JEDEC STANDARD

MultiMediaCard (MMC) Electrical Standard, Standard Capacity
(MMCA, 4.1)

JESD84-B41

JUNE 2007

JEDEC SOLID STATE TECHNOLOGY ASSOCIATION

NOTICE

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.

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

CONTENTS

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

CONTENTS*(continued)*

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

CONTENTS*(continued)*

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

CONTENTS*(continued)*

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

CONTENTS*(continued)*

Page

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

FIGURES

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

FIGURES*(continued)*

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

TABLES

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

TABLES*(continued)*

Foreword

The MultiMediaCard Electrical Specification has been prepared by the MultiMediaCard Association, hereafter referred to as MMCA. The MMCA exists to promote adoption of a global standard for a compact, robust, affordable storage and retrieval device, the MultiMediaCard. Consumers worldwide will benefit from this standard, allowing them to carry with them, information and entertainment that fits their needs, wherever they are, whenever they wish.

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. These high-level differences are detailed in the JEDEC JC-64 Top-Level Specification.

The purpose of the specification is the definition of the eMMC, 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.

This standard implements the SPI standard.

This standard submission is intended to entirely supplant previously submitted MMC specifications.

Introduction

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.

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 ensure compatibility with existing controllers, the cards may offer, in addition to the MultiMediaCard mode, an alternate communication protocol which is based on the SPI standard.

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

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

1 Scope

This document provides a comprehensive definition of the MultiMediaCard, 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.

The eMMC electrical specification implements the Serial Peripheral Interface (SPI) standard.

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
Flash	a type of multiple time programmable non volatile memory
Group	a number of write blocks, composite erase and write protect unit
LOW, HIGH	binary interface states with defined assignment to a voltage level
NSAC	defines the worst case for the clock rate dependent factor of the data access time
MSB, LSB	the Most Significant Bit or Least Significant Bit
OCR	Operation Conditions Register
open-drain	a logical interface operation mode. An external resistor or current source is used to pull the interface level to HIGH, the internal transistor pushes it to LOW

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.

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 support a wide range of applications such 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. Additional features include low power consumption and high data throughput at the memory card interface.

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 ensure compatibility with existing controllers, in addition to the MultiMediaCard mode, cards may offer, an alternative communication protocol based on the SPI standard.

To provide for the forecasted migration of CMOS power (V_{DD}) requirements and for compatibility and integrity of MultiMediaCard systems, two types of MultiMediaCards are defined in this standard specification; they 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 specification is to define the MultiMediaCard, its environment, and its handling. The specification provides guidelines for a systems designers. The specification also defines a tool box (a set of macro functions and algorithms) that contributes to reducing design-in costs.

The document is presented in several chapters. 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 described in Section 9.

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; it also defines the electrical parameters of the interface.

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

Three basic valid file formats for achieving high data interchangeability are defined in Section 14.

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.

Annex C lists all the known errata items related to the JEDEC Non-Volatile Memory Card and Module Electrical Specifications.

NOTE: 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 serve a variety of purposes, including:

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

The following list identifies the main attributes of the MultiMediaCard System:

- Targets portable and stationary applications
- Uses these system voltage (V_{DD}) ranges:

Table 1 — System voltage ranges

	High Voltage MultiMediaCard	Dual Voltage MultiMediaCard
Communication	2.7–3.6	1.65–1.95, 2.7–3.6
Memory Access	2.7–3.6	1.65–1.95, 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, 0–26 MHz or 0–52 MHz
- Has a maximum data rate up to 416 Mbits/sec.
- Has a defined minimum performance
- Maintains card support for three different data bus width modes: 1bit (default), 4bit, and 8bit
- 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 multiple command sets
- Includes attributes of the available operation modes noted here:

Table 2 — MMC operational modes

MultiMediaCard mode	SPI mode
Ten-wire bus (clock, 1 bit command, 8 bit data bus)	Three-wire serial data bus (clock, dataIn, dataOut) + card-specific CS signal.
Card selection is done through an assigned unique card address to maintain backwards compatibility to prior versions of the specification	Card selection via a hardware CS signal
One card per MultiMediaCard bus	Card requires a dedicated CS signal.
Easy identification and assignment of session address	Not available. Card selection via a hardware CS signal
Error-protected data transfer	Optional. A non-protected data transfer mode is available.
Sequential and Single/Multiple block Read/Write commands	Single/Multiple block Read/Write commands

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.

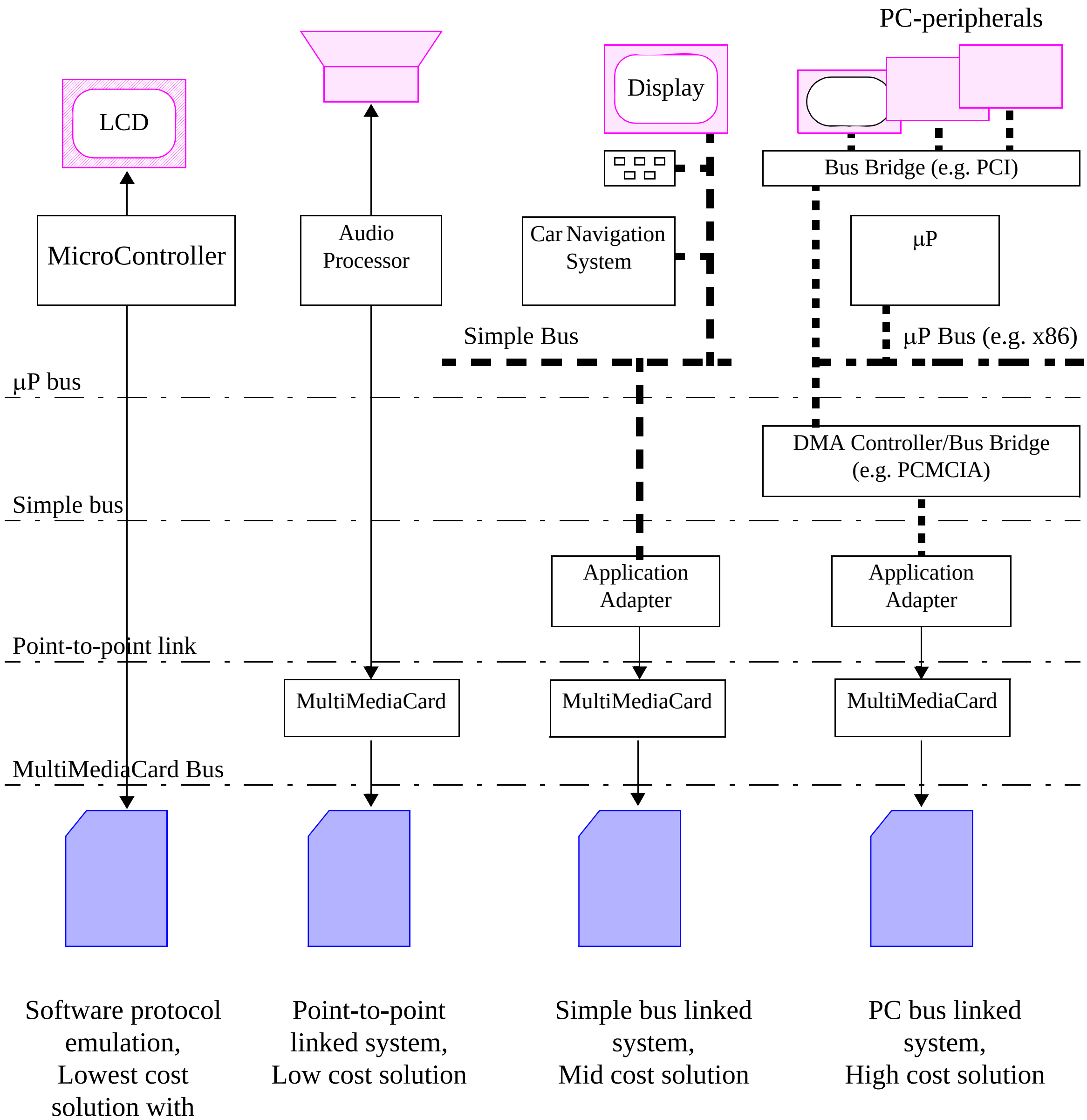


Figure 1 — Topology of MultiMediaCard systems

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.

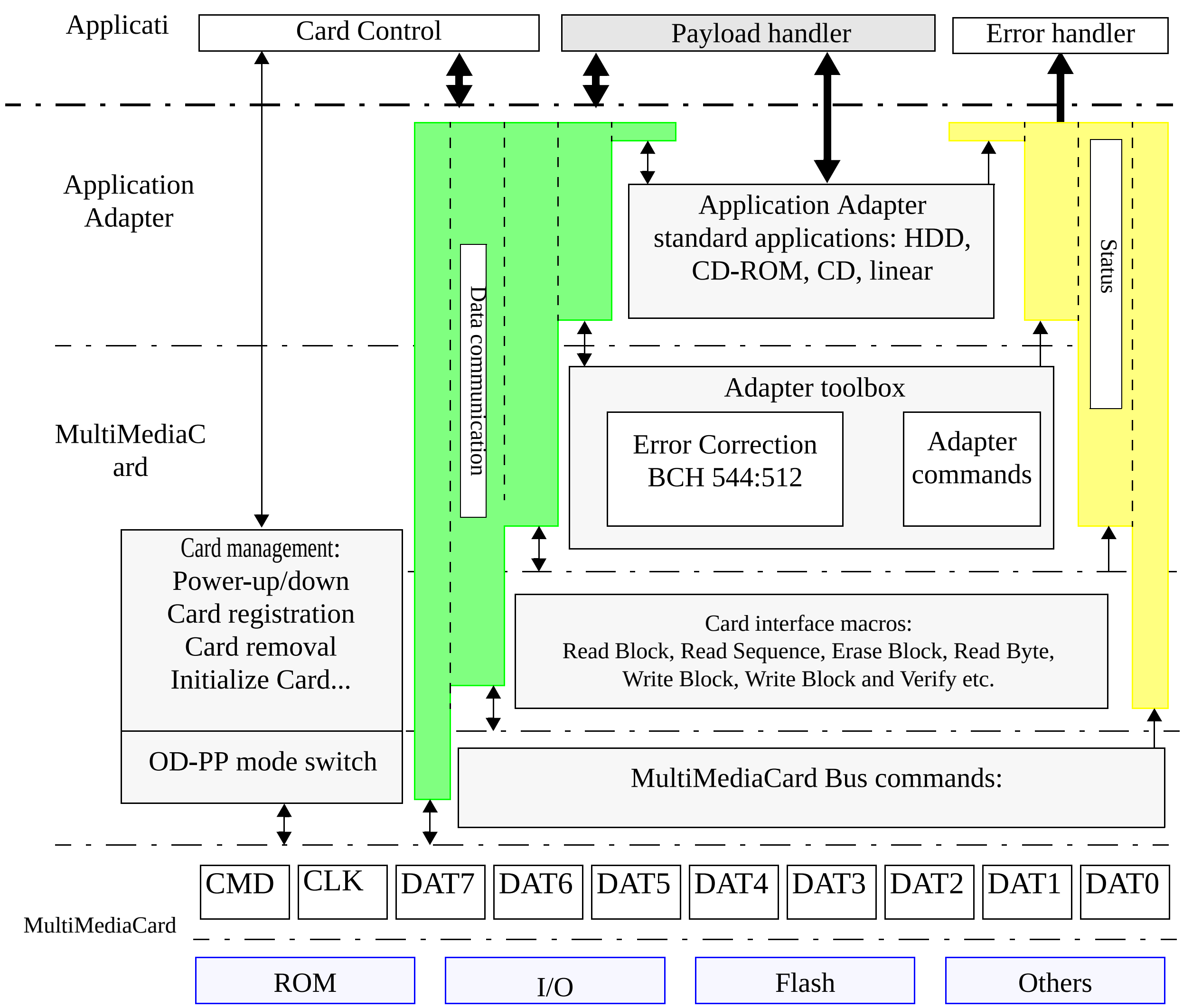


Figure 2 — MultiMediaCard system overview

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

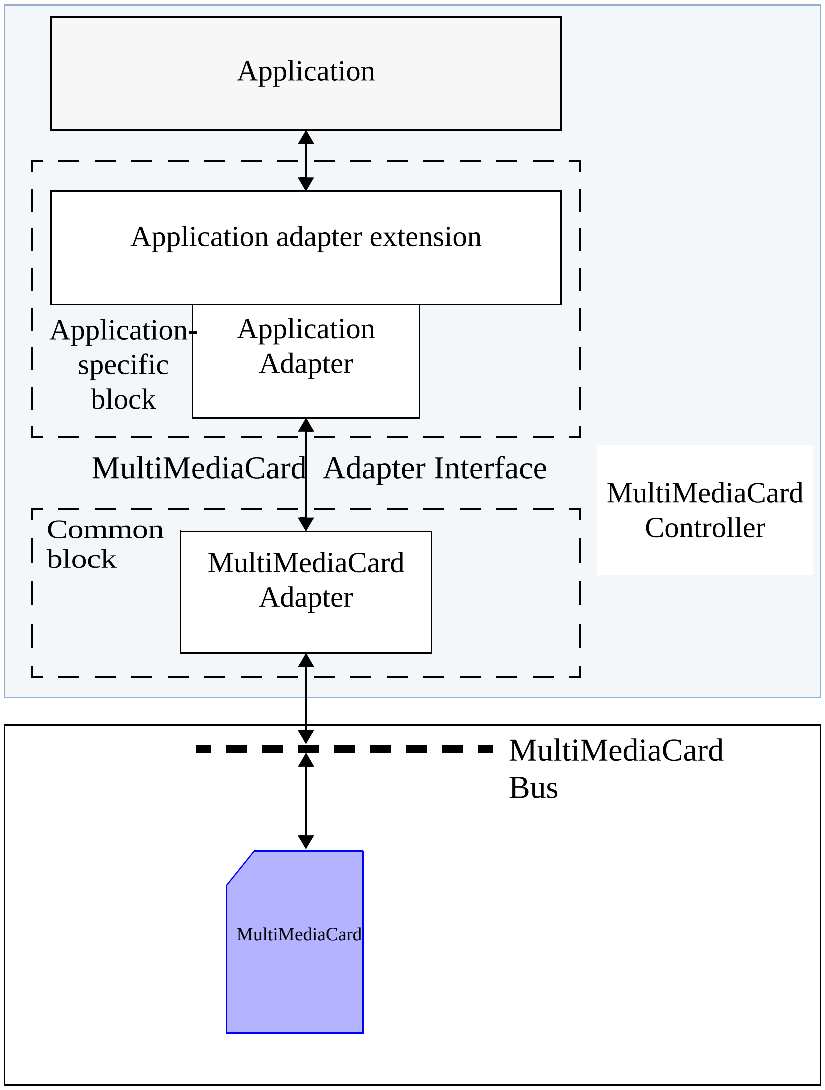


Figure 3 — MultiMediaCard system example

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., displays 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 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. The MultiMediaCard includes internal pull ups for all data lines. 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.

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 — Card contacts

Name[1]	**Type**[2]	**Description**
DAT3	I/O/PP	Data
CMD	I/O/PP/OD	Command/Response
V_{SS1}	S	Supply voltage ground
V_{DD}	S	Supply voltage
CLK	I	Clock
V_{SS2}	S	Supply voltage ground
DAT0[3]	I/O/PP	Data
DAT1	I/O/PP	Data

Table 3 — Card contacts (continued)

Name[1]	Type[2]	Description
DAT2	I/O/PP	Data
DAT4	I/O/PP	Data
DAT5	I/O/PP	Data
DAT6	I/O/PP	Data
DAT7	I/O/PP	Data

NOTE 1: See Table 60 on page 84, for a combined table including SPI pin definitions.

NOTE 2: S: power supply; I: input; O: output; PP: push-pull; OD: open-drain; NC: Not connected (or logical high).

NOTE 3: 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 67):

Table 4 — Card 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.

6.1.1 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.

- MMCplus is defined as normal size R/W or ROM cards that supports 2.7-3.6V operation, x1/x4/x8 bus widths, minimum of 2.4MB/s read/write performance and 26MHz (52MHz optional)
- MMCmobile is defined as reduced size R/W or ROM card that supports 1.65-1.95V 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 Specification versions 3.xx in max 20 MHz clock frequency mode.

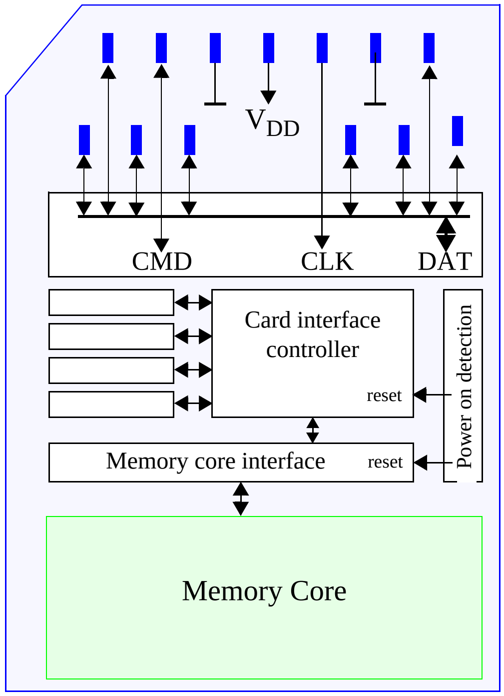


Figure 4 — MultiMediaCard architecture

6.2 Bus Concept

The MultiMediaCard bus is designed to connect either solid-state mass-storage memory or I/O-devices in a card format to multimedia applications. The bus implementation allows the coverage of application fields from low-cost systems to systems with a fast data transfer rate. It is a single master bus with a 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 (see Figure 5 on page 13).

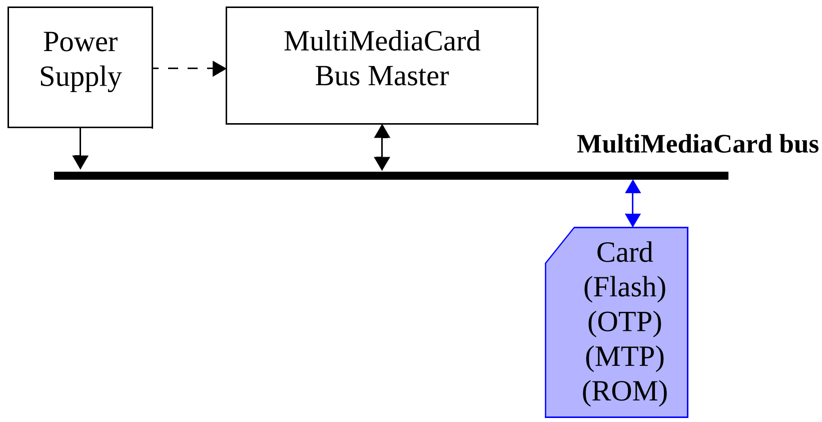


Figure 5 — MultiMediaCard bus system

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.

6.2.1 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.
- 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 6.1.

6.2.2 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) which are defined by the MMCA. Every card manufacturer is required to apply for an unique MID (and optionally OID) number.

The structure of commands, responses and data blocks is described in Section 7 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.

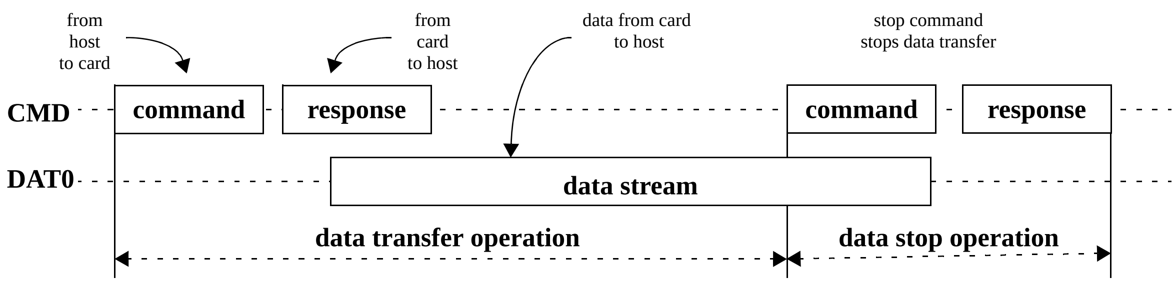


Figure 6 — Sequential read pperation

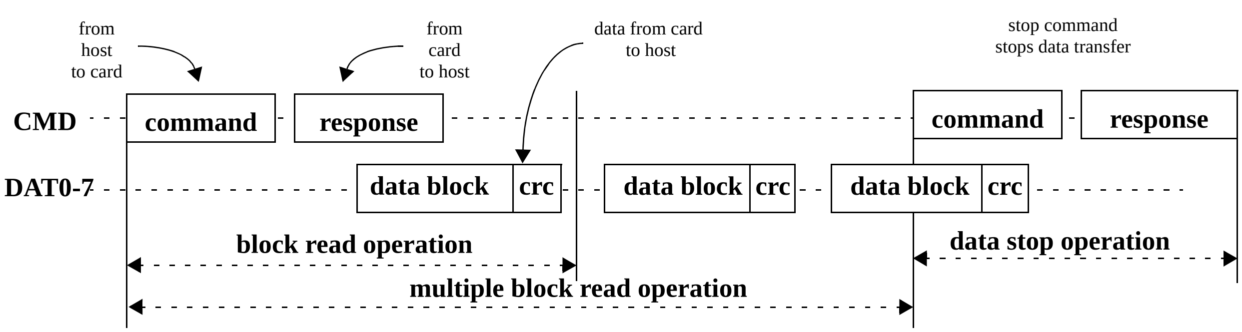


Figure 7 — (Multiple) Block read operation

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

.

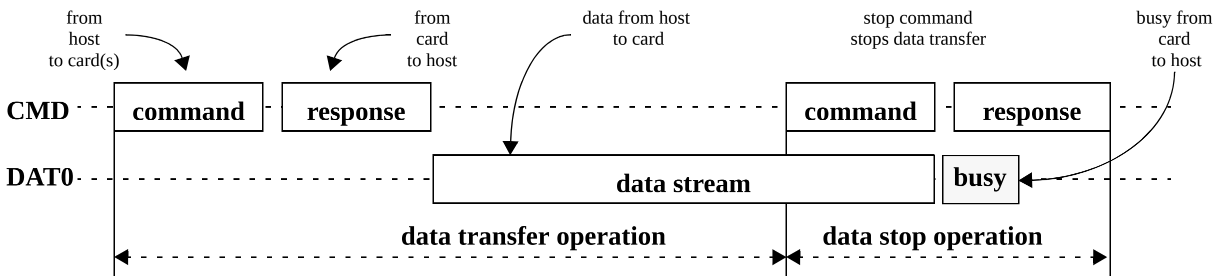


Figure 8 — Sequential write operation

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

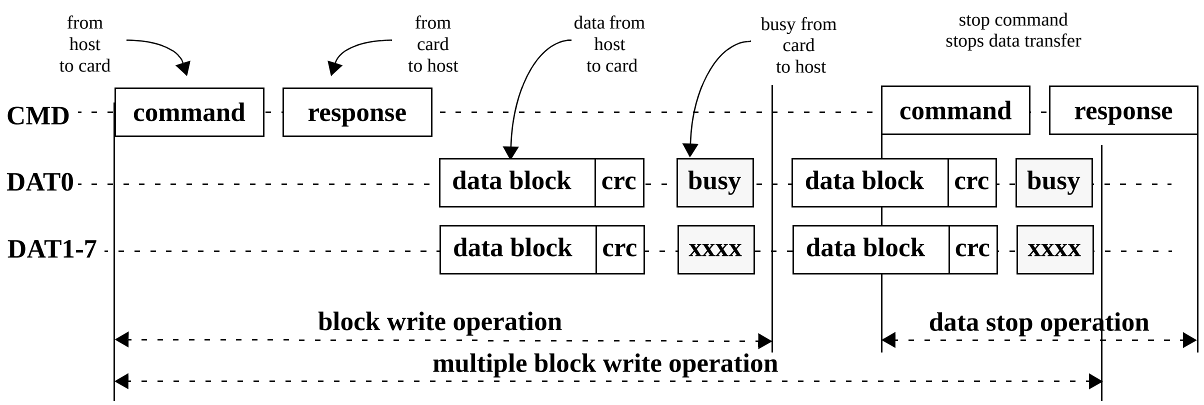


Figure 9 — (Multiple) Block write operation

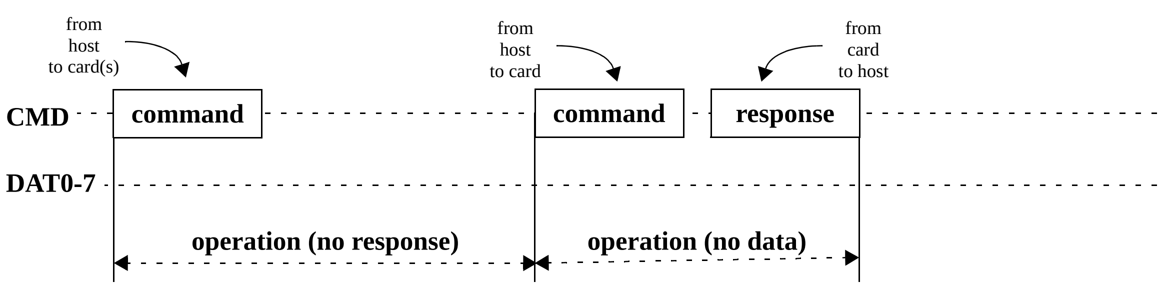


Figure 10 — “No response” and “No data” operations

Command tokens have the following coding scheme:

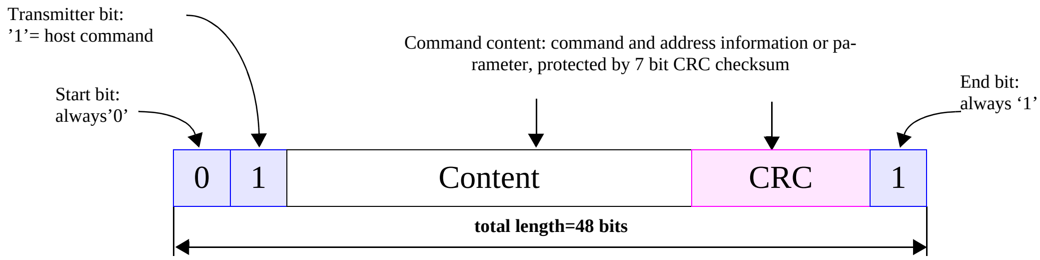


Figure 11 — Command token format

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 commands and response definition is given in Section 7.7 and Section 7.9.

Due to the fact that there is no predefined end in sequential data transfer, no CRC protection is included in this case. The CRC protection algorithm for block data is a 16 bit CCITT polynomial. All used CRC types are described in Section 6.

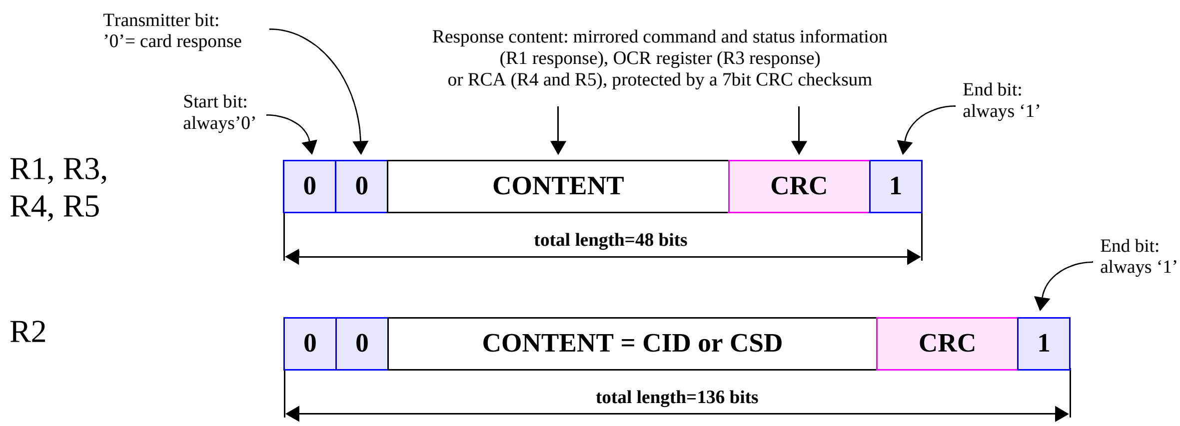


Figure 12 — Response token format

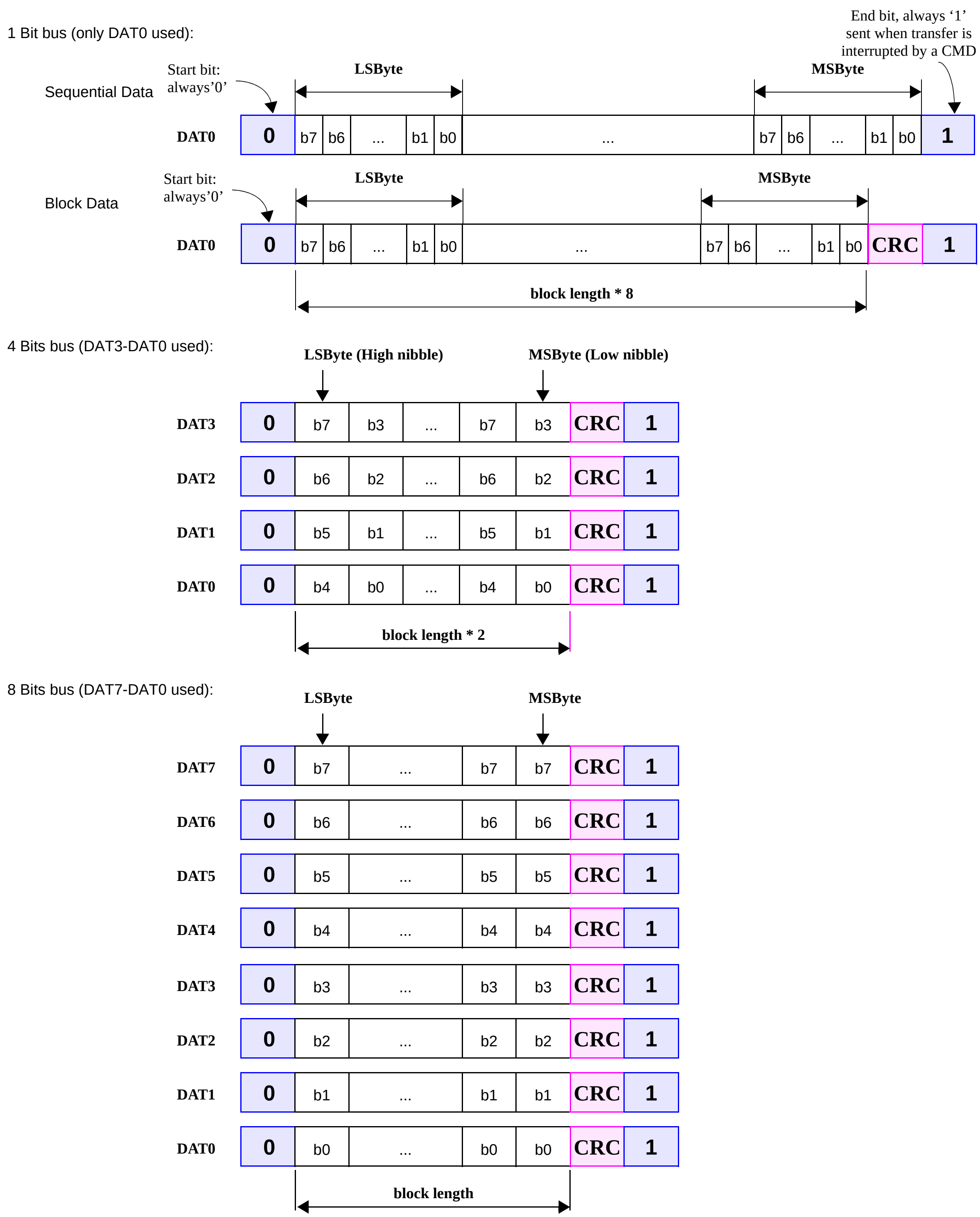


Figure 13 — Data packet format

6.3 Controller Concept

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

- Protocol translation from standard MultiMediaCard bus to application bus
- Data buffering to enable minimal data access latency
- 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

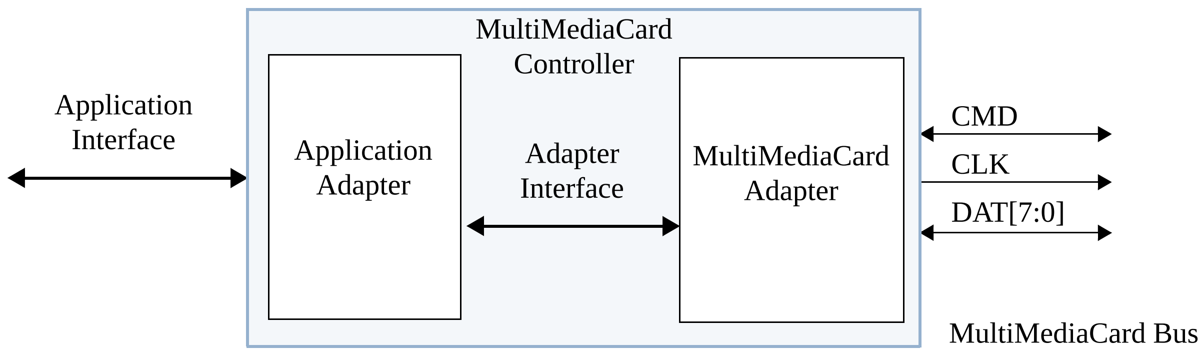


Figure 14 — MultiMediaCard controller scheme

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 109.

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.

6.3.1 Application Adapter Requirements

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

6.3.2 MultiMediaCard Adapter Architecture

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

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

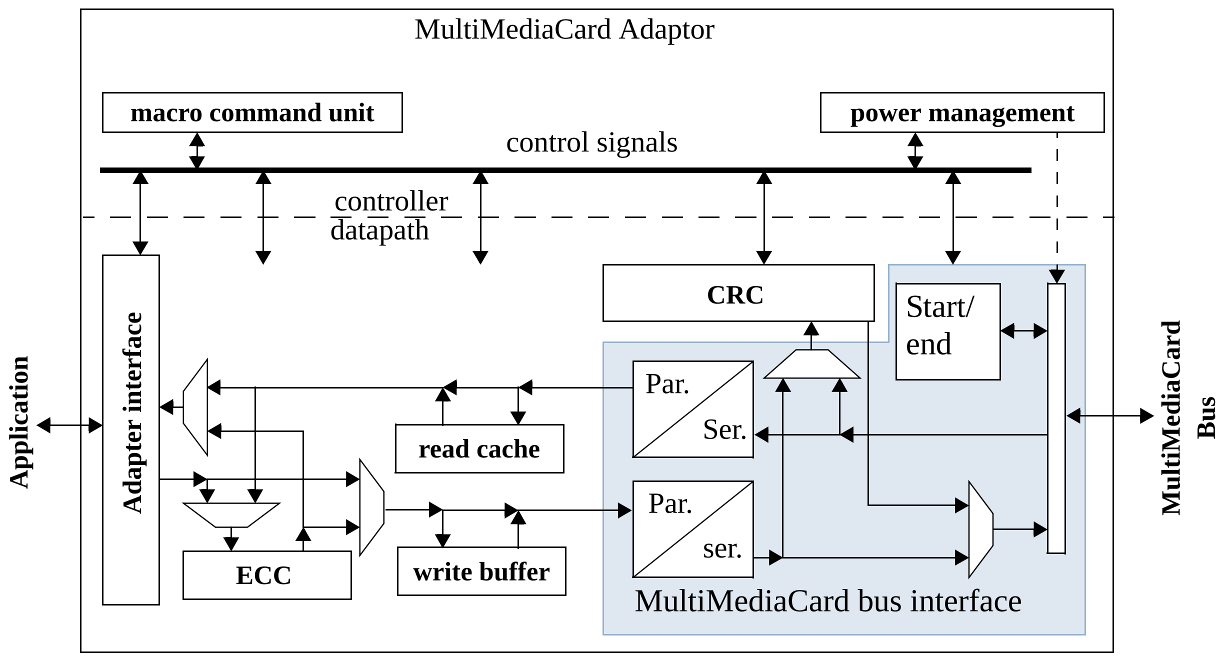


Figure 15 — MultiMediaCard adapter architecture

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 16 for the card identification mode and in Figure 18 for the data transfer mode. The commands are listed in the command tables (Table 14 to Table 22). The dependencies between current state, received command and following state are listed in Table 23 on page 49. 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 (Table 5) shows the dependencies between bus modes, operation modes and card states. Each state in the MultiMediaCard state diagram (see Figure 16 and Figure 18) is associated with one bus mode and one operation mode:

Table 5 — Card states and modes

Card state	**Operation mode**	**Bus mode**
Inactive State	Inactive	Open-drain
Idle State	Card identification mode	
Ready State		
Identification State		

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

Table 5 — Card states and modes (continued)

Card state	Operation mode	Bus mode
Stand-by State	Data transfer mode	Push-pull
Transfer State		
Bus-Test State		
Sending-data State		
Receive-data State		
Programming State		
Disconnect State		
Wait-IRQ State	Interrupt mode	Open-drain

7.2 Card Identification Mode

While in card identification mode the host resets the card, validates operation voltage range, 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.

7.2.1 Card Reset

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

Command GO_IDLE_STATE (CMD0) is the software reset command and puts the card into *Idle State*. It is also used to switch the card into SPI mode. Refer to Section 9 for details.

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 82. The host clocks the bus at the identification clock rate $f_{OD,}$ as described in Section 12.7 on page 122.

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

7.2.2 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 120.)

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 67.) 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. For this, the levels in the OCR register shall be defined accordingly similarly described in Section 8.1.

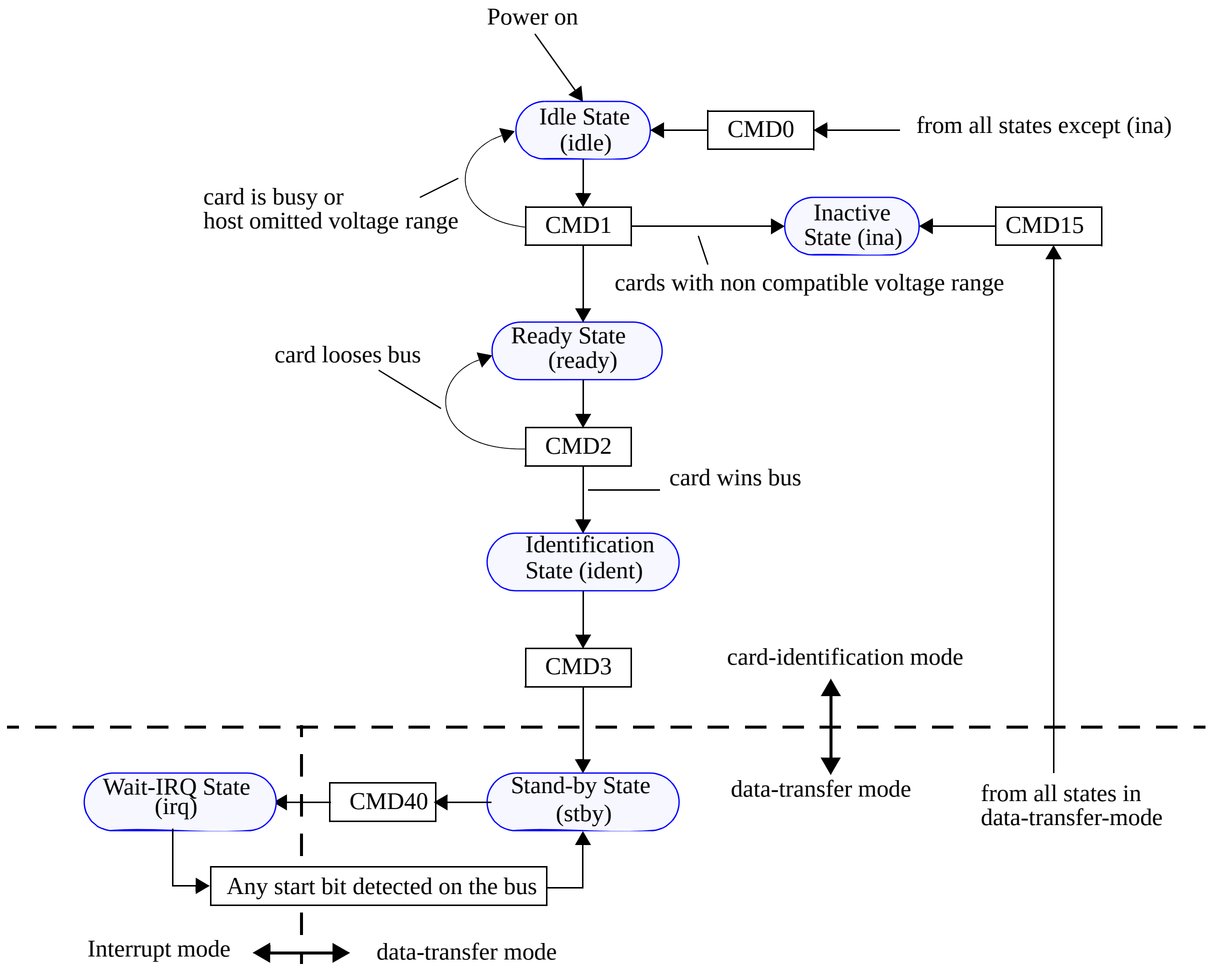


Figure 16 — MultiMediaCard state diagram (card identification mode)

By omitting the voltage range in the command (by setting the argument of CMD1 to 0), the host can query the card and determine the voltage type of the card. This bus query should be used if the host is able to select a common voltage range, or if a notification to the application of a non usable card in the bus is desired. Afterwards, the host must choose a voltage for operation, and reissue CMD1 with this condition, sending incompatible cards into the *Inactive State*.

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

During the initialization procedure, the host is not allowed to change the operating voltage range. Such changes shall be ignored by the card. If there is a real change in the operating conditions, the host must reset the card (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.

If the host intends to operate the Dual Voltage MultiMediaCards in the 1.65V to 1.95V range, it is recommended that the host first validate the operating voltage in the 2.7V to 3.6V range, then power the card down fully, and finally power the card back up to the 1.65V to 1.95V range for operation. Using the 2.7V to 3.6V range initially, which is common to High and 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.

7.2.3 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 122.) 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 7.13 on page 58.)

7.3 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 17.

- 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.

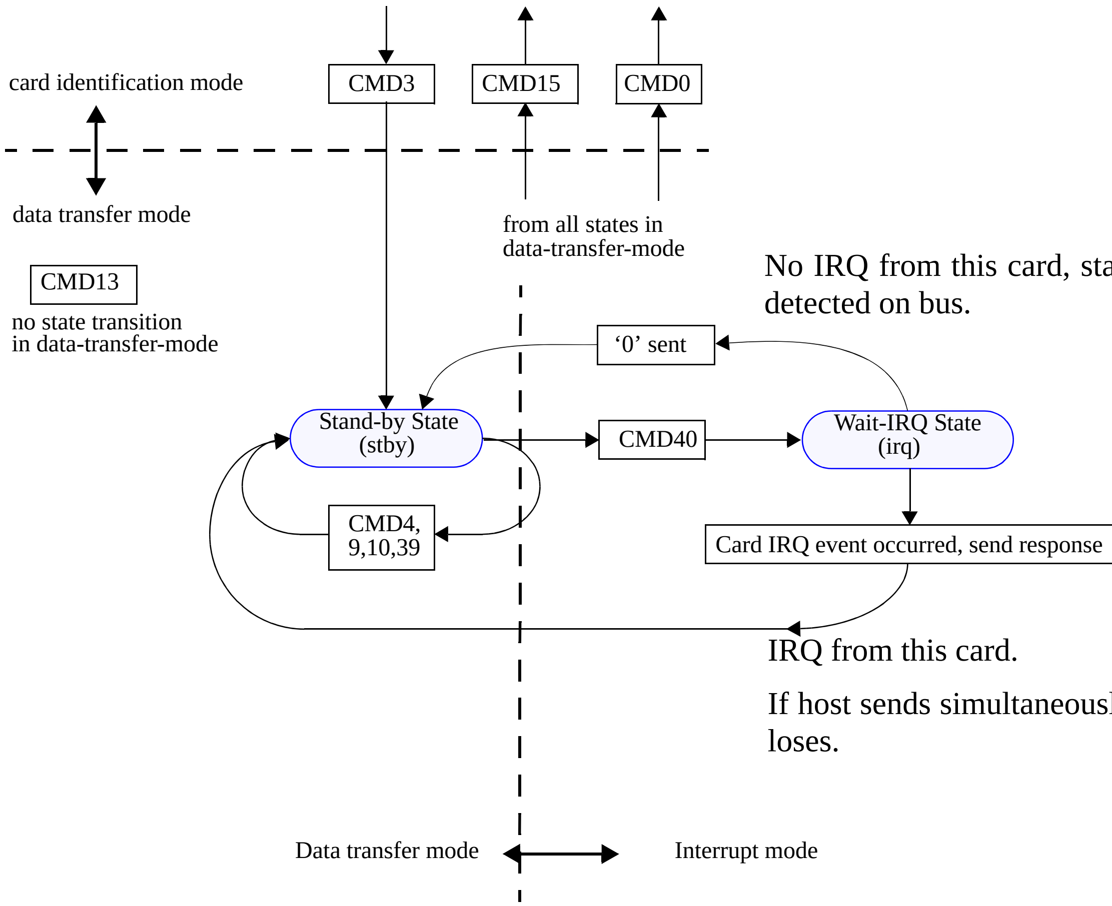


Figure 17 — MultiMediaCard state transition diagram, interrupt mode

- If the host wants to terminate the interrupt mode before an interrupt response is received, it can generate the CMD40 response (with card bit = 0) using the reserved RCA address 0x0000. This will bring the card from Wait-IRQ-State back into the Standby State. Then the host can resume the standard communication procedure.

7.4 Data Transfer Mode

When the card is in *Standby 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 122.) 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.

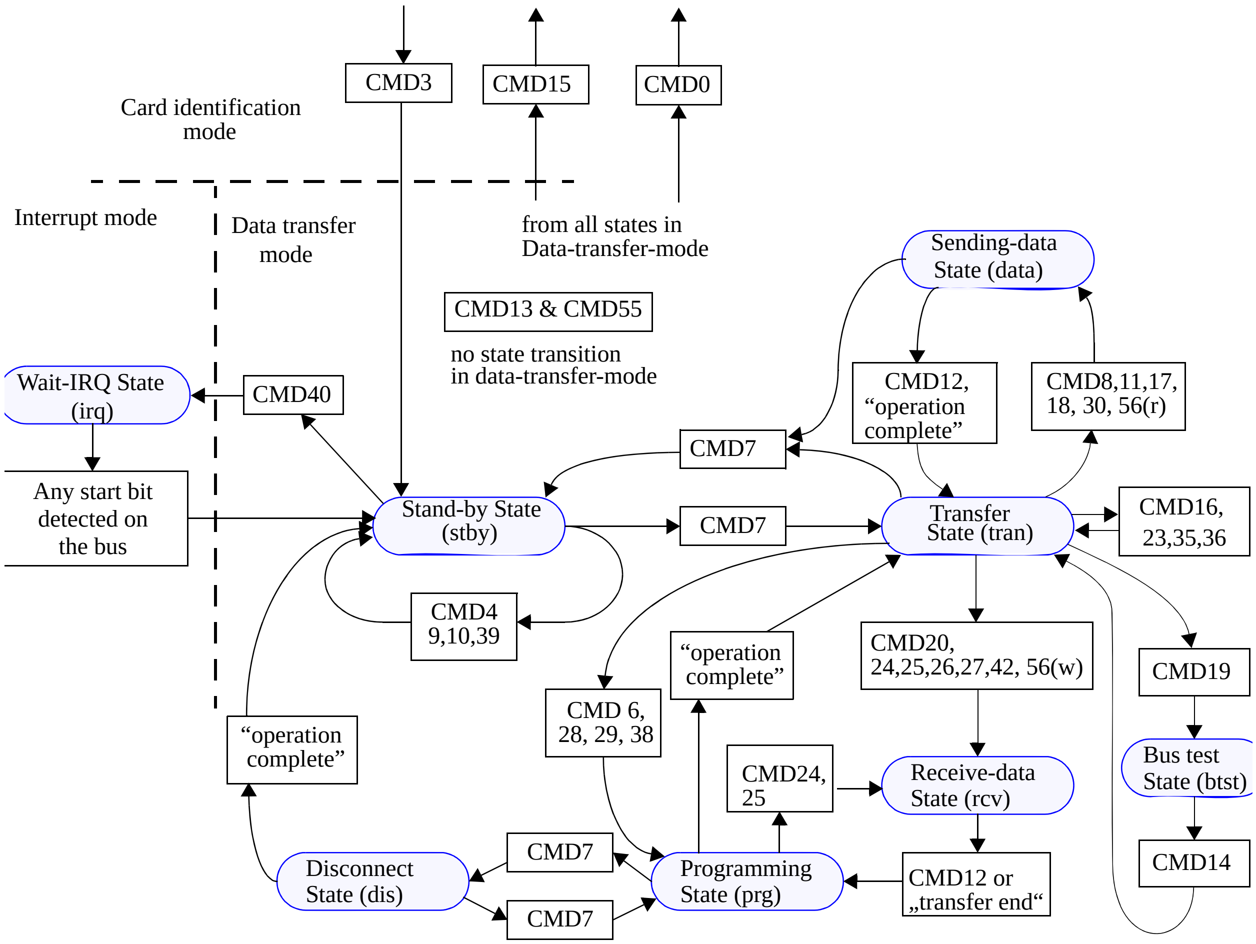


Figure 18 — MultiMediaCard state diagram (data transfer mode)

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.

CMD7 is used to select the card and put it into the *Transfer State*. 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 CMD7 is issued with the reserved relative card address "0x0000", the card is put back to *Stand-by State*. After the card is assigned an RCA it will not respond to identification commands — CMD1, CMD2, or CMD3. (See Section 7.2.3 on page 24).

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 18):

- All data read commands can be aborted any time by the stop command (CMD12). The data transfer will terminate and the card will return to the *Transfer State*. The read commands are: stream read (CMD11), block read (CMD17), multiple block read (CMD18) and send write protect (CMD30).

- All data write commands can be aborted any time by the stop command (CMD12). The write commands must be stopped prior to deselecting the card by CMD7. The write commands are: stream write (CMD20), block write (CMD24 and CMD25), write CID (CMD26), and write CSD (CMD27).
- If a stream write operation is stopped prior to reaching the block boundary and partial blocks are allowed (as defined in the CSD), the part of the last block will be packed as a partial block and programmed. If partial blocks are not allowed the data will be discarded.
- As soon as the data transfer is completed, the card will exit the data write state and move either to the *Programming State* (transfer is successful) or *Transfer State* (transfer failed).
- If a block write operation is stopped and the block length and CRC of the last block are valid, the data will be programmed.
- If data transfer in stream write mode is stopped, not byte aligned, the bits of the incomplete byte are ignored and not programmed.
- The card may provide buffering for stream and block write. This means that the next block can be sent to the card while the previous is being programmed.
  If all write buffers are full, and as long as the card is in *Programming State* (see MultiMediaCard state diagram Figure 18), the DAT0 line will be kept low.
- There is no buffering option for write CSD, write CID, write protection and erase. This means that while the card is busy servicing any one of these commands, no other data transfer commands will be accepted. 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 68), and the other status flags in the Card Status (Section 7.11 on page 53).

7.4.1 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[4].

Table 6 — Card access modes

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.

7.4.2 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 79 on page 123. 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 82. 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.

7.4.3 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 79. If the host tries to write an invalid value, the POWER_CLASS byte is not changed and the SWITCH_ERROR bit is set.

4. 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.

7.4.4 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 data 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 10.) The reverse data pattern sent by the card may optionally include a CRC16 checksum, which is ignored by the host.

The card has pull ups in all data inputs. In cases where the card is connected to only a 1-bit or 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
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	

7.4.5 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 82. 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.

7.4.6 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 26, Figure 27, and Figure 28 on page 61). 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 'CRC', on page 68).

- **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 being defined in Section 8. 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 7.6.2 on page 40) 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 7.6.2) should be used:

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

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

- **Block Read**

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

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

- Open-ended Multiple block read

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

- Multiple block read with pre-defined block count

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

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

If 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).

7.4.7 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 ’CRC’, on page 68) for each received data block prior to the write operation. By this mechanism, writing of erroneously transferred data can be prevented.

- **Stream Write**

Stream write (CMD20) starts the data transfer from the host to the card beginning from the starting address until the host issues a stop command. If partial blocks are allowed (if CSD parameter WRITE_BL_PARTIAL is set) the data stream can start and stop at any address within the card address space, otherwise it shall start and stop only at block boundaries. Since the amount of data to be transferred is not determined in advance, CRC can not be used.

If the 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. 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 7.6.2 on page 40), 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 7.6.2) should be used:

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

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

- **Block Write**

During block write (CMD24 - 27) one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. A card supporting block write shall always be able to accept a block of data defined by WRITE_BL_LEN. If the CRC fails, the card shall indicate the failure on the 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. Two types of multiple block write transactions, identical to the multiple block read, are defined (the host can use either one at any time):

- Open-ended Multiple block write

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

- Multiple block write with pre-defined block count

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

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

If 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 if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) at any time, and the card will respond with its status. The status bit READY_FOR_DATA indicates whether the card can accept new data or whether the write process is still in progress). The host may deselect the card by issuing CMD7 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 if programming is still in progress and the write buffer is unavailable.

7.4.8 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. The content of an explicitly erased memory range shall be 0.

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

If an erase command (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.

7.4.9 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. The segment size is defined in units of WP_GRP_SIZE erase groups as specified in the CSD. The SET_WRITE_PROT command sets the write protection of the addressed write-protect group, and the CLR_WRITE_PROT command clears the write protection of the addressed write-protect group.

The SEND_WRITE_PROT command is similar to a single block read command. The card shall send a data block containing 32 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits. The address field in the write protect commands is a group address in byte units. The card will ignore all LSB’s below the group size.

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.

7.4.10 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 — Command data block 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 a 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 the sent password, 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 7.6.2 on page 40.

7.4.11 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 R1b (card status + busy indication).

7.5 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.

7.6 Error Conditions

7.6.1 CRC and Illegal Command

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

Similarly, if an illegal command has been received, 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 16 on page 23 to Figure 18 on page 26). Table 23 on page 49 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).

7.6.2 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 7.13 on page 58). 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.

- **Force Erase**

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

7.6.3 Read ahead in Stream and multiple block read operation

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

7.7 Minimum Performance

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

7.7.1 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 82). Only the defined values and classes are allowed to be used.

7.7.2 Absolute Minimum

Absolute minimum read and write access performance which all MMCplus and MMCmobile cards has to fulfill is 2.4MB/s. This is the Class A.

7.7.3 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 Section 8.5 on page 82.

7.8 Commands

7.8.1 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.

7.8.2 Command Format

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

Table 12 — Command code length

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

A command always starts with a start bit (always ‘0’), followed by the bit indicating the direction of transmission (host = ‘1’). The next 6 bits indicate the index of the command, this value being interpreted as a binary coded number (between 0 and 63). Some commands need an argument (e.g. an address), which is coded by 32 bits. A value denoted by ‘x’ in the table above indicates this variable is dependent on the command. All commands are protected by a CRC (see ’CRC’, on page 68 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 14 on page 44 through Table 22 on page 48.

7.8.3 Command Classes

The command set of the MultiMediaCard system is divided into several classes (see Table 13 on page 43). Each class supports a subset of card functions.

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

Table 13 — Card command classes

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

7.8.4 Detailed Command Description

The following tables define in detail all MultiMediaCard bus commands. The responses R1-R5 are defined in Section 7.10 on page 51. The registers CID, CSD, EXT_CSD and DSR are described in Section 8.

Table 14 — Basic commands and read stream commands: 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	reserved				
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 7.4.1 on page 27.)
CMD7	ac	[31:16] RCA [15:0] stuff bits	R1b[1]	SELECT/ DESELECT_CARD	Command toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases the card is selected by its own relative address and gets deselected by any other address; address 0 deselects 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	R1b	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
CMD19	adtc	[31:0] stuff bits	R1	BUSTEST_W	A host sends the bus test data pattern to a card.

Table 14 — Basic commands and read stream commands: class 0 and class 1 (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description

NOTE 1 Only from the selected card.
NOTE 2 The addressing capability @ 8-bit address resolution is 2^{32} = 4 Gbyte

Table 15 — 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	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.[1]
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command, or the requested number of data blocks is transmitted

NOTE 1 The transferred data must not cross a physical block boundary, unless READ_BLK_MISALIGN is set in the CSD register.

Table 16 — Stream write commands: class 3

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD20	adtc	[31:0] data address	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				

Table 17 — Block-oriented write commands: class 4

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD23	ac	[31:16] set to 0 [15:0] number of blocks	R1	SET_BLOCK_COUNT	Defines the number of blocks which are going to be transferred in the immediately succeeding multiple block read or write command. If the argument is all 0s, the subsequent read/write operation will be open-ended.
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.[1]
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows or the requested number of block received.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command shall be issued only once. 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 The transferred data must not cross a physical block boundary unless WRITE_BLK_MISALIGN is set in the CSD.

Table 18 — Block-oriented wrtite protection commands: class 6

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.

Table 18 — Block-oriented wrtite protection commands: class 6

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.[1]
CMD31	reserved				

NOTE 1 32 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits are transferred in a payload format via the data 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 19 — 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	R1	ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase
CMD36	ac	[31:0] data address	R1	ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase
CMD37	Reserved. This command index cannot be used in order to maintain backwards compatibility with older versions of the MultiMediaCards.				
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erases all previously selected write blocks

Table 20 — 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. 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 21 — Lock card: class 7

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43 … CMD54	reserved				

Table 22 — 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]	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block shall be set by the SET_BLOCK_LEN command.
CMD57 … CMD59	reserved				

Table 22 — Application-specific commands: class 8 (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD60 … CMD63	reserved for manufacturer				
NOTE 1 RD/WR: "1" the host gets a block of data from the card; "0" the host sends a block of data to the card.					

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

7.9 Card State Transition Table

Table 23 defines the card state transitions in dependency of the received command.

Table 23 — Card state transition table

	Current State											
	idle	ready	ident	stby	tran	data	btst	rcv	prg	dis	ina	irq
Command	**Changes to**											
Class Independent												
CRC error	-	-	-	-	-	-	-	-	-	-	-	stby
command not supported	-	-	-	-	-	-	-	-	-	-	-	stby
Class 0												
CMD0	idle	idle	idle	idle	idle	idle	idle	idle	idle	idle	-	stby
CMD1, card V_{DD} range compatible	ready	-	-	-	-	-	-	-	-	-	-	stby
CMD1, card is busy	idle	-	-	-	-	-	-	-	-	-	-	stby
CMD1, card V_{DD} range not compatible	ina	-	-	-	-	-	-	-	-	-	-	stby
CMD2, card wins bus	-	ident	-	-	-	-	-	-	-	-	-	stby
CMD2, card loses bus	-	ready	-	-	-	-	-	-	-	-	-	stby
CMD3	-	-	stby	-	-	-	-	-	-	-	-	stby
CMD4	-	-	-	stby	-	-	-	-	-	-	-	stby
CMD6	-	-	-	-	prg	-	-	-	-	-	-	stby
CMD7 card is addressed	-	-	-	tran	-	-	-	-	-	prg	-	stby
CMD7 card is not addressed	-	-	-	-	stby	stby	-	-	dis	-	-	stby
CMD8	-	-	-	-	data	-	-	-	-	-	-	stby

Table 23 — Card state transition table (continued)

	Current State											
	idle	**ready**	**ident**	**stby**	**tran**	**data**	**btst**	**rcv**	**prg**	**dis**	**ina**	**irq**
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	-	-	stby
CMD25	-	-	-	-	rcv	-	-	-	rcv	-	-	stby
CMD26	-	-	-	-	rcv	-	-	-	-	-	-	stby
CMD27	-	-	-	-	rcv	-	-	-	-	-	-	stby
Class 6												
CMD28	-	-	-	-	prg	-	-	-	-	-	-	stby
CMD29	-	-	-	-	prg	-	-	-	-	-	-	stby
CMD30	-	-	-	-	data	-	-	-	-	-	-	stby
Class 5												
CMD35	-	-	-	-	tran	-	-	-	-	-	-	stby
CMD36	-	-	-	-	tran	-	-	-	-	-	-	stby
CMD38	-	-	-	-	prg	-	-	-	-	-	-	stby
Class 7												
CMD16	see class 2											
CMD42	-	-	-	-	rcv	-	-	-	-	-	-	stby
Class 8												

Table 23 — Card state transition table (continued)

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

7.10 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 ’CRC7’, on page 110 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 bits. 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 7.11 on page 53.

Table 24 — R1 bits

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

- **R1b** is identical to R1 with an optional busy signal transmitted on the data line DAT0. The card may become busy after receiving these commands based on its state prior to the command reception. Refer to Section 7.13.3 on page 61 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 25 — R2 bits

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 26 — R3 bits

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 27 — R4 bits

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	register addr. [15:8]	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 28 — R5 bits

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

7.11 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 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 30 on page 56 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 29 — Response and execution mode conditions

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	

Table 29 — Response and execution mode conditions (continued)

Bits	Identifier	Type	Det Mode	Value	Description	Clear Cond
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
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

Table 29 — Response and execution mode conditions (continued)

Bits	Identifier	Type	Det Mode	Value	Description	Clear Cond
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 = Read 2 = Ident 3 = Stby 4 = Tran 5 = Data 6 = Rcv 7 = Prg 8 = Dis 9 = Btst 10–15 = 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.	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 29 — Response and execution mode conditions (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 30 — 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		
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		

Table 30 — 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
26							S		R	R		R	X			X		R	S	S		
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.

7.12 Memory Array Partitioning

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

For block oriented commands, the following definition is used:

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

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

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

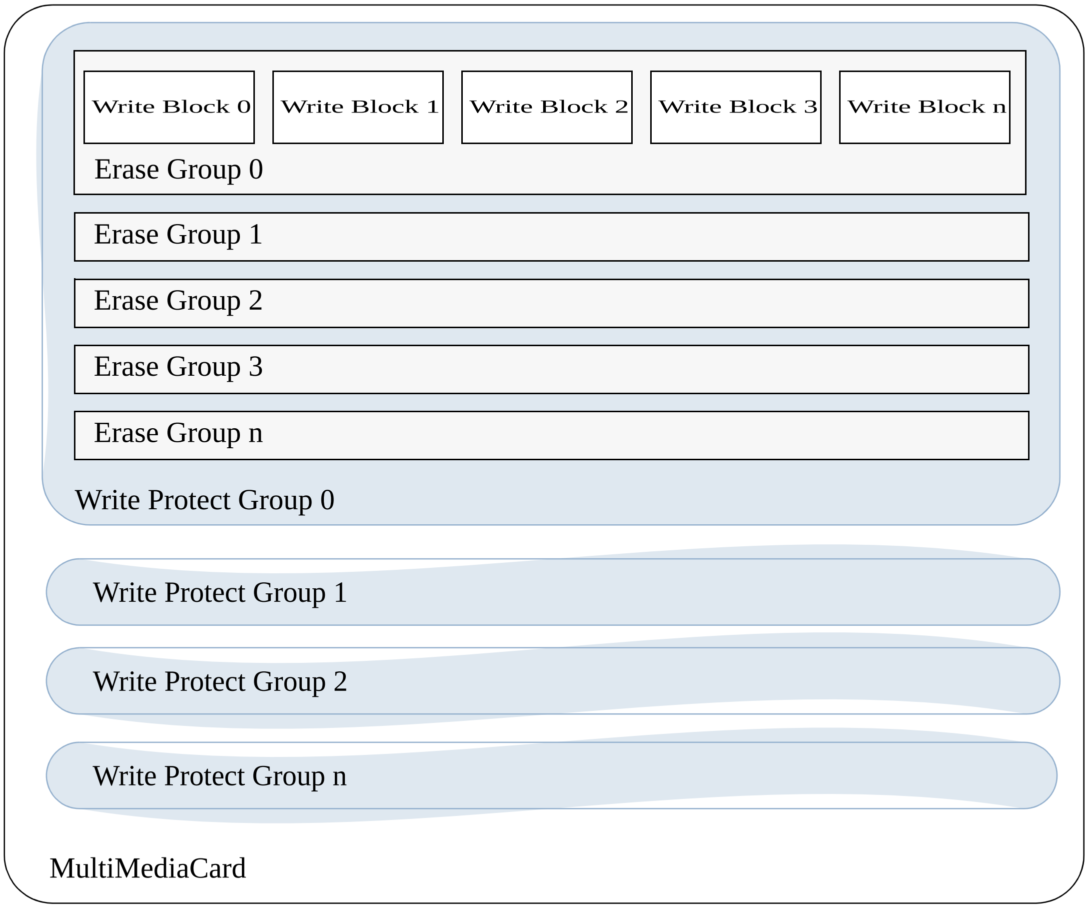


Figure 19 — Memory Array Partitioning

7.13 Timings

All timing diagrams use the following schematics and abbreviations. :

Table 31 — Timing diagram abbreviations

Abbreviation	Definition
S	Start bit (= ‘0’)
T	Transmitter bit (Host = ‘1’, Card = ‘0’)
P	One-cycle pull-up (= ‘1’)
E	End bit (=’1’)
Z	High impedance state (-> = ‘1’)
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 32 on page 64.

7.13.1 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.

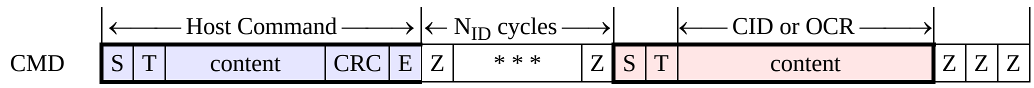


Figure 20 — Identification timing (card identification mode)

- **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.

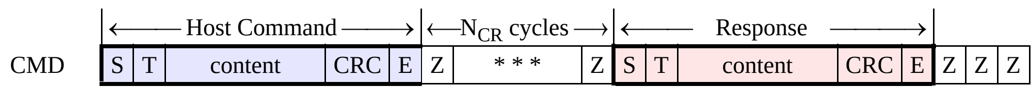


Figure 21 — SET_RCA timing (card identification mode)

- **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:

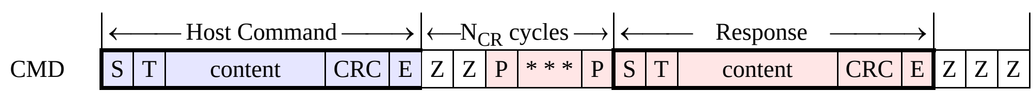


Figure 22 — Command response timing (data transfer mode)

- **R1b Responses**

Some commands, like CMD6, may assert the BUSY signal after responding 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 value is not relevant.

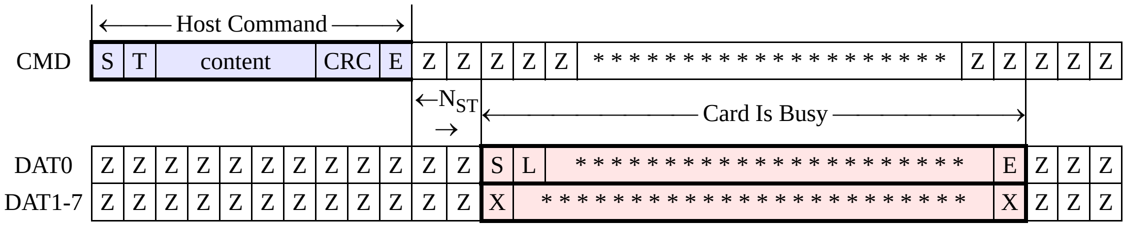


Figure 23 — R1b response timing

- **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.

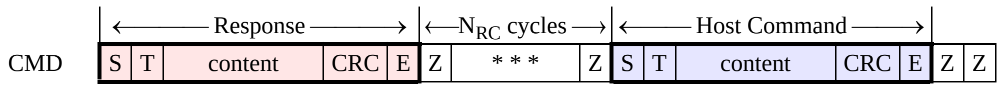


Figure 24 — Timing response end to next command start (data transfer mode)

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

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

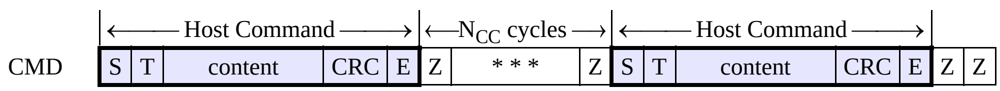


Figure 25 — Timing of command sequences (all modes)

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.

7.13.2 Data Read

- **Single Block Read**

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

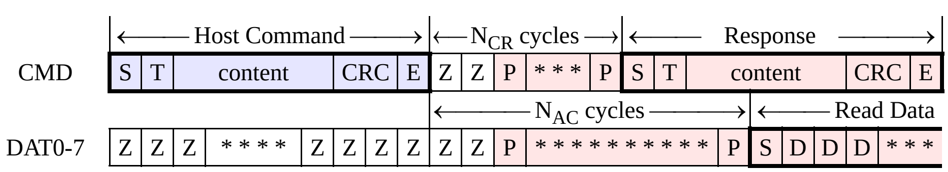


Figure 26 — Single block read timing

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

- **Multiple Block Read**

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

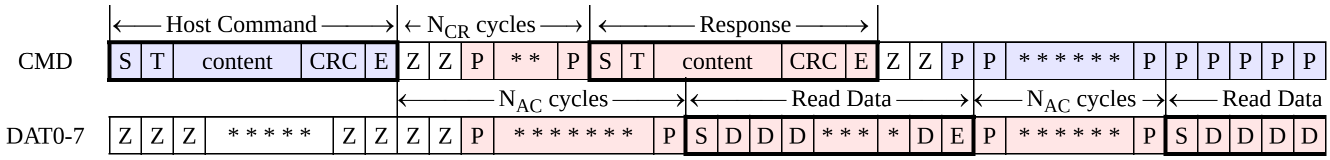


Figure 27 — Multiple block read timing

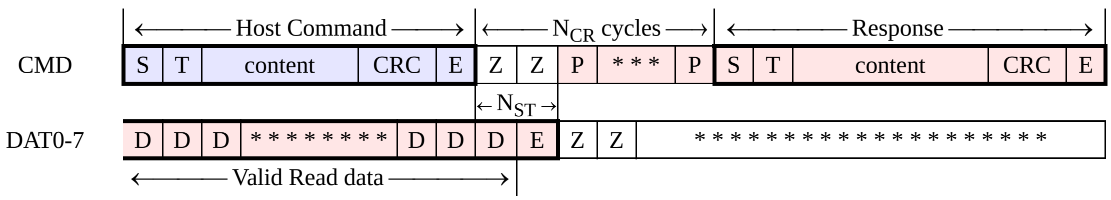


Figure 28 — Stop command timing (CMD12, data transfer mode)

- **Stream Read**

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

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

The data is suffixed with CRC check bits to allow the card to check it for transmission errors. The card sends back the CRC check result as a CRC status token on 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.

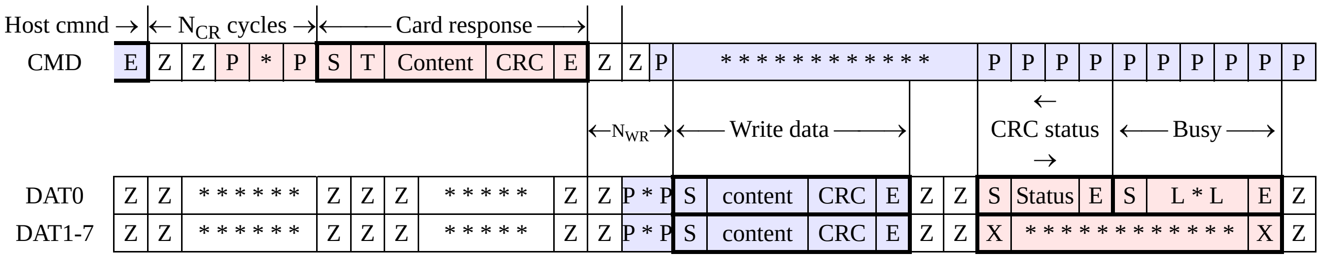


Figure 29 — Block write command timing

If the card does not have a free data receive buffer, the card indicates this condition by pulling down the data line DAT0 to LOW. The card stops pulling down DAT0 as soon as at least one receive buffer for the defined data transfer block length becomes free. This signalling does not give any information about the data write status which must be polled by the host.

- **Multiple Block Write**

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

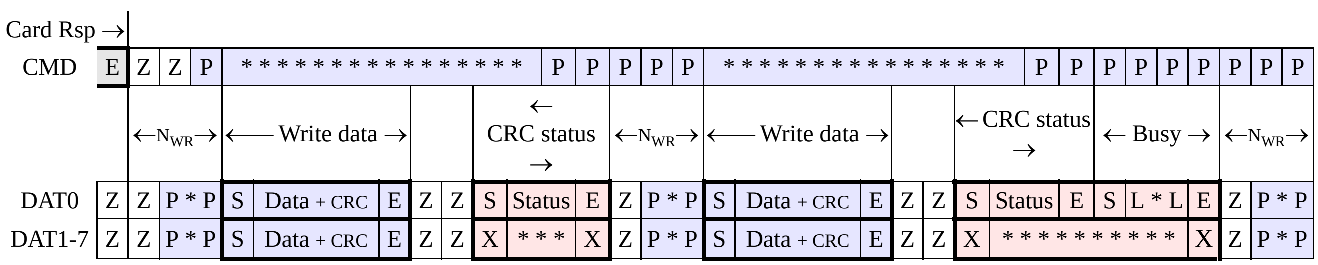


Figure 30 — Multiple block write timing

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

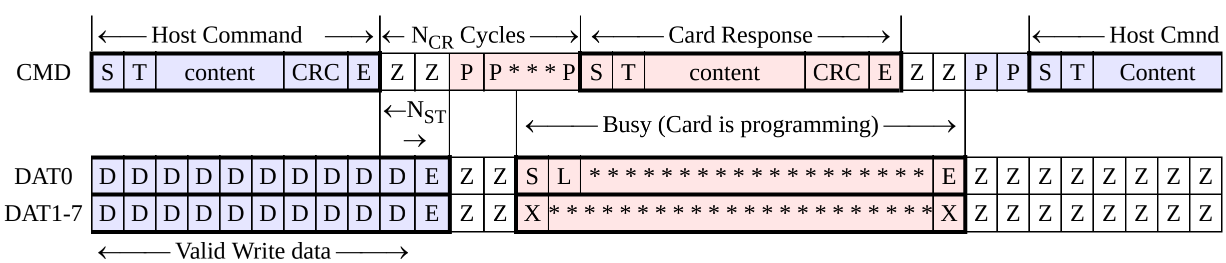
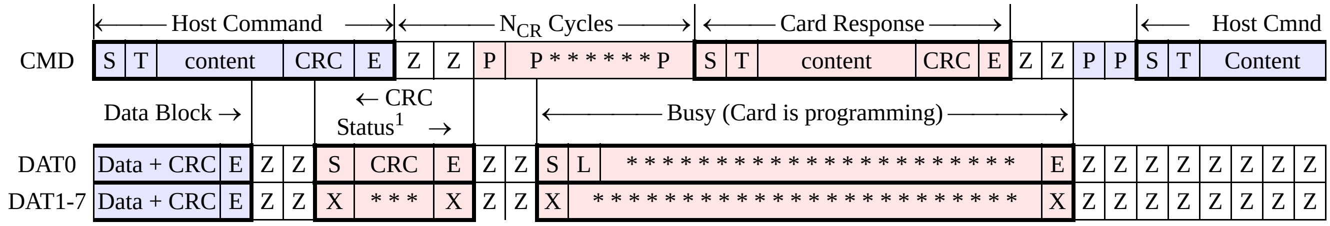


Figure 31 — Stop transmission during data transfer from the host

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



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

Figure 32 — Stop transmission during CRC status transfer from the card

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

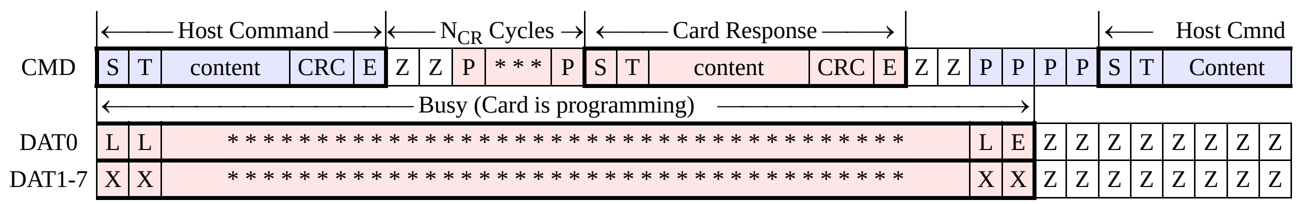


Figure 33 — Stop transmission after last data block. Card is busy programming.

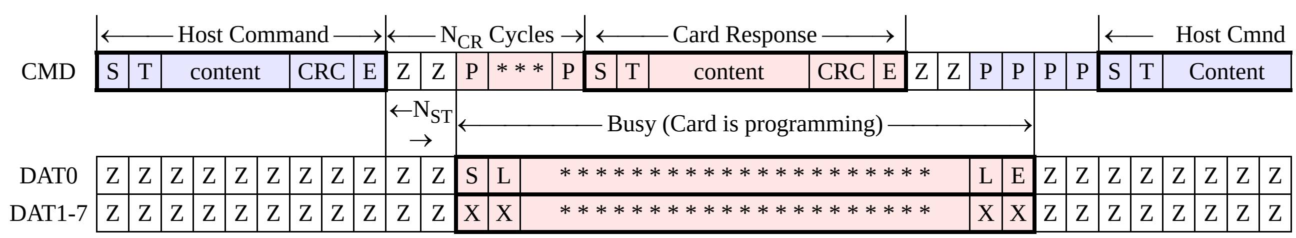


Figure 34 — Stop transmission after last data block. Card becomes busy.

- **Stream Write**

The data transfer starts N_{WR} clock cycles after the card response to the sequential write command was received. The bus transaction is identical to that of a write block command. (See Figure 29 on page 62.) As the data transfer is not block oriented, the data stream does not include the CRC checksum. Consequently the host can not receive any CRC status information from the card. The data stream is terminated by a stop command. The bus transaction is identical to the write block option when a data block is interrupted by the stop command. (See Figure 31 on page 62.)

- **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 34 on page 63.

- **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 34 on page 63.

7.13.4 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.

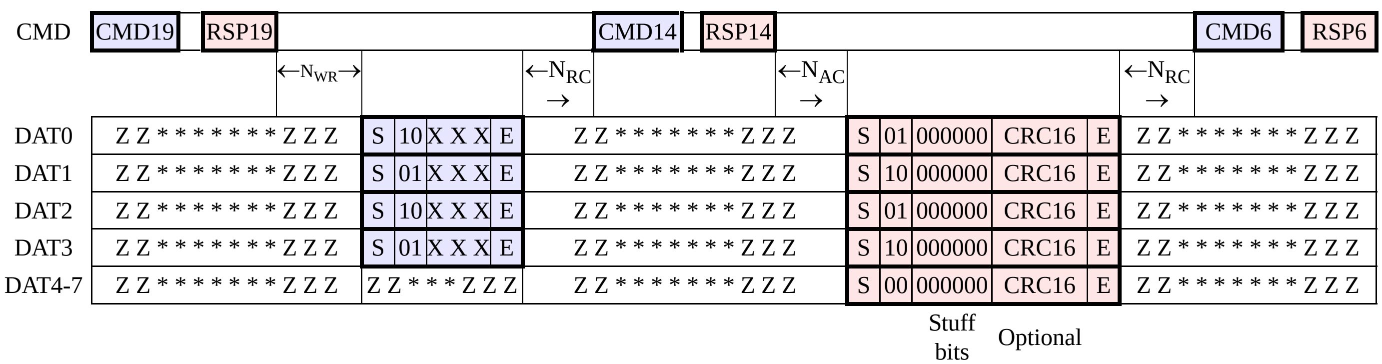


Figure 35 — 4-bit system bus-testing procedure

7.13.5 Timing Values.

Table 32 — Timing parameters

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

Table 32 — Timing parameters

Symbol	Min	Max	Unit
N_{ST}	2	2	clock cycles

NOTE 1 F_{OP} is the MMC clock frequency the host is using for the read operation.
Following is a calculation example:
CSD value for TAAC is 0x26; this is equal to 1.5mSec;
CSD value for NSAC is 0;
The host frequency F_{OP} is 10MHz
$N_{AC} = 10 \times (1.5 \times 10^{-3} \times 10 \times 10^{6} + 0) = 150,000$ 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 7.8 on page 42). 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. 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 33 — OCR register

OCR bit	VDD voltage window	High voltage MultimediaCard	Dual voltage MultiMediaCard
[6:0]	Reserved	000 0000b	00 00000b
[7]	1.65–1.95	0b	1b
[14:8]	2.0–2.6	000 0000b	000 0000b
[23:15]	2.7–3.6	1 1111 1111b	1 1111 1111b
[30:24]	Reserved	000 0000b	000 0000b
[31]	card power up status bit (busy)[1]		

NOTE 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 33, for High Voltage and Dual voltage MultiMediaCards. As long as the card is busy, the corresponding bit (31) is set to LOW, the 'wired-and' operation, described in Section 7.2.2 on page 22 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.

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

Table 34 — CID register

Name	Field	Width	CID-slice
Manufacturer ID	MID	8	[127:120]
OEM/Application ID	OID	16	[119:104]
Product name	PNM	48	[103:56]

Table 34 — CID register (continued)

Name	Field	Width	CID-slice
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. This procedure is established to ensure uniqueness of the CID register.

- **OID**

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

- **PNM**

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

- **PRV**

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

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

- **PSN**

A 32 bits unsigned binary integer.

- **MDT**

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

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

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

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

- **CRC**

CRC7 checksum (7 bits). This is the checksum of the CID contents computed according to 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 entries in the table below are coded as follows: R = readable, W = writable once, E = erasable (multiple writable): .

Table 35 — CSD register

Name	Field	Width	Cell Type	CSD-slice
CSD structure	CSD_STRUCTURE	2	R	[127:126]
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]
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]

Table 35 — CSD register (continued)

Name	Field	Width	Cell Type	CSD-slice
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 36 — CSD structure

CSD_STRUCTURE	CSD structure version	Valid for 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
3	Version is coded in the CSD_STRUCTURE byte in the EXT_CSD register	

- **SPEC_VERS**

Defines the MultiMediaCard Specification version supported by the card.

Table 37 — Specification version

SPEC_VERS	Specification Version Number
0	Allocated by MMCA
1	Allocated by MMCA
2	Allocated by MMCA
3	Allocated by MMCA
4	Version 4.1
5–15	Reserved

- **TAAC**

Defines the asynchronous part of the data access time.

Table 38 — TAAC bit codes

TAAC bit position	Code
2:0	Time unit 0=1ns, 1=10ns, 2=100ns, 3=1µs, 4=10µs, 5=100µs, 6=1ms, 7=10ms

Table 38 — TAAC bit codes (continued)

TAAC bit position	Code
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 32 on page 64 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, and higher, of the specification, the value shall be 20MHz (0x2A):

Table 39 — Transfer speed

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 40.

Table 40 — Card command class support

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 1, 2,4...2048 bytes. (See Section 7.11 on page 53 for details.):

Table 41 — Read block length

READ_BL_LEN	Block length	Remarks
0	2^0 = 1 Byte	
1	2^1 = 2 Bytes	
…		
11	2^{11} = 2048 Bytes	
12–15	Reserved	

- **READ_BL_PARTIAL**

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

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

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

- **WRITE_BLK_MISALIGN**

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

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

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

- **READ_BLK_MISALIGN**

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

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

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

- **DSR_IMP**

Defines if the configurable driver stage is integrated on the card. If set, a driver stage register (DSR) must be implemented also. (See Section 8.6 on page 82.).

Table 42 — Driver stage register implementation

DSR_IMP	DSR type
0	DSR is not implemented
1	DSR implemented

- **C_SIZE**

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

memory capacity = BLOCKNR * BLOCK_LEN

where:

BLOCKNR = (C_SIZE+1) * MULT

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

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

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

- **VDD_R_CURR_MIN, VDD_W_CURR_MIN**

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

Table 43 — Minimal power supply V_{DD} maximum values

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 44 — Maximal power supply V_{DD} maximum values

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**

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 45 — Device size multipliers

C_SIZE_MULT	**MULT**	**Remarks**
0	$2^2 = 4$	
1	$2^3 = 8$	
2	$2^4 = 16$	
3	$2^5 = 32$	
4	$2^6 = 64$	

Table 45 — Device size multipliers (continued)

C_SIZE_MULT	MULT	Remarks
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 46 — R2W factor field format

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

- **WRITE_BL_LEN**

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

- **WRITE_BL_PARTIAL**

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

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

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

- **FILE_FORMAT_GRP**

Indicates the selected group of file formats. This field is read-only for ROM. The usage of this field is shown in Table 47. (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 47 — Card 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.

- **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 48 — Error correction code field formats

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 109. 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 49 — CSD command classes

	Command classes									
CSD Field	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
CSD_STRUCTURE	+	+	+	+	+	+	+	+	+	+
SPEC_VERS	+	+	+	+	+	+	+	+	+	+
TAAC		+	+	+	+	+	+	+	+	
NSAC		+	+	+	+	+	+	+	+	
TRAN_SPEED		+	+	+	+					
CCC	+	+	+	+	+	+	+	+	+	+
READ_BL_LEN			+							
READ_BL_PARTIAL			+							
WRITE_BLK_MISALIGN					+					
READ_BLK_MISALIGN			+							
DSR_IMP	+	+	+	+	+	+	+	+	+	+
C_SIZE_MANT		+	+	+	+	+	+	+	+	
C_SIZE_EXP		+	+	+	+	+	+	+	+	
VDD_R_CURR_MIN		+	+							
VDD_R_CURR_MAX		+	+							
VDD_W_CURR_MIN				+	+	+	+	+	+	
VDD_W_CURR_MAX				+	+	+	+	+	+	
ERASE_GRP_SIZE						+	+	+	+	
WP_GRP_SIZE							+	+	+	
WP_GRP_ENABLE							+	+	+	
DEFAULT_ECC		+	+	+	+	+	+	+	+	

Table 49 — CSD command classes (continued)

	Command classes									
CSD Field	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
R2W_FACTOR				+	+	+	+	+	+	
WRITE_BL_LEN				+	+	+	+	+	+	
WRITE_BL_PARTIAL				+	+	+	+	+	+	
FILE_FORMAT_GRP										
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 50 — Extended CSD register

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]		298		[503:211
Minimum Write Performance for 8bit @ 52 MHz	MIN_PERF_W_8_52	1	R	[210]
Minimum Read Performance for 8bit @ 52 MHz	MIN_PERF_R_8_52	1	R	[209]
Minimum Write Performance for 8bit @ 26 MHz / 4bit @52MHz	MIN_PERF_W_8_26_4_52	1	R	[208]
Minimum Read Performance for 8bit @ 26 MHz / 4bit @52MHz	MIN_PERF_R_8_26_4_52	1	R	[207]
Minimum Write Performance for 4bit @ 26 MHz	MIN_PERF_W_4_26	1	R	[206]
Minimum Read Performance for 4bit @ 26 MHz	MIN_PERF_R_4_26	1	R	[205]
Reserved[1]		1		[204]
Power Class for 26MHz @ 3.6V	PWR_CL_26_360	1	R	[203]
Power Class for 52MHz @ 3.6V	PWR_CL_52_360	1	R	[202]
Power Class for 26MHz @ 1.95V	PWR_CL_26_195	1	R	[201]
Power Class for 52MHz @ 1.95V	PWR_CL_52_195	1	R	[200]

Table 50 — Extended CSD register (continued)

Name	Field	Size (Bytes)	Cell Type	CSD-slice
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]		183		[182:0]

NOTE 1 Reserved bits should read as ‘0.’

- **S_CMD_SET**

This field defines which command sets are supported by the card.

Table 51 — Card-supported command sets

Bit	Command Set
7-3	Reserved
2	Allocated by MMCA
1	Allocated by MMCA
0	Standard MMC

- **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 52 — R/W access performance values

Value	Performance
0x00	For cards not reaching the 2.4MB/s minimum value
0x08	Class A: 2.4MB/s and is the lowest allowed value for MMCplus and MMCmobile(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 (26 MHz 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 53 — 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	
	6-15			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. The only currently valid values for this field are 0x01 and 0x03.

Table 54 — Card types

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

- **CSD_STRUCTURE**

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

Table 55 — CSD structures

CSD_STRUCTURE	CSD structure version	Valid for 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
3	Reserved for future use	
4–255	Reserved for future use	

- **EXT_CSD_REV**

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

Table 56 — Extended CSD register revision parameters

EXT_CSD_REV	Extended CSD Revision
255–2	Reserved
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. It is set to ‘0’ (Standard MMC) after power up and can be changed by a SWITCH command.

- **CMD_SET_REV**

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

Table 57 — Command set revision

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 58. 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 58 — Power class

Bits	Description
[7:4]	Reserved
[3:0]	Card power class code (See Table 53 on page 80)

- **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 (Section 12.7.1 on page 123)

- **BUS_WIDTH**

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

Table 59 — Bus Mode Value

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

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 118. 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

9.1 Introduction

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

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

9.2 SPI Interface Concept

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

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

CS: Host to card Chip Select signal.

CLK: Host to card clock signal

DataIn: Host to card data signal.

DataOut: Card to host data signal.

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

9.3 SPI Bus Topology

The card identification and addressing methods are replaced by a hardware Chip Select (CS) signal. There are no broadcast commands. For every command, a card (slave) is selected by asserting (active low) the CS signal (see Figure 36).

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

The bidirectional CMD and DAT lines are replaced by unidirectional *dataIn* and *dataOut* signals.

The MultiMediaCard pin assignment in SPI mode (compared to MultiMediaCard mode) is given in Table 60 on page 84.

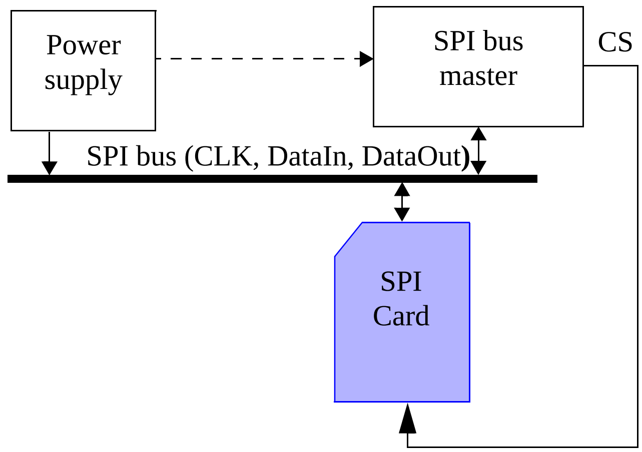


Figure 36 — MultiMediaCard bus system

Table 60 — SPI interface pin configuration

Pin #	MultiMediaCard Mode			SPI Mode		
	Name	**Type[1]**	**Description**	**Name**	**Type**	**Description**
1	DAT3	I/O/PP	Data	CS	I	Chip Select (neg true)
2	CMD	I/O/PP/OD	Command/Response	DI	I/PP	Data In
3	V_{SS1}	S	Supply voltage ground	VSS	S	Supply voltage ground
4	V_{DD}	S	Supply voltage	VDD	S	Supply voltage
5	CLK	I	Clock	SCLK	I	Clock
6	V_{SS2}	S	Supply voltage ground	VSS2	S	Supply voltage ground
7	DAT0	I/O/PP	Data	DO	O/PP	Data Out
8	DAT1	I/O/PP	Data	Not used		
9	DAT2	I/O/PP	Data	Not used		
10	DAT4	I/O/PP	Data	Not used		
11	DAT5	I/O/PP	Data	Not used		
12	DAT6	I/O/PP	Data	Not used		
13	DAT7	I/O/PP	Data	Not used		

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

9.4 MultiMediaCard Registers in SPI Mode

The register usage in SPI mode is summarized in Table 61 (refer to Section 6 for more information). Most of them are inaccessible.

Table 61 — MultiMediaCard registers in SPI mode

Name	Available in SPI mode	Width [Bytes]	Description
CID	Yes	16	Card identification data (serial number, manufacturer ID, etc.)
RCA	No		
DSR	No		
CSD	Yes	16	Card-specific data, information about the card operation conditions.
EXT_CSD	Yes	512	Extended Card-specific data, information about the card supported properties and configured modes
OCR	Yes	32	Operation condition register.

9.5 SPI Bus Protocol

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

Similar to the MultiMediaCard protocol, the SPI messages consist of command, response and data-block tokens (see Section 6 for a detailed description). All communication between host and card is controlled by the host (master). The host starts every bus transaction by asserting the CS signal low.

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

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

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

9.5.1 Mode Selection

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

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

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

9.5.2 Bus Transfer Protection

Every MultiMediaCard token transferred on the bus is protected by CRC bits. In SPI mode, the MultiMediaCard offers a non-protected mode which enables systems built with reliable data links to exclude the hardware or firmware required for implementing the CRC generation and verification functions.

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

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

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

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

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

9.5.3 Data Read

The SPI mode supports single and multiple block read operations. The main difference between SPI and MultiMediaCard modes is that the data and the response are both transmitted to the host on the DataOut signal (refer to Figure 37 and Figure 38). Therefore the card response to the STOP_COMMAND may cut-short and replace the last data block.

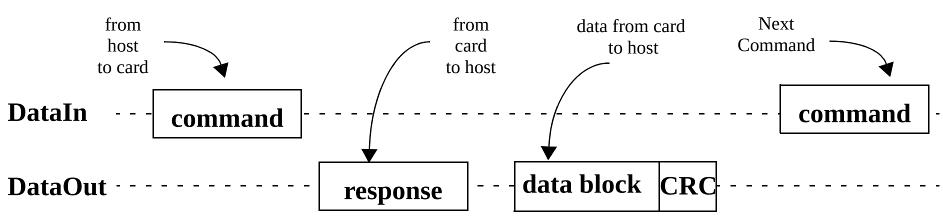


Figure 37 — SPI single block read operation

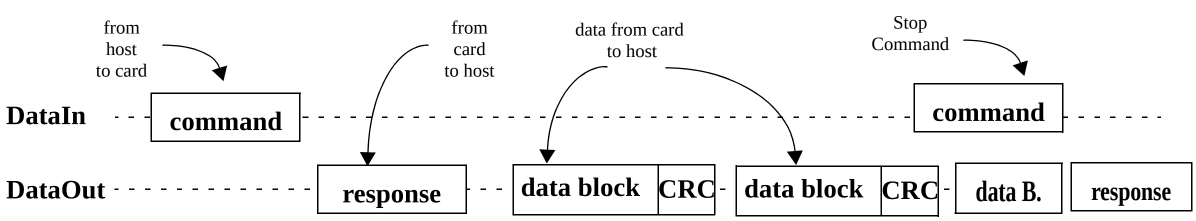


Figure 38 — SPI multiple block read operation

The basic unit of data transfer is a block whose maximum size is defined in the CSD (READ_BL_LEN).If READ_BL_PARTIAL is set, smaller blocks whose starting and ending address are entirely contained within one physical block (as defined by READ_BL_LEN) may also be transmitted. A CRC is appended to the end of each block ensuring data transfer integrity. CMD17 (READ_SINGLE_BLOCK) initiates a single block read. CMD18 (READ_MULTIPLE_BLOCK) starts a transfer of several consecutive blocks. Two types of multiple block read transactions are defined (the host can use either one at any time):

- Open-ended Multiple block read

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

- Multiple block read with pre-defined block count

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

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

If the host provides an out of range address as an argument to either CMD17 or CMD18, or the currently defined block length is illegal for a read operation, the card will reject the command and respond with the ADDRESS_OUT_OF_RANGE or BLOCK_LEN_ERROR bit set, respectively.

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).

In case of a data retrieval error (e.g. out of range, address misalignment, internal error, etc.) detected during data transfer, the card will not transmit any data. Instead (as opposed to MultiMediaCard mode where the card times out), a special data error token will be sent to the host. Figure 39 shows a single block read operation which terminates with an error token rather than a data block.

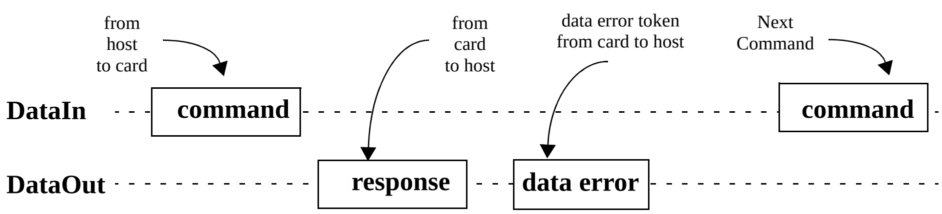


Figure 39 — SPI read operation—data error

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

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

If the host uses partial blocks whose accumulated length is not block aligned, and block misalignment is not allowed, the card shall detect a block misalignment error condition 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.

9.5.4 Data Write

The SPI mode supports single block and Multiple block write commands. Upon reception of a valid write command (CMD24 or CMD25), the card will respond with a response token and will wait for a data block to be sent from the host. CRC suffix, block length and start address restrictions are (with the exception of the CSD parameter WRITE_BL_PARTIAL controlling the partial block write option) identical to the read operation (see Figure 40). If a CRC error is detected it will be reported in the data-response token and the data block will not be programmed.

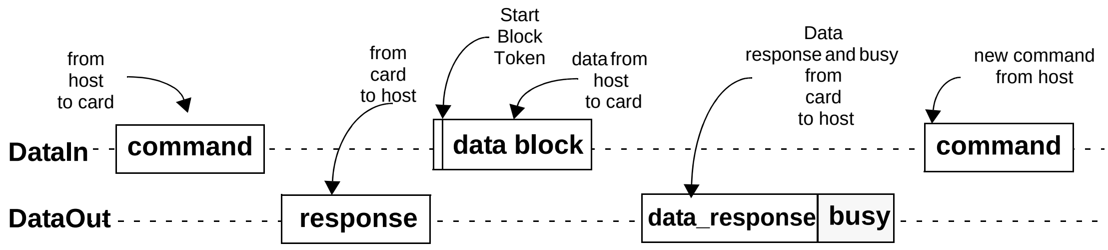


Figure 40 — SPI single block write operation

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

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

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

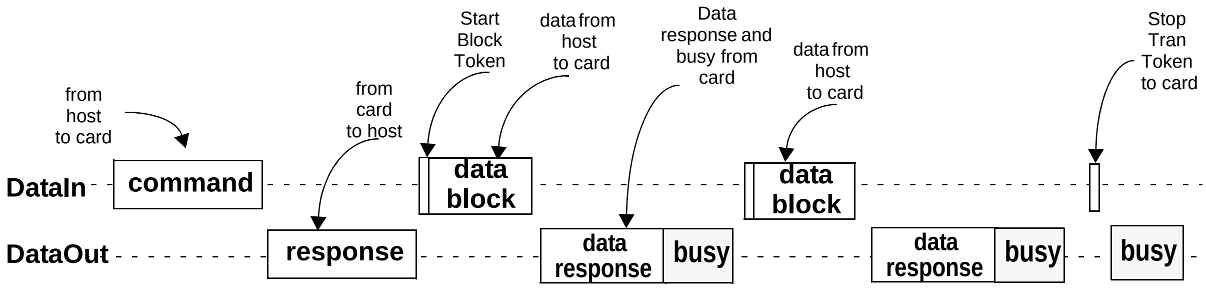


Figure 41 — SPI multiple block write operation

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

- Open-ended Multiple block write

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

- Multiple block write with pre-defined block count

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

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

If the host provides an out of range address as an argument to either CMD17 or CMD18, or the currently defined block length is illegal for a read operation, the card will reject the command, remain in Tran state and respond with the ADDRESS_OUT_OF_RANGE or BLOCK_LEN_ERROR bit set, respectively.

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).

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

If the host uses partial blocks whose accumulated length is not block aligned, and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card shall detect the block misalignment error during the reception of the first misaligned block, abort the write operation, and ignore all further incoming data. The host must abort the operation by sending the 'Stop Tran' token, to which the card will respond with the ADDRESS_MISALIGN bit set.

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

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

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

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

9.5.5 Erase & Write Protect Management

The erase and write protect management procedures in the SPI mode are identical to those of the MultiMediaCard mode. While the card is erasing or changing the write protection bits of the predefined erase groups list, it will be in a busy state and hold the DataOut line low. Figure 42 illustrates a 'no data' bus transaction with and without busy signalling.

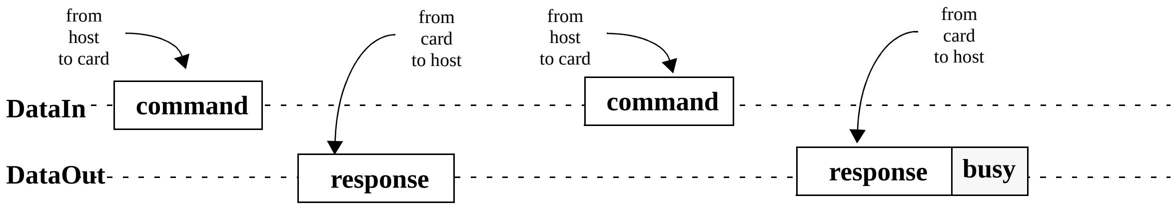


Figure 42 — SPI “no data” operations

9.5.6 Read CID/CSD Registers

Unlike the MultiMediaCard protocol (where the register contents is sent as a command response), reading the contents of the CSD and CID registers in SPI mode is a simple read-block transaction. The card will respond with a standard response token (see Figure 39) followed by a data block of 16 bytes suffixed with a 16 bit CRC.

The data time out for the CSD command cannot be set to the card TAAC since this value is stored in the CSD. Refer to Section 9.8.2 on page 105 for detailed timing. For consistency, read CID transaction is identical to read CSD

9.5.7 Reset Sequence

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

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

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

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

9.5.8 Clock Control

The SPI bus clock signal can be used by the SPI 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 change the clock frequency or shut it down.

There are a few restrictions the SPI host must follow:

- The bus frequency can be changed at any time (under the restrictions of maximum data transfer frequency, defined by the MultiMediaCards)
- It is an obvious requirement that the clock must be running for the MultiMediaCard to output data or response tokens. After the last SPI bus transaction, the host is required, to provide 8 (eight) clock cycles for the card to complete the operation before shutting down the clock. throughout this 8 clocks period

the state of the CS signal is irrelevant. it can be asserted or de-asserted.
Following is a list of the various SPI bus transactions:

- A command / response sequence. 8 clocks after the card response end bit. The CS signal can be asserted or de-asserted during these 8 clocks.
- A read data transaction. 8 clocks after the end bit of the last data block.
- A write data transaction. 8 clocks after the CRC status token.
- The host is allowed to shut down the clock of a "busy" card. The MultiMediaCard will complete the programming operation regardless of the host clock. However, the host must provide a clock edge for the card to turn off its busy signal. Without a clock edge the MultiMediaCard (unless previously disconnected by de-asserting the CS signal) will force the dataOut line down, permanently.

9.5.9 Error Conditions

- **CRC and Illegal Command**

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

There are different kinds of illegal commands:

- Commands which belong to classes not supported by the MultiMediaCard (e.g. interrupt and I/O commands).
- Commands not allowed in SPI mode (e.g. CMD20 - write stream)
- Commands which are not defined (e.g. CMD47).

- **Read, Write, Erase And Force Erase Time-out Conditions**

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

The typical access and program times are defined as follows:

- Read

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

- Write

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

- Erase

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

- Force Erase

  The Force Erase time-out is specified in Section 7.6.2 on page 40

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

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

9.5.10 Memory Array Partitioning

Same as for MultiMediaCard mode.

9.5.11 Card Lock/unlock

Usage of card lock and unlock commands in SPI mode is identical to MultiMediaCard mode. In both cases, the command response is of type R1b. After the busy signal clears, the host should obtain the result of the operation by issuing a GET_STATUS command. Please refer to Section 7.4.10 on page 36 for details.

9.5.12 Application Specific commands

Identical to MultiMediaCard mode with the exception of the APP_CMD status bit (refer to Section 7.4.11 on page 38), which is not available in SPI.

9.6 SPI Mode Transaction Packets

SPI mode transaction packets can be described by one of the following tokens:

- **Command tokens**: various formats and classes that support a set of card functions
- **Response tokens**: signals acknowledging the commands sent
- **Data tokens**: representing data transmission
- **Data error tokens**: identifying data read failure
- **Clearing Status bits**: a SPI mode status returned to the host

9.6.1 Command Tokens

- **Command Format**

All the MultiMediaCard commands are 6 bytes long. The command transmission always starts with the left bit of the bitstring corresponding to the command codeword. All commands are protected by a CRC (see Section 10.2 on page 109). The commands and arguments are listed in Table 63.

Table 62 — SPI mode command formats

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

- **Command Classes**

As in MultiMediaCard mode, the SPI commands are divided into several classes (See Table 66 on page 100). Each class supports a set of card functions. A MultiMediaCard will support the same set of optional command classes in both communication modes (there is only one command class table in the CSD register). The available command classes, and the supported command for a specific class, however, are different in the MultiMediaCard and the SPI communication mode.

Table 63 — SPI mode command classes

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

- **Detailed Command Description**

The following table provides a detailed description of the SPI mode commands. The responses are defined in Section 9.6.2 on page 96. Table 63 lists all MultiMediaCard commands. A “yes” in the SPI mode column indicates that the command is supported in SPI mode. With these restrictions, the command class description in the CSD is still valid. If a command does not require an argument, the value of this field should be set to zero. The reserved commands are also reserved in MultiMediaCard mode.

The card can be switched to a new command space, using the SWITCH command, just as in MultiMediaCard mode; with the only limitation that in SPI mode the bus is always one bit wide.

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

Table 64 — SPI mode commands

CMD INDEX	SPI Mode	Argument	Resp	Abbreviation	Command Description
CMD0	Yes	None	R1	GO_IDLE_STATE	Resets the MultiMediaCard
CMD1	Yes	None	R1	SEND_OP_COND	Activates the card's initialization process
CMD2	No				
CMD3	No				
CMD4	No				
CMD5	reserved				
CMD6	Yes	[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 and modifies the EXT_CSD registers. Access modes are: 00 Command Set 01 Set bits 10 Clear bits 11 Write Byte
CMD7	No				
CMD8	Yes	[31:0] stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data.
CMD9	Yes	None	R1	SEND_CSD	Asks the selected card to send its card-specific data (CSD)
CMD10	Yes	None	R1	SEND_CID	Asks the selected card to send its card identification (CID)
CMD11	No				
CMD12	Yes	None	R1	STOP_TRANSMISSI ON	Stop transmission on multiple block read
CMD13	Yes	None	R2	SEND_STATUS	Asks the selected card to send its status register
CMD14	This command is not applicable in SPI mode and the card should regard it as illegal command				
CMD15	No				
CMD16	Yes	[31:0] block length	R1	SET_BLOCKLEN	selects a block length (in bytes) for all following block commands (read and write).[1]
CMD17	Yes	[31:0] data address	R1	READ_ SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.[2]
CMD18	Yes	[31:0] data address	R1	READ_ MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command or the requested number of data blocks transmitted
CMD19	This command is not applicable in SPI mode and the card should regard it as illegal command				
CMD20	No				
CMD21 … CMD22	reserved				

Table 64 — SPI mode commands (continued)

CMD INDEX	SPI Mode	Argument	Resp	Abbreviation	Command Description
CMD23	Yes	[31:16] set to 0 [15:0] number of blocks	R1	SET_ BLOCK_COUNT	Defines the number of blocks which are going to be transferred in the immediately exceeding multiple block read or write command. If the argument is all 0s, then the subsequent read/write operation will be open-ended.
CMD24	Yes	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.[3]
CMD25	Yes	[31:0] data address	R1	WRITE_ MULTIPLE_BLOCK	Continuously writes blocks of data until a “Stop Tran’ Token or the requested number of blocks received.
CMD26	No				
CMD27	Yes	None	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD
CMD28	Yes	[31:0] data address	R1b[4]	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29	Yes	[31:0] data address	R1b	CLR_WRITE_PROT	If the card has write protection features, this command clears the write protection bit of the addressed group
CMD30	Yes	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card has write protection features, this command asks the card to send the status of the write protection bits[5]
CMD31	reserved				
CMD32 … CMD34	Reserved. These command indexes cannot be used in order to maintain backwards compatibility with older versions of the MultiMediaCards				
CMD35	Yes	[31:0] data address	R1	ERASE_GROUP_ START	Sets the address of the first erase group within a range to be selected for erase
CMD36	Yes	[31:0] data address	R1	ERASE_GROUP_ END	Sets the address of the last erase group within a continuous range to be selected for erase
CMD37	Reserved. This command index cannot be used in order to maintain backwards compatibility with older versions of the MultiMediaCards				
CMD38	Yes	[31:0] stuff bits	R1b	ERASE	Erases all previously selected erase groups
CMD39	No				
CMD40	No				
CMD41	reserved				

Table 64 — SPI mode commands (continued)

CMD INDEX	SPI Mode	Argument	Resp	Abbreviation	Command Description
CMD42	Yes	[31:0] stuff bits.	R1b	LOCK_UNLOCK	Used to Set/Reset the Password or lock/unlock the card. The structure of the data block is described in Section 7.4.10 on page 36. The size of the Data Block is defined by the SET_BLOCK_LEN command.
CMD43 … CMD54	reserved				
CMD55	Yes	[31:0] stuff bits	R1	APP_CMD	Defines to the card that the next command is an application specific command rather than a standard command
CMD56	Yes	[31:1] stuff bits. [0]: RD/WR_ [6]	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block is defined by the SET_BLOCK_LEN command.
CMD57	Reserved				
CMD58	Yes	None	R3	READ_OCR	Reads the OCR register of a card
CMD59	Yes	[31:1] stuff bits [0:0] CRC option	R1	CRC_ON_OFF	Turns the CRC option on or off. A “1” in the CRC option bit will turn the option on, a “0” will turn it off
CMD60 … CMD63	No				

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

9.6.2 Responses

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

- **Format R1**

This response token is sent by the card after every command, with the exception of SEND_STATUS commands. It is one byte long, and the MSB is always set to zero. The other bits are error indications, an error being signaled by a ‘1’. The structure of the R1 format is given in Figure 43. The meaning of the flags is defined as follows:

- **In idle state**: The card is in idle state and running the initializing process.
- **Erase Reset:** An erase sequence was cleared before executing because 'non erase' command (neither of CMD35, CMD36, CMD38 or CMD13) was received.

- **Illegal Command**: An illegal command code was detected or the card did not switch to the requested mode.
- **Communication CRC Error**: The CRC check of the last command failed.
- **Erase Sequence Error**: An error occurred in the sequence of erase commands (CMD35, CMD36, CMD38).
- **Address Misaligned**: A misaligned block is detected during data transfer.
- **Address Out Of Range | Block Length Error**: The command's argument was out of the allowed range for this card.

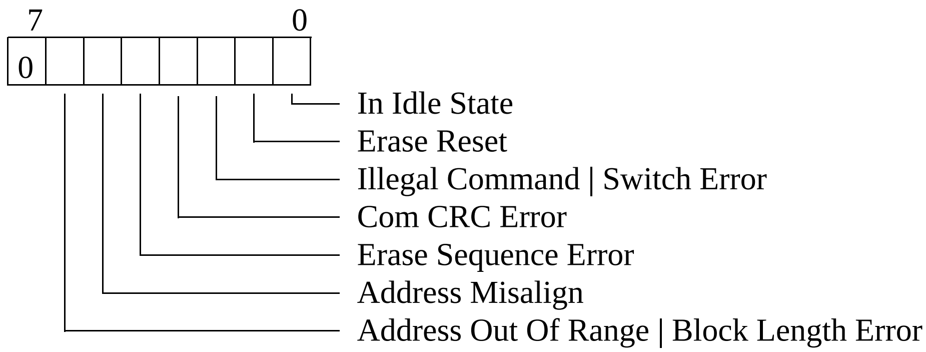


Figure 43 — R1 response format

- **Format R1b**

This response token is identical to the R1 format with the addition of an immediately following busy signal.

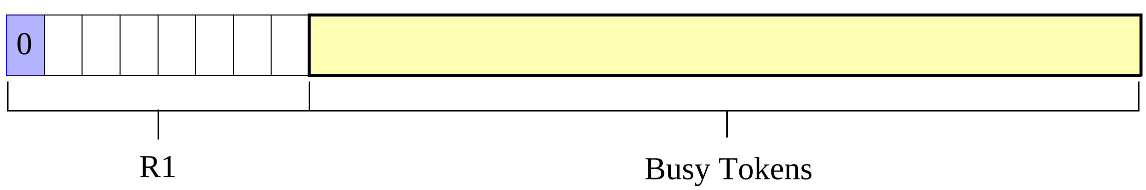


Figure 44 — R1b response format

- **Busy**

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

- **Format R2**

This response token is two bytes long and sent as a response to the SEND_STATUS command. The format is given in Figure 45.

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

- **CSD Overwrite:** This status bit is set if the host is trying to change the ROM section, or reverse the copy bit (set as original) or the permanent WP bit (un-protect) of the CSD register.

- **Erase Param**: An invalid selection of erase groups, for erase.
- **Write Protect Violation**: The command tried to write a write-protected block.
- **Card ECC Failed**: Card internal ECC was applied but failed to correct the data.
- **Card Error**: Generic internal card error, unrelated to the host activities and undefined by the standard.
- **Execution Error**: Generic internal card error, occurred during (and related to) execution of the last host command and undefined by the standard.
- **Write Protect Erase Skip | Lock/Unlock Command Failed**: This status bit has two functions. It is set when the host attempts to erase a write-protected block or if a sequence or password error occurred during a card lock/unlock operation.
- **Card Is Locked**: Set when the card is locked by the user. Reset when it is unlocked.

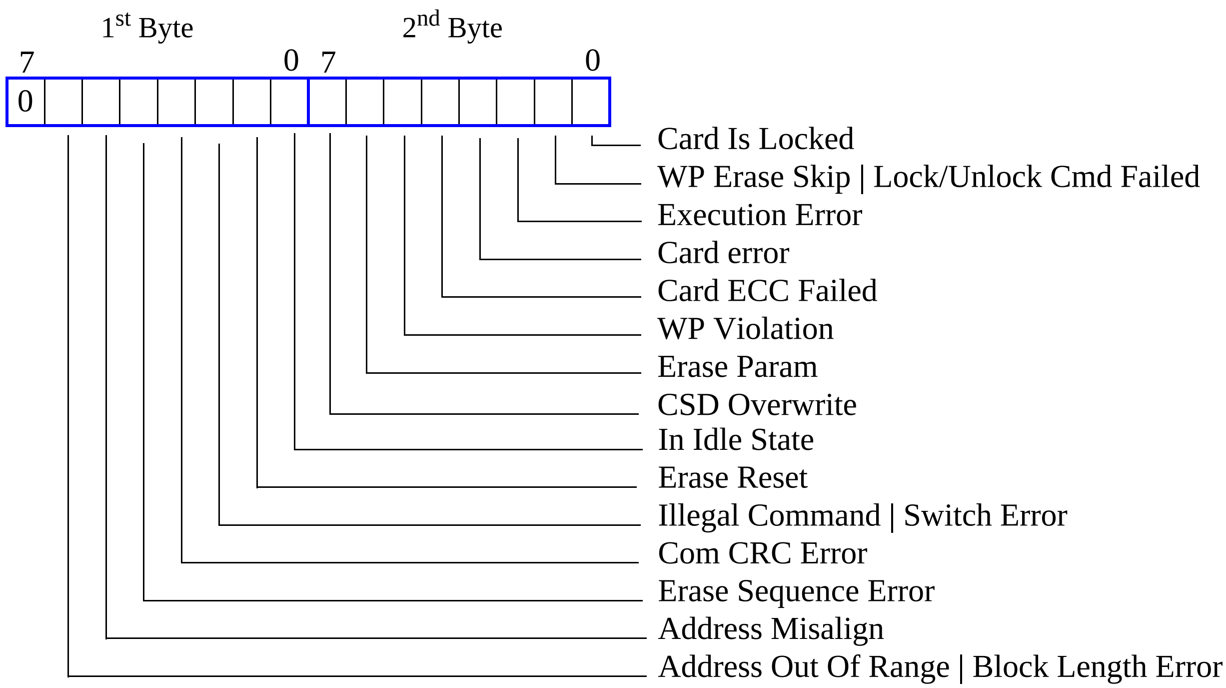


Figure 45 — R2 response format

- **Format R3**

This response token is sent by the card when a READ_OCR command is received. The response length is 5 bytes (see Figure 46). The structure of the first (MSB) byte is identical to response type R1. The other four bytes contain the OCR register.

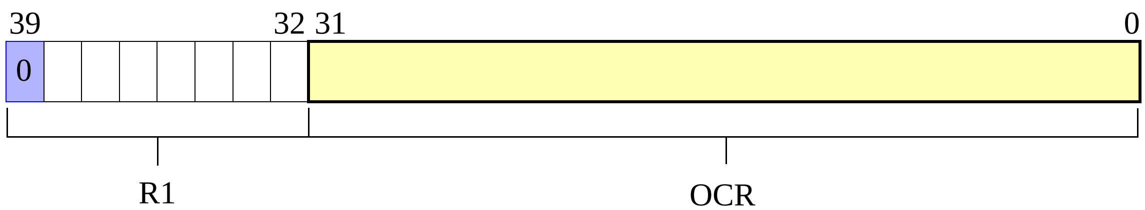


Figure 46 — R3 response format

- **Data Response**

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

7	6				0							
x	x	x	0	Status	1							

Figure 47 — Data response format

The meaning of the status bits is defined as follows:

‘010’ — Data accepted.

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

’110’ — Data Rejected due to a Write Error

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

9.6.3 Data Tokens

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

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

- First byte:

Table 65 — Start data block token format

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

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

9.6.4 Data Error Token

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

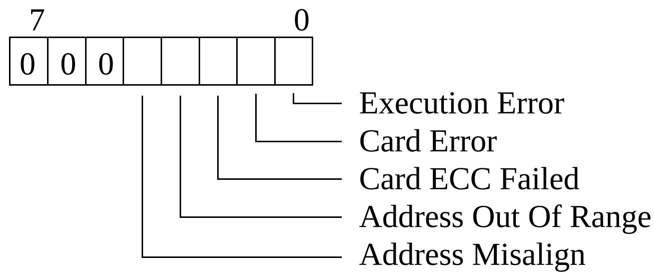


Figure 48 — Data error token

The 5 least significant bits (LSB) are the same error bits as in the response format R2.

9.6.5 Clearing Status Bits

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

All Error bits defined in MultiMediaCard mode, with the exception of underrun and overrun, have the same meaning and usage in SPI mode. There are some differences in the Status bits due to the different protocol (e.g. current state is not defined in SPI mode).

The detection mode and clear condition of Error and Status bits are identical to the MultiMediaCard mode, with one exception, Error bits are cleared when read by the host, regardless of the response format.

The following table describes the various status bits:.

Table 66 — SPI mode status bit descriptions

Identifier	Included in resp	Type	Det Mode	Value	Description
Address Out Of Range	R1 R2 DataErr	E	R	’0’= no error ’1’= error	The command’s address argument was out of the allowed range for this card.
			X		A multiple block read/write operation is attempting to read or write beyond the card capacity (Although it started in a valid address)
Address Mis-align	R1 R2 DataErr	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 mis-aligned to the card physical blocks.
			X		A multiple block read/write operation is attempting to read or write a data block, which is not aligned to the physical blocks of the card (Although it started with a valid address/block-length combi-nation)
Erase Sequence Error	R1 R2	E	R	’0’= no error ’1’= error	An error in the sequence of erase com-mands occurred.

Table 66 — SPI mode status bit descriptions (continued)

Identifier	Included in resp	Type	Det Mode	Value	Description
Erase Param	R2	E	X	’0’= no error ’1’= error	An invalid selection of erase groups, for erase, occurred.
Block Length Error	R1 R2	E	R	’0’= no error ’1’= error	Either the argument of a SET_BLOCKLEN command exceeds the maximum allowed value for the card, or the previously defined block length is illegal for the current command (e.g. the host is issues a write command and the current block length is smaller than the card maximum value and write partial blocks is not allowed)
WP violation	R2	E	X	’0’= not protected ’1’= protected	Attempt to program a write protected block.
Com CRC Error	R1 R2	E	R	’0’= no error ’1’= error	The CRC check of the received com-mand failed.
Illegal Command	R1 R2	E	R	’0’= no error ’1’= error	The received command is not legal for the card state.
Switch Error	R1 R2	E	X	’0’= no error ’1’= error	If set, the card did not switch to the expected mode as requested by the SWITCH command
Card ECC failed	R2 DataErr	E	X	’0’= success ’1’= failure	Card internal ECC was applied but failed to correct the data.
Card Error	R2 DataErr	E	R	’0’= no error ’1’= error	(Undefined by the standard) A card error occurred, which is not related to the host command.
Execution Error	R2 DataErr	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 fail-ures).
WP Erase Skip	R2	S	X	’0’= not protected ’1’= protected	Only partial address space was erased due to existing write protected blocks.
Lock/Unlock Cmd failed	R2	E	X	’0’= no error ’1’= error	Sequence or password error during card lock/unlock operation.
Card Is Locked	R2	S		‘0’ = card is not locked ‘1’ = card is locked	Card is locked by a user password
Erase Reset	R1 R2	E	R	’0’= cleared ’1’= set	An erase sequence was cleared before executing because an out of erase sequence command was received.(other than CMD35, CMD36, CMD38 or CMD13)
In Idle State	R1 R2	S		0 = Card is ready 1 = Card is in idle state	The card enters the idle state after power up or reset command. It will exit this state and become ready upon completion of its initialization procedures.

Table 66 — SPI mode status bit descriptions (continued)

Identifier	Included in resp	Type	Det Mode	Value	Description
CSD Overwrite	R2	E	X	’0’= no error ’1’= error	The host is trying to change the ROM section, or is trying to reverse the copy bit (set as original) or permanent WP bit (un-protect) of the CSD register.

Table 67 on page 103 defines, for each command number, the affected bits in either R1, R2 or Data Error token responses.

An “R” or “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 always valid and marked with “S.”

Table 67 — SPI mode data error response tokens

CMD #	**R2 Response Bit**																**Data Error Token Bit**							
	R1 Response Bit																							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0					R			S					R	X		S								
1					R	R		S						X										
6					R	R/X		S					R	X		S								
8					R	R		S					R	X		S								
9					R	R	R	S					R	X		S								
10					R	R	R	S					R	X		S								
12					R	R		S					R	X		S								
13					R	R		S					R	X		S								
16					R	R	R	S					R	X		S								
17		R	R		R	R	R	S				X	R	X		S				X	X	X	R	X
18		R	R		R	R	R	S				X	R	X		S				X	X	X	R	X
23					R	R	R	S					R	X		S								
24		R	R		R	R	R	S			X		R	X		S								
25		R	R		R	R	R	S			X		R	X		S								
27					R	R	R	S	X				R	X		S								
28		R			R	R	R	S					R	X		S								
29		R			R	R	R	S					R	X		S								
30		R			R	R	R	S					R	X		S								
35		R		R	R	R		S		X			R	X		S								
36		R		R	R	R		S		X			R	X		S								
38				R	R	R		S					R	X	X	S								
42					R	R	R	S					R	X	X	S								
55					R	R	R	S					R	X		S								
56					R	R	R	S					R	X		S								
58					R	R	R	S					R	X		S								
59					R	R	R	S					R	X		S								
Bit is valid for classes		1, 2, 3, 4, 5, 6	2, 4	5	Always	Always	Always	Always	Always	5	3, 4	1, 2	Always	Always	3, 4	Always				Always	Always	1, 2	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.

9.7 Card Registers

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

9.8 SPI Bus Timing Diagrams

All timing diagrams use the following schematics and abbreviations:

Table 68 — SPI bus timing abbreviations

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

All timing values are defined in Table 69. The host must keep the clock running for at least N_{CR} clock cycles after receiving the card response. This restriction applies to both command and data response tokens.

9.8.1 Command / Response

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

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

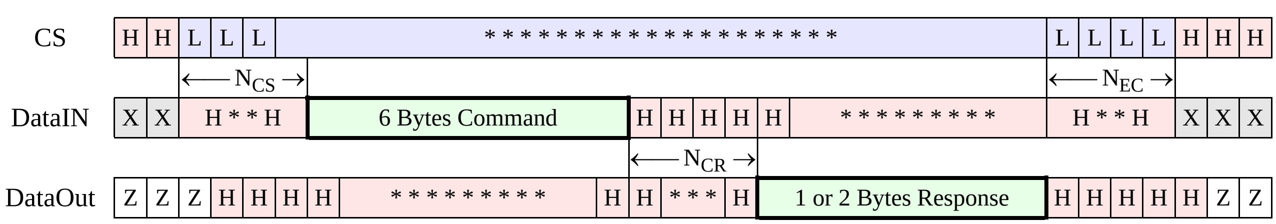


Figure 49 — SPI command/response transaction, card is ready

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

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

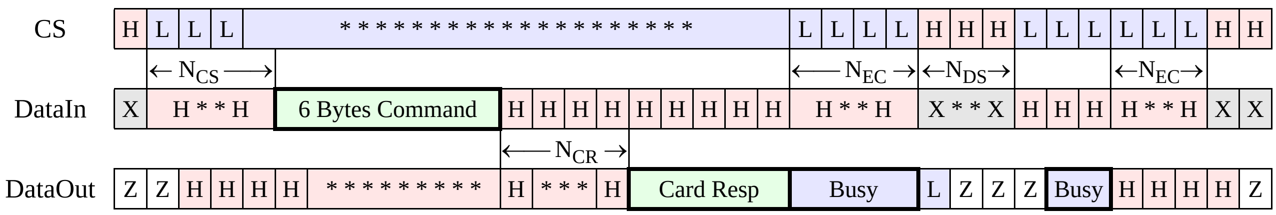


Figure 50 — SPI command/response transaction, card is busy

- **Card Response to Host Command**

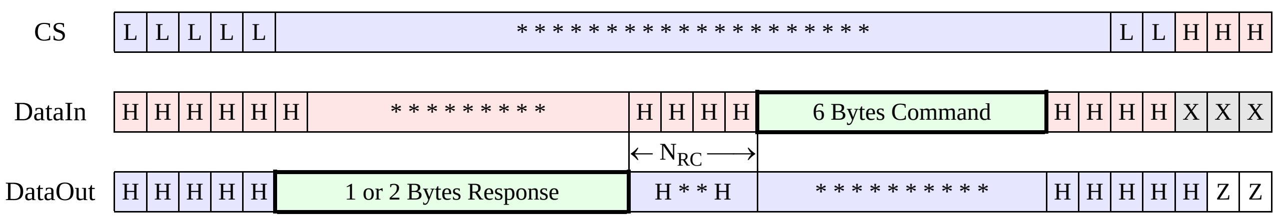


Figure 51 — SPI card response to the next host command

9.8.2 Data read

- **Single Block Read**

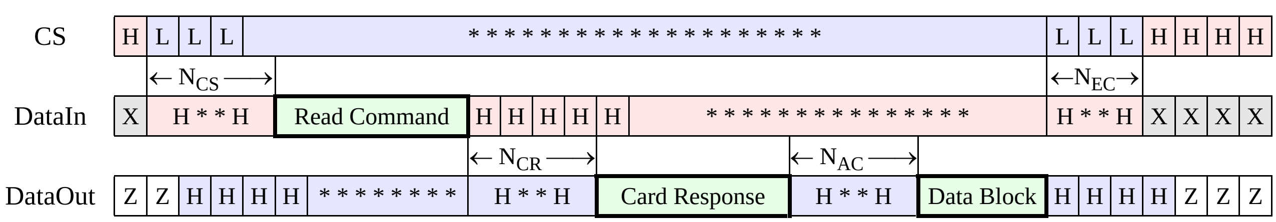


Figure 52 — SPI single block read

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

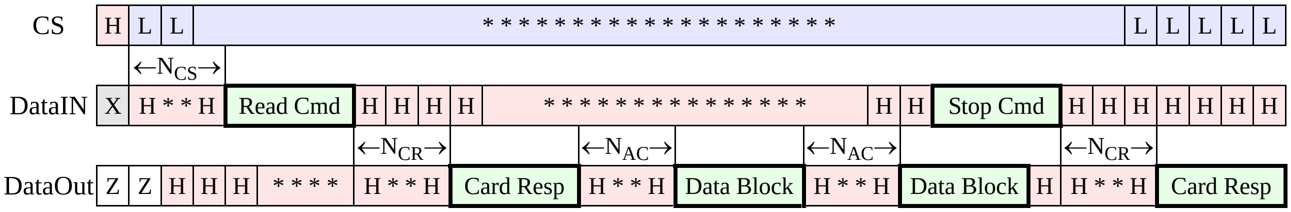


Figure 53 — SPI multiple block read, stop transmission does not overlap data

The timing for de-asserting the CS signal after the last card response is identical to a standard command/response transaction as described in Figure 49;

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

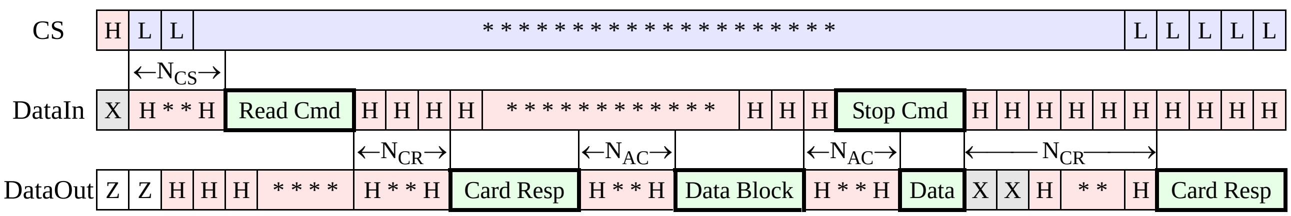


Figure 54 — SPI multiple block read, stop transmission overlaps data

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

The timing for de-asserting the CS signal after the last card response is identical to a standard command/response transaction as described in Figure 49.

- **Reading the CSD and CID registers**

The following timing diagram describes the SEND_CSD and SEND_CID commands bus transaction. The time-out values between the response and the data block is N_{CX} , and not N_{AC} , which is used for data read (since N_{AC} is still unknown at the time the CSD register is read). The SEND_CID transaction complies with the same timing diagram for consistency of the read register commands

.

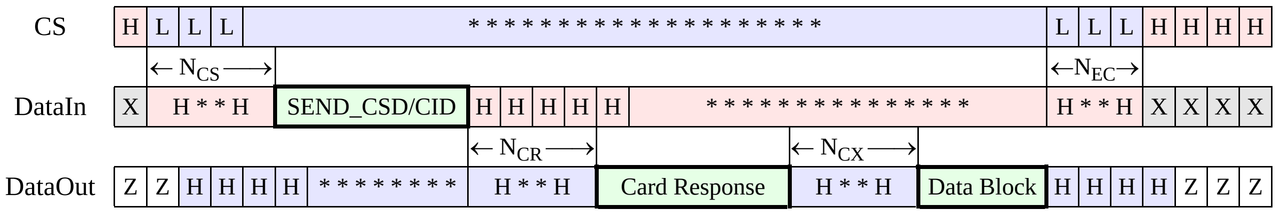


Figure 55 — SPI read CSD and CID registers

9.8.3 Data write

- **Single Block Write**

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

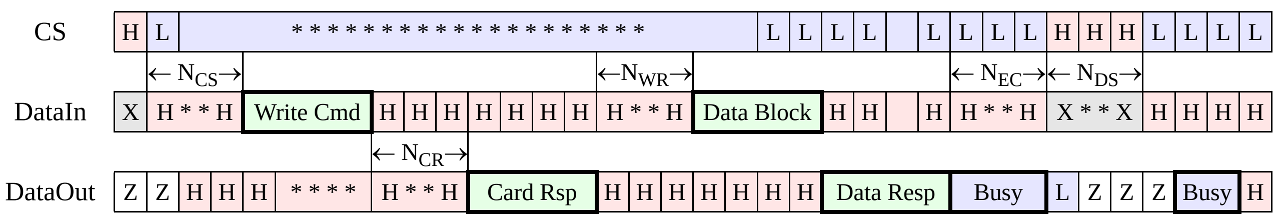


Figure 56 — SPI single block write

- **Multiple Block Write**

The timing behavior of the multiple block write transaction starting from the command up to the first data block is identical to the single block write. Figure 57 describes the timing between the data blocks of a multiple block write transaction. Timing of the 'Stop Tran' token is identical to a standard data block. After the "Stop Tran" token is received ny the card, the data on the DataOut line is undefined for one byte (N_{BR}), after which a Busy token may appear. The host may deselect and reselect the card during every busy period between the data blocks. Timing for toggling the CS signal is identical to the Single block write transaction.

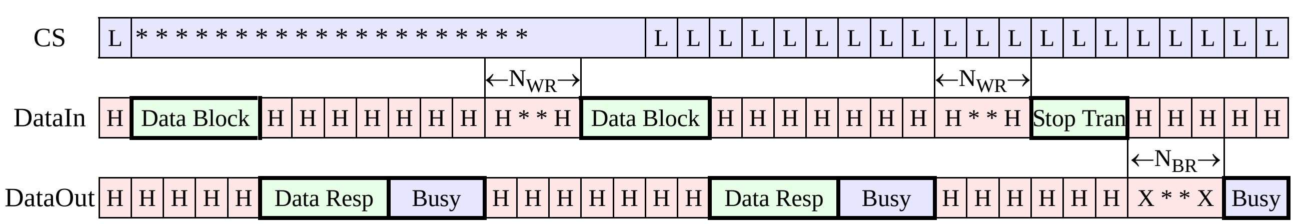


Figure 57 — SPI multiple block write

9.8.4 Timing Values

Table 69 — SPI mode timing

Symbol	Min	Max	Unit
N_{CS}	0	-	8 clock cycles
N_{CR}	1	8	8 clock cycles
N_{CX}	0	8	8 clock cycles
N_{RC}	1	-	8 clock cycles
N_{AC}	1	(10/8) * (TAAC * F_{OP} + 100 * NSAC)[1]	8 clock cycles
N_{WR}	1	-	8 clock cycles
N_{EC}	0	-	8 clock cycles
N_{DS}	0	-	8 clock cycles
N_{BR}	1	1	8 clock cycles
NOTE 1 F_{OP} is the MMC clock frequency the host is using for the read operation.			

9.9 SPI Electrical Interface

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

9.10 SPI Bus Operating Conditions

The bus operating conditions are identical to MultiMediaCard mode.

9.11 Bus Timing

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

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 70 — ECC parameters

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:

$$\text{Generator polynomial } G(x) = x^7 + x^3 + 1$$
$$M(x) = (\text{first bit}) \times x^n + (\text{second bit}) \times x^{n-1} + \ldots + (\text{last bit}) \times x^0$$
$$CRC[6\ldots0] = \text{Remainder}[(M(x) \cdot x^7)/G(x)]$$

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).

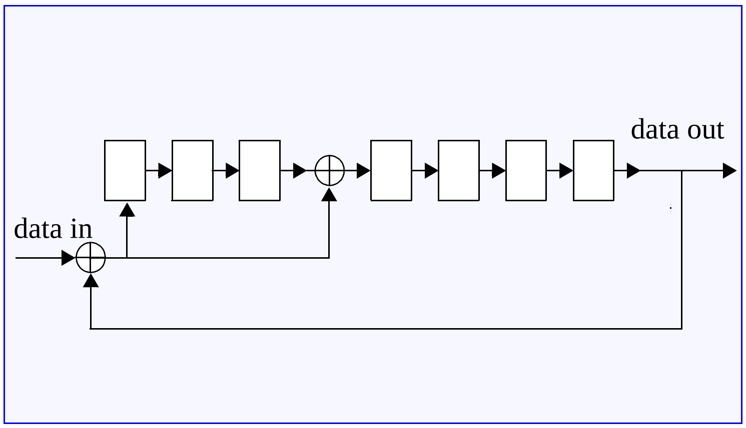


Figure 58 — CRC7 generator/checker

- **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:

$$\text{Generator polynomial } G(x) = x^{16} + x^{12} + x^5 + 1$$
$$M(x) = (\text{first bit}) \times x^n + (\text{second bit}) \times x^{n-1} + \ldots + (\text{last bit}) \times x^0$$
$$CRC[15\ldots0] = \text{Remainder}[(M(x) \cdot x^{16})/G(x)]$$

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.

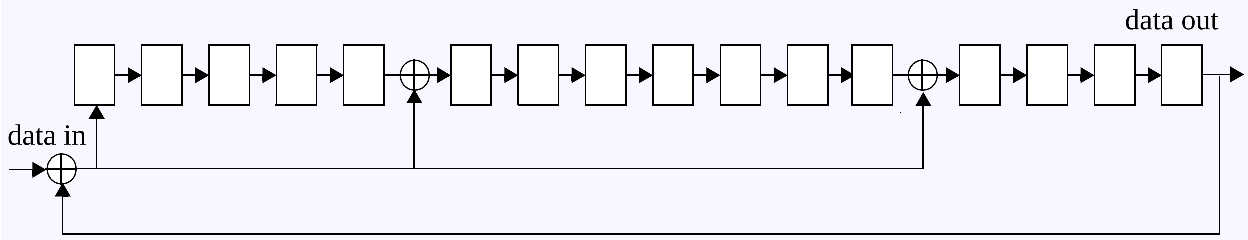


Figure 59 — CRC16 generator/checker

11 MultiMediaCard Mechanical Specification

See applicable JEDEC and MMCA Standards.

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.

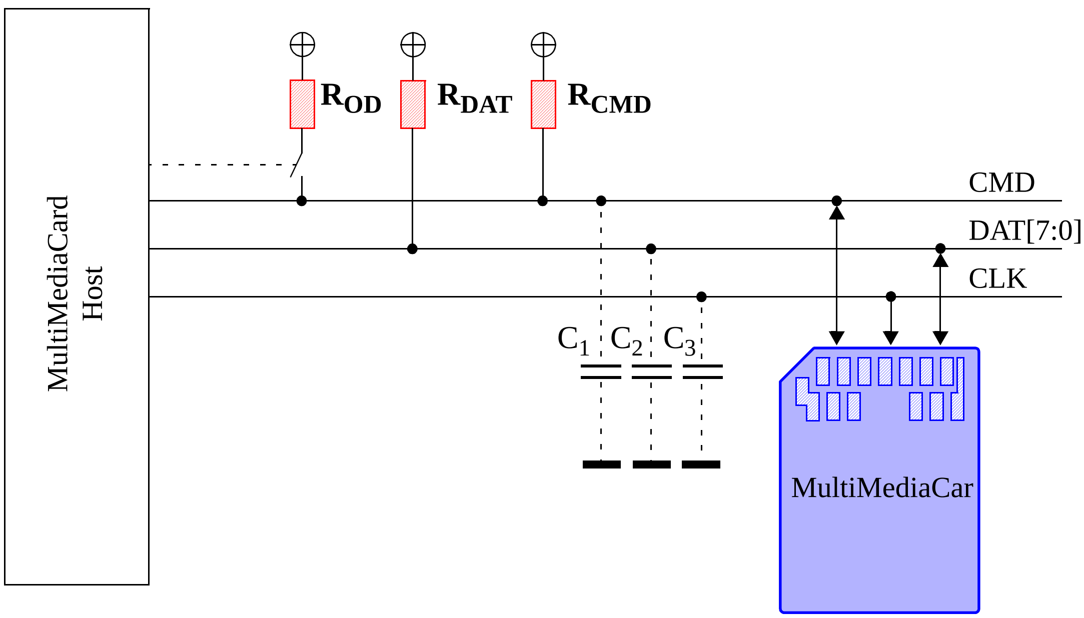


Figure 60 — Bus circuitry diagram

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 120). 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 120.

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).

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.

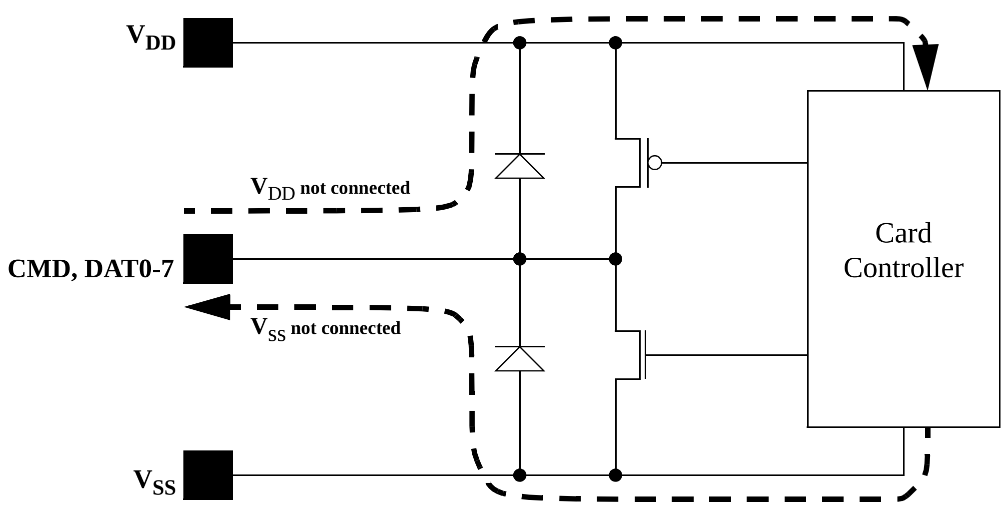


Figure 61 — Improper power supply

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

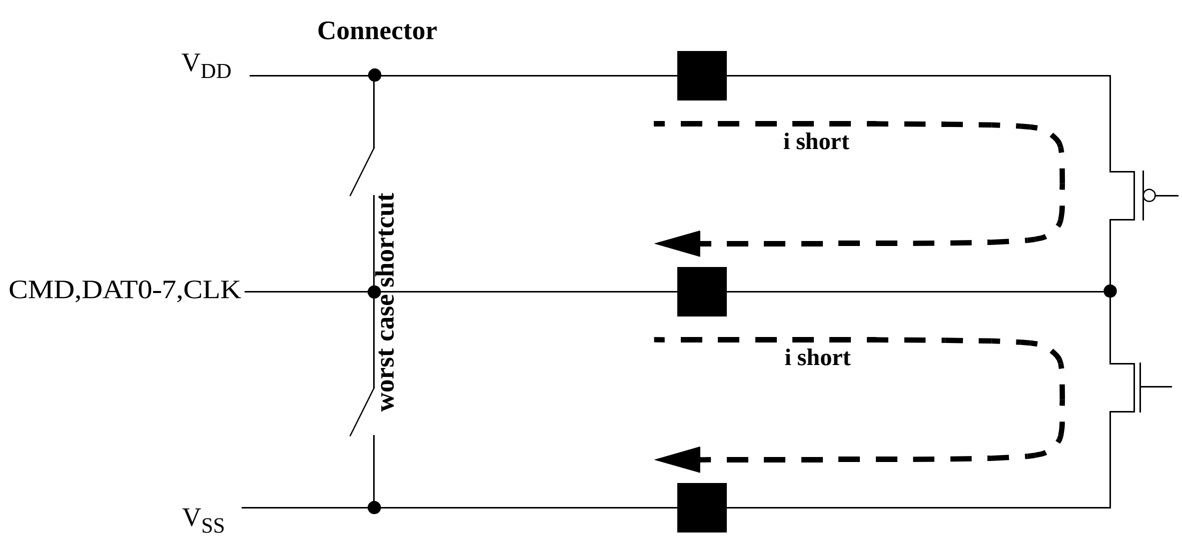


Figure 62 — Shortcut protection

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.

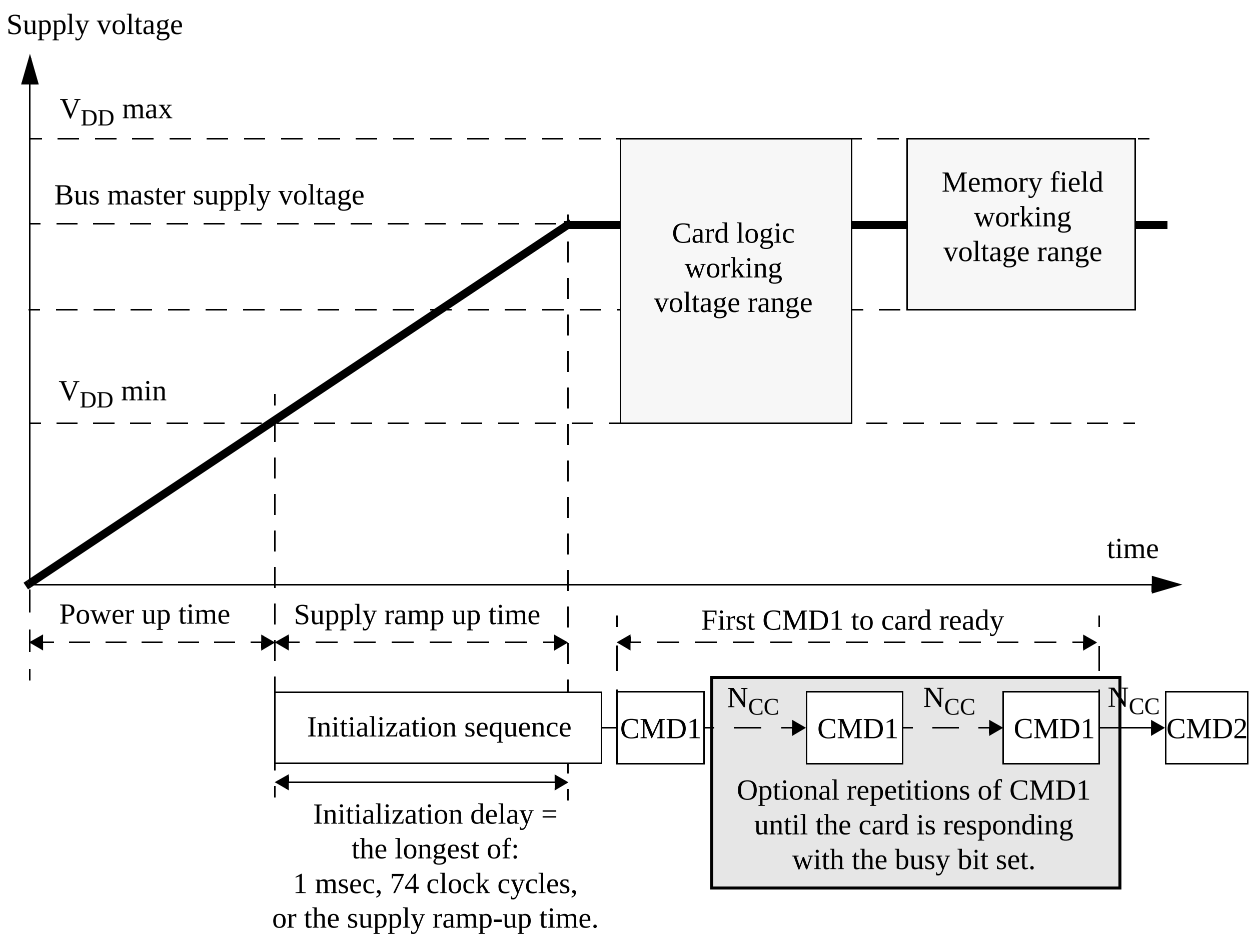


Figure 63 — Power-up diagram

- After power up (including hot insertion, i.e. inserting a card when the bus is operating) the card enters the *idle state*. During this state the card ignores all bus transactions until CMD1 is received.
- 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 µF in parallel with a minimum of 330 ohms. At no time during operation shall the card capacitance on the VDD line exceed 10 µF.
- 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.

- 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. This sequence is a contiguous stream of logical ‘1’s. The sequence length is the longest of: 1msec, 74 clocks or the supply-ramp-up-time; The additional 10 clocks (over the 64 clocks after what the card should be ready for communication) is provided to eliminate power-up synchronization problems.
- Every bus master has to implement CMD1. The CMD1 implementation is mandatory for all MultiMediaCards.

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.

In order to be able to operate the card at an optimal frequency over various buss topologies and number of cards, MultiMediaCards may optionally include programmable card output drivers for the push pull mode. Both the CMD and DAT bus drivers consist of a predriver stage and a complementary driver transistor (Figure 64).

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 71 defines the DSR register contents:

Table 71 — DSR content

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

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

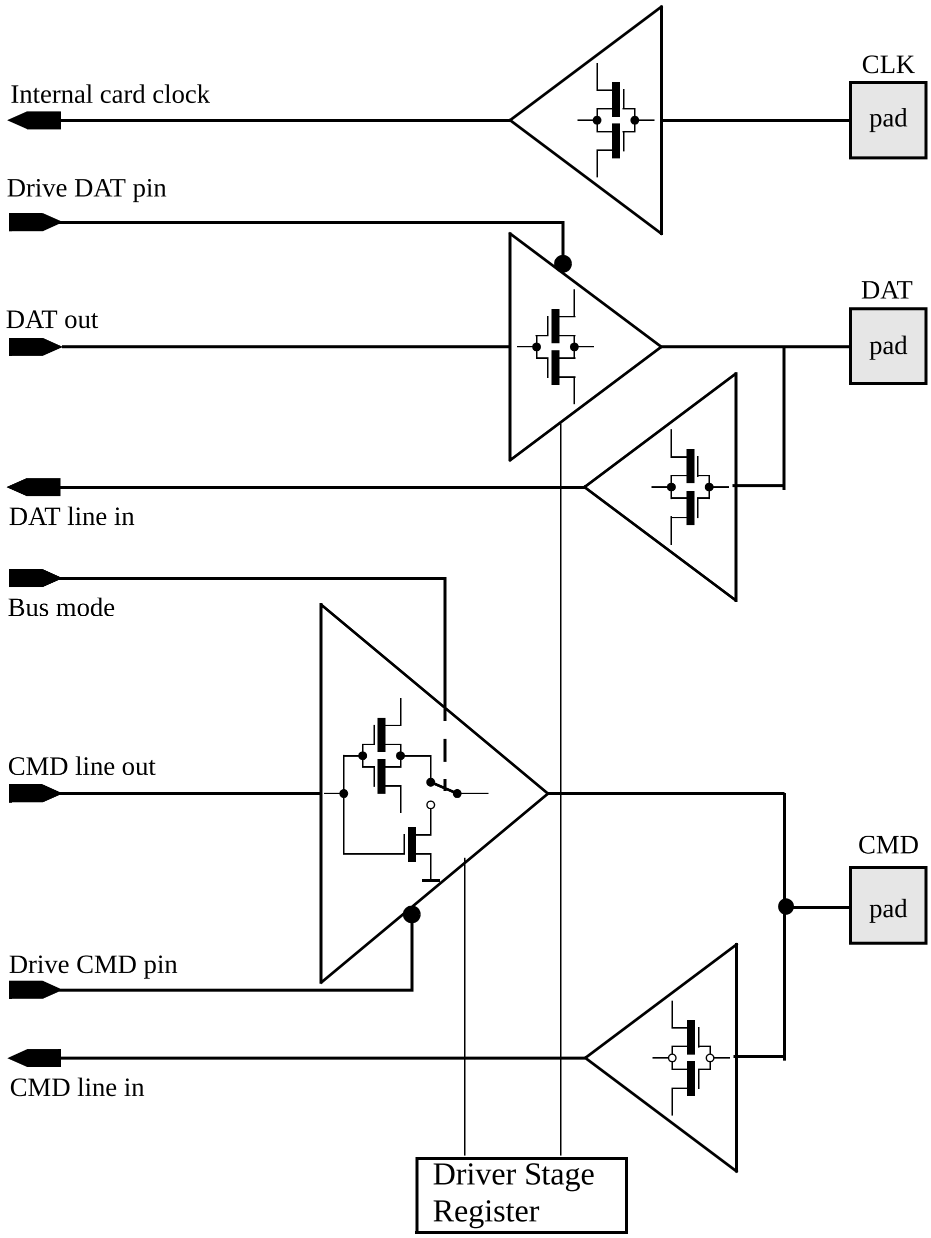


Figure 64 — MultiMediaCard bus driver

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:

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

12.5 Bus Operating Conditions

This section describes the basic bus operating power features

- **General**

Table 72 — General bus operating conditions

Parameter	Symbol	Min	Max.	Unit	Remarks
Peak voltage on all lines		-0.5	3.6	V	
All Inputs					
Input Leakage Current (before initialization sequence[1])		-100	100	μA	
Input Leakage Current (after initialization sequence)		-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 117.					

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

Table 73 — MMC high-voltage power supply voltage

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	

- **Power Supply Voltage—Dual voltage MultiMediaCard**

Table 74 — MMC dual-voltage power supply voltage

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

The current consumption of 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, while the host has not sent yet a valid OCR range, must not exceed 10 mA

- **Bus Signal Line Load**

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

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

Requiring the sum of the host and bus capacitances not to exceed 20 pF:

Table 75 — Bus signal line load parameters

Parameter	Symbol	Min	Max.	Unit	Remark
Pull-up resistance for CMD	R_{CMD}	4.7	100	KOhm	to prevent bus floating
Pull-up resistance for DAT0-7	R_{DAT}	50	100	KOhm	to prevent bus floating
Bus signal line capacitance	C_L		30	pF	Single card
Single card capacitance	C_{CARD}		7	pF	
Maximum signal line inductance			16	nH	f_{PP} <= 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.

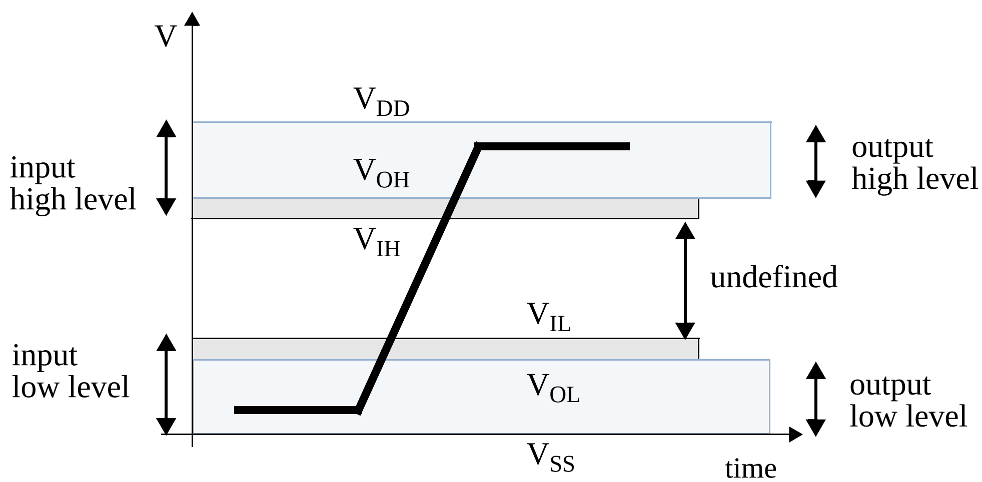


Figure 65 — Bus signal levels

12.6.1 Open-Drain Mode Bus Signal Level

Table 76 — Open-drain signal levels

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 77 — High-voltage MMC push-pull signal levels

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

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

Table 78 — Dual-voltage MMC push-pull signal levels

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

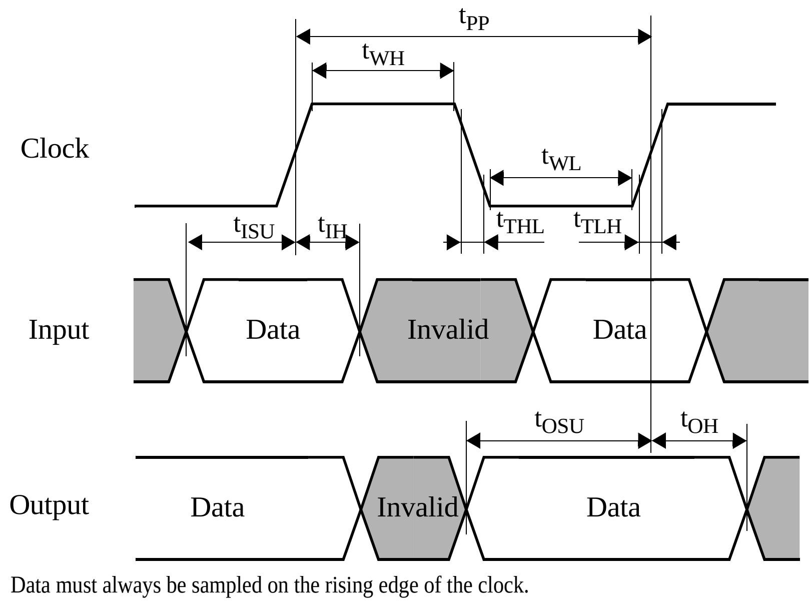


Figure 66 — Timing diagram: data input/output

12.7.1 Card Interface Timings

Table 79 — Card interface timing 26/52 MHz

Parameter	Symbol	Min	Max.	Unit	Remarks
Clock (CLK)[1]					
Clock frequency Data Transfer Mode (PP)[2]	f_{PP}	0	26/52	MHz	C_L <= 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 <= 30 pF
Clock rise time[3]	t_{TLH}		3	ns	C_L <= 30 pF
Clock fall time	t_{THL}		3	ns	C_L <= 30 pF
Inputs CMD, DAT (referenced to CLK)					
Input set-up time	t_{ISU}	3		ns	C_L <= 30 pF
Input hold time	t_{IH}	3		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
Signal rise time[4]	t_{rise}		3	ns	C_L <= 30 pF
Signal fall time	t_{fall}		3	ns	C_L <= 30 pF
NOTE 1 All timing values are measured relative to 50% of voltage level. NOTE 2 A MultiMediaCard shall support the full frequency range from 0-26Mhz, or 0-52MHz. NOTE 3 Rise and fall times are measured from 10%–90% of voltage level. NOTE 4 Rise and fall times are measured from 10%–90% of voltage level.					

Table 80 — Card interface timing 20 MHz

Parameter	Symbol	Min	Max.	Unit	Remarks
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}	3		ns	C_L <= 30 pF
Input hold time	t_{IH}	3		ns	C_L <= 30 pF
Outputs CMD, DAT (referenced to CLK)					

Table 80 — Card interface timing 20 MHz (continued)

Parameter	Symbol	Min	Max.	Unit	Remarks
Output set-up time	t_{OSU}	13.1		ns	C_L <= 30 pF
Output hold time	t_{OH}	9.7		ns	C_L <= 30 pF

NOTE 1 All timing values are measured relative to 50% of voltage level.
NOTE 2 Clock rise and fall times are measured from VIL to VIH of voltage level.

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, Section 8)
- The description of the SPI mode (Section 9)
- Data integrity and error handling (Section 10)
- Mechanical interface parameters, such as: connector type and dimensions and the card form factor (Section 11)
- Electrical interface parameters, such as: power supply, peak and average current consumption and data transfer frequency (Section 12)
- 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 6.1 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 68). 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 7.7 on page 41). The meaning of the entries is as follows:

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

Table 81 — CCC requirements

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
2.7-3.6 volts power supply	Mandatory	Mandatory	Mandatory

Table 81 — CCC requirements (continued)

Function	ROM card class	R/W card class	I/O card class
1.65-1.95 volts power supply	Optional	Optional	Optional
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	Mandatory	Mandatory	Mandatory

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 30 on page 56.

14 File Formats For The MultiMediaCard

The file format specification, for the MultiMediaCard, starting with V4.0 of this document, has been moved into a separate document called the “File Formats Specifications For MultiMediaCards.”

15 Abbreviations and Terms

Block	a number of bytes, basic data transfer unit
Broadcast	a command sent to all cards on the MultiMediaCard bus[5]
CID	Card IDentification number register
CLK	clock signal
CMD	command line or MultiMediaCard bus command (if extended CMDXX)
CRC	Cyclic Redundancy Check
CSD	Card Specific Data register
DAT	data line
DSR	Driver Stage Register
Flash	a type of multiple time programmable non volatile memory
Group	a number of write blocks, composite erase and write protect unit
LOW, HIGH	binary interface states with defined assignment to a voltage level
NSAC	defines the worst case for the clock rate dependent factor of the data access time
MSB, LSB	the Most Significant Bit or Least Significant Bit
OCR	Operation Conditions Register
open-drain	a logical interface operation mode. An external resistor or current source is used to pull the interface level to HIGH, the internal transistor pushes it to LOW
payload	net data
push-pull	a logical interface operation mode, a complementary pair of transistors is used to push the interface level to HIGH or LOW
RCA	Relative Card Address register
ROM	Read Only Memory
stuff bit	filling 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

5. 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.

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 67.

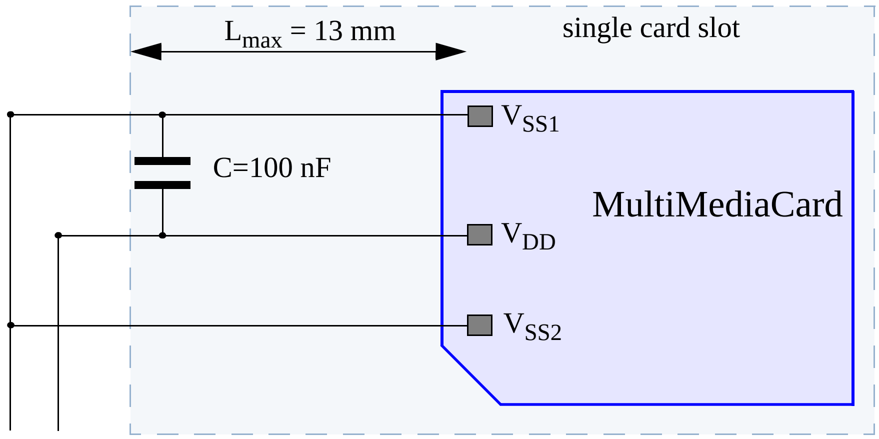


Figure 67 — Power supply decoupling

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 68. 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.

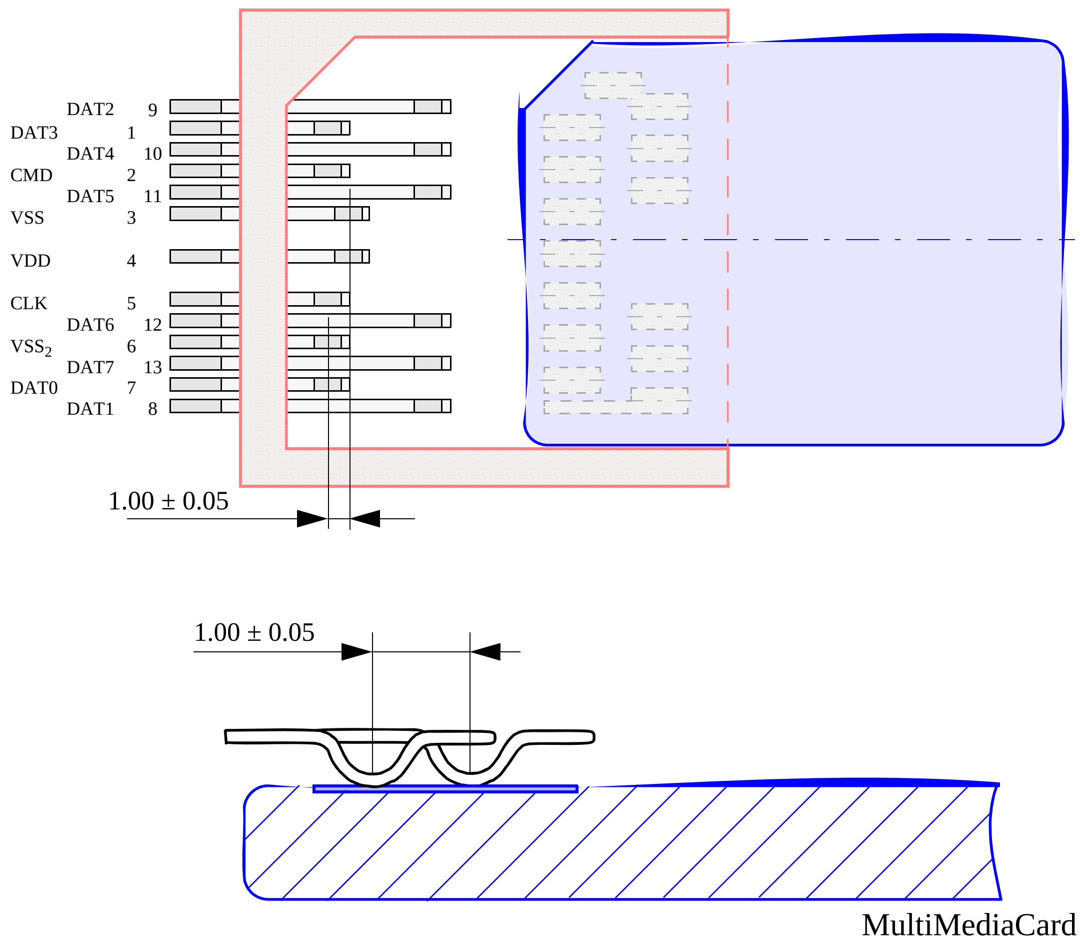


Figure 68 — Modified MultiMediaCard connector for hot insertion

A.3.3 Characteristics

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

- **Mechanical Characteristics**

- Max. number of mating operations	> 10000	
- Contact force	0.2.....0.6 N	
- Total pulling force	min.2 N	DIN IEC 512 part 7
- Total insertion force	max. 40 N	DIN IEC 512 part 7
- Vibration and High Frequency		
- Mechanical frequency range	10...2000 Hz	DIN IEC 512 part 2 and 4
- Acceleration	2 g	
- Shock:		
- Acceleration	5 g	

- **Electrical Characteristics**

DIN IEC 512

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

- **Climatic Characteristics**

DIN IEC 512 part 6-9

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

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

In order to improve compatibility and inter-operability of the card between different applications, it is required that different host 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 possi-

ble 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

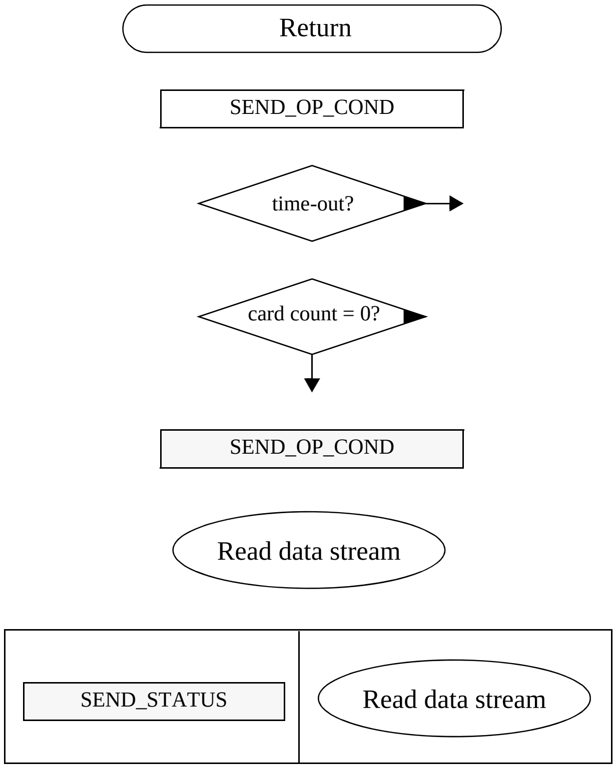
This section defines the way complex MultiMediaCard bus operations (e.g. erase, read, etc.) may be executed using predefined command sequences. Executing these sequences is the responsibility of the MultiMediaCard bus master. Nevertheless, it may be used for host compatibility test purposes.

Table 82 — 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 69 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.



Sequence entry/exit point

MultiMediaCard BUS Command

False exit of condition

True exit of condition

Optional steps in sequence

Host Activity

Parallel activities in the host. The left and right sequences are executed simultaneously.

Figure 69 — Legend for command sequences flow charts

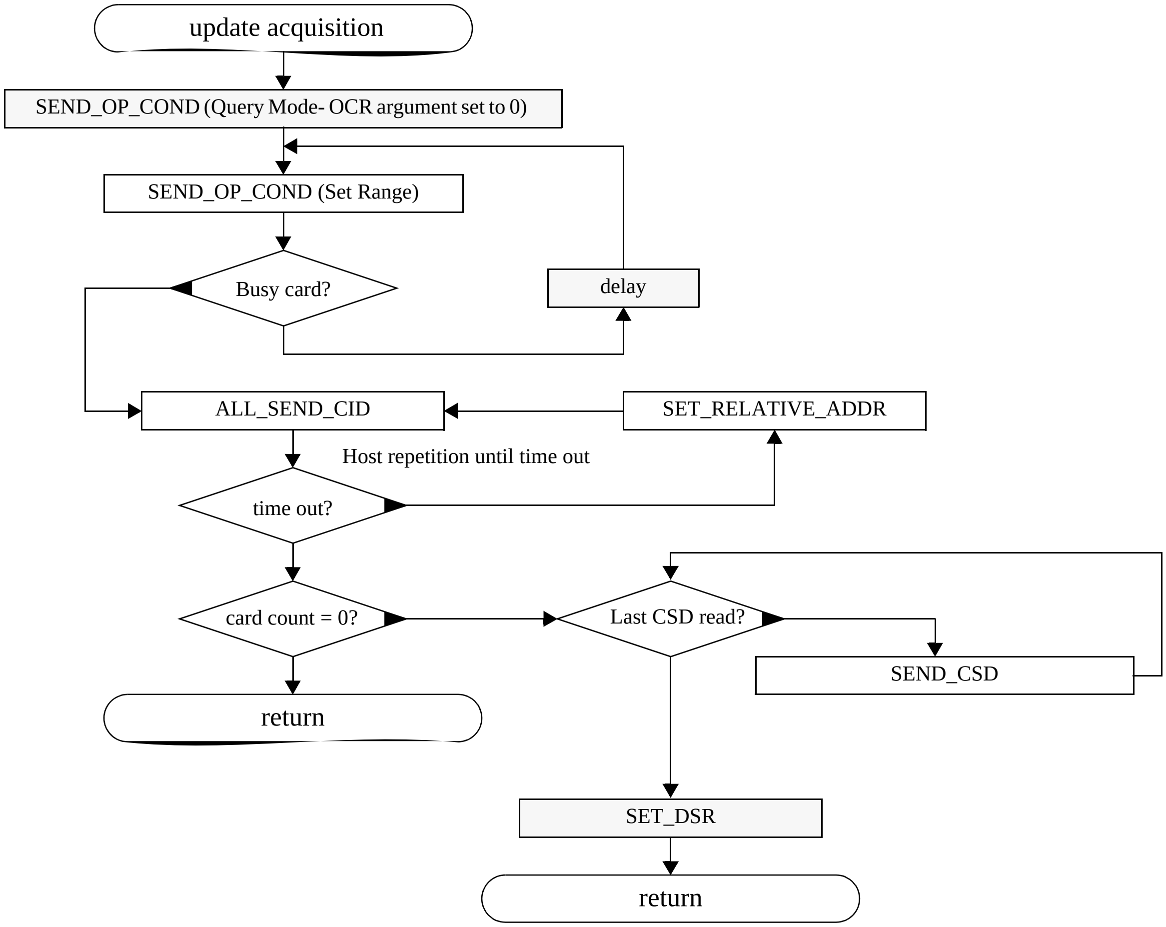


Figure 70 — SEND_OP_COND command flow chart

- **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.

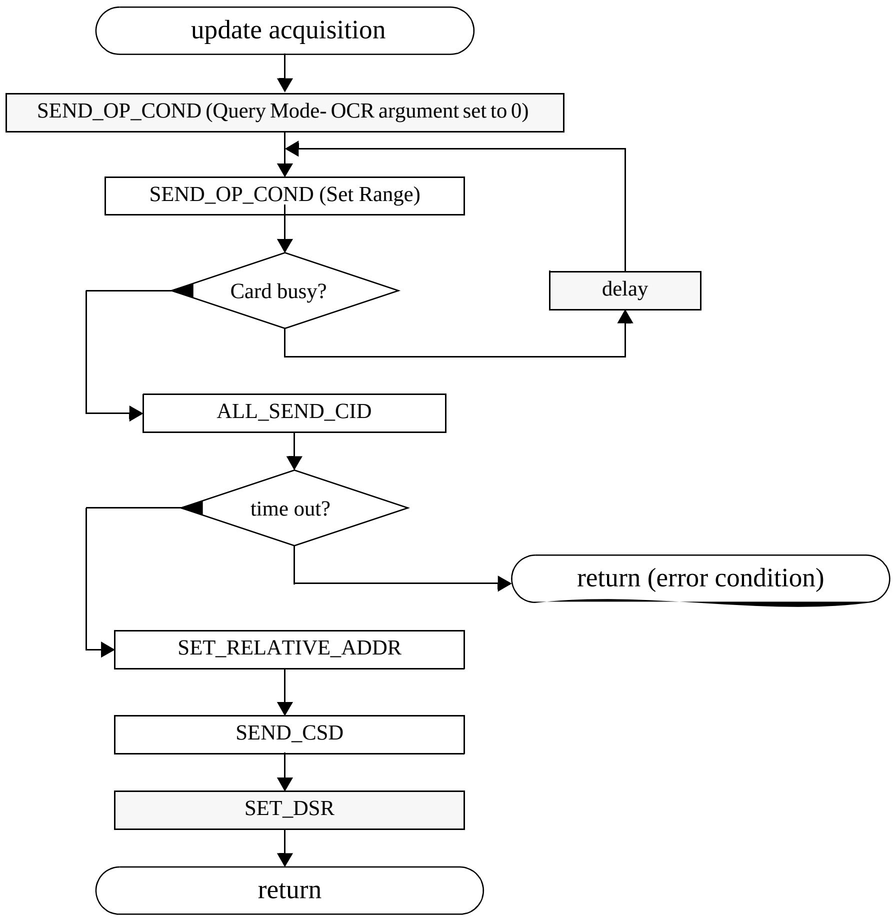


Figure 71 — CIM_SINGLE_CARD_ACQ

- **CIM_SETUP_CARD**

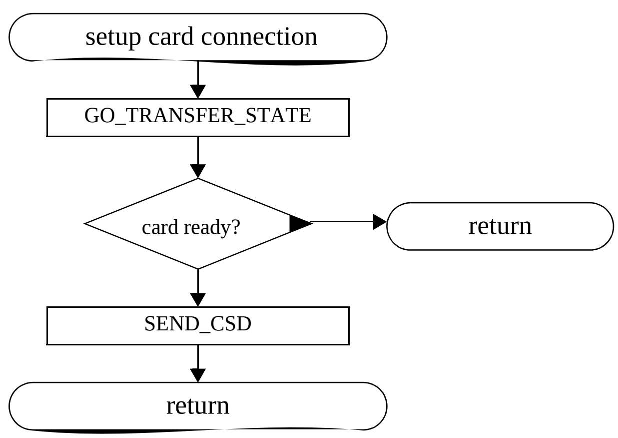


Figure 72 — CIM_SETUP_CARD

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**

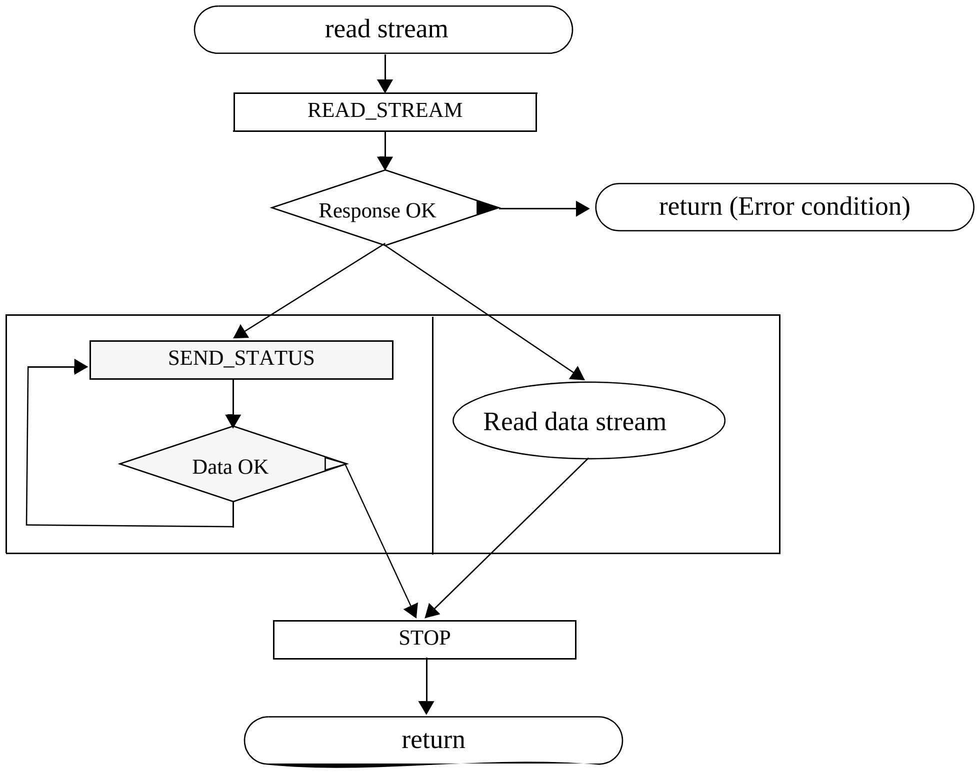


Figure 73 — CIM_STREAM_READ

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**

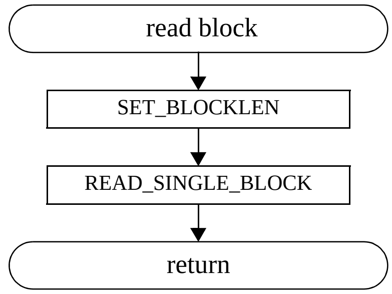


Figure 74 — CIM_READ_BLOCK

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**

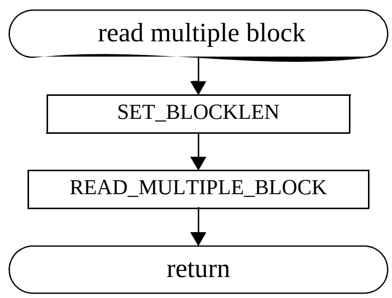


Figure 75 — CIM_READ_MBLOCK

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.

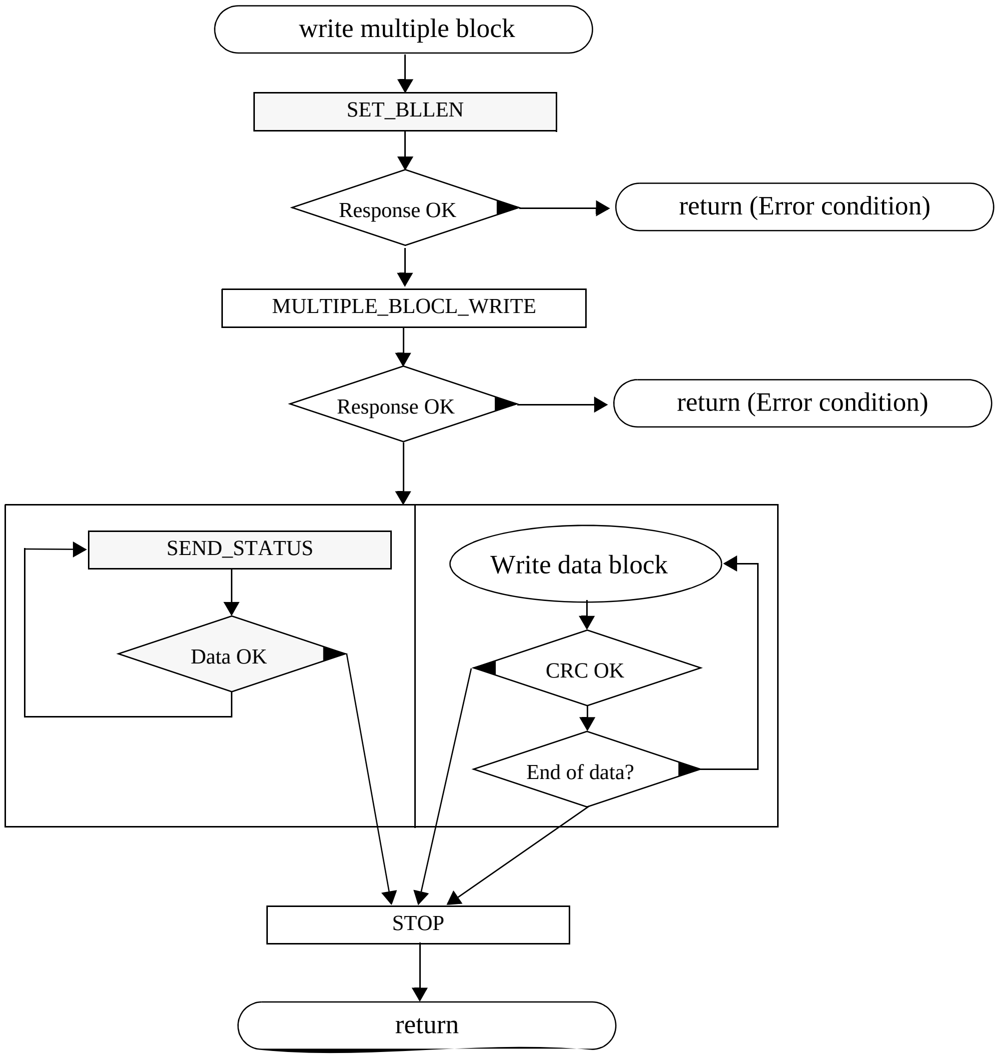


Figure 76 — CIM_WRITE_MBLOCK

- **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.).

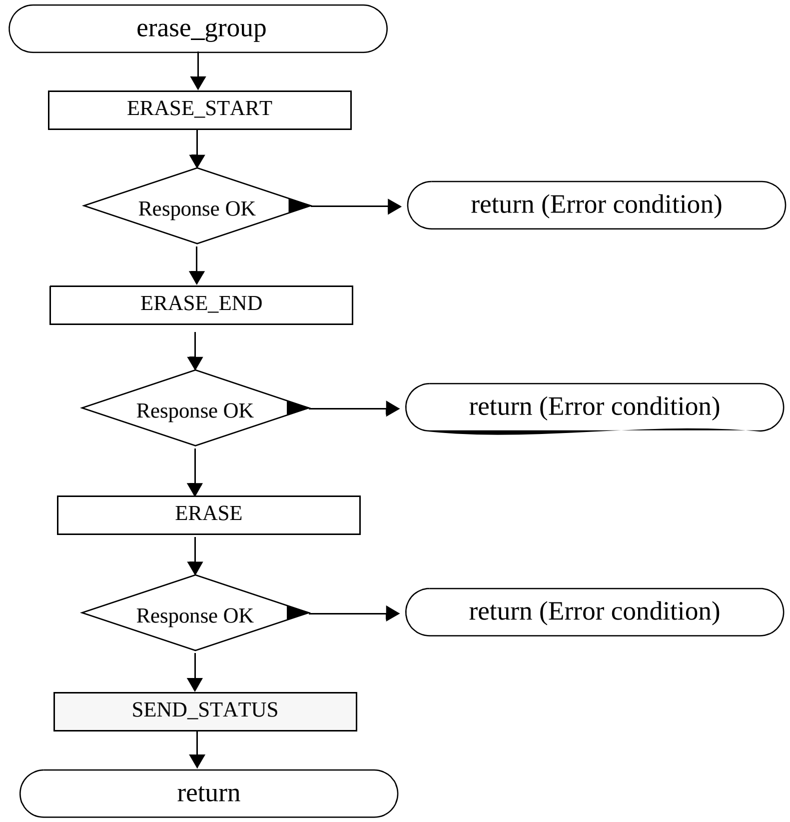


Figure 77 — CIM_ERASE_GROUP

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 Section 12.7 on page 122.

Table 83 — Forward-compatibility 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	

Table 83 — Forward-compatibility interface timing (continued)

Parameter	Symbol	Min	Max.	Unit	Remark
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 High Speed MultiMediaCard Bus Functions

A.7.1 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[6].

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

3- Apply power to the bus, communication voltage range (2.7-3.6V)
4- Set clock to 400KHz, or less
5- Wait for 1ms, then wait for 74 more clock cycles
6- Send CMD0 to reset the bus, keep CS line high during this step.
7- Send CMD1, with the intended voltage range in the argument (either 0x00FF8000 or 0x00000080)
8- Receive R3
9- If the OCR busy bit is ‘0’, repeat steps 5 and 6
10- 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.
11- 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

6. 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.

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.

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

b. CID Retrieval And RCA Assignment

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

c. CSD Retrieval And Host Adjustment

21- Send CMD9
22- Receive R2, and get the card’s CSD from it.
23- Adjust the host parameters, if necessary, 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.7.2 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 Appendix A.7.1

24- Send CMD7 with the card’s RCA to place the card in *tran* state
25- 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)
26- Send CMD6, writing 0x1 to the HS_TIMING byte of the EXT_CSD. The argument 0x03B9_0100 will do it.
26.1-The card might enter BUSY right after R1, if so, wait until the BUSY signal is de-asserted
26.2-After the card comes out of BUSY it is configured for high speed timing
27- Change the clock frequency to the chosen frequency (any frequency between 0 and 26/52MHz).

A.7.3 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 Appendix A.7.1

a. Bus Testing Procedure

28-Send CMD19
29-Send a block of data, over all the bus data lines, with the data pattern as follows (CRC16 is optional):
  29.1-For 8 data lines the data block would be (MSB to LSB): 0x0000_0000_0000_AA55
  29.2-For 4 data lines the data block would be (MSB to LSB): 0x0000_005A
  29.3-For only 1 data line the data block would be: 0x80

	Start	←Test Pattern→								¨æææ Optional æææÆ	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 78 — Test pattern for 8-bit bus

	Start	←Test Pattern→								¨æææ Optional æææÆ	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 79 — Test pattern for 4-bit bus

	Start	←Test Pattern→								¨æææ Optional æææÆ	End
DAT0	0	1	0	0	0	0	0	0	0	CRC16	1
		0x80									

Figure 80 — Test pattern for 1-bit bus

30-Wait for at least N_{CR} clock cycles before proceeding
31-Send CMD14 and receive a block of data from all the available data lines[7]
  31.1-For 8 data lines receive 8 bytes

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

31.2-For 4 data lines receive 4 bytes
31.3-For 1 data line receive 1 byte

32-XNOR the masked data with the data sent in step 27

Table 84 — XNOR

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

33-Mask the result according to the following:
33.1-For 8 data lines the mask is (MSB to LSB): 0x0000_0000_0000_FFFF
33.2-For 4 data lines the mask is (MSB to LSB): 0x0000_00FF
33.3-For 1 data line the mask is 0xC0
34-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

35-Choose the width of bus you want to work with
36-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.
37-The card might signal BUSY after CMD6; wait for the card to be out of BUSY
38-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.
39-The bus is ready to exchange data using the new width configuration.

Annex B: Version Changes

This section lists the changes that have occured between Electrical 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 6.1.)
- The form factor references were removed. (See Section 3.1.2.)
- The CSD_STRUCTURE and SPEC_VERS registers were modified to include only allocations applicable to this Electrical Specification. (See Section 8.3 and Section 8.4.)
- The mechanical specification was removed. (See Section 11.)
- The Appendix A was removed and introduced as a separate document. (See Annex A.)

Annex C: Errata List

This annex consists of the Errata List for Non-Volatile Memory Cards and the Modules Electrical Specification.

C.1 Scope

Scope of this Errata document is to list all the known errata items related to the JEDEC Non-Volatile Memory Card and Module Electrical Specifications. The following sections are listing the errata items per corresponding version of the Electrical Specification. The target is that known errata will be corrected later in the new version of the Electrical Specification.

All references to Chapters, Sections, Figures and Tables are referring to the corresponding version of the Electrical Specification.

C.2 Errata

C.2.1 Electrical Specification Version 4.1

C.2.1.1 Section 4.10, Bullet R1b

The reference included to the paragraph is incorrectly to Section 4.13.3. The correct reference is to Section 4.13 or more precisely 4.13.1.

C.2.1.2 Section 4.13.1, Bullet R1b

The 1st sentence is misleadingly defining the timing of the response as follows: “Some commands, like CMD6, may assert the BUSY signal after responding with R1”. Still it's clearly defined in the Figure 24 and Table 25 that the possible busy indication have to start exactly 2 clock cycles after the end bit of the command. As the N_{CR} is defined to be minimum 2, maximum 64 clock cycles it's possible that the response comes either at the same time or after the indication of busy. Thus the 1st sentence should rather state “Some commands, like CMD6, may assert the BUSY signal and respond with R1.”

C.2.1.3 Section 4.13.1, Figure 24

The response frame is missing from Figure 24. The response frame should appear similarly to Figure 23.

C.2.1.4 Section 4.13.4, Figure 36

The N_{AC} timing is incorrectly starting from end bit of the response frame. The correct starting point is from the end bit of the command frame like in normal data read situation (e.g. Figure 27).

C.2.1.5 Section 4.4, Figure 19 and Section 4.4.1

A card responds with R1b for CMD6. The card may assert busy signal after receiving the command. If the usage of CMD6 was related to the timing mode change the card might be changing the timings during the busy period. It not defined in the Electrical Specification that a host should not be sending any commands (e.g. CMD13) during the busy period in this particular case without being able to handle the situation in which the card may reply with either possible timing modes.

C.2.1.6 2.1.6Section 4.4.4

There is no specific data sending state related to the Bus Test procedure. Thus it's not clear in which state the card should switch from the bus btst-state to tran-state after CMD14. The data-state defines that all read commands during data transfer should be considered as illegal. Similarly there should be no new read commands sent by the host before the Bus Test procedure is finalized.

C.2.1.7 Chapter 10

In the version 4.1 the SPI mode was defined to be mandatory to be supported by all cards. In Chapter 10 the bullet “Mandatory” defines that “any MultiMediaCard host supporting the specified card class must implement this function.” The SPI mode is marked as “Mandatory” in the table in Chapter 10. Still the mandatory support of SPI mode was intended for the cards only (to maximize interoperability). A host is not required to implement both MMC and SPI modes (which would not even be practical).

C.3 Version history

C.3.1 Version 1v0

Initial version of this document.

C.3.2 Version 1v01

Addition of Section 2.1.7.

Standard Improvement Form **JEDEC** **JESD84-B41**

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

MULTIMEDIACARD®
ASSOCIATION