MultiMediaCards 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 Dual Voltage MultiMediaCard.

Embedded MultiMediaCard (eMMC) eMMC/Card, High Capacity, including Reliable Write, Boot, and Sleep Modes

(From BoD ballot, JCB-07-107, formulated under the cognizance of the JC-64 committee on Flash Memory Modules)

1 Scope

This document provides a comprehensive definition of the MMC/eMMC Electrical Interface, its environment, and handling. It also provides design guidelines and defines a tool box of macro functions and algorithms intended to reduce design-in costs.

2 Normative reference

The following normative documents contain provisions that, through reference in this text, constitute provisions of this standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

3 Terms and definitions

For the purposes of this publication, the following abbreviations for common terms apply:

Block	a number of bytes, basic data transfer unit
Broadcast	a command sent to all cards on the MultiMediaCard bus[1]
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
eMMC	embedded MultiMediaCard
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

[1] Broadcast occurs only in MultiMediaCard systems supporting versions prior to 4.0. In version 4.0 and later only one card can be present on the bus.

3 Terms and definitions (continued)

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 0 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

4 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 smart phones, cameras, organizers, PDAs, digital recorders, MP3 players, pagers, electronic toys, etc. Targeted features are high mobility and high performance at a low cost price. These features include low power consumption and high data throughput at the memory card interface.

The MultiMediaCard communication is based on an advanced 13-pin bus. The communication protocol is defined as a part of this standard and referred to as the MultiMediaCard mode.

To provide for the forecasted migration of CMOS power (V_{DD}) requirements and for compatibility and integrity of MultiMediaCard systems, two types of MultiMediaCards 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 Dual 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 contributes to reducing the design-in costs.

The document is split up into several portions. The MultimediaCard Features are described in Section 5.

Section 6 gives a general overview of the system components: card, bus, and host.

The common MultiMediaCard characteristics are described in Section 7. As this description defines an overall set of card properties, you should work with the vendor-specific, product documentation in parallel.

Section 8 describes the card registers.

The SPI mode is removed from this standard.

All error protection techniques employed in this standard are described in Section 10.

Section 11 describes the physical and mechanical properties of the cards and the minimal requirements of the card slots and cartridges.

Section 12 defines the MultiMediaCard bus as a universal communication interface and the electrical parameters of the interface.

The standard compliance criteria for the cards and hosts are described in Section 13.

For achieving high data interchangeability, three basic file formats are defined in Section 14 as valid file formats for the MultiMediaCard

Annex A contains additional information that is informative in nature and not considered a constituent part of this specification. These Application Notes contain useful hints for the circuit and system designers, helping simplify the design process.

Annex B lists the major changes between the previous and the current version of this specification.

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.

5 System features

The MultiMediaCard System has a wide variety of system features, whose comprehensive elements serves several purposes, which include:

- Covering a broad category of applications from smart phones and PDAs to digital recorders and toys
- Facilitating the work of designers who seek to develop applications with their own advanced and enhanced features
- Maintainng compatibility and compliance with current electronic, communication, data and error handling standards.

The following list identifies the main features of the MultiMediaCard System, which:

- Is targeted for portable and stationary applications
- Has these System Voltage (V_{DD}) Ranges:

Table 1 — MultiMediaCard Voltage Modes

	High Voltage MultiMediaCard	Dual Voltage MultiMediaCard
Communication	**2.7 - 3.6**	**1.70 - 1.95, 2.7 - 3.6**[1]
Memory Access	**2.7 - 3.6**	**1.70 - 1.95, 2.7 - 3.6**

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

- Includes MMCplus and MMCmobile definitions
- Is designed for read-only, read/write and I/O cards
- Supports card clock frequencies of 0-20MHz, 0-26MHz or 0-52MHz
- Has a maximum data rate up to 416Mbits/sec.
- Has a defined minimum performance
- Maintains card support for three different data bus width modes: 1-bit (default), 4-bit, and 8-bit
- Includes definition for higher than 2GB of density of memories
- Includes password protection of data
- Supports basic file formats for high data interchangeability
- Includes application specific commands
- Enables correction of memory field errors
- Has built-in write protection features, which may be permanent or temporary
- Includes a simple erase mechanism
- Maintains full backward compatibility with previous MultiMediaCard systems (1 bit data bus, multi-card systems)

- Ensures that new hosts retain full compatibility with previous versions of MultiMediaCards (backward compatibility).
- Supports two form factors: Normal size (24mm x 32mm x 1.4mm) and reduced size (24mm x 18mm x 1.4mm)
- Supports multiple command sets
- Includes attributes of the available operation modes:

Table 2 — MMC System Operational Mode

MultiMediaCard Mode
Ten-wire bus (clock, 1 bit command, 8 bit data bus)
Card selection is done through an assigned unique card address to maintain backwards compatibility to prior versions of the specification
One card per MultiMediaCard bus
Easy identification and assignment of session address
Error-protected data transfer
Sequential and Single/Multiple block Read/Write commands

- Provides a possibility for the host to make sudden power failure safe-update operations for the data content.
- Enhanced power saving method by introducing a sleep functionality.
- Introduces Boot Operation Mode to provide a simple boot sequence method.
- Provides a new CID Register setting to recognize either eMMC or a card.
- Obsoletes the SPI Mode.
- Defines I/O voltage ($V_{CC}Q$) and core voltage (V_{CC}) separately for eMMC.
- Includes eMMC BGA Form Factors:
  11.5mm x 13mm x 1.3mm
  12mm x 16mm x 1.4mm
  12mm x 18mm x 1.4mm
- Defines Erase-unit size and Erase timeout for high-capacity memory.
- Provides access size register indicating one (or multiple) programmable boundary unit(s) of device.
- Obsoletes the Absolute Minimum Performance.
- Introduces eMMC OCR setting and response.
- Defines WP group size for high-capacity devices.
- Introduces optional Alternate Boot Operation Mode.

6 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 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.

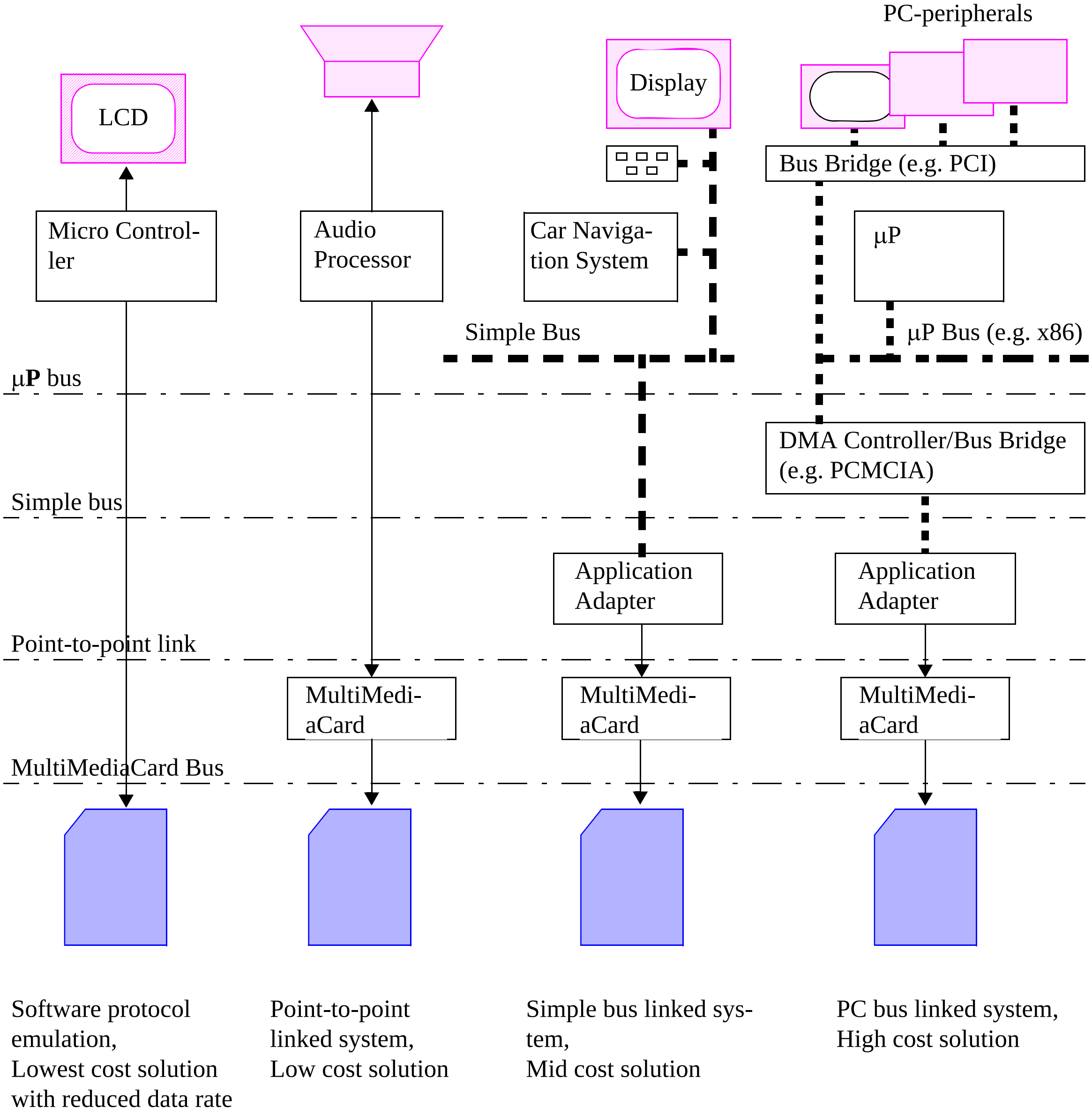


Figure 1 — Topology of MultiMediaCard systems

Four typical types of MultiMediaCard systems can be derived from the diagram shown in Figure 2. The typical systems include:

- Software emulation: reduced data rate, typically 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, such as DMA capabilities

In the first variant, the MultiMediaCard bus protocol is emulated in software using up to ten 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 on page 7.) It does not imply any specific design nor specify rules for implementing parts in hardware or software.

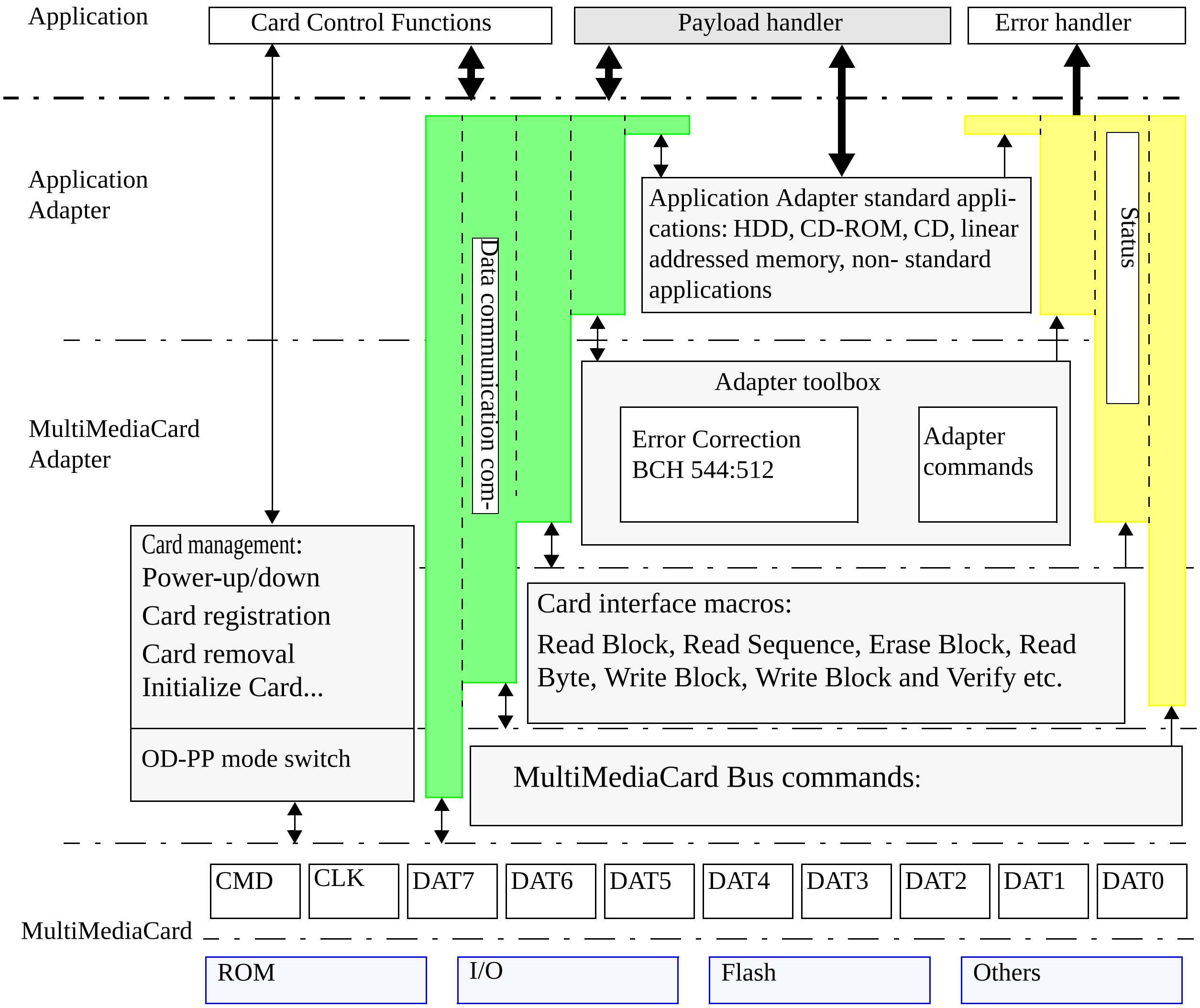


Figure 2 — MultiMediaCard system overview

Figure 3 is a specific design example based on the abstract layer model described in Figure 2 on page 8.

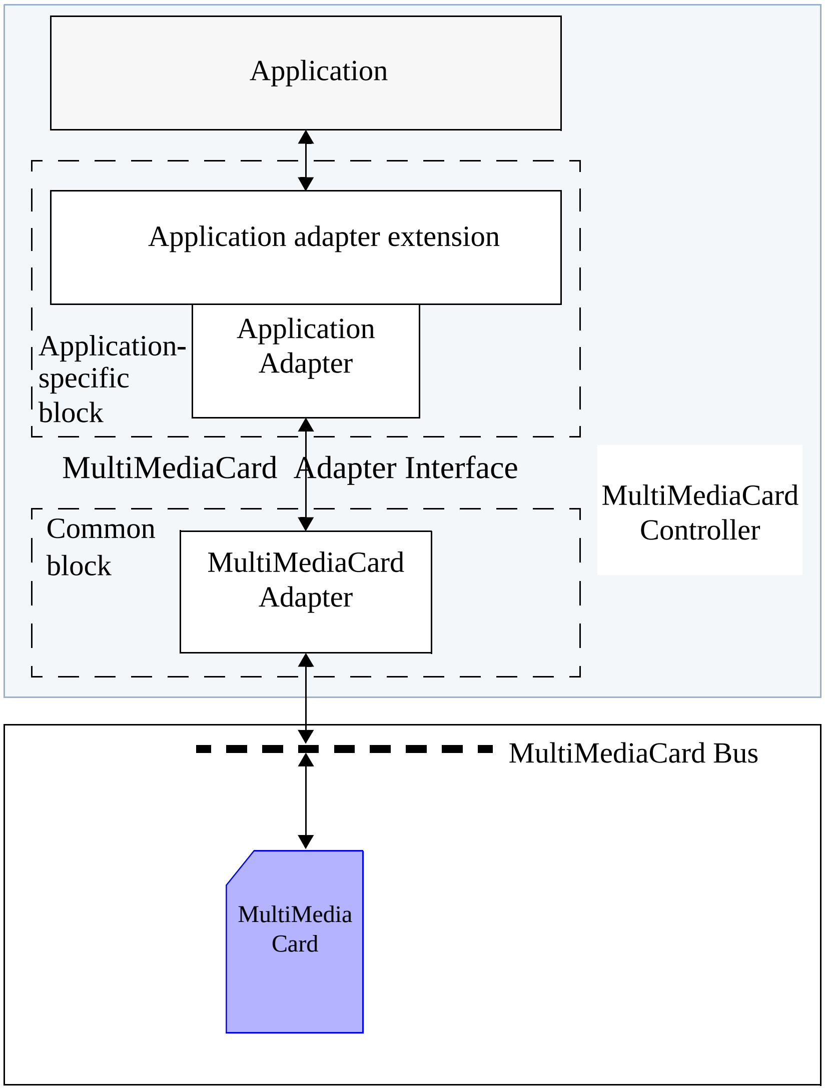

Application
Application adapter extension
Application-specific block
Application Adapter
MultiMediaCard Adapter Interface
MultiMediaCard Controller
Common block
MultiMediaCard Adapter
MultiMediaCard Bus
MultiMedia Card

Figure 3 — MultiMediaCard system example

This MultiMediaCard system contains at least two components:

- The MultiMediaCard
- The MultiMediaCard controller

The MultiMediaCard controller is divided into two major blocks. In some implementations like the example shown in Figure 3, 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:

1: Application adapter — the application specific block, for example, a microprocessor or an adapter to a standard bus like USB or ATA

- 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

2: MultiMediaCard adapter — the common block

- Contains all card specific functions, such as initialization and error correction
- Serves as a bus master for the MultiMediaCard bus
- Implements the standard interface to the card.

6.1 Higher than a density of 2GB

The maximum density possible to be implemented according to the versions up to v4.1 of this specification was limited in practise to 2GB. This was due to the following reasons:

- • Existed 32bit byte-address argument in the command frame (max 4GB could be addressed)
- • Existed formula according to which to calculate the density of a card (max 4GB could be indicated)
- • Capability of the FAT16 File System to address up to 2GB of address space per one partition

The lowest common nominator, 2GB in this case, will set the limit. The implementation of a higher than 2GB of density of memory will not be backwards compatible with the lower densities. First of all the address argument for higher than 2GB of density of memory is changed to be sector address (512B sectors) instead of byte address. Secondly the density of the card is read from the EXT_CSD register instead of CSD register. And finally the system implementation needs to include a File System capable of handling sector type of addresses.

6.2 MMC*plus* and MMC*mobile*

The specification further defines two card types, MMCplus and MMCmobile, to describe R/W or ROM cards with specifically defined mandatory features and attributes. Only cards meeting MMCplus or MMCmobile requirements are eligible to carry the MMCplus or MMCmobile name and logo.

- MMC*plus* is defined as normal size R/W or ROM cards that supports 2.7-3.6V operation, x1/x4/x8 bus widths, and 26MHz (52MHz optional)
- MMC*mobile* is defined as reduced size R/W or ROM card that supports 1.70-1.95V and 2.7-3.6V operations, x1/x4/x8 bus widths, minimum of 2.4MB/s read/write performance and 26MHz (52MHz optional)

Both implementations are backwards compatible with MMCA System Specification versions 3.xx in max 20MHz clock frequency mode.

6.3 Card concept

The MultiMediaCard transfers data via a configurable number of data bus signals. The communication signals are:

- **CLK**: Each cycle of this signal directs a one bit transfer on the command and on all the data lines. The frequency may vary between zero and the maximum clock frequency.
- **CMD**: This signal is a bidirectional command channel used for card initialization and transfer of 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 are sent from the card to the host.

- **DAT0-DAT7**: These are bidirectional data channels. The DAT signals operate in push-pull mode. Only the card or the host is driving these signals at a time. By default, after power up or reset, only DAT0 is used for data transfer. A wider data bus can be configured for data transfer, using either DAT0-DAT3 or DAT0-DAT7, by the MultiMediaCard controller. The MultiMediaCard includes internal pull-ups for data lines DAT1-DAT7. Immediately after entering the 4-bit mode, the card disconnects the internal pull ups of lines DAT1, DAT2, and DAT3. Correspondingly, immediately after entering to the 8-bit mode the card disconnects the internal pull-ups of lines DAT1–DAT7.
- 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.

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

Table 3 — MultiMediaCard interface pin configuration

Name	Type[1]	Description
CLK	I	Clock
DAT0[2]	I/O/PP	Data
DAT1	I/O/PP	Data
DAT2	I/O/PP	Data
DAT3	I/O/PP	Data
DAT4	I/O/PP	Data
DAT5	I/O/PP	Data
DAT6	I/O/PP	Data
DAT7	I/O/PP	Data
CMD	I/O/PP/OD	Command/Response
V_{CC}	S	Supply voltage for NAND (BGA)
$V_{CC}Q$	S	Supply voltage for MMC interface (BGA)
V_{DD}	S	Supply voltage (card)
V_{SS}	S	Supply voltage ground for NAND (BGA)
V_{SS1}	S	Supply voltage ground (card)
V_{SS2}	S	Supply voltage ground (card)
$V_{SS}Q$	S	Supply voltage ground for MMC interface (BGA)

NOTE 1 I: input; O: output; PP: push-pull; OD: open-drain; NC: Not connected (or logical high); S: power supply.
NOTE 2 The DAT0–DAT7 lines for read-only cards are output only.

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

Each card has a set of information registers (see also Section 8 on page 77):

Table 4 — MultiMediaCard registers

Name	Width (bytes)	Description	Implementation
CID	16	Card IDentification number, a card individual number for identification.	Mandatory
RCA	2	Relative Card Address, is the card system address, dynamically assigned by the host during initialization.	Mandatory
DSR	2	Driver Stage Register, to configure the card's output drivers.	Optional
CSD	16	Card Specific Data, information about the card operation conditions.	Mandatory
OCR	4	Operation Conditions Register. Used by a special broadcast command to identify the voltage type of the card.	Mandatory
EXT_CSD	512	Extended Card Specific Data. Contains information about the card capabilities and selected modes. Introduced in specification v4.0	Mandatory

The host may reset the card by switching the power supply off and back on. The 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 card can also be reset by a special command.

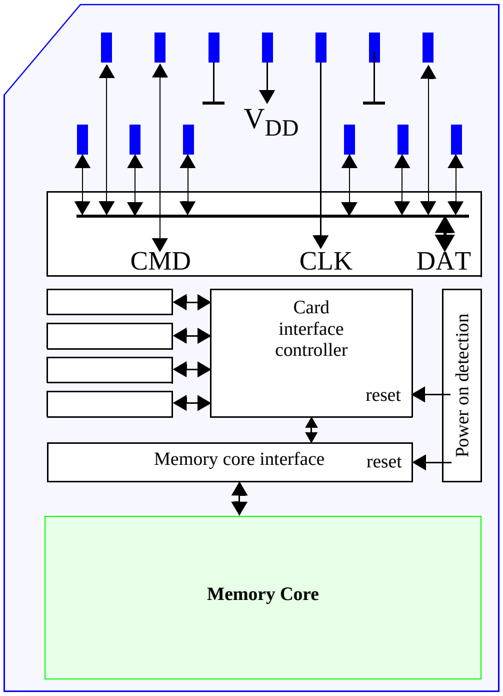


Figure 4 — MultiMediaCard architecture

6.3.1 Form factors

See Chapter 8 for form factor details.

6.4 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 single slave. The MultiMediaCard bus master is the bus controller and the 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.

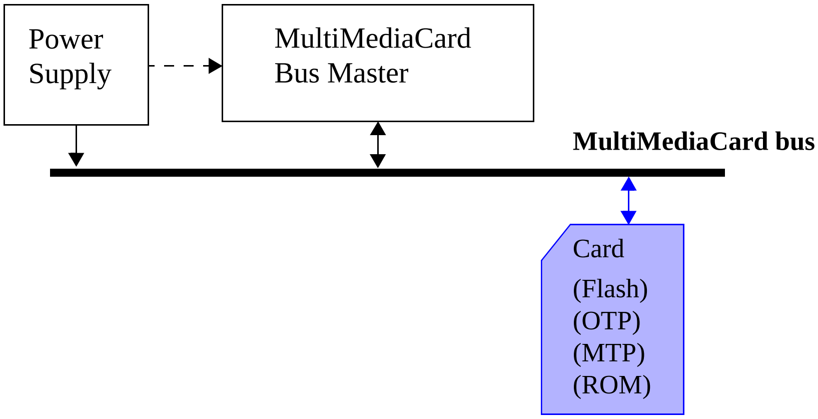


Figure 5 — MultiMediaCard bus system

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

The bus communication uses a special protocol (MultiMediaCard bus protocol). The payload data transfer between the host and the card can be bidirectional.

6.4.1 Bus lines

The bus lines can be divided into three groups:

- Power supply: V_{SS1} and V_{SS2},V_{DD} - used to supply the cards.
  V_{SS}, $V_{SS}Q$, V_{CC}, and $V_{CC}Q$ - used to supply eMMC.
- Data transfer: CMD, DAT0-DAT7 - 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 Section 6.3.

6.4.2 Bus protocol

After a power-on reset, the host must initialize the card 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 to a card. A command is transferred serially on the CMD line.
- response: a response is a token which is sent from the card 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 lines. The number of data lines used for the data transfer can be 1(DAT0), 4(DAT0-DAT3) or 8(DAT0-DAT7).

Card addressing is implemented using a session address, assigned during the initialization phase, by the bus controller to the connected card. A card is identified by its CID number. This method requires the card to have an unique CID number. To ensure uniqueness of CIDs the CID register contains 24 bits (MID and OID fields—see Section 8 starting on page 77) which are defined by the MMCA/JEDEC. Every card manufacturer is required to apply for an unique MID (and optionally OID) number.

Command, response, and data block structures are described in Section 7 starting on page 21.

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 DAT0-DAT7 and CMD lines are transferred synchronous to the host clock.

Two types of data transfer commands are defined:

- Sequential commands[2]: 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.

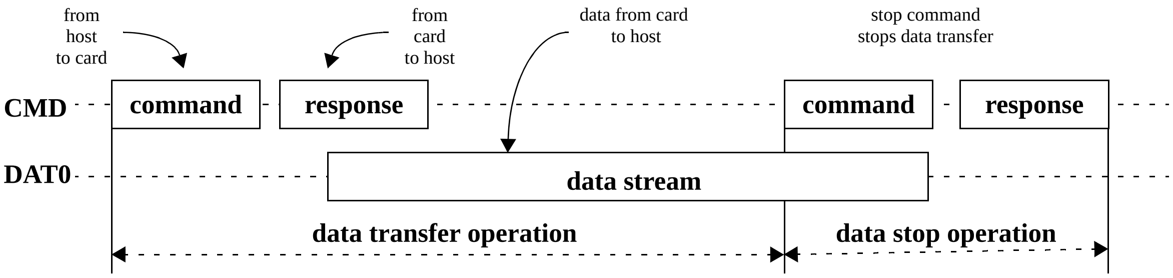


Figure 6 — Sequential read operation

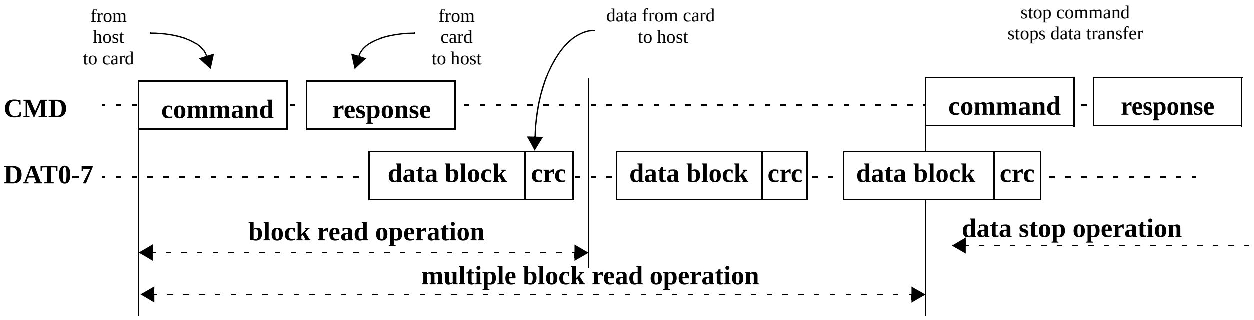


Figure 7 — Multiple-block read operation

[2] Sequential commands are supported only in 1-bit bus mode, to maintain compatibility with previous versions of this specification

.

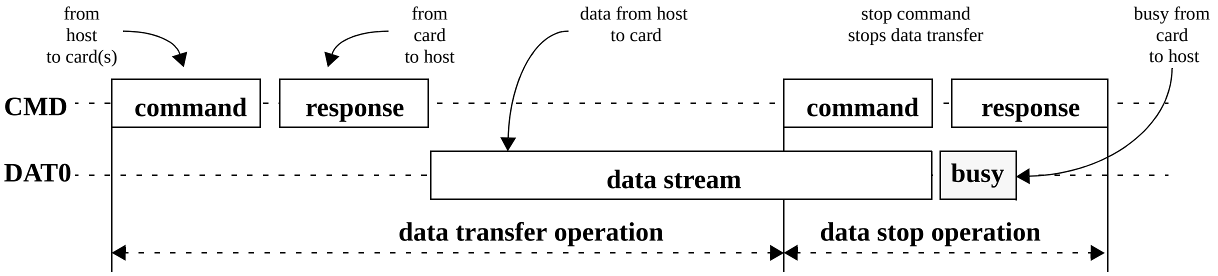


Figure 8 — Sequential write operation

The block write operation uses a simple busy signalling of the write operation duration on the data (DAT0) line. (See Figure 9.)

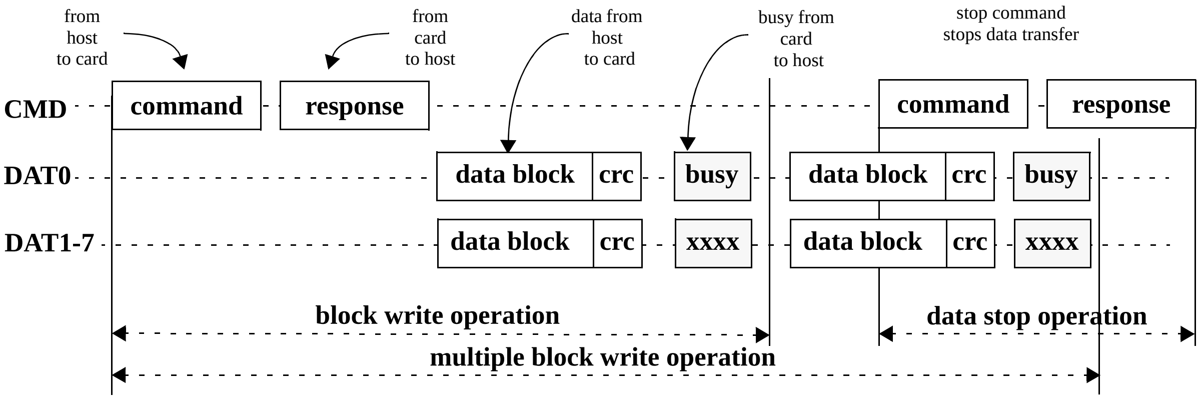


Figure 9 — (Multiple) Block write operation

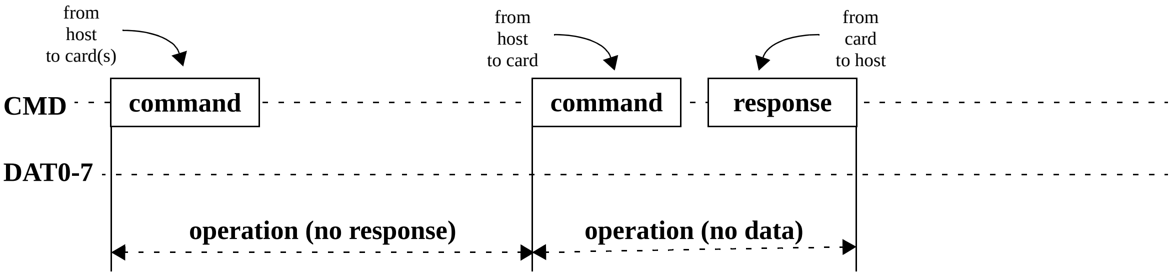


Figure 10 — “No response” and “no data” operations

Command tokens have the following coding scheme:

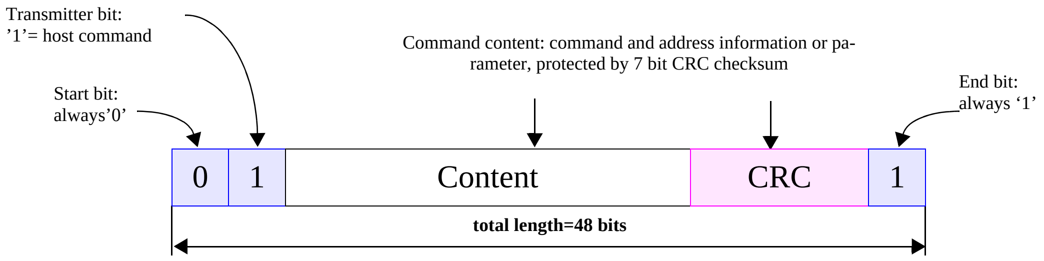


Figure 11 — Command token format

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 command and response definitions are provided in Section 7.9 on page 49 and Section 7.11 on page 59.

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 Section 10 starting on page 103.

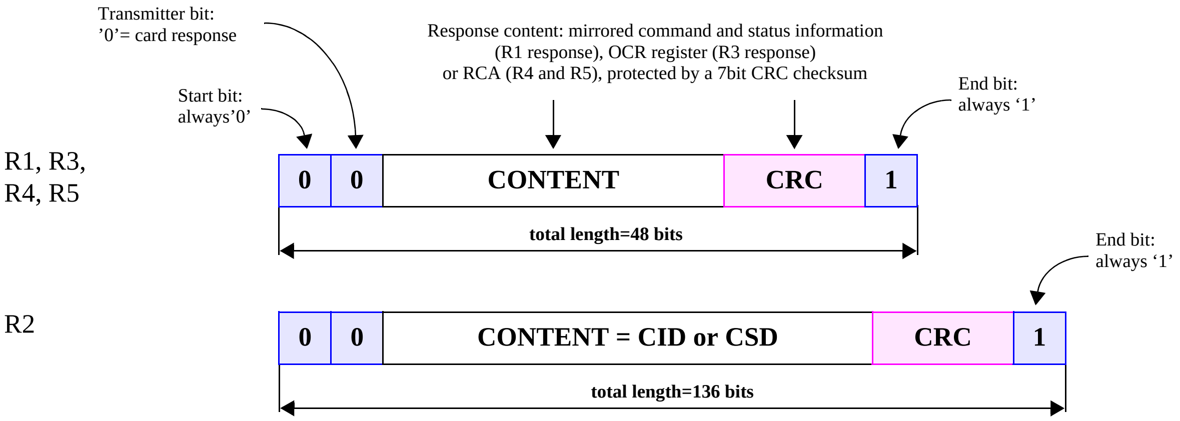


Figure 12 — Response token format

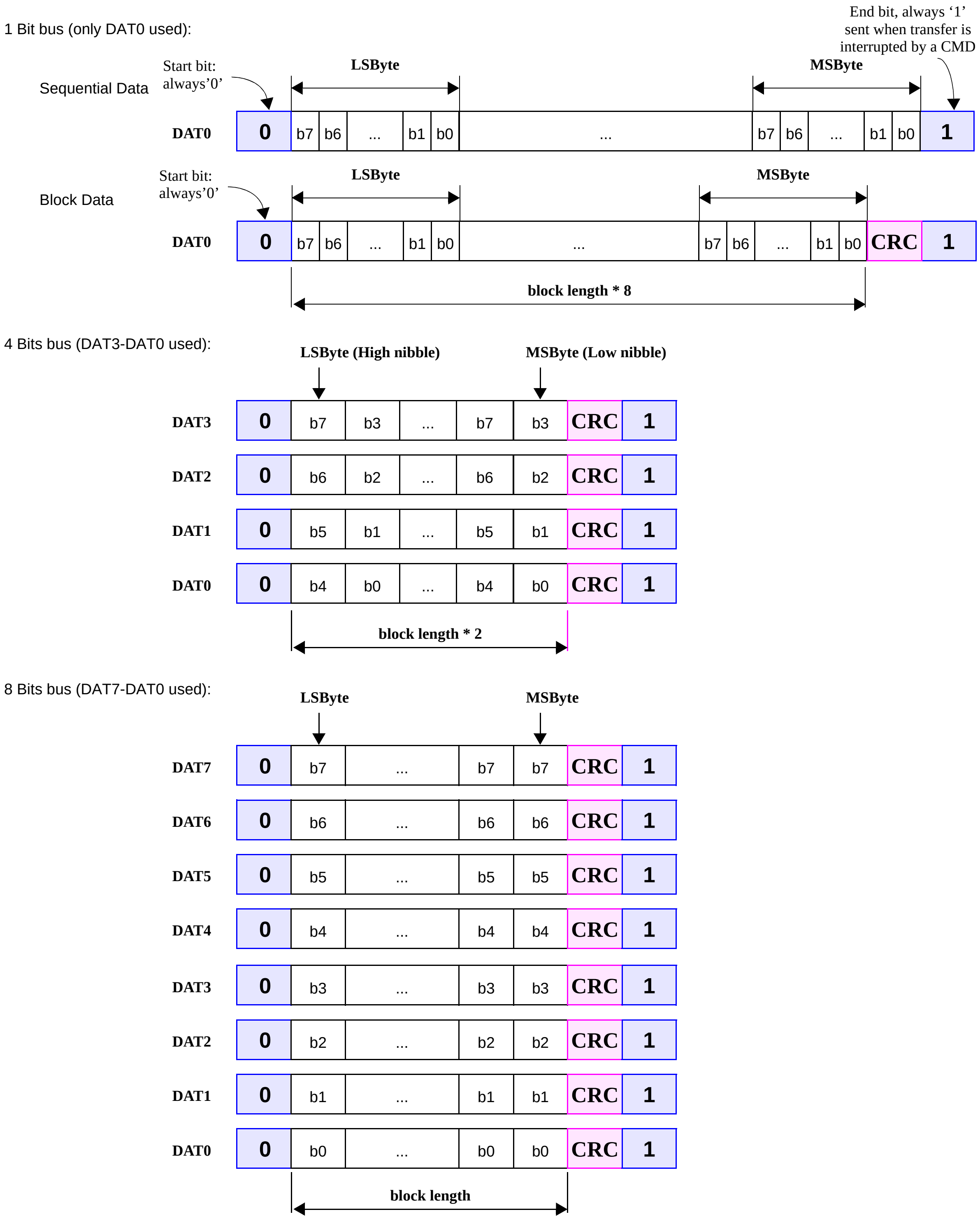


Figure 13 — Data packet format

6.5 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
- Macros for common complex command sequences

The MultiMediaCard controller is the link between the application and the MultiMediaCard bus with its card. 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

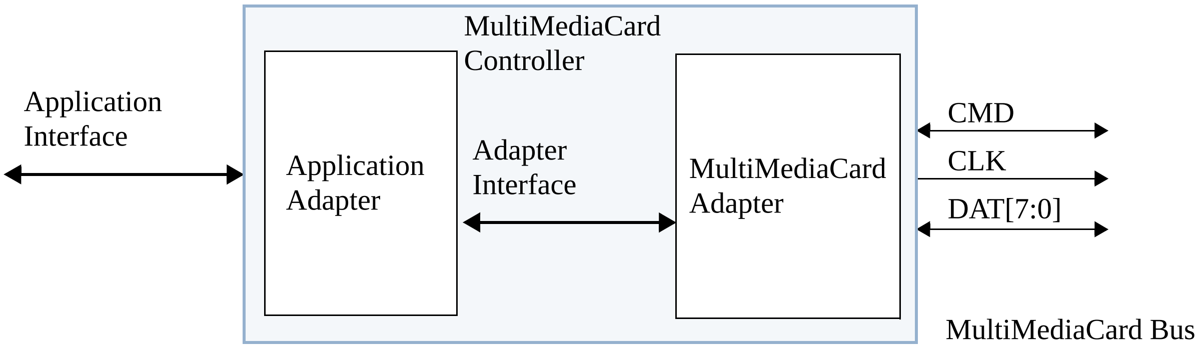


Figure 14 — MultiMediaCard controller scheme

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 for those that are common to most MultiMediaCard systems. 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 error correction codes used are defined in Section 10.1 on page 103.

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.

6.5.1 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 a 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.

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

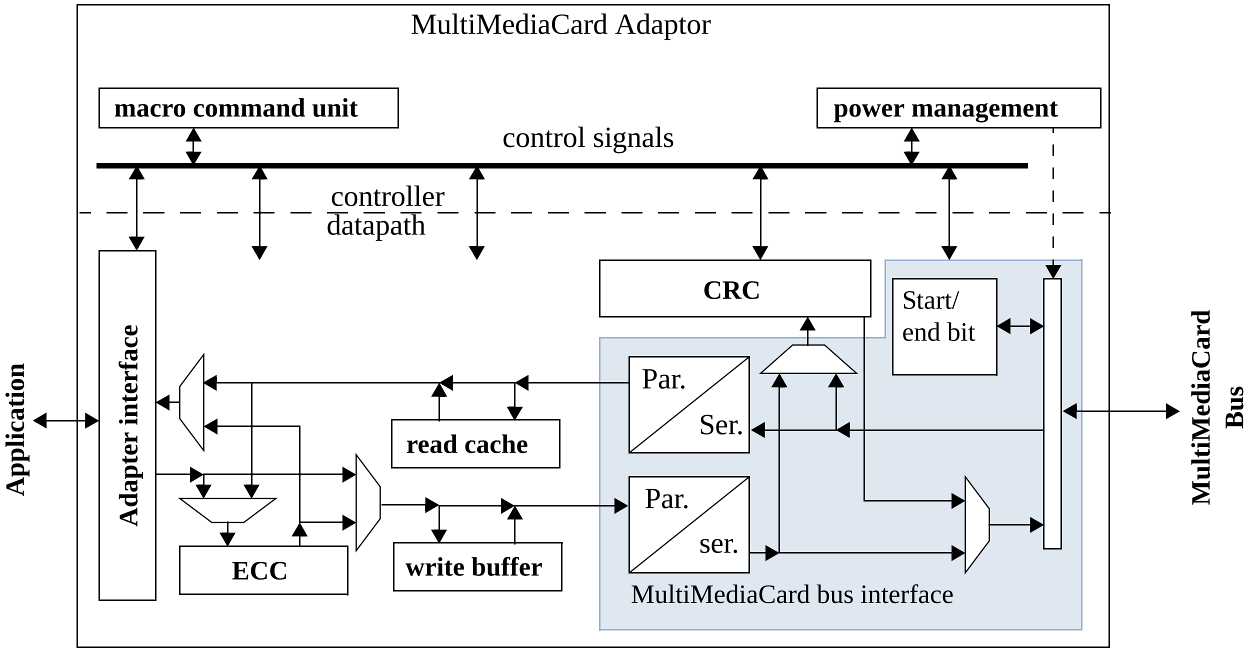


Figure 15 — MultiMediaCard adaptor architecture

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 and read operation. It will, in most cases, improve the system level performance on the application side.The MultiMediaCard bus transports its data with a data rate up to 416 Mbit/sec. This may be 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 MultiMediaCard 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.

7 MultiMediaCard functional description

In the following sections, the different card operation modes are 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.

7.1 General

All communication between host and card 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 in a MultiMediaCard system[3]. 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 20 on page 28 for the card identification mode and in Figure 22 on page 32 for the data transfer mode. The commands are listed in the command tables (Table 12 on page 51 to Table 21 on page 57). The dependencies between current state, received command and following state are listed in Table 22 on page 57. Three operation modes are defined for the MultiMediaCard system (hosts and cards):

- Card identification mode

  The host will be in card identification mode after reset, while it is looking for a card on the bus. The card will be in this mode after reset, until the SET_RCA command (CMD3) is received.

- Interrupt mode

  Host and card enter and exit interrupt mode simultaneously. In interrupt mode there is no data transfer. The only message allowed is an interrupt service request from the card or the host.

- Data transfer mode

  The card will enter data transfer mode once an RCA is assigned to it. The host will enter data transfer mode after identifying the card 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 20 and Figure 22) is associated with one bus mode and one operation mode:

[3] Broadcast commands are kept for backwards compatibility to previous MultiMediaCard systems, where more than one card was allowed on the bus.

Table 5 — Bus modes overview

Card state	Operation mode	Bus mode
Inactive State	Inactive	Open-drain
Pre-Idle State	Boot mode	
Pre-Boot State		
Idle State	Card identification mode	
Ready State		
Identification State		
Stand-by State	Data transfer mode	Push-pull
Sleep State		
Transfer State		
Bus-Test State		
Sending-data State		
Receive-data State		
Programming State		
Disconnect State		
Boot State	Boot mode	
Wait-IRQ State	Interrupt mode	Open-drain

7.2 Boot operation mode

In boot operation mode, the master (MultiMediaCard host) can read boot data from the slave (MMC device) by keeping CMD line low after power-on, or sending CMD0 with argument + 0xFFFFFFFA (optional for slave), before issuing CMD1. The data can be read from either boot area or user area depending on register setting.

7.2.1 Boot partition

There are two partition regions. The minimum size of each boot partition is 128KB. Boot partition size is calculated as follows:

Maximum boot partition size = 128K byte x BOOT_SIZE_MULT

BOOT_SIZE_MULT: the value in Extended CSD register byte [226]

The boot partitions are separated from the user area as shown in Figure 16 on page 23.

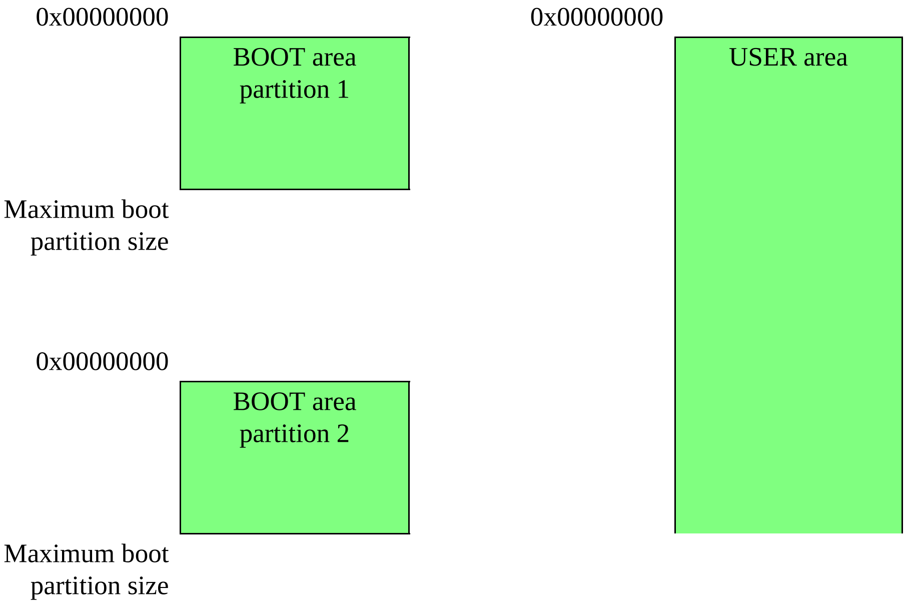


Figure 16 — Memory partition

Slave has boot configuration in Extended CSD register byte [179]. The master can choose the configuration by setting the register using CMD6 (switch). Slave also can be configured to boot from the user area by setting the BOOT_PARTITION_ ENABLE bits in the EXT_CSD register, byte [179] to 111b.

7.2.2 Boot operation

If the CMD line is held LOW for 74 clock cycles and more after power-up before the first command is issued, the slave recognizes that boot mode is being initiated and starts preparing boot data internally. The partition from which the master will read the boot data can be selected in advance using EXT_CSD byte 179, bits [5:3]. The data size that the master can read during boot operation can be calculated as 128KB × BOOT_SIZE_MULT (EXT_CSD byte 226). Within 1 second after the CMD line goes LOW, the slave starts to send the first boot data to the master on the DAT line(s). The master must keep the CMD line LOW to read all of the boot data. The master must use push-pull mode until boot operation is terminated. The master can use the backward-compatible interface timing shown in Table 85 on page 122.

The master can choose to receive boot acknowledge from the slave by setting “1” in EXT_CSD register, byte 179, bit 6, so that the master can recognize that the slave is operating in boot mode.

If boot acknowledge is enabled, the slave has to send acknowledge pattern “010” to the master within 50ms after the CMD line goes LOW. If boot acknowledge is disabled, the slave will not send out acknowledge pattern “0-1-0.”

The master can terminate boot mode with the CMD line HIGH. If the master pulls the CMD line HIGH in the middle of data transfer, the slave has to terminate the data transfer or acknowledge pattern within N_{ST} clock cycles (one data cycle and end bit cycle). If the master terminates boot mode between consecutive blocks, the slave must release the data line(s) within N_{ST} clock cycles.

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.

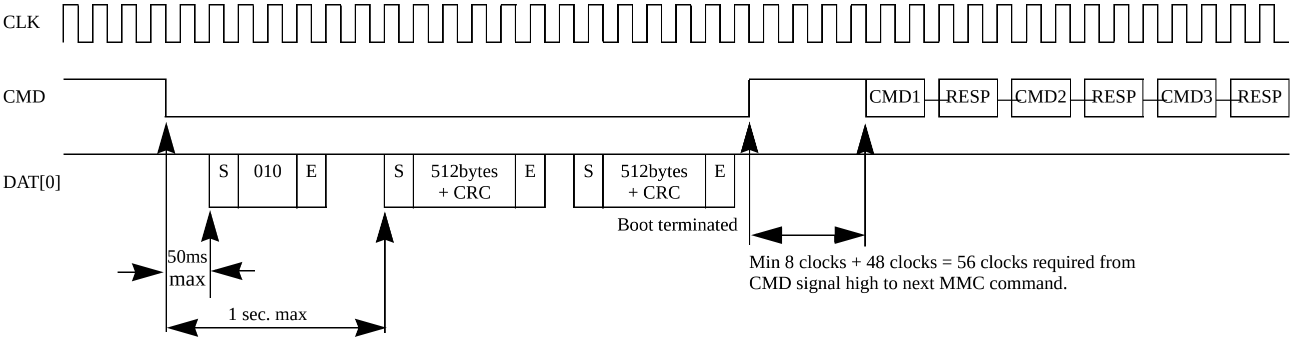


Figure 17 — MultiMediaCard state diagram (boot mode)

Detailed timings are shown in Section 7.14.5 on page 73. Min 8 clocks + 48 clocks = 56 clocks required from CMD signal high to next MMC command.

If the CMD line is held LOW for less than 74 clock cycles after power-up before CMD1 is issued, or the master sends any normal MMC command other than CMD1 and CMD0 with argument 0xFFFFFFFA (if the device supports alternate boot operation) before initiating boot mode, the slave does not respond and will be locked out of boot mode until the next power cycle and enter Idle State.

When BOOT_PARTITION_ENABLE bits are set and master send CMD1 (SEND_OP_COND), slave must enter Card Identification Mode and respond to the command.

If the slave does not support boot operation mode, which is compliant with v4.2 or before, or BOOT_PARTITION_ENABLE bit is cleared, slave automatically enter Idle State after power-on.

7.2.3 Alternative boot operation (device optional)

This boot function is optional for the device. If bit 0 in the extended CSD byte[228] is set to “1,” the device supports the alternative boot operation.

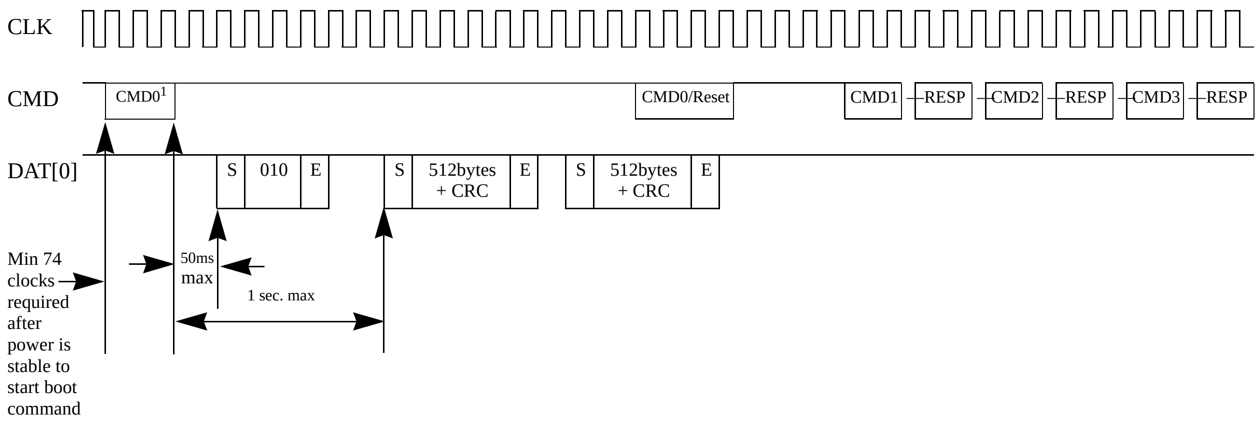
After power-up, if the host issues CMD0 with the argument of 0xFFFFFFFA after 74 clock cycles, before CMD1 is issued or the CMD line goes low, the slave recognizes that boot mode is being initiated and starts preparing boot data internally. The partition from which the master will read the boot data can be selected in advance using EXT_CSD byte 179, bits [5:3]. The data size that the master can read during boot operation can be calculated as 128KB × BOOT_SIZE_MULT (EXT_CSD byte 226. Within 1 second after CMD0 with the argument of 0xFFFFFFFA is issued, the slave starts to send the first boot data to the master on the DAT line(s). The master must use push-pull mode until boot operation is terminated. The master can use the backward-compatible interface timing shown in Table 85 on page 122.

The master can choose to receive boot acknowledge from the slave by setting “1” in EXT_CSD register, byte 179, bit 6, so that the master can recognize that the slave is operating in boot mode.

If boot acknowledge is enabled, the slave has to send the acknowledge pattern “010” to the master within 50ms after the CMD0 with the argument of 0xFFFFFFFA is received. If boot acknowledge is disabled, the slave will not send out acknowledge pattern “010.”

The master can terminate boot mode by issuing CMD0 (Reset). If the master issues CMD0 (Reset) in the middle of a data transfer, the slave has to terminate the data transfer or acknowledge pattern within NST clock cycles (one data cycle and end bit cycle). If the master terminates boot mode between consecutive blocks, the slave must release the data line(s) within N_{ST} clock cycles.

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.



NOTE 1. CMD0 with argument 0xFFFFFFFA

Figure 18 — MultiMediaCard state diagram (alternative boot mode)

Detailed timings are shown in Section 7.14.6 on page 74.

If the CMD line is held LOW for less than 74 clock cycles after power-up before CMD1 is issued, or the master sends any normal MMC command other than CMD1 and CMD0 with argument 0xFFFFFFFA (if the device supports alternate boot operation) before initiating boot mode, the slave does not respond and will be locked out of boot mode until the next power cycle and enter Idle State.

When BOOT_PARTITION_ENABLE bits are set and master send CMD1 (SEND_OP_COND), slave must enter Card Identification Mode and respond to the command.

If the slave does not support boot operation mode, which is compliant with v4.2 or before, or

BOOT_PARTITION_ENABLE bit is cleared, slave automatically enter Idle State after power-on.

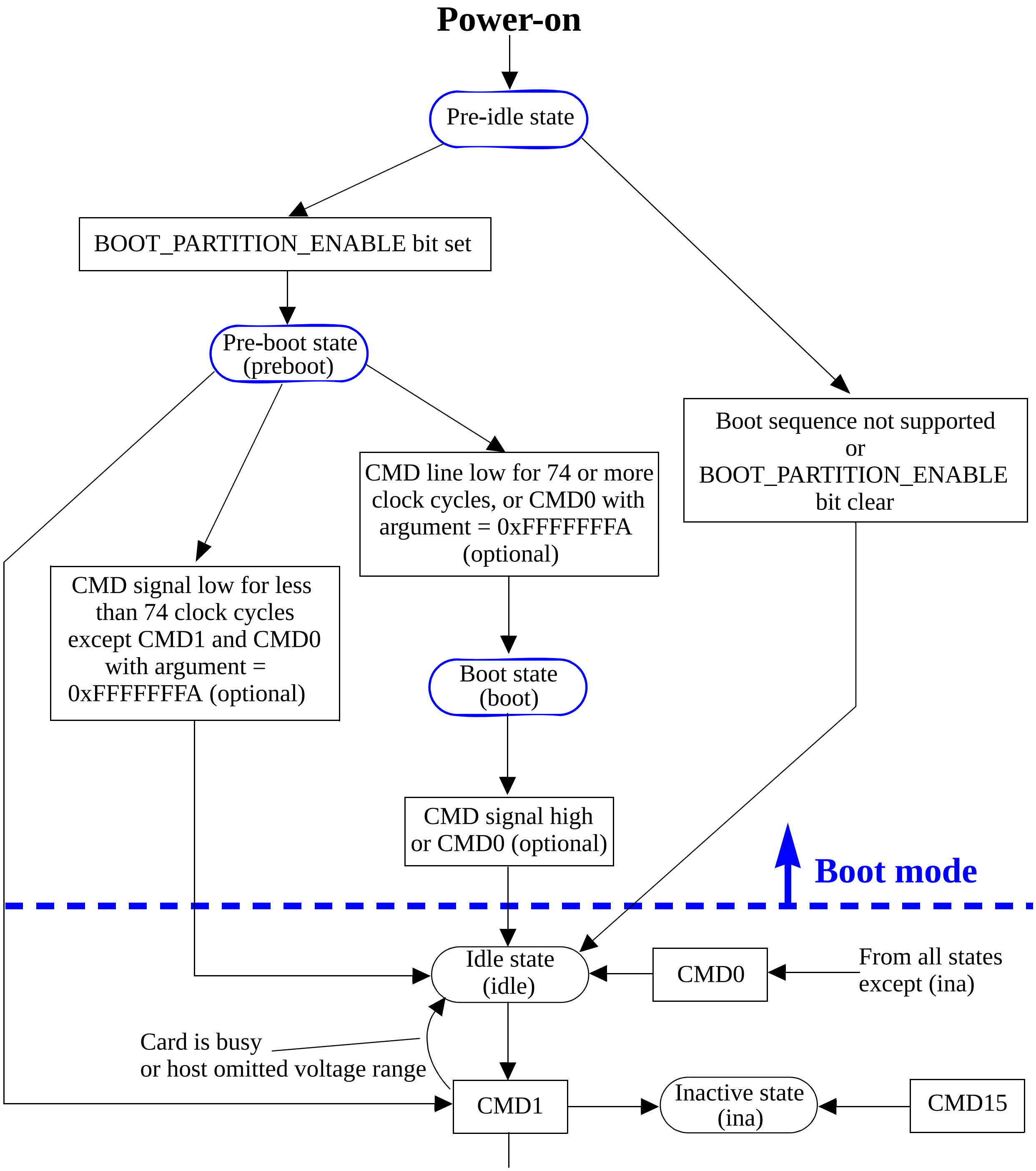

Power-on
Pre-idle state
BOOT_PARTITION_ENABLE bit set
Pre-boot state (preboot)
Boot sequence not supported or BOOT_PARTITION_ENABLE bit clear
CMD line low for 74 or more clock cycles, or CMD0 with argument = 0xFFFFFFFA (optional)
CMD signal low for less than 74 clock cycles except CMD1 and CMD0 with argument = 0xFFFFFFFA (optional)
Boot state (boot)
CMD signal high or CMD0 (optional)
Boot mode
Idle state (idle)
CMD0
From all states except (ina)
Card is busy or host omitted voltage range
CMD1
Inactive state (ina)
CMD15

Figure 19 — MultiMediaCard state diagram (boot mode)

7.2.4 Access to boot partition

After putting a slave into transfer state, master sends CMD6 (SWITCH) to set the BOOT_PARTITION_ACCESS bits in the EXT_CSD register, byte [179]. After that, master can use normal MMC commands to access a boot partition.

Master can program boot data on DAT line(s) using CMD24 (WRITE_BLOCK) or CMD25 (WRITE_MULTIPLE_BLOCK) with slave supported addressing mode i.e. byte addressing or sector addressing. If the master uses CMD25 (WRITE_MULTIPLE_BLOCK) and the writes past the selected partition boundary, the slave will report an “ADDRESS_OUT_OF_RANGE” error. Data that is within the partition boundary will be written to the selected boot partition.

Master can read boot data on DAT line(s) using CMD17 (READ_SINGLE_BLOCK) or CMD18 (READ_MULTIPLE_BLOCK) with slave supported addressing mode i.e. byte addressing or sector addressing. If the master page uses CMD18 (READ_MULTIPLE_BLOCK) and then reads past the selected partition boundary, the slave will report an “ADDRESS_OUT_OF_RANGE” error.

After finishing data access to the boot partition, the BOOT_PARTITION_ACCESS bits should be cleared. Then, non-volatile BOOT_PARTITION_ENABLE bits in the EXT_CSD register should be set to indicate which partition is enabled for booting. This will permit the slave to read data from the boot partition during boot operation.

Master also can access user area by using normal command by clearing BOOT_PARTITION_ACCESS bits in the EXT_CSD register, byte [179] to 000b.

If user area is locked and enabled for boot, data will not be sent out to master during boot operation mode. However, if the user area is locked and one of the two partitions is enabled, data will be sent out to the master during boot operation mode.

7.2.5 Boot bus width configuration

During boot operation, bus width can be configured by non-volatile configuration bits in the Extend CSD register byte[177] bit[0:1]. Bit2 in register byte[177] determines if the slave returns to x1 bus width after a boot operation or if it remains in the configured boot-bus width during normal operation. If boot operation is not executed, the slave will initialize in normal x1 bus width regardless of the register setting.

7.3 Card identification mode

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

7.3.1 Card reset

After power-on by the host, the cards (even if it has been in *Inactive State)* is in MultiMediaCard mode and in *Idle State.*

Command GO_IDLE_STATE (CMD0) is the software reset command and puts the card into *Idle State.*

After power-on, or CMD0, the cards’ output bus drivers are in high-impedance state and the card is initialized with a default relative card address (0x0001) and with a default driver stage register setting, as shown in Section 8.6 on page 99 The host clocks the bus at the identification clock rate $f_{OD,}$ as described in Section 12.7 on page 121.

CMD0 is valid in all states, with the exception of *Inactive* State. While in *Inactive* state the card does not accept CMD0.

7.3.2 Operating voltage range validation

Each type of MultiMediaCard (either High voltage or Dual 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 Section 12.5 on page 117.)

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 Section 8.1 on page 77.) If the card can not perform data transfer in the specified range it must discard itself from further bus operations and go into *Inactive State*. Otherwise, the card shall respond sending back its V_{DD} range, and the eMMC device shall respond with a fixed pattern of either 0x00FF 8080 or 0x40FF 8080, depending on the density. (This will also be true if the operand generated by the host is 0x0000 0000, which does not represent any valid range.) For this, the levels in the OCR register shall be defined accordingly as described in Section 8.1.

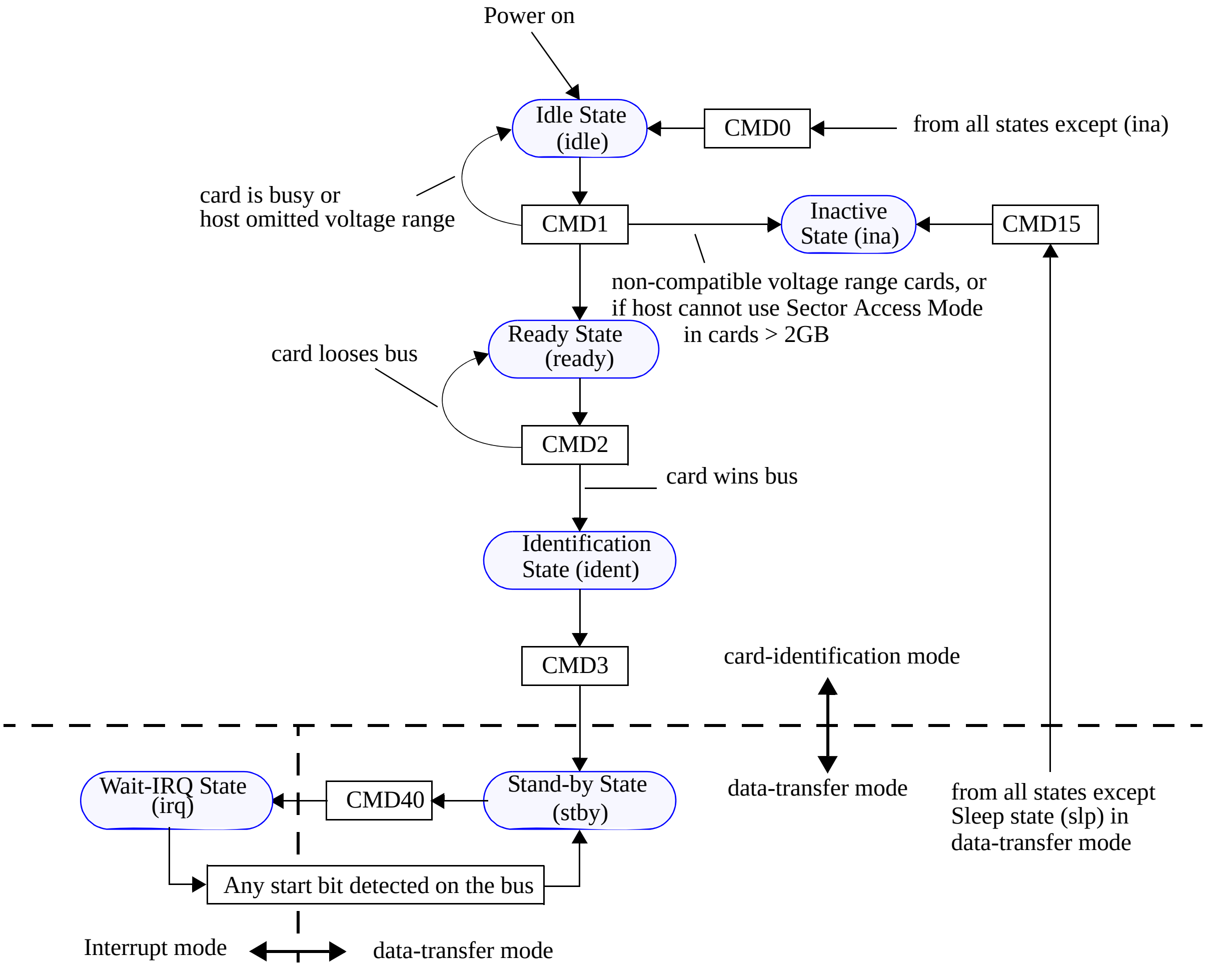


Figure 20 — MultiMediaCard state diagram (card identification mode)

If the host intends to operate the Dual Voltage MultiMediaCards in the 1.70V 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 1.70V to 1.95V range for operation. Using the 2.7V to 3.6V range initially, which is common to High and Dual 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. Dual voltage cards may fail if 1.95 to 2.7V is used.

7.3.3 Access mode validation (higher than 2GB of densities)

The SEND_OP_COND (CMD1) command and the OCR register are also including two bits for the indication of the supported access mode of the memory. The specifically set bits in the CMD1 command argument are indicating to a memory that the host is capable of handling sector type of addressing. The correspondingly set bits in the OCR register are indicating that the card is requiring usage of sector type of addressing. These specific bits of the OCR register are valid only in the last response from the card for CMD1 (card entering Ready state). This kind of two way handshaking is needed so that

• If there is no indication by a host to a memory that the host is capable of handling sector type of addressing

the higher than 2GB of density of memory will change its state to Inactive (similarly to a situation in which there is no common voltage range to work with)

• From the indication of the sector type of addressing requirement in the OCR register the host is able to separate the card from the byte access mode cards and prepare itself

It needs to be taken into account that in a multi card system a byte access mode card (<=2GB) is blocking the OCR response in such way that a sector access mode card (>2GB) is not necessarily recognized as a sector access mode card during the initialization. Thus this needs to be reconfirmed by reading the SEC_COUNT information from the EXT_CSD register.

7.3.4 From busy to ready

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 or access mode setting. Such changes shall be ignored by the card. If there is a real change in the operating conditions, the host must reset the card (using CMD0) and restart the initialization procedure. However, for accessing cards already in *Inactive State*, a hard reset must be done by switching the power supply off and back 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 all cards defined by this standard.

7.3.5 Card identification process

The following explanation refers to a card working in a multi-card environment, as defined in versions of this standard previous to v4.0, and it is maintained for backwards compatibility to those systems.

The host starts the card identification process in open-drain mode with the identification clock rate $f_{OD.}$ (See Section 12.7 on page 121.) 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 its 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 its 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 cards 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 Section 7.14 on page 66.)

7.4 Interrupt mode

The interrupt mode on the MultiMediaCard system enables the master (MultiMediaCard host) to grant the transmission allowance to the slaves (card) 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 card.

The system behavior during the interrupt mode is described in the state diagram in Figure 21.

- The host must ensure that the card is in *Stand-by* State before issuing the GO_IRQ_STATE (CMD40) command. While waiting for an interrupt response from the card, the host must keep the clock signal active. Clock rate may be changed according to the required response time.
- The host sets the card into interrupt mode using GO_IRQ_STATE (CMD40) command.
- A card in Wait-IRQ-State is 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 card is also waiting for a start bit on the command line. Upon detection of a start bit, the card will abort interrupt mode and switch to the *stand-by* state.
- Regardless of winning or losing bus control during CMD40 response, the cards switches to *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.

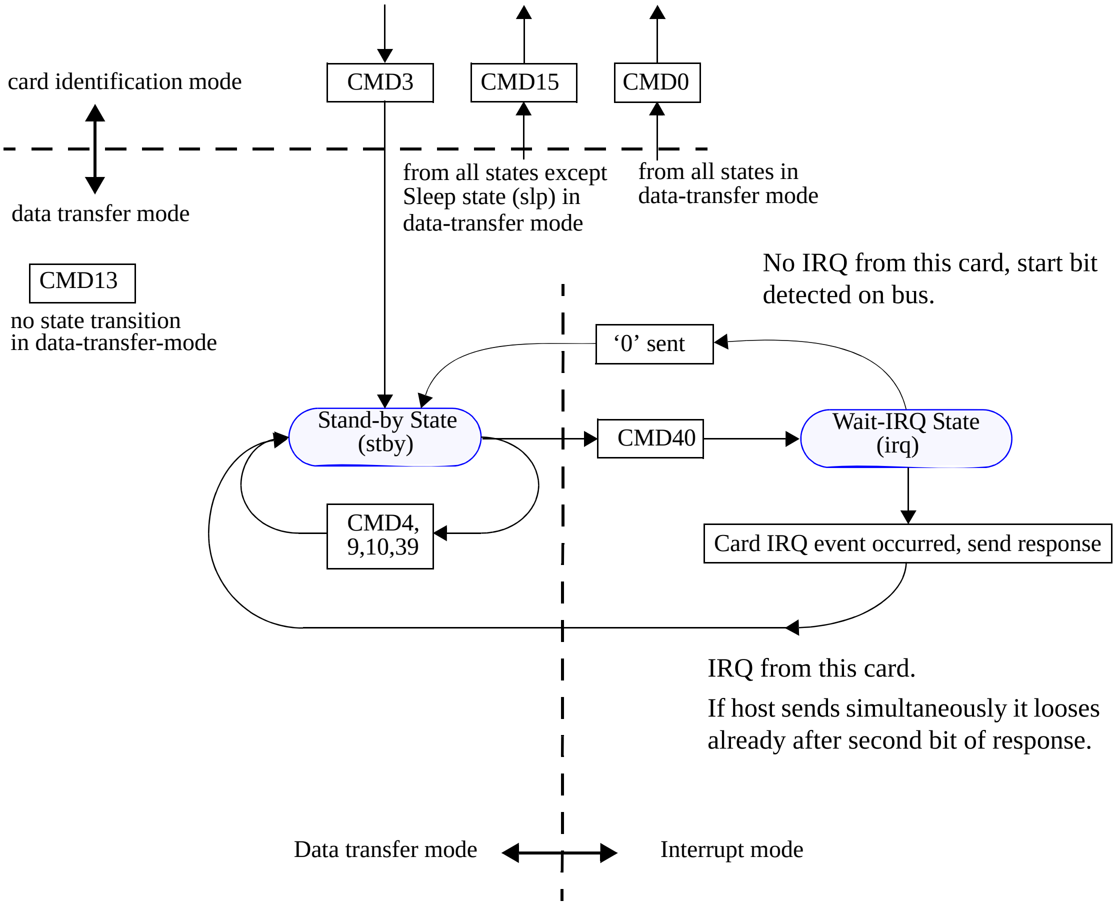

card identification mode
data transfer mode
CMD3
CMD15
CMD0
from all states except Sleep state (slp) in data-transfer mode
from all states in data-transfer mode
CMD13
no state transition in data-transfer-mode
No IRQ from this card, start bit detected on bus.
'0' sent
Stand-by State (stby)
CMD40
Wait-IRQ State (irq)
CMD4, 9,10,39
Card IRQ event occurred, send response
IRQ from this card.
If host sends simultaneously it looses already after second bit of response.
Data transfer mode
Interrupt mode

Figure 21 — 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 0x0000; This will bring the card from Wait-IRQ-State back into the Stand-by-State. Now the host can resume the standard communication procedure.

7.5 Data transfer mode

When the card is in *Stand-by State*, communication over the CMD and DAT lines will be performed in push-pull mode. Until the contents of the CSD register is known by the host, the f_{PP} clock rate must remain at $f_{OD.}$ (See Section 12.7 on page 121.) 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.

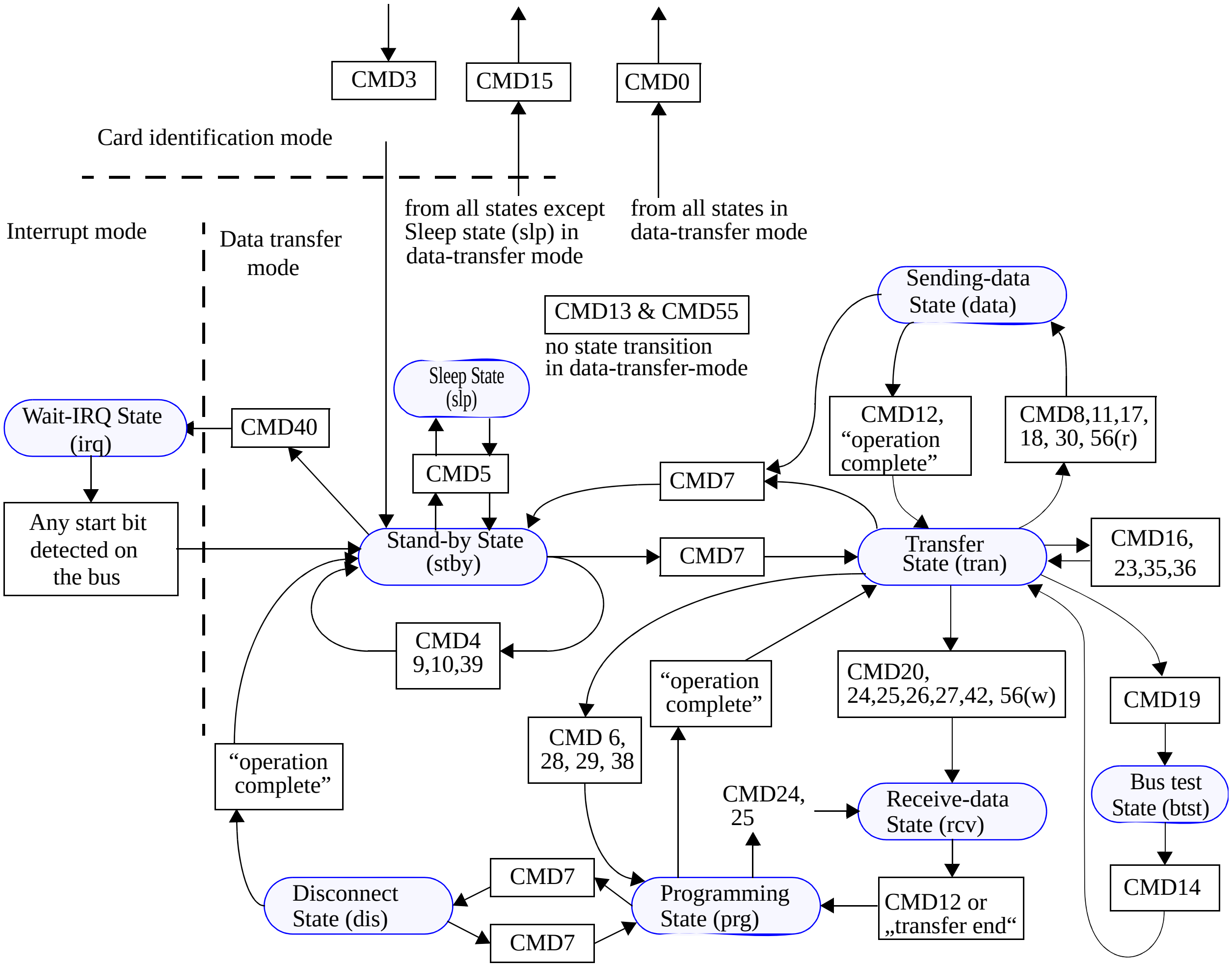


Figure 22 — MultiMediaCard state diagram (data transfer mode)

Note: The busy (Dat0=low) is always active during the prg-state. Due to legacy reasons, a card may still treat CMD24/25 during prg-state (while busy is active) as a legal or illegal command. A host should not send CMD24/25 while the card is in the prg state and busy is active.

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

While the card is in Stand-by State, CMD7 is used to select the card and put it into the *Transfer State* by including card’s relative address in the argument. If the card was previously selected and was in *Transfer State* its connection with the host is released and it will move back to the *Stand-by State* when deselected by CMD7 with any address in the argument that is not equal to card’s own relative address. When CMD7 is issued with the reserved relative card address “0x0000”, the card is put back to *Stand-by State*. Reception of CMD7 with card’s own relative address while the card is in Transfer State is ignored by the card and may be treated as an Illegal Command. After the card is assigned an RCA it will not respond to identification commands — CMD1, CMD2, or CMD3. (See Section 7.3.5 on page 29).

While the card is in Disconnect State, CMD7 is used to select the card and put it into the *Programming*

State by including card's relative address in the argument. If the card was previously selected and was in *Programming State* its connection with the host is released and it will move back to the *Disconnect State* when deselected by CMD7 with any address in the argument that is not equal to card's own relative address. Reception of CMD7 with card's own relative address while the card is in Programming State is ignored by the card and may be treated as an Illegal Command.

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 22):

- 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.
- 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. DAT0 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 DAT0 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.
- Prior to executing the bus testing procedure (CMD19, CMD14), it is recommended to set up the clock frequency used for data transfer. This way the bus test gives a true result, which might not be the case if the bus testing procedure is performed with lower clock frequency than the data transfer frequency.

In the following format definitions, all upper case flags and parameters are defined in the CSD (Section 8.3 on page 79), and the other status flags in the Card Status (Section 7.12 on page 61).

7.5.1 Command sets and extended settings

The card operates in a given command set, by default, after a power cycle or reset by CMD0, it is the MultiMediaCard standard command set, using a single data line, DAT0. The host can change the active command set by issuing the SWITCH command (CMD6) with the 'Command Set' access mode selected.

The supported command sets, as well as the currently selected command set, are defined in the EXT_CSD register. The EXT_CSD register is divided in two segments, a Properties segment and a Modes segment. The Properties segment contains information about the card capabilities. The Modes segment reflects the current selected modes of the card.

The host reads the EXT_CSD register by issuing the SEND_EXT_CSD command. The card sends the EXT_CSD register as a block of data, 512 bytes long. Any reserved, or write only field, reads as ‘0’.

The host can write the Modes segment of the EXT_CSD register by issuing a SWITCH command and setting one of the access modes. All three modes access and modify one of the EXT_CSD bytes, the byte pointed by the Index field.

NOTE 1: The Index field can contain any value from 0–255, but only values 0–191 are valid values. If the Index value is in the 192-255 range the card does not perform any modification and the SWITCH_ERROR status bit is set.

Table 6 — EXT_CSD access mode

Access Bits	Access Name	Operation
00	Command Set	The command set is changed according to the Cmd Set field of the argument
01	Set Bits	The bits in the pointed byte are set, according to the ‘1’ bits in the Value field.
10	Clear Bits	The bits in the pointed byte are cleared, according to the ‘1’ bits in the Value field.
11	Write Byte	The Value field is written into the pointed byte.

The SWITCH command can be used either to write the EXT_CSD register or to change the command set. If the SWITCH command is used to change the command set, the Index and Value field are ignored, and the EXT_CSD is not written. If the SWITCH command is used to write the EXT_CSD register, the Cmd Set field is ignored, and the command set remains unchanged.

The SWITCH command response is of type R1b, therefore, the host should read the card status, using SEND_STATUS command, after the busy signal is de-asserted, to check the result of the SWITCH operation.

7.5.2 High-speed mode selection

After the host verifies that the card complies with version 4.0, or higher, of this standard, it has to enable the high speed mode timing in the card, before changing the clock frequency to a frequency higher than 20MHz.

After power-on, or software reset, the interface timing of the card is set as specified in Table 85 on page 122. For the host to change to a higher clock frequency, it has to enable the high speed interface timing. The host uses the SWITCH command to write 0x01 to the HS_TIMING byte, in the Modes segment of the EXT_CSD register.

The valid values for this register are defined in "HS_TIMING" on page 97. If the host tries to write an invalid value, the HS_TIMING byte is not changed, the high speed interface timing is not enabled, and the SWITCH_ERROR bit is set.

7.5.3 Power class selection

After the host verifies that the card complies with version 4.0, or higher, of this standard, it may change the power class of the card.

After power-on, or software reset, the card power class is class 0, which is the default, minimum current consumption class for the card type, either High Voltage or Dual voltage card. The PWR_CL_ff_vvv bytes, in the EXT_CSD register, reflect the power consumption levels of the card, for a 4 bits bus, an 8 bit bus, at the supported clock frequencies (26MHZ or 52MHz).

The host reads this information, using the SEND_EXT_CSD command, and determines if it will allow the card to use a higher power class. If a power class change is needed, the host uses the SWITCH command to write the POWER_CLASS byte, in the Modes segment of the EXT_CSD register.

The valid values for this register are defined in "PWR_CL_ff_vvv" on page 94. If the host tries to write an invalid value, the POWER_CLASS byte is not changed and the SWITCH_ERROR bit is set.

7.5.4 Bus testing procedure

By issuing commands CMD19 and CMD14 the host can detect the functional pins on the bus. In a first step, the host sends CMD19 to the card, followed by a specific data pattern on each selected data lines. The data pattern to be sent per data line is defined in the table below. As a second step, the host sends CMD14 to request the card to send back the reversed data pattern. With the data pattern sent by the host and with the reversed pattern sent back by the card, the functional pins on the bus can be detected.

Table 7 — Bus testing pattern

Start Bit	**Data Pattern**	**End bit**
0	1 0 x x x x … x x	1

The card ignores all but the two first bits of the data pattern. Therefore, the card buffer size is not limiting the maximum length of the data pattern. The minimum length of the data pattern is two bytes, of which the first two bits of each data line are sent back, by the card, reversed. The data pattern sent by the host may optionally include a CRC16 checksum, which is ignored by the card.

The card detects the start bit on DAT0 and synchronizes accordingly the reading of all its data inputs.

The host ignores all but the two first bits of the reverse data pattern. The length of the reverse data pattern is eight bytes and is always sent using all the card's DAT lines (See Table 8 through Table 9 on page 36.) The reverse data pattern sent by the card may optionally include a CRC16 checksum, which is ignored by the host.

The card has internal pull ups in DAT1–DAT7 lines. In cases where the card is connected to only a 1-bit or a 4-bit HS-MMC system, the input value of the upper bits (e.g. DAT1–DAT7 or DAT4–DAT7) are detected as logical “1” by the card.

Table 8 — 1-bit bus testing pattern

Data line	**Data pattern sent by the host**	**Reversed pattern sent by the card**	**Notes**
DAT0	0, 10xxxxxxxxxx, [CRC16], 1	0, 01000000, [CRC16], 1	Start bit defines beginning of pattern
DAT1		0, 00000000, [CRC16], 1	No data pattern sent
DAT2		0, 00000000, [CRC16], 1	No data pattern sent
DAT3		0, 00000000, [CRC16], 1	No data pattern sent
DAT4		0, 00000000, [CRC16], 1	No data pattern sent

Table 8 — 1-bit bus testing pattern (continued)

Data line	Data pattern sent by the host	Reversed pattern sent by the card	Notes
DAT5		0, 00000000, [CRC16], 1	No data pattern sent
DAT6		0, 00000000, [CRC16], 1	No data pattern sent
DAT7		0, 00000000, [CRC16], 1	No data pattern sent

Table 9 — 4-bit bus testing pattern

Data line	Data pattern sent by the host	Reversed pattern sent by the card	Notes
DAT0	0, 10xxxxxxxxxx, [CRC16], 1	0, 01000000, [CRC16], 1	Start bit defines beginning of pattern
DAT1	0, 01xxxxxxxxxx, [CRC16], 1	0, 10000000, [CRC16], 1	
DAT2	0, 10xxxxxxxxxx, [CRC16], 1	0, 01000000, [CRC16], 1	
DAT3	0, 01xxxxxxxxxx, [CRC16], 1	0, 10000000, [CRC16], 1	
DAT4		0, 00000000, [CRC16], 1	No data pattern sent
DAT5		0, 00000000, [CRC16], 1	No data pattern sent
DAT6		0, 00000000, [CRC16], 1	No data pattern sent
DAT7		0, 00000000, [CRC16], 1	No data pattern sent

Table 10 — 8-bit bus testing pattern

Data line	Data pattern sent by the host	Reversed pattern sent by the card	Notes
DAT0	0, 10xxxxxxxxxx, [CRC16], 1	0, 01000000, [CRC16], 1	Start bit defines beginning of pattern
DAT1	0, 01xxxxxxxxxx, [CRC16], 1	0, 10000000, [CRC16], 1	
DAT2	0, 10xxxxxxxxxx, [CRC16], 1	0, 01000000, [CRC16], 1	
DAT3	0, 01xxxxxxxxxx, [CRC16], 1	0, 10000000, [CRC16], 1	
DAT4	0, 10xxxxxxxxxx, [CRC16], 1	0, 01000000, [CRC16], 1	
DAT5	0, 01xxxxxxxxxx, [CRC16], 1	0, 10000000, [CRC16], 1	
DAT6	0, 10xxxxxxxxxx, [CRC16], 1	0, 01000000, [CRC16], 1	
DAT7	0, 01xxxxxxxxxx, [CRC16], 1	0, 10000000, [CRC16], 1	

7.5.5 Bus width selection

After the host has verified the functional pins on the bus it should change the bus width configuration accordingly, using the SWITCH command.

The bus width configuration is changed by writing to the BUS_WIDTH byte in the Modes Segment of the EXT_CSD register (using the SWITCH command to do so). After power-on, or software reset, the contents of the BUS_WIDTH byte is 0x00.

The valid values for this register are defined in "BUS_WIDTH" on page 97. If the host tries to write an invalid value, the BUS_WIDTH byte is not changed and the SWITCH_ERROR bit is set. This register is write only.

7.5.6 Data read

The DAT0-DAT7 bus line levels are high when no data is transmitted. A transmitted data block consists of a start bit (LOW), on each DAT line, 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), on each DAT line. (See both Figure 30 on page 68, Figure 31 on page 69, and Figure 35 on page 71). The data transmission is synchronous to the clock signal.

The payload for block oriented data transfer is protected by a CRC check sum, on each DAT line (See Section 10.2 on page 103).

- 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 host provides an out of range address as an argument to CMD11, the card will reject the command, remain in *Tran* state and respond with the ADDRESS_OUT_OF_RANGE bit set.

Note that the stream read command works only on a 1 bit bus configuration (on DAT0). If CMD11 is issued in other bus configurations, it is regarded as an illegal 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. As the host sends CMD12 the card will respond with the ADDRESS_OUT_OF_RANGE bit set and return to *Tran* state.

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 defined in Section 8, starting on page 77,. If the host attempts to use a higher frequency, the card will not be able to sustain data transfer, and the content of the further transferred bits is undefined. As the host sends CMD12 the card will respond with the UNDERRUN bit set and return to *Tran* state.

Since the timing constrains in the CSD register are typical (not maximum) values (refer to Section 7.7.2 on page 47) 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 Section 7.7.2) 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 either one of the following conditions occur, the card will reject the command, remain in *Tran* state and respond with the respective error bit set.

- The host provides an out of range address as an argument to either CMD17 or CMD18. ADDRESS_OUT_OF_RANGE is set.
- The currently defined block length is illegal for a read operation. BLOCK_LEN_ERROR is set.
- The address/block-length combination positions the first data block misaligned to the card physical blocks. ADDRESS_MISALIGN is set.

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 then 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 is regarded 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 during the transmission of the first misaligned block and the content of the further transferred bits is undefined. As the host sends CMD12 the card will respond with the ADDRESS_MISALIGN bit set and return to *Tran* state.

If the host sets the argument of the SET_BLOCK_COUNT command (CMD23) to all 0s, then the command is accepted; however, a subsequent read will follow the open-ended multiple block read protocol (STOP_TRANSMISSION command - CMD12 - is required).

If a host had sent a CMD16 for password setting to a higher than 2GB of density of card, then this host MUST re-send CMD16 before read data transfer; otherwise, the card will response a BLK_LEN_ERROR and stay in TRANS state without data transfer since the data block (except in password application) transfer is sector unit (512B). Same error applies to up to 2GB of density of cards in case partial read access are not supported.

7.5.7 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 Section 10.2 on page 103) for each received data block prior to the write operation. By this mechanism, writing of erroneously transferred data can be prevented.

In general, an interruption to a write process should not cause corruption in existing data at any other address.

- 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 host provides an out of range address as an argument to CMD20, the card will reject the command, remain in *Tran* state and respond with the ADDRESS_OUT_OF_RANGE bit set.

Note that the stream write command works only on a 1 bit bus configuration (on DAT0). If CMD20 is issued in other bus configurations, it is regarded as an illegal command.

If the end of the memory range is reached while writing data, and no stop command has been sent yet by the host, the further transferred data is discarded. As the host sends CMD12, the card will respond with the ADDRESS_OUT_OF_RANGE bit set and return to Tran state.

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 than 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 the 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 Section 8, starting on page 77,. If the host attempts to use a higher frequency, the card may not be able to process the data and will stop programming, and while ignoring all further data transfer, wait (in the *Receive-data-State*) for a stop command. As the host sends CMD12, the card will respond with the OVERRUN bit set and return to *Tran* state

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 (see Section 7.7.2 on page 47), using the above calculated frequency may still yield and occasional OVERRUN error. In order to ensure that the card will not experience an OVERRUN situation, the maximum write latency (defined as

10x the typical -refer to Section 7.7.2) 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 DAT0 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. Three types of multiple-block write transactions are defined (the host can use any of these three types 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.

  - Reliable Write: Multiple block write with pre-defined block count and Reliable Write parameters. This transaction is similar to the basic pre-defined multiple-block write (defined in previous bullet) with the following exceptions. The old data pointed to by a logical address must remain unchanged as long as the new data written to same logical address has been successfully programmed. This is to ensure that the target address updated by the reliable write transaction never contains undefined data. Data must remain valid even if a sudden power loss occurs during the programming.

    A maximum of two different sizes of reliable write transactions are supported: 512B and the Reliable Write Sector Count parameter in EXT_CSD (REL_WR_SEC_C) multiplied by 512B.

    The function is activated by setting the Reliable Write Request parameter (bit 31) to “1” in the SET_BLOCK_COUNT command (CMD23) argument. The Reliable Write Sector Count parameter in EXT_CSD indicates the supported write sector count.

    The reliable write function is only possible under the following conditions: the length of the write operation equals the supported reliable write size or 512B, AND the reliable write request is active. Otherwise the transaction is handled as basic pre-defined multiple block case. When the length of the write operation is set to “0,” the operation is executed as a basic, open-ended, multiple-block-write case, even when the reliable write request is active.

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 either one of the following conditions occur, the card will reject the command, remain in *Tran* state and respond with the respective error bit set.

- The host provides an out of range address as an argument to either CMD24 or CMD25. ADDRESS_OUT_OF_RANGE is set.
- The currently defined block length is illegal for a write operation. BLOCK_LEN_ERROR is set.
- The address/block-length combination positions the first data block misaligned to the card physical blocks. ADDRESS_MISALIGN is set.

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 then 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 is regarded 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 during the reception of the first misaligned block, abort the write operation, and ignore all further incoming data. As the host sends CMD12, the card will respond with the ADDRESS_MISALIGN bit set and return to *Tran* state.

If the host sets the argument of the SET_BLOCK_COUNT command (CMD23) to all 0s, then the command is accepted; however, a subsequent write will follow the open-ended multiple block write protocol (STOP_TRANSMISSION command - CMD12 - is required).

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 DAT0 line low. 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 (except in Sleep state). The status bit READY_FOR_DATA indicates whether the card can accept new data or not. The host may deselect the card by issuing CMD7 which will displace the card into the *Disconnect State* and release the DAT0 line without interrupting the write operation. When reselecting the card, it will reactivate busy indication by pulling DAT0 to low. See Section 7.14 on page 66 for details of busy indication

If a host had sent a CMD16 for password setting to a higher than 2GB of density of card, then this host MUST re-send CMD16 before write data transfer; otherwise, the card will response a BLK_LEN_ERROR and stay in TRANS state without data transfer since the data block (except in password application) transfer is sector unit (512B). Same error applies to up to 2GB of density of cards in case partial write access are not supported.

7.5.8 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 when ERASE_GROUP_DEF is disabled, and in the EXT_CSD when ERASE_GROUP_DEF is enabled. The content of an explicitly erased memory range shall be ‘0’ or ‘1’ depending on different memory technology. This value is defined in the EXT_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 for densities up to 2GB, and in sector units for densities greater than 2GB. The card will ignore all LSB’s below the Erase Group size, effectively rounding the address down to the Erase Group boundary for densities up to 2GB.

If an erase command (either CMD35, CMD36, CMD38) is received out of the defined erase sequence, the card shall set the ERASE_SEQ_ERROR bit in the status register and reset the whole sequence.

If the host provides an out of range address as an argument to CMD35 or CMD36, the card will reject the command, respond with the ADDRESS_OUT_OF_RANGE bit set and reset the whole erase sequence.

If an ‘non erase’ command (neither of CMD35, CMD36, CMD38 or CMD13) is received, the card shall respond with the ERASE_RESET bit set, reset the erase sequence and execute the last command. Commands not addressed to the selected card do not abort the erase sequence.

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 DAT0 low. The actual erase time may be quite long, and the host may issue CMD7 to deselect the card.

7.5.9 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. ERASE_GROUP_DEF in EXT_CSD decides the segment size. When set to 0, the segment size is defined in units of WP_GRP_SIZE erase groups as specified in the CSD. When set to 1, the segment size is defined in units of HC_WP_GRP_SIZE erase groups as specified in the EXT_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, for densities up to 2GB, and in sector units for densities greater than 2GB. The card will ignore all LSBs below the group size for densities up to 2GB.

If the host provides an out of range address as an argument to CMD28, CMD29 or CMD30, the card will reject the command, respond with the ADDRESS_OUT_OF_RANGE bit set and remain in the *Tran* state.

7.5.10 Card lock/unlock operation

The password protection feature enables the host to lock the card by 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.

A locked card responds to (and executes) 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.

Table 11 — Lock card data structure

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								

- **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 the 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 a password replacement is attempted with PWD_LEN set to the length of the old pass-word only, the LOCK_UNLOCK_FAILED error bit is set in the status register and the old password is not changed.
  - 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 the 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 the card:
  - Select the 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 the 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 the 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.

The Force Erase time-out is specified in Section 7.7.2 on page 47.

7.5.11 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 R1.

7.5.12 Sleep (CMD5)

A card may be switched between a Sleep state and a Standby state by SLEEP/AWAKE (CMD5). In the Sleep state the power consumption of the memory device is minimized. In this state the memory device reacts only to the commands RESET (CMD0) and SLEEP/AWAKE (CMD5). All the other commands are ignored by the memory device. The timeout for state transitions between Standby state and Sleep state is defined in the EXT_CSD register S_A_timeout. The maximum current consumptions during the Sleep state are defined in the EXT_CSD registers S_A_VCC and S_A_VCCQ.

Sleep command: The bit 15 as set to 1 in SLEEP/AWAKE (CMD5) argument.

Awake command: The bit 15 as set to 0 in SLEEP/AWAKE (CMD5) argument.

The Sleep command is used to initiate the state transition from Standby state to Sleep state. The memory device indicates the transition phase busy by pulling down the DAT0 line. No further commands should be sent during the busy. The Sleep state is reached when the memory device stops pulling down the DAT0 line.

The Awake command is used to initiate the transition from Sleep state to Standby state. The memory device indicates the transition phase busy by pulling down the DAT0 line. No further commands should be sent during the busy. The Standby state is reached when the memory device stops pulling down the DAT0 line.

During the Sleep state the Vcc power supply may be switched off. This is to enable even further system power consumption saving. The Vcc supply is allowed to be switched off only after the Sleep state has been reached (the memory device has stopped to pull down the DAT0 line). The Vcc supply have to be ramped back up at least to the min operating voltage level before the state transition from Sleep state to Standby state is allowed to be initiated (Awake command).

7.6 Clock control

The MultiMediaCard bus clock signal can be used by the host to put the card 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 card, 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 DAT0 line down, forever.

7.7 Error conditions

7.7.1 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, the card shall not change its state, shall not respond 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 20 to Figure 22). Table 22 on page 57 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. CMD44).

7.7.2 Read, write, erase and force 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 Section 7.14 on page 66). 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. If ERASE_GROUP_DEF (EXT_CSD byte [175]) is enabled, ERASE_TIMEOUT_MULT should be used to calculate the duration.

- Force erase

The duration of the Force Erase command using CMD42 is specified to be a fixed time-out of 3 minutes.

7.7.3 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 ADDRESS_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.

7.8 Minimum performance

A MMCplus and MMCmobile card has to fullfill the requirements set for the read and write access performance.

7.8.1 Speed class definition

The speed class definition is for indication of the minimum performance of a card. The classes are defined based on the 150kB/s base value. The minimum performance of the card can then be marked by defined multiples of the base value e.g. 2.4MB/s. Only following speed classes are defined (note that MMCplus and MMCmobile cards are always including 8bit data bus and the categories below states the configuration with which the card is operated):

Low bus category classes (26MHz clock with 4bit data bus operation)

- 2.4 MB/s Class A
- 3.0 MB/s Class B
- 4.5 MB/s Class C
- 6.0 MB/s Class D
- 9.0 MB/s Class E

Mid bus category classes (26MHz clock with 8bit data bus or 52MHz clock with 4bit data bus operation):

- 12.0 MB/s Class F
- 15.0 MB/s Class G
- 18.0 MB/s Class H
- 21.0MB/s Class J

High bus category classes (52MHz clock with 8bit data bus operation):

- 24.0MB/s Class K
- 30.0MB/s Class M
- 36.0MB/s Class O
- 42.0MB/s Class R
- 48.0MB/s Class T

The performance values for both write and read accesses are stored into the EXT_CSD register for electrical reading (see Section 8.5 on page 99). Only the defined values and classes are allowed to be used.

7.8.2 Measurement of the performance

The procedure for the measurement of the performance of the card is defined in detail in the Compliance Documentation. Initial state of the memory in prior to the test is: filled with random data. The test is performed by writing/reading a 64kB chunk of data to/from random logical addresses (aligned to physical block boundaries) of the card. A predefined multiple block write/read is used with block count of 128 (64kB as 512B blocks are used). The performance is calculated as average out of several 64kB accesses.

Same test is performed with all applicable clock frequency and bus width options as follows:

- 52MHz, 8bit bus (if 52MHz clock frequency is supported by the card)
- 52MHz, 4bit bus (if 52MHz clock frequency is supported by the card)
- 26MHz, 8bit bus
- 26MHz, 4bit bus

In case the minimum performance of the card exceeds the physical limit of one of the above mentioned options the card has to also fulfill accordingly the performance criteria as defined in **MIN_PERF_a_b_ff** in "MIN_PERF_a_b_ff" on page 93.

7.9 Commands

7.9.1 Command types

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

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

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.

7.9.2 Command format

All commands have a fixed code length of 48 bits, needing a transmission time of 0.92 microSec @ 52 MHz.

Description	Start Bit	Transmission Bit	Command Index	Argument	CRC7	End Bit
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”

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 Section 10.2 on page 103 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 12 on page 51 through Table 21 on page 57.

7.9.3 Command classes

The command set of the MultiMediaCard system is divided into several classes. (See Table 12 on page 51.) 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 Section 13, starting on page 125, 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.

Table 12 — Supported card command classes (0–56)

Card Command Class (CCC)	Class Description	Supported commands																							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**23**	**24**	**25**
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												
		26	**27**	**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													

7.9.4 Detailed command description

The following tables define in detail all MultiMediaCard bus commands. The responses R1-R5 are defined in Section 7.11 on page 59. The registers CID, CSD, EXT_CSD and DSR are described in Section 8.

Table 13 — Basic commands and read-stream command (class 0 and class 1)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD0	bc	[31:0] stuff bits	—	GO_IDLE_STATE	Resets the card to idle state
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks the card, in idle state, to send its Operating Conditions Register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks the card to send its CID number 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 the card
CMD5	ac	[31:16] RCA [15] Sleep/Awake [14:0] stuff bits	R1b	SLEEP_AWAKE	Toggles the card between Sleep state and Standby state. (See Section 7.5.12 on page 46).
CMD6	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Switches the mode of operation of the selected card or modifies the EXT_CSD registers. (See Section 7.5.1 on page 33.)
CMD7	ac	[31:16] RCA [15:0] stuff bits	R1/R1b[1]	SELECT/DESELECT_CARD	Command toggles a card between the standby 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 the card.
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data.
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[2]	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/R1b[3]	STOP_TRANSMISSION	Forces the card to stop transmission
CMD13	ac	[31:16] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	adtc	[31:0] stuff bits	R1	BUSTEST_R	A host reads the reversed bus testing data pattern from a card.
CMD15	ac	[31:16] RCA [15:0] stuff bits	–	GO_INACTIVE_STATE	Sets the card to inactive state

Table 13 — Basic commands and read-stream command (class 0 and class 1) (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD19	adtc	[31:0] stuff bits	R1	BUSTEST_W	A host sends the bus test data pattern to a card.

NOTE 1. R1 while selecting from Stand-By State to Transfer State; R1b while selecting from Disconnected State to Programming State.
NOTE 2. Data address for media =<2GB is a 32bit byte address and data address for media > 2GB is a 32bit sector (512B) address.
NOTE 3. R1 for read cases and R1b for write cases.

Table 14 — Block-oriented read commands (class 2)

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[1]	R1	READ_SINGLE_ BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.[2]
CMD18	adtc	[31:0] data address[1]	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 is transmitted

NOTE 1. Data address for media =<2GB is a 32bit byte address and data address for media > 2GB is a 32bit sector (512B) address.
NOTE 2. The transferred data must not cross a physical block boundary, unless READ_BLK_MISALIGN is set in the CSD register.

Table 15 — Stream write commands (class 3)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD20	adtc	[31:0] data address[1]	R1	WRITE_DAT_UNTIL_ STOP	Writes a data stream from the host, starting at the given address, until a STOP_TRANSMISSION follows.
CMD21 … CMD22	reserved				

NOTE 1. Data address for media =<2GB is a 32bit byte address and data address for media > 2GB is a 32bit sector (512B) address.

Table 16 — Block-oriented write commands (class 4)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD23	ac	[31] Reliable Write Request [30:16] set to 0 [15:0] number of blocks	R1	SET_BLOCK_COUNT	Defines the number of blocks (read/write) and the reliable writer parameter (write) for a block read or write command. (See Section 7.5.6 and Section 7.5.7)
CMD24	adtc	[31:0] data address[1]	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.[2]
CMD25	adtc	[31:0] data address[1]	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. 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.

NOTE 1. Data address for media =<2GB is a 32bit byte address and data address for media > 2GB is a 32bit sector (512B) address.

NOTE 2. The transferred data must not cross a physical block boundary unless WRITE_BLK_MISALIGN is set in the CSD.

Table 17 — Block-oriented write protection commands (class 6)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD28	ac	[31:0] data address[1]	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 or HC_WP_GRP_SIZE).
CMD29	ac	[31:0] data address[1]	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.

Table 17 — Block-oriented write protection commands (class 6) (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
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.[2]
CMD31	reserved				

NOTE 1. Data address for media =<2GB is a 32bit byte address and data address for media > 2GB is a 32bit sector (512B) address.

NOTE 2. 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 lines. 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.

Table 18 — Erase commands (class 5)

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[1]	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[1]	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

NOTE 1. Data address for media =<2GB is a 32bit byte address and data address for media > 2GB is a 32bit sector (512B) address.

Table 19 — I/O mode commands (class 9)

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 if the write flag is cleared to 0. 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 20 — Lock card commands (class 7)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD42	adtc	[31:0] stuff bits.	R1	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 21 — Application-specific commands (class 8)

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]	R1	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 … CMD63	reserved for manufacturer				

NOTE 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, and their responses (if there are any), shall have a codeword length of 48 bits.

7.10 Card state transition table

Table 22 defines the card state transitions in dependency of the received command.

Table 22 — Card state transitions

	Current State												
	idle	**ready**	**ident**	**stby**	**tran**	**data**	**btst**	**rcv**	**prg**	**dis**	**ina**	slp	**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	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

Table 22 — Card state transitions (continued)

	Current State												
	idle	**ready**	**ident**	**stby**	**tran**	**data**	**btst**	**rcv**	**prg**	**dis**	**ina**	slp	**irq**
Command	**Changes to**												
Class 0 (continued)													
CMD2, card loses bus	-	ready	-	-	-	-	-	-	-	-	-	-	stby
CMD3	-	-	stby	-	-	-	-	-	-	-	-	-	stby
CMD4	-	-	-	stby	-	-	-	-	-	-	-	-	stby
CMD5	-	-	-	slp								stby	stby
CMD6	-	-	-	-	prg	-	-	-	-	-	-	-	stby
CMD7, card is addressed	-	-	-	tran	-	-	-	-	-	prg	-	-	stby
CMD7, card is not addressed	-	-	-	-	stby	stby	-	-	dis	-	-	-	stby
CMD8	-	-	-	-	data	-	-	-	-	-	-	-	stby
CMD9	-	-	-	stby	-	-	-	-	-	-	-	-	stby
CMD10	-	-	-	stby	-	-	-	-	-	-	-	-	stby
CMD12	-	-	-	-	-	tran	-	prg	-	-	-	-	stby
CMD13	-	-	-	stby	tran	data	btst	rcv	prg	dis	-	-	stby
CMD14	-	-	-	-	-	-	tran	-	-	-	-	-	stby
CMD15	-	-	-	ina	ina	ina	ina	ina	ina	ina	-	-	stby
CMD19	-	-	-	-	btst	-	-	-	-	-	-	-	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												
CMD24	-	-	-	-	rcv	-	-	-	rcv[1]	-	-	-	stby
CMD25	-	-	-	-	rcv	-	-	-	rcv[2]	-	-	-	stby
CMD26	-	-	-	-	rcv	-	-	-	-	-	-	-	stby
CMD27	-	-	-	-	rcv	-	-	-	-	-	-	-	stby
Class 6													
CMD28	-	-	-	-	prg	-	-	-	-	-	-	-	stby
CMD29	-	-	-	-	prg	-	-	-	-	-	-	-	stby

Table 22 — Card state transitions (continued)

	Current State												
	idle	**ready**	**ident**	**stby**	**tran**	**data**	**btst**	**rcv**	**prg**	**dis**	**ina**	slp	**irq**
Command	**Changes to**												
Class 6 (continued)													
CMD30	-	-	-	-	data	-	-	-	-	-	-	-	stby
Class 5													
CMD35	-	-	-	-	tran	-	-	-	-	-	-	-	stby
CMD36	-	-	-	-	tran	-	-	-	-	-	-	-	stby
CMD38	-	-	-	-	prg	-	-	-	-	-	-	-	stby
Class 7													
CMD16	see class 2												
CMD42	-	-	-	-	rcv	-	-	-	-	-	-	-	stby
Class 8													
CMD55	-	-	-	stby	tran	data	btst	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												

NOTE 1. Due to legacy considerations, a card may treat CMD24/25 during a prg state—while busy is active—as a legal or an illegal command. A card that treats CMD24/25 during a prg-state—while busy is active—as an illegal command will not change its state to the rcv state. A host should not send CMD24/25 while the card is in prg state and busy is active.

NOTE 2. Due to legacy considerations, a card may treat CMD24/25 during a prg state—while busy is active—as a legal or an illegal command. A card that treats CMD24/25 during a prg state—while busy is active—as an illegal command will not change its state to the rcv state. A host should not send CMD24/25 while the card is in prg state and busy is active.

NOTE 3. As there is no way to obtain state information in boot mode, boot-mode states are not shown in this table.

7.11 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 Section 10.2 on page 103 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 Section 7.12 on page 61.

Table 23 — R1 response

Bit position	47	46	[45:40]	[39:8]	7	0
Width (bits)	1	1	6	32	x	1
Value	“0”	“0”	x	x	CRC7	“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 DAT0. The card may become busy after receiving these commands based on its state prior to the command reception. Refer to Section 7.14 on page 66 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.

Table 24 — R2 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**	**Check bits**	**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.

Table 25 — R3 response

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**	**Check bits**	**OCR register**	**Check bits**	**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.
  The status bit in the argument is set if the operation was successful.

Table 26 — R4 response

Bit position	47	46	[45:40]	[39:8] Argument field				[7:1]	0
Width (bits)	1	1	6	16	1	7	8	7	1
Value	“0”	“0”	“100111”	x	x	x	x	x	“1”
Description	**Start bit**	**Transmission bit**	**CMD39**	**RCA [31:16]**	**Status [15]**	**Register address [14:8]**	**Read register contents [7:0]**	**CRC7**	**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.

Table 27 — R5 response

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**	**Transmission 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**

7.12 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.

Three different attributes are associated with each one of the card status bits:

- Bit type.

  Two types of card status bits are defined:

  (a) **Error bit**. Signals an error condition detected by the card. These bits are cleared as soon as the response (reporting the error) is sent out.

  (b) **Status bit**. These bits serve as information fields only, and do not alter the execution of the command being responded to. These bits are persistent, they are set and cleared in accordance with the card status.

  The “Type” field of Table 28 on page 62 defines the type of each bit in the card status register. The symbol “E” is used to denote an Error bit while the symbol “S” is used to denote a Status bit.

- Detection mode of Error bits.

  Exceptions are detected by the card either during the command interpretation and validation phase (Response Mode) or during command execution phase (Execution Mode). Response mode exceptions are reported in the response to the command that raised the exception. Execution mode exceptions are reported in the response to a STOP_TRANSMISSION command used to terminate the operation or in the response to a GET_STATUS command issued while the operation is being carried out or after the operation is completed.

  The “Det Mode” field of Table 28 defines the detection mode of each bit in the card status register. The symbol “R” is used to denote a Response Mode detection while the symbol “X” is used to denote an Execution Mode detection.

When an error bit is detected in “R” mode the card will report the error in the response to the command that raised the exception. The command will not be executed and the associated state transition will not take place. When an error is detected in “X” mode the execution is terminated. The error will be reported in the response to the next command.

The ADDRESS_OUT_OF_RANGE and ADDRESS_MISALIGN exceptions may be detected both in Response and Execution modes. The conditions for each one of the modes are explicitly defined in the table.

Table 28 — Card status

Bits	Identifier	Type	Det Mode	Value	Description	Clear Cond
31	ADDRESS_ OUT_OF_RANGE	E	R	“0” = no error “1” = error	The command’s address argument was out of the allowed range for this card.	B
			X		A multiple block or stream read/write operation is (although started in a valid address) attempting to read or write beyond the card capacity	
30	ADDRESS_ MISALIGN	E	R	“0” = no error “1” = error	The command’ s address argument (in accordance with the currently set block length) positions the first data block misaligned to the card physical blocks.	B
			X		A multiple block read/write operation (although started with a valid address/block-length combination) is attempting to read or write a data block which does not align with the physical blocks of the card.	
29	BLOCK_LEN_ ERROR	E	R	“0” = no error “1” = error	Either the argument of a SET_BLOCKLEN command exceeds the maximum value allowed for the card, or the previously defined block length is illegal for the current command (e.g. the host issues a write command, the current block length is smaller than the card’s maximum and write partial blocks is not allowed)	B
28	ERASE_SEQ_ ERROR	E	R	“0” = no error “1” = error	An error in the sequence of erase commands occurred.	B
27	ERASE_PARAM	E	X	“0” = no error “1” = error	An invalid selection of erase groups for erase occurred.	B
26	WP_VIOLATION	E	X	“0” = no error “1” = error	Attempt to program a write protected block.	B
25	CARD_IS_LOCKED	S	R	“0” = card unlocked “1” = card locked	When set, signals that the card is locked by the host	A
24	LOCK_UNLOCK_ FAILED	E	X	“0” = no error “1” = error	Set when a sequence or password error has been detected in lock/unlock card command	B
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

Table 28 — Card status (continued)

Bits	Identifier	Type	Det Mode	Value	Description	Clear Cond
21	CARD_ECC_FAILED	E	X	“0” = success “1” = failure	Card internal ECC was applied but failed to correct the data.	B
20	CC_ERROR	E	R	“0” = no error “1” = error	(Undefined by the standard) A card error occurred, which is not related to the host command.	B
19	ERROR	E	X	“0” = no error “1” = error	(Undefined by the standard) A generic card error related to the (and detected during) execution of the last host command (e.g. read or write failures).	B
18	UNDERRUN	E	X	“0” = no error “1” = error	The card could not sustain data transfer in stream read mode	B
17	OVERRUN	E	X	“0” = no error “1” = error	The card could not sustain data programming in stream write mode	B
16	CID/CSD_OVERWRITE	E	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.	B
15	WP_ERASE_SKIP	E	X	“0” = not protected “1” = protected	Only partial address space was erased due to existing write protected blocks.	B
14	Reserved, must be set to 0					
13	ERASE_RESET	E	R	“0” = cleared “1” = set	An erase sequence was cleared before executing because an out of erase sequence command was received (commands other than CMD35, CMD36, CMD38 or CMD13	B
12:9	CURRENT_STATE	S	R	0 = Idle 1 = Ready 2 = Ident 3 = Stby 4 = Tran 5 = Data 6 = Rcv 7 = Prg 8 = Dis 9 = Btst 10 = Slp 11–15 = 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 on the next command. The four bits are interpreted as a binary number between 0 and 15. As there is no way to obtain state information in boot mode, boot-mode states are not shown in this table.	A
8	READY_FOR_DATA	S	R	“0” = not ready “1” = ready	Corresponds to buffer empty signalling on the bus	A
7	SWITCH_ERROR	E	X	“0” = no error “1” = switch error	If set, the card did not switch to the expected mode as requested by the SWITCH command	B
6	Reserved					

Table 28 — Card status (continued)

Bits	Identifier	Type	Det Mode	Value	Description	Clear Cond
5	APP_CMD	S	R	“0” = Disabled “1” = Enabled	The card will expect ACMD, or indication that the command has been interpreted as ACMD	A
4	Reserved					
3:2	Reserved for Application Specific commands					
1:0	Reserved for Manufacturer Test Mode					

The following table defines, for each command responded by a R1 response, the affected bits in the status field. A “R” or a “X” mean the error/status bit may be affected by the respective command (using the R or X detection mechanism respectively). The Status bits are valid in any R1 response and are marked with “S” symbol in the table.

Table 29 — Card status field/command—cross reference

CMD #	Response 1 Format - Status bit #																					
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	13	12:9	8	7	5
0							S		R			R	X						S	S		
1									R	R			X									
2									R	R			X									
3							S		R	R		R	X						S	S		
4							S		R	R		R	X						S	S		
5							S		R	R		R	X					R	S	S		
6							S		R	R		R	X					R	S	S	X	
7							S		R	R		R	X					R	S	S		
8							S		R	R		R	X					R	S	S		
9							S		R	R		R	X					R	S	S		
10							S		R	R		R	X					R	S	S		
11	R						S		R	R	X	R	X	X				R	S	S		
12							S		R	R		R	X						S	S		
13							S		R	R		R	X						S	S		
14							S		R	R		R	X					R	S	S		
15							S		R			R	X					R	S	S		
16			R				S		R	R		R	X					R	S	S		
17	R	R	R				S		R	R	X	R	X					R	S	S		
18	R	R	R				S		R	R	X	R	X					R	S	S		
19							S		R	R		R	X					R	S	S		
20	R					X	S		R	R		R	X		X			R	S	S		
23							S		R	R		R	X					R	S	S		
24	R	R	R			X	S		R	R		R	X					R	S	S		
25	R	R	R			X	S		R	R		R	X					R	S	S		
26							S		R	R		R	X			X		R	S	S		

Table 29 — Card status field/command—cross reference (continued)

CMD #	Response 1 Format - Status bit #																					
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	13	12:9	8	7	5
27							S		R	R		R	X			X		R	S	S		
28	R						S		R	R		R	X					R	S	S		
29	R						S		R	R		R	X					R	S	S		
30	R						S		R	R		R	X					R	S	S		
35	R			R	X		S		R	R		R	X						S	S		
36	R			R	X		S		R	R		R	X						S	S		
38				R			S		R	R		R	X				X		S	S		
39							S		R	R		R	X					R	S	S		
40							S		R	R		R	X					R	S	S		
42							S	X	R	R		R	X					R	S	S		
55							S		R			R	X					R	S	S		S
56							S		R	R		R	X					R	S	S		S
Bit is valid for classes	1, 2, 3, 4, 5, 6	2, 4	2, 4, 7	5	5	3, 4	Always	7	Always	Always	1, 2	Always	Always	1	3	Always	5	Always	Always	Always	Always	Always

Not all Card status bits are meaningful all the time. Depending on the classes supported by the card, the relevant bits can be identified. If all the classes that affect a status bit, or an error bit, are not supported by the card, the bit is not relevant and can be ignored by the host.

7.13 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 when ERASE_GROUP_DEF is disabled, and in the EXT_CSD when ERASE_GROUP_DEF is enabled.
- 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 when ERASE_GROUP_DEF is disabled, and in the EXT_CSD when ERASE_GROUP_DEF is enabled.

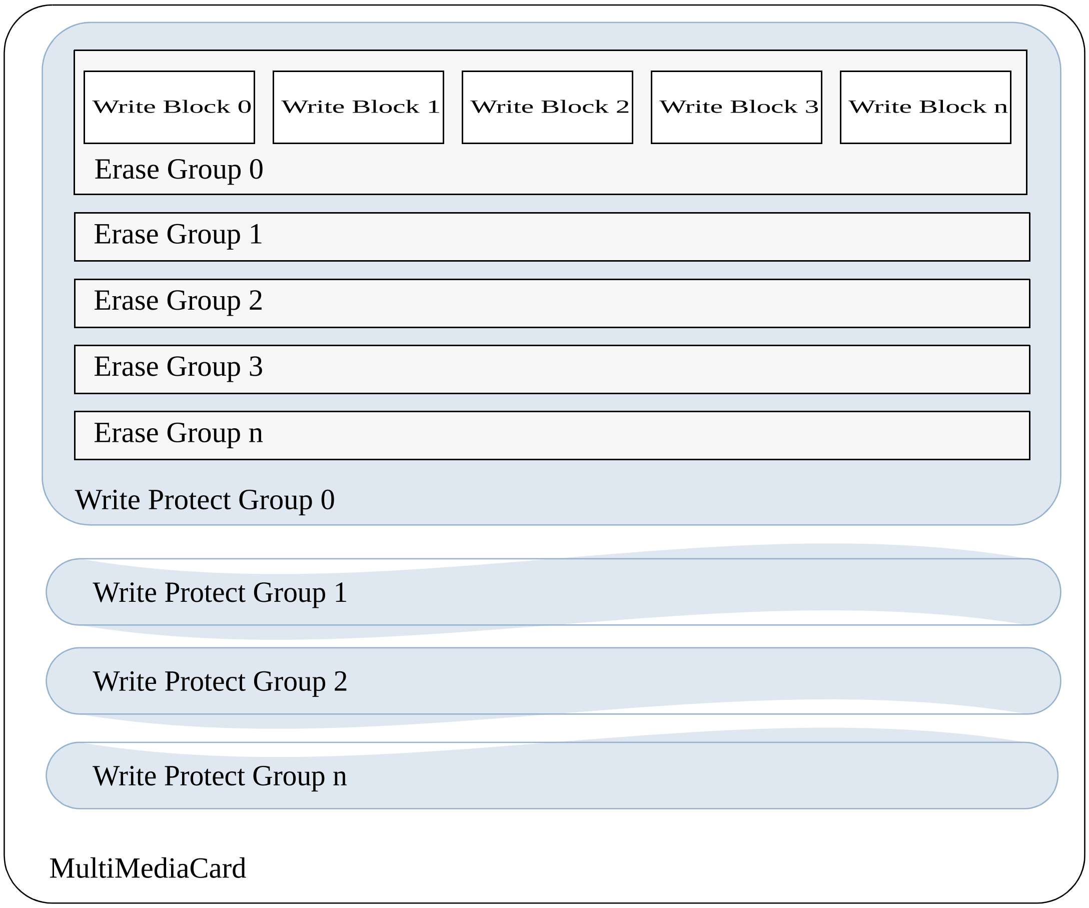


Figure 23 — Memory array partitioning

7.14 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”)
L	One-cycle pull-down (= “0”)
Z	High impedance state (-> = “1”)
X	Driven value, “1” or “0”
D	Data bits
*	Repetition
CRC	Cyclic redundancy check bits (7 bits)
🟥	Card active
🟦	Host active

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 30 on page 75.

7.14.1 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.

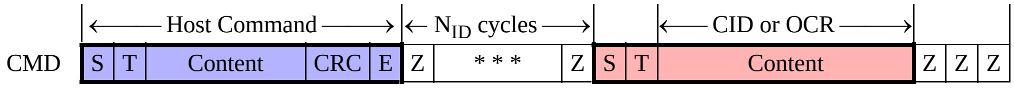


Figure 24 — Identification timing (card identification mode)

- **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.

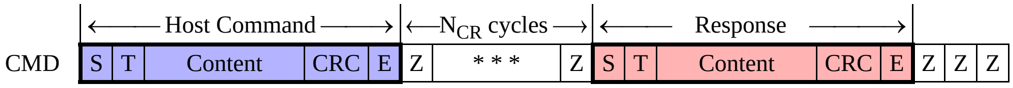


Figure 25 — SET_RCA timing (card identification mode)

- 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:

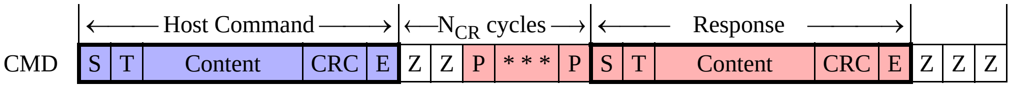


Figure 26 — Command response timing (data transfer mode)

- R1b responses

Some commands, like CMD6, may assert the BUSY signal and respond with R1. If the busy signal is asserted, it is done two clock cycles after the end bit of the command. The DAT0 line is driven low, DAT1–

DAT7 lines are driven by the card though their values are not relevant.

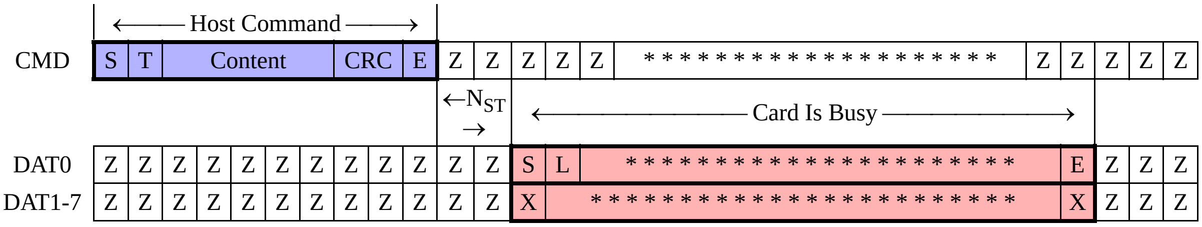


Figure 27 — R1b response timing

- 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.

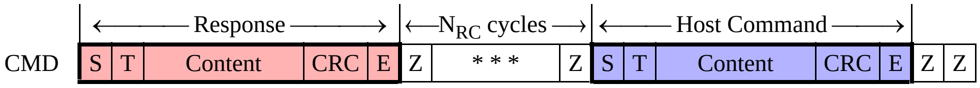


Figure 28 — Timing response end to next command start (data transfer mode)

- 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.

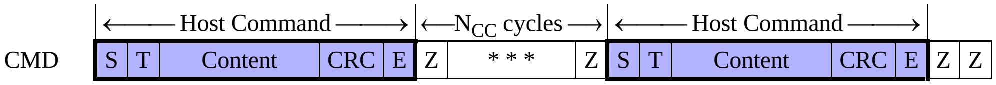


Figure 29 — Timing of command sequences (all modes)

If the ALL_SEND_CID command is not responded by the card after N_{ID} + 1 clock periods, the host can conclude there is no card present in the bus.

7.14.2 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 30. 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.

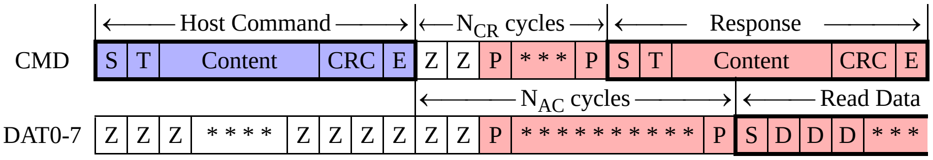


Figure 30 — Single-block read timing

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 31 describes the timing of the data blocks and Figure 32 the response to a stop command. The data transmission stops two clock cycles after the end bit of the stop command.

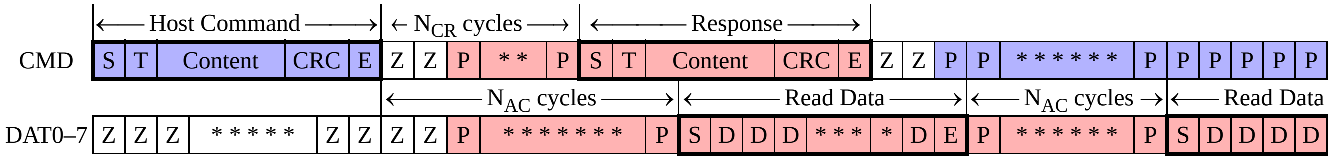

Host Command
N_{CR} cycles
Response
CMD
S T Content CRC E Z Z P ** P S T Content CRC E Z Z P P ****** P P P P P P
N_{AC} cycles
Read Data
N_{AC} cycles
Read Data
DAT0–7
Z Z Z ***** Z Z Z Z P ******* P S D D D *** * D E P ****** P S D D D D

Figure 31 — Multiple-block read timing

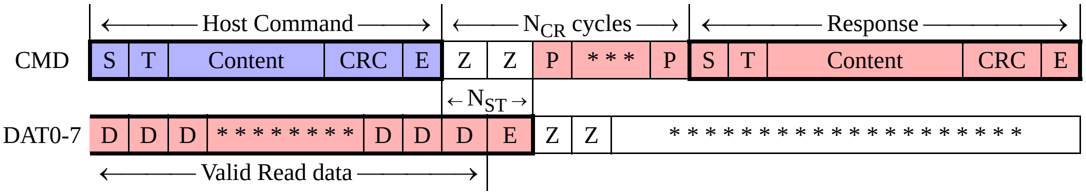

Host Command
N_{CR} cycles
Response
CMD
S T Content CRC E Z Z P *** P S T Content CRC E
N_{ST}
DAT0-7
D D D ******** D D D E Z Z ********************
Valid Read data

Figure 32 — 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 32). 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 32).

7.14.3 Data write

- Single block write

The host selects the 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 33 on page 70. 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 DAT0. In the case of transmission error, occurring on any of the active data lines, the card sends a negative CRC status (‘101’) on DAT0. In the case of successful transmission, over all active data lines, the card sends a positive CRC status (‘010’) on DAT0 and starts the data programming procedure.

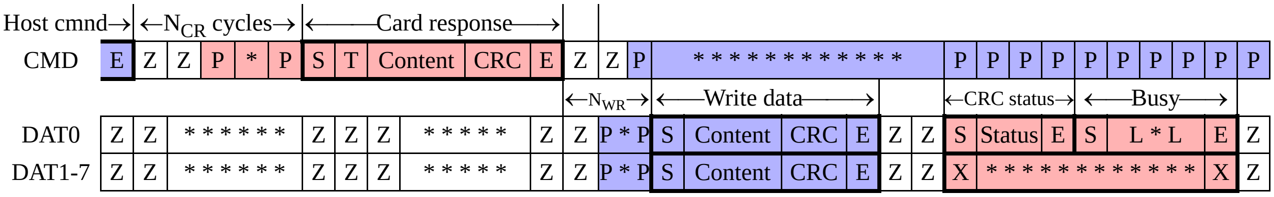

Host cmnd→
←N_{CR} cycles→
←Card response→
CMD
E Z Z P * P S T Content CRC E Z Z P * * * * * * * * * * * * P P P P P P P P P P
←N_{WR}→
←Write data→
←CRC status→
←Busy→
DAT0
Z Z * * * * * * Z Z Z * * * * * Z Z P * P S Content CRC E Z Z S Status E S L * L E Z
DAT1-7
Z Z * * * * * * Z Z Z * * * * * Z Z P * P S Content CRC E Z Z X * * * * * * * * * * * * X Z

Figure 33 — Block write command timing

While the card is programming it indicates busy by pulling down the Dat0 line. This busy status is directly related to Programming state. As soon as the card completes the programming it stops pulling down the Dat0 line.

- 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 34 describes the timing of the data blocks with and without card busy signal.

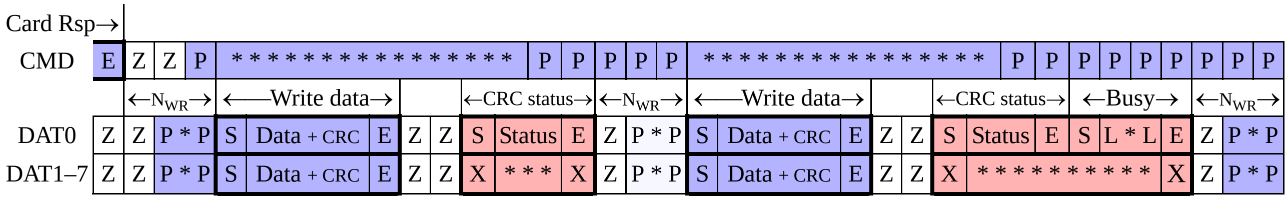

Card Rsp→
CMD
E Z Z P * * * * * * * * * * * * * * * * P P P P P * * * * * * * * * * * * * * * * P P P P P P P P P
←N_{WR}→
←Write data→
←CRC status→
←N_{WR}→
←Write data→
←CRC status→
←Busy→
←N_{WR}→
DAT0
Z Z P * P S Data + CRC E Z Z S Status E Z P * P S Data + CRC E Z Z S Status E S L * L E Z P * P
DAT1–7
Z Z P * P S Data + CRC E Z Z X * * * X Z P * P S Data + CRC E Z Z X * * * * * * * * * * X Z P * P

Figure 34 — Multiple-block write timing

The stop transmission command works similar as in the read mode. Figure 35 on page 71 to Figure 38 on page 72 describe the timing of the stop command in different card states.

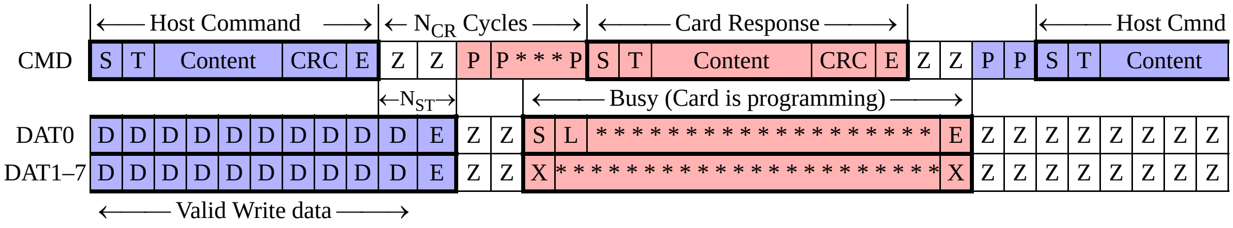
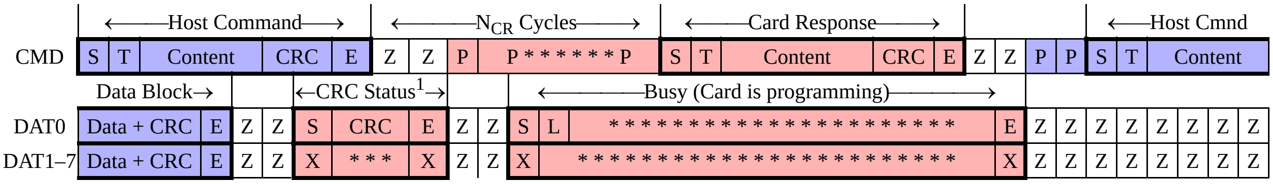

Host Command
N_CR Cycles
Card Response
Host Cmnd
CMD
S T Content CRC E Z Z P P * * * P S T Content CRC E Z Z P P S T Content
N_ST
Busy (Card is programming)
DAT0
D D D D D D D D D D E Z Z S L * * * * * * * * * * * * * * * * * * * E Z Z Z Z Z Z Z Z
DAT1–7
D D D D D D D D D D E Z Z X * X Z Z Z Z Z Z Z Z
Valid Write data

Figure 35 — 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 tokens sent back to the host. Figure 36 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 lines, with one more data bit, an end bit and two Z clocks for switching the bus direction. The received data block, in this case is considered incomplete and will not be programmed.


Host Command
N_CR Cycles
Card Response
Host Cmnd
CMD
S T Content CRC E Z Z P P * * * * * * P S T Content CRC E Z Z P P S T Content
Data Block
CRC Status1
Busy (Card is programming)
DAT0
Data + CRC E Z Z S CRC E Z Z S L * E Z Z Z Z Z Z Z Z
DAT1–7
Data + CRC E Z Z X * * * X Z Z X * X Z Z Z Z Z Z Z Z

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

Figure 36 — 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.

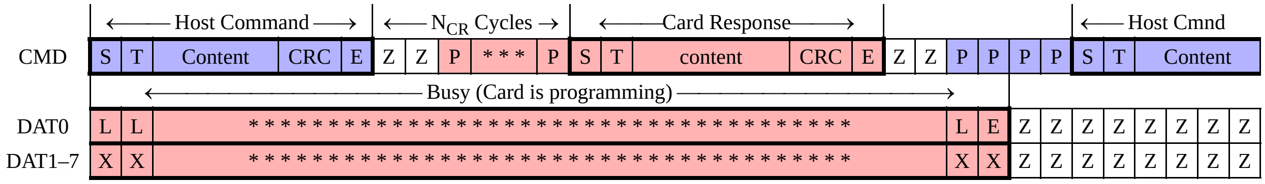

Host Command
N_CR Cycles
Card Response
Host Cmnd
CMD
S T Content CRC E Z Z P * * * P S T content CRC E Z Z P P P P S T Content
Busy (Card is programming)
DAT0
L L * L E Z Z Z Z Z Z Z Z
DAT1–7
X X * X X Z Z Z Z Z Z Z Z

Figure 37 — Stop transmission after last data block; card is busy programming

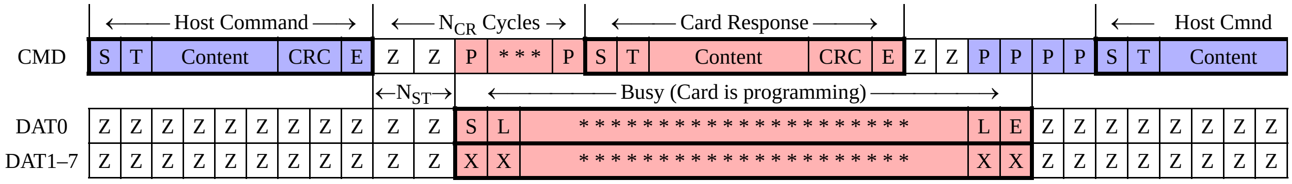

Host Command
NCR Cycles
Card Response
Host Cmnd
CMD
S T Content CRC E Z Z P * * * P S T Content CRC E Z Z P P P P S T Content
NST
Busy (Card is programming)
DAT0
Z Z Z Z Z Z Z Z Z Z Z S L ******************** L E Z Z Z Z Z Z Z Z
DAT1–7
Z Z Z Z Z Z Z Z Z Z Z X X ******************** X X Z Z Z Z Z Z Z Z

Figure 38 — Stop transmission after last data block; card becomes busy

In an open-ended multiple block write case the busy signal between the data blocks should be considered as buffer busy signal. As long as there is no free data buffer available the card should indicate this by pulling down the Dat0 line. The card stops pulling down DAT0 as soon as at least one receive buffer for the defined data transfer block length becomes free. After the card receives the stop command (CMD12), the following busy indication should be considered as programming busy and being directly related to the Programming state. As soon as the card completes the programming, it stops pulling down the Dat0 line.

In pre-defined multiple block write case the busy signal between the data blocks should be considered as buffer busy signal similar to the open-ended multiple block case. After the card receives the last data block the following busy indication should be considered as programming busy and being directly related to the Programming state. The meaning of busy signal (from buffer busy to programming busy) changes at the same time with the state change (from rcv to prg). The busy signal remains “low” all the time during the process and is not released by the card between the state change from rcv to prg. As soon as the card completes the programming, it stops pulling down the Dat0 line.

- 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 33 on page 70.) 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 35 on page 71.)

- 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 DAT0 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 38.

- 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 DAT0. The timing diagram for this command / response / busy transaction is given in Figure 38.

7.14.4 Bus test procedure timing

After reaching the Tran-state a host can initiate the Bus Testing procedure. If there is no response to the CMD19 sent by the host, the host should read the status from the card with CMD13. If there was no response to CMD19, the host may assume that this function is not supported by the card.

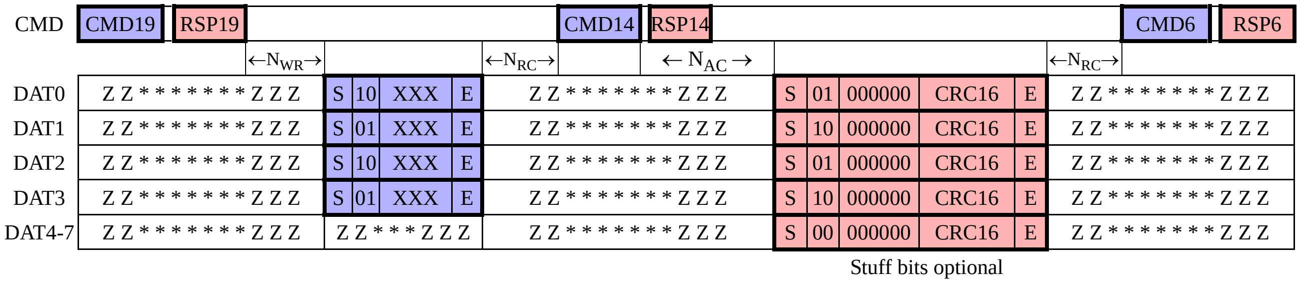
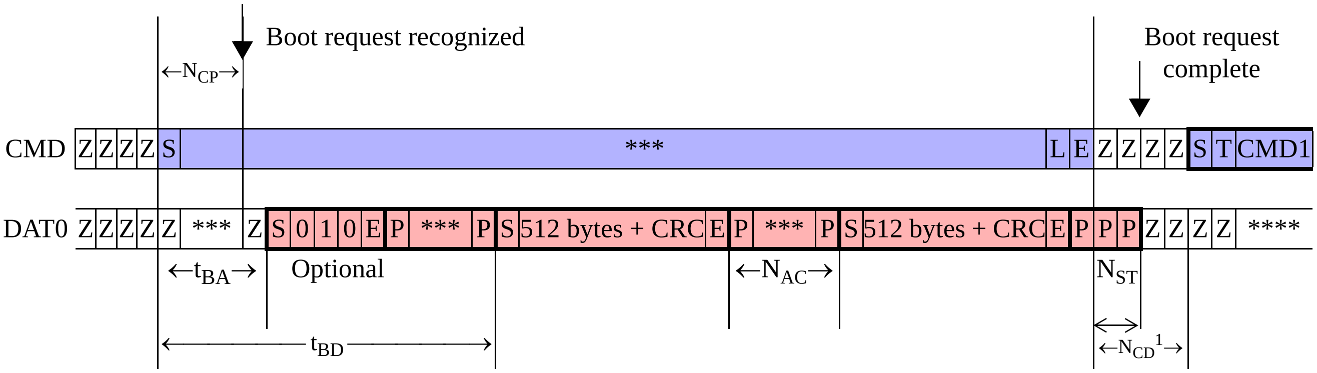


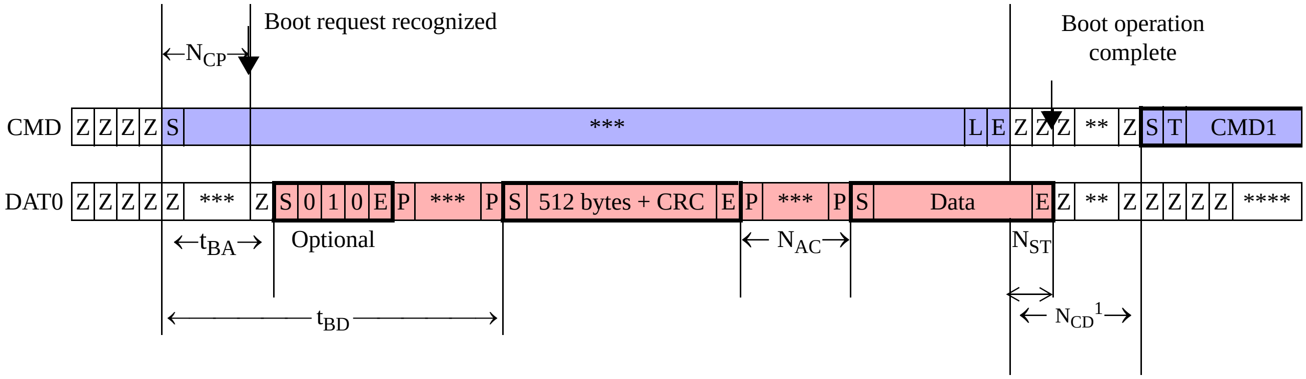
Figure 39 — Bus test procedure timing

7.14.5 Boot operation



NOTE 1. Also refer to Figure 42 on page 74.

Figure 40 — Boot operation, termination between consecutive data blocks



NOTE 1. Also refer to Figure 42 on page 74.

Figure 41 — Boot operation, termination during transfer

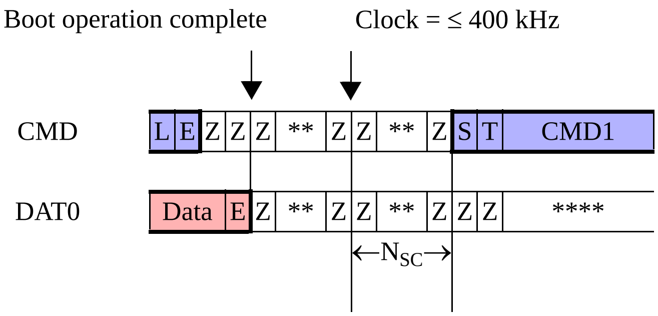
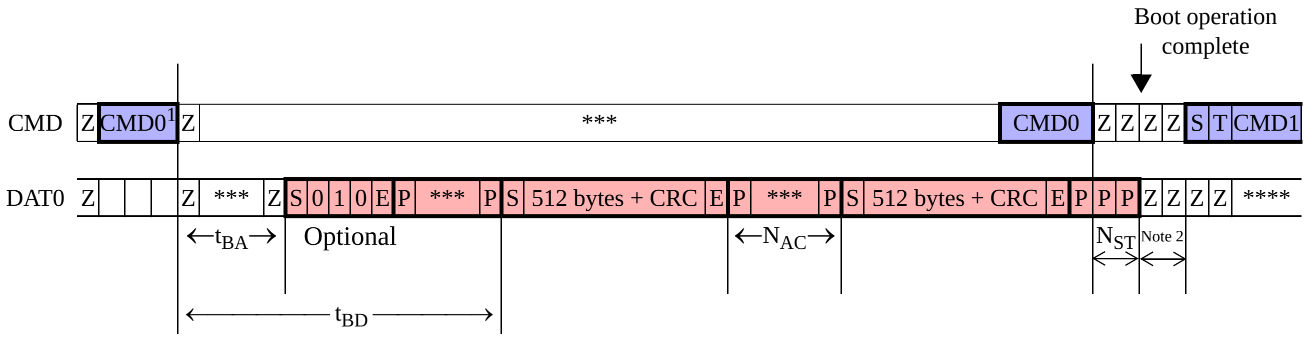


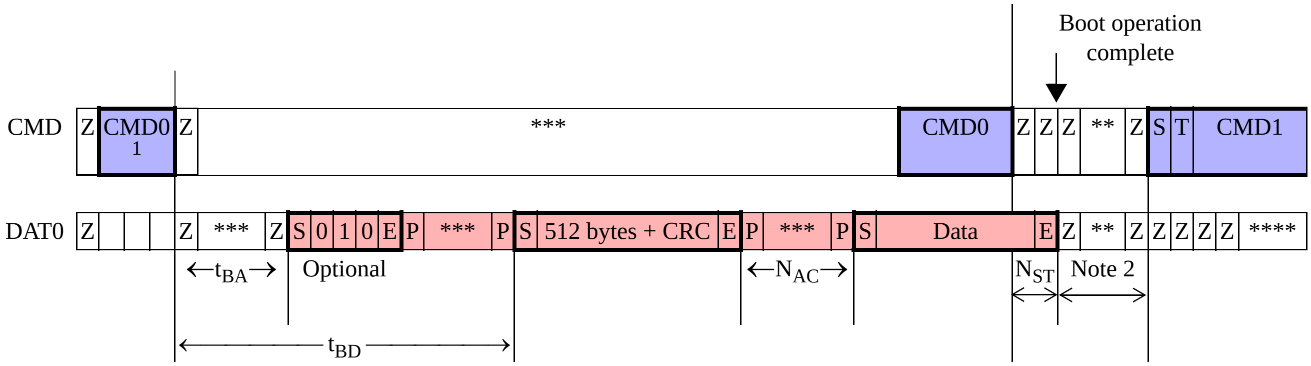
Figure 42 — Bus mode change timing (push-pull to open-drain)

7.14.6 Alternative boot operation (device optional)



NOTE 1. CMD0 with argument 0xFFFFFFFA.
NOTE 2. Refer to Figure 42.

Figure 43 — Alternative boot operation, termination between consecutive data blocks



NOTE 1. CMD0 with argument 0xFFFFFFFA.
NOTE 2. Refer to Figure 42 on page 74.

Figure 44 — Alternative boot operation, termination during transfer

7.14.7 Timing values

Table 30 — Timing parameters

Symbol	Min	Max	Unit
N_{AC}	2	10 * (TAAC * F_{OP} + 100 * NSAC)[1]	Clock cycles
N_{CC}	8	-	Clock cycles
N_{CD}	56	-	Clock cycles
N_{CP}	74	-	Clock cycles
N_{CR}	2	64	Clock cycles
N_{ID}	5	5	Clock cycles
N_{RC}	8	-	Clock cycles
N_{SC}	8	-	Clock cycles
N_{ST}	2	2	Clock cycles
N_{WR}	2	-	Clock cycles
^{t}BA	-	50	ms
^{t}BD	-	1	s

NOTE 1. FOP 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 FOP is 10MHz
$N_{AC} = 10 \times (1.5\times10^{-3} \times 10\times10^{6} + 0) = 150{,}000$ clock cycles

8 Card registers

Within the card interface six registers are defined: OCR, CID, CSD, EXT_CSD, RCA and DSR. These can be accessed only by corresponding commands (see Section 7.9 on page 49). 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. The EXT_CSD register carries both, card specific information and actual configuration parameters.

8.1 OCR register

The 32-bit operation conditions register (OCR) stores the V_{DD} voltage profile of the card and the access mode indication. 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.

Table 31 — OCR register definitions

OCR bit	**VDD voltage window**	**High Voltage MultimediaCard**	**Dual voltage MultiMediaCard and eMMC**
[6:0]	Reserved	000 0000b	00 00000b
[7]	1.70–1.95V	0b	1b
[14:8]	2.0–2.6V	000 0000b	000 0000b
[23:15]	2.7–3.6V	1 1111 1111b	1 1111 1111b
[28:24]	Reserved	000 0000b	000 0000b
[30:29]	Access mode	00b (byte mode) 10b (sector mode)	00b (byte mode) 10b (sector mode)
[31]	(card power up status bit (busy)[1]		

[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 31, for high-voltage MultiMediaCards, dual-voltage MultiMediaCards, and eMMC. As long as the card is busy, the corresponding bit (31) is set to LOW, the ‘wired-and’ operation, described in Section 7.3.2 on page 28 yields LOW, if at least one card is still busy.

8.2 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. Table 32 on page 78 lists these identifiers.

The structure of the CID register is defined in the following sections.

Table 32 — CID fields

Name	Field	Width	CID-slice
Manufacturer ID	MID	8	[127:120]
Reserved		6	[119:114]
Card/BGA	CBX	2	[113:112]
OEM/Application ID	OID	8	[111:104]
Product name	PNM	48	[103:56]
Product revision	PRV	8	[55:48]
Product serial number	PSN	32	[47:16]
Manufacturing date	MDT	8	[15:8]
CRC7 checksum	CRC	7	[7:1]
not used, always “1”	-	1	[0:0]

- 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/JEDEC. This procedure is established to ensure uniqueness of the CID register.

- CBX

CBX indicates the device type.

Table 33 — Device types

[113:112]	Type
00	Card (removable)
01	BGA (embedded)
10, 11	Reserved

- OID

A **8**-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/JEDEC. 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-bit 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 Section 10.

8.3 CSD register

The Card-Specific Data (CSD) 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 CSD Registry entries in the Table 34 below is coded as follows:
R = readable, W = writable once, E = erasable (multiple writable).

Table 34 — CSD fields

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. bus clock frequency	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]

Table 34 — CSD fields (continued)

Name	Field	Width	Cell Type	CSD-slice
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]

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.

Table 35 — CSD register structure

CSD_STRUCTURE	CSD Structure Version	Valid for System Specification Version
0	CSD version No. 1.0	Allocated by MMCA
1	CSD version No. 1.1	Allocated by MMCA
2	CSD version No. 1.2	Version 4.1–4.2–4.3
3	Version is coded in the CSD_STRUCTURE byte in the EXT_CSD register	

- SPEC_VERS

Defines the MultiMediaCard System Specification version supported by the card.

Table 36 — System specification version

SPEC_VERS	System Specification Version Number
0	Allocated by MMCA
1	Allocated by MMCA
2	Allocated by MMCA
3	Allocated by MMCA

Table 36 — System specification version (continued)

SPEC_VERS	System Specification Version Number
4	Version 4.1–4.2–4.3
5–15	Reserved

- TAAC

Defines the asynchronous part of the data access time.

Table 37 — TAAC access-time definition

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	Multiplier factor 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

- 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 30 on page 75 is calculated based on TAAC and NSAC. It has to be computed 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 clock frequency when not in high speed mode. For cards supporting version 4.0, 4.1, and 4.2 of the specification, the value shall be 20MHz (0x2A). For cards supporting version 4.3, the value shall be 26 MHz (0x32).

Table 38 — Maximum bus clock frequency definition

TRAN_SPEED bit	Code
2:0	Frequency unit 0 = 100KHz, 1 = 1MHz, 2 = 10MHz, 3 = 100MHz, 4…7 = reserved
6:3	Multiplier factor 0 = reserved, 1 = 1.0, 2 = 1.2, 3 = 1.3, 4 = 1.5, 5 = 2.0, 6 = 2.6, 7 = 3.0, 8 = 3.5, 9 = 4.0, A = 4.5, B = 5.2, C = 5.5, D = 6.0, E = 7.0, F = 8.0
7	reserved

- 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 12 on page 51.

Table 39 — Supported card command classes

CCC bit	Supported Card Command Class
0	class 0
1	class 1
…	
11	class 11

- READ_BL_LEN

The data block length is computed as $2^{READ\_BL\_LEN}$. The block length might therefore be in the range 1B, 2B,4B...16kB. (See Section 7.12 on page 61 for details.)

Note that the support for 512B read access is mandatory for all cards. And that the cards has to be in 512B block length mode by default after power-on, or software reset. The purpose of this register is to indicate the supported maximum read data block length.

:

Table 40 — Data block length

READ_BL_LEN	Block length	Remark
0	2^0 = 1 Byte	
1	2^1 = 2 Bytes	
…		
11	2^{11} = 2048 Bytes	
12	2^{12} = 4096 Bytes	
13	2^{13} = 8192 Bytes	
14	2^{14} = 16 kBytes	
15	2^{15} = Extension	New register TBD to EXT_CSD

- READ_BL_PARTIAL

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

Up to 2GB of density (byte access mode):

READ_BL_PARTIAL=0 means that only the 512B and 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).

Higher than 2GB of density (sector access mode):

READ_BL_PARTIAL=0 means that only the 512B and the READ_BL_LEN block sizes can be used for block oriented data transfers.

READ_BL_PARTIAL=1 means that smaller blocks than indicated in READ_BL_LEN can be used as well. The minimum block size will be equal to minimum addressable unit, one sector (512B).

- 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 Section 8.6 on page 99.).

Table 41 — DSR implementation code table

DSR_IMP	**DSR type**
0	DSR is not implemented
1	DSR implemented

- C_SIZE

This parameter is used to compute the card capacity for cards up to 2GB of density. Please see "SEC_COUNT" on page 93 for densities greater than 2GB. Note that for card densities greater than 2GB, the maximum possible value should be set to this register (0xFFF).

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:

Table 42 — V_{DD} (min) current consumption

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

The values in these fields are valid when the card is not in high speed mode. When the card is in high speed mode, the current consumption is chosen by the host, from the power classes defined in the PWR_ff_vvv registers, in the EXT_CSD register.

- 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:

Table 43 — V_{DD} (max) current consumption

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

The values in these fields are valid when the card is not in high speed mode. When the card is in high speed mode, the current consumption is chosen by the host, from the power classes defined in the PWR_ff_vvv registers, in the EXT_CSD register.

- C_SIZE_MULT

Note that for higher than 2GB of density of card the maximum possible value should be set to this register (0x7).This parameter is used for coding a factor MULT for computing the total device size (see ‘C_SIZE’). The factor MULT is defined as $2^{C\_SIZE\_MULT+2}$.

Table 44 — Multiplier factor for device size

C_SIZE_MULT	MULT	Remarks
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	$2^9 = 512$	

- 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.

Table 45 — R2W_FACTOR

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

- WRITE_BL_LEN

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

Note that the support for 512B write access is mandatory for all cards. And that the cards has to be in 512B block length mode by default after power-on, or software reset. The purpose of this register is to indicate the supported maximum write data block length.

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

Up to 2GB of density (byte access mode):

WRITE_BL_PARTIAL=’0’ means that only the 512B and 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.

Higher than 2GB of density (sector access mode):

WRITE_BL_PARTIAL=’0’ means that only the 512B and 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 will be equal to minimum addressable unit, one sector (512B).

- 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 46. (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:

Table 46 — File formats

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

A more detailed description is given in Section 14 starting on page 127.

- 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.

Table 47 — ECC type

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

- CRC

The CRC field carries the check sum for the CSD contents. It is computed according to Section 10.2 on page 103. 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.

Table 48 — CSD field command classes

CSD Field	Command Classes									
	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										

Table 48 — CSD field command classes (continued)

CSD Field	Command Classes									
	0	1	2	3	4	5	6	7	8	9
COPY	+	+	+	+	+	+	+	+	+	+
PERM_WRITE_PROTECT	+	+	+	+	+	+	+	+	+	+
TMP_WRITE_PROTECT	+	+	+	+	+	+	+	+	+	+
FILE_FORMAT										
ECC		+	+	+	+	+	+	+	+	
CRC	+	+	+	+	+	+	+	+	+	+

8.4 Extended CSD register

The Extended CSD register defines the card properties and selected modes. It is 512 bytes long. The most significant 320 bytes are the Properties segment, which defines the card capabilities and cannot be modified by the host. The lower 192 bytes are the Modes segment, which defines the configuration the card is working in. These modes can be changed by the host by means of the SWITCH command.

Table 49 — Extended CSD

Name	Field	Size (Bytes)	Cell Type	CSD-slice
Properties Segment				
Reserved[1]		7		[511:505]
Supported Command Sets	S_CMD_SET	1	R	[504]
Reserved[1]		275	TBD	[503:229]
Boot information	BOOT_INFO	1	R	[228]
Reserved[1]		1	TBD	[227]
Boot partition size	BOOT_SIZE_MULTI	1	R	[226]
Access size	ACC_SIZE	1	R	[225]
High-capacity erase unit size	HC_ERASE_GRP_SIZE	1	R	[224]
High-capacity erase timeout	ERASE_TIMEOUT_MULT	1	R	[223]
Reliable write sector count	REL_WR_SEC_C	1	R	[222]
High-capacity write protect group size	HC_WP_GRP_SIZE	1	R	[221]
Sleep current (VCC)	S_C_VCC	1	R	[220]
Sleep current (VCCQ)	S_C_VCCQ	1	R	[219]
Reserved[1]		1	TBD	[218]
Sleep/awake timeout	S_A_TIMEOUT	1	R	[217]
Reserved[1]		1	TBD	[216]
Sector Count	SEC_COUNT	4	R	[215:212]
Reserved[1]		1		[211]
Minimum Write Performance for 8bit at 52MHz	MIN_PERF_W_8_52	1	R	[210]
Minimum Read Performance for 8bit at 52MHz	MIN_PERF_R_8_52	1	R	[209]
Minimum Write Performance for 8bit at 26MHz, for 4bit at 52MHz	MIN_PERF_W_8_26_4_52	1	R	[208]

Table 49 — Extended CSD (continued)

Name	Field	Size (Bytes)	Cell Type	CSD-slice
Minimum Read Performance for 8bit at 26MHz, for 4bit at 52MHz	MIN_PERF_R_8_26_4_52	1	R	[207]
Minimum Write Performance for 4bit at 26MHz	MIN_PERF_W_4_26	1	R	[206]
Minimum Read Performance for 4bit at 26MHz	MIN_PERF_R_4_26	1	R	[205]
Reserved[1]		1		[204]
Power class for 26MHz at 3.6V	PWR_CL_26_360	1	R	[203]
Power class for 52MHz at 3.6V	PWR_CL_52_360	1	R	[202]
Power class for 26MHz at 1.95V	PWR_CL_26_195	1	R	[201]
Power class for 52MHz at 1.95V	PWR_CL_52_195	1	R	[200]
Reserved[1]		3		[199:197]
Card type	CARD_TYPE	1	R	[196]
Reserved[1]		1		[195]
CSD structure version	CSD_STRUCTURE	1	R	[194]
Reserved[1]		1		[193]
Extended CSD revision	EXT_CSD_REV	1	R	[192]
Modes Segment				
Command set	CMD_SET	1	R/W	[191]
Reserved[1]		1		[190]
Command set revision	CMD_SET_REV	1	RO	[189]
Reserved[1]		1		[188]
Power class	POWER_CLASS	1	R/W	[187]
Reserved[1]		1		[186]
High-speed interface timing	HS_TIMING	1	R/W	[185]
Reserved[1]		1		[184]
Bus width mode	BUS_WIDTH	1	WO	[183]
Reserved[1]		1		[182]
Erased memory content	ERASED_MEM_CONT	1	RO	[181]
Reserved[1]		1		[180]
Boot configuration	BOOT_CONFIG	1	R/W	[179]
Reserved[1]		1		[178]
Boot bus width1	BOOT_BUS_WIDTH	1	R/W	[177]
Reserved[1]		1		[176]
High-density erase group definition	ERASE_GROUP_DEF	1	R/W	[175]
Reserved[1]		175		[174:0]

NOTE 1. Reserved bits should read as “0.”

- S_CMD_SET

This field defines the command sets supported by the card.

Table 50 — Card-supported command sets

Bit	Command Set
7–5	Reserved
4	Allocated by MMCA
3	Allocated by MMCA
2	Allocated by MMCA
1	Allocated by MMCA
0	Standard MMC

- BOOT_INFO

Table 51 — Boot information

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved							ALT_BOOT_MODE

Bit[7:1]: Reserved

Bit[0]: ALT_BOOT_MODE

0: Device does not support alternate boot method

1: Device supports alternate boot method.

- BOOT_SIZE_MULT

The boot partition size is calculated from the register by using the following equation:

$$\text{Boot partition size} = 128\text{Kbytes} \times \text{BOOT\_SIZE\_MULT}$$

Table 52 — Boot partition size

Value	Timeout Values
0x00	No boot partition available / Boot mode not supported
0x01	1 × 128Kbytes = 128Kbytes
0x02	2 × 128Kbytes = 256Kbytes
:	:
0xFE	254 × 128Kbytes = 32512Kbytes
0xFF	255 × 128Kbytes = 32640Kbytes

- ACC_SIZE

Table 53 — Access size

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved				SUPER_PAGE_SIZE			

Bit[7:4]: Reserved

Bit[3:0]: SUPER_PAGE_SIZE

This register defines one or multiple of programmable boundary unit which is programmed at the same time. This value can be used by the master for the following cases:

As a guide for format clusters

To prevent format-page misalignment

As a guide for minimum data-transfer size

Super-page size = 512 × $2^{(\mathrm{SUPER\_PAGE\_SIZE}\ -\ 1)}$: $0 < X < 9$

Table 54 — Superpage size

Value	Timeout Values
0x0	Not defined
0x1	512 × 1 = 512 bytes
0x2	512 × 2 = 1K bytes
:	:
0x8	512 × 128 = 64K bytes
0x9–0xF	Reserved

- HC_ERASE_GRP_SIZE

This register defines the erase-unit size for high-capacity memory. If the master enables bit “0” in the extended CSD register byte [175], the slave uses these value for the erase operation.

Erase Unit Size = 512Kbyte × HC_ERASE_GRP_SIZE

Table 55 — Erase-unit size

Value	Timeout Values
0x00	No support for high-capacity erase-unit size
0x01	512Kbyte × 1 = 524,288 bytes
0x02	512Kbyte × 2 = 1,048,576 bytes
:	:
0xFF	512Kbyte × 255 = 133,693,440 bytes

If the ENABLE bit in ERASE_GROUP_DEF is cleared to LOW or HC_WP_GRP_SIZE is set to 0x00, the write protect group size definition would be the original case.

- ERASE_TIMEOUT_MULT

This register is used to calculate erase timeout for high-capacity erase operations and defines the timeout value for the erase operation of one erase group.

Erase Timeout = 300ms × ERASE_TIMEOUT_MULT

If the host executes erase operations for multiple erase groups, the total timeout value should be the multiple of the number of erase groups issued.

If the master enables bit 0 in the extended CSD register byte [175], the slave uses ERASE_TIMEOUT_MULT values for the timeout value.

If ERASE_TIMEOUT_MULT is set to 0x00, the slave must support the previous timeout definition.

Table 56 — Erase timeout values

Value	Timeout Values
0x00	No support for high-capacity erase timeout
0x01	300ms × 1 = 300ms
0x02	300ms × 2 = 600ms
:	:
0xFF	300ms × 255 = 76,500ms

- REL_WR_SEC_C

The reliable write feature requires mandatory sector count 1 (512B) support.

With this register it is also possible to indicate an additional supported sector count.

In applications where only the single-sector write is supported, the value in the register should be “1.” Otherwise, the value should be the multiple of the number of sectors supported.

Table 57 — Reliable write sector count

Name	Field	Size	Cell Type
Reliable Write Sector Count	REL_WR_SEC_C	1	R

- HC_WP_GRP_SIZE

This register defines the write protect group size for high-capacity memory. If the ENABLE bit in ERASE_GROUP_DEF is set to HIGH, the write protect group size would be defined as follows:

Write protect group size = 512KB * HC_ERASE_GRP_SIZE * HC_WP_GRP_SIZE.

Table 58 — Write protect group size

Value	Value definition
0x00	No support for high-capacity write protect group size
0x01	1 high-capacity erase unit size
0x02	2 high-capacity erase unit size
0x03	3 high-capacity erase unit size
:	:
0xFF	255 high-capacity erase unit size

If the ENABLE bit in ERASE_GROUP_DEF is cleared to LOW or HC_WP_GRP_SIZE is set to 0x00, the write protect group size definition would be the original case.

- S_C_VCC, S_C_VCCQ

The S_C_VCC and S_C_VCCQ registers define the max VCC current consumption during the Sleep state (slp). The formula to calculate the max current value is:

Sleep current = 1µA * 2x : register value = X >0

Sleep current = no value (legacy) : register value = 0

Max register value defined is 0x0D which equals 8.192mA. Values between 0x0E and 0xFF are reserved.

Table 59 — S_C_VCC, S_C_VCCQ timeout values

Value	Timeout Values
0x00	Not defined
0x01	1µA × 2^{1} = 2µA
0x02	1µA × 2^{2} = 4µA
:	:
0x0D	1µA × 2^{13} = 8.192mA
0x0E–0xFF	Reserved

- S_A_TIMEOUT

This register defines the max timeout value for state transitions from Standby state (stby) to Sleep state (slp) and from Sleep state (slp) to Standby state (stby). The formula to calculate the max timeout value is:

Sleep/Awake Timeout = 100ns * 2S_A_timeout

Max register value defined is 0x17 which equals 838.86ms timeout. Values between 0x18 and 0xFF are reserved.

Table 60 — Sleep/awake timeout values

Value	Timeout Values
0x00	Not defined
0x01	100ns × 2^{1} = 200ns
0x02	100ns × 2^{2} = 400ns
:	:
0x17	100ns × 2^{23} = 838.86ms
0x18–0xFF	Reserved

- SEC_COUNT

The device density is calculated from the register by multiplying the value of the register (sector count) by 512B/sector. The maximum density possible to be indicated is thus 2 Tera bytes (4 294 967 296 x 512B). The least significant byte (LSB) of the sector count value is the byte [212].

- MIN_PERF_a_b_ff

These fields defines the overall minimum performance value for the read and write access with different bus width and max clock frequency modes. The value in the register is coded as follows. Other than defined values are illegal.

Table 61 — R/W access performance values

Value	Performance
0x00	For cards not reaching the 2.4MB/s value
0x08	Class A: 2.4MB/s and is the next allowed value (16x150kB/s)
0x0A	Class B: 3.0MB/s and is the next allowed value (20x150kB/s)
0x0F	Class C: 4.5MB/s and is the next allowed value (30x150kB/s)
0x14	Class D: 6.0MB/s and is the next allowed value (40x150kB/s)
0x1E	Class E: 9.0MB/s and is the next allowed value (60x150kB/s) This is also the highest class which any MMCplus or MMC mobile card is needed to support in low bus category operation mode (26MHz with 4bit data bus). A MMCplus or MMCmobile card supporting any higher class than this have to support this class also (in low category bus operation mode).
0x28	Class F: Equals 12.0MB/s and is the next allowed value (80x150kB/s)
0x32	Class G: Equals 15.0MB/s and is the next allowed value (100x150kB/s)
0x3C	Class H: Equals 18.0MB/s and is the next allowed value (120x150kB/s)
0x46	Class J: Equals 21.0MB/s and is the next allowed value (140x150kB/s) This is also the highest class which any MMCplus or MMC mobile card is needed to support in mid bus category operation mode (26MHz with 8bit data bus or 52MHz with 4bit data bus). A MMCplus or MMCmobile card supporting any higher class than this have to support this Class (in mid category bus operation mode) and Class E also (in low category bus operation mode)
0x50	Class K: Equals 24.0MB/s and is the next allowed value (160x150kB/s)
0x64	Class M: Equals 30.0MB/s and is the next allowed value (200x150kB/s)
0x78	Class O: Equals 36.0MB/s and is the next allowed value (240x150kB/s)
0x8C	Class R: Equals 42.0MB/s and is the next allowed value (280x150kB/s)
0xA0	Class T: Equals 48.0MB/s and is the last defined value (320x150kB/s)

- PWR_CL_ff_vvv

These fields define the supported power classes by the card. By default, the card has to operate at maximum frequency using 1 bit bus configuration, within the default max current consumption, as stated in the table below. If 4 bit/8 bits bus configurations, require increased current consumption, it has to be stated in these registers.

By reading these registers the host can determine the power consumption of the card in different bus modes. Bits [7:4] code the current consumption for the 8 bit bus configuration. Bits [3:0] code the current consumption for the 4 bit bus configuration.

The PWR_52_vvv registers are not defined for 26MHz MultiMediaCards.

Table 62 — Power classes

Voltage	Value	Max RMS Current	Max Peak Current	Remarks
3.6V	0	100 mA	200 mA	Default current consumption for high voltage cards
	1	120 mA	220 mA	
	2	150 mA	250 mA	
	3	180 mA	280 mA	
	4	200 mA	300 mA	
	5	220 mA	320 mA	
	6	250 mA	350 mA	
	7	300 mA	400 mA	
	8	350 mA	450 mA	
	9	400 mA	500 mA	
	10	450 mA	550 mA	
	11-15			Reserved for future use
1.95V	0	65 mA	130 mA	Default current consumption for Dual voltage cards
	1	70 mA	140 mA	
	2	80 mA	160 mA	
	3	90 mA	180 mA	
	4	100 mA	200 mA	
	5	120 mA	220 mA	
	6	140 mA	240 mA	
	7	160 mA	260 mA	
	8	180 mA	280 mA	
	9	200 mA	300 mA	
	10	250 mA	350 mA	
	11-15			Reserved for future use

The measurement for max RMS current is done as average RMS current consumption over a period of 100ms.

Max peak current is defined as absolute max value not to be exceeded at all.

The conditions under which the power classes are defined are:

- Maximum bus frequency
- Maximum operating voltage
- Worst case functional operation
- Worst case environmental parameters (temperature,...)

These registers define the maximum power consumption for any protocol operation in data transfer mode, Ready state and Identification state.

- CARD_TYPE

This field defines the type of the card.

Table 63 — Card types

Bit	Card Type
7:2	Reserved
1	High-Speed MultiMediaCard @ 52MHz
0	High-Speed MultiMediaCard @ 26MHz

The only currently valid values for this field are 0x01 and 0x03.

- CSD_STRUCTURE

This field is a continuation of the CSD_STRUCTURE field in the CSD register

Table 64 — CSD register structure

CSD_STRUCTURE	CSD structure version	Valid for System Specification Version
0	CSD version No. 1.0	Allocated by MMCA
1	CSD version No. 1.1	Allocated by MMCA
2	CSD version No. 1.2	Version 4.1–4.2–4.3
3–255	Reserved for future use	

- EXT_CSD_REV

Defines the fixed parameters related to the EXT_CSD, according to its revision

Table 65 — Extended CSD revisions

EXT_CSD_REV	Extended CSD Revision
255–4	Reserved
3	Revision 1.3
2	Revision 1.2
1	Revision 1.1
0	Revision 1.0

- CMD_SET

Contains the binary code of the command set that is currently active in the card. The command set can be changed using the Command Set-access type of the SWITCH command (CMD6). Note that while changing the command set with the SWITCH command, bit index values according to the S_CMD_SET register should be used. For backward compatibility, the CMD_SET is set to 0x00 (standard MMC) following power-up. After switching back to the standard MMC command set with the SWITCH command, the value of the CMD_SET is 0x01.

- CMD_SET_REV

Contains a binary number reflecting the revision of the currently active command set. For Standard MMC. command set it is:

Table 66 — Standard MMC command set revisions

Code	MMC Revision
255–1	Reserved
0	v4.0

This field, though in the Modes segment of the EXT_CSD, is read only.

- POWER_CLASS

This field contains the 4-bit value of the selected power class for the card. The power classes are defined in Table 67. The host should be responsible of properly writing this field with the maximum power class it allows the card to use. The card uses this information to, internally, manage the power budget and deliver an optimized performance.

This field is 0 after power-on or software reset.

Table 67 — Power class codes

Bits	Description
[7:4]	Reserved
[3:0]	Card power class code (See Table 62 on page 95)

- HS_TIMING

This field is 0 after power-on, or software reset, thus selecting the backwards compatibility interface timing for the card. If the host writes 1 to this field, the card changes its timing to high speed interface timing (see Section 12.7.1 on page 122)

- BUS_WIDTH

It is set to ‘0’ (1 bit data bus) after power up and can be changed by a SWITCH command.

Table 68 — Bus mode values

Value	Bus Mode
255–3	Reserved
2	8 bit data bus
1	4 bit data bus
0	1 bit data bus

- ERASED_MEM_CONT

This field defines the content of an explicitly erased memory range.

Table 69 — Erased memory content values

Value	Erased Memory Content
255–2	Reserved
1	Erased memory range shall be ‘1’
0	Erased memory range shall be ‘0’

- BOOT_CONFIG

This register defines the configuration for boot operation.

Table 70 — Boot configuration bytes

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	BOOT_ACK	BOOT_PARTITION_ENABLE			BOOT_PARTITION_ACCESS		

Bit 7: Reserved

Bit 6: BOOT_ACK (non-volatile)

0x0 : No boot acknowledge sent (default)

0x1 : Boot acknowledge sent during boot operation

Bit[5:3] : BOOT_PARTITION_ENABLE (non-volatile)

User selects boot data that will be sent to master

0x0 : Device not boot enabled (default)

0x1 : Boot partition 1 enabled for boot

0x2 : Boot partition 2 enabled for boot

0x3–0x6 : Reserved

0x7 : User area enabled for boot

Bit[2:0] : BOOT_PARTITION_ACCESS

User selects boot partition for read and write operation

0x0 : No access to boot partition (default)

0x1 : R/W boot partition 1

0x2 : R/W boot partition 2

0x3–0x7 : Reserved

- BOOT_BUS_WIDTH

This register defines the bus width for boot operation.

Table 71 — Boot bus configuration

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved					RESET_BOOT_BUS_WIDTH	BOOT_BUS_WIDTH	

Bit[7:3] : Reserved

Bit 2: RESET_BOOT_BUS_WIDTH (non-volatile)

0x0 : Reset bus width to x1 after boot operation (default)

0x1 : Retain boot bus width after boot operation

Bit[1:0] : BOOT_BUS_WIDTH (non-volatile)

0x0 : x1 bus width in boot operation mode (default)

0x1 : x4 bus width in boot operation mode

0x2 : x8 bus width in boot operation mode

0x3 : Reserved

- ERASE_GROUP_DEF

This register allows master to select high capacity erase unit size, timeout value, and write protect group size. Bit defaults to “0” on power on.

Table 72 — ERASE_GROUP_DEF

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved							ENABLE

Bit[7:1]: Reserved

Bit0: ENABLE

0x0 : Use old erase group size and write protect group size definition (default)

0x1 : Use high-capacity erase unit size, high capacity erase timeout, and high-capacity write protect group size definition.

8.5 RCA register

The writable 16-bit relative card address (RCA) 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.

8.6 DSR register

The 16-bit driver stage register (DSR) is described in detail in Section 12.4 on page 114. 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.

9 SPI mode

SPI mode was removed in V4.3.

10 Error protection

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.

10.1 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:

Table 73 — Error correction codes

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%.

10.2 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, per data line, 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 $G(x) = x^7 + x^3 + 1$

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

$$CRC[6\ldots0] = \text{Remainder}[(M(x) \cdot x^7)/G(x)]$$

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).

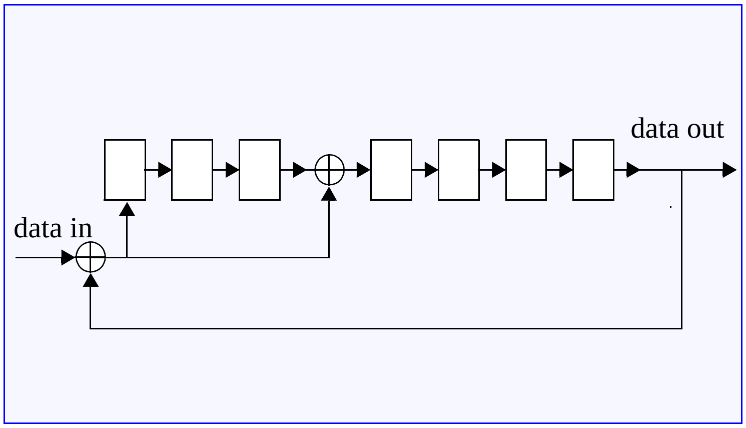


Figure 45 — CRC7 generator/checker

- 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).

The same CRC16 calculation is used for all bus configurations. In 4 bit and 8 bit bus configurations, the CRC16 is calculated for each line separately. Sending the CRC is synchronized so the CRC code is transferred at the same time in all lines.

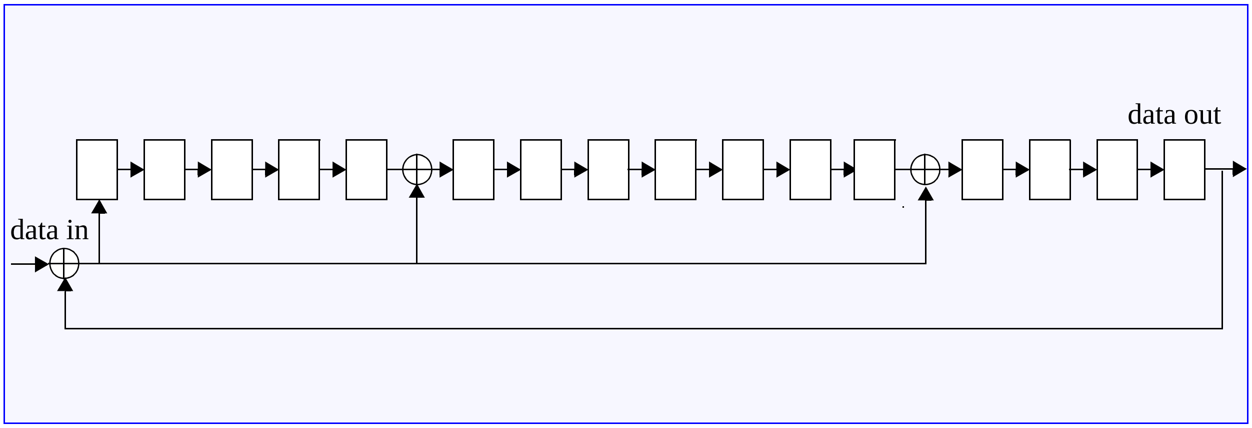


Figure 46 — CRC16 generator/checker

11 MultiMediaCard mechanical specification

See “MultiMediaCard (MMC) Card Mechanical Standard JESD84-C01,” and “MultiMediaCard (MMC) Card Bend, Torque, and Drop Test Specification JESD84-C02” for card applications; see “Embedded MultiMediaCard (eMMC) Mechanical Standard JESD84-C43” for eMMC applications.

12 The MultiMediaCard bus

The MultiMediaCard bus has ten 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.
- DAT0-7: Data lines are bidirectional signals. 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.

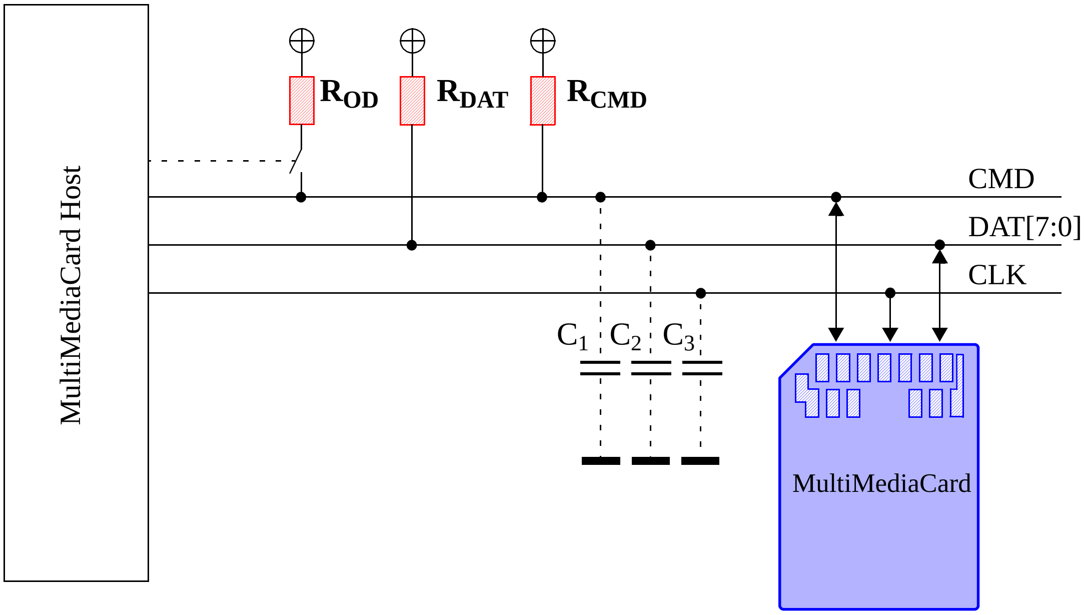


Figure 47 — Bus circuitry diagram

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 lines 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 Section 12.5 on page 117). 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 Section 12.5 on page 117.

12.1 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 Section 11 starting on page 107).

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.

12.2 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.

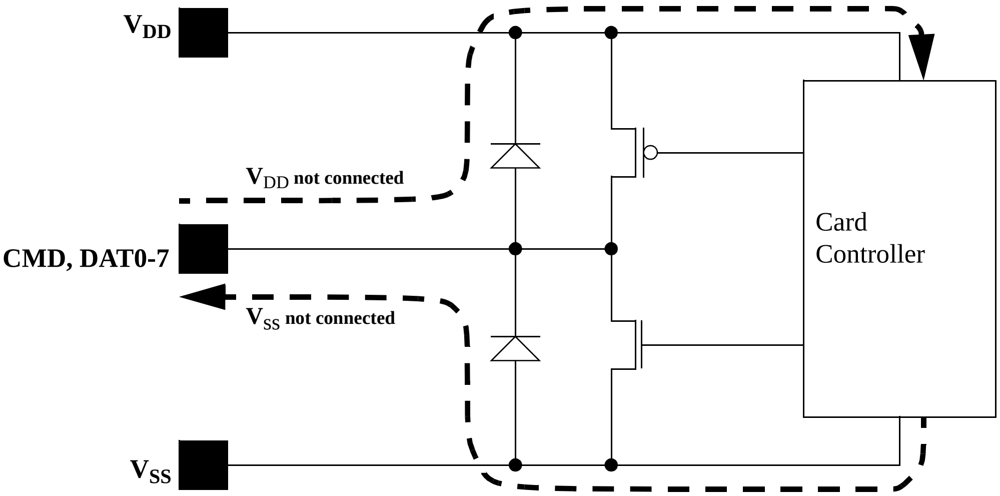


Figure 48 — Improper power supply

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

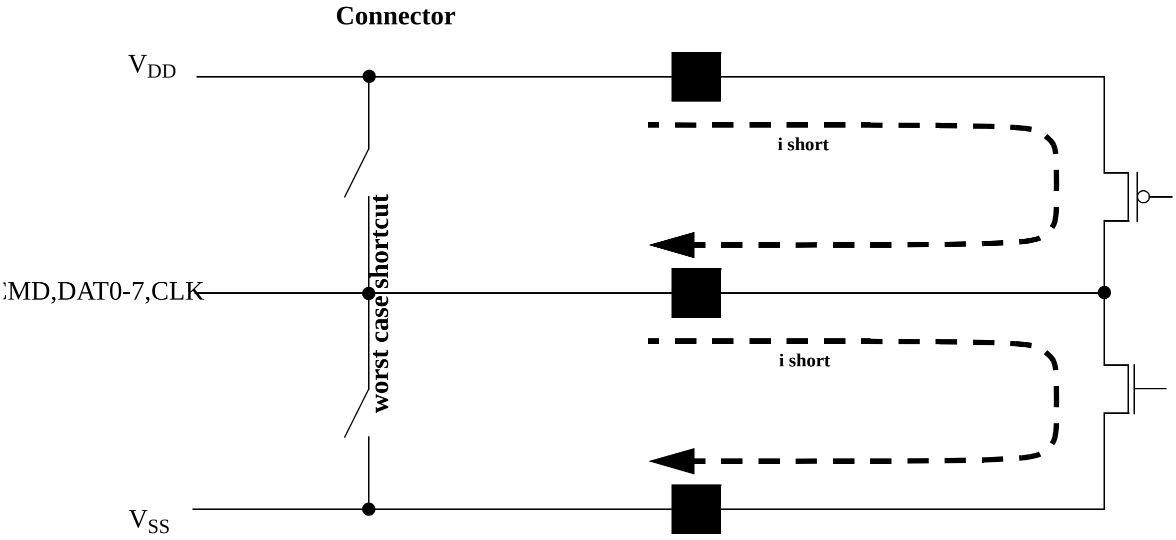


Figure 49 — Shortcut protection

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.

12.3 Power-up

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

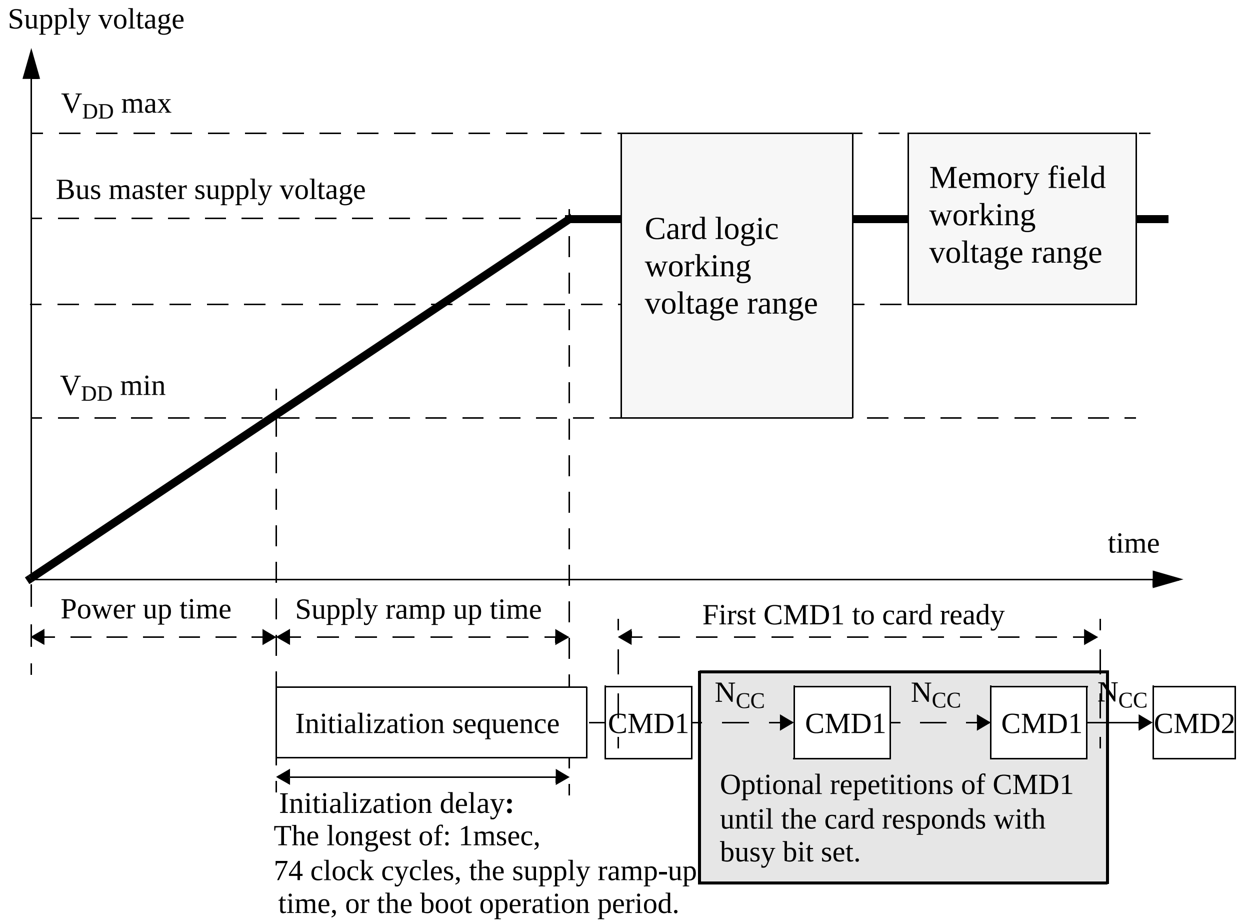

Supply voltage
V_{DD} max
Bus master supply voltage
V_{DD} min
Card logic working voltage range
Memory field working voltage range
time
Power up time
Supply ramp up time
First CMD1 to card ready
Initialization sequence
CMD1
N_{CC}
CMD1
N_{CC}
CMD1
N_{CC}
CMD2
Optional repetitions of CMD1 until the card responds with busy bit set.
Initialization delay:
The longest of: 1msec, 74 clock cycles, the supply ramp-up time, or the boot operation period.

Figure 50 — Power-up diagram

- After power up (including hot insertion, i.e., inserting a card when the bus is operating), the card enters the *pre-idle* state.
- If the card does not support boot mode, or its BOOT_PARTITION_ENABLE bit is cleared, the card moves immediately to the *idle* state. While in the idle state, the card ignores all bus transactions until CMD1 is received. If the card supports only specification v4.2 or earlier versions, it enters the idle state immediately following power-up.
- If the card BOOT_PARTITION_ENABLE bit is set, the card moves to the pre-boot state. The card then waits for boot initiation sequence. Following the boot operation period, the card enters the *idle* state. During the pre-boot state, if the card receives any CMD line transaction other than CMD1 or the boot initiation sequence (keeping the CMD line low for at least 74 clock cycles, or issuing CMD0 with the argument of 0xFFFFFFFA), the card moves to the *idle* state. If the card receives the boot initiation sequence (keeping the CMD line low for at least 74 clock cycles, or issuing CMD0 with the argument

of 0xFFFFFFFA), the card begins boot operation. If boot acknowledge is enabled, the card shall send acknowledge pattern “010” to the host within the specified time. After boot operation is terminated, the card enters the idle state and shall be ready for CMD1 operation. If the card receives CMD1 in the pre-boot state, it begins responding to the command and moves to card identification mode.

- While in the idle state, the card ignores all bus transactions until CMD1 is received.
- The maximum initial load (after power up or hot insertion) that the MultiMediaCard can present on the VDD line shall be a maximum of 10 uF in parallel with a minimum of 330 ohms. At no time during operation shall the card capacitance on the VDD line exceed 10 uF
- CMD1 is a special synchronization command used to negotiate the operation voltage range and to poll the card until it is out of its power-up sequence. Besides the operation voltage profile of the card, 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 the card is not ready. The host has to wait until this bit is cleared. The card shall complete its initialization within 1 second from the first CMD1 with a valid OCR range if boot operation is not executed.
- Getting the card 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 as 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. The sequence length is the longest of: 1msec, 74 clocks, the supply-ramp-up-time, or the boot operation period. 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.

12.3.1 eMMC power-up

An eMMC bus power-up is handled locally in each device and in the bus master. Figure 51 shows the power-up sequence and is followed by specific instructions regarding the power-up sequence.

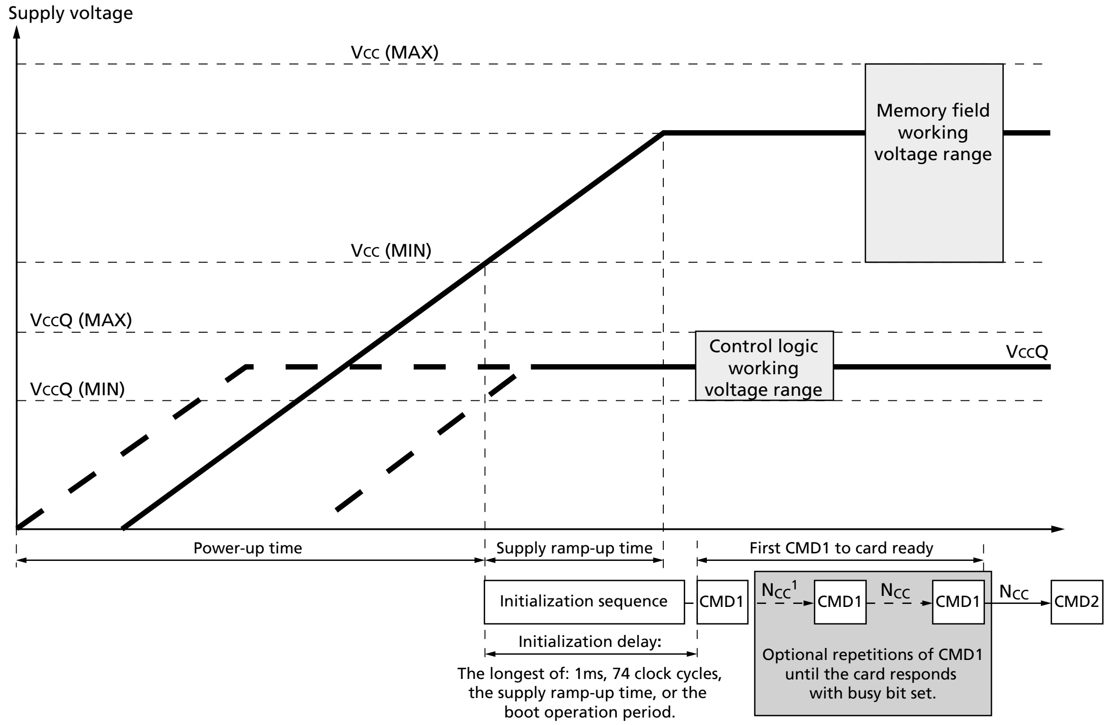

Supply voltage
Vcc (MAX)
Vcc (MIN)
VccQ (MAX)
VccQ (MIN)
Memory field working voltage range
Control logic working voltage range
VccQ
Power-up time
Supply ramp-up time
First CMD1 to card ready
Initialization sequence
CMD1
NCC1
CMD1
NCC
CMD1
NCC
CMD2
Initialization delay:
The longest of: 1ms, 74 clock cycles, the supply ramp-up time, or the boot operation period.
Optional repetitions of CMD1 until the card responds with busy bit set.

Figure 51 — eMMC power-up diagram

12.3.2 eMMC power-up guidelines

An eMMC power-up must adhere to the following guidelines:

- When power-up is initiated, either V_{CC} or V_{CC}Q can be ramped up first, or both can be ramped up simultaneously.
- After power up, the eMMC enters the pre-idle state.
- If the eMMC does not support boot mode or its BOOT_PARTITION_ENABLE bit is cleared, the eMMC moves immediately to the idle state. While in the idle state, the eMMC ignores all bus transactions until CMD1 is received. If the eMMC supports only specification v4.2 or earlier versions, the device enters the idle state immediately following power-up.
- If the BOOT_PARTITION_ENABLE bit is set, the eMMC moves to the pre-boot state, and the eMMC waits for the boot-initiation sequence. Following the boot operation period, the eMMC enters the idle state. During the pre-boot state, if the eMMC receives any CMD-line transaction other than the boot initiation sequence (keeping CMD line low for at least 74 clock cycles, or issuing CMD0 with the argument of 0xFFFFFFFA) and CMD1, the eMMC moves to the Idle state. If eMMC receives the boot initiation sequence (keeping the CMD line low for at least 74 clock cycles, or issuing CMD0 with the argument of 0xFFFFFFFA), the eMMC begins boot operation. If boot acknowledge is enabled, the eMMC shall send acknowledge pattern “010” to the host within the specified time. After boot operation is terminated, the eMMC enters the idle state and shall be ready for CMD1 operation. If the eMMC receives CMD1 in the pre-boot state, it begins responding to the command and moves to the card identification mode.
- While in the idle state, the eMMC 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 device until it is out of its power-up sequence. In addition to the operation voltage profile of the device, the response to CMD1 contains a busy flag indicating that the device is still working on its power-up procedure and is not ready for identification. This bit informs the host that the device is not ready, and the host must wait until this bit is cleared. The device must complete its initialization within 1 second of the first CMD1 issued with a valid OCR range.

- The bus master moves the device out of the idle state. Because the power-up time and the supply ramp-up time depend on application parameters such as the bus length and the power supply unit, the host must ensure that power is built up to the operating level (the same level that 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. The sequence length is the longest of: 1ms, 74 clocks, the supply ramp-up time, or the boot operation period. An additional 10 clocks (beyond the 64 clocks of the power-up sequence) are provided to eliminate power-up synchronization problems.
- Every bus master must implement CMD1.

12.3.3 eMMC power cycling

The master can execute any sequence of V_{CC} and $V_{CC}Q$ power-up/power-down. However, the master must not issue any commands until V_{CC} and $V_{CC}Q$ are stable within each operating voltage range. After the slave enters sleep mode, the master can power-down V_{CC} to reduce power consumption. It is necessary for the slave to be ramped up to V_{CC} before the host issues CMD5 (SLEEP_AWAKE) to wake the slave unit.

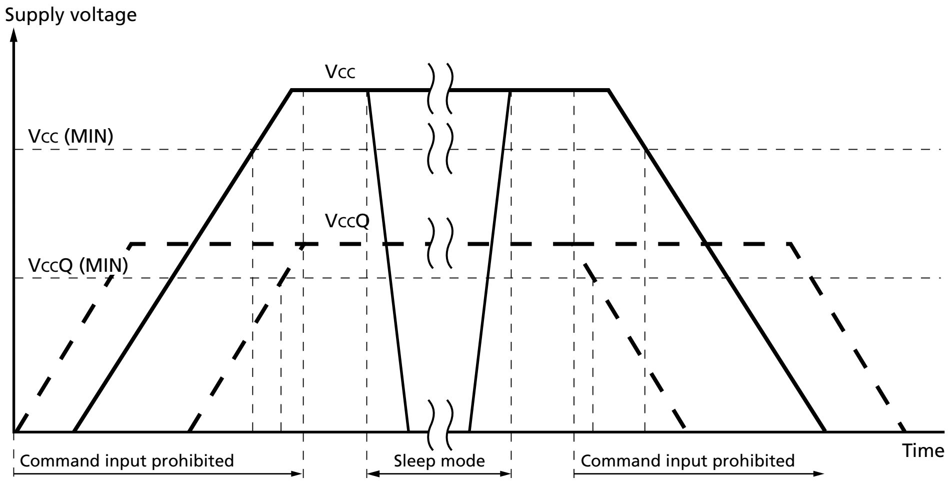


Figure 52 — The eMMC power cycle

12.4 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.

The CMD and DAT bus drivers consist of a predriver stage and a complementary driver transistor (Figure 53).

The DSR register is used to configure the predriver stage output rise and fall time, and the complementary driver transistor size. The proper combination of both allows optimum bus performance.

Table 74 defines the DSR register contents:

Table 74 — DSR register content

	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
$t_{\text{switch-on max}}$	reserved							
$t_{\text{switch-on min}}$								

	15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
$i_{\text{peak min}}$	reserved							
$i_{\text{peak max}}$								

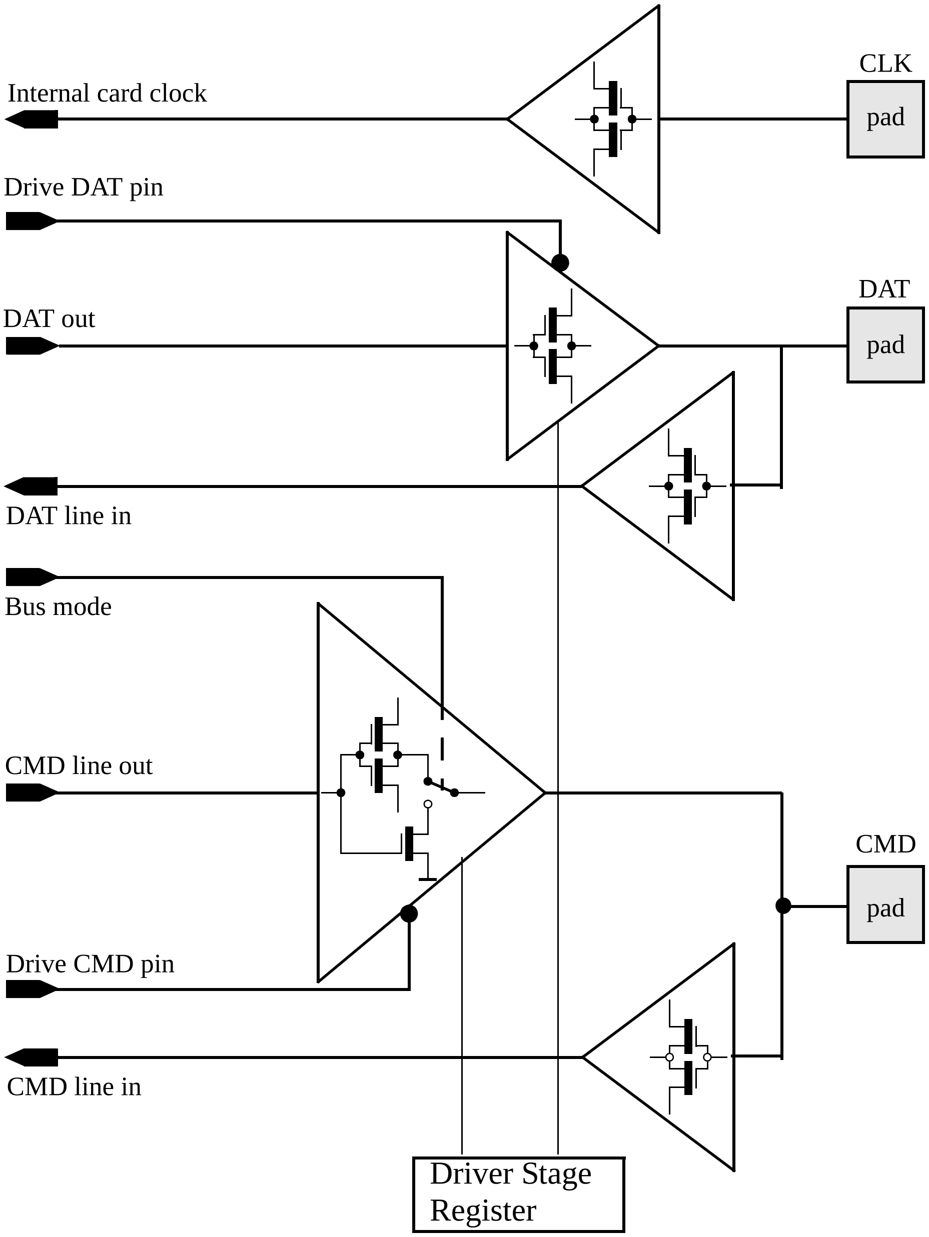


Figure 53 — MultiMediaCard bus driver

All data is valid for the specified operating range (voltage, temperature). The DSR register has two byte codes (e.g. bits 0-7 = 0x02, bits 8-15 = 0x01) that define specific min and max values for the switching speed and current drive of the register, respectively (actual values are TBD). Any combination of switching speed and driving force may be programmed. The selected speed settings must be in accordance with the system frequency. The following relationship must be kept:

$$t_{\text{switch-on-max}} \pm 0.4 * (\mathrm{F_{OD}})^{-1}$$

12.5 Bus operating conditions

Table 75 — General operating conditions

Parameter		**Symbol**	**Min**	**Max.**	**Unit**	**Remark**
Peak voltage on all lines	Card		-0.5	V_{DD} + 0.5	V	
	BGA		-0.5	$V_{CC}Q$ + 0.5	V	
All Inputs						
Input Leakage Current (before initialization sequence and/or the internal pull up resistors connected)			-100	100	µA	
Input Leakage Current (after initialization sequence and the internal pull up resistors disconnected)			-10	10	µA	
All Outputs						
Output Leakage Current (before initialization sequence)			-100	100	µA	
Output Leakage Current (after initialization sequence)			-10	10	µA	
NOTE 1. Initialization sequence is defined in Section 12.3 on page 111						

12.5.1 Power supply: high-voltage MultiMediaCard

Table 76 — Power supply voltage: high-voltage MultiMediaCard

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

12.5.2 Power supply: dual-voltage MultiMediaCard

Table 77 — Power supply voltage: dual-voltage MultiMediaCard

Parameter	**Symbol**	**Min**	**Max.**	**Unit**	**Remarks**
Supply voltage (low voltage range)	V_{DDL}	1.70	1.95	V	1.95V–2.7V range 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 the card for the different card configurations is defined in the power class fields in the EXT_CSD register.

The current consumption of any card during the power-up procedure (except in boot operation), while the host has not sent yet a valid OCR range, must not exceed 10mA.

12.5.3 Power supply: eMMC

In the eMMC, VCC is used for the NAND flash device and its interface voltage; VCCQ is for the controller and the MMC interface voltage shown in Figure 54. The core regulator is optional and only required when VCCQ is in the 3V range. A Creg capacitor must be connected to the VDDi terminal to stabilize regulator output on the system.

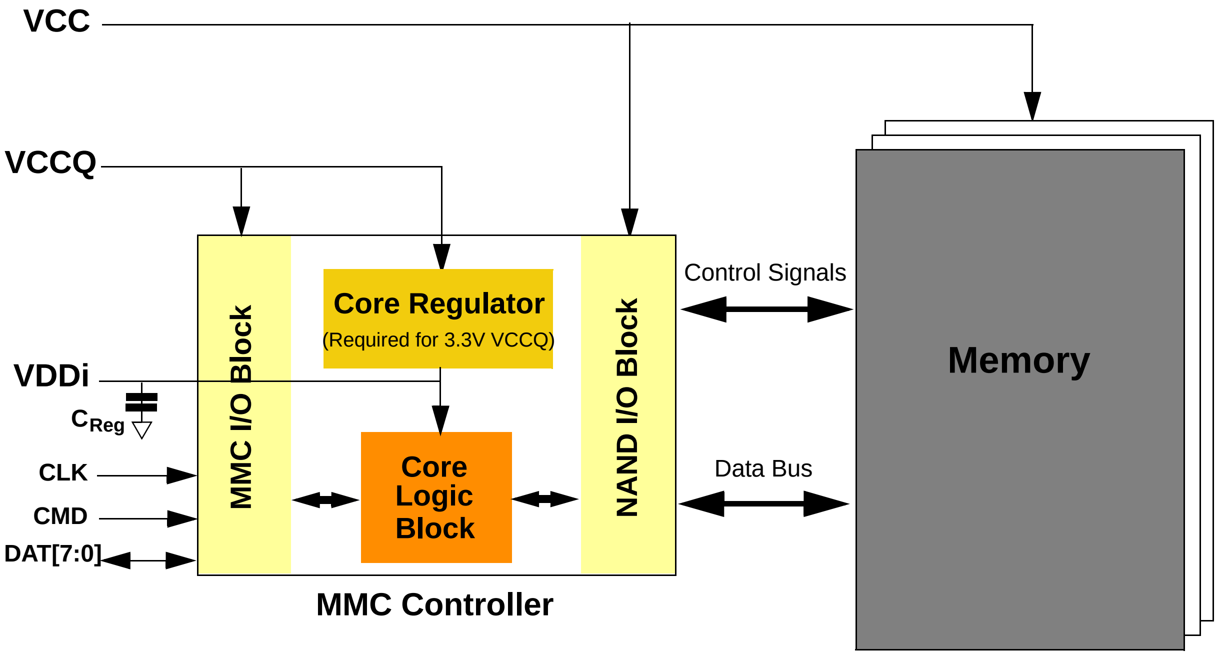


Figure 54 — eMMC internal power diagram

12.5.4 Power supply: eMMC

The eMMC supports one or more combinations of V_{CC} and $V_{CC}Q$ as shown in Table 78. The $V_{CC}Q$ must be defined at equal to or less than V_{CC}. The available voltage configuration is shown in Table 79.

Table 78 — eMMC power supply voltage

Parameter	Symbol	Min	Max	Unit	Remarks
Supply voltage (NAND)	V_{CC}	2.7	3.6	V	
		1.7	1.95	V	
Supply voltage (I/O)	$V_{CC}Q$	2.7	3.6	V	
		1.7	1.95	V	

The eMMC must support at least one of the valid voltage configurations, and can optionally support all valid voltage configurations (see Table 79).

Table 79 — eMMC voltage combinations

		$V_{CC}Q$		
		1.2V[1]	**1.7V–1.95V**	**2.7V–3.6V**
V_{CC}	**2.7V–3.6V**	TBD	Valid	Valid
	1.7V–1.95V	TBD	Valid	NOT VALID

1) The 1.2V voltage range is currently TBD.

12.5.5 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 the card connected to this line,

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

and requiring the sum of the host and bus capacitances not to exceed 20 pF (see Table 80).

Table 80 — Capacitance

Parameter	Symbol	Min	Typ	Max	Unit	Remark
Pull-up resistance for CMD	R_{CMD}	4.7		100	Kohm	to prevent bus floating
Pull-up resistance for DAT0–7	R_{DAT}	50		100	Kohm	to prevent bus floating
Internal pull up resistance DAT1–DAT7	R_{int}	50		150	kohm	to prevent unconnected lines floating
Bus signal line capacitance	C_L			30	pF	Single card
Single card capacitance	C_{MICRO}			12	pF	For MMC*micro*
	C_{MOBILE}			18		For MMC*mobile* and MMC*plus*
	C_{BGA}		7	12		For BGA
Maximum signal line inductance				16	nH	$f_{PP} \leq 52$ MHz

12.6 Bus signal levels

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

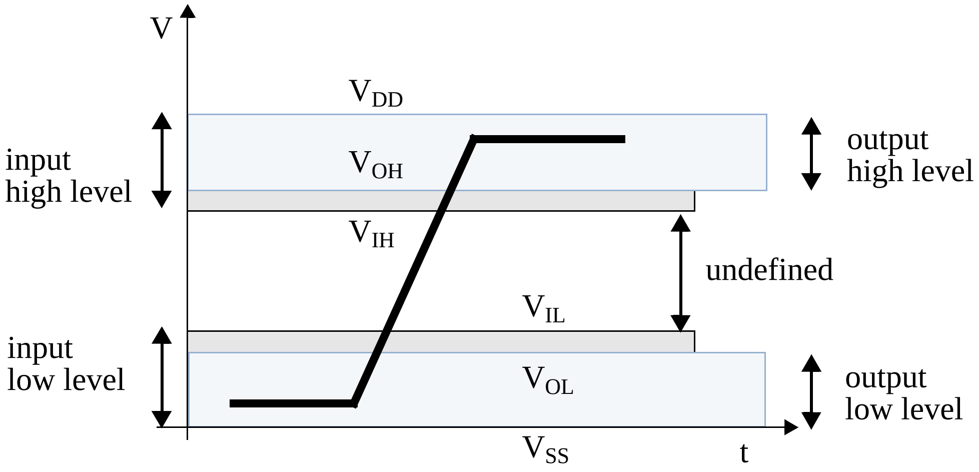


Figure 55 — Bus signal levels

12.6.1 Open-drain mode bus signal level

Table 81 — Open-drain 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.

12.6.2 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:

Table 82 — Push-pull signal level—high-voltage MultiMediaCard

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}	V_{SS} - 0.3	0.25 * V_{DD}	V	

12.6.3 Push-pull mode bus signal level—dual-voltage MultiMediaCard

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

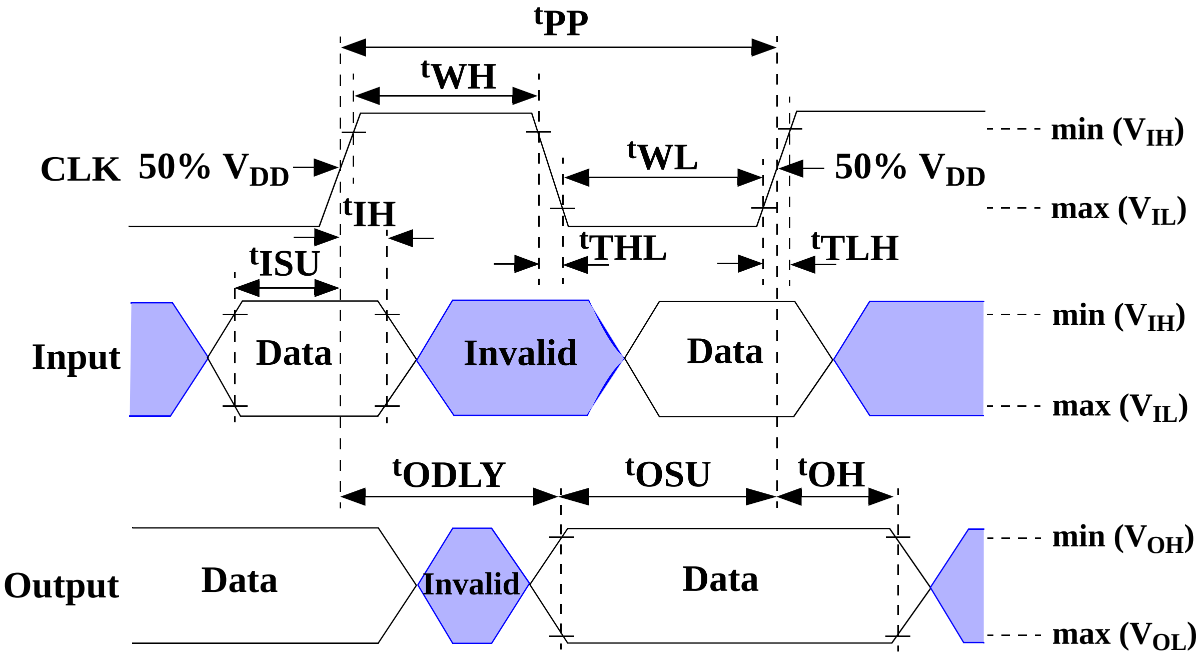
- 2.7V - 3.6V: Identical to the High Voltage MultiMediaCard (refer to Section 12.6.2 on page 120 above).
- 1.95 - 2.7V: Undefined. The card is not operating at this voltage range.
- 1.70 - 1.95V: Compatible with EIA/JEDEC Standard “EIA/JESD8-7 Wide Range” as defined in the

following table.

Table 83 — Push-pull signal level—dual-voltage MultiMediaCard

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	

12.7 Bus timing



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

Figure 56 — Timing diagram: data input/output

12.7.1 Card interface timings

Table 84 — High-speed card interface timing

Parameter	Symbol	Min	Max.	Unit	Remark
Clock CLK[1]					
Clock frequency Data Transfer Mode (PP)[2]	f_{PP}	0	52[3]	MHz	$C_L \leq 30$ pF Tolerance: +100KHz
Clock frequency Identification Mode (OD)	f_{OD}	0	400	kHz	Tolerance: +20KHz
Clock low time	t_{WL}	6.5		ns	$C_L \leq 30$ pF
Clock rise time[4]	t_{TLH}		3	ns	$C_L \leq 30$ pF
Clock fall time	t_{THL}		3	ns	$C_L \leq 30$ pF
Inputs CMD, DAT (referenced to CLK)					
Input set-up time	t_{ISU}	3		ns	$C_L \leq 30$ pF
Input hold time	t_{IH}	3		ns	$C_L \leq 30$ pF
Outputs CMD, DAT (referenced to CLK)					
Output delay time during data transfer	t_{ODLY}		13.7	ns	$C_L \leq 30$ pF
Output hold time	t_{OH}	2.5		ns	$C_L \leq 30$ pF
Signal rise time[5]	t_{RISE}		3	ns	$C_L \leq 30$ pF
Signal fall time	t_{FALL}		3	ns	$C_L \leq 30$ pF

NOTE 1. CLK timing is measured at 50% of V_{DD}.

NOTE 2. A MultiMediaCard shall support the full frequency range from 0-26Mhz, or 0-52MHz

NOTE 3. Card can operate as high-speed card interface timing at 26 MHz clock frequency.

NOTE 4. CLK rise and fall times are measured by min (V_{IH}) and max (V_{IL}).

NOTE 5. Inputs CMD, DAT rise and fall times are measured by min (V_{IH}) and max (V_{IL}), and outputs CMD, DAT rise and fall times are measured by min (V_{OH}) and max (V_{OL}).

Table 85 — Backward-compatible card interface timing

Parameter	Symbol	Min	Max.	Unit	Remark[1]
Clock CLK[2]					
Clock frequency Data Transfer Mode (PP)[3]	f_{PP}	0	26	MHz	$C_L \leq 30$ pF
Clock frequency Identification Mode (OD)	f_{OD}	0	400	kHz	
Clock low time	t_{WL}	10		ns	$C_L \leq 30$ pF
Clock rise time[4]	t_{TLH}		10	ns	$C_L \leq 30$ pF
Clock fall time	t_{THL}		10	ns	$C_L \leq 30$ pF
Inputs CMD, DAT (referenced to CLK)					
Input set-up time	t_{ISU}	3		ns	$C_L \leq 30$ pF
Input hold time	t_{IH}	3		ns	$C_L \leq 30$ pF

Table 85 — Backward-compatible card interface timing (continued)

Parameter	**Symbol**	**Min**	**Max.**	**Unit**	**Remark[1]**
Outputs CMD, DAT (referenced to CLK)					
Output set-up time	t_{OSU}	11.7		ns	$C_L \leq 30$ pF
Output hold time	t_{OH}	8.3		ns	$C_L \leq 30$ pF

NOTE 1. The card must always start with the backward-compatible interface timing. The timing mode can be switched to high-speed interface timing by the host sending the SWITCH command (CMD6) with the argument for high-speed interface select.

NOTE 2. CLK timing is measured at 50% of VDD.

NOTE 3. For compatibility with cards that support the v4.2 standard or earlier, host should not use > 20 MHz before switching to high-speed interface timing.

NOTE 4. CLK rise and fall times are measured by min (V_{IH}) and max (V_{IL}).

13 MultiMediaCard standard compliance

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

- Generic card access and communication protocol (Section 7 starting on page 21, Section 8 starting on page 77)
- The description of the SPI mode was removed from the v.4.3 standard.
- Data integrity and error handling (Section 10 starting on page 103)
- Mechanical interface parameters, such as: connector type and dimensions and the card form factor (Section 11 starting on page 107)
- Electrical interface parameters, such as: power supply, peak and average current consumption and data transfer frequency (Section 12 starting on page 109)
- 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 Section 6.3 on page 10: 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 Section 8.3 on page 79). 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 Section 7.9 on page 49). 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 car
- d class without having implemented this function.
- *Not required*: this function has no use for the specified card class.

Table 86 — MultiMediaCard host requirements for card classes

Function	ROM card class	R/W card class	I/O card class
26–52 MHz transfer rate	Optional	Optional	Optional
20–26 MHz transfer rate	Mandatory	Mandatory	Mandatory
0–20 MHz transfer rate	Mandatory	Mandatory	Mandatory
2.7–3.6V power supply	Mandatory	Mandatory	Mandatory
1.70–1.95V power supply	Optional	Optional	Optional

Table 86 — MultiMediaCard host requirements for card classes (continued)

Function	**ROM card class**	**R/W card class**	**I/O card class**
CCC 0 basic	Mandatory	Mandatory	Mandatory
CCC 1 sequential read	Optional	Optional	Optional
CCC 2 block read	Mandatory	Mandatory	Optional
CCC 3 sequential write	Not required	Optional	Optional
CCC 4 block write	Not required	Mandatory	Optional
CCC 5 erase	Not required	Mandatory	Not required
CCC 6 write protection functions	Not required	Mandatory	Not required
CCC 7 lock card commands	Mandatory	Mandatory	Mandatory
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	Obsolete	Obsolete	Obsolete

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.
- The Card Status bits relevance, according to the supported classes, is defined in Table 29 on page 64.

14 File formats for the MultiMediaCard

The file format specification, for the MultiMediaCard, starting with V4.1 of this document, has been moved into a separate document called the “File Formats Specifications For MultiMediaCards”.

Annex 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 1.

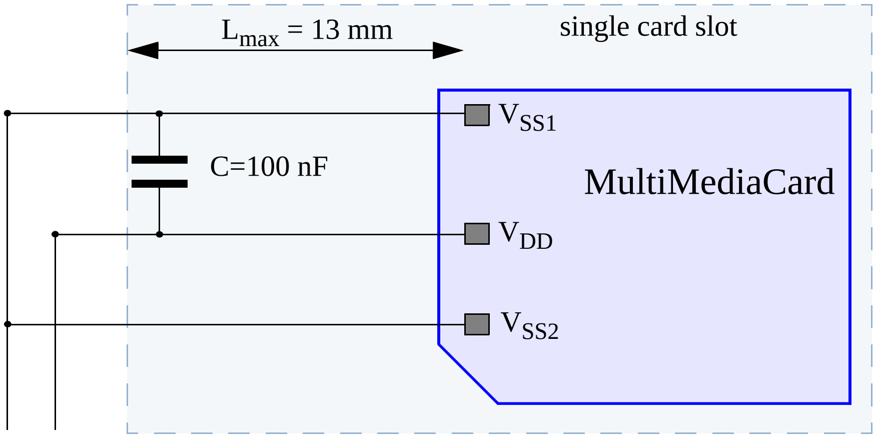


Figure A.1 — Power supply decoupling

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 2. 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.

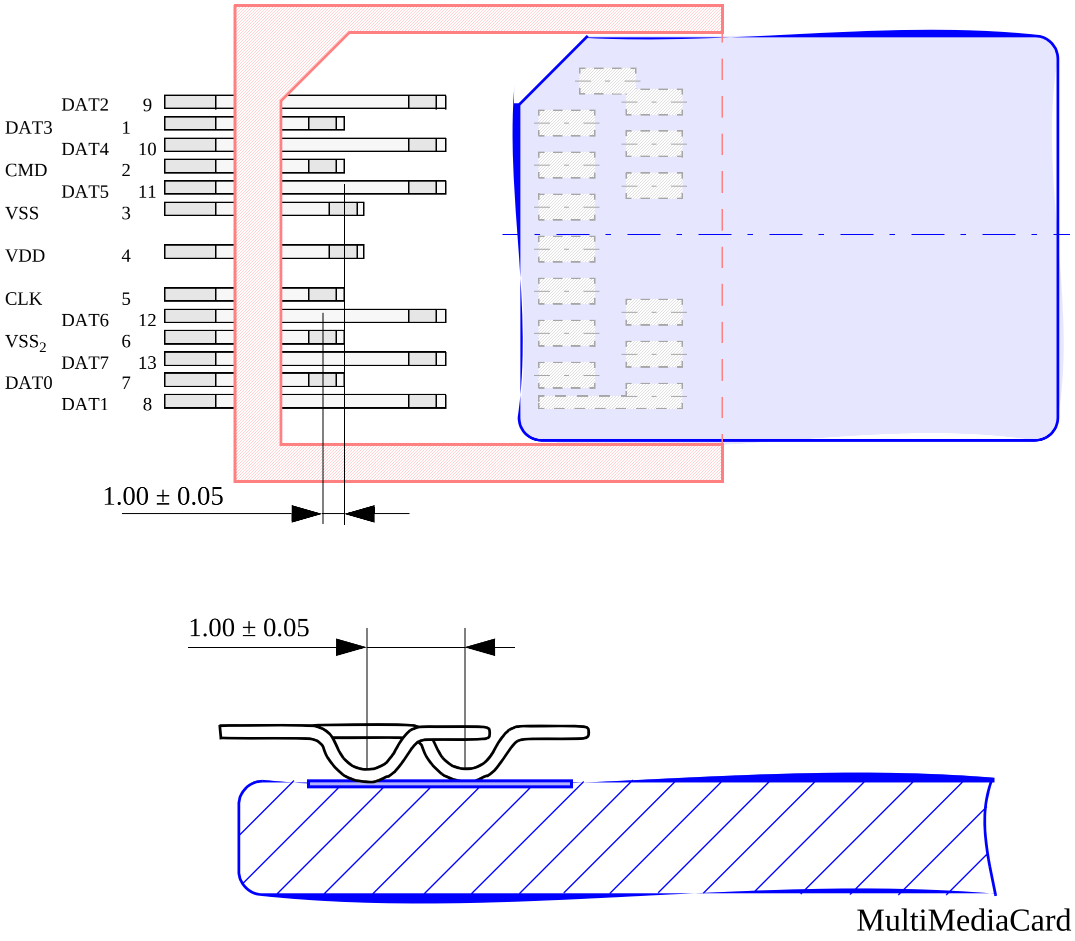


Figure A.2 — Modified MultiMediaCard connector for hot insertion

A.3.3 Characteristics

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

Table A.1 — Mechanical characteristics

Characteristic	**Value**	
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…2000Hz	DIN IEC 512 part 2 and 4
Acceleration	2g	
Shock		
Acceleration	5g	

Table A.2 — Electrical characteristics

DIN IEC 512	**Value**
Contact resistance	100 mOhm
Current-carrying capacity at +25°C	0.5 A
Insulation resistance	> 1000 MOhm, > MOhm after test
Operating voltage	3.3V
Testing voltage	500V
Operating current	100 mA max

Table A.3 — Climatic characteristics

DIN IEC 512 part 6–9	**Range**
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 applications 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. 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.

Table A.4 — Macro commands

Mnemonic	Description
CIM_SINGLE_CARD_ACQ	Starts an identification cycle of a single card.
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.

The MultiMediaCard command sequences are described in the following paragraphs. Figure 3 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.

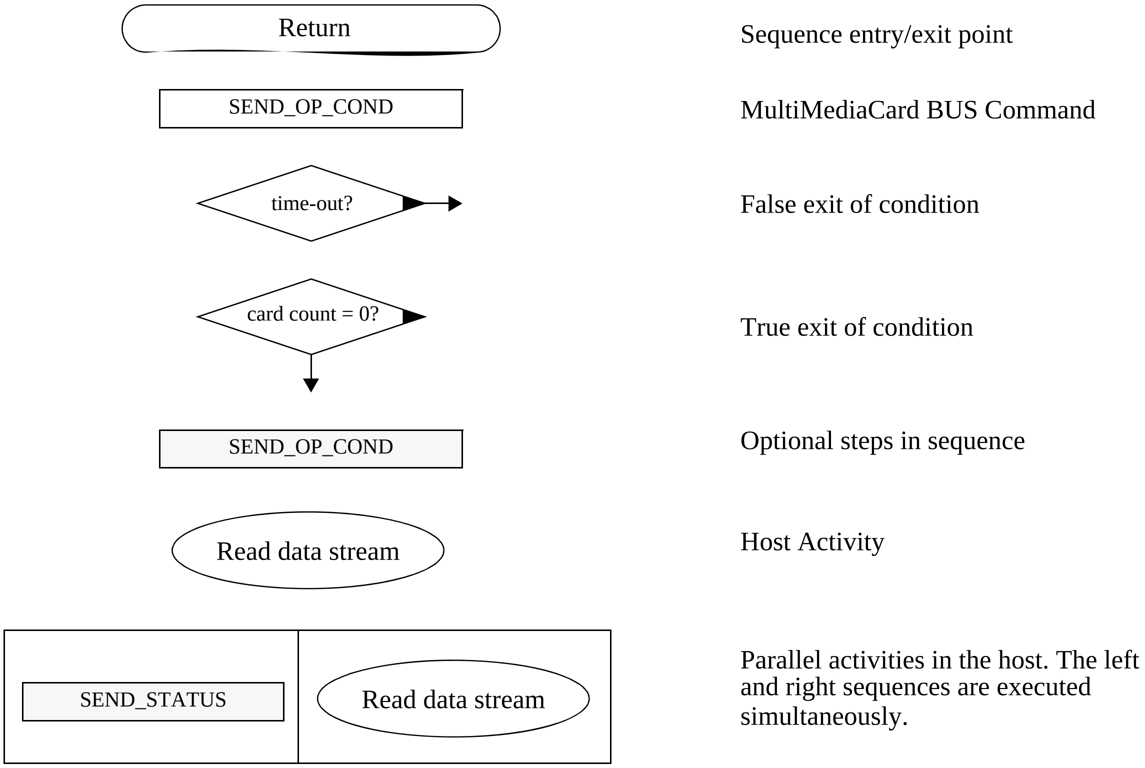

Return
Sequence entry/exit point
SEND_OP_COND
MultiMediaCard BUS Command
time-out?
False exit of condition
card count = 0?
True exit of condition
SEND_OP_COND
Optional steps in sequence
Read data stream
Host Activity
SEND_STATUS
Read data stream
Parallel activities in the host. The left and right sequences are executed simultaneously.

Figure A.3 — Legend for command-sequence flow charts

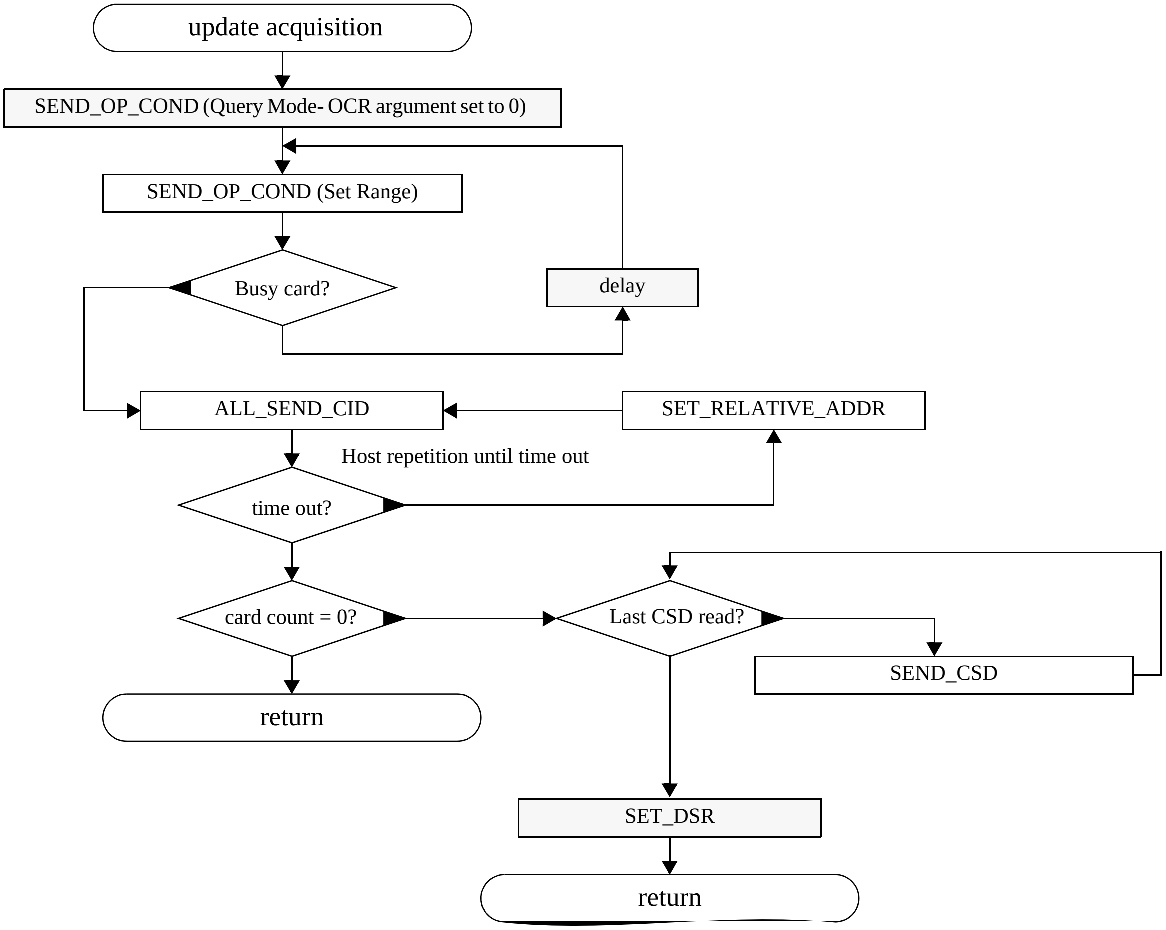


Figure A.4 — SEND_OP_COND command flow chart

- CIM_SINGLE_CARD_ACQ

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.

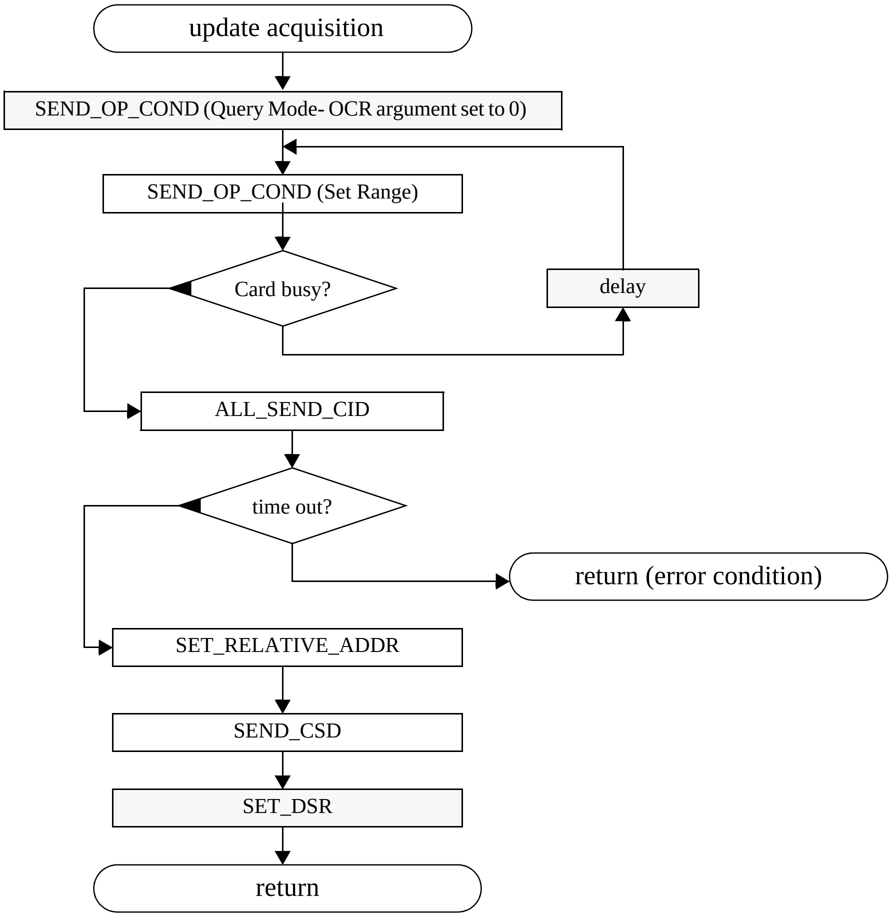


Figure A.5 — CIM_SINGLE_CARD_ACQ

- CIM_SETUP_CARD

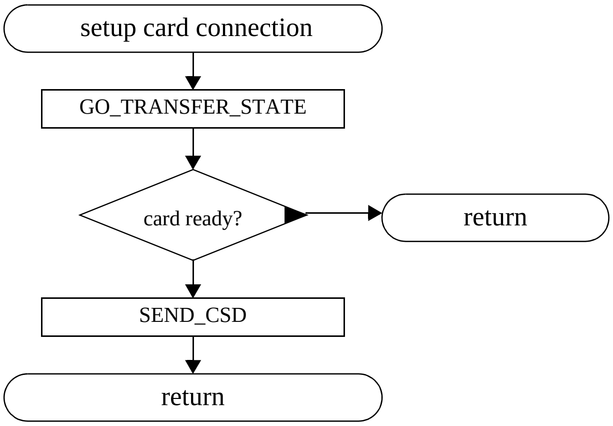


Figure A.6 — CIM_SETUP_CARD

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. 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

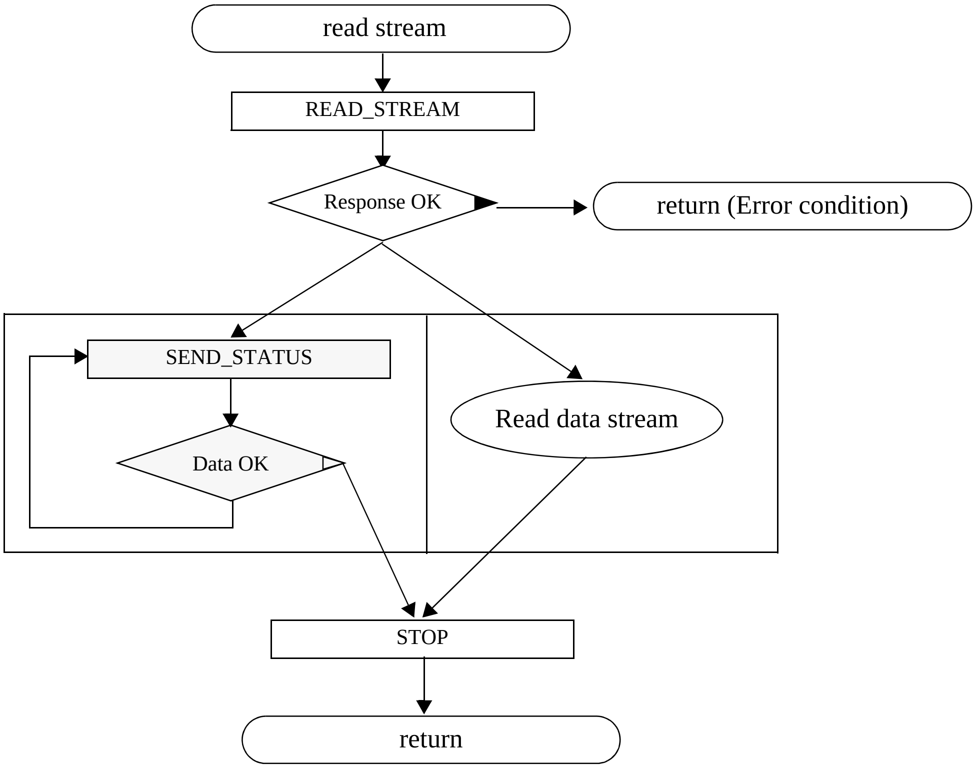


Figure A.7 — CIM_STREAM_READ

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

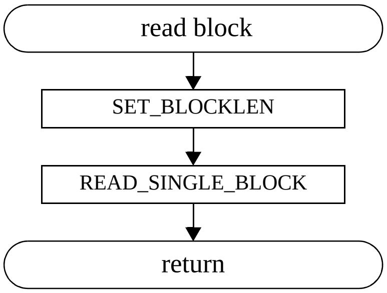


Figure A.8 — CIM_READ_BLOCK

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

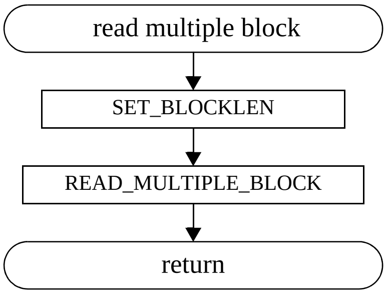


Figure A.9 — CIM_READ_MBLOCK

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.

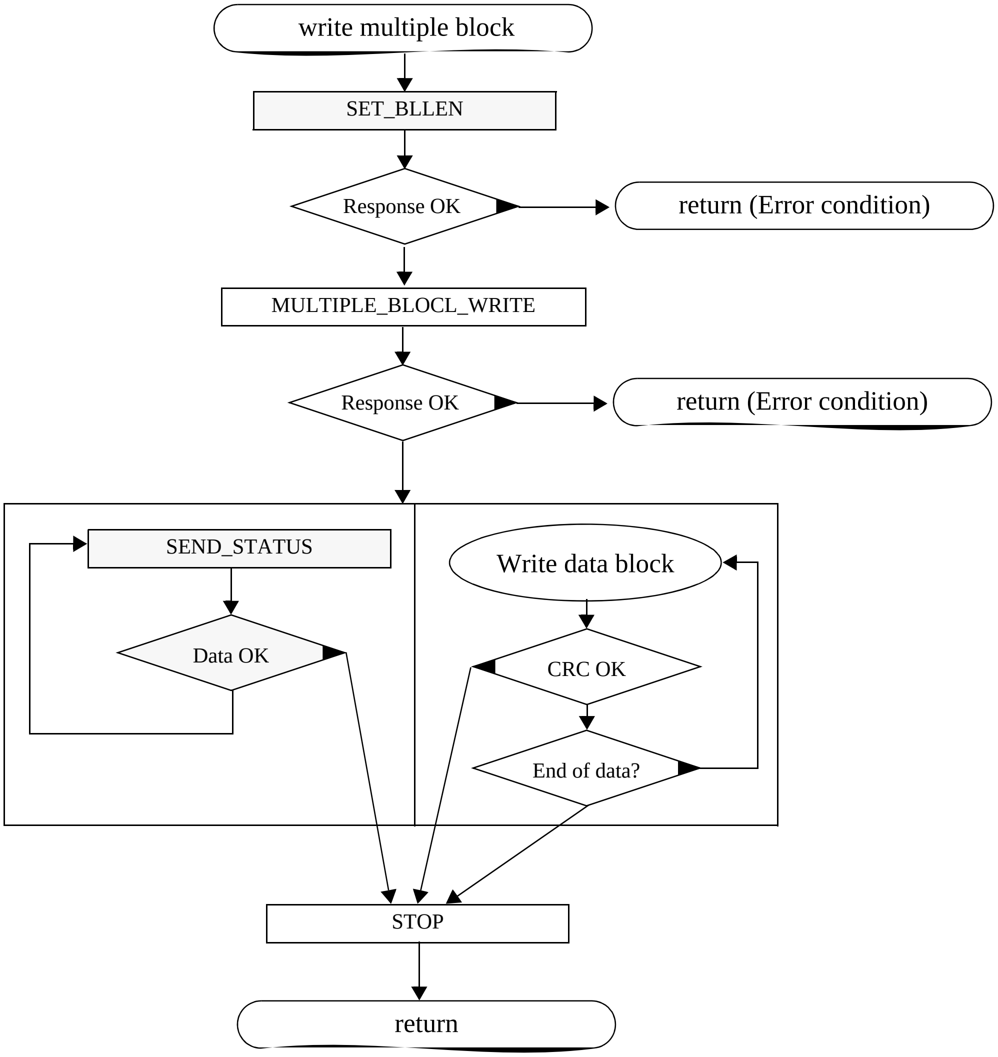


Figure A.10 — CIM_WRITE_MBLOCK

- 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.).

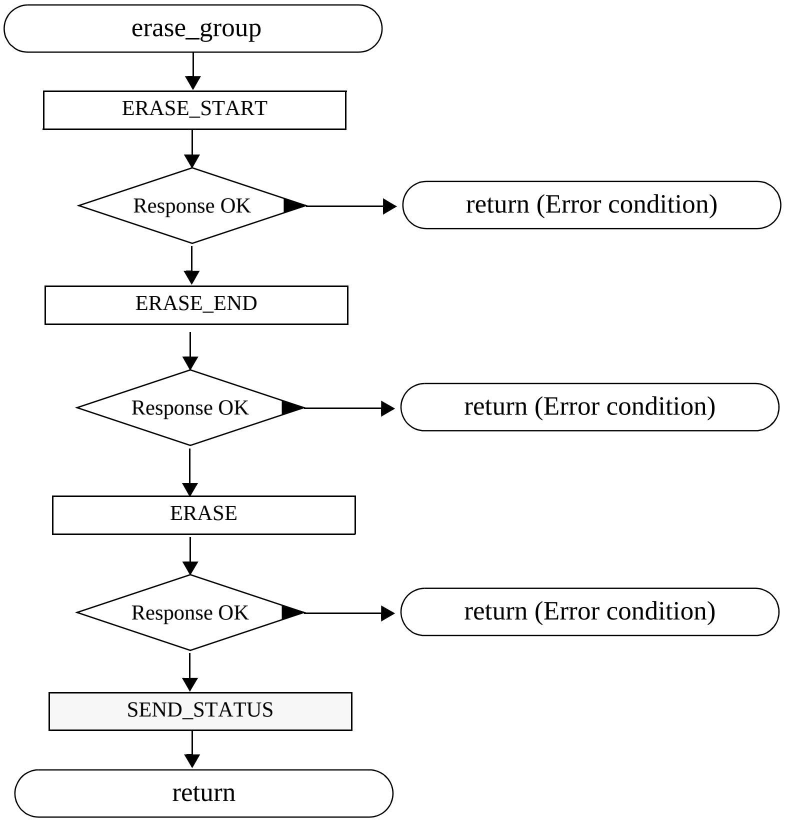


Figure A.11 — CIM_ERASE_GROUP

A.6 Host interface timing

With the introduction of MultiMediaCard specification version 4.0, higher clock speeds are used in both hosts and cards. In order to maintain backward and forward compatibilities, the card, and the host, are required to implement two different sets of timings. One set of timings is the interface timing aimed at high speed systems, working at clock frequencies higher than 20MHz, up to 52MHz. The other set of timing is different for the card and for the host. The card has to maintain backwards compatibility, allowing it to be inserted into an older MultiMediaCard system. The host has to maintain forward compatibility, allowing old MultiMediaCard to be inserted into new high speed MultiMediaCard systems.

Follows the table for the forward compatibility interface timing. The high speed interface timing is already defined in Table 84 on 122.

Table A.5 — Forward-compatible host interface timing

Parameter	Symbol	Min	Max.	Unit	Remark
Clock CLK[1]					
Clock frequency Data Transfer Mode (PP)	f_{PP}	0	20	MHz	C_L <= 30 pF
Clock frequency Identification Mode (OD)	f_{OD}	0	400	kHz	
Clock low time	t_{WL}	10		ns	C_L <= 30 pF
Clock rise time[2]	t_{TLH}		10	ns	C_L <= 30 pF
Clock fall time	t_{THL}		10	ns	C_L <= 30 pF
Inputs CMD, DAT (referenced to CLK)					
Input set-up time	t_{ISU}	4.8		ns	C_L <= 30 pF
Input hold time	t_{IH}	4.4		ns	C_L <= 30 pF
Outputs CMD, DAT (referenced to CLK)					
Output set-up time	t_{OSU}	5		ns	C_L <= 30 pF
Output hold time	t_{OH}	5		ns	C_L <= 30 pF

NOTE 1. All timing values are measured relative to 50% of voltage level
NOTE 2. Rise and fall times are measured from 10%-90% of voltage level.

A.7 Handling of passwords

There is only one length indicator for the password instead of having separate length bytes reserved for both new and old passwords. Due to this there is a possibility for conflict during the password change operation after which the new password does not match to the one which the user set. There has also proven to be various interpretations related to the removal of the lock function in card implementations.

Thus the procedures in the following sections are recommended to be used to enable best possible compatibility over host-card systems.

A.7.1 Changing the password

This applies for the host systems. Instead of using the password replacement function implement the password change as follows:

- First, remove the old password

- Second, set the new password

A.7.2 Removal of the password

This applies to the host systems. Before resetting the password (CLR_PWD) unlock the card.

A.8 High-speed MultiMediaCard bus functions

A.8.1 Bus initialization

There is more than one way to use the new features, introduced in v4.0 of this document. This application note describes a way to switch a high speed MultiMediaCard from the initial lower frequency to the high frequency and different bus configuration.

High Speed MultiMediaCards are backwards compatible, therefore after power up, they behave identically to old cards, with no visible difference[1].

The steps a host can do to identify a High Speed MultiMediaCard, and to put it to high speed mode are described next, from power-up until the card is ready to work at high data rates.

a. Power-up

1- Apply power to the bus, communication voltage range (2.7-3.6V)
2- Set clock to 400KHz, or less
3- Wait for 1ms, then wait for 74 more clock cycles
4- Send CMD0 to reset the bus, keep CS line high during this step.
5- Send CMD1, with the intended voltage range in the argument (either 0x00FF8000 or 0x00000080)
6- Receive R3
7- If the OCR busy bit is ‘0’, repeat steps 5 and 6
8- From the R3 response argument the host can learn if the card is a High Voltage or Dual Voltage card. If the argument is 0x80FF8000 the card is only High Voltage, if the argument is 0x80FF8080 the card is Dual Voltage.
9- If R3 returned some other value, the card is not compliant (since it should have put itself into *inactive* state, due to voltage incompatibility, and not respond); in such a case the host must power down the bus and start its error recovery procedure (the definition of error recovery procedures is host dependent and out of the scope of this application note)

Low-voltage power-up

Do the following steps if low voltage operations are supported by the host; otherwise skip to step 16.

10- If the host is a low voltage host, and recognized a dual voltage card, power down the MMC bus
11- Apply power to the MMC bus, in the low voltage range (1.70 -1.95V)
12- Wait for 1ms, then for 74 more clock cycles
13- Send CMD1 with argument 0x00000080
14- Receive R3, it should read 0x00FF8080
15- If the OCR busy bit is ‘0’, repeat steps 13 and 14

b. CID retrieval and RCA assignment

16- Send CMD2
17- Receive R2, and get the card’s CID
18- Send CMD3 with a chosen RCA, with value greater than 1

[1] Some legacy cards correctly set the ILLEGAL_CMD bit, when the bus testing procedure is executed upon them, and some other legacy cards in the market do not show any error.

c. CSD retrieval and host adjustment

19-Send CMD9
20-Receive R2, and get the card’s CSD from it.
21-If necessary, adjust the host parameters according to the information in the CSD
If the SPEC_VERS indicates a version 4.0 or higher, the card is a high speed card and supports SWITCH and SEND_EXT_CSD commands.
Otherwise the card is an old MMC card.
Regardless of the type of card, the maximum clock frequency that can be set at this point is defined in the TRAN_SPEED field.

A.8.2 Switching to high-speed mode

The following steps are supported by cards implementing version 4.0 or higher. Do these steps after the bus is initialized according to section Annex A.8.1 on page 141.

22-Send CMD7 with the card’s RCA to place the card in *tran* state
23-Send CMD8, SEND_EXT_CSD. From the EXT_CSD the host can learn the power class of the card, and choose to work with a wider data bus (See steps 26-37)
24-Send CMD6, writing 0x1 to the HS_TIMING byte of the EXT_CSD. The argument 0x03B9_0100 will do it.
24.1-The card might enter BUSY right after R1, if so, wait until the BUSY signal is de-asserted
24.2-After the card comes out of BUSY it is configured for high speed timing
25-Change the clock frequency to the chosen frequency (any frequency between 0 and 26/52MHz).

A.8.3 Changing the data bus width

The following steps are optionally done if the card’s power class allows the host to work on a wider bus, within the host power budget. Do these steps after the bus is initialized according to section Annex A.8.1 on page 141.

a. Bus testing procedure

26-Send CMD19
27-Send a block of data, over all the bus data lines, with the data pattern as follows (CRC16 is optional):
27.1-For 8 data lines the data block would be (MSB to LSB): 0x0000_0000_0000_AA55
27.2-For 4 data lines the data block would be (MSB to LSB): 0x0000_005A
27.3-For only 1 data line the data block would be: 0x80

	Start	Test Pattern								Optional	End
DAT7	0	0	1	0	0	0	0	0	0	CRC16	1
DAT6	0	1	0	0	0	0	0	0	0	CRC16	1
DAT5	0	0	1	0	0	0	0	0	0	CRC16	1
DAT4	0	1	0	0	0	0	0	0	0	CRC16	1
DAT3	0	0	1	0	0	0	0	0	0	CRC16	1
DAT2	0	1	0	0	0	0	0	0	0	CRC16	1
DAT1	0	0	1	0	0	0	0	0	0	CRC16	1
DAT0	0	1	0	0	0	0	0	0	0	CRC16	1
		LSB							MSB		
		0x55	0xAA	0x00	0x00	0x00	0x00	0x00	0x00		

Figure A.12 — Bus testing for eight data lines

	Start	Test Pattern								Optional	End
DAT3	0	0	1	0	0	0	0	0	0	CRC16	1
DAT2	0	1	0	0	0	0	0	0	0	CRC16	1
DAT1	0	0	1	0	0	0	0	0	0	CRC16	1
DAT0	0	1	0	0	0	0	0	0	0	CRC16	1
		LSB						MSB			
		0x5A		0x00		0x00		0x00			

Figure A.13 — Bus testing for four data lines

	Start	Test Pattern								Optional	End
DAT0	0	1	0	0	0	0	0	0	0	CRC16	1
		0x80									

Figure A.14 — Bus testing for one data line

28-Wait for at least N_{CR} clock cycles before proceeding
29-Send CMD14 and receive a block of data from all the available data lines[1]
29.1-For 8 data lines receive 8 bytes
29.2-For 4 data lines receive 4 bytes
29.3-For 1 data line receive 1 byte
30-XNOR the masked data with the data sent in step 27

[1] This represents the host expected values. The card always responds to CMD19 over all eight DAT lines.

Table A.6 — XNOR values

A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

31-Mask the result according to the following:
   31.1-For 8 data lines the mask is (MSB to LSB): 0x0000_0000_0000_FFFF
   31.2-For 4 data lines the mask is (MSB to LSB): 0x0000_00FF
   31.3-For 1 data line the mask is 0xC0
32-The result should be 0 for all. Any other result indicates a problem in the card connection to the system; in such a case the host must power down the bus and start its error recovery procedure (the definition of error recovery procedures is host dependent and out of the scope of this application note)

b. Power and bus-width selection

33-Choose the width of bus you want to work with
34-If the power class, for the chosen width, is different from the default power class, send CMD6, and write the POWER_CLASS byte of the EXT_CSD with the required power class.
35-The card might signal BUSY after CMD6; wait for the card to be out of BUSY
36-Send CMD6, writing the BUS_WIDTH byte of the EXT_CSD with the chosen bus width. An argument of 0x03B7_0100 will set a 4-bits bus, an argument 0x03B7_0200 will set an 8-bit bus.
37-The bus is ready to exchange data using the new width configuration.

A.9 Erase-unit size selection flow

The flow chart in Figure 15 shows how the master selects the erase unit size if the master supports the JEDEC MMC Electrical Interface Specification v4.3.

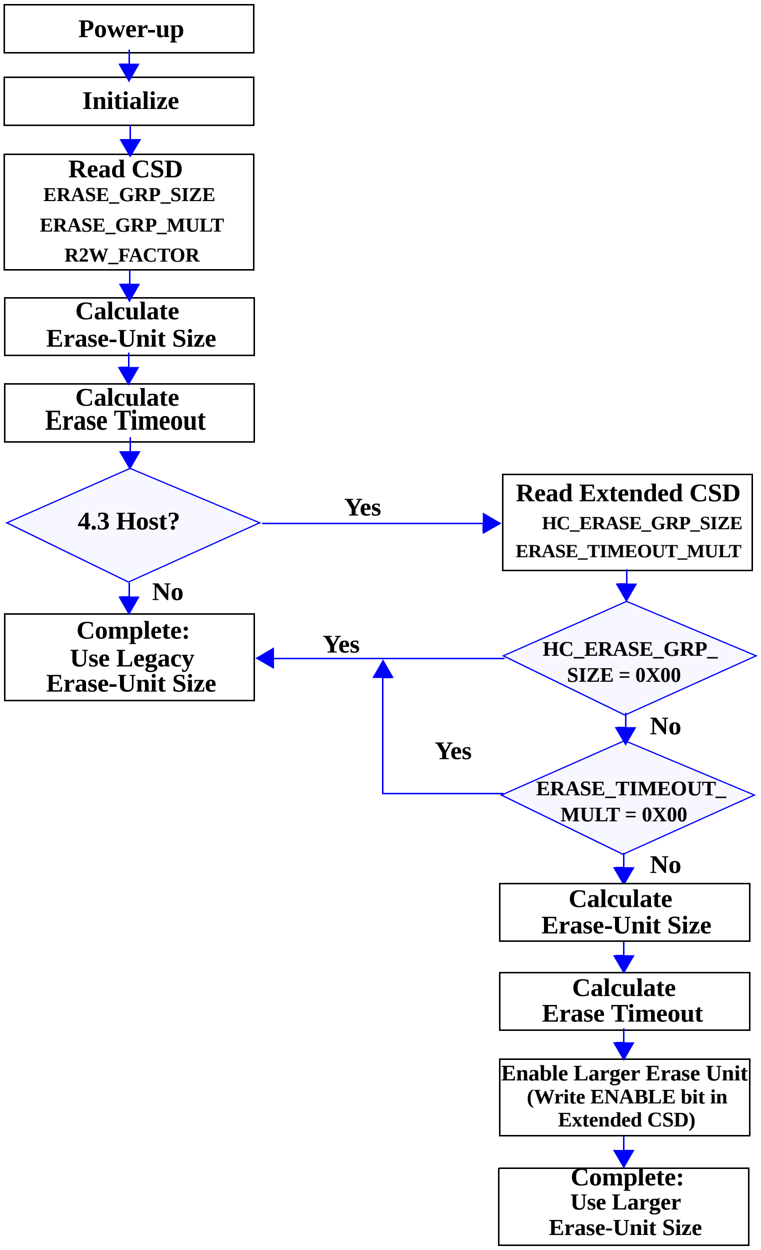


Figure A.15 — Erase-unit size selection flow

Annex B: Changes between system specification versions

B.1 Version 4.1, the first version of this specification

This Electrical Specification is derived from the MMCA System Specification version 4.1. There are no technical changes made. The editorial changes are listed below.

- The pin number references were removed (see Section 6.3 on page 10).
- The form factor references were removed (see Section 6.3 on page 10).
- The CSD_STRUCTURE and SPEC_VERS registers were modified to include only allocations applicable to this Electrical Specification (see Section 8.3 on page 79 and Section 8.5 on page 99).
- The S_CMD_SET allocations were removed from this specification and are defined in detail in a separate Application Note (see Section 8.5 on page 99).
- The mechanical specification was removed (see Section 11 on page 107).
- The Appendix A was removed and introduced as a separate document (see Annex A on page 129).

B.2 Changes from version 4.1 to 4.2

A major new item is handling densities greater then 2GB.

Additional changes include:

- A definition for implementation of media higher than 2GB was introduced (see Section 6.1 on page 10, Section 7 starting on page 21, Section 8 starting on page 77, and Section 9 on page 101).
- The definition for the card pull-up resistors was clarified (see Section 6.3 on page 10, Section 7.5.4 on page 35, and Section 12.5 on page 117).
- Switching between the tran state and standby states by CMD7 was clarified (see Section 7.5 on page 31 and Table 13 on page 52).
- A new register for indication of the state of an erased block was introduced (see Section 7.5.8 on page 42 and Section 8.5 on page 99).
- Command CMD39 argument was clarified (see Table 19 on page 56).
- The definition of busy indication during write operations was partly changed and partly clarified (see Section 7.14.7 on page 75).
- The minimum voltage of the Low-Voltage range was changed from 1.65V to 1.70V (see Section 12.5 on page 117).

B.3 Changes from version 4.2 to 4.3

Major new items added to this specification are the eMMC definition, boot operation, sleep mode, voltage configuration for eMMC, and reliable write. The chapter dedicated to SPI mode was removed.

Additional changes include:

- Added eMMC features (see Section 5 starting on page 5).
- Boot operation mode was introduced (see Section 7.2 on page 22, Section 7.14.5 on page 73, Section 8.5 on page 99, and Section 12.3 on page 111).
- Sector address definition for Erase and Write Protection was defined (see Section 7.5.8 on page 42 and Section 7.5.9 on page 42).

- CID register setting was changed to recognize either eMMC or a card (see Section 8.2 on page 77).
- The chapter defining SPI mode and all SPI-mode references were removed.
- Sleep mode was introduced (see Section 7.5.12 on page 46 and Section 8.5 on page 99).
- Voltage configuration for eMMC was defined (see Section 12.3.1 on page 112 through Section 12.3.3 on page 114, Section 12.5.3 on page 118, and Section 12.5.4 on page 118).
- Reliable Write was defined (see Section 7.5.7 on page 39, under “Block Write,” and Table 49 on page 88).
- Input capacitance for eMMC was defined (see Section 12.5 on page 117).
- New bus timings (setup & Hold) were redefined (see Section 12.7 on page 121).
- Switch command definition was clarified (see Section 7.5.1 on page 33).
- Peak voltage on all signal lines are redefined for card and defined for eMMC (see Section 12.5 on page 117).
- Redefined Access size register (see Section 8.5 on page 99).
- Redefined input capacitance for MMC*micro*, MMC*mobile*, and MMC*plus* (see Section 12.5.5 on page 119).
- Redefined erase-unit size and erase timeout for high-capacity memory (see Section 8.5 on page 99).
- Removed “Absolute Minimum” section formerly section 4.8.2.
- Defined OCR setting and response for eMMC (see Section 7.3.2 on page 28).
- Defined high-capacity WP group size (see Section 7.5.9 on page 42, Section 7.9.4 on page 51, Section 7.13 on page 65, and Section 8.5 on page 99).
- Alternate boot operation (device-optional) introduced (see Section 7.2 on page 22, and Section 7.14.7 on page 75).
- Added “/JEDEC” to “MMCA” as the source of definitions for MID and OID (see Section 6.4.2 on page 13, “MID” on page 78, and “OID” on page 78.)

Standard Improvement Form   JEDEC   84-A43

The purpose of this form is to provide the Technical Committees of JEDEC with input from the industry regarding usage of the subject standard. Individuals or companies are invited to submit comments to JEDEC. All comments will be collected and dispersed to the appropriate committee(s).

If you can provide input, please complete this form and return to:

JEDEC
Attn: Publications Department
2500 Wilson Blvd. Suite 220
Arlington, VA 22201-3834
Fax: 703.907.7583

1. I recommend changes to the following:

□ Requirement, paragraph number: ____________________

□ Test method number: ____________ Paragraph number: ________________

The referenced paragraph number has proven to be:

□ Unclear □ Too rigid □ In error

□ Other: ____

____

2. Recommendations for correction:

____

____

____

____

3. Other suggestions for document improvement:

____

____

____

____

Submitted by:

Name: ______________________ Phone: ______________

Company: ______________________ Email: ______________

Address ______________________

City/State/Zip ______________________ Date: ______________

JEDEC

M®
MULTIMEDIACARD
ASSOCIATION