The MultiMediaCard

System Summary

Based on System Specification Version 3.1

MMCA Technical Committee

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

Table of contents

1 General Description

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

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

This document gives a general overview of the MultiMediaCard system architecture. A detailed description can be found in "MultiMediaCard System Specification Version 3.1", Official Release, March 2001.

The document is split up into several portions. Chapter 3 gives a general overview of the system components - card, bus and host. In the next two chapters the card registers and the MultiMediaCard bus are described. An introduction to the SPI mode is given in Chapter 6. The error protection techniques employed in this standard are described in Chapter 7. For achieving high data interchangeability, three basic file formats are introduced in Chapter 8 as valid file formats for the MultiMediaCard. Finally, the standard compliance criteria for the cards and hosts are given in the last chapter.

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

2 System Features

- **Targeted for portable and stationary applications**
- **System Voltage (Vdd) Range:**

	High Voltage MultiMediaCard	Low Voltage MultiMediaCard
Communication	**2.0 - 3.6**	**1.65 - 3.6**
Memory Access	**Card specific (any sub-range of the communication voltage range)**	

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

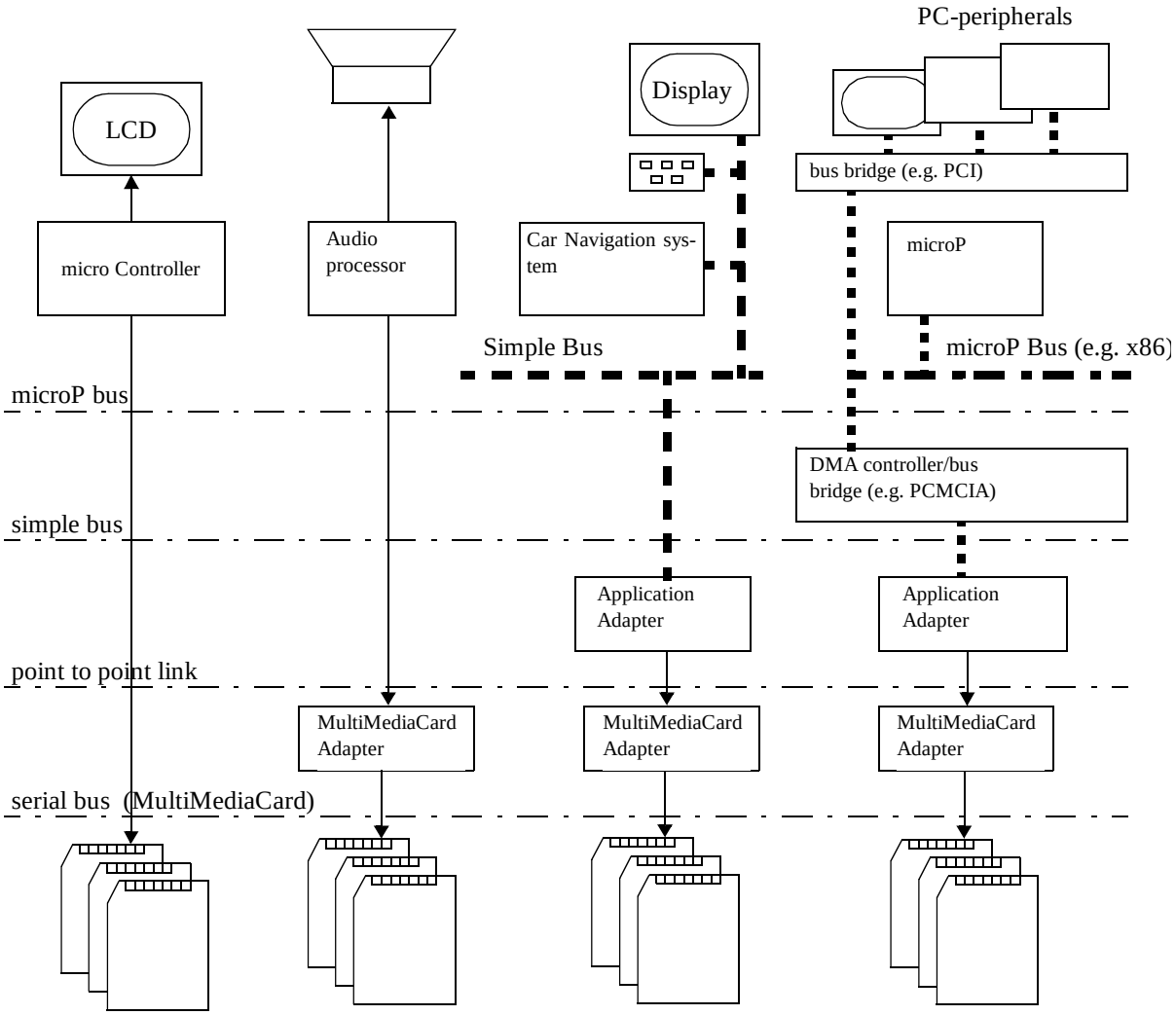
MultiMediaCard Mode	SPI Mode
Three-wire serial data bus (clock, command, data)	Three-wire serial data bus (clock, dataIn, dataOut) + card specific CS signal.
Up to 64k cards addressable by the bus protocol	Card selection via a hardware CS signal
Up to 30 cards stackable on a single physical bus	Card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

3 MultiMediaCard System Concept

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

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

Figure 1 shows four typical architectures of possible MultiMediaCard systems:



software protocol emulation := Lowest cost solution with reduced

point to point linked system :=

simple bus linked system :=

PC bus linked system :=

Figure 1: Topology of MultiMediaCard systems

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

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

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

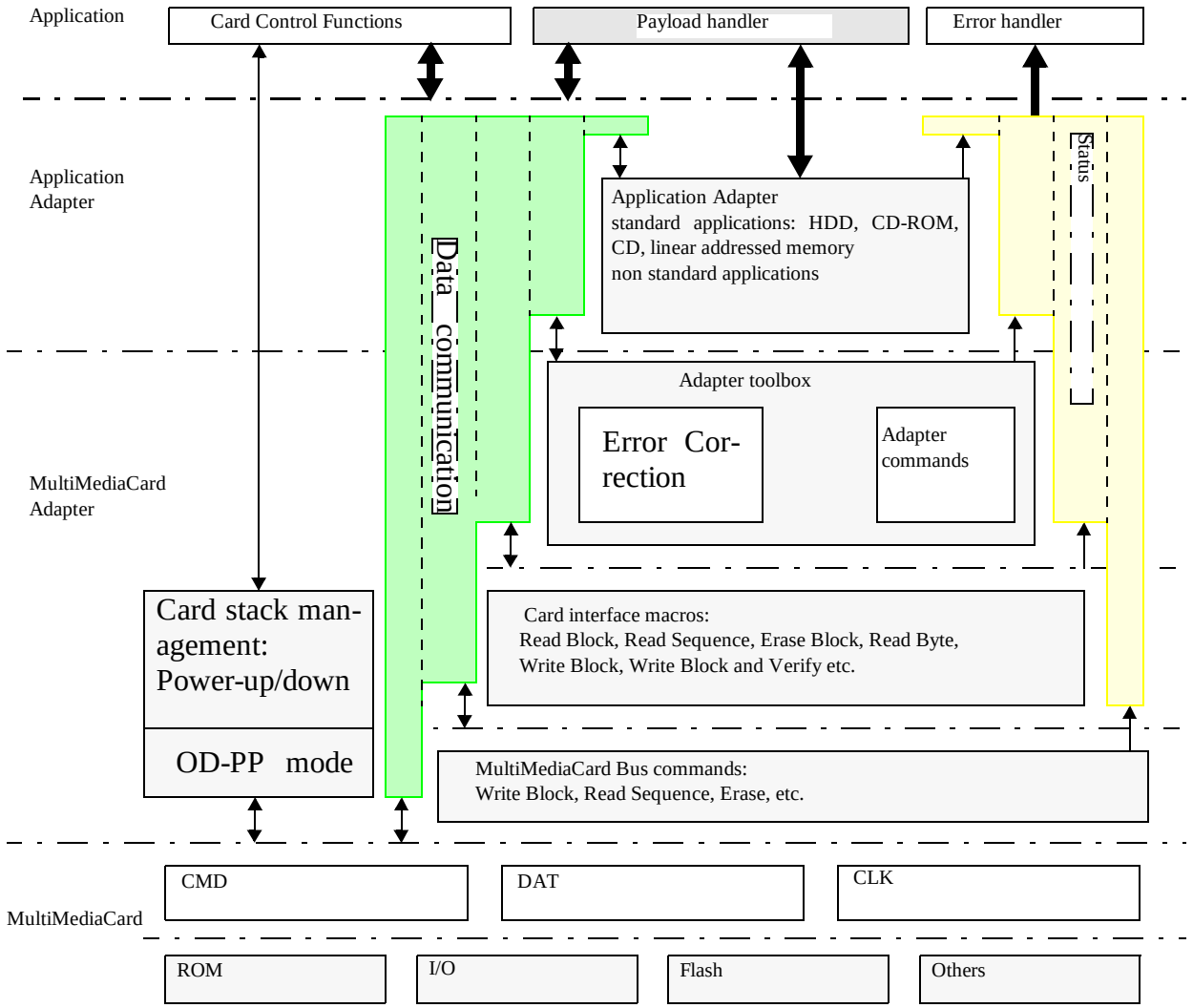


Figure 2: MultiMediaCard system overview

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

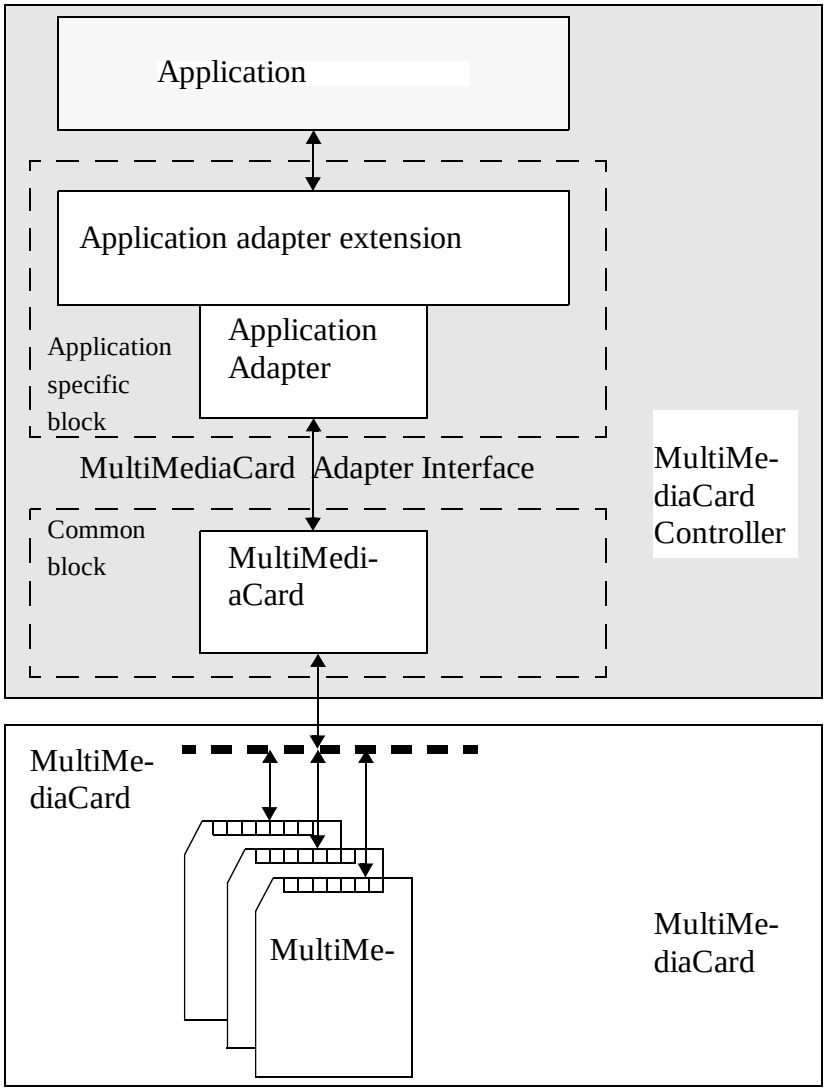


Figure 3: MultiMediaCard system example

This MultiMediaCard system contains at least two components:

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

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

- Application specific block (e.g. a microprocessor or an adapter to a standard bus like USB or ATA) = Application adapter

- Performs application oriented tasks (e.g. display controlling or input decoding for hand-held applications).

- Typically connected as a bus slave for a standard bus

- The common block = MultiMediaCard adapter

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

3.1 Card Concept

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

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

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

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

The MultiMediaCard has the form factor 24mm x 32mm x 1.4mm.

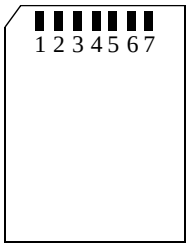


Figure 4: MultiMediaCard shape and interface (bottom view)

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

Pin No.	Name	Type[1]	Description
1	RSV	NC	Reserved for future use
2	CMD	I/O/PP/OD	Command/Response

Table 1: MultiMediaCard pad definition

Pin No.	Name	Type[1]	Description
3	V_{SS1}	S	Supply voltage ground
4	V_{DD}	S	Supply voltage
5	CLK	I	Clock
6	V_{SS2}	S	Supply voltage ground
7	DAT[2]	I/O/PP	Data

Table 1: MultiMediaCard pad definition

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

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

Each card has a set of information registers (see also Chapter 4):

Name	Width	Description
CID	128	Card identification number, card individual number for identification. Mandatory.
RCA	16	Relative card address, local system address of a card, dynamically assigned by the host during initialization. Mandatory.
DSR	16	Driver stage register to configure the card's output drivers. Optional.
CSD	128	Card specific data, information about the card operation conditions. Mandatory
OCR	32	Operation condition register for cards which do not support the full voltage range. Used by a special broadcast command to detect restricted cards. Optional.

Table 2: MultiMediaCard registers

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

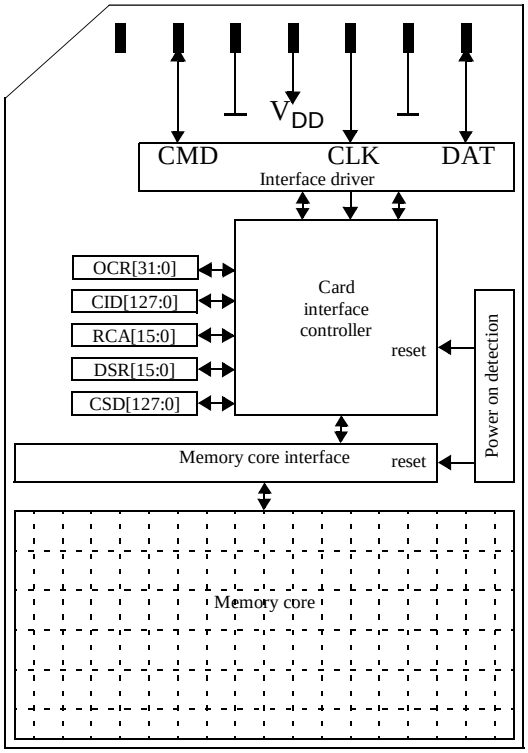


Figure 5: MultiMediaCard architecture

3.2 Bus Concept

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

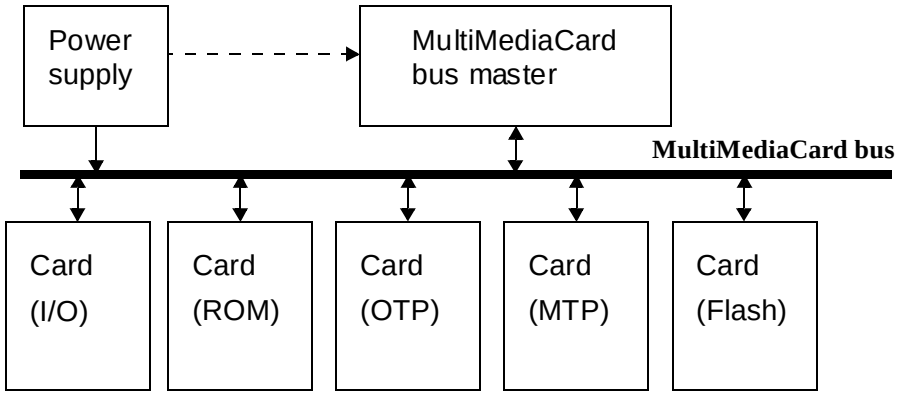


Figure 6: MultiMediaCard bus system

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

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

3.2.1 Bus Lines

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

The bus lines can be divided into three groups:

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

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

3.2.2 Bus Protocol

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

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

Card addressing is implemented using a session address assigned during the initialization phase, by the bus controller to all currently connected cards. Individual cards are identified by their CID number. This method requires that every card will have an unique CID number. To ensure uniqueness of CIDs the CID register contains 24 bits (MID and OID fields - see Chapter 4) which are defined by the MMCA. Every card manufacturers is required to apply for an unique MID (and optionally OID) number.

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

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

Two types of data transfer commands are defined:

- Sequential commands: These commands initiate a continuous data stream, they are terminated only when a stop command follows on the CMD line. This mode reduces the command overhead to an absolute minimum.
- Block-oriented commands: These commands send a data block succeeded by CRC bits.

Both read and write operations allow either single or multiple block transmission. A multiple block transmission is terminated when a stop command follows on the CMD line similarly to the sequential read.

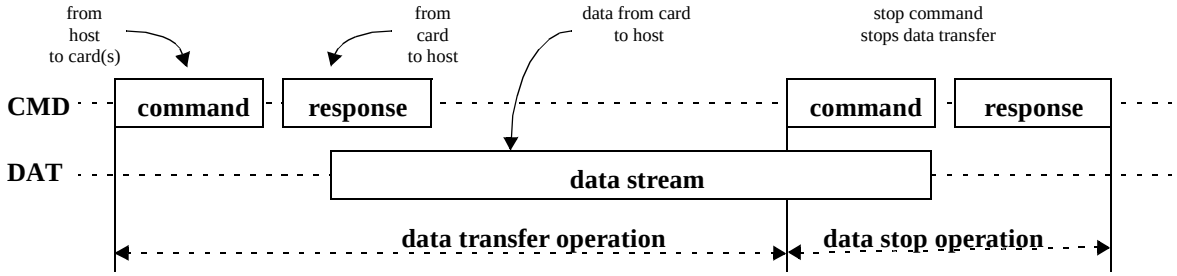


Figure 7: Sequential read operation

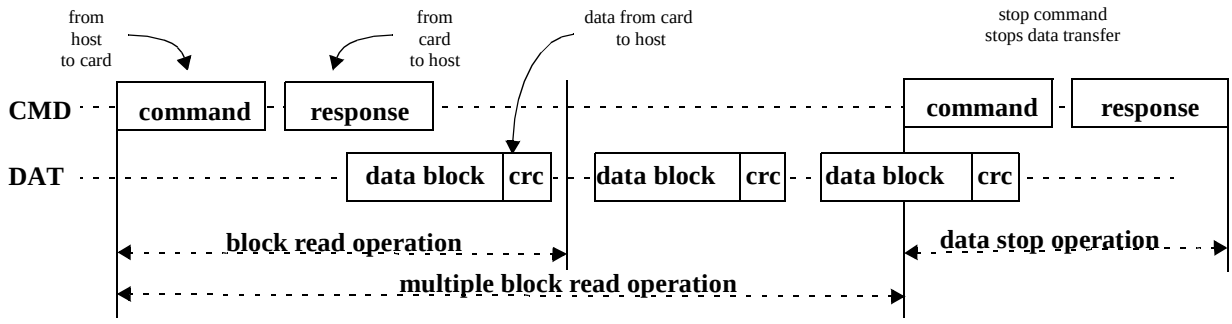


Figure 8: (Multiple) Block read operation

.

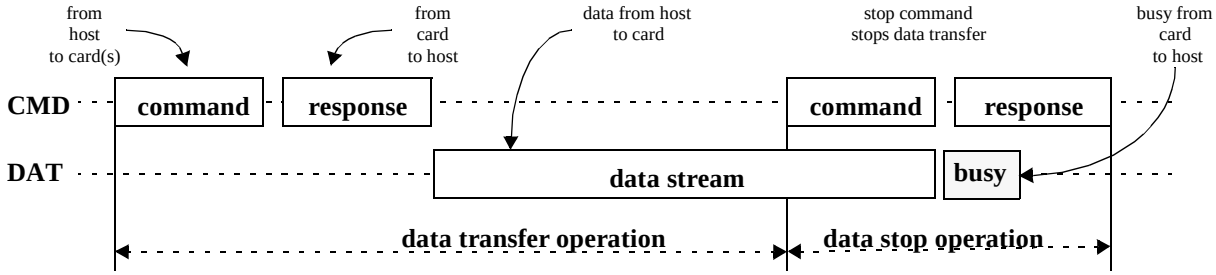


Figure 9: Sequential write operation

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

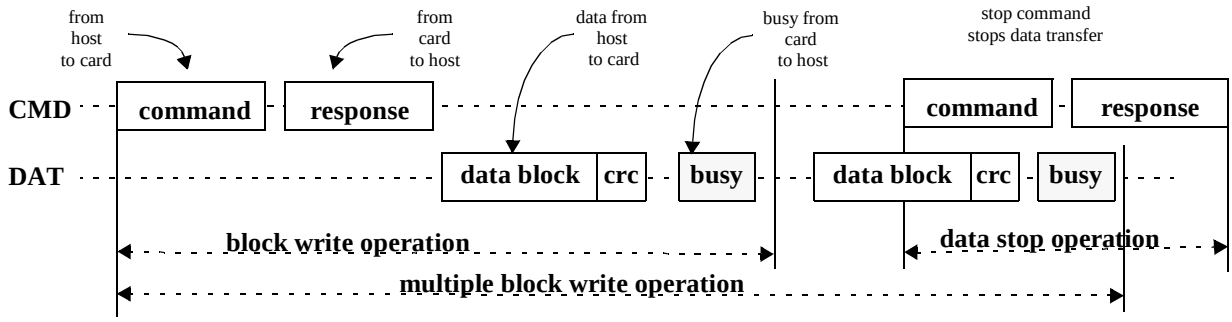


Figure 10: (Multiple) Block write operation

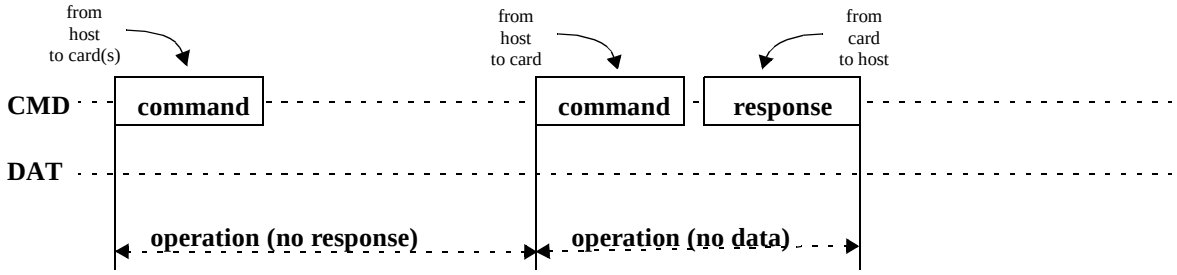


Figure 11: ? no response? and ? no data? operations

Command tokens have the following coding scheme:

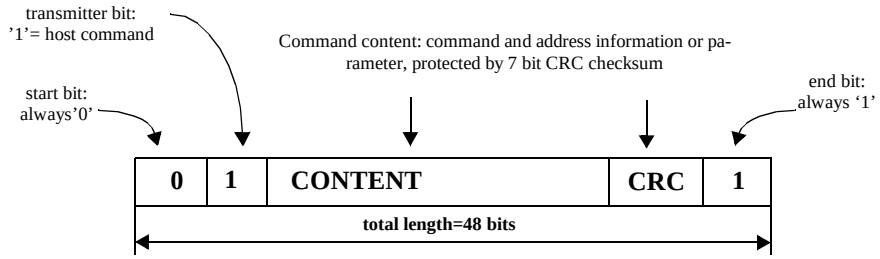


Figure 12: Command token format

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

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

Due to the fact that there is no predefined end in sequential data transfer, no CRC protection is

included in this case. The CRC protection algorithm for block data is a 16 bit CCITT polynomial. All used CRC types are described in Chapter 7.

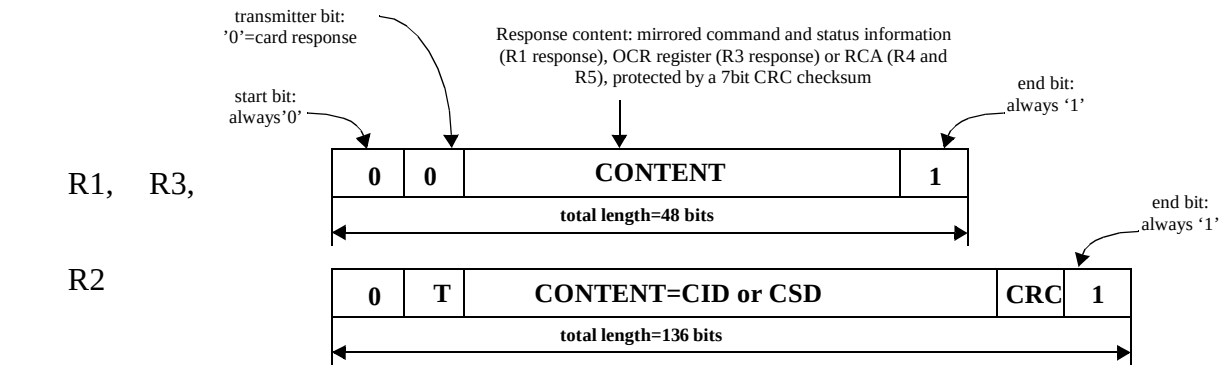


Figure 13: Response token format

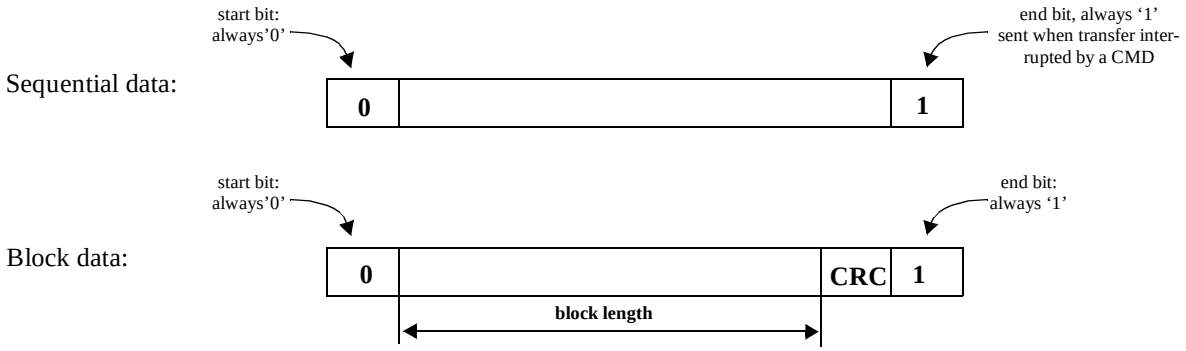


Figure 14: Data packet format

3.3 Controller Concept

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

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

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

- The application adapter: the application oriented part
- The MultiMediaCard adapter: the MultiMediaCard oriented part

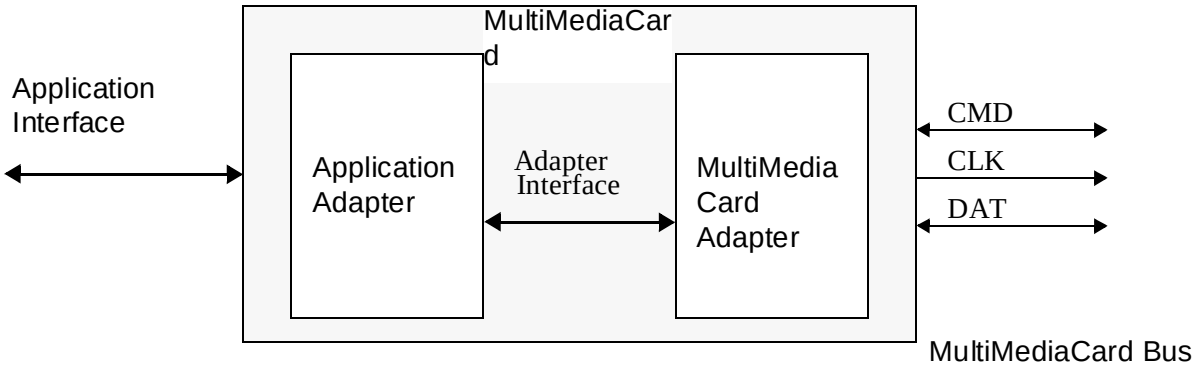


Figure 15: MultiMediaCard controller scheme

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

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

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

3.3.1 Application Adapter Requirements

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

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

3.3.2 MultiMediaCard Adapter Architecture

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

- The controller: macro unit, stack management and power management

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

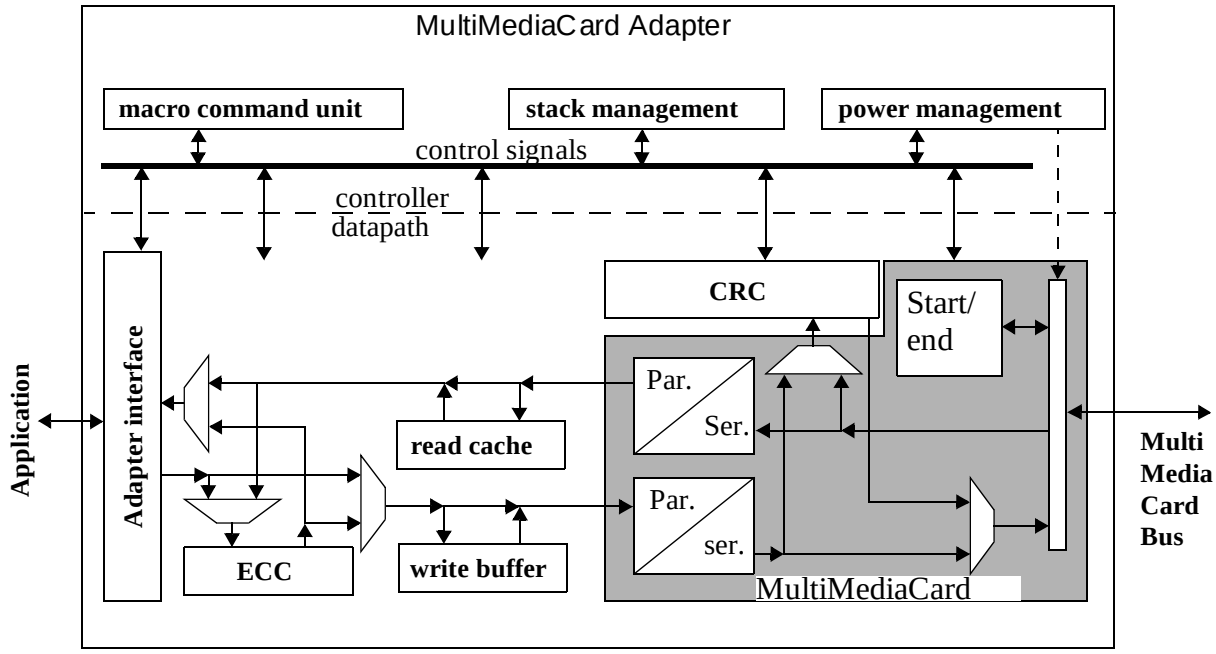


Figure 16: MultiMediaCard adapter architecture

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

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

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

4 Card Registers

Within the card interface five registers are defined: OCR, CID, CSD, RCA and DSR. These can be accessed only by corresponding commands.

The 32-bit operation conditions register (OCR) stores the V_{DD} voltage profile of the card. The register is optional and can be read only.

The 128-bit wide CID register carries the card identification information (Card ID) used during the card identification procedure.

The 128-bit wide Card-Specific Data register (CSD) 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 16-bit relative card address register (RCA) 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 16-bit driver stage register (DSR) 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).

5 MultiMediaCard Bus

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

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

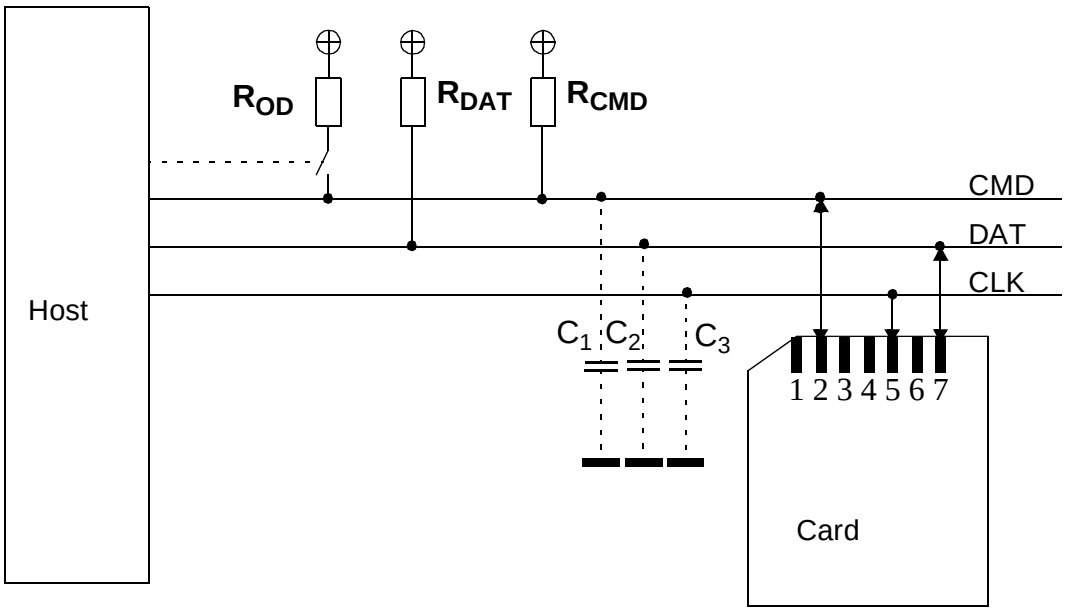


Figure 17: Bus circuitry diagram

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

A constant current source can replace the R_{OD} by achieving a better performance (constant slopes for the signal rising and falling edges). If the host does not allow the switchable R_{OD} implementation, a fixed R_{CMD} can be used. 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 allowed value.

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.

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.

6 SPI Mode

6.1 Introduction

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

The SPI standard defines the physical link only, and not the complete data transfer protocol. The MultiMediaCard SPI implementation uses a subset of the MultiMediaCard protocol and command set. It is intended to be used by systems which require a small number of cards (typically one). 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 system versus MultiMediaCard (fewer cards, hardware CS per card etc.).

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

6.3 SPI Bus Topology

The MultiMediaCard 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 18).

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

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

The SPI interface uses the same 7 signals of the standard MultiMediaCard bus (see Table 3).

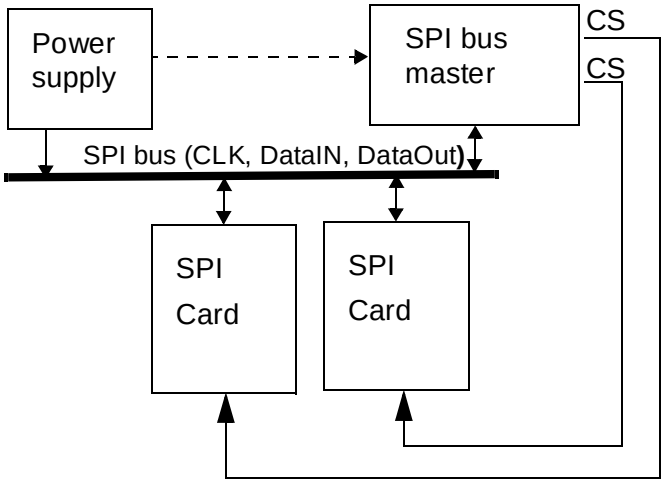


Figure 18: MultiMediaCard bus system

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

Table 3: SPI interface pin configuration

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

6.4 MultiMediaCard Registers in SPI Mode

The registers usage in SPI mode is summarized in Table 4. Most of them are inaccessible.

Name	Available in SPI mode	Width [Bytes]	Description
CID	Yes	16	Card identification data (serial number, manufacturer ID etc.)
RCA	No		
DSR	No		
CSD	Yes	16	Card specific data, information about the card operation conditions.
OCR	Yes		Operation condition register for cards which do not support the full voltage range. Optional.

Table 4: MultiMediaCard registers in SPI mode

6.5 SPI Electrical Interface and Bus Operating Conditions

The electrical interface is identical to MultiMediaCard mode with the exception of the programmable card output drivers option which is not supported in SPI mode.

The bus operating conditions and the bus timing are identical to the MultiMediaCard mode.

7 Error Protection

The CRC is intended for protecting MultiMediaCard commands, responses and data transfer against transmission errors on the MultiMediaCard bus. One 7-bit CRC is generated for every command and checked for every response on the CMD line. For data blocks one 16-bit CRC per transferred block is generated.

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 shortened BCH (542,512) code was chosen as one suitable code for matching the requirement of having high efficiency at lowest costs, leading to a redundancy of 5.9%.

8 File Formats for the MultiMediaCard

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

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

8.1 Hard Disk-like File System with Partition Table

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

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

Table 5: Partition table for hard disk-like file system

Every partition entry consists of the following fields:

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

Table 6: Partition entry description

Byte position	Length (bytes)	Entry description	Value / Range
0x8	4	first sector position relative to beginning of device	number of first sector (linear address)
0xc	4	Number of sectors in partition	between 1 and maximal number of sectors on device

Table 6: Partition entry description

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

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

Byte position	Length (bytes)	Entry description	Value / Range
0x0	3	Jump command	0xeb 0xXX 0x90
0x3	8	OEM name	XXX
0xb	2	Bytes / sector	512
0xd	1	Sectors / cluster	XXX (range: 1 - 64)
0xe	2	Reserved sectors (Number of reserved sectors at the beginning of the media including the boot sector)	1
0x10	1	Number of FAT? s	2
0x11	2	Number of root directory entries	512
0x13	2	Number of sectors on media	XXX (depends on card capacity, if the media has more than 65535 sectors, this field is zero and the 'number of total sectors' is set)
0x15	1	Media descriptor	0xf8 (hard disk)
0x16	2	Sectors / FAT	XXX
0x18	2	Sectors / track	32 (no meaning)
0x1a	2	Number of heads	2 (no meaning)
0x1c	4	Number of hidden sectors	0
0x20	4	Number of total sectors	XXX (depends on capacity)
0x24	1	Drive number	0
0x25	1	Reserved	0
0x26	1	Extended boot signature	0x29
0x27	4	Volume ID or serial number	XXX

Table 7: Boot sector configuration

Byte position	Length (bytes)	Entry description	Value / Range
0x2b	11	Volume label	XXX (ASCII characters padded with blanks if less than 11 characters)
0x36	8	File system type	XXX (ASCII characters identifying the file system type FAT12 or FAT16)
0x3e	448	Load program code	XXX
0x1fe	1	Signature	0x55
0x1ff	1	Signature	0xaa

Table 7: Boot sector configuration

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

8.2 DOS FAT File System without Partition Table

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

8.3 Universal File System for the MultiMediaCard

tbd.

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

9 MultiMediaCard Standard Compliance

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

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

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

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

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

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

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

Function	ROM card class	R/W card class	I/O card class
0-20 MHz transfer rate	sub-range allowed	sub-range allowed	sub-range allowed
2-3.6 volts power supply	sub-range allowed	sub-range allowed	sub-range allowed
CCC 0 basic	mandatory	mandatory	mandatory
CCC 1 and 2 sequential and block read	one of the two mandatory, the other optional	one of the two mandatory, the other optional	optional

Function	ROM card class	R/W card class	I/O card class
CCC 3 and 4 sequential and block write	not required	one of the two mandatory, the other optional	optional
CCC 5 erase	not required	mandatory	not required
CCC 6 write protection functions	not required	mandatory	not required
CCC 7 lock card commands	optional	optional	optional
CCC 8 application specific commands	optional	optional	optional
CCC 9 interrupt and fast read/write	not required	optional	mandatory
DSR	optional	optional	optional
SPI Mode	optional	optional	optional
ECC generation and verification	optional	optional	not required

Remarks 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.

10 Abbreviations and terms

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