JEDEC STANDARD

Embedded Multi-Media Card (*e*•MMC) Electrical Standard (5.1)

JESD84-B51

(Revision of JESD84-B50.1, July 2014)

FEBRUARY 2015

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 refer to www.jedec.org under Standards and Documents for alternative contact information.

Published by
©JEDEC Solid State Technology Association 2015
3103 North 10th Street
Suite 240 South
Arlington, VA 22201-2107

PRICE: Contact JEDEC

Printed in the U.S.A.

EMBEDDED MULTI-MEDIA CARD (*e*•MMC) 5.1 DEVICE

Contents

Figures

Page

Tables

Foreword

This standard has been prepared by JEDEC and the MMC Association, hereafter referred to as MMCA. JEDEC took the basic MMCA specification and adopted it for embedded applications, calling it "*e*•MMC."

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

Introduction

The *e*•MMC is a managed memory capable of storing code and data. It is specifically designed for mobile devices. The *e*•MMC is intended to offer the performance and features required by mobile devices while maintaining low power consumption. The *e*•MMC device contains features that support high throughput for large data transfers and performance for small random data more commonly found in code usage. It also contains many security features.

e•MMC communication is based on an advanced 11-signal bus. The communication protocol is defined as a part of this standard and referred to as the *e*•MMC mode.

The *e*•MMC standard only covers embedded devices, however, the protocol and commands were originally developed for a removable Device. The spec has been updated to remove references to the removable Device but some functions remain to support backward compatibility.

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 implementer.

EMBEDDED MULTIMEDIA DEVICE PRODUCT STANDARD (*e*•MMC 5.1)

(From JEDEC Board Ballot JCB-15-01, formulated under the cognizance of the JC-64.1 Subcommittee on Electrical Specifications and Command Protocols.)

1 Scope

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

2 Normative reference

The following normative documents contain provisions that through reference in this text, constitutes 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. For undated references, the latest edition of the normative document referred to applies.

INCITS, T10 Committee, SPC-4, *SPSCSI Primary Commands*

3 Terms and definitions

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

Address Space Definitions:

Mapped Host Address Space: the area of the *e*•MMC device that can be accessed by a read command from the host software.

Private Vendor Specific Address Space: the area of the *e*•MMC device that cannot be accessed by a read command from the host software. It contains vendor specific internal management data. This data can be either loaded at manufacturing or generated during device operation e.g., Memory Vendor Firmware and mapping tables. It does not contain any data (or portion of data) that was sent from the host to the device.

Unmapped Host Address Space: the area of the *e*•MMC device that cannot be accessed by a read command from the host software. It excludes private vendor specific address space. It may contain old host data or copies of host data.

Block: A number of bytes, basic data transfer unit

CID: Device IDentification register

CLK: Clock signal

CMD: Command line or *e*•MMC bus command (if extended CMDXX)

CRC: Cyclic Redundancy Check

3 Terms and definitions (cont'd)

CSD: Device Specific Data register

DAT: Data line

Data Strobe: Return Clock signal used in HS400 mode

DISCARD: This is a performance command that allows the host to identify regions that aren't needed. It does not require action from the device.

NOTE For data removal see TRIM.

DDR: Dual data rate.

DSR: Driver Stage Register

D-V_{DD}: Positive supply voltage for a cache memory

D-V_{SS}: Positive supply voltage ground for a cache memory

D-V_{DDQ}: Positive supply voltage for a cache memory

D-V_{SSQ}: Positive supply voltage ground for a cache memory

***e*•MMC:** embedded MultiMediaCard (The Cache feature is optional and only supports a single VDDi pin)

***e*^{2}•MMC:** An *e*•MMC device that supports the *e*•MMC Cache feature and three VDDi pins.

Empty Task Queue: A state when no tasks, ready or pending, are queued in the device's task queue.

ERASE: Block erase operation that does not require actual physical NAND erase operation

Flash: A type of multiple time programmable nonvolatile memory

Group: A number of write blocks, composite erase and write protect unit

HS200: High Speed interface timing mode of up to 200 MB/s at 200 MHz Single Date Rate Bus, 1.8 V or 1.2 V I/Os

HS400: High Speed DDR interface timing mode of up to 400 MB/s at 200 MHz Dual Date Rate Bus, 1.8 V or 1.2 V I/Os

ISI: InterSymbol Interference (referred to certain Noise type)

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

Non-Persistent: A part of the storage device that may lose contents after a power cycle

MSB, LSB: 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.

QSR: Queue Status Register

3 Terms and definitions (cont'd)

RCA: Relative Device Address register

Reset: CMD0 with argument of 0x00000000 or 0xF0F0F0F0, H/W reset (or CMD15)

ROM: Read Only Memory

RPMB: Replay Protected Memory Block

SSO: Simultaneous Switching Output (referred as certain type of Noise)

Secure Purge: The process operates on the addressable locations.. When the operation is executed it results in removing all the data from the unmapped host address space. This is done in accordance with the Secure Removal Type parameter value of the EXT_CSD. One or multiple write blocks or write protect groups depending on context.

NOTE The definition of secure purge is technology dependent (the definition above assumes NAND flash). Please refer to the http://www.killdisk.com/dod.htm or the following documents for more details.
DoD 5220.22M (http://www.dtic.mil/whs/directives/corres/html/522022m.htm)
and NIST SP 800-88 (http://csrc.nist.gov/publications/nistpubs/800-88/NISTSP800-88_rev1.pdf)

SQS: Send Queue Status. CMD13 sent with SQS bit set to 1 act as a query for QSR

stuff bit: Filling 0 bits to ensure fixed length frames for commands and responses

TAAC: Defines the time dependent factor of the data access time

three-state driver: A driver stage that has three output driver states: HIGH, LOW and high impedance (meaning that the interface does not have any influence on the interface level)

token: Code word representing a command

TRIM: A command that removes data from a write group. When TRIM is executed the region shall read as '0'. This serves primarily as a data removal command. (See Discard for performance command)

Tuning Process: A process commonly done by the host to find the optimal sampling point of a data input signals. The device may provide a tuning data block as specified for HS200 mode

UI: Unit Interval; It is one bit nominal time. For example, UI=5 ns at 200 MHz SDR, UI = 2.5 ns for 200MHz DDR.

UTC: Universal time coordinated

$\mathbf{V_{DD}}$: represents the common supply in case of a single supply device (V_{CC}=V_{CCQ}) or when related to consumed currents it represents the total consumed current for V_{CC} and V_{CCQ}.

$\mathbf{V_{SS}}$: Positive supply voltage ground for Core Device

$\mathbf{V_{CC}}$: Positive supply voltage for Core

$\mathbf{V_{CCQ}}$: Positive supply voltage for I/O

$\mathbf{V_{SSQ}}$: Positive supply voltage ground for I/O

Write Protection, Permanent: Write and erase prevention scheme, that once enabled, cannot be reversed.

Write Protection, Power-on: Write and erase prevention scheme, that once enabled, can only be reversed when a power failure event, that causes the device to reboot occurs, or the device is reset using the reset pin.

Write protection, Temporary: Write and erase prevention scheme that can be enabled and disabled.

4 System Features

The *e*•MMC device is a managed memory, that defines a mechanism for indirect memory accesses to the memory array. This indirect access is often enabled by a separate controller. The advantage of indirect memory access is that the memory device can perform several background memory management tasks without the involvement of the host software. This results in a simpler flash management layer on the host system.

The *e*•MMC device supports the following features:

- System Voltage (V_{CC} and V_{CCQ}) Ranges (Table 1).

Table 1 — *e*•MMC Voltage Modes

	High Voltage *e*•MMC	**Dual Voltage *e*•MMC[1]**
Communication (V_{CCQ})	2.7 – 3.6	1.1-1.3, 1.70-1.95, 2.7-3.6
Memory Access (V_{CC})	2.7 – 3.6	1.70-1.95, 2.7-3.6
NOTE 1 Refer to Table 199 — *e*•MMC voltage combinations for all the valid combinations of dual voltage devices.		

- Eleven-wire bus (clock, Data Strobe, 1 bit command, 8 bit data bus) and a hardware reset.
  - Clock frequencies of 0-200MHz
  - Three different data bus width modes: 1-bit (default), 4-bit, and 8-bit
- Data protection Mechanisms :
  - Password
  - Permanent
  - Power-On
  - Temporary
- Different types of error protected read and write modes:
  - Single Block
  - Multiple Block
- Data Removal Commands:
  - Erase
  - Trim
  - Sanitize
- Protection methods for data during sudden power failures
- Capability for customized solutions using application specific commands
- Power Saving Sleep mode

4 System Features (cont'd)

- Enhance host and device communication techniques to improve performance
  - Power Off Notification
  - High Priority Interrupt (HPI)
  - Background Operations
  - Partitioning
  - Enhanced Regions
  - Real Time Clock
  - Partition Attributes
  - Context management
  - System data tagging
  - Packed commands
  - Dynamic Device Capacity
  - Optional Volatile Cache
  - Cache Enhancement Barrier
  - Package Case Temperature
  - Command Queuing
  - Enhanced Strobe
  - Package Case Temperature
- Boot Areas that will automatically stream data when using defined boot modes.
- Signed access to a Replay Protected Memory Block.
- Two types of high capacity devices: small 512B sector devices and large 4KB sector devices.

5 *e*•MMC Device and System

5.1 *e*•MMC System Overview

The *e*•MMC specification covers the behavior of the interface and the device controller. As part of this specification the existence of a host controller and a memory storage array are implied but the operation of these pieces is not fully specified.

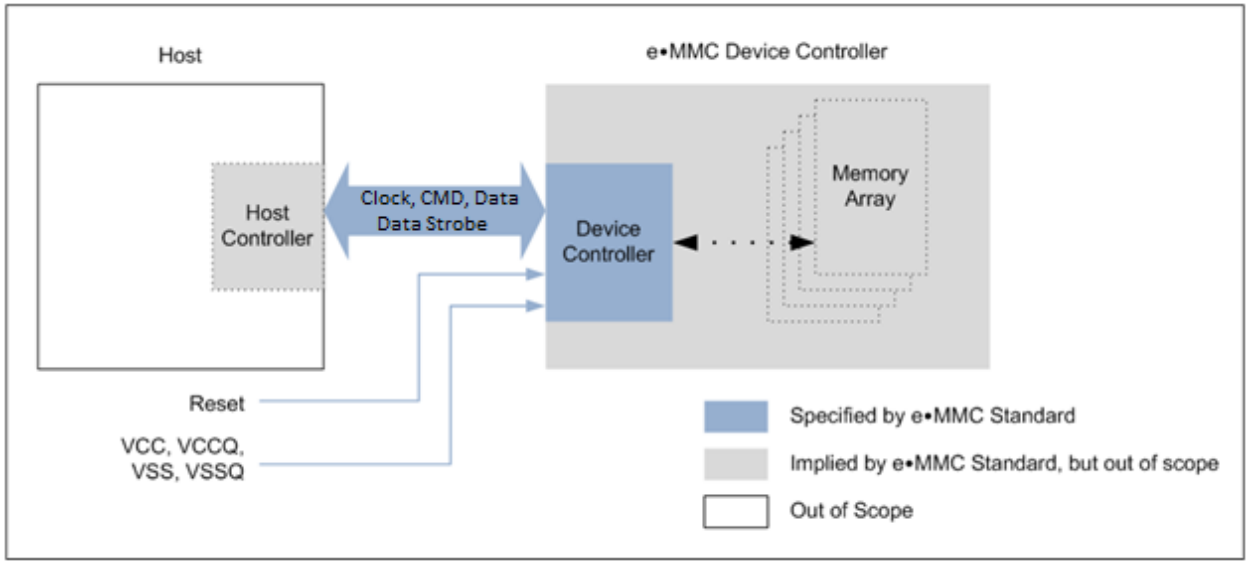


Figure 1 — *e*•MMC System Overview

5.2 Memory Addressing

Previous implementations of the *e*•MMC specification (versions up to v4.1) implemented byte addressing using a 32 bit field. This addressing mechanism permitted for *e*•MMC densities up to and including 2 GB.

To support larger densities the addressing mechanism was update to support sector addresses (512 B sectors). The sector addresses shall be used for all devices with capacity larger than 2 GB.

To determine the addressing mode use the host should read bit [30:29] in the OCR register.

5.3 *e*•MMC Device Overview

The *e*•MMC device 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 either a one bit (1x) or a two bits transfer (2x) on all the data lines. The frequency may vary between zero and the maximum clock frequency.
- **Data Strobe**: This signal is generated by the device and used for output in HS400 mode.
  The frequency of this signal follows the frequency of CLK. For data output each cycle of this signal directs two bits transfer(2x) on the data - one bit for positive edge and the other bit for negative edge. For CRC status response output and CMD response output (enabled only HS400 enhanced strobe mode), the CRC status and CMD Response are latched on the positive edge only, and don't care on the negative edge.
- **CMD**: This signal is a bidirectional command channel used for device 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 *e*•MMC host controller to the *e*•MMC device and responses are sent from the device to the host.

- **DAT0-DAT7**: These are bidirectional data channels. The DAT signals operate in push-pull mode. Only the device 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 *e*•MMC host controller. The *e*•MMC device includes internal pull-ups for data lines DAT1-DAT7. Immediately after entering the 4-bit mode, the device disconnects the internal pull ups of lines DAT1, DAT2, and DAT3. Correspondingly, immediately after entering to the 8-bit mode the device disconnects the internal pull-ups of lines DAT1–DAT7.

The signals on the *e*•MMC interface are described in Table 2.

Table 2 — *e*•MMC interface

Name	Type[1]	Description
CLK	I	Clock
DS	O/PP	Data Strobe
DAT0	I/O/PP	Data
DAT1	I/O/PP	Data
DAT2	I/O/PP	Data
DAT3	I/O/PP	Data
DAT4	I/O/PP	Data
DAT5	I/O/PP	Data
DAT6	I/O/PP	Data
DAT7	I/O/PP	Data
CMD	I/O/PP/OD	Command/Response
RST_n	I	Hardware reset
V_{CC}	S	Supply voltage for Core
V_{CCQ}	S	Supply voltage for I/O
V_{SS}	S	Supply voltage ground for Core
V_{SSQ}	S	Supply voltage ground for I/O

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

5.3 *e*•MMC Device Overview (cont'd)

Each device has a set of information registers (see also clause 7, Device Registers):

Table 3 — *e*•MMC registers

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

The host may reset the device by:

- Switching the power supply off and back on. The device shall have its own power-on detection circuitry that puts the device into a defined state after the power-on.
- A reset signal
- By sending a special command

5.3.1 Bus Protocol

After a power-on reset, the host must initialize the device by a special message-based *e*•MMC bus protocol. Each message is represented by one of the following tokens:

- command: a command is a token that starts an operation. A command is sent from the host to a device . A command is transferred serially on the CMD line.
- response: a response is a token that is sent from the device 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 device 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).

For each data lines, data can be transferred at the rate of one bit (single data rate) or two bits (dual data rate) per clock cycle.

Device addressing is implemented using a session address, assigned during the initialization phase, by the bus controller to the connected device. A device is identified by its CID number. This method requires the device to have a unique CID number. To ensure uniqueness of CIDs the CID register contains 24 bits (MID and OID fields, see 7.2) that are defined by JEDEC/MMCA. Every device manufacturer is required to apply for an unique MID (and optionally OID) number.

e•MMC bus data transfers are composed of command, response, and data block structure tokens. One data transfer is a *bus operation*. Operations always contain a command and a response token. In addition, some operations have a data token.

5.3.1 Bus Protocol (cont'd)

e•MMC commands are 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.

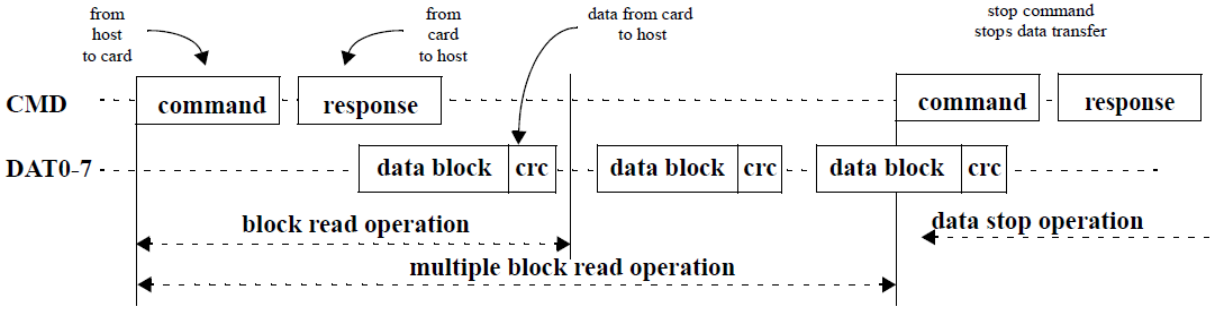


Figure 2 — Multiple-block read operation

The block write operation uses a simple busy signaling of the write operation duration on the data (DAT0) line. (See Figure 3)

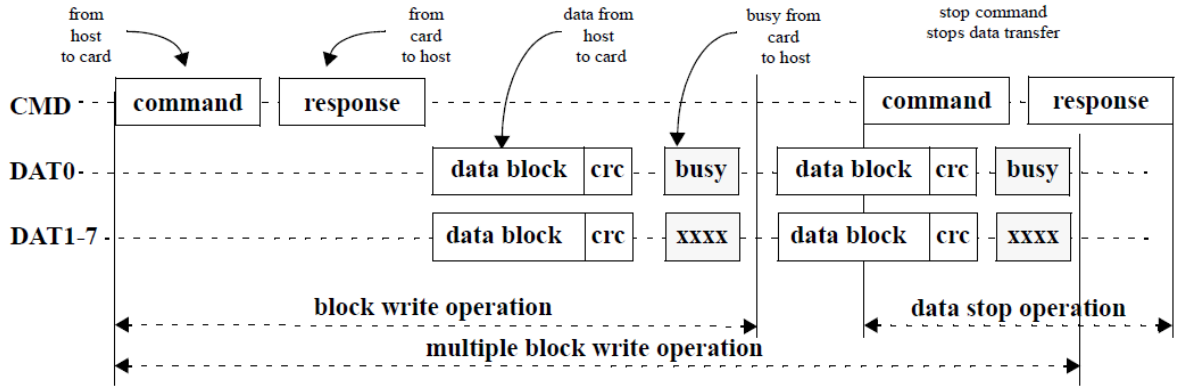


Figure 3 — (Multiple) Block write operation

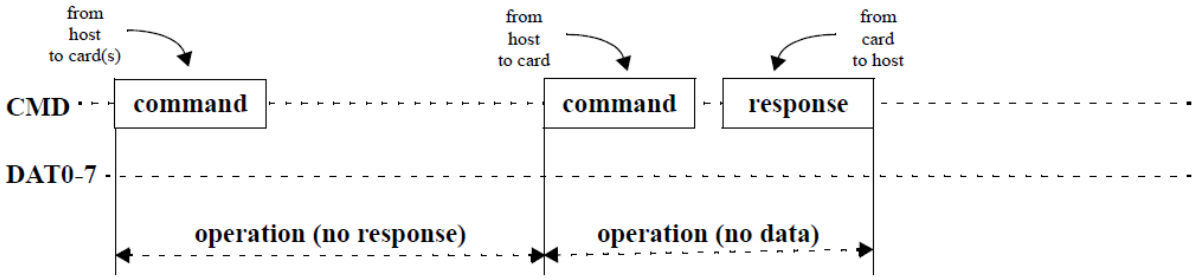


Figure 4 — No response" and "no data" operations

5.3.1 Bus Protocol (cont'd)

Command tokens have the following coding scheme:

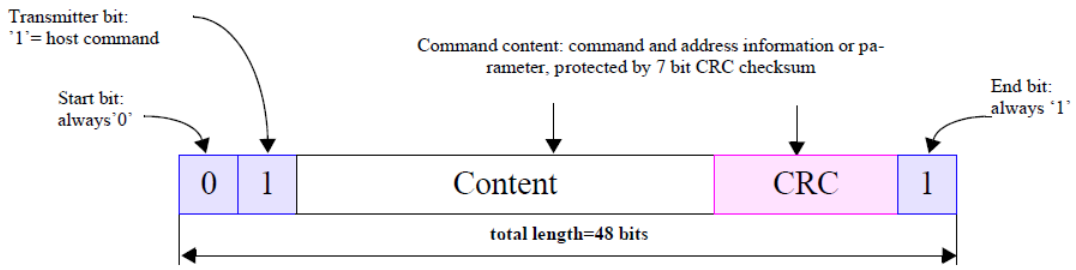


Figure 5 — Command token format

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

Response tokens have five coding schemes depending on their content. The token length is either 48 or 136 bits. The detailed command and response definitions are provided in 6.10 and 6.12.

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 0.

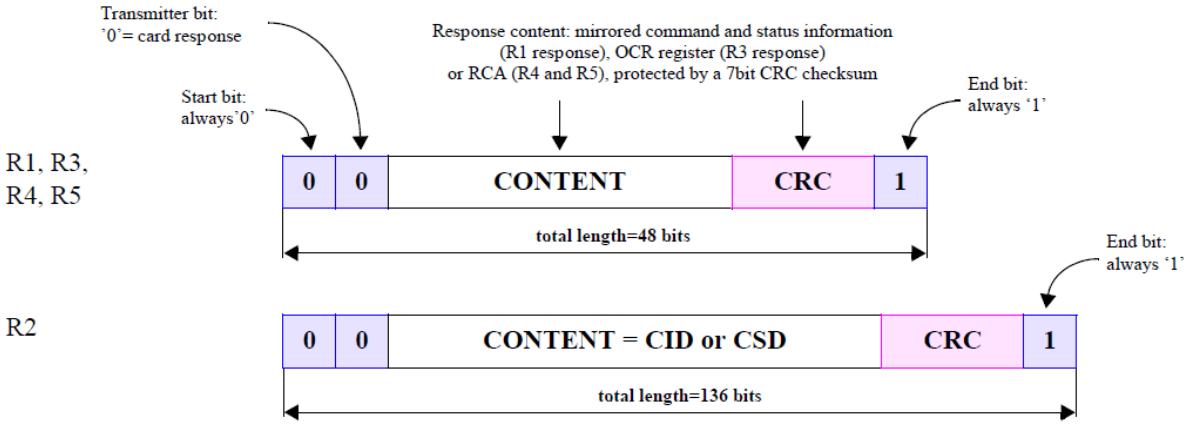


Figure 6 — Response token format

5.3.1 Bus Protocol (cont'd)

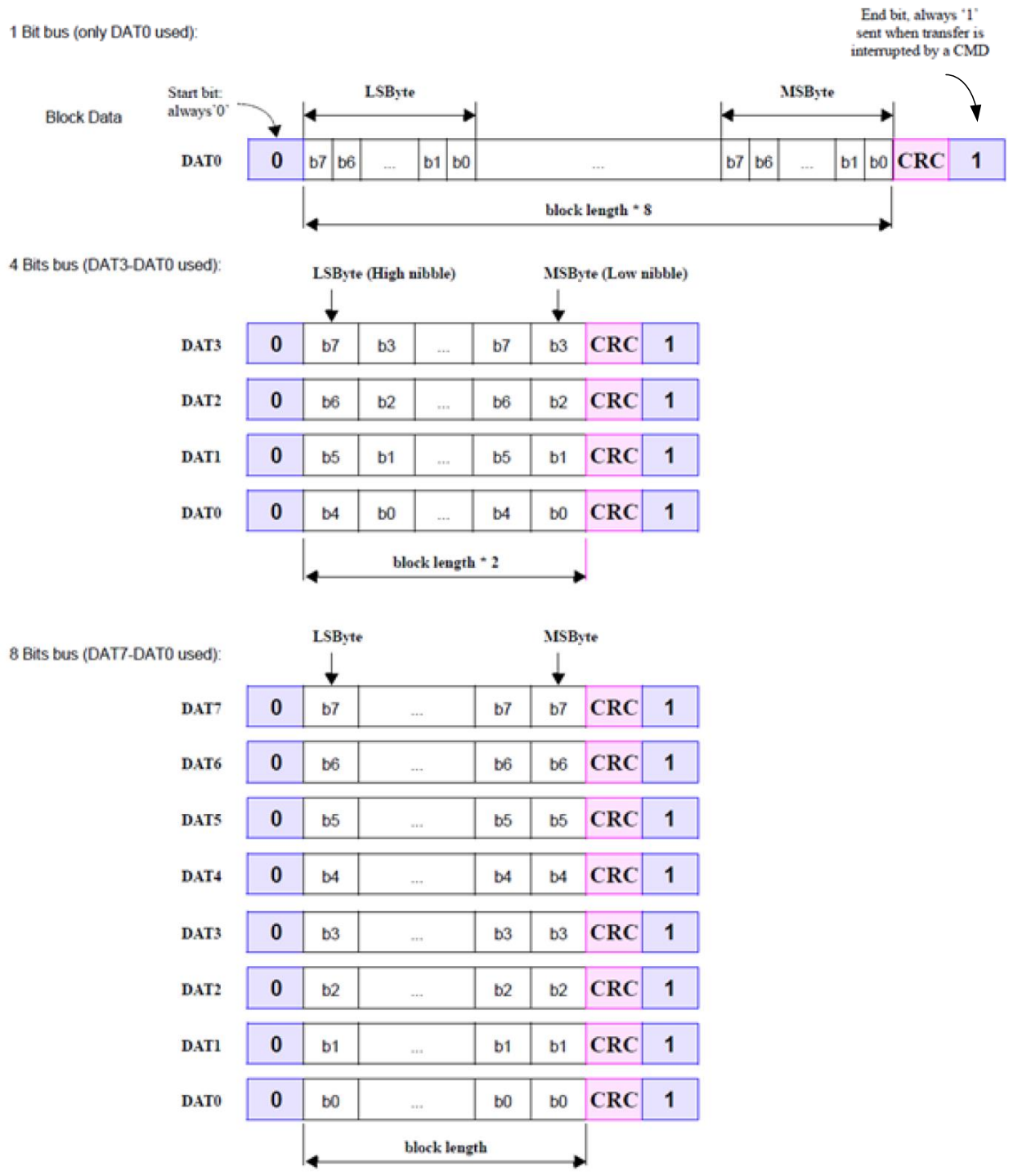
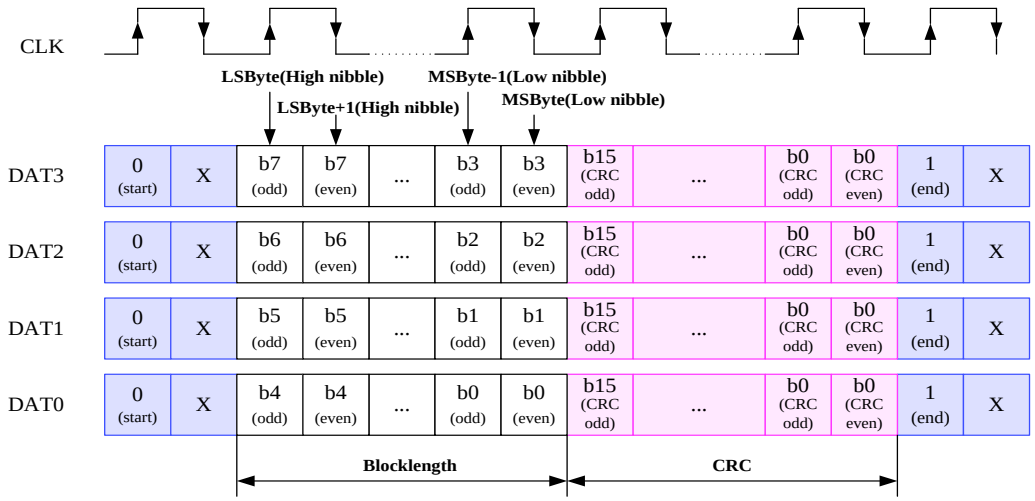


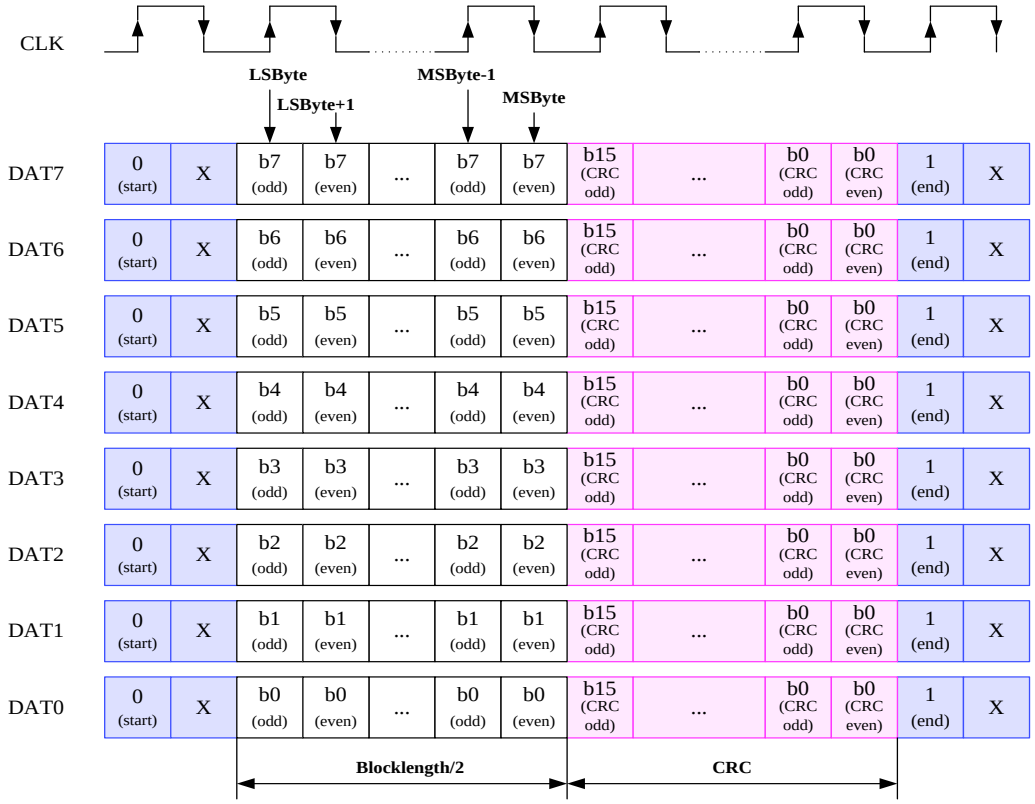
Figure 7 — Data packet format for SDR

5.3.1 Bus Protocol (cont'd)

4 Bits bus DDR (DAT3-DAT0 used):



8 Bits bus DDR (DAT7-DAT0 used):

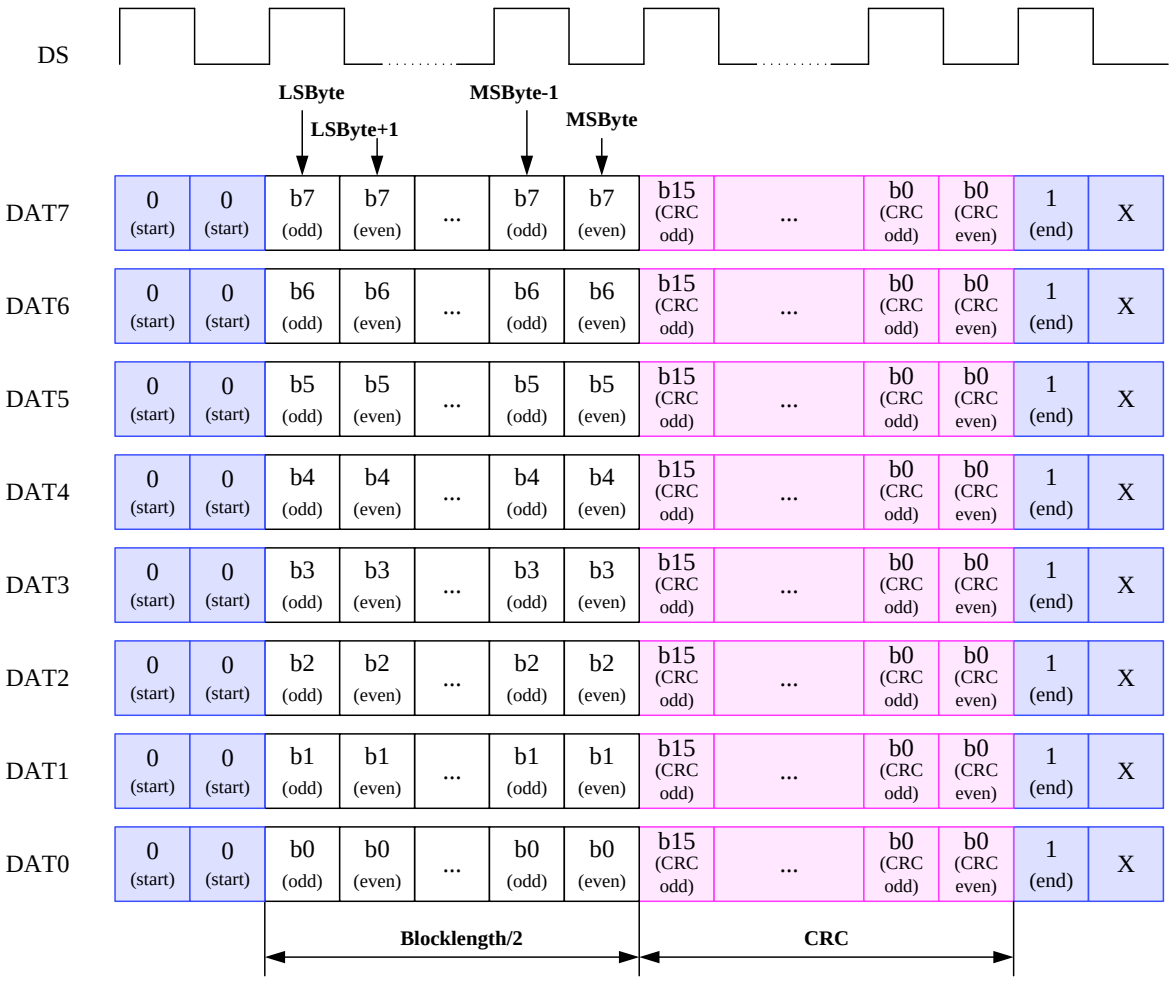


Notice that bytes data are not interleaved but CRC are interleaved
Start and end bits are only valid on the rising edge. ("x": undefined)

Figure 8 — Data packet format for DDR

5.3.1 Bus Protocol (cont'd)

8 Bits bus DDR for HS400 output (DAT7-DAT0 used):

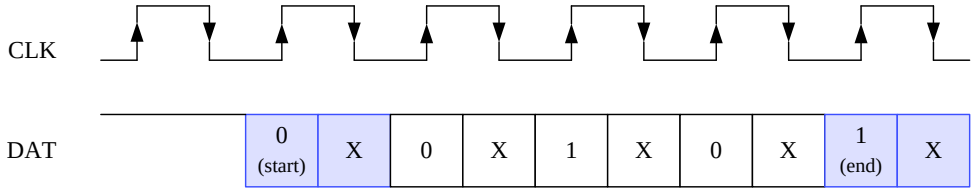


Notice that bytes data are not interleaved but CRC are interleaved
Start bits are valid when Data Strobe is High and Low.
End bits are only valid when Data Strobe is High. ("x": undefined)

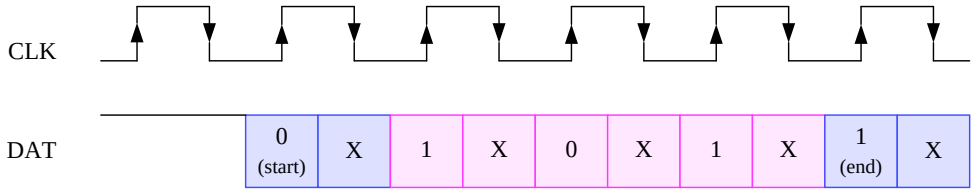
Figure 9 — Data packet format for DDR Read in HS400 mode

5.3.1 Bus Protocol (cont'd)

Positive CRC status token ('010')/Boot acknowledge pattern:



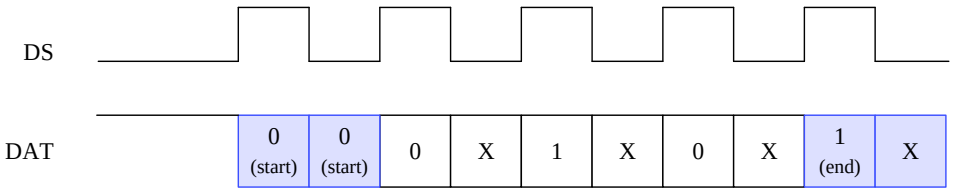
Negative CRC status token ('101'):



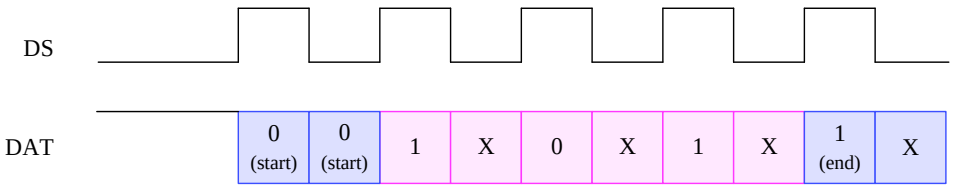
Start, end, CRC status and boot acknowledge bits are only valid on the rising edge ("x" is undefined)

Figure 10 — DDR52 CRC token

Positive CRC status token ('010'):



Negative CRC status token ('101'):



Start bits are valid when Data Strobe is High and Low.
CRC status and end bit are only valid when Data Strobe is High ("x" is undefined)

Figure 11 — HS400 CRC token

5.3.2 Bus Speed Modes

e•MMC defines several bus speed modes. Table 4 summarizes the various modes.

Table 4 — Bus Speed Modes

Mode Name	Data Rate	I/O Voltage	Bus Width	Frequency	Max Data Transfer (implies x8 bus width)
Backwards Compatibility with legacy MMC card	Single	3 V/1.8 V/ 1.2 V	1, 4, 8	0-26 MHz	26 MB/s
High Speed SDR	Single	3 V/1.8 V/ 1.2 V	1,4, 8	0-52 MHz	52 MB/s
High Speed DDR	Dual	3 V/1.8 V/ 1.2 V	4, 8	0-52 MHz	104 MB/s
HS200	Single	1.8 V/1.2 V	4, 8	0-200 MHz	200 MB/s
HS400	Dual	1.8 V/1.2 V	8	0-200 MHz	400 MB/s

5.3.3 HS200 Bus Speed Mode

The HS200 mode offers the following features:

- SDR Data sampling method
- CLK frequency up to 200 MHz Data rate – up to 200 MB/s
- 4 or 8-bits bus width supported
- Single ended signaling with 4 Drive Strengths
- Signaling levels of 1.8 V and 1.2 V
- Tuning concept for Read Operations

5.3.4 HS200 System Block Diagram

Figure 12 shows a typical HS200 Host and Device system. The host has a clock generator, that supplies CLK to the Device. For write operations, clock and data direction are the same, write data can be transferred synchronous with CLK, regardless of transmission line delay. For read operations, clock and data direction are opposite; the read data received by Host is delayed by round-trip delay, output delay and latency of Host and Device. For reads, the Host needs to have an adjustable sampling point to reliably receive the incoming data.

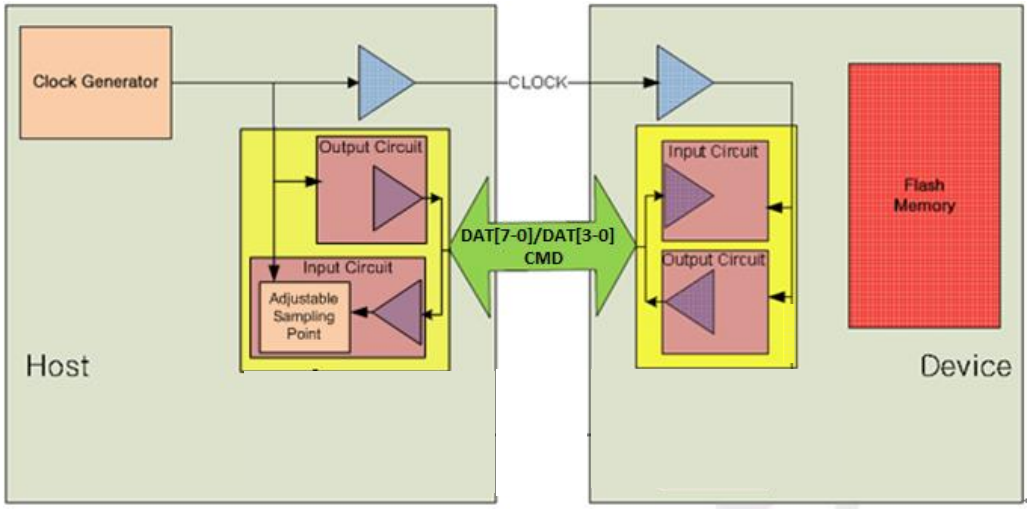


Figure 12 — Host and Device block diagram

5.3.5 HS200 Adjustable Sampling Host

The Host may use adjustable sampling to determine the correct sampling point. A predefined tuning block stored in Device may be used by the Host as an aid for finding the optimal data sampling point. The Host can use CMD21 tuning command to read the tuning block.

5.3.6 HS400 Bus Speed Mode

The HS400 mode has the following features

- DDR Data sampling method
- CLK frequency up to 200 MHz, Data rate is up to 400 MB/s
- Only 8-bit bus width supported
- Signaling levels of 1.8 V and 1.2 V
- Support up to 5 Drive Strengths
- Data strobe signal is toggled only for Data out, CRC response and CMD Response

5.3.7 HS400 System Block Diagram

Figure 13 shows a typical HS400 Host and Device system. The host has a clock generator that supplies CLK to the Device. For read operations, Data Strobe is generated by device output circuit. Host receives the data, CRC Status, and CMD Response (when Enhanced Strobe is enabled), that are aligned to the edge of Data Strobe.

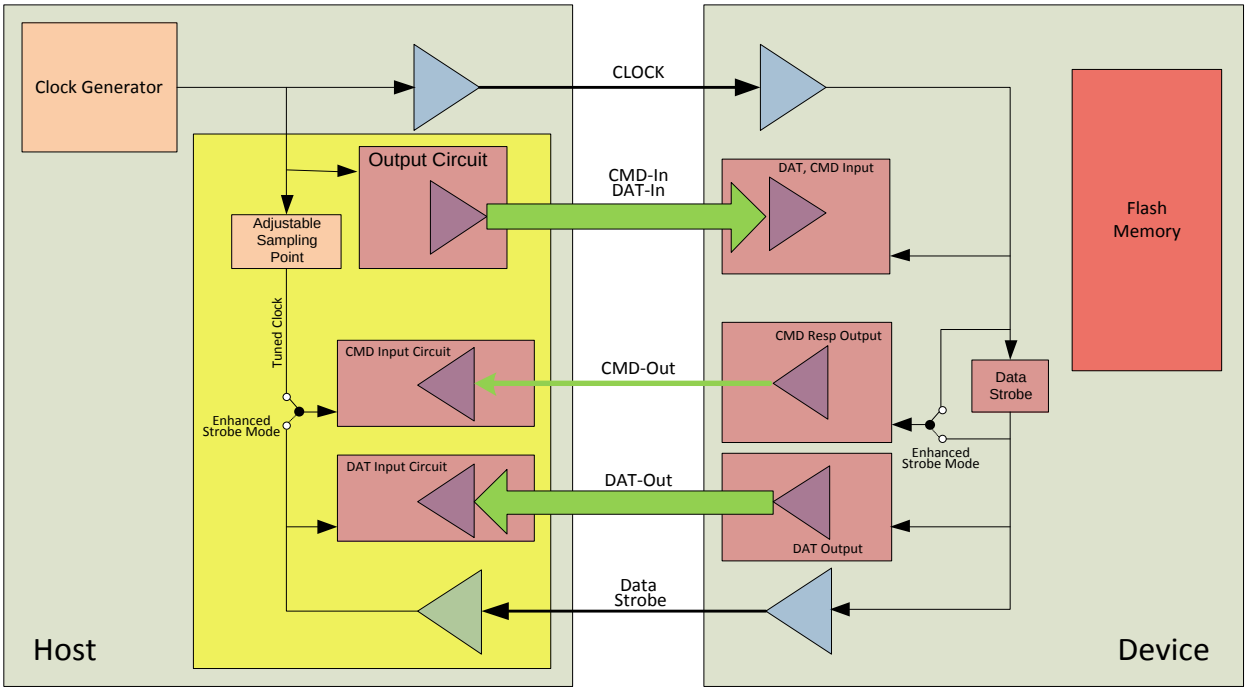


Figure 13 — HS400 Host and Device block diagram

6 *e*•MMC functional description

6.1 *e*•MMC Overview

All communication between host and device are controlled by the host (master). The host sends a command, that results in a device response. A general overview of the command flow is shown in Figure 25 for the device identification mode and in Figure 27 for the data transfer mode. The commands are listed in the command tables (see Table 49 to Table 58). The dependencies between current state, received command and following state are listed in Table 60. Five operation modes are defined for the *e*•MMC system (hosts and devices):

- Boot mode:
  The device will be in boot mode after power cycle, reception of CMD0 with argument of 0xF0F0F0F0 or the assertion of hardware reset signal.
- Device identification mode
  The device will be in device identification mode after boot operation mode is finished or if host and/or device does not support boot operation mode. The device will be in this mode, until the SET_RCA command (CMD3) is received.
- Interrupt mode
  Host and device 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 device or the host.
- Data transfer mode
  The device will enter data transfer mode once an RCA is assigned to it. The host will enter data transfer mode after identifying the device on the bus.
- Inactive mode
  The device will enter inactive mode if either the device operating voltage range or access mode is not valid. The device can also enter inactive mode with GO_INACTIVE_STATE command (CMD15). The device will reset to *Pre-idle* state with power cycle.

Table 5 shows the dependencies between bus modes, operation modes and device states. Each state in the *e*•MMC state diagram (see Figure 25 and Figure 27) is associated with one bus mode and one operation mode.

Table 5 — CMD line modes overview

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

6.2 Partition Management

6.2.1 General

The default area of the memory device consists of a User Data Area to store data, two possible boot area partitions for booting (see 6.3.2) and the Replay Protected Memory Block Area Partition (see 6.6.22) to manage data in an authenticated and replay protected manner. The memory configuration initially consists (before any partitioning operation) of the User Data Area and RPMB Area Partitions and Boot Area Partitions (whose dimensions and technology features are defined by the memory manufacturer).

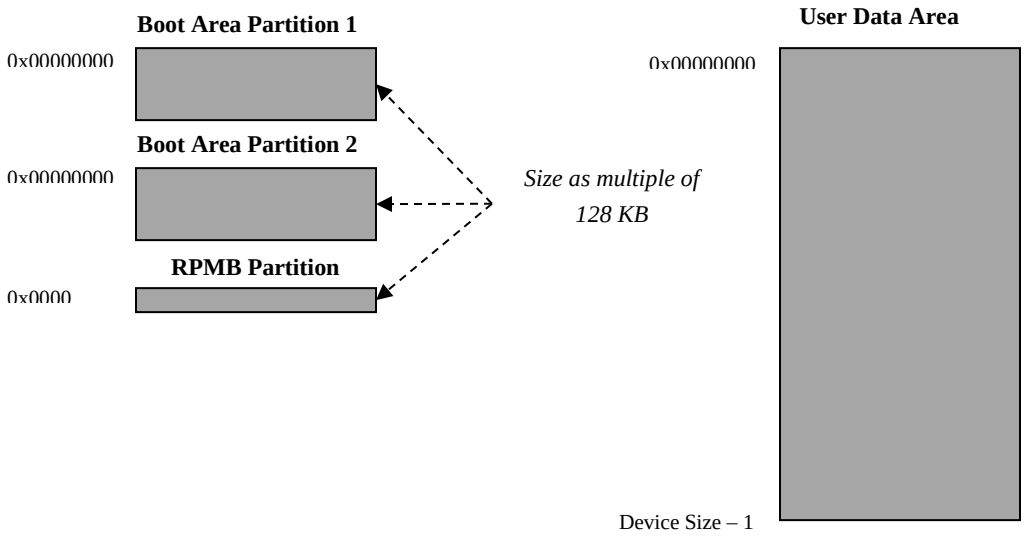


Figure 14 — *e*•MMC memory organization at time zero

The embedded device also offers the host the possibility to configure additional local memory partitions with independent address spaces, starting from logical address 0x00000000, for different usage models.

Therefore the memory block areas can be classified as follows:

- Two Boot Area Partitions, whose size is multiple of 128 KB and where booting from *e*•MMC can be performed.
- One RPMB Partition accessed through a trusted mechanism, whose size is defined as multiple of 128 KB.
- Four General Purpose Area Partitions to store sensitive data or for other host usage models, whose sizes are a multiple of a Write Protect Group.

Each of the General Purpose Area Partitions can be implemented with enhanced or extended technological features (such as better reliability[1]) that distinguish them from the default storage media. If the enhanced storage media feature is supported by the device, boot and RPMB Area Partitions shall be implemented as enhanced storage media by default.

[1] This is cited as an example of an enhanced storage media characteristic, and should not be considered as a necessary definition of enhanced storage media technology. The definition of enhanced storage media should be decided upon by each system manufacturer, and is outside the scope of this standard.

6.2.1 General (cont'd)

Boot and RPMB Area Partitions' sizes and attributes are defined by the memory manufacturer (read-only), while General Purpose Area Partitions' sizes and attributes can be programmed by the host only once in the device life-cycle (one-time programmable).

Moreover, the host is free to configure one segment in the User Data Area to be implemented as enhanced storage media, and to specify its starting location and size in terms of Write Protect Groups. The attributes of this Enhanced User Data Area can be programmed only once during the device life-cycle (one-time programmable).

A possible final configuration can be the following:

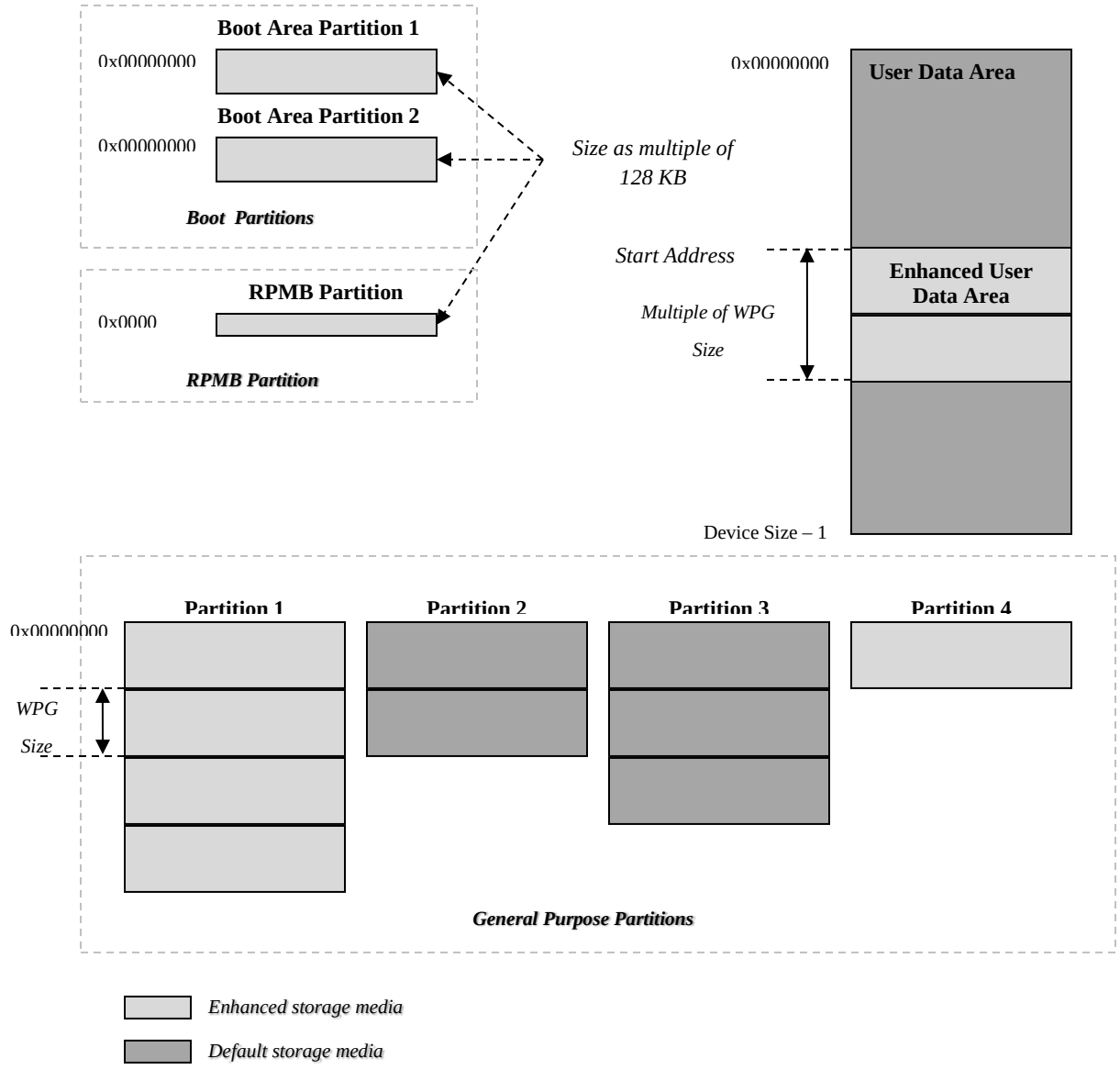


Figure 15 — Example of partitions and user data area configuration

6.2.1 General (cont'd)

General Purpose Partitions and Enhanced User Data Area configuration by the host can have effects on data previously stored (they will be destroyed) and the device initialization time. In particular, the initialization time after first power cycle subsequent to the configuration can exceed the maximum initialization time defined by the specs since the internal controller could execute operations to set up the configurations stated by the host.

More generally also the following initialization phases can be affected by the new configuration. Max power up timings shall be specified in the device technical literature.

6.2.2 Command restrictions

Some restrictions for the commands that can be issued to each partition is defined:

- Boot Partitions
  - Command class 6 (Write Protect) and class 7 (Lock Device) not admitted.
- RPMB Partition
  - Only commands of classes Class0, Class2 and Class4 are admitted. Still usage of any other command than CMD0, CMD6, CMD8, CMD12, CMD13, CMD15 or commands defined in 6.6.22 shall be considered as illegal one.
- General Purpose Partitions
  - Command classes 0, 2, 4, 5, 6 are admitted.
  - Write protection can be set individually for each write protect group in each partition. So the host can set write protection types differently in each write protect group.

In the Enhanced User Data Area, all the commands belonging to the classes admitted in the User Data Area can be issued.

6.2.3 Extended Partitions Attribute

Each General Purpose Partition can have a different extended partition attribute. The list of attribute types includes:

- Default – no extended attribute is set
- System code – a partition that is rarely updated and contains important system files (e.g., containing the executable files of the host operating system)
- Non-Persistent – a partition that is used for temporary information (e.g., swap file to extend the host virtual memory space)

Using the extended attribute, the device can optimize the mixture of storage media characteristics to better suit the intended uses per partition.

A single partition cannot have both enhanced and extended attributes set for it.

6.2.4 Configure partitions

Bit 0 (PARTITIONING_EN) in PARTITIONING_SUPPORT field of the Properties segment in the Extended CSD register indicate if the memory device supports partitioning features. Bit 1 (ENH_ATTRIBUTE_EN) in the same field indicates if the memory device supports enhanced features attribute in the General Purpose Partitions and in the Enhanced User Data Area. Bit 2 (EXT_ATTRIBUTE_EN) in the same field indicates if the memory device supports extended partitions attribute in the General Purpose Partitions. On this specification, Bit 2-0 in PARTITIONING_SUPPORT shall be set to 1.

The attributes of General Purpose Partitions and Enhanced User Data Area can be programmed by the host setting the corresponding values in the Extended CSD registers only once in the device life-cycle. In particular, the host may issue a SWITCH command to set the R/W field of partition features containing the following parameters.

- General Purpose Partitions - size and attribute of max 4 partitions. The fields in the Modes segment of the EXT_CSD register to be set are:
  - GP_SIZE_MULT_GP0 - GP_SIZE_MULT_GP3 for the size
  - PARTITIONS_ATTRIBUTE for the Enhanced attribute
  - EXT_PARTITIONS_ATTRIBUTE for the extended attributes
- Enhanced User Data Area - start address and attribute of the region. The fields in the Modes segment of the EXT_CSD register to be set are:
  - ENH_START_ADDR for the start address
  - ENH_SIZE_MULT for the size
  - PARTITIONS_ATTRIBUTE for the Enhanced attribute

The Enhanced User Data Area start address (ENH_START_ADDR in the Extended CSD) shall be write protect group aligned. It is a group address in byte units, for densities up to 2 GB, and in sector units for densities greater than 2 GB. The device will ignore the LSBs below the write group size and will align the Enhanced User Data Area start address to the Write Protect Group the address (in bytes or sectors) belongs to. The address space of the enhanced user data area is continuous to the address for the rest of the user data area (there is no address gap between the enhanced user data area and the rest of the user data area).

The granularity of General Purpose Partitions and of the Enhanced User Data Area is in units of High Capacity Write Protect Group Sizes (see 7.4). When the partition parameters are configured, ERASE_GROUP_DEF bit in the Extended CSD shall be set to indicate that High Capacity Erase Group Sizes and High Capacity Write Protect Group Sizes are to be used. If the partition parameters are sent to a device by CMD6 before setting ERASE_GROUP_DEF bit, the slave shows SWITCH_ERROR.

Once the device is partitioned and the configuration is stable, all the Command Class 5 and 6 commands will be referred to the high capacity erase groups and write protect groups.

In addition to partitioning parameters fields mentioned before, the host shall set Bit 0 in PARTITION_SETTING_COMPLETED in Modes segment: in this way the host notifies the device that the setting procedure has been successfully completed. This bit setting is to protect partitioning sequence against unexpected power loss event: if a sudden power loss occurs after that partitioning process has been only partially executed, at the next power up the device can detect and invalidate - being this bit not set - the previous incomplete partitioning process giving the host the possibility to repeat and correctly complete it.

6.2.4 Configure partitions (cont'd)

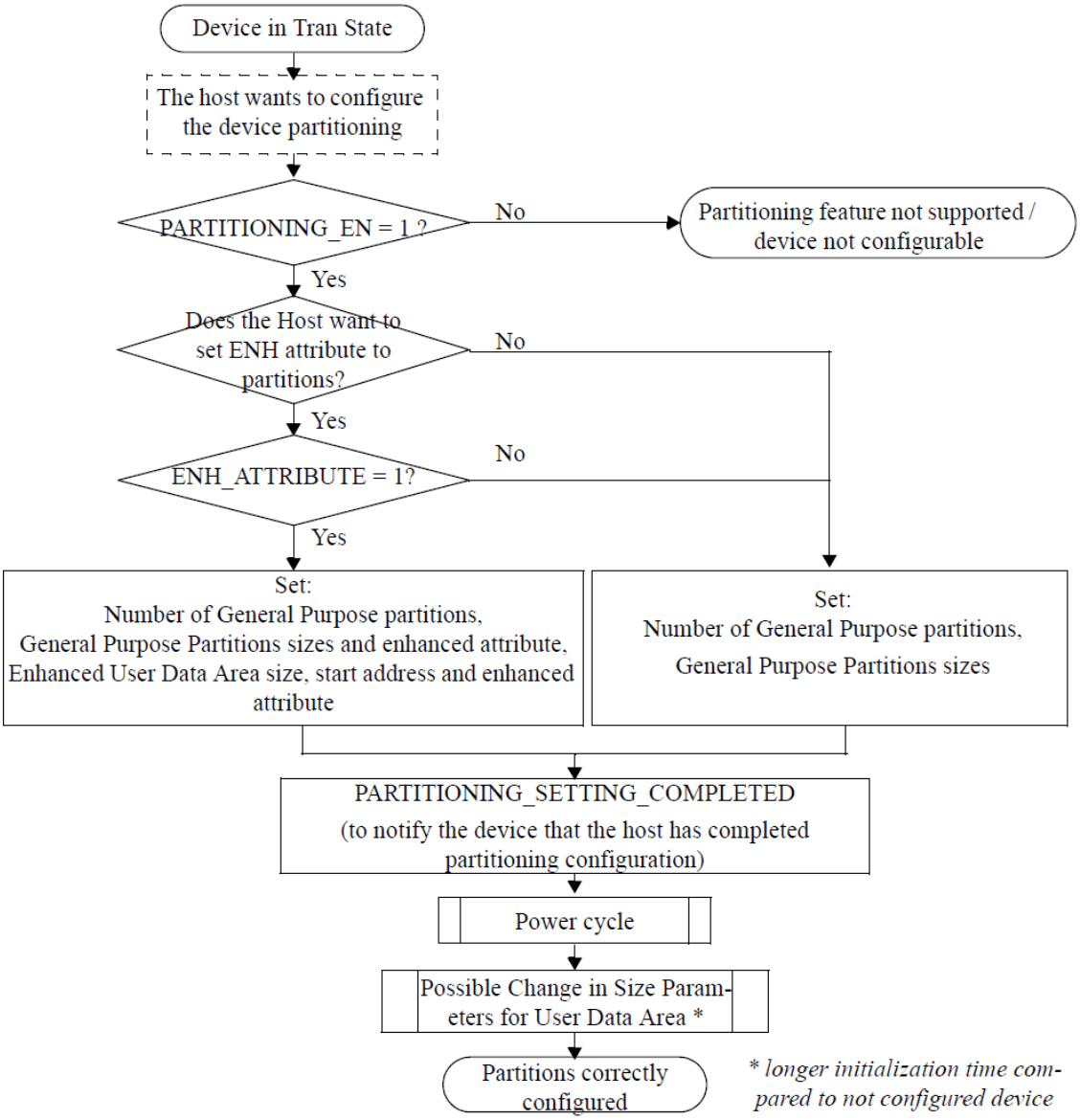


Figure 16 — Flow Chart for General Purpose Partitions & Enhanced User Data Area parameter setting

6.2.4 Configure partitions (cont'd)

A CMD13 shall be issued by the host to make sure that all the parameters are correctly set. If any of the partitioning parameters is not correct a SWITCH_ERROR will be raised by the device. Since the device will not know the total size of configured partitions and user area until PARTITION_SETTING_COMPLETED bit is set, device may show SWITCH_ERROR when host set PARTITION_SETTING_COMPLETED bit, if the total size of the configured partitions and user data area does not fit in the available space of the device. In this case, all the setting will be cleared after the next power cycle. So the host needs to set proper values in each of partition configuration register bytes again.

The device will actually configure itself, according to the partition parameters in the Extended CSD, only after a power cycle. Any valid commands issued after PARTITION_SETTING_COMPLETED bit is set but before a power cycle takes place will be normally executed. Any previous incomplete partitioning configuration sequence before this bit is set will be cancelled upon a power cycle.

After the power cycle following the partition configuration, C_SIZE value for up to 2 GB devices and SEC_COUNT value for more than 2GB devices will be changed to indicate the size of user data area after the configuration. The size compared to 2 GB shall be the size of user data area before configuring partitions (e.g., for more than 2 GB devices before configuring partitions, SEC_COUNT shall keep indicating the size of user data area after configuring partitions, even if the size is decreased to lower than or equal to 2 GB).The size of the user data area includes the size of Enhanced User Data area in the user area. So host may need to read these values after the power cycle to calculate the size of the user data area. Access mode shall keep after configuring partitions.

If the host tries to change General Purpose partitions and Enhanced User Data Area features by using CMD6 after a power up following the configuration procedure, the device will assert the SWITCH_ERROR bit in the status register of CMD 6 response without performing any internal action.

Partitions configuration parameters are stored in one time programmable fields of the Extended CSD register. The host can read them by a CMD8 even though the PARTITION_SETTING_COMPLETED has not yet been set but the execution of partitioning will take place only after the following power up. It is recommended to avoid changes on these parameters after reading them since they are one time programmable fields.

The host shall follow the flow chart in Figure 16 for configuring the parameters of General Purpose Area Partitions and Enhanced User Data Area; otherwise undefined behavior may result.

6.2.5 Access partitions

After every power-up, when the host uses a device that partition(s) are configured, it must set the ERASE_GROUP_DEF bit to high before issuing read, write, erase and write protect commands, because this bit is reset after power-up. Otherwise, these may not work correctly and it may leave the stored data in an unknown state.

Each time the host wants to access a partition the following flow shall be executed:

1) Set PARTITION_ACCESS bits in the PARTITION_CONFIG field of the Extended CSD register in order to address one of the partitions,
2) Issue commands referred to the selected partition,
3) Restore default access to the User Data Area or re-direction the access to another partition.

All the reset events (CMD0 or hardware reset) will restore the access by default to the User Data Area. If an unwanted power loss occurs, the access will be by default restored to the User Data Area. When the host tries to access a partition that has not been created before, the devices sets the SWITCH_ERROR bit in the status register and will not change the PARTITION_ACCESS bits.

6.3 Boot operation mode

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

6.3.1 Device reset to Pre-idle state

The device may enter into *Pre-idle* state through any of the following four mechanisms:

- After power-on by the host, the device (even if it has been in *Inactive* state) is in *Pre-idle* State.
- GO_PRE_IDLE_STATE command (CMD0 with argument of 0xF0F0F0F0) is the software reset command and puts the device into *Pre-idle* State.
- Hardware reset may be used by host resetting a device , moving the device to *Pre-idle* state and disabling power-on period write protect on blocks that had been set as power-on write protect before the reset was asserted. When the device receives GO_PRE_IDLE_STATE command (CMD0 with argument of 0xF0F0F0F0) or assertion of hardware reset signal during sleep state, the device also moves to *Pre-idle* state.

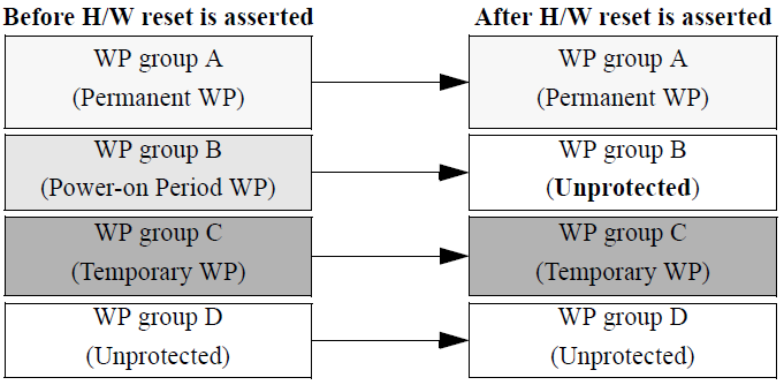


Figure 17 — WP condition transition due to H/W reset assertion

6.3.1 Device reset to Pre-idle state (cont'd)

GO_PRE_IDLE_STATE command or hardware RESET assertion, the device's output bus drivers are in high-impedance state and the device is initialized with a default relative device address (0x0001) and with a default driver stage register setting, as shown in 7.6.

When device powers up, RST_n signal also rises with power source ramp up. So the device may detect rising edge of the RST_n signal at the power up period (either (1), (2), (3) or (4) as shown in below).The device must handle this situation and work properly after the power-up.

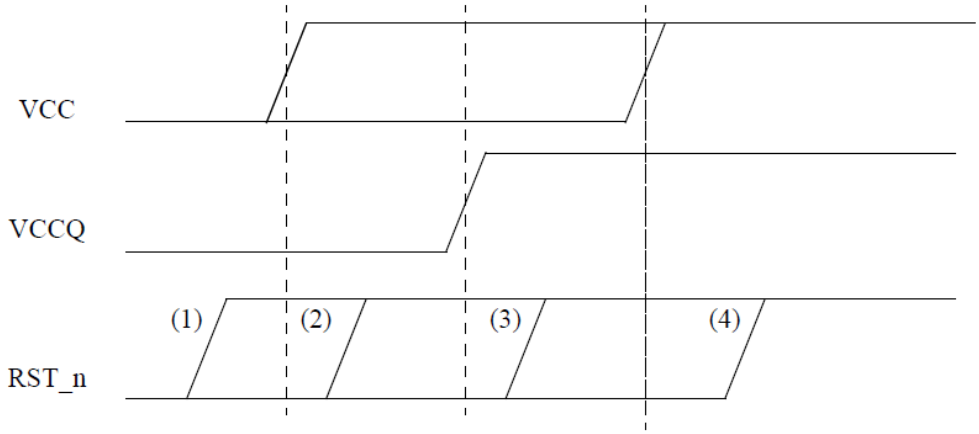


Figure 18 — RST_n signal at the power up period

If the RST_n signal falls before V_{CCQ} fully powers up, the V_{CCQ} rising edge is considered as the falling edge of RST_n signal. In this case, the pulse width of RST_n signal should be measured between the rising edge of RST_n signal and the time V_{CCQ} powers up.

During the device internal initialization sequence right after power-on, device may not be able to detect RST_n signal, because the device may not complete loading RST_n_ENABLE bits of the extended CSD register into the controller yet. However the device already started internal initialization sequence due to power-up, that essentially includes the reset sequence asserted by RST_n signal. The device may not have to do the reset sequence again but it should complete the internal initialization sequence within 1 second. In this case, the initialization delay should be the longest of 1 msec, 74 clock cycles after RST_n asserted or the supply ramp-up time.

6.3.2 Boot partition

There are two partition regions. The minimum size of each boot partition is 128KB. Boot partition size is calculated as follows: Maximum boot partition size = 128K byte x BOOT_SIZE_MULT BOOT_SIZE_MULT: the value in Extended CSD register byte [226] The boot partitions are separated from the user area as shown in Figure 19.

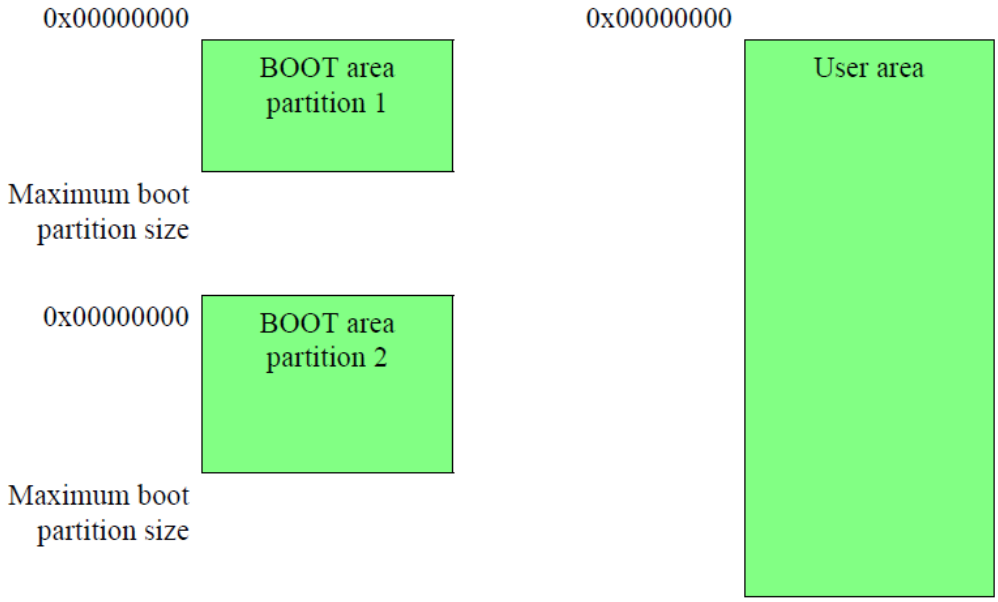


Figure 19 — Memory partition

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

6.3.3 Boot operation

If the CMD line is held LOW for 74 clock cycles and more after power-up or reset operation (either through CMD0 with the argument of 0xF0F0F0F0 or assertion of hardware reset for *e*•MMC, if it is enabled in Extended CSD register byte [162], bits [1:0]) before the first command is issued, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

The partition that from the master will read the boot data can be selected in advance using EXT_CSD byte [179], bits [5:3]. The data size that the master can read during boot operation can be calculated as 128KB × BOOT_SIZE_MULT (EXT_CSD byte [226]). Within 1 second after the CMD line goes LOW, the slave starts to send the first boot data to the master on the DAT line(s). The master must keep the CMD line LOW to read all of the boot data. The master must use push-pull mode until boot operation is terminated.

The master can choose to use single data rate mode with backward-compatible interface timing, single data rate with high-speed interface timing or dual data rate timing (if it supported) shown in 10.6 by setting a proper value in EXT_CSD register byte [177] bits [4:3]. EXT_CSD register byte [228], bit 2 tells the master if the high-speed timing during boot is supported by the device.

The master can also choose to use the dual data rate mode with interface shown in Table 208 during boot by setting “10” in EXT_CSD register byte [177], bits [4:3]. EXT_CSD register byte [228], bit 1 tells the master if the dual data rate mode during boot is supported by the device.

HS200 & HS400 mode is not supported during boot operation.

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

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

In the single data rate mode, data is clocked out by the device and sampled by the host with the rising edge of the clock and there is a single CRC per data line.

In the dual data rate mode, data is clocked out with both the rising edge of the clock and the falling edge of the clock and there are two CRC appended per data line. In this mode, the block length is always 512 bytes, and bytes come interleaved in either 4-bit or 8-bit width configuration. Bytes with odd number (1,3,5, ... ,511) shall be sampled on the rising edge of the clock by the host and bytes with even number (2,4,6, ... ,512) shall be sampled on the falling edge of the clock by the host. The device will append two CRC16 per each valid data line, one corresponding to the bits of the 256 odd bytes to be sampled on the rising edge of the clock by the host and the second for the remaining bits of the 256 even bytes of the block to be sampled on the falling edge of the clock by the host.

All timings on DAT lines shall follow DDR timing mode. The start bit, the end bit and Boot acknowledge bits are only valid on the rising edge of the clock. The value of the falling edge is not guaranteed.

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

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

6.3.3 Boot operation (cont'd)

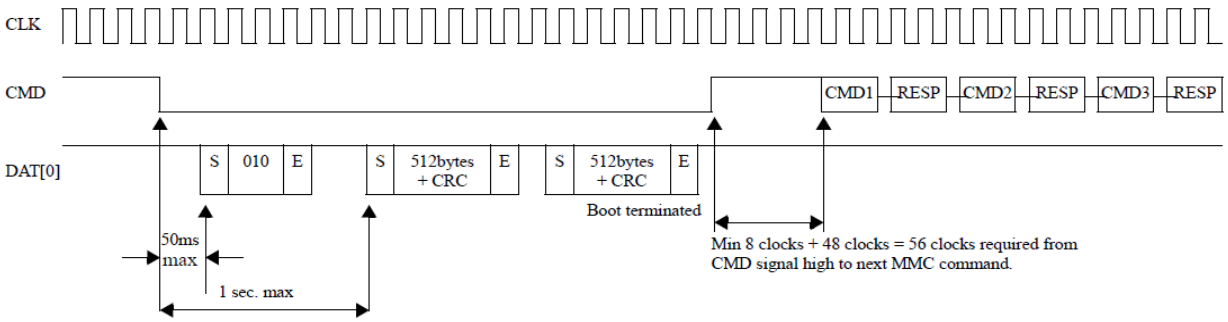


Figure 20 — *e*•MMC state diagram (boot mode)

Detailed timings are shown in 6.15.4.

Min 8 clocks + 48 clocks = 56 clocks required from CMD signal high to next *e*•MMC command. If the CMD line is held LOW for less than 74 clock cycles after power-up before CMD1 is issued, or the master sends any normal *e*•MMC command other than CMD0 with argument 0xFFFFFFFA before initiating boot mode, the slave shall not respond and shall be locked out of boot mode until the next power cycle or hardware reset, and shall enter Idle State.

When BOOT_PARTITION_ENABLE bits are set and master send CMD1 (SEND_OP_COND), slave must enter device Identification Mode and respond to the command. If the slave does not support boot operation mode, per v4.2 or prior, or BOOT_PARTITION_ENABLE bit is cleared, slave automatically enter Idle State after power-on.

6.3.4 Alternative boot operation

This boot function is mandatory for device from v4.4 standard. Device who follows v4.4 standard must show "1" bit 0 in the Extended CSD byte [228]. After power-up or reset operation (either assertion of CMD0 with the argument of 0xF0F0F0F0 or H/W reset if it is enabled), if the host issues CMD0 with the argument of 0xFFFFFFFA after 74 clock cycles, before CMD1 is issued or the CMD line goes low, the slave recognizes that boot mode is being initiated and starts preparing boot data internally. The partition that from the master will read the boot data can be selected in advance using EXT_CSD byte [179], bits [5:3]. The data size that the master can read during boot operation can be calculated as 128KB × BOOT_SIZE_MULT (EXT_CSD byte [226]). Within 1 second after CMD0 with the argument of 0xFFFFFFFA is issued, the slave starts to send the first boot data to the master on the DAT line(s). The master must use push-pull mode until boot operation is terminated. The master can choose to use single data rate mode with backward-compatible interface timing, single data rate with high-speed interface timing or dual data rate timing (if it is supported) shown in 10.7 by setting a proper value in EXT_CSD register byte [177] bit[4:3]. EXT_CSD register byte [228], bit 2 tells the master if the high-speed timing during boot is supported by the device.

The master can choose to receive boot acknowledge from the slave by setting "1" in EXT_CSD register, byte [179], bit 6, so that the master can recognize that the slave is operating in boot mode. If boot acknowledge is enabled, the slave has to send the acknowledge pattern "010" to the master within 50ms after the CMD0 with the argument of 0xFFFFFFFA is received. If boot acknowledge is disabled, the slave will not send out acknowledge pattern "010."

In the single data rate mode, data are clocked out by the device and sampled by the host with the rising edge of the clock and there is a single CRC per data line.

6.3.4 Alternative Boot operation (cont'd)

In the dual data rate mode, data are clocked out with both the rising edge of the clock and the falling edge of the clock and there are two CRC appended per data line. In this mode, the block length is always 512 bytes, and bytes come interleaved in either 4-bit or 8-bit width configuration. Bytes with odd number(1,3,5, ... ,511) shall be sampled on the rising edge of the clock by the host and bytes with even number (2,4,6, ... ,512) shall be sampled on the falling edge of the clock by the host. The device will append two CRC16 per each valid data line, one corresponding to the bits of the 256 odd bytes to be sampled on the rising edge of the clock by the host and the second for the remaining bits of the 256 even bytes of the block to be sampled on the falling edge of the clock by the host.

All timings on DAT lines shall follow DDR timing mode. The start bit, the end bit and Boot acknowledge bits are only valid on the rising edge of the clock. The value of the falling edge is not guaranteed.

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

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

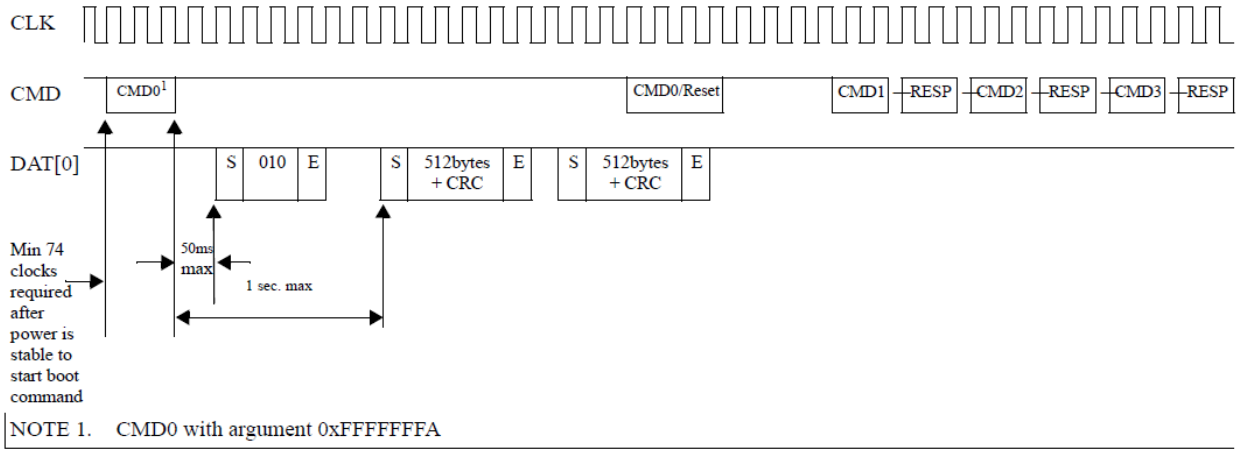


Figure 21 — *e*•MMC state diagram (alternative boot mode)

Detailed timings are shown in 6.15.4.

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

When BOOT_PARTITION_ENABLE bits are set and master send CMD1 (SEND_OP_COND), slave must enter Device Identification Mode and respond to the command. If the slave does not support boot operation mode, per v4.2 or prior, or BOOT_PARTITION_ENABLE bit is cleared, slave automatically enter Idle State after power-on.

6.3.4 Alternative Boot operation (cont'd)

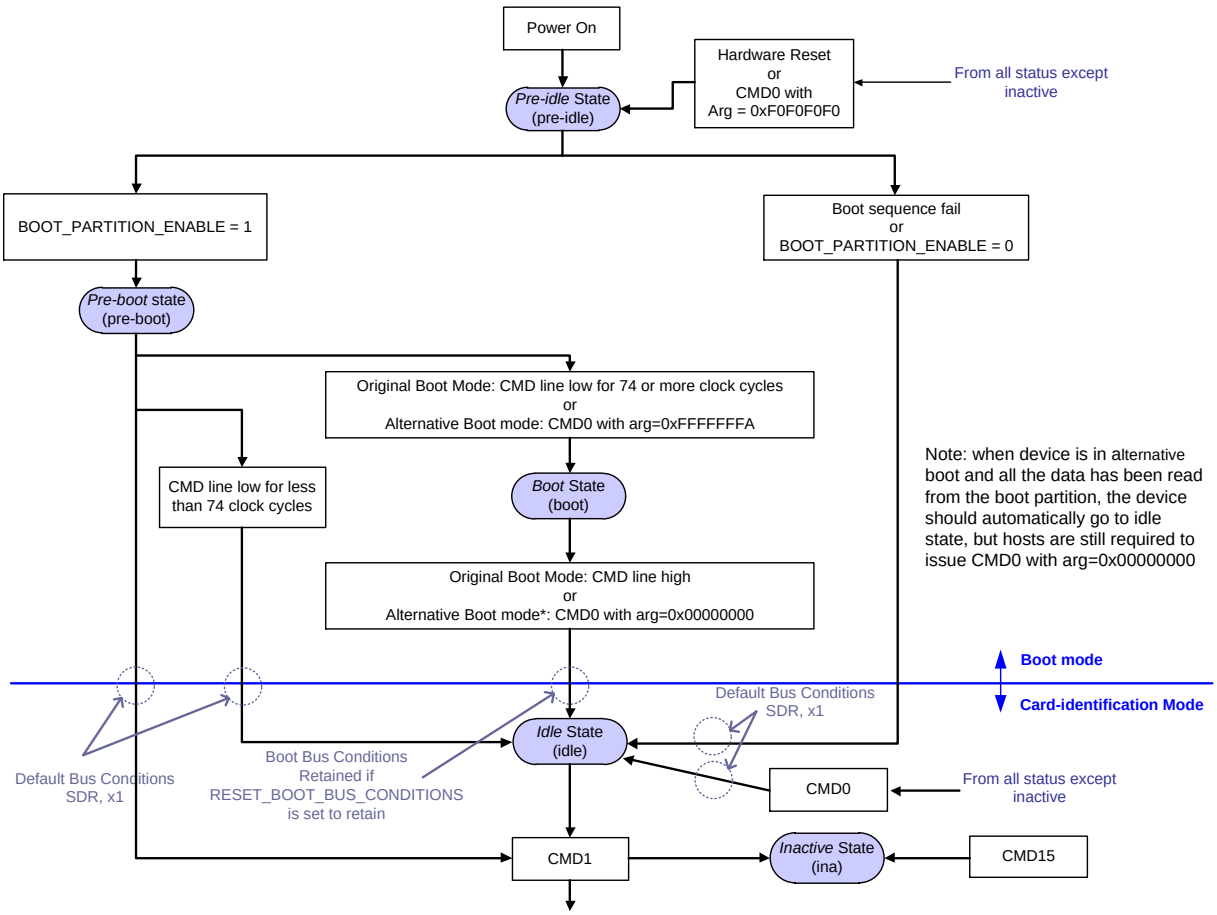


Figure 22 — *e*•MMC state diagram (boot mode)

6.3.4 Alternative Boot operation (cont'd)

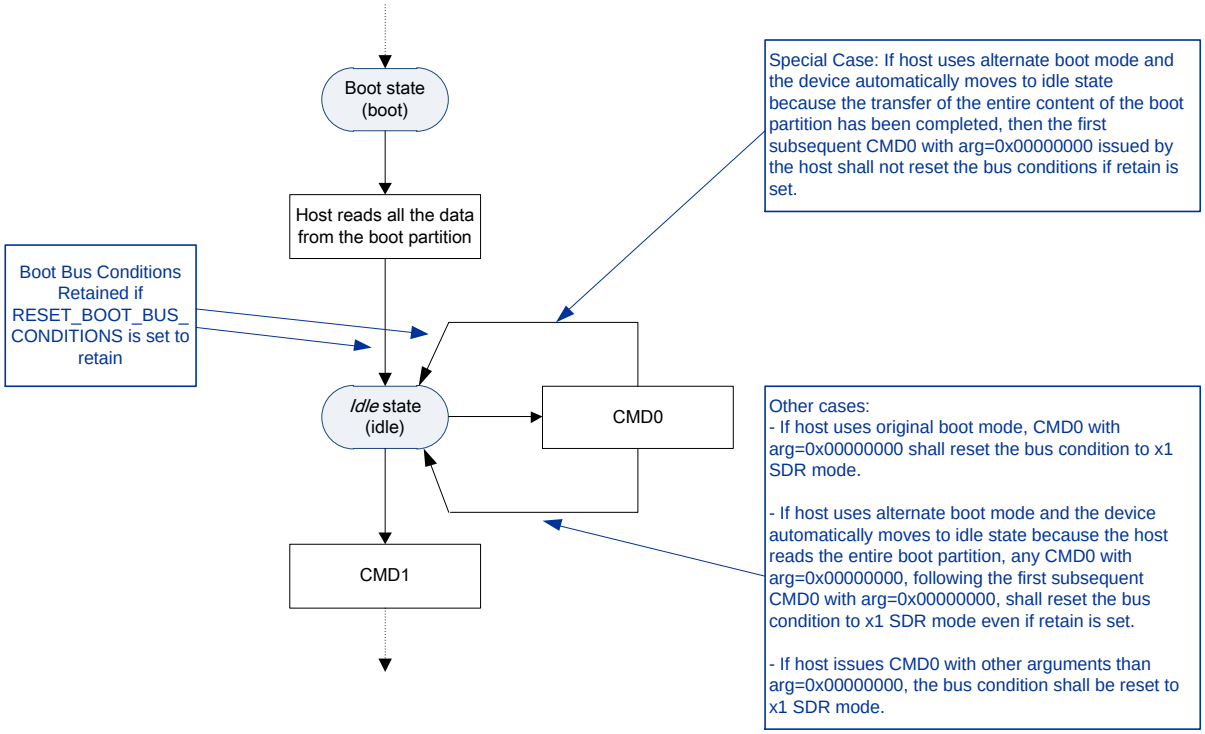


Figure 23 — Clarification of RESET_BOOT_BUS_CONDITIONS behavior when CMD0 is issued in IDLE

6.3.5 Access to boot partition

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

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

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

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

Master also can access user area by using normal command by clearing PARTITION_ACCESS bits in the EXT_CSD register, byte [179] to 000b. If user area is locked and enabled for boot, data will not be sent out to master during boot operation mode. However, if the user area is locked and one of the two partitions is enabled, data will be sent out to the master during boot operation mode.

6.3.6 Boot bus width and data access configuration

During boot operation, bus width can be configured by nonvolatile configuration bits in the Extend CSD register byte[177] bit[0:1]. Bit2 in register byte[177] determines if the slave returns to x1 bus width and single data rate mode with backward compatible timing after a boot operation or if it remains in the configured boot-bus width during normal operation. Bits[4:3] in register byte[177] determines if the data lines are configured for single data rate using backward compatible or high speed timings or dual data rate mode during boot operation. If boot operation is not executed, the slave will initialize in normal x1 bus width, single data rate operation and backward compatible timing regardless of the register setting.

6.3.7 Boot Partition Write Protection

In order to allow the host to protect the boot area against erase or write, the *e*•MMC shall support two levels of write protection for each of the boot areas: Permanent write protection or power-on write protection. This protection can be applied to each boot area individually or to both regions at the same time using the BOOT_WP register (EXT_CSD[173]).

The host must be aware of the following when writing the BOOT_WP register:

- All the bits BOOT_WP register, except the two R/W bits B_PERM_WP_DIS (bit 4) and B_PERM_WP_EN (bit 2), shall only be written once per power cycle. The protection mode intended for both boot areas will be set with a single write.
- Two bits in the register are type R/W, B_PERM_WP_DIS (bit 4) and B_PERM_WP_EN (bit 2) the first write to the boot register will permanently set these bits. The flow chart in Figure 24 should be followed for the first write to register BOOT_WP to properly set these bits.
- If the B_PERM_WP_EN bit is set for only one boot partition, the host should ensure that B_SEC_WP_SEL (bit 7) and B_PERM_WR_SEC_SEL (bit 3) are set correctly to avoid accidentally permanently protecting the other boot Area.

The host has the ability to disable both permanent and power on write protection in the boot area by setting B_PERM_WP_DIS (EXT_CSD[173] bit 4) and B_PWR_WP_DIS (EXT_CSD[173] bit 6). If boot area protection is not required it is recommended that these bits be set in order to ensure that the boot area is not protected unintentionally or maliciously.

Refer to 6.6.40, Secure Write Protect Mode, for further handling of BOOT_WP register when SECURE_WP_EN is set.

6.3.7 Boot Partition Write Protection (cont'd)

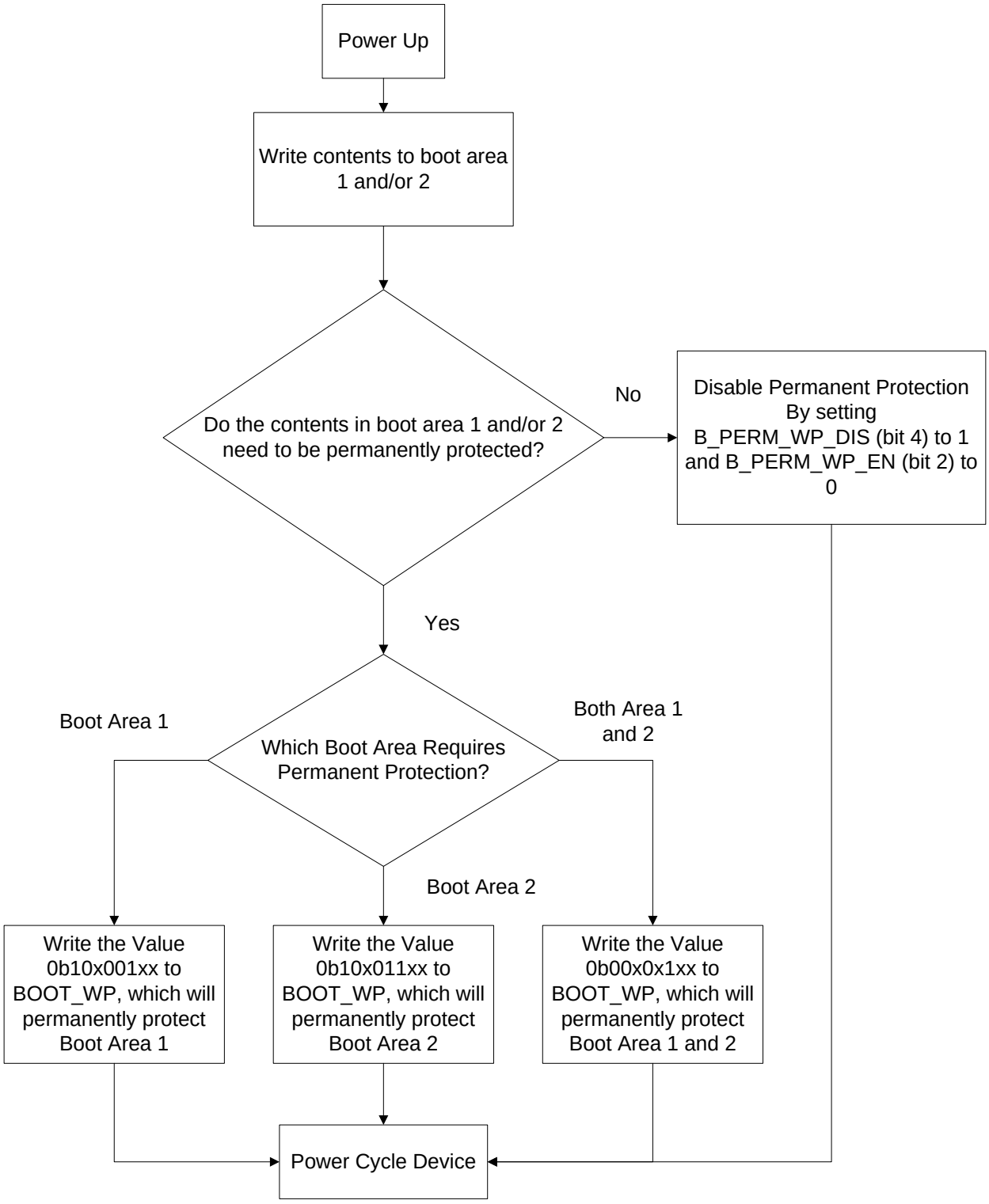


Figure 24 — Setting Ext CSD BOOT_WP[173]

6.4 Device identification mode

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

6.4.1 Device reset

After receiving Command GO_IDLE_STATE (CMD0 with argument of 0x00000000), the device s go to *Idle* State. The following are the cases where the device moves into *Idle* State.

- After completing boot operation,
- After CMD line low for less than 74 clock in pre-boot state,
- After power-up if the device is not boot enabled.

In this state, the devices' output bus drivers are in high-impedance state and the Device is initialized with a default relative Device address (0x0001) and with a default driver stage register setting, as shown in 7.6 The host clocks the bus at the identification clock rate $f_{OD,}$ as described in 10.6. CMD0 with argument of 0x00000000 is valid in all states, with the exception of *Inactive* State. While in *Inactive* state the Device does not accept CMD0 with argument of 0x00000000.

For backward compatibility reason, if device receive CMD0 with argument of other than 0xFFFFFFFA or 0xF0F0F0F0 in any state except *Inactive* state, device shall treat it as Device reset command and move to *Idle* state. CMD0 with argument of 0xFFFFFFFA is a boot initiation command in *Pre-boot* state, but if host issue this command in any state except *Inactive* state and *Pre-boot* state, the device shall treat it as a rest command and move to *Idle* state.

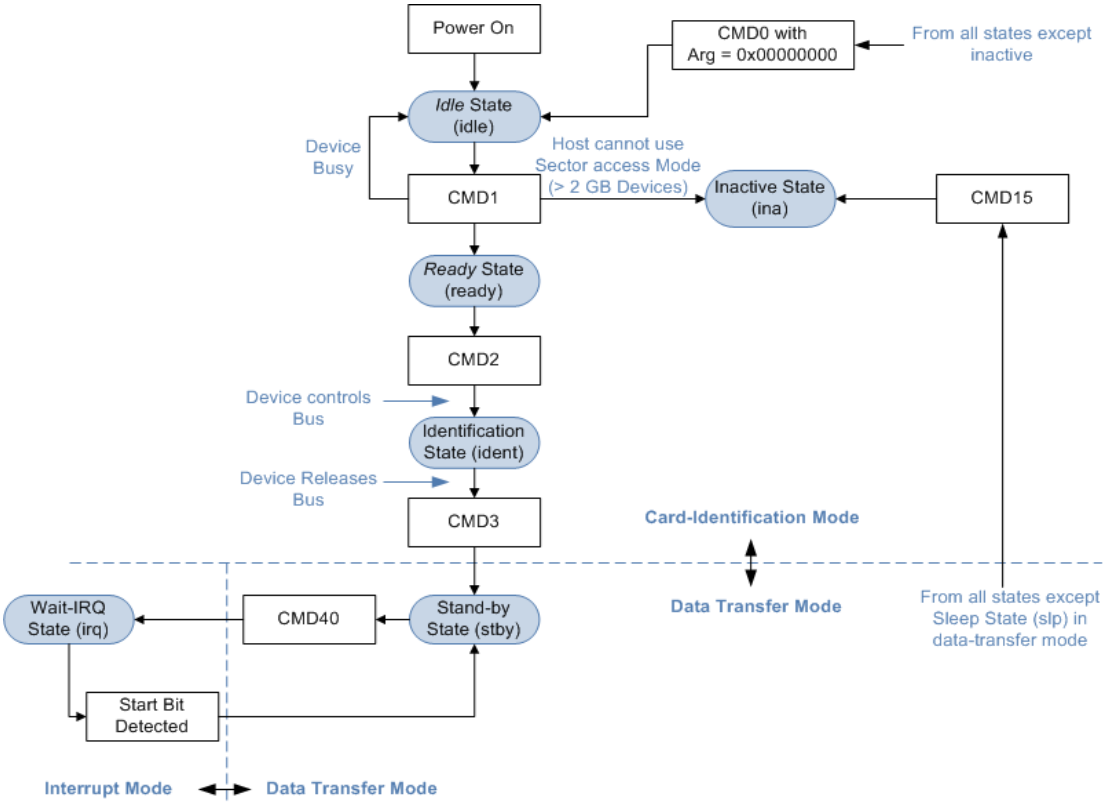


Figure 25 — *e*•MMC state diagram (Device identification mode)

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

For *e*•MMC devices, the voltage range in CMD1 is no longer valid. Regardless of the voltage range indicated by the host, the *e*•MMC devices shall respond with a fixed pattern of either 0x00FF 8080 (capacity less than or equal to 2GB) or 0x40FF 8080 (capacity greater than 2GB) if device is busy, and they shall not move into Inactive state.

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

- If there is no indication by a host to a memory that the host is capable of handling sector type of addressing the higher than 2GB of density of memory will change its state to *Inactive* (similarly to a situation where there is no common voltage range to work with) (exception, if a host send 0x0000 0000 for voltage range validation, device shall not change its state to *Inactive* during voltage range validation stage) This will also be true if the operand generated by the host is 0x0000 0000, and does not represent any valid range.
- From the indication of the sector type of addressing requirement in the OCR register the host is able to separate the device from the byte access mode Devices and prepare itself.

The *e*•MMC devices shall respond with a fixed pattern of either 0x00FF 8080 or 0x40FF 8080 if device is busy, 0x80FF8080 (capacity less than or equal to 2GB) or 0xC0FF8080 (capacity greater than 2 GB) if device is entering Ready state, and they shall not move into *Inactive* state. The host shall ignore the access mode bits if the device is busy.
Due to legacy reasons a host may need to change the voltage of a device. If the host changes the voltage range from one range to another range, the device shall be fully powered down and then powered up to the new voltage range. Dual voltage devices may fail if the voltage range 1.95 V to 2.7 V is used.

The addressing mode should be reconfirmed by the host by reading the SEC_COUNT information from the EXT_CSD register.

6.4.3 From busy to ready

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

During the initialization procedure, the host is not allowed to change the operating voltage range or access mode setting. Such changes shall be ignored by the device. If there is a real change in the operating conditions, the host must reset the device (using CMD0 with argument of 0x00000000) and restart the initialization procedure. However, for accessing devices already in *Inactive* State, a hard reset must be done by switching the power supply off and back on.

The command GO_INACTIVE_STATE (CMD15) can be used to send an addressed device into the *Inactive* State. This command is used when the host explicitly wants to deactivate a device.

The command CMD1 shall be implemented by all devices defined by this standard.

6.4.4 Device identification process

The following explanation refers to a Device working in a multi-Device 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 Device identification process in open-drain mode with the identification clock rate $f_{OD.}$ (See 10.6). The open drain driver stages on the CMD line allow parallel Device operation during Device identification.

After the bus is activated, the host will request the Devices to send its valid operation conditions (CMD1). The response to CMD1 is the 'wired and' operation on the condition restrictions of all Devices in the system. Incompatible Devices are sent into *Inactive* State. The host then issues the broadcast command ALL_SEND_CID (CMD2), asking all Devices for its unique Device identification (CID) number. All unidentified Devices (i.e., those that are in *Ready* State) simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bit stream. Those Devices, 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 Device, there should be only one Device that successfully sends its full CID-number to the host. This Device then goes into *Identification* State. Thereafter, the host issues CMD3 (SET_RELATIVE_ADDR) to assign to this Device a relative Device address (RCA), that is shorter than CID and that will be used to address the Device in the future data transfer mode (typically with a higher clock rate than f_{OD}). Once the RCA is received the Device state changes to the *Stand-by* State, and the Device does not react to further identification cycles. Furthermore, the Device 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 Devices responds to this command, all Devices 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 6.15)

6.5 Interrupt mode

The interrupt mode on the *e*•MMC system enables the master (*e*•MMC host) to grant the transmission allowance to the slaves (Device) 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 Device request for service. Supporting *e*•MMC interrupt mode is an option, both for the host and the Device.

- The system behavior during the interrupt mode is described in the state diagram in Figure 26.
- The host must ensure that the Device is in *Stand-by* State before issuing the GO_IRQ_STATE (CMD40) command. While waiting for an interrupt response from the Device, the host must keep the clock signal active. Clock rate may be changed according to the required response time.
- The host sets the Device into interrupt mode using GO_IRQ_STATE (CMD40) command.
- A Device in Wait-IRQ-State is waiting for an internal interrupt trigger event. Once the event occurs, the Device 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 Device is also waiting for a start bit on the command line. Upon detection of a start bit, the Device will abort interrupt mode and switch to the *stand-by* state.

6.5 Interrupt mode (cont'd)

- Regardless of winning or losing bus control during CMD40 response, the device 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.

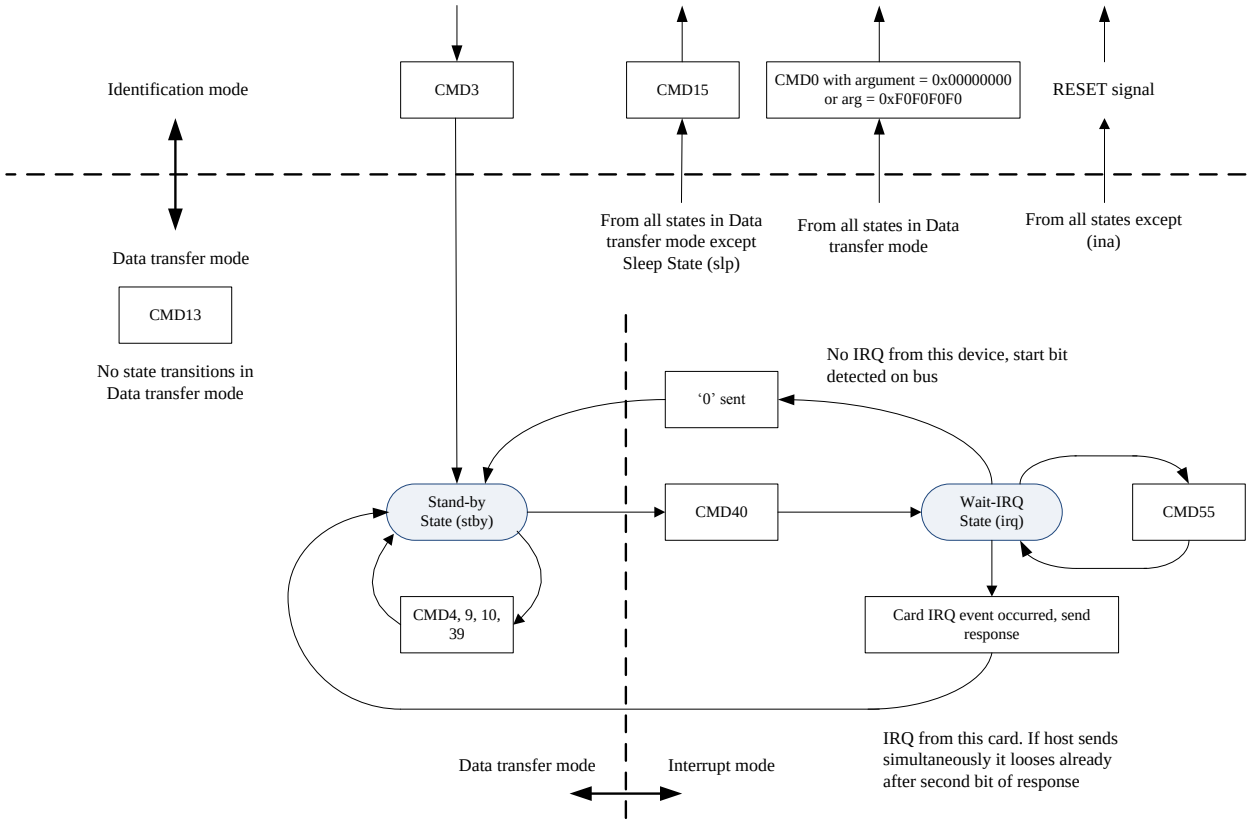
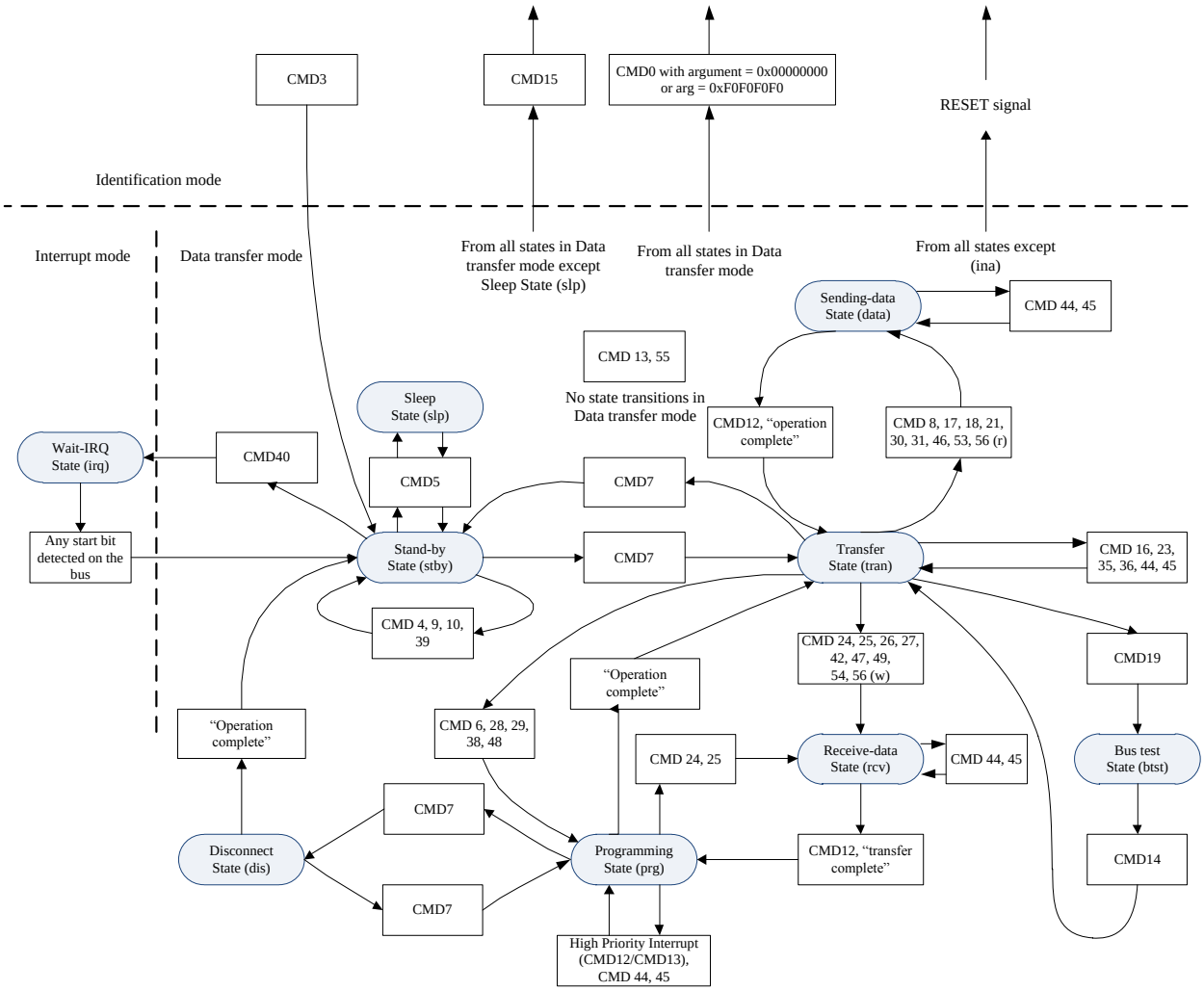


Figure 26 — *e*•MMC state transition diagram, interrupt mode

- If the host wants to terminate the interrupt mode before an interrupt response is received, it can generate the CMD40 response by himself (with Device bit = 0) using the reserved RCA address 0x0000; this will bring the Device from Wait-IRQ-State back into the Stand-by-State. Now the host can resume the standard communication procedure.

6.6 Data transfer mode

When the Device is in *Stand-by* State, communication over the CMD and DAT lines will be performed in push-pull mode. Until the contents of the CSD register is known by the host, the f_{PP} clock rate must remain at f_{OD} (see 10.6). The host issues SEND_CSD (CMD9) to obtain the Device Specific Data (CSD register).



NOTE The busy (DAT0=low) is always active during the prg-state. Due to legacy reasons, a Device may still treat CMD24/25 during prg-state (while busy is active) as a legal or illegal command. A host should not send CMD24/25 while the Device is in the prg state and busy is active.

Figure 27 — *e*•MMC state diagram (data transfer mode)

The broadcast command SET_DSR (CMD4) configures the driver stages of the Device. 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.

6.6 Data transfer mode (cont'd)

While the Device is in *Stand-by* State, CMD7 is used to select the Device and put it into the *Transfer* State by including Device's relative address in the argument. If the Device was previously selected and was in *Transfer* State its connection with the host is released and it will move back to the *Stand-by* State when deselected by CMD7 with any address in the argument that is not equal to Device's own relative address. When CMD7 is issued with the reserved relative Device address "0x0000", the Device is put back to *Stand-by* State. Reception of CMD7 with Device's own relative address while the Device is in *Transfer* State is ignored by the Device and may be treated as an Illegal Command. After the Device is assigned an RCA it will not respond to identification commands: CMD1, CMD2, or CMD3 (see 6.4.4).

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

All data communication in the Data Transfer Mode is point-to point between the host and the selected Device (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 (see Figure 27):

- All data read commands can be aborted any time by the stop command (CMD12). The data transfer will terminate and the Device will return to the *Transfer* State. The read commands are: block read (CMD17), multiple block read (CMD18), send tuning block (CMD21) 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 Device by CMD7. The write commands are: block write (CMD24 and CMD25), write CID (CMD26), and write CSD (CMD27).
- As soon as the data transfer is completed, the Device 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.
- The Device may provide buffering for block write. This means that the next block can be sent to the Device while the previous is being programmed.
- There is no buffering option for write CSD, write CID, write protection and erase. This means that while the Device 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 Device is busy and in the Programming State.
- Parameter set commands are not allowed while Device is programming. Parameter set commands are: set block length (CMD16), and erase group selection (CMD35-36).
- Read commands are not allowed while Device is programming.
- Moving another Device from Stand-by to Transfer State (using CMD7) will not terminate a programming operation. The Device will switch to the Disconnect State and will release the DAT0 line.
- A Device can be reselected while in the Disconnect State, using CMD7. In this case the Device will move to the Programming State and reactivate the busy indication.

6.6 Data transfer mode (cont'd)

- Resetting a Device (using CMD0, CMD15, or hardware reset for *e*•MMC) or power failure will terminate any pending or active programming operation. This may leave some or all of the data addressed by the operation in an unknown state unless Reliable Write was enabled. It is 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, this might not be the case if the bus testing procedure is performed with lower clock frequency than the data transfer frequency.
- The following commands: bus testing (CMD19, CMD14), lock-unlock (CMD42) and set block-length (CMD16) are not allowed once the Device is configured to operate in dual data rate mode and shall not be executed but regarded as illegal commands.

In the following format definitions, all upper case flags and parameters are defined in the CSD (7.3), and the other status flags in the Device Status (6.13).

6.6.1 Command sets and extended settings

The Device operates in a given command set, by default, after a power cycle, reset by CMD0 with argument of 0x00000000 or after boot operation; it is the *e*•MMC 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 Device capabilities. The Modes segment reflects the current selected modes of the Device.

The host reads the EXT_CSD register by issuing the SEND_EXT_CSD command. The Device sends the EXT_CSD register as a block of data, 512 bytes long. Any reserved, or write only field, reads as '0'. The host can write the Modes segment of the EXT_CSD register by issuing a SWITCH command and setting one of the access modes. All three modes access and modify one of the EXT_CSD bytes, the byte pointed by the Index field.

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

Table 6 — EXT_CSD access mode

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

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

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

6.6.2 High-speed modes selection

If host wants to operate a device with a clock above 26 MHz it shall switch the timing mode to one of supported higher speed timing modes: “High Speed” for frequencies between 26 MHz and 52 MHz.“HS200” or “HS400” for frequencies above 52 MHz up to 200 MHz (“High Speed” and either “HS200” or “HS400” timing modes allow operation from 0 to 52 MHz and 0 to 200 MHz, respectively, as given in Table 4).

NOTE While the actual timing change is done, the behavior of any command sent (like CMD13) cannot be guaranteed due to the asynchronous operation. Therefore it is not recommended to use CMD13 to check the busy completion of the timing change indication. In case CMD13 is used the host must ignore CRC errors, if appear.

The following sections describe the mode selections in detail.

6.6.2.1 “High-speed” mode selection

After the host verifies that the Device complies with version 4.0, or higher, of this standard, it has to enable the high speed mode timing in the Device, before changing the clock frequency to a frequency between 26MHz and 52MHz.

After power-on, or software reset, the interface timing of the Device is set as specified in Table A.222. 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 7.4.65. 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.

6.6.2.2 “HS200” timing mode selection

HS200 is valid only at V_{CCQ}= 1.8 V or 1.2 V.

The bus width is set to SDR 4bit or SDR 8bit in HS200 mode.

After the host initializes the device, it must verify that the device supports the HS200 mode by reading the DEVICE_TYPE field in the Extended CSD register. Then it may enable the HS200 timing mode in the device, before changing the clock frequency to a frequency higher than 52MHz.

After power-on or software reset(CMD0), the interface timing of the device is set as the default “Backward Compatible Timing “. Device shall select HS200 Timing mode if required and perform the Tuning process if needed.

6.6.2.2 "HS200" timing mode selection (cont'd)

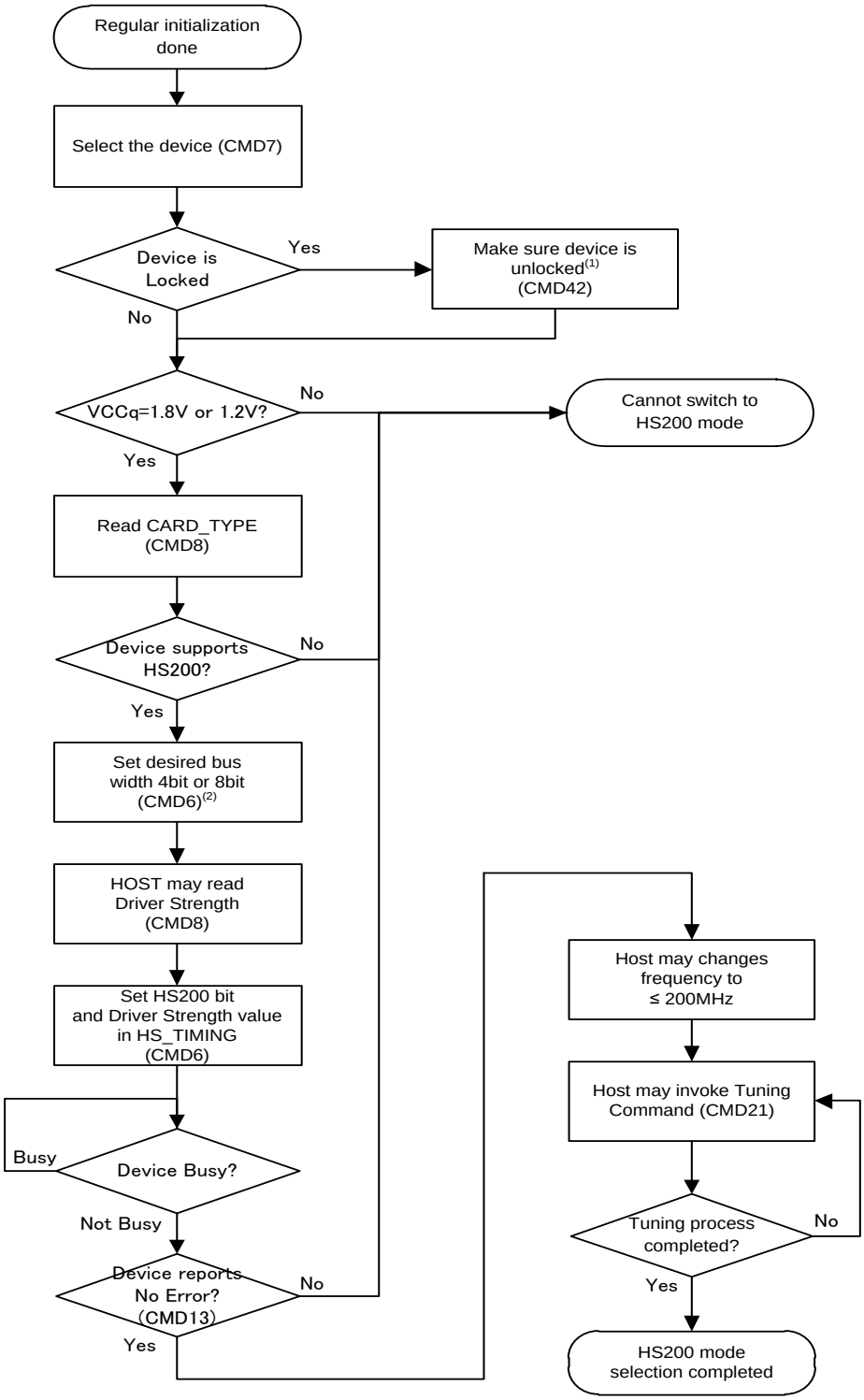
To switch to HS200, host has to perform the following steps:

1) Select the device (through sending CMD7) and make sure it is unlocked (through CMD42)
2) Read the DEVICE_TYPE [196] field of the Extended CSD register to validate if the device supports HS200 at the IO voltage appropriate for both host and device
3) Read the DRIVER_STRENGTH [197] field of the Extended CSD register to find the supported device Driver Strengths. Note: This step can be skipped if changes of driver strength is not needed
4) Set HS200 bit and Driver Strength value in the HS_TIMING [185] field of the Extended CSD register by issuing CMD6. If the host attempts to write an invalid value, the HS_TIMING byte is not changed, the HS200 interface timing is not enabled, the Driver Strength is not changed, and the SWITCH_ERROR bit is set. After the device responds with R1, it might assert Busy signal. Once the busy signal gets de-asserted, the host may send a SEND_STATUS Command (CMD13) using the HS200 timing and after it receives a Trans State indication and No Error it means that the device is set to HS200 timing and the Driver Strength is set to the selected settings.
5) At this point, the host can set the frequency to ≤ 200 MHz.
6) The host may invoke the HS200 tuning sequence, by sending CMD21 to the device (see 6.6.5).

NOTE The host should switch to the required bus width before starting the tuning operation to allow the tuning sequence to be done using the proper bus operating conditions.

6.6.2.2 "HS200" timing mode selection (cont'd)

Figure 28 illustrates the process described in 6.6.4



(1) Locked device does not support CMD21 therefore this check may be done any time before CMD21 is used.
(2) The switch to the required bus width may be done any time before the tuning process starts

Figure 28 — HS200 Selection flow diagram

6.6.2.3 "HS400" timing mode selection

The valid IO Voltage for HS400 is 1.8 V or 1.2 V for V_{CCQ}.

The bus width is set to only DDR 8bit in HS400 mode.

HS400 supports the same commands as DDR52.

After the host initializes the device, host check whether the device supports the HS400 mode by reading the DEVICE_TYPE field in the Extended CSD register. Then it enables the HS400 mode in the device before changing the clock frequency to a frequency higher than 52 MHz.

After power-on or software reset (CMD0), the interface timing of the device is set as the default "Backward Compatible Timing".

In order to switch to HS400 mode, host should perform the following steps:

1) Initialize device with "Backward Compatible Timings",
2) Select the device with CMD7,
3) Read the DEVICE_TYPE [196] field of the Extended CSD register to validate whether the device supports HS400,
4) Read the DRIVER_STRENGTH [197] field of the Extended CSD register to find the supported device Driver Strengths,

   NOTE This step may be skipped if changes of driver strength is not needed.

5) Set the "Selected Driver Strength" parameter in the HS_TIMING [185] field of the Extended CSD register to the appropriate driver strength for HS400 operation and set the "Timing Interface" parameter to 0x2 to switch to HS200 mode,
6) Perform the Tuning Process at the HS400 target operating frequency,

   NOTE Tuning process in HS200 mode is required to synchronize the command response on the CMD line to CLK for HS400 operation.

7) Set the "Timing Interface" parameter in the HS_TIMING [185] field of the Extended CSD register to 0x1 to switch to High Speed mode and then set the clock frequency to a value not greater than 52 MHz,
8) Set BUS_WIDTH[183] to 0x06 to select the dual data rate x8 bus mode,
9) Set the "Timing Interface" parameter in the HS_TIMING [185] field of the Extended CSD register to 0x3 to switch to HS400 mode.

6.6.2.3 "HS400" timing mode selection (cont'd)

Figure 28 illustrates the process described in 6.6.5

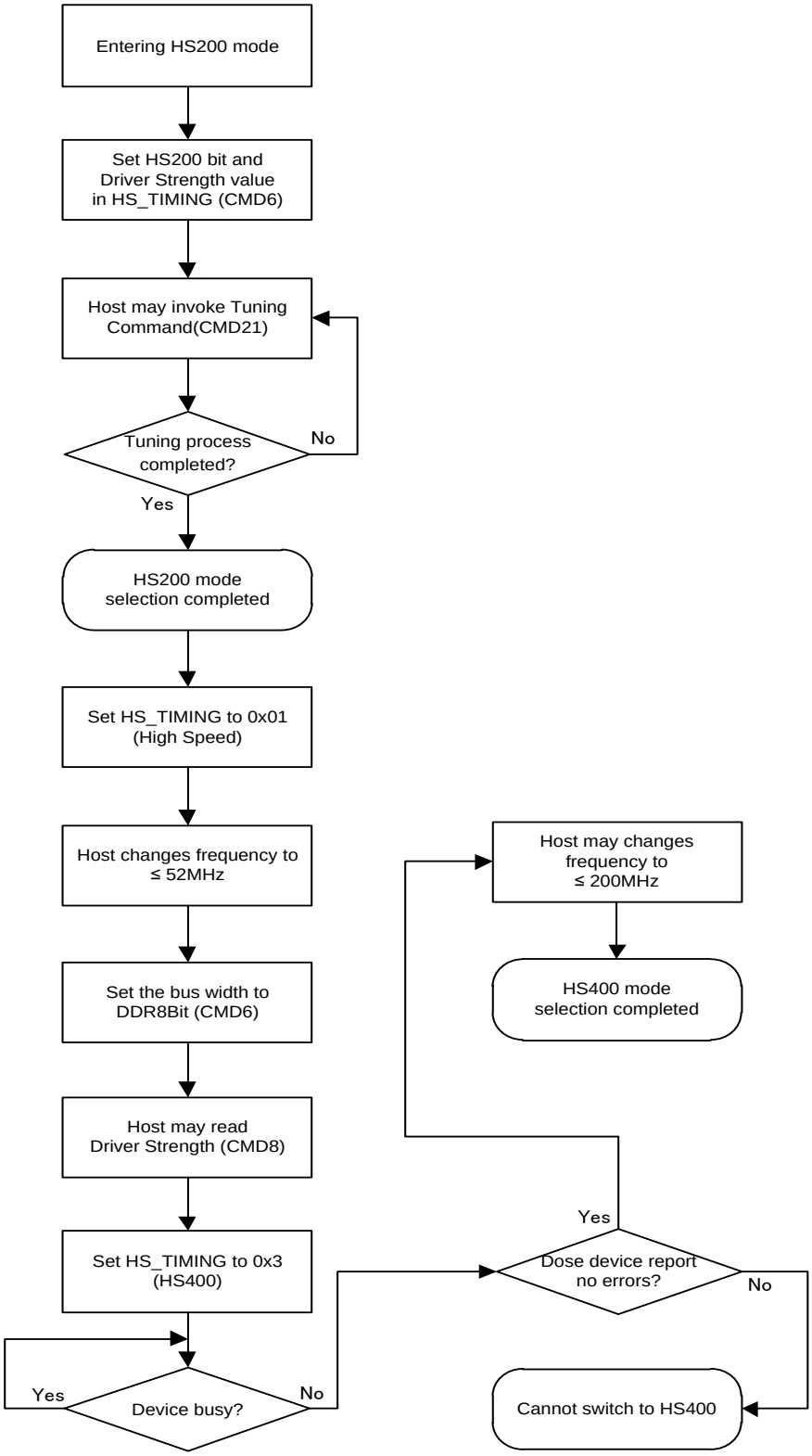


Figure 29 — HS400 Selection flow diagram

6.6.2.3 "HS400" timing mode selection (cont'd)

This selection flow describes how to initialize the *e*•MMC device in HS400 mode while enabling Enhanced Strobe without the need for tuning procedure.

After the host initializes the device, host check whether the device supports the HS400 mode and Enhanced Strobe by reading the DEVICE_TYPE and STROBE_SUPPORT fields in the Extended CSD register.

After power-on or software reset (CMD0), the interface timing of the device is set as the default "Backward Compatible Timing".

In order to switch to HS400 mode with Enhanced Strobe, host should perform the following steps:

1) Initialize device with "Backward Compatible Timings",
2) Select the device with CMD7,
3) Read the DEVICE_TYPE [196] field of the Extended CSD register to validate whether the device supports HS400,
4) Read the STROBE_SUPPORT[184] field of the Extended CSD register to validate whether the device supports Enhanced Strobe,
5) Set the "Timing Interface" parameter in the HS_TIMING [185] field of the Extended CSD register to 0x1 to switch to High Speed mode and then set the clock frequency to a value not greater than 52 MHz,
6) Set BUS_WIDTH[183] to 0x86 to select the dual data rate x8 bus mode and enable Enhanced Strobe (this will be active only after HS400 mode is selected),
7) Read the DRIVER_STRENGTH [197] field of the Extended CSD register to find the supported device Driver Strengths,

   NOTE This step may be skipped if changes of driver strength is not needed, if needed host may change the device Driver Strength.
8) Set the "Timing Interface" parameter in the HS_TIMING [185] field of the Extended CSD register to 0x3 to switch to HS400 mode,
9) Host may set the clock frequency to a value not greater than 200 MHz".

6.6.2.3 "HS400" timing mode selection (cont'd)

Figure 28 illustrates a selection flow of HS400 using Enhanced Strobe.

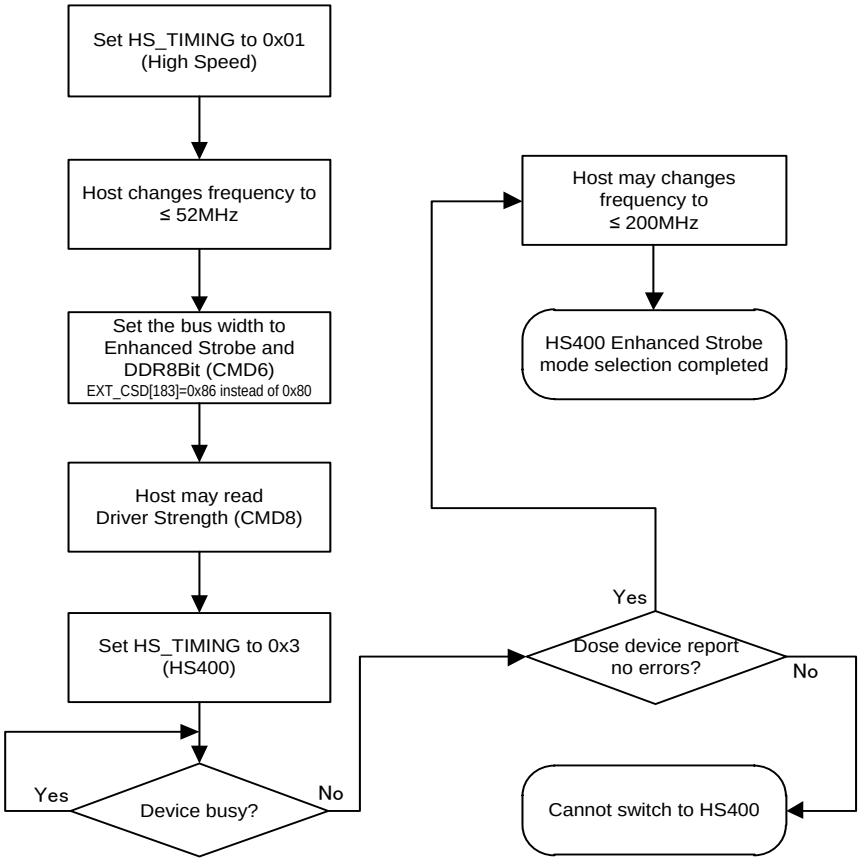


Figure 30 — HS400 (Enhanced Strobe) Selection flow diagram

6.6.3 Power class selection

After the host verifies that the Device complies with v4.0, or higher, of this standard, it may change the power class of the Device. After power-on, or software reset, the Device power class is class 0, which is the default, minimum current consumption class for the Device type, either High Voltage or Dual voltage Device. The PWR_CL_ff_vvv bytes, in the EXT_CSD register, reflect the power consumption levels of the Device, for a 4 bits bus, an 8 bit bus, at the supported clock frequencies (26MHZ,52MHz or 200 MHz).

The host reads this information, using the SEND_EXT_CSD command, and determines if it will allow the Device 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 7.4.55 PWR_CL_ff_vvv [203:200] , PWR_CL_ff_vvv[237:236] , PWR_CL_DDR_ff_vvv [239:238] and PWR_CL_DDR_FF_vvv[253]. If the host tries to write an invalid value, the POWER_CLASS byte is not changed and the SWITCH_ERROR bit is set.

6.6.4 Bus testing procedure

By issuing commands CMD19 and CMD14 in single data rate mode the host can detect the functional pins on the bus. In the dual data rate mode, CMD19 and CMD14 are considered illegal commands. In a first step, the host sends CMD19 to the Device, followed by a specific data pattern on each selected data lines. The data pattern to be sent per data line is defined in Table 7. As a second step, the host sends CMD14 to request the Device to send back the reversed data pattern. With the data pattern sent by the host and with the reversed pattern sent back by the Device, the functional pins on the bus can be detected.

Table 7 — Bus testing pattern

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

The Device ignores all but the first two bits of the data pattern. Therefore, the Device 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 Device, reversed. The data pattern sent by the host may optionally include a CRC16 checksum, this is ignored by the Device.

The Device 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 Device's DAT lines (See Table 8 through Table 10). The reverse data pattern sent by the Device may optionally include a CRC16 checksum, that is ignored by the host.

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

Table 8 — 1-bit bus testing pattern

Data line	Data pattern sent by the host	Reversed pattern sent by the Device	Notes
DAT0	0, 10xxxxxxxxx, [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

6.6.4 Bus testing procedure (cont'd)

Table 9 — 4-bit bus testing pattern

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

6.6.5 Bus Sampling Tuning Concept

Optimal data sampling point in the Host may be found by performing tuning process executed by Host. After Device has entered HS200 mode, the tuning procedure may begin. Using this method, the best sampling point is found by scanning the whole UI with the tuning scheme.

The following is an example of a Host sampling point Tuning scheme (see 5.3.4 for conceptual block diagram of HS200):

1) Sampling Control Block of Host is reset.
2) 'Send Tuning Block' command is issued by Host, to read tuning block.
3) Tuning block is sent by Device as read data. The Host receives it and compares it with a known tuning block pattern.
4) Sampling Control Block of Host is incremented by one step.
5) A read command for the next tuning block is issued by Host.

Steps 3 to 5 should be repeated to cover full UI.

When the entire UI is covered, the Host is able to identify the available valid window. The Host decides the value of the Sampling Control Block (may be set to center of the valid window). After Host sampling point tuning has been completed, Read/Write operations execution may begin.

NOTE This example is given for concept explanation purpose. The Host may use any other implementation according to its design consideration.

6.6.5.1 Sampling Tuning Sequence for HS200

A sampling-tuning sequence may be done by the host when switching to the HS200 mode, to compensate for timing variations due to different silicon processes, PCB loads, voltages, temperatures and other factors. The frequency of use and the implementation of the tuning process will be dependent on the implementation of the host and the system.

Upon host request, the device transmits 128 clocks of data bits containing 64 or 128-bytes block in case of 4 bit or 8 bit, respectively, with a known data pattern to the host, that in turn uses it to find the optimal sampling point for the data lines.

CMD21 (named “Send Tuning Block”) is used for sending the tuning block. CMD21 is followed by R1 type response. CMD21 is valid only in HS200 mode and only when the device is unlocked (Password Lock, see paragraph 6.6.19). In any other state, CMD21 is treated as illegal command. Since the data block following CMD21 is fixed, CMD16 is not required to precede CMD21.

CMD21 follows the timing of a single block read command as described in Figure 31.

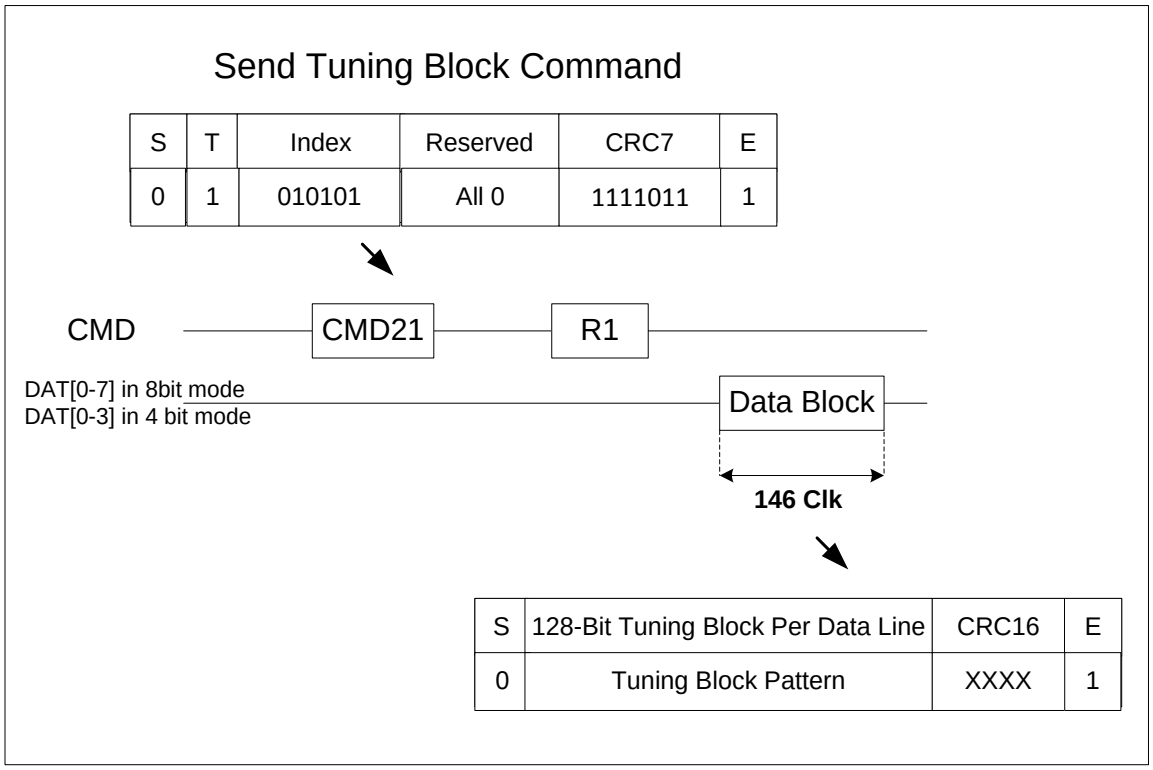


Figure 31 — Send Tuning Command

The host may send a sequence of CMD21s until it has finished its tuning process.

The Device is guaranteed to complete a sequence of 40 times CMD21 executions within 150ms. This is exclusive of any host overhead.

6.6.5.1 Sampling Tuning Sequence for HS200 (cont'd)

Figure 32 defines the tuning block pattern for 8 bit mode. In 4 bit mode the same pattern is used as shown for the lower 4 bits (DAT[3:0]).

	Byte #1	Byte #2	Byte #3	Byte #4	Byte #5	Byte #6	Byte #7	Byte #8	Byte #9	Byte #10	Byte #11	Byte #12	Byte #13	Byte #14	Byte #15	Byte #16	Byte #17	Byte #18	Byte #19	Byte #20	Byte #21	Byte #22	Byte #23	Byte #24	Byte #25	Byte #26	Byte #27	Byte #28	Byte #29	Byte #30	Byte #31	Byte #32
	F	F	0	F	F	F	0	0	F	F	C	C	C	3	C	C	C	3	3	C	C	C	F	F	F	E	F	F	F	E	E	F
	F	F	0	F	F	F	0	0	F	F	C	C	C	3	C	C	C	3	3	C	C	C	F	F	F	E	F	F	F	E	E	F

	Byte #33	Byte #34	Byte #35	Byte #36	Byte #37	Byte #38	Byte #39	Byte #40	Byte #41	Byte #42	Byte #43	Byte #44	Byte #45	Byte #46	Byte #47	Byte #48	Byte #49	Byte #50	Byte #51	Byte #52	Byte #53	Byte #54	Byte #55	Byte #56	Byte #57	Byte #58	Byte #59	Byte #60	Byte #61	Byte #62	Byte #63	Byte #64
	F	F	D	F	F	F	D	D	F	F	F	B	F	F	F	B	B	F	F	F	7	F	F	F	7	7	F	7	B	D	E	F
	F	F	D	F	F	F	D	D	F	F	F	B	F	F	F	B	B	F	F	F	7	F	F	F	7	7	F	7	B	D	E	F

	Byte #65	Byte #66	Byte #67	Byte #68	Byte #69	Byte #70	Byte #71	Byte #72	Byte #73	Byte #74	Byte #75	Byte #76	Byte #77	Byte #78	Byte #79	Byte #80	Byte #81	Byte #82	Byte #83	Byte #84	Byte #85	Byte #86	Byte #87	Byte #88	Byte #89	Byte #90	Byte #91	Byte #92	Byte #93	Byte #94	Byte #95	Byte #96
	F	F	F	0	F	F	F	0	0	F	F	C	C	C	3	C	C	C	3	3	C	C	C	F	F	F	E	F	F	F	E	E
	F	F	F	0	F	F	F	0	0	F	F	C	C	C	3	C	C	C	3	3	C	C	C	F	F	F	E	F	F	F	E	E

	Byte #97	Byte #98	Byte #99	Byte #100	Byte #101	Byte #102	Byte #103	Byte #104	Byte #105	Byte #106	Byte #107	Byte #108	Byte #109	Byte #110	Byte #111	Byte #112	Byte #113	Byte #114	Byte #115	Byte #116	Byte #117	Byte #118	Byte #119	Byte #120	Byte #121	Byte #122	Byte #123	Byte #124	Byte #125	Byte #126	Byte #127	Byte #128
	F	F	F	D	F	F	F	D	D	F	F	F	B	F	F	F	B	B	F	F	F	7	F	F	F	7	7	F	7	B	D	E
	F	F	F	D	F	F	F	D	D	F	F	F	B	F	F	F	B	B	F	F	F	7	F	F	F	7	7	F	7	B	D	E

Figure 32 — Tuning block pattern for 8 bit mode

Figure 33 describes the way the known pattern is spread across DAT[7:0]. It includes only the first 16 bytes for illustration, and the fixed CRC16 value per data line. The order is Left->Right on each data line.

The tuning block purpose is to create "special" signal integrity cases on the bus. This causes high SSO noise, deterministic jitter, ISI and timing errors. Therefore host should switch to the desirable operational bus width before it performs the tuning process.

Therefore, the purpose is to create the worst case data valid window (minimal valid window) that the HS200 system should experience in a specific Host and Device combination.

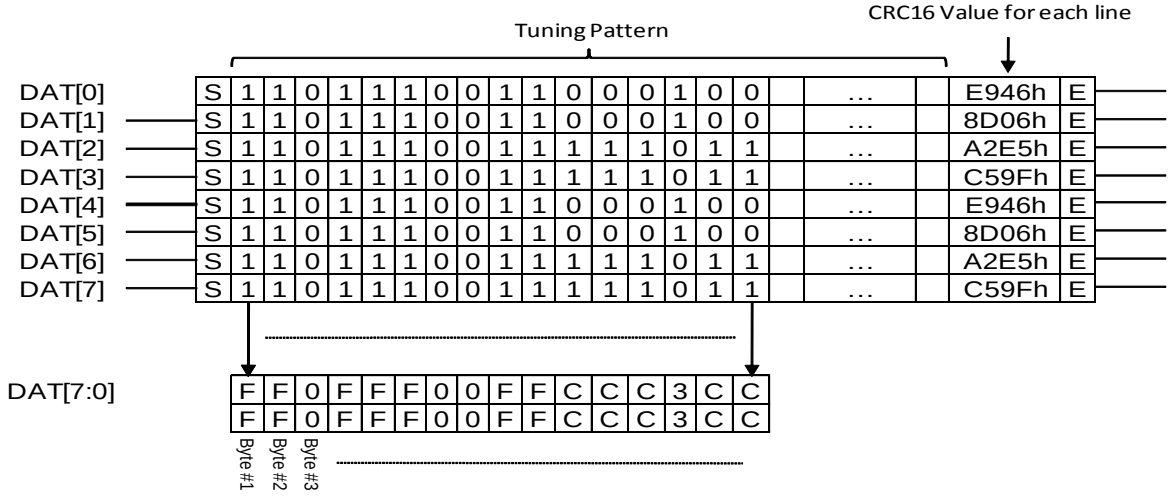


Figure 33 — Tuning block on DAT[7:0]/DAT[3:0] in 8bit/4bit bus width

6.6.6 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 7.4.66. 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.

6.6.7 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-Device ECC is used). The data stream ends with an end bit (HIGH), on each DAT line. (See Figure 42, Figure 43, and Figure 49). The data transmission is synchronous to the clock signal for all interfaces other than HS400, and it is synchronous to the DS signal for HS400 interface.

The payload for block oriented data transfer is protected by one CRC check sum in single data rate mode or by two CRC check sums in dual data rate mode, on each DAT line (See 0).

6.6.7.1 Block read

In single data rate mode the basic unit of data transfer is a block whose maximum size is defined in the CSD (READ_BL_LEN). If READ_BL_PARTIAL is set, smaller blocks whose starting and ending address are entirely contained within one physical block (as defined by READ_BL_LEN) may also be transmitted. A CRC is appended to the end of each block ensuring data transfer integrity. CMD17 (READ_SINGLE_BLOCK) initiates a block read and after completing the transfer, the Device returns to the *Transfer State*. In dual data rate mode, the data size of a block read is always 512 bytes, partial block data read is not supported, and at the end of each block is appended two CRC, one for even bytes and one for odd bytes. 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 Device will continuously transfer data blocks until a stop transmission command is received.

- Multiple block read with pre-defined block count

  The Device 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 Device will start an open-ended multiple block read that can be stopped using the STOP_TRANSMISSION command.

6.6.7.1 Block read (cont'd)

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 occurs, the Device 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 Device physical blocks. ADDRESS_MISALIGN is set.

If the Device 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 Device 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 Device 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 Device 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 Device will respond with the ADDRESS_MISALIGN bit set and return to *Tran* state.

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

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

6.6.8 Data write

The data transfer format of write operation is similar to the data read. For block oriented write data transfer, one CRC check bits in single data rate mode or by two CRC check bits in dual data rate mode are added to each data block. The Device performs a CRC parity check (see 0) for each received data block prior to the write operation. By this mechanism, writing of erroneously transferred data can be prevented.

In general, an interruption to a write process should not cause corruption in existing data at any other address. However, the risk of power being removed during a write operation is different in different applications. Also, for some technologies used to implement *e*•MMC, there is a tradeoff between protecting existing data (e.g., data written by the previous completed write operations), during a power failure, and write performance. The host has the ability to set the type of data reliability desired. When the write data reliability parameters (WR_DATA_REL_USR, WR_DATA_REL_1, WR_DATA_REL_2, WR_DATA_REL_3 and WR_DATA_REL_4) in EXT_CSD (WR_REL_SET) are set to 1 it will indicate to the host that the write mechanism in the associated partition has been implemented to protect existing data in that partition. This means that once a device indicates to the host that a write has successfully completed, the data that was written, along with all previous data written, shall not be corrupted by other operations that are host initiated, controller initiated or accidental. A value of 0 for these bits will indicate that there is some risk to previously written data in those partitions if power is removed. This reliability setting only impacts the reliability of the main user area and the general purpose partitions. The data in the boot partitions and the RPMB partition must have the same reliability that is implied by setting the WR_DATA_REL bit to 1. The reliability of these partitions is not impacted by the value of the WR_DATA_REL bit.

The host has the option of changing the reliability of the writes in one or more partitions on the device. The entire register is considered to be write once so the host has one opportunity to write all of the bits in the register. (Separate writes to change individual bits are not permitted) This write must happen as part of the partitioning process and must occur before the PARTITION_SETTING_COMPLETED bit is set. The changes made to the WR_REL_SET register will not have an impact until the partitioning process is complete (i.e., after the power cycle has occurred and the partitioning has completed successfully). Data reliability settings for partitions that do not exist in the device have no impact on the device.

All write operations must be completed in the order that they arrive. The order restriction is applicable to each write command that a device receives and does not apply to the data within a specific write command.

6.6.8.1 Block write

In single data rate mode, during block write (CMD24 - 27) one or more blocks of data are transferred from the host to the Device with a CRC appended to the end of each block by the host. A Device supporting block write shall always be able to accept a block of data defined by WRITE_BL_LEN. If the CRC fails, the Device 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. In dual data rate mode, the data size of a block write is always 512 bytes, partial block data write is not supported, and at the end of each block is appended two CRC, one for even bytes and one for odd bytes. CMD25 (WRITE_MULTIPLE_BLOCK) starts a transfer of several consecutive blocks. Three types of multiple-block write transactions are defined (the host can use any of these three types at any time):

- Open-ended Multiple-block write

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

- Multiple-block write with pre-defined block count

  The Device 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 Device will start an open-ended multiple-block write that can be stopped using the STOP_TRANSMISSION command.

- Reliable Write: Multiple block write with pre-defined block count and Reliable Write parameters.

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

  - The block size defined by SET_BLOCKLEN(CMD16) is ignored and all blocks are 512 B in length. The data transfers will be in multiple of 512 B sectors or in multiple of eight 512 B sectors if Large Sector size mode is activated. There is no limit on the size of the reliable write.
  - The function is activated by setting the Reliable Write Request parameter (bit 31) to “1” in the SET_BLOCK_COUNT command (CMD23) argument.
  - Reliable write transactions must be sector aligned, if a reliable write is not sector aligned the error bit 19 will be set and the transaction will not complete.
  - If a power loss occurs during a reliable write, each sector being modified by the write is atomic. After a power failure sectors may either contain old data or new data. All of the sectors being modified by the write operation that was interrupted may be in one of the following states: all sectors contain new data, all sectors contain old data or some sectors contain new data and some sectors contain old data.
  - In the case where a reliable write operation is interrupted by a high priority interrupt operation, the sectors that the register marks as completed will contain new data and the remaining sectors will contain old data.
  - The REL_WR_SEC_C [222] register shall be set to 1 and has no impact on the reliable write operation.

6.6.8.1 Block write (cont'd)

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 predefined block count is aborted, the data in the remaining blocks is not defined.

If either one of the following conditions occurs, the Device 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 Device physical blocks. ADDRESS_MISALIGN is set.

If the Device 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 Device 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 Device 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 Device 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 Device 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 Device will report an error and not change any register contents.

Some Devices may require long and unpredictable times to write a block of data. After receiving a block of data and completing the CRC check, the Device will begin writing and hold the DAT0 line low. The host may poll the status of the Device with a SEND_STATUS command (CMD13) at any time, and the Device will respond with its status (except in Sleep state). The status bit READY_FOR_DATA indicates whether the Device can accept new data or not. The host may deselect the Device by issuing CMD7 that will then displace the Device into the *Disconnect* State and release the DAT0 line without interrupting the write operation. When reselecting the Device, it will reactivate busy indication by pulling DAT0 to low. See 6.15 for details of busy indication

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

6.6.9 Erase

In addition to the implicit erase executed by the Device as part of the write operation, provides a host explicit erase function. The erasable unit of the *e•*MMC is the “Erase Group”; Erase group is measured in write blocks that are the basic writable units of the Device. The size of the Erase Group is a Device specific parameter and defined in the CSD when ERASE_GROUP_DEF is disabled, and in the EXT_CSD when ERASE_GROUP_DEF is enabled. The content of an explicitly erased memory range shall be ‘0’ or ‘1’ depending on different memory technology. This value is defined in the EXT_CSD.

Once the erase command completes successfully, the mapped device address range that was erased shall behave as if it was overwritten with all ‘0’ or all ‘1’ depending on the different memory technology. The impact of the erase command should be simply moving the mapped host address range to the unmapped host address range.

NOTE In some cases other flash management tasks may also be completed during the execution of this command.

The host can erase a contiguous range of Erase Groups. Starting the erase process is a three steps sequence. First the host defines the start address of the range using the ERASE_GROUP_START (CMD35) command, next it defines the last address of the range using the ERASE_GROUP_END (CMD36) command and finally it starts the erase process by issuing the ERASE (CMD38) command with argument bits set to zero. See Table 11 for the arguments supported by CMD38. The address field in the erase commands is an Erase Group address, in byte units for densities up to 2GB, and in sector units for densities greater than 2GB. The Device will ignore all LSB's below the Erase Group size, effectively rounding the address down to the Erase Group boundary.

If an erase command (either CMD35, CMD36, CMD38) is received out of the defined erase sequence, the Device shall set the ERASE_SEQ_ERROR bit in the status register and reset the whole sequence. If the host provides an out of range address as an argument to CMD35 or CMD36, the Device will reject the command, respond with the ADDRESS_OUT_OF_RANGE bit set and reset the whole erase sequence.

If an ‘non erase’ command (neither of CMD35, CMD36, CMD38 or CMD13) is received, the Device shall respond with the ERASE_RESET bit set, reset the erase sequence and execute the last command.

Commands not addressed to the selected Device do not abort the erase sequence.

If the erase range includes write protected blocks, they shall be left intact and only the non-protected blocks shall be erased. The WP_ERASE_SKIP status bit in the status register shall be set. As described above for block write, the Device will indicate that an erase is in progress by holding DAT0 low. The actual erase time may be quite long, and the host may issue CMD7 to deselect the Device.

6.6.9 Erase (cont'd)

Table 11 — Erase command (CMD38) Valid arguments

CMD38 Arguments	Command Description	SEC_GB_CL_EN (EXT_CSD[231] bit 4)	SEC_ER_EN (EXT_CSD[231] bit 0)
0x00000000	**Erase:** Erase the erase groups identified by CMD35&36. Controller can perform actual erase at a convenient time. (legacy implementation)	n/a	n/a
0x00000003	**Discard:** Discard the write blocks identified by CMD35 & 36. The controller can perform partial or full the actual erase at a convenient time.	n/a	n/a
0x80000000	**Secure Erase:** Perform a secure purge according to Secure Removal Type in EXT_CSD on the erase groups identified by CMD35 &36 and any copies of those erase groups	n/a	Required
0x80008000	**Secure Trim Step 2:** Perform a secure purge operation on the write blocks according to Secure Removal Type in EXT_CSD and copies of those write blocks that were previously identified using CMD 35,36 and 38 + argument 0x80000001.	Required	Required
0x80000001	**Secure Trim Step 1:** Mark the write blocks, indicated by CMD35&36, for secure erase.	Required	Required
0x00000001	**Trim:** Erase (aka Trim) the write blocks identified by CMD35&36. The controller can perform the actual erase at a convenient time.	Required	n/a

When executing the Erase command the host should note that an erase group contains multiple write blocks that could each contain different pieces of information. When the Erase is executed it will apply to all write blocks within an erase group. Before the host executes the Erase command it should make sure that the information in the individual write blocks no longer needed. So to avoid the deletion of valid data by accident, the Erase command is best used to erase the entire device or a partition. If the host only wishes to purge a single write block a Trim command might be more appropriate.

6.6.10 TRIM

The Trim operation is similar to the default erase operation described in 6.6.9. The Trim function applies the erase operation to write blocks instead of erase groups. The Trim function allows the host to identify data that is no longer required so that the Device can erase the data if necessary during background erase events. The contents of a write block where the trim function has been applied shall be '0' or '1' depending on different memory technology. This value is defined in the EXT_CSD.

Once the trim command completes successfully, the mapped device address range that was trimmed shall behave as if it was overwritten with all ‘0’ or all ‘1’ depending on the different memory technology. The impact of the trim command should be simply moving the mapped host address range to the unmapped host address range.

NOTE In some cases other flash management tasks may also be completed during the execution of this command.

Completing the TRIM process is a three steps sequence. First the host defines the start address of the range using the ERASE_GROUP_START (CMD35) command, next it defines the last address of the range using the ERASE_GROUP_END (CMD36) command and finally it starts the erase process by issuing the ERASE (CMD38) command with argument bit 0 set to one and the remainder of the arguments set to zero. In the case of a TRIM operation both CMD35 and CMD36 identify the addresses of write blocks rather than erase groups.

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

If the host provides an out of range address as an argument to CMD35 or CMD36, the Device will reject the command, respond with the ADDRESS_OUT_OF_RANGE bit set and reset the whole erase sequence. If a “non erase” command (neither of CMD35, CMD36, CMD38 or CMD13) is received, the Device shall respond with the ERASE_RESET bit set, reset the erase sequence and execute the last command. Commands not addressed to the selected Device do not abort the erase sequence.

If the trim range includes write protected blocks, they shall be left intact and only the non-protected blocks shall be erased. The WP_ERASE_SKIP status bit in the status register shall be set. As described above for block write, the Device will indicate that a Trim command is in progress by holding DAT0 low. The actual erase time may be quite long, and the host may issue CMD7 to deselect the Device.

The host should execute the Trim command with caution to avoid unintentional data loss.

Resetting a Device (using CMD0, CMD15, or hardware reset for *e*•MMC) or power failure will terminate any pending or active Trim command. This may leave the data involved in the operation in an unknown state

6.6.11 Sanitize

The Sanitize operation is a feature, in addition to TRIM and Erase that is used to remove data from the device according to Secure Removal Type (see 7.4.120). The use of the Sanitize operation requires the device to physically remove data from the unmapped user address space. A Sanitize operation is initiated by writing a value to the extended CSD[165] SANITIZE_START. While the device is performing the sanitize operation, the busy line is asserted. The device will continue the sanitize operation, with busy asserted, until one of the following events occurs:

- Sanitize operation is complete,
- An HPI is used to abort the operation,
- A power failure, or
- A hardware reset.

After the sanitize operation is completed, no data should exist in the unmapped host address space.

If the sanitize operation is interrupted, either by HPI, power failure, CMD0 or hardware reset, the state of the unmapped host address space cannot be guaranteed. The host must re-initiate the sanitize operation by writing to the SANITIZE_START[165] and allow the operation to complete to be sure that unmapped host address space is clear.

Since the region being operated on is not accessible by the host, applications requiring this feature must work with individual device manufacturers to ensure this operation is performing properly and to understand the impact on device reliability.

6.6.12 Discard

The Discard is similar operation to TRIM. The Discard function allows the host to identify data that is no longer required so that the device can erase the data if necessary during background erase events. The contents of a write block where the discard function has been applied shall be ‘don’t care’. After discard operation, the original data may be remained partially or fully accessible to the host dependent on device. The portions of data that are no longer accessible by the host may be removed or unmapped just as in the case of TRIM. The device will decide the contents of discarded write block.

The distinction between Discard and TRIM, is that a read to a region that was discarded may return some or all of the original data. However, in the case of Trim the entire region shall be unmapped or removed and will return ‘0’ or ‘1’ depending on the memory technology.

When Sanitize is executed, only the portion of data that was unmapped by a Discard command shall be removed by the Sanitize command. The device cannot guarantee that discarded data is completely removed from the device when Sanitize is applied.

Completing the Discard process is a three steps sequence. First the host defines the start address of the range using the ERASE_GROUP_START (CMD35) command, next it defines the last address of the range using the ERASE_GROUP_END (CMD36) command and finally it starts the erase process by issuing the ERASE (CMD38) command with argument bit 0 and bit 1 set to one and the remainder of the arguments set to zero. In the case of a Discard operation both CMD35 and CMD36 identify the addresses of write blocks rather than erase groups.

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

6.6.12 Discard (cont'd)

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

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

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

As described above for block write, the device will indicate that a Discard command is in progress by holding DAT0 low. The actual erase time may be quite long, and the host may issue CMD7 to deselect the device.

The host should execute the Discard command with caution to avoid unintentional data loss.

Resetting a device (using CMD0, CMD15, or hardware reset for *e*•MMC) or power failure will terminate any pending or active Discard command. This may leave the data involved in the operation in an unknown state.

6.6.13 Secure Erase

NOTE Secure Erase is included for backwards compatibility. New system level implementations (based on v4.51 devices and beyond) should use Erase combined with Sanitize instead of secure erase.

In addition to the standard Erase command there is also an optional Secure Erase command. The Secure Erase command differs from the basic Erase command (outlined in 6.6.9) in that it requires the device to execute the erase operation on the memory array when the command is issued and requires the device and host to wait until the operation is complete before moving to the next device operation. Also, the secure erase command requires the device to do a secure purge operation, according to Secure Removal Type in EXT_CSD, outlined in 7.4.120, on the erase groups, and any copies of items in those erase groups, identified for erase.

This command allows applications that have high security requirements to request that the device perform secure operations, while accepting a possible erase time performance impact. The Secure Erase command is executed in the same way the erase command outlined in 6.6.9, except that the Erase (CMD38) command is executed with argument bit 31 set to 1 and the other argument bits set to zero. See Table 11 for details on the argument combinations supported with the Erase (CMD 38) command and Table 12 for the definition of the argument bits associated with the ERASE (CMD38) command.

The host should execute the Secure Erase command with caution to avoid unintentional data loss.

Resetting a device (using CMD0, CMD15, or hardware reset for *e*•MMC) or power failure will terminate any pending or active Secure Erase command. This may leave the data involved in the operation in an unknown state.

Table 12 — Erase command (CMD38) Valid arguments

Bit	Argument	Arguments
31	Secure Request	'1' - Secure form of the command must be performed. '0' - Default in Secure Erase command is performed
15	Force Garbage Collection	'1' – CMD35 & 36 are ignored. Erase is performed on previously identified write blocks. '0' - Command uses the erase groups identified by CMD35&36
1	Discard Enable	'1' – Discard write blocks identified by CMD35 & 36. '0' – Execute an erase. NOTE Bit[0] is also required to be set to 0x1 (in addition to bit[1] being set to 0x1) for the Discard write blocks operation
0	Identify Write Blocks for Erase (or TRIM Enable)	'1' – Mark write block identified by CMD35&36 for erase (bit 31 set) or execute Trim operation (bit 31 cleared). Bit[0] shall be set to 0x1 for the Discard write blocks operation. '0' - Execute an erase.

6.6.14 Secure Trim

NOTE: Secure TRIM is included for backwards compatibility. New system level implementations (based on v4.51 devices and beyond) should use TRIM combined with Sanitize instead of secure trim.

The Secure Trim command is very similar to the Secure Erase command. The Secure Trim command performs a secure purge operation on write blocks instead of erase groups. To minimize the impact on the device's performance and reliability the Secure Trim operation is completed by executing two distinct steps.

In Secure Trim Step 1 the host defines the range of write blocks that it would like to mark for the secure purge. This step does not perform the actual purge operation. The blocks are marked by defining the start address of the range using the ERASE_GROUP_START (CMD35) command, followed by defining the last address of the range using the ERASE_GROUP_END (CMD36) command. In the case of Secure Trim, both ERASE_GROUP_START and ERASE_GROUP_END arguments are identifying write block addresses. Once the range of blocks has been identified the ERASE (CMD 38) with argument bit 31and 0 set to 1 and the remainder of the argument bits set to 0 (See Table 11for the allowable arguments) is applied. This completes Secure Trim Step 1.

Secure Trim Step 1 can be repeated several times, with other commands being allowed in between, until all the write blocks that need to be purged have been identified. It is recommended that the Secure Trim Step 1 is done on as many blocks as possible to improve its efficiency of the secure trim operation.

Secure Trim Step 2 issues the ERASE_GROUP_START (CMD35) and ERASE_GROUP_END (CMD36) with addresses that are in range. Note the arguments used with these commands will be ignored. Then the ERASE (CMD 38) with bit 31 and 15 set to 1 and the remainder of the argument bits to 0 is sent. This step actually performs the secure purge on all the write blocks, according to Secure Removal Type, outlined in 7.4.120, as well as any copies of those blocks, that were marked during Secure Trim Step 1 and completes the secure trim operation. Other commands can be issued to the device in between Secure Trim Step 1 and Secure Trim Step 2.

The host may issue Secure Trim Step 2 without issuing Secure Trim Step 1. This may be required after a power failure event in order to complete unfinished secure trim operations. If Secure Trim Step 2 is done and there are no write blocks marked for erase then Secure Trim Step 2 again will have no impact on the device.

Once a write block is marked for erase using Secure Trim Step 1, it is recommended that the host consider this block as erased. However, if the host does write to a block after it has been marked for erase, then the last copy of the block, that occurred as a result of the modification, will not be marked for erase. All previous copies of the block will remain marked for erase.

If the host application wishes to use the secure TRIM command as the method to remove data from the device, then the host should make sure that it completes secure trim step 1 for a write block before using a write command to overwrite the block. This will ensure that the overwritten data is removed securely from the device the next time that secure Trim step 2 is issued.

If either CMD35, CMD36 or CMD38 is received out of the defined erase sequence, the device shall set the ERASE_SEQ_ERROR bit in the status register and reset the whole sequence for an individual step. If the host provides an out of range address as an argument to CMD35 or CMD36, the device will reject the command, respond with the ADDRESS_OUT_OF_RANGE bit set and reset the whole erase sequence.

6.6.14 Secure Trim (cont'd)

If a 'non erase' command (neither of CMD35, CMD36, CMD38 or CMD13) is received, while the device is performing the operations in Secure Trim Step 1 or Secure Trim Step 2, the device shall respond with the ERASE_RESET bit set, and reset the individual step without completing the operation and execute the last command. Commands not addressed to the selected device do not abort the sequence. Other commands may occur in between the multiple iterations of Secure Trim Step 1 and/or before Secure Trim Step 2 is sent. However, the sequence during each of the steps cannot be interrupted.

If a power failure or reset occurs in between Secure Trim Step 1 and Secure Trim Step 2. The blocks that were identified for the secure purge operation will remain marked. The next time the device sees Secure Trim Step 2 it will purge the blocks that were marked prior to the power failure or reset and along with any blocks that have been identified since that point.

The host should execute the Secure Trim command with caution to avoid unintentional data loss. Resetting a device (using CMD0, CMD15, or hardware reset for *e*•MMC) or power failure during either step 1 or step2 will terminate any pending or active secure trim command. This may leave the data involved in the operation in an unknown state. If the erase range includes write protected blocks, they shall be left intact and only the non-protected blocks shall be erased. The WP_ERASE_SKIP status bit in the status register shall be set.

If the device needs a write block that is marked for Secure Erase and Secure Trim Step 2 has not been issued, the device can issue a secure purge operation on that write block as a background task. As described above for block write, the device will indicate that either Secure Trim Step 1 or Secure Trim Step 2 are in progress by holding DAT0 low. The actual time for the operation may be quite long, and the host may issue CMD7 to deselect the device. If the write block size is changed in between Secure Trim Step 1 and Secure Trim Step 2 then the write block size used during step 1 of the operation will apply.

6.6.15 Write protect management

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

- The entire Device (including the Boot Area Partitions, General Purpose Area Partition, RPMB, and User/Enhanced User Data Area Partition) may be write-protected by setting the permanent or temporary write protect bits in the CSD. When permanent protection is applied to the entire Device it overrides all other protection mechanisms that are currently enabled on the entire Device or in a specific segment. CSD Register bits and Extended CSD Register bits are not impacted by this protection. When temporary write protection is enabled for the entire Device it only applies to those segments that are not already protected by another mechanism. See Table 13 for details.
- Specific segments of the Devices may be permanent, power-on or temporarily write protected. ERASE_GROUP_DEF in EXT_CSD decides the segment size. When set to 0, the segment size is defined in units of WP_GRP_SIZE erase groups as specified in the CSD. When set to 1, the segment size is defined in units of HC_WP_GRP_SIZE erase groups as specified in the EXT_CSD. If host does not set ERASE_GROUP_DEF bit for a device of which high capacity write protect was already set in some of the area in the previous power cycle, then the device may show unknown behavior when host issue write or erase commands to the device, because the write protect group size previously set mismatches the current write protect group size. Similarly if the host set ERASE_GROUP_DEF bit for a device that the default write protect was already set in some of the area in the previous power cycle, then the device may show unknown behavior when host issue write or erase commands to the device. In application, it is mandatory for host to use same ERASE GROUP DEF value to the device all the time.

6.6.15 Write protect management (cont'd)

- The SET_WRITE_PROT command sets the write protection of the addressed write-protect group, group to the type of write protection dictated by the US_PERM_WP_EN (EXT_CSD[171] bit 2) and US_PWR_WP_EN (EXT_CSD[171] bit 0). See Table 13 for the result of the SET_WRITE_PROT(CMD28) command when protection is already enabled in a segment and Table 14 for impact of the different combinations of the protection enable bits. The CLR_WRITE_PROT command clears the temporary write protection of the addressed write-protect group. If the CLR_WRITE_PROT command is applied to a write protection group that has either permanent or power-on write protection then the command will fail.

The ability for the host to set permanent protection for the entire Device and permanent and power-on protection for specific segments can be disabled by setting the CD_PERM_WP_DIS, US_PERM_WP_DIS and US_PWR_WP_DIS bit in the EXT_CSD USER_WP byte. It is recommended to disable all protection modes that are not needed to prevent unused write protection modes from being set maliciously or unintentionally.

The host has the ability to check the write protection status of a given segment or segments by using the SEND_WRITE_PROT_TYPE command (CMD31). When full Device protection is enabled all the segments will be shown as having permanent protection.

The SEND_WRITE_PROT and SEND_WRITE_PROT_TYPE commands are similar to a single block read command. The Device shall send a data block containing 32 or 64 write protection bits, respectively, (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits. The address field in the write protect commands is a group address in byte units, for densities up to 2GB, and in sector units for densities greater than 2 GB.

The Device will ignore all LSBs below the group size for densities up to 2 GB. If the host provides an out of range address as an argument to CMD28, CMD29 or CMD30, the Device will reject the command, respond with the ADDRESS_OUT_OF_RANGE bit set and remain in the *Tran* state.

Table 13 — Write Protection Hierarchy (when disable bits are clear)

Current Protection mode	CSD[12]	Action	Resulting Protection mode
Permanent	N/A	Power failure or hardware reset	Permanent
Permanent	N/A	SET_WRITE_PROT (US_PERM_WP_EN = 0)	Permanent
Power-On	1	Power failure or hardware reset	Temporary
Power-On	0	Power failure or hardware reset	None
Power-On	N/A	SET_WRITE_PROT (US_PERM_WP_EN = 1)	Permanent
Power-On	N/A	SET_WRITE_PROT (US_PERM_WP_EN = 0 and US_PWR_WP_EN = 0)	Power-On
Temporary	1	Power failure or hardware reset	Temporary
Temporary	1	SET_WRITE_PROT (US_PERM_WP_EN = 1)	Permanent
Temporary	1	SET_WRITE_PROT (US_PERM_WP_EN = 0 and US_PWR_WP_EN = 1)	Power-On

Table 14 — Write Protection Types (when disable bits are clear)

US_PERM_WP_EN	US_PWR_WP_EN	Type of protection set by SET_WRITE_PROT command
0	0	Temporary
0	1	Power-On
1	0	Permanent
1	1	Permanent

6.6.16 Extended Security Protocols Pass Through Commands

The extended protocol pass through commands is a feature that allows the host to communicate with an *e*•MMC device with other commonly used security protocols over *e*•MMC protocol using two pass through commands – CMD53 and CMD54.

In order to support Extended Security Protocols communication with the device the following three requirements are essential:

1) Discovery whether the extended protocol pass through commands (CMD53-PROTOCOL_RD and CMD54-PROTOCOL_WR) supported in the device,
2) Receive Extended Protocol data from the device,
3) Send Extended Protocol data to the device.

In order to discover whether the extended protocol pass through commands are supported the host should verify that Command Class 10 is supported by the device (in CCC field in CSD Register).

In order to receive and send extended protocol information CMD53 and CMD54 shall be used.

NOTE CMD53 and CMD54 are the basic infrastructure in *e*•MMC standard that allows the pass through transfer of other protocols. The other protocols that may be implemented over *e*•MMC are referenced in a separate JEDEC Security Extension Standard*.

6.6.16.1 PROTOCOL_RD - CMD53

This command works in similar fashion to the Read Multiple Block command (CMD18) and retrieves security protocol related information from the device. The Security Protocol being referenced is passed in the command arguments. The number of blocks to be transferred shall be provided in advance by using CMD23 (SET_BLOCK_COUNT). Sector size is fix 512Bytes regardless of the NATIVE_SECTOR_SIZE or DATA_SECTOR_SIZE defined in EXT_CSD. Refer to Table 58 for the detailed description of CMD53 fields.

6.6.16.2 PROTOCOL_WR - CMD54

This command works in similar fashion to the multi block Write Multiple Block command (CMD25) and sends security protocol related information to the device. The Security Protocol being referenced is passed in the command arguments. The number of blocks to be transferred shall be provided in advance by using CMD23 (SET_BLOCK_COUNT). Sector size is fix 512Bytes regardless of the NATIVE_SECTOR_SIZE or DATA_SECTOR_SIZE defined in EXT_CSD. The Busy Programming timeout for this command is defined as max 10Sec and it is non interruptible with HPI. In case HPI is used during the Busy Programming time of CMD54 it shall be ignored by device.

Refer to Table 58 for the detailed description of CMD54 fields.

6.6.16.3 Security Protocol Type

The meaning and coding of the Security Protocol argument fields in CMD53 and CMD54 shall be identical to the SECURITY PROTOCOL field in the definition of the SCSI commands SECURITY PROTOCOL IN and SECURITY PROTOCOL OUT respectively, as defined in the INCITS,T10 SPC-4 document. For further information on the Extended Security supported protocols refer to a separate JEDEC Security Extension Standard*.

* As of publication of this document, JEDEC has not published the referenced standard.

6.6.16.4 Security Protocol Information

When received PROTOCOL_RD (CMD53) with Security protocol value equal to 00h device shall return the list of supported protocols as described in Table 15.

Table 15 — Security Protocol Information

Byte Number	Value
0	Reserved
1	Reserved
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	List length (MSB)
7	List length (LSB)
8	Supported security protocols list
…	
M	
M+1	Padding
…	Padding
N	Padding

The values shall be in ascending order starting with 00h. Pad Bytes shall have value ‘0’.

NOTE Devices that support several Security Systems may not support them simultaneously during the same power on session.

6.6.16.5 Error handling

Bit4 of EXCEPTION_EVENTS_STATUS field in the EXT_CSD named EXTENDED_SECURITY_FAILURE indicates error occurrence while usage of PROTOCOL_RD and PROTOCOL_WR commands or in relation to the Extended Protocols usage. In case of such error the host may read the detailed cause of the error in EXT_SECURITY_ERR[505] field of EXT_CSD.

6.6.17 Production State Awareness

e•MMC device could utilize the information of whether it is in production environment and operate differently than it operates in the field.

For example, content that was loaded into the storage device prior to soldering might get corrupted, at higher probability, during device soldering. The *e*•MMC device could use “special” internal operations for loading content prior to device soldering that would reduce production failures and use “regular” operations post-soldering.

PRODUCTION_STATE_AWARENESS [133] field in extended CSD is used as a mechanism through which the host should report to the device whether it is pre or post soldering state.

This standard defines two methods, Manual Mode and Auto Mode, to manage the device production state.

6.6.17 Production State Awareness (cont'd)

The trigger for starting or re-starting the process is setting correctly PRE_LOADING_DATA_SIZE field. Before setting this field the host is expected to make sure that the device is clean and any data that was written before to the device is expected to be erased using CMD35, CMD36 and CMD38.

In case the host erased data, overrode existing data or performed re-partition during production state awareness it should restart the production state awareness process by re-setting PRE_LOADING_DATA_SIZE.

6.6.17.1 Manual Mode

Using Manual Mode method the host explicitly sets the production state by changing the PRODUCTION_STATE_AWARENESS field. The host should set to PRE_SOLDERING_WRITES state when the device is in production prior soldering and before the host loaded content to the device. When loading of content is complete the host should set to PRE_SOLDERING_POST_WRITES state. In this state the host should not write content to the device until soldering is performed. Once soldered, the host should set PRODUCTION_STATE_AWARENESS to Normal state.

In this mode the host is not expected to write any data to the device until PRODUCTION_STATE_AWARENESS is set to PRE_SOLDERING_WRITES.

In case the feature is enabled and the host writes to the device not in PRE_SOLDERING_WRITES state the device may response with an error.

PRODUCTION_STATE_AWARENESS state change may exceed the switch command timeout. The maximum timeout for PRODUCTION_STATE_AWARENESS state change is defined by the device manufacturer in PRODUCTION_STATE_AWARENESS_TIMEOUT [218].

6.6.17.2 Auto Mode

Using Auto Method the host sets the amount of data that would be loaded to the device prior to soldering. The host sets PRODUCTION_STATE_AWARENESS to AUTO_PRE_SOLDERING to signal the device that the Auto mode is used. Once all data have been loaded, the device automatically changes the PRODUCTION_STATE_AWARENESS back to Normal state.

In this mode the host is expected to write data to the device only after it sets PRODUCTION_STATE_AWARENESS to AUTO_PRE_SOLDERING. In case the feature is enabled and the host writes to the device not in AUTO_PRE_SOLDERING state the device may response with an error.

After PRE_LOADING_DATA_SIZE amount of data has been loaded the device switches automatically PRODUCTION_STATE_AWARENESS to Normal.

If the host continues to write data in Normal state (after writing PRE_LOADING_DATA_SIZE amount of data) and before soldering occurred device may respond with an error.

If a power cycle occurred during the data transfer, the amount of data written to device is not cleared. Therefore in this case, host should erase the entire pre-loaded data and set PRE_LOADING_DATA_SIZE to the desired size.

6.6.17.2 Auto Mode (cont'd)

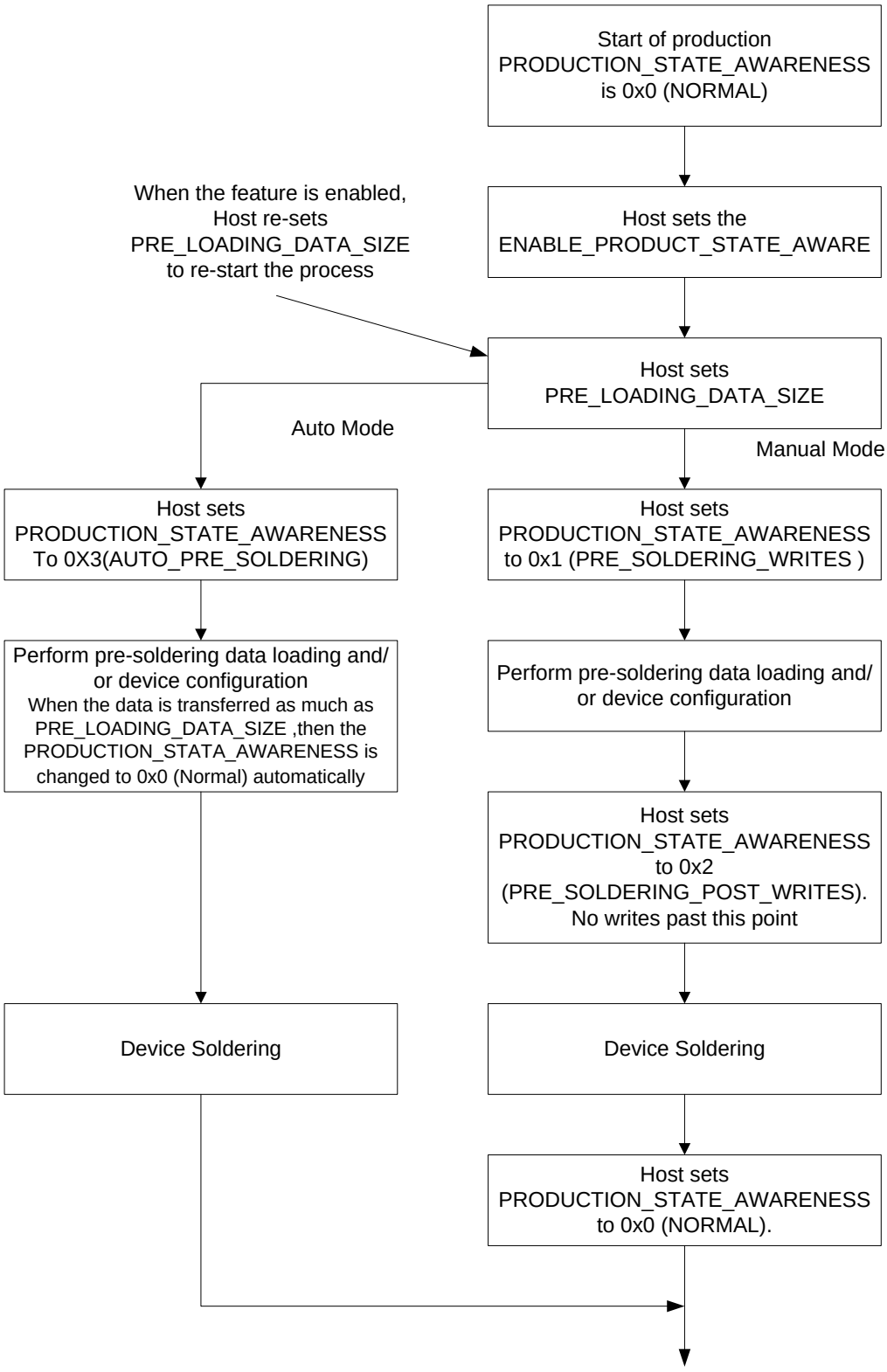


Figure 34 — Recommended Soldering procedure

6.6.18 Field Firmware Update

Field Firmware Updates (FFU) enables features enhancement in the field. Using this mechanism the host downloads a new version of the firmware to the *e*•MMC device and, following a successful download, instructs the *e*•MMC device to install the new downloaded firmware into the device.

In order to start the FFU process the host first checks if the *e*•MMC device supports FFU capabilities by reading SUPPORTED_MODES and FW_CONFIG fields in the EXT_CSD. If the *e*•MMC device supports the FFU feature the host may start the FFU process. The FFU process starts by switching to FFU Mode in MODE_CONFIG field in the EXT_CSD. In FFU Mode host should use closed-ended or open ended commands (CMD17/CMD18/CMD24/CMD25) for downloading the new firmware and reading vendor proprietary data. In this mode, the host should set the argument of these commands to be as defined in FFU_ARG field. In case these commands have a different argument the device behavior is not defined and the FFU process may fail. The host should set Block Length to be DATA_SECTOR_SIZE. Downloaded firmware bundle must be DATA_SECTOR_SIZE size aligned (internal padding of the bundle might be required).

Once in FFU Mode the host may send the new firmware bundle to the device using one or more write commands.

The host could regain regular functionally of write and read commands by setting MODE_CONFIG field in the EXT_CSD back to Normal state. Switching out of FFU Mode may abort the firmware download operation. When switched back-in to FFU Mode, the host should check the FFU Status to get indication about the number of sectors that were downloaded successfully by reading the NUMBER_OF_FW_SECTORS_CORRECTLY_PROGRAMMED in the extended CSD. In case the number of sectors that were downloaded successfully is zero the host should re-start downloading the new firmware bundle from its first sector. In case the number of sectors that were downloaded successfully is positive the host should continue the download from the next sector, that would resume the firmware download operation.

In case MODE_OPERATION_CODES field is supported by the device (defined in SUPPORTED_MODE_OPERATION_CODES bit in FFU_FEATURES field in extended CSD) the host may abort the firmware download process by setting MODE_OPERATION_CODES to FFU_ABORT to inform the device the FFU operation has been aborted. In response the device shall set NUMBER_OF_FW_SECTORS_CORRECTLY_PROGRAMMED field to zero and be ready to receive a new firmware bundle. If the host finished downloading successfully the firmware bundle to the device it may set MODE_OPERATION_CODES to FFU_INSTALL, the device shall set NUMBER_OF_FW_SECTORS_CORRECTLY_PROGRAMMED field to zero, install the new firmware and set MODE_CONFIG to Normal state that would regain regular operation of read and write commands. The host should set MODE_OPERATION_CODES to FFU_INSTALL. If the host performs a CMD0/HW_Reset/Power cycle through the FFU process, prior to successful execution of the FFU_INSTALL command, device may abort the download process.

In case MODE_OPERATION_CODES field is not supported by the device the host sets to NORMAL state and initiates a CMD0/HW_Reset/Power cycle to install the new firmware. In such case the device doesn't need to use NUMBER_OF_FW_SECTORS_CORRECTLY_PROGRAMMED.

In both cases occurrence of a CMD0/HW_Reset/Power cycle before the host successfully downloaded the new firmware bundle to the device may cause the firmware download process to be aborted.

When in FFU_MODE and host sends other commands that are not part of the recommended flow, device behavior may be undefined.

6.6.19 Device lock/unlock operation

The password protection feature enables the host to lock the Device by providing a password that later will be used for unlocking the Device. The password and its size are kept in a 128 bit PWD and 8 bit PWD_LEN registers, respectively. These registers are nonvolatile so that a power cycle will not erase them.

The password protection feature can be disabled permanently by setting the permanent password disable bit in the extended CSD (PERM_PSWD_DIS bit in the EXT_CSD byte [171]). If the host attempts to permanently disable CMD42 features on an *e*•MMC having a password set, the action will fail and the ERROR (bit 19) error bit will be set by the memory device in the Device status register It is recommended to disable the password protection feature on the Device, if it is not required, to prevent it being set unintentionally or maliciously.

An attempt to use password protection features (CMD42) on a Device having password permanently disabled will fail and the LOCK_UNLOCK_FAILED (bit 24) error bit will be set in the status register.

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

On v4.3 and later version devices, the Lock command can be applied to user data area only. So the host can still access the boot partitions, RPMB and General partition area after Lock command is issued to the devices.

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

The Device lock/unlock command (CMD42) has the structure and bus transaction type of a regular single block write command. The transferred data block includes all the required information of the command (password setting mode, PWD itself, Device lock/unlock etc.). Table 16 describes the structure of the command data block.

The Device lock/unlock command (CMD42) can only be performed when the Device operates in single data rate mode. CMD42 is an illegal command in dual data rate mode.

Table 16 — Lock Device data structure

Byte #	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved				ERASE	LOCK_UNLOCK	CLR_PWD	SET_PWD
1	PWD_LEN							
2	Password data							
…								
PWD_LEN +1								

- **ERASE:** '1' Defines Forced Erase Operation (all other bits shall be '0') and only the cmd byte is sent.
- **LOCK/UNLOCK**: '1' = Locks the Device. '0' = Unlock the Device.

  NOTE It is valid to set this bit together with SET_PWD but it is not allowed to set it together with CLR_PWD.
- **CLR_PWD**: '1' = Clears PWD.
- **SET_PWD**: '1' = Set new password to PWD
- **PWD_LEN**: Defines the following password length (in bytes). Valid password lengths are 1 to 16 bytes.
- **PWD:** The password (new or currently used depending on the command).

6.6.19 Device lock/unlock operation (cont'd)

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

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

Setting the password:

- Select the Device (CMD7), if not previously selected already
- Define the block length (CMD16), given by the 8bit Device lock/unlock mode, the 8 bits password size (in bytes), and the number of bytes of the new password. In case that a password *replacement* is done, then the block size shall consider that both passwords, the old and the new one, are sent with the command.
- Send Device Lock/Unlock command (CMD42) with the appropriate data block size on the data line including 16 bit CRC. The data block shall indicate the mode (SET_PWD), the length (PWD_LEN) and the password itself. In case that a password *replacement* is done, then the length value (PWD_LEN) shall include both passwords, the old and the new one, and the PWD field shall include the old password (currently used) followed by the new password. Each of the old and the new password length should not exceed 16 Bytes. In case that the new password length exceeds 16 Bytes, the LOCK_UNLOCK_FAILED error bit is set in the status register and the old password is not changed.
- In case that a password replacement is attempted with PWD_LEN set to the length of the old password only, the LOCK_UNLOCK_FAILED error bit is set in the status register and the old password is not changed.
- In case that the sent old password is not correct (not equal in size and content) then LOCK_UNLOCK_FAILED error bit will be set in the status register and the old password does not change. In case that PWD matches the sent old password then the given new password and its size will be saved in the PWD and PWD_LEN fields, respectively.

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

Reset the password:

- Select the Device (CMD7), if not previously selected already
- Define the block length (CMD16), given by the 8 bit Device lock/unlock mode, the 8 bit password size (in bytes), and the number of bytes of the currently used password.
- Send the Device lock/unlock command (CMD42) with the appropriate data block size on the data line including 16 bit CRC. The data block shall indicate the mode CLR_PWD, the length (PWD_LEN) and the password (PWD) itself (LOCK/UNLOCK bit is don't care). If the PWD and PWD_LEN content match the sent password and its size, then the content of the PWD register is cleared and PWD_LEN is set to 0. If the password is not correct then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

6.6.19 Device lock/unlock operation (cont'd)

Locking the Device:

- Select the Device (CMD7), if not previously selected already
- Define the block length (CMD16), given by the 8 bit Device lock/unlock mode, the 8 bit password size (in bytes), and the number of bytes of the currently used password.
- Send the Device lock/unlock command (CMD42) with the appropriate data block size on the data line including 16 bit CRC. The data block shall indicate the mode LOCK, the length (PWD_LEN) and the password (PWD) itself.
- If the PWD content equals to the sent password then the Device will be locked and the Device-locked status bit will be set in the status register. If the password is not correct then LOCK_UNLOCK_FAILED error bit will be set in the status register.

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

If the password was previously set (PWD_LEN is not '0'), then the Device will be locked automatically after power on reset. An attempt to lock a locked Device or to lock a Device that does not have a password will fail and the LOCK_UNLOCK_FAILED error bit will be set in the status register.

Unlocking the Device:

- Select the Device (CMD7), if not previously selected already.
- Define the block length (CMD16), given by the 8 bit Device lock/unlock mode, the 8 bit password size (in bytes), and the number of bytes of the currently used password.
- Send the Device lock/unlock command (CMD42) with the appropriate data block size on the data line including 16 bit CRC. The data block shall indicate the mode UNLOCK, the length (PWD_LEN) and the password (PWD) itself.

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

NOTE The unlocking is done only for the current power session. As long as the PWD is not cleared the Device will be locked automatically on the next power up. The only way to unlock the Device is by clearing the password. An attempt to unlock an unlocked Device will fail and LOCK_UNLOCK_FAILED error bit will be set in the status register.

6.6.19 Device lock/unlock operation (cont'd)

Forcing erase:

- In case that the user forgot the password (the PWD content) it is possible to erase all the Device data content along with the PWD content. This operation is called *Forced Erase*.
- Select the Device (CMD7), if not previously selected already.

Define the block length (CMD16) to 1 byte (8bit Device lock/unlock command). Send the Device lock/unlock command (CMD42) with the appropriate data block of one byte on the data line including 16 bit CRC. The data block shall indicate the mode ERASE (the ERASE bit shall be the only bit set).

If the ERASE bit is not the only bit in the data field then the LOCK_UNLOCK_FAILED error bit will be set in the status register and the erase request is rejected. If the command was accepted then ALL THE DEVICE CONTENT WILL BE ERASED including the PWD and PWD_LEN register content and the locked Device will get unlocked. In addition, if the Device is temporary write protected it will be unprotected (write enabled), the temporary-write-protect bit in the CSD and all temporary Write-Protect-Groups will be cleared.

If a force erase command is issued and power-on protected or a permanently-write-protected write protect groups exist on the device, the command will fail (Device stays locked) and the LOCK_UNLOCK_FAILED error bit will be set in the status register.

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

The Force Erase time-out is specified in 0. On v4.3 and later version devices, when host issues the Force erase, only the data stored in user data area (including enhanced attribute area) will be erased. The Force erase will not applied to Boot, RPMB and General partition area.

6.6.20 Application-specific commands

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

- **Application-specific command—APP_CMD (CMD55)**

  This command, when received by the Device, will cause the Device to interpret the following command as an application specific command, ACMD. The ACMD has the same structure as of regular *e*•MMC standard commands and it may have the same CMD number. The Device will recognize it as ACMD by the fact that it appears after APP_CMD. The only effect of the APP_CMD is that if the command index of the, immediately, following command has an ACMD overloading, the nonstandard version will be used. If, as an example, a Device has a definition for ACMD13 but not for ACMD7 then, if received immediately after APP_CMD command, Command 13 will be interpreted as the nonstandard ACMD13 but, command 7 as the standard CMD7.

6.6.20 Application-specific commands (cont'd)

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

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

If a non-valid command is sent (neither ACMD nor CMD) then it will be handled as a standard *e*•MMC illegal command error. From the *e*•MMC protocol point of view the ACMD numbers will be defined by the manufacturers without any restrictions.

- **General command—GEN_CMD (CMD56)**

  The bus transaction of the GEN_CMD is the same as the single block read or write commands (CMD24 or CMD17). The difference is that the argument denotes the direction of the data transfer (rather than the address) and the data block is not a memory payload data but has a vendor specific format and meaning. The Device shall be selected ('*tran_state'*) before sending CMD56. If the Device operates in the single data rate mode, the data block size is the BLOCK_LEN that was defined with CMD16. If the Device operates in the dual data rate mode, the data block size is 512 bytes. The response to CMD56 will be R1.

6.6.21 Sleep (CMD5)

A Device may be switched between a Sleep state and a Standby state by SLEEP/AWAKE (CMD5). In the Sleep state the power consumption of the memory device is minimized. In this state the memory device reacts only to the commands RESET (CMD0 with argument of either 0x00000000 or 0xF0F0F0F0 or H/W reset) and SLEEP/AWAKE (CMD5). All the other commands are ignored by the memory device. The timeout for state transitions between Standby state and Sleep state is defined in the EXT_CSD register S_A_TIMEOUT. The maximum current consumptions during the Sleep state are defined in the EXT_CSD registers S_C_VCC and S_C_VCCQ.

Sleep command: The bit 15 as set to 1 in SLEEP/AWAKE (CMD5) argument. Awake command: The bit 15 as set to 0 in SLEEP/AWAKE (CMD5) argument.

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

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

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

6.6.21 Sleep (CMD5) (cont'd)

A locked Device may be placed into Sleep state by first deselecting the Device through CMD7, a Class 0 command that can be executed in the locked state, and then issuing the Sleep command. This would allow the Device to save power consumption while it is locked. The locked Device can subsequently exit sleep and be placed into Standby state through the Awake command.

The reverse sequence, locking the device after it has already entered sleep, is not allowed.

6.6.22 Replay Protected Memory Block

A signed access to a Replay Protected Memory Block is provided. This function provides means for the system to store data to the specific memory area in an authenticated and replay protected manner. This is provided by first programming authentication key information to the *e*•MMC memory (shared secret).

As the system cannot be authenticated yet in this phase the authentication key programming have to take in a secure environment like in an OEM production. Further on the authentication key is utilized to sign the read and write accesses made to the replay protected memory area with a Message Authentication Code (MAC).

Usage of random number generation and count register are providing additional protection against replay of messages where messages could be recorded and played back later by an attacker.

6.6.22.1 The Data Frame for Replay Protected Memory Block Access

The data frame to access the replay protected memory area is including following fields of data.

Table 17 — Data Frame Files for RPMB

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		

Byte order of the RPMB data frame is MSB first, e.g., Write Counter MSB [11] is storing the upmost byte of the counter value.

6.6.22.1 The Data Frame for Replay Protected Memory Block Access (cont'd)

Name: Request/Response Type

- Length: 2B
- Direction: Request (to the memory), Response (from the memory)
- Description: defines the type of request and response to/from the memory. Table 18 is listing the defined requests and responses. The response type is corresponding to the previous Replay Protected Memory Block read/write request.

Table 18 — RPMB Request/Response Message Types

Request Message Types	
0x0001	Authentication key programming request
0x0002	Reading of the Write Counter value -request
0x0003	Authenticated data write request
0x0004	Authenticated data read request
0x0005*	Result read request
0x0006	Authenticated Device Configuration write request
0x0007	Authenticated Device Configuration read request
Response Message Types	
0x0100	Authentication key programming response
0x0200	Reading of the Write Counter value -response
0x0300	Authenticated data write response
0x0400	Authenticated data read response
0x0500	Reserved
0x0600	Authenticated Device Configuration write response
0x0700	Authenticated Device Configuration read response
NOTE There is no corresponding response type for the Result read request because the reading of the result with this request is always relative to previous programming access.	

Name: Authentication Key / Message Authentication Code (MAC)

- Length: 32Bytes (256bits)
- Direction: Request (key or MAC) / Response (MAC)
- Description: the authentication key or the message authentication code (MAC) depending on the request/response type. The MAC will be delivered in the last (or the only) block of data.

Name: Operation Result

- Length: 2B
- Direction: Response
- Description: includes information about the status of the write counter (valid, expired) and successfulness of the access made to the Replay Protected Memory Block. Table 19 represents the RPMB Operation Results data structure. Table 20 is listing the defined results and possible reasons for failures.

Table 19 — RPMB Operation Results data structure

Bit[15:8]	Bit[7]	Bit[6:0]
reserved	Write Counter Status	Operation Result

6.6.22.1 The Data Frame for Replay Protected Memory Block Access (cont'd)

Table 20 — RPMB Operation Results

Operation Results	
0x0000 (0x0080)*	Operation OK
0x0001 (0x0081)	General failure
0x0002 (0x0082)	Authentication failure (MAC comparison not matching, MAC calculation failure)
0x0003 (0x0083)	Counter failure (counters not matching in comparison, counter incrementing failure)
0x0004 (0x0084)	Address failure (address out of range, wrong address alignment)
0x0005 (0x0085)	Write failure (data/counter/result write failure)
0x0006 (0x0086)	Read failure (data/counter/result read failure)
0x0007**	Authentication Key not yet programmed
***** The values in parenthesis are valid in case the Write Counter has expired i.e. reached its max value ****** This value is the only valid Result value until the Authentication Key has been programmed (after which it can never occur again)	

Name: Write Counter

- Length: 4Bytes
- Direction: Request and Response.
- Description: Counter value for the total amount of the successful authenticated data write requests and Authenticated Device Configuration write request made by the host.

Name: Data Address

- Length: 2B
- Direction: Request and Response.
- Description: Address of the data to be programmed to or read from the Replay Protected Memory Block. Address is the serial number of the accessed half sector (256B) and the first address is 0x0000. Address argument in CMD 18 and CMD 25 will be ignored.

Name: Nonce

- Length: 16B
- Direction: Request and Response.
- Description: Random number generated by the host for the Requests and copied to the Response by the *e*•MMC Replay Protected Memory Block engine.

Name: Data

- Length: 256B
- Direction: Request and Response.
- Description: Data to be written or read by signed access.

Name: Block Count

- Length: 2B
- Direction: Request.
- Description: Number of blocks (half sectors, 256B) requested to be read/programmed. This value is equal to the count value in CMD23 argument.

6.6.22.2 Memory Map of the Replay Protected Memory Block

The Replay Protected Memory Block is including following registers and memory partition:

Name: Authentication Key

- Size: 32B
- Type: Write once
- Description: One time programmable authentication key register. This register cannot be overwritten, erased or read. The key is used by the *e*•MMC Replay Protected Memory Block engine to authenticate the accesses when MAC is calculated.

Name: Write Counter

- Size: 4B
- Type: Read only
- Description: Counter value for the total amount of successful authenticated data write requests and Authenticated Device Configuration write request made by the host. Initial value after *e*•MMC production is 0x0000 0000. Value will be incremented by one automatically by the *e*•MMC Replay Protected Memory Block engine along with successful programming accesses. The value cannot be reset. After the counter has reached its maximum value 0xFFFF FFFF it will not be incremented anymore (overflow prevention) and the bit [7] in Operation Result value will be permanently set.

Name: Data

- Size: 128kB min (RPMB_SIZE_MULT x 128kB)
- Type: Read/Write
- Description: Data that can only be read and written via successfully authenticated read/write access. This data may be overwritten by the host but can never be erased.

6.6.22.3 Message Authentication Code Calculation

The message authentication code (MAC) is calculated using HMAC SHA-256 as defined in [HMAC-SHA]. The HMAC SHA-256 calculation takes as input a key and a message. The resulting MAC is 256 bits (32Byte that are embedded in the data frame as part of the request or response.

The key used for the MAC calculation is always the 256 bit Authentication Key stored in the *e*•MMC. The message used as input to the MAC calculation is the concatenation of the fields in the data frames excluding stuff bytes, the MAC itself, start bit, CRC16, and end bit. That is, the MAC is calculated over bytes [283:0] of the data frame in that order.

If several data frames are sent as part of one request or response then the input message to MAC is the concatenation of bytes [283:0] of each data frame in the order that the data frames are sent. The MAC is added only to the last data frame.

EXAMPLE Assume that the write counter is 0x12345678. The host wants to write at address 0x0010 two sectors of where the first sector is 256 bytes of 0xAA and the second sector is 256 bytes of 0xBB. The host must then send to following two requests frames: Where the MAC in the second request frame is an HMAC. See Table 21 for an illustration.

Table 21 — MAC Example

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte(256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
Request frame 1											
	0x0000...	0x0000...	0xAA...	0x0000 ...	0x12345678	0x0010	0x0002	0x0000	0x0003		
Request frame 2											
	0x0000...	MAC	0xBB...	0x0000 ...	0x12345678	0x0010	0x0002	0x0000	0x0003		

SHA-256 calculated over the following message:

0xAA... (total 256 times) ... 0xAA
0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000

0x1234 0x5678 0x0010 0x0002 0x0000 0x0003
0xBB... (total 256 times) ... 0xBB

0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000

0x1234 0x5678 0x0010 0x0002 0x0000 0x0003

The key used for the HMAC SHA-256 calculation is the authentication key stored in the *e*•MMC.

Reference:

[HMAC-SHA] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, July 2006.

6.6.22.4 Accesses to the Replay Protected Memory Block

After putting a slave into transfer state, master sends CMD6 (SWITCH) to set the PARTITION_ACCESS bits in the EXT_CSD register, byte [179]. After that, master can use the Multiple Block read and Multiple Block Write commands (CMD23, CMD18 and CMD25) to access the Replay Protected Memory Block partition. The defined accesses are listed in following sections.

Any undefined set of parameters or sequence of commands, or High Priority Interrupt during Authenticated Write will result in a failed access.

- In Data Programming case the data shall not be programmed
- In Data Read case the data read shall be "0x00" (in multiple block case in all frames)"

A General Failure (0x01) will be indicated in the Results register (in multiple block case in all frames). If there are more than one failure related to an access then the first type of error shall be indicated in the Results register. The order of the error checking is defined under each access below. After finishing data access to the Replay Protected Memory Block partition, the PARTITION_ACCESS bits should be cleared.

6.6.22.4.1 Programming of the Authentication Key

The Authentication Key is programmed with the Write Multiple Block command, CMD25. Prior to the command CMD25 the block count is set to 1 by CMD23, with argument bit [31] set as 1 to indicate Reliable Write type of programming. If block count has not been set to 1 and/or argument bit [31] has not been set to 1 then the subsequent Write Multiple Block command must fail and General Failure shall be indicated.

The key information itself is delivered in data packet. The packet is size of 512B and it is including the request type information and the Authentication Key. The request type value 0x0001 indicates programming of the Authentication Key.

Table 22 — Authentication Key Data Packet

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
0b	0x0000...		0x00	0x00	0x00	0x00	0x00	0x00	0x0001		1b

The busy signaling in the DAT0 line after the CRC status by the *e*•MMC is indicating programming busy of the key. The status can also be polled with CMD13. The status response received in R1 is indicating the generic status condition, excluding the status of successful programming of the key.

The successfulness of the programming of the key should be checked by reading the result register of the Replay Protected Memory Block. The result read sequence is initiated by Write Multiple Block command, CMD25. Prior to CMD25, the block count is set to 1 by CMD23. If block count has not been set to 1 then the subsequent Write Multiple Block command must fail and General Failure shall be indicated.

The request type information is delivered in data packet. The packet is size of 512B and is including the request type information. The request type value 0x0005 indicates Result register read request initiation.

6.6.22.4.1 Programming of the Authentication Key (cont'd)

Table 23 — Result Register Read Request Packet

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
0b	0x0000...	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x0005		1b

The busy signaling in the DAT0 line after the CRC status by the *e*•MMC is indicating request busy. The result itself is read out with the Read Multiple Block command, CMD18. Prior to the read command, the block count is set to 1 by CMD23. If block count has not been set to 1 then the Read Multiple Block command must fail and General Failure shall be indicated.

The result information itself is delivered in the read data packet. The packet size is 512B and is including the response type information and result of the key programming operation. The response type value 0x0100 corresponds to the key programming request.

Table 24 — Response for Key Programming Result Request

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
0b	0x0000...	0x00	0x00	0x00	0x00	0x00	0x00		0x0100		1b

Access to Reply Protected Memory Block is not allowed and not possible before Authentication Key is programmed. The state of the device can be checked by trying to write/read data to/from the Replay Protected Memory Block and then reading the result register. If the Authentication Key is not yet programmed then message 0x07 (Authentication Key not yet programmed) is returned in result field.

If programming of Authentication Key fails then returned result is 0x05 (Write failure). If some other error occurs during Authentication Key programming then returned result is 0x01 (General failure).

6.6.22.4.2 Reading of the Counter Value

The counter read sequence is initiated by Write Multiple Block command, CMD25. Prior to CMD25, the block count is set to 1 by CMD23. If block count has not been set to 1 then the subsequent Write Multiple Block command must fail and General Failure shall be indicated.

The request type information is delivered in data packet. The packet is size of 512B and it is including the request type information and the nonce. The request type value 0x0002 indicates counter value read request initiation.

Table 25 — Counter Read Request Packet

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
0b	0x0000...	0x00	0x00		0x00	0x00	0x00	0x00	0x0002		1b

The busy signaling in the DAT0 line after the CRC status by the *e*•MMC is indicating request busy. The counter value itself is read out with the Read Multiple Block command, CMD18. Prior to the command CMD18, the block count is set to 1 by CMD23. If block count has not been set to 1 then the Read Multiple Block command must fail and General Failure shall be indicated.

The counter value itself is delivered in the read data packet. The packet size is 512B and it is including the response type information, a copy of the nonce received in the request, the write counter value, the MAC and the Result.

Table 26 — Counter Value Response

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
0b	0x0000...		0x00			0x00	0x00		0x0200		1b

If reading of the counter value fails then returned result is 0x06 (Read failure). If some other error occurs then Result is 0x01 (General failure). If counter has expired also bit 7 is set to 1 in returned results (Result values 0x80,0x86 and 0x81, respectively).

6.6.22.4.3 Authenticated Data Write

Data to the Replay Protected Memory Block is programmed with the Write Multiple Block command, CMD25. In prior to the command CMD25 the block count is set by CMD23, with argument bit [31] set as 1 to indicate Reliable Write type of programming access. The block count is the number of the half sectors (256B) to be programmed.

The supported data size of RPMB write access is determined by EN_RPMB_REL_WR (EXT_CSD[166] bit 4). The whole data in one RPMB write access is atomic and shall contain either old data or new data. The start address of each RPMB write access shall be aligned with the transfer size of the corresponding RPMB write access.

- EN_RPMB_REL_WR = 0 : Two different sizes are supported in RPMB partition: 256B (single 512B frame) and 512B (two 512B frames)
- EN_RPMB_REL_WR = 1 : Three different sizes are supported in RPMB partition: 256B (single 512B frame), 512B (two 512B frames), and 8KB (thirty two 512B frames)

If block count has not been set and/or argument bit[31] has not been set to 1and/or the size is different from the supported data size defined by EN_RPMB_REL_WR, then the subsequent Write Multiple Block command must fail and General Failure shall be indicated.

Data itself is delivered in data packet. The packet is size of 512B and it is including the request type information, the block count, the counter value, the start address of the data, the data itself and the MAC. In multiple block write case the MAC is included only to the last packet n, the n-1 packets will include value 0x00. In every packet the address is the start address of the full access (not address of the individual half a sector) and the block count is the total count of the half sectors (not the sequence number of the half a sector).The request type value 0x0003 indicates programming of the data.

Table 27 — Program Data Packet

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
0b	0x0000...			0x00				0x00	0x0003		1b

The busy signaling in the DAT0 line after the CRC status by the *e*•MMC is indicating buffer busy between the sent blocks (in multiple block write case) and programming busy of the key after the last block (or in single block case). The status can also be polled with CMD13. The status response received in R1 is indicating the generic access status condition (e.g., state transitions), excluding the status of successfulness of programming of the data.

When the *e*•MMC receives this message it first checks whether the write counter has expired. If the write counter is expired then *e*•MMC sets the result to 0x85 (write failure, write counter expired). No data is written to the *e*•MMC

Next the address is checked. If there is an error in the address (out of range) then the result is set to 0x04 (address failure). In case of multiple block write, if the Data Address is not aligned to its own data size then the result is set to 0x04 (address failure).No data are written to the *e*•MMC. If the write counter was not expired then the MAC is calculated over bytes [283:0] (request type, result = 0x00, block count, write counter, address, nonce = 0x00 and data), and compares this with the MAC in the request.

6.6.22.4.3 Authenticated Data Write (cont'd)

If the two MAC's are different then *e*•MMC sets the result to 0x02 (authentication failure). No data are written to the *e*•MMC.

If the MAC in the request and the calculated MAC are equal then the *e*•MMC compares the write counter in the request with the write counter stored in the *e*•MMC. If the two counters are different then *e*•MMC sets the result to 0x03 (counter failure). No data are written to the *e*•MMC. If the MAC and write counter comparisons are successful then the write request is considered to be authenticated. The data from the request are written to the address indicated in the request and the write counter is incremented by 1.

If write fails then returned result is 0x05 (write failure). If some other error occurs during the write procedure then returned result is 0x01 (General failure). The successfulness of the programming of the data should be checked by the host by reading the result register of the Replay Protected Memory Block. The result read sequence is initiated by Write Multiple Block command, CMD 25. Prior to CMD25, the block count is set to 1 by CMD23. If block count has not been set to 1 then the subsequent Write Multiple Block command must fail and General Failure shall be indicated.

The request type information is delivered in data packet. The packet is size of 512B and is including the request type information. The request type value 0x0005 indicates result register read request initiation.

Table 28 — Result Register Read Request Packet

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
0b	0x0000...	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x0005		1b

The busy signaling in the DAT0 line after the CRC status by the *e*•MMC is indicating request busy. The result itself is read out with the Read Multiple Block command, CMD18. Prior to the read command, the block count is set to 1 by CMD23. If block count has not been set to 1 then the Read Multiple Block command must fail and General Failure shall be indicated. The result information itself is delivered in the read data packet. The packet size is 512B and includes the response type information, the incremented counter value, the data address, the MAC and result of the data programming operation.

Table 29 — Response for Data Programming Result Request

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
0b	0x0000...		0x00	0x00			0x00		0x0300		1b

6.6.22.4.4 Authenticated Data Read

Data read sequence is initiated by Write Multiple Block command, CMD25. Prior to CMD25, the block count is set to 1 by CMD23. If block count has not been set to 1 then the subsequent Write Multiple Block command must fail and General Failure shall be indicated.

The request type information is delivered in the data packet. The packet is size of 512B and it is including the request type information, the nonce and the data address. The request type value 0x0004 indicates data read request initiation.

Table 30 — Data Read Request Initiation Packet

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
0b	0x0000...	0x00	0x00		0x00		0x00	0x00	0x0004		1b

The busy signaling in the DAT0 line after the CRC status by the *e*•MMC is indicating request busy.

When the *e*•MMC receives this request it first checks the address. If there is an error in the address (out of range) then result is set to 0x04 (address failure). The data read is not valid. The data itself is read out with the Read Multiple Block command, CMD18. Prior to the read command, the block count is set by CMD23.

NOTE The block count of the RPMB read operation is not indicated in the original RPMB Data Read Request packet. Block count set in CMD23 for reading out the RPMB data also indicates the block count for the RPMB read operation. This is intentional for RPMB implementation in *e*•MMC standard that may be different from implementations in other protocol standards.

The block count is the number of the half sectors (256B) to be read. If block count has not been set then the Read Multiple Block command must fail and General Failure shall be indicated. MAC is calculated over bytes [283:0] (response type, nonce, address, block count, write counter = 0x00, data and result). If the MAC calculation fails then returned result is 0x02 (Authentication failure).

The data information itself is delivered in the read data packet. The packet size is 512B and it is including the response type information, the block count, a copy of the nonce received in the request, the data address, the data itself, the MAC and the result. In multiple block read case the MAC is included only to the last packet n, the n-1 packets will include value 0x00. In every packet the address is the start address of the full access (not address of the individual half a sector) and the block count is the total count of the half sectors (not the sequence number of the half a sector).

Table 31 — Read Data Packet

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
0b	0x0000...				0x00				0x0400		1b

If data fetch from addressed location inside *e*•MMC fails then returned result is 0x06 (Read failure). If some other error occurs during the read procedure then returned result is 0x01 (General failure).

6.6.22.4.5 Authenticated Device Configuration Write

Device Configuration Area is programmed through the RPMB procedure by using the Write Multiple Block command, CMD25. In prior to the command CMD25 the block count is set by CMD23, with argument bit [31] set as 1 to indicate Reliable Write type of programming access. The block count is the number of the half sectors (256B) to be programmed and the block count shall be 0x1. Therefore only one size of 256B (single 512B frame) is used.

If block count and argument bit[31] has not been set to 1 then the subsequent Write Multiple Block command shall fail and General Failure shall be indicated in RPMB Operation Results.

Data itself is delivered in data packet. The packet is size of 512B and it is including the request type information, the block count, the counter value, the start address of the data, the data itself and the MAC.

The address is the index of Device Configuration register to be updated, i.e., 0x0001 or 0x0002 is valid index. The request type value 0x0006 indicates programming of the data.

Table 32 — Authenticated Device Configuration Write packet

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
0b	0x0000...		0x00…xx (1 Byte config data)	0x00..0	Current Write Counter	0x0001 or 0x0002	0x00001	0x0000	0x0006		1b

The busy signaling in the Dat0 line after the CRC status by the *e*•MMC is indicating programming busy. The status can also be polled with CMD13. The status response received in R1 is indicating the generic access status condition (e.g., state transitions), excluding the status of successfulness of programming of the data.

When the *e*•MMC receives this message it first checks whether the write counter has expired. If the write counter is expired then *e*•MMC sets the result to 0x85 (write failure, write counter expired). No data is written to the *e*•MMC

If host access reserved area (address[0] and address[3~255]), the device should not return error. No data are written to the *e*•MMC. If the write counter was not expired then the MAC is calculated over bytes [283:0] (request type, result = 0x00, block count, write counter, address, nonce = 0x00 and data), and compares this with the MAC in the request. If the two MAC's are different then *e*•MMC sets the result to 0x02 (authentication failure). No data are written to the *e*•MMC.

If the MAC in the request and the calculated MAC are equal then the *e*•MMC compares the write counter in the request with the write counter stored in the *e*•MMC. If the two counters are different then *e*•MMC sets the result to 0x03 (counter failure). No data are written to the *e*•MMC.

If the MAC and write counter comparisons are successful then the write request is considered to be authenticated. The data from the request are written to the Device Configuration Area indicated by address in the request and the write counter is incremented by 1.

If write fails then returned result is 0x05 (write failure). If some other error occurs during the write procedure then returned result is 0x01 (General failure). The successfulness of the programming of the data should be checked by the host by reading the result register of the Replay Protected Memory Block. The result read sequence is initiated by Write Multiple Block command, CMD 25. Prior to CMD25, the block count is set to 1 by CMD23. If block count has not been set to 1 then the subsequent Write Multiple Block command shall fail and General Failure shall be indicated in RPMB Operation Results.

6.6.22.4.5 Authenticated Device Configuration Write (cont'd)

The request type information is delivered in data packet. The packet is size of 512B and is including the request type information. The request type value 0x0005 indicates result register read request initiation. (Refer the Table 28, Result Register Read Request Packet)

The response type value of Authenticated Device Configuration write response is 0x0600. The busy signaling in the Dat0 line after the CRC status by the *e*•MMC is indicating request busy. The result itself is read out with the Read Multiple Block command, CMD18. Prior to the read command, the block count is set to 1 by CMD23. If block count has not been set to 1 then the Read Multiple Block command shall fail and General Failure shall be indicated in RPMB Operation Results. The result information itself is delivered in the read data packet. The packet size is 512B and includes the response type information, the incremented counter value, index for Device Configuration Area -, the MAC and result of the data programming operation.

Table 33 — Response for Authenticated Device Configuration Write Request

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
0b	0x0000...		0x00	0x00…0	Incremented counter value	Index for Device Conf. Reg	0x0000...	Result code	0x0600		1b

6.6.22.4.6 Authenticated Device Configuration Read

Authenticated Device Configuration read sequence is initiated by Write Multiple Block command, CMD25. Prior to CMD25, the block count is set to 1 by CMD23. If block count has not been set to 1 then the subsequent Write Multiple Block command shall fail and General Failure shall be indicated in RPMB Operation Results.

The request type information is delivered in the data packet. The packet is size of 512B and it is including the request type information and the nonce. The request type value 0x0007 indicates Authenticated Device Configuration read request initiation.

Table 34 — Authenticated Device Configuration Read Initiation packet

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
0b	0x0000...	0x00…0	0x00..0	Host Generated Nonce	0x00…0	0x0000	0x0000	0x0000	0x0007		1b

The busy signaling in the Dat0 line after the CRC status by the *e*•MMC is indicating request busy.
The data itself is read out with the Read Multiple Block command, CMD18. Prior to the read command, the block count is set by CMD23.

The block count shall be set as 0x1. . If block count has not been set as 0x1 then the Read Multiple Block command shall fail and General Failure shall be indicated in RPMB Operation Results.

The data information itself is delivered in the read data packet. The packet size is 512B and it is including the response type information, the block count, a copy of the nonce received in the request, the data address, the data itself, the MAC and the result.

Table 35 — Response for Authenticated Device Configuration Read

Start	Stuff Bytes	Key/ (MAC)	Data	Nonce	Write Counter	Address	Block Count	Result	Req/ Resp	CRC16	End
1bit	196Byte	32Byte (256b)	256Byte	16Byte	4Byte	2Byte	2Byte	2Byte	2Byte	2Byte	1bit
	[511:316]	[315:284]	[283:28]	[27:12]	[11:8]	[7:6]	[5:4]	[3:2]	[1:0]		
0b	0x0000...			Host Generated Nonce	0x00..0	0x0000	0x0001		0x0700		1b

If data fetch from addressed Device Configuration Area inside *e*•MMC fails then returned result is 0x06 (Read failure). If some other error occurs during the read procedure then returned result is 0x01 (General failure).

6.6.23 Dual Data Rate mode selection

After the host verifies that the Device complies with version 4.4, or higher, of this standard, and supports dual data rate mode, it may enable the dual data rate data transfer mode in the Device. After power-on, hardware reset, or software reset, the operating mode of the Device is single data rate, except when the Device begins by performing boot operation, in which case the selection between single or dual data rate mode is determined by EXT_CSD register byte [177] (BOOT_BUS_CONDITIONS) settings specified in Table 147. For the host to change to dual data rate mode, HS_TIMING[3:0] shall be set to 0x1. For the host to change to HS400 mode, HS_TIMING[3:0] shall be set to 0x3 (see 6.6.2) and the Device shall supports this mode as defined in EXT_CSD register [196] (DEVICE_TYPE) specified in Table 137.

The host uses the SWITCH command to write 0x05 (4-bit data width) or 0x06 (8-bit data width) to the BUS_WIDTH byte, in the Modes segment of the EXT_CSD register byte [183]. In HS400 mode, the host is expect to write 0x06(8-bit data width) to the BUS_WIDTH byte, in the Modes segment of the EXT_CSD register byte [183]. The valid values for this register are defined in Table 145.If the host tries to write an invalid value, the BUS_WIDTH byte is not changed, the dual data rate mode is not enabled, and the SWITCH_ERROR bit is set.

Conversely, a Device in the dual data mode may be switched back to single data rate mode by writing new values in the BUS_WIDTH byte.

6.6.24 Dual Data Rate mode operation

After the Device has been enabled for dual data rate operating mode, the block length parameter of CMD17, CMD18, CMD24, CMD25 and CMD56 automatically default to 512 bytes and cannot be changed by CMD16 (SET_BLOCKLEN) command that becomes illegal in this mode.

Therefore, all single or multiple block data transfer read or write will operate on a fixed block size of 512 bytes while the Device remains in dual data rate mode.

CMD 30 and CMD 31 are used in dual data rate mode with their native data size. Additionally to CMD16, CMD42 (LOCK_UNLOCK), CMD14 (BUSTEST_R), CMD19 (BUSTEST_W), CMD11 (READ_DAT_UNTIL_STOP) and CMD 20 (WRITE_DAT_UNTIL_STOP) are considered illegal in the dual data rate mode. If any of these commands are required, the Device shall be switched to operate in single data rate mode before using these. CMD26 and CMD27 are supported in dual data rate mode if EXT_CSD register [130] (PROGRAM_CID_CSD_DDR_SUPPORT) is set.

6.6.25 Background Operations

Devices have various maintenance operations need to perform internally. In order to reduce latencies during time critical operations like read and write, it is better to execute maintenance operations in other times - when the host is not being serviced. Operations are then separated into two types:

- Foreground operations – operations that the host needs serviced such as read or write commands;
- Background operations – operations that the device executes while not servicing the host; Depending on how they can be initiated, there are two types of background operations: manually initiated background operations and autonomously initiated background operations.

Manually initiated method: In order for the device to know when the host does not need it and it can execute background operations, host shall write any value to BKOPS_START (EXT_CSD byte [164]) to manually start background operations. Device will stay busy till no more background processing is needed.

Since foreground operations are of higher priority than background operations, host may interrupt on-going background operations using the High Priority Interrupt mechanism (see 6.6.26). In order for the device to know if host is going to periodically start background operations, host shall set bit 0 (MANUAL_EN) of BKOPS_EN (EXT_CSD byte [163]) to indicate that it is going to write to BKOPS_START periodically. The device may then delay some of its maintenance operations to when host writes to BKOPS_START.

The device reports its background operation status in bits [1:0] of BKOPS_STATUS (EXT_CSD byte [246]), that can be in one of four possible levels: 0x0: No operations required 0x1: Operations outstanding – non critical 0x2: Operations outstanding – performance being impacted 0x3: Operations outstanding – critical

Host shall check the status periodically and start background operations as needed, so that the device has enough time for its maintenance operations, to help reduce the latencies during foreground operations. If the status is at level 3 ("critical"), some operations may extend beyond their original timeouts due to maintenance operations that cannot be delayed anymore. The host should give the device enough time for background operations to avoid getting to this level in the first place.

To allow hosts to quickly detect the higher levels, the URGENT_BKOPS bit in the EXCEPTION_EVENTS_STATUS is set whenever the levels is either 2 or 3. That automatically sets the EXCEPTION_BIT in Device Status. This allows hosts to detect urgent levels on every R1 type response. Hosts shall still read the full status from the BKOPS_STATUS byte periodically and start background operations as needed.

The background operations feature is mandatory for this specification. Bit 0 of BKOPS_SUPPORT (EXT_CSD byte [502]) shall be set.

Autonomously initiated method; a Host that wants to enable the device to perform background operations during device idle time, should signal the device by setting AUTO_EN in BKOPS_EN field [EXT_CSD byte 163] to 1b. When this bit is set, the device may start or stop background operations whenever it sees fit, without any notification to the host.

When AUTO_EN bit is set, the host should keep the device power active. The host may set or clear this bit at any time based on its power constraints or other considerations.

6.6.26 High Priority Interrupt (HPI)

In some scenarios, different types of data on the device may have different priorities for the host. For example, writing operation may be time consuming and therefore there might be a need to suppress the writing to allow demand paging requests in order to launch a process when requested by the user.

The high priority interrupt (HPI) mechanism enables servicing high priority requests, by allowing the device to interrupt a lower priority operation before it is actually completed, within OUT_OF_INTERRUPT_TIME timeout. Host may need to repeat the interrupted operation or part of it to complete the original request.

The HPI command may have one of two implementations in the device:

- CMD12 – based on STOP_TRANSMISSION command when the HPI bit in its argument is set.
- CMD13 – based on SEND_STATUS command when the HPI bit in its argument is set.

Host shall check the read-only HPI_IMPLEMENTATION bit in HPI_FEATURES (EXT_CSD byte [503]) and use the appropriate command index accordingly.

If CMD12 is used with HPI bit set, it differs from the non-HPI command in the allowed state transitions. See Table 60, Device state transition, for the specific transitions for both cases.

HPI shall only be executed during prg-state. Then, it indicates the device that a higher priority command is pending and therefore it should interrupt the current operation and return to tran-state as soon as possible, with a different timeout value.

If HPI is received in states other than prg-state, the device behavior is defined in Table 60, Device state transition. If the state transition is allowed, response is sent but the HPI bit is ignored. If the state transition is not allowed, the command is regarded as an illegal command.

HPI command is accepted as a legal command in prg-state. However, only some commands may be interrupted by HPI. If HPI is received during commands that are not interruptible, a response is sent but the HPI command has no effect and the original command is completed normally, possibly exceeding the OUT_OF_INTERRUPT_TIME timeout. Table 36 shows the commands are interruptible and those that are not.

Table 36 — Interruptible commands

CMD Index	Name	Is interruptible?
CMD24	WRITE_BLOCK	Yes
CMD25	WRITE_MULTIPLE_BLOCK	Yes
CMD38	ERASE	Yes
CMD6	SWITCH, byte BKOPS_START, any value	Yes
CMD6	SWITCH, byte SANITIZE_START, any value	Yes
CMD6	SWITCH, byte POWER_OFF_NOTIFICATION, value POWER_OFF_LONG or SLEEP_NOTIFICATION	Yes
CMD6	SWITCH, byte POWER_OFF_NOTIFICATION, other values	No
CMD6	CACHE_CTRL when used for turning the cache OFF	Yes
CMD6	FLUSH_CACHE	Yes
CMD6	SWITCH, other bytes, any value	No
All others		No

6.6.26 High Priority Interrupt (HPI) (cont'd)

Following a WRITE_MULTIPLE_BLOCK command (CMD25), the device updates the CORRECTLY_PRG_SECTORS_NUM field (EXT_CSD bytes [245:242]) with the number of 512B sectors successfully written to the device. Host may use this information when repeating an interrupted write command – it does not need to re-write again all data sectors, it may skip the correctly programmed sectors and continue writing only the rest of the data that wasn't yet programmed.

In case HPI is interrupting a CMD25 that is part of a packed write command (see 6.6.29), the CORRECTLY_PRG_SECTORS_NUM field will reflect the accumulated packed sectors that were transferred (plus header) – host may calculate from this number the index of the interruptible individual command and the offset of interruption for it.

If HPI is received during a reliable-write command, the first CORRECTLY_PRG_SECTORS_NUM sectors shall contain the new data and all the rest of the sectors shall contain the old data.

The HPI mechanism shall be enabled before it may be used, by setting the HPI_EN bit in the HPI_MGMT field (EXT_CSD byte [161]). The HPI feature is mandatory for this specification. All devices shall have the HPI_SUPPORT bit in HPI_FEATURES field (EXT_CSD byte [503]) set.

6.6.27 Context Management

To better differentiate between large sequential operations and small random operations, and to improve multitasking support, contexts can be associated with groups of read or write commands. Associating a group of commands with a single context allows the device to optimize handling of the data.

A context can be seen as an active session, configured for a specific read/write pattern (e.g., sequential in some granularity). Multiple read or write commands are associated with this context to create some logical association between them, to allow device to optimize performance. For example, a large sequential write pattern may have better performance by allowing the device to improve internal locality and reduce some overheads (e.g., if some large unit of data is allowed to be affected by a power failure as a whole while it is being written to, all of the commands that fill this unit can work faster because they can reduce the overhead usually required to protect each write individually in case of a power failure). Furthermore, handling of multiple concurrent contexts allows the device to better recognize each of the write patterns without getting confused when they are all mixed together.

Device may support one or more concurrent contexts, defined by a context-ID. Context ID #0 always exists for backward compatibility and for context-less data. Each context ID (besides #0) has a configuration field in EXT_CSD to control its behavior.

To use a context, an available context ID shall be picked. Then, it shall be initialized by writing the configuration register. Then, data can be read/written associated to the context by specifying the context ID and the number of blocks (> 0) in SET_BLOCK_COUNT command (CMD23) before sending the read/write command. When the context is no longer used, the configuration register should be updated to close the context ID. A context shall be closed prior to re-configuring it for another configuration/use.

NOTE A Read/Write command using context shall be close-ended command. In case the host uses an open-ended command the device may consider the read/write command a context-less command.

6.6.27.1 Context configuration

Before any context can be used it shall be configured, except for context #0 that is always pre-configured and cannot be changed.

Configuration is done by writing to the configuration register of the specific context ID required. Then, all read commands or write commands that are associated with this context ID shall be sent with the ID. When the context is no longer needed, it should be closed by writing a zero byte to the configuration register.

The configuration registers are an array of 15 byte registers, one for each context (except the fixed, pre-defined #0). To configure a context, the following fields shall be written to the configuration register of the specific context needed:

- Activation and direction mode (read-only, write-only or read/write)
  The direction indicates if all following accesses to this context would be either read-only, write-only or both read/write.
  Writing a non-zero direction to this field is the 'activation code' for the context. A zero in this field means a closed context that can no longer be addressed by its ID until re-configured.
- Large Unit context flag
  This indicates if the context is following Large Unit rules or not
  - If Large Unit context flag is set, then the Large Unit multiplier may be used to specify a larger unit to be used instead of the Large Unit itself
- Reliability mode Controls how data written to a context should respond to interruptions

6.6.27.2 Context direction

A non-zero context can be configured as a read-only context, a write-only context or a read/write context. Context #0 is always treated as a read/write context.

Any read command from any context (including #0) to an address that is part of an active write-only context is not allowed, and may either fail the command or return dummy data.

Any write command from any context (including #0) to an address that is part of an active read-only context is not allowed, and may either fail the command, ignore the data written or cause unexpected data to return in the read commands.

A context that is configured as read/write context may be read while written, as long as the writing follows the context rules.

NOTE Read/write context may have reduced performance compared to read-only or write-only contexts.

6.6.27.3 Large-Unit

The Large Unit is the smallest unit that can be used for large sequential read/write operations, in order to reduce internal overheads and improve performance.

Accessing a Large Unit (both read & write) shall:

- Use a context-ID configured to operate in Large Units
- Always access a full Large Unit, in order and from beginning to end
  - Multiple read/write commands may be used to read or write the Large Unit and they may be interleaved with other accesses and commands, as long as the specific Large Unit is being accessed with its own separate context ID
  - A Large Unit that is being written to shall not be modified outside the scope of the context (e.g., no other writes from other contexts to the address range of the Large Unit shall be used, no erases/trims to that range, etc.)
  - Different Large Units belonging to the same context may be located in non-consecutive addresses on the media, as long as alignment is kept (a Large Unit shall be accessed in order from beginning to end, but only within the range of the specific Large Unit – the next Large Unit can be non-consecutive and even in a lower address)
- When writing a Large Unit context, data shall always be aligned and in multiples of SUPER_PAGE_SIZE

When writing a Large Unit context, last Large Unit before closing the context may be partially written, as long as it is written from the beginning, in order and up to a specific point where it is closed. Host should be aware that the rest of the Large Unit may be padded (by the device) to the end of the Large Unit with some data (may be random).

6.6.27.4 Context Writing Interruption

In case a write command to a context-ID other than #0 is interrupted, the device behaves as if all the writes to the context from its configuration were written in one large write command.

In case of a power failure, RST_n assertion or CMD0 before closing an active context – even if not in the middle of a write command to the specific context (even if not in the middle of any command) – is considered as if the event occurred during writing the entire context.

During a context write operation to a partition with WR_REL_SET set to '1', reliability mode of context would be according to the context Reliability Mode. It means that data written inside a context may be lost by power loss even if WR_REL_SET is set to '1'; whereas during a context write operation to a partition with WR_REL_SET set to '0', reliability mode of the context would be ignored (regardless of the context Reliability Mode).

In any case, once the context is closed the data would be protected according to WR_REL_SET.

The reliable write bit in CMD23 is ignored when context-ID is non-zero. Instead, the context behavior is determined as part of its configuration and is applied to all writes to this context until it is closed. Interruption during any of the writes to a non-zero context may cause some of the data not to be fully programmed on the device. Still no partial sector shall exist – any sector written as part of the non-zero context shall contain either the new data written or its old data before the context was configured. The scope of data that may be affected by the interruption depends on the mode configured:

- For non-Large Unit contexts:
  - MODE0 – Normal mode – Any data written to the context from the time it was configured may be affected
  - MODE1 – Non-Large Unit, reliable mode – Only data written by a specifically interrupted write command (CMD25) may be affected. Any previously completed write to the context shall not be changed because of any interruption.
- For Large Unit contexts:

  NOTE1 The unit N refers to the current Large Unit that is being written to when interruption occurs, unit N-1 refers to the last Large Unit of the context that was written completely before the current one and unit N-2 and earlier are Large Units that were completed before the N-1 unit.

  NOTE 2 Data stored in a large unit that is affected may be partially or entirely invalid.
  - MODE0 – Normal mode – Any unit may be affected: Any data written to the context from the time it was configured may be affected
  - MODE1 – Large Unit, unit-by-unit mode – Unit N may be affected, units N-1 and earlier are not: Any data written to a Large Unit context may affect the entire specific Large Unit accessed. Any previously completed Large Units in the context shall not be changed because of any interruption.
  - MODE2 – Large Unit, one-unit-tail mode – Unit N and N-1 may be affected, units N-2 and earlier are not: Any data written to a Large Unit context may affect the entire specific Large Unit accessed and the entire completed Large Unit that was accessed before the current one. Any other completed Large Units in the context shall not be changed because of any interruption.

In case HPI is used during a write to a non-zero context, the write may still be interrupted like any context-less write. In case HPI is interrupting a write to a Large Unit context (including one that is packed inside a packed-write command), the device shall always stop writing on a SUPER_PAGE_SIZE boundary (and report CORRECTLY_PRG_SECTORS_NUM accordingly), so host may later continue the stopped write from an address that is aligned to SUPER_PAGE_SIZE as required for Large Unit contexts. See also 6.6.26.

6.6.27.5 Large-Unit Multipliers

In order to allow increased performance by parallelism, the device may allow reading or writing in units larger than Large Unit.

The device reports in EXT_CSD the maximum multiplier that is supported.

In order to use a multiplier beyond one, the context should be configured with the multiplier, and all alignment and unit sizes are multiplied by the multiplier.

For example, if the Large Unit size is 1MB, reading data from a Large Unit context configured with a multiplier of 2 shall be done in full units of 2MB and aligned to 2MB. Inside each such unit, same rules apply as if it was a normal Large Unit context with a Large Unit size of 2MB.

6.6.28 Data Tag Mechanism

The mechanism permits the device to receive from the host information about specific data types (for instance file system metadata, time-stamps, configuration parameters, etc.). The information is conveyed before a write multiple blocks operation at well-defined addresses. By receiving this information the device can improve the access rate during the following read and update operations and offer a more reliable and robust storage.

The host should manage the tags accountability (tags allocation, tags – device logical addresses mapping, etc.)

If the SYSTEM_DATA_TAG_SUPPORT (EXT_CSD[499]) field is set to 1, the device supports the functionality on all the ‘Default’ type partitions (user data area in the area that’s not defined as enhanced user data area and general purpose partitions configured with the ‘Default’ attribute)

The host can mark a data segment (Tag Unit) in the addressable space whose length, in bytes, is expressed by the Extended CSD parameter TAG_UNIT_SIZE (always multiple of the sector size). The marking operation shall be Tag Unit aligned and have the granularity of a Tag Unit. If the address of the Write Multiple Blocks command is not Tag Unit aligned but falls in the middle of a Tag Unit, the entire Tag Unit will be marked in order to contain System Data.

The request to mark a Tag Unit or a sequence of Tag Units is based on the use of the SET_BLOCK_COUNT command (CMD23).

Any command that updates data (for instance a Trim, Erase, regular/reliable write, etc.) sent on the logical addresses containing a System Data previously written by the Tagging mechanism determines that the new data will not be opportunely managed to behave as a System Data. Those commands shall be sent in order to cover a single Tag Unit or a multiple of Tag Units.

The resources allocated by the device to accommodate data marked as System Data Tag can be limited. This condition is indicated by the device raising the EXCEPTION_EVENT bit in the Device Status register provided as response to the first R1, R1b command following the internal resource exhaustion event; after receiving this indication the host retrieves the specific information about the exhaustion of resources allocated for the tagging mechanism by checking the EXCEPTION_EVENTS_STATUS field in Extended CSD byte.

NOTE SYSPOOL_EVENT_EN bit in EXCEPTION_EVENTS_CTRL field in Extended CSD shall be set to get the exception event indication in this case.

The data shall still be written to the device regardless of the resources exhaustion; however, it may not have the improved characteristics.

The host should manage the tag operation counting in order to avoid resources exhaustion. If it happens the host should execute operations having the final effect of freeing some of the resources (for instance by issuing a Trim command on some of the addresses containing System Data).

6.6.29 Packed Commands

Read and write commands can be packed in groups of commands (either all read or all write) that transfer the data for all commands in the group in one transfer on the bus, to reduce overheads.

Packed write command can group several write multiple block commands by using SET_BLOCK_COUNT command (CMD23) with the PACKED flag set. Then, WRITE_MULTIPLE_BLOCK (CMD25) shall follow, with the first block containing the packed-command header as described below. Then, all data sectors of the individual packed commands are sent appended together after the header, in the order of appearance in the header. The block count specified in CMD23 in this case shall be the sum of all block counts of the individual writes plus one for the header.

Packed read command can group several read multiple block commands by using SET_BLOCK_COUNT command (CMD23) with the PACKED flag set and a block count of one. Then, WRITE_MULTIPLE_BLOCK (CMD25) shall follow with just the header. Then, CMD23 may be sent again with the packed flag set and a block count equal to the sum of all block counts of the individual reads (CMD23 here is optional – open-ended reading is allowed). Then a READ_MULTIPLE_BLOCK (CMD18) shall follow to read the packed data. Host should not send any other command (except for CMD13 SEND_STATUS) until the packed read command sequence is completed (i.e., inserting command(s) between sending the header and reading the data is not permitted), otherwise device behavior is undefined.

In both cases of read and write, the argument for CMD25 (both for packed write and in case of the header of a packed read) and for CMD18 shall be the same address that is specified by the first individual read or write command in the packed group. If the argument does not match the address of the first individual command, device behavior is undefined.

The maximum number of read or write commands that can be packed in a single packed command is reported by the device in MAX_PACKED_READS and MAX_PACKED_WRITES fields in EXT_CSD respectively.

MAX_PACKED_READS and MAX_PACKED_WRITES may be larger than the number of commands that one block is able to include if the device is 4kB native device. In emulation mode for 4kB native sector device, host can pack commands into only one block and send the block as header. After disabling the emulation mode, host can pack commands into 8 blocks and send the blocks as header.

6.6.29.1 Packed Command Header

The packed command header is sent on the first block (if DATA_SECTOR_SIZE[61] = 0x00) or the first 8 blocks (if DATA_SECTOR_SIZE[61] = 0x01) of the data in case of packed write command and as a separated write command sent before reading the packed data in case of a packed read command.

The structure contains:

- Version of structure – a byte to indicate version for future compatibility; shall be set to 0x01,
- R/W flag – 0x01 for packed read, 0x02 for packed write,
- Number of entries in the table,
- Then, for each entry:
  - Argument of CMD23 of the individual command (4 bytes), formatted as in CMD23, including both the ‘count’ field and the various flags fields (high bits), with the ‘packed’ bit always ‘0’ (except for the ‘packed’ bit, all other CMD23 argument bits are still allowed in the header). Each individual read and write command shall not be open-ended. If any bit[15:0] of argument of CMD23in the header is set to 0, device behavior is undefined."
  - Argument of CMD18 or CMD25 (4 bytes) – the address read/written by the individual command.

The structure contains (in little-endian format, padding shall be all zeros):

Table 37 — Packed Command Structure

Entry index	Offset (Bytes)	Name	Length (Bytes)
-	+0	VERSION	1
	+1	R/W	1
	+2	NUM_ENTRIES (=N)	1
	+3	padding to 8B	5
1	+8	CMD23_ARG_1	4
	+12	CMDxx_ARG_1	4
2	+16	CMD23_ARG_2	4
	+20	CMDxx_ARG_2	4
3	+24	CMD23_ARG_3	4
	+28	CMDxx_ARG_3	4
…			
N	+8N	CMD23_ARG_N	4
	+8N+4	CMDxx_ARG_N	4
-	+8N+8	Padding	till block ends

6.6.29.2 Packed Commands Error Handling

If one of the individual commands causes a failure, the following individual commands within the same packed command are not executed. For packed read, the device may transfer some data (may be random) for the non-executed commands – to complete the number of blocks specified (as host is waiting for all blocks). Alternatively it can stop transferring the data at the point of failure, causing a timeout. For packed write, the device shall ignore the transfer of the non-executed commands.

In any case of failure, the generic error flag in the PACKED_COMMAND_STATUS field in EXT_CSD is set (e.g., bad header, more commands in the pack than device supports, etc.). In case of any error during one of the individual commands within the packed command, the 'error index' flag shall be set as well as the generic error, and the PACKED_FAILURE_INDEX field in EXT_CSD shall report the index in the header block of the failed command.

Additionally, an exception event bit may report a non-zero status in PACKED_COMMAND_STATUS if host enables it by writing a value of '1' to PACKED_EVENT_EN to the EXCEPTION_EVENTS_CTRL. Once enabled, the exception bit in the Device Status (reported on every R1 response) shall be set if PACKED_COMMAND_STATUS is non-zero.

When packed commands are used, reporting of errors in command responses may be deferred to the command after the actual packed read/write transfer (the CMD25 or CMD18 that transfers the packed data). For example, the ADDRESS_OUT_OF_RANGE error may be deferred in case one of the packed commands accessed an address that is out of range.

6.6.30 Exception Events

The exception events allow device to report some exceptions quickly to the host by setting the EXCEPTION_EVENT bit in Device Status. The exception event bit remains set until its cause is cleared.

To control the events that may set this bit, a 16 bit EXCEPTION_EVENTS_CTRL field in EXT_CSD is defined – each bit enables a specific event to set the EXCEPTION_EVENT bit. The EXCEPTION_EVENT bit is therefore a logical-or between all exception events that have their relevant enable-bit set in EXCEPTION_EVENTS_CTRL.

To make it easier to detect the specific exceptions, a 16 bit EXCEPTION_EVENTS_STATUS field in EXT_CSD holds a bit per exception to indicate the one that is currently set. The status bit is set only for exceptions that are enabled in EXCEPTION_EVENTS_CTRL.

6.6.31 Cache

Cache is a temporary storage space in an *e*•MMC device. The cache should in typical case reduce the access time (compared to an access to the main nonvolatile storage) for both write and read. The cache is not directly accessible by the host. This temporary storage space may be utilized also for some implementation specific operations like as an execution memory for the memory controller and/or as storage for an address mapping table etc. however that definition is out of scope of this standard.

The cache is expected to be volatile by nature. Implementation of such volatile cache has some significant effects on some of the requirements of this standard as follows:

- Caching of data shall apply only for the single block read/write (CMD17/24), pre-defined multiple block read/write (CMD23+CMD18/25) and open ended multiple block read/write (CMD18/25+CMD12) commands and excludes any other access e.g., to the register space (e.g., CMD6).
- Data0 busy and status response bit [8] after CMD24, CMD23+CMD25 or CMD25+CMD12 does not anymore necessarily indicate the programming status to the nonvolatile memory but may indicate programming status to the volatile cache (exceptions defined later).
- A Flush operation refers to the requirement, from the host to the device, to write the cached data to the nonvolatile memory. Prior to a flush, the device may autonomously write data to the nonvolatile memory, but after the flush operation all data in the volatile area must be written to nonvolatile memory.
- Data in the cache may (and most probably will) be lost due to a sudden power down. If there was a flush operation ongoing when the power was lost then also any such data may be lost.
- Accesses to the RPMB and Boot partitions while the cache is ON shall still be directed to the nonvolatile storage with same requirements as defined elsewhere in this standard.
- When the cache is turned ON it applies to the *e*•MMC device as whole (When flushing the cache the data for all partitions shall be flushed, this operation is independent of the active partition).
- There is no requirement for flush due to switching between the partitions. (Note: This also implies that the cache data shall not be lost when switching between partitions). Cached data may be lost in SLEEP state, so host should flush the cache before placing the device into SLEEP state.
- The device may invalidate the data in the cache in case of RST_n or CMD0 received.

Support of the cache function and size of the cache are indicated in the CACHE_SIZE byte (EXT_CSD byte [252:249]).

The cache shall be OFF by default after power up, RST_n assertion or CMD0. All accesses shall be directed to the nonvolatile storage like defined elsewhere in this specification. The cache function can be turned ON and OFF by writing to the CACHE_CTRL byte (EXT_CSD byte [33]). Turning the cache ON shall enable behavior model defined in this section. Turning the cache OFF shall trigger flushing of the data to the nonvolatile storage.

The cache may be flushed to the nonvolatile storage by writing to FLUSH_CACHE byte (EXT_CSD byte [32]). The R1b response result shall reflect the status of programming of cached data to the nonvolatile storage. Any error resulted can be read from the status register by CMD13 after the completion of the programming as defined for normal write. If a flush error occurs during the execution of the FLUSH_CACHE or while turning off the cache using the CACHE_CTRL operation, the device shall set the generic error bit, STATUS BIT[19]. If an error occurs as a result of flush operation the device has no responsibility to isolate the error to a specific data area. The error could affect any data written to the cache since the previous flush operation.

6.6.31 Cache (cont'd)

There is no maximum timeout for flushing of the cache as flushing a large amount of cached data may take very unpredictably long time. This applies both for the FLUSH_CACHE and CACHE_CTRL (turning the cache OFF) operations. Host may use the HPI function to interrupt the flush operation. In this case the cache shall not be considered as fully flushed and host should re-initiate the flush.

There are two ways to force data to be programmed directly to the nonvolatile storage while the cache is turned ON:

1) A Reliable Write access shall force the data to be written to the nonvolatile storage and Data0 busy indication reflects the status of the programming as defined elsewhere in this standard,
2) Set the argument bit [24] in the CMD23 in prior to the actual write command. In case the cache is turned OFF then this bit shall be ignored by the memory device. If this bit is set together with the Reliable Write bit in CMD23 then this bit shall be ignored also by the device.

A logical block may or may not be removed from the cache after the flush operation. The data may also be stored to the cache in case of Reliable Write access and access with CMD23 bit [24] set. These are left for the implementation.

If the cache gets totally full and/or the cache is not able to receive the data of the next access (per block count indicated in CMD23 or per initiated single/open ended multiple block write in general) then it shall still be responsibility of the *e*•MMC device to store data of the next access within timeouts specified elsewhere in this specification. The actual algorithm to handle the new data and possible flush of some older cached data is left for the implementation.

ERASE, TRIM and DISCARD commands shall have effect on data in the cache accordingly so that any following read will reflect the Erased, Trimmed or Discarded state of data. There may be copies of an old data in the cache and these are expected to be cleared during power down. The SANITIZE operation shall clean up the unmapped data in the cache.

All functions related to a certain address should still be considered as sequential and ordered, just like with a device without the cache. For example if there is some old data in an address and then some new data is written (potentially cached) to this address. Next there is the write protection set to the very same address and write protect is not a cached operation as such. Still the write protection shall in all cases apply to the new data, not to the old data that may still have been in the nonvolatile memory in this particular time. There is no requirement that the cache in general should be first-in-first-out -type, this is left for the implementation.

In between consecutive flush operations the device is free to write data into the nonvolatile memory out of order. If the host wants to preserve the order it must flush the cache at the point where order is important; the device can only ensure data was written in order after the flush operation has completed.

It shall never be possible to read stale data from the *e*•MMC device due to the cached operation.

The cache feature is optional for an *e*•MMC device.

6.6.32 Features cross matrix

This section summarizes rules and limitations on simultaneous activations of features that are controlled through CMD23 argument, Cache and Partitions.

The following rules are derived from Extended Partition Attributes, Reliable Write, Context Management, Data Tag, Packed Commands and Cache features:

- If CMD23 Reliable Write [31] bit is set then Forced Prg [24] is ignored,
- CMD23 bits for Data Tag [29] and ContextID [28:25] cannot be set in the same command argument,
- In case of Packed Command, in the encapsulating CMD23, packed [30] bit and number of blocks [15:0] are valid; all other parameter bits are '0' and are included in the header (see the "Packed Command" and "Non-Packed Command" description in Table 52),
- Enhanced and Extended partition settings cannot co-exist in same partition,
- Data Tagging works only for default partitions,
- ContextID overrides Reliable Write,
- Cache mode does not apply for Boot and RPMB partitions.

Table 38 — Features Cross Reference Table

Cache On							
Ok	**Packed Commands**						
Ok	Ok, in header (1)	**Context ID**					
Ok	Ok, in header (1)	Not valid	**Data Tag**				
If Reliable Write bit is set, cache is bypassed	Ok, in header (1)	Reliable Write bit is ignored if a Context is open	Ok	**Reliable Write**			
No caching on Boot and RPMB Partitions	No Packed Commands on Boot and RPMB Partitions	Ok	Data tag is valid on default memory type only	Ok	**Partitions (all)**		
If Force programming bit is set, cache is bypassed (2)	Ok, in header (1)	Ok	Ok	If Reliable Write bit is set, Forced Programming bit is ignored	Ok	**Forced programming**	
-	No	No	No	-	Switch to FFU mode could be done from any partition except boot and RPMB	-	**FFU Mode**

NOTE1 The feature bit (ContextID, Data Tag, Reliable Write) is valid in the Packed Command header (first data block of CMD25), and not in the Packed Command argument (argument of the encapsulating CMD23)
NOTE 2 Forced Programming is ignored when the Cache is turned OFF
NOTE 3 If contradicting features are activated in CMD23 argument, device behavior is undefined.
For example, ContextID + Reliable Write + Forced Programming: this combination is undefined since ContextID causes Reliable Write to be ignored, and Reliable Write causes Forced Programming to be ignored

6.6.33 Dynamic Capacity Management

Extensive memory usage and aging of Flash could result in bad blocks. Dynamic Capacity Management provides a mechanism for the memory device to reduce its reported capacity and extend the device life time.

The mechanism to manipulate dynamic capacity is based on: memory array partitioning and the granularity of WP groups (see 6.14). Reducing the capacity is done by releasing of WP-Groups anywhere within the address space of the user area. A released WP-Group will behave as a permanently write protected group and it shall not be read: Writing to an address within a released WP-Group returns a WP error; Reading from an address within a released WP-Group is forbidden and may return an error; Checking write protection (using CMD30) and write protect type (using CMD31) shall report protected groups and permanent write protection accordingly.

Write protected groups cannot be released – they shall be unprotected first before they may be released.

Write protect groups from general purpose partitions cannot be released – only groups from the user area may be released.

The device keeps an indication DYNCAP_NEEDED in EXT_CSD byte [58] to notify the host how many WP-Groups host should release. Host should monitor this byte and release as many WP-Groups as requested for the device to continue to be functional.

If device reports a non-zero DYNCAP_NEEDED and host doesn't release WP-Groups accordingly, the device may degrade in performance and eventually become non-functional.

Additionally, an exception event bit may report the status of DYNCAP_NEEDED if host enables it by writing a value of '1' to DYNCAP_EVENT_EN to the EXCEPTION_EVENTS_CTRL. Once enabled, the exception bit in the Device Status (reported on every R1 response) shall be set if DYNCAP_NEEDED is non-zero.

In order to release WP-Groups and query status of groups, host shall first set CLASS_6_CTRL in EXT_CSD byte [59] to 0x01 (see Table 174) and then use class 6 commands (CMD28-31) to release and/or query the status of WP-Groups. Host shall write 0x00 back to CLASS_6_CTRL before using class 6 command for write protection manipulation.

When CLASS_6_CTRL is set to 0x01:

- CMD28 is used to release a WP-Group
- CMD29 is ignored (no-operation)
- CMD30 returns the status of released or not released WP-Groups
- CMD31 returns a fixed pattern of 64-bit zeros

The dynamic capacity commands and statuses are only supported for high capacity devices and are based on the high capacity write protect group size.

The device user area size shall never change while a device is powered up even if WP-Groups from the end of the user area were released. The device may update its user area following the release of WP-Groups from the end of its address space only after passing through CMD1 after a power cycle, assertion of RST_n signal or CMD0. In any case, the addressing mode of the device doesn't change (device stays in high capacity sector based addressing).

6.6.34 Large sector size

The native sector size is the smallest unit of information that the device manages internally. For a high capacity device, two options may exist for its native sector size:

1) Small 512 B sectors (supported by devices up to and including 256 GB total capacity), or
2) Large 4 KB sectors (supported by all devices).

The device reports its native sector size in NATIVE_SECTOR_SIZE field of EXT_CSD [63].

For backward compatibility, large sectors devices shall work in emulation mode when shipped. The actual sector size host may access and address in any mode is reported by DATA_SECTOR_SIZE. In emulation mode, DATA_SECTOR_SIZE is still 512 B, but the device is emulating the 512 B operations internally that may affect performance. The emulation mode may be disabled by the host. Once emulation mode is disabled, the sector size is set permanently to the native sector size.

All devices with capacities higher than 256 GB shall have native sector size of 4 KB.

For non-high capacity devices the fields NATIVE_SECTOR_SIZE, DATA_SECTOR_SIZE, USE_NATIVE_SECTOR and INI_TIMEOUT_EMU are read only and shall be zero and should be ignored.

Internal sizes reported by the device shall always be a full multiple of the native sector size even in emulation mode, e.g., device capacity, write protect group size, erase group size, super page size, etc.

The units of the internal sizes in the emulation mode and the one after disabling emulation mode shall be same except Tag Unit size. Table 40 shows a list of granularities and units for all internal sizes.

A large sector device shall not support partial access.

6.6.34 Large sector size (cont'd)

Table 39 — *e*•MMC internal sizes and related Units / Granularities

Parameter		Definition	Unit			Granularity		
			512 B Native Sector	4 KB Native Sector		512 B Native Sector	4 KB Native Sector	
				Emulation	Disable Emulation		Emulation	Disable Emulation
Device Capacity	≤ 2GB	BLOCKNR · BLOCK_LEN	512B	512B		512B	4KB	
	> 2GB	SEC_COUNT ·512B	512B	512B		512B	4KB	
Erase Group Size		(ERASE_GRP_SIZE+ 1) · (ERASE_GRP_MULT + 1)	512B	512B		512B	4KB	
Write Protect Group Size		WP_GRP_SIZE	Erase group	Erase group		Erase Group	Erase Group	
High Capacity Erase Size		HC_ERASE_GRP_SIZE · 512KB	512KB	512KB		512 KB	512KB	
High Capacity Write Protect size		HC_ERASE_GRP_SIZE · HC_WP_GRP_SIZE · 512KB	512KB	512KB		512KB	512KB	
Super page size		$2^{(SUPER\_PAGE\_SIZE-1)}$· 512B	512B	512B		512B	4KB	
Boot partition size		BOOT_SIZE_MULT· 128KB	128KB	128KB		128KB	128KB	
RPMB partition size		RPMB_SIZE_MULT· 128KB	128KB	128KB		128KB	128KB	
Tag Unit Size		$2^{TAG\_UNIT\_SIZE}$·Sector Size	512B	512B	4KB	512B	512B	4KB
Tag Resource Size		((N· TAG_UNIT_SIZE)· $2^{TAG\_RES\_SIZE}$)/2^{10}	Tag Unit Size	Tag Unit Size		Tag Unit Size	Tag Unit Size	
Large Unit size		1MB · (LARGE_UNIT_SIZE_M1 + 1)	1MB	1MB		1MB	1MB	
Cache size		CACHE_SIZE·1Kb	1Kb	1Kb		1Kb	32Kb (=4KB)	
Reliable Write Sector size		REL_WR_SEC_C	512B	512B		512B	4KB	
Max Enhanced area size		MAX_ENH_SIZE_MULT·HC_WP_GRP_SIZE·HC_ERASE_GRP_SIZE·512KB	512 KB	512KB		512KB	512KB	
General purpose partition size		(GP_SIZE_MULT_X_2·2^{16}+ GP_SIZE_MULT_X_1·2^{8}+ GP_SIZE_MULT_X_0·2^{0}) ·HC_WP_GRP_SIZE ·HC_ERASE_GRP_SIZE ·512KB	512 KB	512KB		512KB	512KB	
Enhanced user data area size		(ENH_SIZE_MULT_2·2^{16}+ ENH_SIZE_MULT_1·2^{8}+ ENH_SIZE_MULT_0·2^{0}) ·HC_WP_GRP_SIZE ·HC_ERASE_GRP_SIZE ·512KB	512KB	512KB		512KB	512KB	

6.6.34.1 Disabling emulation mode

To disable the emulation mode for large 4KB sector devices, host may write 0x01 to the USE_NATIVE_SECTOR field in EXT_CSD [62] and then go through a power cycle, passing through CMD1.

Setting USE_NATIVE_SECTOR does not immediately change the DATA_SECTOR_SIZE. Only after a power cycle following setting of USE_NATIVE_SECTOR to 0x01 shall the device change DATA_SECTOR_SIZE. The following initialization timeout for CMD1 is then defined by the INI_TIMEOUT_EMU.

Any valid commands issued after USE_NATIVE_SECTOR is set to 0x01 but before a power cycle takes place will be normally executed.

Disabling emulation mode is only supported on devices that do not support partitions or before partitions are configured. Once partitions are configured, disabling of emulation mode is no longer allowed.

Setting Bit 0 in PARTITION_SETTING_COMPLETED and writing 0x01 in USE_NATIVE_SECTOR without a power cycle between the two operations is not allowed.

Some internal sizes reported by the device may change after successfully disabling of the emulation mode.

After a successful disabling of the emulation mode, the content of the User Data Area is undefined.

6.6.34.2 Native 4KB sector behavior

The following restrictions apply for a 4 KB sector device that is in native mode (not emulation mode):

- Data transfers on the bus are still using 512 B CRC-protected blocks, but data shall only be transferred in multiple of 8 such blocks (always multiples of 4 KB),
- Sector addressing is still used, but sector addresses shall always be aligned to 8 (4 KB),
- Sector counts shall be multiples of 8 (4 KB), e.g., in SET_BLOCK_COUNT (CMD23), and CORRECTLY_PRG_SECTORS_NUM field in EXT_CSD,
- Single block Write and Read Command (CMD17 and CMD24) are not supported. If the host attempts to issue CMD17 or CMD24 the device behavior is undefined,
- Arguments for read multiple block command (CMD18) and multiple block write command (CMD25) shall always be aligned to 8 (4 KB),
- If a power loss occurs during a reliable write, each 4 KB sector being modified by the write is atomic. After a power failure sectors may either contain old data or new data . All of the sectors being modified by the write operation that was interrupted may be in one of the following states: all sectors contain new data, all sectors contain old data or some sectors contain new data and some sectors contain old data,
- Start and end addresses in erase/trim/discard commands shall always be aligned to 8 (4 KB), and
- RPMB partition, if exists, is an exception to the above: Since read/write commands to the RPMB partition are data frames packets and not actual read/write operations, access to RPMB partition shall still be done in 512 B blocks.

Not following any of the above restrictions may result in undefined behavior.

Table 40 summarizes data sector size, address mode and reliable write granularity depending on the device density range, device sector size and emulation mode enablement state:

Table 40 — Admitted Data Sector Size, Address Mode and Reliable Write granularity

Device density range	Native Sector size	Emulation mode	Data Sector size	Address mode	Reliable Write granularity
`≤ 2GB	**N/A**	N/A	512B	Byte	512B
2GB < Density ≤ 256GB	512B	N/A	512B	Sector (512B)	512B
2GB < Density ≤ 256GB	4KB	On (when device is shipped)	512B	Sector (512B)	512B
2GB < Density ≤ 256GB	4KB	Off	4KB	Sector (512B) (1)	4KB
Density > 256GB	4KB (mandatory)	On (when device is shipped)	512B	Sector (512B)	512B
Density > 256GB	4KB (mandatory)	Off	4KB	Sector (512B) (1)	4KB

NOTE 1 Each data transfer has a minimum length of 8 Sectors (4KB) and the addresses shall be aligned to 8 (4KB)

6.6.35 Real Time Clock Information

Providing real time clock information to the device may be useful for internal maintenance operations.

Host may provide either absolute time (based on UTC) if available, or relative time. This feature provides a mechanism for the host to update both real time clock and relative time updates (see CMD49).

The host should send the time information on the following events:

- After power up, as early as possible
- After waking up from sleep state, as early as possible
- Periodically (hourly, daily, …)

To set the real time clock, CMD49 command (SET_TIME) shall be used. This command that behaves like CMD24 (WRITE_BLOCK) in its state transitions and timeouts and shall always be configured to transfer a 512B information block using CMD16 (SET_BLOCKLEN). The information block is formatted as little-endian block with the following structure:

Table 41 — Real Time Clock Information Block Format

Bytes	Field Name
1	Version of Structure: Shall be set to 0x01
1	RTC_INFO_TYPE
8	SECONDS_PASSED
10-511	Reserved (shall be set to all '0')

The meaning of the SECONDS_PASSED field is determined by the RTC_INFO_TYPE field:

Table 42 — RTC_INFO_TYPE Field Description

Value	Name	Description
0x01	Absolute time	SECONDS_PASSED is the number of seconds that passed from January 1st, year 0 AD till the current time in UTC (universal time coordinated)
0x02	Set-base for relative time	SECONDS_PASSED is ignored, and the time of sending the SET_TIME command with this parameter is set as a reference for future relative time
0x03	Relative time	SECONDS_PASSED is the number of seconds that passed since the most recent call to SET_TIME with the parameter RTC_INFO_TYPE=2 (set-base)
0x00, 0x04-0xFF	-	Reserved, not allowed

6.6.35.1 Periodic Wake-up

Host may update the PERIODIC_WAKEUP register (EXT_CSD[131]) to indicate to the device how often host shall wake it up. Host shall then repeat the following sequence at least as often as configured by PERIODIC_WAKEUP register: Power up the device and execute at least one background operation (by writing to BKOPS_START) that is run until completion (without interruptions), before powering down the device. Host may execute other operations before powering down the device as long as the device is not powered down for a period longer than that configured in PERIODIC_WAKEUP.

Host may only increase the frequency in the PERIODIC_WAKEUP register.

6.6.36 Power Off Notification

The host should notify the device before it powers the device off. This allows the device to better prepare itself for being powered off. Power the device off means to turn off all its power supplies. In particular, the host should issue a power off notification (POWER_OFF_LONG, POWER_OFF_SHORT) if it intends to turn off both V_{CC} and V_{CCQ} power supplies or it may use to a power off notification (SLEEP_NOTIFICATION) if it intends to turn-off V_{CC} after moving the device to Sleep state.

To indicate to the device that power off notification is supported by the host, a supporting host shall first set the POWER_OFF_NOTIFICATION byte in EXT_CSD [34] to POWERED_ON (0x01). To execute a power off, before powering the device down the host changes the value to either POWER_OFF_SHORT (0x02) or POWER_OFF_LONG (0x03). Host waits for the busy line to be de-asserted. Once the setting has changed to either 0x02 or 0x03, host may safely power off the device.

The host may issue SLEEP_AWAKE (CMD5) to enter or to exit from Sleep state if POWER_OFF_NOTIFICATION byte is set to POWERED_ON. Before moving to Standby state and then to Sleep state, the host sets POWER_OFF_NOTIFICATION to SLEEP_NOTIFICATION and waits for the DAT0 line de-assertion. While in Sleep (slp) state V_{CC} (Memory supply) may be turned off as defined in 6.6.21. Removing power supplies other than V_{CC} while the device is in the Sleep (slp) state may result in undefined device behavior. Before removing all power supplies, the host should transition the device out of Sleep (slp) state back to Transfer state using CMD5 and CMD7 and then execute a power off notification setting POWER_OFF_NOTIFICATION byte to either POWER_OFF_SHORT or POWER_OFF_LONG.

If host continues to send commands (e.g., CMD8) to the device after switching to the power off setting (POWER_OFF_LONG, POWER_OFF_SHORT or SLEEP_NOTIFICATION) or performs HPI during its busy condition, the device shall restore the POWER_OFF_NOTIFICATION byte to POWERED_ON.

If host tries to change POWER_OFF_NOTIFICATION to 0x00 after writing another value there, a SWITCH_ERROR is generated.

The difference between the two power-off modes is how urgent the host wants to turn power off. The device should respond to POWER_OFF_SHORT quickly under the generic CMD6 timeout. If more time is acceptable, POWER_OFF_LONG may be used and the device shall respond to it within the POWER_OFF_LONG_TIME timeout.

While POWER_OFF_NOTIFICATION is set to POWERED_ON, the device expects the host to :

- Keep the device power supplies alive (both V_{CC} and V_{CCQ}) and in their active mode,
- Not power off the device intentionally before changing POWER_OFF_NOTIFICATION to either POWER_OFF_LONG or POWER_OFF_SHORT, and
- Not power off V_{CC} intentionally before changing POWER_OFF_NOTIFICATION to SLEEP_NOTIFICATION and before moving the device to Sleep state.

Before moving to Sleep state hosts may set the POWER_OFF_NOTIFICATION byte to SLEEP_NOTIFICATION (0x04) if aware that the device is capable of autonomously initiating background operations for possible performance improvements. Host should wait for the busy line to be de-asserted. Busy line may be asserted up the period defined in SLEEP_NOTIFICATION_TIME byte in EXT_CSD [216]. Once the setting has changed to 0x04 host may set the device into Sleep mode (CMD7+CMD5). After getting out from Sleep the POWER_OFF_NOTIFICATION byte will restore its value to POWERED_ON. HPI may interrupt the SLEEP_NOTIFICATION operation. In that case POWER_OFF_NOTIFICATION byte will restore to POWERED_ON.

6.6.37 Cache Enhancement Barrier

Barrier function provides a way to perform a delayed in-order flushing of a cached data. The main motivation for using barrier commands is to avoid the long delay that is introduced by flush commands. There are cases where the host is not interested in flushing the data right away, however it would like to keep an order between different cached data batches. The barrier command enables the host achieving the in-order goal but without paying the flush delay, since the real flushing can be delayed by the device to some later idle time. The formal definition of the barrier rule is as follows:

Denote a sequence of requests Ri, i=0,..,N. Assuming a barrier is set between requests Rx and Rx+1 (0<x<N) then all the requests R0..Rx must be flushed to the nonvolatile memory before any of the requests Rx+1..RN.

Between two barriers the device is free to write data into the nonvolatile memory in any order. If the host wants to preserve a certain order it shall flush the cache or set another barrier at a point where order is important.

The barrier is set by writing to the BARRIER bit of the FLUSH_CACHE byte (EXT_CSD byte [32]). Any error resulted can be read from the status register by CMD13 after the completion of the programming as defined for a normal write request. The error could affect any data written to the cache since the previous flush operation.

The device shall support any number of barrier commands between two flush commands. In case of multiple barrier commands between two flush commands a subset of the cached data may be committed to the nonvolatile memory according to the barrier rule. Internally, a device may have an upper limit on the barrier amount it can absorb without flushing the cache. That is, if the host exceeds this barrier amount, the device may issue, internally, a normal flush.

The device shall expose its barrier support capability via the BARRIER_SUPPORT byte (EXT_CSD byte [486]). If a device does not support barrier function this register shall be zero. If a device supports barrier function this register shall be one.

Assuming the device supports barrier function, if the BARRIER bit of the FLUSH_CACHE byte is set, a barrier operation shall be executed.

If the cache gets totally full and/or the cache is not able to receive the data of the next access (per block count indicated in CMD23 or per initiated single / open ended multiple block write in general) then it shall still be the responsibility of the *e*•MMC device to store the data of the next access within the timeouts that are specified elsewhere in this specification. The actual algorithm to handle the new data and possible flush of some older cached data is left for the implementation.

NOTE When issuing a force-programming write request (CMD23 with bit 24 on) or a reliable write request (CMD23 with bit 31 on), the host should be aware that the data will be written to the nonvolatile memory, potentially, before any cached data, even if a barrier command was issued. Therefore, if the writing order to the nonvolatile memory is important, it is the responsibility of the host to issue a flush command before the force-programming or the reliable-write request.

In order to use the barrier function, the host shall set bit 0 of BARRIER_EN (EXT_CSD byte [31]).

The barrier feature is optional for an *e*•MMC device.

6.6.38 Cache Flushing Policy

The host may require the device to flush data from the cache in an in-order manner. From time to time, to guarantee in-order flushing, the host may command the device to flush the device cache or may use a barrier command.

However, if the *e*•MMC device flushing policy is to flush data from the cache in an in-order manner, cache barrier commands or flush commands operations (In case goal is to guarantee the flushing order) are redundant and impose a needless overhead to the device and host.

FIFO bit in CACHE_FLUSH_POLICY field (EXT_CSD byte [240]) is used by the device to indicate to the host that the device cache flushing policy is First-In-First-Out; this means that the device guarantees that the order of the flushing of data would be the in same order that data was written to the cache. When the FIFO bit is set it is recommended for the host not to send cache barrier commands or flush operations which goal is to guarantee the flushing order as they are redundant and impose a burden to the system.

However, if the FIFO bit is set to 1b and the device supports the cache barrier mechanism, the host may still send barrier commands without getting an error. Sending these commands will not change the device behavior as device flushes cache in-order anyway.

The CACHE_FLUSH_POLICY field is read-only field and never change its value either by the host or device.

6.6.39 Command Queuing

6.6.39.1 Overview

To facilitate command queuing in *e*•MMC, the device manages an internal task queue that the host can queue data transfer tasks.

Initially the task queue is empty. Every task is issued by the host and initially queued as pending. The device controller works to prepare pending tasks for execution. When a task is ready for execution its state changes to "ready for execution". The exact meaning of "ready for execution" is left for device implementation.

The host tracks the state of all queued tasks and may order the execution of any task, that is marked as "ready for execution", by sending a command indicating its task ID. When the execute command is received (CMD46/CMD47) the device executes the data transfer transaction.

For example, in order to queue a write transaction the host sends a CMD44 indicating the task's parameters. The device responds and the host sends a CMD45, indicating the start block address.

The device regards the two commands as a single task in the queue and sends a response indicating success if no error is detected. This exchange may be executed on the CMD line while a data transfer, or busy state, is ongoing on the DAT lines. The host tracks the state of the queue using CMD13.

At a later time, when data transfer is not in progress, the host issues a CMD47, ordering the device to execute a task from the queue, providing the Task ID in its argument. The device responds with an R1 response and the data transfer starts.

NOTE If hosts need to access RPMB partition, the host should disable the Command Queue mechanism and access RPMB partition not through the command queue.

General Purpose partitions may be accessed when command queuing is enabled. The queue must be empty when CMD6 is sent (to switch partitions or to disable command queuing). Sending CMD6 while the queue is not empty shall be regarded as illegal command (see 6.6.39.9, Supported Commands).

Prior to enabling command queuing, the block size shall be set to 512 B. Device may respond with an error to CMD46/CMD47 if block size is not 512 B.

The device does not guarantee the order that queued tasks are processed. In cases where ordering is important (e.g., commands with overlapping LBAs), the host is responsible to ensure it.

6.6.39.2 QUEUED_TASK_PARAMS - CMD44

CMD44 is the first step in queuing a data transfer task. The command encodes parameters that are necessary for queuing the task and executing the operation. The arguments are: Block Count, Task ID, Priority, Data Direction, and additional parameters (i.e., Tag Request, Context ID, Reliable Write, Forced Programming).

The encoded Task ID must not be already in use by another task in the queue. A Task ID is considered available for reuse if the task identified by it is completed. A data read task is considered completed when the last data block has been fully transferred over the *e•*MMC bus. A data write task is considered completed when the busy signal following the last data block is released.

There are 2 priority levels: high priority tasks and simple priority tasks. The device is required to give priority to queued high-priority tasks by making the preparations for their execution before it prepares simple tasks. As such, a high priority task should normally transfer to "ready for execution" state before simple tasks that were queued before it. However, it is acceptable that, at the time of receiving the high-priority task request, the device may have started preparing simple tasks for execution, but has not marked them as "ready for execution" in the QSR. The device may mark such tasks as "ready for execution" before the new high-priority task.

The handling of error cases related to CMD44 is listed in Table 44.

CMD44 should always be followed by a CMD45 to complete the queuing operation. If the next command issued by the host is not CMD45, or CMD45 is issued but an error condition is detected, device behavior is undefined and host is expected to resend CMD44 and CMD45.

CMD44 may be issued while data transfer is on-going on the DAT lines, when the lines are idle (e.g., during NAC time) or when the device indicates BUSY state, except when the device is BUSY due to execution of CMD48.

6.6.39.3 QUEUED_TASK_ADDRESS - CMD45

CMD45 shall be issued directly after (and only after) CMD44. The argument of a CMD45 is the start block address for the related transaction (similar to CMD18, for example). When receiving CMD45, the device queues the task with the following parameters (extracted from the two commands CMD44 and CMD45): Task ID, Block Count, Block Address, Priority, Data Direction, Tag Request, Context ID, Reliable Write, and Forced Programming.

The handling of error cases related to CMD45 is listed in Table 44.

CMD45 may be issued while data transfer is on-going on the DAT lines or when the device indicates BUSY state or when the lines are idle (e.g., during NAC time). CMD45 should not be issued when device executes CMD48.

6.6.39.4 EXECUTE_READ_TASK - CMD46

In order to execute a data read task that is already queued the host issues CMD46. The argument of CMD46 is the Task ID of the requested task. The designated task must be a data read task (Data Direction = 1), that is marked as “ready for execution” in the Queue Status Register.

If the requested task (as indicated by the Task ID) is indeed a read task and is ready for execution, the device shall respond with R1 response and start the data transfer.

When the last data block has been fully transferred over the e•MMC bus, the device should clear "ready for execution" for the task.

The handling of error cases related to CMD46 is listed in Table 44.

6.6.39.5 EXECUTE_WRITE_TASK - CMD47

In order to execute a data write task that is already queued the host issues CMD47. The argument of CMD47 is the Task ID of the requested task. The designated task must be a data write task (Data Direction = 0), that is marked as “ready for execution” in the Queue Status Register.

If the requested task (as indicated by the Task ID) is indeed a write task and is ready for execution, the device shall respond with R1 response and the host will start the data transfer.

When the busy signal following the last data block is released, the device should clear "ready for execution" for the task.

The handling of error cases related to CMD47 is listed in Table 44.

6.6.39.6 CMDQ_TASK_MGMT - CMD48

CMD48 is used for various Command Queuing task management requests. The argument includes a TM op-code, which encodes the requested operation and a TaskID when relevant to the op-code. The list of valid TM op-codes and the respective descriptions of the operations is given in Table 43.

CMD48 may only be issued when no data transfer is taking place.

Table 43 — Task Management op-codes

TM op-code	Description	Task ID Required?
0h	Reserved	N/A
1h	**Discard entire queue:** Device shall discard all tasks in the queue[1] and QSR shall be cleared.	no
2h	**Discard Task:** Device shall discard designated task[2] and QSR designated task shall be cleared.	yes
3h-Fh	Reserved	
NOTE 1 If the queue is empty, device shall execute command without an error. NOTE 2 If Task ID does not exist, device shall execute command without an error.		

6.6.39.7 SEND_STATUS - CMD13

An option is added to CMD13 for reading the Queue Status Register (QSR) by the host. If bit [15] in CMD13's argument is set, then the device shall send an R1 Response with the QSR instead of the Device Status.

When the device, in its response to CMD13, indicates that one or more tasks are 'ready for execution', the host should select one of these tasks for execution, and not send additional CMD13s in expectation that additional tasks would become 'ready for execution'.

6.6.39.8 Error handling

A number of error cases are introduced by the command queuing mechanism. The new error cases are listed in Table 44, along with a description of how each case is handled by the *e*•MMC device.

Table 44 — Error handling for Command Queue

	Command	Error(s) Description	Response	Error Handling
1	CMD44/CMD45/ CMD46/CMD47/ CMD48	Received when Command Queuing is not enabled	No Response	Illegal Command
2	CMD44	Task ID already in use; Task ID exceeds CMDQ_DEPTH; Block Count equals zero	No Response	Illegal Command
3	CMD44	Error in other parameters (e.g., ContextID)	OK (no error indication)	Type 'X' errors
4	CMD45	Sent not immediately after CMD44	No Response	Illegal Command
5	CMD45	Type 'R' errors (e.g., OOR)	Error in Response	Error returned in Response
6	CMD45	Type 'X' errors (e.g., WP violation)	OK (no error indication)	Type 'X' errors
7	CMD46/CMD47	Error in parameters: non-existent Task ID; Task ID is not ready for execution; Wrong direction	No Response	Illegal Command
8	CMD48	Invalid TM op-code	OK (no error indication)	Type 'X' error (ERROR)

6.6.39.9 Supported Commands

Command queuing can be enabled and disabled using CMDQ_MODE_EN field in the extended CSD. The queue must be empty prior to disabling it. Trial of disabling non-empty queue shall be regarded as illegal command.

When command queuing is enabled (CMDQ Mode En bit in CMDQ_MODE_EN field is set to '1') class 11 commands are the only method through which data transfer tasks can be issued. Existing data transfer commands, namely CMD18/CMD17 and CMD25/CMD24, are not supported when command queuing is enabled.

When command queuing is disabled, CMD17/18 and CMD24/25 are supported, as they are in previous revisions of *e*•MMC.

When the task queue is enabled and empty host should only send class11 commands in Trans state. However, if the device is in Trans state and processes a sequence of commands (e.g., erase sequence) host should complete the sequence before sending class11 commands.

When command queuing mode is enabled, some legacy commands may still be used, while other are treated as illegal. The legality status of commands in different queuing modes is listed in Table 45.

If CMD12 is issued during CMD46 data transfer, the device moves back to Transfer state and the operation can be considered completed (QSR[i] = 1 → 0).

If CMD12 is issued during CMD47 data transfer, the device moves to Programming state and the operation can be considered completed (QSR[i] = 1 → 0) only when all the transferred data have been programmed.

Table 45 — Supported Commands for Command Queue

Mode	Task Queue status	Commands	Status
CMDQ_MODE_EN is set to 0.	N/A	Class 11 commands	Illegal
		All other commands	Legal (same as *e*•MMC5.0)
CMDQ_MODE_EN is set to 1.	Empty	CMD17, CMD18, CMD 24, CMD 25	Illegal
		Class 11 commands	Legal
		All other commands	Legal (same as *e*•MMC5.0)
	Non-empty	Class 11 commands	Legal
		CMD0	Legal (all tasks in the queue and QSR will reset)
		CMD12	Legal
		CMD13	Legal
		All other commands	Illegal

6.6.40 Secure Write Protect Mode

Any application running on the host may issue write protection by updating fields of write protection related EXT_CSD, like USER_WP[171], BOOT_WP[173], by issuing CMD6, CMD8, CMD28 and CMD29. To prevent un-authorized changes of such write protection related EXT_CSD fields, host should enter the secure write protect mode by setting the SECURE_WP_EN field in SECURE_WP_MODE_CONFIG to 0x1.

If host enters the secure write protection mode, those write protection related fields like USER_WP[171] , BOOT_WP[173], TMP_WRITE_PROTECT[12] and PERM_WRITE_PROTECT[13] are updated only when SECURE_WP_MASK is set as 0x1. Those two SECURE_WP_MODE_CONFIG and SECURE_WP_MODE_ENABLE fields are defined in Device Configuration area shall be updated only by Authenticated Device Configuration write request.

Table 46 — Device Configuration Area

Name	Field	Size (Bytes)	Cell Type	Address
Reserved		253	TBD	[255~3]
Secure Write Protect Configuration	SECURE_WP_MODE_CONFIG	1	R/W/E_P	[2]
Secure Write Protect Enable	SECURE_WP_MODE_ENABLE	1	R/W/E	[1]
Reserved		1	TBD	[0]

6.7 Clock control

The *e*•MMC bus clock signal can be used by the host to put the Device into energy saving mode, or to control the data flow (to avoid under-run or over-run conditions) on the bus. The host is allowed to lower the clock frequency or shut it down.

There are a few restrictions the host must follow:

- The bus frequency can be changed at any time (under the restrictions of maximum data transfer frequency, defined by the Device, and the identification frequency defined by the standard document).

It is an obvious requirement that the clock must be running for the Device to output data or response tokens. After the last *e*•MMC bus transaction, the host is required, to provide 8 (eight) clock cycles for the Device to complete the operation before shutting down the clock. The various bus transactions are as follows:

- A command with no response. 8 clocks after the host command end bit.
- A command with response. 8 clocks after the Device response end bit.
- A read data transaction. 8 clocks after the end bit of the last data block.
- A write data transaction. 8 clocks after the CRC status token.

The host is allowed to shut down the clock of a "busy" Device. The Device will complete the programming operation regardless of the host clock. However, the host must provide a clock edge for the Device to turn off its busy signal. Without a clock edge the Device (unless previously disconnected by a deselect command (CMD7)) will force the DAT0 line down, forever.

6.8 Error conditions

6.8.1 CRC and illegal command

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

Similarly, if an illegal command has been received, the Device shall not change its state, shall not respond and shall set the ILLEGAL_COMMAND error bit in the status register. Only the non-erroneous state branches are shown in the state diagrams, (see Figure 25 through Figure 27). Table 60 contains a complete state transition description.

There are different kinds of illegal commands:

- Commands that belong to classes not supported by the Device (e.g., write commands in read only Devices).
- Commands not allowed in the current state (e.g., CMD2 in Transfer State).
- Commands that are not defined (e.g., CMD50).

NOTE The COM_CRC_ERROR and ILLEGAL_COMMAND errors are detected during the command interpretation and validation phase (Response Mode). However, since the Device does not respond to commands with COM_CRC_ERROR or ILLEGAL_COMMAND error, the errors are reported in the response of the following valid command.

6.8.2 Time-out conditions

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

- **Read**

The read access time is defined as the sum of the two times given by the CSD parameters TAAC and NSAC (see 6.15). These Device parameters define the typical delay between the end bit of the read command and the start bit of the data block. This number is Device dependent.

- **Write**

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

- **Erase / Secure Erase**

The duration of an erase command will be (order of magnitude) the number of Erase blocks to be erased multiplied by the block write delay. If ERASE_GROUP_DEF (EXT_CSD byte [175]) is enabled, ERASE_TIMEOUT_MULT should be used to calculate the duration.

Secure Erase timeout is calculated based on the Erase Timeout and additional SEC_ERASE_MULT

factor (EXT_CSD byte [230]).

- **TRIM/DISCARD/Secure TRIM**

The TRIM/DISCARD function timeout is calculated based on the TRIM_MULT factor (EXT_CSD byte [232]). Secure TRIM timeout is calculated based on the Erase Timeout and additional SEC_TRIM_MULT factor (EXT_CSD byte [229])

- **Force erase**

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

- **High-Priority Interrupt (HPI)**

OUT_OF_INTERRUPT_TIME (EXD_CSD byte[198]) defines the maximum time between the end bit of CMD12/13, arg[0]=1 to the DAT0 release by the device.

- **Partition Switch**

PARTITION_SWITCH_TIME (EXD_CSD byte[199]) defines the maximum time between the end bit of the SWITCH command (CMD6) to the DAT0 release by the device, when switching partitions by changing PARTITION_ACCESS bits in PARTITION_CONFIG field (EXT_CSD byte [179]).

6.8.3 Read ahead in multiple block read operation

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

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

6.9 Minimum performance

An *e*•MMC Device has to fulfill the requirements set for the read and write access performance.

6.9.1 Speed class definition

The speed class definition is for indication of the minimum performance of a Device. The classes are defined based on respectively the 150kB/s base value for single data rate operation (normal mode) and 300kB/s base value for dual data rate operation. The minimum performance of the Device can then be marked by defined multiples of the base value e.g., 2.4MB/s (SDR) or 4.8MB/s (DDR). Only following speed classes are defined. Speed class is not defined and is not applicable for *e*•MMC operating in HS200 mode and HS400 mode.

NOTE *e*•MMC Devices always including 8 bit data bus and the categories below states the configuration that the Device is operated.

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

- 2.4 MB/s (sdr) or 4.8 MB/s (ddr) Class A
- 3.0 MB/s (sdr) or 6.0 MB/s (ddr) Class B
- 4.5 MB/s (sdr) or 9.0 MB/s (ddr) Class C
- 6.0 MB/s (sdr) or 12.0 MB/s (ddr) Class D
- 9.0 MB/s (sdr) or 18.0 MB/s (ddr) Class E

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

- 12.0 MB/s (sdr) or 24.0 MB/s (ddr) Class F
- 15.0 MB/s (sdr) or 30.0 MB/s (ddr) Class G
- 18.0 MB/s (sdr) or 36.0 MB/s (ddr) Class H
- 21.0MB/s (sdr) or 42.0 MB/s (ddr) Class J

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

- 24.0 MB/s (sdr) or 48.0 MB/s (ddr) Class K
- 30.0 MB/s (sdr) or 60.0 MB/s (ddr) Class M
- 36.0 MB/s (sdr) or 72.0 MB/s (ddr) Class O
- 42.0 MB/s (sdr) or 84.0 MB/s (ddr) Class R
- 48.0 MB/s (sdr) or 96.0 MB/s (ddr) Class T

The performance values for both write and read accesses are stored into the EXT_CSD register for electrical reading (see 7.4). Only the defined values and classes are allowed to be used.

6.9.2 Measurement of the performance

Initial state of the memory in prior to the test is: filled with random data. The test is performed by writing/reading a 64 kB chunk of data to/from random logical addresses (aligned to physical block boundaries) of the Device. A predefined multiple block write/read is used with block count of 128 (64 kB as 512B blocks are used). The performance is calculated as average out of several 64kB accesses. Same test is performed with all applicable clock frequency and bus width options as follows:

- 52 MHz, 8bit bus in the dual data mode (if 52 MHz clock frequency and dual data mode is supported by the Device)
- 52 MHz, 8bit bus (if 52 MHz clock frequency is supported by the Device)
- 52 MHz, 4bit bus (if 52 MHz clock frequency is supported by the Device)
- 26 MHz, 8bit bus
- 26 MHz, 4bit bus

In case the minimum performance of the Device exceeds the physical limit of one of the above mentioned options the Device has to also fulfill accordingly the performance criteria as defined in MIN_PERF_a_b_ff.

6.10 Commands

6.10.1 Command types

There are four kinds of commands defined to control the *e*•MMC:

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

All commands and responses are sent over the CMD line of the *e*•MMC bus. The command transmission always starts with the left bit of the bit string corresponding to the command codeword.

6.10.2 Command format

All commands have a fixed code length of 48 bits, needing a transmission time of 0.92 micro second at 52 MHz.

Table 47 — Command Format

Description	Start Bit	Transmission Bit	Command Index	Argument	CRC7	End Bit
Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width (bits)	1	1	6	32	7	1
Value	“0”	“1”	x	x	x	“1”

A command always starts with a start bit (always ‘0’), followed by the bit indicating the direction of transmission (host = ‘1’). The next 6 bits indicate the index of the command, this value being interpreted as a binary coded number (between 0 and 63). Some commands need an argument (e.g., an address), that is coded by 32 bits. A value denoted by ‘x’ in Table 47 indicates this variable is dependent on the command. All commands are protected by a CRC (see 0 for the definition of CRC7). Every command codeword is terminated by the end bit (always ‘1’). All commands and their arguments are listed in Table 48 through Table 58.

6.10.3 Command classes

The command set of the *e*•MMC system is divided into several classes. (See Table 48)

Each class supports a subset of Device functions.

Class 0 is mandatory and shall be supported by all Devices. The other classes are either mandatory only for specific Device types or optional (refer to clause 11, for detailed description of supported command classes as a function of Device type). By using different classes, several configurations can be chosen (e.g., a block writable Device). The supported Device Command Classes (CCC) are coded as a parameter in the Device specific data (CSD) register of each Device, providing the host with information on how to access the Device.

Table 48 — Supported Device command classes (0–56)

Device Command Class (CCC)	Class Description	Supported Commands																							
		0	1	2	3	4	5	6	7	8	9	10	11 (2)	12	13	14	15	16	17	18	19	20 (2)	21 (1)	23	24
class 0	basic	+	+	+	+	+	+	+	+	+	+	+		+	+	+	+				+				
class 1	Obsolete																								
class 2	block read																	+	+	+				+	
class 3	Obsolete																								
class 4	block write																	+						+	+
class 5	erase																								
class 6	write protection																								
class 7	lock Device																	+							
class 8	Application-specific																								
class 9	I/O mode																								
class 10	Security Protocols																								
class11	Command Queuing																								
class12	Reserved																								

Device Command Class (CCC)	Class Description	Supported Commands																							
		25	26	27	28	29	30	31	35	36	38	39	40	42	44	45	46	47	48	49	53	54	55	56	
class 0	basic																								
class 1	Obsolete																								
class 2	block read																								
class 3	Obsolete																								
class 4	block write	+	+	+																+					
class 5	Erase								+	+	+														
class 6	write protection				+	+	+	+																	
class 7	lock Device													+											
class 8	Application-specific																						+	+	
class 9	I/O mode											+	+												
class 10	Security Protocols																				+	+			
class11	Command Queuing														+	+	+	+	+						
class 12	Reserved																								

NOTE 1 Mandatory only for devices that supports HS200 mode.

NOTE 2 Obsolete.

6.10.4 Detailed command description

Table 49 through Table 59 define in detail all *e*•MMC bus commands. The responses R1-R5 are defined in 6.12. The registers CID, CSD, EXT_CSD and DSR are described in clause 7.

Table 49 — Basic commands (class 0 and class 1)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD0	bc	[31:0] 00000000	—	GO_IDLE_STATE	Resets the Device to *idle* state
	bc	[31:0] F0F0F0F0	—	GO_PRE_IDLE_STATE	Resets the Device to *pre-idle* state
	—	[31:0]FFFFFFFA	—	BOOT_INITIATION	Initiate alternative boot operation
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks Device, in *idle* state, to send its Operating Conditions Register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks Device to send its CID number on the CMD line
CMD3	ac	[31:16] RCA [15:0] stuff bits	R1	SET_RELATIVE_ADDR	Assigns relative address to the Device
CMD4	bc	[31:16] DSR [15:0] stuff bits	—	SET_DSR	Programs the DSR of the Device
CMD5	ac	[31:16] RCA [15]Sleep/Awake [14:0] stuff bits	R1b	SLEEP_AWAKE	Toggles the Device between Sleep state and Standby state. (See 6.6.21).
CMD6	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Switches the mode of operation of the selected Device or modifies the EXT_CSD registers. (See 6.6.1)
CMD7	ac	[31:16] RCA [15:0] stuff bits	R1/ R1b[1]	SELECT/DESELECT_CARD	Command toggles a device between the standby and transfer states or between the programming and disconnect states. NOTE In both cases the Device is selected by its own relative address and gets deselected by any other address; address 0 deselects the Device.
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	Device sends its EXT_CSD register as a block of data.
CMD9	ac	[31:16] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed Device sends its Device-specific data (CSD) on the CMD line.
CMD10	ac	[31:16] RCA [15:0] stuff bits	R2	SEND_CID	Addressed Device sends its Device identification (CID) on CMD the line.
CMD11				obsolete	The response to CMD11 will be undefined.
CMD12	ac	[31:16] RCA[2] [15:1] stuff bits [0] HPI	R1/ R1b[3]	STOP_TRANSMISSION	Forces the Device to stop transmission. If HPI flag is set the device shall interrupt its internal operations in a well-defined timing.
CMD13	ac	[31:16] RCA [15] SQS [14:1] stuff bits [0] HPI	R1	SEND_STATUS	In case SQS bit = 0: Addressed Device sends its status register. If HPI flag is set the device shall interrupt its internal operations in a well-defined timing. In case SQS bit = 1: indicate that this is a QSR query. In response device shall send the QSR (Queue Status Register). In this case HPI must be set to '0'.
CMD14	adtc	[31:0] stuff bits	R1	BUSTEST_R	A host reads the reversed bus testing data pattern from a Device.
CMD15	ac	[31:16] RCA [15:0] stuff bits	–	GO_INACTIVE_STATE	Sets the Device to *inactive* state
CMD19	adtc	[31:0] stuff bits	R1	BUSTEST_W	A host sends the bus test data pattern to a Device.

NOTE 1 R1 while selecting from Stand-By State to Transfer State; R1b while selecting from Disconnected State to Programming State.
NOTE 2 RCA in CMD12 is used only if HPI bit is set. The argument does not imply any RCA check on the device side.
NOTE 3 R1 for read cases and R1b for write cases.

6.10.4 Detailed command description (cont'd)

Table 50 — Block-oriented read commands (class 2)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address[1]	R1	READ_SINGLE_ BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.[2]
CMD18	adtc	[31:0] data address[1]	R1	READ_MULTIPLE_ BLOCK	Continuously transfers data blocks from Device to host until interrupted by a stop command, or the requested number of data blocks is transmitted If sent as part of a packed read command, the argument shall contain the first read data address in the pack (address of first individual read command inside the pack). (See 6.6.27.1)
CMD21	adtc	[31:0] stuff bits	R1	SEND_TUNING_ BLOCK	128 clocks of tuning pattern (64 byte in 4 bit mode or 128 byte in 8 bit mode) is sent for HS200 optimal sampling point detection.

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

Table 51 — Class 3 commands

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD20				Obsolete	The response to CMD20 will be undefined.
CMD22	Reserved				

6.10.4 Detailed command description (cont'd)

Table 52 — Block-oriented write commands (class 4)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD23 (default)	ac	[31] Reliable Write Request [30] '0' non- packed [29] tag request [28:25] context ID [24]: forced programming [23:16] set to 0 [15:0] number of blocks	R1	SET_BLOCK_COUNT	**non-packed command version** Defines the number of blocks (read/write) and the reliable writer parameter (write) for a block read or write command. (See 6.6.7 and 6.6.8) May contain a tag request (see 6.6.28) or a context ID (see 6.6.27). Tag and Context ID cannot be used together in the same command, if one is used the other must be set to zero. The context ID is an identifier (0 to 15) that associates the read/write command with the specific context. When bit 24 is set to 1, forced programming enabled, data shall be forcefully programmed to nonvolatile storage instead of volatile cache while cache is turned ON.
CMD23 (packed)	ac	[31] set to 0 [30] '1' packed [29:16] set to 0 [15:0] number of blocks	R1	SET_BLOCK_COUNT	**packed command version** Defines the number of blocks (read/write) for the following packed write command or for the header of the following packed read command. (See 6.6.27.1). For packed write commands, the number of blocks should include the total number of blocks all packed commands plus one for the header block. For packed read commands, the number of blocks should equal one as only the header is sent inside the following CMD25. After that, a separate normal read command is sent to get the packed data.
CMD24	adtc	[31:0] data address[1]	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.[2]
CMD25	adtc	[31:0] data address[1]	R1	WRITE_MULTIPLE_ BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows or the requested number of block received. If sent as a packed command (either packed write, or the header of packed read) the argument shall contain the first read/write data address in the pack (address of first individual command inside the pack). (See 6.6.27.1)
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the Device identification register. This command shall be issued only once. The Device contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD49	adtc	[31:0] stuff bits	R1	SET_TIME	Sets the real time clock according to the RTC information in the 512 B data block.

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

6.10.4 Detailed command description (cont'd)

Table 53 — Block-oriented write protection commands (class 6)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD28	ac	[31:0] data address[1]	R1b	SET_WRITE_PROT	**If CLASS_6_CTRL=0x00:** If the Device has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the Device specific data (WP_GRP_SIZE or HC_WP_GRP_SIZE). **If CLASS_6_CTRL=0x01:** This command releases the specified addressed group.
CMD29	ac	[31:0] data address[1]	R1b	CLR_WRITE_PROT	**If CLASS_6_CTRL=0x00:** If the Device provides write protection features, this command clears the write protection bit of the addressed group. **If CLASS_6_CTRL=0x01:** This command is ignored.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	**If CLASS_6_CTRL=0x00:** If the Device provides write protection features, this command asks the Device to send the status of the write protection bits.[2] **If CLASS_6_CTRL=0x01:** This command asks the device to send the status of released groups. A bit '0' means the specific group is valid and accessible, a bit '1' means the specific group was released and it cannot be used.[3]
CMD31	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT_TYPE	**If CLASS_6_CTRL=0x00:** This command sends the type of write protection that is set for the different write protection groups[4.] **If CLASS_6_CTRL=0x01:** This command returns a fixed pattern of 64-bit zeros in its payload.

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

NOTE 2 32 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits are transferred in a payload format via the data lines. The last (least significant) bit of the protection bits corresponds to the first addressed group. If the addresses of the last groups are outside the valid range, then the corresponding write protection bits shall be set to zero.

NOTE 3 32 released status bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits are transferred in a payload format via the data lines. The last (least significant) bit of the released bits corresponds to the first addressed group. If the addresses of the last groups are outside the valid range, then the corresponding released bits shall be set to zero.

NOTE 4 64 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits are transferred in a payload format via the data lines. Each set of two protection bits shows the type of protection set for each of the write protection groups. The definition of the different bit settings are shown below. The last (least significant) two bits of the protection bits correspond to the first addressed group. If the addresses of the last groups are outside the valid range, then the corresponding write protection bits shall be set to zero.
"00" Write protection group not protected
"01" Write protection group is protected by temporary write protection
"10" Write protection group is protected by power-on write protection
"11" Write protection group is protected by permanent write protection

6.10.4 Detailed command description (cont'd)

Table 54 — Erase commands (class 5)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD32 … CMD34	Reserved. These command indexes cannot be used in order to maintain backwards compatibility with older versions of *e*•MMCs				
CMD35	ac	[31:0] data address[1,2]	R1	ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase
CMD36	ac	[31:0] data address[1,2]	R1	ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase
CMD37	Reserved. This command index cannot be used in order to maintain backwards compatibility with older versions of *e*•MMCs				
CMD38	ac	[31] Secure request[4] [30:16] set to 0 [15] Force Garbage Collect request[4] [14:2] set to 0 [1] Discard Enable [0] Identify Write Blocks for Erase (or TRIM Enable)	R1b	ERASE	Erases all previously selected write blocks according to argument bits[3] When all argument bits are zero CMD38 will perform an erase on erase group(s). When Bit 0 = 1 and Bit 1=0 then CMD38 will perform a TRIM on the sector(s). When Bit 0 =1 and Bit 1=1 then CMD38 will perform a DISCARD on the sector(s) To maintain backward compatibility the device must not report an error if bits 31 and 15 are set. The device behavior when these are set is undefined. All other argument settings should trigger an ERROR.

NOTE 1 Data address for media =<2 GB is a 32 bit byte address and data address for media > 2 GB is a 32 bit sector (512B) address.
NOTE 2 The Device will ignore all LSB's below the Erase Group size, effectively rounding the address down to the Erase Group boundary.
NOTE 3 Table 11 and Table 12 give a description of the argument bits and a list of supported argument combinations.
NOTE 4 Argument bit 15 is an optional feature that is only supported if SEC_GB_CL_EN (EXT_CSD[231] bit 4) is set. Argument bit 31 is an optional feature that is only supported if SEC_ER_EN (EXT_CSD[231] bit 0) is set

6.10.4 Detailed command description (cont'd)

Table 55 — I/O mode commands (class 9)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD39	ac	[31:16] RCA [15:15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8 bit (register) data fields. The command addresses a Device and a register and provides the data for writing if the write flag is set. The R4 response contains data read from the addressed register if the write flag is cleared to 0. This command accesses application dependent registers that are not defined in the *e*•MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode
CMD41	Reserved				

Table 56 — Lock Device commands (class 7)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD42	adtc	[31:0] stuff bits.	R1	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the Device. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43	Reserved				

Table 57 — Application-specific commands (class 8)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the Device that the next command is an application specific command rather than a standard command
CMD56	adtc	[31:1] stuff bits. [0]: RD/WR1	R1	GEN_CMD	Used either to transfer a data block to the Device or to get a data block from the Device for general purpose / application specific commands. The size of the data block shall be set by the SET_BLOCK_LEN command.
CMD57 … CMD59	Reserved				
CMD60 … CMD63	Reserved for manufacturer				
NOTE	RD/WR: "1" the host gets a block of data from the Device. "0" the host sends block of data to the Device.				

6.10.4 Detailed command description (cont'd)

Table 58 — Security Protocols (class 10)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD50-52	Reserved				
CMD53	adtc	[16:31] Security Protocol Specific [15:8] Security Protocol [7:0] reserved	R1	PROTOCOL_RD	Continuously transfers data blocks from device to host. Number of data blocks shall be defined by a preceding CMD23. Data Transfer may be interrupted by a STOP_TRANSMISSION command, This command is not supported if sent as a packed command. Block size is always 512bytes.
CMD54	adtc	[16:31] Security Protocol Specific [15:8] Security Protocol [7:0] reserved	R1	PROTOCOL_WR	Continuously transfers data blocks from host to device. Number of data blocks shall be defined by a preceding CMD23. Data Transfer may be interrupted by a STOP_TRANSMISSION command, This command is not supported if sent as a packed command. Block size is always 512bytes.

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

6.10.4 Detailed command description (cont'd)

Table 59 — Command Queue (Class 11)

CMD INDEX	Type	Argument	Resp	Abbreviation	Command Description
CMD44	ac	[31] Reliable Write Request [30] DD: "1" read / "0" write [29] Tag request [28:25] Context ID [24]: Forced Programming [23] Priority: "0" simple / "1" high [22:21] reserved [20:16] Task ID [15:0] Number of Blocks	R1	QUEUED_TASK _PARAMS	Defines Data Direction (DD) of operation (read/write), Priority (high/simple), Task ID, and block count of queued task. CMD44 settings shall be compliant with the indications given in Table 34, Features Cross Reference Table.
CMD45	ac	[31:0] Start block address	R1	QUEUED_TASK _ADDRESS	Defines the block address of queued task
CMD46	adtc	[31:21] reserved [20:16] Task ID [15:0] reserved	R1	EXECUTE_READ _TASK	Device shall execute task from the queue whose ID is encoded in the argument
CMD47	adtc	[31:21] reserved [20:16] Task ID [15:0] reserved	R1	EXECUTE_WRITE _TASK	Device shall execute task from the queue whose ID is encoded in the argument
CMD48	ac	[31:21] reserved [20:16]: TaskID [15:4]: reserved [3:0] TM op-code (Table 5)	R1b	CMDQ_TASK _MGMT	Device shall discard a specific task or entire queue (all tasks in the queue) [20:16] when TM op-code = 2h these bits represent TaskID. When TM op-code = 1h these bits are reserved.

6.11 Device state transition table

Table 60 defines the Device state transitions in dependency of the received command.

Table 60 — Device state transitions

	Current State												
	idle	**ready**	**ident**	**stby**	**tran**	**data**	**btst**	**rcv**	**prg**	**dis**	**ina**	**slp**	**irq**
Command	**Changes to**												
Class Independent													
CRC error	-	-	-	-	-	-	-	-	-	-	-	-	stby
command not supported	-	-	-	-	-	-	-	-	-	-	-	-	stby
Class 0													
CMD0 (arg=0x00000000)	idle	idle	idle	idle	idle	idle	idle	idle	idle	idle	-	idle	stby
CMD0 (arg=0xF0F0F0F0)	Pre-idle	Pre-idle	Pre-idle	Pre-idle	Pre-idle	Pre-idle	Pre-idle	Pre-idle	Pre-idle	Pre-idle	-	Pre-idle	stby
CMD1, Device V_{CCQ} range[6] compatible	ready	-	-	-	-	-	-	-	-	-	-	-	stby
CMD1, Device is busy	idle	-	-	-	-	-	-	-	-	-	-	-	stby
CMD1, Device V_{CCQ} range[6] not compatible	ina	-	-	-	-	-	-	-	-	-	-	-	stby
CMD2, Device wins bus	-	ident	-	-	-	-	-	-	-	-	-	-	stby
CMD2, Device loses bus	-	ready	-	-	-	-	-	-	-	-	-	-	stby
CMD3	-	-	stby	-	-	-	-	-	-	-	-	-	stby
CMD4	-	-	-	stby	-	-	-	-	-	-	-	-	stby
CMD5	-	-	-	slp								stby	stby
CMD6	-	-	-	-	prg	-	-	-	-	-	-	-	stby
CMD7, Device is addressed	-	-	-	tran	-	-	-	-	-	prg	-	-	stby
CMD7, Device is not addressed	-	-	-	-	stby	stby	-	-	dis	-	-	-	stby
CMD8	-	-	-	-	data	-	-	-	-	-	-	-	stby
CMD9	-	-	-	stby	-	-	-	-	-	-	-	-	stby
CMD10	-	-	-	stby	-	-	-	-	-	-	-	-	stby
CMD12, arg[0] = 0	-	-	-	-	-	tran	-	prg	-	-	-	-	stby
CMD12, arg[0] = 1	-	-	-	-	-	-	-	-	prg	-	-	-	stby
CMD13, arg[0] = 0 or 1	-	-	-	stby	tran	data	btst	rcv	prg	dis	-	-	stby
CMD14	-	-	-	-	-	-	tran	-	-	-	-	-	stby
CMD15	-	-	-	ina	ina	ina	ina	ina	ina	ina	-	-	stby
CMD19	-	-	-	-	btst	-	-	-	-	-	-	-	stby
Class 1													
CMD11 (obsolete)	-	-	-	-	-	-	-	-	-	-	-	-	-
Class 2													
CMD16	-	-	-	-	tran	-	-	-	-	-	-	-	stby
CMD17	-	-	-	-	data	-	-	-	-	-	-	-	stby
CMD18	-	-	-	-	data	-	-	-	-	-	-	-	stby
CMD21	-	-	-	-	data	-	-	-	-	-	-	-	stby
CMD23	-	-	-	-	tran	-	-	-	-	-	-	-	stby
Class 3													
CMD20 (obsolete)	-	-	-	-	-	-	-	-	-	-	-	-	-

6.11 Device state transition table (cont'd)

Table 61 — Device state transitions (cont'd)

	Current State												
	idle	**ready**	**ident**	**stby**	**tran**	**data**	**btst**	**rcv**	**prg**	**dis**	**ina**	**slp**	**Irq**
Command	**Changes to**												
Class 4													
CMD16	see class 2												
CMD23	see class 2												
CMD24	-	-	-	-	rcv	-	-	-	rcv1	-	-	-	stby
CMD25	-	-	-	-	rcv	-	-	-	rcv2	-	-	-	stby
CMD26	-	-	-	-	rcv	-	-	-	-	-	-	-	stby
CMD27	-	-	-	-	rcv	-	-	-	-	-	-	-	stby
CMD49	-	-	-	-	rcv	-	-	-	-	-	-	-	stby
Class 6													
CMD28	-	-	-	-	prg	-	-	-	-	-	-	-	stby
CMD29	-	-	-	-	prg	-	-	-	-	-	-	-	stby
CMD30	-	-	-	-	data	-	-	-	-	-	-	-	stby
CMD31	-	-	-	-	data	-	-	-	-	-	-	-	stby
Class 5													
CMD35	-	-	-	-	tran	-	-	-	-	-	-	-	stby
CMD36	-	-	-	-	tran	-	-	-	-	-	-	-	stby
CMD38	-	-	-	-	prg	-	-	-	-	-	-	-	stby
Class 7													
CMD16	see class 2												
CMD42	-	-	-	-	rcv	-	-	-	-	-	-	-	stby
Class 8													
CMD55	-	-	-	stby	tran	data	btst	rcv	prg	dis	-	-	irq
CMD56; RD/WR = 0	-	-	-	-	rcv	-	-	-	-	-	-	-	stby
CMD56; RD/WR = 1	-	-	-	-	data	-	-	-	-	-	-	-	stby
Class 9													
CMD39	-	-	-	stby	-	-	-	-	-	-	-	-	stby
CMD40	-	-	-	irq	-	-	-	-	-	-	-	-	stby
Class 10													
CMD53	-	-	-	-	data	-	-	-	-	-	-	-	Stby
CMD54	-	-	-	-	rcv	-	-	-	-	-	-	-	Stby
Class 11													
CMD44	-	-	-	-	tran	data	-	rcv	prg	-	-	-	-
CMD45	-	-	-	-	tran	data	-	rcv	prg	-	-	-	-
CMD46	-	-	-	-	data	-	-	-	-	-	-	-	-
CMD47	-	-	-	-	rcv	-	-	-	-	-	-	-	-
CMD48	-	-	-	-	prg	-	-	-	-	-	-	-	-
Class 12													
CMD41; CMD43; CMD50…CMD52, CMD57…CMD59	Reserved												
CMD60…CMD63	Reserved for Manufacturer												

6.11 Device state transition table (cont'd)

Table 62 — Device state transitions (cont'd)

	Current State												
	idle	**ready**	**ident**	**stby**	**tran**	**data**	**btst**	**rcv**	**prg**	**dis**	**ina**	**slp**	**irq**
Command	**Changes to**												
NOTE 1 Due to legacy considerations, a Device may treat CMD24/25 during a prg state—while busy is active—as a legal or an illegal command. A Device that treats CMD24/25 during a prg-state—while busy is active—as an illegal command will not change its state to the rcv state. A host should not send CMD24/25 while the Device is in prg state and busy is active.													
NOTE 2 Due to legacy considerations, a Device may treat CMD24/25 during a prg state—while busy is active—as a legal or an illegal command. A Device that treats CMD24/25 during a prg state—while busy is active—as an illegal command will not change its state to the rcv state. A host should not send CMD24/25 while the Device is in prg state and busy is active.													
NOTE 3 As there is no way to obtain state information in boot mode, boot-mode states are not shown in this table.													
NOTE 4 For details on Pre-Idle, Pre-Boot and Boot state and CMD0 (0xFFFFFFFA) transitions, refer to 6.3.													
NOTE 5 CMD44-CMD48 state transitions described in this table relate to scenario that Command Queue feature is enabled (CMDQ_MODE_EN=xxxxxxx1b).													
NOTE 6 VDD range in case of High Voltage MultimediaCard.													

6.12 Responses

All responses are sent via the command line CMD. The response transmission always starts with the left bit of the bit string corresponding to the response codeword. The code length depends on the response type. A response always starts with a start bit (always '0'), followed by the bit indicating the direction of transmission (Device = '0'). A value denoted by 'x' in Table 63 – Table 67 indicates a variable entry. All responses, except for the type R3, are protected by a CRC (see 0 for the definition of CRC7). Every command codeword is terminated by the end bit (always '1').

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

- **R1** (normal response command): code length 48 bit. The bits 45:40 indicate the index of the command to be responded to, this value being interpreted as a binary coded number (between 0 and 63). The status of the Device is coded in 32 bits. The Device status is described in 6.13.

Table 63 — R1 response

Description	Start bit	Transmission bit	Command index	Device status	CRC7	End bit
Bit position	47	46	[45:40]	[39:8]	7	0
Width (bits)	1	1	6	32	x	1
Value	"0"	"0"	x	x	CRC7	"1"

- **R1b** is identical to R1 with an optional busy signal transmitted on the data line DAT0. The Device may become busy after receiving these commands based on its state prior to the command reception. Refer to 6.15 for detailed description and timing diagrams.

6.12 Responses (cont'd)

- **R2** (CID, CSD register): code length 136 bits. The contents of the CID register are sent as a response to the commands CMD2 and CMD10. The contents of the CSD register are sent as a response to CMD9. Only the bits [127...1] of the CID and CSD are transferred, the reserved bit [0] of these registers is replaced by the end bit of the response.

Table 64 — R2 response

Description	Start bit	Transmission bit	Check bits	CID or CSD register incl. internal CRC7	End bit
Bit position	135	134	[133:128]	[127:1]	0
Width (bits)	1	1	6	127	1
Value	"0"	"0"	111111	x	"1"

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

Table 65 — R3 Response

Description	Start bit	Transmission bit	Check bits	OCR register	Check bits	End bit
Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width (bits)	1	1	6	32	7	1
Value	"0"	"0"	"111111"	x	"1111111"	"1"

- **R4** (Fast I/O): code length 48 bits. The argument field contains the RCA of the addressed Device, the register address to be read out or written to, and its contents. The status bit in the argument is set if the operation was successful.

Table 66 — R4 response

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

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

Table 67 — R5 response

Description	Start bit	Transmission bit	CMD40	RCA [31:16] of winning Device or of the host	[15:0] Not defined. May be used for IRQ data	CRC 7	End bit
Bit position	47	46	[45:40]	[39:8] Argument field		[7:1]	0
Width (bits)	1	1	6	16	16	7	1
Value	"0"	"0"	"101000"	x	x	x	"1"

6.13 Device status

The response format R1 contains a 32-bit field named *Device status*. This field is intended to transmit the Device's status information. Two different attributes are associated with each one of the Device status bits:

- Bit type. Two types of Device status bits are defined:
  1) Error bit. Signals an error condition was detected by the device. These bits are cleared as soon as the response (reporting the error) is sent out. Clear Cond. B
  2) Status bit. These bits serve as information fields only, and do not alter the execution of the command being responded to. These bits are persistent; they are set and cleared in accordance with the Device status. Clear Cond. A

  The "Type" field of Table 68 defines the type of each bit in the Device status register. The symbol "E" is used to denote an Error bit while the symbol "S" is used to denote a Status bit.

- Detection mode of Error bits.

  Exceptions are detected by the Device either during the command interpretation and validation phase (Response Mode) or during command execution phase (Execution Mode). Response mode exceptions are reported in the response to the command that raised the exception. Execution mode exceptions are reported in the response to a STOP_TRANSMISSION command used to terminate the operation or in the response to a SEND_STATUS command issued while the operation is being carried out or after the operation is completed. The "Det Mode" field of Table 68 defines the detection mode of each bit in the Device status register. The symbol "R" is used to denote a Response Mode detection while the symbol "X" is used to denote an Execution Mode detection.

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

  The ADDRESS_OUT_OF_RANGE and ADDRESS_MISALIGN exceptions may be detected both in Response and Execution modes. The conditions for each one of the modes are explicitly defined in Table 68.

When Command Queue feature is enabled (CMDQ_MODE_EN=xxxxxxx1b) and R1 response is returned to a CMD13 command with SQS = 1, R1 response format is defined as QSR.

6.13 Device status (cont'd)

Table 68 — Device status

Bits	Identifier	Type	Det Mode	Value	Description	Clear Cond
31	ADDRESS_ OUT_OF_RANGE	E	R	"0" = no error "1" = error	The command's address argument was out of the allowed range for this Device.	B
			X		A multiple block operation is (although started in a valid address) attempting to read or write beyond the Device capacity	
30	ADDRESS_ MISALIGN	E	R	"0" = no error "1" = error	The command's address argument (in accordance with the currently set block length) positions the first data block misaligned to the Device physical blocks.	B
			X		A multiple block read/write operation (although started with a valid address/block length combination) is attempting to read or write a data block that does not align with the physical blocks of the Device.	
29	BLOCK_LEN_ ERROR	E	R	"0" = no error "1" = error	Either the argument of a SET_BLOCKLEN command exceeds the maximum value allowed for the Device, or the previously defined block length is illegal for the current command (e.g., the host issues a write command, the current block length is smaller than the Device's maximum and write partial blocks is not allowed)	B
28	ERASE_SEQ_ ERROR	E	R	"0" = no error "1" = error	An error in the sequence of erase commands occurred.	B
27	ERASE_PARAM	E	X	"0" = no error "1" = error	An invalid selection of erase groups for erase occurred.	B
26	WP_VIOLATION	E	X	"0" = no error "1" = error	Attempt to program a write protected block	B
25	DEVICE_IS_LOCKED	S	R	"0" = Device unlocked "1" = Device locked	When set, signals that the Device is locked by the host	A
24	LOCK_UNLOCK_ FAILED	E	X	"0" = no error "1" = error	Set when a sequence or password error has been detected in lock/unlock Device command	B
23	COM_CRC_ERROR	E	R	"0" = no error "1" = error	The CRC check of the command failed.	B
22	ILLEGAL_ COMMAND	E	R	"0" = no error "1" = error	Command not legal for the Device state	B
21	DEVICE_ECC_ FAILED	E	X	"0" = success "1" = failure	Device internal ECC was applied but failed to correct the data.	B
20	CC_ERROR	E	R	"0" = no error "1" = error	(Undefined by the standard) A Device error occurred that is not related to the host command.	B
19	ERROR	E	X	"0" = no error "1" = error	(Undefined by the standard) A generic Device error related to the (and detected during) execution of the last host command (e.g., read or write failures).	B
18	obsolete				Hosts should ignore this bit	

Bits	Identifier	Type	Det Mode	Value	Description	Clear Cond
17	obsolete				Hosts should ignore this bit	
16	CID/CSD_ OVERWRITE	E	X	“0” = no error “1” = error	Can be either one of the following errors: - The CID register has been already written and cannot be overwritten - The read only section of the CSD does not match the Device content. - An attempt to reverse the copy (set as original) or permanent WP (unprotected) bits was made.	B
15	WP_ERASE_SKIP	E	X	“0” = not protected “1” = protected	Only partial address space was erased due to existing write protected blocks.	B
14	Reserved, must be set to 0					
13	ERASE_RESET	E	R	“0” = cleared “1” = set	An erase sequence was cleared before executing because an out of erase sequence command was received (commands other than CMD35, CMD36, CMD38 or CMD13	B
12:9	CURRENT_STATE	S	R	0 = Idle 1 = Ready 2 = Ident 3 = Stby 4 = Tran 5 = Data 6 = Rcv 7 = Prg 8 = Dis 9 = Btst 10 = Slp 11–15 = reserved	The state of the Device when receiving the command. If the command execution causes a state change, it will be visible to the host in the response on the next command. The four bits are interpreted as a binary number between 0 and 15. NOTE As there is no way to obtain state information in boot mode, boot-mode states are not shown in this table.	A
8	READY_FOR_ DATA	S	R	“0” = not ready “1” = ready	Corresponds to buffer empty signaling on the bus	A
7	SWITCH_ERROR	E	X	“0” = no error “1” = switch error	If set, the Device did not switch to the expected mode as requested by the SWITCH command	B
6	EXCEPTION_EVE NT	S	R	"0" = no event "1" = an exception event has occurred	If set, one of the exception bits in field EXCEPTION_EVENTS_STATUS was set to indicate some exception has occurred. Host should check that field to discover the exception has occurred to understand what further actions are needed in order to clear this bit.	A
5	APP_CMD	S	R	“0” = Disabled “1” = Enabled	The Device will expect ACMD, or indication that the command has been interpreted as ACMD	A
4	Reserved					
3:2	Reserved for Application Specific commands					
1:0	Reserved for Manufacturer Test Mode					

6.13 Device status (cont'd)

Table 69 defines, for each command responded by a R1 response, the affected bits in the status field. A "R" or a "X" mean the error/status bit may be affected by the respective command (using the R or X detection mechanism respectively). The Status bits are valid in any R1 response and are marked with "S".

Table 69 — Device Status field/command - cross reference

CMD#	Response 1 Format – Status bit #																						
	31	30	29	28	27	26	25	24	23*	22*	21	20	19	18	17	16	15	13	12:9	8	7	6	5
0							S		R			R	X						S	S		S	
1									R	R			X										
2									R	R			X										
3							S		R	R		R	X						S	S		S	
4							S		R	R		R	X						S	S		S	
5							S		R	R		R	X					R	S	S		S	
6							S		R	R		R	X					R	S	S	X	S	
7							S		R	R		R	X					R	S	S		S	
8							S		R	R		R	X					R	S	S		S	
9							S		R	R		R	X					R	S	S		S	
10							S		R	R		R	X					R	S	S		S	
11 (1)																							
12							S		R	R		R	X						S	S		S	
13							S		R	R		R	X						S	S		S	
14							S		R	R		R	X					R	S	S		S	
15							S		R			R	X					R	S	S		S	
16			R				S		R	R		R	X					R	S	S		S	
17	R	R	R				S		R	R	X	R	X					R	S	S		S	
18	R	R	R				S		R	R	X	R	X					R	S	S		S	
19							S		R	R		R	X					R	S	S		S	
20 (1)																							
21							S		R	R	X	R	X					R	S	S		S	
23							S		R	R		R	X					R	S	S		S	
24	R	R	R				S		R	R		R	X					R	S	S		S	
25	R	R	R				S		R	R		R	X					R	S	S		S	
26							S		R	R		R	X			X		R	S	S		S	
27							S		R	R		R	X			X		R	S	S		S	
28	R						S		R	R		R	X					R	S	S		S	
29	R						S		R	R		R	X					R	S	S		S	
30	R			R	X		S		R	R		R	X					R	S	S		S	
35	R			R	X		S		R	R		R	X						S	S		S	
36				R			S		R	R		R	X						S	S		S	
38							S		R	R		R	X				X		S	S		S	
39							S		R	R		R	X					R	S	S		S	
40							S		R	R		R	X					R	S	S		S	
42							S	X	R	R		R	X					R	S	S		S	
53	R	R	R				S		R	R	X	R	X					R	S	S		S	
54	R	R	R				S		R	R		R	X					R	S	S		S	
55							S		R			R	X					R	S	S		S	S
56							S		R	R		R	X					R	S	S		S	S

NOTE 1 Commands are obsolete

6.13 Device status (cont'd)

Table 70 — Response 1 Status Bit Valid

CMD#	Response 1 Format – Status bit #																						
	31	30	29	28	27	26	25	24	23*	22*	21	20	19	18	17	16	15	13	12:9	8	7	6	5
Bit is valid for classes	1, 2, 3, 4, 5, 6, 10	2, 4, 10	2, 4, 7, 10	5	5	3, 4, 10	ALWAYS	7	ALWAYS	ALWAYS	1, 2, 10	ALWAYS	ALWAYS	1	3	ALWAYS	5	ALWAYS	ALWAYS	ALWAYS	ALWAYS	ALWAYS	ALWAYS
* The COM_CRC_ERROR and ILLEGAL_COMMAND errors are detected during the command interpretation and validation phase (Response Mode). However, since the Device does not respond to commands with COM_CRC_ERROR or ILLEGAL_COMMAND error, the errors are reported in the response of the following valid command.																							

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

6.14 Memory array partitioning

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

For block oriented commands, the following definition is used:

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

Special erase and write protect commands are defined:

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

6.14 Memory array partitioning (cont'd)

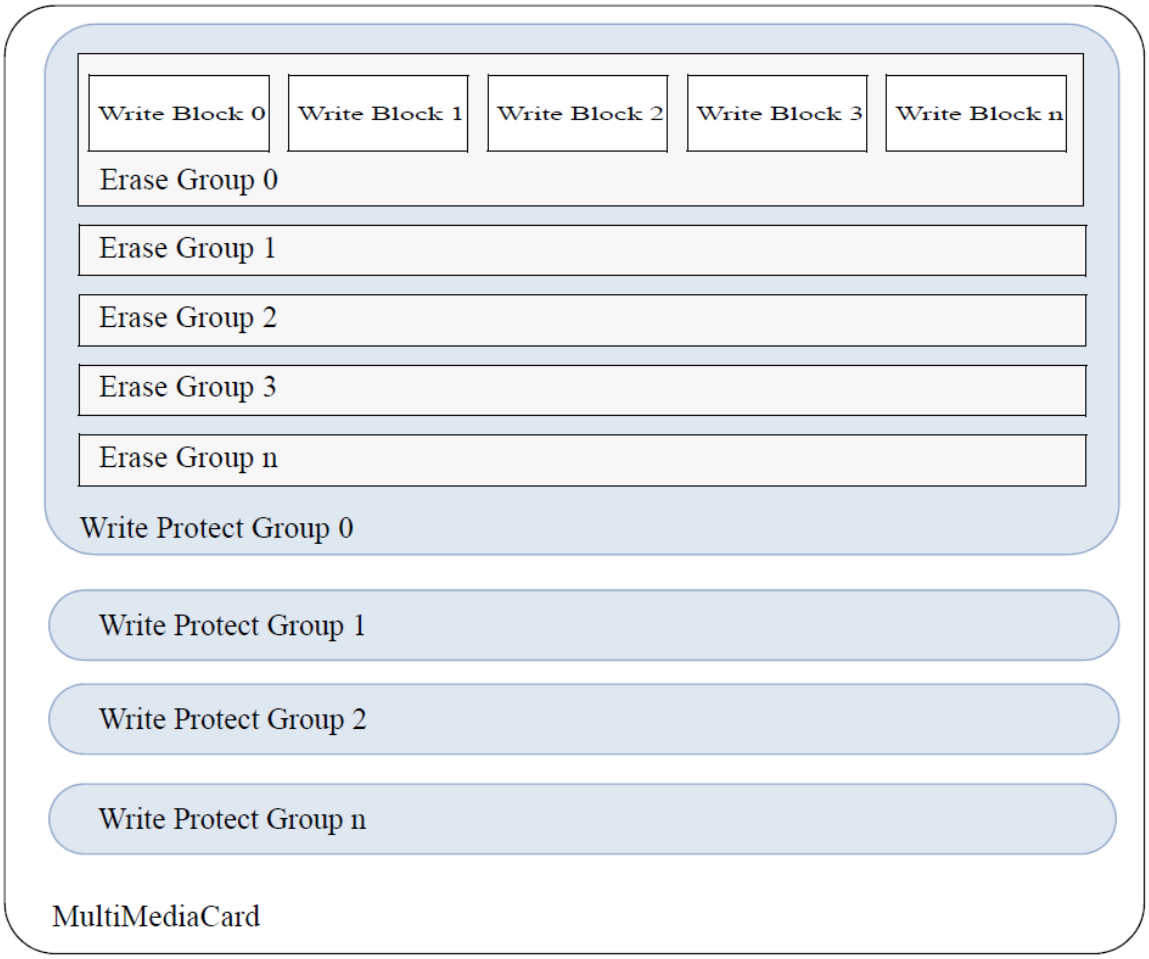


Figure 35 — Memory array partitioning

6.15 Timings

All timing diagrams use the following schematics and abbreviations:

S	Start bit (= “0”)
T	Transmitter bit (Host = “1,” Device = “0”)
P	One-cycle pull-up (= “1”)
E	End bit (= “1”)
L	One-cycle pull-down (= “0”)
Z	High impedance state (-> = “1”)
X	Driven value, “1” or “0”
D	Data bits
*	Repetition
CRC	Cyclic redundancy check bits (7 bits)
	Device active
	Host active

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

All timing values are defined in Table 71.

6.15.1 Command and response

Both host command and Device response are clocked out with the rising edge of the host clock in either the single data rate or dual data rate modes.

- Device identification and Device operation conditions timing

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

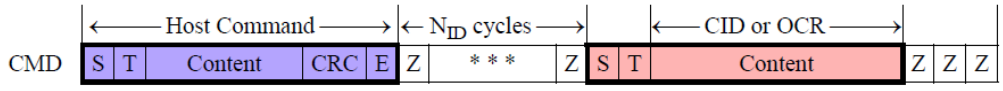


Figure 36 — Identification timing (Device identification mode)

- Assign a Device relative address

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

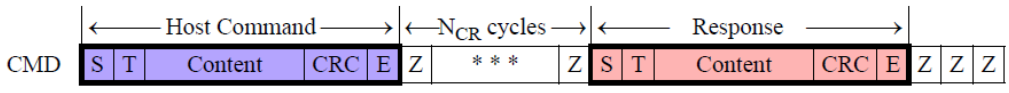


Figure 37 — SET_RCA timing (Device identification mode)

6.15.1 Command and response (cont'd)

- Data transfer mode

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

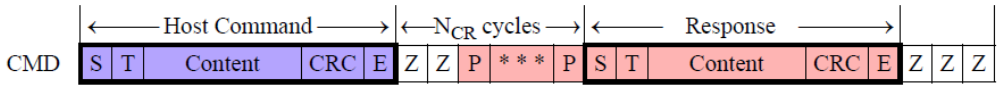


Figure 38 — Command response timing (data transfer mode)

- R1b responses

  Some commands, like CMD6, may assert the BUSY signal and respond with R1. If the busy signal is asserted, it is done two clock cycles after the end bit of the command. The DAT0 line is driven low, DAT1-DAT7 lines are driven by the Device though their values are not relevant.

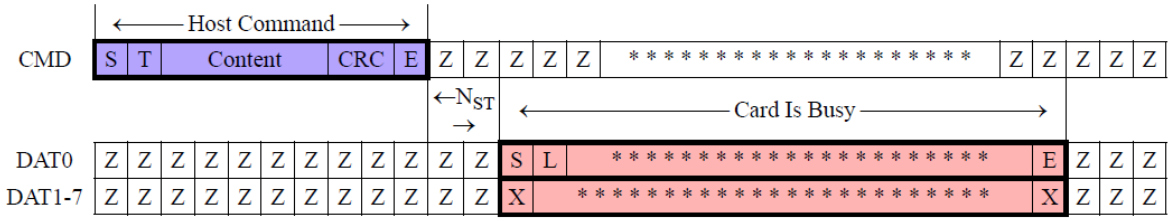


Figure 39 — R1b response timing

- Last Device response—next host command timing

  After receiving the last Device response, the host can start the next command transmission after at least N_{RC} clock cycles. This timing is relevant for any host command.

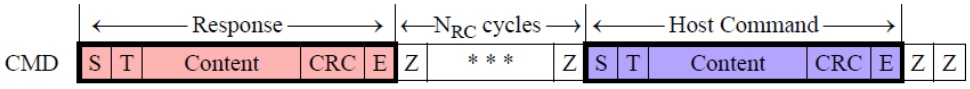


Figure 40 — Timing response end to next command start (data transfer mode)

- Last host command—next host command timing

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

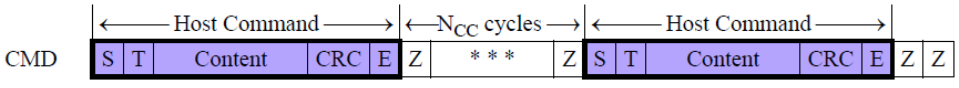


Figure 41 — Timing of command sequences (all modes)

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

6.15.2 Data read

Data read can be made in single data rate mode or in dual rate mode.

In the single data rate mode, data are clocked out by the Device and sampled by the host with the rising edge of the clock and there is a single CRC per data line.

In the dual data rate mode, data are clocked out with both the rising edge of the clock and the falling edge of the clock and there are two CRC appended per data line. In this mode, the block length is always 512 bytes, and bytes come interleaved in either 4-bit or 8-bit width configuration. Bytes with odd number (1,3,5, ... ,511) shall be sampled on the rising edge of the clock by the host and bytes with even number (2,4,6, ... ,512) shall be sampled on the falling edge of the clock by the host. The Device will append two CRC16 per each valid data line, one corresponding to the bits of the 256 odd bytes to be sampled on the rising edge of the clock by the host and the second for the remaining bits of the 256 even bytes of the block to be sampled on the falling edge of the clock by the host.

Remark: only the data block and the two CRC use both edges of the clock. Start and end bits are only valid on the rising edge of the clock. The value on the falling edge is not guaranteed.[2]

- Single block read

  The host selects one Device for data read operation by CMD7, and sets the valid block length for block oriented data transfer by CMD16(CMD16 only applies to single data rate mode). The basic bus timing for a read operation is given in Figure 42. The sequence starts with a single block read command (CMD17) that specifies the start address in the argument field. The response is sent on the CMD line as usual.

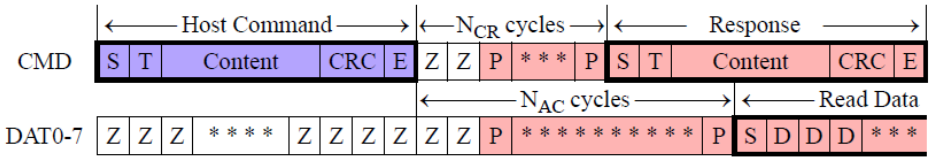


Figure 42 — Single-block read timing

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

[2] Due to ambiguity of previous versions of standard, start and end bits may either be valid for both the rising and falling edge or only for the rising edge. To ensure backward compatibility, the standard allows for both interpretations and does not mandate the value on the falling edge.

6.15.2 Data read (cont'd)

- Multiple block read

  In multiple block read mode, the Device sends a continuous flow of data blocks following the initial host read command. The data flow is terminated by a stop transmission command (CMD12). Figure 43 describes the timing of the data blocks and Figure 44 the response to a stop command. The data transmission stops two clock cycles after the end bit of the stop command.

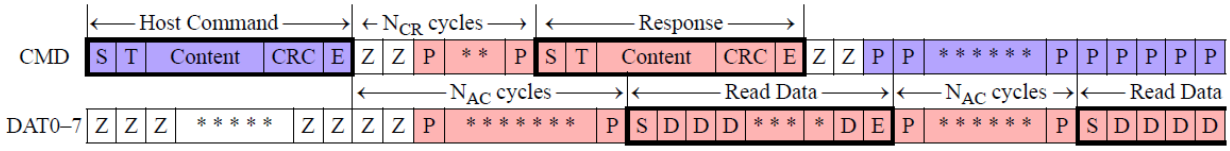


Figure 43 — Multiple-block read timing

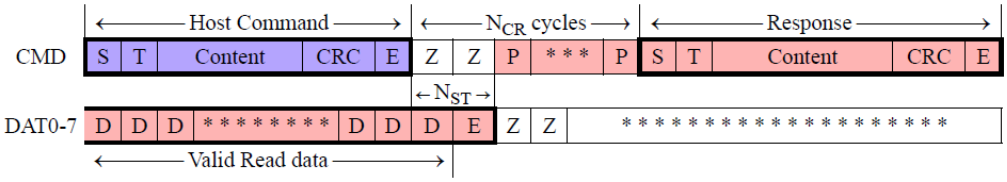


Figure 44 — Stop command timing (CMD12, data transfer mode)

6.15.3 Data write

Data write can be made in single data rate mode or in dual rate mode.

In the single data rate mode, data are clocked out by the host and sampled by the Device with the rising edge of the clock and there is a single CRC per data line. In the dual data rate mode, data are clocked out with both the rising edge of the clock and the falling edge of the clock and there are two CRC appended per data line. In this mode, the block length is always 512 bytes, and bytes come interleaved in either 4-bit or 8-bit width configuration. Bytes with odd number (1,3,5,...,511) shall be sampled on the rising edge of the clock by the Device and bytes with even number (2,4,6,...,512) shall be sampled on the falling edge of the clock by the Device. The host will append two CRC16 per valid data line, one corresponding to bits of the 256 odd bytes to be sampled on the rising edge of the clock by the Device and the second for the remaining bits of the 256 even bytes of the block to be sampled on the falling edge of the clock by the Device.

Remark: In a similar manner to the data read, only the data block and the two CRC use both edges of the clock. Start, end and the 3-bit CRC token status bits are only valid on the rising edge of the clock. The value on the falling edge is not guaranteed.[3]

[3] Due to ambiguity of previous versions of standard, start and end bits may either be valid for both the rising and falling edge or only for the rising edge. To ensure backward compatibility, the standard allows for both interpretations and does not mandate the value on the falling edge

6.15.3Data write (cont'd)

- Single block write

  The host selects the Device for data write operation by CMD7. The host sets the valid block length for block oriented data transfer by CMD16 (CMD16 only applies to single data rate mode).

  The basic bus timing for a write operation is given in Figure 45. The sequence starts with a single block write command (CMD24) that determines (in the argument field) the start address. It is responded by the Device on the CMD line as usual. The data transfer from the host starts N_{WR} clock cycles after the Device response was received.

  The data is suffixed with CRC check bits to allow the Device to check it for transmission errors. The Device sends back the CRC check result as a CRC status token on DAT0. In the case of transmission error, occurring on any of the active data lines, the Device sends a negative CRC status ('101') on DAT0. In the case of successful transmission, over all active data lines, the Device sends a positive CRC status ('010') on DAT0 and starts the data programming procedure.

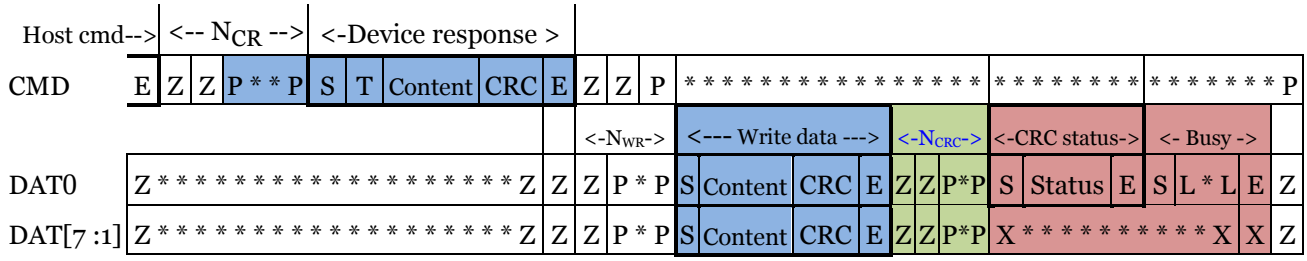


Figure 45 — Block write command timing

N_{CRC}, is a timing parameter defined for HS200 and HS400. HS200 and HS400 device shall support N_{CRC}. N_{CRC} specifies timing from the end bit of data block in a Write operation, till the start bit of the CRC status.

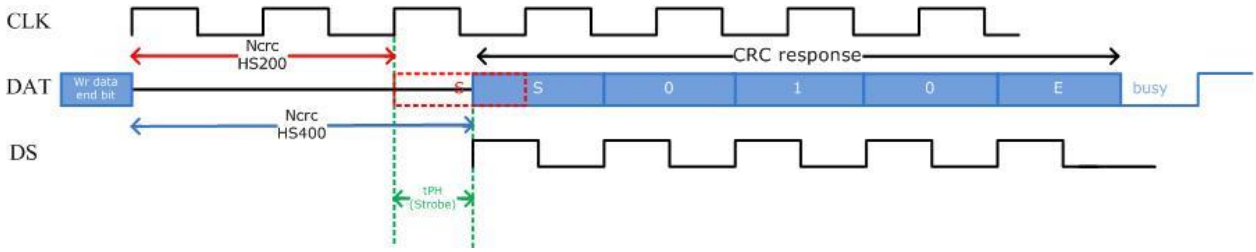


Figure 46 — N_{CRC} timing

For HS400, after CRC Status Response, the Device outputs the BUSY signal on DAT0 after tPH_DATASTROBE + tPERIOD + tRQ from CLK rising edge that is the reference for the End Bit. After this time, the host may start sampling safely the BUSY signal.

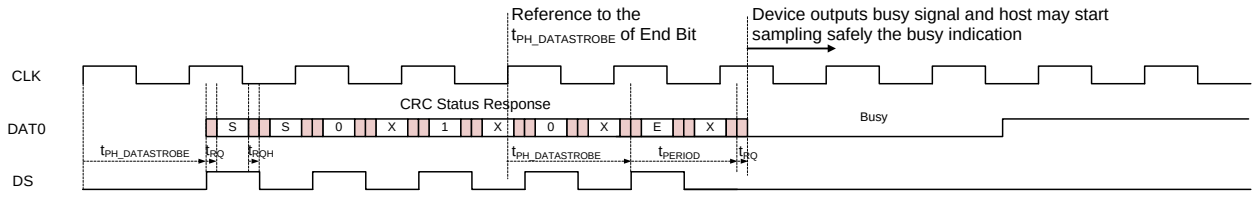


Figure 47 — BUSY Signal after CRC Status Response

6.15.3 Data write (cont'd)

While the Device is programming it indicates busy by pulling down the DAT0 line. This busy status is directly related to Programming state. As soon as the Device completes the programming it stops pulling down the DAT0 line.

- Multiple block write

In multiple block write mode, the Device expects continuous flow of data blocks following the initial host write command. The data flow is terminated by a stop transmission command (CMD12). Figure 48 describes the timing of the data blocks with and without Device busy signal.

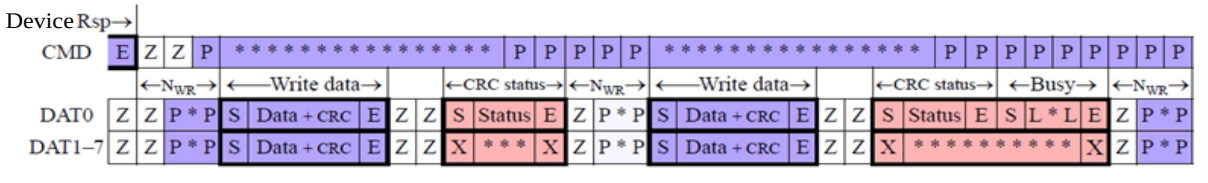


Figure 48 — Multiple-block write timing

The stop transmission command works similar as in the read mode. Figure 49 through Figure 52 describe the timing of the stop command in different Device states.

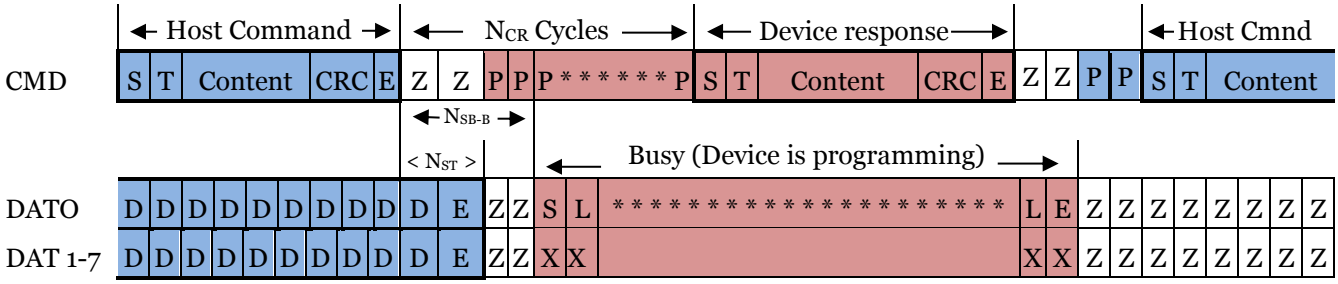
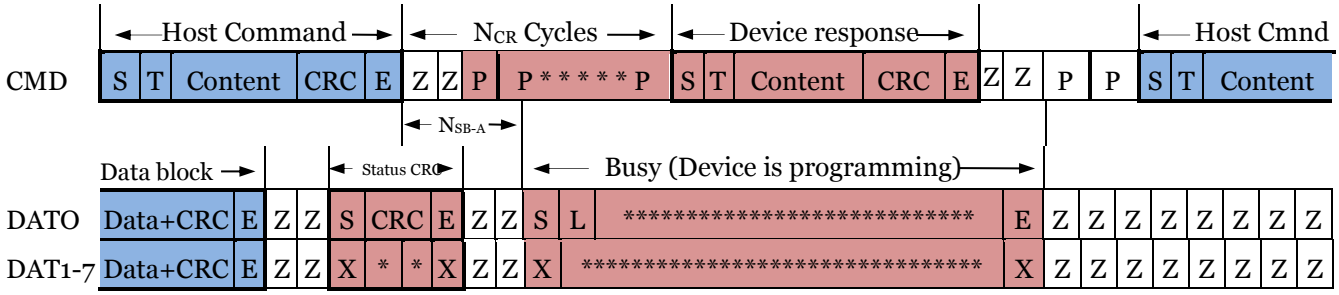


Figure 49 — Stop transmission during data transfer from the host

The Device will treat a data block as successfully received and ready for programming only if the CRC data of the block was validated and the CRC status tokens sent back to the host. 0 is an example of an interrupted (by a host stop command) attempt to transmit the CRC status block. The sequence is identical to all other stop transmission examples. The end bit of the host command is followed, on the data lines, with one more data bit, an end bit and two Z clocks for switching the bus direction (number of Z clocks may vary from 2 to 4 in HS200 and HS400 as described in 6.15.8.1). The received data block, in this case is considered incomplete and will not be programmed.

6.15.3 Data write (cont'd)



NOTE 1 The Device CRC status response is interrupted by the host

Figure 50 — Stop transmission during CRC status transfer from the Device

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

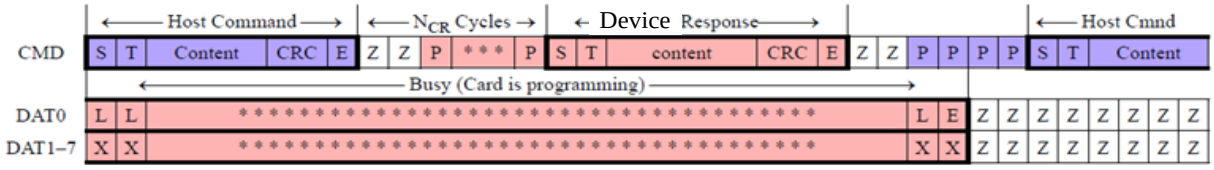


Figure 51 — Stop transmission after last data block; Device is busy programming

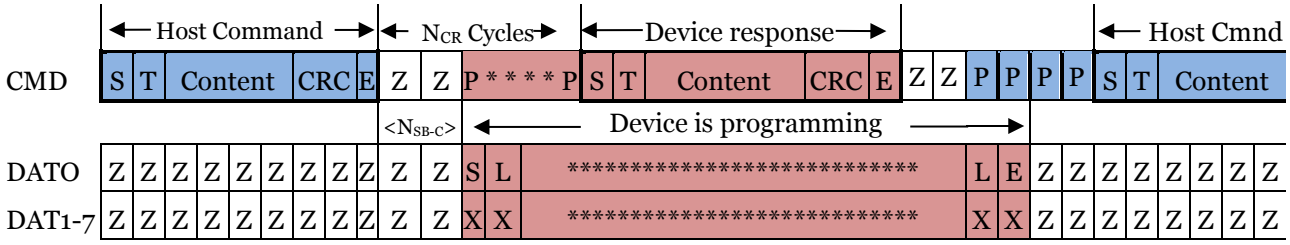


Figure 52 — Stop transmission after last data block; Device becomes busy

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

6.15.3 Data write (cont'd)

In pre-defined multiple block write case the busy signal between the data blocks should be considered as buffer busy signal similar to the open-ended multiple block case. After the Device receives the last data block the following busy indication should be considered as programming busy and being directly related to the Programming state. The meaning of busy signal (from buffer busy to programming busy) changes when the state changes (from rcv to prg). The busy signal remains "low" all the time during the process and is not released by the device between the rcv to prg states. As soon as the Device completes the programming, it stops pulling down the DAT0 line.

- Erase, set, and clear write protect timing

  The host must first select the erase groups to be erased using the erase start and end command (CMD35, CMD36). The erase command (CMD38), once issued, will erase all selected erase groups. Similarly, set and clear write protect commands start a programming operation as well. The Device will signal "busy" (by pulling the DAT0 line low) for the duration of the erase or programming operation. The bus transaction timings are identical to the variation of the stop transmission described in Figure 52.

- Reselecting a busy Device

  When a busy Device that is currently in the dis state is reselected it will reinstate its busy signaling on the data line DAT0. The timing diagram for this command/response/busy transaction is given in Figure 52.

6.15.4 Bus test procedure timing

After reaching the Tran-state in the single data rate mode a host can initiate the Bus Testing procedure. The Bus Testing procedure is invalid in dual data rate mode. If there is no response to the CMD19 sent by the host, the host should read the status from the Device with CMD13. If there was no response to CMD19, the host may assume that this function is not supported by the Device.

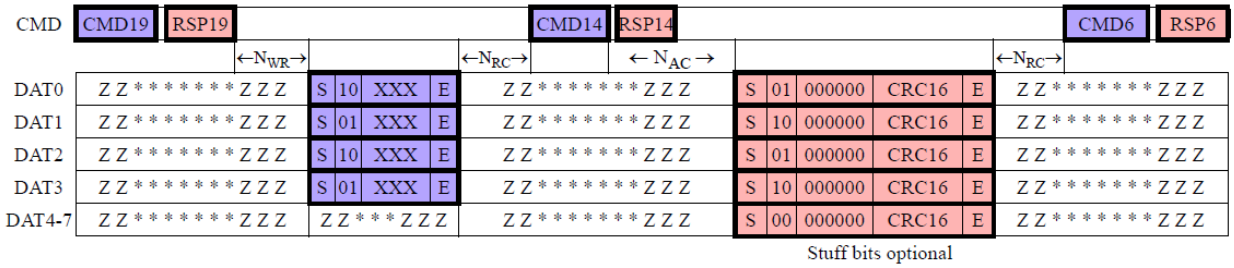


Figure 53 — Bus test procedure timing

6.15.5 Boot operation

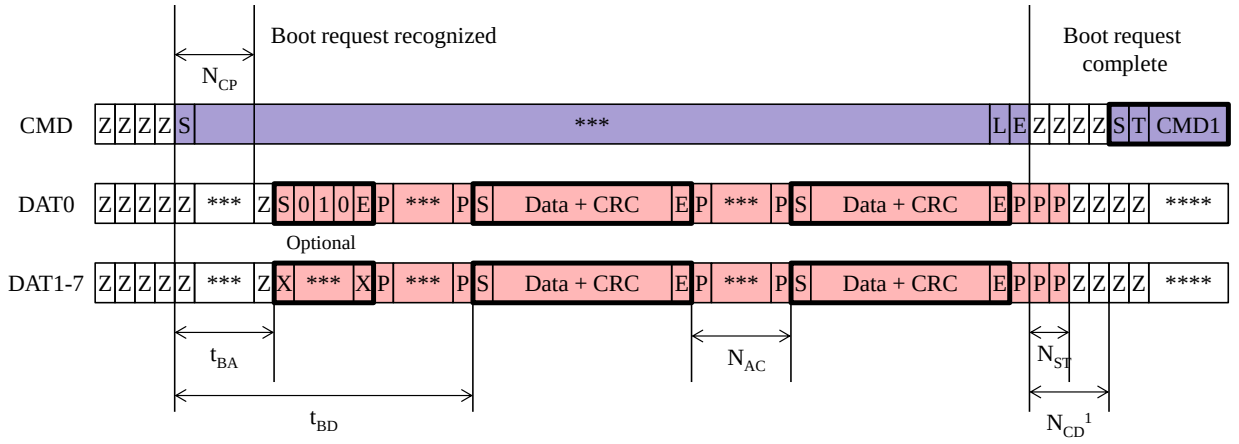
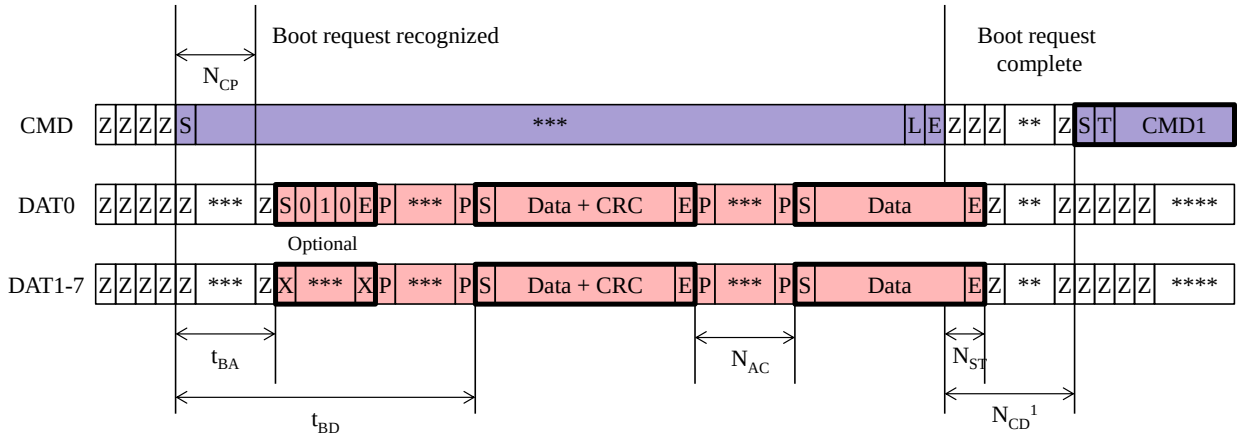


Figure 54 — Boot operation, termination between consecutive data blocks



NOTE 1 Also refer to Figure 56.

Figure 55 − Boot operation, termination during transfer

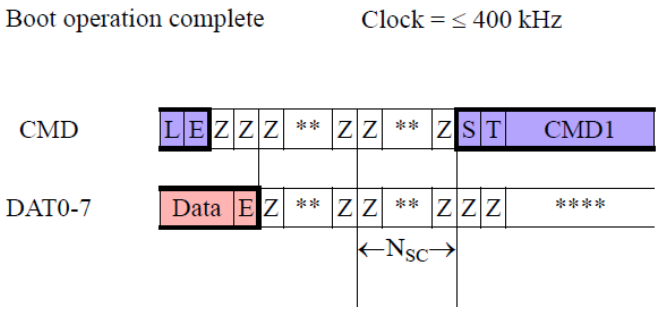
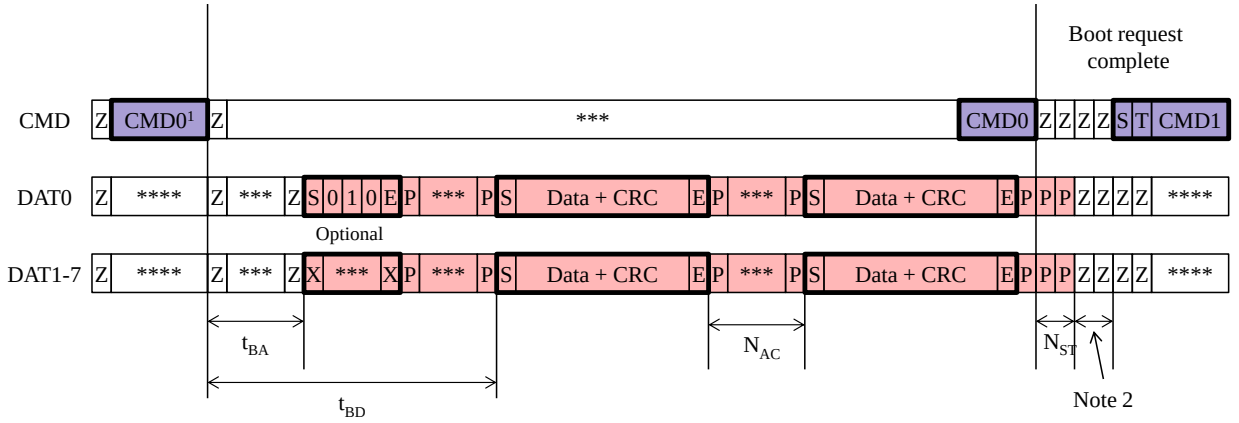


Figure 56 — Bus mode change timing (push-pull to open-drain)

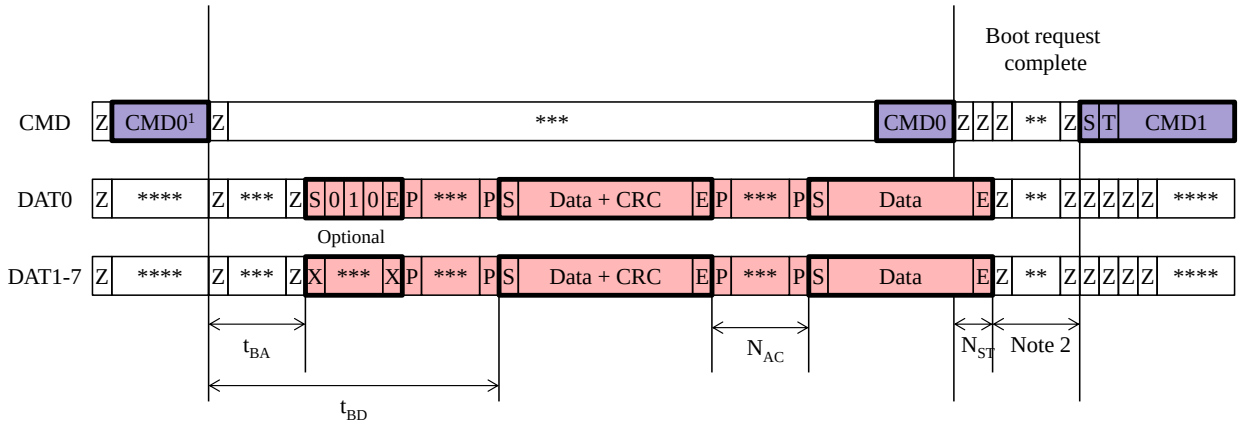
6.15.6 Alternative boot operation



NOTE 1 CMD0 with argument 0xFFFFFFFA.

NOTE 2 Refer to Figure 56.

Figure 57 — Alternative boot operation, termination between consecutive data blocks



NOTE 1 CMD0 with argument 0xFFFFFFFA.

NOTE 2 Refer to Figure 56.

Figure 58 — Alternative boot operation, termination during transfer

6.15.7 Timing Values

Table 71 — Timing Parameters

Symbol	Min	Max	Unit
N_{AC}	2	10 * (TAAC * FOP + 100 * NSAC)[1]	Clock cycles
N_{CC}	8	-	Clock cycles
N_{CD}	56	-	Clock cycles
N_{CP}	74	-	Clock cycles
N_{CR}	2	64	Clock cycles
N_{ID}	5	5	Clock cycles
N_{RC}	8	-	Clock cycles
N_{SC}	8	-	Clock cycles
N_{ST}	2	2	Clock cycles
N_{WR}	2	-	Clock cycles
$\mathbf{N}_{SB\text{-}A}$**(2)**	4	4	Clock cycles
$\mathbf{N}_{SB\text{-}B}$**(2)**	4	4	Clock cycles
$\mathbf{N}_{SB\text{-}C}$**(2)**	2	2	Clock cycles
t_{BA}	-	50	ms
t_{BD}	-	1	s

NOTE 1
FOP is the MMC clock frequency the host is using for the read operation.
Following is a calculation example:
CSD value for TAAC is 0x26; this is equal to 1.5mSec;
CSD value for NSAC is 0;
The host frequency FOP is 10MHz
NAC = 10 × (1.5×10-3 × 10×106 + 0) = 150,000 clock cycles

NOTE 2
NSB-A: Device is driving DAT0 (e.g., CRC response) – no need in direction change (see Figure 50)
NSB-B: Host is transmitting DAT0 (e.g., during write data transfer) – need to allow direction change (see Figure 49)
NSB-C: DAT0 in Tri State – no need in direction change (see Figure 51)

6.15.8 Timing changes in HS200 and HS400 mode

6.15.8.1 Timing values

Table 72 describes the differences and additions in timing parameters compared to Table 71.

The difference regarding N_{ST} and N_{SB} are caused by the phase variation between CLK and DAT that is defined as t_{PH} (0 - 2 UI) in HS200. The phase variation between CLK and DataStrobe is defined as $t_{PH\_DATASTROBE}$ (0 - 4 UI) in HS400.

Table 72 — Timing Parameters for HS200 and HS400 mode

Parameter	Min.	Max.	Unit
N_{AC}	8	-	clock cycles
N_{CRC}		(See note.1)	
N_{ST}	2	4	clock cycles
N_{SB-A}	4	6	clock cycles
N_{SB-B}	4	6	
N_{SB-C}	2	4	
NOTE 1 N_{CRC} shall be between 2 to 8 clock cycles in HS200 and (2 x tPeriod) to (8 x tPeriod + tPH_dataStrobe) in HS400.			

6.15.8.2 Read Block Gap

The host may suspend read data transfer from the device by stopping CLK, and resume data transfer by resuming CLK. In HS200 and HS400, the t_{PH} variation requires that DAT[7:0] will be considered asynchronous to CLK. As a result, it takes a few clock cycles to stop CLK by detecting the end bit of a data block, because synchronization is required.

Therefore, the host shall stop CLK between data blocks rather than within a data block. The minimum gap between two consecutive blocks (N_{AC}) is defined as 8, leaving room for the host to stop CLK before the device starts to transfer the next block of data.

Figure 59 describes an example of stopping CLK at the Read Block Gap. In that case t_{PH} is more than 1UI. Clock position 0 is a trigger point, which outputs the end bit of data block. The device starts to count N_{AC} based on CLK cycles from clock position 0. The number indicated above CLK is the counter value. The device shall wait until at least clock 9 to output the next data block. Then host needs to stop CLK before clock 9. N_{AC}(min.)=8 provides enough timing for the host to stop CLK.

NOTE The host is advised to take a special care when command or response is communicated at the time when the host stops the clock in a Read Block Gap. Due to the asynchronism of the lines, stopping the clock while command or response are displayed may result in an uncertain behavior

6.15.8.2 Read Block Gap (cont'd)

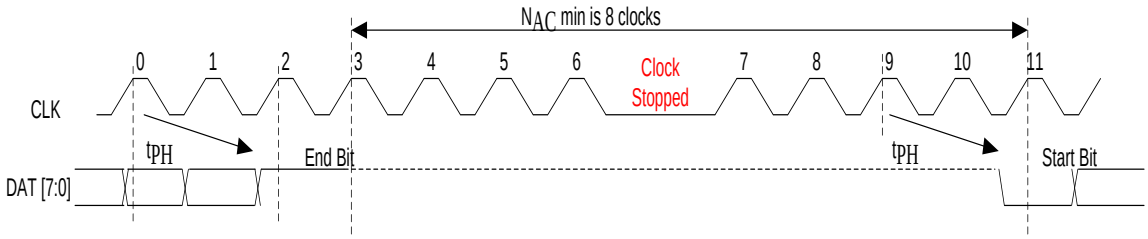


Figure 59 — Clock Stop Timing at Block Gap in Read Operation

Implementation Guide:

If the host stops the clock during read data block transfer, it should recognize the number of correctly-transferred data bits, assuming that sampling timing has drifted during the clock suspension.

6.15.8.3 CMD12 Timing Modification in Write Operation

In HS200 and HS400 mode, the output delay from the device to the host may vary more than 1 clock cycle. The host needs to take special care about the relations between CMD and Data. This applies specifically to the relations between CMD12 (STOP_TRANSMISSION) and the Data CRC Status response.

In order to assure that the last data block in a Multiple Block Write operation is written successfully, the host shall not interrupt the data CRC Status response sent by the device. It requires that the host will output the end bit of CMD12 after it has received the end bit of the data CRC response from the device.

In case where the device detects the end bit of CMD12 prior to outputting the end bit of the CRC status response, the last data block is not assured to be written to the nonvolatile memory of the device.

Figure 60 describes border timing that assures write block is written successfully. Host needs to output the end bit of CMD12 after host receive the end bit of CRC Status.

Adjusting end bit of CMD12 to CRC status requires specific hardware. Alternatively, the host can take a simpler approach, and start issuing CMD12 after receiving the CRC status of the last data block.

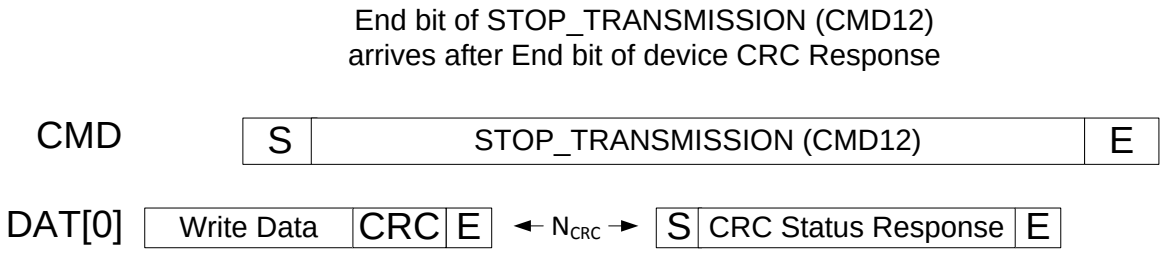


Figure 60 — Border Timing of CMD12 in Write Operation

6.15.8.4 CMD12 Timing Modification in Read Operation

Figure 61 describes CMD12 border timing in read operation. The minimum Read Block Gap length N_{AC} is 8 clocks. In order to allow the device to output the entire last read data block successfully, the end bit of CMD12 (clock #1 in the figure) shall not arrive earlier than 2 clocks before the end bit of the last data block (clock #3). If CMD12 is issued earlier than this timing, the last read data block may be misinterpreted (for example, end bit of data block is not indicated).

To avoid transferring of the following data block (after the last block the host intends to read), or OUT_OF_RANGE error (in case the last block the host reads is near the last logical block of the partition), the end bit of CMD12 shall arrive no later than 3 clocks before the end of N_{AC} clocks after the end of the last data block read by the host. If the end bit of CMD12 arrives after that clock cycle, the device may start transferring the next data block.

Adjusting end bit of CMD12 to read data block requires specific hardware. Host may take another method to stop multiple-block read operation, for example issue CMD12 after receiving the last data block. By this method, next data block may start to output and aborted by CMD12. The last block read indicates out of range error.

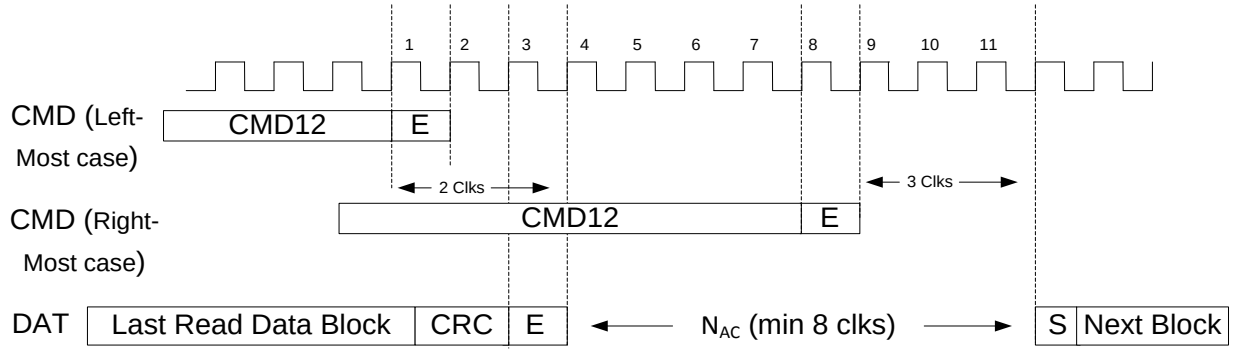


Figure 61 — Border Timing of CMD12 in Read Operation

6.15.8.5 R1b Timing

"Busy" signaling (by pulling DAT0 line low) during an R1b command execution adheres to the same bus timing as described in Figure 39. In all legacy modes other than HS200 or HS400 mode, R1b busy indication starts 2 clocks after the end bit of the command. In HS200 and HS400 mode, R1b busy indication starts 2 to 4 clocks after the end bit of the command.

6.15.8.6 Reselecting a Busy Device

When a busy device that is currently in the disconnect state is reselected it will reinstate its busy signaling on DAT0. The bus timing of reselecting a device is the same as shown in Figure 51.

In non HS200 or HS400 mode, the selected device starts indicating busy 2 clocks after the end bit of CMD7. In HS200 & HS400 mode, the selected device starts indicating busy 2 to 4 clocks after the end bit of CMD7.

6.15.9 Enhanced Strobe in HS400 Mode

Support of Enhanced Strobe mode is indicated by STROBE_SUPPORT[184] register of EXT_CSD. The following explanation relates to devices that supports Enhanced Strobe mode.

In HS400 a STROBE signal is provided by the device to the host. The STROBE signal is provided only in a limited time periods. The time periods that STROBE signal is provided while in HS400 mode are defined by Enhanced Strobe bit in BUS_WIDTH [183] register of EXT_CSD.

While Enhanced Strobe bit is set to “0” the STROBE is provided during DATA Out and CRC Response time periods. Data Out and CRC response are synched to the STROBE clock signals (as described in Figure 62).

While Enhanced Strobe bit is set to “1” the STROBE is provided during DATA Out, CRC Response and also during CMD Response.

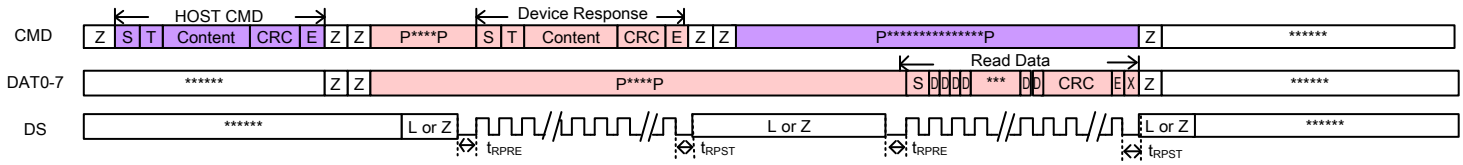


Figure 62 — Enhanced Strobe signals for CMD Response and Data Out (Read operation)

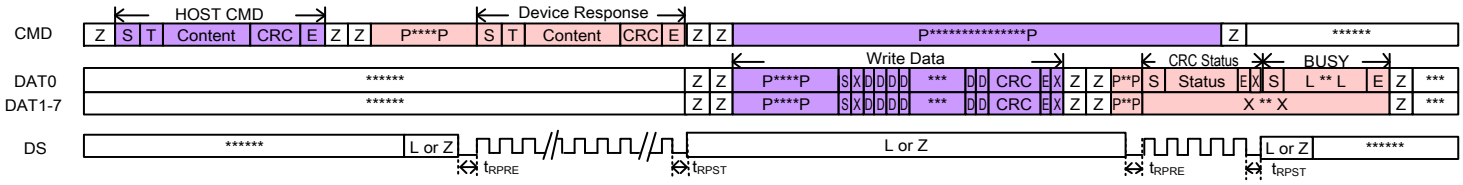


Figure 63 — Enhanced Strobe signals for CMD Response and CRC Response (Write operation)

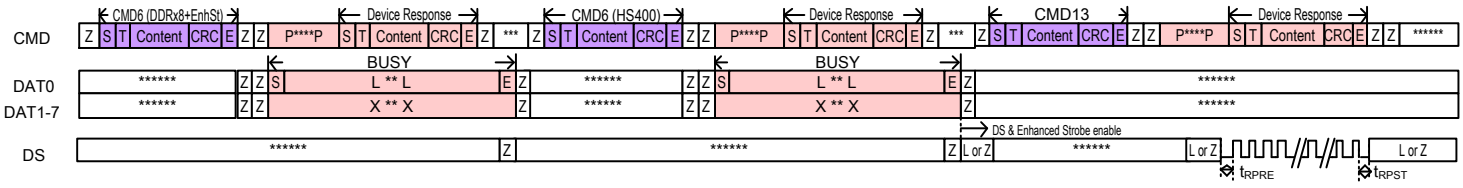
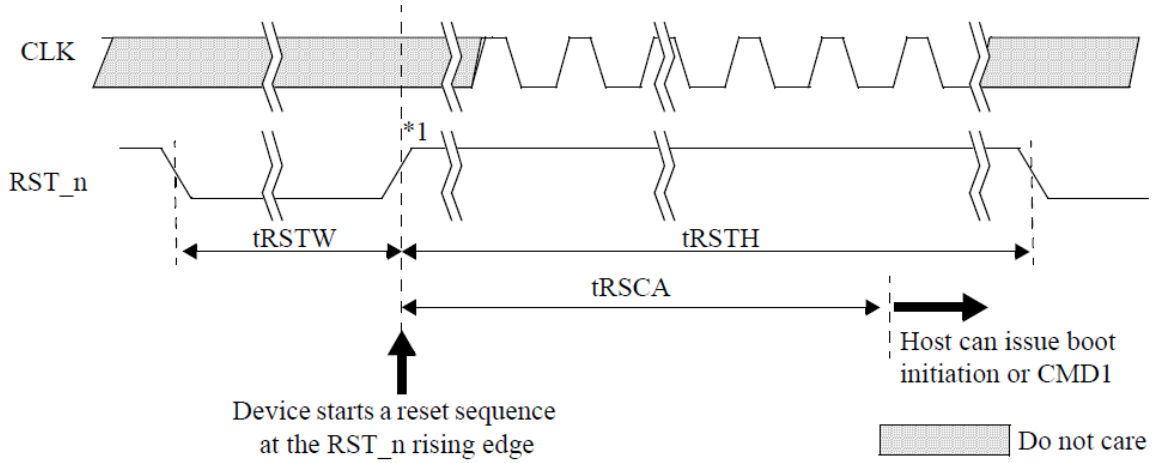


Figure 64 — HS400 mode change with Enhanced Strobe

In Enhanced Strobe mode DATA OUT, CRC Response and CMD Response are all synched to STROBE clocks. The timing relation between CMD Response output signals and STROBE clocks is the same as defined for DATA Out to STROBE clocks. Refer to HS400 timing diagrams (Figure 62).

For detailed t_{RPRE} and t_{RPST} timing refer to Table 218 in 10.10.3.

6.15.10 H/W Reset Operation



NOTE 1 Device will detect the rising edge of RST_n signal to trigger internal reset sequence.

Figure 65 — H/W reset waveform

Table 73 — H/W reset timing parameters

Symbol	Comment	Min	Max	Unit
tRSTW	RST_n pulse width	1		[us]
tRSCA	RST_n to Command time	200[1]		[us]
tRSTH	RST_n high period (interval time)	1		[us]
NOTE 1 74 cycles of clock signal required before issuing CMD1 or CMD0 with argument 0xFFFFFFFA.				

6.15.11 Noise filtering timing for H/W Reset

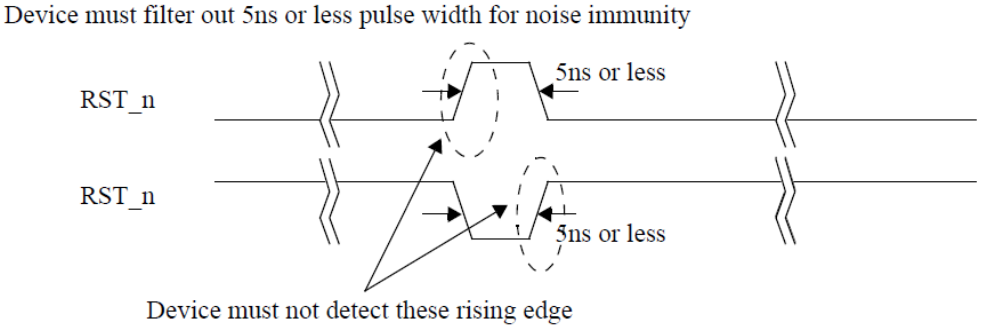


Figure 66 — Noise filtering timing for H/W reset

Device must not detect 5ns or less of positive or negative RST_n pulse. Device must detect more than or equal to 1us of positive or negative RST_n pulse width.

6.15.12 Additional Timing changes in HS400 mode

6.15.12.1 Write Timing

In HS400 mode, default Start Bit position is valid at rising edge. In addition, the host may stop CLK between data blocks or within a data block. The Data Strobe is used only in read operation. The Data Strobe is always High-Z (not driven by the device and pulled down by R_{DS}) or Driven Low in write operation, except during CRC status response.

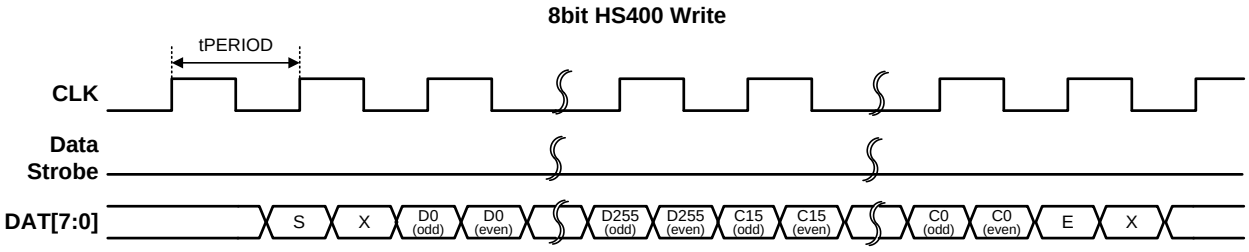


Figure 67 — HS400 Write Timing with data block size of 512 bytes

6.15.12.2 Read Timing

The Data Strobe is toggled only during data valid period (Start bit, Data, CRC16, End bit, CRC status token). In HS400 mode, Start Bit position is valid at both rising and falling edge. In addition, the host may stop CLK between data blocks and not within a data block.

Data Strobe shall be driven t_{RPRE} prior to the first Data Strobe rising edge and t_{RPST} after the last falling edge.

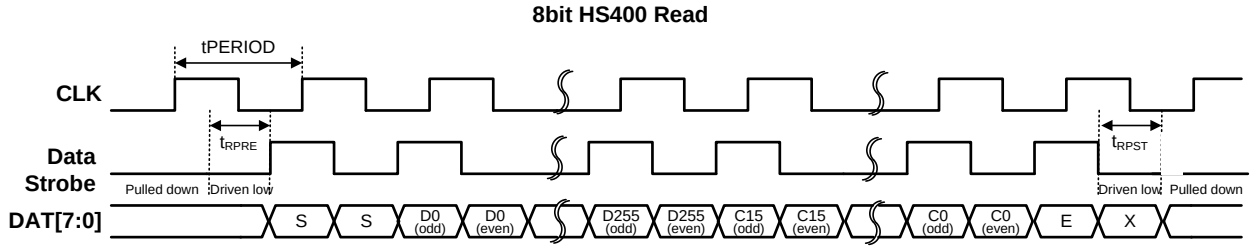


Figure 68 — HS400 Read Timing with data block size of 512 bytes

7 Device Registers

Within the Device interface seven registers are defined: OCR, CID, CSD, EXT_CSD, RCA, DSR and QSR. These can be accessed only by corresponding commands (see 6.10). The OCR, CID, CSD and QSR registers carry the Device/content specific information, while the RCA and DSR registers are configuration registers storing actual configuration parameters. The EXT_CSD register carries both, Device specific information and actual configuration parameters.

7.1 OCR register

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

Table 74 — OCR register definitions

OCR bit	VCCQ voltage window[2]	High Voltage MultimediaCard	Dual voltage MultimediaCard and *e*•MMC
[6:0]	Reserved	000 0000b	000 0000b
[7]	1.70 – 1.95V	0b	1b
[14:8]	2.0 – 2.6V	000 0000b	000 0000b
[23:15]	2.7 – 3.6V	1 1111 1111b	1 1111 1111b
[28:24]	Reserved	0 0000b	0 0000b
[30:29]	Access Mode	00b (byte mode) 10b (sector mode)	00b (byte mode) 10b (sector mode)
[31]	(Card power up status bit (busy))[1]		

NOTE 1 This bit is set to LOW if the Device has not finished the power up routine.

NOTE 2 VDD voltage window in case of High Voltage MultimediaCard.

The supported voltage range is coded as shown in Table 74 for *e*•MMC devices. As long as the Device is busy, the corresponding bit (31) is set to LOW, the ‘wired-and’ operation, described in 6.4.4 yields LOW, if at least one Device is still busy.

7.2 CID register

The Device IDentification (CID) register is 128 bits wide. It contains the Device identification information used during the Device identification phase (*e*•MMC protocol). Every individual flash or I/O Device shall have an unique identification number. Every type of *e*•MMC Device shall have a unique identification number. Table 75 lists these identifiers. The structure of the CID register is defined in this section.

Table 75 — CID Fields

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

7.2.1 MID [127:120]

MID is an 8 bit binary number that identifies the device manufacturer. The MID number is controlled, defined and allocated to an *e*•MMC manufacturer by JEDEC. This procedure is established to ensure uniqueness of the CID register.

7.2.2 CBX [113:112]

CBX indicates the device type.

Table 76 — Device Types

[113:112]	Type
00	Device (removable)
01	BGA (Discrete embedded)
10	POP
11	Reserved

7.2.3 OID [111:104]

OID is an 8-bit binary number that identifies the Device OEM and/or the Device contents (when used as a distribution media either on ROM or FLASH Devices). The OID number is controlled, defined and allocated to an *e*•MMC manufacturer by JEDEC. This procedure is established to ensure uniqueness of the CID register.

7.2.4 PNM [103:56]

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

7.2.5 PRV [55:48]

The product revision, PRV, is composed of two Binary Coded Decimal (BCD) digits, four bits each, representing an “n.m” revision number. The “n” is the most significant nibble and “m” is the least significant nibble. As an example, the PRV binary value field for product revision “6.2” will be: 0110 0010.

7.2.6 PSN [47:16]

PSN is a 32-bit unsigned binary integer.

7.2.7 MDT [15:8]

The manufacturing date, MDT, is composed of two hexadecimal digits, four bits each, representing a two digits date code m/y; The "m" field, most significant nibble, is the month code. 1 = January. The "y" field, least significant nibble, is the year code. 0 = 1997. As an example, the binary value of the MDT field for production date "April 2000" will be: 0100 0011

For *e*•MMC 4.41 and later devices, indicated by a value larger than 4 in EXT_CSD_REV [192], the 4-bit "y" field shall roll over after 2012, so that y=0 shall be used for 2013. See Table 77 for a list of valid y values for specific *e*•MMC versions.

Table 77 — Valid MDT "y" Field Values

MDT y Field [11:8]	Year
0000	1997, or 2013 if EXT_CSD_REV [192] > 4
0001	1998, or 2014 if EXT_CSD_REV [192] > 4
0010	1999, or 2015if EXT_CSD_REV [192] > 4
0011	2000, or 2016 if EXT_CSD_REV [192] > 4
0100	2001, or 2017 if EXT_CSD_REV [192] > 4
0101	2002, or 2018 if EXT_CSD_REV [192] > 4
0110	2003, or 2019 if EXT_CSD_REV [192] > 4
0111	2004, or 2020 if EXT_CSD_REV [192] > 4
1000	2005, or 2021 if EXT_CSD_REV [192] > 4
1001	2006, or 2022 if EXT_CSD_REV [192] > 4
1010	2007, or 2023 if EXT_CSD_REV [192] > 4
1011	2008, or 2024 if EXT_CSD_REV [192] > 4
1100	2009, or 2025 if EXT_CSD_REV [192] > 4
1101	2010 only
1110	2011 only
1111	2012 only

7.2.8 CRC [7:1]

The CRC7 checksum (7 bits). This is the checksum of the CID contents computed according to 0.

7.3 CSD register

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

R: Read only.
W: One time programmable and not readable.
R/W: One time programmable and readable.
W/E: Multiple writable with value kept after power failure, H/W reset assertion and any CMD0 reset and not readable.
R/W/E: Multiple writable with value kept after power failure, H/W reset assertion and any CMD0 reset and readable.
R/W/C_P: Writable after value cleared by power failure and HW/rest assertion (the value not cleared by CMD0 reset) and readable.
R/W/E_P: Multiple writable with value reset after power failure, H/W reset assertion and any CMD0 reset and readable.
W/E_P: Multiple writable with value reset after power failure, H/W reset assertion and any CMD0 reset and not readable.

7.3 CSD Register (cont'd)

Table 78 — CSD Fields

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

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

7.3 CSD Register (cont'd)

7.3.1 CSD_STRUCTURE [127:126]

CSD_STRUCTURE describes the version of the CSD structure.

Table 79 — CSD register structure

CSD_STRUCTURE	CSD Structure version	Valid for System Specification Version
0	CSD version No. 1.0	Allocated by MMCA
1	CSD version No. 1.1	Allocated by MMCA
2	CSD version No. 1.2	Version 4.1–4.2–4.3-4.41-4.5-4.51-5.0-5.01-5.1
3	Version is coded in the CSD_STRUCTURE byte in the EXT_CSD register	

7.3.2 SPEC_VERS [125:122]

SPEC_VERS defines the *e*•MMC System Specification version supported by the Device.

Table 80 — System specification version

SPEC_VERS	System Specification Version
0	Allocated by MMCA
1	Allocated by MMCA
2	Allocated by MMCA
3	Allocated by MMCA
4	Version 4.1–4.2–4.3-4.4-4.41-4.5-4.51-5.0-5.01-5.1
5–15	Reserved

7.3.3 TAAC [119:112]

TAAC defines the asynchronous part of the data access time.

Table 81 — TAAC access-time definition

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

7.3.4 NSAC [111:104]

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

The total access time N_{AC} as expressed in Table 71 is calculated based on TAAC and NSAC. It has to be computed by the host for the actual clock rate. The read access time should be interpreted as a typical delay for the first data bit of a data block or stream.

7.3 CSD Register (cont'd)

7.3.5 TRAN_SPEED [103:96]

Table 82 defines the clock frequency when not in high speed mode. For Devices supporting versions v4.0, v4.1, and v4.2 of the standard, the value shall be 20 MHz (0x2A). For Devices supporting v4.3, the value shall be 26 MHz (0x32).

Table 82 — Maximum bus clock frequency definition

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

7.3.6 CCC [95:84]

The *e*•MMC command set is divided into subsets (command classes). The Device command class register CCC defines command classes that are supported by this Device. A value of '1' in a CCC bit means that the corresponding command class is supported. For command class definition refer to Table 83.

Table 83 — Supported Device command classes

CCC Bit	Supported Device Command Class
0	class 0
1	class 1
…	
11	class 11

7.3.7 READ_BL_LEN [83:80]

The data block length is computed as $2^{READ\_BL\_LEN}$. The block length might therefore be in the range 1B, 2B, 4B...16kB. (See 6.13 for details.)

NOTE The support for 512B read access is mandatory for all Devices. And that the Devices has to be in 512B block length mode by default after power-on, or software reset.

The purpose of this register is to indicate the supported maximum read data block length:

Table 84 — Data block length

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

7.3 CSD Register (cont'd)

7.3.8 READ_BL_PARTIAL [79]

READ_BL_PARTIAL defines whether partial block sizes can be used in block read commands.

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

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

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

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

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

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

7.3.9 WRITE_BLK_MISALIGN [78]

Defines if the data block to be written by one command can be spread over more than one physical block of the memory device. The size of the memory block is defined in WRITE_BL_LEN.
WRITE_BLK_MISALIGN=0 signals that crossing physical block boundaries is invalid.
WRITE_BLK_MISALIGN=1 signals that crossing physical block boundaries is allowed.

7.3.10 READ_BLK_MISALIGN [77]

Defines if the data block to be read by one command can be spread over more than one physical block of the memory device. The size of the memory block is defined in READ_BL_LEN.
READ_BLK_MISALIGN=0 signals that crossing physical block boundaries is invalid.
READ_BLK_MISALIGN=1 signals that crossing physical block boundaries is allowed.

7.3.11 DSR_IMP [76]

DSR_IMP defines if the configurable driver stage is integrated on the Device. If set, a driver stage register (DSR) must be implemented also. (See 7.6.)

Table 85 — DSR implementation code table

CCC Bit	Supported Device Command Class
0	DSR is not implemented
1	DSR is implemented

7.3 CSD Register (cont'd)

7.3.12 C_SIZE [73:62]

The C_SIZE parameter is used to compute the device capacity for devices up to 2 GB of density. See 7.4.52, SEC_COUNT [215:212] , for details on calculating densities greater than 2 GB. When the device density is greater than 2GB, the maximum possible value should be set to this register (0xFFF). This parameter is used to compute the device capacity.

The memory capacity of the device is computed from the entries C_SIZE, C_SIZE_MULT and READ_BL_LEN as follows:

Memory capacity = BLOCKNR * BLOCK_LEN where BLOCKNR = (C_SIZE+1) * MULT

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

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

Therefore, the maximal capacity that can be coded is 4096*512*2048 = 4 GBytes.

Example: A 4 MByte device with BLOCK_LEN = 512 can be coded by C_SIZE_MULT = 0 and C_SIZE = 2047. When the partition configuration is executed by host, device will re-calculate the C_SIZE value that can indicate the size of user data area after the partition.

7.3.13 VDD_R_CURR_MIN [61:59] and VDD_W_CURR_MIN [55:53]

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

Table 86 — V_{DD} (min) current consumption

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

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

7.3.14 VDD_R_CURR_MAX [58:56] and VDD_W_CURR_MAX [52:50]

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

Table 87 — V_{DD} (max) current consumption

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

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

7.3 CSD Register (cont'd)

7.3.15 C_SIZE_MULT [49:47]

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

If the density of the device is higher than 2GB, the maximum possible value should be set to this register (i.e., 0x7).

Table 88 — Multiplier factor for device size

C_SIZE_MULT	MULT	Remarks
0	$2^2 = 4$	
1	$2^3 = 8$	
2	$2^4 = 16$	
3	$2^5 = 32$	
4	$2^6 = 64$	
5	$2^7 = 128$	
6	$2^8 = 256$	
7	$2^9 = 512$	

7.3.16 ERASE_GRP_SIZE [46:42]

The content of this register is a 5 bit binary coded value, used to calculate the size of the erasable unit of the Device. The size of the erase unit (also referred to as erase group) is determined by the ERASE_GRP_SIZE and the ERASE_GRP_MULT entries of the CSD, using the following equation: size of erasable unit = (ERASE_GRP_SIZE + 1) * (ERASE_GRP_MULT + 1) This size is given as minimum number of write blocks that can be erased in a single erase command.

7.3.17 ERASE_GRP_MULT [41:37]

ERASE_GRP_MULT is a 5 bit binary coded value used for calculating the size of the erasable unit of the Device. See 7.3.16 for detailed description.

7.3.18 WP_GRP_SIZE [36:32]

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

7.3.19 WP_GRP_ENABLE [31]

WP_GRP_ENABLE indicates whether write protection of regions is possible. A value of '0' means no group write protection possible.

7.3.20 DEFAULT_ECC [30:29]

DEFAULT_ECC is set by the Device manufacturer. It defines the ECC code that is recommended for use. The field definition is the same as for the ECC field described later.

7.3 CSD Register (cont'd)

7.3.21 R2W_FACTOR [28:26]

R2W_FACTOR defines the typical block program time as a multiple of the read access time. Table 89 defines the field format.

Table 89 — R2W_FACTOR

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

7.3.22 WRITE_BL_LEN [25:22]

WRITE_BL_LEN defines the block length for write operations. See READ_BL_LEN for field coding. Support for 512 B write access is mandatory for all devices and the device must be in 512B block length mode by default after power-on, or software reset. The purpose of this register is to indicate the supported maximum write data block length.

7.3.23 WRITE_BL_PARTIAL[21]

WRITE_BL_PARTIAL defines whether partial block sizes can be used in block write commands.

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

WRITE_BL_PARTIAL='0' means that only the 512B and the WRITE_BL_LEN block size can be used for block oriented data write.

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

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

WRITE_BL_PARTIAL='0' means that only the 512B and the WRITE_BL_LEN block size can be used for block oriented data write.

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

7.3.24 CONTENT_PROT_APP [16]

This field in the CSD indicates whether the content protection application is supported. *e•*MMC devices that implement the content protection application will have this bit set to '1';

7.3.25 FILE_FORMAT_GRP [15]

FILE_FORMAT_GRP indicates the selected group of file formats. This field is read-only for ROM. The usage of this field is shown in Table 90. (See FILE_FORMAT)

7.3.26 COPY [14]

COPY defines if the contents is original (= '0') or has been copied (='1'). The COPY bit for OTP and MTP devices, sold to end consumers, is set to '1' that identifies the device contents as a copy. The COPY bit is a one time programmable bit.

7.3 CSD Register (cont'd)

7.3.27 PERM_WRITE_PROTECT [13]

This register permanently protects the whole device (boot, RPMB and all user area partitions) content against overwriting or erasing (all data write and erase commands for the device are permanently disabled). The default value is '0', i.e., not permanently write protected.

Setting permanent write protection for the entire Device will take precedence over any other write protection mechanism currently enabled on the Device. The ability to permanently protect the Device by setting PERM_WRITE_PROTECT(CSD[13]) can be disabled by setting CD_PERM_WP_DIS (EXT_CSD[171] bit 6). If CD_PERM_WP_DIS is set and the master attempts to set PERM_WRITE_PROTECT(CSD[13]) the operation will fail and the ERROR (bit 19) error bit will be set in the status register.

7.3.28 TMP_WRITE_PROTECT [12]

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

Temporary write protection only applies to the write protection groups on the Device where another write protection mechanism (Password, Permanent or Power-On) has not already been enabled.

When SECURE_WP_MASK is set user area is updatable regardless of TMP_WRITE_PROTECT[12].

7.3.29 FILE_FORMAT [11:10]

FILE_FORMAT indicates the file format on the Device. This field is read-only for ROM. The following formats are defined:

Table 90 — File formats

FILE_FORMAT_GRP	FILE_FORMAT	Remarks
0	0	Hard disk-like file system with partition table
0	1	DOS FAT (floppy-like) with boot sector only (no partition table)
0	2	Universal File Format
0	3	Others / Unknown
1	0, 1, 2, 3	Reserved

7.3.30 ECC [9:8]

ECC defines the ECC code that was used for storing data on the Device. This field is used by the host (or application) to decode the user data. Table 91 defines the field format.

Table 91 — ECC type

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

7.3 CSD Register (cont'd)

7.3.31 CRC [7:1]

The CRC field carries the check sum for the CSD contents. It is computed according to 0. The checksum has to be recalculated by the host for any CSD modification. The default corresponds to the initial CSD contents. Table 92 lists the correspondence between the CSD entries and the command classes. A '+' entry indicates that the CSD field affects the commands of the related command class.

Table 92 — CSD field command classes

CSD Field	Command Classes										
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
CSD_STRUCTURE	+	+	+	+	+	+	+	+	+	+	+
SPEC_VERS	+	+	+	+	+	+	+	+	+	+	+
TAAC		+	+	+	+	+	+	+	+		+
NSAC		+	+	+	+	+	+	+	+		+
TRAN_SPEED		+	+	+	+						+
CCC	+	+	+	+	+	+	+	+	+	+	+
READ_BL_LEN			+								+
READ_BL_PARTIAL			+								+
WRITE_BLK_MISALIGN					+						+
READ_BLK_MISALIGN			+								+
DSR_IMP	+	+	+	+	+	+	+	+	+	+	+
C_SIZE_MANT		+	+	+	+	+	+	+	+		+
C_SIZE_EXP		+	+	+	+	+	+	+	+		+
VDD_R_CURR_MIN		+	+								+
VDD_R_CURR_MAX + +		+	+								+
VDD_W_CURR_MIN				+	+	+	+	+	+		+
VDD_W_CURR_MAX				+	+	+	+	+	+		+
ERASE_GRP_SIZE						+	+	+	+		
WP_GRP_SIZE							+	+	+		
WP_GRP_ENABLE							+	+	+		
DEFAULT_ECC		+	+	+	+	+	+	+	+		+
R2W_FACTOR				+	+	+	+	+	+		
WRITE_BL_LEN				+	+	+	+	+	+		+
WRITE_BL_PARTIAL				+	+	+	+	+	+		+
FILE_FORMAT_GRP											
COPY	+	+	+	+	+	+	+	+	+	+	+
PERM_WRITE_PROTECT	+	+	+	+	+	+	+	+	+	+	+
TMP_WRITE_PROTECT	+	+	+	+	+	+	+	+	+	+	+
FILE_FORMAT											
ECC		+	+	+	+	+	+	+	+		+
CRC	+	+	+	+	+	+	+	+	+	+	+

7.4 Extended CSD register

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

Multi bytes field is interpreted in little endian byte order.

R: Read only.

W: One time programmable and not readable.

R/W: One time programmable and readable.

W/E: Multiple writable with value kept after power failure, H/W reset assertion and any CMD0 reset and not readable.

R/W/E: Multiple writable with value kept after power failure, H/W reset assertion and any CMD0 reset and readable.

R/W/C_P: Writable after value cleared by power failure and HW/rest assertion (the value not cleared by CMD0 reset) and readable.

R/W/E_P: Multiple writable with value reset after power failure, H/W reset assertion and any CMD0 reset and readable.

W/E_P: Multiple writable with value reset after power failure, H/W reset assertion and any CMD0 reset and not readable.

Table 93 — Extended CSD

Name	Field	Size (Bytes)	Cell Type	CSD-slice
Properties Segment				
Reserved[1]		6	TBD	[511:506]
Extended Security Commands Error	EXT_SECURITY_ERR	1	R	[505]
Supported Command Sets	S_CMD_SET	1	R	[504]
HPI features	HPI_FEATURES	1	R	[503]
Background operations support	BKOPS_SUPPORT	1	R	[502]
Max packed read commands	MAX_PACKED_READS	1	R	[501]
Max packed write commands	MAX_PACKED_WRITES	1	R	[500]
Data Tag Support	DATA_TAG_SUPPORT	1	R	[499]
Tag Unit Size	TAG_UNIT_SIZE	1	R	[498]
Tag Resources Size	TAG_RES_SIZE	1	R	[497]
Context management capabilities	CONTEXT_CAPABILITIES	1	R	[496]
Large Unit size	LARGE_UNIT_SIZE_M1	1	R	[495]
Extended partitions attribute support	EXT_SUPPORT	1	R	[494]
Supported modes	SUPPORTED_MODES	1	R	[493]
FFU features	FFU_FEATURES	1	R	[492]
Operation codes timeout	OPERATION_CODE_TIMEOUT	1	R	[491]
FFU Argument	FFU_ARG	4	R	[490:487]
Barrier support	BARRIER_SUPPORT	1	R	[486]
Reserved[1]		177	TBD	[485:309]
CMD Queuing Support	CMDQ_SUPPORT	1	R	[308]
CMD Queuing Depth	CMDQ_DEPTH	1	R	[307]

Name	Field	Size (Bytes)	Cell Type	CSD-slice
Reserved[1]		1	TBD	[306]
Number of FW sectors correctly programmed	NUMBER_OF_FW_SECTORS_CORRECTLY_PROGRAMMED	4	R	[305:302]
Vendor proprietary health report	VENDOR_PROPRIETARY_HEALTH_REPORT	32	R	[301:270]
Device life time estimation type B	DEVICE_LIFE_TIME_EST_TYP_B	1	R	[269]
Device life time estimation type A	DEVICE_LIFE_TIME_EST_TYP_A	1	R	[268]
Pre EOL information	PRE_EOL_INFO	1	R	[267]
Optimal read size	OPTIMAL_READ_SIZE	1	R	[266]
Optimal write size	OPTIMAL_WRITE_SIZE	1	R	[265]
Optimal trim unit size	OPTIMAL_TRIM_UNIT_SIZE	1	R	[264]
Device version	DEVICE_VERSION	2	R	[263:262]
Firmware version	FIRMWARE_VERSION	8	R	[261:254]
Power class for 200MHz, DDR at VCC= 3.6V	PWR_CL_DDR_200_360	1	R	[253]
Cache size	CACHE_SIZE	4	R	[252:249]
Generic CMD6 timeout	GENERIC_CMD6_TIME	1	R	[248]
Power off notification(long) timeout	POWER_OFF_LONG_TIME	1	R	[247]
Background operations status	BKOPS_STATUS	1	R	[246]
Number of correctly programmed sectors	CORRECTLY_PRG_SECTORS_NUM	4	R	[245:242]
1st initialization time after partitioning	INI_TIMEOUT_AP	1	R	[241]
Cache Flushing Policy	CACHE_FLUSH_POLICY	1	R	[240]
Power class for 52MHz, DDR at V_{CC} = 3.6V	PWR_CL_DDR_52_360	1	R	[239]
Power class for 52MHz, DDR at V_{CC} = 1.95V	PWR_CL_DDR_52_195	1	R	[238]
Power class for 200MHz at V_{CCQ} =1.95V, V_{CC} = 3.6V	PWR_CL_200_195	1	R	[237]
Power class for 200MHz, at V_{CCQ} =1.3V, V_{CC} = 3.6V	PWR_CL_200_130	1	R	[236]
Minimum Write Performance for 8bit at 52MHz in DDR mode	MIN_PERF_DDR_W_8_52	1	R	[235]
Minimum Read Performance for 8bit at 52MHz in DDR mode	MIN_PERF_DDR_R_8_52	1	R	[234]
Reserved[1]		1	TBD	[233]
TRIM Multiplier	TRIM_MULT	1	R	[232]
Secure Feature support	SEC_FEATURE_SUPPORT	1	R	[231]
Secure Erase Multiplier	SEC_ERASE_MULT	1	R	[230]
Secure TRIM Multiplier	SEC_TRIM_MULT	1	R	[229]
Boot information	BOOT_INFO	1	R	[228]
Reserved[1]		1	TBD	[227]
Boot partition size	BOOT_SIZE_MULT	1	R	[226]
Access size	ACC_SIZE	1	R	[225]
High-capacity erase unit size	HC_ERASE_GRP_SIZE	1	R	[224]
High-capacity erase timeout	ERASE_TIMEOUT_MULT	1	R	[223]
Reliable write sector count	REL_WR_SEC_C	1	R	[222]
High-capacity write protect group size	HC_WP_GRP_SIZE	1	R	[221]

Name	Field	Size (Bytes)	Cell Type	CSD-slice
Sleep current (V_{CC})	S_C_VCC	1	R	[220]
Sleep current (V_{CCQ})	S_C_VCCQ	1	R	[219]
Production state awareness timeout	PRODUCTION_STATE_AWARENESS_TIMEOUT	1	R	[218]
Sleep/awake timeout	S_A_TIMEOUT	1	R	[217]
Sleep Notification Timout[1]	SLEEP_NOTIFICATION_TIME	1	R	[216]
Sector Count	SEC_COUNT	4	R	[215:212]
Secure Write Protect Information	SECURE_WP_INFO	1	R	[211]
Minimum Write Performance for 8bit at52 MHz	MIN_PERF_W_8_52	1	R	[210]
Minimum Read Performance for 8bit at 52 MHz	MIN_PERF_R_8_52	1	R	[209]
Minimum Write Performance for 8bit at 26 MHz, for 4bit at 52MHz	MIN_PERF_W_8_26_4_52	1	R	[208]
Minimum Read Performance for 8bit at 26 MHz, for 4bit at 52MHz	MIN_PERF_R_8_26_4_52	1	R	[207]
Minimum Write Performance for 4bit at 26 MHz	MIN_PERF_W_4_26	1	R	[206]
Minimum Read Performance for 4bit at 26 MHz	MIN_PERF_R_4_26	1	R	[205]
Reserved[1]		1	TBD	[204]
Power class for 26 MHz at 3.6 V 1 R	PWR_CL_26_360	1	R	[203]
Power class for 52 MHz at 3.6 V 1 R	PWR_CL_52_360	1	R	[202]
Power class for 26 MHz at 1.95 V 1 R	PWR_CL_26_195	1	R	[201]
Power class for 52 MHz at 1.95 V 1 R	PWR_CL_52_195	1	R	[200]
Partition switching timing	PARTITION_SWITCH_TIME	1	R	[199]
Out-of-interrupt busy timing	OUT_OF_INTERRUPT_TIME	1	R	[198]
I/O Driver Strength	DRIVER_STRENGTH	1	R	[197]
Device type	DEVICE_TYPE	1	R	[196]
Reserved[1]		1	TBD	[195]
CSD STRUCTURE	CSD_STRUCTURE	1	R	[194]
Reserved[1]		1	TBD	[193]
Extended CSD revision	EXT_CSD_REV	1	R	[192]
Modes Segment				
Command set	CMD_SET	1	R/W/E_P	[191]
Reserved[1]		1	TBD	[190]
Command set revision	CMD_SET_REV	1	R	[189]
Reserved[1]		1	TBD	[188]
Power class	POWER_CLASS	1	R/W/E_P	[187]
Reserved[1]		1	TBD	[186]
High-speed interface timing	HS_TIMING	1	R/W/E_P	[185]
Strobe Support	STROBE_SUPPORT	1	R	[184]
Bus width mode	BUS_WIDTH	1	W/E_P	[183]
Reserved[1]		1	TBD	[182]
Erased memory content	ERASED_MEM_CONT	1	R	[181]

Name	Field	Size (Bytes)	Cell Type	CSD-slice
Reserved[1]		1	TBD	[180]
Partition configuration	PARTITION_CONFIG	1	R/W/E & R/W/E_P	[179]
Boot config protection	BOOT_CONFIG_PROT	1	R/W & R/W/C_P	[178]
Boot bus Conditions	BOOT_BUS_CONDITIONS	1	R/W/E	[177]
Reserved[1]		1	TBD	[176]
High-density erase group definition	ERASE_GROUP_DEF	1	R/W/E_P	[175]
Boot write protection status registers	BOOT_WP_STATUS	1	R	[174]
Boot area write protection register	BOOT_WP	1	R/W & R/W/C_P	[173]
Reserved[1]		1	TBD	[172]
User area write protection register	USER_WP	1	R/W, R/W/C_P & R/W/E_P	[171]
Reserved[1]		1	TBD	[170]
FW configuration	FW_CONFIG	1	R/W	[169]
RPMB Size	RPMB_SIZE_MULT	1	R	[168]
Write reliability setting register	WR_REL_SET	1	R/W	[167]
Write reliability parameter register	WR_REL_PARAM	1	R	[166]
Start Sanitize operation	SANITIZE_START	1	W/E_P	[165]
Manually start background operations	BKOPS_START	1	W/E_P	[164]
Enable background operations handshake	BKOPS_EN	1	R/W & R/W/E	[163]
H/W reset function	RST_n_FUNCTION	1	R/W	[162]
HPI management	HPI_MGMT	1	R/W/E_P	[161]
Partitioning Support	PARTITIONING_SUPPORT	1	R	[160]
Max Enhanced Area Size	MAX_ENH_SIZE_MULT	3	R	[159:157]
Partitions attribute	PARTITIONS_ATTRIBUTE	1	R/W	[156]
Partitioning Setting	PARTITION_SETTING_COMPLETED	1	R/W	[155]
General Purpose Partition Size	GP_SIZE_MULT	12	R/W	[154:143]
Enhanced User Data Area Size	ENH_SIZE_MULT	3	R/W	[142:140]
Enhanced User Data Start Address	ENH_START_ADDR	4	R/W	[139:136]
Reserved[1]		1	TBD	[135]
Bad Block Management mode	SEC_BAD_BLK_MGMNT	1	R/W	[134]
Production state awareness	PRODUCTION_STATE_AWARENESS	1	R/W/E	[133]
Package Case Temperature is controlled	TCASE_SUPPORT	1	W/E_P	[132]
Periodic Wake-up	PERIODIC_WAKEUP	1	R/W/E	[131]
Program CID/CSD in DDR mode support	PROGRAM_CID_CSD_DDR_SUPPORT	1	R	[130]

Name	Field	Size (Bytes)	Cell Type	CSD-slice
Reserved[1]		2	TBD	[129:128]
Vendor Specific Fields	VENDOR_SPECIFIC_FIELD	64	<vendor specific>	[127:64]
Native sector size	NATIVE_SECTOR_SIZE	1	R	[63]
Sector size emulation	USE_NATIVE_SECTOR	1	R/W	[62]
Sector size	DATA_SECTOR_SIZE	1	R	[61]
1st initialization after disabling sector size emulation	INI_TIMEOUT_EMU	1	R	[60]
Class 6 commands control	CLASS_6_CTRL	1	R/W/E_P	[59]
Number of addressed group to be Released	DYNCAP_NEEDED	1	R	[58]
Exception events control	EXCEPTION_EVENTS_CTRL	2	R/W/E_P	[57:56]
Exception events status	EXCEPTION_EVENTS_STATUS	2	R	[55:54]
Extended Partitions Attribute	EXT_PARTITIONS_ATTRIBUTE	2	R/W	[53:52]
Context configuration	CONTEXT_CONF	15	R/W/E_P	[51:37]
Packed command status	PACKED_COMMAND_STATUS	1	R	[36]
Packed command failure index	PACKED_FAILURE_INDEX	1	R	[35]
Power Off Notification	POWER_OFF_NOTIFICATION	1	R/W/E_P	[34]
Control to turn the Cache ON/OFF	CACHE_CTRL	1	R/W/E_P	[33]
Flushing of the cache	FLUSH_CACHE	1	W/E_P	[32]
Control to turn the Barrier ON/OFF	BARRIER_CTRL	1	R/W	[31]
Mode config	MODE_CONFIG	1	R/W/E_P	[30]
Mode operation codes	MODE_OPERATION_CODES	1	W/E_P	[29]
Reserved[1]		2	TBD	[28:27]
FFU status	FFU_STATUS	1	R	[26]
Pre loading data size	PRE_LOADING_DATA_SIZE	4	R/W/E_P	[25:22]
Max pre loading data size	MAX_PRE_LOADING_DATA_SIZE	4	R	[21:18]
Product state awareness enablement	PRODUCT_STATE_AWARENESS_ENABLEMENT	1	R/W/E & R	[17]
Secure Removal Type	SECURE_REMOVAL_TYPE	1	R/W & R	[16]
Command Queue Mode Enable	CMDQ_MODE_EN	1	R/W/E_P	[15]
Reserved[1]		15	TBD	[14:0]

NOTE 1 Reserved bits should read as “0.”
NOTE 2 Obsolete values should be don’t care.

7.4 Extended CSD register (cont'd)

7.4.1 EXT_SECURITY_ERR [505]

This field is a read only field through which the device indicates to the host the extended security error that occurred in relation to the Extended Security Protocol usage (this byte should be read in case that Bit4, EXTENDED_SECURITY_FAILURE, of the EXCEPTION_EVENTS_STATUS field in the EXT_CSD is set). The error bits are cleared by the given clear condition.

Table 94 — EXT_SECURITY_ERR byte description

Bit #	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Name	Reserved						ACCESS_DENIED (3)	SEC_INVALID_COMMAND_PARAMETERS
Clear Cond.							(2)	(1)

NOTE 1 Cleared by next valid PROTOCOL_RD/WR command acceptation
NOTE 2 Cleared by next valid WR/RD command acceptation
NOTE 3 ACCESS_DENIED error indication that the host attempts to access an area without the proper access rights, using a regular RD/WR command

7.4.2 S_CMD_SET [504]

This field defines the command sets supported by the Device.

Table 95 — Device-supported command sets

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

7.4.3 HPI_FEATURES [503]

This field indicates if the HPI feature is supported and the implementation that is used by the device.

Table 96 — HPI features

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved						HPI_IMPLEMENTATION	HPI_SUPPORT

Bit[7:2]: Reserved

Bit[1]: HPI_IMPLEMENTATION

0x0 : HPI mechanism implementation based on CMD13

0x1 : HPI mechanism implementation based on CMD12

Bit[0]: HPI_SUPPORT

0x0 : Obsolete

0x1 : HPI mechanism supported (default)

7.4 Extended CSD register (cont'd)

7.4.4 BKOPS_SUPPORT [502]

This field indicates if the background operations feature is supported by the device.

Table 97 — Background operations support

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

Bit[7:1]: Reserved

Bit[0]: SUPPORTED

0x0 : Obsolete
0x1 : Background operations are supported. The fields BKOPS_STATUS, BKOPS_EN, BKOPS_START and URGENT_BKOPS are supported. (default)

7.4.5 MAX_PACKED_READS [501]

This field describes the maximum number of commands that can be packed inside a packed read command. The mandatory minimum value for this field is 5 (MAX_PACKED_READS shall be 5 or higher)

7.4.6 MAX_PACKED_WRITES [500]

This field describes the maximum number of commands that can be packed inside a packed write command. The mandatory minimum value for this field is 3 (MAX_PACKED_WRITES shall be 3 or higher)

7.4.7 DATA_TAG_SUPPORT [499]

This field indicates if the Data Tag mechanism features are supported.

Bit[7:1]: Reserved

Bit[0]: SYSTEM_DATA_TAG_SUPPORT

0x0 : System Data Tag mechanism not supported (default)
0x1 : System Data Tag mechanism supported

7.4.8 TAG_UNIT_SIZE [498]

This field is used by the host to calculate the size of a Tag Unit in Bytes.

Tag Unit Size = $2^{TAG\_UNIT\_SIZE}$•Sector Size

The ranges that can be covered by the parameter are:

- From 512 Bytes to 128 Kbytes in case of Sector Size = 512 Bytes
- From 4Kbytes to 1 Mbytes in case of Sector Size = 4 Kbytes

7.4.9 TAG_RES_SIZE [497]

This field is defined to inform the host about the maximum quantity of resources in bytes allocated by the device to allocate system data by the tagging mechanism:

System Data Tag Resources Size = [(N• Tag Unit Size) • $2^{TAG\_RES\_SIZE}$]/2^{10}

(N• Tag Unit Size) represents the device capacity in terms of Tag Unit Size

The range of valid TAG_RES_SIZE values is from 0 to 6; Values in range of 7 to 0xFF are reserved.

The formula covers a range from about 0.1% to 6.25% of the device capacity.

7.4 Extended CSD register (cont'd)

7.4.10 CONTEXT_CAPABILITIES [496]

This field describes the capabilities of context management.

Table 98 — Context Management Context Capabilities

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	LARGE_UNIT_MAX_MULTIPLIER_M1			MAX_CONTEXT_ID			

MAX_CONTEXT_ID is the highest context ID that may be used in contexts.

The mandatory minimum value for this field is 5 plus the default ID #0 (MAX_CONTEXT_ID shall be 5 or higher)

LARGE_UNIT_MAX_MULTIPLIER_M1 is the highest multiplier that can be configured for Large Unit contexts, minus one. Large Unit contexts may be configured to have a multiplier in the range:

$1 \leq \text{multiplier} \leq (\text{LARGE\_UNIT\_MAX\_MULTIPLIER\_M1} + 1)$

7.4.11 LARGE_UNIT_SIZE_M1 [495]

This field describes the size of the Large Unit, minus one.

Large Unit size = 1MB * (LARGE_UNIT_SIZE_M1 + 1)

7.4.12 EXT_SUPPORT [494]

This field describes the extended partitions attribute support.

A bit value of '1' denotes a supported type. On this specification, bits 0 and 1 are set.

Table 99 — Extended CSD Register Support

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved						Non-persistent	System code

7.4.13 SUPPORTED_MODES [493]

Bit [0]: '0' FFU is not supported by the device. '1' FFU is supported by the device.

Bit [1]: '0' Vendor Specific Mode (VSM) is not supported by the device. '1' Vendor Specific mode is supported by the device.

Table 100 — SUPPORTED_MODES

Others	Bit 1	Bit 0
Reserved	VSM	FFU

7.4.14 FFU_FEATURES [492]

Table 101 — FFU FEATURES

Others	Bit 0
Reserved	SUPPORTED_MODE_OPERATION_CODES

SUPPORTED_MODE_OPERATION_CODES:

- '0' – Device does not support MODE_OPERATION_CODES field
- '1' – Device support MODE_OPERATION_CODES field

7.4 Extended CSD register (cont'd)

7.4.15 OPERATION_CODES_TIMEOUT [491]

This field indicates the maximum timeout for the SWITCH command (CMD6) when setting a value to the MODE_OPERATION_CODES field.

The formula to calculate the max timeout value is:

Operation Codes Timeout = 100us * 2^ OPERATION_CODES_TIMEOUT

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

Table 102 — MODE_OPERATION_CODES timeout definition

Value	Timeout Values
0x00	Not defined
0x01	100us x 2^1 = 200us
0x02	100us x 2^2 = 400us
..	..
0x17	100us x 2^{23} = 838.86s
0x18 – 0xFF	Reserved

7.4.16 FFU_ARG [490-487]

Using this field the device reports to the host the value that the host should set as an argument for the read and write commands in FFU mode.

7.4.17 BARRIER_SUPPORT [486]

This field indicates whether the device supports the barrier command. Value 0x00 indicates that the device does not support barrier, and value 0x01 indicates that the device supports barrier. All other values are reserved.

7.4.18 CMDQ_SUPPORT [308]

This field indicates whether command queuing is supported by the device.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved							CMDQ Support

Bit encoding:

- [7:1]: Reserved
- [0]: Command queuing support:
    - 0: Command queuing is not supported
    - 1: Command queuing is supported

7.4 Extended CSD register (cont'd)

7.4.19 CMDQ_DEPTH [307]

This field is used to calculate the depth of the queue supported by the device. The maximum depth allowed by the standard is 32. The range of allowed Task IDs is 0 through N.

For example, if N = 0x5 the queue depth is 0x6 and the valid task IDs are 0, 1, 2, 3, 4, 5.

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

Bit encoding:

- [7:5]: Reserved
- [4:0]: N, a parameter used to calculate the Queue Depth of task queue in the device. Queue Depth = N+1.

7.4.20 NUMBER_OF_FW_SECTORS_CORRECTLY_PROGRAMMED [305-302]

Using this field the device reports the number of sectors, from the firmware bundle, that were programmed correctly into the device. The value is in terms of 512 Bytes or in multiple of eight 512Bytes sectors (4KBytes) depending on the value of the DATA_SECTOR_SIZE field.

7.4.21 VENDOR_PROPRIETARY_HEALTH_REPORT [301-270]

This Fields reserved for vendor proprietary health report.

7.4.22 DEVICE_LIFE_TIME_EST_TYP_B [269]

This field provides an estimated indication about the device life time that is reflected by the averaged wear out of memory of Type B relative to its maximum estimated device life time.

Table 103 — Device life time estimation type B value

Value	Description
0x00	Not defined
0x01	0% - 10% device life time used
0x02	10% -20% device life time used
0x03	20% -30% device life time used
0x04	30% - 40% device life time used
0x05	40% - 50% device life time used
0x06	50% - 60% device life time used
0x07	60% - 70% device life time used
0x08	70% - 80% device life time used
0x09	80% - 90% device life time used
0x0A	90% - 100% device life time used
0x0B	Exceeded its maximum estimated device life time
Others	Reserved

7.4 Extended CSD register (cont'd)

7.4.23 DEVICE_LIFE_TIME_EST_TYP_A [268]

This field provides an estimated indication about the device life time that is reflected by the averaged wear out of memory of Type A relative to its maximum estimated device life time.

Table 104 — Device life time estimation type A value

Value	Description
0x00	Not defined
0x01	0% - 10% device life time used
0x02	10% -20% device life time used
0x03	20% -30% device life time used
0x04	30% - 40% device life time used
0x05	40% - 50% device life time used
0x06	50% - 60% device life time used
0x07	60% - 70% device life time used
0x08	70% - 80% device life time used
0x09	80% - 90% device life time used
0x0A	90% - 100% device life time used
0x0B	Exceeded its maximum estimated device life time
Others	Reserved

7.4.24 PRE_EOL_INFO [267]

This field provides indication about device life time reflected by average reserved blocks.

Table 105 — Pre EOL info value

Value	Pre-EOL Info.	Description
0x00	Not Defined	
0x01	Normal	Normal
0x02	Warning	Consumed 80% of reserved block
0x03	Urgent	
0x04 ~ 0xFF	Reserved	

7.4 Extended CSD register (cont'd)

7.4.25 OPTIMAL_READ_SIZE [266]

This field provides the minimum optimal (for the device) read unit size for the different partitions.

Table 106 — Optimal read size value

Value	Optimal Read Size
0x00	Not defined
0x01	4KB x 1 = 4KB
0x02	4KB x 2 = 8KB
…	
0xFF	4KB x 255 = 1020KB

7.4.26 OPTIMAL_WRITE_SIZE [265]

This field provides the minimum optimal (for the device) write unit size for the different partitions. "Optimal" relates to the minimum wearout done by the device.

Table 107 — Optimal write size value

Value	Optimal Write Size
0x00	Not defined
0x01	4KB x 1 = 4KB
0x02	4KB x 2 = 8KB
…	
0xFF	4KB x 255 = 1020KB

7.4.27 OPTIMAL_TRIM_UNIT_SIZE [264]

This field provides the minimum optimal (for the device) trim unit size for the different partitions. "Optimal" relates to the minimum wearout done by the device.

Table 108 — Optimal trim unit size value

Value	Optimal Trim unit Size
0x00	Not defined
0x01	4KB x 1 = 4KB
0x02	4KB x 2 = 8KB
0x03	4KB x 4 = 16KB
…	
0x15	4KB x 2^20 = 4GB
Reserved	

7.4.28 DEVICE_VERSION [263-262]

This field provides the device version.

7.4 Extended CSD register (cont'd)

7.4.29 FIRMWARE_VERSION [261-254]

This field provides the device firmware version.

7.4.30 CACHE_SIZE [252:249]

This field indicates the existence and size of the volatile cache in the *e*•MMC device. Value 0x00 indicates that there is no cache existing. Any value higher than 0x00 indicates that there is a cache existing and the size of it. The size is indicated as multiple of kilobits. The value of the CACHE_SIZE register informs the multiplier. The MSB of the multiplier is in the [252] byte and LSB in [249] byte.

Size of the Cache = CACHE_SIZE x 1kb

7.4.31 GENERIC_CMD6_TIME [248]

This field indicates the default maximum timeout for a SWITCH command (CMD6) unless a specific timeout is defined when accessing a specific field. Additionally, this field doesn't define the timeout when starting a background operation (writing to field BKOPS_START[164]) or starting a sanitize operation (writing to field SANITIZE_START[165]) or flushing the cache (writing to field FLUSH_CACHE[32]). Time is expressed in units of 10-milliseconds.

Table 109 — Generic Switch Timeout Definition

Value	Timeout value Definition
0x00	Not defined
0x01	10ms x 1 = 10 ms
0x02	10ms x 2 = 20 ms
…	…
0xFF	10 ms x 255 = 2550 ms

7.4.32 POWER_OFF_LONG_TIME [247]

This field indicates the maximum timeout for the SWITCH command (CMD6) when notifying the device that power is about to be turned off by writing POWER_OFF_LONG to POWER_OFF_NOTIFICATION[34] byte. In case other modes are set in POWER_OFF_NOTIFICATION, the timeout is the generic CMD6 timeout. Time is expressed in units of 10-milliseconds.

Table 110 — Power off long switch timeout definition

Value	Timeout value Definition
0x00	Not defined
0x01	10ms x 1 = 10 ms
0x02	10ms x 2 = 20 ms
…	…
0xFF	10 ms x 255 = 2550 ms

7.4 Extended CSD register (cont'd)

7.4.33 BKOPS_STATUS [246]

This field indicates the level of background operations urgency.

Table 111 — Background operations status

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

Bit[7:2]: Reserved

Bit[1:0]: OUTSTANDING

0x0 : No operations required

0x1 : Operations outstanding (non critical)

0x2 : Operations outstanding (performance being impacted)

0x3 : Operations outstanding (critical)

7.4.34 CORRECTLY_PRG_SECTORS_NUM [245:242]

This field indicates how many sectors were successfully programmed by the last WRITE_MULTIPLE_BLOCK command (CMD25). The value is in terms of 512 Bytes or in multiple of eight 512Bytes sectors (4KBytes) depending on the value of the DATA_SECTOR_SIZE field

Table 112 — Correctly programmed sectors number

CORRECTLY_PRG_SECTORS_NUM	
EXT_CSD[245]	CORRECTLY_PRG_SECTORS_NUM_3
EXT_CSD[244]	CORRECTLY_PRG_SECTORS_NUM_2
EXT_CSD[243]	CORRECTLY_PRG_SECTORS_NUM_1
EXT_CSD[242]	CORRECTLY_PRG_SECTORS_NUM_0

Number of correctly programmed sectors =

$[CORRECTLY\_PRG\_SECTORS\_NUM\_3 * 2^{24} + CORRECTLY\_PRG\_SECTORS\_NUM\_2 * 2^{16} + CORRECTLY\_PRG\_SECTORS\_NUM\_1 * 2^{8} + CORRECTLY\_PRG\_SECTORS\_NUM\_0 * 2^{0}]$

NOTE It is recommended for hosts to refer to this field for avoiding re-writing again the successfully programmed sectors when repeating an interrupted write command.

7.4.35 INI_TIMEOUT_AP [241]

This register indicates the maximum initialization timeout during the first power up after successful partitioning of an *e*•MMC device. Note that all of the initialization timeouts during consecutive power ups will have timeout max 1s (like in case of non-partitioned *e*•MMC device).

Table 113 — Initialization Time out value

Value	Timeout Value
0x00	Not defined
0x01	100ms × 1 = 100 ms
...	...
0xFF	100ms × 255 = 25500 ms

7.4 Extended CSD register (cont'd)

7.4.36 CACHE_FLUSH_POLICY [240]

Table 114 — Cache Flushing Policy

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

Bit [0]: FIFO

0b: Device flushing policy is not provided by the device.

1b: Device is using a FIFO policy for cache flushing (see 6.6.34.1 Cache Flushing Policy).

Bits [1-7]: Reserved.

7.4.37 TRIM_MULT [232]

This register is used to calculate the TRIM and DISCARD function timeout. The following formula defines the timeout value for the TRIM and DISCARD operation of inside a logical erase group.

TRIM Timeout = 300ms x TRIM_MULT

If the host executes TRIM or DISCARD operation including write sectors belonging to multiple erase groups, the total timeout value should be the multiple of the number of the erase groups involved.

Table 115 — TRIM/DISCARD Time out value

Value	Timeout Value
0x00	Not defined
0x01	300ms × 1 = 300 ms
...	...
0xFF	300ms × 255 = 76500 ms

7.4 Extended CSD register (cont'd)

7.4.38 SEC_FEATURE_SUPPORT [231]

This byte allows the host to determine the secure data management features that are supported on the Device. The bits in this register determine whether the ERASE (CMD38) command arguments are supported and whether the host can manage the way defected portions of the memory array are retired by using SEC_BAD_BLK_MGMNT (EXT_CSD[134])

Table 116 — SEC Feature Support

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	SEC_ SANITIZE	Reserved	SEC_GB_ CL_EN	Reserved	SEC_BD_BLK_EN	Reserved	SECURE_ ER_EN(1)

(1) See 6.6.13 and 6.6.14

Bit 7: Reserved

Bit 6: SEC_SANITIZE

0x1: Device supports the sanitize operation.

0x0: Device does not support the sanitize operation.

Bit 5: Reserved

Bit 4: SEC_GB_CL_EN (R)

0x0: Device does not support the secure and insecure trim operations

0x1: Device supports the secure and insecure trim operations. This bit being set means that argument bits 15 and 0 are supported with CMD38

Bit 3: Reserved

Bit 2: SEC_BD_BLK_EN (R)

0x0 : Device does not support the automatic erase operation on retired defective portions of the array.

0x1 : Device supports the automatic erase operation on retired defective portions of the array. This bit being set enables the host to set SEC_BAD_BLK_MGMNT (EXT_CSD[134]).

Bit 1: Reserved

Bit 0: SECURE_ER_EN (R)

0x0: Secure purge operations are not supported on the device

0x1: Secure purge operations are supported. This bit being set allows the host to set bit 31 of the argument for the ERASE (CMD38) Command

7.4 Extended CSD register (cont'd)

7.4.39 SEC_ERASE_MULT [230]

This register is used to calculate Secure_Erase function timeout. The following formula defines the time out value for the Secure Erase operation of one logical erase group.

Secure Erase Timeout = 300ms x ERASE_TIMEOUT_MULT x SEC_ERASE_MULT.

If the host executes Secure Erase operation including erase groups the total timeout value should be the multiple of the number of the erase groups involved.

SEC_ERASE_MULT should represent the worst-case timeout value of all supported SECURE_REMOVAL_TYPE modes in EXT_CSD[16].

Table 117 — Secure Erase time-out value

Value	Timeout Value
0x00	Not defined
0x01	300ms × ERASE_TIMEOUT_MULT x 1
...	...
0xFF	300ms × ERASE_TIMEOUT_MULT x 255

7.4.40 SEC_TRIM_MULT [229]

This register is used to calculate Secure_TRIM function timeout. The following formula defines the time-out value for the Secure_TRIM operation of one logical erase group.

Secure TRIM Timeout = 300ms x ERASE_TIMEOUT_MULT x SEC_TRIM_MULT

If the host executes Secure Trim operation including write sectors belonging to multiple erase groups, the total timeout value should be the multiple of the number of the erase groups involved.

SEC_TRIM_MULT should represent the worst-case timeout value of all supported SECURE_REMOVAL_TYPE modes in EXT_CSD[16].

Table 118 — Secure Erase time-out value

Value	Timeout Value
0x00	Not defined
0x01	300ms × ERASE_TIMEOUT_MULT x 1
...	...
0xFF	300ms × ERASE_TIMEOUT_MULT x 255

7.4 Extended CSD register (cont'd)

7.4.41 BOOT_INFO [228]

Table 119 — Boot information

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

Bit[7:3]: Reserved

Bit[2]: HS_BOOT_MODE

0: Device does not support high speed timing during boot.

1: Device supports high speed timing during boot.

Bit[1]: DDR_BOOT_MODE

0: Device does not support dual data rate during boot.

1: Device supports dual data rate during boot.

Bit[0]: ALT_BOOT_MODE

0x0 : Device does not support alternative boot method (obsolete)

0x1 : Device supports alternative boot method. Device must show "1" since this is mandatory in v4.4 standard

The only currently valid values for this register are 0x0, 0x1, 0x05, and 0x07. A device supporting dual data rate mode during boot shall also have bit 2 set.

7.4.42 BOOT_SIZE_MULT [226]

The boot partition size is calculated from the register by using the following equation: Boot Partition size = 128Kbytes × BOOT_SIZE_MULT

Table 120 — Boot partition size

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

7.4 Extended CSD register (cont'd)

7.4.43 ACC_SIZE [225]

Table 121 — Access size

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

Bit[7:4]: Reserved

Bit[3:0]: SUPER_PAGE_SIZE

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

- As a guide for format clusters
- To prevent format-page misalignment
- As a guide for minimum data-transfer size

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

Table 122 — Superpage size

Value	Super-page Size
0x0	Not defined
0x1	512 × 1 = 512 bytes
0x2	512 × 2 = 1K bytes
:	:
0x8	512 × 128 = 64K bytes
0x9–0xF	Reserved

7.4.44 HC_ERASE_GRP_SIZE [224]

This register defines the erase-unit size for high-capacity memory. If the master enables bit "0" in the extended CSD register byte [175], the slave uses these value for the erase operation. Erase Unit Size = 512Kbyte × HC_ERASE_GRP_SIZE

Table 123 — Erase-unit size

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

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

7.4 Extended CSD register (cont'd)

7.4.45 ERASE_TIMEOUT_MULT [223]

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

Erase Timeout = 300 ms × ERASE_TIMEOUT_MULT

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

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

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

Table 124 — Erase timeout values

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

7.4.46 REL_WR_SEC_C [222]

The REL_WR_SEC_C [222] register shall be set to 1 and has no impact on the reliable write operation.

Table 125 — Reliable write sector count

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

7.4.47 HC_WP_GRP_SIZE [221]

This register defines the write protect group size for high-capacity memory. If the ENABLE bit in ERASE_GROUP_DEF is set to HIGH, the write protect group size would be defined as follows: Write protect group size = 512KB * HC_ERASE_GRP_SIZE * HC_WP_GRP_SIZE.

Table 126 — Write protect group size

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

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

7.4 Extended CSD register (cont'd)

7.4.48 S_C_VCC[220] and S_C_VCCQ[219]

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

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

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

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

Table 127 — S_C_VCC, S_C_VCCQ Sleep Current

Value	Value definition
0x00	Not defined
0x01	1µA × 21 = 2µA
0x02	1µA × 22 = 4µA
:	:
0x0D	1µA × 213 = 8.192mA
0x0E–0xFF	Reserved

7.4.49 PRODUCTION_STATE_AWARENESS_TIMEOUT [218]

This field indicates the maximum timeout for the SWITCH command (CMD6) when setting a value to the PRODUCTION_STATE_AWARENESS [133] field.

The formula to calculate the max timeout value is:

Production State Timeout = 100us * 2^ PRODUCTION_STATE_AWARENESS_TIMEOUT

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

Table 128 — Production State Awareness timeout definition

Value	Timeout Values
0x00	Not defined
0x01	100us x 2^1 = 200us
0x02	100us x 2^2 = 400us
..	..
0x17	100us x 2^{23} = 838.86s
0x18 – 0xFF	Reserved

7.4 Extended CSD register (cont'd)

7.4.50 S_A_TIMEOUT [217]

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

Sleep/Awake Timeout = 100ns * 2^S_A_TIMEOUT

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

Table 129 — Sleep/awake timeout values

Value	Timeout Values
0x00	Not defined
0x01	100ns × 21 = 200ns
0x02	100ns × 22 = 400ns
:	:
0x17	100ns × 223 = 838.86ms
0x18–0xFF	Reserved

7.4.51 SLEEP_NOTIFICATION_TIME [216]

This field indicates the maximum timeout for the SWITCH command (CMD6) when notifying the device that it is about to be move to sleep state (slp) by writing SLEEP_NOTIFICATION to POWER_OFF_NOTIFICATION [34] byte. Time is expressed in units of 10-millimicroseconds. The formula to calculate the max timeout value is:

Sleep Notification Time out value = 10us * 2^SLEEP_NOTIFICATION_TIME

Max register value defined is 0x17 which equals 83.88s timeout. Values between 0x18 and 0xFF are reserved.

Table 130 — Sleep Notification timeout values

Value	Timeout Values
0x00	Not defined
0x01	$10us \times 2^1 = 20us$
0x02	$10us \times 2^2 = 40us$
:	:
0x17	$10us \times 2^{23} = 83.88s$
0x18–0xFF	Reserved

7.4.52 SEC_COUNT [215:212]

The device density is calculated from the register by multiplying the value of the register (sector count) by 512B/sector as shown in following equation.

Device density = SEC_COUNT x 512B

The maximum density possible to be indicated is thus 4 294 967 295x 512B.

The addressable sector range for the device will be from Sector 0 to Sector (SEC_COUNT-1).

The least significant byte (LSB) of the sector count value is the byte [212].

When the partition configuration is executed by host, device will re-calculate the SEC_COUNT value that can indicate the size of user data area after the partition.

7.4 Extended CSD register (cont'd)

7.4.53 SECURE_WP_INFO[211]

The SECURE_WP_SUPPORT field indicates whether the device is supporting secure write protection mode. The SECURE_WP_EN_STATUS is showing the value of SECURE_WP_EN defined in Authenticated Device Configuration Area.

Table 131 — SECURE_WP_INFO

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reserved						SECURE_WP_EN_STATUS	SECURE_WP_SUPPORT

Bit[7:2] : Reserved

Bit[1] : SECURE_WP_EN_STATUS (R)

0x0: Legacy Write Protection mode.

0x1: Secure Write Protection mode.

Bit[0] : SECURE_WP_SUPPORT (R)

0x0: Secure Write Protection is NOT supported by this device

0x1: Secure Write Protection is supported by this device

7.4 Extended CSD register (cont'd)

7.4.54 MIN_PERF_a_b_ff [210/:205] and MIN_PERF_DDR_a_b_ff [235:234]

These fields define the overall minimum performance value for the read and write access with different bus width and max clock frequency modes. The value in the register is coded as in Table 132. Values, other than those defined, are considered illegal.

Table 132 — R/W access performance values

Value	Performance
Single Data Rate mode	
0x00	For Devices not reaching the 2.4MB/s value
0x08	Class A: 2.4MB/s and is the next allowed value (16x150kB/s)
0x0A	Class B: 3.0MB/s and is the next allowed value (20x150kB/s)
0x0F	Class C: 4.5MB/s and is the next allowed value (30x150kB/s)
0x14	Class D: 6.0MB/s and is the next allowed value (40x150kB/s)
0x1E	Class E: 9.0MB/s and is the next allowed value (60x150kB/s)
0x28	Class F: Equals 12.0MB/s and is the next allowed value (80x150kB/s)
0x32	Class G: Equals 15.0MB/s and is the next allowed value (100x150kB/s)
0x3C	Class H: Equals 18.0MB/s and is the next allowed value (120x150kB/s)
0x46	Class J: Equals 21.0MB/s and is the next allowed value (140x150kB/s) This is also the highest class that any mobile *e*•MMC Device is needed to support in mid bus category operation mode (26MHz with 8bit data bus or 52MHz with 4bit data bus). A Device supporting any higher class than this have to support this Class (in mid category bus operation mode) and Class E also (in low category bus operation mode)
0x50	Class K: Equals 24.0MB/s and is the next allowed value (160x150kB/s)
0x64	Class M: Equals 30.0MB/s and is the next allowed value (200x150kB/s)
0x78	Class O: Equals 36.0MB/s and is the next allowed value (240x150kB/s)
0x8C	Class R: Equals 42.0MB/s and is the next allowed value (280x150kB/s)
0xA0	Class T: Equals 48.0MB/s and is the last defined value (320x150kB/s)
Dual Data Rate mode	
0x00	For Devices not reaching the 4.8MB/s value
0x08	Class A: Equals 4.8MB/s and is the next allowed value (16x300kB/s)
0x0A	Class B: Equals 6.0MB/s and is the next allowed value (20x300kB/s)
0x0F	Class C: Equals 9.0MB/s and is the next allowed value (30x300kB/s)
0x14	Class D: Equals 12.0MB/s and is the next allowed value (40x300kB/s)
0x1E	Class E: Equals 18.0MB/s and is the last defined value (60x300kB/s)
0x28	Class F: Equals 24.0MB/s and is the next allowed value (80x300kB/s)
0x32	Class G: Equals 30.0MB/s and is the next allowed value (100x300kB/s)
0x3C	Class H: Equals 36.0MB/s and is the next allowed value (120x300kB/s)
0x46	Class J: Equals 42.0MB/s and is the last defined value (140x300kB/s)
0x50	Class K: Equals 48.0MB/s and is the next allowed value (160x300kB/s)
0x64	Class M: Equals 60.0MB/s and is the next allowed value (200x300kB/s)
0x78	Class O: Equals 72.0MB/s and is the next allowed value (24 0x300kB/s)
0x8C	Class R: Equals 84.0MB/s and is the next allowed value (280x300kB/s)
0xA0	Class T: Equals 96.0MB/s and is the last defined value (320x300kB/s)

7.4.55 PWR_CL_ff_vvv [203:200] , PWR_CL_ff_vvv[237:236] , PWR_CL_DDR_ff_vvv [239:238] and PWR_CL_DDR_ff_vvv[253]

These fields define the supported power classes by the Device. By default, the Device has to operate at maximum frequency using 1 bit bus configuration, within the default max current consumption, as stated in Table 133. If 4 bit/8 bits bus configurations require increased current consumption, it has to be stated in these registers.

7.4.55 PWR_CL_ff_vvv [203:200] , PWR_CL_ff_vvv[237:236] , PWR_CL_DDR_ff_vvv [239:238] and PWR_CL_DDR_ff_vvv[253] (cont'd)

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

The PWR_52_vvv registers are not defined for 26MHz *e*•MMCs.

PWR_CL_DDR_200_vvv[253] registers are for HS400 mode,

Table 133 — Power classes

Voltage	Value	Max RMS Current	Max Peak Current	Remarks
3.6V	0	100 mA	200 mA	Default current consumption for high voltage Devices
	1	120 mA	220 mA	
	2	150 mA	250 mA	
	3	180 mA	280 mA	
	4	200 mA	300 mA	
	5	220 mA	320 mA	
	6	250 mA	350 mA	
	7	300 mA	400 mA	
	8	350 mA	450 mA	
	9	400 mA	500 mA	
	10	450 mA	550 mA	
	11	500mA	600mA	
	12	600mA	700mA	
	13	700mA	800mA	
	14	800mA	900mA	
	15	>800mA	>900mA	
1.95V	0	65 mA	130 mA	Default current consumption for Dual voltage Devices
	1	70 mA	140 mA	
	2	80 mA	160 mA	
	3	90 mA	180 mA	
	4	100 mA	200 mA	
	5	120 mA	220 mA	
	6	140 mA	240 mA	
	7	160 mA	260 mA	
	8	180 mA	280 mA	
	9	200 mA	300 mA	
	10	250 mA	350 mA	
	11	300mA	400mA	
	12	350mA	450mA	
	13	400mA	500mA	
	14	500mA	600mA	
	15	>500mA	>600mA	

7.4.55 PWR_CL_ff_vvv [203:200] , PWR_CL_ff_vvv[237:236] , PWR_CL_DDR_ff_vvv [239:238] and PWR_CL_DDR_ff_vvv[253] (cont'd)

The measurement for max RMS current is done as average RMS current consumption over a period of 100ms. Max peak current is defined as absolute max value not to be exceeded at all. The conditions under which the power classes are defined are:

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

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

Device may specify in their datasheet the performance per Power Class and whether the package case (Tc) temperature conditions shall be met as given in Annex A.10

7.4.56 PARTITION_SWITCH_TIME [199]

This field indicates the maximum timeout for the SWITCH command (CMD6) when switching partitions by changing PARTITION_ACCESS bits in PARTITION_CONFIG field (EXT_CSD byte [179]). Time is expressed in units of 10-milliseconds.

Table 134 — Partition switch timeout definition

Value	Timeout value definition
0x00	Not defined
0x01	10ms x 1 = 10ms
0x02	10ms x 2 = 20ms
…	…
0xFF	10ms x 255 = 2550ms

7.4.57 OUT_OF_INTERRUPT_TIME [198]

This field indicates the maximum timeout to close a command interrupted by HPI – time between the end bit of CMD12/13 to the DAT0 release by the device. Time is expressed in units of 10-milliseconds.

Table 135 — Out-of-interrupt timeout definition

Value	Timeout value definition
0x00	Not defined
0x01	10ms x 1 = 10ms
0x02	10ms x 2 = 20ms
:	:
0xFF	10ms x 255 = 2550ms

7.4 Extended CSD register (cont'd)

7.4.58 DRIVER_STRENGTH [197]

Indicates the I/O driver strength types that are supported by a device.

Table 136 — Supported Driver Strengths

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

Bit [7:5]: Reserved

Bit [4]: Type 4

0x0 : not supported

0x1 : supported

Bit [3]: Type 3

0x0 : not supported

0x1 : supported

Bit [2]: Type 2

0x0 : not supported

0x1 : supported

Bit [1]: Type 1

0x0 : not supported

0x1 : supported

Bit [0]: Type 0

0x0 : N/A

0x1 : supported (mandatory)

Refer to 10.5.4.1 for the Driver Types Definition.

7.4 Extended CSD register (cont'd)

7.4.59 DEVICE_TYPE [196]

This field defines the type of the Device.

Table 137 — Device types

Bit	Device Type
7	HS400 Dual Data Rate *e*•MMC at 200 MHz – 1.2 V I/O
6	HS400 Dual Data Rate *e*•MMC at 200 MHz – 1.8 V I/O
5	HS200 Single Data Rate *e*•MMC at 200 MHz - 1.2 V I/O
4	HS200 Single Data Rate *e*•MMC at 200 MHz - 1.8 V I/O
3	High-Speed Dual Data Rate *e*•MMC at 52 MHz - 1.2 V I/O
2	High-Speed Dual Data Rate *e*•MMC at 52 MHz - 1.8 V or 3 V I/O
1	High-Speed *e*•MMC at 52 MHz - at rated device voltage(s)
0	High-Speed *e*•MMC at 26 MHz - at rated device voltage(s)

The - currently valid values for this field are including followings; 0x01, 0x03, 0x07, 0x0B, 0x0F, 0x13, 0x17, 0x1B,0x1F, 0x23, 0x27, 0x2B,0x2F, 0x33, 0x37, 0x3B,0x3F, 0x41, 0x43, 0x47, 0x4B, 0x4F, 0x51, 0x53, 0x57, 0x5B, 0x5F, 0x61, 0x63, 0x67, 0x6B, 0x6F, 0x71, 0x73, 0x77, 0x7B, 0x7F, 0x81, 0x83, 0x87, 0x8B, 0x8F,0x91, 0x93, 0x97, 0x9B, 0x9F, 0xA1, 0xA3, 0xA7, 0xAB, 0xAF, 0xB1, 0xB3, 0xB7, 0xBB, 0xBF, 0xC1, 0xC3, 0xC7, 0xCB, 0xCF, 0xD1, 0xD3, 0xD7, 0xDB, 0xDF, 0xE1, 0xE3, 0xE7, 0xEB, 0xEF, 0xF1, 0xF3, 0xF7, 0xFB, 0xFF . Ex) A dual voltage 1.2 V/1.8 V device that supports 52 MHz DDR mode at 1.8 V and not at 1.2 V will be coded 0x7.

A dual voltage 1.2 V/1.8 V device that supports 52 MHz Double-Data-Rate mode at 1.8 V and 26 Mhz/52 MHz Single-Data-Rate mode at 1.2 V will be also coded 0x7. For all the device types that cover several voltage ranges, the data sheet of the device shall specify the specific supported voltage range for V_{CC} and V_{CCQ}.

Dual Data Rate mode support is optional

7.4.60 CSD_STRUCTURE [194]

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

Table 138 — CSD register structure

CSD_STRUCTURE	CSD structure version	Valid for System Specification Version
0	CSD version No. 1.0	Allocated by MMCA
1	CSD version No. 1.1	Allocated by MMCA
2	CSD version No. 1.2	Version 4.1–4.2–4.3-4.41-4.5-4.51-5.0-5.01-5.1
3–255	Reserved for future use	

7.4 Extended CSD register (cont'd)

7.4.61 EXT_CSD_REV [192]

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

Table 139 — Extended CSD revisions

EXT_CSD_REV	Extended CSD Revision
255–9	Reserved
8	Revision 1.8 (for MMC v5.1)
7	Revision 1.7 (for MMC v5.0, v5.01)
6	Revision 1.6 (for MMC v4.5, v4.51)
5	Revision 1.5 (for MMC v4.41)
4	Revision 1.4 (Obsolete)
3	Revision 1.3 (for MMC v4.3)
2	Revision 1.2 (for MMC v4.2)
1	Revision 1.1 (for MMC v4.1)
0	Revision 1.0 (for MMC v4.0)

7.4.62 CMD_SET [191]

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

7.4.63 CMD_SET_REV [189]

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

Table 140 — Standard MMC command set revisions

Code	MMC Revision
255–1	Reserved
0	v4.0

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

7.4.64 POWER_CLASS [187]

This field contains the 4-bit value of the selected power class for the Device. The power classes are defined in Table 141. The host should be responsible of properly writing this field with the maximum power class it allows the Device to use. The Device uses this information to, internally, manage the power budget and deliver an optimized performance.

Table 141 — Power class codes

Bits	Description
[7:4]	Reserved
[3:0]	Device power class code (See Table 133)

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

7.4 Extended CSD register (cont'd)

7.4.65 HS_TIMING [185]

Table 142 — HS_TIMING (timing and driver strength)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Selected Driver Strength				Timing Interface			

- This field is used by the host to select both Timing Interface and Driver Strength. This byte is composed from two fields, each represented by a nibble.
- Timing Interface [0:3]: This field is 0 after power-on, H/W reset or software reset, thus selecting the backwards compatibility interface timing for the device. If the host sets 1 to this field, the device changes its timing to high speed interface timing (see 10.6.1). If the host sets value 2 the device changes its timing to HS200 interface timing (see 10.8.1). If the host sets HS_TIMING[3:0] to 0x3, the device changes its timing to HS400 interface timing (see 10.10)

Table 143 — HS_TIMING Interface values

Value	Timing Interface	Remarks
0x0	Selecting backwards compatibility interface timing	
0x1	High Speed	
0x2	HS200	
0x3	HS400	

- Selected Driver Strength [4:7]: Using this field the host sets the required Driver Strength from device. The default and mandatory value is (0x0). Refer to 10.5.4.1 for the Driver Types Definition.

7.4.66 STROBE_SUPPORT [184]

This register indicates whether a device supports Enhanced Strobe mode for operation modes that STROBE is used (ie HS400).

Value "0x0" indicates No support of Enhanced Strobe mode
Value "0x1" indicates device supports Enhanced Strobe mode

In case device supports Enhanced Strobe mode it may be enabled in BUS_WIDTH [183] register of EXT_CSD.

7.4 Extended CSD register (cont'd)

7.4.67 BUS_WIDTH [183]

It is set to '0' (1 bit data bus) after power up and can be changed by a SWITCH command.
Bus Width, Normal or DDR mode and Strobe mode (for HS400) are defined through BUS_WIDTH register.

Table 144 — BUS_WIDTH

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Enhanced Strobe	Reserved	Reserved	Reserved	Bus Mode Selection			

Bit 7:

0x0: Strobe is provided only during Data Out and CRC response [Default]
0x1: Strobe is provided during Data Out, CRC response and CMD Response

The support of Enhanced Strobe mode is optional for devices. STROBE_SUPPORT[184] register of EXT_CSD indicates whether a device supports that mode.

Bit[3-0]: define the Bus Mode Selection as defined in Table 145

Table 145 — Bus Mode Selection

Value	Bus Mode
15–7	Reserved
6	8 bit data bus (dual data rate)
5	4 bit data bus (dual data rate)
4–3	Reserved
2	8 bit data bus
1	4 bit data bus
0	1 bit data bus

HS_TIMING must be set to "0x1" before setting BUS_WIDTH for dual data rate operation (values 5 or 6)

7.4.68 ERASED_MEM_CONT [181]

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

Table 146 — Erased memory content values

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

7.4 Extended CSD register (cont'd)

7.4.69 PARTITION_CONFIG (before BOOT_CONFIG) [179]

This register defines the configuration for partitions.

Table 147 — Boot configuration bytes

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	BOOT_ACK	BOOT_PARTITION_ENABLE			PARTITION_ACCESS		
	R/W/E	R/W/E			R/W/E_P		

Bit 7: Reserved

Bit 6: BOOT_ACK (R/W/E)

0x0 : No boot acknowledge sent (default)

0x1 : Boot acknowledge sent during boot operation Bit

Bit[5:3] : BOOT_PARTITION_ENABLE (R/W/E)

User selects boot data that will be sent to master

0x0 : Device not boot enabled (default)

0x1 : Boot partition 1 enabled for boot

0x2 : Boot partition 2 enabled for boot

0x3–0x6 : Reserved

0x7 : User area enabled for boot

Bit[2:0] : PARTITION_ACCESS (before BOOT_PARTITION_ACCESS, R/W/E_P)

User selects partitions to access

0x0 : No access to boot partition (default)

0x1 : R/W boot partition 1

0x2 : R/W boot partition 2

0x3 : R/W Replay Protected Memory Block (RPMB)

0x4 : Access to General Purpose partition 1

0x5 : Access to General Purpose partition 2

0x6 : Access to General Purpose partition 3

0x7 : Access to General Purpose partition 4

7.4 Extended CSD register (cont'd)

7.4.70 BOOT_CONFIG_PROT[178]

This register defines boot configuration protection

Table 148 — Boot configuration protection

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved			PERM_BOOT_ CONFIG_PROT	Reserved			PWR_BOOT_ CONFIG_PROT
			R/W				R/W/C_P

Bit [7:5]: Reserved

Bit [4]: PERM_BOOT_CONFIG_PROT (R/W)

0x0 : PERM_BOOT_CONFIG_PROT is not enabled (default)

0x1 : Permanently disable the change of boot configuration register bits relating boot mode operation (BOOT_PARTITION_ENABLE, BOOT_ACK, RESET_BOOT_BUS_CONDITIONS, BOOT_MODE and BOOT_BUS_WIDTH).

Bit[3:1] : Reserved

Bit[0] : PWR_BOOT_CONFIG_PROT (R/W/C_P)

0x0 : PWR_BOOT_CONFIG_PROT is not enabled (default)

0x1 : Disable the change of boot configuration register bits relating to boot mode operation (BOOT_PARTITION_ENABLE, BOOT_ACK, RESET_BOOT_BUS_CONDITIONS, BOOT_MODE and BOOT_BUS_WIDTH) from at this point until next power cycle or next H/W reset operation (but not CMD0 Reset operation).

If PERM_BOOT_CONFIG_PROT is enabled, whether PWR_BOOT_CONFIG_PROT is enable or not, BOOT mode is permanently locked and cannot reversed.

7.4.71 BOOT_BUS_CONDITIONS [177]

This register defines the bus width for boot operation.

Table 149 — Boot bus configuration

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved			BOOT_MODE		RESET_BOOT_BUS_CONDITIONS	BOOT_BUS_WIDTH	

Bit[7:5] : Reserved

Bit [4:3] : BOOT_MODE (nonvolatile)

0x0 : Use single data rate + backward compatible timings in boot operation (default)

0x1 : Use single data rate + High Speed timings in boot operation mode

0x2 : Use dual data rate in boot operation

0x3 : Reserved

NOTE HS200 and HS400 is not supported during BOOT operation.

7.4 Extended CSD register (cont'd)

7.4.65 BOOT_BUS_CONDITIONS [177] (cont'd)

Bit [2]: RESET_BOOT_BUS_CONDITIONS (nonvolatile)

0x0 : Reset bus width to x1, single data rate and backward compatible timings after boot operation (default)
0x1 : Retain BOOT_BUS_WIDTH and BOOT_MODE values after boot operation. This is relevant to Push-pull mode operation only.

Bit[1:0] : BOOT_BUS_WIDTH (nonvolatile)

0x0 : x1 (sdr) or x4 (ddr) bus width in boot operation mode (default)
0x1 : x4 (sdr/ddr) bus width in boot operation mode
0x2 : x8 (sdr/ddr) bus width in boot operation mode
0x3 : Reserved

The settings in the BOOT_BUS_CONDITIONS register control any data transfer in boot mode, while the HS_TIMING and the BUS_WIDTH register control the behavior in other modes. The BOOT_BUS_CONDITIONS register is nonvolatile, while the HS_TIMING and BUS_WIDTH registers will be reset after a power cycle, hardware reset, or a CMD0 (unless sent to exit boot mode). This requires software to set the BUS_WIDTH and HS_TIMING registers after completing the boot procedure.

To avoid having to write these registers after completing the boot operation, the host may set the RESET_BOOT_BUS_CONDITIONS register to '1' which will automatically set the values of BUS_WIDTH and HS_TIMING registers to the values set in the BOOT_BUS_CONDITIONS register, when exiting boot mode. In this case, CMD0 to exit boot mode does not reset BUS_WIDTH and HS_TIMING. However, if CMD0 is sent in other cases (with any argument), it shall reset the BUS_WIDTH and HS_TIMING to their reset values (1-bit bus width and single data rate, backward-compatible interface timing respectively).

Any power cycle or hardware reset shall reset the BUS_WIDTH and HS_TIMING to their reset values (1-bit bus width and single data rate, backward-compatible interface timing respectively).

If the RESET_BOOT_BUS_CONDITIONS bit is set after boot operation, this will not affect bus width and interface timing until the device goes through the Boot state. Bus width and timing mode transitions are shown in Table 150.

Table 150 — Bus Width and Timing Mode Transition

RESET_BOOT_BUS_CONDITIONS bit	Event	BUS_WIDTH, HS_TIMING
1	Power cycle or hardware reset, entering boot mode	BOOT_BUS_WIDTH, BOOT_MODE
1	Power cycle or hardware reset, skipping boot mode	x1, SDR (backward compatibility)
1	CMD0, exiting boot mode	BOOT_BUS_WIDTH, BOOT_MODE
1	CMD0, anywhere but when exiting boot mode	x1, SDR (backward compatibility)
0	Any power cycle, hardware reset or CMD0	x1, SDR (backward compatibility)

7.4 Extended CSD register (cont'd)

7.4.72 ERASE_GROUP_DEF [175]

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

Table 151 — ERASE_GROUP_DEF

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

Bit[7:1]: Reserved

Bit0: ENABLE

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

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

7.4.73 BOOT_WP_STATUS [174]

This byte allows the host to read the current protection status of the boot partitions.

Table 152 — BOOT_WP_STATUS

Bit [7:4]	Bit [3:2]	Bit [1:0]
Reserved	B_AREA_2_WP	B_AREA_1_WP
	R	R

Bit[7:3]: Reserved

Bit[3:2]: B_AREA_2_WP (R)

0x00: Boot Area 2 is not protected

0x01: Boot Area 2 is Power on protected

0x10: Boot Area 2 is Permanently Protected

0x11: Reserved

Bit[1:0]**:** B_AREA_1_WP (R)

0x00: Boot Area 1 is not protected

0x01: Boot Area 1 is Power on protected

0x10: Boot Area 1 is Permanently Protected

0x11: Reserved

7.4 Extended CSD register (cont'd)

7.4.74 BOOT_WP [173]

This byte allows the host to apply permanent or power-on write protection to the boot area. Also, the register allows the master to disable either power-on or permanent write protection or both. The default state of the bits is zero. This register can only be written once per power cycle. The process for setting boot area write protection is outlined in 6.3.7.

Table 153 — BOOT area Partitions write protection

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
B_ _SEC_WP _SEL	B_PWR_WP_ DIS	Reserved	B_PERM_WP_ DIS	B_PER M_WP_ SEC_SE L	B_PERM_WP_E N	B_PWR_ WP_SEC _SEL	B_PWR_WP_ EN
R/W/C_P	R/W/C_P		R/W	R/W/C_ P	R/W	R/W/C_ P	R/W/C_P

Bit[7]: B_SEC_WP_SEL (R/W/C_P)

0x0: B_PERM_WP_EN(bit2) and B_PWR_WP_EN (bit 0) apply to both boot partitions and B_PERM_WP_SEC_SEL (bit 3) and B_PWR_WP_SEC_SEL (bit 1) have no impact

0x1: B_PERM_WP_EN(bit2) and B_PWR_WP_EN (bit 0) apply to only boot partition selected by B_PERM_WP_SEC_SEL (bit 3) and B_PWR_WP_SEC_SEL (bit 1) respectively.

NOTE Once a device has been set to enable protection on the boot partitions separately this cannot be reverted for a power cycle. This bit enables a mix of Permanent protect, power on protected boot partitions.

Bit[6]: B_PWR_WP_DIS (R/W/C_P)

0x0:Master is permitted to set B_PWR_WP_EN(bit 0)

0x1: Disable the use of B_PWR_WP_EN(bit 0). This bit must be zero if PWR_WP_EN is set.

Bit[5]: Reserved

Bit[4]: B_PERM_WP_DIS (R/W)

0x0: Master is permitted to set B_PERM_WP_EN(bit 2)

0x1: Permanently disable the use of B_PERM_WP_EN(bit 2). This bit must be zero if B_PERM_WP_EN is set. This bit has no impact on the setting of CSD[13].

Bit[3]: B_PERM_WP_SEC_SEL(R/W/C_P)

0x0: B_PERM_WP_EN(Bit 2) applies to boot Area1 only, if B_SEC_WP_SEL (bit 7 is set)

0x1: B_PERM_WP_EN(Bit 2) applies to boot Area2 only, if B_SEC_WP_SEL (bit 7 is set)

Bit[2]: B_PERM_WP_EN (R/W)

0x0: Boot region is not permanently write protected.

0x1: Boot region is permanently write protected. This bit must be zero if B_PERM_WP_DIS is set. When read, this bit only indicates if permanent protection has been set specifically for a boot region. How permanent protection has been applied depends on the setting of bits 7 and 3. To verify boot region protection read BOOT_WP_STATUS[174]. This bit may be zero if the whole device is permanently protected using CSD[13].

7.4 Extended CSD register (cont'd)

7.4.68 BOOT_WP [173] (cont'd)

Bit[1]: B_PWR_WP_SEC_SEL(R/W/C_P)

0x0: B_PWR_WP_EN(Bit 0) applies to boot Area1 only, if B_SEC_WP_SEL (bit 7 is set)

0x1: B_PWR_WP_EN(Bit 0) applies to boot Area2 only, if B_SEC_WP_SEL (bit 7 is set)

Bit[0]: B_PWR_WP_EN (R/W/C_P)

0x0: Boot region is not power-on write protected.

0x1: Enable Power-On Period write protection to the boot area. This bit must be zero if B_PWR_WP_DIS (bit 6) is set. When read, this bit only indicates if power on protection has been set specifically for a boot region. How power on protection has been applied depends on the setting of bits 7 and 1. To verify boot region protection read BOOT_WP_STATUS[174].

An attempt to set both the disable and enable bit for a given protection mode (permanent or power-on) in a single switch command will have no impact. If both permanent and power on protection are applied to the same partition(s), permanent protection will take precedence and the partition(s) will be permanently protected.

If the host enables Power-On write protection to a boot partition after enabling Permanent write protection to the other boot area, the host shall set Bit 3 as same value as set for Permanent write protection to the other boot partition (see A.11 for details).

7.4.75 USER_WP [171]

This byte allows the host to apply permanent or power-on write protection to all the partitions in the user area. Also, the register allows the host to disable the different protection modes that apply to the user area.

Table 154 — User area write protection

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERM_PSWD_DIS	CD_PERM_WP_DIS	Reserved	US_PERM_WP_DIS	US_PWR_WP_DIS	US_PERM_WP_EN	Reserved	US_PWR_WP_EN
R/W	R/W		R/W	R/W/C_P	R/W/E_P		R/W/E_P

Bit[7]: PERM_PSWD_DIS (R/W)

0x0: Password protection features are enabled.

0x1: Password protection features (ERASE (Forcing erase), LOCK, UNLOCK, CLR_PWD, SET_PWD) are disabled permanently.

Bit[6]: CD_PERM_WP_DIS (R/W)

0x0: Host is permitted to set PERM_WRITE_PROTECT (CSD[13]).

0x1: Disable the use of PERM_WRITE_PROTECT (CSD[13]).

Bit[5]: Reserved

Bit[4]: US_PERM_WP_DIS (R/W)

0x0: Permanent write protection can be applied to write protection groups.

0x1: Permanently disable the use of permanent write protection for write protection groups within all the partitions in the user area from the point this bit is set forward. Setting this bit does not impact areas that are already protected.

Bit[3]: US_PWR_WP_DIS (R/W/C_P)

0x0: Power-on write protection can be applied to write protection groups.

0x1: Disable the use of power-on period write protection for write protection groups within all the partitions in the user area from the point this bit is set until power is removed or a hardware reset occurs. Setting this bit does not impact areas that are already protected.

Bit[2]: US_PERM_WP_EN (R/W/E_P)

0x0: Permanent write protection is not applied when CMD28 is issued.

0x1: Apply permanent write protection to the protection group indicated by CMD28. This bit cannot be set if US_PERM_WP_DIS is set.

Bit[1]: Reserved

Bit[0]: US_PWR_WP_EN (R/W/E_P)

0x0: Power-on write protection is not applied when CMD28 is issued.

0x1: Apply Power-On Period protection to the protection group indicated by CMD28. This bit cannot be set if US_PWR_WP_DIS is set. This bit has not impact if US_PERM_WP_EN is set.

This field is set to zero after power on or hardware reset.

Issuing CMD28 when both US_PERM_WP_EN and US_PWR_WP_EN, will result in the write protection group being permanently protected.

7.4.76 FW_CONFIG [169]

The Update_Disable bit disables the possibility to update the firmware of the *e*•MMC.

Table 155 — FW Update Disable

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

Bit[7:1]: Reserved

Bit[0]: Update_Disable

0x0: FW updates enabled.

0x1: FW update disabled permanently

7.4.77 RPMB_SIZE_MULT [168]

The RPMB partition size is calculated from the register by using the following equation: RPMB partition size = 128kB x RPMB_SIZE_MULT

Table 156 — RPMB Partition Size

Value	Value definition
0x00	No RPMB partition available.
0x01	1 x 128Kbyte = 128Kbytes
0x02	2 x 128Kbyte = 256Kbytes
:	:
0x80	128x128Kbyte = 16384Kbytes
0x81-0xFF	Reserved[1]
NOTE 1 RPMB data frame format definition supports a maximum of 16 MB. This limits the size of the RPMB area.	

7.4.78 WR_REL_SET [167]

The write reliability settings register indicates the reliability setting for each of the user and general area partitions in the device.

Table 157 — Write reliability setting

Name	Field	Bit	Type
Write Data Reliability (user Area)	WR_DATA_REL_USR	0	R (if HS_CTRL_REL =0) R/W (if HS_CTRL_REL =1)
Write Data Reliability Partition 1	WR_DATA_REL_1	1	R (if HS_CTRL_REL =0) R/W (if HS_CTRL_REL =1)
Write Data Reliability Partition 2	WR_DATA_REL_2	2	R (if HS_CTRL_REL =0) R/W (if HS_CTRL_REL =1)
Write Data Reliability Partition 3	WR_DATA_REL_3	3	R (if HS_CTRL_REL =0) R/W (if HS_CTRL_REL =1)
Write Data Reliability Partition 4	WR_DATA_REL_4	4	R (if HS_CTRL_REL =0) R/W (if HS_CTRL_REL =1)
Reserved		7:5	

Bit[7:5]: Reserved Bit

Bit[4]: WR_DATA_REL_4

0x0: In general purpose partition 4, the write operation has been optimized for performance and existing data in the partition could be at risk if a power failure occurs.

0x1: In general purpose partition 4, the device protects previously written data if power failure occurs.

Bit[3]: WR_DATA_REL_3

0x0: In general purpose partition 3, the write operation has been optimized for performance and existing data in the partition could be at risk if a power failure occurs.

0x1: In general purpose partition 3, the device protects previously written data if power failure occurs.

Bit[2]: WR_DATA_REL_2

0x0: In general purpose partition 2, the write operation has been optimized for performance and existing data in the partition could be at risk if a power failure occurs.

0x1: In general purpose partition 2, the device protects previously written data if power failure occurs.

Bit[1]: WR_DATA_REL_1

0x0: In general purpose partition 1, the write operation has been optimized for performance and existing data in the partition could be at risk if a power failure occurs.

0x1: In general purpose partition 1, the device protects previously written data if power failure occurs.

Bit[0]: WR_DATA_REL_USR

0x0: In the main user area, write operations have been optimized for performance and existing data could be at risk if a power failure occurs.

0x1: In the main user area, the device protects previously written data if power failure occurs.

7.4.79 WR_REL_PARAM [166]

On this specification write reliability shall be supported. Both bits in this register should be set to 1.

Table 158 — Write reliability parameter register

Name	Field	Bit	Type
Host controlled data reliability	HS_CTRL_REL	0	R
Reserved		1	
Enhanced Reliable Write	EN_REL_WR	2	R
Reserved		3	
Enhanced RPMB Reliable Write	EN_RPMB_REL_WR	4	R
Reserved		7:5	

Bit[7:5]: Reserved

Bit[4]: EN_RPMB_REL_WR (R)

0x0: RPMB transfer size is either 256B (single 512B frame) or 512B (two 512B frame).

0x1: RPMB transfer size is either 256B (single 512B frame), 512B (two 512B frame), or 8KB (Thirty two 512B frames).

Bit[3]: Reserved

Bit[2]: EN_REL_WR (R)

0x0: obsolete

0x1: The device supports the enhanced definition of reliable write

Bit[1]: Reserved

Bit[0]: HS_CTRL_REL (R)

0x0: obsolete

0x1: All the WR_DATA_REL parameters in the WR_REL_SET registers are R/W.

7.4.80 SANITIZE_START[165]

Writing any value to this field shall manually start a sanitize operation. Device shall stay busy until sanitize is complete.

7.4.81 BKOPS_START [164]

Writing any value to this field shall manually start background operations. Device shall stay busy till no more background operations are needed.

7.4.82 BKOPS_EN [163]

This field allows the host to indicate to the device if it is expected to periodically manually start background operations by writing to the BKOPS_START field.

Table 159 — Background operations enable

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved						AUTO_EN	MANUAL_EN

Bit[7:2]: Reserved

Bit[1]: AUTO_EN (R/W/E) default value is vendor specific.

0b: Device shall not perform background operations while not servicing the host.

1b: Device may perform background operations while not servicing the host.

Bit[0]: MANUAL_EN (R/W)

- 0x0: Host does not support background operations handling and is not expected to write to BKOPS_START field.
- 0x1: Host is indicating that it shall periodically write to BKOPS_START field to manually start background operations.

7.4.83 RST_n_FUNCTION [162]

For backward compatibility reason, RST_n signal is temporary disabled in device by default. Host may need to set the signal as either permanently enable or permanently disable before it uses the Device.

Table 160 — H/W reset function

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

Bit[7:2]: Reserved

Bit[1:0]: RST_n_ENABLE (Readable and Writable once)

0x0: RST_n signal is temporarily disabled (default)

0x1: RST_n signal is permanently enabled

0x2: RST_n signal is permanently disabled

0x3: Reserved

By default, RST_n_ENABLE is set to 0x0, meaning RST_n is temporarily disabled. Host can change the value to either 0x1 (permanently enabled) or 0x2 (permanently disabled). Once host sets the value to either one, the value cannot be changed again.

Once host sets RST_n_ENABLE bits to 0x2 (permanently disabled), the Device will not accept the input of RST_n signal permanently. During the disable period, the Device has to take care that any state of RST_n (high, low and floating) will not cause any issue (i.e., mal function and high leakage current in the input buffer) in the device.

When RST_n_ENABLE is set to 0x1 (permanently enabled), the Device accepts the input of RST_n permanently. Host cannot change the bits back to the disabled values. Also, when host set RST_n_ENABLE to 0x1, the Device must not start resetting internal circuits by triggering the register bit change. Internal reset sequence must be triggered by RST_n rising edge but not by the register change.

Since Device does not have any internal pull up or pull down resistor on RST_n terminal, host has to pull up or down RST_n to prevent the input circuits from flowing unnecessary leakage current when RST_n is enabled.

7.4.84 HPI_MGMT [161]

This field allows the host to activate the HPI mechanism.

Table 161 — HPI management

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

Bit[7:1]: Reserved

Bit[0]: HPI_EN

0x0 : HPI mechanism not activated by the host (default)

0x1 : HPI mechanism activated by the host

7.4.85 PARTITIONING_SUPPORT [160]

This register defines supported partition features. This standard requires that all values in this register are set.

Table 162 — Partitioning Support

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved					EXT_ATTRIBUTE_EN	ENH_ATTRIBUTE_EN	PARTITIONING_EN

Bit[7:3]: Reserved

Bit[2]: EXT_ATTRIBUTE_EN

0x0: n/a.

0x1: Device can have extended partitions attribute

Bit[1]: ENH_ATTRIBUTE_EN

0x0: obsolete

0x1: Device can have enhanced technological features in partitions and user data area

Bit[0]: PARTITIONING_EN

0x0: obsolete.

0x1: Device supports partitioning features

7.4.86 MAX_ENH_SIZE_MULT [159:157]

This register defines maximum amount of memory area that can have the enhanced attribute. The Write Protect Group size refers to the high capacity definition.

Table 163 — Max. Enhanced Area Size

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MAX_ENH_SIZE_MULT_2							
MAX_ENH_SIZE_MULT_1							
MAX_ENH_SIZE_MULT_0							

Max Enhanced Area = MAX_ENH_SIZE_MULT x HC_WP_GRP_SIZE x HC_ERASE_GRP_SIZE x 512kBytes

$\sum$ Enhanced general partition Size(i) + Enhanced user data area $\leq$ Max Enhanced Area

7.4.87 PARTITIONS_ATTRIBUTE [156]

This register bits sets enhanced attribute in general purpose partitions.

Table 164 — Partitions Attribute

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved			ENH_4	ENH_3	ENH_2	ENH_1	ENH_USR

Bit[7:5]: Reserved

Bit[4]: ENH_4

0x0: Default

0x1: Set Enhanced attribute in General Purpose partition 4

Bit[3]: ENH_3

0x0: Default

0x1: Set Enhanced attribute in General Purpose partition 3

Bit[2]: ENH_2

0x0: Default

0x1: Set Enhanced attribute in General Purpose partition 2

Bit[1]: ENH_1

0x0: Default

0x1: Set Enhanced attribute in General Purpose partition 1

Bit[0]: ENH_USR

0x0: Default

0x1: Set Enhanced attribute in User Data Area

7.4.88 PARTITION_SETTING_COMPLETED [155]

Default value states that any partitions configuration procedure has been issued by the host. The bit is set to notify the device that the definition of parameters has been completed and the device can start its internal configuration activity. If a sudden power loss occurs and this bit has not been set yet, the configuration of partitions shall be invalidated and must be repeated.

Table 165 — Partition Setting

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

7.4.89 GP_SIZE_MULT_GP0 - GP_SIZE_MULT_GP3 [154:143]

This register defines general purpose partition size. General Purpose Partitions size shall be expressed in terms of high capacity write protect groups.

Table 166 — General Purpose Partition Size

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GP_SIZE_MULT_X_2							
GP_SIZE_MULT_X_1							
GP_SIZE_MULT_X_0							

GP_SIZE_MULT_X_Y

Where X refers to the General Purpose Partition (from 1 to 4) and Y refers to the factors in the formula (from 0 to 2), so;

General_Purpose_Partition_X Size =

$(GP\_SIZE\_MULT\_X\_2 \times 2^{16} + GP\_SIZE\_MULT\_X\_1 \times 2^{8} + GP\_SIZE\_MULT\_X\_0 \times 2^{0}) \times HC\_WP\_GRP\_SIZE \times HC\_ERASE\_GRP\_SIZE \times 512kBytes$

GPP1:

- GP_SIZE_MULT_1_0 = EXT_CSD[143]
- GP_SIZE_MULT_1_1 = EXT_CSD[144]
- GP_SIZE_MULT_1_2 = EXT_CSD[145]

GPP2:

- GP_SIZE_MULT_2_0 = EXT_CSD[146]
- GP_SIZE_MULT_2_1 = EXT_CSD[147]
- GP_SIZE_MULT_2_2 = EXT_CSD[148]

GPP3:

- GP_SIZE_MULT_3_0 = EXT_CSD[149]
- GP_SIZE_MULT_3_1 = EXT_CSD[150]
- GP_SIZE_MULT_3_2 = EXT_CSD[151]

GPP4:

- GP_SIZE_MULT_4_0 = EXT_CSD[152]
- GP_SIZE_MULT_4_1 = EXT_CSD[153]
- GP_SIZE_MULT_4_2 = EXT_CSD[154]

7.4.90 ENH_SIZE_MULT [142:140]

This register defines enhanced user data area size. Enhanced User Data Area size is defined in terms of High Capacity Write Protect Groups.

Table 167 — Enhanced User Data Area Size

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ENH_SIZE_MULT_2							
ENH_SIZE_MULT_1							
ENH_SIZE_MULT_0							

Enhanced User Data Area x Size =

$$(ENH\_SIZE\_MULT\_2 \times 2^{16} + ENH\_SIZE\_MULT\_1 \times 2^{8} + ENH\_SIZE\_MULT\_0 \times 2^{0}) \times HC\_WP\_GRP\_SIZE \times HC\_ERASE\_GRP\_SIZE \times 512kBytes$$

7.4.91 ENH_START_ADDR [139:136]

This register defines starting address of the enhanced user data area.

Start address of the Enhanced User Data Area segment in the User Data Area (expressed in bytes or in sectors in case of high capacity devices).

Table 168 — Enhanced User Data Start Address

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ENH_START_ADDR_3							
ENH_START_ADDR_2							
ENH_START_ADDR_1							
ENH_START_ADDR_0							

7.4.92 SEC_BAD_BLK_MGMNT [134]

In some memory array technologies that are used for *e*•MMC portions of the memory array can become defective with use. In these technologies the Device will recover the information from the defective portion of the memory array before it retires the block. This register bit, when set, requires the device to perform an erase on the contents of the defective region before it is retired. This feature requires only those bits that are not defective in the region to be erased.

Table 169 — Secure Bad Block management

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

Bit[7:1]: Reserved

Bit[0]: SEC_BAD_BLK (R/W)

0x0: (Default) Feature disabled

0x1: All data must be erased from defective memory array regions before they are retired from use. SEC_BD_BLK_EN (EXT_CSD[231] bit 2) must be set in order to use this bit.

7.4.93 PRODUCTION_STATE_AWARENESS [133]

Through this field the host reports to the device its production state.

This field has the following values:

- NORMAL (Field) – This value represents a state in which the device is the field and the device uses "regular" operations.
- PRE_SOLDERING_ WRITES – This value represents a state in which the device is in production prior soldering and before the host loaded content to the device. The host sets the device to this state for loading the content to the device.
- PRE_SOLDERING_POST_WRITES - This value represents a state in which the device is in production and the host completed to load the content to the device. The host sets the device to this state after content was loaded and just before soldering. Once transferred to this state the host should not write content to the device.
- AUTO_PRE_SOLDERING –This value should be set by the host if auto pre-soldering data is desired. If the data is transferred as much as PRE_LOADING_DATA_SIZE, then the device state is changed back to normal state by changing the value of PRODUCTION_STATA_AWARENESS to 0x0(Normal) automatically. i.e., no separate command to change to normal state is needed.

Table 170 — PRODUCTION_STATE_AWARENESS states

Value	Value definition
0x00	NORMAL(Field)
0x01	PRE_SOLDERING_WRITES
0x02	PRE_SOLDERING_POST_WRITES
0x03	AUTO_PRE_SOLDERING
0x04 – 0x0F	Reserved
0x10 – 0x1F	Reserved for Vendor Proprietary Usage
0x20 – 0xFF	Reserved

7.4.94 TCASE_SUPPORT [132]

e•MMC device may condition their maximum performance to cases where the host conforms to the T_{case} control temperature tables as given in Annex A.10. In that case host that wish to utilize the maximum performance and conforms to the given Case temperature (T_c) tables shall set the TCASE_SUPPORT bits accordingly:

- TCASE_SUPPORT = 0x01: Table A.227 in Annex A.10 is supported. Heat relief through Case only is assumed.
- TCASE_SUPPORT = 0x10: Table A.227 in Annex A.10 is supported. Heat relief through Case and PCB/Balls is assumed.

If TCASE_SUPPORT bit is ="0x00" the above mentioned device may limit the maximum available performance. Host should set the TCASE_SUPPORT value as 0x01 or 0x10 as soon as possible for the device to run its maximum available performance.

7.4.95 PERIODIC_WAKEUP [131]

How often the host shall wake up the device.

Table 171 — PERIODIC_WAKEUP

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WAKEUP_UNIT 0x0 = infinity (no wakeups) 0x1 = months 0x2 = weeks 0x3 = days 0x4 = hours 0x5 = minutes 0x6, 0x7 reserved			WAKEUP_PERIOD				

Where the period between wakeups is WAKEUP_PERIOD in units of WAKEUP_UNIT. For example, a value of 01000110b means: WAKEUP_UNIT=010b=weeks, WAKEUP_PERIOD=00110b=6 => 6 weeks.

If WAKEUP_UNIT is 0, WAKEUP_PERIOD is ignored and the period between wake ups is infinity (no wake ups).

7.4.96 PROGRAM_CID_CSD_DDR_SUPPORT [130]

This field indicates if the CMD26 and CMD27 are supported in dual data rate mode by the device.

Table 172 — CMD26 and CMD27 in DDR mode Support

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved							PROGRAM_CID_CSD_DDR_ SUPPORT

Bit[7:1]: Reserved

Bit[0]: PROGRAM_CID_CSD_DDR_SUPPORT (R)

0x0: (Default) CMD26 and CMD27 must be used in single data rate mode.

0x1: CMD26 and CMD27 are considered legal in both single data rate and dual data rate mode.

7.4.97 VENDOR_SPECIFIC_FIELD [127:64]

The purpose and type of these fields are reserved for definition by the device manufacturer.

7.4.98 NATIVE_SECTOR_SIZE [63]

This field indicates the native size of sectors supported by the device:

0x00:Native sector size is 512B

0x01:Native sector size is 4KB

0x02-0xFF: Reserved

7.4.99 USE_NATIVE_SECTOR [62]

This field controls if sector size of 512B is emulated on a native sector size other than 512B:

0x00:Device is emulating a 512B sector size or uses a native 512B sector size

0x01:Device is using the larger than 512B native sector size

0x02-0xFF: Reserved

When shipped, a large sector device is always configured with USE_NATIVE_SECTOR=0. If device supports larger than 512B native sector size, host may disable the emulation mode by writing 0x01 followed by a power cycle. Emulation mode cannot be re-enabled.

If NATIVE_SECTOR_SIZE is 512B, writing this field shall fail with SWITCH_ERROR.

7.4.100 DATA_SECTOR_SIZE [61]

This field indicates the current sector size that can be accessed or addressed:

0x00:Data sector size is 512B

0x01:Data sector size is 4KB

0x02-0xFF: Reserved

7.4.101 INI_TIMEOUT_EMU [60]

This register indicates the maximum initialization timeout during the first power up after successful disabling of the 512B emulation mode.

Table 173 — Initialization Time out value

Value	Timeout Value
0x00	Not defined
0x01	100ms × 1 = 100 ms
...	...
0xFF	100ms × 255 = 25500 ms

7.4.102 CLASS_6_CTRL[59]

This field controls the usage of class 6 command set (CMD28, CMD29, CMD30 and CMD31). By setting this field to 0x00, class 6 command set is used for WP. By setting this field to 0x1, class 6 command set is used to manipulate the dynamic capacity functionality. Setting any other value (0x02-0xFF) is forbidden.

Table 174 — Class 6 usage

Value	Value definition
0x00	Write Protect (Default)
0x01	Dynamic Capacity
0x02-0xFF	Reserved

7.4.103 DYNCAP_NEEDED [58]

This field is a read only field through which the device indicates to the host the amount of WP-Groups that the device requests to be released from the user area address space.

7.4.104 EXCEPTION_EVENTS_CTRL [57:56]

Each bit enables the use of the relevant exception event bit, allowing it to raise the EXCEPTION_EVENT bit in Device Status.

Table 175 — EXCEPTION_EVENTS_CTRL[56]

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	Reserved	Reserved	EXTENDED_SECURITY_EN	PACKED_EVENT_EN	SYSPOOL_EVENT_EN	DYNCAP_EVENT_EN	Reserved

Table 176 — EXCEPTION_EVENTS_CTRL[57]

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reserved							

NOTE For backward compatibility reasons, if BKOPS_SUPPORT bit [0] is set, then the urgent background operations event (URGENT_BKOPS) is always enabled and cannot be disabled. All other enable bits start disabled after power up until set by host.

7.4.105 EXCEPTION_EVENTS_STATUS [55:54]

Each bit reports the status of a specific exception event.

Table 177 — EXCEPTION_EVENTS_STATUS[54]

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	Reserved	Reserved	EXTENDED_SECURITY_FAILURE	PACKED_FAILURE	SYSPOOL_EXHAUSTED	DYNCAP_NEEDED	URGENT_BKOPS

Table 178 — EXCEPTION_EVENTS_STATUS[55]

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reserved							

- URGENT_BKOPS – Urgent background operations needed: if set, the device needs to perform background operations urgently. Host can check EXT_CSD field BKOPS_STATUS for the detailed level.
- DYNCAP_NEEDED – Dynamic capacity needed: If set, device needs some capacity to be released.
- SYSPOOL_EXHAUSTED – System resources pool exhausted: If set, system resources pool has no more available resources and some data needs to be untagged before other data can be tagged
- PACKED_FAILURE – Packed command failure: If set, the last packed command has failed. Host may check EXT_CSD field PACKED_COMMAND_STATUS for the detailed cause.
- EXTENDED_SECURITY_FAILURE – An error caused while using the PROTOCOL_WR (CMD54) or PROTOCOL_RD (CMD53) commands or in relation to the Extended Protocols usage. If set the host may check the EXT_CSD field EXT_SECURITY_ERR [505] for the detailed cause.

7.4.106 EXT_PARTITIONS_ATTRIBUTE [53:52]

This register bits sets extended attribute in general purpose partitions.

Table 179 — First Byte EXT_PARTITIONS_ATTRIBUTE[52]

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EXT_2				EXT_1			

Table 180 — Second Byte EXT_PARTITIONS_ATTRIBUTE[53]

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
EXT_4				EXT_3			

Each four-bits nibble describes the extended attribute of a specific general purpose partition. The specific meaning of each type follows:

0x0: Default (no extended attribute)
0x1: System code
0x2: Non-persistent
0x3-0xF: Reserved

Bit[15:12]: Extended partition attribute for general purpose partition 4

Bit[11:8]:Extended partition attribute for general purpose partition 3

Bit[7:4]:Extended partition attribute for general purpose partition 2

Bit[3:0]: Extended partition attribute for general purpose partition 1

7.4.107 CONTEXT_CONF [51:37]

CONTEXT_CONF is an array of 15 bytes, each controlling the configuration of the relevant ID, starting with ID #1 associated to CONTEXT_CONF[37] up to ID #15 associated to CONTEXT_CONF[51]. Context ID #0 is reserved for context-less operation and has no configuration register.

Each configuration register holds the following format:

Table 181 — CONTEXT_CONF configuration format

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reliability Mode		Large Unit Multiplier			Large Unit Context	Config direction and activate	

Bit[7:6]: Reliability mode

0x0: MODE0 (normal)

0x1: MODE1 (non-Large Unit, reliable mode or Large Unit unit-by-unit mode)

0x2: MODE2 (Large Unit, one-unit-tail mode)

0x3: Reserved

Bit[5:3]: Large Unit multiplier

If Large Unit context is set, then the unit is multiplied by (Bit[5:3] + 1),
else it is ignored

Bit[2]: Large Unit context

0x0: Context is not following Large Unit rules

0x1: Context follows Large Unit rules

Bit [1:0]: Activation and direction selection

00b: Context is closed and is no longer active

01b: Context is configured and activated as a write-only context and according to the rest of the bits in this configuration register

10b: Context is configured and activated as a read-only context and according to the rest of the bits in this configuration register

11b: Context is configured and activated as a read/write context and according to the rest of the bits in this configuration register

7.4.108 PACKED_COMMAND_STATUS [36]

This register reports the status of the last packed command.

Table 182 — Packed Command Status Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Indexed Error	Error

In case any error occurs during a packed command, the ‘Error’ bit (bit 0) shall be set.

If the error is a result of one of the individual commands inside the packed command, its index is reported in PACKED_FAILURE_INDEX [35] and ‘Indexed Error’ bit (bit 1) is set as well.

7.4.109 PACKED_FAILURE_INDEX [35]

If the ‘Indexed Error’ bit (bit 1) in PACKED_COMMAND_STATUS is set, this field specifies the index in the header of the failed command.

7.4.110 POWER_OFF_NOTIFICATION [34]

This field allows host to notify the device before the device is powered off. Values not in Table 183 are invalid and setting them will result in SWITCH_ERROR.

NOTE *e•*MMC device should be able to guard against sudden power loss even when POWER_OFF_NOTIFICATION is set to 0x01 (POWER_ON) since unintentional power loss event may still occur.

Table 183 — Valid POWER_OFF_NOTIFICATION values

Value	Name	Description
0x00	NO_POWER_NOTIFICATION	Power off notification is not supported by host, device shall not assume any notification
0x01	POWERED_ON	Host shall notify before powering off the device, and keep power supplies alive and active until then
0x02	POWER_OFF_SHORT	Host is going to power off the device, The device shall respond within GENERIC_CMD6_TIME.
0x03	POWER_OFF_LONG	Host is going to power off the device The device shall respond within POWER_OFF_LONG_TIME.
0x04	SLEEP_NOTIFICATION	Host is going to put the device in Sleep Mode. The device shall respond within SLEEP_NOTIFICATION_TIME

7.4.111 CACHE_CTRL [33]

The cache shall be turned ON and OFF by writing the CACHE_EN bit. The status of CACHE_CTRL can be read from this byte.

Bit [7:1]: Reserved

Bit [0]: CACHE_EN

0x0: Cache is OFF

0x1: Cache is ON

Table 184 — CACHE ENABLE

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

7.4.112 FLUSH_CACHE [32]

The settings in FLUSH_CACHE controls the timing when data will be flushed to the nonvolatile storage.

A barrier command is issued by setting BARRIER bit; All data cached before the barrier shall be flushed to the nonvolatile memory before any request after the barrier command.

Data in the cache shall be flushed to the nonvolatile storage by setting the FLUSH bit.

Bit [7:2]: Reserved

Bit [1]: BARRIER

0x0: Reset value

0x1: Set barrier

Bit [0]: FLUSH

0x0: Reset value

0x1: Triggers the flush

Table 185 — FLUSH CACHE

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved						BARRIER	FLUSH

7.4.113 BARRIER_CTRL [31]

The Barrier feature shall be turned ON and OFF by writing the BARRIER_EN bit. The status of BARRIER_CTRL can be read from this byte.

Bit [7:1]: Reserved

Bit [0]: BARRIER_EN

0x0: Barrier feature is OFF

0x1: Barrier feature is ON

Table 186 — BARRIER_CTRL

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

7.4.114 MODE_CONFIG [30]

Using this field the host can change the mode of the device.

Table 187 — Valid MODE_CONFIG values

Value	Name	Description
0x00	Normal Mode	To keep the compatibility
0x01	FFU Mode	
0x10	Vendor Specific Mode	
Others	Reserved	

7.4.115 MODE_OPERATION_CODES [29]

Using this field the host sets the operation to be performed at the selected mode.

In case MODE_CONFIG is set to FFU Mode, MODE_OPERATION_CODES could have the following values:

Table 188 — Valid MODE_OPERATION_CODES values

Value	Name	Description
0x00	Reserved	
0x01	FFU_INSTALL	
0x02	FFU_ABORT	
Others	Reserved	

7.4.116 FFU_STATUS [26]

Using this field the device reports the status of the FFU process

Table 189 — FFU Status codes

Value	Description
0x00	Success
0x01 – 0xF	Reserved
0x10	General error
0x11	Firmware install error
0x12	Error in downloading firmware
Others	Reserved

7.4.117 PRE_LOADING_DATA_SIZE [25-22]

This filed is used to set the size of the contents to be loaded on to the device during pre-loading.

PRE_LOADING_DATA_SIZE represents the number of sectors which the host is expected to write to all partitions in the pre-soldering state.

Pre_Loading_Data_Size = PRE_LOADING_DATA_SIZE * Sector Size

If PRE_LOADING_DATA_SIZE is 0xFFFFFFFF, the ranges that can be covered by the parameter are:.

- If Sector size is 512 bytes - Pre_Loading_Data_Size= (2^32 -1) * 512B = 2TB
- If Sector size is 4K bytes - Pre_Loading_Data_Size= (2^32 -1) * 4KB = 16TB

NOTE Host should set the value of PRE_LOADING_DATA_SIZE not to exceed the value of MAX_ PRE_LOADING_DATA_SIZE. In case the value of the PRE_LOADING_DATA_SIZE exceeds the value of the MAX PRE_LOADING_DATA_SIZE, device shall generate switch error and the value will not be set.

7.4.118 MAX_PRE_LOADING_DATA_SIZE [21-18]

This field indicates the maximum data size that can be loaded to the device. This value shall be set by the device vendor.

Max_Pre_Loading_Data_Size = MAX PRE_LOADING_DATA_SIZE * Sector Size

If MAX_PRE_LOADING_DATA_SIZE is 0xFFFFFFFF, the ranges that can be covered by the parameter are:

- If Sector size is 512 bytes – Max Pre_Loading_Data_Size= (2^32 -1) * 512B = 2TB
- If Sector size is 4K bytes – Max Pre_Loading_Data_Size= (2^32 -1) * 4KB = 16TB

NOTE For RPMB partition each data frame which contains 256 bytes of data is counted in PRE_LOADING_DATA_SIZE as one sector.

7.4.119 PRODUCT_STATE_AWARENESS_ENABLEMENT [17]

This field is composed of 2 sub-fields:

- Capabilities [0:3] used to indicate the capabilities of product state awareness. Host should refer this value to decide which mode can be used by the host vendor. This sub-field is read only. In case the host enables a mode which is not supported by the device a switch error shall be returned by the device.
  - Bit 0 – '1' indicates that Manual mode is supported by the device
  - Bit 1 - '1' indicates that Auto mode is supported by the device
- Enablement [4:7] Host will enable the product state awareness feature by setting this register
  - Bit 4
    - '0' - Production State Awareness is disabled.
    - '1' - Production State Awareness is enabled.
  - Bit 5
    - '0' – Manual mode is enabled.
    - '1' – Auto mode is enabled.

NOTE Bit 5 is ignored by the device when bit 4 is set to '0'.

Table 190 — Production State Awareness Enablement

	Enablement (R/W/E)				**Capabilities (R)**			
Bit #	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Function	Reserved		Mode	Production State Awareness enable (2)	Reserved		Auto mode Supported	Manual mode Supported
Clear Condition	(1)				Never			

NOTE 1 Cleared when PRODUCTION_STATE_AWARENESS is changed to Normal (either automatically or by setting PRODUCTION_STATE_AWARENESS to Normal)

NOTE 2 This bit could be set to '1' only once.

7.4.120 SECURE_REMOVAL_TYPE [16]

This field indicates how information is removed from the physical memory during a Purge operation. The first 4 bits (Bit0~Bit3) indicate the capability of the *e*•MMC device. The next two bits indicate configurable option. It can be set once during system integration by host.

Table 191 — Secure Removal Type

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved		Configure Secure Removal Type		Supported Secure Removal Type			
		R/W		R			

Supported Secure Removal Type

Bit [7:6]: Reserved

Bit [5:4]: Configure Secure Removal Type

0x0: information removed by an erase of the physical memory
0x1: information removed by an overwriting the addressed locations with a character followed by an erase
0x2: information removed by an overwriting the addressed locations with a character, its complement, then a random character
0x3: information removed using a vendor defined

Bit [3:0]: Supported Secure Removal Type

Bit 0: information removed by an erase of the physical memory
Bit 1: information removed by an overwriting the addressed locations with a character followed by an erase
Bit 2: information removed by an overwriting the addressed locations with a character, its complement, then a random character
Bit 3: information removed using a vendor defined.

7.4.121 CMDQ_MODE_EN [15]

This field is used by the host enable command queuing mechanism if supported by the device.

Table 192 — Command Queue Mode Enable

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved							CMDQ Mode En

Bit encoding:

- [7:1]: Reserved
- [0]: Command queuing enable:
  - 0: Command queuing is disabled. Host should clear the queue empty using CMD48 prior disabling the queue.
  - 1: Command queuing is enabled

To maintain backward compatibility with hosts, which do not support command queuing, when the command queuing is disabled other functionality of the device is as if the device does not support command queuing.

7.5 RCA register

The writable 16-bit relative Device address (RCA) register carries the Device address assigned by the host during the Device identification. This address is used for the addressed host-Device communication after the Device identification procedure. The default value of the RCA register is 0x0001. The value 0x0000 is reserved to set all Devices into the *Stand-by State* with CMD7.

7.6 DSR register

The 16-bit driver stage register (DSR) is described in detail in 10.2. It can be optionally used to improve the bus performance for extended operating conditions (depending on parameters like bus length, transfer rate or number of Devices). The CSD register carries the information about the DSR register usage. The default value of the DSR register is 0x404.

7.7 QSR

The 32-bit Queue Status Register (QSR) carries the state of tasks in the queue at a specific point in time. The host may read this register through device response to SEND_QUEUE_STATUS command (CMD13 with bit [15]="1"), R1's argument will be the 32-bit Queue Status Register (QSR). Each bit in the QSR represents the task who's ID corresponds to the bit index. If bit QSR[*i*] = "0", then the queued task with a Task ID *i* is not ready for execution. It is the host's responsibility to track the state of tasks so it can determine if the task is queued and pending, or the Task ID is unused. If bit QSR[*i*] = "1", then the queued task with Task ID *i* is ready for execution.

7.8 Authenticated Device Configuration Area

7.8.1 Authenticated Device Configuration Area[1] : SECURE_WP_MODE_ENABLE

The byte is to enter/exit the secure write protection mode.

Table 193 — SECURE_WP_MODE_ENABLE

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reserved							SECURE_WP_EN

If host want a device to enter the secure Write Protection mode, host set the SECURE_WP_EN bit as '0x1' in this register using Authenticated Device Configuration Write request. This register can be read using Authenticated Device Configuration Read request. If there are already write protected groups or write protected boot partitions, those will be preserved when entering or exiting secure Write protected mode.

Bit[7:1] : Reserved

Bit[0] : SECURE_WP_EN (R/W/E)

The default value of this field is 0x0.

0x0 : Legacy Write Protection mode, i.e., TMP_WRITE_PROTECT[12] , PERM_WRITE_PROTECT[13] is updated by CMD27. USER_WP[171], BOOT_WP[173] and BOOT_WP_STATUS[174] are updated by CMD6.

0x1 : Secure Write Protection mode. The access to the write protection related EXT_CSD and CSD fields depends on the value of SECURE_WP_MASK bit in SECURE_WP_MODE_CONFIG field.

7.8.2 Authenticated Device Configuration Area[2] : SECURE _WP_MODE_CONFIG

In secure write protected mode, the updatability of USER_WP[171], BOOT_WP[173], TMP_WRITE_PROTECT[12] and PERM_WRITE_PROTECT[13] are controlled by this mask value.

Table 194 — SECURE _WP_MODE_CONFIG

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reserved							SECURE_WP_MASK

Bit[7:1] : Reserved

Bit[0] : SECURE_WP_MASK (R/W/E_P)

The default value of this field is 0x0.

- 0x0: Disabling updating WP related EXT_CSD and CSD fields. CMD27 (Program CSD) will generate generic error for setting TMP_WRITE_PROTECT[12] , PERM_WRITE_PROTECT[13]. CMD6 for updating USER_WP[171], BOOT_WP[173] and BOOT_WP_STATUS[174] generates SWITCH_ERROR. If a force erase command is issued, the command will fail (Device stays locked) and the LOCK_UNLOCK_FAILED error bit will be set in the status register. If CMD28 or CMD29 is issued, then generic error will be occurred. Power-on Write Protected boot partitions will keep protected mode after power failure, H/W reset assertion and any CMD0 reset. The device keeps the current value of BOOT_WP_STATUS in the EXT_CSD register to be same after power cycle, H/W reset assertion, and any CMD0 reset.
- 0x1: Enabling updating WP related EXT_CSD and CSD fields. I.e TMP_WRITE_PROTECT[12] , PERM_WRITE_PROTECT[13] , USER_WP[171], BOOT_WP[173] and BOOT_WP_STATUS[174] are accessed using CMD6, CMD8 and CMD27. If a force erase command is issued and accepted, then ALL THE DEVICE CONTENT WILL BE ERASED including the PWD and PWD_LEN register content and the locked Device will get unlocked. If a force erase command is issued and power-on protected or a permanently-write-protected write protect groups exist on the device, the command will fail (Device stays locked) and the LOCK_UNLOCK_FAILED error bit will be set in the status register. An attempt to force erase on an unlocked Device will fail and LOCK_UNLOCK_FAILED error bit will be set in the status register. Write Protection is applied to the WPG indicated by CMD28 with the WP type indicated by the bit[2] and bit[0] of USER_WP[171]. All temporary WP Groups and power-on Write Protected boot partitions become writable/erasable temporarily which means write protect type is not changed. All power-on and permanent WP Groups in user area will not become writable/erasable temporarily. Those temporarily writable/erasable area will become write protected when this bit is cleared to 0x0 by the host or when there is power failure, H/W reset assertion and any CMD0 reset. The device keeps the current value of BOOT_WP_STATUS in the EXT_CSD register to be same after power cycle, H/W reset assertion, and any CMD0 reset.

8 Error protection

The CRC is intended for protecting *e*•MMC commands, responses and data transfer against transmission errors on *e*•MMC bus. One CRC is generated for every command and checked for every response on the CMD line. For data blocks one CRC per transferred block is generated.

8.1 Error correction codes (ECC)

In order to detect data defects on the Devices 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 Device, 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 *e*•MMC adapter.

If an *e*•MMC requires external error correction (external means outside of the Device), then an ECC algorithm has to be implemented in the *e*•MMC host. The DEFAULT_ECC field in the CSD register defines the recommended ECC algorithm for the Device.

The shortened BCH (542,512) code was chosen for matching the requirement of having high efficiency at lowest costs. Table 195 gives a brief overview of this code.

Table 195 — Error correction codes

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

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

8.2 Cyclic redundancy codes (CRC)

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

8.2.1 CRC7

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

Generator polynomial $G(x) = x^7 + x^3 + 1$

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

$CRC\ [6:0] = \text{Remainder}[(M(x) \times x^7)/ G(x)]$

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

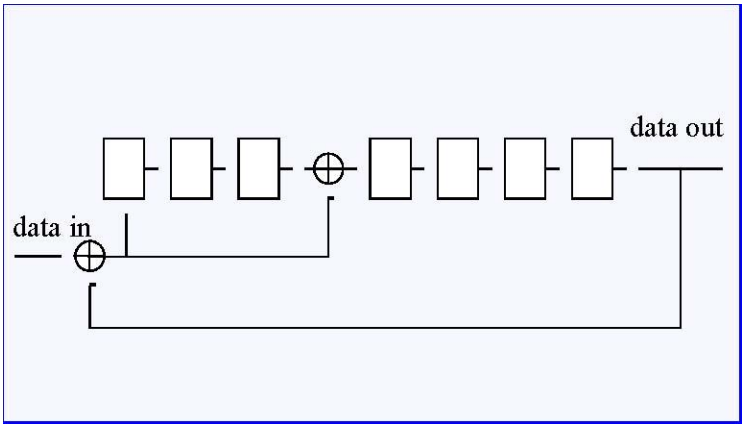


Figure 69 — CRC7 generator/checker

8.2.2 CRC16

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

Generator polynomial G(x) = x16 + x12 + x5 + 1

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

$CRC\ [15:0] = \text{Remainder}[(M(x) \times x^{16})/\ G(x)]$

CRC registers are initialized to zero. The first bit is the first data bit of the corresponding block. The degree *n* of the polynomial denotes the number of bits of the data block decreased by one (e.g., n = 4095 for a block length of 512 bytes). The generator polynomial Gx is a standard CCITT polynomial. The code has a minimal distance d=4 and is used for a payload length of up to 2048 Bytes (n <= 16383).

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

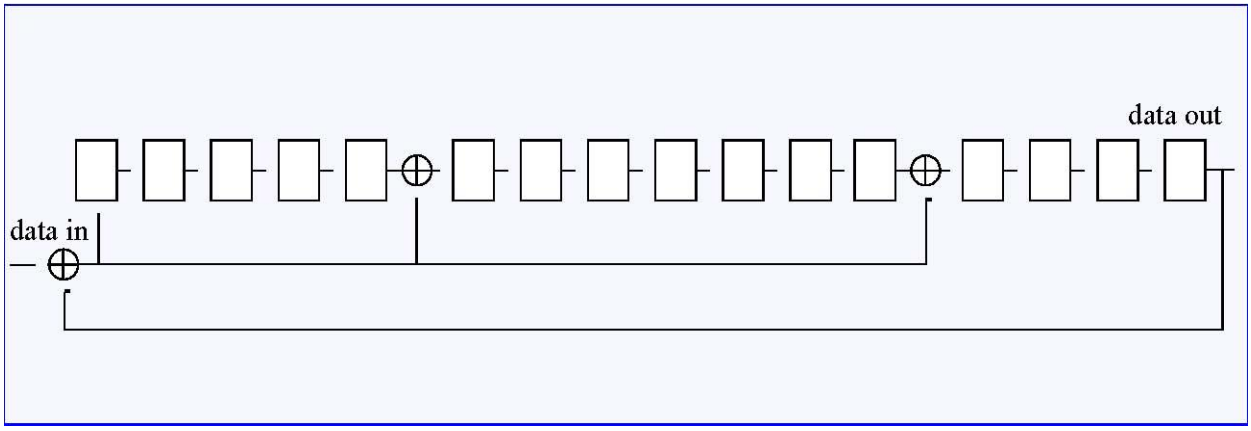


Figure 70 — CRC16 generator/checker

9 *e*•MMC mechanical standard

e•MMC and e^2•MMC discrete and *e*•MMC multichip ballouts are defined in JESD21C within the MCP or PoP section.

10 The *e*•MMC bus

The *e*•MMC bus has eleven communication lines:

- CMD: Command is a bidirectional signal. The host and Device drivers are operating in two modes, open drain and push/pull.
- DAT0-7: Data lines are bidirectional signals. Host and Device drivers are operating in push-pull mode
- CLK: Clock is a host to Device signal. CLK operates in push-pull mode
- Data Strobe: Data Strobe is a Device to host signal. Data Strobe operates in push-pull mode.

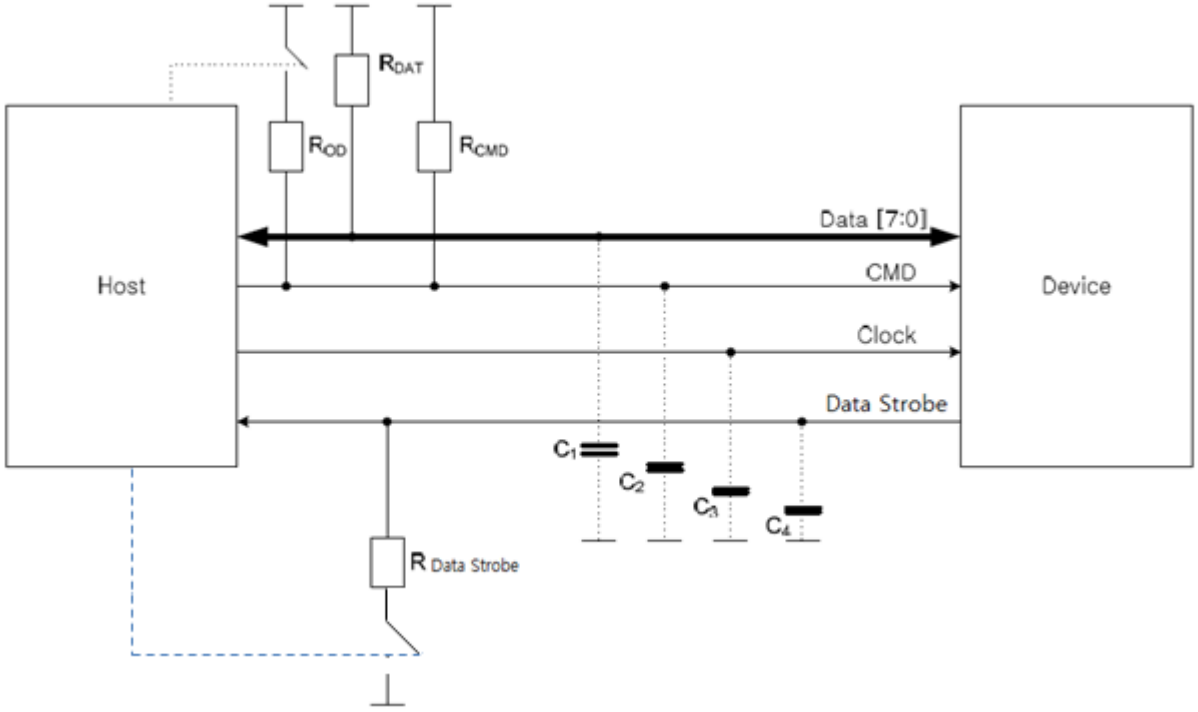


Figure 71 — Bus circuitry diagram

The R_{OD} is switched on and off by the host synchronously to the open-drain and push-pull mode transitions. The host does not have to have open drain drivers, but must recognize this mode to switch on the R_{OD}. R_{DAT} and R_{CMD} are pull-up resistors protecting the CMD and the DAT lines against bus floating device when all device 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 one given in.

$R_{Data\ Strobe}$ is pull-down resistor used in HS400 device.

10.1 Power-up

The power up of the *e*•MMC bus is handled locally in the Device and in the bus master.

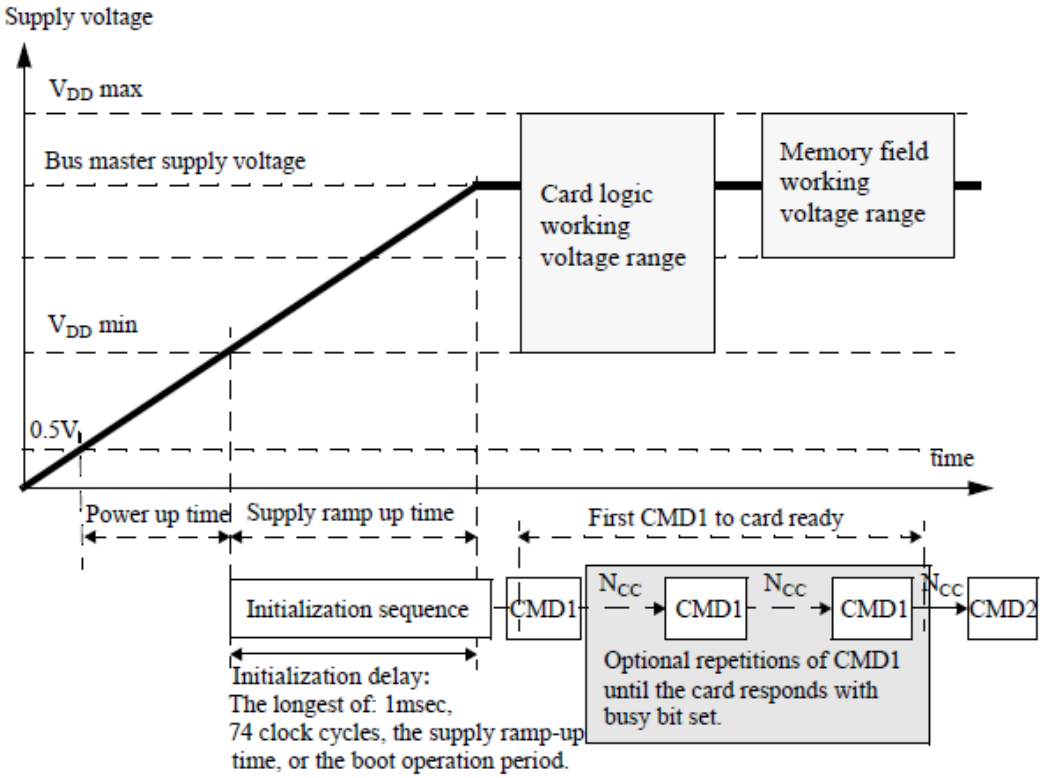


Figure 72 — Power-up diagram

- After power up (including hot insertion, i.e., inserting a Device when the bus is operating), the Device enters the *pre-idle* state. The power up time of the supply voltage should be less than the specified t_{PRU} for the Bus master supply voltage.
- If the Device does not support boot mode, or its BOOT_PARTITION_ENABLE bit is cleared, the Device moves immediately to the *idle* state. While in the *idle* state, the Device ignores all bus transactions until CMD1 is received. If the Device supports only standard v4.2 or earlier versions, it enters the *idle* state immediately following power-up.
- If the Device BOOT_PARTITION_ENABLE bit is set, the Device moves to the *pre-boot* state. The Device then waits for boot initiation sequence. Following the boot operation period, the Device enters the *idle* state. During the *pre-boot* state, if the Device receives any CMD line transaction other than CMD1 or the boot initiation sequence (keeping the CMD line low for at least 74 clock cycles, or issuing CMD0 with the argument of 0xFFFFFFFA), the Device moves to the *idle* state. If the Device receives the boot initiation sequence (keeping the CMD line low for at least 74 clock cycles, or issuing CMD0 with the argument of 0xFFFFFFFA), the Device begins boot operation. If boot acknowledge is enabled, the Device shall send acknowledge pattern “010” to the host within the specified time. After boot operation is terminated, the Device enters the *idle* state and shall be ready for CMD1 operation. If the Device receives CMD1 in the *pre-boot* state, it begins responding to the command and moves to Device identification mode.
- While in the *idle* state, the Device ignores all bus transactions until CMD1 is received.

10.1 Power-up (cont'd)

- The maximum initial load (after power up or hot insertion) that the *e*•MMC can present on the VCC and V_{CCQ} line shall be a maximum of 10 uF in parallel with a minimum of 330 Ω. At no time during operation shall the Device capacitance on the V_{CCQ} line exceed 10 uF
- CMD1 is a special synchronization command used to negotiate the operation voltage range and to poll the Device until it is out of its power-up sequence. Besides the operation voltage profile of the Device, the response to CMD1 contains a busy flag, indicating that the Device is still working on its power-up procedure and is not ready for identification. This bit informs the host that the Device is not ready. The host has to wait until this bit is cleared.
  - The Device shall complete its initialization within 1 second from the first CMD1 with a valid OCR range if boot operation is not executed.
  - Getting the Device out of *idle* state is up to the responsibility of the bus master. Since the power up time and the supply ramp up time depend on application parameters as the bus length and the power supply unit, the host must ensure that the power is built up to the operating level (the same level which will be specified in CMD1) before CMD1 is transmitted.
  - After power up the host starts the clock and sends the initializing sequence on the CMD line. The sequence length is the longest of: 1msec, 74 clocks, the supply-ramp-up-time, or the boot operation period. The additional 10 clocks (over the 64 clocks after what the Device should be ready for communication) is provided to eliminate power-up synchronization problems.
  - Every bus master has to implement CMD1. The CMD1 implementation is mandatory for all *e*•MMCs.

10.1.1 *e*•MMC power-up

An *e*•MMC bus power-up is handled locally in each device and in the bus master. Figure 73 shows the power-up sequence and is followed by specific instructions regarding the power-up sequence

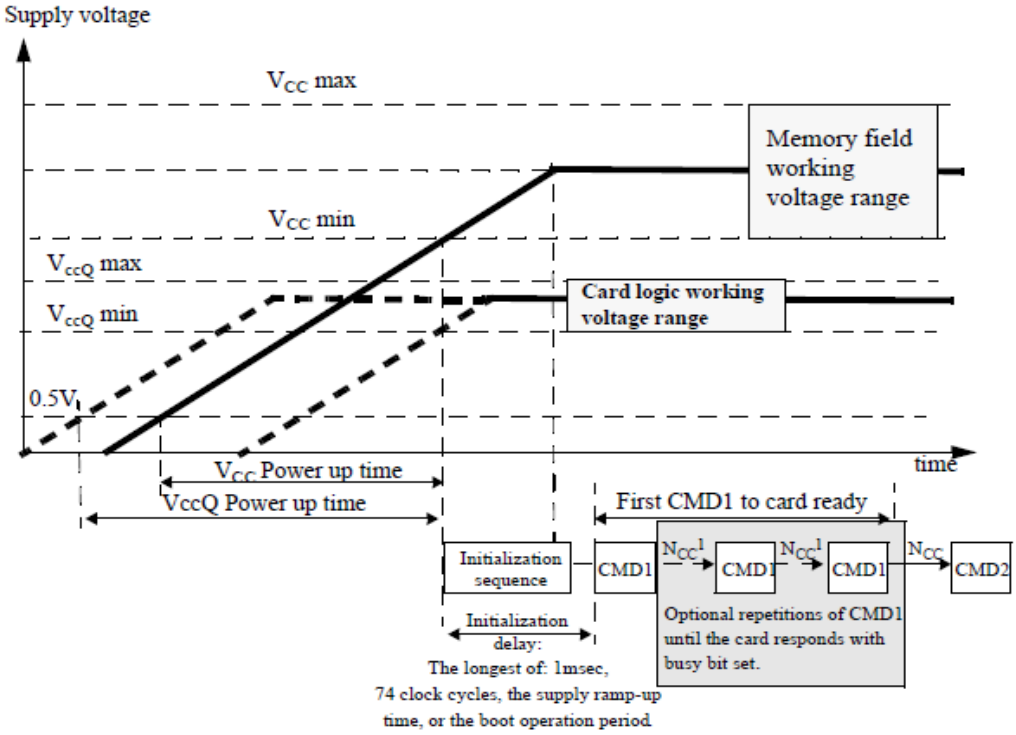


Figure 73 — *e*•MMC power-up diagram

10.1.2 *e*•MMC power-up guidelines

An *e*•MMC power-up must adhere to the following guidelines:

- When power-up is initiated, either V_{CC} or V_{CCQ} can be ramped up first, or both can be ramped up simultaneously.
- After power up, the *e*•MMC enters the *pre-idle* state. The power up time of each supply voltage should be less than the specified t_{PRU} (t_{PRUH}, t_{PRUL} or t_{PRUV}) for the appropriate voltage range.
- If the *e*•MMC does not support boot mode or its BOOT_PARTITION_ENABLE bit is cleared, the *e*•MMC moves immediately to the *idle* state. While in the *idle* state, the *e*•MMC ignores all bus transactions until CMD1 is received. If the *e*•MMC supports only standard v4.2 or earlier versions, the device enters the *idle* state immediately following power-up.
- If the BOOT_PARTITION_ENABLE bit is set, the *e*•MMC moves to the *pre-boot* state, and the *e*•MMC waits for the boot-initiation sequence. Following the boot operation period, the *e*•MMC enters the *idle* state. During the *pre-boot* state, if the *e*•MMC receives any CMD-line transaction other than the boot initiation sequence (keeping CMD line low for at least 74 clock cycles, or issuing CMD0 with the argument of 0xFFFFFFFA) and CMD1, the *e*•MMC moves to the *Idle* state. If *e*•MMC receives the boot initiation sequence (keeping the CMD line low for at least 74 clock cycles, or issuing CMD0 with the argument of 0xFFFFFFFA), the *e*•MMC begins boot operation. If boot acknowledge is enabled, the *e*•MMC shall send acknowledge pattern "010" to the host within the specified time. After boot operation is terminated, the *e*•MMC enters the *idle* state and shall be ready for CMD1 operation. If the *e*•MMC receives CMD1 in the *pre-boot* state, it begins responding to the command and moves to the Device identification mode.
- While in the *idle* state, the *e*•MMC ignores all bus transactions until CMD1 is received.
- CMD1 is a special synchronization command used to negotiate the operation voltage range and to poll the device until it is out of its power-up sequence. In addition to the operation voltage profile of the device, the response to CMD1 contains a busy flag indicating that the device is still working on its power-up procedure and is not ready for identification. This bit informs the host that the device is not ready, and the host must wait until this bit is cleared. The device must complete its initialization within 1 second of the first CMD1 issued with a valid OCR range.
    - If the *e*•MMC device was successfully partitioned during the previous power up session (bit 0 of EXT_CSD byte [155] PARTITION_SETTING_COMPLETED successfully set) then the initialization delay is (instead of 1s) calculated from INI_TIMEOUT_PA (EXT_CSD byte [241]). This timeout applies only for the very first initialization after successful partitioning. For all the consecutive initialization 1sec timeout will apply.
- The bus master moves the device out of the *idle* state. Because the power-up time and the supply ramp-up time depend on application parameters such as the bus length and the power supply unit, the host must ensure that power is built up to the operating level (the same level that will be specified in CMD1) before CMD1 is transmitted.
- After power-up, the host starts the clock and sends the initializing sequence on the CMD line. The sequence length is the longest of: 1ms, 74 clocks, the supply ramp-up time, or the boot operation period. An additional 10 clocks (beyond the 64 clocks of the power-up sequence) are provided to eliminate power-up synchronization problems.
- Every bus master must implement CMD1.
- All above rules apply also for e^2•MMC (V_{CC} and V_{CCQ}). In addition the e^2•MMC D-V_{DD} must ramp up before D-V_{DDQ}. There is no restriction for order of ramping up V_{CC}/V_{CCQ} versus D-V_{DD}/D-V_{DDQ}.

10.1.3 *e*•MMC power cycling

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

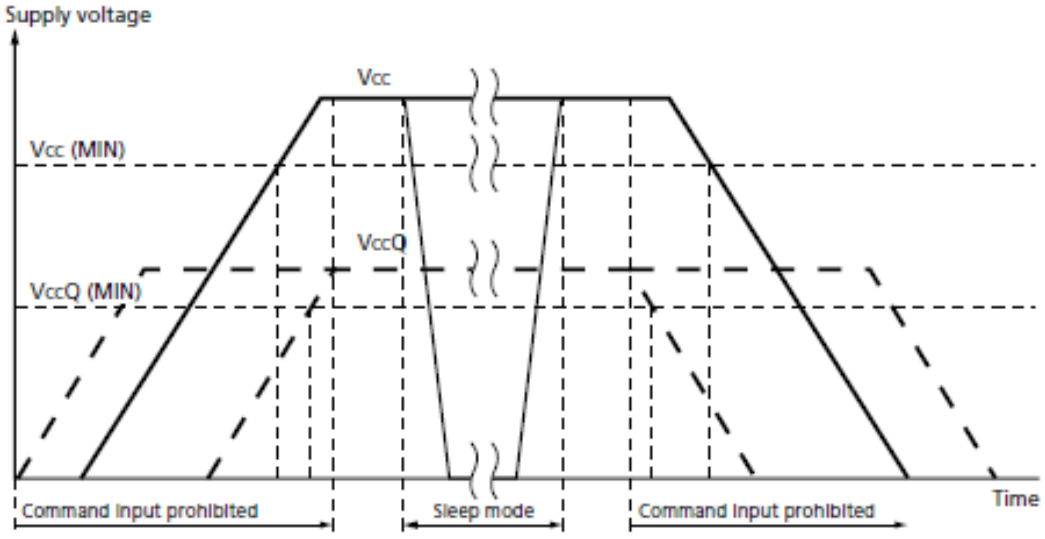


Figure 74 — *e*•MMC power cycle

If V_{CC} or V_{CCQ} are below 0.5 V for longer than 1 ms, the slave shall always return to the *pre-idle* state, and perform the appropriate boot behavior, as appropriate. The slave will behaves as in a standard power-up condition once the voltages have returned to their functional ranges.

An exception to this behavior is if the device is in sleep state, in which the voltage on V_{CC} is not monitored.

In addition to the above rules, in case of e^2•MMC, the D-V_{DDQ} shall be ramped down before D-V_{DD} and also D-V_{DDQ} / D-V_{DD} shall both be in stable state within the operation voltage range before any commands may be issued. Both D-V_{DDQ} and D-V_{DD} may be powered-down while the device is in Sleep state. Still again both D-V_{DDQ} and D-V_{DD} shall be powered-up before host issues CMD5 to wake-up the slave device.

µs

10.2 Programmable Device output driver

The bus capacitance of each line of the *e*•MMC bus is the sum of the bus master capacitance, the bus capacitance itself and the capacitance of each inserted Device. The sum of host and bus capacitance are fixed for one application, but may vary between different applications. The Device load may vary in one application with each of the inserted Devices.

The CMD and DAT bus drivers consist of a pre-driver stage and a complementary driver transistor (Figure 75). The DSR register is used to configure the pre-driver stage output rise and fall time, and the complementary driver transistor size. The proper combination of both allows optimum bus performance. Table 196 defines the DSR register contents:

Table 196 — DSR register content

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

	7	6	5	4	3	2	1	0
$i_{peak\ min}$	Reserved							
$I_{peak\ max}$								

10.2 Programmable Device output driver (cont'd)

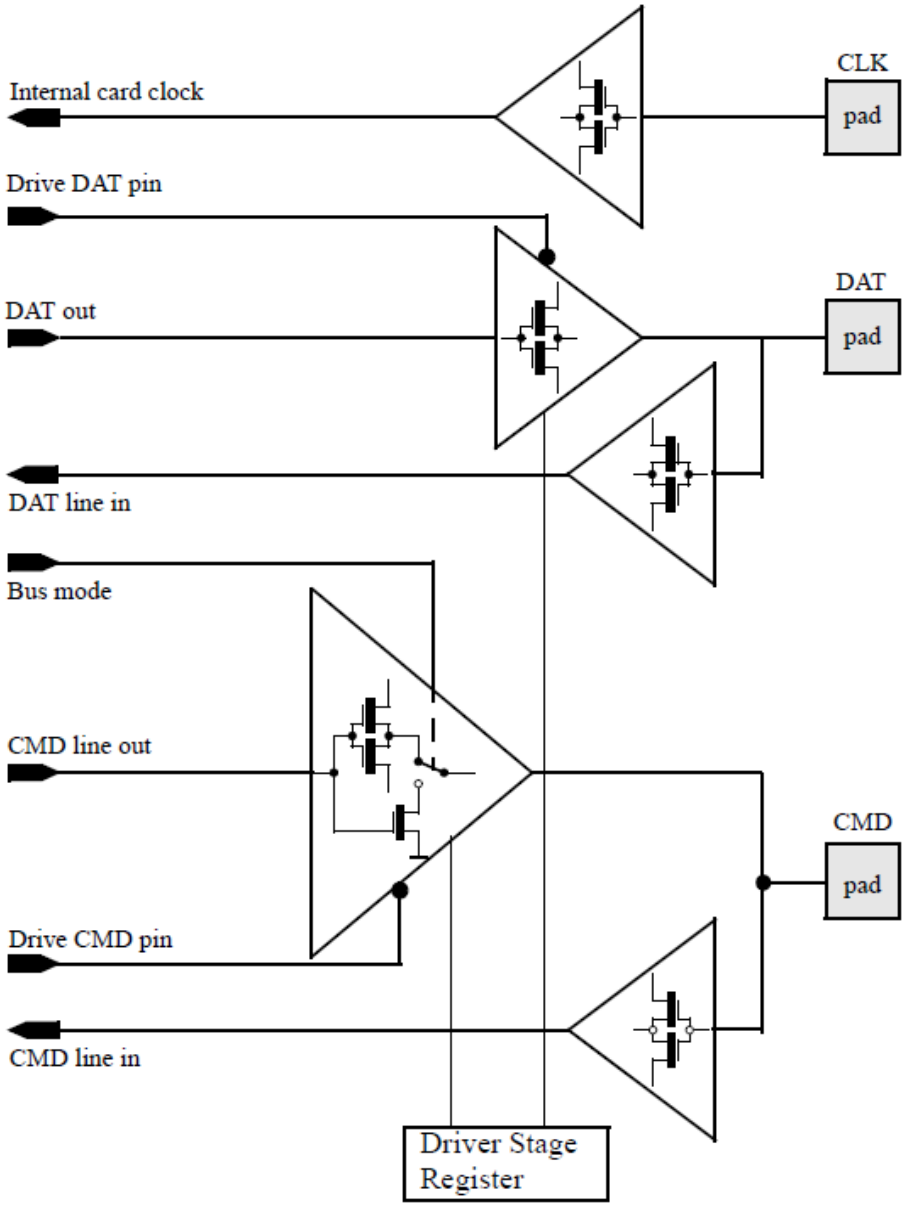


Figure 75 — *e*•MMC bus driver

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

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

10.3 Bus operating conditions

Table 197 — General operating conditions

Parameter	Symbol	Min	Max	Unit	Remarks
Peak voltage on all lines		-0.5	V_{CCQ} + 0.5	V	
All Inputs					
Input Leakage Current (before initialization sequence and/or the internal pull up resistors connected)		-100	100	µA	
Input Leakage Current (after initialization sequence and the internal pull up resistors disconnected)		-2	2	µA	
All Outputs					
Output Leakage Current (before initialization sequence)		-100	100	µA	
Output Leakage Current (after initialization sequence)		-2	2	µA	
NOTE 1 Initialization sequence is defined in 10.1					

10.3.1 Power supply: *e*•MMC

In the *e*•MMC, V_{CC} is used for the memory device; V_{CCQ} is for the controller and the MMC interface voltage. Either V_{CC} or V_{CCQ} is for memory interface voltage shown in Figure 76. The core regulator is optional and only required when internal core logic voltage is regulated from V_{CCQ}. A C_{reg} capacitor must be connected to the VDDi terminal to stabilize regulator output on the system.

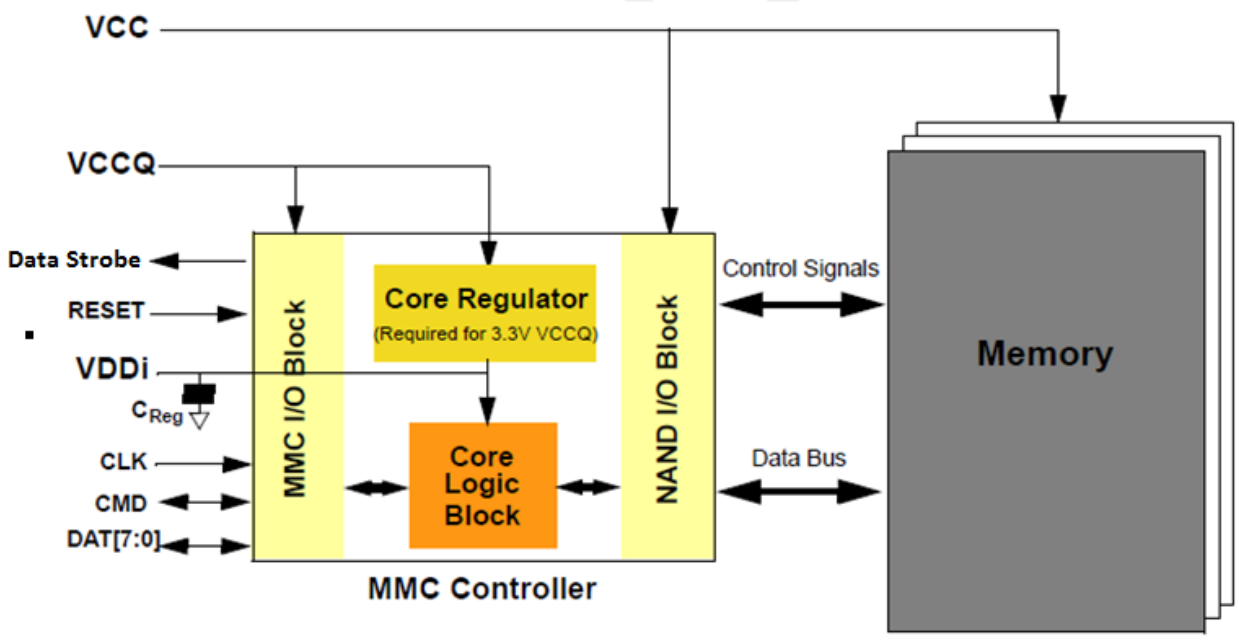


Figure 76 — *e*•MMC internal power diagram (as an example)

10.3.2 Power supply: *e*^{2}•MMC

In the *e*•MMC, V_{CC} is used for the memory device; V_{CCQ} is for the controller and the MMC. Either V_{CC} or V_{CCQ} is used for memory interface voltage shown in Figure 77. The core regulator is optional and only required when internal core logic voltage is regulated from V_{CCQ}. A C_{reg} capacitor must be connected to the VDDi terminal to stabilize regulator output on the system.

D-V_{DDQ} and D-V_{DD} are for the cache memory. D-V_{DD} is for the core of the cache memory and D-V_{DDQ} for the IO between memory controller and cache memory.

VDDi2 and VDDi3 are other internal voltage terminals. Each of those is generated from one of the external power supplies through a regulator to reduce power consumption or to avoid noise interference internally. C_{reg2} and C_{reg3} capacitors must be connected to VDDi2 and VDDi3 respectively. These are device optional for *e*^{2}•MMC devices.

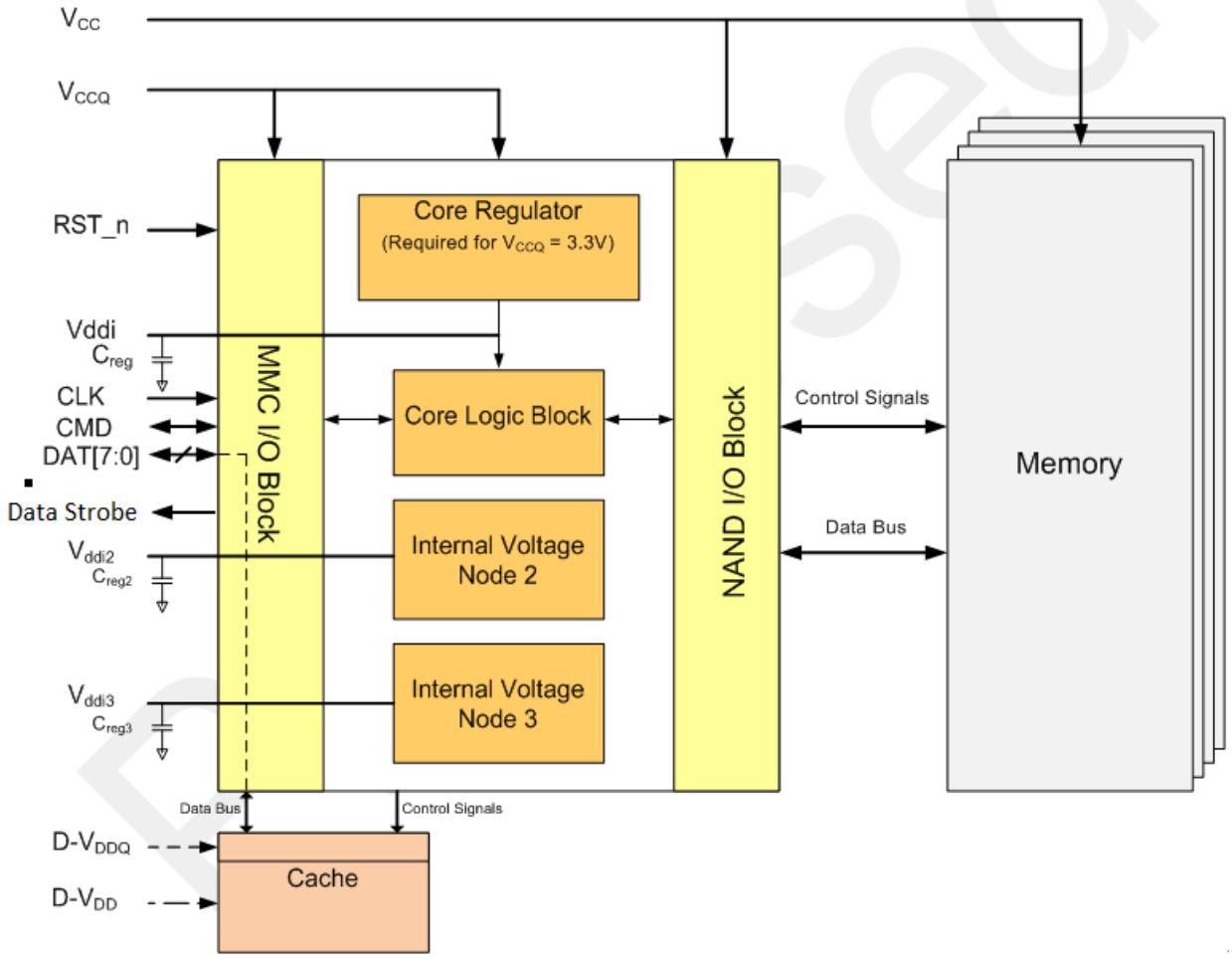


Figure 77 — *e*^{2}•MMC internal power diagram (as an example)

10.3.3 Power supply Voltages

The *e*•MMC supports one or more combinations of V_{CC} and V_{CCQ} and either combinations of D-V_{DDQ} and D-V_{DD} as shown in Table 197. The V_{CCQ} must be defined at equal to or less than V_{CC}. The available voltage configuration is shown in Table 198.

Table 198 — *e*•MMC power supply voltage

Parameter	Symbol	Min	Max	Unit	Remarks
Supply voltage (NAND)	V_{CC}	2.7	3.6	V	
		1.7	1.95	V	
Supply voltage (I/O)	V_{CCQ}	2.7	3.6	V	
		1.70	1.95	V	
		1.1	1.3	V	
Supply voltage (cache) (e^2•MMC)	D-V_{DD}	1.7	1,9	V	
Supply voltage (cache I/O) (e^2•MMC)	D-V_{DDQ}	1.7	1.9	V	
		1.14	1.3	V	
Supply power-up for 3.3 V	tPRUH		35	ms	
Supply power-up for 1.8 V	tPRUL		25	ms	
Supply power-up for 1.2 V	tPRUV		20	ms	

The *e*•MMC must support at least one of the valid voltage configurations, and can optionally support all valid voltage configurations, see Table 199.

Table 199 — *e*•MMC voltage combinations

		V_{CCQ}		
		1.1 V-1.3 V	**1.70 V–1.95 V**	**2.7 V–3.6 V**
V_{CC}	**2.7 V–3.6 V**	Valid	Valid	Valid (1)
	1.7 V–1.95 V	Valid	Valid	NOT VALID

NOTE 1 V_{CCQ} (I/O) 3.3 V range is not supported in either HS200 or HS400 devices

10.3.4 Bus signal line load

The total capacitance C_L of each line of the *e*•MMC bus is the sum of the bus master capacitance C_{HOST}, the bus capacitance C_{BUS} itself, and the capacitance C_{DEVICE} of the Device connected to this line,

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

and requiring the sum of the host and bus capacitances not to exceed 20 pF, see Table 200**Error! Reference source not found.Error! Reference source not found.**.

Table 200 — Capacitance and Resistors

Parameter	Symbol	Min	Typ	Max	Unit	Remark
Pull-up resistance for CMD	R_{CMD}	4.7		100(1)	kΩ	to prevent bus floating
Pull-up resistance for DAT0–7	R_{DAT}	10		100(1)	kΩ	to prevent bus floating
Internal pull up resistance DAT1–DAT7	R_{int}	10		150	kΩ	to prevent unconnected lines floating
Bus signal line capacitance	C_L			30	pF	Single Device
Single Device capacitance	C_{DEVICE}			6	pF	
Maximum signal line inductance				16	nH	$f_{PP} \leq 52$ MHz
V_{DDi} capacitor value	C_{REG}(2)	0.1			uF	To stabilize regulator output when target device bus speed mode is either backward-compatible, high speed SDR, high speed DDR, or HS200.
		1			uF	To stabilize regulator output when target device bus speed mode is HS400.
V_{DDi2} capacitor value (*e*^{2}•MMC)	C_{REG2}	1			uF	To stabilize internal regulated voltage
V_{DDi3} capacitor value (*e*^{2}•MMC)	C_{REG3}	1			uF	To stabilize internal regulated voltage
V_{CCQ} decoupling capacitor	C_{H1}	1			uF	(3), (4), (5)

NOTE 1 Recommended maximum pull-up is 30 kΩ for 1.2 V and 50 kΩ for 1.8 V interface supply voltages. A 3 V part, may use the whole range up to 100 kΩ.

NOTE 2 Recommended value for C_{REG}, C_{REG2} and C_{REG3} might be different between *e*•MMC device vendors.

NOTE Confirm the maximum value and the accuracy of the capacitance with *e*•MMC vendor because the electrical characteristics of the regulator inside *e*•MMC is affected by the fluctuation of the capacitance.

NOTE 3 C_{H1} is V_{CCQ}-V_{SSQ} decoupling capacitor required for HS200&HS400 *e*•MMC device.

NOTE 4 C_{H1} should be placed adjacent to V_{CCQ}-V_{SSQ} balls (around DAT[7..0] balls), It should be located as close as possible to the balls defined in order to minimize connection parasitics.

NOTE 5 *e*•MMC device vendor may have more specific requirements for CH1 placement. Please confirm such requirements with specific *e*•MMC device vendor.

10.3.5 HS400 reference load

The circuit in Figure 78 shows the reference load used to define the HS400 Device Output Timings and overshoot / undershoot parameters.

The reference load is made up by the transmission line and the $C_{REFERENCE}$ capacitance.

The reference load is not intended to be a precise representation of the typical system environment nor a depiction of the actual load presented by a production tester.

System designers should use IBIS or other simulation tools to correlate the reference load to system environment. Manufacturers should correlate to their production test conditions.

Delay time (td) of the transmission line has been introduced to make the reference load independent from the PCB technology and trace length.

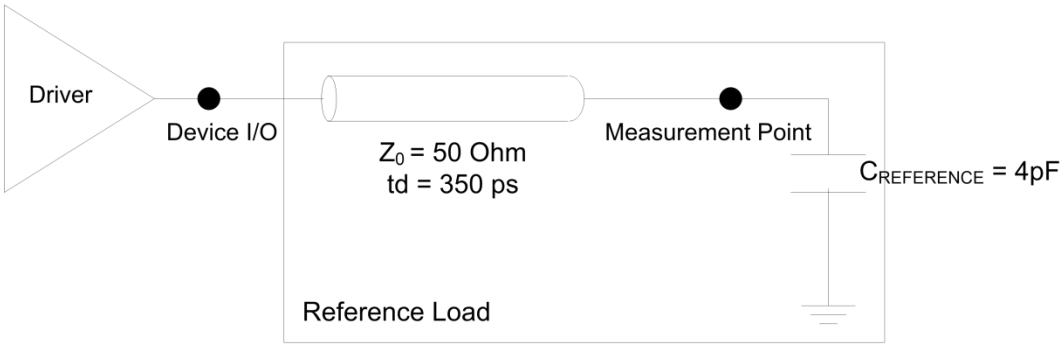


Figure 78 — HS400 reference load

10.4 Overshoot/Undershoot Specification

Table 201 — AC Overshoot/Undershoot Specification

		V_{CCQ}	Unit
		1.70 V – 1.95 V	
Maximum peak amplitude allowed for overshoot area. (See Figure 79)	Max	0.9	V
Maximum peak amplitude allowed for undershoot area. (See Figure 74)	Max	0.9	V
Maximum area above V_{CCQ} (See Figure 74)	Max	1.5	V-ns
Maximum area below V_{SSQ} (See Figure 74)	Max	1.5	V-ns

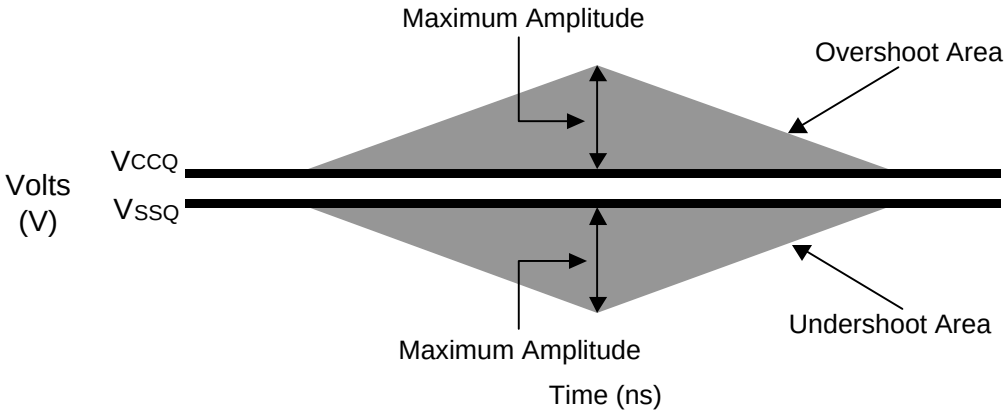


Figure 79 — Overshoot/Undershoot definition

10.5 Bus signal levels

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

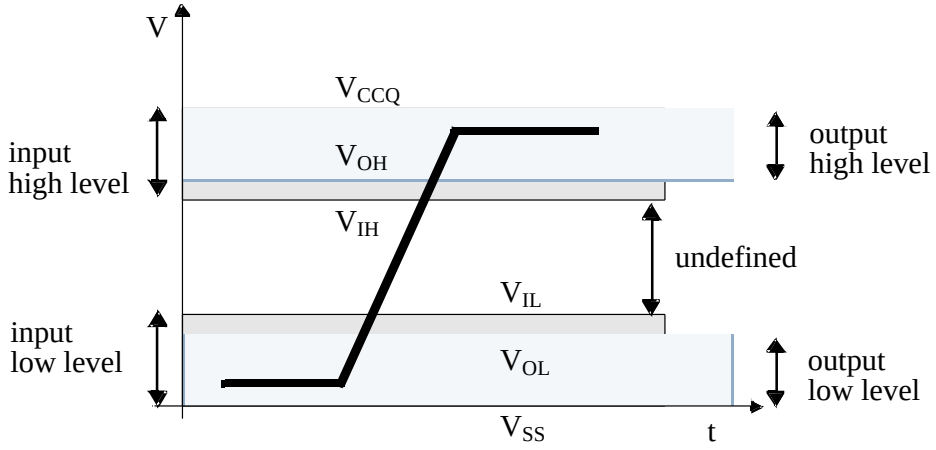


Figure 80 — Bus signal levels

10.5.1 Open-drain mode bus signal level

Table 202 — Open-drain bus signal level

Parameter	Symbol	Min	Max	Unit	Conditions
Output HIGH voltage	V_{OH}	V_{CCQ} - 0.2		V	NOTE 1
Output LOW voltage	V_{OL}		0.3	V	I_{OL} = 2 mA
NOTE 1 Because Voh depends on external resistance value (including outside the package), this value does not apply as device specification. Host is responsible to choose the external pull-up and open drain resistance value to meet Voh Min value.					

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

10.5.2 Push-pull mode bus signal level— *e*•MMC

The device input and output voltages shall be within the following specified ranges for any V_{CCQ} of the allowed voltage range

For 2.7 V - 3.6 V V_{CCQ} range (compatible with JESD8C.01)

Table 203 — Push-pull signal level—high-voltage *e*•MMC

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

For 1.70 V - 1.95 V V_{CCQ} range as defined in Table 204.

Table 204 — Push-pull signal level—1.70 V -1.95 V V_{CCQ} voltage Range

Parameter	Symbol	Min	Max	Unit	Conditions
Output HIGH voltage	V_{OH}	V_{CCQ} - 0.45V		V	I_{OH} = -2mA
Output LOW voltage	V_{OL}		0.45V	V	I_{OL} = 2mA
Input HIGH voltage	V_{IH}	0.65 * V_{CCQ} (1)	V_{CCQ} + 0.3	V	
Input LOW voltage	V_{IL}	V_{SS} - 0.3	0.35 * V_{CCQ}(2)	V	
NOTE 1 0.7 * V_{DD} for MMC4.3 and older revisions. NOTE 2 0.3 * V_{DD} for MMC4.3 and older revisions.					

For 1.1 V - 1.3 V V_{CCQ} range as defined in Table 205.

Table 205 — Push-pull signal level—1.1 V-1.3 V V_{CCQ} range *e*•MMC

Parameter	Symbol	Min	Max	Unit	Conditions
Output HIGH voltage	V_{OH}	0.75*V_{CCQ}		V	I_{OH} = -2 mA
Output LOW voltage	V_{OL}		0.25*V_{CCQ}	V	I_{OL} = 2 mA
Input HIGH voltage	V_{IH}	0.65 * V_{CCQ}	V_{CCQ} + 0.3	V	
Input LOW voltage	V_{IL}	V_{SS} - 0.3	0.35 * V_{CCQ}	V	
NOTE Both the 1.95 V - 2.7 V range and the 1.3 V – 1.70 V are undefined. The *e*•MMC device does not operate at this voltage range.					

10.5.3 Bus Operating Conditions for HS200 and HS400

The bus operating conditions for HS200 and HS400 devices is the same as specified in 10.5.1 through 10.5.2. The only exception is that V_{CCQ} = 3.3 V is not supported.

10.5.4 Device Output Driver Requirements for HS200 and HS400

10.5.4.1 Driver Types Definition

Driver Type-0 is defined as mandatory for *e*•MMC HS200 and HS400 Device While four additional Driver Types (1, 2, 3 and 4) are defined as optional, to allow the support of wider Host loads. The Host may select the most appropriate Driver Type of the Device (if supported) to achieve optimal signal integrity performance.

NOTE Drive strength definitions are same for 1.8 V signaling level and for 1.2 V signaling level.

Driver Type-0 is targeted for transmission line, based distributed system with 50 Ω nominal line impedance. Therefore, it is defined as 50 Ω nominal driver.

For HS200, when tested with C_L = 15pF Driver Type-0 shall meet all AC characteristics (see 10.5.4.2) and HS200 Device output timing requirements (see 10.8.3). The test circuit defined in 10.5.4.3 is used for testing of Driver Type-0.

For HS400, when tested with the reference load defined in 10.3.5, Driver Type-0 or Driver Type-1 or Driver Type-4 shall meet all AC characteristics (see 10.5.4.2) and HS400 Device output timing requirements (see 10.10.2).

The Optional Driver Types are defined with reference to Driver Type-0.

Table 206 summarizes the nominal impedance characteristics for the five Driver Types.

Table 206 — I/O driver strength types

Driver Type Values	Support	Nominal Impedance	Approximated driving capability compared to Type-0	Remark
0x0	Mandatory	50 Ω	x1	Default Driver Type. Supports up to 200 MHz operation.
0x1	Optional	33 Ω	x1.5	Supports up to 200 MHz operation.
0x2	Optional	66 Ω	x0.75	The weakest driver that supports up to 200 MHz operation.
0x3	Optional	100 Ω	x0.5	For low noise and low EMI systems. Maximal operating frequency is decided by Host design.
0x4	Optional	40 Ω	x1.2	

NOTE 1 Support of Driver Type-0 is mandatory for HS200 and HS400 device.
NOTE 2 Nominal impedance is defined by I-V characteristics of output driver at 0.9 V when V_{CCQ} = 1.8 V.
NOTE 3 Nominal impedance is defined by I-V characteristics of output driver at 0.6 V when V_{CCQ} = 1.2 V.

If Device supports the optional Driver Types, the Host may use them to optimize the signal integrity in its system. To do so, the Host designer may simulate its specific system, using a Device driver models. Host can select the optimal Driver Type that may drive the Host system load at the desired operating frequency with minimal noise generated.

10.5.4.2 Driver Type-0 AC Characteristics

Table 207 is the mandatory requirement from Driver Type-0, for HS200 and HS400 Device.

AC requirements for HS200 device are defined in 10.8.

AC requirements for HS400 device are defined in 10.10.

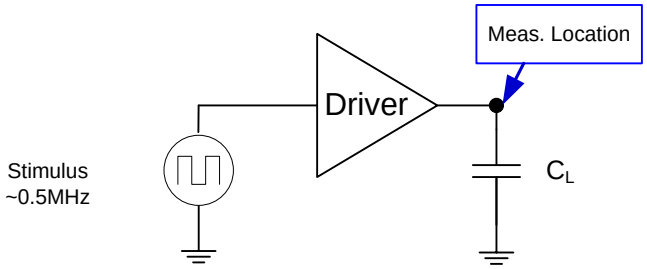
The I-V curves (current-voltage characteristics) of Driver Types 1, 2, 3 and 4 are approximately x1.5, x0.75 and x0.5 from the default Driver Type-0.

Table 207 — Driver Type-0 AC Characteristics

Parameter	Symbol	Min.	Typ.	Max.	Units	Remark
Rise/Fall Time	T_{R0}, T_{F0}	0.40	0.88	1.32	ns	C_L =15pF, Note 1
Ratio of fall time to rise time	R_{FR}	0.7	1.0	1.4	-	$R_{FR} = T_{F0} / T_{R0}$, Note 2, 3
NOTE 1 T_{R0} is measured between V_{OL} to V_{OH}, T_{F0} is measured between V_{OH} to V_{OL}. NOTE 2 Worst case R_{FR} is expected when the P and N CMOS processes are unbalanced. R_{FR} is defined for all possible specific operating condition points. (R_{FR} shall be verified individually for each valid operating point with fixed temperature, voltage and process condition)						

10.5.4.3 Driver Type-0 Test Circuit

The test circuit as describes in Figure 81 is used to verify driver characteristics defined in 10.5.4.2. Measured characteristics should meet specification under all corner conditions (process and environmental).



NOTE 1 C_L is total equivalent lumped capacitance for each Driver.

NOTE 2 C_L incorporates device die load, device package load and equivalent lumped load external to the device.

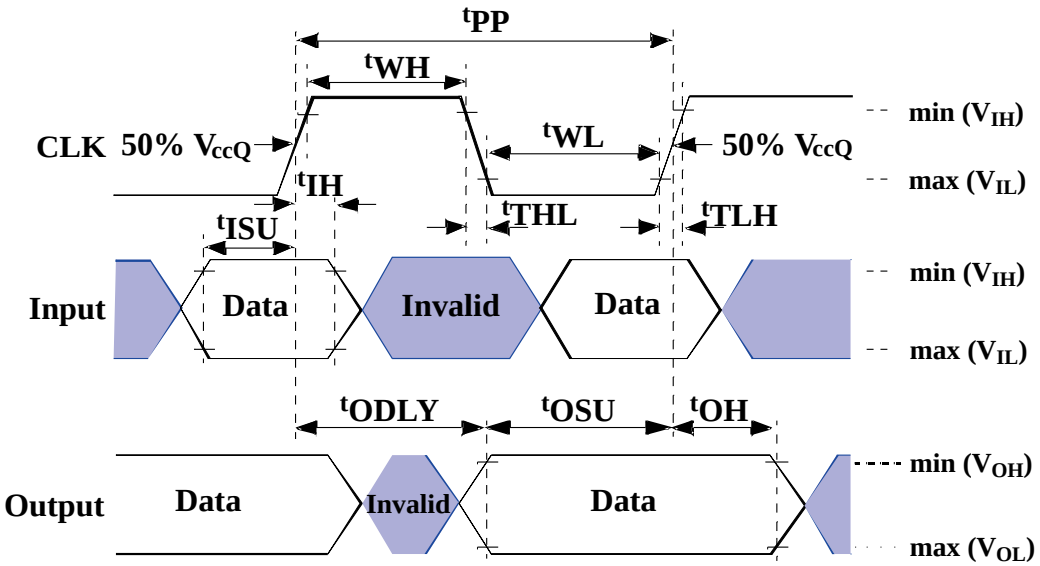
NOTE 3 In distributed transmission lines only part of the line capacitance considered as load for the Driver.

Figure 81 — Outputs test circuit for rise/fall time measurement

10.5.4.4 Driver Type Selection

The levels of Driver Types supported by the device are indicated in the DRIVER_STRENGTH [197] field of the Extended CSD register. The host sets the desired level by writing (through CMD6) to the “Selected Driver Strength" field in HS_TIMING[185] byte of the Extended CSD register.

10.6 Bus timing



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

Figure 82 — Timing diagram: data input/output

10.6.1 Device interface timings

Table 208 — High-speed Device interface timing

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

NOTE 1 CLK timing is measured at 50% of V_{CCQ}.
NOTE 2 A *e*•MMC shall support the full frequency range from 0 Mhz - 26 Mhz, or 0 MHz - 52 MHz
NOTE 3 Device can operate as high-speed Device interface timing at 26 MHz clock frequency.
NOTE 4 CLK rise and fall times are measured by min (V_{IH}) and max (V_{IL}).
NOTE 5 Inputs CMD, DAT rise and fall times are measured by min (V_{IH}) and max (V_{IL}), and outputs CMD, DAT rise and fall times are measured by min (V_{OH}) and max (V_{OL}).

10.6.1 Device interface timings (cont'd)

Table 209 — Backward-compatible Device interface timing

Parameter	Symbol	Min	Max	Unit	Remark[1]
Clock CLK[2]					
Clock frequency Data Transfer Mode (PP)[3]	f_{PP}	0	26	MHz	$C_L \leq 30$ pF
Clock frequency Identification Mode (OD)	f_{OD}	0	400	kHz	
Clock high time	t_{WH}	10			$C_L \leq 30$ pF
Clock low time	t_{WL}	10		ns	$C_L \leq 30$ pF
Clock rise time[4]	t_{TLH}		10	ns	$C_L \leq 30$ pF
Clock fall time	t_{THL}		10	ns	$C_L \leq 30$ pF
Inputs CMD, DAT (referenced to CLK)					
Input set-up time	t_{ISU}	3		ns	$C_L \leq 30$ pF
Input hold time	t_{IH}	3		ns	$C_L \leq 30$ pF
Outputs CMD, DAT (referenced to CLK)					
Output set-up time[5]	t_{OSU}	11.7		ns	$C_L \leq 30$ pF
Output hold time[5]	t_{OH}	8.3		ns	$C_L \leq 30$ pF

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

NOTE 2 CLK timing is measured at 50% of V_{CCQ}.

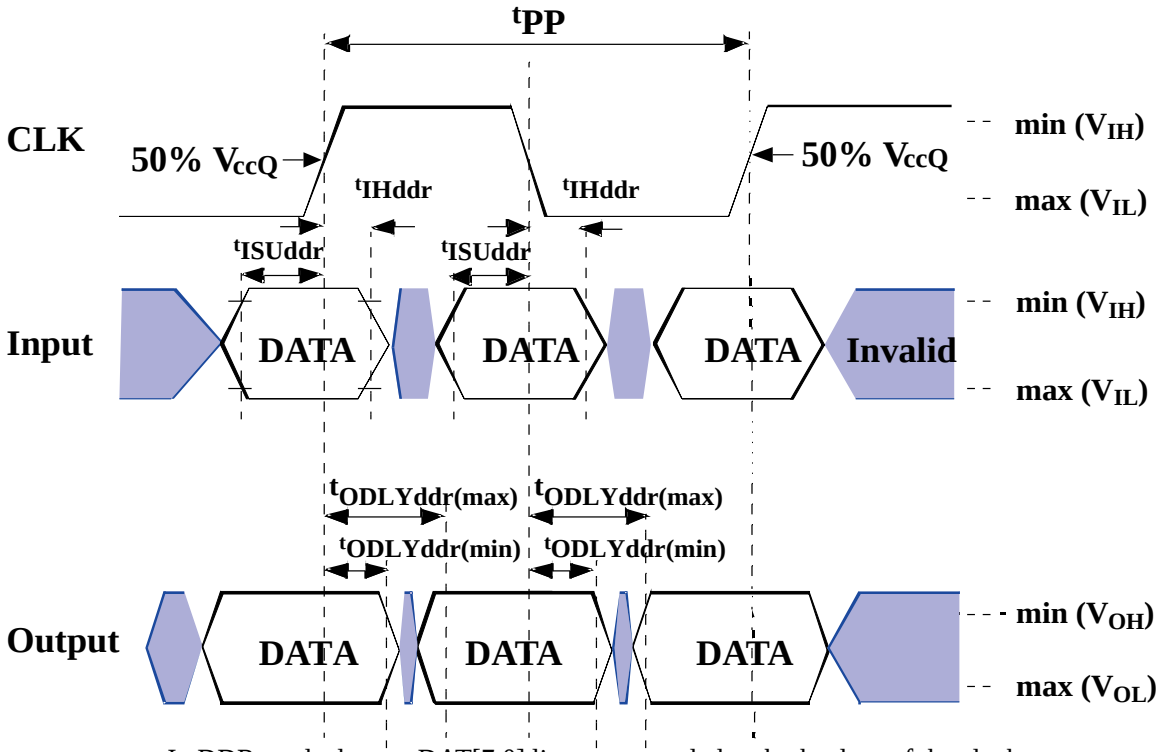
NOTE 3 For compatibility with Devices that support the v4.2 standard or earlier, host should not use > 26 MHz before switching to high-speed interface timing.

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

NOTE 5 t_{OSU} and t_{OH} are defined as values from clock rising edge. However, there may be Devices or devices which utilize clock falling edge to output data in backward compatibility mode. Therefore, it is recommended for hosts either to set t_{WL} value as long as possible within the range which will not go over t_{CK}-$t_{OH(min)}$ in the system or to use slow clock frequency, so that host could have data set up margin for those devices. In this case, each device which utilizes clock falling edge might show the correlation either between t_{WL} and t_{OSU} or between tCK and t_{OSU} for the device in its own datasheet as a note or its' application notes.

10.7 Bus timing for DAT signals during 2x data rate operation

These timings applies to the DAT[7:0] signals only when the device is configured for dual data mode operation. In this dual data mode, the DAT signals operates synchronously of both the rising and the falling edges of CLK. the CMD signal still operates synchronously of the rising edge of CLK and therefore complies with the bus timing specified in 10.6, therefore there is no timing change for the CMD signal.



In DDR mode data on DAT[7:0] lines are sampled on both edges of the clock (not applicable for CMD line)

Figure 83 — Timing diagram: data input/output in dual data rate mode

10.7.1 Dual data rate interface timings

Table 210 — High-speed dual rate interface timing

Parameter	Symbol	Min	Max	Unit	Remark
Input CLK(1)					
Clock duty cycle		45	55	%	Includes jitter, phase noise
Clock rise time	t_{TLH}		3	ns	CL≤ 30 pF
Clock fall time	t_{THL}		3	ns	CL≤ 30 pF
Input CMD (referenced to CLK-SDR mode)					
Input set-up time	t_{ISUddr}	3		ns	CL ≤ 20 pF
Input hold time	t_{IHddr}	3		ns	CL ≤ 20 pF
Output CMD (referenced to CLK-SDR mode)					
Output delay time during data transfer	t_{ODLY}		13.7	ns	CL ≤ 20 pF
Output hold time	t_{OH}	2.5		ns	CL ≤ 20 pF
Signal rise time	t_{RISE}		3	ns	CL ≤ 20 pF
Signal fall time	t_{FALL}		3	ns	CL ≤ 20 pF
Input DAT (referenced to CLK-DDR mode)					
Input set-up time	t_{ISUddr}	2.5		ns	CL ≤ 20 pF
Input hold time	t_{IHddr}	2.5		ns	CL ≤ 20 pF
Output DAT (referenced to CLK-DDR mode)					
Output delay time during data transfer	$t_{ODLYddr}$	1.5	7	ns	CL ≤ 20 pF
Signal rise time (DAT0-7)(2)	t_{RISE}		2	ns	CL ≤ 20 pF
Signal fall time (DAT0-7)	t_{FALL}		2	ns	CL ≤ 20 pF

NOTE 1 CLK timing is measured at 50% of V_{CCQ}.
NOTE 2 Inputs DAT rise and fall times are measured by min (V_{IH}) and max (V_{IL}), and outputs DAT rise and fall times are measured by min (V_{OH}) and max (V_{OL})

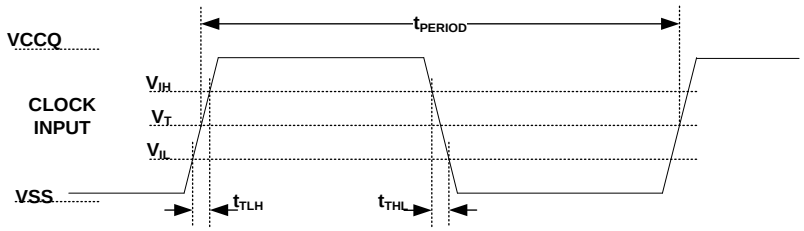
10.8 Bus Timing Specification in HS200 mode

10.8.1 HS200 Clock Timing

Host CLK Timing in HS200 mode shall conform to the timing specified in Figure 84 and Table 211.

CLK input shall satisfy the clock timing over all possible operation and environment conditions. CLK input parameters should be measured while CMD and DAT lines are stable high or low, as close as possible to the Device.

The maximum frequency of HS200 is 200 MHz. Hosts can use any frequency up to the maximum that HS200 mode allows.



NOTE 1 V_{IH} denote V_{IH}(min.) and V_{IL} denotes V_{IL}(max.).
NOTE 2 V_T = 50% of V_{CCQ}, indicates clock reference point for timing measurements.

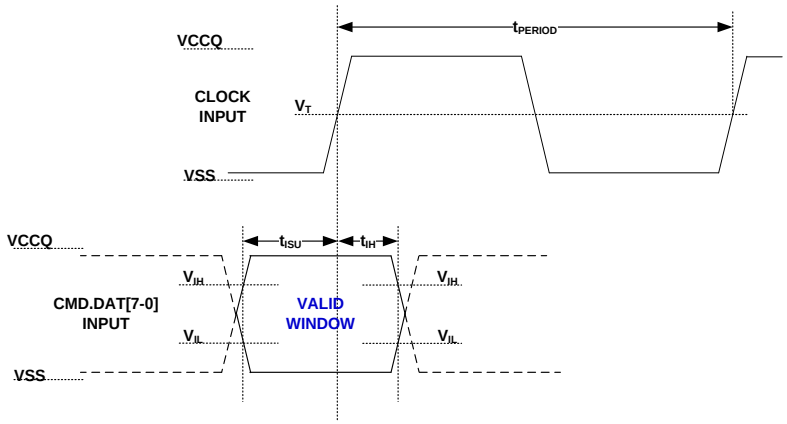
Figure 84 — HS200 Clock signal timing

Table 211 — HS200 Clock signal timing

Symbol	Min	Max	Unit	Remark
t_{PERIOD}	5	-	ns	200 MHz (max), between rising edges
t_{TLH}, t_{THL}	-	0.2· t_{PERIOD}	ns	t_{TLH}, t_{THL} < 1ns (max) at 200 MHz, C_{DEVICE} = 6 pF, The absolute maximum value of t_{TLH}, t_{THL} is 10ns regardless of clock frequency.
Duty Cycle	30	70	%	

10.8.2 HS200 Device Input Timing

Figure 85 and Table 212 define Device input timing.



NOTE 1 t_{ISU} and t_{IH} are measured at V_{IL}(max) and V_{IH}(min).
NOTE 2 V_{IH} denote V_{IH}(min) and V_{IL} denotes V_{IL}(max).

Figure 85 — HS200 Device input timing

Table 212 — HS200 Device input timing

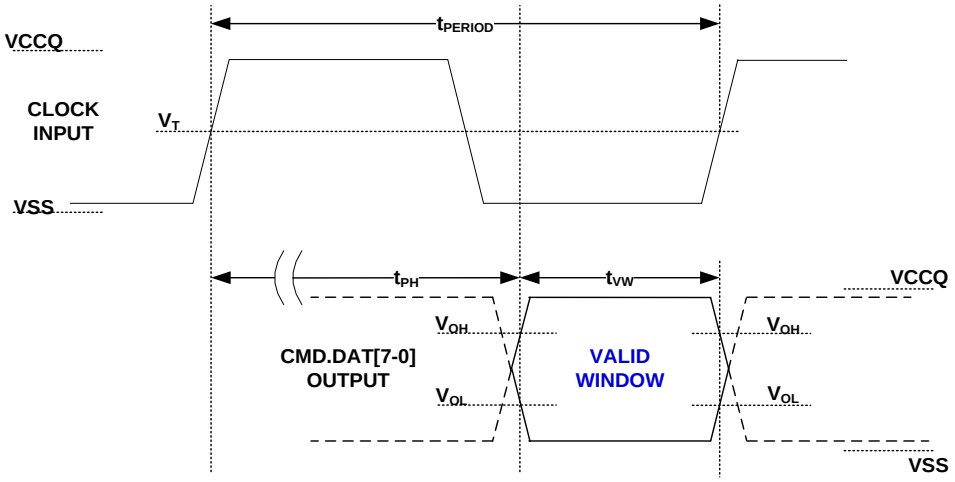
Symbol	Min	Max	Unit	Remark
t_{ISU}	1.40	-	ns	C_{DEVICE} ≤**6** pF
t_{IH}	0.8		ns	C_{DEVICE} ≤**6** pF

10.8.3 HS200 Device Output Timing

t_{PH} parameter is defined to allow device output delay to be longer than t_{PERIOD}. After initialization, the t_{PH} may have random phase relation to the clock. The Host is responsible to find the optimal sampling point for the Device outputs, while switching to the HS200 mode.

Figure 86 and Table 213 define Device output timing.

While setting the sampling point of data, a long term drift, which mainly depends on temperature drift, should be considered. The temperature drift is expressed by Δ_{TPH}. Output valid data window (t_{VW}) is available regardless of the drift (Δ_{TPH}) but position of data window varies by the drift, as describes in Figure 87.



NOTE V_{OH} denotes V_{OH}(min) and V_{OL} denotes V_{OL}(max).

Figure 86 — HS200 Device output timing

Table 213 — Output timing

Symbol	Min	Max	Unit	Remark
t_{PH}	0	2	UI	Device output momentary phase from CLK input to CMD or DAT lines output. Does not include a long term temperature drift.
Δ_{TPH}	-350 (ΔT= -20 °C)	+1550 (ΔT= 90 °C)	ps	Delay variation due to temperature change after tuning. Total allowable shift of output valid window (T_{VW}) from last system Tuning procedure Δ_{TPH} is 2600ps for ΔT from -25 °C to 125 °C during operation.
t_{VW}	0.575	-	UI	t_{VW} = 2.88ns at 200 MHz Using test circuit in Figure 81 including skew among CMD and DAT lines created by the Device. Host path may add Signal Integrity induced noise, skews, etc. Expected T_{VW} at Host input is larger than 0.475UI.
NOTE Unit Interval (UI) is one bit nominal time. For example, UI=5ns at 200 MHz.				

10.8.3 HS200 Device Output Timing (cont'd)

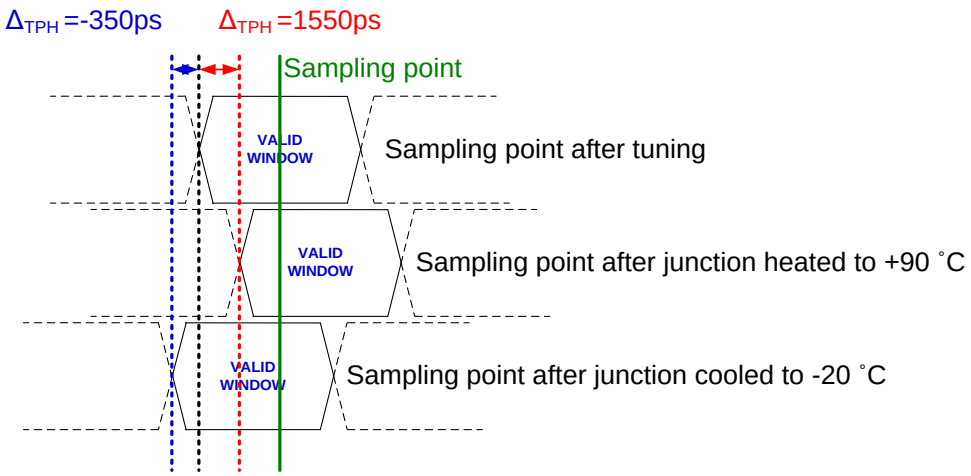


Figure 87 — Δ_{TPH} consideration

Implementation Guide:
Host should design to avoid sampling errors that may be caused by the Δ_{TPH} drift.
It is recommended to perform tuning procedure while Device wakes up, after sleep.
One simple way to overcome the Δ_{TPH} drift is by reduction of operating frequency.

10.9 Temperature Conditions

Table 214 — Temperature Conditions

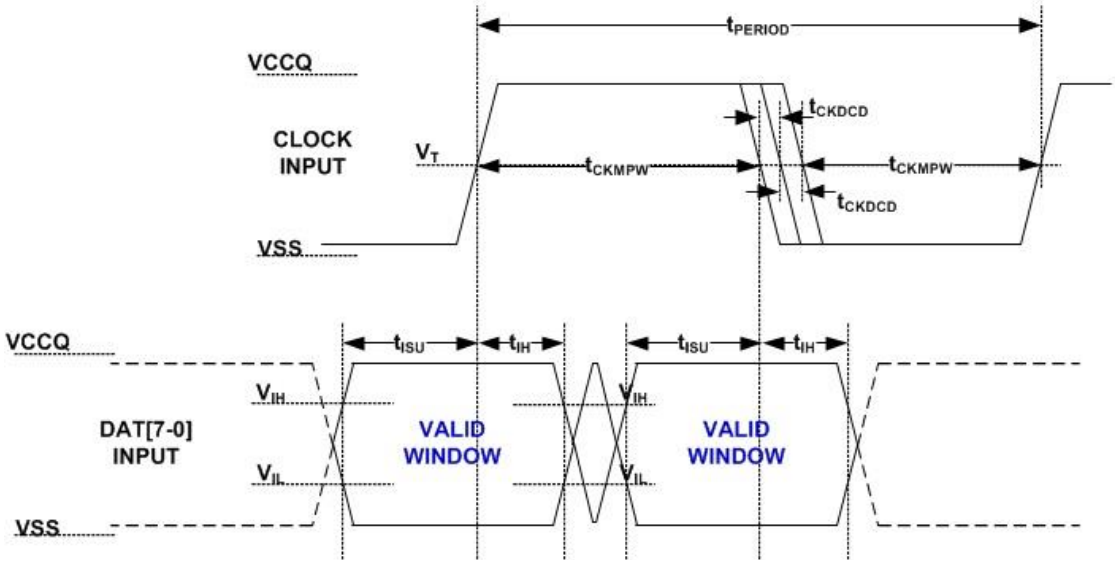
Condition	Ambient Temperature[1] – Ta
Operating	-25 °C to +85 °C
Storage	-40 °C to +85 °C
NOTE 1 To achieve optimized power/performance, refer to Annex A.10 for package case (Tcase) temperature control nominal time. For example, UI = 5ns at 200MHz.	

10.10 Bus Timing Specification in HS400 mode

10.10.1 HS400 Device Input Timing

The CMD input timing for HS400 mode is the same as CMD input timing for HS200 mode.

Figure 88 and Table 215 show Device input timing.



NOTE V_T = 50% of V_{CCQ}, indicates clock reference point for timing measurements.

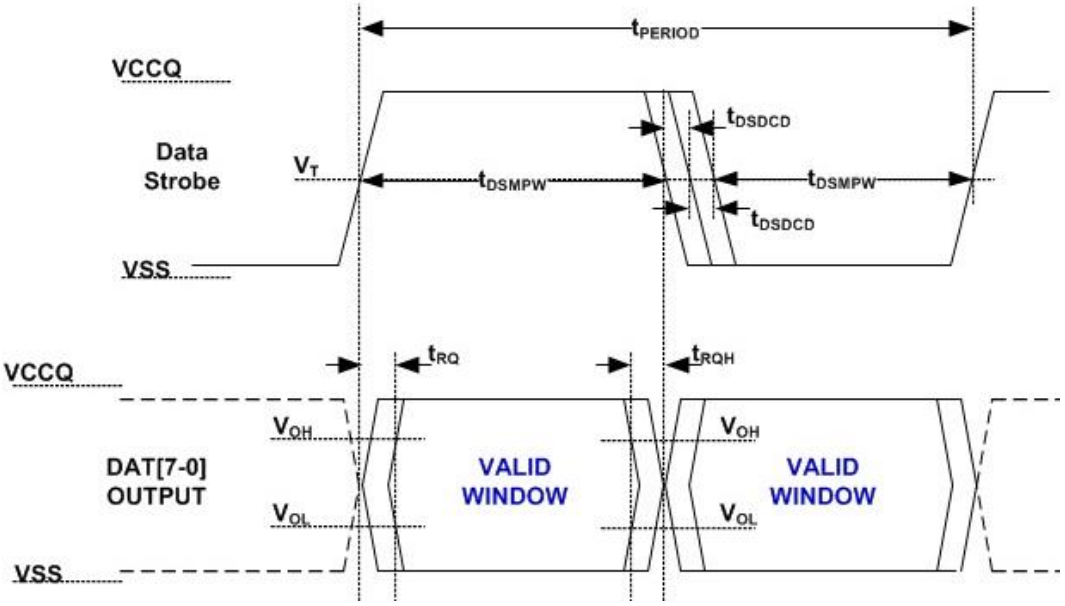
Figure 88 — HS400 Device Data input timing

Table 215 — HS400 Device input timing

Parameter	Symbol	Min	Max	Unit	Remark
Input CLK					
Cycle time data transfer mode	t_{PERIOD}	5			200 MHz(max), between rising edges With respect to V_T.
Slew rate	SR	1.125		V/ns	With respect to V_{IH}/V_{IL}.
Duty cycle distortion	t_{CKDCD}	0.0	0.3	ns	Allowable deviation from an ideal 50% duty cycle. With respect to V_T. Includes jitter, phase noise
Minimum pulse width	t_{CKMPW}	2.2		ns	With respect to V_T.
Input DAT (referenced to CLK)					
Input set-up time	t_{ISUddr}	0.4		ns	$C_{Device} \leq 6$ pF With respect to V_{IH}/V_{IL}.
Input hold time	t_{IHddr}	0.4		ns	$C_{Device} \leq 6$ pF With respect to V_{IH}/V_{IL}.
Slew rate	SR	1.125		V/ns	With respect to V_{IH}/V_{IL}.

10.10.2 HS400 Device Output Timing

The Data Strobe is used to read data in HS400 mode. The Data Strobe is toggled only during data read or CRC status response.



NOTE V_T = 50% of V_{CCQ}, indicates clock reference point for timing measurements.

Figure 89 — HS400 Device output timing

Table 216 — HS400 Device Output timing

Parameter	Symbol	Min	Max	Unit	Remark
Data Strobe					
Cycle time data transfer mode	t_{PERIOD}	5			200 MHz(max), between rising edges With respect to V_T
Slew rate	SR	1.125		V/ns	With respect to V_{OH}/V_{OL} and HS400 reference load
Duty cycle distortion	t_{DSDCD}	0.0	0.2	ns	Allowable deviation from the input CLK duty cycle distortion (t_{CKDCD}) With respect to V_T Includes jitter, phase noise
Minimum pulse width	t_{DSMPW}	2.0		ns	With respect to V_T
Read pre-amble	t_{RPRE}	0.4	-	t_{PERIOD}	Max value is specified by manufacturer. Value up to infinite is valid
Read post-amble	t_{RPST}	0.4	-	t_{PERIOD}	Max value is specified by manufacturer. Value up to infinite is valid
Output DAT (referenced to Data Strobe)					
Output skew	t_{RQ}		0.4	ns	With respect to V_{OH}/V_{OL} and HS400 reference load
Output hold skew	t_{RQH}		0.4	ns	With respect to V_{OH}/V_{OL} and HS400 reference load.
Slew rate	SR	1.125		V/ns	With respect to V_{OH}/V_{OL} and HS400 reference load

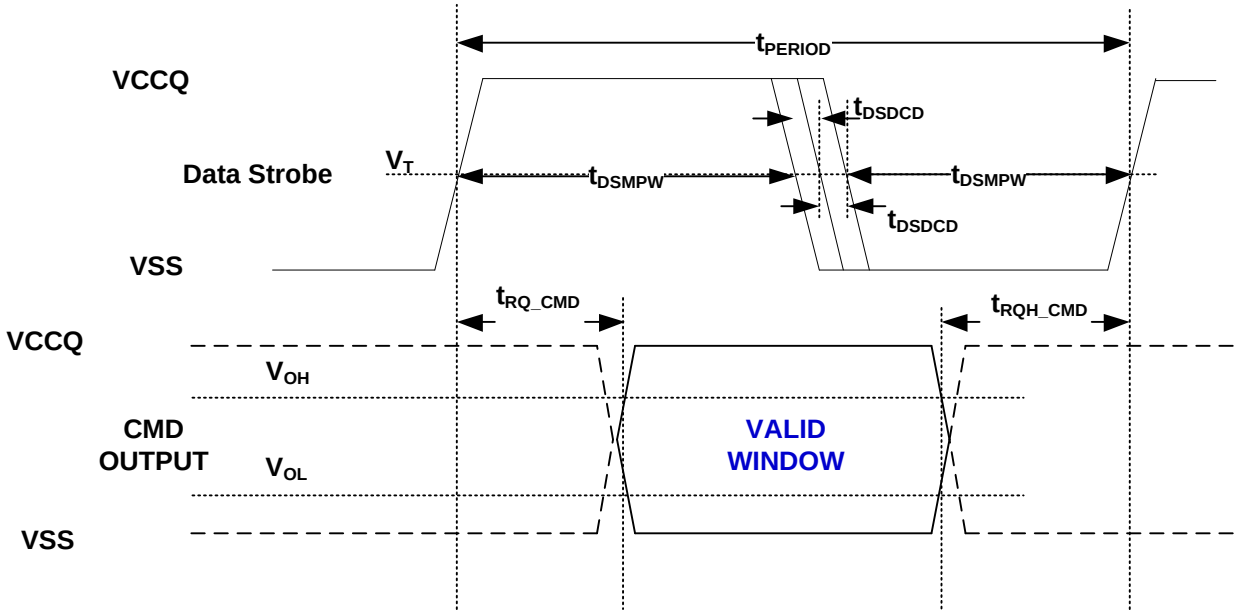
10.10.2 HS400 Device Output Timing (cont'd)

Table 217 — HS400 Capacitance and Resistors

Parameter	Symbol	Min	Max	Unit	Remark
Pull-up resistance for CMD	R_{CMD}	4.7	100(1)	kΩ	
Pull-up resistance for DAT0-7	R_{DAT}	10	100(1)	kΩ	
Pull-down resistance for Data Strobe	R_{DS}	10	100(1)	kΩ	
Internal pull up resistance DAT1-DAT7	R_{int}	10	150	kΩ	
Single Device capacitance	C_{Device}		6	pF	
NOTE 1 Recommended maximum value is 30 kΩ for 1.2 V and 50 kΩ for 1.8 V interface supply voltages.					

10.10.3 HS400 Device Command Output Timing

The Data Strobe is used to response of any command in HS400 mode.



NOTE V_T = 50% of V_{CCQ}, indicates clock reference point for timing measurements.

Figure 90 — HS400 CMD Response timing

Table 218 — HS400 CMD Response timing

Parameter	Symbol	Min	Max	Unit	Remark
Data Strobe					
Cycle time data transfer mode	t_{PERIOD}	5			200 MHz(max), between rising edges With respect to V_T
Slew rate	SR	1.125		V/ns	With respect to V_{OH}/V_{OL} and HS400 reference load
Duty cycle distortion	t_{DSDCD}	0.0	0.2	ns	Allowable deviation from the input CLK duty cycle distortion (t_{CKDCD}) With respect to V_T Includes jitter, phase noise
Minimum pulse width	t_{DSMPW}	2.0		ns	With respect to V_T
Read pre-amble	t_{RPRE}	0.4	-	t_{PERIOD}	Max value is specified by manufacturer. Value up to infinite is valid
Read post-amble	t_{RPST}	0.4	-	t_{PERIOD}	Max value is specified by manufacturer. Value up to infinite is valid
CMD Response (referenced to Data Strobe)					
Output skew(CMD)	t_{RQ_CMD}		0.4	ns	With respect to V_{OH}/V_{OL} and HS400 reference load
Output hold skew(CMD)	t_{RQH_CMD}		0.4	ns	With respect to V_{OH}/V_{OL} and HS400 reference load
Slew rate	SR	1.125		V/ns	With respect to V_{OH}/V_{OL} and HS400 reference load

11 *e*•MMC standard compliance

The *e*•MMC standard provides all the necessary information required for media exchangeability and compatibility.

- Generic Device access and communication protocol (clause 6, clause 7)
- Data integrity and error handling (clause 8)
- Mechanical interface parameters, Device form factor (clause 9)
- Electrical interface parameters, such as: power supply, peak and average current consumption and data transfer frequency (clause 10)

However, due to the wide spectrum of targeted *e*•MMC 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 *e*•MMC standard feature, in a specific *e*•MMC system. Therefore, many of the parameters are configurable and can be tailored per implementation.

A device is compliant with the standard as long as all of its configuration parameters are within the valid range. An *e*•MMC host is compliant as long as it supports at least one *e*•MMC class as defined below. Every provider of *e*•MMC system components is required to clearly specify (in its product manual) all the *e*•MMC specific restrictions of the device.

e•MMCs (slaves) provide their configuration data in the Device Specific Data (CSD) register (refer to 7.3). The *e*•MMC protocol includes all the necessary commands for querying this information and verifying the system concept configuration. *e*•MMC hosts (masters) are required (as part of the system boot-up process) to verify host-to-Device compatibility with each of the Devices connected to the bus.

Table 219 summarizes the requirements from a *e*•MMC host for each Device class (CCC = Device command class, see 6.10). The meaning of the entries is as follows:

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

11 *e*•MMC standard compliance (cont'd)

Table 219 — *e*•MMC host requirements for Device classes

Function	*e*•MMC	e^2•MMC
52 – 200 MHz transfer rate	Optional	Optional
26–52 MHz transfer rate	Mandatory	Mandatory
20–26 MHz transfer rate	Mandatory	Mandatory
0–26 MHz transfer rate	Mandatory	Mandatory
2.7 V–3.6 V power supply	Optional	Optional
1.70 V–1.95 V power supply	Optional	Optional
CCC 0 Basic	Mandatory	Mandatory
CCC 1 Sequential Read	Obsolete	Obsolete
CCC 2 Block Read	Mandatory	Mandatory
CCC 3 Sequential Write	Obsolete	Obsolete
CCC 4 Block Write	Mandatory	Mandatory
CCC 5 Erase	Mandatory	Mandatory
CCC 6 Write Protection Functions	Mandatory	Mandatory
CCC 7 Lock Device Commands	Mandatory	Mandatory
CCC 8 Application Specific Commands	Optional	Optional
CCC 9 Interrupt and Fast Read/Write	Optional	Optional
CCC10 Security Protocols	Optional	Optional
CCC11 Command Queuing	Optional	Optional
DSR	Optional	Optional
SPI Mode	Obsolete	n/a

Comments on the optional functions:

- The interrupt command is intended for reducing the overhead on the host side required during polling for some events.
- The setting of the DSR allows the host to configure the *e*•MMC bus in a very flexible, application dependent manner
- The external ECC in the host allows the usage of extremely low-cost Devices.
- The Device Status bits relevance, according to the supported classes, is defined in Table 69.
- Sanitize and bad block management are features that enable the device to be used in secure applications.
- The Trim and discard command allows the host to assist with the optimization of the internal Device garbage collection operations

11 *e*•MMC standard compliance (cont'd)

Table 220 — New Features List for device type

Function	*e*•MMC	e^2•MMC
Boot	Mandatory	Mandatory
RPMB	Mandatory	Mandatory
Reset Pin	Mandatory	Mandatory
Write Protection (including Perm & Temp)	Mandatory	Mandatory
1.2 V I/O	Optional	Optional
Dual Data Rate timing	Optional	Optional
HS200	Optional	Optional
Multi Partitioning	Mandatory	Mandatory
Secure Erase/Secure Trim	Optional	Optional
Trim	Mandatory	Mandatory
High Priority Interrupt	Mandatory	Mandatory
Background Operation	Mandatory	Mandatory
Enhance Reliable Write	Mandatory	Mandatory
Discard Command	Mandatory	Mandatory
Security Features	Mandatory	Mandatory
Partition types	Mandatory	Mandatory
Context ID	Mandatory	Mandatory
Data Tag	Mandatory	Mandatory
Packed commands	Mandatory	Mandatory
Real Time Clock	Mandatory	Mandatory
Dynamic Device Capacity	Mandatory	Mandatory
Power Off Notification	Mandatory	Mandatory
Thermal Spec	Mandatory	Mandatory
Minimum Sector Size = 4 KB (≤256 GB)	Optional	Optional
Minimum Sector Size = 4 KB (>256 GB)	Mandatory	Mandatory
Cache	Optional	Mandatory
Extended Security Protocols	Optional	Optional
HS400	Optional	Optional
Field Firmware Update	Optional	Optional
Product State Awareness	Optional	Optional
Secure Removal Type	Optional	Optional
Device Health Report	Optional	Optional
Command Queuing	Optional	Optional
Enhanced Strobe	Optional	Optional
Cache Flushing Report	Mandatory	Mandatory
BKOPS Control	Mandatory	Mandatory
Cache Barrier	Optional	Optional
RPMB Throughput Improve	Optional	Optional
Secure Write Protection	Optional	Optional

Annex A (informative) Application Notes

A.1 Device Payload block length and ECC types handling

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

– block length type and
– external ECC.

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

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

A.2 Description of method for storing passwords on the Device

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

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

Get the password (from the user, from a local storage on the device, or something else). The password can be of any length, and in any character set.

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

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

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

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

After running this string through SHA-1, it becomes:

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

Of which the first 128 bits are:

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

Which is then used as the password for the device.

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

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

A.3 *e*•MMC macro commands

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

Table A.221 — Macro commands

Mnemonic	Description
CIM_SINGLE_DEVICE_ACQ	Starts an identification cycle of a single Device.
CIM_SETUP_DEVICE	Select a Device by writing the RCA and reads its CSD.
CIM_READ_BLOCK	Sets the block length and the starting address and reads a data block from the Device.
CIM_READ_MBLOCK	Sets the block length and the starting address and reads (continuously) data blocks from the Device. Data transfer is terminated by a stop command.
CIM_WRITE_BLOCK	Sets the block length and the starting address and writes a data block from the Device.
CIM_WRITE_MBLOCK	Sets the block length and the starting address and writes (continuously) data blocks to the Device. Data transfer is terminated by a stop command.
CIM_ERASE_GROUP	Erases a range of erase groups on the Device.
Mnemonic	Description
CIM_TRIM	Erases a number of ranges of write blocks on the Device.
CIM_US_PWR_WP	Applies power-on write protection to a write protection group on the Device.
CIM_US_PERM_WP	Applies permanent write protection to a write protection group on the Device.

The *e*•MMC command sequences are described in the following paragraphs. Figure A.91 provides a legend for the symbols used in the sequence flow charts. The status polling by CMD13 can explicitly be done any time after a response to the previous command has been received.

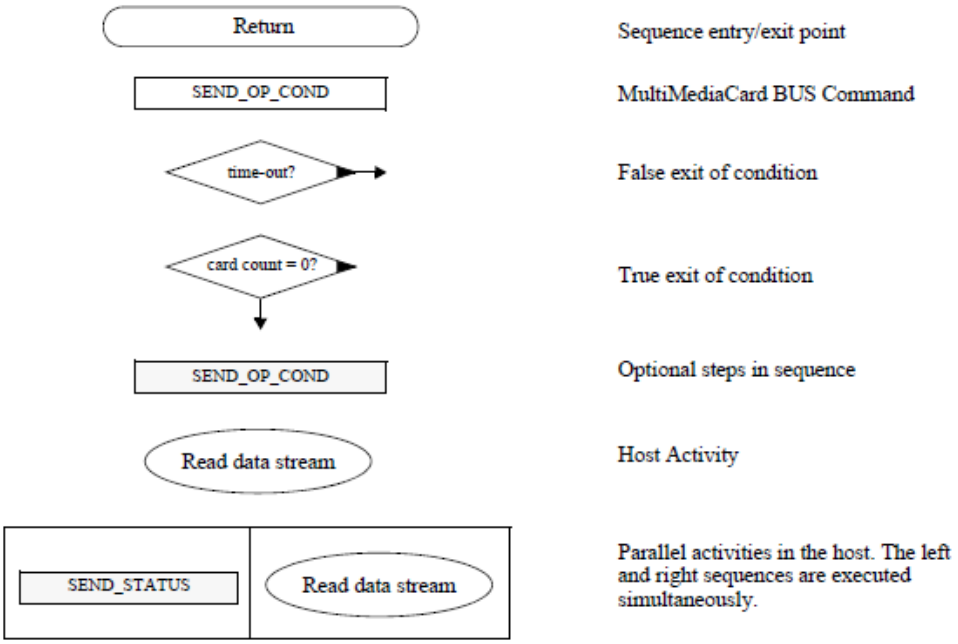


Figure A.91 — Legend for command-sequence flow charts

A.3 *e*•MMC macro commands (cont'd)

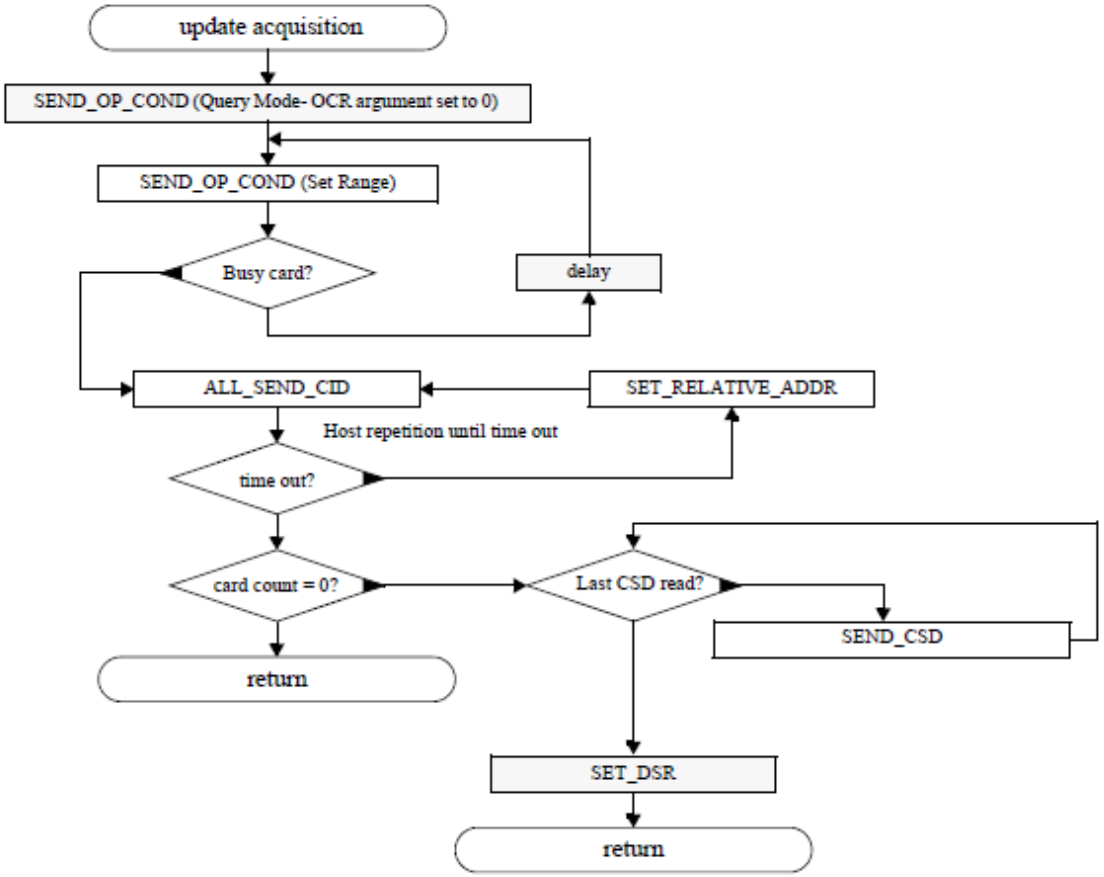


Figure A.92 — SEND_OP_COND command flow chart

A.3 *e*•MMC macro commands (cont'd)

- CIM_SINGLE_DEVICE_ACQ

The host knows that there is a single Device in the system and, therefore, does not have to implement the identification loop. In this case only one ALL_SEND_CID is required. Similarly, a single SEND_CSD is sufficient.

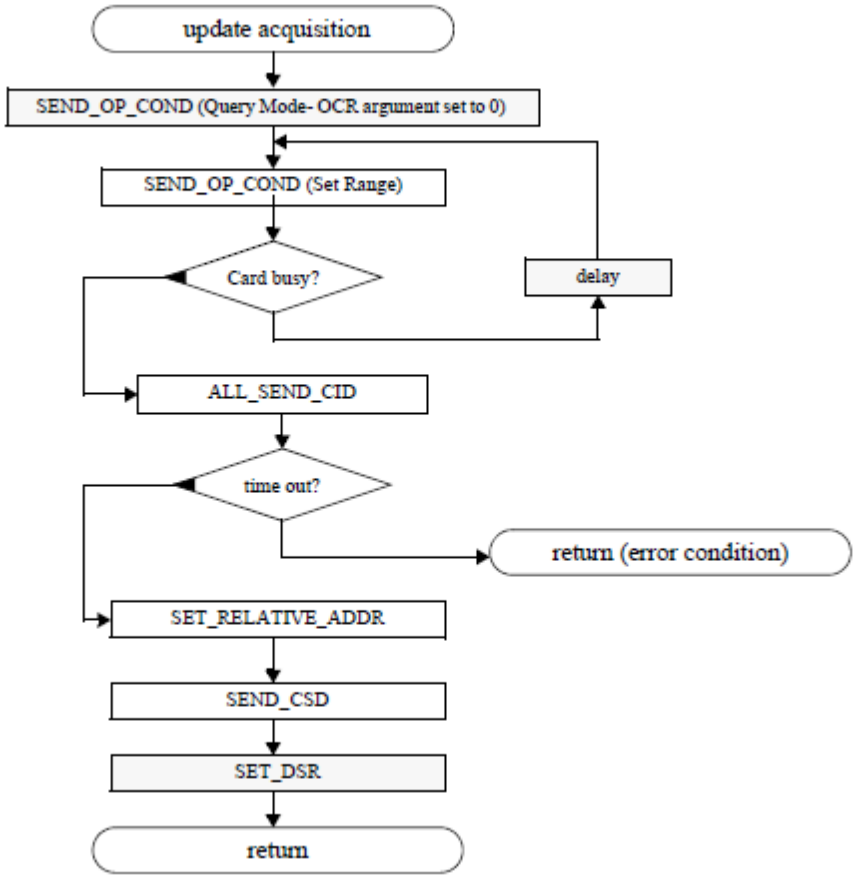


Figure A.93 — CIM_SINGLE_DEVICE_ACQ

A.3 *e*•MMC macro commands (cont'd)

- CIM_SETUP_DEVICE

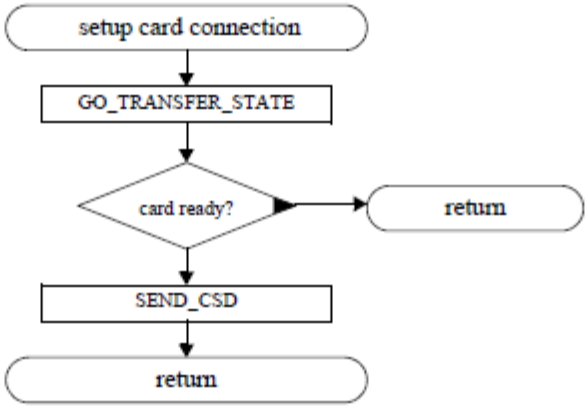


Figure A.94 — CIM_SETUP_DEVICE

The setup Device connection procedure (CIM_SETUP_DEVICE) links the bus master with a single Device. The argument required for this command is the RCA of the chosen Device. A single Device is selected with GO_TRANSFER_STATE (CMD7) command by its RCA. The response indicates whether the Device is ready or not. If the Device confirms the connection, the adapter will read the Device specific data with SEND_CSD (CMD9). The information within the response is used to configure the data path and controller options.

A.3 *e*•MMC macro commands (cont'd)

- CIM_READ_BLOCK

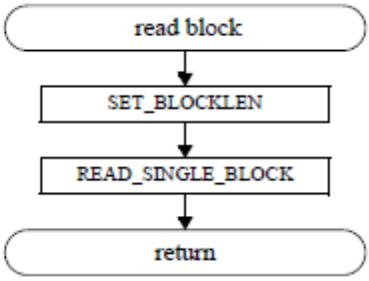


Figure A.95 — CIM_READ_BLOCK

The read block procedure (CIM_READ_BLOCK) reads a data block from a Device. The arguments required for this command are the block length (4 bytes) and the starting address of the block (4 bytes). This operation also includes a data portion (in this case, the read block). The procedure starts by setting the required block length with the SET_BLOCKLEN (CMD16) command. If the Device accepts this setting, the data block is transferred via command READ_SINGLE_BLOCK (CMD17), starting at the given address.

- CIM_READ_MBLOCK

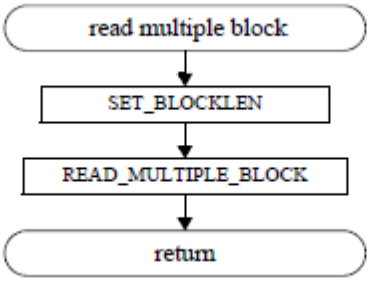


Figure A.96 — CIM_READ_MBLOCK

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

- CIM_WRITE_BLOCK

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

A.3 *e*•MMC macro commands (cont'd)

- CIM_WRITE_MBLOCK

The sequence of write multiple block starts with an optional SET_BLOCK_LEN command. If there is no change in block length this command can be omitted. If the Device accepts the two starting commands the host will begin sending data blocks on the data line. After each data block the host will check the Device response on the DAT line. If the CRC is OK, the Device is not busy and the host will send the next block if there are more data blocks.

While sending data blocks, the host may query the Device status register (using the SEND_STATUS conned) to poll any new status information the Device may have (e.g., WP_VIOLATION, MISALIGMENT, etc.) The sequence must be terminated with a STOP command.

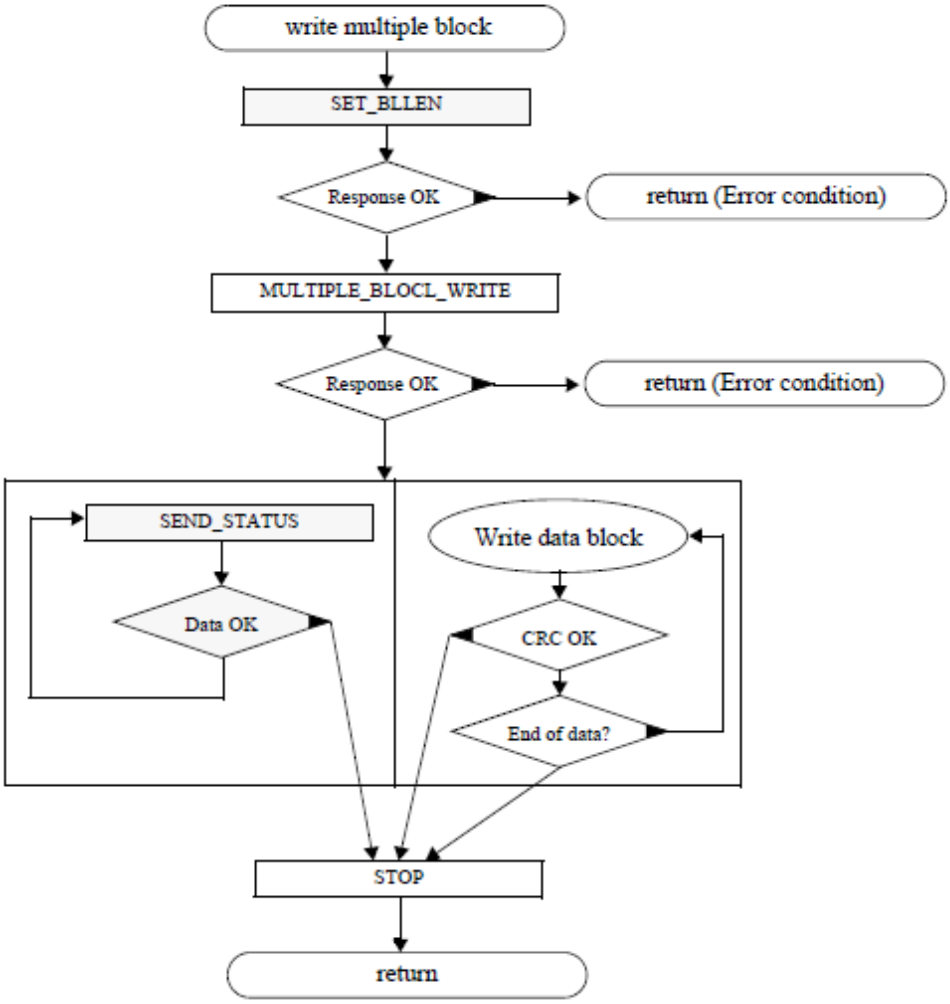


Figure A.97 — CIM_WRITE_MBLOCK

A.3 *e*•MMC macro commands (cont'd)

- CIM_ERASE_GROUP

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

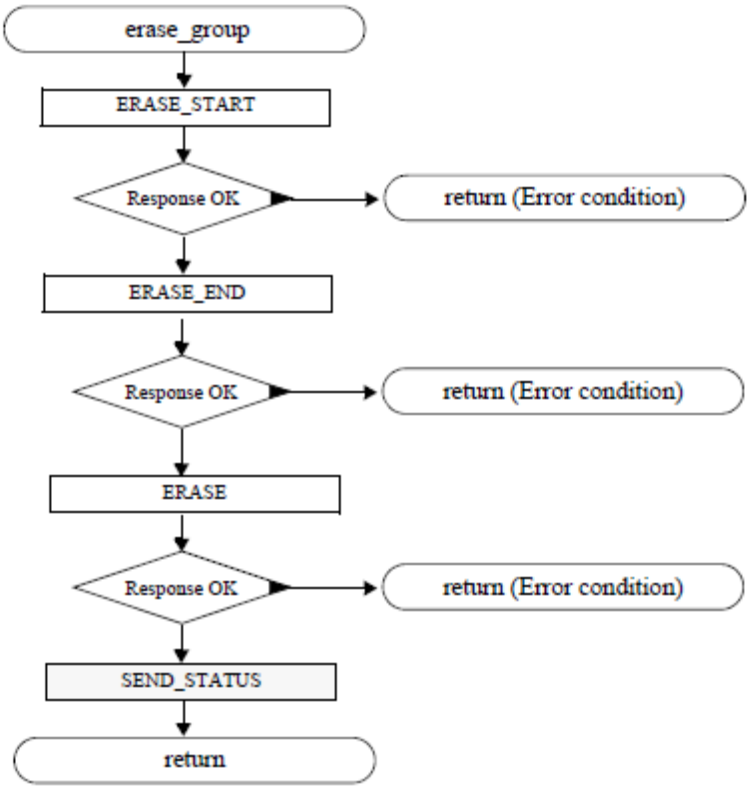


Figure A.98 — CIM_ERASE_GROUP

A.3 *e*•MMC macro commands (cont'd)

- CIM_TRIM

The trim procedure starts with ERASE_START (CMD35) and ERASE_END (CMD36) commands, these commands are used to select write block. Once the write blocks are selected the host will send an ERASE (CMD38) command with argument bit 0 set to one and the remainder of the bits set to zero. It is recommended that the host terminates the sequence with a SEND_STATUS (CMD13) to poll any additional status information the Device may have (e.g., WP_ERASE_SKIP, etc.).

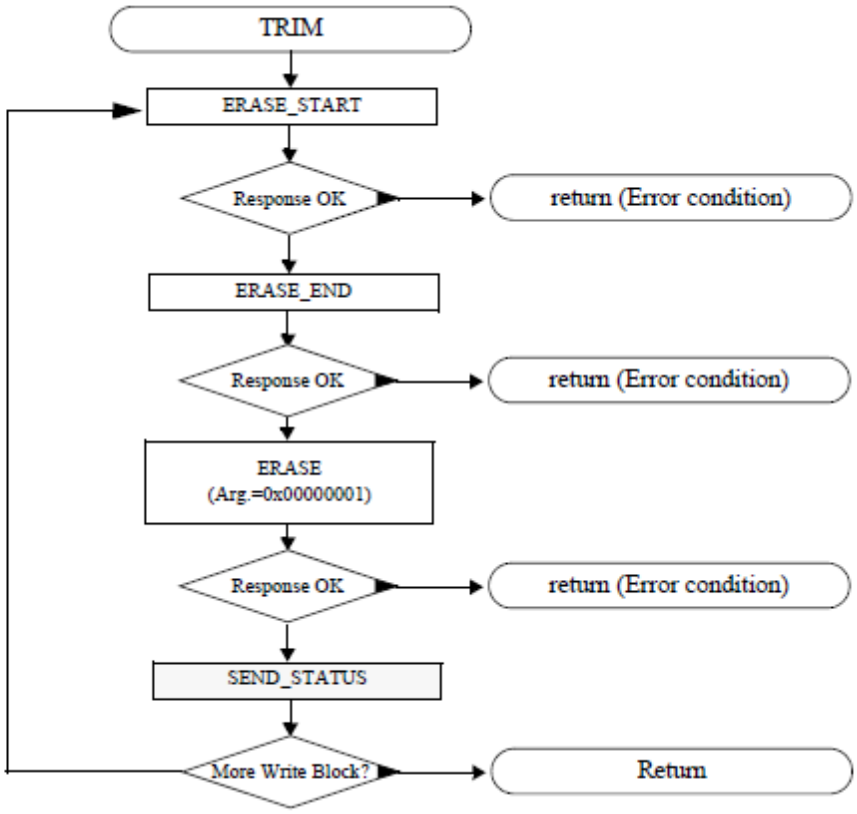


Figure A.99 — CIM_TRIM

A.3 *e*•MMC macro commands (cont'd)

• CIM_US_PWR_WP

The minimum required sequence to apply Power-On write protection to a write protection group is to set US_PWR_WP_EN (EXT_CSD[171] bit 0) and then use the SET_WR_PROT(CMD28) command.

It is recommended to disable permanent write protection, if it is not needed, before issuing the first power-on write protection sequence since if an area is permanently protected then power-on write protection cannot be applied.

The host can check if power-on protection has been disabled before following the minimum required sequence to apply power-on protection by reading US_DIS_PWR_WP (EXT_CSD[171] bit 3). Also, the host can verify the protection status of the write group after the required sequence has been executed by using the SEND_WR_PROTECT_TYPE (CMD31) command.

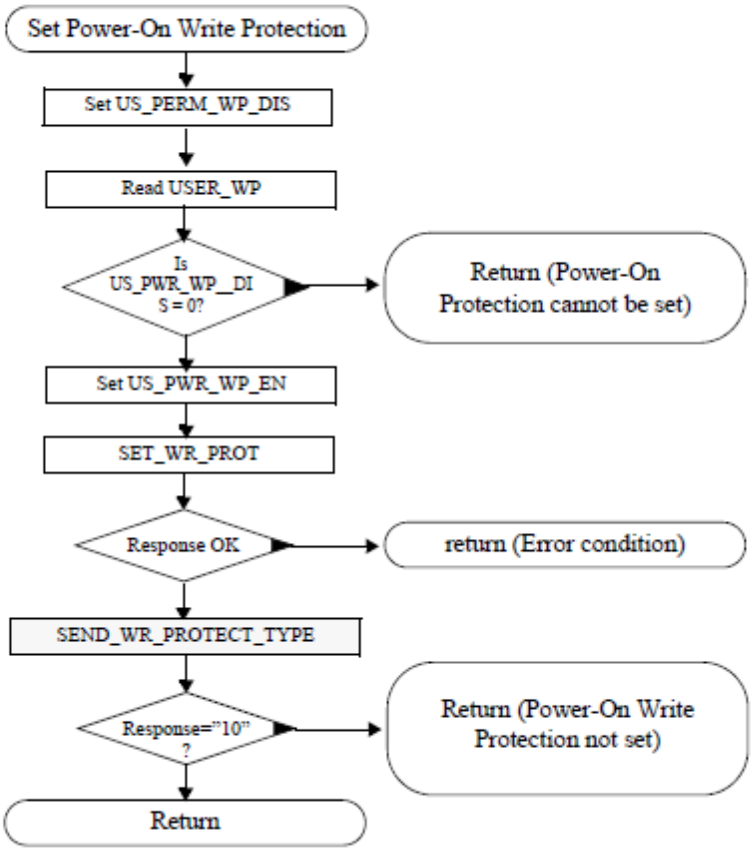


Figure A.100 — CIM_US_PWR_WP

A.3 *e*•MMC macro commands (cont'd)

- CIM_US_PERM_WP

The minimum required sequence to apply permanent write protection to a write protection group is to set US_PERM_WP_EN (EXT_CSD[171] bit 2) and then use the SET_WR_PROT(CMD28) command.

The host has the option to check that permanent protection is not disabled before setting permanent write protection by reading US_DIS_PERM_WP (EXT_CSD[171] bit 4). Also, the host can verify the protection status of the write group after the required sequence has been executed by using the SEND_WR_PROTECT_TYPE (CMD31) command.

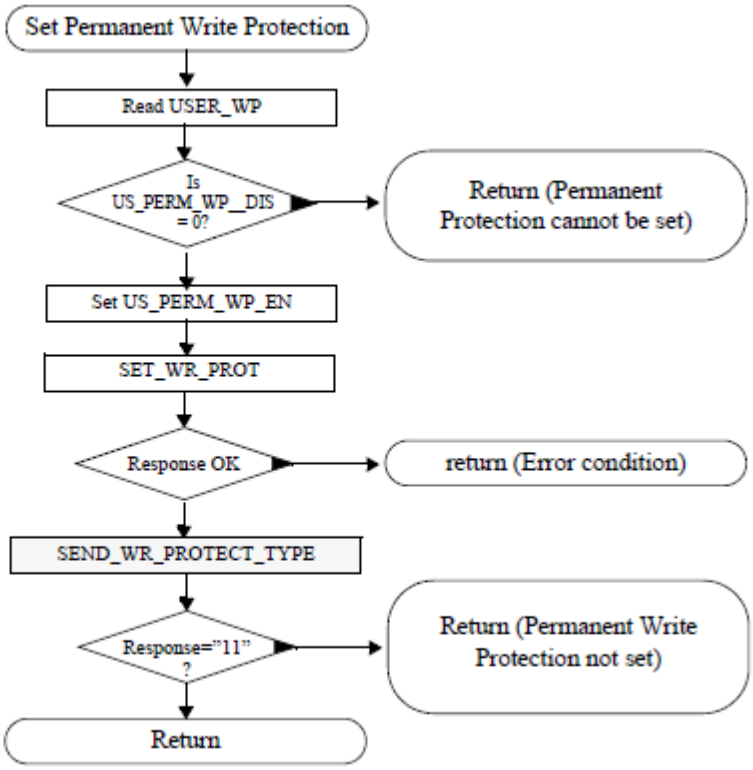


Figure A.101 — CIM_US_PERM_WP

A.4 Host interface timing

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

Table A.222 defines the forward compatibility interface timing. The high speed interface timing is already defined in Table 208.

Table A.222 — Forward-compatible host interface timing

Parameter	Symbol	Min	Max	Unit	Remark
Clock CLK[1]					
Clock frequency Data Transfer Mode (PP)	f_{PP}	0	20	MHz	C_L <= 30 pF
Clock frequency Identification Mode (OD)	f_{OD}	0	400	kHz	
Clock low time	t_{WL}	10		ns	C_L <= 30 pF
Clock rise time[2]	t_{TLH}		10	ns	C_L <= 30 pF
Clock fall time	t_{THL}		10	ns	C_L <= 30 pF
Inputs CMD, DAT (referenced to CLK)					
Input set-up time	t_{ISU}	4.8		ns	C_L <= 30 pF
Input hold time	t_{IH}	4.4		ns	C_L <= 30 pF
Outputs CMD, DAT (referenced to CLK)					
Output set-up time	t_{OSU}	5		ns	C_L <= 30 pF
Output hold time	t_{OH}	5		ns	C_L <= 30 pF
NOTE 1 All timing values are measured relative to 50% of voltage level NOTE 2 Rise and fall times are measured from 10%-90% of voltage level.					

A.5 Handling of passwords

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

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

A.5.1 Changing the password

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

- First, remove the old password
- Second, set the new password

A.5.2 Removal of the password

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

A.6 High-speed *e*•MMC bus functions

A.6.1 Bus initialization

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

High Speed *e*•MMCs are backwards compatible, therefore after power up, they behave identically to old Devices, with no visible difference.[4] The steps a host can do to identify a High Speed *e*•MMC, and to put it to high speed mode are described next, from power-up until the Device is ready to work at high data rates.

a. Power-up
   1. Apply power to the bus, communication voltage range (1.70 V – 1.95 V or 2.7 V – 3.6 V/ 1.1 V – 1.3 V opt)
   2. Set clock to 400KHz, or less
   3. Wait for 1ms, then wait for 74 more clock cycles , voltage supply ramp-up time, boot operation duration (see Figure 73)
   4. Send CMD1with the address mode required by the host (0x80FF8080 - capacity less than or equal to 2GB – or 0xC0FF8080 - capacity greater than 2GB)
   5. Receive R3
   6. If the OCR busy bit is "0"(0x00FF8080 or 0x40FF8080 in R3 response and the host shall ignore the access bits), repeat steps 5 and 6
   7. When the device returns 0x80FF8080 or 0xC0FF8080, it moves to Ready State if compatible to the host CMD1 argument
   8. If R3 returned some other value, the Device is not compliant (since it should have put itself into *inactive* state, due to voltage incompatibility, and not respond); in such a case the host must power down the bus and start its error recovery procedure (the definition of error recovery procedures is host dependent and out of the scope of this application note)

Low-voltage power-up

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

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

b. CID retrieval and RCA assignment
   15. Send CMD2
   16. Receive R2, and get the Device's CID
   17. Send CMD3 with a chosen RCA, with value greater than 1

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

A.6.1 Bus initialization (cont'd)

c. CSD retrieval and host adjustment

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

A.6.2 Switching to high-speed mode

The following steps are supported by Devices implementing version 4.0 or higher. For switching to HS200 mode introduced in v4.5 refer to 6.6.2.2. Do these steps after the bus is initialized according to A.6.1, Bus initialization.

21. Send CMD7 with the Device's RCA to place the Device in tran-state
22. Send CMD8, SEND_EXT_CSD. From the EXT_CSD the host can learn the power class of the Device, and choose to work with a wider data bus (See steps 26-37)
23. Send CMD6, writing 0x1 to the HS_TIMING byte of the EXT_CSD. The argument 0x03B9_0100 will do it.

    23.1-The Device might enter BUSY right after R1, if so, wait until the BUSY signal is de-asserted

    23.2-After the Device comes out of BUSY it is configured for high speed timing

24. Change the clock frequency to the chosen frequency (any frequency between 0 and 26/52MHz).

A.6.3 Changing the data bus width

The following steps are optionally done if the Device's power class allows the host to work on a wider bus, within the host power budget. Do these steps after the bus is initialized according to A.6.1 Bus initialization.

a. Bus testing procedure

25. Send CMD19
26. Send a block of data, over all the bus data lines, with the data pattern as follows (CRC16 is optional):

    26.1-For 8 data lines the data block would be (MSB to LSB): 0x0000_0000_0000_AA55

    26.2-For 4 data lines the data block would be (MSB to LSB): 0x0000_005A

    26.3-For only 1 data line the data block would be: 0x80

A.6.3 Changing the data bus width (cont'd)

Table A.223 — Bus testing for eight data lines

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

Table A.224 — Bus testing for four data lines

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

Table A.225 — Bus testing for one data line

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

27. Wait for at least NCR clock cycles before proceeding
28. Send CMD14 and receive a block of data from all the available data lines[5]

    28.1-For 8 data lines receive 8 bytes

    28.2-For 4 data lines receive 4 bytes

    28.3-For 1 data line receive 1 byte
29. XNOR the masked data with the data sent in step 27

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

A.6.3 Changing the data bus width (cont'd)

Table A.226 — XNOR values

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

30. Mask the result according to the following:

    30.1-For 8 data lines the mask is (MSB to LSB): 0x0000_0000_0000_FFFF

    30.2-For 4 data lines the mask is (MSB to LSB): 0x0000_00FF

    30.3-For 1 data line the mask is 0xC0

31. The result should be 0 for all. Any other result indicates a problem in the Device connection to the system; in such a case the host must power down the bus and start its error recovery procedure (the definition of error recovery procedures is host dependent and out of the scope of this application note)

b. Power and bus-width selection

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

A.7 Erase-unit size selection flow

The flow chart in Figure A.102 shows how the master selects the erase unit size if the master supports the JEDEC MMC Electrical Interface standard v4.3.

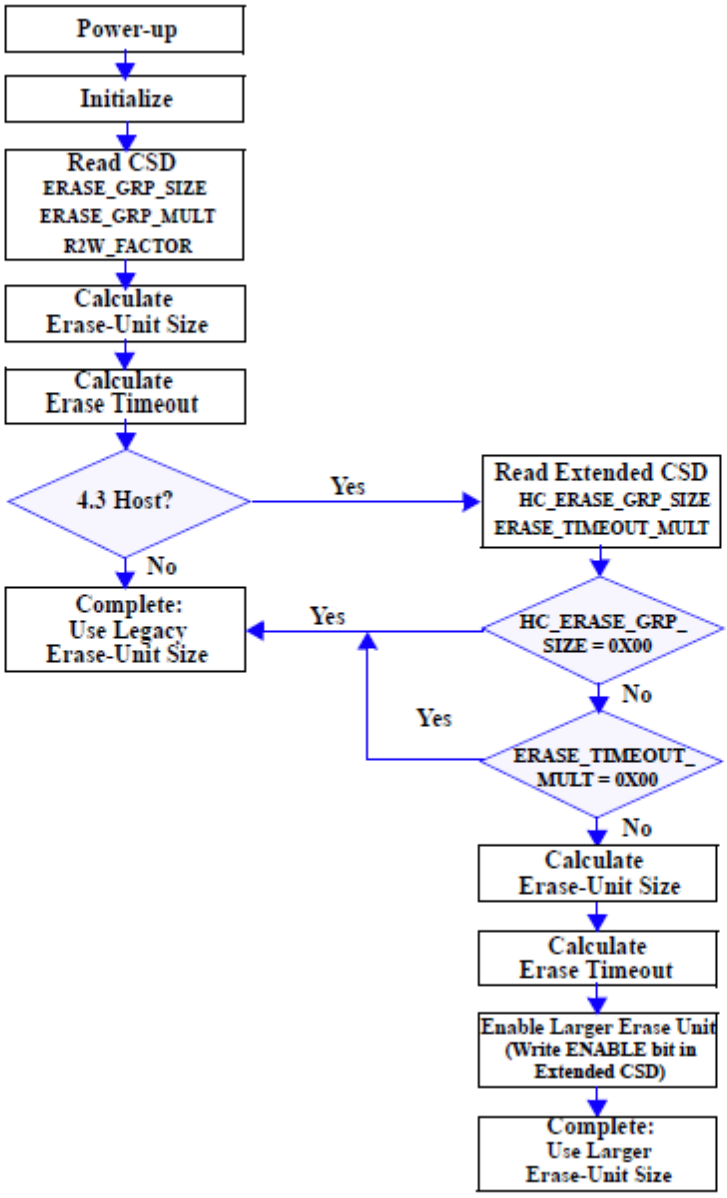


Figure A.102 — Erase-unit size selection flow

A.8 HPI background and one of possible solutions

- **Background - issues with HPI**

There are concerns with the High Priority Interrupt (HPI).

One case is when the interruptible operation (e.g., write) is being repeatedly interrupted in order to allow execution of the High Priority Read operations. There is concern that the delaying the write operation for too long time can stall the host system. If it really would be a problem or not is hard to predict because of different system characteristics; different types of applications that will be launched by High Priority Read, different types of applications that might have their write operations interrupted, different timing of the systems, etc.

Another case is when a process with high priority has requested a write operation and a process with lower priority has requested a new code page, and in the case that the write operation is being interrupted by the HPI, we are facing a priority violation against the intention of the scheduling defined in the system.

- **One of possible solutions**

There might be different solutions to resolve the observed issues with the HPI.

One of possible solutions to the observed issues is to resolve a conflict at the host by letting the write operation come through when a dedicated timer has been expired. With an adjustable timeout value depending on the context, flexibility is gained making it possible to adopt the method for different needs.

Using the following examples, a possible method is explained for the two most interesting use cases. Assume that we want to protect the write commands not being delayed by more than 1s after the write command has been originally sent to the memory device. In such case we need to set the timeout value 1s for each write command. A timer will be started when the write command has been sent to the memory device. If HPI is requested while the write command is still ongoing, the driver in the host will check the timer value. If the timer value has expired compared to the timeout (1s), the HPI will not be able to interrupt the write command. On the other hand, if the timer value is lower than the timeout (1s), then the HPI will interrupt the write operation and the requested read operation will be provided. When the write operation has finally been concluded, the timer will be reset.

Assume another case, protecting high priority write operations in Linux. When a write operation has been requested, the check will be made of the priority of the process that has requested this operation. If it is a real time process or a kernel process, the timeout value will be set to zero and a timer will be started in the same way as in the example above. If HPI is requested while the write command is still ongoing, the driver in the host will check the timer value. The timer value will be checked towards the timeout (0s), indicating the expiration of the timer that means that the write command will never be interrupted. When the write operation has been concluded, the timer will be reset.

A.9 Stop transmission timing

Previous versions of the *e*•MMC spec did not describe the timing of the stop command in all different device states. Also, they did not fully specify whether the received block is valid, after the stop command is received, for all cases. The following section is intended to clarify this behavior for future designs. However, since these clarifications did not exist for previous versions of the spec not all devices may adhere to this clarification. It is for this reason that a host should take the necessary precautions to ensure that the STOP command functions as expected in its design and should not rely on the following behavior. Figure A.103 describes busy signal timing of the stop command just before CRC status transfer from the device. Considering backward compatibility, the end bit of CRC status is followed by two Z clocks even the device doesn't need two Z bits between CRC status and busy signal. This behavior is based on Figure 44-Stop transmission during CRC status transfer from the device. However, some devices may not to adhere to Figure A.103. It is for this reason that a host should ensure behavior of the device and is recommended to indicate busy signal 4 clock cycles after STOP command.

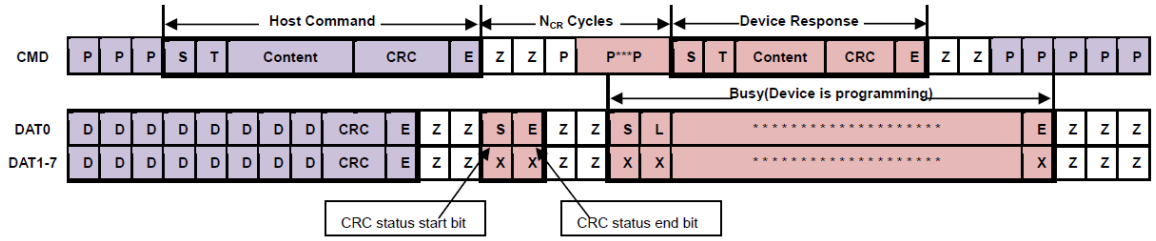


Figure A.103 — Stop transmission just before CRC status transfer from the device

Figure A.104 and Figure A.105 describe data block validity. End bit which is followed 2 clock cycles after STOP command on Figure A.104 follows CRC status, however, device considers CRC status is interrupted. For this reason, the received data block on Figure A.104 is considered incomplete and will not be programmed.

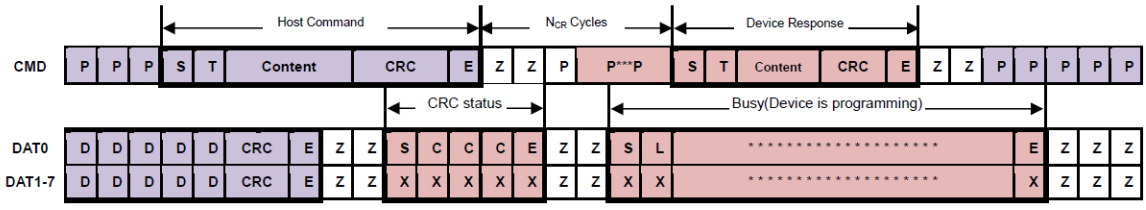
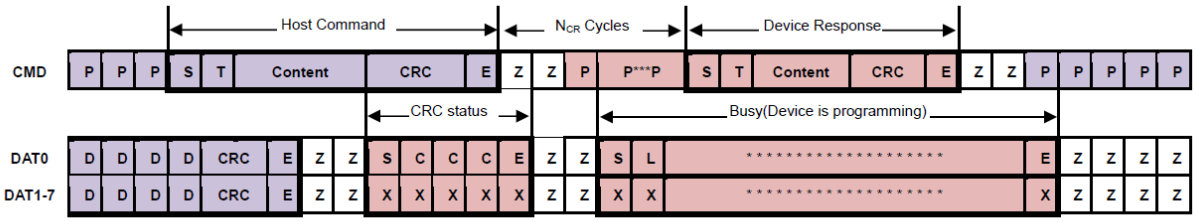


Figure A.104 — Stop transmission during CRC status transfer from the device

A.9 Stop transmission timing (cont'd)

The received data block on Figure A.105 is considered complete and the device will program it.



NOTE For Stop transmission in HS200 Mode, the time from the End bit of Stop transmission until the Busy starts (named Nsb-a) may vary. Please refer to 6.15.8, "Timing Changes in HS200 Mode", for further info on HS200 case."

Figure A.105 — Stop transmission during CRC status transfer from the device

A.10 Temperature Conditions per Power Classes (T_{case} controlled)

Increasing *e*•MMC performance increases the heat generated by the device.

Simulation data of future *e*•MMC devices operating in higher performance shows:

a. Memory and controller junction temperature may exceed its maximum allowed limit for silicon device operation if natural air convection is relied upon as the only means of heat removal and Ambient Temperature limit is the only given standard spec parameter
b. In addition, system level solution designed by host manufactures, such as removing heat from the top surface of the package, is required to improve the thermal management challenge

In order to enable more efficient system solution utilizing high speed operations following set of package case temperature per different current consumptions is defined.

It is assumed that Device vendors may specify in data sheets the Max performance supported per certain Power Class and condition the maximum available performance upon system solution that conforms to the T_c temperatures given in Table A.227. Hosts that conform to the given table shall set the TCASE_SUPPORT bits in the EX_CSD register. If TCASE_SUPPORT bits are not set(= "00") device may limit its maximum performance levels as a self-protection mechanism.

T_c: package case temperature

T_J: junction temperature

q_{JC}: thermal resistance between junction & package case

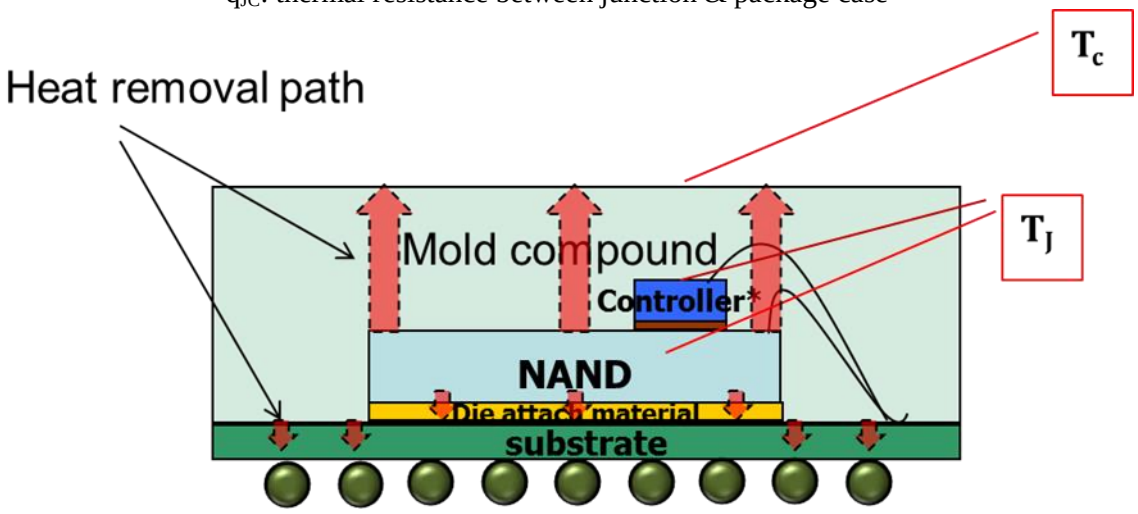


Figure A.106 — Heat Removal - Nomenclatures

Heat path through conduction from solder-balls into PCB will be dependent upon the host PCB designs.

Two cases were assumed:

- Heat removal through package case is the main path
- Heat removal is 50% through case and 50% through Balls/PCB.

A.10 Temperature Conditions per Power Classes (Tcase controlled) (cont'd)

Table A.227 provides set of T_c temperatures for the various power class current levels.

Package surface to be held at or below the T_c temperature shown for given current consumption.

Having T_c defined provides a reference for Device manufacturers and Host vendors assuming that each will take its own responsibility on the following:

- Memory Device Vendors: Making thermally efficient packages: Assuring the storage device junction temp never to exceed its maximum T_j as long as T_c is kept per *e*•MMC as shown in Table A.227.
- Host Vendors: Efficient cooling system to assure that the package case temp never exceeds the T_c as defined in Table A.227.

Table A.227 — Package Case Temp (Tc) per current consumption

Max RMS Current (mA) / Pkg Case Temp T_c (°C) for Standard Package Type "14x18" & "12x16"	100	150	200	250	300	350	400	450
Heat relief through package case : 100% Heat relief through Balls/PCB : 0%	85	85	85	85	75	75	75	75
Heat relief through package case : 50% Heat relief through Balls/PCB : 50%	85	85	85	85	85	85	85	85

NOTE Packages types are as defined in JEDEC, MO-276D. "14x18" = Package type "AC" and "12x16" = Package type "AA"

A.11 Handling write protection for each boot area individually

The following steps describe how to enable power-on write protection to a boot area when the other boot area is permanently write protected.

1) When permanent write protection was applied to BOOT Area 2, BOOT_WP register would be like:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
B_ _SEC_WP_ SEL	B_PWR_WP_ DIS	Reserved	B_PERM_WP_DI S	B_PERM _WP_SEC _SEL	B_PERM_WP_EN	B_PWR_ WP_SEC _SEL	B_PWR_WP_EN
1	0	x	0	1	1	0	0

2) According to 7.4.49, some bits of BOOT_WP register can only be written once per power cycle, the device power shall be turned off and on again to apply write protection to the other Boot area. BOOT_WP register value would be like below after power cycle:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
B_ _SEC_WP_ SEL	B_PWR_WP_ DIS	Reserved	B_PERM_WP_DI S	B_PERM _WP_SEC _SEL	B_PERM_WP_EN	B_PWR_ WP_SEC _SEL	B_PWR_WP_EN
0	0	x	0	0	1	0	0

The host should take notice that B_SEC_WP_SEL(bit 7) and B_PER_WP_SEC_SEL(bit3) turned to zero as they are type R/W/C_P.

3) In order to apply power-on write protection to BOOT Area 1, following values shall be set:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
B_ _SEC_WP_ SEL	B_PWR_WP_ DIS	Reserved	B_PERM_WP_DI S	B_PERM _WP_SEC _SEL	B_PERM_WP_EN	B_PWR_ WP_SEC _SEL	B_PWR_WP_EN
1	0	x	0	1	1	0	1

When trying to independently power-on write protect the other boot partition, the configuration of bit3 in EXT_CSD173 is required to be in agreement with the value read back from BOOT_WP_STATUS[174].

Especially in this case, Boot Area 2 was previously permanently write protected, two bits B_SEC_WP_SEL(bit7) and B_PER_WP_SEC_SEL(bit3) must be set. Otherwise, both Boot areas will be permanent write protected unintentionally.

A.12 Field Firmware Update

Figure A.107 describes the positive oriented FFU flow.

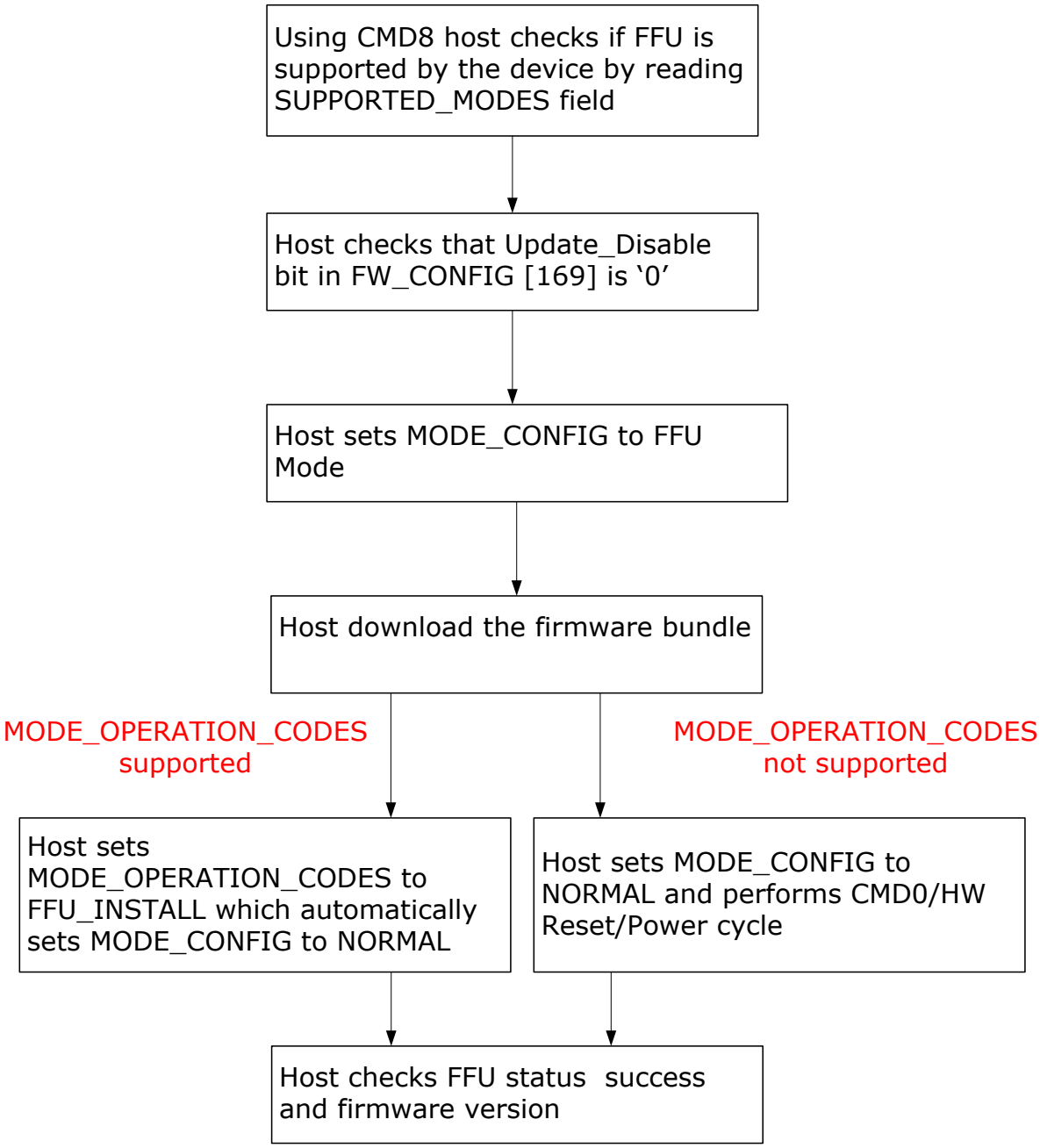


Figure A.107 — FFU flow

A.13 Command Queue: Command Flows (Informative)

The commands sequences involved in queuing commands and executing tasks are described in the following subsections.

A.13.1 Queuing a Transaction (CMD44+CMD45)

To queue a data transfer task the host issues a CMD44 followed by a CMD45. CMD44 encodes the Block Count, Task ID, Data Direction, and Priority. CMD45 encodes the transaction LBA. The host shall make sure that the Task ID encoded in CMD44 represents an ID, which is not already in use in the device's queue.

The device responds to each command with an R1-type response. The response indicates if the transaction is received and the task is queued successfully.

In order to queue a task, both CMD44 and CMD45 must be issued in sequence. If CMD44 is not successful (e.g., the requested Task ID is already in use), the device treats the command as illegal and does not respond. In that case, CMD45 must not be issued by the host.

If CMD45 is not issued after CMD44, the queuing operation is cancelled.

After CMD45 is issued successfully and an R1 response is returned, the task is considered queued in the device.

CMD44 and CMD45 may be issued while data transfer is on-going on the DAT lines or when the device indicates BUSY state or when the lines are idle (e.g., during NAC time).

The host may queue any number of tasks using the CMD44+CMD45 flow, as long as the queue is not full (i.e., there are Task IDs available).

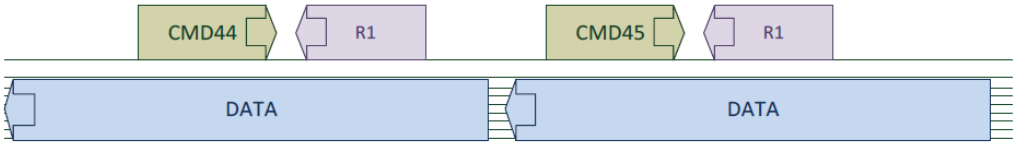


Figure A.108 — Queuing a transaction

A.13.2 Checking the Queue Status (SEND_STATUS - CMD13)

In order to determine, which tasks are ready for execution, the host can read the Queue Status Register (QSR). In order to read the QSR, the host issues SEND_STATUS, with bit[15] in its argument set. The device responds with an R1 with the QSR in its argument.

SEND_STATUS may be issued while data transfer is on-going on the DAT lines or when the device indicates BUSY state or when the lines are idle (e.g., during NAC time).

A.13.3 Execution of a Queued Task (CMD46/CMD47)

In order to execute a previously queued task, the host issues a CMD46 or a CMD47, encoding the Task ID of the requested task. CMD46 is used for the execution of Read commands, and CMD47 is used for Write commands. The host shall only issue CMD46/CMD47 for tasks, which are shown as “ready for execution” in the QSR.

If the task is ready for execution, and its Data Direction matches the command index (CMD46 for Read, CMD47 for Write) the device issues an appropriate R1 response and the data transfer starts. If there is any error, the device does not respond and no data transfer takes place.

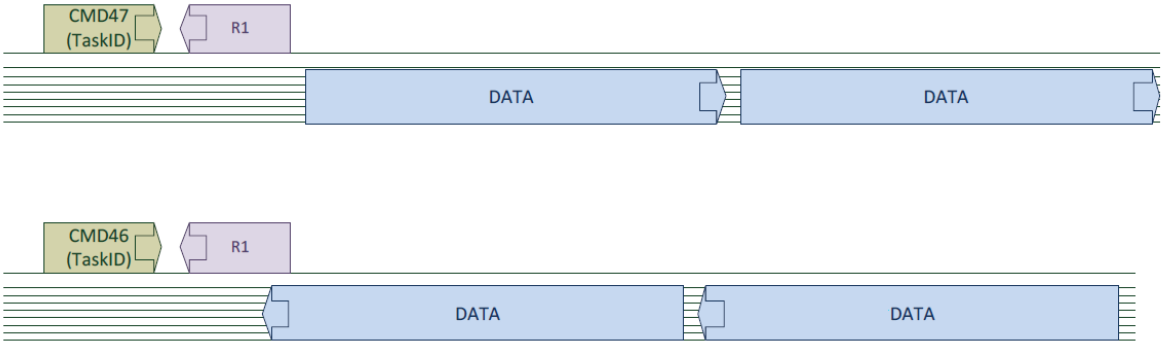


Figure A.109 — Execution of a queued task

Annex B (Normative) Host Controller Interface for Command Queuing

B.1. Introduction

B.1.1. Background

Command Queuing (CQ) feature is introduced to *e*•MMC standard in v5.1. CQ includes new commands for issuing tasks to the device, for ordering the execution of previously issued tasks, and for additional task management functions.

In order to optimally exploit CQ, the partition between hardware and software in the host should be designed such that host hardware executes the bus protocol and provides a task-level interface to software and host software issues tasks to the hardware and is notified when they are completed.

B.1.2. Overview and Scope

This annex describes the Command Queuing Host Controller Interface (CQHCI): the hardware/ software interface of a module which is capable of executing the hardware functions related to CQ. These functions are: processing task information provided by software, communication with the device using the bus protocol for issuing tasks and ordering task execution, copying data to/from the system memory, and generation of interrupts.

The objective of CQHCI is to provide a uniform interface method of accessing the Command Queuing hardware functions in *e*•MMC so that a standard or common software driver can be provided for these functions. The common driver would work with Command Queuing hardware from any vendor. This annex includes a description of the hardware/software interface between system software and the CQ hardware in the host controller.

The specification described in this annex is intended for hardware designers, system builders and software developers.

B.1.3. Feature Summary

The Command Queuing HCI has the following features:

- Task-level protocol between host software and host hardware,
- Non-blocking issuance of data transfer tasks using a task list in host memory,
- Optional notification upon completion via interrupt, with interrupt coalescing,
- Direct-Command (DCMD) tasks through which software can send any *e*•MMC command, by its index and argument, to the device, using CQHCI,
- Queue-Barrier (QBR) feature through which software can control the execution order of tasks, and
- Task management and error recovery executed by software, by halting the CQ hardware at known locations.

B.1.4. Outside of Scope

This annex does not contain information relevant to implementation of an *e*•MMC host controller. The *e*•MMC host controller, the methods which software uses it to manage the *e*•MMC bus protocol, device control functions, such as power management, and the host controller's software interface are left for implementation.

B.1.5. Acronyms and Conventions

B.1.5.1. Acronyms

The following acronyms are used in this annex:

CQ	Command Queuing
CQE	Command Queuing Engine
CQHCI	Command Queuing Host Controller Interface
DCMD	Direct Command
QBR	Queue Barrier
QSR	Queue Status Register
SQS	Send Queue Status
TDL	Task Descriptor List

B.1.5.2. Conventions

The conventions used for registers in this annex are defined.

Hardware shall return ‘0’ for all bits and registers that are marked as reserved, and host software shall write all reserved bits and registers with the value of ‘0’.

Inside the register section, the following abbreviations are used:

RO	Read Only
ROC	Read Only and Read to clear
RW	Read Write
R/W	Read Write. The value read may not be the last value written.
RW1C	Read / Write ‘1’ to clear
RW1S	Read / Write ‘1’ to set
RWAC	Read / Write, Auto-clear: Writing 1 triggers an operation. Hardware clears register when operation completed. Writing 0 has no effect
WO	Write Only

B.2. Architecture Overview

The hardware unit executing the CQ functions is referred in this annex as Command Queuing Engine (CQE). The CQE is responsible for managing the interface between host software and *e*•MMC device, and the data transfers. The CQE may be implemented as a unit external to the *e*•MMC host controller in the host system, as illustrated in **Figure B.110**.

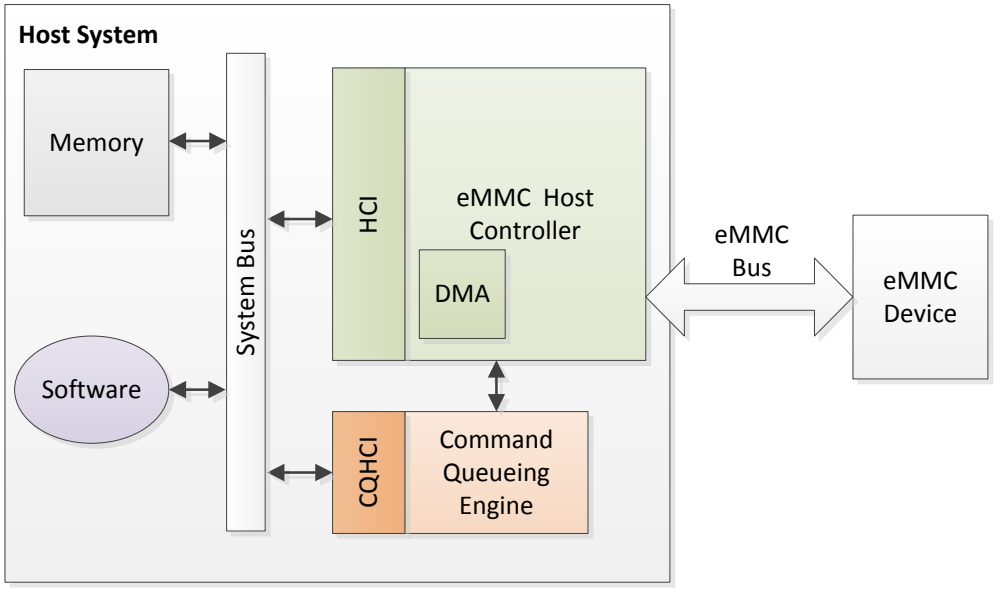


Figure B.110 — Proposed Host System Architecture, with CQE

The CQE receives tasks from software via a Task Descriptor List in system memory, and a doorbell register. The CQE issues CQ commands, QUEUED_TASK_PARAMS (CMD44) and QUEUED_TASK_ADDRESS (CMD45), to the *e*•MMC device and also stores the task's information. CQE also reads the device's queue status register (QSR), decides which task to execute and issues the EXECUTE commands: EXECUTE_READ_TASK (CMD46) or EXECUTE_WRITE_TASK (CMD47).

The functionality of CQE is described in this section.

B.2.1. Task Issuance: Task Descriptor List / Doorbell Register

To implement the queuing of tasks on the device, software issues the tasks asynchronously using a Task Descriptor List (TDL) which is located in the host memory space. The TDL is located in a memory location known to the CQE, and is comprised of up to 32 fixed-size slots. Each slot is comprised of one Task Descriptor and one Transfer Descriptor.

To issue a task, software selects an available TDL slot and constructs 2 descriptors in it:

- **A Task Descriptor**, which encodes all the information defining the Task. For example: Address, Block Count, and Priority. This information is later passed to the device.
  The Task Descriptor may, alternatively, encode any arbitrary command which is sent directly to the device (DCMD Descriptor, see B.2.5)
- **A Transfer Descriptor,** which points either to one continuous data buffer to/from which data is transferred (TRAN Descriptor) or to a scatter/gather list of any length (LINK Descriptor).

B.2.1 Task Issuance: Task Descriptor List / Doorbell Register (cont'd)

Figure B.111 illustrates the structures of the CQHCI. As an example, in Figure , slot #0 stores a Data Transfer Task with a TRAN descriptor, slot #1 stores a Data Transfer Task with a LINK descriptor, pointing to a scatter/gather list. Slot #31 stores a DCMD descriptor.

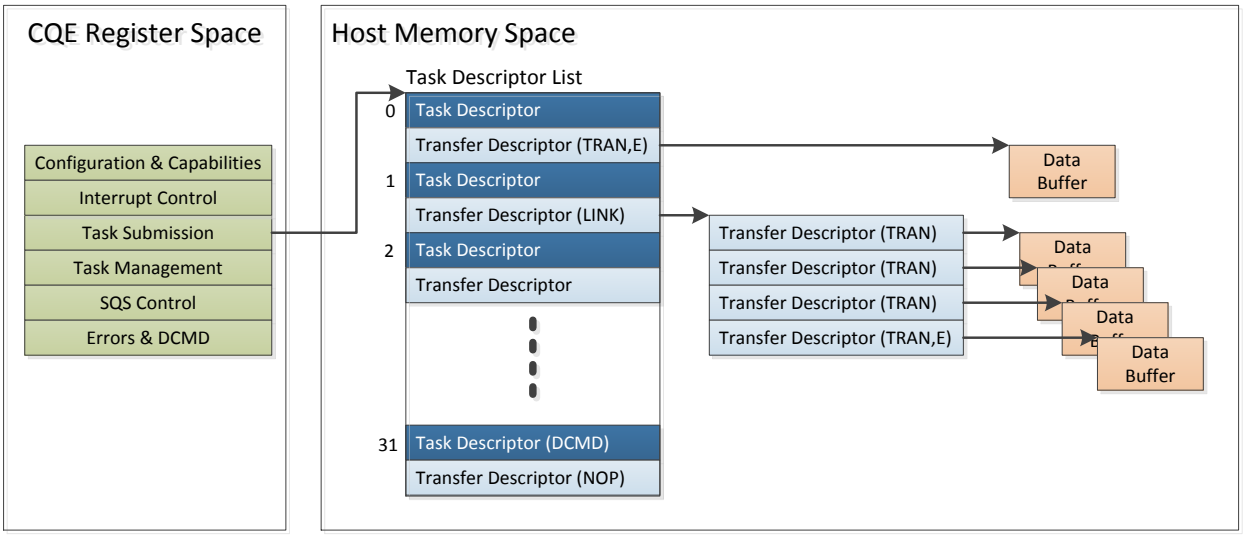


Figure B.111 — Command Queuing HCI General Architecture

B.2.2. Task Processing by Host Hardware

Once the 2 descriptors are written in slot *i*, software issues the task by writing '1' to bit *i* in CQTDBR register. This action signals to CQE to process the task. Software must not write CQTDBR to issue the task before valid descriptors are ready in the appropriate slot.

CQE reads the descriptors. Based on the information encoded in the Task Descriptor, CQE generates QUEUED_TASK_PARAMS (CMD44) and QUEUED_TASK_ADDRESS (CMD45), and sends them to the device.

B.2.3. Task Selection and Execution

CQE is also in charge of reading the device's Queue Status Register (QSR) to determine which tasks are ready for execution, selecting tasks for execution, and ordering their execution. These actions are described in this subsection.

Reading the QSR: When task(s) are queued in the device, CQE reads the QSR to determine which tasks are ready for execution. The QSR is read using SEND_ QUEUE_STATUS (CMD13) commands.

If a data transfer is ongoing, CQE sends the SQS towards the end of the data transfer, as configured in CQSSC1.CBC register field. If the bus is idle, CQE periodically sends SQS commands, as configured in CQSSC1.CIT register field.

Selecting a Task: When one or more data transfer tasks are ready for execution and the bus is idle (or a previous data transfer is about to end), CQE is expected to select a task for execution. The selected task must be marked as ready for execution by the device.

Ordering Task Execution: When the bus is idle, CQE sends EXECUTE_READ_TASK (CMD46) or EXECUTE_WRITE_TASK (CMD47) ordering the device to execute a task, identified by its Task ID.

B.2.3 Task Selection and Execution (cont'd)

CQE also feeds the task's Transfer Descriptor(s) to a DMA engine as a pointer to the data buffer in the host memory.

It is recommended that CQE selects tasks and sends the EXECUTE command as early as possible, to avoid unnecessary degradation in performance and power.

A task's execution is considered completed once the data transfer phase is completed.

B.2.4. Task Completion: Interrupts and Interrupt Coalescing

Upon the completion of a data transfer task, after the data has been transferred, an interrupt may be generated, if requested. If the INT bit in the Task Descriptor is set, a Task Complete (TCC) interrupt is generated. If the INT bit is not set, the transfer is counted towards coalescing. DCMD tasks are not counted towards coalescing.

The Interrupt Coalescing mechanism allows the host to moderate its interrupt load. The mechanism is also timer-protected to avoid high latency in low-load cases.

B.2.5. Direct Command (DCMD) Submission

When CQ is enabled, software still retains the ability to efficiently issue e•MMC commands by their index and argument. To use the DCMD feature, it must be enabled in CQCFG register. In order to issue a direct command, software writes a DCMD Task Descriptor to slot #31 of the TDL and rings the doorbell register (CQTDBR). The command's index and argument are encoded directly in the DCMD Task Descriptor (in the fields otherwise used for Address and Block Count).

The DCMD feature is targeted for the transmission of non-data commands which are allowed when CQ mode is enabled, such as CMD0, CMD12, or CMD13.

When processing the Task Descriptor, CQE constructs the command by its index and argument, sends it to the device, and stores the response in a dedicated register.

The DCMD Task Descriptor is identified by the CQE by its placement in slot #31 of the TDL. Therefore, it may only be issued via slot #31, and may only be sent one at a time. The Transfer Descriptor following the DCMD descriptor is ignored by CQE.

B.2.6. Queue-Barrier (QBR) Tasks

To enable host's control on the ordering between tasks, a task can be marked as a Queue-Barrier (QBR) Task by setting the QBR bit in the Task Descriptor. The following are guaranteed by the host controller:

- A QBR Task is only sent to the device after all the tasks issued before it have been executed (i.e., when the task queue in the device is empty).
- New tasks (i.e., tasks issued after the QBR Task) are sent to the device after the QBR task execution is completed.

The QBR feature, in conjunction with the DCMD feature, is targeted for transmission of commands which are allowed when CQ mode is enabled, when the device's task queue is empty. Such commands include, for example, erase commands (CMD35/CMD36/CMD38), cache flush/barrier (CMD6), and switching between General Purpose Partitions (CMD6).

B.2.6 Queue-Barrier (QBR) Tasks (cont'd)

Specifically, the following rules apply:

a) If the task marked with QBR is a Data Transfer task, CQE may send the related QUEUED_ TASK_PARAMS (CMD44) and QUEUED_ TASK_ADDRESS (CMD45) commands to the device only after receiving the response for the EXECUTE_READ_TASK(CMD46)/ EXECUTE_WRITE_TASK(CMD47) for the last data transfer task in the device's queue (data transfer may still be on-going).
b) If the task marked with QBR is a DCMD task, CQE may send the related command only after the all previously issued tasks have completed, including the associated data transfers.
c) Commands related to tasks which are queued after the QBR task, may only be sent after the QBR task execution is completed.

B.2.7. Halt Feature

In some cases, software may need to execute commands directly on the bus (i.e., not using DCMD). Task management commands (CMD48) or error recovery procedures are examples. In these cases, software should halt the CQE in order to gain control over the bus and issue its own commands.

This is done by writing '1' to the Halt bit in CQCTL register. The CQE does not stop immediately. Instead, it waits until the current operation (e.g., data transfer) is completed, and then halts and does not issue additional commands.

The exact behavior of CQE on a halt request is defined as follows:

a) If a data transfer operation has started: CQE will not issue additional commands on the bus, will wait until the data transfer completes as planned (including DMA transfer), and then notify software.
b) If task queuing is in progress [QUEUED_TASK_PARAMS (CMD44) and QUEUED_ TASK_ADDRESS (CMD45)]: CQE will complete the operation, sending both commands and processing the responses, and then notify software
   NOTE If an error is detected in the response to QUEUED_TASK_PARAMS (CMD44), CQE will halt immediately and not issue QUEUED_ TASK_ADDRESS (CMD45).
c) If command transmission is in progress (any other command): CQE will complete the command transmission, receive and process the response, and then notify software.

After CQE has halted, the device waits for the next command, so it is effectively halted as well. CQE notifies software that it is halted by setting the Halt bit on CQCTL register and, optionally by an interrupt (CQIS.HAC). It is guaranteed that neither the device nor the host hardware (i.e., CQE) will initiate any operation while CQE is in "halt" state.

When software wants the CQE to resume operation, it writes '0' to the Halt bit in CQCTL. When '0' is written, CQE continues its normal operation, based on the current state,

In some error conditions, the Halt bit may never actually set, because a response has not been received from the device. In these cases, software may assume the CQE is halted after waiting for a long enough time. See more on error recovery in 0.

B.2.8. Error Detection and Recovery

Error conditions are described in 6.8. Some errors, such as command timeout, or data CRC errors, are already detected by the *e*•MMC host controller. *e*•MMC host controller shall notify CQE when such errors occur.

Response mode errors, which are detected by software in legacy *e*•MMC, require detection in CQE.

When an error is detected CQE does the following:

a) Stops issuing commands
b) If the error is detected during a command/response transfer: Stores the Task ID and Command Index of said command
c) If the error is detected during a data transfer: Stores the Task ID and Command Index of command which started said data transfer
d) In the case of Response Mode error detection, CQE generates a RED interrupt

The specific handling of error conditions by CQE hardware is explained in Table B.228.

Table B.228 — Handling of Error Conditions in CQE

Error Condition	Description	Handling in CQE
Response Time-out	Response not received within N_{CR} clock cycles. See 6.8.1.	*e*•MMC controller detects error, generates error interrupt, and notifies CQE. CQE stores Task IDs and Command Indices in CQTERRI (see register description), and does not issue additional commands. Software Halts CQE and runs error recovery procedure. Halt operation may not complete due to error condition, in which case CQE disable/enable is required.
Time-out Conditions	See 6.8.2	
Data CRC Error	CRC error detected during data transfer (read or write)	
Error Response (R mode / X mode)	Error bit set in the device status field of a R1 response. See 6.13.	CQE detect error in response (according to CQRMEM configuration), stores Task IDs and Command Indices in CQTERRI, triggers RED interrupt, and does not issue additional command. Software Halts CQE and runs error recovery procedure.

Software is expected to halt CQE every time it runs an error recovery procedure, even though CQE is already stopped. When software has completed its procedure, it orders CQE to resume, by writing ‘0’ the Halt field of CQCTL. When instructed by software, CQE immediately resumes its operation, according to its updated state.

In some error cases, a response may not be received from the device, so CQE will not set the Halt bit. In such a case, software should wait a time-out which is long enough, based on the definitions in 6.8.2 and assume the device is stuck.

Also in these cases, software may write ‘0’ to the Halt field of CQCTL to order CQE to resume its operation.

B.3. Data Structures

This section describes the descriptor structures used in *e*•MMC Command Queuing. In order to queue a task, host software writes a pair of descriptors in system memory: a Task Descriptor, indicating the parameters of the task (see B.3.1) and a Transfer Descriptor, indicating the location of data buffers in the host memory space (see B.3.2).

Task descriptors are also used for queuing DCMD commands (See B.3.3).

All Descriptors are organized in the host memory in the Task Descriptor List, which is described in B.3.5.

B.3.1. Task Descriptor for Data Transfer Tasks

The Task Descriptor is created by software to specify to the CQE all the details of the queued task. A part of the contents of the Task Descriptor are passed to the device using CMD44 and CMD45 and are also saved in the CQE for later use.

The structure of the Task Descriptor is shown in Table B.229, and the field definition is given in Table B.231. The size of the Task Descriptor's can be configured using the Task Descriptor Size field in CQCFG Register. When this bit field bit is '0' the descriptor size is 64 bits as shown in Table B.229. When Task Descriptor Size field equals '1' the descriptor size is 128 bits. The structure of the upper 64 bits in that case is as shown in Table B.230.

Table B.229 — Task Descriptor Structure; Lower 64 bits (Data Transfer tasks)

Block Address	**Block Count**	**Task Parameters Field**							**Attribute**			
63:32	31:16	15	14	13	12	11	10:7	6	5:3	2	1	0
xxxx_xxxxh	xxxxh	Reliable Write	QBR	Priority	Data Direction	Tag request	Context	Forced programming	Act=101	Int	End=1	Valid=1

Table B.230 — Task Descriptor Structure; Upper 64 bits

Vendor Specific	**Reserved**
127 96	95 64
Implementation Specific	00000000h

B.3.1 Task Descriptor for Data Transfer Tasks (cont'd)

Table B.231 — Task Descriptor Fields

Field Name	Bit Loc.	Description	Encoded in CMD
Valid	0	1 = The descriptor is effective and should be processed by hardware 0 = The descriptor line should not be used	
End	1	Shall always be set to 1. Every Task Descriptor is standalone.	
Int	2	Indicates the interrupt generation policy required for this task. 1 = Hardware shall generate an interrupt upon the task's completion 0 = Hardware shall count task's completion for interrupt coalescing	
Act	5:3	Shall be set to b101, to indicate this is a Task Descriptor	
Forced Programming	6	If 1, FP is enabled. Data shall be forcefully programmed to nonvolatile storage instead of volatile cache while cache is turned ON	CMD44
Context ID	10:7	As described in 6.6.30	CMD44
Tag Request	11	As described in 6.6.31	CMD44
Data Direction	12	Indicates the direction of data transfer 1 = Device to Host (Read) / 0 = Host to Device (Write)	CMD44
Priority	13	1 = high / 0= simple	CMD44
Queue Barrier (QBR)	14	As described in B.3.4.	
Reliable Write	15	As described in 6.6.11	CMD44
Block Count	31:16	Number of blocks to be read/written	CMD44
Block Address	63:32	Data block address	CMD45

B.3.2. Transfer Descriptors

The 2nd descriptor describing a data transfer task is the Transfer Descriptor. The Transfer Descriptor may be either a TRAN Descriptor pointing directly to a single data buffer, or a LINK Descriptor, pointing to a list of descriptors. If a TRAN descriptor is used, its END attribute shall be set to indicate that the entire transfer is contained in a single data buffer.

Command queuing is to be used with fixed block size. Changing the block size shall be done after clearing and discarding all of the tasks in the queue.

The descriptor structures, for 32-bit and 64-bit addressing modes, are shown in Table B.232**Error! Reference source not found.** and Table B.233, respectively. The field definitions are listed in Table B.234.

B.3.2 Transfer Descriptors (cont'd)

Table B.232 — Transfer Descriptor Structure (32-bit addressing)

Address	Length	Reserved	Attribute			
63:32	31:16	15 6	5:3	2	1	0
xxxx_xxxxh	xxxxh	0000000000	Act	Int	End	Valid

Table B.233 — Transfer Descriptor Structure (64-bit addressing)

Reserved	Address	Length	Reserved	Attribute			
127:96	95:32	31:16	15 6	5:3	2	1	0
0	xxxx_xxxxh	xxxxh	0000000000	Act	Int	End	Valid

Table B.234 — Transfer Descriptor Fields

Field Name	Bit Loc.	Description
Valid	0	1 = The descriptor is effective and should be processed by hardware 0 = The descriptor line should not be used
End	1	1 = The descriptor is the last descriptor in a descriptor list 0 = Additional descriptors follow this descriptor In the case of a TRAN descriptor, the value of this bit shall be 1
Int	2	The value of this bit shall be set to 0 in command queuing.
Act	5:3	100: TRAN – Address field of descriptor points to a data buffer 110: LINK – Address field of descriptor points to a another descriptor 000: NOP – no operation. Others: reserved
Reserved	15:6	Reserved
Length	31:16	Length of data buffer in bytes. A value of 0000 means 64 KB.
Address (32-bit)	63:32	Data buffer address in host memory, in 32-bit addressing mode. Address shall be set on 32-bit boundary (Lower 2 bits set to 0)
Address (64-bit)	95:32	Data buffer address in host memory, in 64-bit addressing mode. Address shall be set on 64-bit boundary (Lower 3 bits set to 0)
Reserved	127:96	Reserved (64-bit addressing mode only)

B.3.3. Task Descriptor for Direct-Command (DCMD) Tasks

Host driver may issue non-data commands to the *e*•MMC device by writing a DCMD Task Descriptors to slot #31 of the TDL (when DCMD is enabled in CQCFG register). In case of DCMD Task Descriptor, the meanings of some fields are changed. The descriptor structure is shown in Table B.235, and the field definition is listed in Table B.236.

Similarly to Data Transfer Task Descriptors, the size of the DCMD Task Descriptor is set by the Task Descriptor Size field in CQCFG Register. When this bit field bit is '0' the descriptor size is 64 bits, as shown in Table B.235. When Task Descriptor Size field equals '1' the descriptor size is 128 bits. The structure of the upper 64 bits in that case is as shown in Table B.230.

B.3.3 Task Descriptor for Direct-Command (DCMD) Tasks (cont'd)

When processing a DCMD Task Descriptor, the hardware sends a command to the device, with the index and argument encoded in bits 21:16 and 63:32 of the Task Descriptor, respectively.

Table B.235 — Task Descriptor Structure: Lower 64 bits (for DCMD tasks)

Command Argument	Command Parameters				Task Parameters Field			Attribute			
63:32	31 25	24 23	22	21 16	15	14	13 6	5 3	2	1	0
xxxx_xxxxh	Reserved (0000000)	Response Type	CMD timing	CMD index	Rsvd (0)	QBR	Reserved (0000000)	Act= 101	Int	End =1	Valid =1

The Command Timing bit lets software control whether the command is issued during data transfer (e.g., CMD13) or only during idle time. Using the R1b bit, software informs hardware whether it should expect a busy time following the command's response.

The Transfer Descriptor following a DCMD Task Descriptor is expected to be with ACT=NOP and END bit set.

Task descriptor for direct-command task shall only be queued in task index 31 of the TDL.

Table B.236 — Task Descriptor Fields (for DCMD tasks)

Field Name	Bit Loc.	Description
Valid	0	1 = The descriptor is effective and should be processed by hardware 0 = The descriptor line should not be used
End	1	Shall always be set to 1. Every Task Descriptor is standalone.
Int	2	Indicates the interrupt generation policy required for this task. 1 = Hardware shall generate an interrupt upon the task's completion 0 = Hardware shall not generate an interrupt upon the task's completion. Interrupt coalescing is not used with DCMD.
Act	5:3	Shall be set to b101, to indicate this is a Task Descriptor
Reserved	13:6	
Queue Barrier (QBR)	14	As described in B.2.6
Reserved	15	
CMD Index	21:16	The index of the command to be sent to device
CMD Timing	22	1 = Command may be sent to device during data activity or busy time 0 = Command may **not** be sent to device during data activity or busy time NOTE Software shall be 0 if response type is b11 (R1b)
Response Type	24:23	This field indicates to the host controller the response expected to be received from the device. Value / Description: b00 — No Response Expected b01 — Reserved b10 — R1, R4, R5 b11 — R1b NOTE R2 and R3 are not supported in DCMD
Reserved	31:25	
CMD Argument	63:32	The argument of the command to be sent to the device

B.3.4. Task Descriptor for Queue-Barrier Task (QBR)

A task whose Task Descriptor's QBR bit is set, will be held by CQE and will only be sent to the device after all the tasks queued before it have completed execution. Additionally, CQE holds all tasks which are issued after the QBR Task, and only sends them to the device after the QBR Task has completed.

QBR bit may be set for both Data Transfer Task Descriptors and DCMD Task Descriptors.

B.3.5. Task List

The Task Descriptors for all tasks are organized as a list (Task Descriptor List, TDL) in a location in the host memory indicated in Task Descriptor List Base Address (TDLBA) registers. Every slot in the TDL is comprised of a Task Descriptor, followed by a single Transfer Descriptor.

The size of each entry in the TDL is therefore dependent upon the configured size of the Task Descriptors (64 or 128 bits), and the Transfer Descriptors (64 or 128 bits), so it varies between 16 Bytes and 32 Bytes.

The location of each slot in the TDL is therefore calculates as:

$$ADDR_i = TDLBA + i \times (SIZE_{TASKDESC} + SIZE_{TRANSFERDESC})$$

Changing the value of TDLBA is not allowed when command queue mode is enabled. To do that host software shall exit command queue mode, reset CQE and only then reconfigure TDLBA, before issuing any new tasks.

B.4. CQE Registers

The registers listed in this section comprise the register interface of the CQE module.

B.4.1. Register Map

CQE's register map appears in Table B.237. The registers in the table appear with their respective offsets from CQBASE, the base address of CQE.

Table B.237 — CQE Register Map

	Offset from CQBASE	Symbol	Register Name
CFG and CAP	00h	CQVER	Command Queuing Version
	04h	CQCAP	Command Queuing Capabilities (reserved)
	08h	CQCFG	Command Queuing Configuration
	0Ch	CQCTL	Command Queuing Control
Interrupt Control	10h	CQIS	Command Queuing Interrupt Status
	14h	CQISTE	Command Queuing Interrupt Status Enable
	18h	CQISGE	Command Queuing Interrupt Signal Enable
	1Ch	CQIC	Command Queuing Interrupt Coalescing
Task Submission	20h	CQTDLBA	Command Queuing Task Descriptor List Base Address
	24h	CQTDLBAU	Command Queuing Task Descriptor List Base Address Upper 32 bits
	28h	CQTDBR	Command Queuing Task Doorbell
	2Ch	CQTCN	Command Queuing Task Completion Notification
Task Management	30h	CQDQS	Command Queuing Device Queue Status
	34h	CQDPT	Command Queuing Device Pending Tasks
	38h	CQTCLR	Command Queuing Task Clear
SQS and DCMD	40h	CQSSC1	Command Queuing Send Status Configuration 1
	44h	CQSSC2	Command Queuing Send Status Configuration 2
	48h	CQCRDCT	Command Queuing Command Response for Direct-Command Task
Error handling	50h	CQRMEM	Command Queuing Response Mode Error Mask
	54h	CQTERRI	Command Queuing Task Error Information
	58h	CQCRI	Command Queuing Command Response Index
	5Ch	CQCRA	Command Queuing Command Response Argument

B.4.2. CQBASE+00h: CQVER – Command Queuing Version

This register provides information about the version of the *e*•MMC CQ standard which is implemented by the CQE, in BCD format. The current version is 5.1

Bit	Type	Reset	Description
31:12	RO	0	Reserved
11:08	RO	5	*e*•MMC Major Version Number (digit left of decimal point), in BCD format
07:04	RO	1	*e* •MMC Minor Version Number (digit right of decimal point), in BCD format
03:00	RO	0	*e* •MMC Version Suffix (2nd digit right of decimal point), in BCD format

B.4.3.CQBASE+04h: CQCAP – Command Queuing Capabilities

This register indicates hardware capabilities.

Bit	Type	Reset	Description
31:16	RO	0	Reserved
15:12	RO	Imp. Spec.	**Internal Timer Clock Frequency Multiplier (ITCFMUL)** ITCFMUL and ITCFVAL indicate the frequency of the clock used for interrupt coalescing timer and for determining the SQS polling period. See ITCFVAL definition for details. **Field Value Description:** 0h = 0.001 MHz 1h = 0.01 MHz 2h = 0.1 MHz 3h = 1 MHz 4h = 10 MHz Other values are reserved
11:10	RO	0	Reserved
09:00	RO	Imp. Spec.	**Internal Timer Clock Frequency Value (ITCFVAL)** ITCFMUL and ITCFVAL indicate the frequency of the clock used for interrupt coalescing timer and for determining the polling period when using periodic SEND_QUEUE_ STATUS (CMD13) polling. The clock frequency is calculated as ITCFVAL* ITCFMUL. For example, to encode 19.2 MHz, ITCFVAL shall be C0h (= 192 decimal) and ITCFMUL shall be 2h (0.1 MHz): 192 * 0.1 MHz = 19.2 MHz.

B.4.4. CQBASE+08h: CQCFG – Command Queuing Configuration

This register controls CQE behavior affecting the general operation of command queuing module or operation of multiple tasks in the same time.

Bit	Type	Reset	Description
31:13	RO	0000	Reserved
12	RW	0	**Direct Command (DCMD) Enable** This bit indicates to the hardware whether the Task Descriptor in slot #31 of the TDL is a Data Transfer Task Descriptor, or a Direct Command Task Descriptor. CQE uses this bit when a task is issued in slot #31, to determine how to decode the Task Descriptor. **Bit Value Description** 1 = Task descriptor in slot #31 is a DCMD Task Descriptor 0 = Task descriptor in slot #31 is a Data Transfer Task Descriptor
11:09	RO	0000	Reserved
08	RW	0	**Task Descriptor Size** This bit indicates whether the task descriptor size is 128 bits or 64 bits as detailed in Data Structures section. This bit can only be configured when Command Queuing Enable bit is '0' (command queuing is disabled) **Bit Value Description** 1 = Task descriptor size is 128 bits 0 = Task descriptor size is 64 bits
07:01	RO	0	**Reserved**
00	RW	0	**Command Queuing Enable** Software shall write '1' this bit when in order to enable command queuing mode (i.e., enable CQE). When this bit is 0, CQE is disabled and software controls the *e*•MMC bus using the legacy *e*•MMC host controller. Before software writes '1' to this bit, software shall verify that the *e*•MMC host controller is in idle state and there are no commands or data transfers ongoing. When software wants to exit command queuing mode, it shall clear all previous tasks if such exist before setting this bit to 0.

B.4.5. CQBASE+0Ch: CQCTL – Command Queuing Control

This register controls CQE behavior affecting the general operation of command queuing module or operation of multiple tasks in the same time.

Bit	Type	Reset	Description
31:09	RO	0000	Reserved
08	RWAC	0	**Clear All Tasks** Software shall write '1' this bit when it wants to clear all the tasks sent to the device. This bit can only be written when CQE is in halt state (i.e., Halt bit is 1). When software writes 1, the value of the register is updated to '1', and CQE shall reset CQTDBR register and all other context information for all unfinished tasks. Then CQE will clear this bit. Software should poll on this bit until it is set to back 0 and may then resume normal operation, by clearing the Halt bit. CQE does not communicate to the device that the tasks were cleared. It is software's responsibility to order the device to discard the tasks in its queue using CMDQ_TASK_MGMT command. Writing '0' to this register shall have no effect.
07:01	RO	0	**Reserved**
00	R/W	0	**Halt** Host software shall write '1' to the bit when it wants to acquire software control over the *e*•MMC bus and disable CQE from issuing commands on the bus. For example, issuing a Discard Task command (CMDQ_TASK_MGMT) When software writes '1', CQE shall complete the ongoing task if such a task is in progress. Once the task is completed and CQE is in idle state, CQE shall not issue new commands and shall indicate so to software by setting this bit to 1. Software may poll on this bit until it is set to 1, and may only then send commands on the *e*•MMC bus. In order to exit halt state (i.e., resume CQE activity), software shall clear this bit (write '0'). Writing '0' when the value is already '0' shall have no effect.

B.4.6. CQBASE+10h: CQIS – Command Queuing Interrupt Status

This register indicates pending interrupts that require service. Each bit in this registers is asserted in response a specific event, only if the respective bit is set in CQISTE register.

Bit	Type	Reset	Description
31:04	Rsvd	0	Reserved
03	RW1C	0	**Task Cleared (TCL)** This status bit is asserted (if CQISTE.TCL=1) when a task clear operation is completed by CQE. The completed task clear operation is either an individual task clear (CQTCLR) or clearing of all tasks (CQCTL).
02	RW1C	0	**Response Error Detected Interrupt (RED)** This status bit is asserted (if CQISTE.RED=1) when a response is received with an error bit set in the device status field. The contents of the device status field are listed in 6.13. Software uses CQRMEM register to configure which device status bit fields may trigger an interrupt, and which are masked.
01	RW1C	0	**Task Complete Interrupt (TCC)** This status bit is asserted (if CQISTE.TCC=1) when at least one of the following two conditions are met: 1) A task is completed and the INT bit is set in its Task Descriptor 2) Interrupt caused by Interrupt Coalescing logic (see 0)
00	RW1C	0	**Halt Complete Interrupt (HAC)** This status bit is asserted (if CQISTE.HAC=1) when halt bit in CQCTL register transitions from 0 to 1 indicating that host controller has completed its current ongoing task and has entered halt state.

B.4.7. CQBASE+14h: CQISTE – Command Queuing Interrupt Status Enable

This register enables and disables the reporting of the corresponding interrupt to host software in CQIS register. When a bit is set (‘1’) and the corresponding interrupt condition is active, then the bit is CQIS is asserted. Interrupt sources that are disabled (‘0’) are not indicated in the CQIS register. This register is bit-index matched to CQIS register.

Bit	Type	Reset	Description
31:04	Rsvd	0	Reserved
03	RW	0	**Task Cleared Status Enable (TCL)** 1 = CQIS.TCL will be set when its interrupt condition is active 0 = CQIS.TCL is disabled
02	RW	0	**Response Error Detected Status Enable (RED)** 1 = CQIS.RED will be set when its interrupt condition is active 0 = CQIS.RED is disabled
01	RW	0	**Task Complete Status Enable (TCC)** 1 = CQIS.TCC will be set when its interrupt condition is active 0 = CQIS.TCC is disabled
00	RW	0	**Halt Complete Status Enable (HAC)** 1 = CQIS.HAC will be set when its interrupt condition is active 0 = CQIS.HAC is disabled

B.4.8. CQBASE+18h: CQISGE – Command Queuing Interrupt Signal Enable

This register enables and disables the generation of interrupts to host software. When a bit is set (‘1’) and the corresponding bit in CQIS is set, then an interrupt is generated. Interrupt sources that are disabled (‘0’) are still indicated in the CQIS register. This register is bit-index matched to CQIS register.

Bit	Type	Reset	Description
31:04	Rsvd	0	Reserved
03	RW	0	**Task Cleared Signal Enable (TCL)** When set and CQIS.TCL is asserted, the CQE shall generate an interrupt
02	RW	0	**Response Error Detected Signal Enable (TCC)** When set and CQIS.RED is asserted, the CQE shall generate an interrupt
01	RW	0	**Task Complete Signal Enable (TCC)** When set and CQIS.TCC is asserted, the CQE shall generate an interrupt
00	RW	0	**Halt Complete Signal Enable (HAC)** When set and CQIS.HAC is asserted, the CQE shall generate an interrupt

B.4.9. CQBASE+1Ch: CQIC – Interrupt Coalescing

This register controls the interrupt coalescing feature.

Bit	Type	Reset	Description
31	RW	0	**Interrupt Coalescing Enable/Disable:** When set to '0' by software, command responses are neither counted nor timed. Interrupts are still triggered by completion of tasks with INT=1 in the Task Descriptor. When set to '1', the interrupt coalescing mechanism is enabled and coalesced interrupts are generated
30:21	RO	0	**Reserved**
20	RO	0	**Interrupt Coalescing Status Bit (ICSB):** This bit indicates to software whether any tasks (with INT=0) have completed and counted towards interrupt coalescing (i.e., ICSB is set if and only if IC counter > 0). **Bit Value Description** 1 = At least one task completion has been counted (IC counter >0) 0 = No task completions have occurred since last counter reset (IC counter =0)
19:17	RO	0	**Reserved**
16	WO	0	**Counter and Timer Reset(ICCTR):** When host driver writes '1', the interrupt coalescing timer and counter are reset
15	WO	0	**Interrupt Coalescing Counter Threshold Write Enable (ICCTHWEN):** When software writes '1', the value ICCTH is updated with the contents written at the same cycle. When software writes '0', the value in ICCTH is not updated. NOTE Write operations to ICCTH are only allowed when the task queue is empty.
14:13	RO	0	**Reserved**
12:08	RW	0	**Interrupt Coalescing Counter Threshold (ICCTH):** Software uses this field to configure the number of task completions (only tasks with INT=0 in the Task Descriptor) which are required in order to generate an interrupt. Counter Operation: As data transfer tasks with INT=0 complete, they are counted by CQE. The counter is reset by software during the interrupt service routine. The counter stops counting when it reaches the value configured in ICCTH. The maximum allowed value is 31 NOTE When ICCTH is 0, task completions are not counted, and counting-based interrupts are not generated. In order to write to this field, the ICCTHWEN bit must be set at the same write operation.
07	WO	0	**Interrupt Coalescing Timeout Value Write Enable (ICTOVALWEN):** When software writes '1', the value ICTOVAL is updated with the contents written at the same cycle. When software writes '0', the value in ICTOVAL is not updated. NOTE Write operations to ICTOVAL are only allowed when the task queue is empty.
06:00	RW	0	**Interrupt Coalescing Timeout Value (ICTOVAL):** Software uses this field to configure the maximum time allowed between the completion of a task on the bus and the generation of an interrupt. Timer Operation: The timer is reset by software during the interrupt service routine. It starts running when a data transfer task with INT=0 is completed, after the timer was reset. When the timer reaches the value configured in ICTOVAL field it generates an interrupt and stops. The timer's unit is equal to 1024 clock periods of the clock whose frequency is specified in the Internal Timer Clock Frequency field CQCAP register. The minimum value is 01h (1024 clock periods) and the maximum value is 7Fh (127*1024 clock periods). For example, a CQCAP field value of 0 indicates a 19.2 MHz clock frequency (period = 52.08 ns). If the setting in ICTOVAL is 10h, the calculated polling period is 16*1024*52.08 ns= 853.33 us NOTE When ICTOVAL is 0, the timer is not running, and timer-based interrupts are not generated. In order to write to this field, the ICTOVALWEN bit must be set at the same write operation.

B.4.10. CQBASE+20h: CQTDLBA – Command Queuing Task Descriptor List Base Address

This register is used for configuring the lower 32 bits of the byte address of the head of the Task Descriptor List in the host memory.

Bit	Type	Reset	Description
31:00	RW	0	**Task Descriptor List Base Address (TDLBA)** This register stores the LSB bits (bits 31:0) of the byte address of the head of the Task Descriptor List in system memory. The size of the task descriptor list is 32 * (Task Descriptor size + Transfer Descriptor size) as configured by Host driver. This address shall be set on 1 KByte boundary: The lower 10 bits of this register shall be set to 0 by software and shall be ignored by CQE.

B.4.11. CQBASE+24h: CQTDLBAU – Command Queuing Task Descriptor List Base Address Upper 32 Bits

This register is used for configuring the upper 32 bits of the byte address of the head of the Task Descriptor List in the host memory.

Bit	Type	Reset	Description
31:00	RW	0	**Task Descriptor List Base Address (TDLBA)** This register stores the MSB bits (bits 63:32) of the byte address of the head of the Task Descriptor List in system memory. The size of the task descriptor list is 32 * (Task Descriptor size + Transfer Descriptor size) as configured by Host driver. This register is reserved when using 32-bit addressing mode.

B.4.12. CQBASE+28h: CQTDBR – Command Queuing Task Doorbell

Using this register, software triggers CQE to process a new task.

Bit	Type	Reset	Register Field Explanation
31:00	RW1S	0	**Command Queuing Task Doorbell** Software shall configure TDLBA and TDLBAU, and enable CQE in CQCFG before using this register. Writing 1 to bit *n* of this register triggers CQE to start processing the task encoded in slot *n* of the TDL. CQE always processes tasks in-order according to the order submitted to the list by CQTDBR write transactions. CQE processes Data Transfer tasks by reading the Task Descriptor and sending QUEUED_TASK_PARAMS (CMD44) and QUEUED_TASK_ADDRESS (CMD45) commands to the device. CQE processes DCMD tasks (in slot #31, when enabled) by reading the Task Descriptor, and generating the command encoded by its index and argument. The corresponding bit is cleared to '0' by CQE in one of the following events: a) When a task execution is completed (with success or error) b) The task is cleared using CQTCLR register c) All tasks are cleared using CQCTL register d) CQE is disabled using CQCFG register Software may initiate multiple tasks at the same time (batch submission) by writing 1 to multiple bits of this register in the same transaction. In the case of batch submission: CQE shall process the tasks in order of the task index, starting with the lowest index. If one or more tasks in the batch are marked with QBR, the ordering of execution will be based on said processing order. Writing 0 by software shall have no impact on the hardware, and will not change the value of the register bit.

B.4.13. CQBASE+2Ch: CQTCN – Task Completion Notification

This register is used by CQE to notify software about completed tasks.

Bit	Type	Reset	Description
31:00	RW1C	0	**Task Complete Notification** CQE shall set bit *n* of this register (at the same time it clears bit *n* of CQTDBR) when a task execution is completed (with success or error). When receiving interrupt for task completion, software may read this register to know which tasks have finished. After reading this register, software may clear the relevant bit fields by writing 1 to the corresponding bits.

B.4.14. CQBASE+30h: CQDQS – Device Queue Status

This register stores the most recent value of the device's queue status.

Bit	Type	Reset	Description
31:00	RO	0	**Device Queue Status** Every time the Host controller receives a queue status register (QSR) from the device, it updates this register with the response of status command, i.e., the device's queue status.

B.4.15. CQBASE+34h: CQDPT – Device Pending Tasks

This register indicates to software which tasks are queued in the device, awaiting execution.

Bit	Type	Reset	Description
31:00	RO	0	**Device Pending Tasks** Bit *n* of this register is set if and only if QUEUED_TASK_PARAMS (CMD44) and QUEUED_TASK_ADDRESS (CMD45) were sent for this specific task and if this task hasn't been executed yet. CQE shall set this bit after receiving a successful response for CMD45. CQE shall clear this bit after the task has completed execution. Software needs to read this register in the task-discard procedure, when the controller is halted, to determine if the task is queued in the device. If the task is queued, the driver sends a CMDQ_TASK_MGMT (CMD48) to the device ordering it to discard the task. Then software clears the task in the CQE. Only then the software orders CQE to resume its operation using CQCTL register.

B.4.16. CQBASE+38h: CQTCLR – Task Clear

This register is used for removing an outstanding task in the CQE.

The register should be used only when CQE is in Halt state.

Bit	Type	Reset	Description
31:00	RW	0	**Command Queuing Task Clear** Writing 1 to bit *n* of this register orders CQE to clear a task which software has previously issued. This bit can only be written when CQE is in Halt state as indicated in CQCFG register Halt bit. When software writes '1' to a bit in this register, CQE updates the value to '1', and starts clearing the data structures related to the task. CQE clears the bit fields (sets a value of 0) in CQTCLR and in CQTDBR once clear operation is complete. Software should poll on the CQTCLR until it is cleared to verify clear operation was complete. Writing to this register only clears the task in the CQE and does not have impact on the device. In order to discard the task in the device, host software shall send CMDQ_TASK _MGMT while CQE is still in Halt state. Host driver is not allowed to use this register to clear multiple tasks at the same time. Clearing multiple tasks can be done using CQCTL register. Writing 0 to a register bit shall have no impact.

B.4.17. CQBASE+40h: CQSSC1 – Send Status Configuration 1

The register controls the when SEND_QUEUE_STATUS commands are sent.

Bit	Type	Reset	Description
31:20	RO	0	Reserved
19:16	RW	1	**Send Status Command Block Counter (CBC)** This field indicates to CQE when to send SEND_QUEUE_STATUS (CMD13) command to inquire the status of the device's task queue. A value of *n* means CQE shall send status command on the CMD line, during the transfer of data block **BLOCK_CNT-*n***, on the data lines, where BLOCK_CNT is the number of blocks in the current transaction. A value of 0 means that SEND_QUEUE_STATUS (CMD13) command shall not be sent during the transaction. Instead it will be sent only when the data lines are idle. A value of 1 means that STATUS command is to be sent during the last block of the transaction.
15:00	RW	h1000	**Send Status Command Idle Timer (CIT)** This field indicates to CQE the polling period to use when using periodic SEND_QUEUE_STATUS (CMD13) polling. Periodic polling is used when tasks are pending in the device, but no data transfer is in progress. When a SEND_QUEUE_STATUS response indicating that no task is ready for execution, CQE counts the configured time until it issues the next SEND_QUEUE_STATUS. Timer units are clock periods of the clock whose frequency is specified in the Internal Timer Clock Frequency field CQCAP register. The minimum value is 0001h (1 clock period) and the maximum value is FFFFh (65535 clock periods). Default interval is: 4096 clock periods. For example, a CQCAP field value of 0 indicates a 19.2 MHz clock frequency (period = 52.08 ns). If the setting in CQSSC1.CIT is 1000h, the calculated polling period is 4096*52.08 ns= 213.33 us

B.4.18. CQBASE+44h: CQSSC2 – Send Status Configuration 2

This register is used for configuring RCA field in SEND_QUEUE_STATUS command argument.

Bit	Type	Reset	Description
31:16	RO	0	Reserved
15:00	RW	0	**Send Queue Status RCA** This field provides CQE with the contents of the 16-bit RCA field in SEND_QUEUE_ STATUS (CMD13) command argument. CQE shall copy this field to bits 31:16 of the argument when transmitting SEND_ QUEUE_STATUS (CMD13) command.

B.4.19. CQBASE+48h: CQCRDCT – Command Response for Direct-Command Task

This register is used for passing the response of a DCMD task to software.

Bit	Type	Reset	Description
31:00	RO	0	**Direct Command Last Response** This register contains the response of the command generated by the last direct-command (DCMD) task which was sent. CQE shall update this register when it receives the response for a DCMD task. This register is considered valid only after bit 31 of CQTDBR register is cleared by CQE.

B.4.20. CQBASE+50h: CQRMEM – Response Mode Error Mask

This register controls the generation of Response Error Detection (RED) interrupt.

Bit	Type	Reset	Description
31:00	RW	FDF9A080h	**Response Mode Error Mask** This bit is used as in interrupt mask on the device status filed which is received in R1/R1b responses. **Bit Value Description (for any bit i):** 1 = When a R1/R1b response is received, with bit *i* in the device status set, a RED interrupt is generated 0 = When a R1/R1b response is received, bit *i* in the device status is ignored The reset value of this register is set to trigger an interrupt on all “Error” type bits in the device status, as defined in 6.13. NOTE Responses to CMD13 (SQS) encode the QSR, so they are ignored by this logic.

B.4.21. CQBASE+54h: CQTERRI - Task Error Information

This register is updated by CQE when an error occurs on data or command related to a task activity.

When such error is detected by CQE or indicated by the *e*•MMC controller CQE stores in CQTERRI the task IDs and the command indices of the commands which were executed on the command line and data lines when the error occurred.

Software is expected to use this information in the error recovery procedure. See B.2.8 for more details.

Bit	Type	Reset	Description
31	RO	0	**Data Transfer Error Fields Valid** This bit is updated when an error is detected by CQE, or indicated by *e*•MMC controller. If a data transfer is in progress when the error is detected/indicated, the bit is set to 1. If a no data transfer is in progress when the error is detected/indicated, the bit is cleared to 0.
30:29	RO	0	**Reserved**
28:24	RO	0	**Data Transfer Error Task ID** This field indicates the ID of the task which was executed on the data lines when an error occurred. The field is updated if a data transfer is in progress when an error is detected by CQE, or indicated by *e*•MMC controller.
23:22	RO	0	**Reserved**
21:16	RO	0	**Data Transfer Error Command Index** This field indicates the index of the command which was executed on the data lines when an error occurred. The index shall be set to EXECUTE_READ_TASK (CMD46) or EXECUTE_WRITE_TASK (CMD47) according to the data direction. The field is updated if a data transfer is in progress when an error is detected by CQE, or indicated by *e*•MMC controller.
15	RO	0	**Response Mode Error Fields Valid** This bit is updated when an error is detected by CQE, or indicated by *e*•MMC controller. If a command transaction is in progress when the error is detected/indicated, the bit is set to 1. If a no command transaction is in progress when the error is detected/indicated, the bit is cleared to 0.
14:13	RO	0	**Reserved**
12:08	RO	0	**Response Mode Error Task ID** This field indicates the ID of the task which was executed on the command line when an error occurred. The field is updated if a command transaction is in progress when an error is detected by CQE, or indicated by *e*•MMC controller.
07:06	RO	0	Reserved
05:00	RO	0	**Response Mode Error Command Index** This field indicates the index of the command which was executed on the command line when an error occurred. The field is updated if a command transaction is in progress when an error is detected by CQE, or indicated by *e*•MMC controller.

B.4.22. CQBASE+58h: CQCRI – Command Response Index

This register stores the index of the last received command response.

Bit	Type	Reset	Description
31:06	RO	0	Reserved
05:00	RO	0	**Last Command Response index** This field stores the index of the last received command response. CQE shall update the value every time a command response is received.

B.4.23. CQBASE+5Ch: CQCRA – Command Response Argument

This register stores the argument of the last received command response.

Bit	Type	Reset	Description
31:00	RO	0	**Last Command Response Argument** This field stores the argument of the last received command. CQE shall update the value every time a command response is received.

B.5. Command Queuing Interrupt in e•MMC Host Controller

In order to make CQE compatible with existing *e*•MMC host controllers, an interrupt bit is added to the general interrupt register structure of the *e*•MMC host controller.

Specifically, the added fields are the Command Queuing Interrupt support in bit 14 of the Interrupt, Interrupt Status Enable, and Interrupt Signal Enable registers.

B.5.1. Normal Interrupt Status Register (Offset 030h)

Bit	Type	Reset	Description
14	RW1C	0	**Command Queuing Interrupt** This interrupt is asserted when at least one of the bits in CQIS register is set. This interrupt is cleared only by clearing the source interrupt in CQIS register.

B.5.2. Normal Interrupt Status Enable Register (Offset 034h)

Bit	Type	Reset	Description
14	RW	0	**Command Queuing Status Enable**

B.5.3. Normal Interrupt Signal Enable Register (Offset 038h)

Bit	Type	Reset	Description
14	RW	0	**Command Queuing Signal Enable**

B.6. Theory of Operation (Informative)

This section provides information to software developers regarding the behavior expected from the host software such that it optimally uses CQE for the implementation of e•MMC Command Queuing.

B.6.1. Command Queuing Initialization Sequence

In order to start using Command Queuing and initialize CQE hardware, the following steps should be executed by host software:

1. Initialize and enable Command Queuing in the device (beyond the scope of this section).
2. Configure Task Descriptor size in CQCFG register.
3. Configure CQTDLBA and CQTDLBAU to point to the memory location allocated to the TDL in host memory
4. Configure CQSSC1 to control when SEND_QUEUE_STATUS commands are sent to the device by CQE
5. Configure CQIC register to control the interrupt coalescing feature: enable/disable, set interrupt count and timer protection
6. Configure CQRMEM to control which errors may trigger a RED interrupt (if different from reset values)
7. Write '1' to CQCFG to enable CQE activity

B.6.2. Task Issuance Sequence

When host software needs to issue a data Transfer Task to the CQE and e•MMC device, it utilizes Task Descriptor List.

The following is the steps for host software to build a Task and issue it:

1. Find an empty transfer request slot by reading the CQTDBR. An empty transfer request slot has its respective bit cleared to '0' in the CQTDBR.
2. Build a Task Descriptor at the 1st entry of the empty slot. Task Descriptor field values:
   a. Valid =1, to indicate the descriptor is effective.
   b. End =1, as required for Task Descriptors.
   c. Int=1, if an interrupt on completion is required. Otherwise, Int=0.
   d. Act = b101, to indicate a Task Descriptor
   e. Data Direction =1 for read commands, and 0 for write commands.
   f. Priority =1 for high priority, and 0 for simple requests.
   g. QBR =1 if Queue Barrier functionality is required. 0 otherwise.
   h. Forced Programming, Context ID, Tag Request, Reliable Write, Block Count, and Block Address programmed according to application requirements.
3. Build a Transfer Descriptor at the 2nd entry of the empty slot. Transfer Descriptor field values:
   a. Valid =1, to indicate the descriptor is effective.
   b. End =1, if a TRAN descriptor is used, to indicate the task only has one data buffer. End =0 if LINK descriptor is used.
   c. Int=0. Ignore in Command queuing.
   d. Act = b100, for TRAN descriptors, pointing directly to the task's single data buffer. Act = b110, for LINK descriptors, point to a scatter/gather list (indirect)
   e. Address and Length programmed according to the data buffer supplied by the application.
4. If more than one transfer is requested, repeat step 1-3 for all needed transfers
5. Set CQTDBR to ring the doorbell register to indicate to the CQE that one or more transfer requests are ready to be sent to the attached device. Host software shall only write a '1' to the bit position that corresponds to new tasks; all other bit positions within CQTDBR should be written with a '0', which indicates no change to their current values.

B.6.2 Task Issuance Sequence (cont'd)

Following the steps below, the host hardware starts processing the task and sends it to the device. The sequence for queuing a task, between the host software, CQE and the *e*•MMC device, is given in Figure .

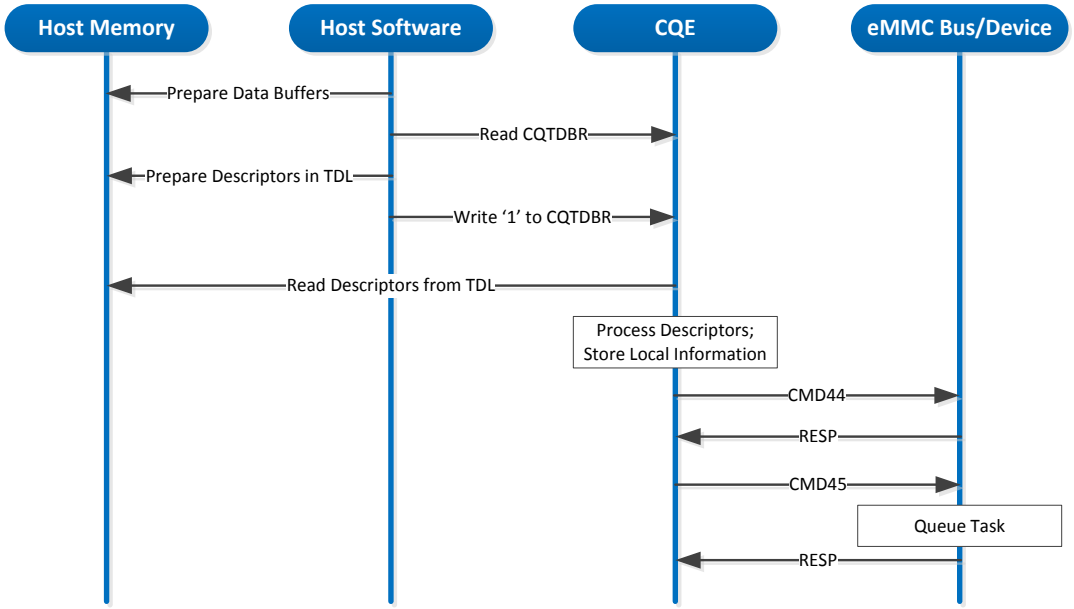


Figure B.112 — Task Queuing Sequence

B.6.3. Task Completion Sequence

The CQE is responsible for task execution, communication with the device and moving the data to the buffers in the host memory. When the task execution is completed, an interrupt may be generated, if requested, or as determined by Interrupt Coalescing mechanism.

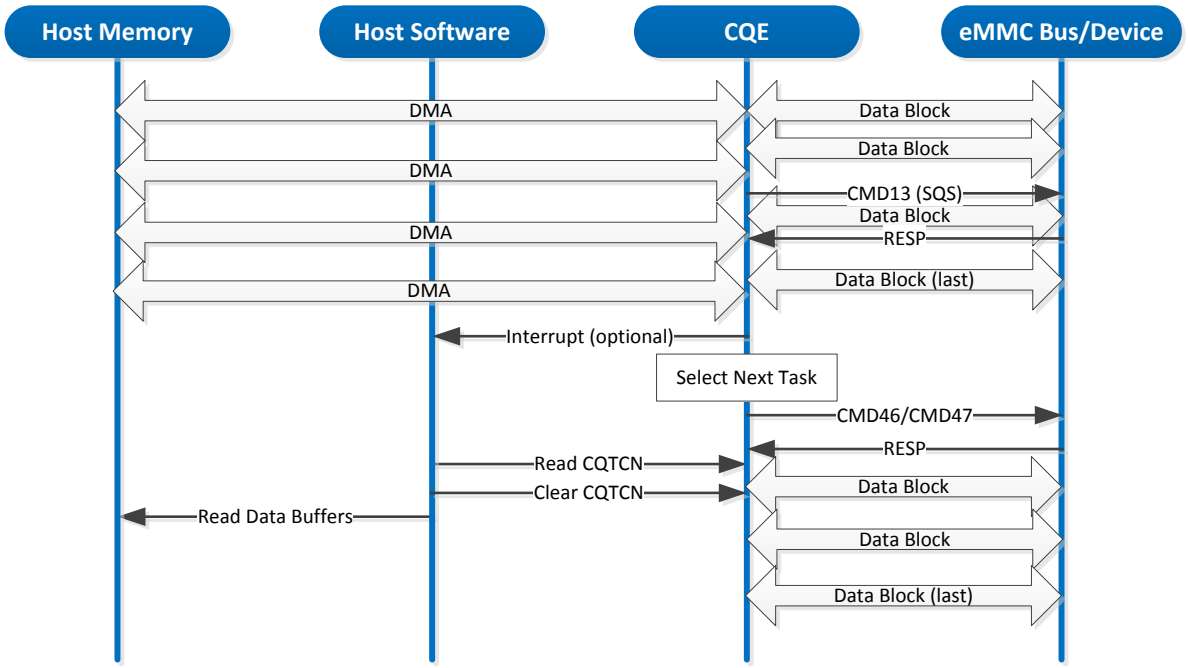


Figure B.113 — Task Execution and Completion Sequence

In the example in Figure B.113, CQE sends a CMD13 during the 2nd-to-last block of a data transfer to determine the queue status, in order to decide which task to queue next. The response to CMD13 may indicate that no task is ready for execution. In such a case the host is required to send CMD13 again, at a later time, which is controlled by CQSSC1.CIT register.

When the task is completed, an interrupt may be generated to host software, and an EXECUTE_READ_TASK (CMD46) or EXECUTE_WRITE_TASK (CMD47) is sent to the device, ordering it to execute the next task.

When receiving the interrupt, host software takes the following steps:

1. Read CQTCN to determine which task(s) has (have) completed. Each bit which is set in CQTCN represents (by its index) a task which has completed but wasn't yet served by software.
2. For every task completed:
   a. Clear appropriate CQTCN bit
   b. Pass a completion notification to the requesting application

B.6.4. Task Discard Sequence (inc. Halting CQE)

In order to discard a task, host software should notify the device and the CQE separately. The task is discarded by the device when a CMDQ_TASK_MGMT (CMD48) is sent. The CQE is issued a "task clear" command using the CQTCLR register. Both operations need to be done, without a specific ordering between them, as long as the CQE is kept in Halt state.

The procedure is illustrated in Figure B.114.

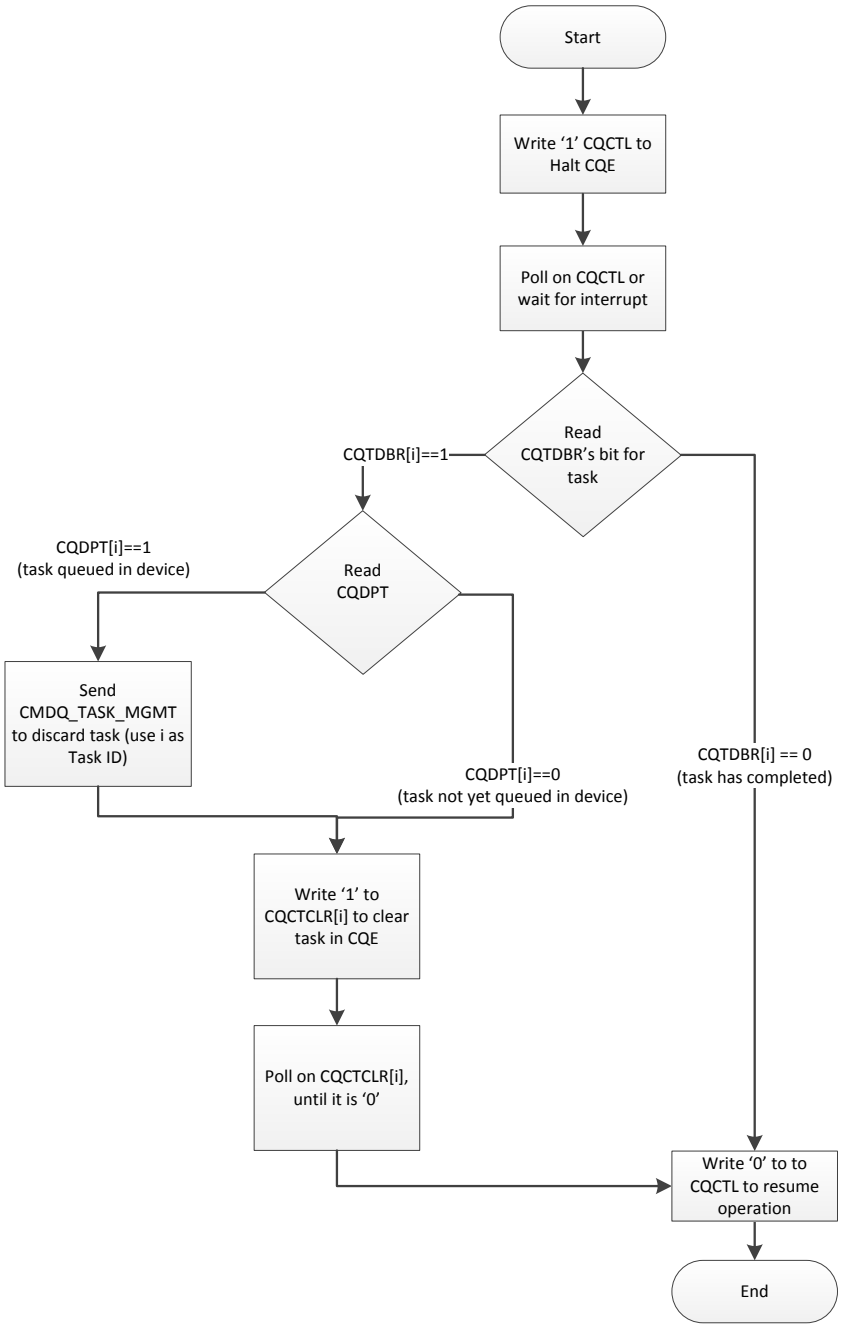


Figure B.114 — Task Discard and Clear Sequence Diagram

B.6.5. Error Detect and Recovery

Error conditions are described in 6.8. The handling of error conditions by CQE is described in B.2.8.

The following steps should be taken by software when responding error interrupts from CQE and *e*•MMC host controller, when CQ mode is in use:

1. Read *e*•MMC host controller Error Interrupt Status register and determine error is related to CQE
2. Write ‘1’ to CQCTL.Halt to halt CQE
3. Wait for CQCTL.Halt to read ‘1’. In some error cases, this may not happen, so software should proceed to the next step after a sufficient time-out.
4. Read CQCRI and CQCRA to determine last response’s index and argument
5. Read CQTERRI to determine the transmitted command’s index and the index of the task to which it is related
6. Perform error-specific recovery procedure
7. Write ‘0’ to CQCTL to resume operation

Annex C (informative) Changes between system specification versions

C.1. Version 4.1, the first version of this standard

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

- The pin number references were removed.
- The form factor references were removed.
- The CSD_STRUCTURE and SPEC_VERS registers were modified to include only allocations applicable to this Electrical standard.
- The S_CMD_SET allocations were removed from this standard and are defined in detail in a separate Application Note.
- The mechanical standard was removed.
- The Appendix A was removed and introduced as a separate document.

C.2. Changes from version 4.1 to 4.2

A major new item is handling densities greater than 2GB. Additional changes include:

- A definition for implementation of media higher than 2GB was introduced.
- The definition for the Device pull-up resistors was clarified.
- Switching between the tran-state and standby states by CMD7 was clarified.
- A new register for indication of the state of an erased block was introduced.
- Command CMD39 argument was clarified.
- The definition of busy indication during write operations was partly changed and partly clarified.
- The minimum voltage of the Low-Voltage range was changed from 1.65V to 1.70V.

C.3. Changes from version 4.2 to 4.3

- Major new items added to this standard are the *e*•MMC definition, boot operation, sleep mode, voltage configuration for *e*•MMC, and reliable write. The chapter dedicated to SPI mode was removed. Additional changes include:
- Added new *e*•MMC features to the feature list.
- Boot operation mode was introduced.
- Sector address definition for Erase and Write Protection was defined.
- CID register setting was changed to recognize either *e*•MMC or a Device.
- The chapter defining SPI mode and all SPI-mode references were removed.
- Sleep mode was introduced.
- Voltage configuration for *e*•MMC was defined.
- Reliable Write was defined.
- Input capacitance for *e*•MMC was defined.
- New bus timings (setup & Hold) were redefined.
- Switch command definition was clarified.
- Peak voltage on all signal lines are redefined for Device and defined for *e*•MMC.

- Redefined Access size register.
- Redefined input capacitance for MMC*micro*, MMC*mobile*, and MMC*plus*.
- Redefined erase-unit size and erase timeout for high-capacity memory.
- Removed "Absolute Minimum" section formerly 4.8.2.
- Defined OCR setting and response for *e*•MMC.
- Defined high-capacity WP group size.
- Alternative boot operation (device-optional) introduced.
- Added "/JEDEC" to "MMCA" as the source of definitions for MID and OID.

C.4. Changes from version 4.3 to 4.4

- Major new items added to this standard are the Dual Data Rate mode, Multiple Partition Supports and Security Enhancement. Additional changes include:
- Introduce of Partition Management features with enhanced storage option.
- Add *Pre-idle* reset arguments at CMD0.
- Clarify CMD1 for Voltage Operation Range and Access mode validation.
- Introduce of New Secure Features, Replay Protected Memory Block.
- Introduce of dual data rate interface with maximum 104MB/s.
- Introduce of Secure Erase, Secure TRIM.
- Introduce of TRIM
- Introduce of New Time value for Secure Erase, Trim.
- Enhancement of write protection with H/W reset and non-reversible register setting.
- Introduce of Replay Protected Memory Block and access control.
- Alternative boot operation was changed as mandatory for device instead of device-optional.
- Introduce of H/W reset signal.
- Introduce New Extended CSD registers.
- Clarify maximum density calculation from SEC_COUNT.
- Clarify of tOSU timing for compatibility.

C.5. Changes from version 4.4 to 4.41

- Major new items added to this standard are the Background Operations and High Priority Interrupt. Additional changes include:
- Added clarification for command operation in RPMB partition.
- Added clarification of address sequence for user area including enhanced area and error condition for partition configuration.
- Added clarification for error behavior when partition is configured without setting ERASE_GROUP_DEF and clarification for the device behavior in case the device received a write/erase command when the condition of ERASE_GROUP_DEF bit has been changed from the previous power cycle.
- Add a clarification for C_SIZE and SEC_COUNT after configuring partitions.
- Added Clarification for the configuration of boot and alternative boot operation.
- Introduce of Enhanced Reliable Write.
- Added clarification for LOCK_UNLOCK feature of the *e*•MMC.

- Introduce of Background Operations.
- Introduce of High Priority Interrupt mechanism, HPI background and one of possible solutions.
- Introduce New Extended CSD registers.
- Corrected equation of General purpose partition size and Enhanced user data area size.
- Removed “File formats for the e.MMC” section formerly section 14.

C.6. Changes from version 4.41 to 4.5

- Removed references to cards in the spec and made an embedded only spec
- Removed stream write and read
- Removed secure erase and Secure trim
- Added HS200 mode
- Added e^2•MMC which supports 2 additional internal voltage nodes and the optional cache command.
- Added the Discard command
- Added the Sanitize command
- Added enhance host and device communication techniques to improve Performance:
  - Extended partition types
  - Packed commands
  - Context IDs
  - Data Tag
- Added Enhance host and device communication techniques to improve Reliability: real time clock and dynamic device capacity
- Added Thermal Specification and Package Case Temperature
- Allocated a region for vendor specific registers
- Provided a path to move from a minimum 512 B access to a minimum 4 KB access
- Updated boot protection to allow boot area protection to be set independently
- Clarified operation for:
  - DDR timing for CRC status, boot acknowledge, start bit and end bit timing
  - Boot diagram
  - CID and CSD in DDR mode
  - Lock/unlock command password replacement
  - Stop transmission
  - RPMB
  - Resistor value in 1.2 V mode
  - SDR voltage range
- Removed “Errata” Annex B

C.7. Changes from version 4.5 to 4.51

- Added Extended Security Commands
- Re-introduced Secure Erase and Secure Trim
- Clarification on e^2•MMC
- Clarified boot bus width retain behavior
- Editorial Clarifications on Context ID, Packet commands, Correctly Programmed Sector Count, Cache, Driver Strength, Illegal commands

C.8. Changes from version 4.51 to 5.0

- Introduced of HS400
  - Data Strobe Line
- Introduced Production State Awareness
- Introduced Field Firmware Update
- Added Sleep Notification in Power Off Notification
- Introduced Device Health Report
- Added Secure Removal Type
- Added overshoot and undershoot
- Added reference load for HS400
- Added pre-amble and post-amble
- Clarified RPMB address failure on Read operation.
- Clarified I/O timing values
- Edit to support Ver.5.0 in EXT_CSD_REV[192]

  NOTE A host system should work properly considering future *e*•MMC version. For example, a host system is expected not to exit only due to the EXT_CSD_REV[192] value greater than 7 which will be used for next *e*•MMC revision."

C.9. Changes from version 5.0 to 5.01

- Added HS400 selection flow diagram
- Clarified clock stopping during read and write operations in HS400 mode
- Editorial Clarifications
- Fixed typos, cross references and formatting
- Generated new figures to improve images quality and correct errors

C.10. Changes from version 5.01 to 5.1

- Added Command Queuing
- Added Enhanced Strobe in HS400 Mode
- Added Cache Barrier
- Added Cache Flushing report
- Added RPMB Throughput Improve. Support write data size 8KB (thirty two 512B frames)
- Added Background Operation Control
- Added Secure Write Protection
- Added Host Controller Interface for Command Queuing as normative reference
- Clarification on HS200/HS400 V_T
- Edit to support v5.1 in EXT_CSD_REV[192]
- Fixed typos, cross references and formatting
- Generated new figures to improve images quality and correct errors

Standard Improvement Form **JEDEC** _____

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

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

JEDEC
Attn: Publications Department
3103 North 10th Street
Suite 240 South
Arlington, VA 22201-2107

Fax: 703.907.7583

1. I recommend changes to the following:

☐ Requirement, clause number _____

☐ Test method number _____ Clause number _____

The referenced clause number has proven to be:

☐ Unclear ☐ Too Rigid ☐ In Error

☐ Other _____

2. Recommendations for correction:

3. Other suggestions for document improvement:

Submitted by
Name: _____ Phone: _____
Company: _____ E-mail: _____
Address: _____
City/State/Zip: _____ Date: _____


JEDEC®