

JEDEC STANDARD

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

JESD84-B41

JUNE 2007

JEDEC SOLID STATE TECHNOLOGY ASSOCIATION



NOTICE

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

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

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

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

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

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

Published by

©JEDEC Solid State Technology Association 2007

2500 Wilson Boulevard

Arlington, VA 22201-3834

This document may be downloaded free of charge; however JEDEC retains the copyright on this material. By downloading this file the individual agrees not to charge for or resell the resulting material.

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.

All rights reserved

PLEASE!

DON'T VIOLATE THE LAW!

This document is copyrighted by JEDEC and may not be reproduced without permission.

Organizations may obtain permission to reproduce a limited number of copies through entering into a license agreement. For information, contact:

JEDEC Solid State Technology Association
2500 Wilson Boulevard
Arlington, Virginia 22201-3834
or call (703) 907-7559

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

CONTENTS

	Page
Foreword	xi
Introduction	xi
1 Scope	1
2 Normative reference	1
3 Terms and definitions	1
4 General Description	3
5 System Features	5
6 MultiMediaCard System Concept	7
6.1 Card Concept	10
6.1.1 MMCplus and MMCmobile	11
6.2 Bus Concept	12
6.2.1 Bus Lines	13
6.2.2 Bus Protocol	13
6.3 Controller Concept	18
6.3.1 Application Adapter Requirements	18
6.3.2 MultiMediaCard Adapter Architecture	19
7 MultiMediaCard Functional Description	21
7.1 General	21
7.2 Card Identification Mode	22
7.2.1 Card Reset	22
7.2.2 Operating Voltage Range Validation	22
7.2.3 Card Identification Process	24
7.3 Interrupt Mode	24
7.4 Data Transfer Mode	25
7.4.1 Command Sets And Extended Settings	27
7.4.2 High Speed Mode Selection	28
7.4.3 Power Class Selection	28
7.4.4 Bus Testing Procedure	29
7.4.5 Bus Width Selection	30
7.4.6 Data Read	30
7.4.7 Data Write	32
7.4.8 Erase	35

MultiMediaCard (MMC) Electrical Standard, Standard Capacity**CONTENTS(*continued*)**

	Page
7.4.9 Write Protect Management	35
7.4.10 Card Lock/Unlock Operation	36
7.4.11 Application specific commands	38
7.5 Clock Control	39
7.6 Error Conditions	39
7.6.1 CRC and Illegal Command	39
7.6.2 Read, Write, Erase And Force Erase Time-out Conditions	40
7.6.3 Read ahead in Stream and multiple block read operation	40
7.7 Minimum Performance	41
7.7.1 Speed Class Definition	41
7.7.2 Absolute Minimum	41
7.7.3 Measurement of the Performance	41
7.8 Commands	42
7.8.1 Command Types	42
7.8.2 Command Format	42
7.8.3 Command Classes	42
7.8.4 Detailed Command Description	44
7.9 Card State Transition Table	49
7.10 Responses	51
7.11 Card Status	53
7.12 Memory Array Partitioning	57
7.13 Timings	58
7.13.1 Command and Response	59
7.13.2 Data Read	60
7.13.3 Data Write	61
7.13.4 Bus Test Procedure Timing	64
7.13.5 Timing Values.	64
8 Card Registers	67
8.1 OCR Register	67
8.2 CID Register	67
8.3 CSD Register	68
8.4 Extended CSD Register	77
8.5 RCA Register	82
8.6 DSR Register	82
9 SPI Mode	83

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

CONTENTS(*continued*)

	Page
9.1 Introduction	83
9.2 SPI Interface Concept	83
9.3 SPI Bus Topology	83
9.4 MultiMediaCard Registers in SPI Mode	85
9.5 SPI Bus Protocol	85
9.5.1 Mode Selection	85
9.5.2 Bus Transfer Protection	86
9.5.3 Data Read	86
9.5.4 Data Write	88
9.5.5 Erase & Write Protect Management	89
9.5.6 Read CID/CSD Registers	90
9.5.7 Reset Sequence	90
9.5.8 Clock Control	90
9.5.9 Error Conditions	91
9.5.10 Memory Array Partitioning	92
9.5.11 Card Lock/unlock	92
9.5.12 Application Specific commands	92
9.6 SPI Mode Transaction Packets	92
9.6.1 Command Tokens	92
9.6.2 Responses	96
9.6.3 Data Tokens	99
9.6.4 Data Error Token	99
9.6.5 Clearing Status Bits	100
9.7 Card Registers	104
9.8 SPI Bus Timing Diagrams	104
9.8.1 Command / Response	104
9.8.2 Data read	105
9.8.3 Data write	107
9.8.4 Timing Values	108
9.9 SPI Electrical Interface	108
9.10 SPI Bus Operating Conditions	108
9.11 Bus Timing	108
10 Error protection	109
10.1 Error Correction Codes (ECC)	109
10.2 Cyclic Redundancy Codes (CRC)	109

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

CONTENTS(*continued*)

	Page
11 MultiMediaCard Mechanical Specification	113
12 The MultiMediaCard Bus	115
12.1 Hot Insertion and Removal	115
12.2 Power Protection	116
12.3 Power Up	117
12.4 Programmable Card Output Driver	118
12.5 Bus Operating Conditions	120
12.6 Bus Signal Levels	121
12.6.1 Open-Drain Mode Bus Signal Level	121
12.6.2 Push-Pull Mode Bus Signal Level - High Voltage MultiMediaCard	121
12.6.3 Push-Pull Mode Bus Signal Level - Dual voltage MultiMediaCard	122
12.7 Bus Timing	122
12.7.1 Card Interface Timings	123
13 MultiMediaCard Standard Compliance	125
14 File Formats For The MultiMediaCard	127
15 Abbreviations and Terms	129
Annex A: Application Notes	131
A.1 Power Supply Decoupling	131
A.2 Payload Block Length and ECC Types Handling	131
A.3 Connector	131
A.3.1 General	131
A.3.2 Card Insertion and Removal	132
A.3.3 Characteristics	132
A.4 Description of method for storing passwords on the card	133
A.5 MultiMediaCard Macro Commands	134
A.6 Host Interface Timing	141
A.7 High Speed MultiMediaCard Bus Functions	142
A.7.1 Bus Initialization	142
A.7.2 Switching to High Speed Mode	143
A.7.3 Changing the Data Bus Width	143
Annex B: Version Changes	147
B.1 Version 4.1, the First Version of This Specification	147

MultiMediaCard (MMC) Electrical Standard, Standard Capacity**CONTENTS(*continued*)**

	Page
Annex C: Errata List	149
C.1 Scope	149
C.2 Errata	149
C.2.1 Electrical Specification Version 4.1	149
C.3 Version history	150
C.3.1 Version 1v0	150
C.3.2 Version 1v01	150

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

FIGURES

	Page
Figure 1 — Topology of MultiMediaCard systems	7
Figure 2 — MultiMediaCard system overview	8
Figure 3 — MultiMediaCard system example	9
Figure 4 — MultiMediaCard architecture	12
Figure 5 — MultiMediaCard bus system	13
Figure 6 — Sequential read operation	14
Figure 7 — (Multiple) Block read operation	14
Figure 8 — Sequential write operation	15
Figure 9 — (Multiple) Block write operation	15
Figure 10 — “No response” and “No data” operations	15
Figure 11 — Command token format	16
Figure 12 — Response token format	16
Figure 13 — Data packet format	17
Figure 14 — MultiMediaCard controller scheme	18
Figure 15 — MultiMediaCard adapter architecture	19
Figure 16 — MultiMediaCard state diagram (card identification mode)	23
Figure 17 — MultiMediaCard state transition diagram, interrupt mode	25
Figure 18 — MultiMediaCard state diagram (data transfer mode)	26
Figure 19 — Memory Array Partitioning	58
Figure 20 — Identification timing (card identification mode)	59
Figure 21 — SET_RCA timing (card identification mode)	59
Figure 22 — Command response timing (data transfer mode)	59
Figure 23 — R1b response timing	60
Figure 24 — Timing response end to next command start (data transfer mode)	60
Figure 25 — Timing of command sequences (all modes)	60
Figure 26 — Single block read timing	60
Figure 27 — Multiple block read timing	61
Figure 28 — Stop command timing (CMD12, data transfer mode)	61
Figure 29 — Block write command timing	62
Figure 30 — Multiple block write timing	62
Figure 31 — Stop transmission during data transfer from the host	62
Figure 32 — Stop transmission during CRC status transfer from the card	63
Figure 33 — Stop transmission after last data block. Card is busy programming.	63
Figure 34 — Stop transmission after last data block. Card becomes busy.	63
Figure 35 — 4-bit system bus-testing procedure	64
Figure 36 — MultiMediaCard bus system	84
Figure 37 — SPI single block read operation	86
Figure 38 — SPI multiple block read operation	86
Figure 39 — SPI read operation—data error	87
Figure 40 — SPI single block write operation	88
Figure 41 — SPI multiple block write operation	88

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

FIGURES(continued)

	Page
Figure 42 — SPI “no data” operations	90
Figure 43 — R1 response format	97
Figure 44 — R1b response format	97
Figure 45 — R2 response format	98
Figure 46 — R3 response format	98
Figure 47 — Data response format.	99
Figure 48 — Data error token.	100
Figure 49 — SPI command/response transaction, card is ready	104
Figure 50 — SPI command/response transaction, card is busy	105
Figure 51 — SPI card response to the next host command	105
Figure 52 — SPI single block read.	105
Figure 53 — SPI multiple block read, stop transmission does not overlap data	106
Figure 54 — SPI multiple block read, stop transmission overlaps data.	106
Figure 55 — SPI read CSD and CID registers	107
Figure 56 — SPI single block write	107
Figure 57 — SPI multiple block write	107
Figure 58 — CRC7 generator/checker	110
Figure 59 — CRC16 generator/checker	111
Figure 60 — Bus circuitry diagram	115
Figure 61 — Improper power supply	116
Figure 62 — Shortcut protection	116
Figure 63 — Power-up diagram	117
Figure 64 — MultiMediaCard bus driver.	119
Figure 65 — Bus signal levels	121
Figure 66 — Timing diagram: data input/output	122
Figure 67 — Power supply decoupling	131
Figure 68 — Modified MultiMediaCard connector for hot insertion.	132
Figure 69 — Legend for command sequences flow charts	135
Figure 70 — SEND_OP_COND command flow chart	136
Figure 71 — CIM_SINGLE_CARD_ACQ	137
Figure 72 — CIM_SETUP_CARD	137
Figure 73 — CIM_STREAM_READ	138
Figure 74 — CIM_READ_BLOCK.	139
Figure 75 — CIM_READ_MBLOCK	139
Figure 76 — CIM_WRITE_MBLOCK	140
Figure 77 — CIM_ERASE_GROUP	141
Figure 78 — Test pattern for 8-bit bus	144
Figure 79 — Test pattern for 4-bit bus	144
Figure 80 — Test pattern for 1-bit bus	144

MultiMediaCard (MMC) Electrical Standard, Standard Capacity

TABLES

	Page
Table 1 — System voltage ranges	5
Table 2 — MMC operational modes	6
Table 3 — Card contacts	10
Table 4 — Card registers	11
Table 5 — Card states and modes	21
Table 6 — Card access modes	28
Table 7 — Bus-testing data pattern	29
Table 8 — 1-bit bus-testing pattern	29
Table 9 — 4-bit bus-testing pattern	30
Table 10 — 8-bit bus-testing pattern	30
Table 11 — Command data block structure	36
Table 12 — Command code length	42
Table 13 — Card command classes	43
Table 14 — Basic commands and read stream commands: class 0 and class 1	44
Table 15 — Block-oriented read commands: class 2	45
Table 16 — Stream write commands: class 3	45
Table 17 — Block-oriented write commands: class 4	46
Table 18 — Block-oriented write protection commands: class 6	46
Table 19 — Erase commands: class 5	47
Table 20 — I/O mode commands: class 9	48
Table 21 — Lock card: class 7	48
Table 22 — Application-specific commands: class 8	48
Table 23 — Card state transition table	49
Table 24 — R1 bits	51
Table 25 — R2 bits	52
Table 26 — R3 bits	52
Table 27 — R4 bits	52
Table 28 — R5 bits	52
Table 29 — Response and execution mode conditions	53
Table 30 — Card status field/command cross reference	56
Table 31 — Timing diagram abbreviations	58
Table 32 — Timing parameters	64
Table 33 — OCR register	67
Table 34 — CID register	67
Table 35 — CSD register	69
Table 36 — CSD structure	70
Table 37 — Specification version	70
Table 38 — TAAC bit codes	70
Table 39 — Transfer speed	71
Table 40 — Card command class support	71
Table 41 — Read block length	72
Table 42 — Driver stage register implementation	72

MultiMediaCard (MMC) Electrical Standard, Standard Capacity**TABLES(continued)**

	Page
Table 43 — Minimal power supply VDD maximum values	73
Table 44 — Maximal power supply VDD maximum values.....	73
Table 45 — Device size multipliers	73
Table 46 — R2W factor field format	74
Table 47 — Card file formats.....	75
Table 48 — Error correction code field formats	76
Table 49 — CSD command classes	76
Table 50 — Extended CSD register	77
Table 51 — Card-supported command sets	78
Table 52 — R/W access performance values.....	79
Table 53 — Power classes.....	80
Table 54 — Card types.....	81
Table 55 — CSD structures.....	81
Table 56 — Extended CSD register revision parameters.....	81
Table 57 — Command set revision	81
Table 58 — Power class.....	82
Table 59 — Bus Mode Value.....	82
Table 60 — SPI interface pin configuration	84
Table 61 — MultiMediaCard registers in SPI mode.....	85
Table 62 — SPI mode command formats	92
Table 63 — SPI mode command classes	93
Table 64 — SPI mode commands	94
Table 65 — Start data block token format	99
Table 66 — SPI mode status bit descriptions.....	100
Table 67 — SPI mode data error response tokens	103
Table 68 — SPI bus timing abbreviations.....	104
Table 69 — SPI mode timing.....	108
Table 70 — ECC parameters.....	109
Table 71 — DSR content	118
Table 72 — General bus operating conditions.....	120
Table 73 — MMC high-voltage power supply voltage	120
Table 74 — MMC dual-voltage power supply voltage.....	120
Table 75 — Bus signal line load parameters.....	121
Table 76 — Open-drain signal levels.....	121
Table 77 — High-voltage MMC push-pull signal levels.....	122
Table 78 — Dual-voltage MMC push-pull signal levels	122
Table 79 — Card interface timing 26/52 MHz	123
Table 80 — Card interface timing 20 MHz	123
Table 81 — CCC requirements	125
Table 82 — Macro commands	134
Table 83 — Forward-compatibility interface timing	141
Table 84 — XNOR.....	145

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 (VDD) 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 VDD. 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.

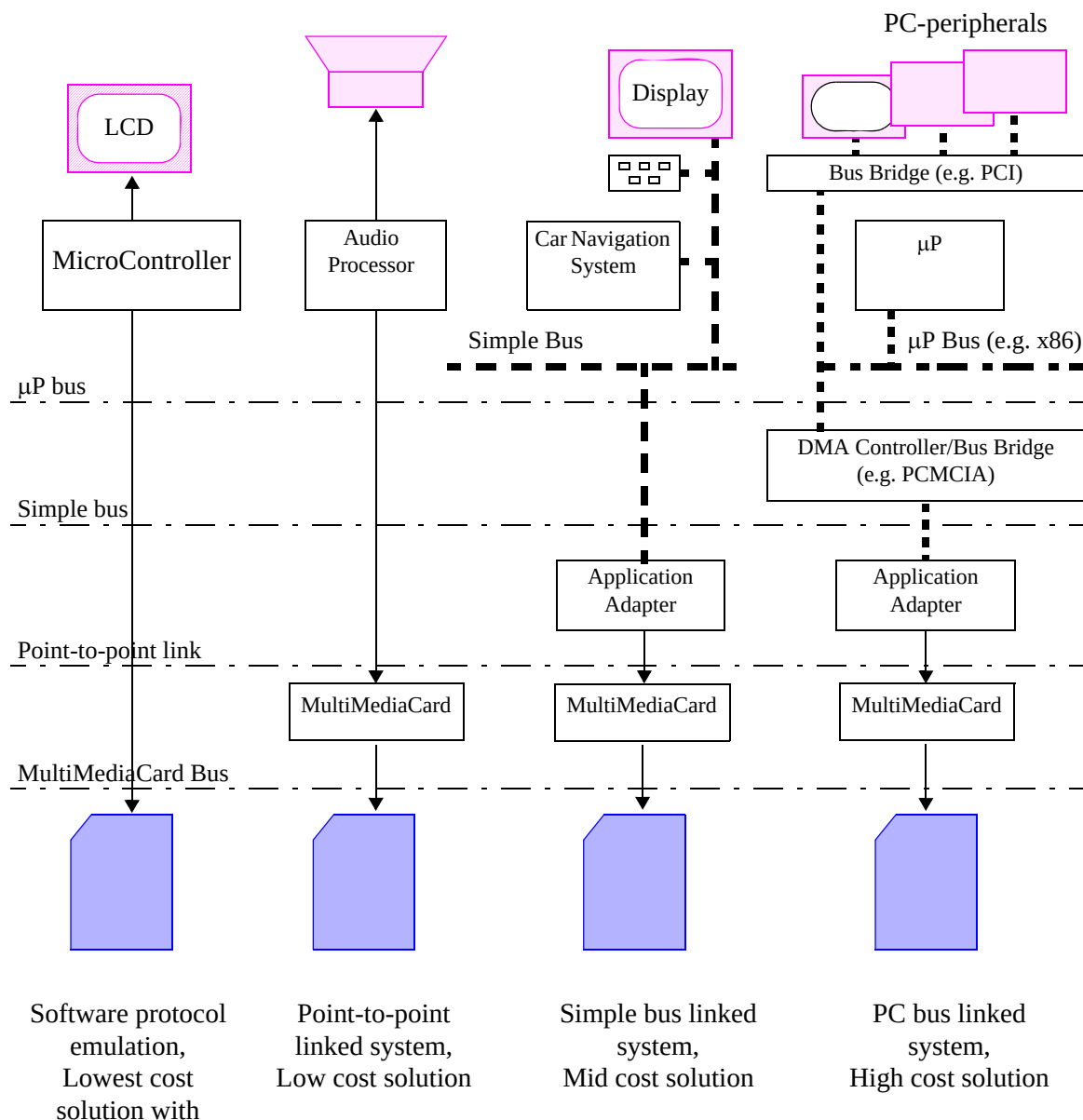


Figure 1 — Topology of MultiMediaCard systems

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.

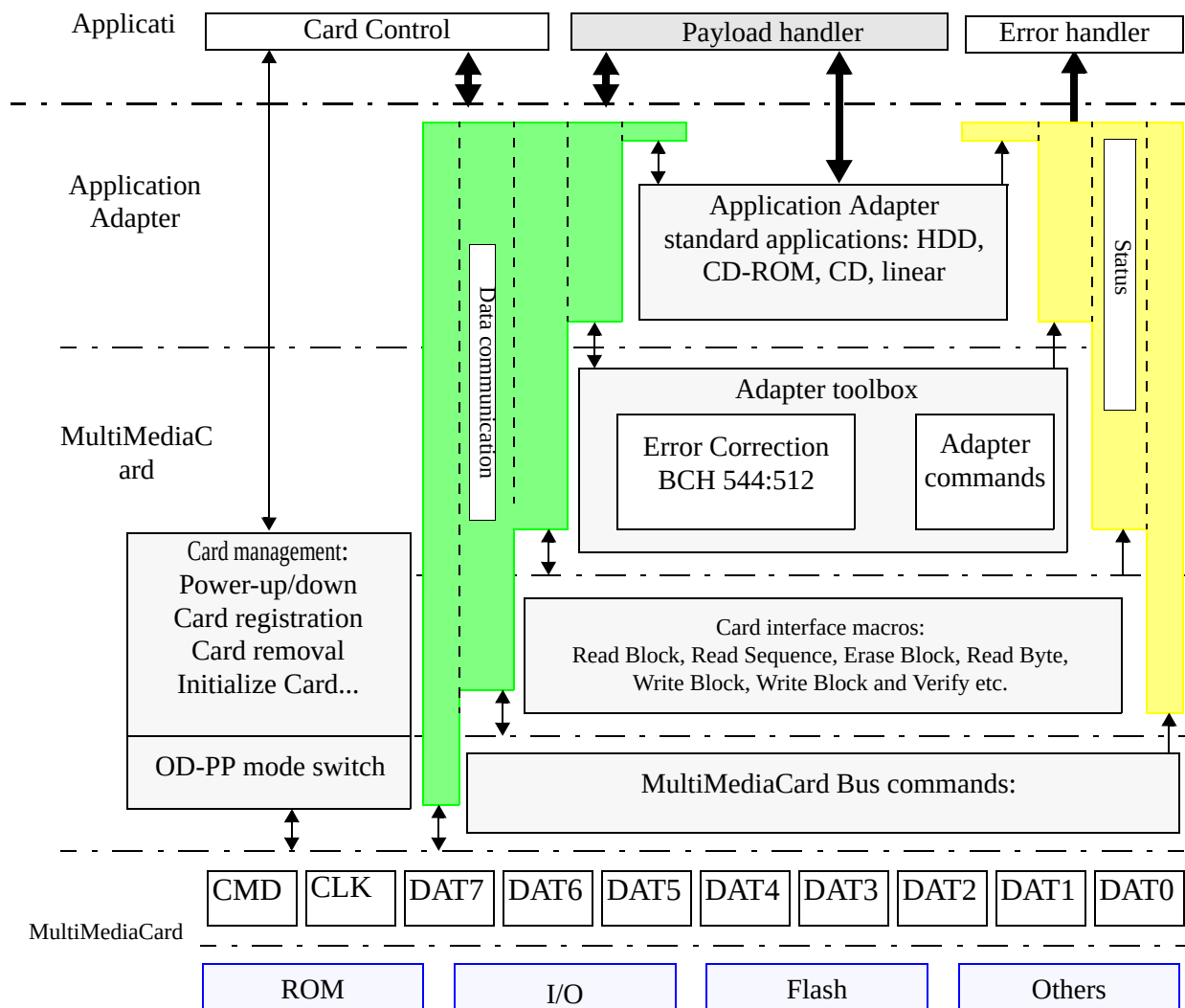


Figure 2 — MultiMediaCard system overview

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

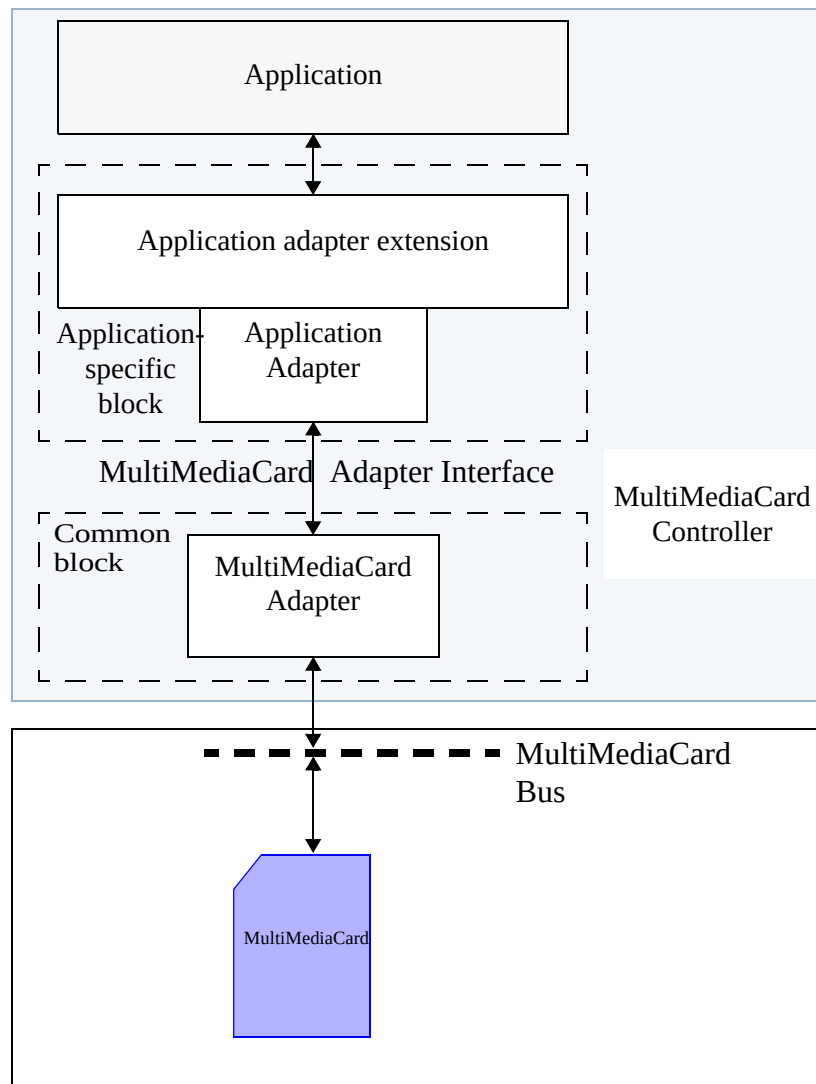


Figure 3 — MultiMediaCard system example

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_CS D	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 MMCplus and MMCmobile

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.

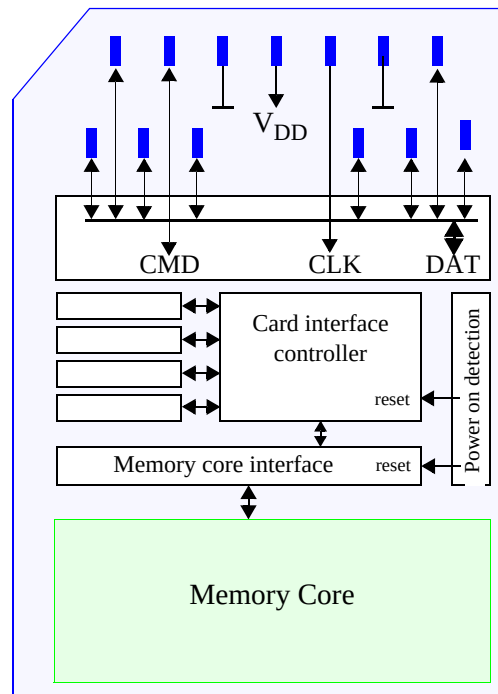


Figure 4 — MultiMediaCard architecture

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](#)).

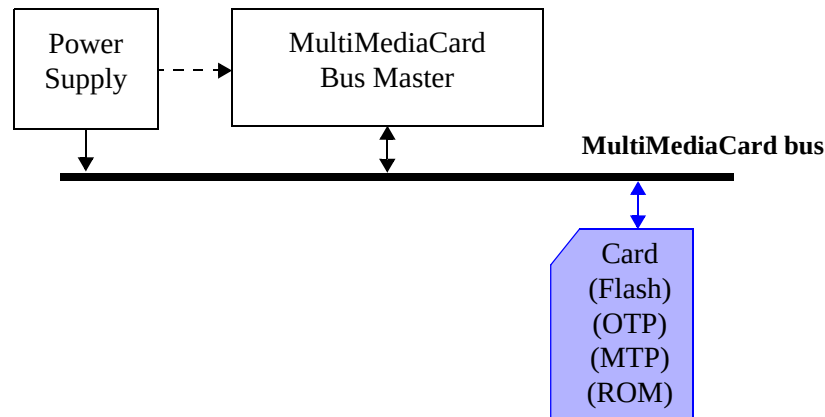


Figure 5 — MultiMediaCard bus system

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 a 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 a 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 commands²: These commands initiate a continuous data stream, they are terminated only when a stop command follows on the CMD line. This mode reduces the command overhead to an absolute minimum.
- Block-oriented commands: These commands send a data block succeeded by CRC bits. Both read and write operations allow either single or multiple block transmission. A multiple block transmission is terminated when a stop command follows on the CMD line similarly to the sequential read.

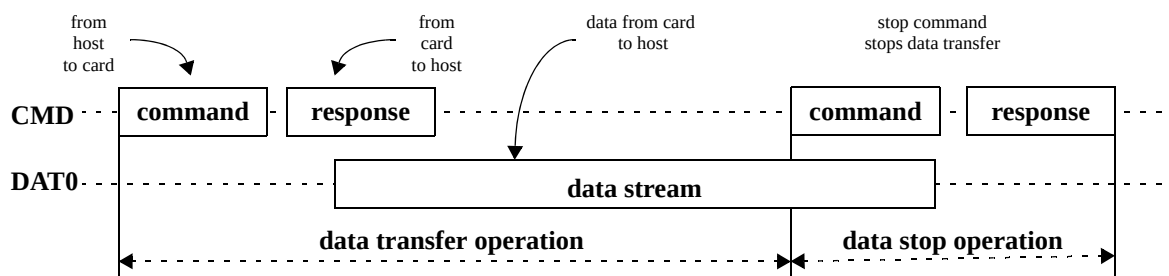


Figure 6 — Sequential read operation

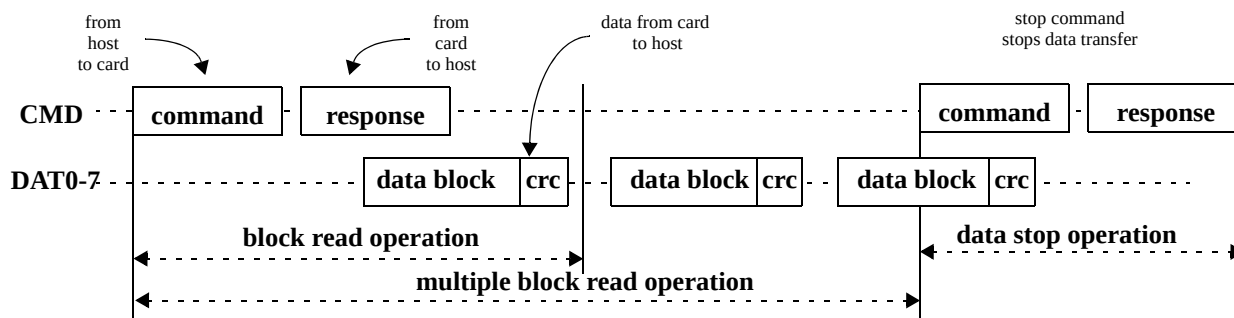


Figure 7 — (Multiple) Block read operation

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

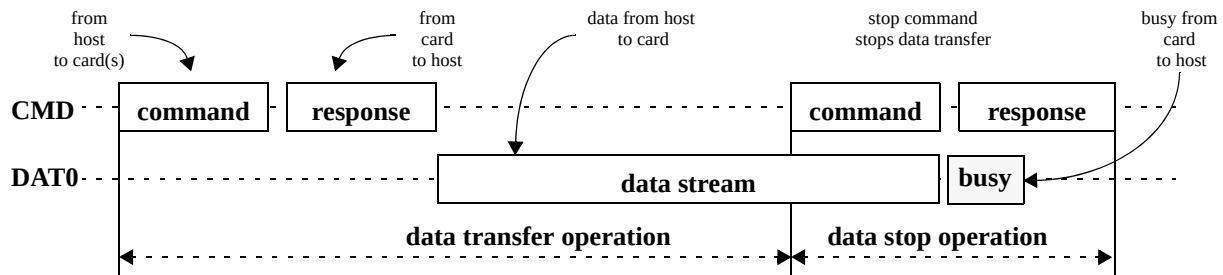


Figure 8 — Sequential write operation

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

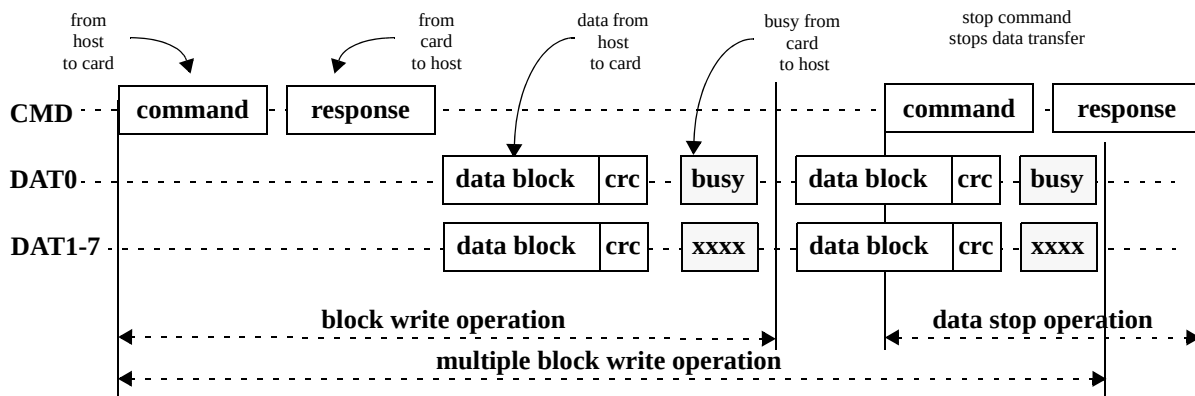


Figure 9 — (Multiple) Block write operation

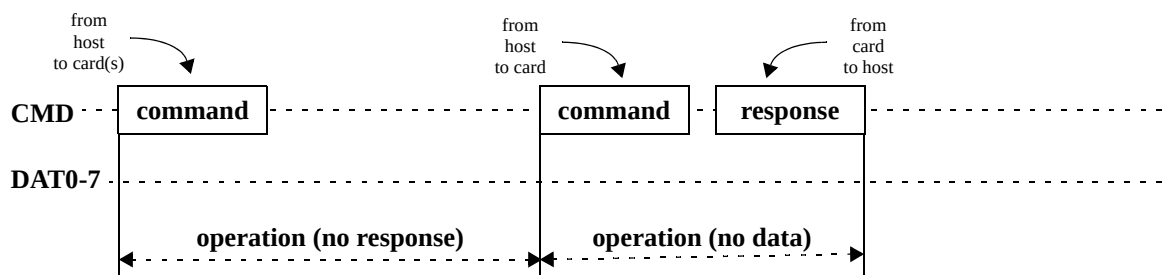


Figure 10 — “No response” and “No data” operations

Command tokens have the following coding scheme:

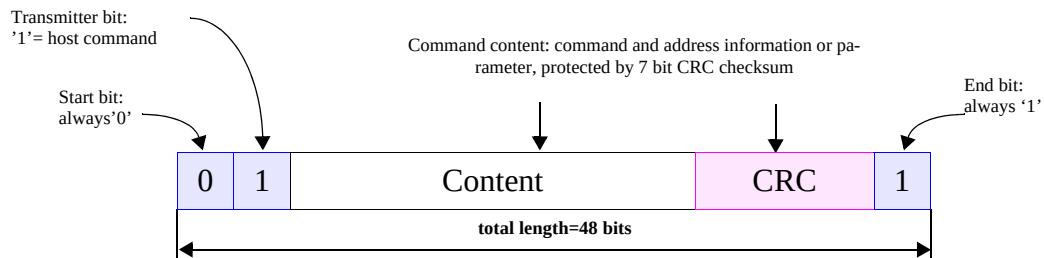


Figure 11 — Command token format

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

Response tokens have five coding schemes depending on their content. The token length is either 48 or 136 bits. The detailed 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](#).

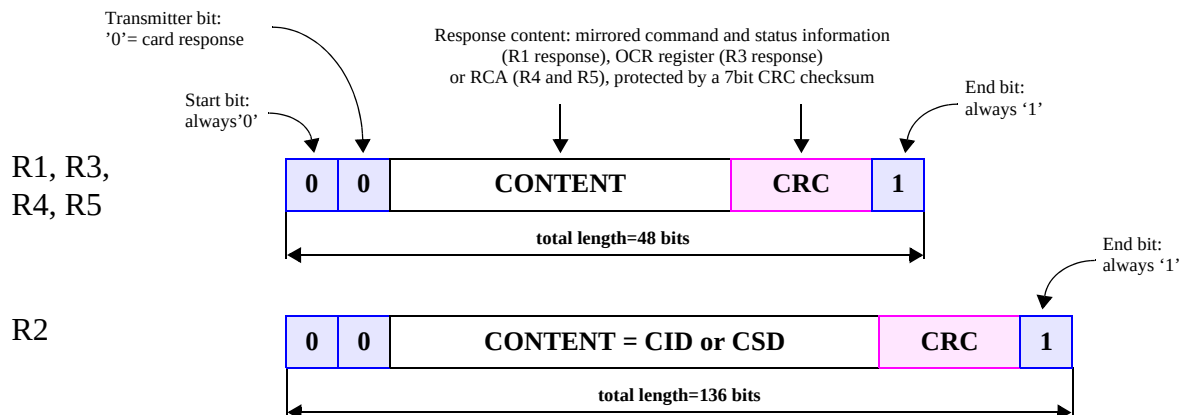
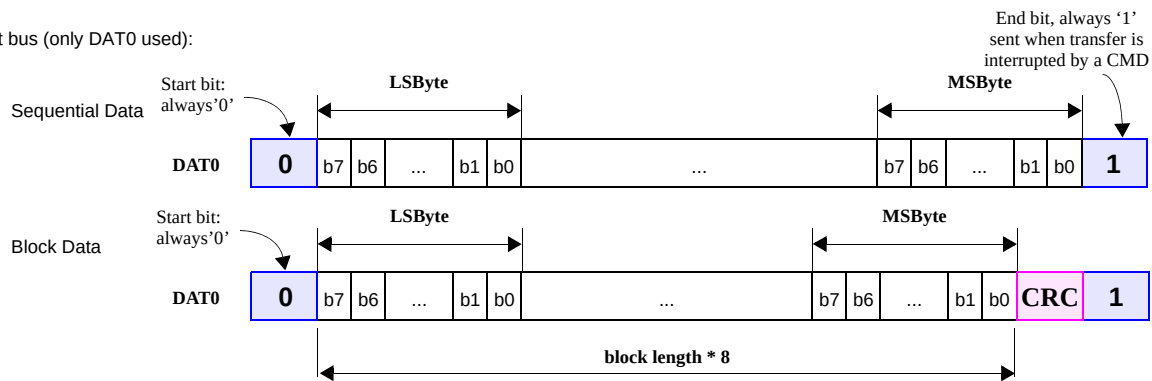
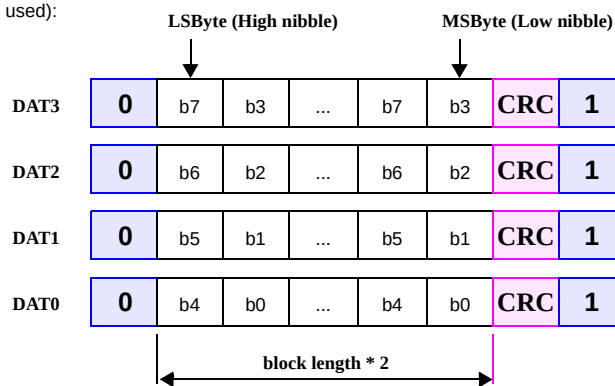


Figure 12 — Response token format

1 Bit bus (only DAT0 used):



4 Bits bus (DAT3-DAT0 used):



8 Bits bus (DAT7-DAT0 used):

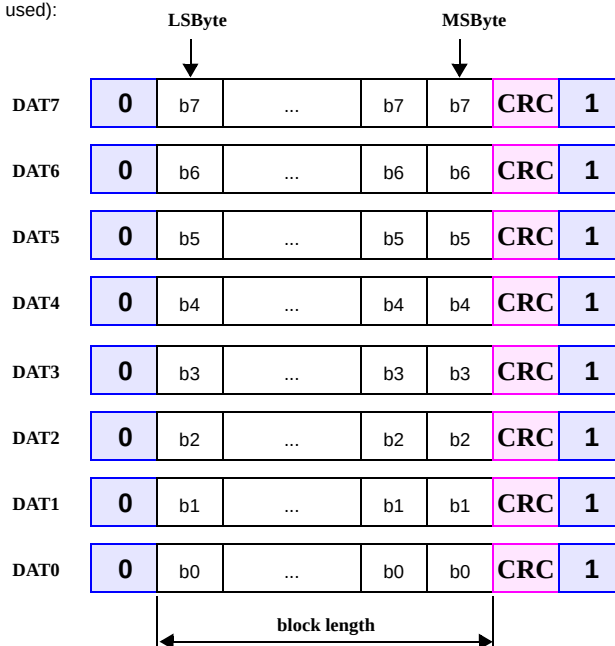


Figure 13 — Data packet format

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:

- 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 diagram illustrates the MultiMediaCard Controller architecture. It features a central light blue box labeled "MultiMediaCard Controller". Inside this box, there are two white rectangular blocks: "Application Adapter" on the left and "MultiMediaCard Adapter" on the right. These two blocks are connected by a horizontal double-headed arrow labeled "Adapter Interface". To the left of the "Application Adapter" block, outside the controller box, is a horizontal double-headed arrow labeled "Application Interface". To the right of the "MultiMediaCard Adapter" block, outside the controller box, are three horizontal lines representing a bus. The top line is a double-headed arrow labeled "CMD". The middle line is a single-headed arrow pointing right, labeled "CLK". The bottom line is a double-headed arrow labeled "DAT[7:0]". Below these three lines, the text "MultiMediaCard Bus" is written.

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.

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.

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

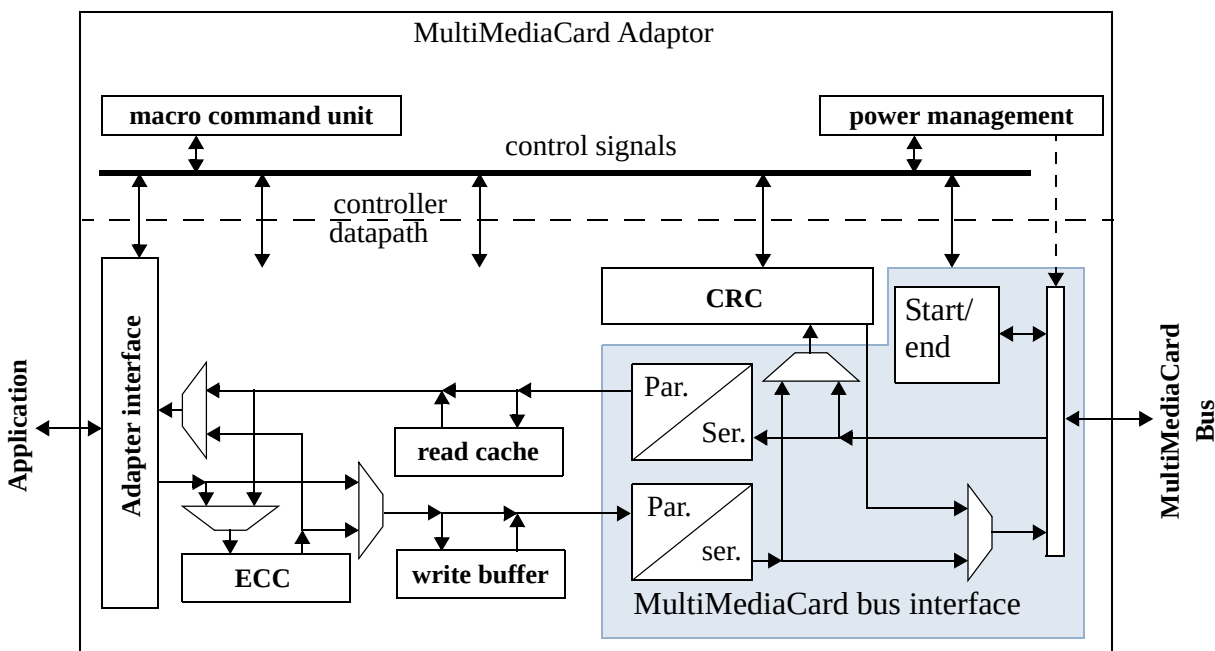


Figure 15 — MultiMediaCard adapter architecture

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 system³. 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](#).



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 $V_{DD} < 2.0V$ 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.

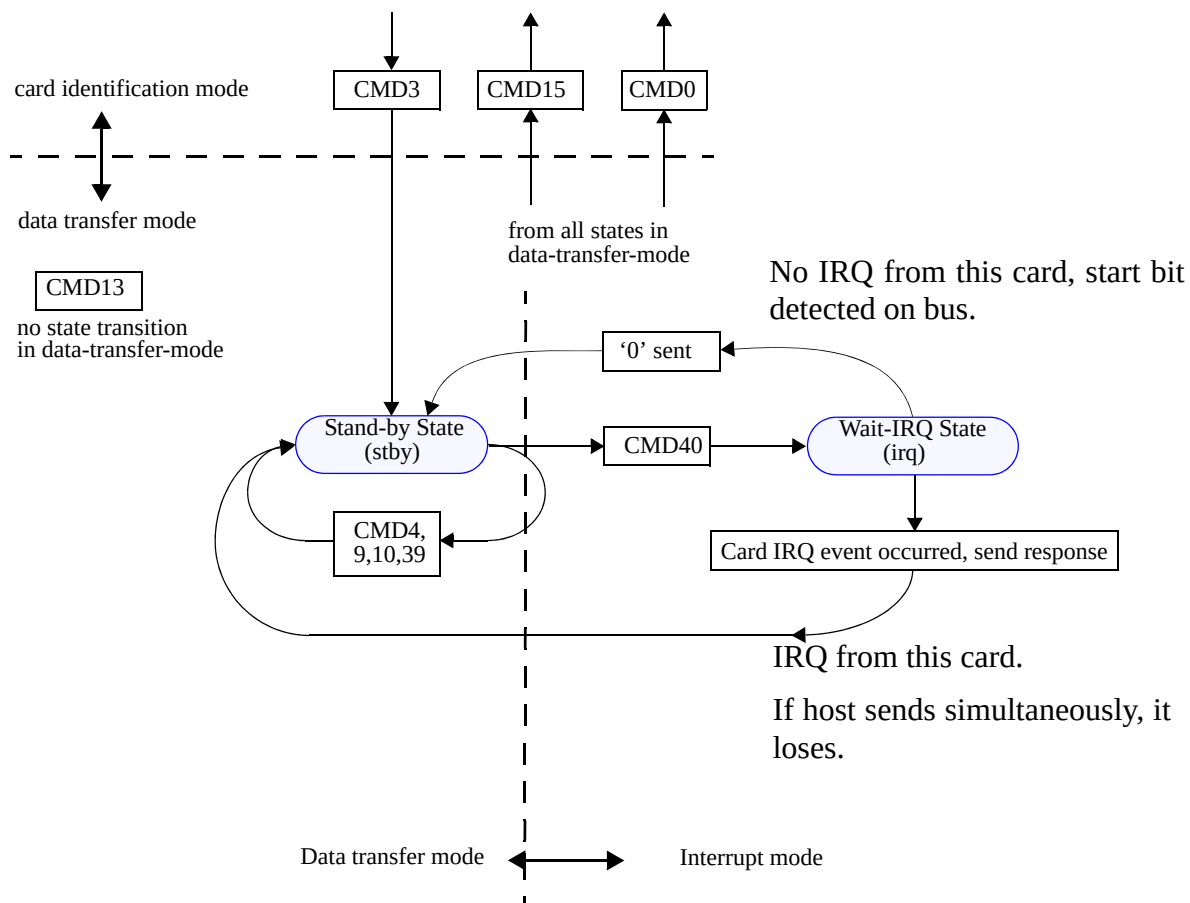


Figure 17 — MultiMediaCard state transition diagram, interrupt mode

- 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.

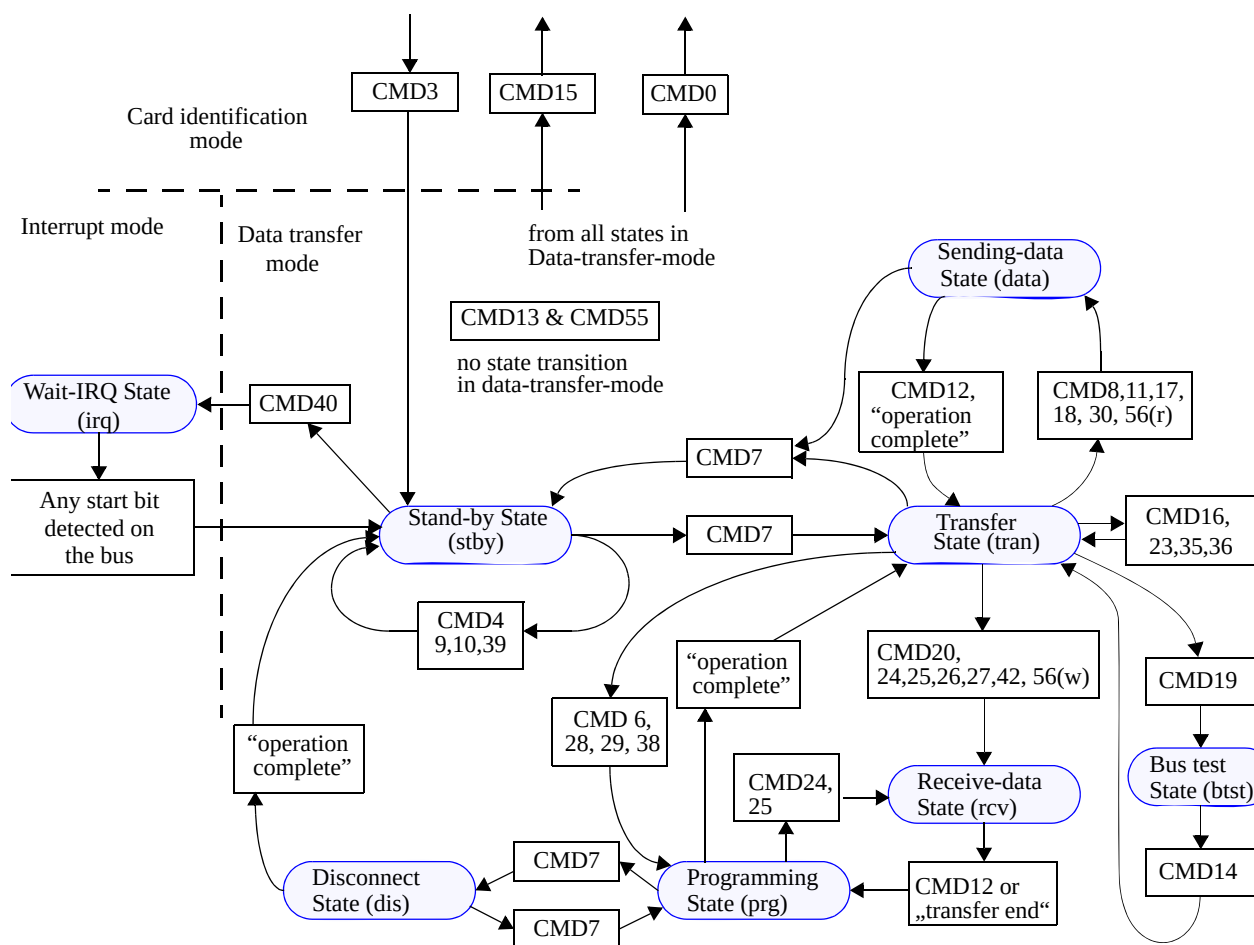


Figure 18 — MultiMediaCard state diagram (data transfer mode)

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 set-

ting one of the access modes. All three modes access and modify one of the EXT_CSD bytes, the byte pointed by the Index field⁴.

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_TRANSMISSION 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 constraints 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 an 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_TRANSMISSION 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.