JEDEC STANDARD

Universal Flash Storage (UFS)

JESD220

FEBRUARY 2011

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 2011
3103 North 10th Street
Suite 240 South
Arlington, VA 22201-2107

PRICE: Contact JEDEC

Printed in the U.S.A.

PLEASE!

DON'T VIOLATE
THE
LAW!

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

UNIVERSAL FLASH STORAGE (UFS)

Contents

UNIVERSAL FLASH STORAGE (UFS)

Contents (cont'd)

UNIVERSAL FLASH STORAGE (UFS)

Contents (cont'd)

Contents (cont'd)

Contents (cont'd)

Contents (cont'd)

Figures

UNIVERSAL FLASH STORAGE (UFS)

Contents (cont'd)

UNIVERSAL FLASH STORAGE (UFS)

Contents (cont'd)

Tables

Contents (cont'd)

Contents (cont'd)

Contents (cont'd)

Foreword

This standard has been prepared by JEDEC. The purpose of this standard is definition of an UFS Universal Flash Storage electrical interface and an UFS memory device. This standard defines a unique UFS feature set and includes the feature set of eMMC Specification as a subset. This standard references also several other standard specifications by MIPI (M-PHY and UniPro Specifications) and INCITS T10 (SBC, SPC and SAM Specifications) organizations.

Introduction

The UFS electrical interface is a universal serial communication bus which can be utilized for different type of applications. It’s based on MIPI M-PHY standard as physical layer for optimized performance and power. Architectural model references the INCITS T10 SAM model for ease of adoption.

The UFS device is a universal data storage and communication media. It is designed to cover a wide area of applications as smart phones, cameras, organizers, PDAs, digital recorders, MP3 players, internet tablets, electronic toys, etc.

UNIVERSAL FLASH STORAGE (UFS)

(From JEDEC Board Ballot JCB-11-13, formulated under the cognizance of the JC-64.1 Committee on Electrical Specifications and Command Protocols.)

1 SCOPE

This standard specifies the characteristics of the UFS electrical interface and the memory device. Such characteristics include (among others) low power consumption, high data throughput, low electromagnetic interference and optimization for mass memory subsystem efficiency. The UFS electrical interface is based on an advanced differential interface by MIPI M-PHY standard which together with the MIPI UniPro standard forms the interconnect of the UFS interface. The architectural model is referencing the INCITS T10 SAM standard and the command protocol is based on INCITS T10 (SCSI) SPC and SBC standards.

Universal Flash Storage (UFS) is a simple, high performance, mass storage device with a serial interface. It is primarily for use in mobile systems, between host processing and mass storage memory devices. The following is the summary of the UFS features.

1.1 General Features

- Target Performance
    - UFS version 1.0: 1.25Gbps (Gear1) is mandatory, Support for ~3Gbps (Gear2) is optional ~5.8Gbps max per lane, future UFS revision
- Target Host Applications
    - Mobile phone, UMPC, DSC, PMP, MP3 and any other applications require mass storage , bootable mass storage, and external card
- Target Device Types
    - External Card
        - Micro size for mobile and portable devices
        - Full size or adaptor for DSC and large devices
    - Embedded Packages
        - Mass Storage and Bootable Mass Storage
    - Future expansion of device class types
        - I/O devices, camera, wireless … etc
- Topology: One device per UFS port. A topology to support multiple devices on a single interface is planned to future revision
- UFS Command Set Layer: Simplified SCSI command set based on SBC and SPC. UFS will not modify these SBC and SPC Compliant commands. Option for defining UFS Native command and future extension exist.UFS Transport Protocol Layer: Jedec to define the supported protocol layer, i.e. UTP for SCSI. This does not exclude the support of other protocol in UFS Transport Protocol Layer.
- UFS Interconnect Layer:
    - MIPI M-Phy^{SM} [MIPI M-Phy]
    - MIPI UniPro^{SM} [MIPI UniPro]

1 SCOPE (cont'd)

1.2 Interface Features

- Three Power Supplies: separate Power Supply for I/O & Core
  - VCCQ supply: 1.2V ; logic, controller, and I/O
  - VCCQ2 supply: 1.8V; Controller & IO
  - supply: 1.8/3.3 V ; ie NVM
- Signaling as defined by MIPI M-PHY
  - 400mVp/240mVp (not terminated),
  - 200mVp/120mVp (terminated)
- 8b10b line coding, as defined by MIPI M-PHY
- High reliability – BER under 10-10
- Two signaling schemes supported
  - Low-speed mode with PWM signaling scheme
  - High-Speed burst mode
- Multiple gears defined for both Low-Speed & High-Speed mode

1.3 Functional Features

UFS functional features are NAND management features. These include

- Similar functional features as eMMC
- Defines Write Protect group size for high-capacity devices
- Boot Operation Mode
- Device enumeration & discovery
- Supports Multiple partitions (LUNs) with partition Management
- Supports Multiple User Data Partition with Enhanced User Data Area options
- Support for boot partitions and RPMB partition
- Reliable write operation
- Background operations
- Secure operations, Purge, Erase and Trim to enhance data security
- Write Protection options, including Permanent & Power-On Write Protection
- Signed access to a Replay Protected Memory Block
- HW Reset Signals

2 UFS ARCHITECTURE OVERVIEW

2.1 UFS Top level Architecture

Figure 2-1 shows the Universal Flash Storage (UFS) top level architecture.

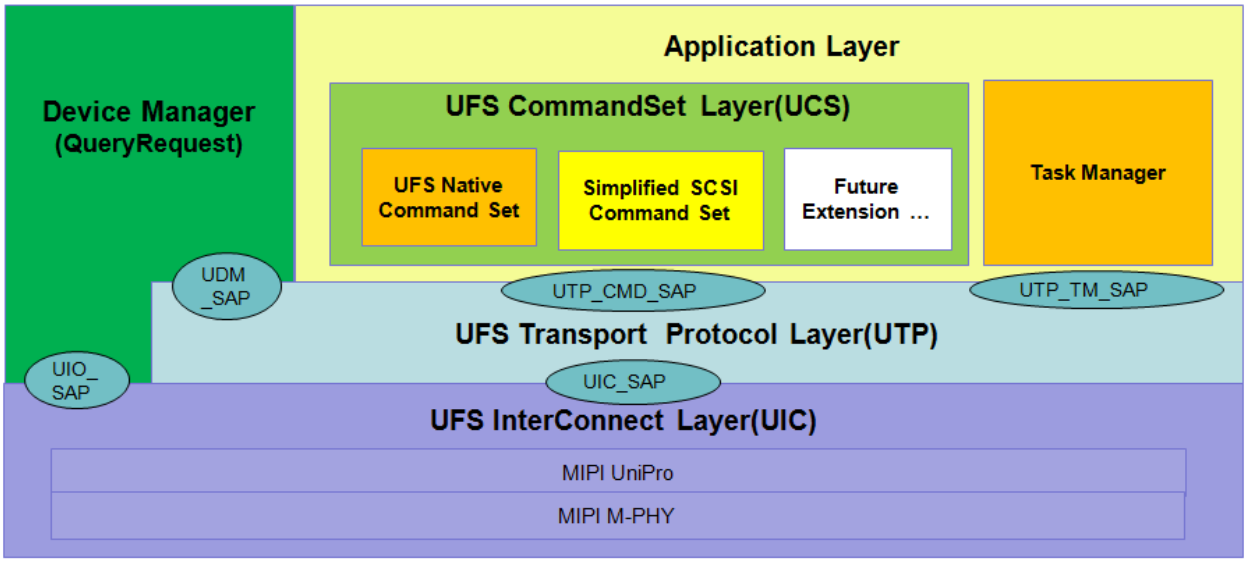

Figure 2-1 — UFS Top Level Architecture

UFS communication is a layered communication architecture. It is based on SCSI SAM-5 architectural model [SAM].

Application Layer

The application layer consists of UFS command set layer (UCS), device manager and Task Manager. The UCS will handle the normal commands like read, write, and so on. UFS may support multiple command sets. UFS is designed to be protocol agnostic. But for Version 1.0, UFS have decided to use SCSI as the baseline protocol layer. A simplified SCSCI command set was selected for UFS. UFS Native command set can be supported when it is needed to extend the UFS functionalities.

The Task Manager handles commands meant for command queue control. The Device Manager will provide device level control like Query Request and lower level link-layer control.

UFS Device Manager

The device manager has the following two responsibilities:

- Handling device level operations.
- Managing device level configurations.

Device level operations include operations like device sleep, device deep sleep etc. These mainly involve power management of the device.

Device level configuration is managed by the device manager by maintaining and storing a set of descriptors. The device manager would handle commands like query request which is meant for modifying and retrieving configuration information of the device.

2.1 UFS Top level Architecture (cont'd)

Service Access Points

As seen from the diagram the device manager interacts with lower layers using the following two service access points

- UDM_SAP
- UIO_SAP

UDM_SAP is the service access point exposed by the UTP for the device manager to allow handling of device level operations and configurations. For example the handling of query request for descriptors would be done using this service access point. The following diagram depicts the usage of the service access point.

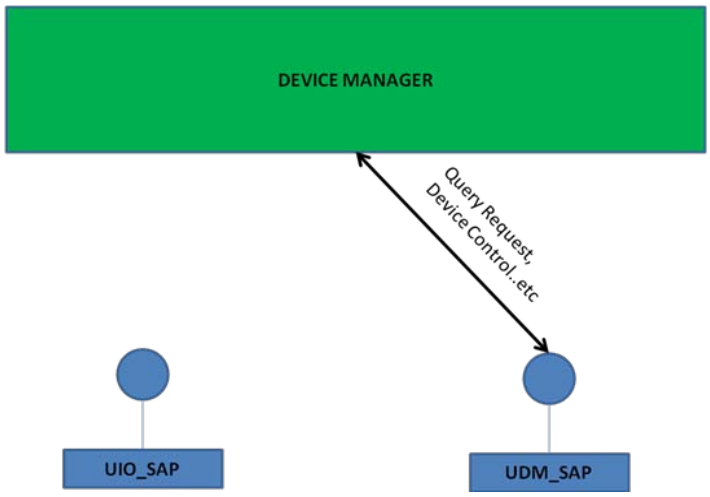

Figure 2-2 — Usage of UDM_SAP

UIO_SAP is the service access point exposed by the UIC layer for the device manager to trigger the reset of the UIC layer and also to service a reset request from the host. The following diagram depicts the usage of the service access point.

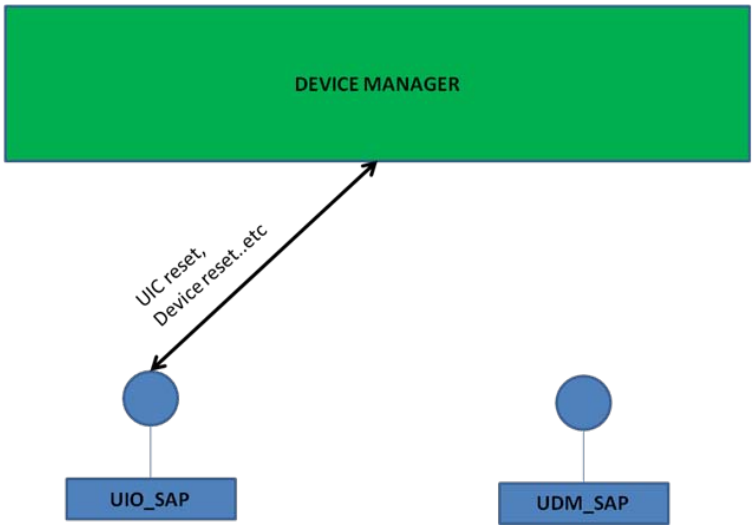

Figure 2-3 — Usage of UIO_SAP

2.1 UFS Top level Architecture (cont'd)

UIO_SAP

UIO_SAP is the service access point exposed by the UIC layer. In Unipro/M-PHY, UIO_SAP corresponds to DME_SAP. The DME_SAP have Service Primitives including resetting the entire UniPro protocol stack and UFS device reset, and so on.

- UNIPRORESET : It is used when the UniPro stack has to be reset.
- ENDPOINTRESET: It is used when UFS Host wants the UFS Device to perform an reset.

For the detailed internal mechanism, refer the UniPro spec released by MIPI. (MIPI is Mobile Industry Processor Interface).

UDM_SAP

UDM_SAP is the service access point exposed by the UTP layer to the Device Manager for UFS device-level functions. UDM_SAP corresponds to the Query Request and Query Response Functions defined by the UFS UTP specification.

Most details in the Query Function Transport Protocol Services, and the Query Request and Query Response UPIU sections.

UFS Transport Protocol Layer

The UFS Transport Protocol (UTP) layer provides services for the higher layer . UPIU is "UFS Protocol Information Unit" which is exchanged between UTP layers between UFS host and UFS device. For example, if Host-side UTP receives the request from application layer or Device manager, UTP generates UPIU for that request and transports the generated UPIU to the peer UTP in UFS device side. The 3 services provided by UTP are like following. (1) UFS Device Manger Service Access Point (UDM_SAP) to perform the device level management like descriptor access. (2) UTP Command Service Access Point (UTP_CMD_SAP) to transport commands, and (3) UTP Task-Management Service Access Point (UTP_TM_SAP) to transport task-management function like "abort task" function.

UFS Interconnect Layer

The lowest layer is UFS Inter-connect Layer (UIC) handles connection between UFS host and UFS device. UIC consists of MIPI UniPro and MIPI M-PHY. The UIC provides 2 services to upper layer. The UIC Service Access Point (UIC_SAP) to transport UPIU between UFS host and UFS device. The UIC_SAP corresponds to T_SAP in UniPro. The UIC IO control Service Access Point (UIO_SAP) to manage UIC. The UIO_SAP corresponds to DME_SAP in UniPro.

UFS Topology

In UFS specification version 1.0, one device per UFS port is adopted. Future revision of the UFS specification will evaluate different topologies.

2.2 UFS System Model

Figure 2-4 shows an example of UFS system. It shows how host UFS Host and UFS device is connected, the position of UFS host controller and its related UFSHCI interface.

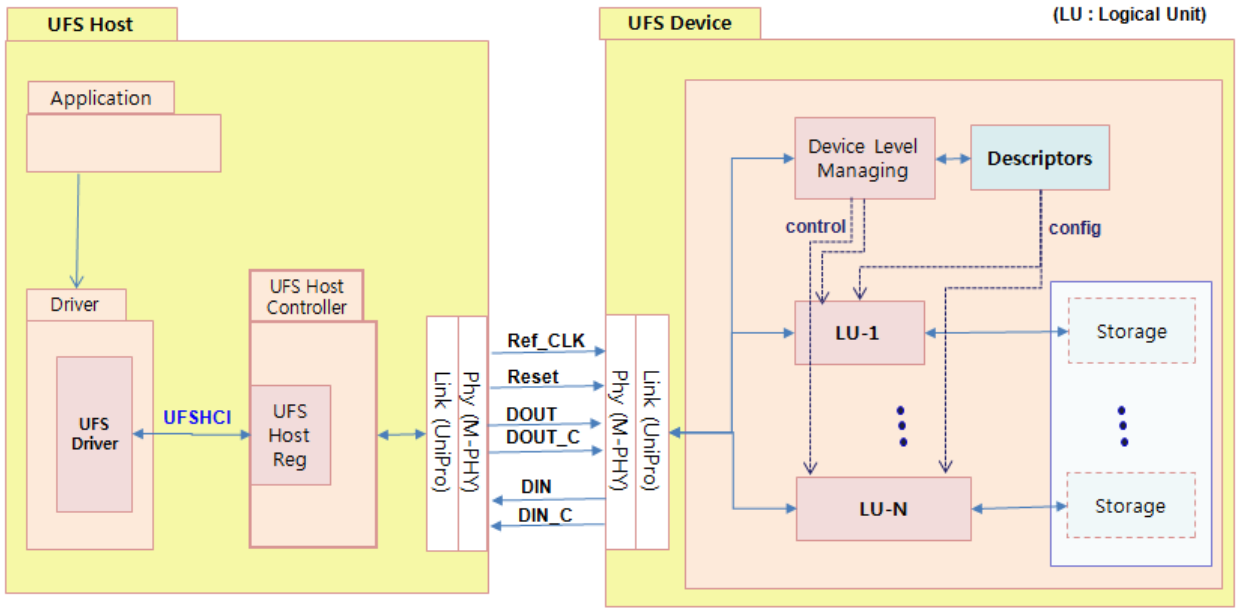

Figure 2-4 — UFS System Model

The UFS host consists of the application which wishes to communicate with the UFS device. It communicates with the device using the UFS driver. The UFS driver is meant for managing the UFS host controller through the UFSHCI (UFS Host Controller Interface). The UFSHCI is basically a set of registers exposed by the host controller.

Figure 2-4 also indicates the UFS interface between the UFS host and the UFS device. The link layer and the physical layer on the host and device side make up the UFS interface. UniPro has been adopted as the link layer for UFS while M-PHY has been adopted as the physical layer for UFS. The M-PHY is differential, dual simplex PHY that include TX and RX pairs.

Potential UFS devices can be memory card (full size and micro size), embedded bootable mass storage devices, IO devices etc. A UFS device comprises of multiple logical units, device manager and descriptors. The device manager performs device level functions like power management while the logical unit performs functions like read, write etc. The descriptors are meant for storage of configuration related information.

2.3 Booting & Enumeration

UFS booting from a bootable storage device will initiate after the UFS InterConnect Layer (MIPI M-PHY and Unipro) is power-up to its default state. Code can be read from the appropriate boot partition, or as desired, boot ROM code can re-configure MIPI M-Phy and Unipro to appropriate setting before reading code from boot partition.

Multiple boot partitions may be available from one UFS device. However, only one partition may be active at power-up. Appropriate descriptors are available to configure the default boot partition.

During booting, all accesses to boot partition are supported via SCSI commands.

2.4 UFS Physical Layer Signals

The UFS Physical layer defines the PHY portion of the UFS interface that connects two UFS devices. This is based on MIPI M-PHY specification. Note that UFS interface can support multiple lanes. For every lane, the PHY consist of two differential pairs, one transmit pair and one receive pair.

In a multi-lane architecture, lanes may be added in multiples of 2 (i.e., 2 or 4) and, numbered accordingly. An equal number of downstream and upstream lanes must be provided in each link.

The following table summarizes the basic signals required for an embedded UFS device. Note that only the single lane, per direction, per link, configuration is shown. A memory card solution would require a subset of these signals. And a full detail specification of the signals would include in the PHY chapter and the UFS mechanical specification.

Table 2-1 — UFS Signals

Name	Type	Description
REF_CK	Input	Reference Clock: Relatively low speed clock common to all UFS devices in the chain, used as a reference for the PLL in each device.
DIN_0_t DIN_0_c	Input	Downstream lane input zero: first lane differential input true and complement signal pair.
DOUT_0_t DOUT_0_c	Output	Upstream lane output zero: first lane differential output true and complement signal pair.
RESET_N	Input	UFS Device HW Reset signal

2.5 UFS Link Layer – MIPI Unipro

The UFS Link Layer is based on MIPI Unipro specification. It is responsible for management of the link, including the PHY. Note that device management is outside the scope of the link layer and is the responsibility of the upper layer SW.

The basic interface to the link layer is Unipro definition of a C-Port. C-Port is used for all data transfer as well as all control and configuration messages. Multiple C-Ports can be supported on a UFS device and the # of C-Ports will be implementation dependent. For multiple C-Ports implementation, channels must be bind to the appropriate C-Port.

Traffic sent over Unipro link can be classified as TC0 or TC1 traffic class with TC1 as higher priority traffic class.

JEDEC UFS will take advantage of 3 basic types of Unipro services. These include data transfer service, config/control/status service, and interrupt/notification service.

For more details regarding the link layer, please refer to the link layer chapter and MIPI Unipro specification.

2.6 MIPI UniPro Related Attributes

In general the UniPro related attributes, values and use of them are defined in the MIPI UniPro Specification. The attributes may be generic for all UniPro applications and thus out of scope of this document. Following attributes are still defined in this specification specifically for UFS application as follows in the table.

Table 2-2 ManufacturerID and DeviceClass Attributes

Attribute	AttributeID*	Value	Description
ManufacturerID	0x5004		MIPI manufacturer ID. MIPI MID shall be used in this Attribute also for UFS applications. The ID can be requested from MIPI.
DeviceClass	0x5002	Memory = 0x02 Host = 0x03	UniPro DeviceClass ID for UFS application
* Reference MIPI Alliance Specification for Device Descriptor Block [MIPI-DDB]			

2.7 UFS Transport Protocol (UTP) Layer

As mentioned previously, the Transport Layer is responsible for encapsulating the protocol into the appropriate frame structure for the link layer. UFS is protocol agnostic and thus any protocol will need the appropriate translation layer. For UFS Version1.0 specification, this is UTP (UFS Transport Protocol).

Currently, all accesses are support only through SCSI. However, UFS TG reserves the right to establish the appropriate API/service/extension for future features or requirements.

A design feature of UTP is to provide a flexible packet architecture that will assist the UFS controller in directing the encapsulated command, data and status packets into and out of system memory. The intention is to allow the rapid transmittal of data between the host system memory and the UFS device with minimal host processor intervention. Once the data structures are set up in host memory and the target device is notified, the entire commanded transaction can take place between the UFS device and the host memory. The means by which the UFS controller transfers data into and out of host memory is via a hardware and/or firmware mechanism that is beyond the scope of this document. See the UFS controller specification for further information.

A second feature of the UTP design is that once a device receives a command request notification from the host, the device will control the pacing and state transitions needed to satisfy the data transfers and status completion of the request. The idea here is that the device knows its internal condition and state and when and how to best transfer the data that makes up the request. It is not necessary for the host system or controller to continuously poll the device for “ready” status or for the host to estimate when to start a packet transfer. The device will start the bus transactions when it determines its conditions and status are optimal. This approach cuts down on the firmware and logic needed within the host to communicate with a device. It also affords the maximum possible throughput with the minimum number of bus transactions needed to complete the operation.

2.7.1 Architectural Model

The SCSI Architecture Model SAM-5 [SAM] is used as the general architectural model for UTP. The SAM Architecture is a client-server model or more commonly a request-response architecture.

2.7.1.1 Client-Server Model

A client-server transaction is represented as a procedure call with inputs supplied by the application client on the Initiator. The procedure call is processed by the server and returns outputs and a procedure call status. Client-server relationships are not symmetrical. A client only originates requests for service. A server only responds to such requests.

Initiator (host) and Target (UFS device) are mapped into UFS physical network devices. An Initiator may request processing of a command or a task management function through a request directed to the Target. Device service requests are used to request the processing of commands and task manager requests are used to request the processing of task management functions.

Target is a Logical Unit (LU) contained within a UFS device. A UFS device will contain one or more Logical Units. A Logical Unit is an independent processing entity within the device.

An Initiator request is directed to a single Logical Unit within a device. A Logical Unit will receive and process the client command or request. Each Logical Unit has an address within the Target device called a Logical Unit Number (LUN).

2.7.1.1 Client-Server Model (cont'd)

Communication between the Initiator and Target is divided into a series of messages. These messages are formatted into UFS Protocol Information Units (UPIU) as defined within this specification. There are

a number of different UPIU types defined. All UPIU structures contain a common header area at the beginning of the data structure (lowest address). The remaining fields of the structure vary according to the type of UPIU.

A Task is a command or sequence of actions that perform a requested service. A Logical Unit contains a task queue that will support the processing of one or more Tasks. The Task queue is managed by the Logical Unit. A unique Initiator provided Task Tag is generated by the Initiator when building the Task. This Task Tag is used by the Target and the Initiator to distinguish between multiple Tasks. All transactions and sequences associated with a particular Task will contain that Task

Tag in the transaction associated data structures. Only one command within a Task can be active. Only one Task can be processed at a time within a Logical Unit. If a device contains multiple Logical Units, it could have the ability to process multiple Tasks simultaneously or concurrently if so designed.

Command structures consist of Command Descriptor Blocks (CDB) that contain a command opcode and related parameters, flags and attributes. The description of the CDB content and structure are defined in detail in the [SAM] and [SPC] and device specific (RBC, etc.) specifications.

2.7.1.1 Client-Server Model (cont'd)

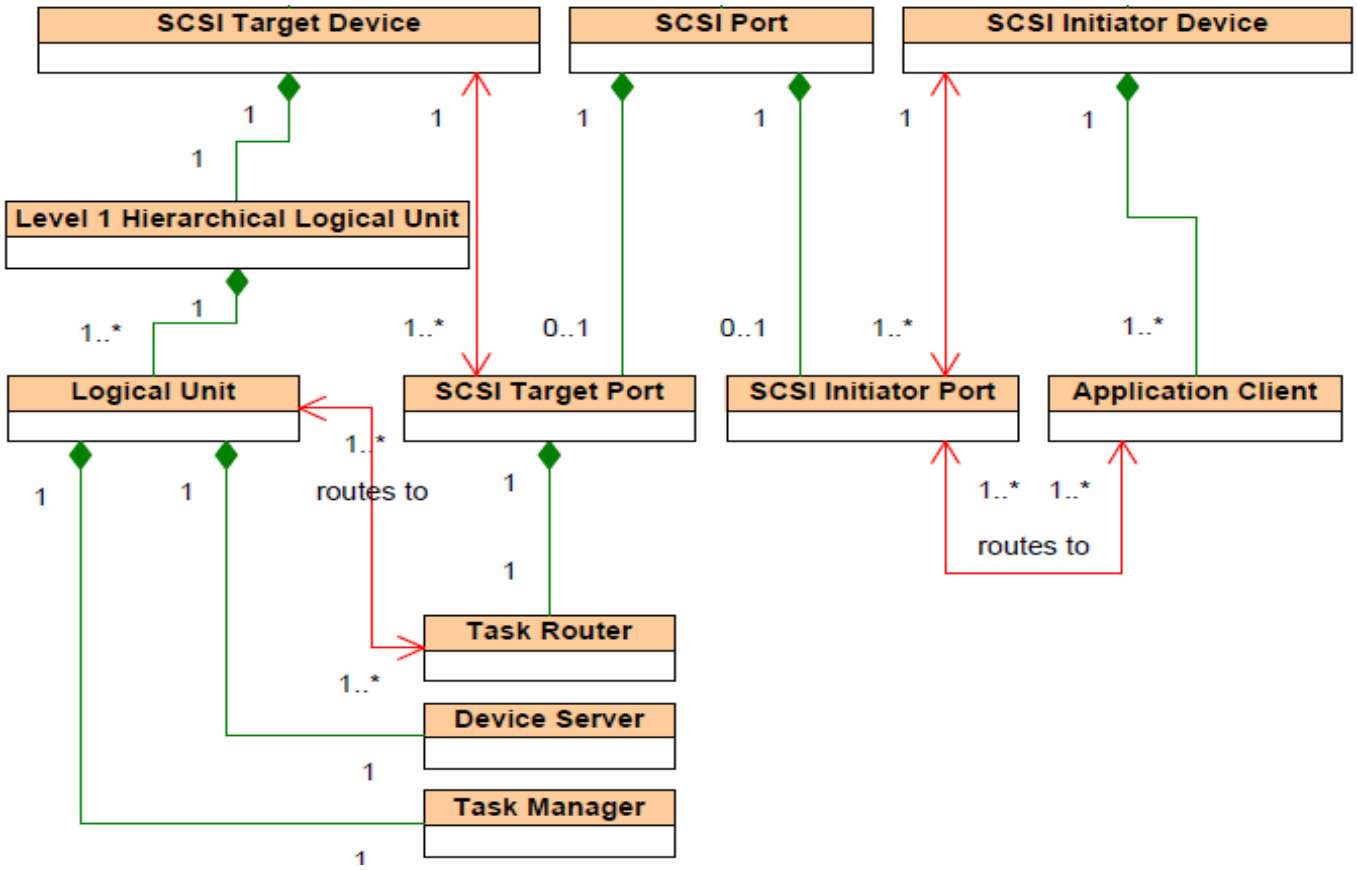

Figure 2-5 — SCSI Domain Class Diagram

2.7.1.1 Client-Server Model (cont'd)

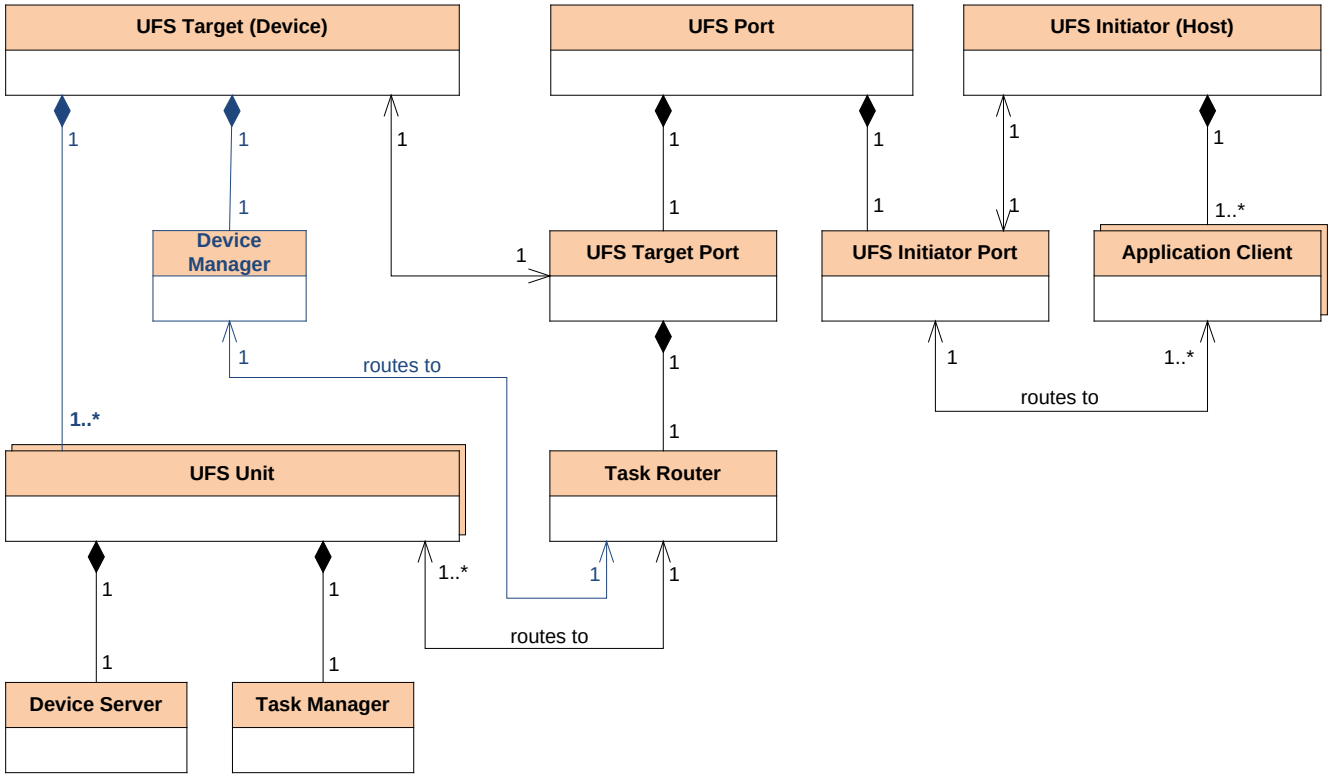

Figure 2-6 — UFS Domain Class Diagram

2.7.1.2 CDB, Status, Task Management

UTP adopts Command Descriptor Block (CDB) format for commands, device status data hierarchy and reporting method, and task management functions of outstanding commands, described in [SAM]. Regardless of the command protocol to be delivered by UTP, SCSI CDB, Status and Task Management Functions should be adopted uniformly in UFS devices.

2.7.1.3 Nexus

The nexus represents the relationship among the Initiator, Target, Logical Unit and Command (Task)

- Nexus notation: I_T_L_Q nexus; where I =Initiator, T = Target, L = Logical Unit, Q = Command

There shall only be one Initiator, the host, and one target, the UFS device in UFS definition. There shall be one or more logical units in a UFS device. The command identifier (i.e., the Q in an I_T_L_Q nexus) is assigned by the Initiator to uniquely identify one command in the context of a particular I_T_L nexus, allowing more than one command to be outstanding for the I_T_L nexus at the same time.

2.7.1.4 SCSI Command Model

All command requests originate from application clients in an Initiator. An application Client requests the processing of a command with the following procedure call:

- Service Response = Execute Command (IN (I_T_L_Q Nexus, CDB, Task Attribute, [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [CRN], [Command Priority]), OUT ([Data-In Buffer], [Sense Data], [Sense Data Length], Status, [Status Qualifier]))

Parameter fields in the UTP Command, Response, Ready-to-Transfer, Data-Out, Data-In UPIU headers contain the requisite information for the input and output arguments of the Execute Command procedure call in compliance with SAM-5.

2.7.1.5 SCSI Task Management Functions

An application client requests the processing of a task management function with the following procedure call:

- Service Response = Function name (IN (Nexus), OUT ([Additional Response Information])

Parameters fields in the UTP Task Management Request and Task Management Response UPIU headers contain the requisite information for the input and output arguments of the Task Management Function procedure call in compliance with SAM-5.

2.8 UFS Application and Command Layer

UFS interface is designed to be protocol agnostic interface. However, as mentioned previously, SCSI has been selected as the baseline protocol layer for version 1.0 UFS specification. Descriptors are available to identify and select the appropriate protocol for UFS interface.

The primary functions of the Command Set Layer are to establish a method of data exchange between the UFS Host & Devices and to provide fundamental device management capability. SCSI SBC and SPC commands are the baseline for UFS. UFS will not modify the SBC and SPC Compliant commands. The goal is to maximize re-use and leverage of the SW codebase available on platforms (PC, netbook, MID) that are already supporting SCSI.

Options are available to define UFS Native commands and extension as needed.

UFS Simplified SCSI command set includes

1. SBC compliant commands, from Table 10 [SBC].

    - FormatUnit
    - Inquiry
    - Read(6) & Read(10)
    - ReadCapacity(10)
    - ReportLun
    - RequestSense
    - Send Diagnostic
    - TestUnitReady

2.8 UFS Application and Command Layer (cont'd)

2. SPC Compliant commands, from Table 44 of [SPC].

   - Inquiry
   - ReportLun
   - TestUnitReady

3. SCSI Operational commands for UFS applications & compatible with existing SCSI driver

   - Write(6) & Write(10)->
   - Start Stop Unit
   - Power Mode, UFS Specific (TBD)
   - SynchronizeCache(10)
   - Verify

4. Value-added commands for UFS.

   - Read(16), Write(16) & ReadCapacity(16), applicable to 512B block device w capacity larger than 2TB in any LU

Please refer to UFS Command Set Layer chapter for more details.

3 UFS ELECTRICAL: CLOCK, RESET, SIGNALS & SUPPLIES

3.1 Embedded UFS Signals

The Figure 3-1 below represents a conceptual drawing of UFS memory device. Utilization of internal regulators and connection of those to different parts of the sub-system may differ per implementation.

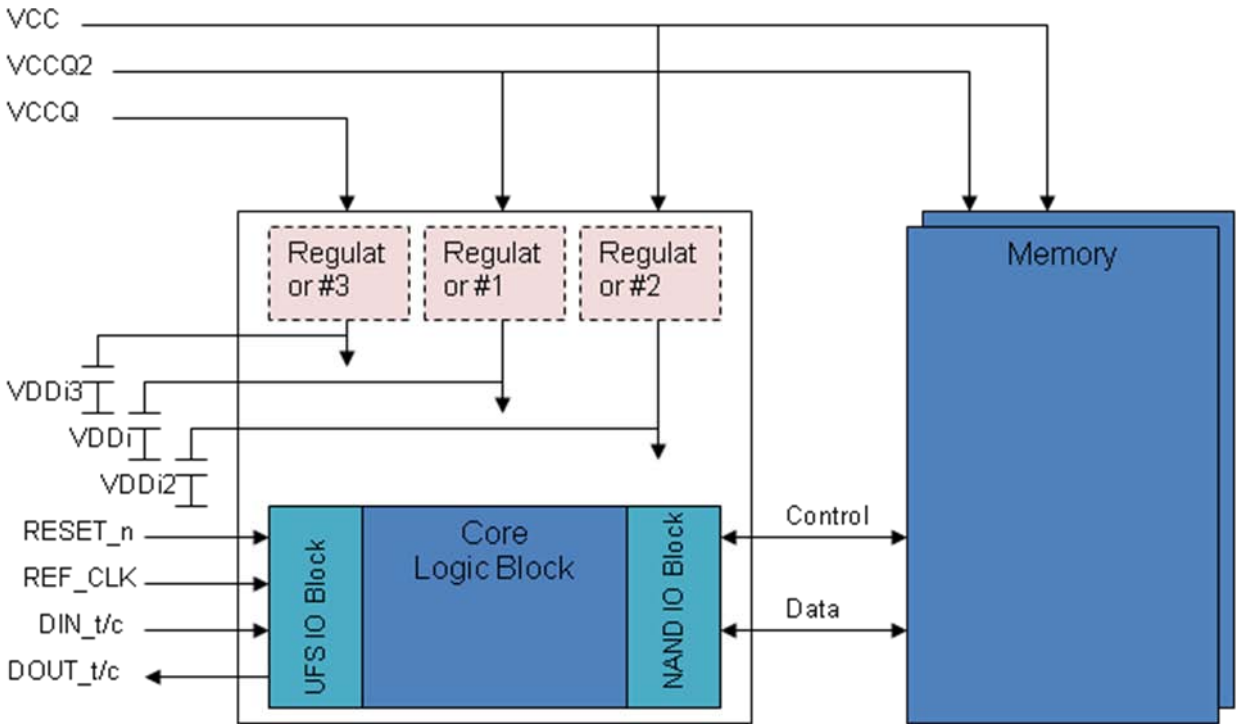

Figure 3-1 — USF Device Block Diagram

3.1 Embedded UFS Signals (cont'd)

Table 3-1 — Signal Name and Definitions

Name	Type	Description
VCC	Input	Supply voltage for the memory devices
VCCQ	Input	Supply voltage used typically for the memory controller and optionally for the PHY interface and any other internal very low voltage block
VCCQ2	Input	Supply voltage used typically for the PHY interface and the memory controller and any other internal low voltage block
VDDi*	Input	Input terminal to provided bypass capacitor for internal regulator typically related to the memory controller
VDDi2*	Input	Input terminal to provide bypass capacitor for internal regulator #2, typically related to memory IF
VDDi3*	Input	Input terminal to provide bypass capacitor for internal regulator #3
VSS	Input	Ground
RESET_n	Input	Input HW RESET signal. This is an active low signal
REF_CLK	Input	Input reference clock. When not active, this signal should be pull-down or driven low by the host SoC.
DIN_t DIN_c	Input	Downstream data lane: differential input signals into UFS device from the host
DOUT_t DOUT_c	Output	Upstream data lane: differential output signals from the UFS device to the host
C+	Input	Optional charge pump capacitor, positive terminal. For more information, see 3.6
C-	Input	Optional charge pump capacitor, negative terminal. For more information, see 3.6
CPout1, CPout2	Input	Charge pump output capacitor. For more information, see 3.6

* If there is no internal regulator requiring output capacitor then VDDi pins should be internally connected as follows:

- VDDi to VCCQ2
- VDDi2 to VCC
- VDDi3 to VCCQ

3.2 UFS Memory Card Signals

[Reserved for future Memory Card definition]

3.3 RESET_n Signal

To meet the requirements of the JEDEC standard JESD8-12A, the RESET_N signal voltages shall be within the following specified ranges for VCCQ.

Table 3-2 — RESET_n Signal Electrical Parameters

Parameter	Symbol	Min	Max	Unit	Notes
Input HIGH voltage	VIH	0.65*VCCQ	VCCQ+0.3	V	For VCCQ as defined in Table 3-3
Input LOW voltage	VIL	VSS-0.3	0. 35*VCCQ	V	For VCCQ as defined in Table 3-3
Input Capacitance	Cin		10	pF	
Input leakage Current	Ilkg		10	µA	

3.4 Power Supplies

Table 3-3 — UFS Supply Voltages

Parameter	Symbol	Min	Max	Unit	Notes
Supply Voltage (memory)	VCC	2.7	3.6	V	
		1.70	1.95	V	
Supply Voltage (controller and IO)	VCCQ	1.1	1.3	V	JESD8-12A
Supply Voltage (secondary for controller and IO)	VCCQ2	1.65	1.95	V	
Supply power-up for 3.3V	tPRUH		35	ms	
Supply power-up for 1.8V	tPRUL		25	ms	
Supply power-up for 1.2V	tPRUV		20	ms	
Internal regulator capacitors	VDDi, VDDi2, VDDi3	0.1 1 1		µF µF µF	

3.4 Power Supplies (cont'd)

UFS device must support at least one of the valid voltage configurations, and can optionally support all valid voltage configurations highlighted within the table below.

Table 3-4 — Voltage configurations for Embedded UFS

Embedded UFS		VCCQ	VCCQ2
		1.1V-1.3V	1.65V-1.95V
VCC	2.7V-3.6V	Valid	
	1.7V-1.95V	Valid	

Table 3-5 — Voltage configurations for UFS Card

UFS Card		VCCQ
		1.1V-1.3V
VCC	2.7V-3.6V	Valid
	1.7V-1.95V	Not Valid

3.5 Reference Clock

The M-PHY Specification defines the reference clock optional for the State Machine Type I. As the PWM signaling is self-clocked the reference clock is not required for the data latching. Still existence of the reference clock may be utilized to enable lower BER and faster HS mode PLL/DLL locking. Thus a UFS host shall implement a UFS device square wave single ended reference clock driver. And a UFS memory device shall implement a square wave single ended reference clock input. Details of the characteristics of the reference clock as followed.

3.5 Reference Clock (cont'd)

Table 3-6 — Reference Clock Electrical Characteristics

Parameter	Symbol	Min	Max	Unit	Notes
Frequency	f_{ref}	19.2 26 38.4 52		MHz	These are the favorable frequencies to achieve the HS-Burst Rates A & B with integer multipliers
Frequency Error	f_{ERROR}	-150	+150	ppm	
DC Output High Voltage	V_{OH}	0.75 VCCQ		V	Note 1, 2
DC Output Low Voltage	V_{OL}		0.25 VCCQ	V	
Input High Voltage	V_{IH}	0.65 VCCQ		V	Note 2
Input Low Voltage	V_{IL}		0.35 VCCQ	V	
Clock Rise Time	T_{rise}		2	ns	Measured (note 1) from 20 to 80% of VOL_max to VOH_min (note 3)
Clock Fall Time	T_{fall}		2	ns	Measured (note 1) from 20 to 80% of VOL_max to VOH_min (note 3)
Duty Cycle	T_{DC}	45	55	%	Measured (note 1) at the crossings of the RefClk signal with the midpoint reference (note 3)
Phase Noise	N		-66	dBc	Integrated phase noise from 50kHZ to 10MHz
Noise Floor Density	$N_{density}$		-140	dBc/Hz	White noise floor
Test Load Impedance	RL_{Test}	10		kΩ	Resistive and Capacitive load to ground (NOTE 1, 4, 6)
	CL_{Test}	20		pF	
Input Impedance	RL_{RX}	10		kΩ	Note 5
	CL_{RX}		10	pF	

3.5 Reference Clock (cont'd)

Table 3-6 — Reference Clock Electrical Characteristics (cont'd)

NOTE 1 Output load is defined as 20 pF shunted by 10 KΩ

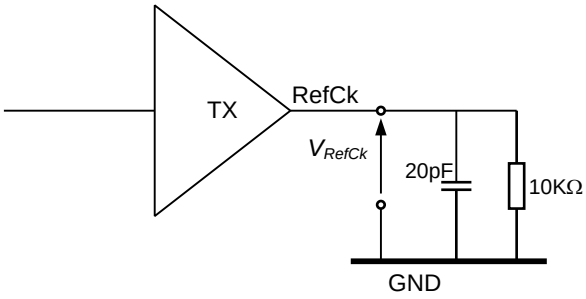

Figure 3-2 — Test Load Impedance

NOTE 2 Following graph shows Output driver and Input receiver levels. RefCk Driver AC voltage (e.g. ring back) shall be kept inside Output Voltage limit.

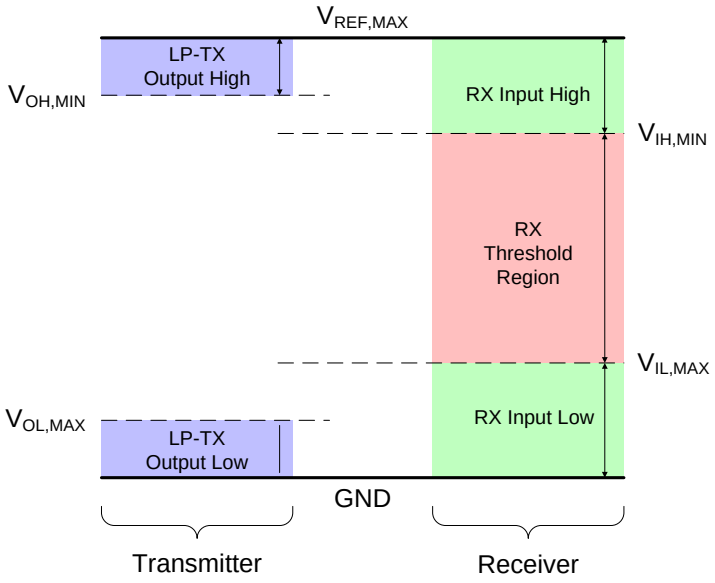

Figure 3-3 — Output driver and Input receiver levels

3.5 Reference Clock (cont'd)

Table 3-6 — Reference Clock Electrical Characteristics (cont'd)

NOTE 3 Rise and Fall time are measured in 20-80% window defined by VOL,max and VOH,min. Duty Cycle is measured at the crossing point of the RefCk signal with Vmid, defined as:

$$V_{mid} = \frac{V_{ol,\max} + V_{oh,\min}}{2}$$

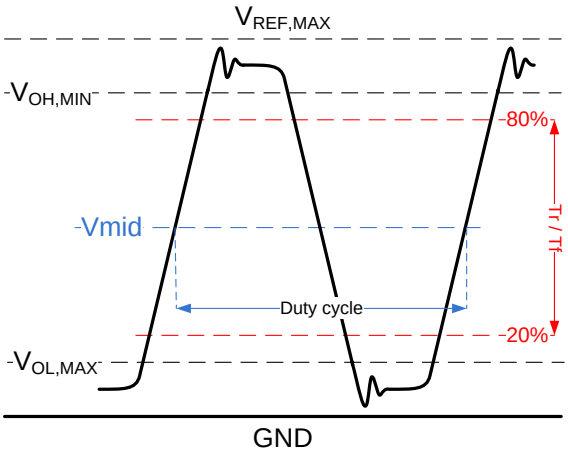

Figure 3-4 — Rise time, Fall time and Duty Cycle

NOTE 4 Including transmitter output, Tx package, interconnect, Rx package and Rx input impedance

NOTE 5 Including Rx package and Rx input impedance

Reference Clock buffer might be required when long interconnect are used. The distance between a Tx/Rx set is application dependant and might be defined as the following example: Assuming nominal interconnect capacitance provides around 1pf/cm, Tx package pin capacitance is about 3pF, Rx package pin capacitance is about 10pF load and max Tx load is 20pF so the maximum interconnect length is

$$\frac{(20\,pF - 3\,pF - 10\,pF)}{1\,pF\,/\,cm} = 7cm$$

Distance between a RefCk driver and its receiver should not exceed 7cm in this example.

NOTE 6 The Test Load Impedance is placed at the output of the driver, with the shortest interconnect.

3.6 External Charge Pump Capacitors (Optional)

In order to produce memory devices that can accommodate a low voltage core supply (Vcc = 1.8 V) an internal Charge Pump circuit may be required.

Charge Pump circuit requires extra-sized passive components. An optional usage of external Charge Pump capacitors is provided. Figure 3-5 shows the electrical connections required in case of Charge Pump implementation that uses external capacitors.

3.6 External Charge Pump Capacitors (Optional) (cont'd)

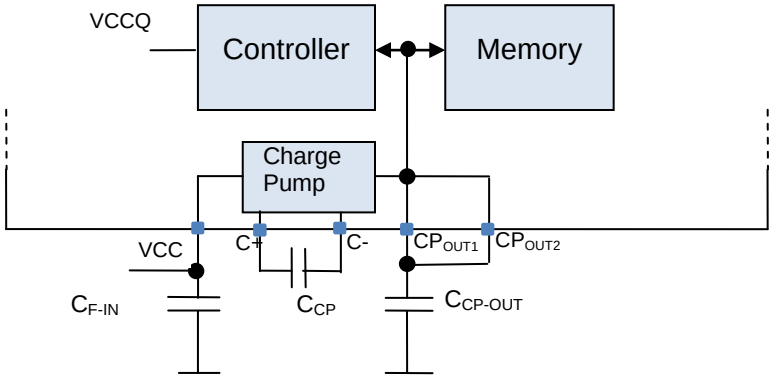

Figure 3-5 — Electrical Connections of External Charge Pump Capacitors

Table 3-7 provides description of the capacitors to be used.

Table 3-7 — CP Capacitors Description

Capacitor Name	Min	Typ	Max	Description
C_{F-IN}	TBD	4.7uF	TBD	When charge pump is used, this capacitor is used as the charge pump input bypass capacitor. When charge pump is not used this capacitor is used as a bypass capacitor for the memory.
C_{CP-OUT}	TBD	4.7uF	TBD	Charge pump output bypass capacitor
C_{CP}	TBD	0.22uF	TBD	Charge pump flying capacitor

The Charge Pump Capacitors are optional for Embedded UFS devices. Even though all defined embedded UFS packages shall include ball placement for the given optional capacitors. The ball names and description are given in Table 3-6.

Table 3-8 — CP related ball names

Ball Name	Description
C+	CCP capacitor's positive terminal
C-	CCP capacitor's negative terminal
CPout1, CPout2	Charge pump output capacitor (2 balls)[1]
1) Two CPout balls are required to reduce inductance, improve ripple and transient response NOTE The given capacitors shall be placed close to the memory device to minimize the inductance. As a guideline for package design – it is recommended to place the CP related balls close to each other and close to the edge of the package.	

4 RESET, POWER-UP & POWER-DOWN

4.1 Reset

Following sub-sections define the means for resetting the UFS Device or a layer of it.

4.1.1 Power-on Reset

The host may reset the UFS Device by switching the VCCQ and VCCQ2 (and VCC) power supplies off and back on. The UFS Device shall have its own power-on detection circuitry which puts the UFS Device and all the different layers of it into a defined state after the power-on.

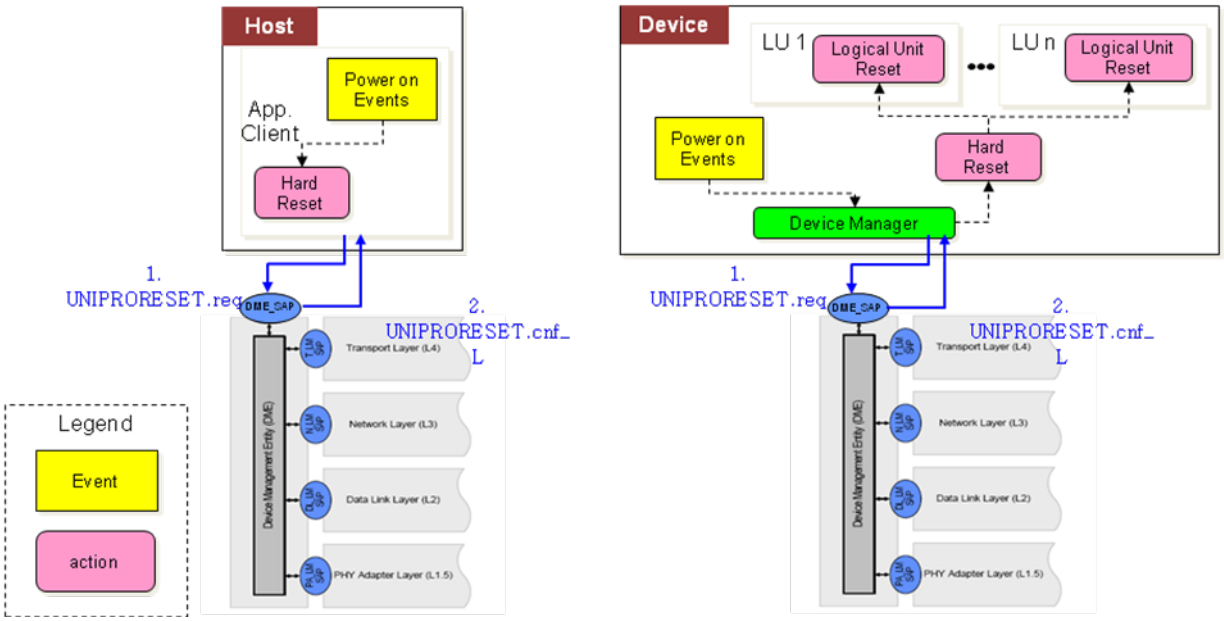

Figure 4-1 — Power-on Reset

4.1.2 HW Reset Pin

A dedicated HW reset signal is also defined for the UFS Device. This signal is applicable for embedded UFS devices only.

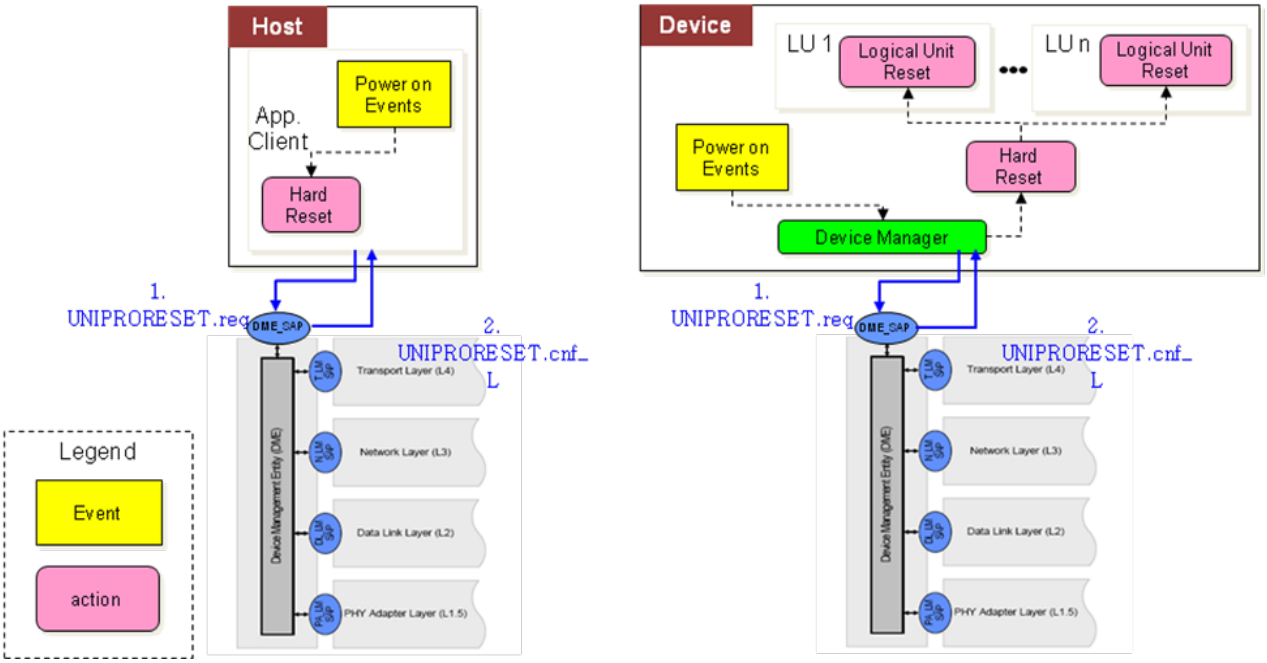

Figure 4-2 — HW Reset

Table 4-1 — Reset timing parameters

Symbol	Comment	Min	Max	Unit
tRSTW	RST_n Pulse Width	1		µs
tRSTH	RST_n High Period (Interval)	1		µs
tRSTF[1]	RST_n filter	100		ns
NOTE The UFS Device must not detect 100ns or less of positive or negative RST_n pulse. The UFS Device must detect more than or equal to 1us of positive or negative RST_n pulse width.				

4.1.3 EndPointReset

The EndPointReset feature is defined in the MIPI UniPro Specification.

Function Call from Host Application Client to Host UniPro via UIO_SAP: ENDPOINTRESET.REQ = 1

Device Manager shall receive the EndPointReset function call from the Device UniPro via UIO_SAP and executes the EndPointReset function.

A UFS Device shall completely reset itself on reception of an EndPointReset. A UFS Host shall ignore the reception of an EndPointReset from a UFS device.

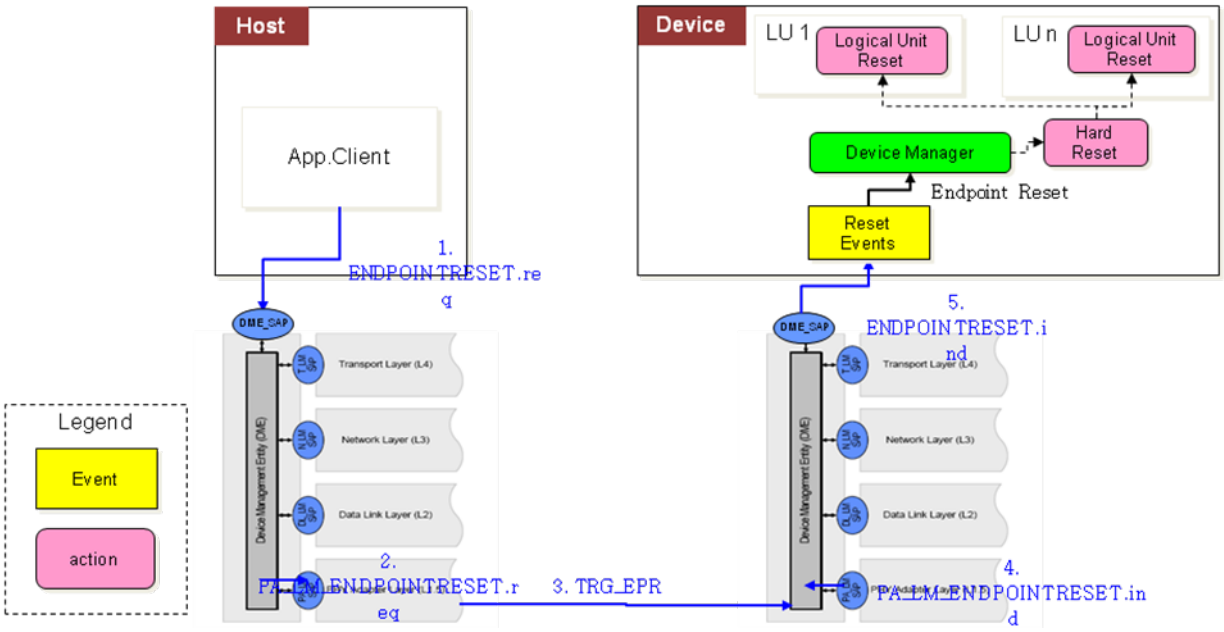

Figure 4-3 — EndPointReset

4.2 Logical Unit Reset

The Logical Unit Reset feature is defined in the SCSI Architectural Model Specification [SAM]. This Reset is triggered via the SCSI Task Management features described in 7.8.4.7.

LU reset (LU 0-7):

Function Call from Host Application Client to Host UTP via UTP_TM_SAP: Task management LOGICAL UNIT RESET (IN (I_T_L Nexus)).

LU Task manager shall receive the function call from Device UTP via UTP_TM_SAP and executes the LU reset function.

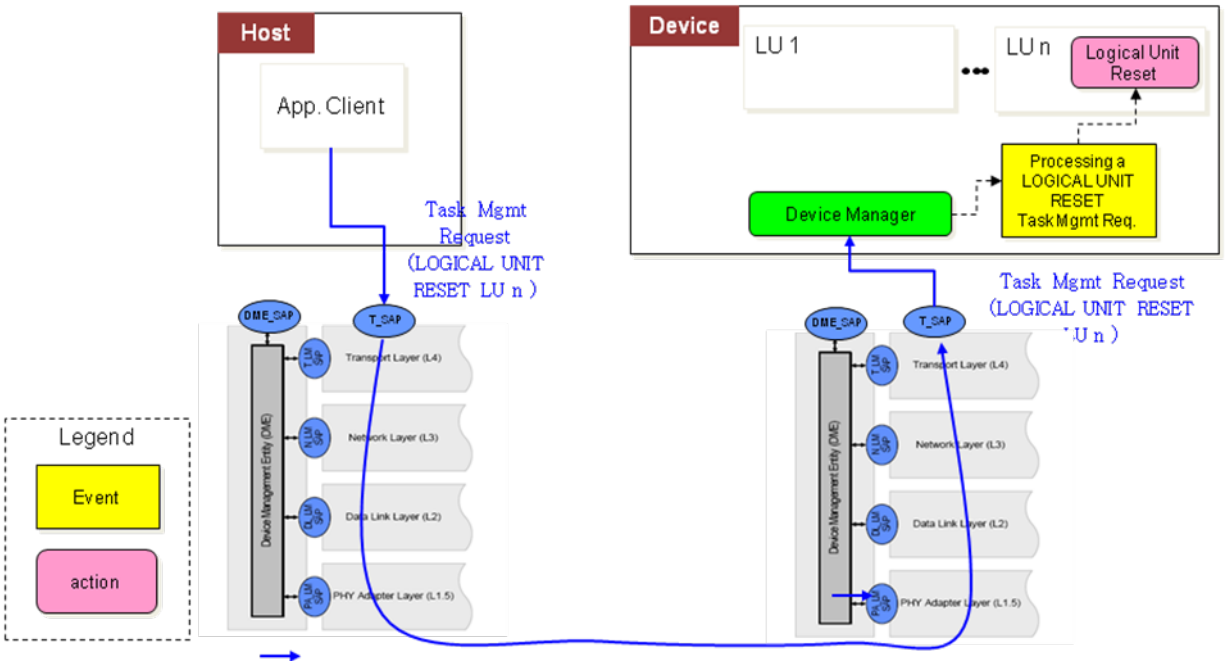

Figure 4-4 — Logical Unit Reset

4.3 Other Resets

In addition to the direct ways to initiate UFS Device Reset there are some indirect methods existing also.

Host System resets its own UniPro stack: UniPro Stack reset activity in host system side also UniPro Stack reset in the UFS Device side via the DME_LINKLOST.ind message. In case UFS Device will receive such DME_LINKLOST.ind message from host system it shall start process of re-initializing its own UniPro stack. In addition also all UFS Device level activity has to stop and different blocks shall be reset.

4.4 Summary of Resets and Devices Behavior

Table 4-2 and Table 4-3 summarize the different types of Resets and the UFS device behavior related to them.

Table 4-2 — Reset States

Reset Type	Initiator	Initial State	Target State	Boot
Power-on	Host	All	PRE-Active	Enabled
HW Pin Reset	Host	All	PRE-Active	Enabled
EndPointReset	Host, Device Manager	All	PRE-Active	Enabled
LU Reset	Host, UFS Device Driver	Active	Active	Disabled
Host System UniPro Reset	Host	All	PRE-Active	Enabled

Table 4-3 — UniPro Attributes and Description Reset

Reset Type	UniPro Stack and Attributes	Programmable Device Descriptors	Programmable LU Descriptors	Queue (LU)	Power-on protected Descriptors
Power-on	Reset	Reset	Reset	Reset (all)	Reset
HW Pin Reset	Reset	Reset	Reset	Reset (all)	Reset
EndPointReset	Reset	Reset	Reset	Reset (all)	Not affected
LU Reset	Not affected	Not affected	Reset	Reset (LU)	Not affected
Host System UniPro Reset	Reset	Reset	Reset	Reset (all)	No affected

4.5 UFS Power Modes

The device supports multiple power modes, which are controlled by the START_STOP_UNIT command and Power Mode Control Register. These are independent of the bus state of the upstream or downstream links, which are controlled independently.

In order to minimize power consumption in a variety of operating environments, UFS devices will support four basic power modes. One where the device is working, one where it is awaiting the next instruction, one where it has been put to sleep until the host wants it, and a final mode where it can be turned off completely. These four modes should cover the need for the host to control the power consumed by the device, while still maintaining appropriate responsiveness from the device. There are also three transitional modes needed to facilitate the change from one mode to the next.

While in Active Mode processing instructions, there are several possible power scenarios. UFS devices can be expected to be battery powered. However, they may be plugged directly into a power source to recharge those batteries. During those times, a larger current may be available, and large amounts of data may be processed at the same time. There is also the possibility that the device is attached to a mobile device with a failing battery, in which case minimal power consumption is a requirement. Finally, there is the possibility that the host would know nothing of the device with which it is paired, and the device would need to be configured to operate within the host's current requirements.

In order to support these varied scenarios, UFS can support up to 16 Active configurations, each with its own current profile. The host can choose from either pre-defined or user-defined current profiles to deliver the highest performance possible. The details of the system are described in Active Mode.

4.5.1 Active Mode

In the active current mode, the device is responding to a command or performing a background operation. The host link is in either STALL or HS-BURST mode (if in high-speed mode), or SLEEP or PWM-BURST (if in low-speed mode).

The maximum power consumption in Active is determined by the SELECT_ACTIVE_CONSUMPTION_LEVEL attribute. This 4-bit field selects from among the 16 different current consumption modes described by the ACTIVE_CONSUMPTION_VCC, ACTIVE_CONSUMPTION_VCCQ, and ACTIVE_CONSUMPTION_VCCQ2 parameters. For example, when the SELECT_ACTIVE_CONSUMPTION attribute is set to N, the maximum current consumed on VCC would be specified by ACTIVE_CONSUMPTION_VCC[N], the maximum current consumed on VCCQ would be specified by ACTIVE_CONSUMPTION_VCCQ[N], and the maximum current consumed on VCCQ2 would be specified by ACTIVE_CONSUMPTION_VCCQ2[N]. The assumption is that the ACTIVE_CONSUMPTION_LEVELs are ordered in terms of performance: that is, that level0 is lower performance than level1, which is lower than level2, and so on until levelF which would be the highest performance. The host can then read in the ACTIVE_CONSUMPTION_VCCx parameters, and choose the highest performance level which fits within its current limitations on each supply.

The default value of the SELECT_ACTIVE_CONSUMPTION_LEVEL is 0x0, which always has a maximum combined current of 50 mA on all power supplies. Embedded devices should primarily use settings of 0x6 and 0xC, for normal (battery) and high (plugged in) power operating modes. The current consumption for those two modes, and the Sleep and PowerDown mode should be available in the specification. Removable devices shall (and embedded devices may) support any of the other setting values.

4.5.1 Active Mode (cont'd)

Active Mode can be entered from Pre-Active mode after the completion of all setup necessary to handle commands. Bootable devices initialize to Active Mode, as do non-removable devices if the POR_DEVICE_STATE bits has been set.

The following power mode may be Idle (after completion of a command and/or background operations), or Pre-Sleep (by START_STOP_UNIT with PowerCondition = 2), or Pre-PowerDown (by START_STOP_UNIT with PowerCondition = 3). The details of the START_STOP_UNIT command are in Power Management Command: START_STOP_UNIT.

All supported commands are available in Active Mode.

4.5.2 Idle

The Idle power mode is reached when the device is not processing any instruction. The host link is in either STALL or SLEEP mode. If background operations are continuing, the device should be considered Active.

This mode can only be entered from an Active Mode, and the following state is always an Active Mode. The receipt of any command will transition the device into Active Mode.

4.5.3 Pre-Active

The Pre-Active Mode is a transitional mode associated with Active Mode. The power consumed will be no more than that consumed in Active mode. The device will remain in this mode until all of the preparation needed to accept commands has been completed.

Pre-Active Mode can be entered from Pre-Sleep, Sleep, Pre-PowerDown, or PowerDown (START_STOP_UNIT with POWER CONDITION = 1). The following mode is always Active.

The REQUEST_SENSE will be responded to with the SENSE_KEY set to NO_SENSE, and the additional sense code set to LOGICAL_TRANSITIONING_TO_ANOTHER_POWER_CONDITION. The START_STOP_UNIT with POWER_CONDITION !=1 or any other commands will be terminated with a "CHECK_CONDITION" status, with the sense key set to NOT_READY, with the additional sense code set to LOGICAL_UNIT_IN PROCESS OF BECOMING_READY.

4.5.4 UFS-Sleep

The UFS-Sleep Mode allows for deep-power down of the device.

VCC can be deasserted in this state.

The UFS-Sleep mode can be entered from Pre-Sleep. Removable devices always initialize into UFS-Sleep Mode. Embedded devices may initialize into UFS-Sleep, if the POR_DEVICE_STATE field has not been set.

In this mode, no operations can be performed except REQUEST_SENSE or START_STOP_UNIT. The REQUEST_SENSE command will be responded to with the SENSE_KEY set to NO_SENSE, and the additional sense code set to LOGICAL_UNIT_NOT_READY, INITIALIZING COMMAND REQUIRED. Other commands will be terminated with a CHECK_CONDITION status, with the sense key set to ILLEGAL_COMMAND. The host link should be in to HIBERN8 state, although they are actually under host control and can come up and down independently of the UFS-Sleep Mode.

4.5.5 Pre-Sleep

The Pre-Sleep Mode is a transitional mode associated with UFS-Sleep entry. The power consumed will be no more than that consumed in Active mode. The device will automatically advance to Sleep Mode once any outstanding operations and management activities have been completed.

Pre-Sleep can be entered from Active (START_STOP_UNIT with POWER CONDITION = 2). The following mode is Sleep or Pre-Active (START_STOP_UNIT with POWER CONDITION = 1).

The REQUEST_SENSE command will be responded to with the SENSE_KEY set to NO_SENSE, and the additional sense code set to LOGICAL_UNIT_TRANSITIONING_TO_ANOTHER_POWER_CONDITION. Queue management functions will operate normally. Other incoming commands will be terminated with a "CHECK_CONDITION" status, with the sense key set to ILLEGAL_COMMAND.

4.5.6 UFS-PowerDown

The UFS-PowerDown mode is the maximum power saving mode. All non-volatile information may be lost, and either power supply can be deasserted. The host link can be in either HIBERN8 or DISABLE state.

This mode can be entered from the Pre-PowerDown Mode, at the completion of the mode transition.

In this mode, no operations can be performed except REQUEST_SENSE or START_STOP_UNIT. The REQUEST_SENSE command will be responded to with the SENSE_KEY set to NO_SENSE, and the additional sense code set to LOGICAL_UNIT_NOT_READY, INITIALIZING COMMAND REQUIRED. Other commands will be terminated with a "CHECK_CONDITION" status, with the sense key set to ILLEGAL_COMMAND.

After receipt of an update from the PHY and START_STOP_UNIT command, the device can switch to the Pre-Active Mode.

4.5.7 Pre-PowerDown

The Pre-PowerDown mode is a transitional mode associated with UFS-PowerDown entry. The power consumed will be no more than that consumed in Active mode. The device will automatically advance to PowerDown Mode once any outstanding operations and management activities have been completed.

During Pre-PowerDown, any dynamic data in the system should be stored to memory, since power can be deasserted at any point in PowerDown Mode.

Pre-PowerDown can be entered from Active or Sleep (START_STOP_UNIT with POWER CONDITION = 2). The following mode is PowerDown or Pre-Active (START_STOP_UNIT with POWER CONDITION = 1).

The REQUEST_SENSE command will be responded to with the SENSE_KEY set to NO_SENSE, and the additional sense code set to LOGICAL_UNIT_TRANSITIONING_TO_ANOTHER_POWER_CONDITION. Queue management functions will operate normally. Other incoming commands will be terminated with a "CHECK_CONDITION" status, with the sense key set to ILLEGAL_COMMAND.

4.5.8 Power State Machine

The relationship amongst the different power modes is shown in the following diagram.

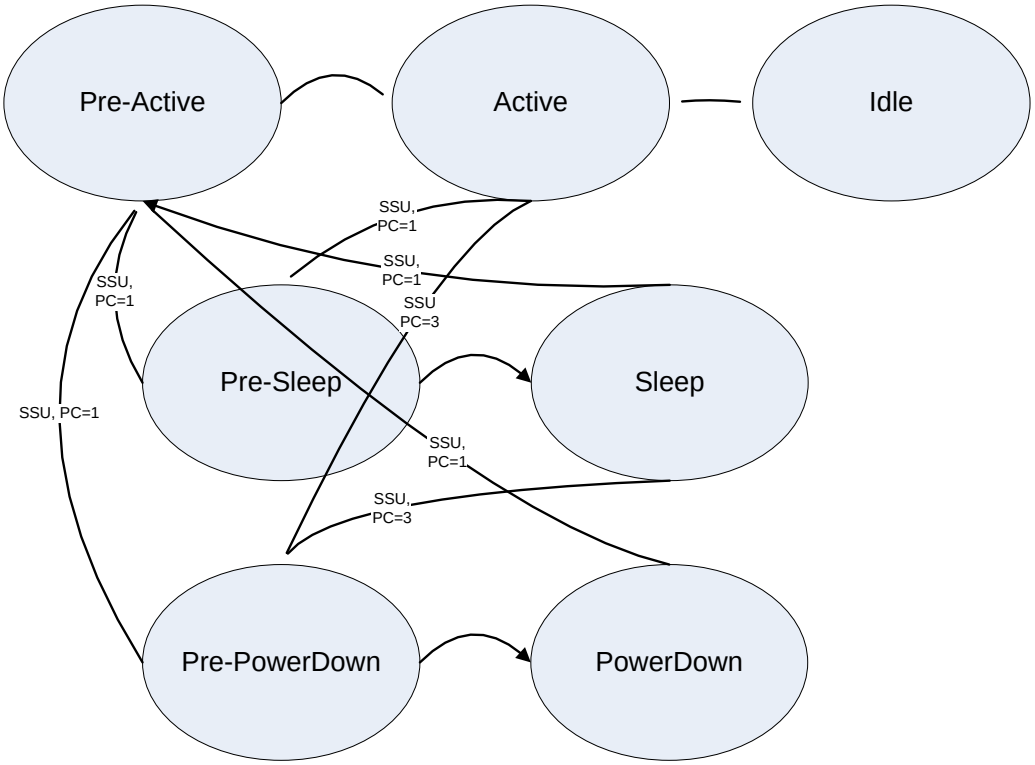

Figure 4-5 — Power Modes

The current mode can be read from the DEVICE_POWER_STATE parameter.

By setting the IMMED field during the SSU command, the device can be instructed to respond at the entrance to the transitional mode; once the command is received. If the IMMED field is not set, the response will not be sent until the change out of the transitional mode. If the IMMED field is not set, and the device changes to Active, Sleep, or PowerDown modes, a NO_SENSE will be indicated. If the IMMED field is not set and the devices leaves the transitional mode to go to another transitional mode (Pre-Sleep to Pre-Active, or Pre-PowerDown to Pre-Active), the response will be NOT_READY, with the additional key of START_STOP_UNIT_COMMAND_IN_PROGRESS.

4.5.9 Power Management Command: START_STOP_UNIT

When the START_STOP_UNIT command is sent to an LU, it can be used to enable/disable that LU, flush any outstanding operations, or load/eject medium. When the START_STOP_UNIT command is sent to the well-known LU, it can be used to select the device power mode.

Table 4-4 — START STOP UNIT command

Bit Byte	7	6	5	4	3	2	1	0
0	Operation Code (1Bh)							
1	Reserved (0000000b)							IMMED
2	Reserved (00h)							
3	Reserved (0000b)				Power Condition Modifier (Rsvd)			
4	Power Condition (0000b)				Rsvd	No Flush	LOEJ (Rsvd)	Start
5	Control (00h)							

NOTE The Power_CONDITION field selects the desired mode. If the command is sent to a particular LU, the Power Condition field should be ignored.

Table 4-5 — Power Condition field

IMMED	No Flush	Power Condition	Start	LU	Action
0	-	-	-	-	Response is sent after change is complete.
1	-	-	-	-	Response is sent immediately after command decode
-	0	-	-	-	Dynamic data should be flushed to non-volatile storage
-	1	-	-	-	No requirements regarding dynamic data
-	-	0x0	0	Particular	Stop the designated LU.
-	-	0x0	1	Particular	Start the designated LU.
-	-	0x1		WKLU	Cause a transition to the Active Power Mode
-	-	0x2		WKLU	Cause a transition to the UFS_Sleep Power Mode
-	-	0x3		WKLU	Cause a transition to the UFS_PowerDown Mode

NOTE WKLU refers to the well-known logic unit number for UFS START_STOP_UNIT: 0x50.

4.6 Power Mode Control

A series of parameter and configuration values are used to control the active power modes.

Table 4-6 — Power Mode Control parameters and configurations

Parameter Name	Size	Type	Description
DEVICE_POWER_STATE	1 Byte	RO	Current device power mode.
bActiveConsumptionVCC 15:0]	2 Byte	RO	Sixteen descriptors describing the maximum current consumed from VCC in each of the 16 active consumption modes.
bActiveConsumptionVCCQ 15:0]	2 Bytes	RO	Sixteen descriptors describing the maximum current consumed from VCCQ in each of the 16 active consumption mode.
bActiveConsumptionVCCQ2 [15:0]	2 Bytes	RO	Sixteen descriptors describing the maximum current consumed from VCCQ2 in each of the 16 active consumption modes.
Attribute Name	**Size**	**Type**	**Description**
SELECT_ACTIVE_CONSUMPTION_MODE	4b	R/W	If set, the device is allowed to use the larger power budgets associated with the Active power mode. Default=0.
bPORDeviceState	1b	R/W	At next boot, if set the device initializes to Active Mode, otherwise to UFS-Sleep Mode. Default=1.
Each of the 48 ACTIVE_CONSUMPTION parameters is formatted thusly: 15:14 Units – 0x0: nA; 0x1: uA; 0x2: mA; 0x3: A 9:0 Value – The maximum current expected in each condition.			

5 UFS PHY – MIPI M-PHY

5.1 Termination

The M-TX shall be terminated as defined in the M-PHY Specification.

The M-RX shall include switchable differential termination. By default the M-RX termination shall be off in PWM-BURST state and may be turned on by the protocol. The termination shall be on by default in HS-BURST state and may be turned off. There shall be no termination in SLEEP and STALL states. During DISABLE and HIBERNATE states M-TX drives High-Z while M-RX drives the lane by a 'Dif-Z keeper'. Dif-Z keeper means M-RX drives a weak differential zero on the lane.

M-RX of a SUBLINK in a LINK may have different termination settings.

The supported termination settings are defined in the Capability Attributes in Table 5-1 and Table 5-2. The termination is controlled via the Configuration Attributes. The receiver termination resistance is defined in the M-PHY Specification.

Timings of the termination enable and disable are defined in the M-PHY Specification.

5.2 Drive Levels

The M-PHY Specification defines two different drive amplitudes: the large amplitude (LA) and the small amplitude (SA). The UFS IF utilizes both of these drive amplitudes as defined in the M-PHY Specification.

Every M-TX in every LINK will start communication with LA after power up or reset. Option to switch to SA mode shall be supported via a Configuration Attribute.

SUBLINKS in a LINK shall communicate with same amplitude

5.3 PHY State machine

The UFS IF shall implement the Type I state machine.

The M-PHY Specification defines two different signaling schemes for low-speed (LS) mode: Non-Return-to-Zero (NRZ) and Pulse-Width-Modulation (PWM). The UFS IF shall utilize the PWM signaling scheme in the LS mode as defined by the M-PHY Specification for the State Machine Type I.

The UFS IF shall implement both Mode-LLC and Configuration-LCC line control commands as defined in the M-PHY Specification for the state machine Type I.

5.4 HS Burst

A UFS device shall support the HS-G1 (A/B) GEAR. Support for the higher GEARS, HS-G2(A/B) and HS-G3(A/B), is optional. The supported GEAR(s) is (are) indicated in the Capability Attribute Table 5-1 and Table 5-2. The lower HS GEAR(s) shall be supported also by a higher GEAR device.

The HS-1A GEAR shall be the active one by default after power up or reset.

SUBLINKS in a LINK may communicate with different HS-GEAR or PWM-GEAR.

5.4.1 HS Prepare Length Control

The HS PREPARE_LENGTH defines the time to move from STALL to HS-BURST. At reset M-TX settles the HS PREPARE_LENGTH = 15.

A UFS M-RX shall support HS PREPARE_LENGTH ≤ 15.

5.4.2 HS Sync Length Control

The HS_SYNC_LENGTH defines the number of synchronization words before a HS Burst. In UFS Device the SYNC sequence shall be generated by the M-TX. Support for protocol controlled synchronization is optional. M-TX starts at reset with HS_SYNC_LENGTH = 15, in COARSE type.

A UFS device M-RX shall support RX_HS_SYNC_LENGTH_Capability with SYNC_range = COARSE, SYNC_LENGTH ≤8. For a low-power M-PHY implementation it is recommended to have a RX_HS_SYNC_LENGTH_Capability with SYNC_range = COARSE, SYNC_LENGTH ≥ 4 .

5.4.3 Slew Rate Control

A UFS Device may include configurable Slew Rate control as per the M-PHY Specification. If the Slew Rate Control is supported by a UFS Device then the number of N (steps) shall be $4 \leq N \leq 8$.

5.5 PWM Burst

A UFS device shall support the PWM-G1 (default, mandated by the M-PHY Specification), PWM-G2, PWM-G3 and PWM-G4 GEARS. PWM-G0 shall not be supported. The PWM-G5, PWM-G6 and PWM-G7 are optional. The supported PWM-GEARS are indicated in the Capability Attributes Table 5-1 and Table 5-2.

The PWM-G1 shall be the active one by default after power up or reset.

SUBLINKS in a LINK may communicate with different PWM-GEAR or HS-GEAR.

5.5.1 LS Prepare Length Control

The LS PREPARE_LENGTH defines the time to move from SLEEP to PWM-BURST. At reset M-TX settles the LS PREPARE_LENGTH = 15.

A UFS device M-RX shall support LS PREPARE_LENGTH ≤ 15.

5.6 UFS PHY Attributes

The MIPI M-PHY includes several configurable attributes. There is range of values defined for the attributes but it is left for the application to fix the actual required values inside the range. Following is the list of such attributes. The UFS application specific requirement for the values can be found from Table 5-1 and Table 5-2.

5.6 UFS PHY Attributes (cont'd)

Table 5-1 — UFS PHY M-TX Capability Attributes

Attribute Name	Attribute ID	Range	UFS Value	Notes
TX_HSMODE_Capability	0x01	No=0, Yes=1	1	
TX_HSGEAR_Capability	0x02	1=G1, 2=G1+G2, 3=G1+G2+G3	1-3	Value 1 is mandatory, Values 2 and 3 are optional
TX_PWMG0_Capability	0x03	0=No, 1=Yes	0	A UFS device shall not support PWMG0
TX_PWMGEAR_Capability	0x04	1-7	1-7	A UFS device shall support PWM-GEARs 1-4. PWM-GEAR 5-7 are optional.
TX_Amplitude_Capability	0x05	SA=1, LA=2, Both=3	3	UFS IF device shall support both drive levels
TX_ExternalSYNC_Capability	0x06	0=False 1=True	0-1	Value 0 mandatory, Value 1 optional.
TX_HS_Unterminated_LINE_Drive_Capability	0x07	0=No, 1=Yes	1	UFS device shall support un-terminated HS burst
TX_LS_Terminated_LINE_Drive_Capability	0x08	0=No, 1=Yes	1	UFS device shall support terminated LS burst
TX_Min_SLEEP_NoConfig_Time_Capability	0x09	1-15	1-15	
TX_Min_STALL_NoConfig_Time_Capability	0x0A	1-15	1-15	
TX_Min_SAVE_- Config_Time_Capability	0x0B	1-250	1-250	0- 10000ns (40ns steps)
TX_REF_CLOCK_SHARED_Capability	0x0C	0=No, 1=Yes	1	
TX_PHY_MajorMinor_Release_Capability	0x0D	B[7:4]		Major version number, 0 to 9
		B[3:0]		Minor version number, 0 to 9
TX_PHY_Editorial_Release_Capability	0x0E	B[7:0]		1 to 99

5.6 UFS PHY Attributes (cont'd)

Table 5-2 — UFS PHY M-RX Capability Attributes

Attribute Name	Attribute ID	Range	UFS Value	Notes
RX_HSMODE_Capability	0x81	No=0, Yes=1	1	
RX_HSGEAR_Capability	0x82	1=G1, 2=G1+G2, 3=G1+G2+G3	1-3	Value 1 is mandatory, Values 2 and 3 are optional
RX_PWMG0_Capability	0x83	0=No, 1=Yes	0	A UFS device shall not support PWMG0
RX_PWMGEAR_Capability	0x84	1-7	1-7	A UFS device shall support PWM-GEARs 1-4. PWM-GEARs 5-7 are optional.
RX_HS_Unterminated_Capability	0x85	0=No, 1=Yes	1	UFS device shall support un-terminated HS burst
RX_LS_Terminated_Capability	0x86	0=No, 1=Yes	1	UFS device shall support terminated LS burst
RX_Min_SLEEP_NoConfig_Time_Capability	0x87	1-15	1-15	
RX_Min_STALL_NoConfig_Time_Capability	0x88	1-15	1-15	
RX_Min_SAVE_Config_Time_Capability	0x89	1-250	1-250	0- 10000ns (40ns steps)
RX_REF_CLOCK_SHARED_Capability	0x8A	0=No, 1=Yes	1	
RX_HS_SYNC_LENGTH	0x8B	b[7:6]=0-1 b[5:0]=0-15	b[7:6]=0-1 b[5:0]=0-8	Mandatory values are: b[7:6]=1 (COARSE) b[5:0]=0-8 Recommended values are b[7:6]=1 (COARSE) b[5:0]=0
RX_HS_PREPARE_LENGTH	0x8C	0-15	0-15	Value 15 mandatory, Values 0- 14 optional.
RX_LS_PREPARE_LENGTH	0x8D	0-15	0-15	Value 15 mandatory, Values 0-14 optional.
RX_PWM_Burst_Closure_Length_Capability	0x8E	0-255	0-255	
RX_Min_ActivateTime_Capability	0x0F	1-9	1-9	Value 9 is mandatory, Values 1-8 are optional, defined in steps of 100us
RX_PHY_MajorMinor_Release_Capability	0x0D	0-9		Major version number, 0 to 9
		B[3:0]		Minor version number, 0 to 9
RX_PHY_Editorial_Release_Capability	0x0E	B[7:0]		1 to 99

5.7 Operation Timings

5.7.1 Reference Clock Timings

The reference clock shall be turned ON before initiation of the state transition to STALL. The clock may be turned OFF again after both links has reached either HIBERNATE or SLEEP states.

5.8 Electrical characteristics

5.8.1 Transmitter Characteristics

As defined in the M-PHY Specification.

5.8.2 Receiver Characteristics

As defined in the M-PHY Specification.

6 UFS INTERCONNECT LAYER

6.1 Overview

UFS builds on the MIPI Unified Protocol (UniPro) as its Interconnect (Service Delivery Subsystem) to provide basic transfer capabilities to the UFS Transport Protocol (UTP) Layer. On the data plane UTP and UniPro communicate via the Service Primitives of the UniPro Transport Layer CPorts (T_CO_SAPs). Control plane interaction (e.g., discovery, enumeration and configuration of the Link) between higher layer protocol functions of UFS and UniPro are accomplished using the Device Management Entity Service Primitives as defined by the UniPro specification.

6.2 Architectural Model

UniPro is internally composed of several sub-layers which are all well defined by the MIPI UniPro specification [MIPI-UniPro]. In the context of UFS the entire UniPro protocol stack shall be viewed as a black box model (see Figure below) to the greatest extent possible. The following sections therefore only:

- Specify number and type of the required interfaces between UFS and UniPro
- Specify the mapping between UFS and UniPro addressing scheme
- Select optional features and definable attributes of the UniPro specification

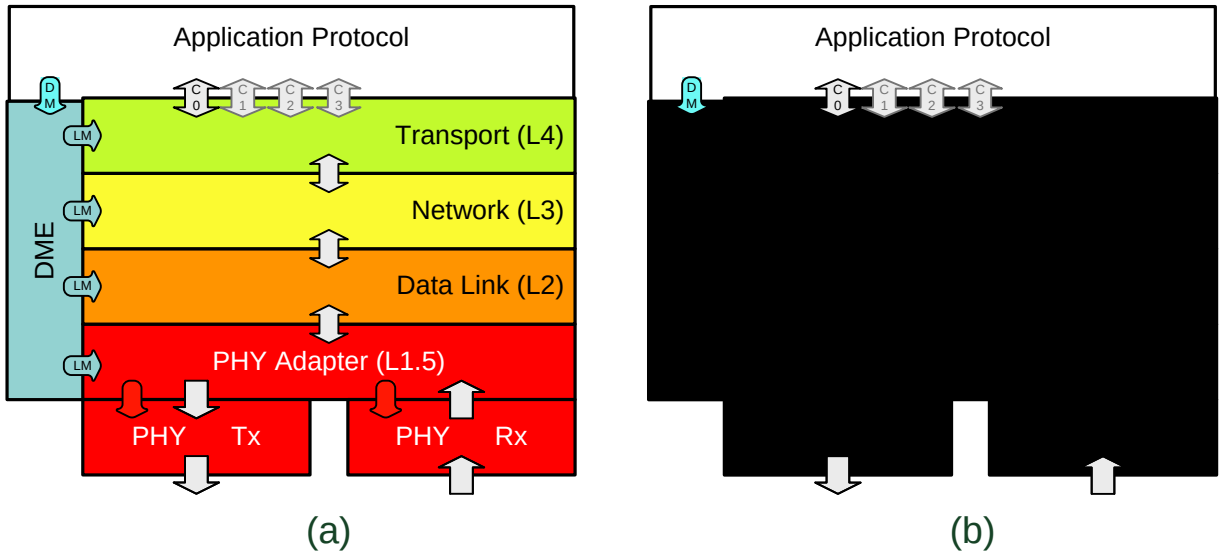

Figure 6-1 — UniPro internal layering view (left) and UniPro Black Box view (right)

6.3 UniPro/UFS Transport Protocol Interface (Data Plane)

UniPro provides CPorts as conceptual interfaces to applications or protocol layers on top of UniPro. CPorts can be viewed as instantiations of the T_CO_SAPs as specified in section 8.8 of the UniPro specification. The physical implementation of a T_CO_SAP was deliberately not defined in MIPI as implementers should be free to chose, e.g. a SW implementations of higher UniPro layers, HW implementations based on buffering per CPort or DMA channels per CPort, etc.

A Service Access Point (SAP) provides Service Primitives (SP) which can be used by specifications of applications or protocols as UFS on top of UniPro to define their interactions. For more information on the concept of SAP/SP in protocol specifications please refer to Annex C of the UniPro specification.

The T_CO_SAP provides the following core data transfer service primitives (see UniPro specification, 8.8.1):

- T_CO_DATA.req(MessageFragment, EOM)
  - Issued by service user of UniPro to send a message (Fragment) transfer

    NOTE Whenever a UFS layer requests the UIC layer to transfer data that UFS layer shall ensure that the last fragment of said data will be transmitted with the EOM flag set. One way to ensure such a behavior is for the UFS layer to invoke this UIC data transfer service primitive only once per atomic protocol data unit (e.g., once per UFS Transport Layer ‘UPIU’) with the EOM flag set to ‘true’ always.
- T_CO_DATA.cnf_L(L4CPortResultCode)
  - Issued by UniPro to report the result of a Message (Fragment) transfer request
- T_CO_DATA.ind(MessageFragment, EOM, SOM, MsgStatus)
  - Issued by UniPro to deliver a received Message (Fragment) towards the service user
  - EOM informs the service user that this is the last Message Fragment (EndOfMessage)
  - SOM informs the service user that this is the first Message Fragment (StartOfMessage)
- T_CO_DATA.rsp_L( )
  - Issued by a service user of UniPro to report readiness to receive the next Message (Fragment)

Flow control

UFS will not make use of the End-to-End Flow Control feature of UniPro for data communication as the UFS Transport Layer already avoids any overflow by a strict Host/Client communication model, tagged command queues and Device side throttling of Data transfers.

Object sizes

A UniPro Message can be of any size and its content is not interpreted in any way by UniPro. Messages can be delivered from/to UniPro as multiple Message Fragments.

A Message Fragment is a portion of a Message that can be passed to, or received by, a CPort. Received Fragments are not generally identical to transmitted Fragments. Message Fragments may or may not carry an End-of-Message (EoM) flag.

6.4 UniPro/UFS Control Interface (Control Plane)

UniPro provides access to its Device Management Entity (DME) via a Service Access Point (DME SAP) with the following services exposed to UFS allowing control of properties and the behavior of UniPro:

DME Configuration Primitives

- DME_GET / DME_SET
  - Provide read/write access to all UniPro and M-PHY attributes of the local UniPort
- DME_PEER_GET (optional) / DME_PEER_SET (optional)
  - Provide read/write access to all UniPro and M-PHY attributes of the remote UniPort

NOTE The order in which attributes are set is in some cases relevant for UniPro's correct operation. Therefore higher UFS layers shall preserve the ordering of DME Configuration Primitives invocations by UFS applications. If internally generated by UFS itself, DME Configuration Primitives shall be issued correctly ordered as defined by the UniPro specification.

DME Control Primitives

- DME_POWERON (optional) / DME_POWEROFF (optional)
  - Allow to power up or power down all UniPro layers (L1.5 through L4)
- DME_ENABLE
  - Allow enabling of the entire local UniPort (UniPro L1.5 -L4 and MPHY)
- DME_RESET
  - Allows to reset the entire local UniPort (UniPro L1.5-L4 and MPHY)
- DME_ENDPOINTRESET
  - Allows to send an end-point reset request command to a remote UniPort
- DME_LINKSTARTUP
  - Allows locally to startup the Link and informs about remote link startup invocation
- DME_HIBERNATE_ENTER / DME_HIBERNATE_EXIT
  - Allow to put the entire Link into HIBERNATE power mode and to wake the Link up
    - Affects the local and the remote UniPort (UniPro L1.5-L4 and MPHY)

  NOTE After exit from Hibernate all UniPro Transport Layer attributes (including L4 T_PeerDeviceID, L4 T_PeerCPortID, L4 T_ConnectionState, etc.) will be reset to their reset default values. All required attributes must be restored properly on both ends before communication can resume.
- DME_POWERMODE
  - Allows to change the power mode of one or both directions of the MPHY Link
- DME_TEST_MODE (optional)
  - Allows to set the remote UniPro Device on the Link in a specific test mode
- DME_LINKLOST
  - Indication of the UniPro stack towards higher layers that the Link has been lost
- DME_ERROR
  - Indication of the UniPro stack towards higher layers that an error condition has been encountered in one of the UniPro Layers

6.5 UniPro/UFS Transport Protocol Address Mapping

UniPro has fundamentally two levels of addressing to control the exchange of information between remote UniPro entities

Network Layer (L3): Device ID, lowest level of addressability

- Provided for future UniPro networks of devices. During connection establishment the side creating a connection uses this value to select the physical entity on the remote end of the connection. It shall be considered static for the lifetime of this connection.

Transport Layer (L4): CPort ID, highest level of end-to-end addressability

- During connection establishment the side creating a connection uses this value to select the logical entity inside the targeted UniPro device on the remote end of the connection. It shall be considered static for the lifetime of this connection.

UFS will adopt the addressing notation of the SCSI Architecture Model (SAM-5) [SAM] which defines an ‘Address Nexus’ (I_T_L_Q) which is composed of

- Initiator Port Identifier (I)
- Target Port Identifier (T)
- Logical Unit Number (L)
- Command Identifier (Q).

An I_T_L_Q Nexus uniquely defines a specific command slot (Q) inside a specific Logical Unit (L) connected to a specific Device Target Port (T) accessed through a specific Host Initiator Port (I).

UFS Interconnect Layer (UniPro) addresses are only related to the I_T part of the Nexus:

- UFS 1.0 systems shall only support a single UFS Host containing a single UFS Initiator Port
  - Hence, UFS 1.0 only requires and uses a single UniPro CPort on the Host side
- UFS 1.0 systems shall only support a single UFS Device containing a single UFS Target Port
  - Hence, UFS 1.0 only requires and uses a single UniPro CPort on the Device side

Mapping Rules

- UFS Port IDs (I and T) shall be 12 bit wide each
  - The most significant 7 bits of each UFS Port ID shall contain the UniPro Network Layer Device ID of the entity (Host or Device) containing said UFS Port
    - The UniPro Network Layer Device ID reset value shall be 0 for the Host
    - The UniPro Network Layer Device ID reset value shall be 1 for the Device
  - The least significant 5 bits of each UFS Port ID shall contain the UniPro Transport Layer CPort ID which said UFS Port uses to communicate to the remote entity
    - The UniPro Transport Layer CPort ID reset value shall be 0 for the Host
      - The UniPro Transport Layer CPort ID reset value shall be 0 for the Device

6.5 UniPro/UFS Transport Protocol Address Mapping (cont'd)

As a result of the aforementioned mapping after reset the I_T Nexus which UFS uses defaults to:

- UFS 1.0 I_T Nexus = [0b0000000|0b00000|0b0000001|0b00000] = 0x20 = 0d32

The single UniPro connection between the UTP layer of an UFS 1.0 Host and the UTP layer of an UFS 1.0 Device can be uniquely identified by the UFS I_T Nexus above.

NOTE

- The I_T Nexus elements (Device IDs and CPort IDs) can be modified by the Host after reset using the DME Service Primitives:
    - The "I" Nexus element may be modified by the Host using the DME_SET primitive
    - The "T" Nexus element may be modified by the Host using DME_PEER_SET primitive

All attributes of the CPort on the Host side (including, e.g. "T_ConnectionState") can be checked and modified by the Host using the DME_GET and DME_SET primitives after reset.

All attributes of the CPort on the Device side (including, e.g. the "T_ConnectionState") can be checked and modified by the Host using the DME_PEER_GET and DME_PEER_SET primitives after reset.

6.6 Options and Tunable Parameters of UniPro

MIPI UniPro has been designed as a versatile protocol specification and as such has several options and parameters which an application like UFS should specify for its specialized UniPro usage scenario. Annex G of the UniPro specification details all of the possible choices.

The remaining sections of this chapter define the specific requirements towards these options and parameters for UFS 1.0. They apply to UniPro implementations for the UFS Host side as well as to UniPro implementations for the UFS Device side if not explicitly stated otherwise below.

6.6.1 UniPro PHY Adapter

- For MIPI M-PHY related attribute values and implementation options as defined by UFS please refer to Chapter 5: UFS PHY – MIPI M-PHY on page 34.

6.6.2 UniPro Data Link Layer

- Shall implement the Data Link Layer Traffic Class "Best Effort" (TC 0)
- Data Link Layer Traffic Class 1 (TC1: 'Low Latency') is not required
- TX preemption capability is not required
- Shall provide at least DL_MTU bytes of Data Link Layer RX and TX buffering
- Shall support transmission and reception of maximum sized L2 frames (DL_MTU)

6.6.3 UniPro Network Layer

- Shall support transmission and reception of maximum sized L3 packets (N_MTU)

6.6.4 UniPro Transport Layer

- UFS Hosts and Devices shall implement at least 1 CPort
  - Note: UFS 1.0 only requires and uses a single CPort on either side of the Link.
- UFS does not mandate any CPort arbitration scheme beyond the UniPro default if more than one CPort is implemented
- Shall support the UniPro Test Feature
- UFS does not require the UniPro End-to-End Flow Control mechanism
  - UFS will not use ‘Controlled Segment Dropping’ (CSD)
    - Hence CSD shall be disabled
- Shall support transmission and reception of maximum sized L4 segments (T_MTU).

6.6.5 UniPro Device Management Entity Transport Layer

Table 6-1 details which DME Service Primitives are defined as mandatory or optional by UniPro. Additionally, it details which DME Service Primitives are required to be implemented by UFS Hosts and by UFS Devices.

Table 6-1 — DME Service Primitives

DME Service Primitive	UniPro	UFS Host	UFS Device	UFS Usage Limitations
DME_GET	Mandatory	Implement	Implement	
DME_SET	Mandatory	Implement	Implement	UFS Device shall not use this primitive to modify the local PA_PWRMode attribute
DME_PEER_GET	Optional	Implement	Illegal	Shall not be used by UFS Device
DME_PEER_SET	Optional	Implement	Illegal	Shall not be used by UFS Device
DME_POWERON	Optional	Optional	Illegal	Shall not be used by UFS Device
DME_POWEROFF	Optional	Optional	Illegal	Shall not be used by UFS Device
DME_ENABLE	Mandatory	Implement	Implement	
DME_RESET	Mandatory	Implement	Implement	UFS Device shall only use this primitive after DME_LINKLOST.ind
DME_ENDPOINTRESET	Mandatory	Implement	Implement	Shall not be used by UFS Device
DME_LINKSTARTUP	Mandatory	Implement	Implement	
DME_LINKLOST	Mandatory	Implement	Implement	Event indication only
DME_HIBERNATE_ENTER	Mandatory	Implement	Implement	Shall not be used by UFS Device
DME_HIBERNATE_EXIT	Mandatory	Implement	Implement	Shall not be used by UFS Device
DME_POWERMODE	Mandatory	Implement	Implement	Shall not be used by UFS Device
DME_TEST_MODE	Optional	Optional	Illegal	Shall not be used by UFS Host
DME_ERROR	Mandatory	Implement	Implement	Event indication only
DME_RXPWRSTATE	*D-PHY only*	*N/A*	*N/A*	*N/A for UFS*

NOTE A UFS Device shall only use DME Service Primitives if their usage is explicitly defined by the UFS specification. In such a case, UFS specifically defines the allowed parameter range and the sequence of actions.

6.6.6 UniPro Attributes

To optimize the UFS Boot procedure the UFS UIC implementation shall use the default reset values for all UniPro Attributes as defined by the MIPI UniPro specification. As an exception to this, the reset values of Network Layer Attributes and specific and specific Attributes for *CPort 0* shall reflect the settings which have been defined in the sections above and therefore shall contain the values as depicted in the table below:

Table 6-2 — UniPro Attribute

UniPro Attribute Name	UFS Host Reset Value	UFS Device Reset Value	UniPro 1.4 Reset Value
N_DeviceID	0	1	0
N_DeviceID_valid	TRUE	TRUE	FALSE
T_PeerDeviceID	1	0	N/A
T_PeerCPortID	0	0	N/A
T_E2EFC_CSD	2 (E2E_FC off, CSD off)	2 (E2E_FC off, CSD off)	1 (E2E_FC on)
T_ConnectionState	1 (CONNECTED)	1 (CONNECTED)	0 (IDLE)
T_TrafficClass	0	0	Any

7 UFS Transport Protocol (UTP) Layer

7.1 Overview

The SCSI Architecture Model (SAM-5) is used as the general architectural model for UTP. While the model uses the SCSI command set as the command set, it is not necessary to use SCSI commands exclusively. And, the SAM Task Management features for task management. A task is generally a SCSI command or service request.

The SAM Architecture is a client-server model or more commonly a request-response architecture. Clients are called Initiators and servers are called Targets. Initiators and Targets are mapped into UFS physical network devices. An Initiator issues commands or service requests to a Target that will perform the service requested. A Target is a Logical Unit (LU) contained within a UFS device. A UFS device will contain one or more Logical Units. A Logical Unit is an independent processing entity within the device.

A client request is directed to a single Logical Unit within a device. A Logical Unit will receive and process the client command or request. Each Logical Unit has an address within the Target device called a Logical Unit Number (LUN).

Communication between the Initiator and Target is divided into a series of messages. These messages are formatted into UFS Protocol Information Units (UPIU) as defined within this specification. There are a number of different UPIU types defined. All UPIU structures contain a common header area at the beginning of the data structure (lowest address). The remaining fields of the structure vary according to the type of UPIU.

A Task is a command or sequence of actions that perform a requested service. A Logical Unit contains a task queue that will support the processing of one or more Tasks. The Task queue is managed by the Logical Unit. A unique Initiator provided Task Tag is generated by the Initiator when building the Task. This Task Tag is used by the Target and the Initiator to distinguish between multiple Tasks. All transactions and sequences associated with a particular Task will contain that Task Tag in the transaction associated data structures.

Only one command within a Task set can be active. Only one Task can be processed at a time within a Logical Unit. If a device contains multiple Logical Units, it could have the ability to process multiple Tasks simultaneously or sequentially if so designed.

Command structures consist of Command Descriptor Blocks (CDB) that contain a command opcode and related parameters, flags and attributes. The description of the CDB content and structure are defined in detail in [SAM] and [SPC] and device specific (RBC, etc.) specifications.

A command transaction consists of a Command, an optional Data Phase, and a Status Phase. These transactions are represented in the form of UPIU structures. The Command Phase delivers the command information and supporting parameters from the Initiator to the Target. If a Data Phase is required, the direction of data flow is relative to the Initiator (host). A data WRITE travels from Initiator to Target. A data READ travels from Target to Initiator. At the completion of the command request the Target will deliver a response to the Initiator during the Status Phase. The response will contain the status and a UFS response status indicating successful completion or failure of the command. If an error is indicated the response will contain additional detailed UFS error information.

Depending upon system and network requirements or restrictions some UFS devices may be able to act as a Target or Initiator, though not simultaneously. Others may be designated as fixed Initiators or Targets.

7.2 UTP and Unipro Specific Overview

UTP will deliver commands, data and responses as standard message packets (T_SDU) over the Unipro network.

The UFS transactions will be grouped into data structures called UFS Protocol Information Units (UPIU).

There are UPIU's defined for UFS SCSI commands, responses, data in and data out, task management, utility functions, vendor functions, transaction synchronization and control. The list is extensible for future additions.

For enumeration and configuration, UFS supports a system of Descriptors, Attributes and Flags that define and control the specifics of the device, including operating characteristics, interfaces, number of logical units, operating speeds, power profiles, etc. The system is a hierarchical tree of related elements. It is easy to expand and infinitely extensible.

7.2.1 Phases

The SCSI- based Command protocol requires that the UPIU packets follow the transitions required to execute a command. Briefly, a command execution requires the sending of a Command UPIU, zero or more Data In or **Data Out UPIU** packets and terminates with a **Response UPIU** that contains the status.

7.2.2 Phase Collapse

In certain instances a SCSI Command and a **Data Out UPIU** packet or a Data In and **SCSI Response UPIU** packet can be collapsed into a single packet, thereby decreasing the number of UPIU packets that need to be transmitted. This decreases overhead and makes the overall transmission more efficient.

7.2.3 Data Pacing

A device may have limited memory resources for buffering or limited processing throughput. During a Command that requires a large Data Out transaction the Target device can pace the Data Out phase by sending a **Ready To Transfer UPIU** when it is ready for the next **Data Out UPIU**. In addition, the **Ready To Transfer UPIU** contains and embedded transfer context that can be used initiate a DMA transfer on a per packet basis at the initiating device (the Host).

During the Data In phase, no **Ready To Transfer UPIU** is required as the initiating Host has the ability to specify the size of the Data In transfer and thereby is able to allocate in advance the appropriate memory resources for the incoming data. A device issued **Data In UPIU** packet also contains an embedded DMA context that can be used to initiate a DMA transfer on a per packet basis.

7.2.4 Unipro

In keeping with the requirements of the Unipro Protocol the UFS Initiator and Target will divide its transactions into Unipro messages that will contain UPIU's. Unipro messages can handle T_SDU messages of theoretically unlimited size. UFS will impose a practicable limit on the maximum T_SDU message size. Currently, the limit is 65600 bytes which includes a 32-byte UPIU header, 32-bytes for potentially one extended header area and 65536 bytes of data segment. The minimum message size is determined by the basic header format, which is 32 bytes. There is a possibility that in the future this value will increase to allow a larger data segment area.

7.3 UFS Transport Protocol Transactions

7.3.1 Overview

UFS transactions consist of packets called UFS Protocol Information Units (UPIU) that travel between devices on the Unipro bus. A transaction begins between an Initiator and a Target in the form of a Request-Response operation. The Initiator starts the sequence of transactions by sending a request to a Target device and LUN. The Target will then respond with a series of transactions that eventually end in a response transaction.

All UFS UPIU's consist of a single basic header segment, possibly one or more extended header segments and zero or more data segments.

A basic header segment has a fixed length of 12 bytes. The minimum UPIU size is 32 bytes which includes a basic header segment and transaction specific fields.

The maximum UPIU size is unlimited but a practical limitation should be established for system use. Currently a reasonable limit of 65600 bytes is suggested. This size will support 64 bytes of UPIU header fields plus a 65600 byte data segment.

The UPIU format is flexible enough to be easily extended to support future transactions and larger data segments and will allow the application of this protocol to network protocols other than Unipro.

7.4 Service Delivery Subsystem

The Service Delivery Subsystem is an I/O system that transmits service requests and responses between Initiators and logical units within Targets connected via a physical or logical bus. The UFS UTP attempts to define a protocol that is independent of the Service Delivery Subsystem. This will allow for the easy porting of UTP to different Service Delivery Subsystems.

Currently, UFS is using the MIPI Unipro bus and the MIPI M-PHY as the Service Delivery Subsystem. For convenience and to aid in better understanding, portions of this specification will directly reference Unipro and M-PHY. Regardless of these references, the UTP protocol is independent of the Service Delivery Subsystem and should be able to port to other I/O systems.

UPIU structures will be handed off to MIPI Unipro as Unipro Service Data Units (T_SDU). Currently, the Unipro T_SDU requires no additional headers or trailer wrapped around the UPIU structure. This means that the T_SDU size will be exactly the UPIU size. The minimum size T_SDU will be 32 bytes. The maximum T_SDU size will be 65600 bytes.

7.4.1 UPIU Transaction Codes

Every UPIU data structure contains a Transaction Code. This code defines the content and implied function or use of the UPIU data structure. The following chart lists currently defined transaction codes.

Table 7-1 — UPIU Transaction Codes

Initiator To Target	T+Opcode	Target to Initiator	T+Opcode
NOP Out	00 0000b	NOP In	10 0000b
Command	00 0001b	Response	10 0001b
Data Out	00 0010b	Data In	10 0010b
Reserved	00 0011b	Reserved	10 0011b
Task Management Request	00 0100b	Task Management Response	10 0100b
Reserved	01 0001b	Ready To Transfer	11 0001b
Reserved	01 0010b	Reserved	11 0010b
Reserved	01 0100b	Reserved	11 0100b
Reserved	01 0101b	Reserved	11 0101b
Query Request	01 0110b	Query Response	11 0110b
Reserved	01 1100b	Reserved	11 1100b
Reserved	01 1101b	Reserved	11 1101b
Reserved	01 1110b	Reserved	11 1110b
Reserved	01 1111b	Reserved	11 1111b

7.4.1 UPIU Transaction Codes (cont'd)

Table 7-2 — UPIU Transaction Code Definitions

IPIU Data Structure	Description
NOP Out	The NOP Out transaction acts as a ping from an initiator to a target. It can be used to check for a connection path to a device and LUN.
NOP In	The NOP In transaction is a target response to an initiator when responding to a NOP In request.
Command	The Command transaction originates in the Initiator (host) and is sent to a logical unit within a Target device. A Command UPIU will contain a Command Descriptor Block as the command and the command parameters. When using the phase collapse feature the UPIU will also contain a data segment that would have been sent during the DATA OUT phase. This represents the COMMAND phase of the command.
Response	The Response transaction originates in the Target and is sent back to the Initiator (host). A Response UPIU will contain a command specific operation status and other response information. When using the phase collapse feature, the UPIU will also contain a data segment that would have been sent during the DATA IN phase. This represents the STATUS phase of the command.
Data Out	The Data Out transaction originates in the Initiator (host) and is used to send data from the Initiator to the Target (device). This represents the DATA OUT phase of a command.
Data In	The Data In transaction originates in the Target (device) and is used to send data from the Target to the Initiator (host). This represents the DATA IN phase of a command.
Task Management Request	This transaction type carries SCSI Architecture Model (SAM) task management function requests originating at the Initiator and terminating at the Target. The standard functions are defined by the SAM- 5 specification. Addition functions might be defined by UFS.
Task Management Response	This transaction type carries SCSI Architecture Model (SAM) task management function responses originating in the Target and terminating at the Initiator.
Ready To Transfer	The Target device will send a Ready To Transfer transaction when it is ready to receive the next Data Out UPIU and has sufficient buffer space to receive the data. The Target can send multiple Ready To Transfer UPIU if it has buffer space to receive multiple Data Out UPIU packets. The maximum data buffer size is negotiated between the Initiator and Target during enumeration and configuration. The **Ready To Transfer UPIU** contains a DMA context and can be used to setup and trigger a DMA action within a host controller.
Query Request	This transaction originates in the Initiator and is used to request descriptor data from the Target. This transaction is defined outside of the Command and Task Management functions and is defined exclusively by UFS.
Query Response	This transaction originates in the Target and provides requested descriptor information to the Initiator in response of the Query Request transaction. This transaction is defined outside of the Command and Task Management functions and is defined exclusively by UFS.

7.5 General UFS Protocol Information Unit Format

The diagram represents the general structure of a UPIU. All UPIU's will contain a fixed size and location basic header and additional fields as required to support the transaction type.

Table 7-3 — General format of the UFS Protocol Information Unit

General UPIU Format				
Transaction Type		Flags	LUN	Task Tag
Reserved	Command Set Type	Query Function	Response	Status
Total EHS Length		Device Information	Data Segment Length	
Transaction Specific Fields				
Extra Header Segment (EHS) 1				
…				
Extra Header Segment (EHS) N				
Header E2ECRC (omit if HD=0)				
Data Segment				
Data E2ECRC (omit if DD=0)				

7.5.1 Overview

UPIU total size will vary depending upon the UPIU transaction type but all UPIU sizes will be an integer multiple of 32-bits, meaning they will be addressed on a 4- byte boundary. If the aggregation of data and header segments does not end on a 32-bit boundary then additional padding will be added to round up the UPIU to the next 32-bit, 4- byte address boundary.

The UPIU size can be fixed or variable depending upon the Transaction Type field and extension flags. Some Transaction Types will have different lengths for the same code others will always be a fixed size. In addition, any UPIU can be extended if necessary to include extra header and data segments. The general format allows for extension and has flags and size fields defined within the structure to indicate to the processing entity where the extension areas are located within the structure and their size (not including padding) and in some cases the type of extension data.

7.5.2 Basic Header Format

This is the format of the basic header contained within every UPIU structure. This data packet will be sent between Initiators and Targets and will be part of a larger function specific UPIU. There is enough information in this header to allow the Initiator or the Target to track the destination and the source, the function request, if additional data and parameters are required and whether they are included in this UPIU or will follow in subsequent UPIU's.

7.5.2 Basic Header Format (cont'd)

The smallest sized UPIU is currently defined to have 32 bytes. The 32 bytes area will contain the basic header plus additional fields. This means that the smallest datum sent over the Service Delivery Subsystem will be 32 bytes.

Table 7-4 — Basic Header Format

Basic UPIU Header Format				
Transaction Type		Flags	LUN	Task Tag
Reserved	Command Set Type	Query Function	Response	Status
Total EHS Length		Device Information	Data Segment Length	

Table 7-5 — Basic Header fields

Flag	Description
Transaction Type	The Transaction Type indicates the type of request or response contained within the data structure. The Transaction Type contains an opcode as defined in 7.4.1, UPIU Transaction Codes. Transaction Type Format Transaction Type Bits: HD | DD | T | OPCODE HD The HD bit when set to '1' specifies that an end-to-end CRC of all Header Segments is included within the UPIU. The CRC fields include all fields within the header area. The CRC is placed at the 32-bit word location following the header. DD The DD bit when set to '1' specifies that an end-to-end CRC of the Data Segment is included with the UPIU. The 32-bit CRC is calculated over all the fields within the Data Segment. The 32-bit CRC word is placed at the end of the Data Segment. This will be the last word location of the UPIU. T The T bit when set to '1' indicates that the source of this UPIU is the Target device. The T bit when set to '0' indicates that the source of this UPIU is the Initiator device. The T bit can be considered part of the 5-bit opcode field that follows. OPCODE The 5-bit OPCODE field indicates the transaction code and the format of the UPIU. This code indicates the operation that is represented within the data fields of the UPIU and the number and location of the defined fields within the UPIU.

7.5.2 Basic Header Format (cont'd)

Table 7-5 — Basic Header fields (cont'd)

<table>
<tr><td>Flags</td><td>
The content of the Flags field vary with the Transaction Type opcode. For all opcodes, bit 7 always indicates if the UPIU is the final one of a series or only one expected.
<table>
<tr><th></th><th colspan="4">Operational Flags</th><th colspan="2">Reserved</th><th colspan="2">Task Attributes</th></tr>
<tr><td>UPIU Type</td><td>Bit 7</td><td>Bit 6</td><td>Bit 5</td><td>Bit 4</td><td>Bit 3</td><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td></tr>
<tr><td>Command</td><td>F</td><td>R</td><td>W</td><td>-</td><td>-</td><td>-</td><td colspan="2">ATTR</td></tr>
<tr><td>Response</td><td>F</td><td>O</td><td>U</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>Data Out</td><td>F</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>Data In</td><td>F</td><td>O</td><td>U</td><td>S</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>Ready-to-Transfer</td><td>F</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>Query Request</td><td>F</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>Query Response</td><td>F</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
</table>
ATTR Definition
<table>
<tr><th>Task Attribute</th><th>Bit 1</th><th>Bit 0</th></tr>
<tr><td>Simple</td><td>0</td><td>0</td></tr>
<tr><td>Ordered</td><td>0</td><td>1</td></tr>
<tr><td>Head of Queue</td><td>1</td><td>0</td></tr>
<tr><td>ACA (Not Used)</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>Response</td><td>If a response is required from a Target this field indicates whether the requested function succeeded or failed. This field is reserved for an Initiator.</td></tr>
<tr><td>Status</td><td>When coming from a Target, this field contains the SCSI status of the last requested SCSI command when the opcode is set to SCSI Response. Otherwise it contains an opcode specific status or is reserved. For an Initiator it is a reserved field.</td></tr>
<tr><td>Reserved</td><td>All fields marked as reserved shall contain a value of zero.</td></tr>
<tr><td>LUN</td><td>This field contains the Logical Unit Number to which a request is targeted. Target devices will contain at a minimum one logical unit numbered unit 0. This field is generated by the Initiator and maintained by the Target and Initiator for all UPIU transactions relating to a single request or task.</td></tr>
</table>

7.5.2 Basic Header Format (cont'd)

Table 7-5 — Basic Header fields (cont'd)

Task Tag	The Task Tag is generated by the Initiator when creating a task request. This field will be maintained by the Initiator and Target for all UPIU transactions relating to a single task. The Initiator will contain a register or variable that represents the Task Tag value. The Initiator will generate unique Task Tag by incrementing the internal variable when creating a new task request. All task related UPIU will contain this value in the Task Tag field. (A task request will be made up of a series of UPIU transactions).
Command Set Type	Command set type is four bit s. This field indicates the command set type the Command and Response UPIU is associated with.
Query Function	This field is used in Query Request and Query Response UPIU's to define the Query Function.
Device Information	This field provides device-level information required by specific UFS functionality in any Response UPIU.
Total Extra Header Segment Length	This field represents the size in 32-bit units (DWORDS) of all Extra Header Segments contained within the UPIU. This field is used if additional header segments are needed. Currently, UTP does not use Extra Header Segments. The value in this field is the number of total number of bytes in all EHS divided by four. If the number of each EHS segment is not a multiple of four, the segment will be padded with zeros to the next 32-bit boundary. $Total\ EHS\ Length\ value = (Total\ Extra\ Header\ Segment\ Bytes) \div 4$ The maximum size of all EHS fields combined is 1024 bytes. A value of zero in this field indicates that there are no EHS contained within the UPIU.
Data Segment Length	The Data Segment Length field contains the number of valid bytes within the Data Segment of the UPIU. When the number of bytes within the Data Segment is not a multiple of four then the last 32-bit field will be padded with zeros to terminate on the next nearest 32-bit boundary. The number of 32-bit units (DWORDS) that make up the Data Segment is calculated as follows: $Data\ Segment\ DWORDS = INTEGER(Data\ Segment\ Length + 3) \div 4$ The data segment can contain a maximum of 65535 bytes. A value of zero in this field indicates that there is no Data Segment within the UPIU.
Transaction Specific Fields	Additional fields as required by certain Transaction Codes are located within this area. For UTP, this area starts at byte address 12 within the UPIU and terminates on a 32 byte boundary at byte address 31. Since all UPIU contain a 12 byte Basic Header this leaves 20 bytes remaining for this area.
Extra Header Segments	The Extra Header Segments exist if the Total EHS Length field contains a non- zero value. For UTP, this area will start at byte address 32 within the UPIU. The UPIU may contain zero or more EHS. Currently, UTP does not use EHS.
Data Segment	The Data Segment field starts on the next 32- bit (DWORD) boundary after the EHS area within the UPIU. For UTP, there are no EHS areas used meaning that the Data Segment will begin at byte address 32 (byte address 36 if E2ECRC is enabled) within the UPIU. The Data Segment will be a multiple of 32- bits, thereby making the UPIU packet size a multiple of 4 bytes. The Data Segment Length field can contain a value that is not a multiple of 4 bytes but the Data Segment area will be padded with zeros to fill to the next nearest 32- bit (DWORD) boundary. The Data Segment Length field indicates the number of valid bytes within the Data Segment.

7.5.3 Command UPIU

7.5.3.1 Overview

The Command UPIU contains the basic UPIU header plus additional information needed to specify a command. The Initiator will generate this UPIU and send it to a Target to request a SCSI command service to be performed by the Target.

Table 7-6 — Command UPIU

Command UPIU				
xx00 0001b		Flags	LUN	Task Tag
Reserved	Command Set Type	Reserved	Reserved	Reserved
Total EHS Length		Reserved	Data Segment Length	
Expected Data Transfer Length				
CDB[0]		CDB[1]	CDB[2]	CDB[3]
CDB[4]		CDB[5]	CDB[6]	CDB[7]
CDB[8]		CDB[9]	CDB[10]	CDB[11]
CDB[12]		CDB[13]	CDB[14]	CDB[15]
Header E2ECRC (omit if HD=0)				
Data [0]		Data [1]	Data [2]	Data [3]
…		…	…	…
Data[Length-4]		Data[Length-3]	Data[Length-2]	Data[Length-1]
Data E2ECRC (omit if DD=0)				

7.5.3.2 Basic Header

The first 12 bytes of the Command UPIU contain the Basic Header as described in 7.5. Specific details follow below.

Table 7-7 — Command UPIU fields

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Transaction Type</td><td>A type code value of xx00 0001b indicates a Command UPIU.</td></tr>
<tr><td>Flags</td><td>Detail flags & description
<table>
<tr><th>Flag</th><th>Description</th></tr>
<tr><td>Flags.F</td><td>For a write operation a value of ‘1’ in the .F flag indicates that no Data Out UPIU’s will follow this UPIU thereby indicating this as the final outgoing UPIU being sent to the Target during this operation.
For a write operation a value of ‘0’ in the .F flag and a non-zero value in the Expected Data Transfer field indicate that one or more Data Out UPIU’s will follow this UPIU. The Target will request the Initiator to send additional SCSI Data Out UPIU’s by sending the Ready To Transfer UPIU to the Initiator.
For a read operation the .F flag should contain a value of ‘1’, indicating that this is the final outgoing UPIU to the Target during this operation.</td></tr>
<tr><td>Flags.R</td><td>A value of ‘1’ in the .R flag indicates that the command requires that a data read operation (incoming data) from Target to Initiator is required. If .R is set to ‘1’ then .W must be set to ‘0’. If .R and .W are set to ‘0’ then no data transfer is required for this command and the Expected Data Transfer Length field is ignored.</td></tr>
<tr><td>Flags.W</td><td>A value of ‘1’ in the .W flag indicates that the command requires a data write operation (outgoing data) from Initiator to Target. If .W is set to ‘1’ then .R must be set to ‘0’. If .W and .R are set to ‘0’ then no data transfer is required for this command and the Expected Data Transfer Length field is ignored.</td></tr>
<tr><td>Flags.ATTR</td><td>The .ATTR field contains the task attribute value as defined by the SAM-5 specification.
ATTR Definition
<table>
<tr><th>Task Attribute</th><th>Bit 1</th><th>Bit 0</th></tr>
<tr><td>Simple</td><td>0</td><td>0</td></tr>
<tr><td>Ordered</td><td>0</td><td>1</td></tr>
<tr><td>Head of Queue</td><td>1</td><td>0</td></tr>
<tr><td>ACA (Not Used)</td><td>1</td><td>1</td></tr>
</table></td></tr>
</table></td></tr>
</table>

7.5.3.2 Basic Header (cont'd)

Table 7-7 — Command UPIU fields (cont'd)

<table>
<tr><td>Data Segment Length</td><td>A non-zero value in the Data Segment Length field indicates that this UPIU contains all or part of the outgoing data for a write operation. It is an error if this field contains a non-zero value and the Flags.W field is set to '0'.

If Flags.W is set to '1' and the Data Segment Length is equal to the Expected Data Transfer Length then this UPIU contains all the data to be transferred to the target in the Data Segment field (phase collapse operation).

The Data Segment area always starts and ends on a 32-bit (DWORD) boundary. If necessary the area is padded out to the next nearest 32-bit boundary. The number of 32-bit (DWORD) units contained in the Data Segment area is calculated by:

$Data\ Segment\ DWORDS = INTEGER(Data\ Segment\ Length + 3) \div 4$

The Data Segment Length field indicates the actual number of valid bytes in the data segment and could be a value that is not a multiple of four. For example, if the Data Segment Length indicates 11 bytes, the Data Segment area will occupy 3 DWORDS or 12 bytes (INTEGER(11+3)/4 = 3 DWORDS) which results in 1 byte of padding.

The amount of outgoing data to be included within a Command UPIU for a write operation shall be limited. The host application client should query the device to determine the maximum allowable data size.</td></tr>
<tr><td>Expected Data Transfer Length</td><td>The Expected Data Transfer Length field contains a value that represents the number of bytes that are required to complete the SCSI command request and the number of bytes that the Initiator expects to be transferred to/from the Target. This field is valid only if one of the Flags.W or Flags.R bits are set to '1'.

For a write operation the .W flag shall be set to '1' and the .R flag shall be set to '0' and the value in this field represents the number of bytes that the Initiator expects to transfer to the Target.

For a read operation the .R flag shall be set to '1' and the .W flag shall be set to '0' and the value in this field represents the number of bytes that the Initiator expects the Target to transfer to it.

This model requires that the Initiator (Host) will allocate sufficient buffer space to receive the full size of the data requested by a command that requires a Data In operation. That size measured in bytes shall be the value in Expected Data Transfer Length field. This requirement is important in order to realize the full throughput of the Data In phase without the use of additional handshaking UPIU's.

The Initiator may request a Data Out size larger than the size of the Target's receive buffer. In this case, the Target will pace the Data Out UPIU's by sending Request To Transfer UPIU's as needed. The Initiator will not send a Data Out UPIU before it receives a Request To Transfer UPIU.</td></tr>
<tr><td>CDB</td><td>The CDB fields contain the Command Descriptor Block. This area is an array of 16 bytes that will contain a standard Command Descriptor Block as defined by one of the supported UFS Command Set Types. For SCSI commands, specifications such as SPC-4 or RBC can be referenced. For UFS Specific Commands, the current UFS command specification would be referenced. Up to a 16 byte CDB can be utilized. The CDB size is determined implicitly indicated by the group bits of the operation code field in CDB[0] for SCSI, which is the SCSI command operation code. For other commands, the CDB size is dependent upon the command opcode</td></tr>
<tr><td>Command Set Type</td><td>The Command Set Type field will specify an enumerated value that indicates which particular command set is used to define the command bytes in the CDB fields. This field is defined for the Command UPIU and the Response UPIU. This field is reserved in all other UPIU's. The Target device will use this field to indicate the type of command that is in the CDB field. The currently supported command types are listed in the table below.

<table>
<tr><th colspan="2">Command Set Type</th></tr>
<tr><td>0000</td><td>SCSI Command Set (SPC, RBC, SBC)</td></tr>
<tr><td>0001</td><td>UFS Specific Command Set</td></tr>
<tr><td>0010 - 1111</td><td>Reserved</td></tr>
<tr><td>F0..FFh</td><td>Vendor Specific Set</td></tr>
</table>
</td></tr>
</table>

7.5.4 Response UPIU

7.5.4.1 Overview

The Response UPIU contains the basic UPIU header plus additional information indicating the command and system level status resulting from the successful or failed execution of a Command Request UPIU. The Target will generate this UPIU and send it to the Initiator after it has completed the requested task.

Table 7-8 — Response UPIU

Response UPIU				
xx10 0001b		Flags	LUN	Task Tag
Reserved	Command Set Type	Reserved	Response	SCSI Status
Total EHS Length		Device Information	Data Segment Length	
Residual Transfer Count				
Reserved				
Reserved				
Reserved				
Reserved				
Header E2ECRC (omit if HD=0)				
Sense Data Length			Sense Data[0]	Sense Data[1]
…		…	…	…
Sense Data[14]		Sense Data[15]	Sense Data[16]	Sense Data[17]
Data E2ECRC (omit if DD=0)				

7.5.4.2 Basic Header

The first 12 bytes of the Response UPIU contain the Basic Header as described in section 7.5. Specific details follow below.

Table 7-9 — Response UPIU fields

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Transaction Type</td><td>A type code value of xx10 0001b indicates a Response UPIU.</td></tr>
<tr><td>Flags</td><td>Detail flags & description
<table>
<tr><th>Flag</th><th>Description</th></tr>
<tr><td>Flags.F</td><td>The .F flag will be set to ‘1’ as this UPIU will be the final (one and only) in the sequence of SCSI UPIU’s.</td></tr>
<tr><td>Flags.O</td><td>The .O flag will be set to ‘1’ to indicate that a data overflow occurred during the task execution. The Residual Transfer Count field will indicate the number of bytes not transferred from the Target to the Initiator. The Residual Transfer Count will be set to the absolute value difference of the Expected Data Transfer Length (from Command UPIU) and the actual number of bytes available to be transferred from the Target to the Initiator. See below for further explanation.</td></tr>
<tr><td>Flags.U</td><td>The .U flag will be set to ‘1’ to indicate that a data underflow occurred during the task execution. The Residual Transfer Count field will indicate the number of bytes that were not transferred out from the Initiator to the Target. The Residual Transfer Count will be set to the absolute value difference of the Expected Data Transfer Length (from Command UPIU) and the actual number of bytes transferred from the Initiator to the Target. See below for further explanation.</td></tr>
</table></td></tr>
<tr><td>Response</td><td>The Response field will contain the UFS response that indicates the UFS defined overall success or failure of the series of Command, Data and Response UPIU’s that make up the execution of a task.
Table 7-10 — UTP Response Values
<table>
<tr><th>Opcode</th><th>Response Description</th></tr>
<tr><td>00h</td><td>Target Success</td></tr>
<tr><td>01h</td><td>Target Failure</td></tr>
<tr><td>02h-7Fh</td><td>Reserved</td></tr>
<tr><td>80h-FFh</td><td>Vendor Specific</td></tr>
</table></td></tr>
</table>

7.5.4.2 Basic Header (cont'd)

Status

The Status field contains the command set specific status for a specific command issued by the Initiator. The Status field is command set specific. The Command Set Type field will indicate with which command set the status is associated. Specific command sets may or may not define detailed extended status indicated as Sense Data. If the command requires extended status, that information will be stored in the Sense Data field.

SCSI Command Set Status

When the Command Set Type field indicates SCSI Command Set the Status field will contain the standard SPC-4 defined SCSI status value. Possible values are listed in the table below. See the SPC-4 or SAM-5 for detailed definition of the status conditions.

A GOOD status indicates successful SCSI completion and therefore no Sense Data will be returned.

A status other CHECK CONDITION requires that the Data Segment contain Sense Data for the failed command.

Other status values may or may not return Sense Data. In this case a non-zero value in the Data Segment Length field indicates that this UPIU contains Sense Data in the Data Segment area.

'**M**' indicates mandatory implementation of this field and the value specified if fixed. '**O**' indicates that the support of this field is optional; if it is not supported then a value of zero should be inserted in the field otherwise the value will be indicated as described. n/a indicates "not applicable" to UFS.

Table 7-11 — SCSI Status Values

Opcode	Response Description	Use
00h	GOOD	M
02h	CHECK CONDITION	M
04h	CONDITION MET	n/a
08h	BUSY	M
18h	RESERVATION CONFLICT	O
28h	TASK SET FULL	M
30h	ACA ACTIVE	n/a
40h	TASK ABORTED	M

- **GOOD** - This status indicates that the device has completed the command without error.
- **CHECK CONDITION** - This status indicates that the device has completed the command with error or other actions are required to process the result. Valid Sense Data for the last command processed will be returned within the response UPIU when this status occurs.
- **CONDITION MET -** Not used for UFS.
- **BUSY -** This status indicates that the logical unit is busy. When the logical unit is unable to accept a command this status will be returned. Issuing the command at a later time is the standard recovery action.
- **RESERVATION CONFLICT -** This status is returned when execution of the command will result in a conflict of an existing reservation. UFS may support reserving areas of the device depending upon the device type and capabilities.
- **TASK SET FULL -** This status is returned when the logical unit cannot process the command due to a lack of resources such as task queue being full or memory needed for command execution is temporarily unavailable.
- **ACA ACTIVE -** This status is returned when an ACA condition exists. See SAM-5 for further definition.
- **TASK ABORTED -** This status is returned when a command is aborted by a command or task management function.

7.5.4.2 Basic Header (cont'd)

<table>
<tr><td>Device Information</td><td>Device-level information (defined in 10.3 - Host Device Interaction).

bit 0: Background Operation (see 10.3.4.6.3)
bit 1: Release LBA range (see 10.3.6.1.1.2)
others: Reserved</td></tr>
<tr><td>Data Segment Length</td><td>The Data Segment Length field will contain the number of valid bytes in the Data Segment. In the Response UPIU the Data Segment will contain the Sense Data bytes and the Sense Data Length field.

When this field contains zero it indicates that there is no Data Segment area in the UPIU and therefore no Sense Data is returned.

For example, the typical number of Sense Data bytes is 18, therefore this field would contain a value of 20 (18 bytes of Sense Data + 2 bytes for Sense Data Length = 20 bytes).

As stated previously, the Data Segment field size will located on a 32-bit (DWORD) boundary. The Data Segment Length value indicated the number of "valid" bytes in the Data Segment area and therefore can be less than a multiple of 32-bits.</td></tr>
<tr><td>Residual Transfer Count</td><td>This field is valid only if one of the Flags.U or Flags.O fields are set to '1', otherwise this field will contain zero.

When the Flags.O field is set to '1' then this field indicates the number of bytes that were not transferred in (to the Initiator) because the Expected Data Transfer Length field contained a value that was lower than the Target expected to transfer. In other words, the Target has more bytes to send to complete the request but the Initiator is not expecting more than the amount indicated in the Expected Data Transfer Length.

When the Flags.U field is set to '1' then this field indicates the number of bytes that were not transferred out (of the Initiator) to the Target. In other words, the Initiator has more bytes to send but the Target terminated the data out transfer before the number of bytes indicated in the Expected Data Transfer Length field was actually transferred.

Table 7-12 — Flags and Residual Count Relationship
<table>
<tr><th>.O</th><th>.U</th><th>Residual Count</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Expected Data Length bytes transferred</td></tr>
<tr><td>1</td><td>0</td><td>N</td><td>Target expected to send N more bytes to Initiator</td></tr>
<tr><td>0</td><td>1</td><td>N</td><td>Initiator expected to send N more bytes to Target</td></tr>
<tr><td>1</td><td>1</td><td>X</td><td>Illegal Condition</td></tr>
</table></td></tr>
<tr><td>Sense Data Area</td><td>The Sense Data fields will contain standard 18 byte SPC-4 defined sense data when using format for a Response Code value of 70h. See SPC-4 for further information.

SCSI Sense Data Area

The Sense Data fields will contain standard 18 byte SPC-4 defined sense data when using format for a Response Code value of 70h. See SPC-4 for further information.

Sense Data consists of three levels of error codes, each in increasing detail. The purpose is to provide the host a means to determine the cause of an error or exceptional condition at various levels of detail. The Sense Key provides a general category of what error or exceptional condition occurred and has caused the current command from successfully completing. Further and finer error detail is provided in the Additional Sense Code field (ASC). The Additional Sense Code Qualifier (ASCQ) field refines the error information even further. It is required to implement the Sense Key value when indicating an error or exceptional condition. It is NOT required to implement the ASC or ASCQ values; a value of zero can be placed in these fields if the implementation does not require more refined error detail.

All SCSI commands that terminate in error or exceptional condition will automatically return Sense Data, relieving the host from issuing a subsequent REQUEST SENSE command to retrieve the additional sense error information.

The Sense Data area is within the Data Segment area of the UPIU, therefore the Sense Data area will be placed on a 32-bit (DWORD) boundary. The Sense Data area will be padded with zeros to place the data on the next nearest 32-bit value if the length of valid Sense Data bytes is not a multiple of 32-bits. As currently defined, UFS devices will return 18 bytes of Sense Data plus 2 bytes of Sense Data Length for a total of 20 bytes or 5 DWORDS within the Data Segment of the UPIU.</td></tr>
</table>

7.5.4.2 Basic Header (cont'd)

Sense Data Length

The Sense Data Length field indicates the number of valid Sense Data bytes that follow. The number of valid bytes may or may not be a multiple of 32-bits. If not a multiple of 32-bits, the data will be padded with zeros to the next 32-bit boundary.

A successfully executed command will not normally need to return Sense Data, therefore the Sense Data Length in this case will contain a value of zero.

A command that terminated in error or an exception may or may not return Sense Data. If the Sense Data Length indicates a value of zero, then that error condition did not return Sense Data. A zero value in the Data Segment Length also indicates that no Sense Data was returned. Otherwise, the Sense Data Length will contain a value that indicates the number of additional bytes of Sense Data information.

SCSI Sense Data Length

Currently, the Sense Data Length field for UFS devices will indicate a value of 18 when using the SCSI Command Set.

Sense Data Format

The following table describes a typical sense data structure that gives detailed error information about the previously executed command. Typically 18 bytes are returned when the AdditionalSenseLength field is set to zero

'**M**' indicates mandatory implementation of this field and the value specified if fixed. '**O**' indicates that the support of this field is optional; if it is not supported then a value of zero should be inserted in the field otherwise the value will be indicated as described.

Table 7-13 — Typical SCSI Sense Data Format

START	END	BITS	NAME	DESCRIPTION	Use
0	0	7:7	Valid	'1' if Information field is valid Default value = '0'	O
0	0	6:0	ResponseCode	Value of 70h for standard sense data response	M
1	1	7:0	Obsolete	Not used Default value = 00h	M
2	2	7:7	FILEMARK	File mark found Default value = 0b	M
2	2	6:6	EOM	End of media detected Default value = 0b	M
2	2	5:5	ILI	Incorrect length detected Default value = 0b	M
2	2	4:4	Reserved	Default value = 0b	M
2	2	3:0	SenseKey	Sense Key code is the general SCSI error code for previous command (see SENSE KEY table below))	M
3	6	-	Information	Sense Information	O
7	7		AdditionalSenseLength	Length in bytes of additional sense information Value = 10 (0Ah) indicating 10 additional bytes (bytes 8 through 17)	M
8	11	-	CommandSpecific	Command Specific Information Default value = "0"	M
12	12	7:0	ASC	Additional Sense Code is an additional, more specific error code (see SCSI spec) Default value = 00h	
13	13	7:0	ASCQ	Additional Sense Code Qualifier qualifies the Additional Sense Code (see SCSI spec) Default value = 00h	M
14	14	7:0	FRUC	Field replaceable unit code Default value = 00h	M
15	17	-	SenseKeySpecific	Sense key specific information Default value = 00h	M

7.5.4.2 Basic Header (cont'd)

Sense Key

The **Sense Key** value provides a means to categorize errors and exceptional conditions. The Sense Key indicates a particular type of error. The **Additional Sense Code** and **Additional Sense Code Qualifier** can be used to further detail and describe the condition that the **Sense Key** indicates.

Sense Keys are specific to the action performed by a particular command.

Sense Key	
00h	NO SENSE – Indicates that there is no specific sense key information to be reported. This would be the result of a successfully executed command.
01h	RECOVERED ERROR – Indicates that the last command completed successfully after error recovery actions were performed by the target. Further detail may be determined by examining the additional sense bytes (ASC and ASCQ fields).
02h	NOT READY – Indicates that the logical unit addressed cannot be accessed at this time.
03h	MEDIUM ERROR – Indicates that the last command was unsuccessful due to a non-recoverable error condition due to a flaw in the media or failed error recovery.
04h	HARDWARE ERROR – Indicates that the target detected a non-recoverable hardware error.
05h	ILLEGAL REQUEST – Indicates that there was an illegal parameter value in a command descriptor block or within additional parameter data supplied with some commands. If the target detects an invalid parameter in the command descriptor block then it shall terminate the command without altering the media.
06h	UNIT ATTENTION – Indicates that the unit has been reset or unexpectedly powered-on or that removable media has changed. The UNIT ATTENTION condition will remain set until a REQUEST SENSE or AUTO-REQUEST SENSE has been issued to the target.
07h	DATA PROTECT – Indicates that a command that reads or writes the medium was attempted on a block that is protected from this operation. The read or write operation shall not be performed.
08h	BLANK CHECK - Indicates that a target encountered blank or unformatted media while reading or writing.
09h	VENDOR SPECIFIC – This Sense Key is available for reporting vendor specific error or exceptional conditions.
0Ah	RESERVED
0Bh	ABORTED COMMAND – Indicates that the target aborted the execution of the command. The initiator may be able to recover by retrying the command.
0Ch	RESERVED
0Dh	VOLUME OVERFLOW - Indicates that a buffered peripheral device has reached the end-of-partition and data may remain in the buffer that has not been written to the medium.
0Eh	MISCOMPARE – Indicates that the source data did not match the data read from the media.
0Fh	RESERVED

7.5.5 Data Out

7.5.5.1 Overview

The Data Out UPIU contains the basic UPIU header plus additional information needed to manage the data out transfer. The data transfer flows from Initiator to Target (WRITE). The Data Out UPIU will usually contain a data segment. It is possible to have a NULL Data Out UPIU.

The Data Out UPIU is sent in response to a Target generated Ready To Transfer UPIU.

Table 7-14 — SCSI Data Out UPIU

Data Out UPIU			
xx00 0010b	Flags	LUN	Task Tag
Reserved	Reserved	Reserved	Reserved
Total EHS Length	Reserved	Data Segment Length	
Data Buffer Offset			
Data Transfer Count			
Reserved			
Reserved			
Reserved			
Header E2ECRC (omit if HD=0)			
Data[0]	Data[1]	Data[2]	Data[3]
…	…	…	…
Data[Length -4]	Data[Length -3]	Data[Length -2]	Data[Length -1]
Data E2ECRC (omit if DD=0)			

7.5.5.2 Basic Header

The first 12 bytes of the Command UPIU contain the Basic Header as described in section 7.5. Specific details follow below.

Table 7-15 — SCSI Data Out UPIU fields

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Transaction Type</td><td>A type code value of 02h indicates a Data Out UPIU.</td></tr>
<tr><td>Flags</td><td>Detail flags & description
<table>
<tr><th>Flag</th><th>Description</th></tr>
<tr><td>Flags.F</td><td>A value of ‘1’ in the .F flag indicates that this is the last UPIU of a sequence of Data Out UPIU’s, i.e. the data transfer is finished.</td></tr>
</table></td></tr>
<tr><td>Data Segment Length</td><td>The value in this field indicates the size in bytes of the data payload within the Data Segment area.
The Data Segment area should be entirely filled to a 32-bit (DWORD) boundary unless the UPIU has the .F field set to ‘1’ indicating the final UPIU (no padding permitted unless last UPIU). In this case the Data Segment Length will indicate the number of valid bytes within the Data Segment.
The Data Segment area always starts and ends on a 32-bit (DWORD) boundary. If necessary the area is padded out to the next nearest 32-bit boundary.</td></tr>
<tr><td>Data Buffer Offset</td><td>The Data Buffer Offset field contains the offset of this UPIU data payload within the complete data transfer area. The sum of the Data Buffer Offset and the Data Segment Length should not exceed the Expected Transfer Length that was indicated in the Command UPIU.
This field permits out of order sequencing of the Data Out UPIU packets. Therefore the order of the Data Out UPIU packets do not have to be sequential.</td></tr>
<tr><td>Data Transfer Count</td><td>This field indicates the number of bytes that the Initiator is transferring to the Target in this UPIU. This value is the number of bytes that are contained within the Data Segment of the UPIU. The maximum number of bytes that can be transferred within a single Data Out UPIU packet is 65535 bytes. Therefore, multiple Data Out UPIU packets will need to be issued by the Initiator if the Expected Data Transfer Length of the original command requires more than 65535 bytes to be transferred.
This field and the Data Segment Length field of the UPIU will contain the same value. This field is intended to be used along with the Data Buffer Offset field as part of a DMA context.</td></tr>
<tr><td>Data Segment</td><td>This is the Data Segment area that contains the data payload. Currently, the UFS defined maximum data payload size is 65535 bytes.</td></tr>
</table>

7.5.6 Data In

7.5.6.1 Overview

The Data In UPIU contains the basic UPIU header plus additional information needed to manage the data in transfer. Data in flows from Target to Initiator (READ). The Data In UPIU will usually contain a data segment. It is possible to have a NULL Data In UPIU.

Table 7-16 — SCSI Data In UPIU

Data In UPIU			
xx10 0010b	Flags	LUN	Task Tag
Reserved	Reserved	Reserved	Status (applicable if Phase Collapse is Used)
Total EHS Length	Reserved	Data Segment Length	
Data Buffer Offset			
Data Transfer Court			
Reserved			
Reserved			
Reserved			
Header E2ECRC (omit if HD=0)			
Data[0]	Data[1]	Data[2]	Data[3]
…	…	…	…
Data[Length -4]	Data[Length -3]	Data[Length -2]	Data[Length -1]
Data E2ECRC (omit if DD=0)			

7.5.6.2 Basic Header

The first 12 bytes of the SCSI Command UPIU contain the Basic Header as described in 7.5. Specific details follow.

Table 7-17 — SCSI Data In UPIU fields

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Transaction Type</td><td>A type code value of xx10 0010b indicates a Data In UPIU.</td></tr>
<tr><td>Flags</td><td>Detail flags & description
<table>
<tr><th>Flag</th><th>Description</th></tr>
<tr><td>Flags.F</td><td>A value of ‘1’ in the .F flag indicates that this is the last UPIU of a sequence of Data In UPIU’s, i.e. the data transfer is finished.</td></tr>
<tr><td>Flags.S</td><td>A value of ‘1’ in the .S flag indicates that this UPIU contains SCSI Status information. This flag indicates the phase collapse of the Status phase meaning that this UPIU contains the last data block and a Status that did not end in an exception. Status Phase collapse cannot be used if the Status ends in an exception condition.</td></tr>
<tr><td>Flags.O</td><td>The .O field is set to ‘1’ it indicates an data overflow occurred. This field is only valid if the .S field is set to ‘1’.</td></tr>
<tr><td>Flags.U</td><td>When the .U flag is set to ‘1’ it indicates a data underflow occurred. This field is only valid if the .S field is set to ‘1’.</td></tr>
</table></td></tr>
<tr><td>Data Segment Length</td><td>The value in this field indicates the size in bytes of the data payload within the Data Segment area.
The Data Segment area should be entirely filled to a 32-bit (DWORD) boundary unless the UPIU has the .F field set to ‘1’ indicating the final UPIU (no padding permitted unless last UPIU). In this case the Data Segment Length will indicate the number of valid bytes within the Data Segment.
The Data Segment area always starts and ends on a 32-bit (DWORD) boundary. If necessary the area is padded out to the next nearest 32-bit boundary.</td></tr>
<tr><td>Data Buffer Offset</td><td>The Data Buffer Offset field contains the offset of this UPIU data payload within the complete data transfer area. The sum of the Data Buffer Offset and the Data Segment Length should not exceed the Expected Transfer Length that was indicated in the Command UPIU.
This field permits out of order sequencing of the Data In UPIU packets. Therefore the order of the SCSI In UPIU packets do not have to be sequential.</td></tr>
<tr><td>Data Transfer Count</td><td>This field indicates the number of bytes that the Target is transferring to the Initiator. This value is the number of bytes that are contained within the Data Segment of the UPIU. The maximum number of bytes that can be transferred within a single Data In UPIU packet is 65535 bytes. Therefore, this value may result in multiple Data In UPIU packets being issued by the Initiator to satisfy this request.</td></tr>
<tr><td>Data Segment</td><td>This is the Data Segment area that contains the data payload. Currently, the UFS defined maximum data payload size is 65535 bytes. The actual size of the data segment area will be rounded up to the next higher 32-bit address value if the Data Transfer Count is not a multiple of 4 bytes.</td></tr>
<tr><td>Status</td><td>Applicable only if Phase Collapse is used. The returned Status must be GOOD, otherwise a separate Response UPIU is sent with the status information in the Status Phase.</td></tr>
</table>

7.5.7 Ready to Transfer

7.5.7.1 Overview

The Ready To Transfer UPIU is issued by the Target when it is ready to receive data blocks when processing a SCSI command that requires a data out transfer (WRITE). The Target may request the data in sequence or out of order by setting the appropriate fields within the UPIU.

The Initiator will respond to one or more **Ready To Transfer UPIU** packets with one or more **Data Out UPIU** packets, enough to satisfy the **Expected Transfer Length** that was indicated within the associated **Command UPIU**. The maximum number of bytes that can be sent within a single **Data Out UPIU** packet is equal to bMaxDataOutSize attribute value.

Table 7-18 — Ready To Transfer UPIU

Ready To Transfer UPIU			
xx11 0001b	Flags	LUN	Task Tag
Reserved	Reserved	Reserved	Status
Total EHS Length	Reserved	Data Segment Length	
Data Buffer Offset			
Data Transfer Count			
Reserved			
Reserved			
Reserved			
Header E2ECRC (omit if HD=0)			

7.5.7.2 Basic Header

The first 12 bytes of the Command UPIU contain the Basic Header as described in 7.5. Specific details follow.

Table 7-19 — Ready To Transfer UPIU fields

Field	Description
Transaction Type	A type code value of xx11 0001b indicates a Ready to Transfer UPIU.
Flags	Detail flags & description **Flag**: **Description** Flags.F: The Flags.F field will always contain a value of '1', indicating that this is the last and only UPIU of this sequence.
Data Segment Length	The Data Segment Length field shall contain zero as there is no Data Segment in this UPIU.
Data Buffer Offset	The Data Buffer Offset field indicates to the Initiator the location of the beginning of the segment of data to send. The Target may request data from the Initiator in several blocks, not necessarily in sequential order. The sum of the Data Buffer Offset and the Requested Data Transfer Length should not exceed the Expected Transfer Length that was indicated in the Command UPIU.
Data Transfer Count	This field indicates the number of bytes the Target is requesting. The maximum number of bytes that can be transferred within a single Data Out UPIU is equal to bMaxDataOutSize attribute value

7.5.8 Task Management Request

The Task Management Request UPIU is used by the Initiator to manage the execution of one or more tasks within the Target. The Task Management Request function closely follows the SAM-5 model.

Table 7-20 — Task Management Request UPIU

Task Management Request UPIU			
xx00 0100b	Flags	LUN	Task Tag
Reserved	Reserved	Function	Reserved
Total EHS Length	Reserved	Data Segment Length	
Input Parameter 1			
Input Parameter 2			
Input Parameter 3			
Reserved			
Reserved			
Header E2ECRC (omit if HD=0)			

Table 7-21 — Task Management Request UPIU

Field	Description
Input Parameter 1	LUN of the logical unit operated on by the task management function.
Input Parameter 2	Task Tag of the task/command operated by the task management function.
Input Parameter 3	Reserved

7.5.8.1 Task Management Function

Function	Value	Description
Abort Task	01h	Abort specific task in queue in a specific LU. Identify by LUN and Task Tag
Abort Task Set	02h	Abort the task queue list in a specific LU. Identify by LUN.
Clear Task Set	04h	Clear the task queue list in specific LU. Identify by LUN. Equivalent to Abort Task Set.
Logical Unit Reset	08h	Reset the designated LU. Identify by LUN
Query Task	80h	Query a specific task in a queue list in a specific LU. Identify by LUN and Task Tag. If the specific task is present in the queue, Function Succeeded is returned in the response. If the specific task is not present in the queue, Function Complete is returned in the response.
Query Task Set	81h	Query a specific LU to see if there is any Task in queue. Identify by LUN. If there is one of more tasks present in the queue, Function Succeeded is returned in the response. If no task is present in the queue, Function Complete is returned in the response.

7.5.9 Task Management Response

The Task Management Response UPIU is sent by the Target in response to a Task Management Request from the Initiator. The Task Management Response function closely follows the SAM-5 model.

Table 7-22 — Task Management Response UPIU

Task Management Response UPIU			
xx10 0100b	Flags	LUN	Task Tag
Reserved	Reserved	Response	Reserved
Total EHS Length	Reserved	Data Segment Length	
Output Parameter 1			
Output Parameter 2			
Reserved			
Reserved			
Reserved			
Header E2ECRC (omit if HD=0)			

Table 7-23 — Task Management Response UPIU fields

Field	Description
Output Parameter 1	Service Response (see table below)
Output Parameter 2	Reserved

7.5.9.1 Task Management Service Response

Table 7-24 — Task Management Service Response

Service Response	Value
Task Management Function Complete	00h
Task Management Function Not Supported	04h
Task Management Function Failed	05h
Task Management Function Succeeded	08h
Incorrect Logical Unit Number	09h

7.5.10 Query Request

The **Query Request UPIU** is used to transfer data between the **Initiator** and **Target** that is outside domain of standard user data transfers for command read and writes.

The **Query Request UPIU** can be used to read and write parametric data to or from the **Target**. It can be used to get information for configuration or enumeration, to set or clear bus or overall device conditions, to set or reset global flag values, parameters or attributes, to set or get power or bus or network information or to get or set descriptors, to get serial numbers or GUID's (globally unique identifiers), etc.

The Query Request UPIU follows the general UPIU format with a field defined for QUERY FUNCTION. Value of LUN field is ignored in a Query Request transaction.

The Transaction Specific Fields are defined specifically for each type of operations which are defined by the **OPCODE** field. For the STANDARD READ REQUEST and STANDARD WRITE REQUEST the fields are defined as in the table.

The Data Segment Area is optional depending upon the QUERY FUNCTION value. The Data Segment Length field will be set to zero if there is no data segment in the packet.

Table 7-25 — Query Request UPIU

Query Request UPIU			
xx01 0110b	Flags	LUN	Task Tag
Reserved	QUERY FUNCTION	Reserved	Reserved
Total EHS Length	Reserved	Data Segment Length	
Transaction Specific Fields			
Transaction Specific Fields			
Transaction Specific Fields			
Transaction Specific Fields			
Reserved			
Header E2ECRC (omit if HD=0)			
Data[0]	Data[1]	Data[2]	Data[3]
…	…	…	…
Data[Length -4]	Data[Length -3]	Data[Length -2]	Data[Length -1]
Data E2ECRC (omit if DD=0)			

7.5.10.1 Overview

Queries are used to read and write data structures between the Host and the Device. This data is outside the scope of normal device reads or writes; data that would be considered system data, configuration data, production information, descriptors, special parameters and flags and other.

For UFS the Query function will generally be used to read or write DESCRIPTORS, ATTRIBUTES and FLAGS. There are also a range of Vendor Specific operations that can be used to transfer vendor specific data between Host and Device.

All these items reside within the device memory and used by the device to control or define its operation.

Below is a short overview of the most common data structures that are transferred using the QUERY REQUEST function. Please see the related section for more detail on these data structures.

7.5.10.2 Descriptors

A Descriptor is a block or page of parameters that describe something about a Device. For example, there are Device Descriptors, Configuration Descriptors, Unit Descriptors, etc.

7.5.10.3 Attributes

An Attribute is a single parameter that represents a specific range of numeric values that can be set or read. This value could be a byte or word or floating point number. For example, baud rate or block size would be an attribute.

7.5.10.4 Flags

A Flag is a single Boolean value that represents a TRUE or FALSE, ‘0’ or ‘1’, ON or OFF type of value. A Flag can be cleared or reset, set, toggled or read. Flags are useful to enable or disable certain functions or modes or states within the device.

7.5.10.5 Query Functions

The **Query Function** field holds the requested query type describing the query function to perform. Common query functions are listed in the following table. Currently, there are two general query functions defined: Read Request and Write Request. Additional **Transaction Specific Fields** will be used to specify further information needed for the transaction. These fields can describe the specific operation to perform, the target data or information to access, the amount of data to transfer and additional parameters and data.

Table 7-26 — Query Functions

QUERY FUNCTION	
00h	NOP READ
01h	STANDARD READ REQUEST
02h-3Fh	Reserved
40-7Fh	Vendor Specific Read Functions
80h	NOP WRITE
81h	STANDARD WRITE REQUEST
82h-BFh	Reserved
C0h-FFh	Vendor Specific Write Functions

Query Function Field Values

7.5.10.6 Standard Read Request

The **Standard Read Request** function type is used to read requested information from a **Target** device. The **Target** device will return the requested information to the **Initiator** within a **Query Response UPIU** packet.

7.5.10.7 Standard Write Request

The **Standard Write Request** function type is used to write information and data to a **Target** device. The information and data to write to the **Target** will be included within the **Data Segment** field of the **Query Request UPIU** packet.

7.5.10.8 Transaction Specific Fields

The transaction specific fields are defined specifically for each type of operations which are defined byte the **OPCODE** field. For the STANDARD READ REQUEST and STANDARD WRITE REQUEST the fields are defined in Table 7-27.

Table 7-27 — Transaction specific fields

Transaction Specific Fields for Standard Read/Write Request			
OPCODE	OSF[0]	OSF[1]	OSF[2]
OSF[3]	OSF[4]	OSF[5]	
OSF[6]			
OSF[7]			

7.5.10.8.1 OPCODE

The opcode indicates the operation to perform. Possible opcode values are listed in Table 7-28.

Table 7-28 — Query Function opcode values

OPCODE	
00h	NOP
01h	READ DESCRIPTOR
02h	WRITE DESCRIPTOR
03h	READ ATTRIBUTE
04h	WRITE ATTRIBUTE
05h	READ FLAG
06h	SET FLAG
07h	CLEAR FLAG
08h	TOGGLE FLAG
09h-EFh	Reserved
F0h-FFh	Vendor Specific

7.5.10.8.2 OSF

The OSF field is an Opcode Specific Field. The OSF fields will be defined for each specific OPCODE.

7.5.10.9 Read Descriptor Opcode

The **READ DESCRIPTOR OPCODE** is used to retrieve a UFS DESCRIPTOR from the Target device. A descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields are used to select a particular descriptor and to read a number of descriptor bytes. The OSF fields are defined as listed in Table 7-29.

Table 7-29 — Read descriptor

Transaction Specific Fields for READ DESCRIPTOR OPCODE			
01h	DESCRIPTOR IDN	INDEX	SELECTOR
Reserved	Reserved	LENGTH	
Reserved			
Reserved			

7.5.10.9.1 DESCRIPTOR IDN

The DESCRIPTOR IDN field contains a value that indicates the particular type of descriptor to retrieve. For example, it could indicate a DEVICE DESCRIPTOR or UNIT DESCRIPTOR or STRING DESCRIPTOR. Some descriptor types are unique and can be fully identified by the Descriptor Type value. Other descriptors can exist in multiple forms, such as STRING DESCRIPTORS, and must be furthered identified with subsequent fields.

7.5.10.9.2 INDEX

The INDEX value is used to further identify a particular descriptor. For example, there may be multiple STRING DESCRIPTORS defined. In the case of multiple descriptors the INDEX field is used to select a particular one. Multiple descriptors are indexed starting from 0 through 255. The actual index value for a particular descriptor will be provided by other means, usually contained within a field of some other related descriptor.

7.5.10.9.3 SELECTOR

The SELECTOR field may be needed to further identify a particular descriptor. In most cases this value is 0.

7.5.10.9.4 LENGTH

The LENGTH field is used to indicate the number of bytes to read of the descriptor. These bytes will be returned in a **QUERY RESPONSE UPIU** packet. This is the requested length to read, which may be less than or equal to or greater than the number of bytes within the actual descriptor. If less than or equal to the actual descriptor size, the number of bytes specified will be returned. If the LENGTH is greater than the descriptor size, the response will pad the data segment.

7.5.10.10 Write Descriptor Opcode

The **WRITE DESCRIPTOR OPCODE** is used to write a UFS DESCRIPTOR from the Host to the Target device. A descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields are used to select a particular descriptor. The OSF fields are defined as listed in Table 7-30.

Table 7-30 — Write Descriptor

Transaction Specific Fields for WRITE DESCRIPTOR OPCODE			
02h	DESCRIPTOR IDN	INDEX	SELECTOR
Reserved	Reserved	LENGTH	
Reserved			
Reserved			

7.5.10.10.1 DESCRIPTOR IDN

The DESCRIPTOR IDN field contains a value that identifies a particular of descriptor to write. For example, it could indicate a DEVICE DESCRIPTOR or UNIT DESCRIPTOR or STRING DESCRIPTOR. Some descriptor types are unique and can be fully identified by the Descriptor IDN value. Other descriptors can exist in multiple forms, such as STRING DESCRIPTORS, and must be furthered identified with subsequent fields.

7.5.10.10.2 INDEX

The INDEX value is used to further identify a particular descriptor. For example, there may be multiple STRING DESCRIPTORS defined. In the case of multiple descriptors the INDEX field is used to select a particular one. Multiple descriptors are indexed starting from 0 through 255. The actual index value for a particular descriptor will be provided by other means, usually contained within a field of some other related descriptor.

7.5.10.10.3 SELECTOR

The SELECTOR field may be needed to further identify a particular descriptor. In most cases this value is 0.

7.5.10.10.4 LENGTH

The LENGTH field is used to indicate the number of descriptor bytes to write. The entire descriptor must be written; there is no partial write or update possible. These bytes will be contained within the DATA SEGMENT area of the **QUERY REQUEST UPIU** packet. The DATA SEGMENT LENGTH field of the UPIU should also be set to this same value.

7.5.10.11 READ ATTRIBUTE OPCODE

The **READ ATTRIBUTE OPCODE** is used to retrieve a UFS ATTRIBUTE from the Target device. An attribute can be an 8-bit, 16-bit, 24-bit or 32-bit value. There are up to 256 possible ATTRIBUTES, identified by an identification number, IDN, which ranges from 0 to 255. The OSF fields for this opcode are listed in Table 7-31.

Table 7-31 — Read Attribute

Transaction Specific Fields for READ ATTRIBUTE OPCODE			
03h	ATTRIBUTE IDN	Reserved	Reserved
Reserved	Reserved	LENGTH	
Reserved			
Reserved			

7.5.10.11.1 ATTRIBUTE IDN

The ATTRIBUTE IDN contains a value that identifies a particular ATTRIBUTE to retrieve from the Target device.

7.5.10.11.2 LENGTH

The LENGTH field is used to indicate the length in bytes of the attribute. This value must be the exact size of the ATTRIBUTE; there is no padding or shortening of the attribute value. These bytes will be returned in a **QUERY RESPONSE UPIU** packet. This is the requested length to read, which, for an ATTRIBUTE, must be the exact number of bytes returned in the response.

7.5.10.12 Write Attribute Opcode

The **WRITE ATTRIBUTE OPCODE** is used to write a UFS ATTRIBUTE to the Target device. An attribute can be an 8-bit, 16-bit, 24-bit or 32-bit value. There are up to 256 possible ATTRIBUTES, identified by an identification number, IDN, which ranges from 0 to 255. The OSF fields for this opcode are listed in the following table

Table 7-32 — Write Attribute

Transaction Specific Fields for WRITE ATTRIBUTE OPCODE			
04h	ATTRIBUTE IDN	Reserved	Reserved
Reserved	Reserved	LENGTH	
VALUE [31:24]	VALUE [23:16]	VALUE [15:8]	VALUE [7:0]
Reserved			

7.5.10.12.1 ATTRIBUTE IDN

The ATTRIBUTE IDN contains a value that identifies a particular ATTRIBUTE to retrieve from the Target device.

7.5.10.12.2 LENGTH

The LENGTH field is used to indicate the number of bytes to write into the ATTRIBUTE. This length must be the exact length of the defined ATTRIBUTE.

7.5.10.12.3 VALUE [31:0]

The 32-bit VALUE field contains the data value of the ATTRIBUTE. The VALUE is a right justified, big Endian value. Unused upper byte fields should be set to zero.

7.5.10.13 Read Flag Opcode

The **READ FLAG OPCODE** is used to retrieve a UFS FLAG value from the Target device. An FLAG is a fixed size single byte value that represents a Boolean value. There can be defined up to 256 possible FLAG values. A FLAG is identified by its FLAG IDN, an identification number that ranges in value from 0 to 255. The OSF fields for this opcode are listed in Table 7-33.

The FLAG data, either ‘1’ (0x01) or ‘0’ (0x00), is returned within the Transaction Specific Fields area of a **QUERY RESPONSE UPIU** packet.

Table 7-33 — Read Flag

Transaction Specific Fields for READ FLAG OPCODE			
05h	FLAG IDN	Reserved	Reserved
Reserved	Reserved	Reserved	Reserved
Reserved			
Reserved			

7.5.10.13.1 FLAG IDN

The FLAG IDN field contains a value that identifies a particular FLAG to retrieve from the Target device.

7.5.10.13.2 Operation

The Boolean value of the addressed flag is returned in a Query Response UPIU.

7.5.10.14 Set Flag

Table 7-34 — Set Flag

Transaction Specific Fields for READ FLAG OPCODE			
06h	FLAG IDN	Reserved	Reserved
Reserved	Reserved	Reserved	Reserved
Reserved			
Reserved			

7.5.10.14.1 FLAG IDN

The FLAG IDN field contains a value that identifies a particular FLAG to set in the Target device.

7.5.10.14.2 Operation

The Boolean value of the addressed flag is set to TRUE or ‘1’.

7.5.10.15 Clear Flag

Table 7-35 — Clear Flag

Transaction Specific Fields for READ FLAG OPCODE			
07h	FLAG IDN	Reserved	Reserved
Reserved	Reserved	Reserved	Reserved
Reserved			
Reserved			

7.5.10.15.1 FLAG IDN

The FLAG IDN field contains a value that identifies a particular FLAG to clear in Target device.

7.5.10.15.2 Operation

The Boolean value of the addressed flag is cleared to FALSE or ‘0’.

7.5.10.16 Toggle Flag

Table 7-36 — Toggle Flag

Transaction Specific Fields for TOGGLE FLAG OPCODE			
08h	FLAG IDN	Reserved	Reserved
Reserved	Reserved	Reserved	Reserved
Reserved			
Reserved			

7.5.10.16.1 FLAG IDN

The FLAG IDN field contains a value that identifies a particular FLAG to toggle in the Target device.

7.5.10.16.2 Operation

The Boolean value of the addressed flag is set to the negated current value.

7.5.11 Query Response

The **Query Response UPIU** is used to transfer data between the **Target** and **Initiator** in response to a Query Request UPIU.

The **Query Response UPIU** is used to return parametric data to or from the requesting **Initiator**. It can be used to get device information for configuration or enumeration, to set or clear bus or overall device conditions, to set or reset global flag values, parameters or attributes, to set or get power or bus or network information or to get or set descriptors, to get serial numbers or GUID's (globally unique identifiers), etc.

The Query Response UPIU follows the general UPIU format a field defined for QUERY FUNCTION. Value of LUN field is ignored in a Query Response transaction.

The transaction specific fields are defined specifically for each type of operations which are defined byte the **OPCODE** field of the STANDARD READ REQUEST and STANDARD WRITE REQUEST function.

The Data Segment Area is optional depending upon the QUERY FUNCTION value. The Data Segment Length field will be set to zero if there is no data segment in the packet.

Table 7-37 — Query Response

Query Response UPIU			
xx11 0110b	Flags	LUN	Task Tag
Reserved	QUERY FUNCTION	RESPONSE	Reserved
Total EHS Length	Reserved	Data Segment Length	
Transaction Specific Fields			
Transaction Specific Fields			
Transaction Specific Fields			
Transaction Specific Fields			
Reserved			
Header E2ECRC (omit if HD=0)			
Data[0]	Data[1]	Data[2]	Data[3]
…	…	…	…
Data[Length -4]	Data[Length -3]	Data[Length -2]	Data[Length -1]
Data E2ECRC (omit if DD=0)			

7.5.11.1 Overview

The Query Response function will respond to the QUERY FUNCTION that was sent from the Initiator in the Query Request UPIU. The Query Response may or may not return data depending upon the function. If data needs to be returned it will be returned in the Data Segment of the UPIU or one of the Transaction Specific Fields.

7.5.11.2 Query Function

The QUERY FUNCTION field will contain the original QUERY FUNCTION value that was sent in the corresponding Query Request UPIU.

7.5.11.3 Response

The QUERY RESPONSE field indicates the completion code of the action taken in response to the Query Request UPIU. Possible values are listed in Table 7-38.

Table 7-38 — Query Response Code

QUERY RESPONSE – RESPONSE FIELD VALUES	
00h	SUCCESS
01h-FEh	Reserved
FFh	FAILURE

7.5.11.4 Transaction Specific Fields

The transaction specific fields are defined specifically for each type of operations which are defined byte the **OPCODE** field. For the STANDARD READ REQUEST and STANDARD WRITE REQUEST the fields are defined in Table 7-39.

Table 7-39 — Transaction Specific Fields

Transaction Specific Fields for Standard Read/Write Request			
OPCODE	OSF[]0]	OSF[1]	OSF[2]
OSF[3]	OSF[4]	OSF[5]	
OSF[6]			
OSF[7]			

7.5.11.4.1 OPCODE

The opcode indicates the operation to perform. Possible opcode values are listed in Table 7-40.

Table 7-40 — Query Function opcode Values

OPCODE	
00h	NOP
01h	READ DESCRIPTOR
02h	WRITE DESCRIPTOR
03h	READ ATTRIBUTE
04h	WRITE ATTRIBUTE
05h	READ FLAG
06h	SET FLAG
07h	CLEAR FLAG
08h	TOGGLE FLAG
09h-EFh	Reserved
F0h-FFh	Vendor Specific

7.5.11.4.2 OSF

The OSF field is an Opcode Specific Field. The OSF fields will be defined for each specific OPCODE.

7.5.11.5 Read Descriptor Opcode

The **READ DESCRIPTOR OPCODE** is used to retrieve a UFS DESCRIPTOR from the Target device. A descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields are used to select a particular descriptor and to read a number of descriptor bytes. The OSF fields are defined as listed in Table 7-41.

The READ DESCRIPTOR OPCODE is returned in response to a Query Request UPIU containing the same value in the OPCODE field.

Table 7-41 — Read Descriptor

Transaction Specific Fields for READ DESCRIPTOR OPCODE			
01h	DESCRIPTOR IDN	INDEX	SELECTOR
Reserved	Reserved	LENGTH	
Reserved			
Reserved			

7.5.11.5.1 DESCRIPTOR IDN

The DESCRIPTOR IDN field contains the same DESCRIPTOR IDN value sent from the corresponding Query Request UPIU.

7.5.11.5.2 INDEX

The Index field value returned is the same INDEX value of the corresponding Query Request UPIU.

7.5.11.5.3 SELECTOR

The SELECTOR field returned is the same SELECTOR value of the corresponding Query Request UPIU.

7.5.11.5.4 LENGTH

The LENGTH field is used to indicate the number of bytes returned in response to the corresponding Query Response UPIU. This value could be less than the requested size if the size of the data item is smaller than the size requested in the corresponding Query Request UPIU.

7.5.11.6 Write Descriptor Opcode

The **WRITE DESCRIPTOR OPCODE** is used to receive a UFS DESCRIPTOR from the Host to write into the Target device. A descriptor can be a fixed or variable length. There are up to 256 possible descriptor types. The OSF fields are used to select a particular descriptor. The OSF fields are defined as listed in Table 7-42.

Table 7-42 Write Descriptor

Transaction Specific Fields for WRITE DESCRIPTOR OPCODE			
02h	DESCRIPTOR IDN	INDEX	SELECTOR
Reserved	Reserved	LENGTH	
Reserved			
Reserved			

7.5.11.6.1 DESCRIPTOR IDN

The Descriptor IDN field contains the same DESCRIPTOR IDN value sent from the corresponding Query Request UPIU.

7.5.11.6.2 INDEX

The Index field value returned is the same INDEX value of the corresponding Query Request UPIU.

7.5.11.6.3 SELECTOR

The SELECTOR field returned is the same SELECTOR value of the corresponding Query Request UPIU.

7.5.11.6.4 LENGTH

The LENGTH field is used to indicate the number of descriptor bytes to write.. The entire descriptor must be written; there is no partial write or update possible.

7.5.11.7 Read Attribute Opcode

The **READ ATTRIBUTE OPCODE** is used to retrieve a UFS ATTRIBUTE from the Target device. An attribute can be an 8-bit, 16-bit, 24-bit or 32-bit value. There are up to 256 possible ATTRIBUTES, identified by an identification number, IDN, which ranges from 0 to 255.

7.5.11.7.1 RESPONSE DATA FORMAT

The response to a READ ATTRIBUTE request will be returned in a **QUERY RESPONSE UPIU**. A success or failure code for the entire operation will be contained within the **RESPONSE** field.

The attribute data will be returned within the transaction specific fields. The Transaction Specific fields are formatted as indicated in the following table. The first two 32-bits words of those fields will echo the first two 32-bit words of the transaction specific fields of the **QUERY REQUEST UPIU**. The third word will contain the **ATTRIBUTE** data. The **LENGTH** field will contain the number of valid bytes returned within the 32-bit field. This value can range from 0 to 4.

Table 7-43 — Read Attribute Response Data Format

Transaction Specific Fields for READ ATTRIBUTE OPCODE			
03h	ATTRIBUTE IDN	Reserved	Reserved
Reserved	Reserved	LENGTH	
VALUE [31:24]	VALUE [23:16]	VALUE [15:8]	VALUE [7:0]
Reserved			

7.5.11.8 Write Attribute Opcode

The **WRITE ATTRIBUTE OPCODE** is used to write a UFS ATTRIBUTE to the Target device. An attribute can be an 8-bit, 16-bit, 24-bit or 32-bit value. There are up to 256 possible ATTRIBUTES, identified by an identification number, IDN, which ranges from 0 to 255. The OSF fields for this opcode are listed in the following table

Table 7-44 — Write Attribute

Transaction Specific Fields for WRITE ATTRIBUTE OPCODE			
04h	ATTRIBUTE IDN	Reserved	Reserved
Reserved	Reserved	LENGTH	
VALUE [31:24]	VALUE [23:16]	VALUE [15:8]	VALUE [7:0]
Reserved			

7.5.11.8.1 LENGTH

The LENGTH field is used to indicate the number of bytes to write into the ATTRIBUTE. This length must be the exact length of the defined ATTRIBUTE.

7.5.11.8.2 VALUE [31:0]

The 32-bit VALUE field contains the data value of the ATTRIBUTE. The VALUE is a right justified, big Endian value. Unused upper byte fields should be set to zero.

7.5.11.9 Read Flag Opcode

The **READ FLAG OPCODE** is used to retrieve a UFS FLAG value from the Target device. An FLAG is a fixed size single byte value that represents a Boolean value. There can be defined up to 256 possible FLAG values. A FLAG is identified by its FLAG IDN, an identification number that ranges in value from 0 to 255. The FLAG data, either '1' or '0', is returned within the Transaction Specific Fields area of a **QUERY RESPONSE UPIU** packet.

7.5.11.9.1 RESPONSE DATA FORMAT

The response to a READ ATTRIBUTE request will be returned in a **QUERY RESPONSE UPIU**. A success or failure code for the entire operation will be contained within the **RESPONSE** field.

The attribute data will be returned within the transaction specific fields. The Transaction Specific fields are formatted as indicated in the following table. The first two 32-bits words of those fields will echo the first two 32-bit words of the transaction specific fields of the **QUERY REQUEST UPIU**. The third word will contain the **FLAG** data, a '0' or '1' value.

Table 7-45 — Read Flag Response Data Format

Transaction Specific Fields for READ ATTRIBUTE OPCODE			
03h	FLAG IDN	Reserved	Reserved
Reserved	Reserved	Reserved	Reserved
Reserved	Reserved	Reserved	FLAG VALUE
Reserved			

7.5.11.10 Set Flag

Table 7-46 — Set Flag

Transaction Specific Fields for READ FLAG OPCODE			
06h	FLAG IDN	Reserved	Reserved
Reserved	Reserved	Reserved	Reserved
Reserved			
Reserved			

7.5.11.11 Clear Flag

Table 7-47 — Clear Flag

Transaction Specific Fields for READ FLAG OPCODE			
07h	FLAG IDN	Reserved	Reserved
Reserved	Reserved	Reserved	Reserved
Reserved			
Reserved			

7.5.11.12 Toggle Flag

Table 7-48 — Toggle Flag

Transaction Specific Fields for READ FLAG OPCODE			
08h	FLAG IDN	Reserved	Reserved
Reserved	Reserved	Reserved	Reserved
Reserved			
Reserved			

7.6 Logical Units

7.6.1 Overview

This section gives more details on the definition of Logical Unit in the UFS Specification.

7.6.2 UFS SCSI Domain

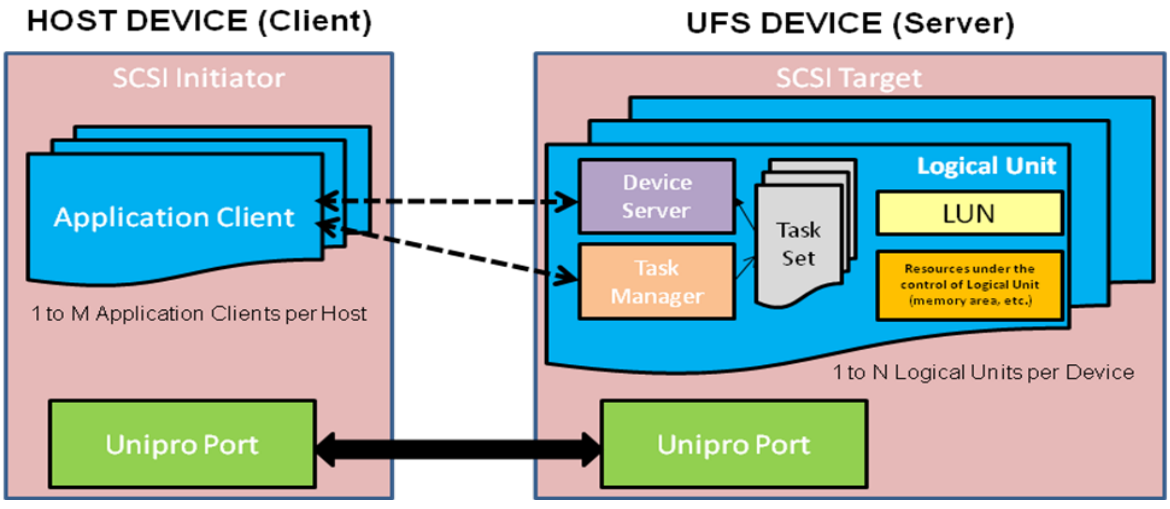

Figure 7-1 — UFS SCSI Domain

7.6.3 UFS Logical Unit Definition

A Logical Unit (LU) is an externally addressable, independent, processing entity that processes SCSI tasks (commands) and performs task management functions.

- A UFS device contains 1 or more Logical Units
- Each Logical Unit is independent of other Logical Units in a device
- UFS shall support up to 8 logical units in addition to the well-known logical units as defined in this specification.

Commands addressed to LU i are handled by LU i exclusively, not visible, handled or processed by LU j (or any other LU)

A Logical Unit contains the followings:

- DEVICE SERVER: A conceptual object within a logical unit that processes SCSI commands.
- TASK MANAGER: A conceptual object within a logical unit that controls the sequencing of commands and performs task management functions.
- TASK SET: A conceptual group of 1 or more commands (a list, queue, etc.)

7.6.4 Well-Known Logical Unit Definition

Well-Known Logical Units, as defined by SCSI, support very specific types of commands, usually only one or two commands such as REPORT LUNS command to allow an Application Client to issue requests to receive specific information usually relating to the entire device.

In the UFS specification, additional Well-Known Logical Units are defined for specific UFS functions, including Boot and RPMB.

7.6.5 Logical Unit Addressing

LUN Field in UPIU

7	6	5	4	3	2	1	0
WLUN_ID	UNIT_NUMBER_ID						

WLUN_ID specifies whether an LU or a W-LU is being addressed.

A value of ‘0’ indicates an LU is being addressed
A value of ‘1’ indicates an W-LU is being addressed

UNIT_NUMBER_ID specifies the unit number

Figure 7-2 — Logical Unit Addressing

Logical Units are addressed by an 8-bit value in the LUN field of the UPIU header. The MSB=’1’ (WLUN_ID) indicates Well-Known LU address.

The UFS 8-bit LUN number field supports two types of LUN addressing:

- If WLUN_ID bit = ‘0’ then the UNIT_NUMBER_ID field addresses a standard logical unit (LUN)
- If WLUN_ID bit = ‘1’ then the UNIT_NUMBER_ID field addresses a WELL KNOW LOGICAL UNIT (W-LUN)

Up to 128 LUN’s and up to 128 W-LUN’s

- 0 <= UNIT_NUMBER_ID <= 127
- 0 <= UNIT_NUMBER_ID <= 127

7.6.6 Well-Known Logical Unit Defined in UFS

The following Well-Known Logical Units are defined in UFS specification for SCSI and UFS specific functions

- SCSI REPORT LUNS W-LU
  - Responds to REPORT LUNS and INQUIRY commands as defined in SPC-4
  - REPORT LUNS LU = 01h (as defined by SPC-4
  - UFS LUN encoding: 01h.OR.80h = 81h (WLUN_ID bit set to 1)
- UFS Device Well-Known LU
  - Well-known for UFS device level interaction
  - UFS Device Well-Known LU shall respond to START_STOP command as defined in Power Mode chapter
  - UFS Device W-LUN = 50h
  - UFS LUN encoding: 50h.OR.80h = D0h (WLUN_ID bit set to 1)
- Boot W-LU
  - The default Boot W-LUN referenced by host device to address the logical unit that contains the valid boot code at the system startup. Boot W-LUN is a virtual reference to the actual LU containing boot code, as designated by the host.
  - Boot W-LU shall support only READ command
  - UFS Boot W-LUN = 30h
  - UFS LUN encoding: 30h.OR.80h = B0h (WLUN_ID bit set to 1)
- RPMB W-LU
  - RPMB W-LU supports the RPMB function and shall respond only to the Security In and Security Out commands.
  - UFS RPMB W-LUN = 44h
  - UFS LUN encoding: 44h.OR.80h = C4h (WLUN_ID bit set to 1)

7.6.7 Translation of 8-bit UFS LUN to 64-bit SCSI LUN Address

The SCSI Architecture Model describes a 64-bit LUN addressing scheme. The value of C1h in the first 8 bits of the 64-bit address indicates a Well-Known LUN address in the SCSI definition. Translation of the 8-bit UFS LUN address to 64-bit SCSI address is as follows:

- UFS LUN = PQh → SAM LUN = 00h PQh 00h 00h 00h 00h 00h 00h
  - Example: UFS LUN 06h →SAM LUN 0006000000000000h
- UFS W-LUN = XYh.OR.80h → SAM W-LUN = C1h XYh 00h 00h 00h 00h 00h 00h
  - Example: UFS W-LUN 81h = 01h.OR80H → SAM W-LUN C101000000000000h (SAM W-LUN)

7.6.7 Translation of 8-bit UFS LUN to 64-bit SCSI LUN Address (cont'd)

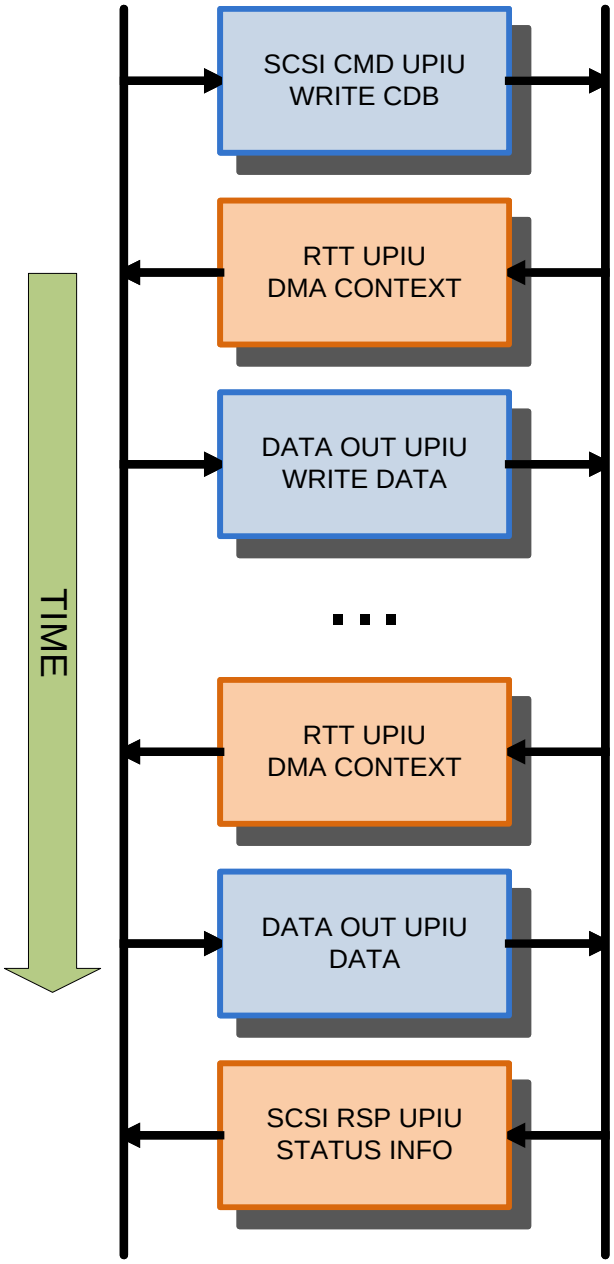

Figure 7-3 — SCSI Write

7.6.7 Translation of 8-bit UFS LUN to 64-bit SCSI LUN Address (cont'd)

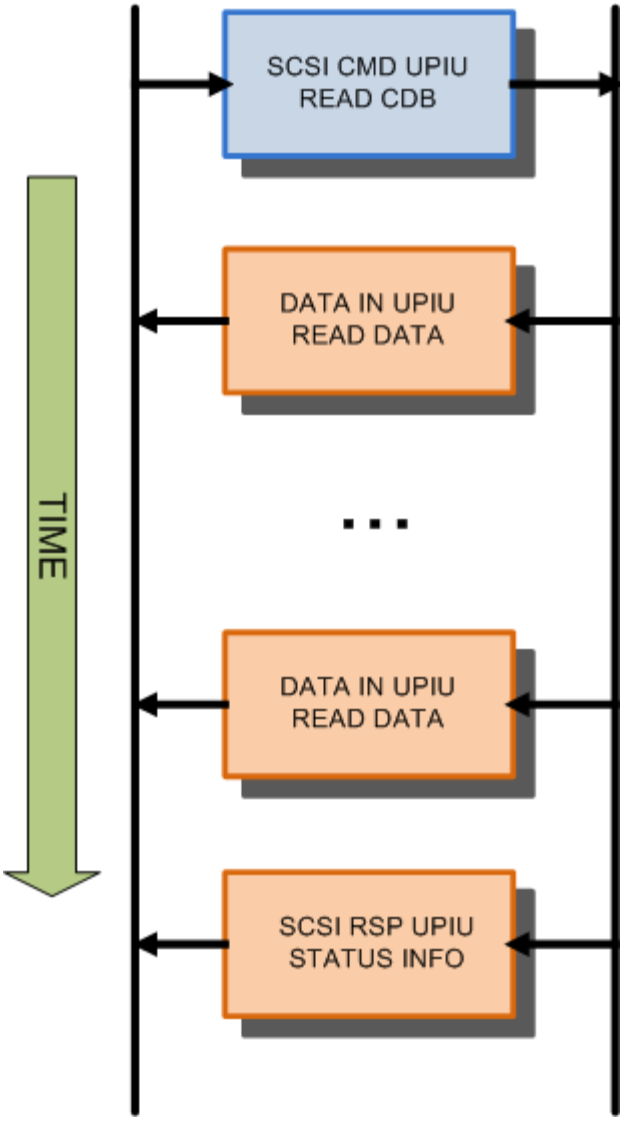

Figure 7-4 — SCSI Read

7.7 UFS Initiator Port and Target Port Attributes

Table 7-49 — UFS Initiator Port and Target Port Attributes

Attribute	Value
Maximum CDB Length	16 bytes
Command Identifier Size	16 bits
Task Attributes Supported	Simple, Head of Queue, Ordered, ACA (not supported)
Maximum Data-In Buffer Size	FFFFh
Maximum Data-Out Buffer Size	FFFFh
Maximum CRN	TBD
Command Priority Supported	No
Maximum Sense Data Length	FFFFh
Status Qualifier Supported	No
Additional Response Information Supported	Yes
Bidirectional Commands Supported	No
Task Management Functions Supported	Abort Task, Abort Task Set, Clear Task Set, Logical Unit Reset, Query Task, Query Task Set

7.7.1 Execute Command procedure call transport protocol services

All SCSI transport protocol standards shall define the SCSI transport protocol specific requirements for implementing the Send SCSI Command request, the SCSI Command Received indication, the Send Command Complete response, and the Command Complete Received confirmation SCSI transport protocol services.

All SCSI initiator devices shall implement the Send SCSI Command request and the Command Complete Received confirmation SCSI transport protocol services as defined in the applicable SCSI transport protocol standards. All SCSI target devices shall implement the SCSI Command Received indication and the Send Command Complete response SCSI transport protocol services as defined in the applicable SCSI transport protocol standards.

Initiator	Target
Send SCSI Command request	SCSI Command Received indication
Command Complete Received confirmation	Send Command Complete response

7.7.1 Execute Command procedure call transport protocol services (cont'd)

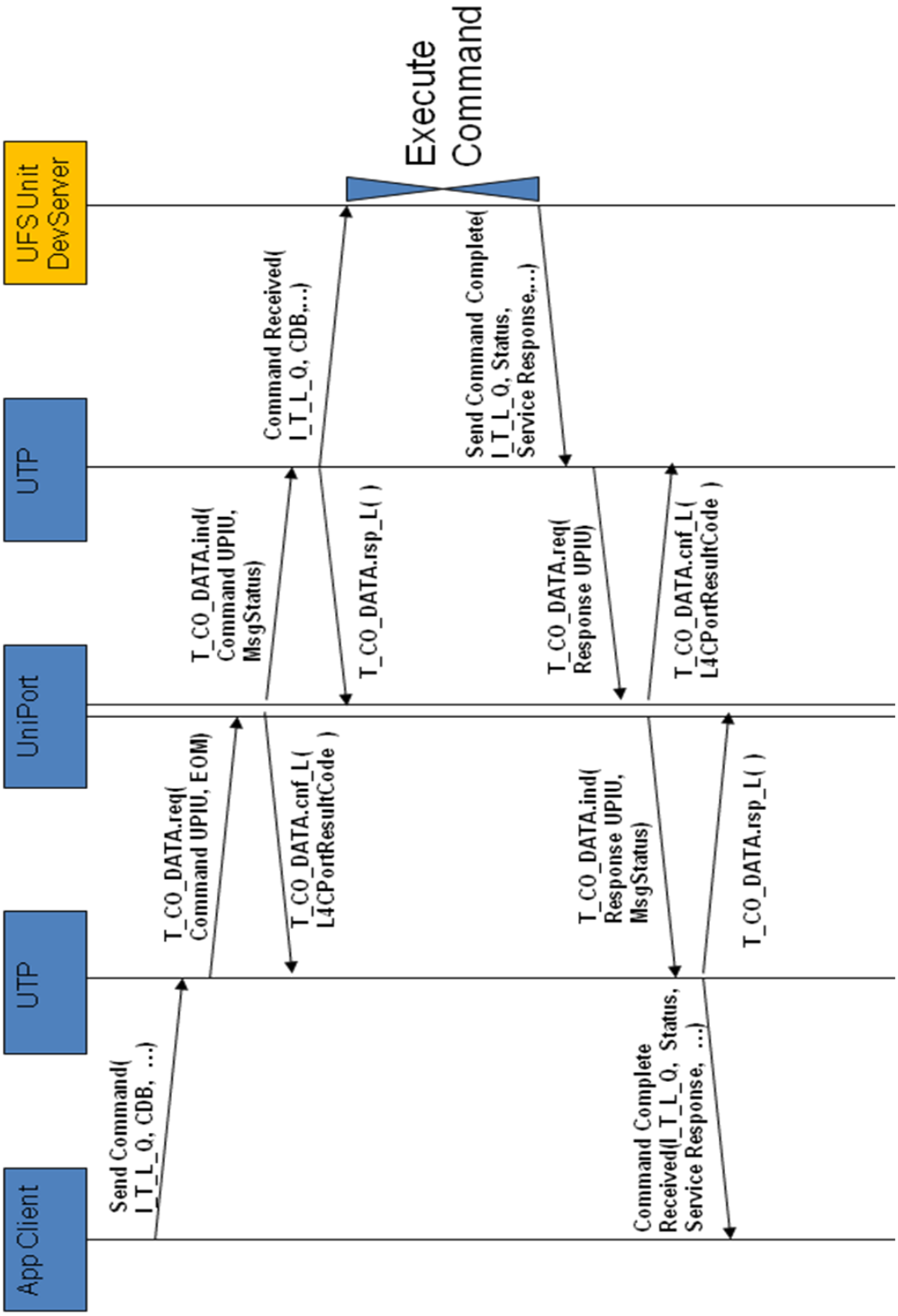

Figure 7-5 — Command w/o Data Phase

7.7.2 Send SCSI Command transport protocol service

An application client uses the Send Command transport protocol service request to request a UFS Initiator port transmit a COMMAND UPIU via UniPort

- Send SCSI Command (IN (I_T_L_Q Nexus, CDB, Task Attribute, [Data-in Buffer Size], [Data-out Buffer], [Data-out Buffer Size], [CRN], [Command Priority], [First Burst Enabled]))

7.8 Implementation

Table 7-50 — Send SCSI Command transport protocol service

Argument	Implementation
I_T_L_Q Nexus	I specifies the initiator port to send the COMMAND UPIU T specifies the target port to which the COMMAND UPIU is to be sent L specifies the LUN field in the COMMAND UPIU Q specifies the Task Tag field in the COMMAND UPIU
CDB	Specifies the CDB field in the COMMAND UPIU
Task Attribute	Specifies the Flag.ATTR of the Flag field in the COMMAND UPIU
[Data-in Buffer Size]	Expected Data Transfer Length
[Data-out Buffer]	Internal to UFS Initiator port
[Data-out Buffer Size]	Expected Data Transfer Length
[CRN]	TBD
[Command Priority]	Ignored
[First Burst Enabled]	An argument specifying that a SCSI transport protocol specific number of bytes from the Data-Out Buffer shall be delivered to the logical unit without waiting for the device server to invoke the Receive Data-Out SCSI transport protocol service. A non-zero value in the Data Segment Length field indicates First Burst Enabled.

7.8.1 SCSI Command Received transport protocol

A UFS target port uses the SCSI Command Received transport protocol service indication to notify a task manager that it has received a COMMAND UPIU.

- SCSI Command Received (IN (I_T_L_Q Nexus, CDB, Task Attribute, [CRN], [Command Priority], [First Burst Enabled))

Table 7-51 — SCSI Command Received transport protocol

Argument	Implementation
I_T_L_Q Nexus	I indicates the initiator port from which COMMAND UPIU was received T indicates the target port which receives the COMMAND UPIU L indicates the LUN field in the COMMAND UPIU Q indicates the Task Tag field in the COMMAND UPIU
CDB	Specifies the CDB field in the COMMAND UPIU
Task Attribute	Specifies the Flag.ATTR of the Flag field in the COMMAND UPIU
[CRN]	TBD
[Command Priority]	Ignored
[First Burst Enabled]	An argument specifying that a SCSI transport protocol specific number of bytes from the Data-Out Buffer shall be delivered to the logical unit without waiting for the device server to invoke the Receive Data-Out SCSI transport protocol service. A non-zero value in the Data Segment Length field indicates First Burst Enabled.

7.8.2 Send Command Complete transport protocol service

A device server uses the Send Command Complete transport protocol service response to request that a UFS target port transmit a RESPONSE UPIU.

- Send Command Complete (IN (I_T_L_Q Nexus, [Sense Data], [Sense Data Length], Status, [Status Qualifier], Service Response))

A device server shall only call Send Command Complete () after receiving SCSI Command Received ().

A device server shall not call Send Command Complete () for a given I_T_L_Q nexus until:

a) all its outstanding Receive Data-Out () calls for that I_T_L_Q nexus have been responded to with Data-Out Received (); and
b) all its outstanding Send Data-In () calls for that I_T_L_Q nexus have been responded to with Data-In Delivered ().

7.8.3 Send Command Complete transport protocol service (cont'd)

Table 7-52 — Send Command Complete transport protocol service

Argument	Implementation
I_T_L_Q Nexus	I specifies the initiator port to send the RESPONSE UPIU T specifies the target port to which the RESPONSE UPIU is to be sent L specifies the LUN field in the RESPONSE UPIU Q specifies the Task Tag field in the RESPONSE UPIU
[Sense Data]	Specifies the Sense Data field in the RESPONSE UPIU
[Send Data Length]	Specifies the Sense Data Length field in the RESPONSE UPIU
Status	Specifies the Status Length field in the RESPONSE UPIU
[Status Qualifier]	Ignored
Service Response	Specifies the Response field in the RESPONSE UPIU

7.8.3 Command Complete Received transport protocol service

A UFS initiator port uses the Command Complete Received transport protocol service confirmation to notify an application client that it has received a response for its COMMAND UPIU.

- Command Complete Received (IN (I_T_L_Q Nexus, [Data-in Buffer], [Sense Data], [Sense Data Length], Status, [Status Qualifier], Service Response))

Table 7-53 — Command Complete Received transport protocol service

Argument	Implementation
I_T_L_Q Nexus	I specifies the initiator port to send the RESPONSE UPIU T specifies the target port to which the RESPONSE UPIU is to be sent L specifies the LUN field in the RESPONSE UPIU Q specifies the Task Tag field in the RESPONSE UPIU
[Data-out Buffer]	Internal to UFS Initiator port
[Sense Data]	Specifies the Sense Data field in the RESPONSE UPIU
[Send Data Length]	Specifies the Sense Data Length field in the RESPONSE UPIU
Status	Specifies the Status Length field in the RESPONSE UPIU
[Status Qualifier]	Ignored
Service Response	Specifies the Response field in the RESPONSE UPIU

7.8.4 Data transfer SCSI transport protocol services

The data transfer services provide mechanisms for moving data to and from the SCSI initiator port while processing commands. All SCSI transport protocol standards shall define the protocols required to implement these services.

The application client's Data-In Buffer and/or Data-Out Buffer each appears to the device server as a single, logically contiguous block of memory large enough to hold all the data required by the command.

7.8.4.1 Send Data-In transport protocol service

A device server uses the Send Data-In transport protocol service request to request that a UFS target port sends data.

- Send Data-In (IN (I_T_L_Q Nexus, Device Server Buffer, Request Byte Count))

A device server shall only call Send Data-In () during a read command.

A device server shall not call Send Data-In () for a given I_T_L_Q nexus after it has called Send Command Complete () for that I_T_L_Q nexus (e.g., a RESPONSE UPIU with for that I_T_L_Q nexus) or called Task Management Function Executed for a task management function that terminates that task (e.g., an ABORT TASK).

Table 7-54 — Send Data-In transport protocol service

Argument	Implementation
I_T_L_Q Nexus	I_T_L_Q of the corresponding COMMAND UPIU
Device Server Buffer	Internal to device server
Request Byte Count	Specifies the length of the read data specified by the command

7.8.4.2 Data-In Delivered transport protocol service

This confirmation notifies the device server that the specified data was successfully delivered to the application client buffer, or that a UniPro delivery subsystem error occurred while attempting to deliver the data.

- Data-In Delivered (IN (I_T_L_Q Nexus, Delivery Result))

Table 7-55 — Data-In Delivered transport protocol service

Argument	Implementation
I_T_L_Q Nexus	I_T_L_Q of the corresponding COMMAND UPIU
Delivery Result	DELIVERY SUCCESSFUL: The data was delivered successfully. DELIVERY FAILURE: A UniPro service delivery error occurred while attempting to deliver the data.

7.8.5.2 Data-In Delivered transport protocol service (cont'd)

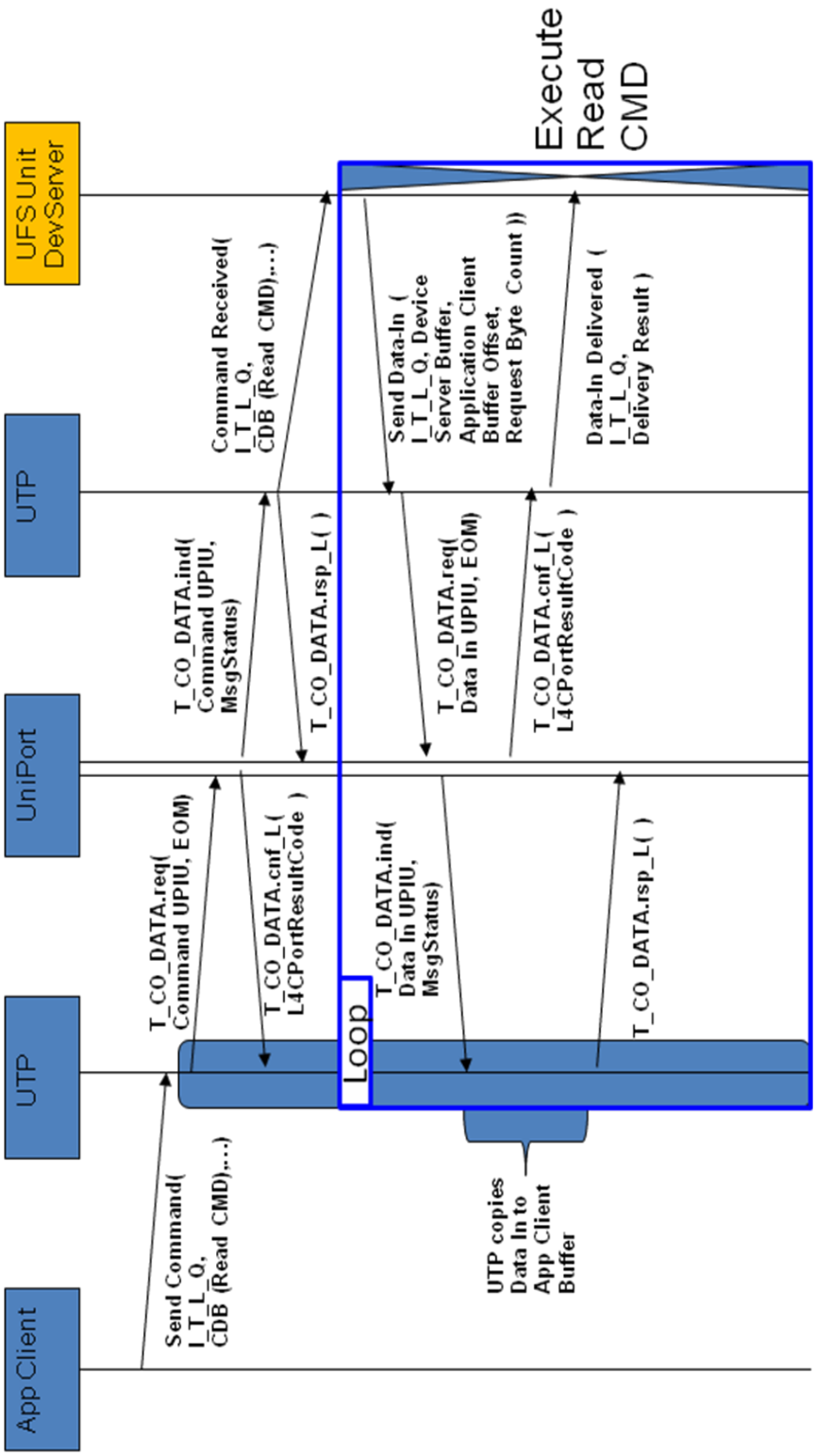

Figure 7-6 — Command + Read Data Phase 1/2

7.8.5.2 Data-In Delivered transport protocol service (cont'd)

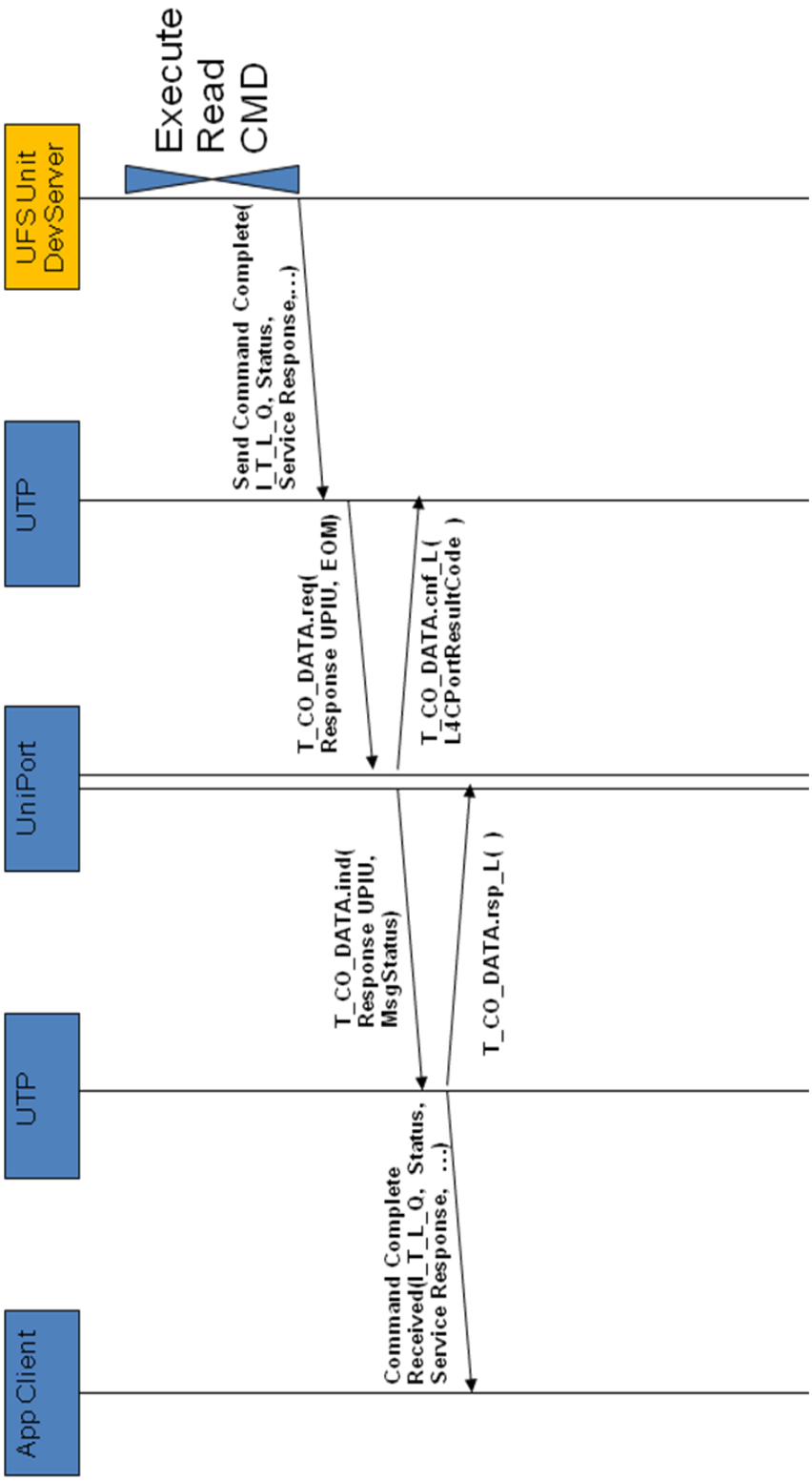

Figure 7-7 — Command + Read Data Phase 2/2Receive Data-Out transport protocol service

7.8.5.2 Data-In Delivered transport protocol service (cont'd)

A device server uses the Receive Data-Out transport protocol service request to request that a UFS target port receives data

- Receive Data-Out (IN (I_T_L_Q Nexus, Application Client Buffer Offset, Request Byte Count, Device Server Buffer))

A device server shall only call Receive Data-Out () during a write command.

A device server shall not call Receive Data-Out () for a given I_T_L_Q nexus after a Send Command Complete () has been called for that I_T_L_Q nexus or after a Task Management Function Executed () has been called for a task management function that terminates that command (e.g., an ABORT TASK).

Table 7-56 — Receive Data-Out transport protocol service

Argument	Implementation
I_T_L_Q Nexus	I_T_L_Q of the corresponding COMMAND UPIU
Application Client Buffer Offset	Ignored
Device Server Buffer	Internal to device server
Request Byte Count	Ignored

7.8.4.3 Data-Out Received transport protocol service

A UFS target port uses the Data-Out Received transport protocol service indication to notify a device server that it has received data.

- Data-out Received (IN (I_T_L_Q Nexus, Delivery Result))

Table 7-57 — Data-Out Received transport protocol service

Argument	Implementation
I_T_L_Q Nexus	I_T_L_Q of the corresponding COMMAND UPIU
Delivery Result	DELIVERY SUCCESSFUL: The data was delivered successfully. DELIVERY FAILURE: A UniPro service delivery error occurred while attempting to deliver the data.

7.8.5.3 Data-Out Received transport protocol service (cont'd)

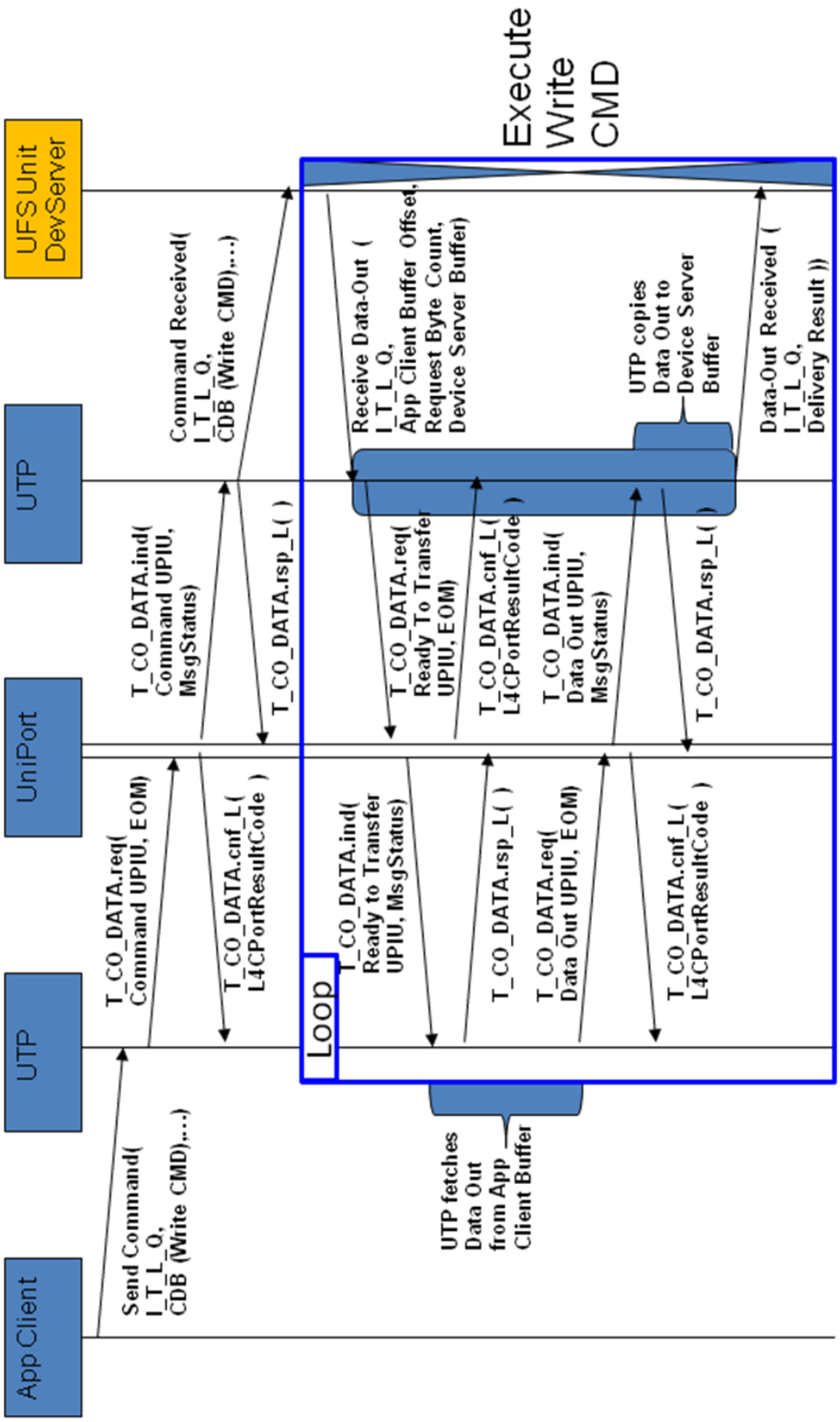

Figure 7-8 — Command + Write Data Phase ½

7.8.5.3 Data-Out Received transport protocol service (cont'd)

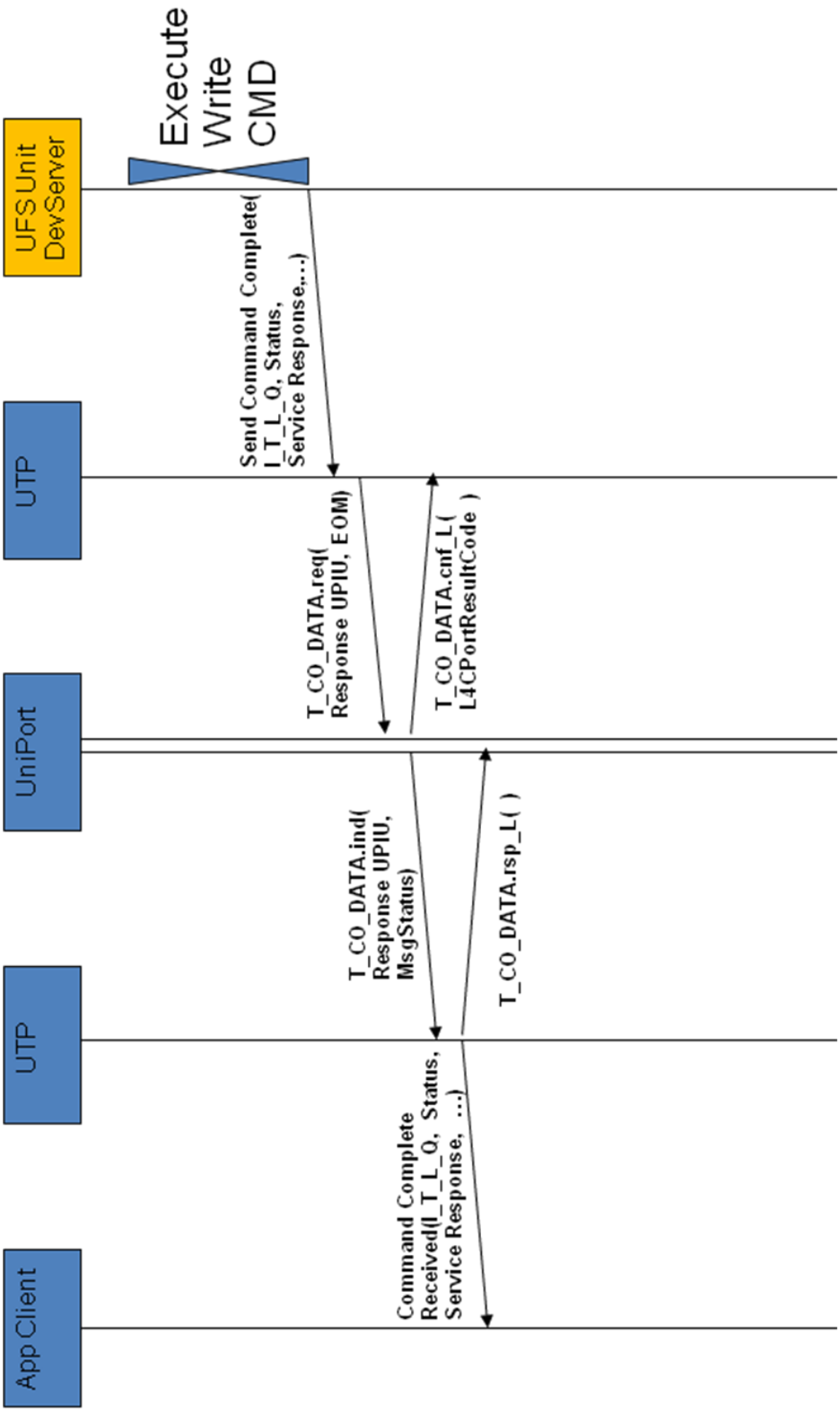

Figure 7-9 — Command + Write Data Phase 2/2

7.8.5.3 Data-Out Received transport protocol service (cont'd)

Task Management Function procedure calls

An application client requests the processing of a task management function by invoking the SCSI transport protocol services:

- Service Response = Function name (IN (Nexus), OUT ([Additional Response Information])

Table 7-58 — Task Management Function procedure calls

Task Management Function (Function name)	Nexus argument
Abort Task	I_T_L_Q Nexus
Abort Task Set	I_T_L Nexus
Clear Task Set	I_T_L Nexus
Logical Unit Reset	I_T_L Nexus
Query Task	I_T_L_Q Nexus
Query Task Set	I_T_L Nexus

One of the following SCSI transport protocol specific service responses shall be returned:

Table 7-59 — SCSI transport protocol service responses

FUNCTION COMPLETE	A task manager response indicating that the requested function is complete. Unless another response is required, the task manager shall return this response upon completion of a task management request supported by the logical unit or SCSI target device to which the request was directed.
FUNCTION SUCCEEDED	A task manager response indicating that the requested function is supported and completed successfully. This task manager response shall only be used by functions that require notification of success (e.g., QUERY TASK, QUERY TASK SET)
FUNCTION REJECTED	A task manager response indicating that the requested function is not supported by the logical unit or SCSI target device to which the function was directed.
INCORRECT LOGICAL UNIT NUMBER	A task router response indicating that the function requested processing for an incorrect logical unit number.
SERVICE DELIVERY OR TARGET FAILURE	The request was terminated due to a service delivery failure SCSI target device malfunction. The task manager may or may not have successfully performed the specified function.

7.8.4.4 ABORT TASK

This function shall be supported by all logical units.

The task manager shall abort the specified command, if it exists. Previously established conditions, including mode parameters, and reservations shall not be changed by the ABORT TASK function.

A response of FUNCTION COMPLETE shall indicate that the command was aborted or was not in the task set. In either case, the SCSI target device shall guarantee that no further requests or responses are sent from the command.

All SCSI transport protocol standards shall support the ABORT TASK task management function.

Service Response = ABORT TASK (IN (I_T_L_Q Nexus))

7.8.4.5 ABORT TASK SET

This function shall be supported by all logical units.

The task manager shall abort all commands in the task set that were received on the specified I_T nexus. Commands received on other I_T nexuses or in other task sets shall not be aborted. This task management function performed is equivalent to a series of ABORT TASK requests.

All pending status and sense data for the commands that were aborted shall be cleared. Other previously established conditions, including mode parameters, and reservations shall not be changed by the ABORT TASK SET function.

All SCSI transport protocol standards shall support the ABORT TASK SET task management function.

Service Response = ABORT TASK SET (IN (I_T_L Nexus))

7.8.4.6 CLEAR TASK SET

This function shall be supported by all logical units.

The task manager shall abort all commands in the task set If the TST field is set to 001b (i.e., per I_T nexus) in the Control mode page (see SPC-4), there is one task set per I_T nexus. As a result, no other I_T nexuses are affected and CLEAR TASK SET is equivalent to ABORT TASK SET.

All pending status and sense data for the task set shall be cleared. Other previously established conditions, including mode parameters, and reservations shall not be changed by the CLEAR TASK SET function.

All SCSI transport protocol standards shall support the CLEAR TASK SET task management function.

Service Response = CLEAR TASK SET (IN (I_T_L Nexus))

7.8.4.7 LOGICAL UNIT RESET

This function shall be supported by all logical units.

Before returning a FUNCTION COMPLETE response, the logical unit shall perform the logical unit reset functions.

All SCSI transport protocol standards shall support the LOGICAL UNIT RESET task management function.

Service Response = LOGICAL UNIT RESET (IN (I_T_L Nexus))

7.8.5 QUERY TASK

UFS transport protocols shall support QUERY TASK.

The task manager in the specified logical unit shall:

1. If the specified command is present in the task set, then return a service response set to FUNCTION SUCCEEDED; or

2. If the specified command is not present in the task set, then return a service response set to FUNCTION COMPLETE.

Service Response = QUERY TASK (IN (I_T_L_Q Nexus))

7.8.5.1 QUERY TASK SET

UFS transport protocols shall support QUERY TASK SET.

The task manager in the specified logical unit shall:

1. If there is any command present in the task set specified I_T nexus, then return a service response set to FUNCTION SUCCEEDED; or

2. If there is no command present in the task set specified I_T nexus, then return a service response set to FUNCTION COMPLETE.

Service Response = QUERY TASK SET (IN (I_T_L Nexus))

7.8.5.2 Task Management SCSI Transport Protocol Services

UFS standard shall define the SCSI transport protocol specific requirements for implementing the Send Task Management Request request, the Task Management Request Received indication, the Task Management Function Executed response, and the Received Task Management Function Executed confirmation SCSI transport protocol services.

A SCSI transport protocol standard may specify different implementation requirements for the Send Task Management Request request SCSI transport protocol service for different values of the Function Identifier argument.

All SCSI initiator devices shall implement the Send Task Management Request request and the Received Task Management Function Executed confirmation SCSI transport protocol services as defined in the applicable SCSI transport protocol standards.

All SCSI target devices shall implement the Task Management Request Received indication and the Task Management Function Executed response SCSI transport protocol services as defined in the applicable SCSI transport protocol standards.

7.8.6.2 Task Management SCSI Transport Protocol Services (cont'd)

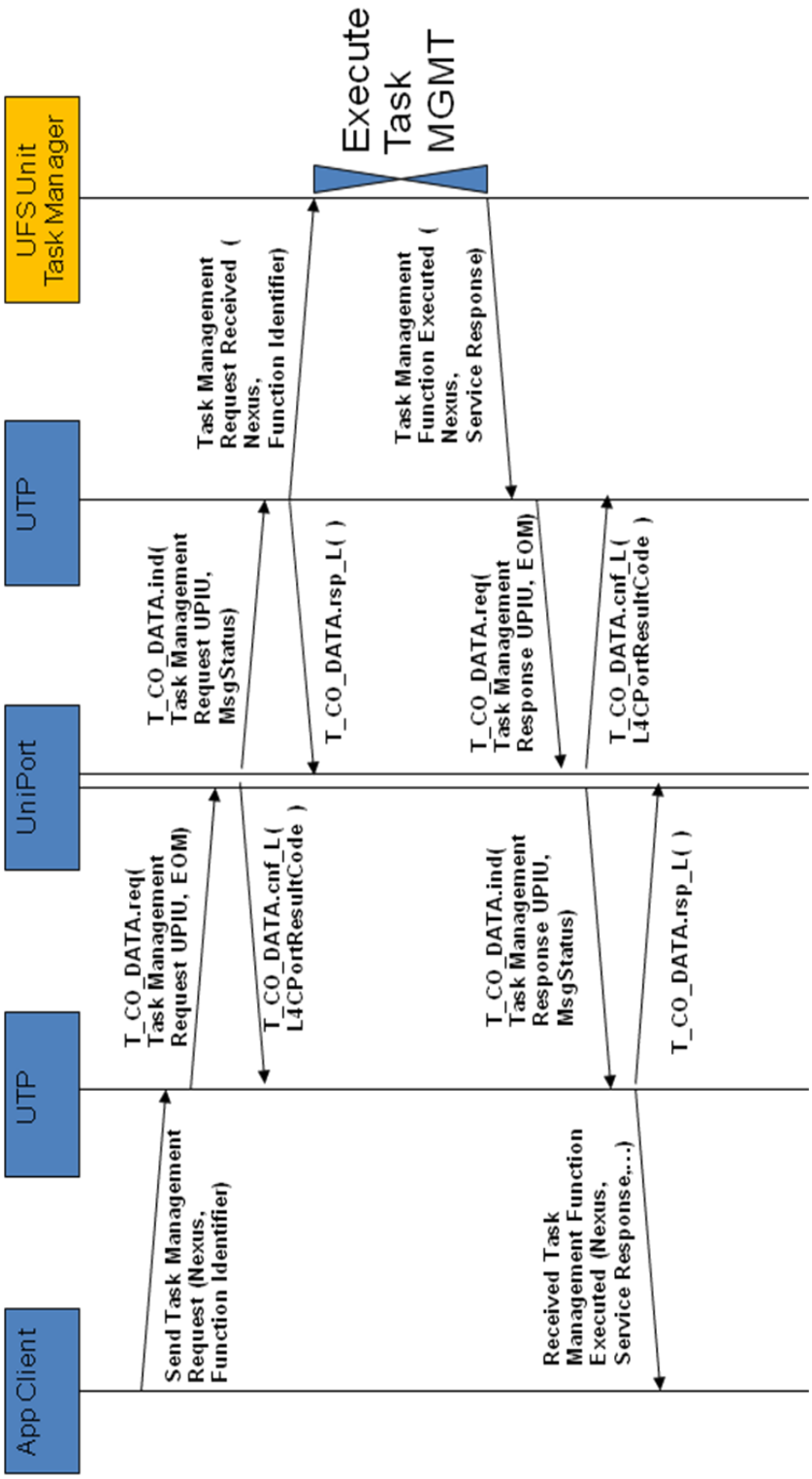

Figure 7-10 — Task Management FunctionSend Task Management Request SCSI transport protocol service request

7.8.6.2 Task Management SCSI Transport Protocol Services (cont'd)

An application client uses the Send Task Management Request SCSI transport protocol service request to request that a SCSI initiator port send a task management function.

Send Task Management Request SCSI transport protocol service request:

Send Task Management Request (IN (Nexus, Function Identifier))

Table 7-60 — Send Task Management Request SCSI transport protocol service request

Argument	Implementation
Nexus	I_T nexus, I_T_L nexus, or I_T_L_Q nexus
Function Identifier:	Argument encoding the task management function to be performed.

7.8.5.3 Task Management Request Received SCSI transport protocol service indication

A task router uses the Task Management Request Received SCSI transport protocol service indication to notify a task manager that it has received a task management function.

Task Management Request Received SCSI transport protocol service indication:

Task Management Request Received (IN (Nexus, Function Identifier))

Table 7-61 — Task Management Request Received SCSI transport protocol service indication

Argument	Implementation
Nexus	I_T nexus, I_T_L nexus, or I_T_L_Q nexus
Function Identifier:	Argument encoding the task management function to be performed.

7.8.5.4 Task Management Function Executed SCSI transport protocol service response

A task manager uses the Task Management Function Executed SCSI transport protocol service response to request that a SCSI target port transmit task management function executed information.

Task Management Function Executed SCSI transport protocol service response:

Task Management Function Executed (IN (Nexus, Service Response, [Additional Response Information]))

7.8.6.4 Task Management Function Executed SCSI transport protocol service response (cont'd)

Table 7-62 — Task Management Function Executed SCSI transport protocol service response

Argument	Implementation
Nexus	I_T nexus, I_T_L nexus, or I_T_L_Q nexus
Service Response	FUNCTION COMPLETE
	FUNCTION SUCCEEDED:
	FUNCTION REJECTED
	INCORRECT LOGICAL UNIT NUMBER
	SERVICE DELIVERY OR TARGET FAILURE
Additional Response Information	The Additional Response Information output argument for the task management procedure call

7.8.5.5 Received Task Management Function Executed SCSI transport protocol service confirmation

A SCSI initiator port uses the Received Task Management Function Executed SCSI transport protocol service confirmation to notify an application client that it has received task management function executed information.

Received Task Management Function Executed SCSI transport protocol service confirmation:

Received Task Management Function Executed (IN (Nexus, Service Response, [Additional Response Information]))

Table 7-63 — Received Task Management Function Executed SCSI transport protocol service confirmation

Argument	Implementation
Nexus	I_T nexus, I_T_L nexus, or I_T_L_Q nexus
Service Response	FUNCTION COMPLETE
	FUNCTION SUCCEEDED:
	FUNCTION REJECTED
	INCORRECT LOGICAL UNIT NUMBER
	SERVICE DELIVERY OR TARGET FAILURE
Additional Response Information	The Additional Response Information output argument for the task management procedure call

7.8.6 Query Function transport protocol services

UFS defines Query Function to get/set UFS-specific device-level registers and parameters (not part of SCSI definition).

7.8.6.1 Send Query Request UFS transport protocol service

An application client uses the Send Query Request UFS transport protocol service request to request that a UFS initiator port send a Query Request function.

Send Query Request UFS transport protocol service request:

Send Query Request(I_T Nexus, Operation: Read|Write, Type: UFS|Vendor, Identifier:Descriptor|Attribute|Flag,Length:n, [Index], [Selector], [WriteData])

Table 7-64 — Send Query Request UFS transport protocol service

Argument	Implementation
Nexus	I_T nexus
Operation	Argument encoding the operation to be performed
Type	Indicates UFS defined or vendor-specific operation
Identifier	Identifier for descriptor type, attribute or flag
Length	Number of bytes to read or write
Index	Index reference for the descriptor, attribute or flag
Selector	Reserved
WriteData	Data to be written in a write query request

7.8.7 Query Function transport protocol services (cont'd)

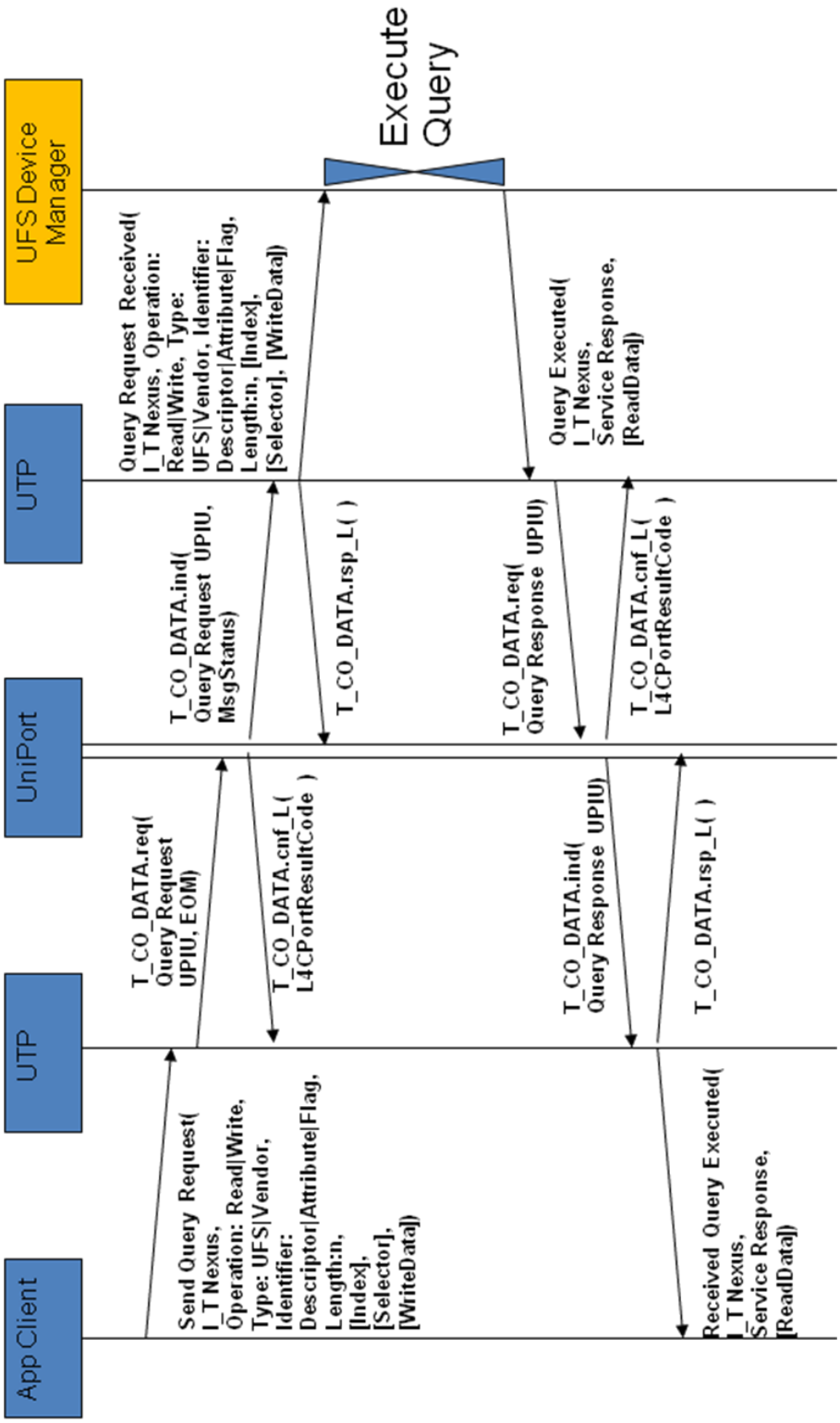

Figure 7-11 — UFS Query Function

7.8.6.2 Query Request Received UFS transport protocol service indication

A UFS target port uses the Query Request Received UFS transport protocol service indication to notify a UFS device manager that it has received a query function.

Query Request Received UFS transport protocol service indication:

Query Request Received(I_T Nexus, Operation: Read|Write, Type: UFS|Vendor, Identifier:Descriptor|Attribute|Flag,Length:n, [Index], [Selector], [WriteData])

Table 7-65 — Query Request Received UFS transport protocol service indication

Argument	Implementation
Nexus	I_T nexus
Operation	Argument encoding the operation to be performed
Type	Indicates UFS defined or vendor-specific operation
Identifier	Identifier for descriptor type, attribute or flag
Length	Number of bytes to read or write
Index	Index reference for the descriptor, attribute or flag
Selector	Reserved
WriteData	Data to be written in a write query request

7.8.6.3 Query Function Executed UFS transport protocol service response

A device manager uses the Query Function Executed UFS transport protocol service response to request that a UFS target port transmit query function executed information.

Query Function Executed UFS transport protocol service response:

Query Executed(I_T Nexus, Service Response,[ReadData])

Table 7-66 — Query Function Executed UFS transport protocol service response

Argument	Implementation
Nexus	I_T nexus
Service Response	FUNCTION SUCCEEDED
	FUNCTION FAILED
ReadData	Data to be returned in a read query request

7.8.6.4 Received Query Function Executed UFS transport protocol service confirmation

A UFS initiator port uses the Received Query Function Executed UFS transport protocol service confirmation to notify an application client that it has received task management function executed information.

Received Query Function Executed UFS transport protocol service confirmation:

Received Query Executed(I_T Nexus, Service Response,[ReadData])

Table 7-67 — Received Query Function Executed UFS transport protocol service confirmation

Argument	Implementation
Nexus	I_T nexus
Service Response	FUNCTION SUCCEEDED
	FUNCTION FAILED
ReadData	Data to be returned in a read query request

8 UFS PROTOCOL LAYER – SCSI COMMANDS

8.1 Universal Flash Storage Command Layer (UCL) Introduction

This chapter defines the mandatory SCSI command set supported by the UFS device. The commands are based on UFS Native Commands and SCSI Primary Commands specification (SPC-4) [SPC] / SCSI Block Commands specification (SBC-3) [SBC].

The UFS native commands (USC) set are defined by JEDEC to support flash storages and UFS native needs bases. The UFS SCSI commands set are based on selection of SCSI SPC and SBC; SCSI Primary Commands specification (SPC-4) and SCSI Block Commands specification (SBC-3). Both commands types share similar command descriptor block (CDB) format.

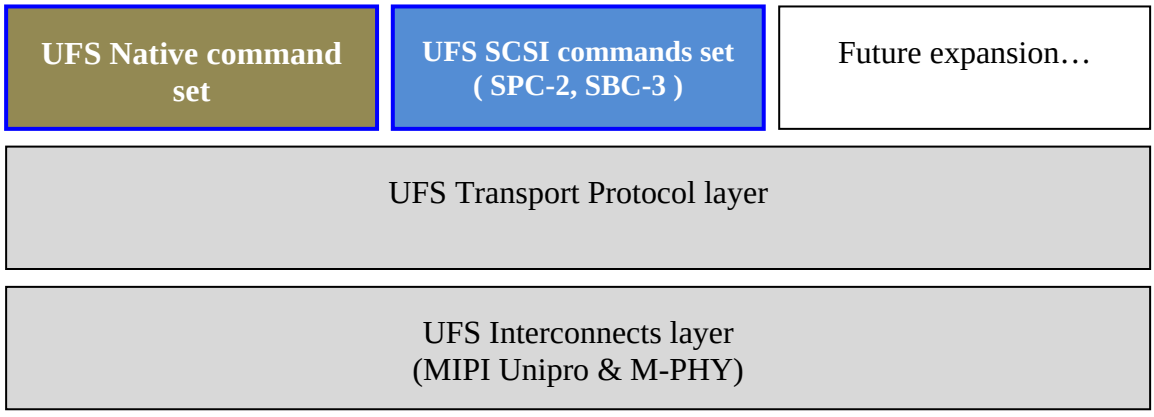

Figure 8-1 – UFS Command Layer

Full supports commands set will be updated with final version based on UFS TG consensus.

8.1.1 The Command Descriptor Block (CDB)

SCSI commands are communicated by sending the SCSI Command Descriptor Block (CDB) to the device. There are only fixed length CDB format for UFS, unlike SCSI which has additional Variable Length CDB format.

All UFS CDBs shall have an OPERATION CODE field as their first byte and these values shall be defined by each SCSI and UFS. Detail SCSI CDB usages and structure are defined in SPC-4, chapter 4.3.Common CDB fields.

The General Common CDB fields are defined in SPC- 4, section 4.3.4 Common CDB fields section.

8.1.1.1 Operation code

The first byte of a SCSI and USC CDB shall contain an operation code identifying the operation being requested by the CDB.

The OPERATION CODE of the CDB contains a GROUP CODE and OPERATION CODE fields. The GROUP CODE field provides for eight groups of command codes and the OPERATION CODE provides 32 command codes in each group.

8.2 Universal Flash Storage native commands (UNC)

UFS Native commands will be defined by JEDEC based on needs based.

8.3 Universal Flash Storage SCSI Commands

The Basic Universal Flash Storage (UFS) SCSI Commands are compatible with SCSI primary commands, SPC-4 and block commands, SBC-3.

Table 8-1 — UFS SCSI Command Set

Command name	OpCode	Command Support	
		Embedded	Removable
INQUIRY	12h	M	M
READ (6)	08h	M	M
READ (10)	28h	M	M
READ (16)	88h	O	O
READ CAPACITY (10)	25h	M	M
READ CAPACITY (16)	25h	O	O
REPORT LUNS	A0h	M	M
START STOP UNIT	1Bh	M	M
TEST UNIT READY	00h	M	M
VERIFY (10)	2Fh	M	M
WRITE (6)	0Ah	M	M
WRITE (10)	2Ah	M	M
WRITE (16)	8Ah	O	O
FORMAT UNIT	04h	M	M
SEND DIAGNOSTIC	1Dh	M	M
SYNCHRONIZE CACHE	35h	M	M
M: mandatory O: optional			

8.3 Universal Flash Storage SCSI Commands (cont'd)

Table 8-2 — UFS SCSI Command Set (additional commands needed for full functionality and SW driver compatibility)

Command name	OpCode	Command Support	
		Embedded	Removable
MODE SELECT (10)	55h	M	M
MODE SENSE (10)	5Ah	M	M
REQUEST SENSE	03h	M	M
UNMAP	42h	M	M
READ BUFFER	3Ch	O	O
WRITE BUFFER	3Bh	O	O
M: mandatory O: optional			

8.3.1 INQUIRY Command

The INQUIRY command (see Table 8-3) is a request for information regarding the logical units and UFS target device be sent to the application client. Refer to [SPC] specification for more details regarding the INQUIRY command.

Table 8-3 — INQUIRY command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (12h)							
1	Reserved						Obsolete	EVPD
2	PAGE CODE							
3	(MSB)	ALLOCATION LENGTH						
4								(LSB)
5	CONTROL							

- Allocation Length = Number of response bytes to return

8.3.1.1 VITAL PRODUCT DATA

When EVPD = 1, the device server shall return the vital product data specified by the PAGE CODE field per SCSI SPC specification definition. Support for vital product data is optional for UFS.

8.3.1.2 STANDARD INQUIRY DATA

When EVPD =0 and Page Code = 0, the Standard INQUIRY DATA is responded to INQUIRY command. The standard INQUIRY data (see **Table 8-4**) shall contain at least 36 bytes. The INQUIRY Data Structure is as followed with the bits specific to UFS highlighted & described.

The INQUIRY command requests that information regarding the logical unit and SCSI target device be sent to the Application Client.

The INQUIRY command may be used by an Application Client after a hard reset or power on condition to determine information about the device for system configuration If a INQUIRY command is received with a pending UNIT ATTENTION condition (i.e., before the device server reports CHECK CONDITION status), the device server shall perform the INQUIRY command

INQUIRY information is returned in standard INQUIRY RESPONSE data structure (described below)

- Client requests number of bytes to return
- First 36 bytes are defined for UFS as standard
- Requesting 0 is OK, 36 returns complete UFS defined record
- Requesting more bytes than defined will result in truncation to max number device has defined

The Device Server should process the INQUIRY command even when an error occurs that prohibits normal command completion

- When in UNIT ATTENTION
- During other conditions that may affect medium access

The Command CDB shall be sent in a single Command UPIU

Table 8-4 — INQUIRY DATA Format

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE = 00h				
1	RMB	Reserved						
2	VERSION							
3	Obsolete	Obsolete	NORMACA	HISUP	RESPONSE DATA FORMAT			
4	ADDITIONAL LENGTH (n-4)							
5	SCCS	ACC	TPGS		3PC	Reserved		PROTECT
6	Obsolete	ENCSERV	VS	MULTIP	Obsolete	Obsolete	Obsolete	ADDR16
7	Obsolete	Obsolete	WBUS16	SYNC	Obsolete	Obsolete	CMDQUE	VS
8	(MSB)	VENDOR IDENTIFICATION						
15								(LSB)
16	(MSB)	PRODUCT IDENTIFICATION						
31								(LSB)
32	(MSB)	**PRODUCT REVISION LEVEL**						
35								(LSB)

8.3.1.3 Inquiry Command Data Response

- Data returned from an INQUIRY command will be transferred to the Application Client in a single DATA IN UPIU
- The Device Server will transfer up to 36 Bytes of Response Data in the Data Segment area of a Data In UPIU
    - Return 36 bytes if Allocation Length in CDB >= 36
    - Return Allocation Length bytes if Allocation Length in CDB < 36
    - An Allocation Length of zero is not an error
        - No data will be returned, go directly to Status phase
- Data will be returned in the indicated Response Data Format described below
- No DATA IN UPIU will be transferred if an error occurs

8.3.1.4 Inquiry Response Data

Table 8-5 — Inquiry Response Data

Byte	Bit	Value	Description
0	7:5	0000b	PERIPHERAL QUALIFIER: 0
0	4:0	00000b	PERIPHERAL TYPE: Direct Access Device
1	7	0b	RMB: Medium not removable
1	6:0	0000000b	RESERVED
2	7:0	06h	VERSION: Conformance to SPC-4
3	7:4	0000b	N/A
3	3:0	0010b	RESPONSE DATA FORMAT: Type 2
4	7:0	31d	ADDITIONAL LENGTH: 31 bytes
5	7:0	00h	N/A
6	7:0	00h	N/A
7	7:2	000000b	N/A
7	1:1	1b	CMDQUE: Support command management (SAM)
7	0:0	0b	N/A
8:15	7:0	ASCII	VENDOR IDENTIFICATION: Left justified (e.g., “Micron “)
16:31	7:0	ASCII	PRODUCT IDENTIFICATION: Left justified (e.g., “UFS MSD M33-X ”)
32:35	7:0	ASCII	PRODUCT REVISION LEVEL: Left justified (e.g., “1.23”)

8.3.1.5 Inquiry Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested data is successfully transferred, the INQUIRY command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure rarely occurs but for few reasons. When the INQUIRY command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - HARDWARE ERROR (hardware failure)
- Will not fail due to a pending UNIT ATTENTION condition

8.3.2 MODE SELECT (10) Command

MODE SELECT command provides a means for the application client to specify medium, logical unit, or peripheral device parameters to the device server.

- Parameters are managed by means of parameter pages called Mode Pages
    - UFS will support specific pages
        - CONTROL, CACHING, READ-WRITE ERROR RECOVERY, VENDOR
- Writes parameters to one or more parameter pages in a list
    - The Application Client can specify a single, multiple or all supported pages in a single command
- Complementary command to the MODE SENSE command

The Command CDB shall be sent in a single Command UPIU

Table 8-6 — MODE SELECT (10) Command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (55h)							
1	Reserved			PF=1	Reserved			SP
2								
3								
4				Reserved				
5								
6								
7	PARAMETER LIST LENGTH							
8								
9	CONTROL							

8.3.2.1 Mode Select Command Parameters

Table 8-7 — Mode Select Command Parameters

Byte	Bit	Description
1	4:4	**PF**: PAGE FORMAT. A page format bit set to zero specifies that all parameters after the block descriptors are vendor specific. A PF bit set to one specifies that the MODE SELECT parameters following the header and block descriptor(s) are structured as pages of related parameters as defined in the SCSI standard.
1	0:0	**SP**: SAVE PAGES. A save pages (SP) bit set to zero specifies that the device server shall perform the specified MODE SELECT operation, and shall not save any mode pages. If the logical unit implements no distinction between current and saved mode pages and the SP bit is set to zero, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST. An SP bit set to one specifies that the device server shall perform the specified MODE SELECT operation, and shall save to a nonvolatile vendor specific location all the saveable mode pages including any sent in the Data-Out Buffer. Individual Mode pages that are saved are specified by the saveable parameter bit (PS) that is returned in the first byte of each mode page when read via the MODE SENSE command. If the PS bit is set to one in the MODE SENSE data, then the mode page shall be saveable when issuing a MODE SELECT command with the SP bit set to one. If the logical unit does not implement saved pages and the SPI bit is set to one, the command shall terminate with a CHECK CONDITION status and a sense key set to ILLEGAL REQUEST.
7:8	7:0	**PARAMETER LIST LENGTH**: Specifies length in bytes of the mode parameter list that the Application Client buffer will transfer to the Device Server. A PARAMETER LIST LENGTH of zero specifies that no data shall be transferred, this shall not be considered an error.

8.3.2.2 Mode Select Command Data Transfer

The Device Server will request transfer PARAMETER LIST LENGTH number of bytes from a buffer in the Application Client by issuing a one or more Ready To Transfer UPIU's (RTT) followed by a Data Out UPIU per RTT and will subsequently write those to the appropriate Mode Page area within the Device Server

- The data transferred from the Application Client will be contained within the Data Segment of the Data Out UPIU
- After delivery the Device Server will sort through the parameter list to identify the individual mode pages

Zero or an incomplete number of RTT's and DATA OUT UPIU's could be requested if a write error occurs before the entire data transfer is complete

The Parameter List will be formatted as indicated by the Mode Parameter Layout

- Header (8 bytes)
- Block Descriptor (0 bytes for UFS)
- Page(s) Data (N bytes, dependent up number of pages sent)

8.3.2.3 Mode Select Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested data is successfully transferred and written, the MODE SELECT command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the MODE SELECT command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (CDB or parameter errors)
    - MEDIUM ERROR (medium failure, ecc, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset or power-on)

8.3.3 MODE SENSE (10) Command

The MODE SENSE command provides a means for a Device Server to report parameters to an Application Client

- Parameters are managed by means of parameter pages called Mode Pages
    - UFS will support specific pages
        - CONTROL, BACKGROUND CONTROL, CACHING, POWER CONDITION, VENDOR
- Reads parameter pages in a list
    - The Application Client may request any one or all of the supported pages from the Device Server
    - If all pages requested, they will be returned in ascending page order
- Mode Sense returns Device Specific Parameter in header
    - IMPORTANT: Indicates active Write Protected (WP) indication for Direct Access Device
- Complementary command to the MODE SELECT command

The Command CDB shall be sent in a single Command UPIU

8.3.3 MODE SENSE (10) Command (cont'd)

Table 8-8 — MODE SENSE (10) Command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (5Ah)							
1	Reserved (000b)			0b	DBD (1b)	Reserved (000b)		
2	PC		PAGE CODE					
3	SUBPAGE CODE							
4	Reserved							
5								
6								
7	ALLOCATION LENGTH							
8								
9	CONTROL = 00h							

8.3.3.1 Mode Sense Command Parameters

Table 8-9 — Mode Sense Command Parameters

Byte	Bit	Description
2	7:6	**PC**: The page control (PC) field specifies the type of mode parameter values to be returned in the mode pages. See table below for definition.
2	5:0	**PAGE CODE**: Specifies which mode page to return. The Page and Subpage code specify the page to return.
3	7:0	**SUBPAGE CODE**: Specifies which subpage mode page to return
7:8	7:0	**ALLOCATION LENGTH**: Specifies the maximum number of bytes the Device Server will transfer to the Application Client

8.3.3.2 Page Control Function

Table 8-10 — Page Control Function

Page Control (PC)	Description	Use
00b	Current Values	Return the current operational mode page parameter values. Must be supported.
01b	Changeable Values	Return a bit mask denoting values that are changeable. Changeable bits in the mode parameters will have a value of '1', non-changeable will have a value of '0'. Return ILLEGAL REQUEST Sense Key if not supported.
10b	Default Values	Return the default values of the mode parameters. Must be supported.
11b	Saved Values	Return the saved values of the mode parameters. Return ILLEGAL REQUEST Sense Key if not supported.

- UFS shall be required to support PC = 00b and PC = 10b
- Other formats are implementation dependent

8.3.3.3 Mode Sense Command Data Transfer

The Device Server will transfer up to Allocation Length number of data bytes of Mode Parameter Data to the Application Client.

- Less than Allocation Length will be transferred if Device Server contains less bytes

The Device Server will transfer the data as indicated by the Mode Parameter Layout

- Header (8 bytes)
- Block Descriptor (0 bytes for UFS)
- Page Data (N bytes, dependent up page requested)

Data will be transferred from the Device Server to the Application Client via a series of Data In UPIU's

- The data transferred from the Device Server will be contained within the Data Segment of the Data In UPIU

Zero or an incomplete number of DATA IN UPIU's will be transferred if an error occurs before the entire data transfer is complete.

8.3.3.4 Mode Sense Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested data is successfully transferred and written, the MODE SENSE command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure occurs when a requesting a page or subpage that is not supported. A STATUS response of CHECK CONDITION will be returned with a SENSE KEY indicating ILLEGAL REQUEST
- Failure can occur for numerous reasons. When the MODE SENSE command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (CDB errors)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset or power-on)

8.3.4 READ (6) Command

The READ (6) command (see Table 8-11) requests that the Device Server read from the medium the specified number of logical block(s) and transfer them to the Application Client.

The Command CDB shall be sent in a single Command UPIU.

Table 8-11 — READ (6) UFS Command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (08h)							
1	Reserved			(MSB)				
2	LOGICAL BLOCK ADDRESS							
3								(LSB)
4	TRANSFER LENGTH							
5	CONTROL = 00h							

8.3.4.1 Read (6) Command Parameters

- LOGICAL BLOCK ADDRESS: Address of first block
- TRANSFER LENGTH: Number of contiguous logical blocks of data that shall be read and transferred. A transfer length of zero specifies that 256 logical blocks shall be read. This condition shall not be considered an error.

8.3.4.2 Read (6) Command Data Transfer

- The Device Server will read the specified logical block(s) from the medium and transfer them to the Application Client in a series of Data In UPIU's
- Zero or an incomplete number of DATA IN UPIU's could be transferred if a read error occurs before the entire data transfer is complete

8.3.4.3 Read (6) Command Status Response

- Status response will be sent in a single RESPONSE UPIU
- If all requested data is successfully read and transferred, the READ command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the READ command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ecc, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset or power-on)
    - Etc.

8.3.5 READ (10) Command

The READ (10) command (see Table **8-12**) requests that the Device Server read from the medium the specified number of logical block(s) and transfer them to the Application Client

The Command CDB shall be sent in a single Command UPIU

Table 8-12 — READ (10) UFS Command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (28h)							
1	RDPROTECT (0)			DPO (0)	FUA	Reserved	FUA_NV (0)	Obsolete
2	(MSB)							
3				LOGICAL BLOCK ADDRESS				
4								
5								(LSB)
6	Reserved			GROUP NUMBER (0)				
7	(MSB)							
8				TRANSFER LENGTH				(LSB)
9	CONTROL = 00h							

- The RDPROTECT, FUA_NV and GROUP NUMBER files are set to '0' for UFS.
- DPO set to '0' for UFS.

8.3.5.1 Read (10) Command Parameters

- FUA: Force Unit Access
  '0' = The Device Server may read the logical blocks from the cache and/or the medium.
  '1' = The Device Server shall read the logical blocks from the medium. If a cache contains a more recent version of a logical block, then the device server shall write the logical block to the medium before reading it.
- LOGICAL BLOCK ADDRESS: Address of first block
- TRANSFER LENGTH: Number of contiguous logical blocks of data that shall be read and transferred. A transfer length of zero specifies that no logical blocks will be read. This condition shall not be considered an error.

8.3.5.2 Read (10) Command Data Transfer

- The Device Server will read the specified logical block(s) from the medium and transfer them to the Application Client in a series of Data In UPIU's
- Zero or an incomplete number of DATA IN UPIU's could be transferred if a read error occurs before the entire data transfer is complete

8.3.5.3 Read (10) Command Status Response

- Status response will be sent in a single RESPONSE UPIU
- If all requested data is successfully read and transferred, the READ command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the READ command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
  - ILLEGAL REQUEST (range or CDB errors)
  - MEDIUM ERROR (medium failure, ecc, etc.)
  - HARDWARE ERROR (hardware failure)
  - UNIT ATTENTION (unexpected reset or power-on)
  - Etc.

8.3.6 READ (16) Command

The READ (16) command (see Table 8-11) requests that the Device Server read from the medium the specified number of logical block(s) and transfer them to the Application Client

The Command CDB shall be sent in a single Command UPIU

Table 8-13 — READ (16) UFS Command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (88h)							
1	RDPROTECT (0)			DPO (0)	FUA	Reserved	FUA_NV (0)	Reserved
2	(MSB)							
….	LOGICAL BLOCK ADDRESS							
9								(LSB)
10	(MSB)							
….	TRANSFER LENGTH							
13								(LSB)
14	Ignore	Reserved		GROUP NUMBER				
15	CONTROL = 00h							

- The RDPROTECT, FUA_NV and GROUP NUMBER files are set to ‘0’ for UFS.
- DPO set to ‘0’ for UFS.

8.3.6.1 Read (16) Command Parameters

- **FUA**: Force Unit Access
  ‘0’ = The Device Server may read the logical blocks from the cache and/or the medium. ‘1’ = The Device Server shall read the logical blocks from the medium. If a cache contains a more recent version of a logical block, then the device server shall write the logical block to the medium before reading it.
- **LOGICAL BLOCK ADDRESS**: Address of first block
- **TRANSFER LENGTH**: Number of contiguous logical blocks of data that shall be read and transferred. A transfer length of zero specifies that no logical blocks will be read. This condition shall not be considered an error.

8.3.6.2 Read (16) Command Data Transfer

- The Device Server will read the specified logical block(s) from the medium and transfer them to the Application Client in a series of Data In UPIU's
- Zero or an incomplete number of DATA IN UPIU's could be transferred if a read error occurs before the entire data transfer is complete

8.3.6.3 Read (16) Command Status Response

- Status response will be sent in a single RESPONSE UPIU
- If all requested data is successfully read and transferred, the READ command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the READ command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ecc, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset or power-on)
    - Etc.

8.3.7 READ CAPACITY (10) Command

The READ CAPACITY (10) command provides a means for the UFS host to request the current capacity of the UFS device. Refer to **SBC-3** specification for more details regarding the READ CAPACITY (10) command.

- The READ CAPACITY (10) command requests that the device server transfer 8 bytes of parameter data describing the capacity and medium format of the direct-access block device
- Use to dynamically find number and size of addressable blocks (sectors) on the medium
- Can be issued per Logical Unit if each LU maintains an independent addressable space

The Command CDB shall be sent in a single Command UPIU

Table 8-14 — READ CAPACITY (10)

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (25h)							
1	Reserved							Obsolete
2	(MSB)							
3	LOGICAL BLOCK ADDRESS							(LSB)
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							PMI
9	CONTROL = 00h							

- PMI = 0 for UFS
- Logical Block Address = 0 for UFS

8.3.7.1 Read Capacity (10) Data Response

- Data returned from a READ CAPACITY (10) command will be transferred to the Application Client in a single DATA IN UPIU
- The Device Server will transfer 8 Bytes of Capacity Data in the Data Segment area of a Data In UPIU
- Data will be returned in the indicated
  Read Capacity Parameter format described below
- No DATA IN UPIU will be transferred if an error occurs

8.3.7.2 Read Capacity (10) Parameter Data

Table 8-15 — Read Capacity (10) Parameter Data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				RETURNED LOGICAL BLOCK ADDRESS				
2								
3								(LSB)
4	(MSB)							
5				LOGICAL BLOCK LENGTH				
6								
7								(LSB)

- Returned Logical Block Address = Last addressable block on medium under control of logical unit (LU)
    - If the number of logical blocks exceeds the maximum value that is able to be specified in the RETURNED LOGICAL BLOCK ADDRESS field, then the device server shall set the RETURNED LOGICAL BLOCK ADDRESS field to FFFF_FFFFh. The application client should then issue a READ CAPACITY (16) command to request that the device server transfer the READ CAPACITY (16) parameter data to the data-in buffer.
- Logical Block Length = size of block in bytes
    - For UFS minimum size shall be 512 bytes

8.3.7.3 Read Capacity (10) Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested data is successfully transferred, the READ CAPACITY command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for a number of reasons. When the READ CAPACITY command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset, power-on, media changed)
    - Etc.

8.3.8 READ CAPACITY (16) Command

The READ CAPACITY (16) command provides a means for the UFS host to request the current capacity of the UFS device. Refer to **SBC-3** specification [SBC] for more details regarding the READ CAPACITY (16) command.

- The READ CAPACITY (16) command requests that the device server transfer 32 bytes of parameter data describing the capacity and medium format of the direct-access block device
- Use to dynamically find number and size of addressable blocks (sectors) on the medium
- Can be issued per Logical Unit if each LU maintains an independent addressable space

The Command CDB shall be sent in a single Command UPIU

Table 8-16 — READ CAPACITY (16)

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (10h)				
2	(MSB)							
….	LOGICAL BLOCK ADDRESS							
9								(LSB)
10	(MSB)							
….	ALLOCATION LENGTH							
13								(LSB)
14	Reserved							PMI
15	CONTROL = 00h							

- PMI = 0 for UFS
- Logical Block Address = 0 for UFS
- Allocation Length = the maximum number of bytes that the application client has allocated
- for the returned parameter data.

8.3.8.1 Read Capacity (16) Data Response

- Data returned from a READ CAPACITY (16) command will be transferred to the Application Client in a single DATA IN UPIU
- The Device Server will transfer 8 Bytes of Capacity Data in the Data Segment area of a Data In UPIU
- Data will be returned in the indicated
  Read Capacity Parameter format described below
- No DATA IN UPIU will be transferred if an error occurs

8.3.8.2 Read Capacity (16) Parameter Data

Table 8-17 — Read Capacity (16) Parameter Data

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
….		RETURNED LOGICAL BLOCK ADDRESS						
7								(LSB)
8	(MSB)							
….		LOGICAL BLOCK LENGTH						
11								(LSB)
12	Reserved				P_TYPE			PROT_EN
13	P_I_EXPONENT				LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT			
14	TPE	TPRZ	(MSB)	LOWEST ALIGNED LOGICAL BLOCK ADDRESS				
15								(LSB)
16								
….		Reserved						
31								

- Returned Logical Block Address = Last addressable block on medium under control of logical unit (LU)
- Logical Block Length = size of block in bytes
    - For UFS minimum size shall be 512 bytes
- P_TYPE, PROT_EN are set to ‘0’ for UFS
- The P_I_EXPONENT field can be ignored for PROT_EN = ‘0’
- Logical Blocks per Physical Block Exponent (vendor-specific information)
    - 0: one or more physical blocks per logical block
    - n>0: 2^n logical blocks per physical block
- TPE is set to ‘0’ if PROVISIONING TYPE parameter in UFS Configuration Descriptor is set to 0b00000000. TPE is set to ‘1’ if PROVISIONING TYPE is set to 0b00000010 or 0b00000011.
- TPRZ value is set by PROVISIONING TYPE parameter in UFS Configuration Descriptor. If the thin provisioning read zeros (TPRZ) bit is set to one, then, for an unmapped LBA specified by a read operation, the device server shall send user data with all bits set to zero to the data in buffer. If the TPRZ bit is set to zero, then, for an unmapped LBA specified by a read operation, the device server shall send user data with all bits set to any value to the data in buffer. Lowest Aligned Logical Block Address indicates the LBA of the first logical block that is located at the beginning of a physical block (vendor-specific information)

8.3.8.3 Read Capacity (16) Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested data is successfully transferred, the READ CAPACITY command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for a number of reasons. When the READ CAPACITY command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset, power-on, media changed)
    - Etc.

8.3.9 START STOP UNIT

The START STOP UNIT command requests that the device server change the power condition of the logical unit or load or eject the medium.

- Enable or disable the direct-access block device for medium access operations by controlling power

The Command CDB shall be sent in a single Command UPIU

Table 8-18 — START STOP UNIT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Bh)							
1	Reserved							IMMED
2	Reserved				Power Condition Modifier			
3	Reserved							
4	POWER CONDITIONS				Reserved	NO_FLUSH	LOEJ	START
5	CONTROL = 00h							

IMMED = 0 : Return STATUS (Response UPIU) after operation is completed

IMMED = 1 : Return STATUS after CDB is decoded

8.3.9.1 START STOP UNIT Parameters

Power Condition Modifier shall be set to ‘0’ (reserved) in UFS spec.

Use of the other parameters is defined in the Power Mode chapter.

8.3.9.2 Start Stop Unit Data Response

- The START STOP UNIT command does not have a data phase
- No DATA IN or DATA OUT UPIU’s are transferred

8.3.9.3 Start Stop Unit Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If IMMED=1 in the CDB, the STATUS response will be sent to the Application Client before the device operations have been completed
    - Usually used for shutting down
- If the requested operation is successful, the START STOP UNIT command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for few reasons. When the START STOP UNIT command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION

8.3.10 TEST UNIT READY Command

The TEST UNIT READY command provides a means to check if the logical unit is ready. This is not a request for a self-test.

- If the logical unit is able to accept an appropriate medium-access command without returning CHECK CONDITION status, this command shall return a GOOD status.
- If the logical unit is unable to become operational or is in a state such that an Application Client action (e.g., START UNIT command) is required to make the logical unit ready, the command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY (02H).

The Command CDB shall be sent in a single Command UPIU

Table 8-19 — TEST UNIT READY command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (00h)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved							
5	CONTROL = 00h							

8.3.10.1 Test Unit Read Data Response

- The TEST UNIT READY command does not have a data response
- No DATA IN or DATA OUT UPIU’s are transferred

8.3.10.2 Test Unit Read Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the command succeeds without error, the TEST UNIT READY command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the TEST UNIT READY command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (CDB errors)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset or power-on)
    - Etc.

8.3.11 REPORT LUNS Command

The REPORT LUNS command requests that the peripheral device logical unit inventory be sent to the Application Client.

- The logical unit inventory is a list that shall include the logical unit numbers of all logical units accessible to a UFS Application Client
- If a REPORT LUNS command is received with a pending unit attention condition (i.e., before the device server reports CHECK CONDITION status), the device server shall perform the REPORT LUNS command

The Command CDB shall be sent in a single Command UPIU

Table 8-20 — REPORT LUNS command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A0h)							
1	Reserved							
2	SELECT REPORT							
3	(MSB)	Reserved						
5								(LSB)
6	(MSB)	ALLOCATION LENGTH						
9								(LSB)
10	Reserved							
11	CONTROL							

8.3.11.1 Report LUNS Command Parameters

Table 8-21 — Report LUNS Command Parameters

Byte	Bit	Description
2	7:0	**SELECT REPORT**: Specifies the type of logical unit addresses that shall be reported. The report types available are listed in the table below. UFS will support all report types.
6:9	7:0	**ALLOCATION LENGTH**: Specifies the maximum number of bytes of buffer space that the Application Client has allocated for data reception.

8.3.11.2 Report LUNS Command Select Report Field Values

Table 8-22 — Report LUNS Command Select Report Field Values

REPORT CODE	DESCRIPTION
00h	The list shall contain all accessible logical units using the single level LUN structure using the peripheral device addressing method, if any. If there are no logical units, the LUN LIST LENGTH field shall be zero.
01h	The list shall contain all accessible well known logical units, if any using the well known logical unit extended addressing format. If there are no well known logical units, the LUN LIST LENGTH field shall be zero.
02h	The list shall contain all accessible logical units using their respective addressing format.
03h-FFh	Reserved

8.3.11.3 Report LUNS Data Response

- Data returned from a REPORT LUNS command will be transferred to the Application Client in a one or more DATA IN UPIU's
- Most likely one DATA IN UPIU
- The Device Server will transfer less than or equal to Allocation Length data bytes to the Application Client.
- Less if Device Server has less total data than requested
- Data will be returned in the indicated Parameter Data Format described below
- Each reportable logical unit will produce 8 bytes of data in a format described below

8.3.11.4 Report LUNS Parameter Data Format

Table 8-23 — Report LUNS Parameter Data Format

Byte	Description
0:3	**LUN LIST LENGTH**: Length = N – 7. This field shall contain the length in bytes of the LUN list that is available to be transferred. The LUN list length is the number of logical unit numbers in the logical unit inventory multiplied by eight.
4:7	**RESERVED**: 00000000h
8:15	**FIRST LUN RECORD**: 8 byte record that contains the last LUN
…	… next LUN record …
N-7:N	**LAST LUN RECORD:** 8 byte record that contains the last LUN

- Total List Length = LUN LIST LENGTH + 8, (8*number of LUN's + 8)
- Each LUN record is 8 bytes in length
- UFS uses two formats:
    - The single level LUN structure using peripheral device addressing method
    - The well known logical unit extended addressing format

8.3.11.5 Report LUNS LUN Addressing Formats

Table 8-24 — Single level LUN structure using peripheral device addressing method

Peripheral Device Addressing Method								
Byte	7	6	5	4	3	2	1	0
0	00b		000000b					
1	LUN							
2	NULL (0000h)							
3								
4	NULL (0000h)							
5								
6	NULL (0000h)							
7								

- Format used for standard Logical Unit addressing
- LUN = Logical Unit Number
    - For UFS: 00h <= LUN <= 7Fh and WLUN_ID bit = ‘0’

Table 8-25 — Well Known Logical Unit Extended Addressing Format

Well Known Logical Unit Extended Addressing Format								
Byte	7	6	5	4	3	2	1	0
0	11b		00b		0001b			
1	W-LUN							
2	NULL (0000h)							
3								
4	NULL (0000h)							
5								
6	NULL (0000h)							
7								

- Format used for Well Know Logical Unit addressing
- W-LUN = Well Known Logical Unit Number
    - For UFS: 00h <= W-LUN <= 7Fh and WLUN bit = ‘1’

8.3.11.6 Report LUNS Status Response

- Status response will be sent in a single RESPONSE UPIU
- If all requested data is successfully read and transferred, the REPORT LUNS command will terminate with a STATUS response of GOOD
- The REPORT LUNS command will succeed when a pending UNIT ATTENTION condition exists
- Failure can occur for very few reasons, mainly for illegal values in the CDB. When the REPORT LUNS command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (CDB errors)

8.3.11.7 UFS LUNS Format

Table 8-26 — UFS LUNS Format

7	6	5	4	3	2	1	0
WLUN_ID	UNIT_NUMBER_ID						

The UFS 8-bit LUN number field supports two types of LUN addressing:

- If WLUN_ID bit = ‘0’ then the UNIT_NUMBER_ID field addresses a standard logical unit (LUN)
- If WLUN_ID bit = ‘1’ then the UNIT_NUMBER_ID field addresses a WELL KNOW LOGICAL UNIT (W-LUN)

Up to 128 LUN’s and up to 128 W-LUN’s

- 0 <= UNIT_NUMBER_ID <= 127
- 0 <= UNIT_NUMBER_ID <= 127

Table 8-27 defines the logical unit number for UFS well know logical units (WLUN_ID bit set to ‘1’)

Table 8-27 — Well know logical unit numbers

Well know logical unit	WLUN_ID	UNIT_NUMBER_ID	Logical Unit Number
REPORT LUNS	1b	01h	81h
UFS Device	1b	50h	D0h
RPMB	1b	44h	C4h
BOOT	1b	30h	B0h

The UFS 8-bit unit number field in the UPIU will be converted into the 64-bit SCSI SAM-5 unit number format as described:

- UFS LUN PQh → SAM LUN: 00h PQh 00h 00h 00h 00h 00h 00h
    - Example: UFS LUN 06h = SAM LUN 0006000000000000h
- UFS W-LUN: XYh .OR.80h → SAM LUN: C1h XYh 00h 00h 00h 00h 00h 00h
    - Example: UFS WLUN 81h (01h .OR.80h) = SAM LUN: C101000000000000h

8.3.12 VERIFY (10)

The VERIFY command requests that the UFS device verify that the Device Server verify that the specified logical block(s) and range on the medium can be accessed.

- Logical units that contain cache shall write referenced cached logical blocks to the medium for the logical unit before verification

The Command CDB shall be sent in a single Command UPIU

Table 8-28 — VERIFY Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Fh)							
1	Reserved							
2	(MSB)							
3				LOGICAL BLOCK ADDRESS				
4								
5								(LSB)
6	Reserved							
7	(MSB)							
8				VERIFICATION LENGTH				(LSB)
9	CONTROL							

8.3.12.1 Verify Command Parameters

Table 8-29 — Verify Command Parameters

Byte	Bit	Description
2:5	7:0	**LOGICAL BLOCK ADDRESS**: Address of first block
7:8	7:0	**VERIFICATION LENGTH**: Number of contiguous logical blocks of data that shall be transferred and written. A transfer length of zero specifies that no logical blocks will be written. This condition shall not be considered an error.

8.3.12.2 Verify Command Data Transfer

The VERIFY command does not have a data response

- No DATA IN or DATA OUT UPIU’s are transferred

8.3.12.3Verify Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If all requested data is successfully verified against the medium, the VERIFY command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- If a byte-by-byte comparison is unsuccessful for any reason, then the Device Server shall terminate the command with CHECK CONDITION status with the Sense Key set to MISCOMPARE
- Other failures can occur for numerous reasons. When the WRITE command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
- ILLEGAL REQUEST (range or CDB errors)
- MEDIUM ERROR (medium failure, ecc, etc.)
- HARDWARE ERROR (hardware failure)
- UNIT ATTENTION (unexpected reset or power-on)
- Etc.

8.3.13 WRITE (6) Command

The WRITE (6) UFS command requests that the Device Server transfer the specified number of logical blocks(s) from the Application Client and write them to the medium

The Command CDB shall be sent in a single Command UPIU

Table 8-30 — WRITE (6) Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (0Ah)							
1	Reserved			(MSB)				
2	LOGICAL BLOCK ADDRESS							
3								(LSB)
4	TRANSFER LENGTH							
5	CONTROL = 00h							

8.3.13.1 Write (6) Command Parameters

- LOGICAL BLOCK ADDRESS: Address of first block
- TRANSFER LENGTH: Number of contiguous logical blocks of data that shall be transferred and written. A transfer length of zero specifies that no logical blocks will be written. This condition shall not be considered an error.

8.3.13.2 Write (6) Command Data Transfer

The Device Server will transfer the specified logical block(s) from the Application Client by issuing a series of Ready To Transfer UPIU's (RTT) followed by a Data Out UPIU per RTT and will subsequently write them

Zero or an incomplete number of RTT's and DATA OUT UPIU's could be requested if a write error occurs before the entire data transfer is complete.

8.3.13.3 Write (6) Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If all requested data is successfully transferred and written, the WRITE command will terminate with a STATUS response of GOOD
- If the logical blocks are transferred directly to a cache then the Device Server may complete the command with a GOOD status prior to writing the logical blocks to the medium
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the WRITE command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ecc, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset or power-on)
    - Etc.

8.3.14 WRITE (10) Command

The WRITE (10) UFS command requests that the Device Server transfer the specified number of logical blocks(s) from the Application Client and write them to the medium

The Command CDB shall be sent in a single Command UPIU

Table 8-31 — WRITE (10) Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Ah)							
1	WRPROTECT (0)			DPO (0)	FUA	Reserved	FUA_NV (0)	Obsolete
2	(MSB)							
3				LOGICAL BLOCK ADDRESS				
4								
5								(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)							
8				TRANSFER LENGTH				(LSB)
9	CONTROL = 00h							

- The WDPROTECT, FUA_NV and GROUP NUMBER files are set to ‘0’ for UFS.
- DPO set to ‘0’ for UFS.

8.3.14.1 Write (10) Command Parameters

- **FUA**: Force Unit Access – ‘0’ = The Device Server shall write the logical blocks to the cache and/or the medium. ‘1’ = The Device Server shall write the logical blocks to the medium, and shall not complete the command with GOOD status until all the logical blocks have been written on the medium without error.
- **LOGICAL BLOCK ADDRESS**: Address of first block
- **TRANSFER LENGTH**: Number of contiguous logical blocks of data that shall be transferred and written. A transfer length of zero specifies that no logical blocks will be written. This condition shall not be considered an error.

8.3.14.2 Write(10) Command Data Transfer

The Device Server will transfer the specified logical block(s) from the Application Client by issuing a series of Ready To Transfer UPIU's (RTT) followed by a Data Out UPIU per RTT and will subsequently write them

Zero or an incomplete number of RTT's and DATA OUT UPIU's could be requested if a write error occurs before the entire data transfer is complete

8.3.14.3 Write (10) Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If all requested data is successfully transferred and written, the WRITE command will terminate with a STATUS response of GOOD
- If the logical blocks are transferred directly to a cache then the Device Server may complete the command with a GOOD status prior to writing the logical blocks to the medium
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the WRITE command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ecc, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset or power-on)
    - Etc.

8.3.15 WRITE (16) Command

The WRITE (16) UFS command requests that the Device Server transfer the specified number of logical blocks(s) from the Application Client and write them to the medium.

The Command CDB shall be sent in a single Command UPIU.

Table 8-32 — WRITE (16) Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Ah)							
1	WRPROTECT (0)			DPO (0)	FUA	Reserved	FUA_NV (0)	Reserved
2	(MSB)							
….	LOGICAL BLOCK ADDRESS							
9								(LSB)
10	(MSB)							
….	TRANSFER LENGTH							
13								(LSB)
14	Ignore	Reserved		GROUP NUMBER				
15	CONTROL = 00h							

- The WDPROTECT, FUA_NV and GROUP NUMBER files are set to ‘0’ for UFS.
- DPO set to ‘0’ for UFS.

8.3.15.1 Write (16) Command Parameters

- **FUA**: Force Unit Access – ‘0’ = The Device Server shall write the logical blocks to the cache and/or the medium. ‘1’ = The Device Server shall write the logical blocks to the medium, and shall not complete the command with GOOD status until all the logical blocks have been written on the medium without error.
- **LOGICAL BLOCK ADDRESS**: Address of first block
- **TRANSFER LENGTH**: Number of contiguous logical blocks of data that shall be transferred and written. A transfer length of zero specifies that no logical blocks will be written. This condition shall not be considered an error.

8.3.15.2 Write (16) Command Data Transfer

The Device Server will transfer the specified logical block(s) from the Application Client by issuing a series of Ready To Transfer UPIU's (RTT) followed by a Data Out UPIU per RTT and will subsequently write them

Zero or an incomplete number of RTT's and DATA OUT UPIU's could be requested if a write error occurs before the entire data transfer is complete

8.3.15.3 Write (16) Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If all requested data is successfully transferred and written, the WRITE command will terminate with a STATUS response of GOOD
- If the logical blocks are transferred directly to a cache then the Device Server may complete the command with a GOOD status prior to writing the logical blocks to the medium
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the WRITE command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ecc, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset or power-on)
    - Etc.

8.3.16 REQUEST SENSE Command

The REQUEST SENSE Command requests that the Device Server transfer parameter data containing sense data information to the Application Client.

- Sense Data describes error or exception condition and/or current operational status of device
    - i.e. get the device "status"
- UFS devices will return a fixed format data record of 18 bytes of sense data as described below
- Three tiered error code for detailed status
    - Sense Key: main indicator
    - ASC: additional information
    - ASCQ: qualifier
- If a REQUEST SENSE command is received with a pending UNIT ATTENTION condition (i.e., before the device server reports CHECK CONDITION status), the device server shall perform the REQUEST SENSE command and not clear the UNIT ATTENTION condition

8.3.16 REQUEST SENSE Command (cont'd)

The Command CDB shall be sent in a single Command UPIU

Table 8-33 — REQUEST SENSE Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (03h)							
1	Reserved (0000000b)							0b
2	(MSB)			Reserved				
3								(LSB)
4	Allocation Length (18d)							
5	CONTROL							

8.3.16.1 Request Sense Data Response

- Data returned from a REQUEST SENSE command will be transferred to the Application Client in a single DATA IN UPIU
- The Device Server will transfer up to 18 Bytes of Response Data in the Data Segment area of a Data In UPIU
    - Return 18 bytes if Allocation Length in CDB >= 18
    - Return Allocation Length bytes if Allocation Length in CDB < 18
    - An Allocation Length of zero is not an error
        - No data will be returned, go directly to Status phase
- Data will be returned in the indicated Sense Data Format described below

8.3.16.2 Se

8.3.16.3 nse Data

Table 8-34 — Sense Data

Byte	Bit	Value	Field Name: Description
0	7:7	Variable	VALID: ‘1’ indicates INFORMATION field contains valid data
0	6:0	1110000b	RESPONSE CODE: 70h
1	7:0	0h	OBSOLETE: 00h
2	7:4	0000b	RESERVED: 0 for UFS MSD
2	3:0	Variable	SENSE KEY: Error or exception indicator (see table below)
3:6	7:0	Variable	INFORMATION (bits 31-0): error dependent information
7	7:0	31d	ADDITIONAL SENSE LENGTH: 31 bytes
8:11	7:0	00h	COMMAND SPECIFIC: Additional command specific info, 0 for UFS 1.0
12	7:0	Variable	ADDITIONAL SENSE CODE (ASC): 00h for UFS 1.0
13	7:0	Variable	ADDITIONAL SENSE CODE QUALIFIER (ASCQ): 00h for UFS 1.0
14	7:0	00h	FIELD REPLACEABLE UNIT CODE (FRUC): 00h
15	7:7	0b	SKSC: ‘1’ indicates SENSE KEY SPECIFIC field contains valid data
15	6:0	000000b	SENSE KEY SPECIFIC (bits 22-16): 0
16	7:0	00h	SENSE KEY SPECIFIC (bits 15-8): 0
17	7:0	00h	SENSE KEY SPECIFIC (bits 7-0): 0

8.3.16.4 Sense Key

Table 8-35 — Sense Key

KEY	NAME	DESCRIPTION
0H	NO SENSE	No additional information or no error.
1H	RECOVERED ERROR	Command completed successfully with some recovery action
2H	NOT READY	Logical unit is not accessible
3H	MEDIUM ERROR	Command failed due to flaw in medium or recording error
4H	HARDWARE ERROR	Failure due to hardware (controller, device, parity, etc.)
5H	ILLEGAL REQUEST	Failure due to illegal address or parameter or unsupported command
6H	UNIT ATTENTION	Unit attention condition established (unit reset, power-on, removal, etc)
7H	DATA PROTECT	Failure due to access of protected area of medium
8h	BLANK CHECK	Not applicable for block device (UFSMSD)
9h	VENDOR SPECIFIC	Not applicable for UFS device
AH	COPY ABORTED	Not applicable for UFS device
BH	ABORTED COMMAND	Device aborted execution of the command
CH	Obsolete	-
DH	VOLUME OVERFLOW	Failure to write all data because end of range or partition
EH	MISCOMPARE	Indicates source data did not match data read from medium
FH	RESERVED	Reserved

8.3.16.5 ASC and ASCQ

Sense Key used for normal error handling during operation

- ASC:ASCQ codes are mainly used for detailed diagnostic and logging (post-mortem) information. Generally, except for a certain few, they’re not

ASC – Additional Sense Code

- For UFS 1.0, default value = 00h, which means no additional information provided

ASCQ – Additional Sense Code Qualifier

- For UFS 1.0, default value = 00h

8.3.16.6 Request Sense Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested data is successfully transferred, the REQUEST SENSE command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure is very rare. When the REQUEST SENSE command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
- Will not fail due to a pending UNIT ATTENTION condition
- If a REQUEST SENSE command is received with a pending UNIT ATTENTION condition (i.e., before the device server reports CHECK CONDITION status), the device server shall perform the REQUEST SENSE command and will not clear the UNIT ATTENTION condition

8.3.17 FORMAT UNIT Command

The FORMAT UNIT command requests that the Device Server format the medium into Application Client accessible logical blocks as specified in the parameter lists. The Device Server may also certify the medium and create control structures for the management of the medium and defects. The degree that the medium is altered by the command is vendor specific.

The Command CDB shall be sent in a single Command UPIU

Table 8-36 — FORMAT UNIT Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (04h)							
1	FMTPINFO		LONGLIST	FMTDATA	CMPLST	DEFECT LIST FORMAT		
2	Vendor Specific							
3	(MSB)							
4	Reserved							(LSB)
5	CONTROL							

8.3.17.1 Format Unit Command Parameters

Table 8-37 — Format Unit Command Parameters

Byte	Bit	Description
1	7:6	**FMTPINFO**: Specifies the FORMAT PROTECTION INFORMATION as detailed in SBC3.
1	5	**LONGLIST**: If set to '0' then the parameter list, if any, will use the SHORT parameter list header as defined in SBC3. If set to '1' then the parameter list uses the LONG format.
1	4	**FMTDATA**: If set to '1' specifies that parameter list data shall be transferred during the DATA OUT phase.
1	3	**CMPLST**: Complete List. '0' indicates that the parameter list contains a partial growing list of defects. A '1' indicates the list is complete. See SBC3.
1	2:0	**DEFECT LIST FORMAT**: If the FMTDATA bit is set to one, then the DEFECT LIST FORMAT field specifies the format of the address descriptors in the defect list. See SBC-3 [SBC].
2	7:0	**VENDOR SPECIFIC**: Vendor specified field.

8.3.17.2 Format Unit Command Data Transfer

The FORMAT UNIT command will transfer out the number of bytes specified within the Parameter List header format.

The Device Server will request the transfer of the specified bytes from the Application Client by issuing a series of Ready To Transfer UPIU (RTT) followed by a Data Out UPIU containing the number of bytes to transfer.

8.3.17.3 Format Unit Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested format of the medium is performed successfully then the command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Other failures can occur for numerous reasons. When the FORMAT UNIT command fails, a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ecc, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset or power-on)
    - Etc.

8.3.18 SEND DIAGNOSTIC Command

The SEND DIAGNOSTIC command requests that the Device Server perform diagnostic operations on the SCSI target device, on the logical unit or on both. Logical units that support this command shall implement, at a minimum, the default self-test feature when the SELFTEST bit is set to '1'.

The Command CDB shall be sent in a single Command UPIU.

Table 8-38 — SEND DIAGNOSTIC Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Dh)							
1	SELF-TEST CODE			PF	0	SELFTEST	DEVOFFL	UNITOFFL
2	Reserved							
3	(MSB)			Parameter List Length				
4								(LSB)
5	CONTROL							

8.3.18.1 Send Diagnostic Parameters

Table 8-39 — Send Diagnostic Parameters

Byte	Bit	Description
1	7:5	**SELF-TEST CODE**: Specifies the self-test code as defined in SPC4.
1	4	**PF**: Specifies the format of any parameter list sent by the Application Client
1	2	**SELFTEST**: Specifies the device server shall perform a self test.
1	1	**DEVOFFL**: If set to '0' the device server will perform a self-test without exhibiting any effects on the logical unit. If set to '1' the device server may perform a self-test that affects the logical unit.
1	0	**UNITOFFL**: If set to '0' the device server will perform a self-test without exhibit no effects on the user accessible medium of the logical unit. If set to '1' the device server may perform operations that affect the user accessible medium.
1	5	**PARAMETER LIST LENGTH**: Specifies the length in bytes that shall be transferred from the Application Client to the device server. A parameter list length of zero specifies that no data shall be transferred.

8.3.18.2 Send Diagnostic Command Data Transfer

The SEND DIAGNOSTIC command will transfer out the number of bytes specified by the Parameter List Length. If that value is zero then no data out transfer will occur.

The Device Server will request the transfer the specified bytes from the Application Client by issuing a series Ready To Transfer UPIU (RTT) followed by a Data Out UPIU containing the number of bytes to transfer.

8.3.18.3 Send Diagnostic Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested diagnostics are performed successfully then the command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Other failures can occur for numerous reasons. When the SEND DIAGNOSTIC command fails, a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ecc, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset or power-on)
    - Etc.

8.3.19 SYNCHRONIZE CACHE Command

The SYNCHRONIZE CACHE (10) command requests that the device server ensure that the specified logical blocks have their most recent data values recorded in non-volatile cache and/or on the medium, based on the SYNC_NV bit.

Table 8-40 — SYNCHRONIZE CACHE Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (35h)							
1	Reserved					SYNCH_NV	IMMED	0b
2	(MSB)							
3				LOGICAL BLOCK ADDRESS				
4								
5								(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB)							
8	NUMBER OF LOGICAL BLOCKS							(LSB)
9	CONTROL							

8.3.19.1 Synchronize Cache Command Parameters

Table 8-41 — Synchronize Cache Command Parameters

<table>
<tr><th>Byte</th><th>Bit</th><th>Description</th></tr>
<tr><td>1</td><td>2</td><td>SYNC_NV: The SYNC_NV bit specifies whether the device server is required to synchronize volatile and non-volatile caches.
<table>
<tr><th rowspan="2">Code</th><th colspan="2">Device Server requirement to
synchronize logical blocks currently in the</th></tr>
<tr><th>Volatile Cache</th><th>Non-Volatile Cache</th></tr>
<tr><td>0</td><td>Device server shall synchronize to the medium</td><td>Device server shall synchronize to the medium</td></tr>
<tr><td>1</td><td>If a non-volatile cache is present, then the device server shall synchronize to non-volatile cache or the medium. If a non-volatile cache is not present, then the device server shall synchronize to the medium</td><td>No requirement</td></tr>
</table></td></tr>
<tr><td>1</td><td>1</td><td>IMMED: An immediate (IMMED) bit set to zero specifies that the device server shall not return status until the operation has been completed. An IMMED bit set to one specifies that the device server shall return status as soon as the CDB has been validated. If the IMMED bit is set to one, and the device server does not support the IMMED bit, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.</td></tr>
<tr><td>2:5</td><td>7:0</td><td>LOGICAL BLOCK ADDRESS: The LOGICAL BLOCK ADDRESS field specifies the LBA of the first logical block accessed by this command. If the specified LBA exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.</td></tr>
<tr><td>6</td><td>4:0</td><td>GROUP NUMBER: The GROUP NUMBER field specifies the group into which attributes associated with the command should be collected. A GROUP NUMBER field set to zero specifies that any attributes associated with the command shall not be collected into any group. In UFS the value is ignored by UFS v1.0 device. UFS v1.0 host is intended to use fixed value ‘0’ for this field.</td></tr>
<tr><td>7:8</td><td>7:0</td><td>NUMBER OF LOGICAL BLOCKS: The NUMBER OF LOGICAL BLOCKS field specifies the number of logical blocks that shall be synchronized, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A NUMBER OF LOGICAL BLOCKS field set to zero specifies that all logical blocks starting with the one specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium shall be synchronized. If the LBA plus the number of logical blocks exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

A logical block within the range that is not in cache is not considered an error.</td></tr>
<tr><td>9</td><td>7:0</td><td>Control: This CONTROL byte are defined in SAM-5. But, In UFS the value is ignored by UFS v1.0 device. UFS v1.0 host is intended to use fixed value ‘0’ for this field. That is, No vendor specific interpretation and Normal ACA are assumed.</td></tr>
</table>

8.3.19.2 Synchronize Cache Command Data Transfer

The SYNCHRONIZE CACHE command does not have a data transfer phase

- No DATA IN or DATA OUT UPIU's are transferred

8.3.19.3 Synchronize Cache Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested diagnostics are performed successfully then the command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Other failures can occur for numerous reasons. When the SYNCHRONIZE CACHE command fails, a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ecc, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset or power-on)
    - Etc.

8.3.20 UNMAP Command

The UNMAP command shall require the device server to cause one or more LBAs to be unmapped (de-allocated).

UFS defines that a logical unit shall be either Full Provisioning or Thin Provisioning as described in SCSI SBC. To use UNMAP command, SCSI SBC requires that a logical unit to be thin-provisioned and support logical block provisioning management. UNMAP command is not supported in a full-provisioned logical unit.

UFS shall require a thin provisioned logical unit to have sufficient physical memory resources to support the logical block address space when delivered by manufacturer. Mapped State and De-Allocated State are mandatory in a UFS thin provisioned logical unit.

8.3.20 UNMAP Command (cont'd)

Table 8-42 — UNMAP Command Descriptor Block

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (42h)							
1	Reserved							ANCHOR
2	(MSB)							
3				Reserved				
4								
5								(LSB)
6	Reserved				GROUP NUMBER			
7	(MSB)							
8				PARAMETER LIST LENGTH				(LSB)
9	CONTROL							

- GROUP NUMBER = '0'
- ANCHOR = 0 for UFS. If ANCHOR = 1, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

8.3.20.1 UNMAP parameter list

The UNMAP parameter list contains the data sent by an application client along with an UNMAP command. Included in the data are an UNMAP parameter list header and block descriptors for LBA extents to be processed by the device server for the UNMAP command. The LBAs specified in the block descriptors may contain overlapping extents, and may be in any order.

Table 8-43 — UNMAP parameter list

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	UNMAP DATA LENGTH (n-1)						
1								(LSB)
2	(MSB)	UNMAP BLOCK DESCRIPTOR DATA LENGTH (n-7)						
3								(LSB)
4	(MSB)	Reserved						
….								
7								(LSB)
UNMAP block descriptors								
8	(MSB)	UNMAP block descriptor (first)						
….								
23								(LSB)
…..								
n-15	(MSB)	UNMAP block descriptor (last)						
….								
n								(LSB)

- The UNMAP DATA LENGTH field specifies the length in bytes of the following data that is available to be transferred from the data-out buffer. The unmap data length does not include the number of bytes in the UNMAP DATA LENGTH field.
- The UNMAP BLOCK DESCRIPTOR DATA LENGTH field specifies the length in bytes of the UNMAP block descriptors that are available to be transferred from the data-out buffer. The unmap block descriptor data length should be a multiple of 16. If the unmap block descriptor data length is not a multiple of 16, then the last unmap block descriptor is incomplete and shall be ignored. If the UNMAP BLOCK DESCRIPTOR DATA LENGTH is set to zero, then no unmap block descriptors are included in the UNMAP parameter data. This condition shall not be considered an error.

8.3.20.2 UNMAP block descriptor

Table 8-44 — UNMAP block descriptor

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
….				UNMAP LOGICAL BLOCK ADDRESS				
7								(LSB)
8	(MSB)							
….				NUMBER OF LOGICAL BLOCKS				
11								(LSB)
12								
….				Reserved				
15								

- The UNMAP LOGICAL BLOCK ADDRESS field contains the first LBA of the UNMAP block descriptor to be unmapped.
- The NUMBER OF LOGICAL BLOCKS field contains the number of LBAs to be unmapped beginning with the LBA specified by the UNMAP LOGICAL BLOCK ADDRESS field.
- LBA's to be unmapped should align with the Erase_Block_Size in the Unit Descriptor of the LU where possible (but not required) to minimize performance degradation.
- If the NUMBER OF LOGICAL BLOCKS is set to zero, then no LBAs shall be unmapped for this UNMAP block descriptor. This condition shall not be considered an error.
- If the LBA specified by the UNMAP LOGICAL BLOCK ADDRESS field plus the number of logical blocks exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.
- In UFS, if Vital Product Data Page is not supported by the UFS device, default values of MAXIMUM UNMAP LBA COUNT and MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT are set as follows.
    - MAXIMUM UNMAP LBA COUNT = LBA count reported in READ CAPACITY
    - MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT = 1
- If Vital Product Data Page is supported by the device, the MAXIMUM UNMAP LBA COUNT and MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT are set by the device manufacturer in the Block Limits VPD page. If the total number of logical blocks specified in the UNMAP block descriptor data exceeds the value indicated in the MAXIMUM UNMAP LBA COUNT field in the Block Limits VPD page or if the number of UNMAP block descriptors exceeds the value of the MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT field in the Block Limits VPD page, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

8.3.21 READ BUFFER Command

The READ BUFFER command is used in conjunction with the WRITE BUFFER command for

- testing logical unit buffer memory
- testing the integrity of the service delivery subsystem
- Downloading microcode
- Retrieving error history and statistics
- Tunneling commands and data

The READ BUFFER command transfers a specified number of data bytes from a specified offset within a specified buffer in the Device Server to a buffer in the Application Client

The Command CDB shall be sent in a single Command UPIU

Table 8-45 — READ BUFFER UFS Command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (3Ch)							
1	RESERVED (000b)			MODE				
2	BUFFER ID							
3	(MSB)							
4				BUFFER OFFSET				
5								(LSB)
6	(MSB)							
7				ALLOCATION LENGTH				
8								(LSB)
9	CONTROL (00h)							

8.3.21.1 Read Buffer Command Parameters

Table 8-46 — Read Buffer Command Parameters

Byte	Bit	Description
1	4:0	**MODE**: Specifies the function of this command. UFS 1.0 with use DATA MODE (02h) to transfer UFS specific data. See table below for more detail.
2	7:0	**BUFFER ID**: Specifies a buffer within the logical unit. Buffer 0 shall be supported. If more than one buffer is supported, then additional BUFFER ID codes shall be assigned contiguously, beginning with one.
3:5	7:0	**BUFFER OFFSET**: Specifies the byte offset within the specified buffer from which data shall be transferred.
6:8	7:0	**ALLOCATION LENGTH**: Specifies the maximum number of bytes of buffer space that the Application Client has allocated for data reception.

8.3.21.2 Read Buffer Command Mode Field Values

Table 8-47 — Read Buffer Command Mode Field Values

MODE	DESCRIPTION
00h	Not used in UFS
01h	Vendor Specific
02h	Data
03h-1Ch	Not used in UFS
1Dh-1Fh	Reserved

- UFS 1.0 shall use a MODE value of 02h, indicating Data Mode. Data Mode will transfer from the Device Server up to Allocation Length bytes, starting at Buffer Offset, of non-descript data from the buffer specified by Buffer ID.
- The definition and structure of the data being transferred in Data Mode will be described in other UFS documents that specify the functional description of the UFS specific extension operation that uses the READ BUFFER command.

8.3.21.3 Read Buffer Command Data Transfer

- The Device Server will read up to Allocation Length number of data bytes from the specified Buffer Offset within a buffer specified by the Buffer ID in the Device Server and transfer them to a buffer in the Application Client
    - Less than Allocation Length will be transferred if Device Server contains less bytes
- Data will be transferred from the Device Server to the Application Client via a series of Data In UPIU's
    - The data transferred from the Device Server will be contained within the Data Segment of the Data In UPIU
- Zero or an incomplete number of DATA IN UPIU's will be transferred if an error occurs before the entire data transfer is complete

8.3.21.4Read Buffer Command Status Response

- Status response will be sent in a single RESPONSE UPIU
- If all requested data is successfully read and transferred, the READ BUFFER command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the READ BUFFER command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ecc, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset or power-on)
    - Etc.

8.3.22 WRITE BUFFER Command

The WRITE BUFFER command is used in conjunction with the READ BUFFER command for

- testing logical unit buffer memory
- testing the integrity of the service delivery subsystem
- Downloading microcode
- Retrieving error history and statistics
- Tunneling commands and data

The WRITE BUFFER command transfers a specified number of data bytes from a buffer in the Application Client to a specified buffer in the Device Server at a specified buffer offset

8.3.22 WRITE BUFFER Command (cot'd)

The Command CDB shall be sent in a single Command UPIU

Table 8-48 — WRITE BUFFER UFS Command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (3Bh)							
1	RESERVED (000b)			MODE				
2	BUFFER ID							
3	(MSB)							
4				BUFFER OFFSET				
5								(LSB)
6	(MSB)							
7				PARAMETER LIST LENGTH				
8								(LSB)
9	CONTROL (00h)							

8.3.22.1 Write Buffer Command Parameters

Table 8-49 — Write Buffer Command Parameters

Byte	Bit	Description
1	4:0	**MODE**: Specifies the function of this command. UFS 1.0 with use DATA MODE (02h) to transfer UFS specific data and MODES 04h-07h, 0Eh-0Fh for microcode download. See table below for more detail.
2	7:0	**BUFFER ID**: Specifies a buffer within the logical unit. Buffer 0 shall be supported. If more than one buffer is supported, then additional BUFFER ID codes shall be assigned contiguously, beginning with one.
3:5	7:0	**BUFFER OFFSET**: Specifies the byte offset within the specified buffer from which data shall be transferred.
6:8	7:0	**PARAMETER LIST LENGTH**: Specifies the maximum number of bytes the Application Client buffer will transfer to the Device Server.

8.3.22.2 Write Buffer Command Mode Field Values

Table 8-50 — Write Buffer Command Mode Field Values

MODE	DESCRIPTION
00h	Not used in UFS
01h	8.3.23 Vendor Specific
02h	Data
03h	Not used in UFS
04h	Download microcode and activate
05h	Download microcode, save and activate
06h	Download microcode with offsets and activate
07h	Download microcode with offsets, save and activate
08h-0Dh	Not used in UFS
0Eh	Download microcode with offsets, save and defer activation
0Fh	Activate deferred microcode
10h-1Ch	Not used in UFS
1Dh-1Fh	Reserved

- UFS 1.0 shall use a MODE value of 02h, indicating Data Mode. Data Mode will transfer from the Application Client Parameter List Length bytes of non-descript data destined for the Device Server for the buffer specified by Buffer ID at the offset specified by Buffer Offset.
- The definition and structure of the data being transferred in Data Mode will be described in other UFS documents that specify the functional description of the UFS specific extension operation that uses the WRITE BUFFER command.
- UFS 1.0 may use a MODE value of 04h-07h or 0Eh-0Fh for microcode download. The format and structure is will be specified in subsequent UFS documents.

8.3.23.1 Write Buffer Command Data Transfer

The Device Server will request transfer of the specified number of bytes from a buffer in the Application Client by issuing a series of Ready To Transfer UPIU's (RTT) followed by a Data Out UPIU per RTT and will subsequently write those bytes to the specified Buffer Offset within the buffer specified by the Buffer ID in the Device Server

- The data transferred from the Application Client will be contained within the Data Segment of the Data Out UPIU

Zero or an incomplete number of RTT's and DATA OUT UPIU's could be requested if a write error occurs before the entire data transfer is complete

8.3.23.2Write Buffer Command Status Response

- STATUS response will be sent in a single RESPONSE UPIU
- If the requested data is successfully transferred and written, the WRITE BUFFER command will terminate with a STATUS response of GOOD
- If the unit is not ready to accept a new command (e.g. still processing previous command) a STATUS response of BUSY will be returned
- Failure can occur for numerous reasons. When the WRITE BUFFER command fails a STATUS response of CHECK CONDITION will be returned along with an appropriate SENSE KEY, such as
    - ILLEGAL REQUEST (range or CDB errors)
    - MEDIUM ERROR (medium failure, ecc, etc.)
    - HARDWARE ERROR (hardware failure)
    - UNIT ATTENTION (unexpected reset or power-on)
    - Etc.

8.4 Mode Pages

This section describes the mode pages used with MODE SELECT command and MODE SENSE command. Subpages are identical to mode pages except that they include a SUBPAGE CODE field that further differentiates the mode page contents.

8.4.1 Mode Page Overview

8.4.1.1 Mode Page/Subpage Codes

- UFS will support limited, specific page codes
- UFS will support no subpages (subpage = 0)

Table 8-51 — Summary of mode page codes

Page Code	Subpage Code	Description	Page Format
00h	Vendor specific	Vendor specific page	Vendor specific format
01h to 1Fh (SCSI SPECIFIC)	00h	Device specific STANDARD page (subpage 0)	Page 0 format
	01h to DFh	Device specific SUBPAGE	Subpage format
	E0h to FEh	Vendor specific SUBPAGE	Subpage Format
	FFh	Return all SUBPAGES for the specified **device** specific mode page	Page 0 format for subpage 00h, subpage format for subpages 01h to FEh
20h to 3Eh (VENDOR SPECIFIC)	00h	Vendor specific STANDARD page (subpage 0)	Page 0 format
	01h to FEh	Vendor specific SUBPAGE	Subpage format
	FFh	Return all SUBPAGES for the specified **vendor** specific mode page	Page 0 format for subpage 00h, subpage format for subpages 01h to FEh
3Fh (Return ALL pages)	00h	Return all STANDARD pages (subpage 0)	Page 0 format
	01h to FEh	Reserved	NA
	FFh	Return all subpages for all mode pages	Page 0 format for subpage 00h, sub_page format for subpages 01h to FEh

8.4.1.2 Mode parameter list format

General format for reading or writing mode pages.

UFS will not implement Block Descriptor field

Table 8-52 — UFS Mode parameter list

Bit Byte	7	6	5	4	3	2	1	0
	Mode Parameter Header							
	Block Descriptor(s)							
	Mode Page(s) or Subpages or Vendor specific pages							

8.4.1.3 Mode Parameter Header

The mode parameter header that is used by the MODE SELECT (10) command and the MODE SENSE (10) command is defined in the following table.

Table 8-53 — UFS Mode parameter header (10)

Bit Byte	7	6	5	4	3	2	1	0
0	Mode Data Length (N – 2)							
1								
2	Medium Type (00h)							
3	WP	Reserved (00b)		0b	Reserved (0000b)			
4	Reserved (0000h)							
5								
6	Block Descriptor Length (00h)							
7								

When using the MODE SENSE command, the MODE DATA LENGTH field indicates the length in bytes of the following data that is available to be transferred. The mode data length does not include the number of bytes in the MODE DATA LENGTH field. When using the MODE SELECT command, this field is reserved.

8.4.1.4 Mode Parameter Header Detail

Table 8-54 — Mode Parameter Header Detail

Byte	Bit	Description
0:1	7:0	**MODE DATA LENGTH**: Indicates the length in bytes of data following this field that is available to transfer. This value does not include the size of this field (2 bytes). For MODE SENSE 10-byte CDB, this value will be calculated as 6 + page data bytes.
2	7:0	**MEDIUM TYPE**: Indicates the medium type of the device. For UFS this value shall be set to 00h, indicating Data Medium.
3	7:7	**WP**: Indicates the write protection status of the medium when using Mode Sense (reading). Don't care bit for Mode Select (writing).
3	6:0	**DEVICE SPECIFIC PARAMETER**: Direct access device specific value. For UFS this field will contain (0000000b).
6:7	7:0	**BLOCK DESCRIPTOR LENGTH**: Length of block descriptor in parameter list. For UFS this value shall be 00h indicating that there is no block descriptor(s) used in the parameter list.

8.4.1.5 Page 0 Format

Table 8-55 — Page 0 Format

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	Page Code					
1	Page Length (N – 1)							
2	Mode Parameters							
N								

Table 8-56 — Page 0 Format parameters

Byte	Bit	Description
0	7:7	**PS**: Indicates the page parameters can be saved. When using the MODE SENSE command, the PS bit set to one indicates that the mode page may be saved by the logical unit in a nonvolatile, vendor specific location. A PS bit set to zero indicates that the device server is not able to save the supported parameters. When using the MODE SELECT command, the PS bit is reserved.
0	6:6	**SPF**: Indicates SUBPAGE format. When set to zero indicates that the PAGE 0 format is being used. When set to one, indicates the SUBPAGE mode page format is being used.
0	5:0	**PAGE CODE**: Indicates the format and parameters for particular mode page.
1	7:0	**PAGE LENGTH**: Indicates the size in bytes of the following mode page parameters.
2:N	7:0	**MODE PARAMETERS**: The contents of the indicated mode page.

8.4.1.6 Subpage Format

Table 8-57 — Subpage Format

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	Page Code					
1	Subpage Code							
2	Page Length (N – 3)							
3								
4	Mode Parameters							
N								

Table 8-58 — Subpage Format parameters

Byte	Bit	Description
0	7:7	**PS**: Indicates the page parameters can be saved. When using the MODE SENSE command, the PS bit set to one indicates that the mode page may be saved by the logical unit in a nonvolatile, vendor specific location. A PS bit set to zero indicates that the device server is not able to save the supported parameters. When using the MODE SELECT command, the PS bit is reserved.
0	6:6	**SPF**: Indicates SUBPAGE format. When set to zero indicates that the PAGE 0 format is being used. When set to one, indicates the SUBPAGE mode page format is being used.
0	5:0	**PAGE CODE**: Page and Subpage code indicates the format and parameters for particular mode page.
1	7:0	**SUBPAGE CODE:** Page and subpage code indicates the format and parameters for particular mode page.
2:3	7:0	**PAGE LENGTH**: Indicates the size in bytes of the following mode page parameters.
4:N	7:0	**MODE PARAMETERS**: The contents of the indicated mode page.

8.4.2 UFS Supported Pages

The followings are the mode pages to be supported by UFS. UFS does not define any subpage currently.

Table 8-59 — UFS Supported Pages

PAGE NAME	PAGE CODE	SUBPAGE CODE	DESCRIPTION
CONTROL	0Ah	00h	Return CONTROL mode page
READ-WRITE ERROR RECOVERY	01h	00h	Return READ-WRITE ERROR RECOVERY mode page
CACHING	08h	00h	Return CACHING mode page
ALL PAGES	3Fh	00	Return all mode pages (not including subpages)
ALL SUBPAGES	3Fh	FFh	Return all mode pages and subpages

8.4.2.1 Control Mode Page

The Control mode page provides controls over SCSI features that are applicable to all device types (e.g., task set management and error logging)

Table 8-60 — Control Mode Page

Bit Byte	7	6	5	4	3	2	1	0
0	0	SPF (0b)	PAGE CODE (0Ah)					
1	PAGE LENGTH (0Ah)							
2	TST		0b	0b	0b	0b	0b	0b
3	0001b				0b	00b		0b
4	0b	0b	00b		SWP	000b		
5	0b	0b	000b			000b		
6	0000h							
7								
8	BUSY TIMEOUT PERIOD							
9								
10	0000h							
11								

8.4.2.2 Control Mode Page Parameters

Table 8-61 — Control Mode Page Parameters

Byte	Bit	Description
1	7:6	**TST**: Indicates TASK SET type. 00b indicates the device maintains a single task set for all logical units. 01b indicates the device maintains an individual task set for each logical unit.
4	3:3	**SWP**: A software write protect (SWP) bit set to one specifies that the logical unit shall inhibit writing to the medium after writing all cached or buffered write data, if any. When SWP is one, all commands requiring writes to the medium shall be terminated with CHECK CONDITION status, with the sense key set to DATA PROTECT
8:9	7:0	**BUSY TIMEOUT PERIOD**: The BUSY TIMEOUT PERIOD field specifies the maximum time, in 100 milliseconds increments, that the application client allows for the device server to return BUSY status for commands from the application client. A 0000h value in this field is undefined. An FFFFh value in this field is defined as an unlimited period.

8.4.2.3 Read-Write Error Recovery Mode Page

The Read-Write Error Recovery mode page specifies the error recovery parameters the device server shall use during any command that performs a read or write operation to the medium (e.g., READ command, WRITE command, or a WRITE AND VERIFY command).

Table 8-62 — Read-Write Error Recovery Mode Page

Bit Byte	7	6	5	4	3	2	1	0
0	0	SPF (0b)	PAGE CODE (01h)					
1	PAGE LENGTH (0Ah)							
2	1b	0b	0b	0b	0b	0b	0b	0b
3	READ RETRY COUNT							
4	00h							
5	00h							
6	00h							
7	0b	00000b					0b	0b
8	WRITE RETRY COUNT							
9	Reserved (00h)							
10	RECOVERY TIME LIMIT							
11								

8.4.2.4 Read-Write Error Recovery Parameters

Table 8-63 — Read-Write Error Recovery Parameters

Byte	Bit	Description
3	7:0	**READ RETRY COUNT**: The READ RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during read operations.
8	7:0	**WRITE RETRY COUNT**: The WRITE RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during write operations.
10:11	7:0	**RECOVERY TIME LIMIT**: The RECOVERY TIME LIMIT field specifies in milliseconds the maximum time duration that the device server shall use for data error recovery procedures. When both a retry count and a recovery time limit are specified, the field that specifies the recovery action of least duration shall have priority.

8.4.2.5 Caching Mode Page

The Caching mode page defines the parameters that affect the use of the cache.

Table 8-64 — Caching Mode Page

Bit Byte	7	6	5	4	3	2	1	0
0	0	SPF (0b)	PAGE CODE (08h)					
1	PAGE LENGTH (12h)							
2	0b	0b	0b	0b	0b	WCE	0b	RCD (0b)
3	0000b				0000b			
4	0000h							
5								
8	0000h							
9								
10	0000h							
11								
12	0b	0b	DRA	Vendor Specific (00b)		Reserved (00b)		0b
13	00h							
14	0000h							
15								
16	Reserved (00h)							
17	000000h							
18								
19								

8.4.2.6 Caching Mode Page Parameters

Table 8-65 — Caching Mode Page Parameters

Byte	Bit	Description
2	2:2	**WCE**: WRITE BACK CACHE ENABLE. A writeback cache enable bit set to zero specifies that the device server shall complete a WRITE command with GOOD status only after writing all of the data to the medium without error. A WCE bit set to one specifies that the device server may complete a WRITE command with GOOD status after receiving the data without error and prior to having written the data to the medium.
2	0:0	**RCD**: READ CACHE DISABLE. A read cache disable bit set to zero specifies that the device server may return data requested by a READ command by accessing either the cache or medium. A RCD bit set to one specifies that the device server shall transfer all of the data requested by a READ command from the medium (i.e., data shall not be transferred from the cache)..
12	5:5	**DRA**: DISABLE READ AHEAD. A disable read-ahead bit set to one specifies that the device server shall not read into the pre-fetch buffer any logical blocks beyond the addressed logical block(s). A DRA bit set to zero specifies that the device server may continue to read logical blocks into the pre-fetch buffer beyond the addressed logical block(s).

9 UFS SECURITY

This section summarizes the security features and the implementation details that will be in the UFS spec. These features include: Secure mode operation, data and register protection, RPMB and reset.

9.1 UFS Security Feature Support Requirements

The UFS command set will be divided into different command classes. One of the command classes will be the security class. The security features outlined in this specification are mandatory for all devices. Supporting security features also means that the device will have one LUN that will offer the RPMB functionality. Also, all LUNs on the device will allow the user to select a secure mode and different types of write protection for the entire LUN.

9.2 Secure Mode

9.2.1 Description

UFS devices will be used to store user's personal and/or corporate data information. The UFS device must provide a way to remove the data permanently from the device when requested, ensuring that it cannot be retrieved using reverse engineering on the memory device.

The UFS device will support a secure and insecure mode of operation. When the device is in the secure mode of operation all operations that result in the removal or retiring of information on the device will purge this information in a secure manner, as outlined in section**Error! Reference source not found.**.

The secure mode will be applied at the Logical unit, so different Logical unit can have different secure modes.

9.2.1.1 Commands and Operations Impacted

Examples of operations that secure mode will impact:

- Erase
- Trim
- Overwrite operations
- Bad Block Management
- Background operations that result in data removal (both host initiated and device initiated)

9.2.2 Requirements

9.2.2.1 Secure Removal

The way in which data is removed securely from the device is dependent on the type of memory technology that is used to implement the UFS device. Three common methods that apply to most memory types implemented at the time of this spec are:

1. The device controller should issue an erase operation to the addressed location.
2. The device controller should overwrite the addressed locations with a single character and erase the device.
3. The device controller should overwrite the addressed locations with a character, its complement, then a random character

UFS secure device manufacturers are required to support the secure removal method required for their specific flash media.

9.2.2.2 Erase Operation

Erase is an operation that moves data from the mapped host address space to the unmapped address space. The regions in the mapped host address space where erase and Trim were applied will be set to the erased value = '0'. This operation places no requirement on what the device is required to do with the data in the unmapped host address space. After an erase or trim is executed software on the host should not be able to retrieve the erased or trimmed data.

The difference between the erase and Trim command is the minimum size of the unit that they operate on. The minimum data range that an erase operates on is the smallest region that can be written.

9.2.2.3 Discard Operation

Discard is a non-secure variant of the Trim/Erase functionality. The distinction between Discard and Trim/Erase is the device behavior where the device is not required to guarantee that host would not retrieve the original data from one or more LBA's that were marked for erase when a Read operation is directed to the LBA's.

9.2.2.4 Purge Operation

The Purge operation operates on the unmapped host address space. 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 that is set. This mode allows the host system to protect against die level attacks.

9.2.3 Implementation

9.2.3.1 Erase

The Trim and Erase functionality is fulfilled by the UNMAP command on an LU the TPRZ bit in the READ CAPACITY(16) parameter data is set to ‘one’.

For an LU with the TPRZ = 1, UFS defines that all LBA’s contained in the UNMAP command shall be unmapped. An unmapped LBA shall remain unmapped until a WRITE operation is directed at the LBA, and physical memory resources are available to be allocated by the device server to complete the operation without error. READ operation to an unmapped (de-allocated) LBA shall return value of ‘zero’.

LBA’s to be trimmed/erased may align to multiples of the Erase_Block_Size in the Unit Descriptor of the LU where possible to minimize performance impact.

TPRZ bit shall be set to ‘one’ for the LU in the Configuration Descriptor.

9.2.3.2 Discard

When an LU with TPRZ = 0 in the READ CAPACITY(16) parameter data, the device behavior corresponds to the Discard functionality. In Discard, LBA’s contained in the UNMAP command should be unmapped by the device server but there is no specific requirement to do so. READ operation to an LBA following UNMAP may return any value, including the original data. LBA’s to be discard may align to multiples of the Erase_Block_Size in the Unit Descriptor of the LU where possible to minimize performance impact.

TPRZ bit shall be set to ‘zero’ for the LU in the Configuration Descriptor.

9.2.3.3 Purge operation

The Purge operation shall be implemented via Query Function with Attributes and Flags.

- A PURGE_EN bit shall exist as a Write Only flag
    - When the bit is set to 0 it will be disabled (default)
    - When set to one it will be enabled.
    - This bit can only be set when the LU queue is empty.
    - This bit is automatically cleared by the UFS device when the operation completes or an error condition occurs.
- A PURGE_ERR descriptor bit should be defined. (read only)
    - 0 no error exists (default)
    - 1 Purge operation was not run because the queue was not empty.
    - When the PURGE bit is set this bit will be reset to zero unless a new error occurs.
- A PURGE_STATUS bit shall be defined in an attribute. (read only)
    - 00 no status (default)
    - 01 Purge in progress
    - 10 Purge was stopped prematurely.
    - 11 Purge completed successfully.

9.2.3.3 Purge operation (cont'd)

- If the Purge operation is in progress any commands placed into the LU queue will fail.The device shall return the sense key"NOT READY" to show that the command failed because the Purge was in progress.
- If the host must execute a command when the Purge is in progress it must send a START_STOP_UNIT command to the device. When the stop unit command is executed it will set the PURGE_STATUS bit to 10.
- If a power failure occurs the Purge status bit shall be reset to 00. There is no way for the device to indicate that this failed.

9.2.3.4 Wipe Device

To combine the activities of a Purge and an erase operation a FORMAT_UNIT command will be issued.

9.2.3.5 Parameter: SECURE_MODE

- SECURE_MODE shall be defined for each logical unit (n is the logical unit number). UFS defines SECURE_MODE shall be tied to the TPRZ value setting in the READ CAPACITY parameter data.
- TPRZ value of '1' indicates the device is in secure mode. In this mode all operations shall be performed using the mode defined by SECURE_REMOVAL_TYPE. Only one type of removal type can be defined for an entire device.
- TPRZ value of '0' indicates the device is in normal mode.
- SECURE_MODE (i.e. TPRZ bit) shall be set once in Configuration Descriptor.

9.2.3.6 Parameter: SECURE_REMOVAL_TYPE

- In the secure parameter page a parameter represented by two bits whose type is defined by SECURE_MODE_PROTECT shall be defined.
- Value of '11' will result in the information being removed using a vendor defined mechanism.
- Value of '10' will result in all information being removed by overwriting the addressed locations with a character, its complement, then a random character.
- Value of '01' will result in all information being removed by overwriting the addressed locations with a single character followed by an erase.
- Value of '00' will result in all information being removed by an erase of the physical memory (default).
- Device manufacturers are only required to support the mechanism required by their memory array technology.
- SECURE_REMOVAL_TYPE is set once in Configuration Descriptor.

9.3 Device Data Protection

9.3.1 Description and Requirements

9.3.1.1 Partition Write Protection

The largest area in a UFS device that can be protected will be a partition. The following protection modes will be available at the partition level.

- Permanent write and erase protection (permanent read only)
- Power on and reset write and erase protection (read only for power on period)
- Temporary write and erase protection

Write protection will not be implemented in the RPMB block.

There should also be a method to read the type of protection that is currently enabled for a partition.

For the case where smaller regions with in a partition are protected, the highest method of protection defined for that region shall be protected. The order of protection mode priority is set as:

1. Permanent
2. Power On
3. Temporary
4. No protection Implementation

9.3.1.2 Parameters: Partition Write Protection

- Each logical unit will have protection bits that indicate the protection status of a logical unit.
- LU_x_PROT (3 bits)
    - PERM_PROT (R/W)
        - ‘1’ LU is permanently protected. Values of LU_PWR_PROT and LU_TMP_PROT are ignored.
        - ‘0’ LU is not permanently protected.
    - LU_PWR_PROT (R/W_P)
        - ‘1’ LU is power on protected. Value of LU_TMP_PROT is ignored.
        - ‘0’ LU is not power on protected.
    - LU_TMP_PROT (R/W/E)
        - ‘1’ LU is temporarily protected.
        - ‘0’ LU is not temporarily protected.

9.4 RPMB

9.4.1 Description

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 UFS device memory (shared secret).

As the system can not 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

9.4.2 RPMB Logical Unit Description

The RPMB is contained in a unique logical unit (LU) whose size is defined in the LU descriptor. It has a minimum size of 128 KB. The contents of the RPMB LU can only be read and written via successfully authenticated read and write accesses. The data may be overwritten by the host but can never be erased.

If the device has the ability to support enhanced regions the RPMB block must be implemented as an enhance region.

All data writes will be directed to the LU specified in the original command, in this case, the RPMB LUN. When writing the data for an RPMB command it will be done as a reliable write.

The attributes to indicate the Reliable Write and enhanced region requirements are indicated in the RPMB LU descriptor.

All accesses to the RPMB will reference the specific RPMB LUN.

9.4.3 Requirements

9.4.3.1 Memory Map of RPMB LU

- Authentication Key
    - Type W (write only) - Write once, not erase or readable
    - 32 bytes
    - Authentication key register which is used to authenticate accesses when MAC is calculated.
- Write Counter
    - Type R (Read only)
    - 4 bytes
    - Counter value for the total amount of successful authenticated data write requests made by the host. The initial value of this register after production is 0x00000000. The value will be incremented by one automatically by the RPMB block along with each successful programming access. The value cannot be reset. After the counter has reached a maximum value 0xffffffff it will not be incremented anymore (overflow prevention).

9.4.3.1 Memory Map of RPMB LU (cont'd)

- RPBM Data Area
    - Type R/W (Read/Write)
    - Multiples of 128Kbytes defined in RPMB LU descriptor
        - 128Kbytes min., 16Mbytes max.
    - Data which 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.

9.4.3.2 Algorithm and Key for MAC 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), which 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 UFS device. The message used as input to the MAC calculation is the concatenation of the fields in the RPMB packet.

MAC calculation includes a 16Byte nonce (random number) generated by the host.

Reference:

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

9.4.3.3 RPMB Message Components

Each RPMB message should include specific components. These components are displayed in Table 9-1.

Table 9-1 — RPMB Message Components

Component Name	Length	Required for Requests	Required for Responses	Description
Req/Resp Message Type	2 bytes	Yes	Yes	See 9.4.4.1 & 9.4.4.2
Authentication Key and Message Authentication Code (MAC)	32 bytes (256bits)	Yes (Key or MAC)	Yes (MAC)	
Result	2 bytes	No	Yes	See 9.4.4.3
Write Counter	4 bytes	Yes	Yes	Total amount of successful authenticated data write requests.
Address	2 bytes	Yes	Yes	Address of data to be programmed to or read from the RPMB. Address is the serial number of the half sector (256B).
Nonce	16 bytes	Yes	Yes	Random number generated by the host for the requests and copied to response by the RPMB engine.
Data	256 Bytes	Yes	Yes	Data to be written or read by signed access.
Block Count	2 bytes	Yes	No	Number of blocks (half sectors, 256 B) requested to be read/programmed.

9.4.3.4 Messages from Host to Device

The following message types must be defined to support RPMB. These messages will be sent from the UFS Host to the UFS device.

- Authentication Key Programming request
- Read write Counter value
- Authenticated data write request
- Authenticated data read request
- Result Read Request

9.4.3.5 Messages from Device to Host

The following message types must be defined to support RPMB. These messages will be sent to the Host from the Device. The value in parenthesis will be the numerical value assigned to represent the request type.

- Authentication key programming response
- Write Counter value (response to request)
- Authenticated Data write response
- Authenticated Data read response

Table 9-2 — Request Message and 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
Response Message Types	
0x0100	Authentication key programming response
0x0200	Reading of the Write Counter value response
0x0300	Authenticated data write response
0x0400	Authenticated date read response

9.4.3.6

9.4.3.7 RPMB Operation Result

- RPMB Operation Result shall contain two bytes.
- The MSB shall indicate if the write counter has expired or (i.e. reached its max. value) not
  - Value of 0x8 will represent an expired write counter
  - Value of 0x0 will represent a valid write counter
- The LSB shall indicate the operation status
  - Operation Okay (0x0)
  - General Failure (0x1)
  - Authentication Failure (0x2)
    - MAC comparison not matching, MAC calculation failure
  - Counter Failure (0x3)
    - Counters not matching in comparison, counter incrementing failure
  - Address Failure (0x4)
    - Address out of range, wrong address alignment
  - Write Failure (0x5)
    - Data, Counter or result write failure
  - Read Failure (0x6)
    - Data, Counter or result write failure
  - Authentication key not yet programmed (0x7)
    - This value is the only valid result until the authentication key has been programmed (after which it can never occur again)

Table 9-3 — RPMB Operation Result data structure

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

9.4.3.7 RPMB Operation Result (cont'd)

Table 9-4 — RPMB Operation Results

Operation Results (The values in parenthesis are valid when Write Counter has expired)	
0x00 (0x80)	Operation OK
0x01 (0x81)	General failure
0x02 (0x82)	Authentication failure (MAC comparison not matching, MAC calculation failure)
0x03 (0x83)	Counter failure (counters not matching in comparison, counter incrementing failure)
0x04 (0x84)	Address failure (address out of range, wrong address alignment)
0x05 (0x85)	Write failure (data/counter/result write failure)
0x06 (0x86)	Read failure (data/counter/result read failure)
0x07	Authentication Key not yet programmed. This value is the only valid Result value until the Authentication Key has been programmed. Once the key is programmed, this Result value will no longer be used.

9.4.4 Implementation

9.4.4.1 Data Frame for RPMB Message

RPMB Message Data Frame size is 512Bytes, organized in the manner below.

Table 9-5 — RPMB Message Data Frame

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

9.4.4.2 MAC Calculation

The key used for the MAC calculation is always the 256 bit Authentication Key stored in the device. Input to the MAC calculation is the concatenation of the fields in the RPBM message data frame excluding stuff bytes and the MAC itself – i.e., 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 in which the data frames are sent. The MAC is added only to the last data frame.

9.4.4.3 RPMB Message Data Frame Delivery

The RPMB functions will be implemented by using Security Protocol commands -- Security Protocol In and Security Protocol Out -- to deliver the RPMB message data frame between the host and UFS device.

9.4.5 Security Protocol In/Out Commands

Security Protocol In/Out commands described in SPC-4 is used to encapsulate and deliver data packets of any security protocol between host and UFS device without interpreting, dis-assembling or re-assembly the data packets for delivery.

The Security Protocol In/Out commands contains a Security Protocol field. A unique Security Protocol ID is assigned by T10 for JEDEC UFS application.

- Security Protocol = ECh for JEDEC UFS application

UFS specification further assigns unique identifiers in the Security Protocol Specific field of the Security In/Out commands for RPMB and all other future security protocols used in UFS application.

- RPMB Protocol ID = 0001h

9.4.5.1 CDB format of Security Protocol In/Out commands

Table 9-6 — CDB format of Security Protocol In/Out commands

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	SECURITY PROTOCOL (UFS = ECh)							
2	SECURITY PROTOCOL SPECIFIC FIELD (UFS Protocol ID: UFS = 0001h)							
3								
4	INC_512	RESERVED						
5	RESERVED							
6	ALLOCATION LENGTH							
7								
8								
9								
10	RESERVED							
11	CONTROL							

For UFS v1.0, Security Protocol In/Out commands shall consider the unique Security Protocol ID assigned to JEDEC UFS application as the only valid Security Protocol ID.

For UFS v1.0, INC_512 bit is set to default = 0b.

9.4.5.2 Security Protocol In/Out Command Opcodes

- Security Protocol In = A2h
- Security Protocol Out = B5h

9.4.5.3 Security Protocol Information Query

As required by T10 SPC-4, the SECURITY PROTOCOL value of 00h shall be supported if the SECURITY PROTOCOL IN command is supported by the device. The security protocol information security protocol (i.e., the SECURITY PROTOCOL field set to 00h in a SECURITY PROTOCOL IN command) is used to transfer security protocol related information from the target logical unit.

When the SECURITY PROTOCOL field is set to 00h in a SECURITY PROTOCOL IN command, the 2-byte SECURITY PROTOCOL SPECIFIC field shall contain a numeric value as defined below.

Table 9-7 — Security Protocol Information Query

Security Protocol Specific Field Value	Description	Support
0000h	Supported security protocol list	Mandatory
0001h	Certificate data	Mandatory
0002h-FFFFh	Reserved	

9.4.5.4 Supported security protocols list description

If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0000h in a SECURITY PROTOCOL IN command, the parameter data shall have the format shown in Table 9-8.

Table 9-8 — Supported security protocols list

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
5								
6	SUPPORTED SECURITY PROTOCOL LIST LENGTH (m-7)							
7								
8	SUPPORTED SECURITY PROTOCOL [first] (00h)							
m	SUPPORTED SECURITY PROTOCOL [last]							
m+1	Pad Bytes (optional)							
n								

Security Protocol 00h and the UFS Security Protocol ID = ECh are the only valid Security Protocol ID’s supported by UFS v1.0.

9.4.5.5 Certificate data description

If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0001h in a SECURITY PROTOCOL IN command, the parameter data shall have the format shown in Table 9-9.

Table 9-9 — Certificate data

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	CERTIFICATE LENGTH (m-3)							
3								
4	CERTIFICATE							
m								
m+1	Pad Bytes (optional)							
n								

For UFS v1.0, the Device Server does not have a certificate to transfer, the CERTIFICATE LENGTH field shall be set to 0000h.

9.4.6 RPMB Operations

9.4.6.1 Request Type Message Delivery

- Host sends Req type message to UFS device to request an operation by the UFS device or to deliver data to be written into the RPMB memory block.
- To deliver a Req type message, the host sends the Security Protocol Out command in a Command UPIU in the Command Phase of a SCSI transaction.

Table 9-10 — Security Protocol Out command

Command UPIU			
xx00 0001b	Flag.W = 1 .R=0	LUN	Task Tag
Command Set = 01h	Reserved	Reserved	Reserved
Total EHS Length = 0	Reserved	Data Segment Length = 0	
Expected Data Transfer Length = 512			
Security Procotol Out CDB			
Header E2ECRC (omit if HD=0)			

- The device indicates to the host that it is ready for the operation by returning a Ready-to-Transfer UPIU.

9.4.6.1 Request Type Message Delivery (cont'd)

Table 9-11 — Ready To Transfer

Ready To Transfer UPIU			
xx11 0001b	Flag.F = 1	LUN	Task Tag
Reserved	Reserved	Reserved	Reserved
Total EHS Length = 0	Reserved	Data Segment Length = 0	
Data Buffer Offset = 0			
Data Transfer Count = 512			
Reserved			
Reserved			
Reserved			
Header E2ECRC (omit if HD=0)			

- The entire RPBM message data frame is then delivered to the device by a Data-Out UPIU in the Data Phase.

Table 9-12 — RPBM message data frame

Data Out UPIU			
xx00 0010b	Flag.F = 1	LUN	Task Tag
Reserved	Reserved	Reserved	Reserved
Total EHS Length = 0	Reserved	Data Segment Length = 512	
Data Buffer Offset = 0			
Data Transfer Count = 512			
Reserved			
Reserved			
Reserved			
Header E2ECRC (omit if HD=0)			
RPMB Data Frame (512B)			
Data E2ECRC (omit if DD=0)			

9.4.6.1 Request Type Message Delivery (cont'd)

- To complete the operation, the device returns a Response UPIU with the status of the operation in the Status Phase.

Table 9-13 — Response UPIU

Response UPIU			
xx10 0001b	Flag.F = 1	LUN	Task Tag
Reserved	Reserved	Response	Status
Total EHS Length = 0	Reserved	Data Segment Length = 0	
Residual Transfer Count = 0			
Reserved			
Reserved			
Reserved			
Reserved			
Header E2ECRC (omit if HD=0)			
Sense Data Length		Sense Data[0]	Sense Data[1]
…	…	…	…
Sense Data[14]	Sense Data[15]	Sense Data[16]	Sense Data[17]
Data E2ECRC (omit if DD=0)			

- The host/device transactions can be reduced by utilizing Phase Collapse described in the UTP chapter.

9.4.6.2 Response Type Message Delivery

- Host sends Resp type message to UFS device to read the result of a previous operation request or to read data from the RPMB memory block.
- To deliver a Resp type message, the host sends the Security Protocol In command in a Command UPIU in the Command Phase of a SCSI transaction.

Table 9-14 — Response Type Message Delivery: Command UPIU

Command UPIU			
xx00 0001b	Flag.W=0 .R=1	LUN	Task Tag
Command Set = 01h	Reserved	Reserved	Reserved
Total EHS Length = 0	Reserved	Data Segment Length = 0	
Expected Data Transfer Length = 512			
Security Protocol In CDB			
Header E2ECRC (omit if HD=0)			

- The device returns the result or data requested in the RPMB message. The entire RPBM message data frame is delivered from the device to the host in a Data-In UPIU in the Data Phase.

Table 9-15 — Response Type Message Delivery: Data In UPIU

Data In UPIU			
Xx10 0010b	Flag.F = 1	LUN	Task Tag
Reserved	Reserved	Reserved	Reserved
Total EHS Length = 0	Reserved	Data Segment Length = 512	
Data Buffer Offset = 0			
Data Transfer Count = 512			
Reserved			
Reserved			
Reserved			
Header E2ECRC (omit if HD=0)			
RPMB Data Frame (512B)			
Data E2ECRC (omit if DD=0)			

9.4.6.2 Response Type Message Delivery (cont'd)

- To complete the operation, the device sends a Response UPIU with the status of the operation in the Status Phase.

Table 9-16 — Response Type Message Delivery: Response UPIU

Response UPIU			
xx10 0001b	Flag.F = 1	LUN	Task Tag
Reserved	Reserved	Response	Status
Total EHS Length = 0	Reserved	Data Segment Length = 0	
Residual Transfer Count = 0			
Reserved			
Reserved			
Reserved			
Reserved			
Header E2ECRC (omit if HD=0)			
Sense Data Length		Sense Data[0]	Sense Data[1]
…	…	…	…
Sense Data[14]	Sense Data[15]	Sense Data[16]	Sense Data[17]
Data E2ECRC (omit if DD=0)			

- The host/device transactions can be reduced by utilizing Phase Collapse described in the UTP chapter.

9.4.6.3 Authentication Key Programming

- The Authentication Key programming is initiated by a Security Protocol Out command
- The RPMB data frame delivered from the host to the device includes the Request Message Type = 0x0001h and the Authentication Key
- The device returns “Good” status in status response when Authentication Key programming is completed
- Host initiates the Authentication Key programming verification process by issuing a Security Protocol Out command with delivery of a RPMB data frame contains the Request Message Type = 0x0005h
- The device returns “Good” status in status response when the verification result is ready for retrieval
- Host retrieves the verification result by issuing a Security Protocol In command
- Device returns the RPBM data frame containing the Response Message Type = 0x0100h and the Result code [3:2]

  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).

  Access to Reply Protected Memory Block is not allowed/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.

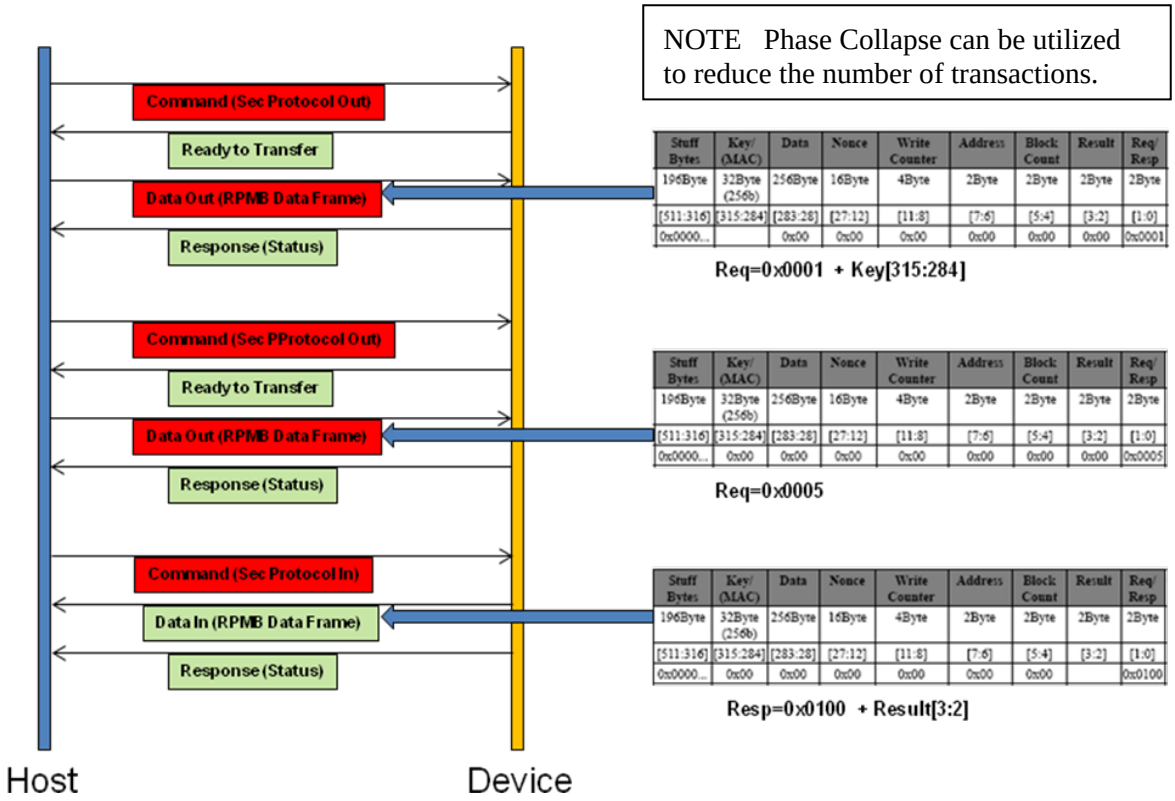

Figure 9-1 — Authentication Key Programming Flow

9.4.6.4 Read Counter Value

- The Read Counter Value sequence is initiated by a Security Protocol Out command
- The RPMB data frame delivered from the host to the device includes the Request Message Type = 0x0002h
- When the host received a “Good” status in the status response from the device, it sends a Security Protocol In command to the device to retrieve the Counter value
- The device returns a RPMB data frame with Response Message Type = 0x0200 + Result[3:2] + Counter[11:8] + Nonce[27:12] + MAC[315:284]

  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).

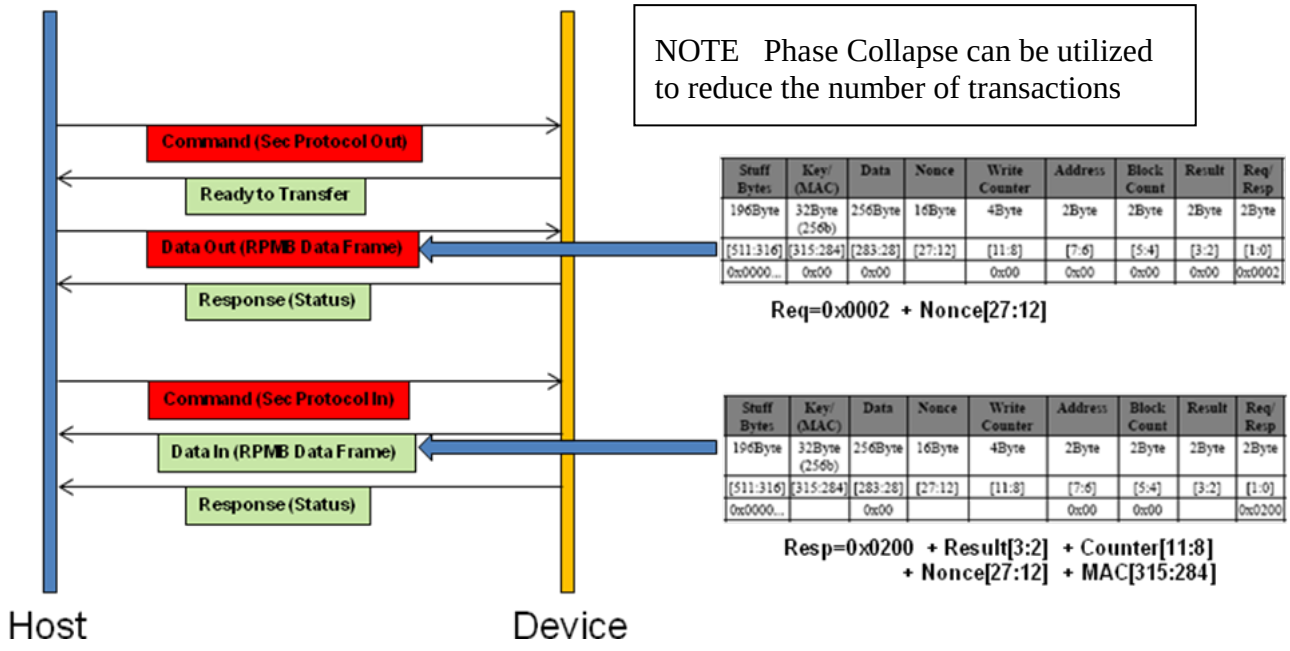

Figure 9-2 — Read Counter Value Flow

9.4.6.5 Authenticated Data Write

- The Authentication Key programming is initiated by a Security Protocol Out command
- The RPMB data frame delivered from the host to the device includes the Request Message Type = 0x0003h, Block Count, Address, Write Counter, Data and MAC
    - When the device receives this RPMB message data frame, it first checks whether the write counter has expired. If the write counter is expired then the device sets the result to 0x85 (write failure, write counter expired). No data is written to the RPMB data area.
    - 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). No data are written to the RPMB data area.
    - If the write counter was not expired then the device calculates the MAC of request type, block count, write counter, address and data, and compares this with the MAC in the request. If the two MAC's are different, then the device sets the result to 0x02 (authentication failure). No data are written to the RPMB data area.
    - If the MAC in the request and the calculated MAC are equal then the device compares the write counter in the request with the write counter stored in the device. If the two counters are different then the device sets the result to 0x03 (counter failure). No data are written to the RPMB data area.
    - 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).
    - 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 half sectors).
- The device returns "Good" status in status response when Authenticated Data Write operation is completed regardless of whether the Authenticated Data Write is successful or not
- The successfulness of the programming of the data should be checked by the host by reading the result register of the RPMB. Host initiates the Authenticated Data Write verification process by issuing a Security Protocol Out command with delivery of a RPMB data frame contains the Request Message Type = 0x0005h
- The device returns "Good" status in status response when the verification result is ready for retrieval
- Host retrieves the verification result by issuing a Security Protocol In command
- Device returns the RPBM data frame containing the Response Message Type = 0x0100h and the Result code [3:2]

9.4.6.5 Authenticated Data Write (cont'd)

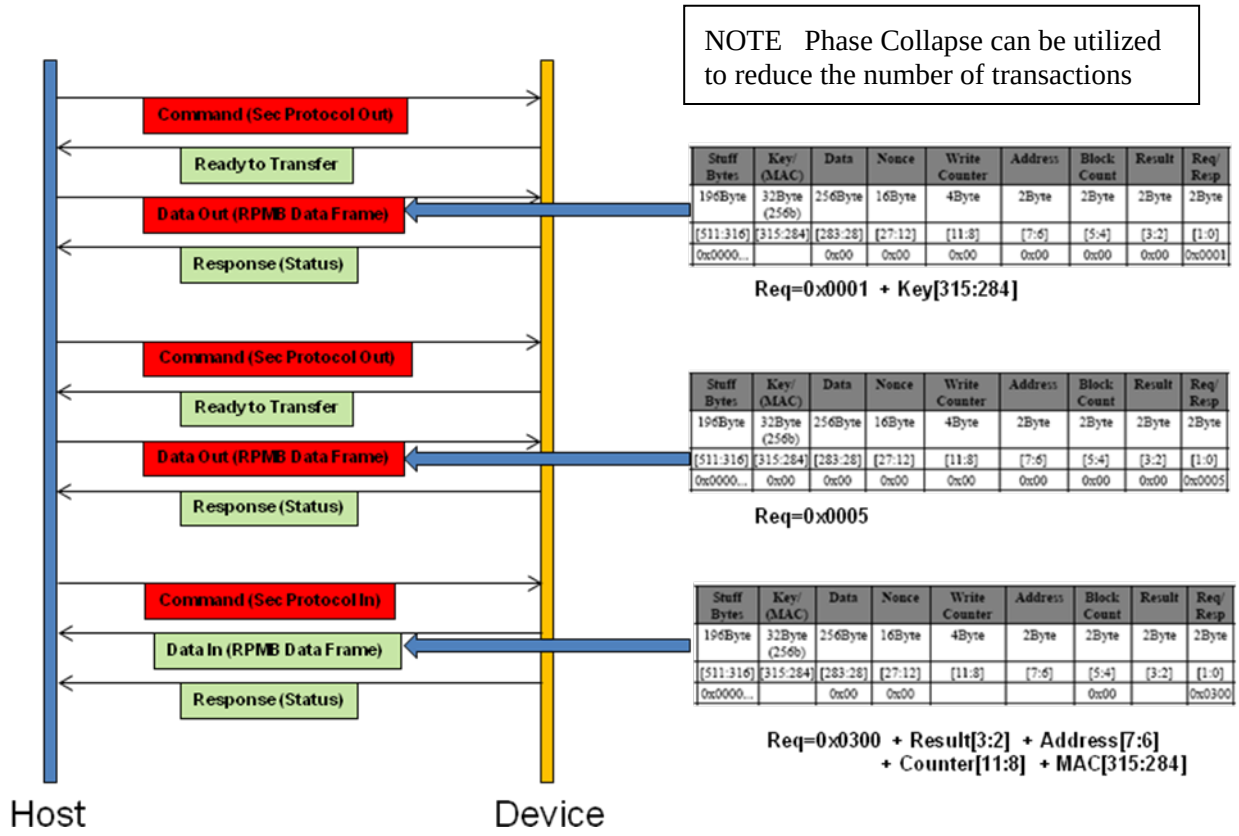

Figure 9-3 — Authenticated Data Write Flow

9.4.6.6 Authenticated Data Read

- The Authenticated Data Read sequence is initiated by a Security Protocol Out command
- The RPMB data frame delivered from the host to the device includes the Request Message Type = 0x0004h
  - When the device receives this request it first checks the address. If there is an error in the address then result is set to 0x04 (address failure). The data read is not valid.
  - After successful data fetch the MAC is calculated from response type, nonce, address, data and result. If the MAC calculation fails then returned result is 0x02 (Authentication failure).
- When the host received a "Good" status in the status response from the device, it sends a Security Protocol In command to the device to retrieve the data
- The device returns a RPMB data frame with Response Message Type = 0x0400 + Result[3:2] + Block Count[5:4] + Counter[11:8] + Nonce[27:12] + Data[283:28] + MAC[315:284]
- 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 half sectors).

9.4.6.6 Authenticated Data Read (cont'd)

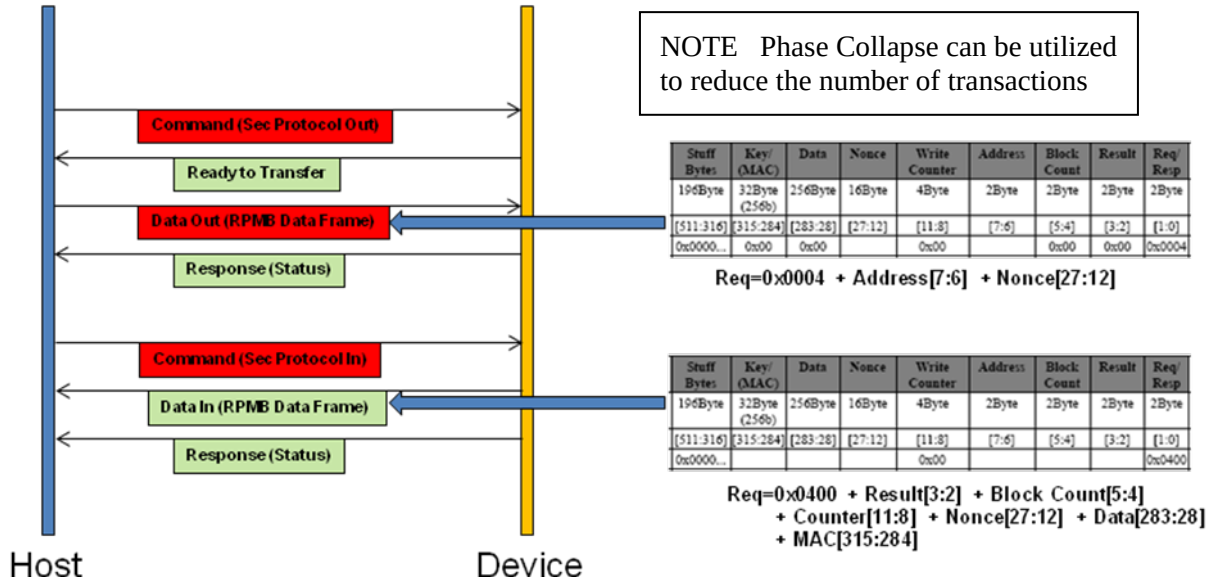

Figure 9-4 — Authenticated Read Flow

9.5 Malware Protection

The UFS spec will also have the option to protect boot, bus configuration settings and other important device configuration settings so that once they are set they cannot be modified. The implementation of the protection of these parameters will be defined within the spec where the parameter is defined.

9.6 Reset

The reset requirements for an embedded and removable UFS device will be different. This is a result of the limited pins available for the removable UFS device.

There will be different types of resets available for a UFS device.

Table 9-17 — Reset Types

Type	Description
PHY reset	Squelch Mode reset implemented at PHY level. This puts the phy into the sleep state.
Soft reset	This will be a reset command implemented at the protocol layer that will terminate the current operation on the specific device.
Hardware reset	An external pin that will terminate operations on all UFS devices and cause the device to return to the power up state or “reboot” state if supported.
Power on reset	This is an implied reset that is supported if the hardware reset is not supported. The MIPI PHY requires that both the host and device controllers be held in a reset state until both are powered.

The host controller:

- Initiator of the PHY and soft reset.
- It is not mandatory for a host to support the hardware reset pin.
- It is recommended that the host either select to permanently disable or enable the hardware reset pin.

The embedded UFS device:

- Must have a hardware reset source that it not controlled by software. This will be implemented by a separate hardware reset pin on the device, which will be active low. This reset pin will be disabled by default and either permanently disabled or enabled via a parameter on UFS device.
- The embedded device will support both the PHY and soft reset as well.
- Will implement the internal power on reset to comply with the PHY.

The removable UFS device:

- Will not have a dedicated reset pin.
- Will support both the PHY and soft reset as well.
- Will implement the internal power on reset to comply with the PHY.

9.6.1 Implementation

9.6.1.1 RST_ENABLE Parameter

- Parameter is R/W
- ‘11’ undefined
- ‘10’ hardware reset is permanently enabled
- ‘01’ hardware reset is permanently disabled
- ‘00’ hardware reset is disabled (default)

9.6.1.2 Bus Components

It is expected that the UFS device will not require any external components on any receive, transmit or control lines between the host and the device. Any passive components that may be required must be incorporated as part of the host or device.

9.7 Mechanical

Packaging and requirements for UFS embedded device should adhere to the following guidelines if possible

- Reset and data transfer pins should be located in the second (PoP) or third row (MCP) in from the side of the package to prevent access.

9.8 UFS Security vs. eMMC

Table 9-18 — UFS Security vs eMMC

Item	eMMC	UFS
Secure Device Class	RPMB is required and security features optional (register indicates support)	If a device is a secure device it must support all secure features otherwise support is optional
Secure Removal	Defined a secure version of both the Trim and Erase command.	A parameter will be set in each partition to indicate whether any removal operation will be secure or not.
Secure Removal Method	Only one method supported	Supports different removal types to support different requirements and different technologies.
Force Garbage Collection	Not Supported	Allow the host to issue a device clean which would remove all garbage from the device.
Data protection	Data protection can be applied to a write group or to the entire device.	Data protection only to a partition.
Protection Types	Permanent Power on Password Temporary	Permanent Power on Temporary

10 UFS FUNCTIONAL DESCRIPTIONS

10.1 UFS Boot

10.1.1 Boot Requirements

Some computing systems can have the need to download the system boot loader from an external non-volatile source. This task can be accomplished through an internal Boot ROM contained in the Host SOC whose code when executed determines a minimal initialization of the system to start the boot code transfer.

UFS Specification version 1.0 defines the boot procedure for a point to point connection.

The boot functionality shall ensure a complete configurability in order to be adapted to different system requirements.

Moreover security features should be guaranteed for the overall operation to ensure boot data integrity and no corruption of boot code.

10.1.2 Boot Configuration

During boot operation the UFS Host Controller retrieves the system boot code stored in single embedded UFS devices.

In UFS Specification version 1.0 the boot mechanism shall be applied to a point-to-point topology (see Figure 10-1):

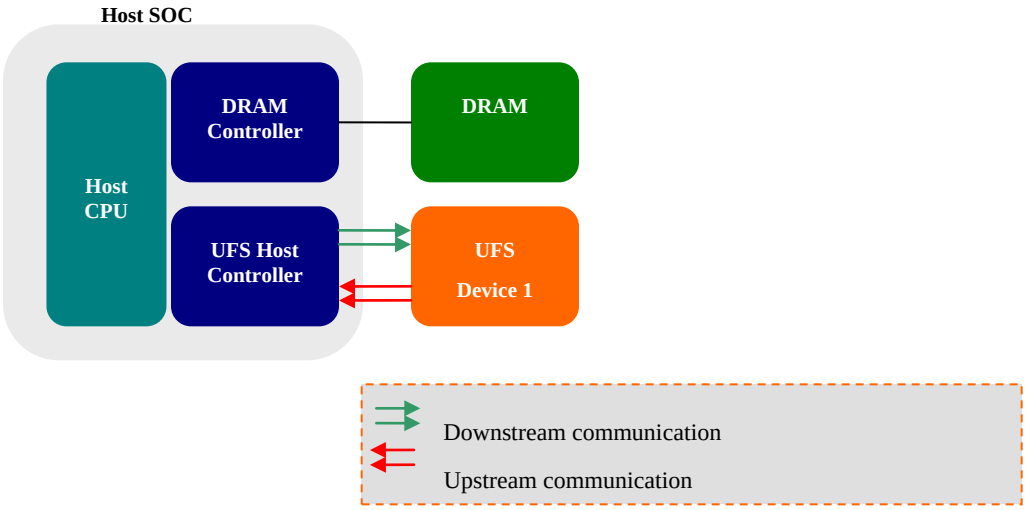

Figure 10-1 — UFS System Diagram

Several fields of the Configuration Descriptor shall be configured to define the device behavior during boot. Note that only the embedded UFS device supports the boot feature.

For a UFS bootable device, the bBootEnable field in the device Configuration Descriptor shall be set to enable the boot process.

10.1.2 Boot Configuration (cont'd)

Two LUs (Boot LU A, Boot LU B) can be used to store the Boot code but only one is active during the boot process. The user shall configure the LUs by writing the corresponding Logical Unit Configuration part of the Configuration descriptor. In particular the size shall be defined through the # OF ALLOCATION BLOCKS field, and the LUN shall be associated to Boot LU A or Boot LU B through the bBootLunID field.

The Boot Logical Unit active during the boot shall be selected configuring the bBootLunEn attribute, as described in Table 10-1.

Table 10-1 — bBootLunEn Attribute

bBootLunEn	Description
0b000	Boot LU A = disabled Boot LU B = disabled
0b001	Boot LU A = enabled Boot LU B = disable
0b010	Boot LU A = disable Boot LU B = enabled
Others	Reserved

The host shall not attempt to set bBootLunEn to 'Reserved' values, and UFS device shall generate an error in case of an attempt to set 'Reserved' values and not execute the request.

When bBootLunEn attribute is 0b000 the boot feature is disabled, the device behaves as if bBootEnable would be equal to 0.

The active Boot LU will be mapped onto the Well-Known LU 0hB0 once the bBootLunEn has been properly configured.

10.1.2 Boot Configuration (cont'd)

Figure 10-2 shows an example of a UFS device having eight LUs: LU 1 and LU 4 are configured, respectively, as Boot LU A and Boot LU B. In particular, LU 1 is the active one (bBootLunEn = 0b001).

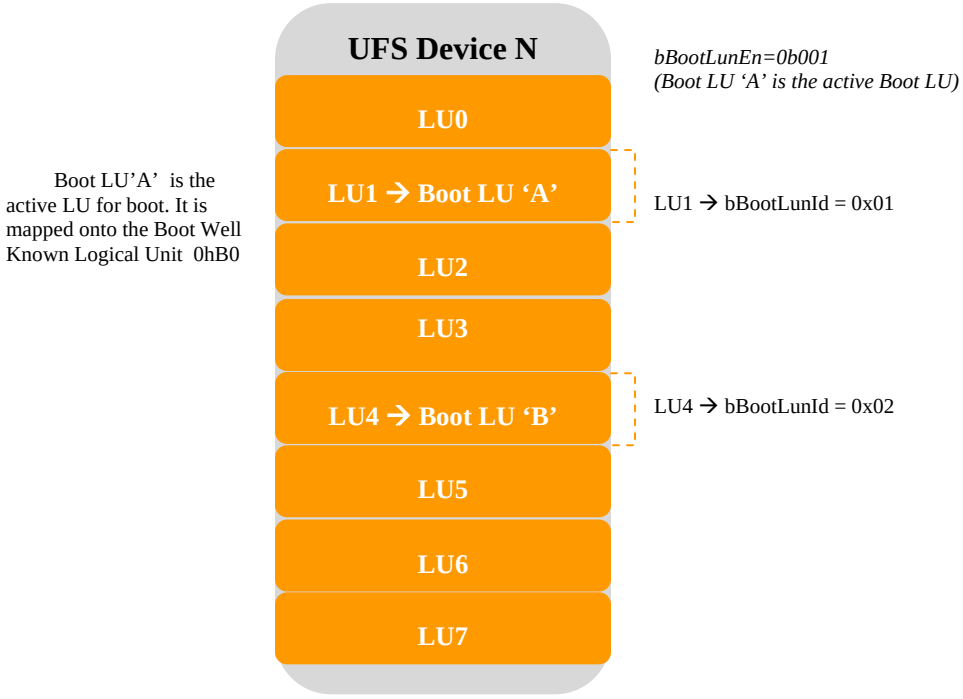

Figure 10-2 — Example of UFS Device Memory Organization for Boot

10.1.3 Boot Process

Boot execution is made up of the following steps: Initialization, Boot Transfer and Initialization Completion.

10.1.3.1 Initialization

The initialization phase starts after power-up and involves the entire UFS stack. At the end of this phase, the UniPro boot sequence shall be completed, and the UTP layer shall be capable of accessing configuration descriptors (if the bShortBootEnable field of the Configuration Descriptor is ’0’) and exchanging UPIU for SCSI READ command.

Each single layers in the UFS Protocol stack execute the initialization process on both UFS Host and UFS Device sides.

Physical Layer (M-PHY)

After reset events, the Physical layer will move from DISABLED state to LOW SPEED BURST state

Link Layer (UniPro)

On Host and Device side UniPro Boot sequence takes place:

1. DME will send a Layer Management Reset Request and each layer will generate a specific acknowledge (Layer Management Reset Confirm).
2. DME will issue a Link Management Enable Layer Request for each single UniPro layer.
3. When a UniPro layer has completed its own initialization process a Link Management Enable Layer Confirm is issued.
4. The DME initiates the Link StartUp sequence which consists of a multiphase handshake exchanging UniPro trigger events to establish initial link communication, in both directions, between UFS host and device.

10.1.3.2 Boot Transfer

The following steps can be executed only if bBootEnable field is set.

Link Configuration. The host may configure the Link Attributes (i.e. Gear, HS Serie, PWM Mode in Rx and Tx) by DME primitives at UniPro level.

The UFS Host Controller may optionally discover relevant device info for the boot process by accessing the Device Descriptor (i.e. Device Class/Subclass, Boot Enable, Boot LUs size, etc.). The UFS Host Controller is allowed to access the Device Descriptor only if the bShortBootEnable is ‘0’, otherwise it can be accessed only when the device has completed the initialization phase.

Boot code download. The UFS Host Controller reads the well known boot LU by issuing SCSI READ(10) commands and the UFS device will start to send the boot code on the Upstream Link.

During this phase only the well-known boot LU is accessible. The well known boot LU shall accept read command, while other LUs may not be ready and the host shall not access them.

10.1.3.3 Initialization Completion

After the host has completed the boot code download from the configured logical unit, it shall complete the initialization process. In particular, the host shall set the bDeviceInit flag to communicate the UFS device that it can proceed complete the complete initialization. The device will reset the bDeviceInit flag when the initialization is complete. The host shall poll the bDeviceInit flag to check the completion of the process. When the bDeviceInit is reset, the device is ready to accept any command.

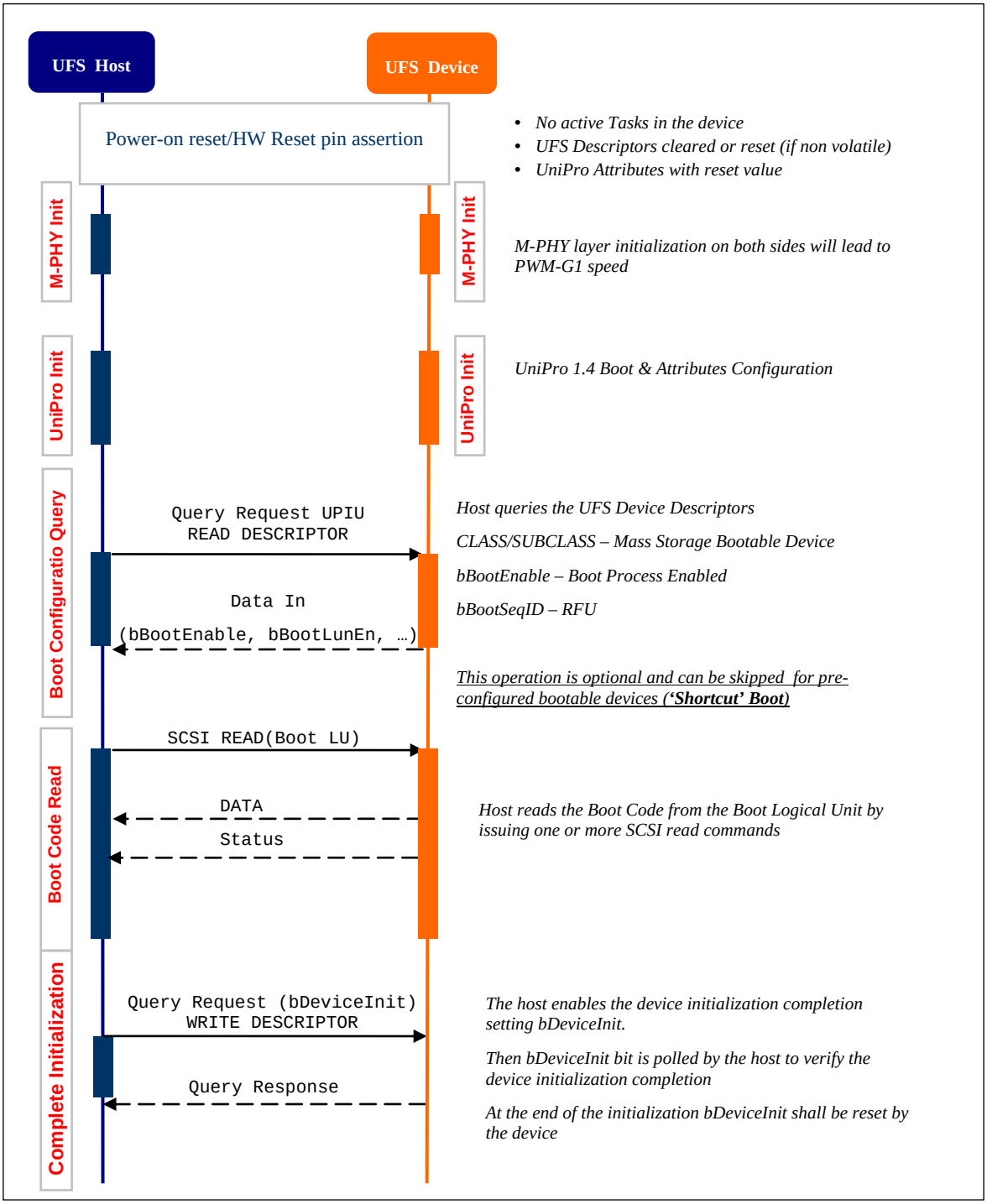

Figure 10-3 — Boot Procedure Sequence Diagram

10.1.4 Boot LUNs Operations

Boot Logical Units shall be programmed to store the boot code during the system manufacturing phase and they may be also updated during the system lifecycle. The same area can be read to verify its content.

Therefore the following operations are permitted on the Boot LUs:

1. boot code program – for boot code upload/update
2. boot code read – to verify the content programmed
3. boot code removal – to remove the content of the Boot LU

Those operations can be executed regardless the bBootEnable field value in the Device Descriptor.

10.1.5 Configurability

The boot process is configurable through several fields in the Configuration Descriptors (see paragraph 10.3.7) to adapt it to different usage models and system features.

The following fields refer to boot capabilities:

- Boot Device Enable
- Boot LU(s) Size
- Configured Boot LU (active LU for Boot)
- Boot Sequence ID (for future specs evolutions)
- Upstream/Downstream configuration after boot

These parameters are non volatile and are characterized by different levels of changeability (R/W, R/W/E, etc.). The R/W (one time writeable) fields may be programmed during the system manufacturing phase.

10.1.6 Security

10.1.6.1Boot Area Protection

Boot areas might be protected in order to avoid boot code alteration by a third party, so the write/erase protection mechanism defined for Logical Units can be applied to the Boot Logical Units as well.

Optionally a power-on read protection can be set-up on Boot Logical Units after completing the device initialization.

10.1.6.2 Boot Configuration Parameters Protection

Boot configuration register settings might be protected in order to make it inviolable by a third party.

10.2 Partition Management

10.2.1 Requirements

The functionality aims to provide a mechanism to let an external application define and use a virtual memory organization which could easily fit different usage models in a versatile way.

Besides segmenting the available addressable space, the mechanism introduces the possibility of differentiating each partition through dedicated functionalities and features.

The specification gives the procedure to configure the partitions in terms of number of partitions, size, attribute. Security features are added following spec [2].

A UFS memory area can be partitioned in different parts each one with independent logical address ranges and singularly accessible.

Moreover each partition can be defined for a specified use and with peculiar attributes in order to be adapted to different UFS host usage models and Operating Systems requirements.

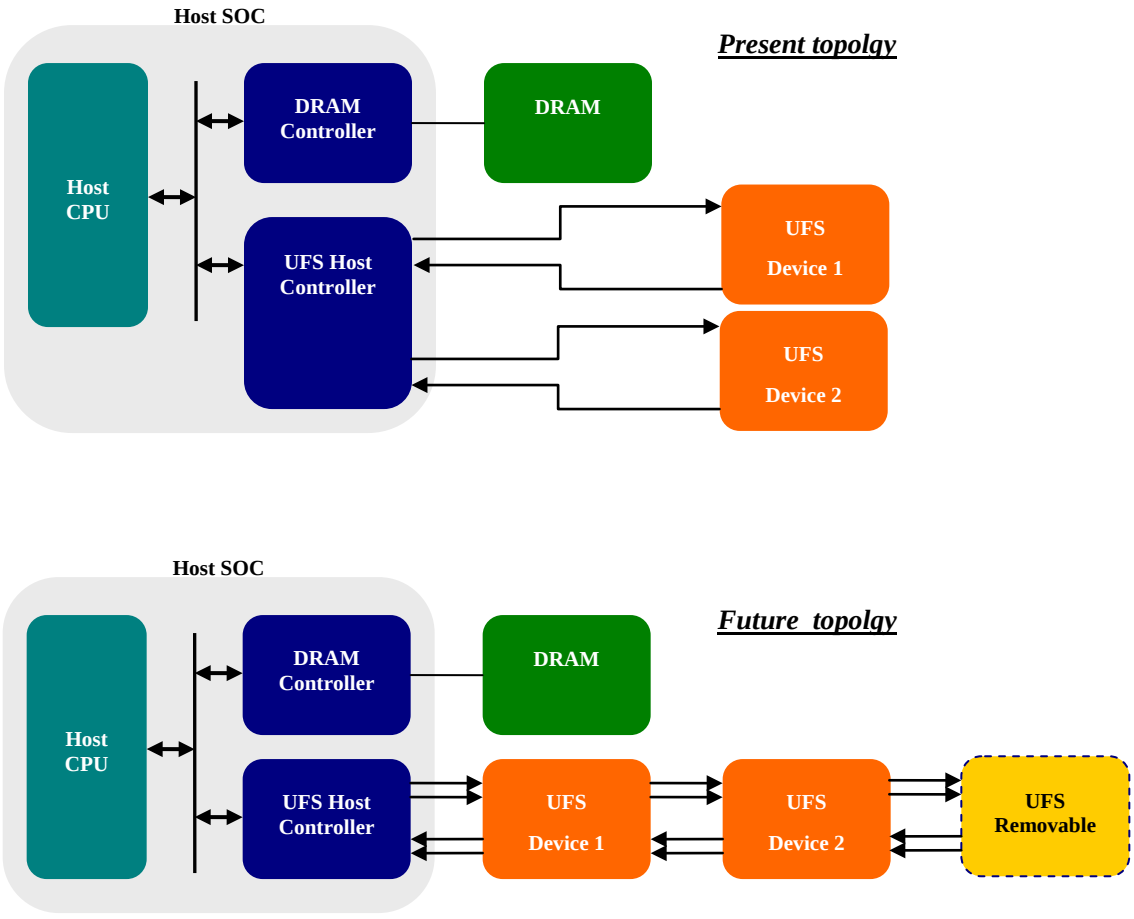

Figure 10-4 — UFS System diagram

10.2.2 Partitions features

An UFS device shall be partitioned by the UFS Host Controller in different memory areas at Transport Layer level. In particular, starting from the SCSI over UniPro nomenclature, partitions will be denoted as LUNs and will be characterized by the fact that they will have independent logical addressable spaces starting from the logical address 0x00000000.

Some of the LUNs which can be configured on the device will have a specific purpose while others can be ‘general purpose’ areas in order to fulfill different host software/hardware requirements:

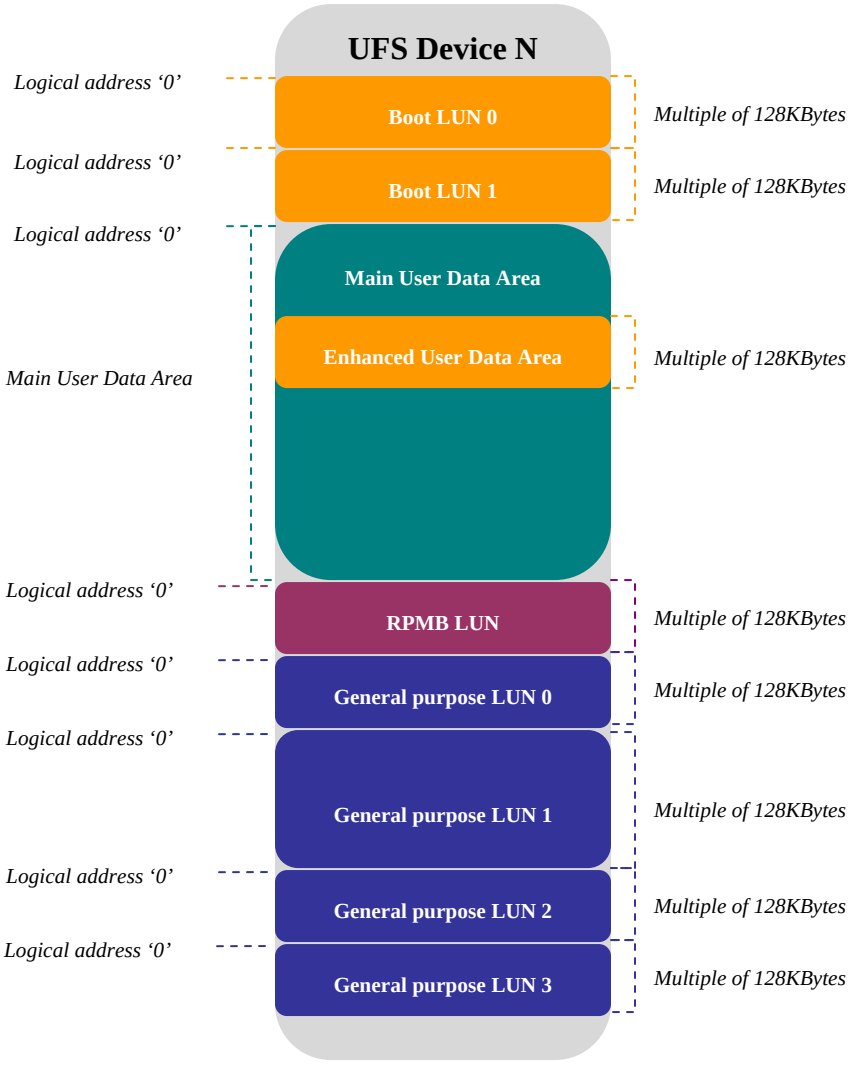

Figure 10-5 — UFS Device Memory Organization

Each LUN will have a physical implementation on the non-volatile storage media.

In particular, the LUNs organization in a single UFS device will be the following one:

- 2 Boot LUNS
- 1 RPMB LUN
- 4 General Purpose LUNs

10.2.2 Partitions features (cont'd)

The two boot LUNs will contain the system boot code and will be involved in the boot procedure (see [4]); the RPMB LUN is accessed by authenticated operations by a well defined security algorithm (see [2]). The four general purpose partitions will be used to fulfill other use cases.

Common features of each LUN are:

- independent logical addressable spaces (starting from logical address 0x00000000 up to the LUN size)
- LUN size defined as multiple of 128 Kbytes

Moreover each LUN can be characterized by one ore more attributes specified by each memory manufacturer. Examples of attributes to differentiate LUNs properties are the following ones:

- High performance attribute – higher write/read performances vs. more relaxed data reliability in the specified LUN
- High data reliability attribute – more robust data reliability to the detriment of write and read performances
- High reliability attribute – the area is characterized by a higher read/write endurance
- No data management attribute – the data management algorithms (i.e. wear leveling) can be disabled in the selected LUN
- Accessibility properties – i.e. ROM LUNs
- Proprietary security attribute – not standardized security algorithm implemented in the partition

The definition of these attributes is left open in order to accomplish different needs and vendors specific implementations.

Beside the LUNs also the Main User Data Area can contain sub-areas characterized by a certain attribute (i.e., write/read high performance area) with the peculiarity of having contiguous logical addresses inside the Main User Data Area.

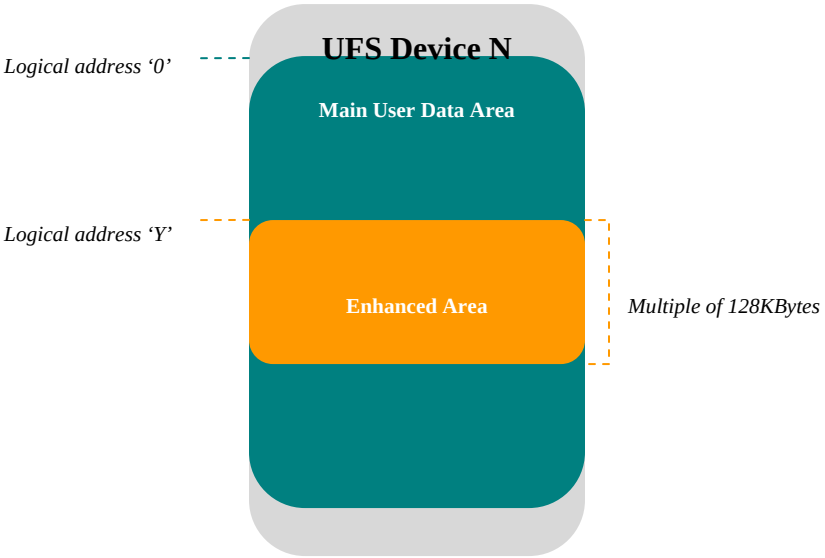

Figure 10-6 — Sub-areas inside the Main User Data Areas

10.2.3 Partitions configuration

In a UFS device supporting partitioning features, the following LUNs will be mandatory at time zero:

- 2 Boot LUNs/Boot Area inside the Main User Data Memory Area
- 1 RPMB LUN

Instead, the UFS Host Controller shall configure LUNs according to Table 10-2:

Table 10-2 — LUNs configurable by the UFS Host Controller

LUN type	Configurable parameters
General Purpose LUN 1 – 4	Size Attribute
Enhanced Area	Size Attribute Start Address

The configurable LUNs is a '*one-time-programmable*' operation.

It is recommended to execute LUNs configuration during the system manufacturing phase.

10.2.4 LUNs Access

Enhanced LUN in the Main User Data Area, Boot LUNs, General Purpose LUNs and RPMB LUN access shall be enabled writing in a proper register. After doing it, the selected LUN is accessible by SCSI command in case of Bootable Mass Storage devices or XiP commands in case of XiP devices.

10.2.5 LUNs Protection

Mechanisms of protection can be defined for each LUN. Write /erase protection features types are:

- Permanent write and erase protection (permanent read only)
- Power on and reset write and erase protection (read only for power on period)
- Temporary write and erase protection

Write protection will not be implemented in the RPMB LUN. Boot partition can be entirely protected in one of the three modes defined. In General Purpose LUNs and User Data / Enhanced User Data Area, Write / Erase protections can be set up on smaller regions.

Status and type of protection in the small regions shall be tracked by the device and can be read by the host.

10.3 Host Device Interaction

10.3.1 Overview

This proposal provides mechanisms for the host and device to communicate about pending operations so that the performance and/or reliability of the device can be improved.

10.3.2 Applicable Devices

All the features described in this document should be implemented on all UFS devices. The extent to which the features are implemented will be up to the device manufacturer. It is expected that poor implementations will result in lower device performance or reliability and higher max power consumption in the low power modes.

10.3.3 COMMAND QUEUE: Inter-LU Priority

The specific details of how commands interact in a queue of a specific LU are handled in other chapters. This section outlines how the queues within a device interact with each other. The UFS specification acknowledges that there may be many different implementations of a UFS controller. For example, it may be implemented as a single or multithreaded processor. In order to make the UFS spec implementation agnostic this section outlines a parameter that allow the host to communicate to the device its priorities so that the device can take this into account when executing commands from the host.

In the implementation case where several queues are serviced from a single execution unit, it is necessary for the host to either designate all queues has having the same priority or designate a single Queue has having a higher priority. A parameter value shall be defined to allow the host to designate a queue as having high or normal priority. In the case where a queue is designated as having a higher priority, whenever a command enters the queue with the high priority it will be executed as soon as possible resulting in commands from other queues being stalled.

One example of where a host may want to take advantage of this feature is when the host has allocated one LU to be a code execution unit and another unit to be a mass storage unit. In this example the execution of code takes priority over mass storage transfers, so the host would set the parameter bit associated with the code LU to be high priority and leave the remaining queues as lower priority.

The queue supports two explicit priorities (and a third implicit low priority):

1. High priority – This is used for high priority requests. Any commands in this queue will have higher priority than commands in other priority queues. For example, when servicing demand-paging applications, only read commands can have this priority.
2. Normal priority – This is used for all regular commands which do not belong to priority #1
3. (Lowest priority – This is an implicit priority used by background operations (not represented in real commands, background operations are executed internally))

10.3.3.1 Implementation

The following parameter will be defined to set the queue priority, per LU:

QUEUE_PRIORITY	0 = Normal Priority Queue 1 = High Priority Queue Reset value = 0 (normal)

10.3.4 BACKGROUND OPERATION MODE

10.3.4.1 Description

A managed device requires time to execute management tasks. The background operation mode grants the device time to execute the commands associated with these flash management tasks. Flash management operations could include, but are not limited to, wear-leveling, bad-block management, wipe, trim and garbage collection. The operations completed during the background operation period are determined by the device manufacturer and are not covered by the UFS spec.

10.3.4.2 Purpose

10.3.4.2.1 Performance Improvement

The intent of this mode is to improve a device response to host commands by allowing the device to postpone device management activities that occur as a result of host initiated operations to periods when the host is not using the device.

Systems that will use a UFS device tend to have peak periods of activity, in which the best response possible is needed from the UFS device. These peak periods are followed by idle periods, where the host can allow the device to do device management operations. Allowing better communication between the host and the device on when these idle periods will occur allows the UFS device to perform more optimally in the system.

The device will still be permitted to do device management when the host initiates operations, however the downside of doing this may be poorer device performance. This mode will just give device manufacturers the option to delay device management operations to improve performance. It is recommended that devices postpone as many tasks as possible to take full advantage of the possible performance improvements associated with this feature.

10.3.4.2.2 Host Power Management

This mode will also allow the host to control when the device uses power to perform management activities. The host will have more knowledge about the power consumed by the system and can make the appropriate tradeoffs about when to use system power and when to conserve it.

An example of where host control over power consumed by the device could be an advantage would be the case where the system has very little battery power and the UFS device has a lot of unused memory. In this case the host may not wish for the device to perform clean up operations but conserve the power for more critical system functions.

Allowing the host to communicate with the device on when activities can be performed will allow better system power management, which can be controlled by the host.

10.3.4.3 Background Status

A UFS device needs some mechanism to inform the host how critical it is for the device to complete management operations. The device will have a parameter that allows it to communicate this status to the host. This will be reported in four different levels:

- No operations required
- Operations outstanding (non critical)
- Operations outstanding (performance being impacted)
- Operations outstanding (critical)

The device will also use a single bit in the response that is sends to the host that will allow the host to easily monitor the UFS device's need for background operations. These two levels will be:

- No immediate need to service (corresponds to no operations and non critical)
- Service (corresponds to performance being impacted and critical)

It is expected that the host will respond as soon as possible when the status changes to service since if the background operations are not properly managed than the device could fail to operate in an optimal way.

In the case where the device status is operations outstanding (critical) this will mean that the device will only respond to mode sense and mode select commands. The host should put in all possible measures to ensure the device never reaches this state since it means the device is not longer able to operate.

The point at which the device enters each of these states is up to the manufacturer and is not covered by the UFS spec.

10.3.4.4 Operation Initiation

There is no explicit command in UFS to start the background operation. There is a mode bit defined that indicates whether the device is allowed to execute background operations. If the command queue is empty, the background operation mode bit is set and the device is active then the device will be allowed to execute any internal operations required.

The host will halt a background operation by sending a command to the device. The timeout defined for command response time, for when the background mode is enabled, will take into account the need to stop background operations. The host can minimize the device response time by either turning off background operations mode during critical performance times or by doing a read with high priority set.

10.3.4.5 Power Failure

It is the device's responsibility to ensure that the data in the device is not corrupted if a power failure occurs during a background operation.

10.3.4.6 Implementation

10.3.4.6.1 Background Operation Enable Mode

BACKGROUND_OPS_EN mode will be added to the host device interaction mode page.

- 0 = Device is not permitted to run background operations when the queue is empty.
- 1 =Device is permitted to run background operations when the queue is empty

The default value of this mode will be enabled.

10.3.4.6.2 Background Status Parameter

BACKGROUND_OPS_STATUS is a two bit parameter defined as follows: 0 = not required

- 1 = required, not critical
- 2 = required, performance impact
- 3 = critical.

10.3.4.6.3 Background Status Device Response Indicator

BACKGROUND_OPS_STATUS will be a bit defined in the SCSI device response.

- 0 = not required or not critical
- 1 = required.

10.3.5 POWER OFF NOTIFICATION

A UFS Host will notify the Device when it is going to power the Device off by requesting the device to move to the SLEEP or UFS-PowerDown modes. This will give the Device time to cleanly complete any ongoing operations. The Device will respond to the Host when it is ready for power off, meaning that the device entered the SLEEP or UFS-PowerDown modes. Host can then power off the device without the risk of data loss.

10.3.6 DYNAMIC DEVICE CAPACITY

Common storage devices assume a fixed capacity. This presents a problem as the Device ages, and get closer to its end of life. When some blocks of the device become too old to be used reliably, spare blocks are reallocated to replace them. A Device contains some spare blocks for this purpose, as well as for some housekeeping operations. However, when all spares are consumed, the Device can no longer meet its fixed capacity definition - and it stops being functional (some become read-only, some stop responding completely). This can be frustrating for the end user.

A simple solution would be to allow the capacity to be dynamic. If the Device is allowed to reduce its reported capacity, it can reallocate useable blocks as new spares to compensate for aging. To support this, the Device needs to report to the Host how much more spare blocks it needs, and the Host should relinquish some used blocks to let the Device reallocate them as spares.

UFS Host should read the Device capacity upon initialization. The capacity will indicate the total useable capacity as well as the amount of spare blocks that the Device may currently need. A special status bit will be used by the Device on each command to indicate if it needs more spares. Host should monitor this bit, and if set, relinquish some used blocks.

To relinquish blocks, Host should reorder the logical data stored on the Device so that some area no longer contains valid data. Then, Host shall report to the Device of the freed area, and the Device will reduce the dynamic capacity and reallocate that area as new spare blocks. Note that the freed area shall comply with some size and alignment restrictions which are reported by the Device to make the area suitable for conversion to spare blocks.

In case the device is split to several LUs, each LU can report it needs some blocks separately. Several LUs may report needing blocks even though they share resources, so freeing blocks from one of them may satisfy the others too. Host should therefore check each LU separately before continuing to free blocks.

10.3.6 DYNAMIC DEVICE CAPACITY (cont'd)

The host may release space anywhere in the Logical Address range. If the space is released at the end of the address range than the device's capacity may be reduced after the next power up. Device capacity never changes while power is still on. If the address range is released in another location the device capacity will not change.

10.3.6.1 Implementation

10.3.6.1.1 Host and Device handshake

10.3.6.1.1.1 Initialization time

1. Host will query the device information structure.
2. Device replies to Host with the exported user capacity and the required capacity need to be released by the Host for device use.
3. The required capacity from the device is represented by a number that should be multiplied by CAPACITY_DECREASE_CHUNK_SIZE defined in device configuration data, also need to take into consideration the CAPACITY_DECREASE_ALIGHNMENT value that is defined also in device configuration data.
4. Host send to device RELEASE_LBA_RANGE command with the amount of data requested by the device.

RELEASE_LBA_RANGE command should give the Start LBA and count to the device.

10.3.6.1.1.2 On any UFS command

A UFS command will include release LBA range status in each UFS command response. The status is represented by 1 bit –

- 0 = not required
- 1 = required

In case the status from device says that release LBA range is required, the Host should issue the command sequence as done in the initialization time.

10.3.7 DATA RELIABILITY

10.3.7.1 Description

The UFS device will have the ability to define the level of data reliability during normal operation and power failure per LUN.

There are two components to the data reliability that will be defined for UFS. The first component deals with the data currently being written and the second will deal with the data that has already been written to the device.

The first component, reliable write, when set, will mean that if the device losses power during a write the data will be either the old data or the new written data when the device recovers from power failure. The resolution for the old and new data will be 512 B aligned. The figure below shows some of the possible scenarios that the host will see when it recovers from power failure during a 4 KB write with the reliable write enabled.

10.3.7.1 Description (cont'd)

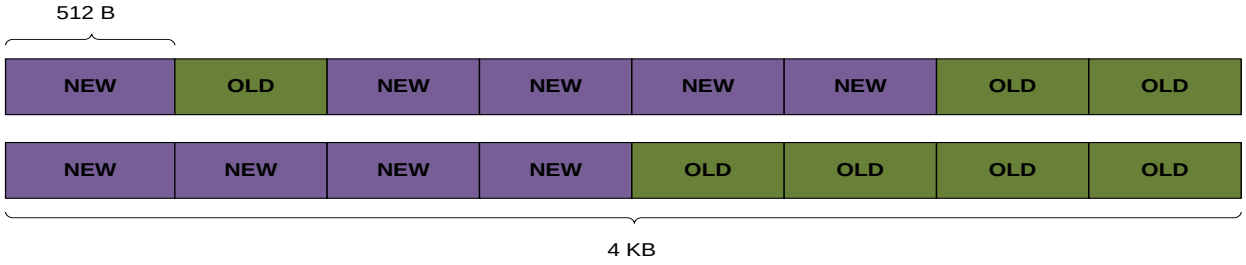

Figure 10-7 — Example of data status after a power failure during reliable write

The reliable write component will have two types of settings. The first will allow the user to set the parameter bit so that it is set until the host resets it. The second will allow the device to automatically reset the reliable write bit after the reliable write has completed in the device.

The second component, device data reliability, will allow the host to define the level of protection that must be applied to existing data on the device. In some technologies that will be used to implement UFS devices, the device has the option to potentially sacrifice some of the existing data on a device during a power failure in order to get better write performance from the device. Depending on the application for the end device the host can select per LUN whether the data in that LUN should be protected during power failure, which may have a performance impact, or to select device performance with the risk of losing data during a power failure.

This bit will be set once per LUN and will be set when the device is partitioned into LUNs. This setting only applies to general purpose partitions. Any LUNs defined as RPMB or boot will automatically select data reliability.

10.3.7.2 Implementation

The parameters listed in Table 10-3 will be defined to control device data reliability.

Table 10-3 — Parameters for controlling device reliability

RELIABLE_WRITE_MANUAL_RST	When set all write commands will be reliable write until this bit is reset by the host
RELIABLE_WRITE_AUTO_RST	When set the next write executed will be a reliable write, after the execution of the writ the bit will be reset.
DATA_RELIABLITLITY_LUN_x	The value of x will be from 0 to 7. If the bit is set the device will protect all device data during power failure. Settings for LUNs that have not been configured on the device will have no impact.

10.3.8 Detailed Implementation Summary

Implementation for sections Command Queue and Power Modes are defines as part of other proposals.

10.3.8.1 SCSI/UFS Status Bits (Device Information field of Response UPIU)

The flowing flags are reported after every SCSI/UFS command.

Table 10-4 — SCSI/UFS Status Bits

Offset	Name	Width (bits)	Description
TBD	BACKGROUND_OPS_FLAG	1	0 = not required 1 = required
TBD	DYNCAP_NEEDED_FLAG	1	0 = not required 1 = required

10.3.8.2 DEVICE Status Registers

The following status flags are read only and are reported by the device.

Bytes/bits not specified are reserved and return 0.

Table 10-5 — DEVICE Status Registers

Address	Name	Width (bytes)	Description
TBD	DYNCAP_BLOCK_SIZE (Implementation TBD)	4	The manageable block size for dynamic capacity manipulations (in MB)

10.3.8.3 LU Status Registers

The following status flags are read only and are reported by the LU. Each LU has its own set.

Bytes/bits not specified are reserved and return 0.

Table 10-6 — LU Status Registers

Address	Name	Width (bytes)	Description
TBD	DYNCAP_NEEDED (Implementation TBD)	4	Number of dynamic capacity blocks the device needs host to release

10.3.8.4 Device Mode Registers

The following flags can be read and written, and they control the device behavior.

Bytes/bits not specified are reserved and shall be set to 0.

Table 10-7 — DEVICE Mode Registers

Address	Name	Width (bytes)	Description

10.3.8.5 LU Mode Registers

The following flags can be read and written, and they control the LU behavior. Each LU has its own set. Bytes/bits not specified are reserved and shall be set to 0.

Table 10-8 — LU Mode Registers

Address	Name	Width (bytes)	Description
TBD	RELIABLE_WRITE_ACTIVE (Implementation TBD)	1	Bit [1:0]: 0 = Following writes are not reliable 1 = Only the next write is reliable 2 = All next writes are reliable
TBD	DATA_RELIABILITY (Implementation TBD)	1	Bit [0]: 0 = LU reliability not needed 1 = LU reliability needed
TBD	QUEUE_PRIORITY	1	Bit [0]: 0 = Normal Priority Queue 1 = High Priority Queue Reset value = 0 (normal)

11 UFS DESCRIPTORS

A descriptor is a data structure with a defined format. Descriptors are accessed via **Query Request UPUI** packets. Descriptors are independently addressable data structures. Descriptors may be stand alone and unique per device or they may be interrelated and linked in a hierarchical fashion to other descriptors by parameters defined within the top-level descriptor. Descriptors can range in size from 2 bytes through 256 bytes. A 2 byte descriptor is an empty descriptor. All descriptors have a length value as their first element. This length represents the entire length of the descriptor, including the length byte. All descriptors have a type identification as their second byte. A descriptor can be partially read and indexed in order to retrieve portions beyond the starting point. All descriptors are read using the CONFIGURATION READ sub-function specified in the DATA DESCRIPTOR HEADER.

A Descriptor is a block or page of parameters that describe something about a Device. For example, there are Device Descriptors, Configuration Descriptors, Unit Descriptors, etc.
Descriptors are read-only with the exception of Configuration Descriptor which is write-once – to be used to configure the UFS device during system integration

Table 11-1 specifies the descriptor identification values.

Table 11-1 — Descriptor identification values

DESCRIPTOR IDN	DESCRIPTOR TYPE
00h	DEVICE
01h	CONFIGURATION
02h	UNIT
03h	BOOT
04h	INTERCONNECT
05h	STRING
06h	LANGUAGE ID
07h	GEOMETRY
08h	POWER
09h..FFh	RFU

11.1 Descriptor Types

Descriptors are classified into types, as indicated in the table above. Some descriptors are singular entities, such as a Device descriptor. Others can have multiple copies, depending upon the quantity defined, such as UNIT descriptors. See diagram below.

11.2 Descriptor Indexing

Each descriptor type has an index number associated with it. The index value starts at zero and increments by one for each additional descriptor that is instantiated. For example, the first and only Device Descriptor has an index value of 0. The first String Descriptor will have an index value of 0. The Nth String Descriptor will have an index value of N-1.

11.3 Accessing Descriptors

Descriptors are accessed by requesting a descriptor IDN and a descriptor INDEX. For example, to request the fifth Unit descriptor, the request would reference TYPE UNIT (numeric value = 2) and INDEX 4 (numeric value = N-1).

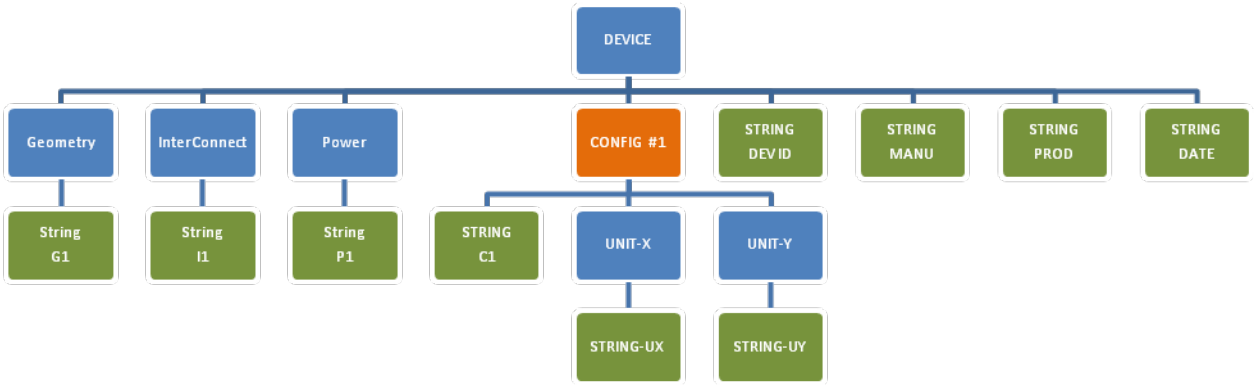

Figure 11-1 — Descriptor Organization

All descriptors are read-only, except Configuration Descriptor which is write-once. Configuration Descriptor is used to configure the UFS device during system integration. Parameter settings in the Configuration Descriptor are used to calculate and populate the parameter fields in the Unit Descriptors – an internal operation by the device.

11.4 Descriptor Page Definitions

11.4.1 Generic Descriptor Format

The format of all descriptors begins with a header which contains the length of the descriptor and a type value that identifies the specific type of descriptor. The length value includes the length of the header plus any additional data.

Table 11-2 — Generic Descriptor Format

DEVICE DESCRIPTOR			
OFFSET	**SIZE**	**NAME**	**DESCRIPTION**
0	1	bLength	Size of this descriptor inclusive = N
1	1	bDescriptorType	Descriptor Type Identifier
2..N-1	N-2	DATA	Descriptor Information

11.4.1 Generic Descriptor Format (cont'd)

For unit descriptors, the format is changed to add an indication for the unit being address.

Table 11-3 — Logical Unit Descriptor Format

UNIT DESCRIPTOR			
OFFSET	**SIZE**	**NAME**	**DESCRIPTION**
0	1	bLength	Size of this descriptor inclusive = N
1	1	bDescriptorType	Descriptor Type Identifier
2	1	Unit_INDEX	Unit being address - 00h to 07h
3..N-1	N-3	DATA	Descriptor Information

11.4.2 Device Descriptor

This is the main descriptor and should be the first descriptor retrieved as it specifies the device class and sub-class and the protocol (command set) to use to access this device and the maximum number of logical units contained within the device.

Table 11-4 — Device Descriptor

DEVICE DESCRIPTOR				
OFFSET	**SIZE**	**NAME**	**VALUE**	**DESCRIPTION**
0x00	1	bLength	0x13	Size of this descriptor inclusive = N
0x01	1	bDescriptorType	00h	Descriptor Type Identifier
0x02	1	bDevice	00h	Device type 00h – Device
0x03	1	bDeviceClass	00h	UFS Device Class 00h - Mass Storage RFU
0x04	1	bDeviceSubClass		UFS Subclass. Mass Storage Subclass 00h – Embedded Bootable 01h – Embedded Non-Bootable 02h – Removable , Non-Bootable RFU RFU Subclass
0x05	1	bProtocol		Protocol supported by UFS Device * 00h - SCSI * RFU
0x06	1	bNumberLU		For Version1.0, default is 1 & the maximum # of LU 8 LUs. To set this, OEMs must set the equivalent byte in the Configuration Descriptor
0x07	1	bBootEnable		Indicate whether the device is enabled for boot. To set this, OEMs must set the equivalent byte in the Configuration Descriptor
0x08	1	bShortBootEnable		Indicate whether the device is enabled for shortcut boot To set this, OEMs must set the equivalent byte in the Configuration Descriptor
0x09	1	bBootSeqID		Reserved for future use
0x0A	1	bSecurityLU		Support for security LU 00h – not supported 01h – RPMB RFU
0x0B	2	bcdSpecVersion	0100h	BCD version of supported UFS specification by the device, i.e., Version 3.21=0321H
0x0D	2	bcdManufactureDate		Manufacturing Date
0x0F	1	iManufacturerID	INDEX	Manufacturer ID. Index to string 01h which contain manufacturer string
0x10	1	iDeviceID	INDEX	Device ID. Index to string 02h which contain DEVICE ID string
0x11	1	iSerialNumberID	INDEX	Serial Number String ID. Index to string 03h which contain SN ID string
0x12	1	iOemID	INDEX	OEM ID. Index to string 04h which contain OEM ID string

11.4.3 UFS Interconnect Descriptor

The UFS Interconnect Descriptor describes critical maximal and minimal parameters associated with MIPI M-PHY & MIPI Unipro.

Table 11-5 — Interconnect Descriptor

INTERCONNECT DESCRIPTOR				
OFFSET	**SIZE**	**NAME**	**VALUE**	**DESCRIPTION**
0x0	1	bLength	0x06	Size of this descriptor inclusive = N
0x1	1	bDescriptorType	04h	Descriptor Type Identifier
0x2	2	bcdUniproVersion		MIPI Unipro Release Version Number
0x4	2	bcdMphyVersion		MIPI M-PHY Release Version Number

11.4.4 UFS Geometry Descriptor

The UFS Geomertry Descriptor describes the device geometric parameters.

Table 11-6 — Geometry Descriptor

GEOMETRY DESCRIPTOR				
OFFSET	**SIZE**	**NAME**	**VALUE**	**DESCRIPTION**
0x00	1	bLength	0x37	Byte of device geometry information
0x01	1	bDescriptorType	07h	Descriptor Type Identifier
0x02	1	bMediaTechnology		Reserve
0x03	1	bDataType		RFU for next revision. Support for data typing Default is ‘0’ – not supported
0x04	8	bTotalSectorCount		Total number of addressable 512B sectors in a raw device for total capacity calculation (default = normal memory type)
0x0C	1	bNativeBlockSize		Media Native Block size, in number of 512B sectors. This is optional parameter, 0=not available
0x0D	4	bSegmentSize		Equivalent to erase block size depending on memory technology and implementation, In number of 512B sectors
0x11	1	bAllocationUnitSize		In number of Segments Each LU can be allocated as a multiple of Allocation Unit
0x12	1	bMinAddrBlockSize		Minimum addressable block size; in number of 512B sectors; default = 1
0x13	1	bOptimalReadBlockSize		Optimal read block size; in number of 512B sectors This is optional parameter, 0=not available
0x14	1	bOptimalWriteBlockSize		Optimal write block size; in number of 512B sectors This is optional parameter, 0=not available

Table 11-6 — Geometry Descriptor (cont'd)

0x15	1	bMaxInBufferSize		Max. data-in buffer size in number of 512B sectors
0x16	1	bMaxOutBufferSize		Max. data-out buffer size in number of 512B sectors
0x17	2	bMaxDataSizeCollapseMode		Max. data size in bytes in a Phase Collapse operation supported by the device; no larger than 512B; 0= phase collapse operation not supported
0x19	1	bDataOrdering		Support for out-of-order data transfer 00h (default): out-of-order data transfer is not supported by the device, in-order data transfer is required. 01h: out-of-order data transfer is supported by the device
0x1A	16	Reserved		RFU
0x2A	1	bNumEnhancedMemory		The # of enhanced memory types of memories in the UFS device, in addition normal memory. - Default is '0' – i.e., normal memory only
0x2B	1	bEnhancedType01	01h	Support for Enhanced Memory Type 01. Memory type includes - 00h – Normal Memory - 01h – Enhanced Memory Type 01 - 02h – Enhanced Memory Type 02 - Etc.
0x2C	4	MAX # OF ALLOCATION UNITS		Max. available quantity of Enhanced Memory Type 1 for the entire device In number of Allocation Unit
0x30	1	CAPACITY ADJUSTMENT FACTOR		Factor used to adjust the memory capacity of this memory property from normal memory: CAPACITY = NORMAL_MEMORY_CAPACITY / FACTOR
0x31	1	bEnhancedType02	02h	Support for Enhanced Memory Type 02. Memory type includes - 00h – Normal Memory - 01h – Enhanced Memory Type 01 - 02h – Enhanced Memory Type 02 - Etc.
0x32	4	MAX # OF ALLOCATION UNITS		Max. available quantity of Frequent Access Memory type for the entire device In number of Allocation Unit
0x36	1	CAPACITY ADJUSTMENT FACTOR		Factor used to adjust the memory capacity of this memory property from normal memory: CAPACITY = NORMAL_MEMORY_CAPACITY / FACTOR

11.4.5 Configuration Parameters

This descriptor contains Device configuration parameters.

Table 11-7 — Configuration Descriptor

CONFIGURATION DESCRIPTOR				
OFFSET	**SIZE**	**NAME**	**VALUE**	**DESCRIPTION**
0x00	1	bLength		Size of this descriptor inclusive = N
0x01	1	bDescriptorType	01h	Configuration Descriptor Type
0x02	1	bNumberLU		Configure the number of LU the device will support.
0x03	1	SECURE_REMOVAL_TYPE		• 0b00000000 will result in all information being removed by an erase of the physical memory. (default) • 0b00000010 will result in all information being removed by overwriting the addressed locations with a character, its complement, then a random character. • 0b00000001 will result in all information being removed by overwriting the addressed locations with a single character followed by an erase. • 0b00000011 will result in the information being removed using a vendor defined mechanism.
0x04	14	Reserved		Reserved
0x12	1	BOOT CONFIG LENGTH		Length of boot configuration setting; default is 8 bytes
0x13	1	bBootEnable		No = 00h, Yes = 01h Note: This status will also be reflected in the Device Descriptor
0x14	1	bShortBootEnable		No = 00h, Yes = 01h Note: This status will also be reflected in the Device Descriptor
0x15	1	bBootSeqID		Reserved for future use
0x16	5	RESERVED		Reserved
	1	LU CONFIG LENGTH	10h	Length of one LUN configuration setting; default is 16 bytes
	1	bLUNum	##h	Definition for LU Number ##h. Default LU if no other LU's are defined
	1	LU ENABLED		No = 00h, Yes = 01h
	1	bBootLunID		0b00000000 = not bootable 0b00000001 = Boot LU A 0b00000010 = Boot LU B 0b00000100 = Boot LU C Etc.

Table 11-7 — Configuration Descriptor (cont'd)

	1	LU WRITE PROTECT		00h: not write protected; 01h: LU write protected when Power-On Write Protect Flag is set; 02h: LU permanently write protected when Permanent Write Protect Flag is set.
	1	Reserved		
	1	LU_QUEUE_PRIORITY		Inter-LU queue priority setting 00h = Normal Priority Queue 01h = High Priority Queue Default value is 00h
	1	MEMORY TYPE		Input memory type reference. Default is Normal Memory = 00h
	4	# OF ALLOCATION BLOCKS		Number of blocks assigned to this memory property type in this LU; in units of Allocation Block of normal memory
	1	DATA TYPE SUPPORT		Reserved; default = 00h
	1	bLogicalBlockSize		Size of addressable logical blocks; number of 512B sectors; default = 1
	1	PROVISIONING TYPE		Enable/disable LU Thin Provisioning, and set SECURE_MODE and TPRZ bit in READ CAPACITY parameter data 0b00000000 = Thin Provisioning is disabled (default) 0b00000010 = Thin Provisioning is enabled and TPRZ = 0 0b00000011 = Thin Provisioning is enabled and TPRZ = 1 All others reserved
	2	RESERVED		Reserved
	1	LUN 1	01h	
	1	LU ENABLED		No = 00h, Yes = 01h
	1	bBootLunID		0b00000000 = not bootable 0b00000001 = Boot LU A 0b00000010 = Boot LU B 0b00000100 = Boot LU C Etc.
	1	LU WRITE PROTECT		00h: not write protected; 01h: LU write protected when Power-On Write Protect Flag is set; 02h: LU permanently write protected when Permanent Write Protect Flag is set.
	1	Reserved		
	1	LU_QUEUE_PRIORITY		Inter-LU queue priority setting 00h = Normal Priority Queue 01h = High Priority Queue Default value is 00h

Table 11-7 — Configuration Descriptor (cont'd)

	1	MEMORY TYPE		Input memory type reference. Default is Normal Memory = 00h
	4	# OF ALLOCATION BLOCKS		Number of blocks assigned to this memory property type in this LU; in units of Allocation Block of normal memory
	1	DATA TYPE SUPPORT		Reserved; default = 00h
	1	bLogicalBlockSize		Size of addressable logical blocks; number of 512B sectors; default = 1
	1	PROVISIONING TYPE		Enable/disable LU Thin Provisioning, and set SECURE_MODE and TPRZ bit in READ CAPACITY parameter data 0b00000000 = Thin Provisioning is disabled (default) 0b00000010 = Thin Provisioning is enabled and TPRZ = 0 0b00000011 = Thin Provisioning is enabled and TPRZ = 1 All others reserved
	2	RESERVED		
		-		
		-		
		-		
		-		
	1	LUN 7	07h	
	1	LU ENABLED		No = 00h, Yes = 01h
	1	bBootLunID		0b00000000 = not bootable 0b00000001 = Boot LU A 0b00000010 = Boot LU B 0b00000100 = Boot LU C Etc.
	1	LU WRITE PROTECT		00h: not write protected; 01h: LU write protected when Power-On Write Protect Flag is set; 02h: LU permanently write protected when Permanent Write Protect Flag is set.
	1	Reserved		
	1	LU_QUEUE_PRIORITY		Inter-LU queue priority setting 00h = Normal Priority Queue 01h = High Priority Queue Default value is 00h
	1	MEMORY TYPE		Input memory type reference. Default is Normal Memory = 00h
	4	# OF ALLOCATION BLOCKS		Number of blocks assigned to this memory property type in this LU; in units of Allocation Block of normal memory

Table 11-7 — Configuration Descriptor (cont'd)

	1	DATA TYPE SUPPORT		Reserved; default = 00h
	1	bLogicalBlockSize		Size of addressable logical blocks; number of 512B sectors; default = 1
	1	PROVISIONING TYPE		Enable/disable LU Thin Provisioning, and set SECURE_MODE and TPRZ bit in READ CAPACITY parameter data 0b00000000 = Thin Provisioning is disabled (default) 0b00000010 = Thin Provisioning is enabled and TPRZ = 0 0b00000011 = Thin Provisioning is enabled and TPRZ = 1 All others reserved
	2	RESERVED		

11.4.6 Power Parameters Descriptor

This descriptor contains information about the power capabilities and power states of the device.

Table 11-8 — Power Descriptor

POWER DESCRIPTOR – 08h				
OFFSET	**SIZE**	**NAME**	**VALUE**	**DESCRIPTION**
	1	bLength		Byte of device geometry information
	1	bDescriptorType	00h	Descriptor Type Identifier
	32	bActiveConsumptionVCC(15:0)		2byte max. VCC current value for each of the 16 power modes starting with Power Mode 00
	32	bActiveConsumptionVCCQ(15:0)		2byte max. VCCQ current value for each of the 16 power modes starting with Power Mode 00
	32	bActiveConsumptionVCCQ2(15:0)		2byte max. VCCQ2 current value for each of the 16 power modes starting with Power Mode 00

11.4.7 Unit Descriptor

This page describes specific characteristics and capabilities of an individual logical unit, for example geometry of the device and maximum addressable item. There are up to eight unit descriptors.

Table 11-9 — Unit Descriptor

UNIT DESCRIPTOR				
OFFSET	**SIZE**	**NAME**	**VALUE**	**DESCRIPTION**
0	1	bLength		Size of this descriptor inclusive = N
1	1	bDescriptorType	02h	Configuration Descriptor Type
2	1	Unit_INDEX	00h to 07h	Unit Index
3	1	LU ENABLED		No = 00h, Yes = 01h
	1	bBootLunID		0b00000000 = not bootable 0b00000001 = Boot LU A 0b00000010 = Boot LU B 0b00000100 = Boot LU C Etc.
5	1	LU WRITE PROTECT		00h: not write protected; 01h: LU write protected when Power-On Write Protect Flag is set; 02h: LU permanently write protected when Permanent Write Protect Flag is set.
6	1	QUEUE DEPTH		Queue depth available in this LU. Queue depth of '0' means best effort by device to service the command task
7	1	LU_QUEUE_PRIORITY		Inter-LU queue priority setting 00h = Normal Priority Queue 01h = High Priority Queue Default value is 00h
8	1	MEMORY TYPE		Input memory type reference. Default is Normal Memory = 00h
9		DATA TYPE SUPPORT		Reserved; default = 00h
10	1	bLogicalBlockSize		Size of addressable logical blocks; number of 512B sectors; default = 1
11:18	8	LOGICAL BLOCK COUNT		Total number of addressable Logical Blocks in the LU
19:22	4	ERASE BLOCK SIZE		In number of Logical Blocks
23	1	PROVISIONING TYPE		0b00000000 = Thin Provisioning is disabled (default) 0b00000010 = Thin Provisioning is enabled and TPRZ = 0 0b00000011 = Thin Provisioning is enabled and TPRZ = 1 All others reserved
24:31		RESERVED		Reserved

11.4.8 RPMB Unit Descriptor

Table 11-10 — RPMB Unit Descriptor

RPMB UNIT DESCRIPTOR				
OFFSET	**SIZE**	**NAME**	**VALUE**	**DESCRIPTION**
0	1	bLength		Size of this descriptor inclusive = N
1	1	bDescriptorType	02h	Configuration Descriptor Type
2	1	INDEX	C4h	Descriptor Index
3	1	LU ENABLED		No = 00h, Yes = 01h
	1	bBootLunID	0b00000000	0b00000000 = not bootable 0b00000001 = Boot LU A 0b00000010 = Boot LU B 0b00000100 = Boot LU C Etc.
5	1	LU WRITE PROTECT	00h	00h: not write protected; 01h: LU write protected when Power-On Write Protect Flag is set; 02h: LU permanently write protected when Permanent Write Protect Flag is set.
6	1	QUEUE DEPTH		Queue depth available in this LU. Queue depth of ‘0‘ means best effort by device to service the command task
7	1	LU_QUEUE_PRIORITY		Inter-LU queue priority setting 00h = Normal Priority Queue 01h = High Priority Queue Default value is 00h
8	1	MEMORY TYPE		Input memory type reference. Default is Normal Memory = 00h
9	1	DATA TYPE SUPPORT		Reserved; default = 00h
10	1	bLogicalBlockSize		Size of addressable logical blocks; number of 512B sectors; For RPMB, default = 1
11:18	8	LOGICAL BLOCK COUNT		Total number of addressable Logical Blocks of the LU For RPMB, Logical Block Count should be multiples of 256 (i.e. 128Kbytes)
11:22	4	ERASE BLOCK SIZE	00h	In number of Logical Blocks For RPMB, Erase Block Size is ignored; set to ‘0’
23	1	PROVISIONING TYPE		0b00000000 = Thin Provisioning is disabled (default)
24:31		RESERVED		Reserved

11.4.9 MANUFACTURER ID String

This descriptor contains the UNICODE manufacturer name string that may consist of up to 127 UNICODE characters. Number of UNICODE characters is calculated by $(LENGTH - 2) \div 2$

Table 11-11 — Manufacturer ID String

MANUFACTURER_ID STRING			
OFFSET	**SIZE**	**NAME**	**DESCRIPTION**
0	1	bLength	Size of this descriptor inclusive
1	1	bDescriptorType	00h
2	2	UC[0]	Unicode string character
4	-	-	-
LENGTH-2	2	UC[(LENGTH-2)÷2-1]	Unicode string character

11.4.10 DEVICE_ID String

This descriptor contains the UNICODE device name string that may consist of up to 127 UNICODE characters. Number of UNICODE characters is calculated by $(LENGTH - 2) \div 2$

Table 11-12 — Device_ID String

DEVICE_ID STRING DESCRIPTOR			
OFFSET	**SIZE**	**NAME**	**DESCRIPTION**
0	1	bLength	Size of this descriptor inclusive
1	1	bDescriptorType	00h
2	2	UC[0]	Unicode string character
4	-	-	-
LENGTH-2	2	UC[(LENGTH-2)÷2-1]	Unicode string character

11.4.11 OEM_ID String

This descriptor contains the UNICODE OEM_ID string that may consist of up to 127 UNICODE characters. Number of UNICODE characters is calculated by $(LENGTH - 2) \div 2$.

Table 11-13 — OEM_ID String

OEM_ID STRING DESCRIPTOR			
OFFSET	**SIZE**	**NAME**	**DESCRIPTION**
0	1	bLength	Size of this descriptor inclusive
1	1	bDescriptorType	00h
2	2	UC[0]	Unicode string character
4	-	-	-
LENGTH-2	2	UC[(LENGTH-2)÷2-1]	Unicode string character

11.4.12 SERIAL_NUMBER String

This descriptor contains the UNICODE serial number string that may consist of up to 127 UNICODE characters. Number of UNICODE characters is calculated by $(LENGTH - 2) \div 2$.

Table 11-14 — Serial Number String Descriptor

SERIAL NUMBER STRING DESCRIPTOR			
OFFSET	**SIZE**	**NAME**	**DESCRIPTION**
0	1	bLength	Size of this descriptor inclusive
1	1	bDescriptorType	00h
2	2	UC[0]	Unicode string character
4	-	-	-
LENGTH-2	2	UC[(LENGTH-2)÷2-1]	Unicode string character

11.4.13 Flags

A Flag is a single Boolean value that represents a TRUE or FALSE, ‘0’ or ‘1’, ON or OFF type of value. A Flag can be cleared or reset, set, toggled or read. Flags are useful to enable or disable certain functions or modes or states within the device.

Access property type (read/write, write-only, read-only, power-on-reset, permanent, etc.) is defined for each flag in the table below.

Table 11-15 — Flags

FLAGS			
Index	**Name**	**Type**	**Description**
01	bDeviceInit	Read/Write	Host set flag to initiate device-side initialization after boot process is completed. Device resets flag when device initialization is completed.
02	WP_PERM	Write once	To set permanent write protection on all identified parameters; cannot be toggled or cleared once it is set
03	WP_POR	Power-on reset	To set power-on write protection on all identified parameters; cannot be toggled or cleared until the next power-on reset
04	BACKGROUND_OPS_EN	Read/Write	0= Device is not permitted to run background operations when the queue is empty. 1=Device is permitted to run background operations when the queue is empty Default is ‘1’
05	Reserved		Reserved
06	PURGE_EN	Write Only	0 = Purge operation is disabled (default) 1 = Purge operation is enabled This bit can only be set when the LU queue is empty. This bit is automatically cleared by the UFS device when the operation completes or an error condition occurs
07	PURGE_ERR	Read Only	0: no error exists (default) 1: purge operation was not run because the queue was not empty. When the PURGE_EN bit is set, this bit will be reset to zero unless a new error occurs.
08	OUT_OF_ORDER DATA_EN	Write once	0 = Out-of-order data transfer is disabled (default) 1 = Out-of-order data transfer is enabled. This bit shall have effect only when bDataOrdering = 01h

11.4.14 Attributes

An Attribute is a 32-bit (4 bytes) parameter that represents a specific range of numeric values that can be set or read. For example, block size would be an attribute.

Access property type (read/write, write-only, read-only, power-on-reset, permanent, etc.) is defined for each attribute in the table below.

Table 11-16 — Attributes

ATTRIBUTES				
Index	**Name**	**Type**	**Value**	**Description**
00	bBootLunEn	Power-on reset		Set to the bBootLunID of Boot LU to use to boot. When bBootLunEn field is 0b00000000 the boot feature is disabled, the device behaves as if bBootEnable would be equal to 0
02	bDeviceState	Read only		Current device state/mode
03	bActiveConsumpMode	Read/Write		The device is allowed to use the larger power budgets associated with the Active power mode set in this attribute. Default=0.
04	bPORDeviceState	Power-on reset		At next boot, if set the device initializes to Active Mode, otherwise to UFS-Sleep Mode.
05	BACKGROUND_OPS_STATUS	Read only		Device health status for background operation 00h = not required 01h = required, not critical 02h = required, performance impact 03h = critical.
06	PURGE_STATUS	Read only		Status of the Purge operation 0b00000000 = no status (default) 0b00000001 = Purge in progress 0b00000010 = Purge was stopped prematurely. 0b00000011 = Purge completed successfully.
07	bMaxDataInSize	Read/Write		Maximum data-in data packet size allowable to be sent by device to host. Not to exceed the bMaxInBufferSize parameter. If bMaxDataInSize > bMaxInBufferSize, data packet size in Data-In UPIU sent by device to host shall default to bMaxInBufferSize This parameter can be written by the host only when all LU task queues are empty.
08	bMaxDataOutSize	Read/Write		Maximum data-out data packet size the host is capable to send to device in a Data-Out UPIU. Not to exceed the bMaxOutBufferSize parameter. If bMaxDataOutSize > bMaxOutBufferSize, device shall default to bMaxOutBufferSize in the Ready-to-Transfer UPIU sent to the host to request data. This parameter can be written by the host only when all LU task queues are empty.

11.5 GET DESCRIPTORS

11.5.1 Details

A Query Request operation with READ DESCRIPTOR opcode is sent by the host to the device. The host builds a **QUERY REQUEST UPIU** places a **READ DESCRIPTOR** opcode within the UPIU, sets the appropriate values in the required fields and sends that UPIU to the target device.

Upon reception of the **QUERY REQUEST UPIU** the device will decode the **READ DESCRIPTOR** opcode field and retrieve the descriptor indicated by the **DESCRIPTOR IDN QUERY REQUEST UPIU** field. When the device is ready to return data to the Host the device will construct a **QUERY RESPONSE UPIU,** set the appropriate fields and place the entire retrieved descriptor within a single Data Segment area of the **QUERY RESPONSE UPIU.**

Upon transmission of the **QUERY RESPONSE UPIU** the device will consider the Query Request operation complete.

Upon reception of the **QUERY RESPONSE UPIU** the host will retrieve the requested descriptor from the Data Segment area, and decode the Status and other fields within the UPIU and take the appropriate completion action. Upon reception of the **QUERY RESPONSE UPIU** the host will consider that the Query Request operation is complete.

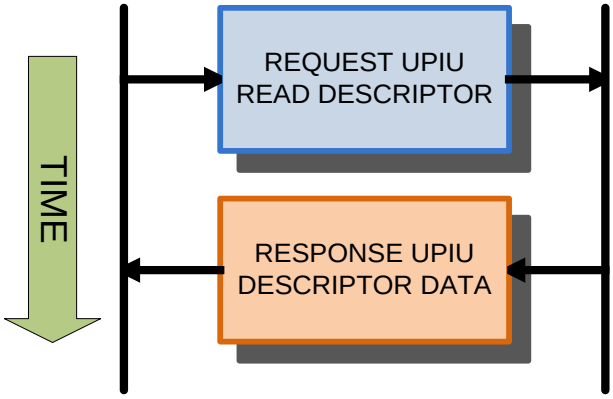

Figure 11-2 — Read Request Descriptor

11.6 SET DESCRIPTORS

11.6.1 Details

A Query Request operation with WRITE DESCRIPTOR opcode is sent by the host to the device. The host builds a **QUERY REQUEST UPIU** places a **WRITE DESCRIPTOR** opcode within the UPIU, sets the appropriate values in the required fields and sends that UPIU to the target device. A **QUERY REQUEST UPIU** with a **WRITE DESCRIPTOR** opcode will additionally include a data segment that contains the descriptor data the Host is sending to the Device.

Upon reception of the **QUERY REQUEST UPIU** the device will decode the **WRITE DESCRIPTOR** opcode field and access the descriptor indicated by the **DESCRIPTOR IDN QUERY REQUEST UPIU** field. The device will extract the descriptor data within the data segment of the UPIU and will internally overwrite the addressed descriptor with the extracted data. When the device has completed this process it will then notify the Host of the success or failure of this operation by returning to the Host a **QUERY RESPONSE UPIU** with the **RESPONSE** field containing the response code.

11.6.1 Details (cont'd)

When the device has decided that the data transfer has completed the device will build a **QUERY RESPONSE UPIU**, place a Status code within **RESPONSE** field the UPIU, set the appropriate values in the required fields within the UPIU and send the UPIU to the host. After the transmission of the **QUERY RESPONSE UPIU** the target device will consider the operation complete.

Upon reception of the **QUERY RESPONSE UPIU** the host will decode the **RESPONSE** field and other fields within the UPIU and take the appropriate completion action. Upon reception of the **QUERY RESPONSE UPIU** the host will consider that the operation is complete.

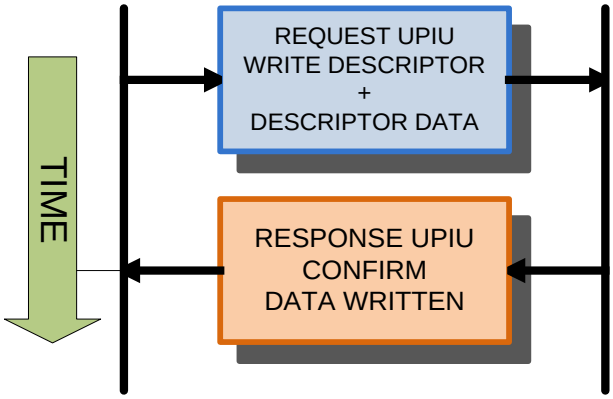

Figure 11-3 — Write Request Descriptor

Annex A (informative) - Host Controller Interface (HCI) Overview

The UFS Host Controller Interface specification describes the register-level Host Controller interface for UFS Controller. This also includes description of the HW-SW interface to the host controller as well as the data flow model.

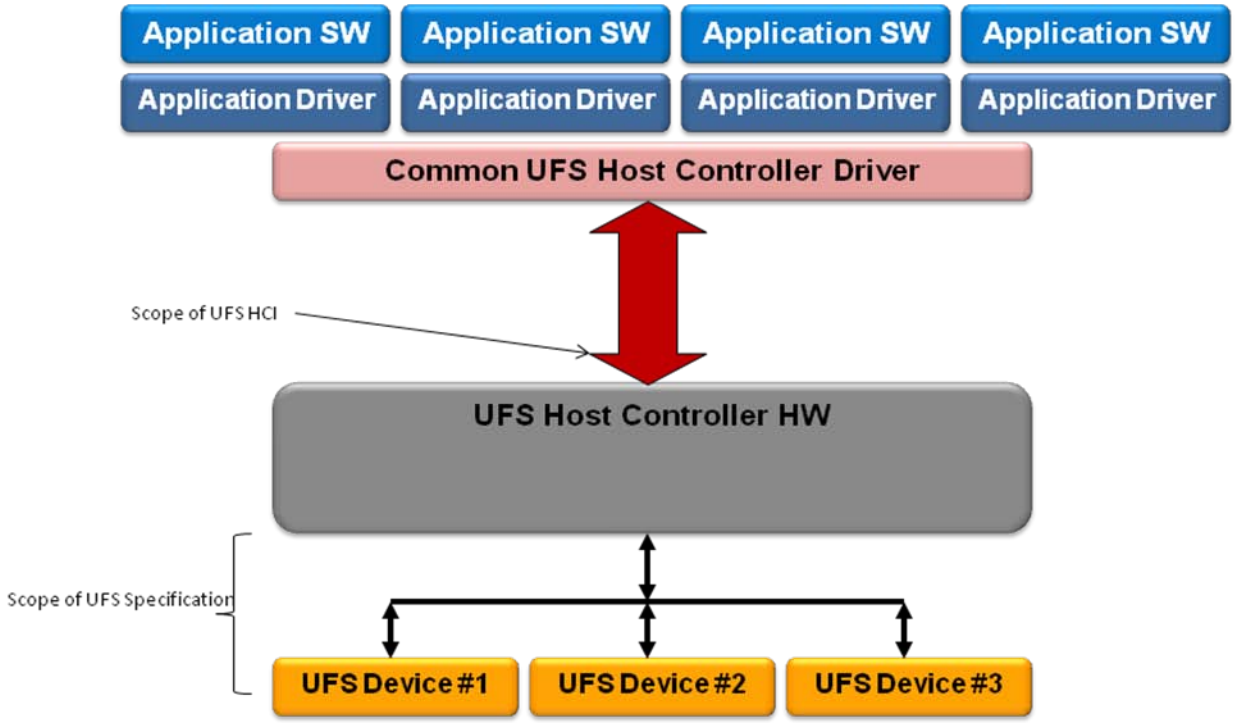

Figure 11-4 — UFS HW-SW Overview

The intention is to provide a standard/coming programming model at the Host/OS Driver level for UFS. It will also

Provide standard programming model for UFS

Common Register set for OS driver

Low-level driver can be custom per host

Provide mechanism to managed the Host Controller, the data transfer engine

Provide bus/link management and power management capabilities, including device power management

De-couple the host controller interface from the UFS protocol

Does not define implementation of the Host Controller, but provide OEMs & vendors the mechanism to add differentiators and value in their implementation

A.1 UFS HCI Architecture

The figure below provides an architectural overview of the UFS Host Controller Interface and description of the different components.

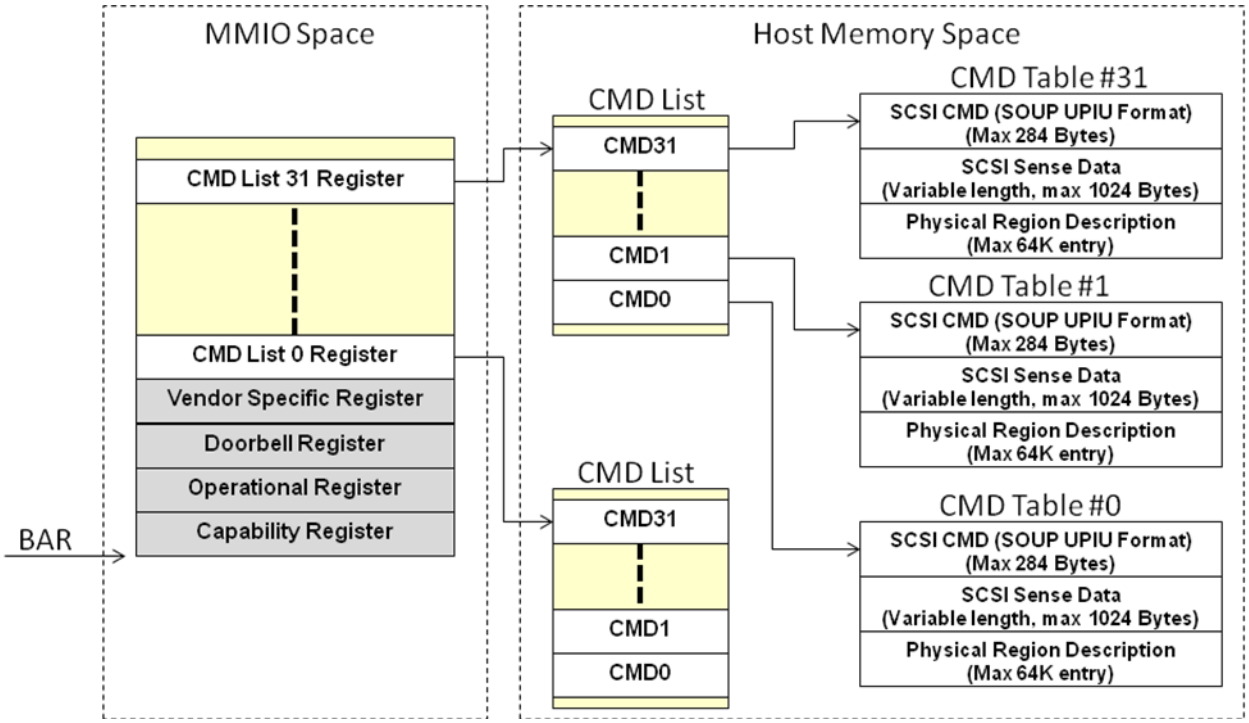

Figure 11-5 — UFS Host Controller Architectural Overview

The UFS Host Controller Interface consists on two memory map regions

MMIO region. This register set include the basic control and configuration of UFS bus and devices. The configuration & control information include bus management, power management, doorbells, interupts, device enumeration & discovery, etc.

Host memory region. System memory or DRAM where the command queue/list, scatter/gather list, and data buffer resides. These lists and buffers are allocated and managed by UFS Host Controller driver and are used to perform the data transfer to/from the UFS devices.

Here are some definitions for the different blocks in **Error! Reference source not found.**. The assumption is that a single Host Controller shall support up to N different UFS devices.

The Base Adress Register (BAR) provides a single reference to the data structures used by the HCI. Using the BAR the Capability Registers, Operational Registers, Command List registers, vendor specific register, and the Doorbell Array can be accessed. The later registers can reside in contiguous memory or in separate memory addresses.

The Capability Registers are read-only registers, defining the capabilities and restrictions of the host controller implementation. These values are used as parameters to the host controller driver.

The Operational Registers specify host controller configuration and runtime modifiable state, and are used by system software to control and monitor the operational state of the host controller.

A.1 UFS HCI Architecture (cont'd)

Command List 0-31 Registers.

Doorbell Array Register. This is an array of N+1 Doorbell Registers, which supports up to N UP devices. Each Doorbell Register provides the system software with a mechanism for notifying the Host Controller if it has Device related work to perform. A DB Target field in the Doorbell Register is written with a value that identifies the reason for "ringing" the doorbell. Doorbell Register #0 is allocated to the Host Controller for Command Ring management.

Vendor Specific Registers.

And the following describes the components residing in Host memory region

Command List. Max of 32 command list is supported by the HCI. 32 Commands can be queued in the command list. It will also support command reordering and out-of-order data delivery.

Command Table. Include the command used to describe a transaction, command/management task. The command is in SOUP UPIU format. The table also includes fields for SCSI CDB, SCSI Sense Data, and the scatter/gather list for the transaction.

Scatter/Gather List. Part of the command table, the list include pointer to all the data or region of data to be transfer.

Data Buffer. Data to be transfer, located in host memory.

Annex B (normative) – References

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

[MIPI-M-Phy] MIPI Alliance Specification for M-PHYSM Specification, Version 1.00.00

[MIPI-UniPro] MIPI Alliance Specification for Unified Protocol (UniProSM), Version 1.40.00

[MIPI-DDB] MIPI Alliance Specification for Device Descriptor Block (DDB), Version

[SAM] SCSI Architecture Model – 5 (SAM-5), Revision 05, 19 May 2010

[SPC] T10 Specification: SCSI Primary Commands – 4 (SPC-4), Revision 27, 11 October 2010

[SBC] T10 Specification: SCSI Block Commands – 3 (SBC-3), Revision 24, 05 August 2010

Annex C (normative) – Terms, Definitions, Letter Symbols, and Keywords

For the purposes of this standard, the following terms, definitions, letter symbols, and keywords apply:

BOOT_ID	Boot Identification Number
CDB	Command Descriptor Block
CPORT	A CPort is a Service Access Point on the UniPro Transport Layer (L4) within a Device that is used for Connection-oriented data transmission
CRB	Command Request Block
DEVID	Device ID
DID	Device ID
DMA	Direct Memory Access
EOF	End of Packet
GB	Gigabyte
HCI	Host Controller Interface
IU	Information Unit
KB	Kilobyte
LUN	Logical Unit Number
MB	Megabyte
MIPI	Mobile Industry Processor Interface
NA	Not applicable
NU	Not used
PDU	Protocol Data Unit
PLL	Phase-Locked Loop
PTR	Pointer
PWM	Pulse Width Modulation
RBC	
RFU	Reserved for future use
RPMB	Replay Protected Memory Block
SDU	Service Data Unit
SID	Segment ID
SOF	Start of Packet
SPC	SCSI Primary Commands
SRB	SCSI Request Block
T_PDU	MIPI Unipro Protocol Data Unit
T_SDU	MIPI Unipro protocol Service Data Unit
UFS	Universal Flash Storage
UPIU	UFS Protocol Information Unit
UTP	UFS Transport Protocol
XCDB	
XiP	eXecute in Place

Annex C (normative) – Terms, Definitions, Letter Symbols, and Keywords (cont'd)

C.1 Definitions

Address Nexus: An address nexus is a combination of parameters that identify a connection between one bus device to another. For the UFS bus and address nexus consists of a bus device ID for the source, a bus device ID for the destination and a logical unit number for the destination.

Byte: An 8-bit data value with most significant bit labeled as bit 7 and least significant bit as bit 0.

Command Descriptor Block: The structure used to communicate commands from an application client to a device server. A CDB may have a fixed length of up to 16 bytes or a variable length of between 12 and 260 bytes.

Command Request Block: The bytes the make up a SCSI command that are encapsulated within a SCSI Request Block.

Device: An addressable device on the UFS bus usually a target that contains at least one LUN.

Device ID: The bus address of a UFS device.

Doubleword: A 32-bit data value with most significant bit labeled as bit 31 and least significant bit as bit 0.

Dword: A 32-bit data value, a Doubleword.

Gigabyte: 1,073,741,824 or 230 bytes.

Host: An addressable device on the UFS bus which is usually the main CPU that hosts the UFS bus.

Initiator: An initiator is an addressable device on the UFS bus which is the originator of a command request or message to a target device. As currently defined within UFS, the host or host controller is always acts as the initiator. UFS devices cannot be initiators.

Kilobyte: 1024 or 210bytes.

Logical Unit: A logical unit is an internal entity of a bus device that performs a certain function or addresses a particular space or configuration within a bus device. UFS devices can have up to eight logical units.

Logical Unit Number: A numeric value that identifies a logical unit within a device.

Megabyte: 1,048,576 or 220 bytes.

Quadword: A 64-bit data value with most significant bit labeled as bit 63 and least significant bit as 0.

Segment: A specified number of sequentially addressed bytes representing a data structure or section of a data structure.

Segment ID: A 16-bit value that represents an index into a table or an address of a segment descriptor or simply an absolute value that is an element of an absolute address.

SCSI Request Block: A data packet that contains a multi-byte SCSI command and additional contextual information needed to carry out the command operation. A SCSI Request Block is built by the host and is targeted at a particular bus device.

Target: A target is an addressable device on the UFS bus which is the destination or target of a command or message sent by an initiating device.

Task: A task is a SCSI command which includes all transactions to complete all data transfers and a status response that will satisfy the requirements of the requested services of the command.

Annex C (normative) – Terms, Definitions, Letter Symbols, and Keywords (cont'd)

C.1 Definitions (cont'd)

Transaction: A UFS bus primitive action which results in transmission of serial data packets between a target and initiator.

UFS Protocol Information Unit: Information transfer (communication) between a UFS host and device is done through messages which are called UFS Protocol Information Units. These messages are UFS defined data structures that contain a number of sequentially addressed bytes arranged as various information fields.

Unit: A bus device.

Unit Attention: A condition of a bus device utilizing the SCSI protocol where it needs to be serviced before it can continue processing requests and responses.

Word: A 16-bit data value with most significant bit labeled as bit 15 and least significant bit as bit 0.

C.2 Keywords

Several keywords are used to differentiate levels of requirements and options, as follow:

Expected - A keyword used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

Ignored - A keyword that describes bits, bytes, quadlets, or fields whose values are not checked by the recipient.

Mandatory - A keyword that indicates items required to be implemented as defined by this standard.

May - A keyword that indicates flexibility of choice with no implied preference.

Optional - A keyword that describes features which are not required to be implemented by this standard. However, if any optional feature defined by the standard is implemented, it shall be implemented as defined by the standard.

Reserved - A keyword used to describe objects—bits, bytes, and fields—or the code values assigned to these objects in cases where either the object or the code value is set aside for future standardization. Usage and interpretation may be specified by future extensions to this or other standards. A reserved object shall be zeroed or, upon development of a future standard, set to a value specified by such a standard. The recipient of a reserved object shall not check its value. The recipient of a defined object shall check its value and reject reserved code values.

Shall - A keyword that indicates a mandatory requirement. Designers are required to implement all such mandatory requirements to assure interoperability with other products conforming to this standard.

Should - A keyword indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase "it is strongly recommended".

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