JEDEC STANDARD

Universal Flash Storage Host Controller Interface (UFSHCI)

Version 5.0

JESD223G

(Revision of JESD223F, December 2025)

February 2026

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.

All risk and liability relating to the use of JEDEC standards is assumed by the user, who agrees to indemnify and hold JEDEC harmless.

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.

PRICE: Contact JEDEC
3103 10th Street North, Suite 240S, Arlington, VA 22201

For information, contact:

JEDEC Solid State Technology Association
3103 10th Street North
Suite 240S
Arlington, VA 22201
https://www.jedec.org/contact

This page intentionally left blank.

UNIVERSAL FLASH STORAGE HOST CONTROLLER INTERFACE (UFSHCI), VERSION 5.0

Contents

Contents (cont'd)

Page

UNIVERSAL FLASH STORAGE HOST CONTROLLER INTERFACE (UFSHCI), VERSION 5.0

Contents (cont'd)

Page

UNIVERSAL FLASH STORAGE HOST CONTROLLER INTERFACE (UFSHCI), VERSION 5.0

Contents (cont'd)

Page

Figures

Tables

UNIVERSAL FLASH STORAGE HOST CONTROLLER INTERFACE (UFSHCI), VERSION 5.0

(From JEDEC Board ballot JCB-24-47, formulated under the cognizance of the JC-64.1 Subcommittee on Electrical Specifications and Command Protocols.)

1 Scope

This standard describes a functional specification of the Host Controller Interface (HCI) for Universal Flash Storage (UFS). The objective of UFSHCI is to provide a uniform interface method of accessing the UFS hardware capabilities so that a standard/common Driver can be provided for the Host Controller. The common Driver would work with UFS host controller from any vendor. This standard includes a description of the hardware/software interface between system software and the host controller hardware. It is intended for hardware designers, system builders and software developers. This standard is a companion document to [UFS], Universal Flash Storage (UFS). The reader is assumed to be familiar with [UFS], [MIPI-UniPro], and [MIPI-M-PHY].

Clause 4 provides a brief overview of the architectural overview of UFS. Clause 5 describes the register interface of UFSHCI. Clause 6 describes the data structure used by UFSHCI. Clause 7 provides a theory of operation for UFSHCI. Clause 8 describes the error recovery process for UFSHCI.

2 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. For undated references, the latest edition of the normative document referred to applies.

[MIPI-M-PHY], *MIPI Alliance Specification for M-PHY℠, Version 6.0,* November 2025.

[MIPI-UniPro], *MIPI Alliance Specification for Unified Protocol (UniPro℠), Version 3.0,* September 2025

[UFS], JEDEC JESD220H, *Universal Flash Storage (UFS 5.0)*

[JEP106], JEDEC JEP106, *Standard Manufacturer's Identification Code, JEP106BK,* September 2024.

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

[IEEE 1619-2007], *IEEE Standard for Cryptographic Protection of Data Block-oriented Storage Devices,* 1619-2007.

3 Acronyms, Terms and Definitions, Keywords, and Conventions

3.1 Acronyms

CQ Completion Queue
DID Device ID
GB Gigabyte
HCI Host Controller Interface
KB Kilobyte (Binary (computing/memory notation))
KiB kibibyte (Standardized (IEC notation))
LUN Logical Unit Number
MCQ Multi-Circular Queue
MIPI Mobile Industry Processor Interface
MB Megabyte
NA Not applicable
PRDT Physical Region Description Table
SQ Submission Queue
UCD UTP Command Descriptor
UFS Universal Flash Storage
UPIU UFS Protocol Information Unit
UTP UFS Transport Protocol
UTRD UTP Transfer Request Descriptor
UTMRD UTP Task Management Request Descriptor

3.2 Terms and Definitions

Byte: An 8-bit data value with most significant bit labeled as bit 7 and least significant bit as bit 0.

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 (GB): 1,073,741,824 or 2^{30} bytes.

Host: An addressable device on the UFS bus which is usually the main CPU that hosts the UFS bus.

Kilobyte (KB or KiB): 1024 or 2^{10} bytes.

Logical Unit Number: A numeric value that identifies a logical unit within a device.

3.2 Terms and Definitions (cont'd)

Megabyte (MB): 1,048,576 or 2^{20} bytes.

Quadword: A 64-bit data value with most significant bit labeled as bit 63 and least significant bit as 0.

UFS Protocol Information Unit: Information transfer (communication) between a UFS host and device is done through messages which are called UFS Protocol Information Units.

NOTE The messages are UFS defined data structures that contain a number of sequentially addressed bytes arranged as various information fields.

UTP Transfer Request Descriptor: A data structure in system memory that contains a UTP command and additional contextual information needed to carry out the command operation.

NOTE The command is limited to UFS adopted INCITS T10 draft standard command sets (see [UFS]), UFS native command set and Device Management function that uses UTP protocol. A UTP Transfer Request Descriptor is built by the host and is targeted at the attached UFS device.

UTP Task Management Request Descriptor: A data structure in system memory that contains a UTP Task Management Function and the additional contextual information needed to execute the function.

NOTE A UTP Transfer Request Descriptor is built by the host and is executed by the attached UFS device.

Unit: A bus device.

Word: A 16-bit data value with most significant bit labeled as bit 15 and least significant bit as bit 0.

Zero-Based value: A numeric value N (N > 0) represented by N-1.

3.3 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, quad lets, 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.

3.3 Keywords (cont'd)

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.

3.4 Conventions

The conventions used for registers in this standard are defined in the clauses that follow.

Hardware shall return '0' for all bits and registers that are marked as reserved, and host software shall write all reserved bits and registers with the value of '0'.

Inside the register clause, the following abbreviations are used:

HwInit The default state is dependent on device and system configuration. The value is initialized at reset, either by an expansion ROM, or in the case of integrated devices, by a platform BIOS.

Impl Spec Implementation Specific – the controller has the freedom to choose its implementation.

RO Read Only

ROC Read Only and Read to clear

RW Read Write

R/W Read Write. The value read may not be the last value written.

RWC Read/Write '1' to clear

RWS Read/Write '1' to set

WO Write Only

4 Architectural Overview

UFS is a simple, high performance, serial interface. It is primarily for use in mobile systems, between host processing and NVM mass storage devices.

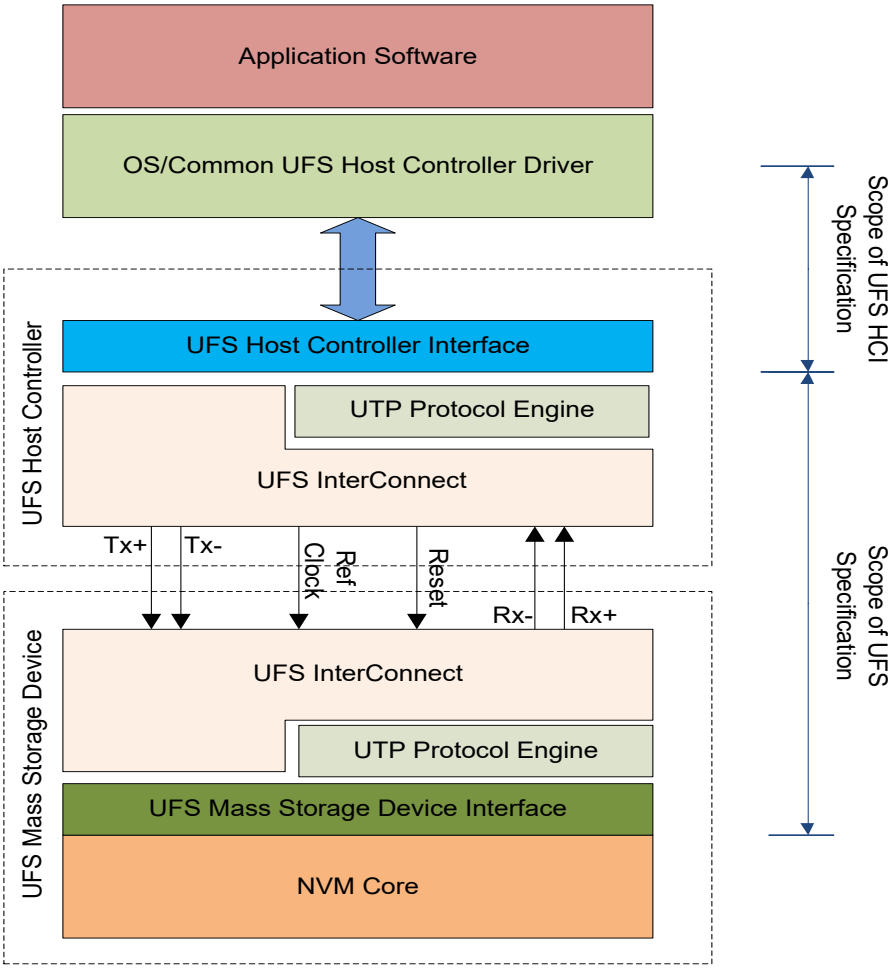


Figure 1 — UFS Architecture Overview

This standard defines the register-level host controller interface for Universal Flash Storage (UFS). The UFS Host Controller is responsible for managing the interface between host SW and UFS device and the data transfer. This includes interface management, power management, and control. Also included in this standard is the data transfer & programming model.

4.1 Outside of Scope

This standard does not contain information relevant to implementation of the UFS Interconnect as this is wholly described in the MIPI UniPro Specification [MIPI-UniPro]. This standard can be applied to any system bus interface. However, this standard does not define a system bus that may be used in an UFS Host Controller implementation.

4.2 Interface Architecture

UFS host software uses a combination of a host register set and Transfer Request Descriptors in system memory to communicate with host controller hardware. Figure 2 illustrates a conceptual block diagram of UFS Host Controller Interface.

In Version 4.0, the HCI spec added Multi-Circular Queue (MCQ) definition to improve UFS storage performance. To support UFS3.1 device, the legacy single Doorbell definition via UTP Transfer Request List will continue to exist and operate independent of Multi-Circular Queue definition. However, only one should be active during operation, either Single Doorbell, or MCQ.

As Figure 2 highlighted, MCQ will have its own capabilities register, and its own config, status, attributes, and interrupt registers. Please refer to 4.4.1, Multi-Circular Queue (MCQ) Definition, and clause 5, UFS Host Controller Register Interface, for more details.

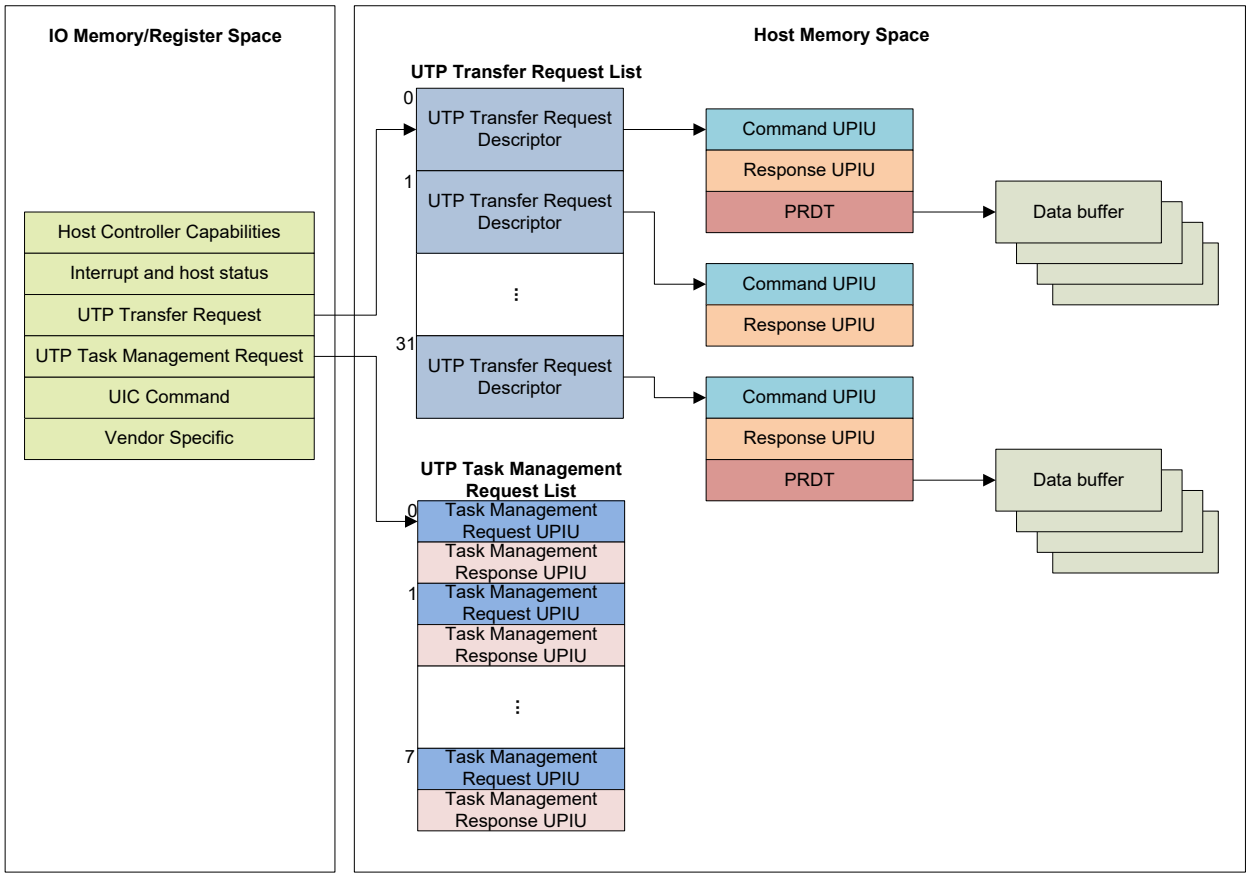


Figure 2 — General Architecture of UFS Host Controller Interface.

4.2 Interface Architecture (cont'd)

UFSHCI defines two interface spaces.

- **MMIO Space**. In this space, a set of hardware registers are defined as the host controller interface to system software. It is normally implemented as Memory-Mapped I/O (MMIO) space, consisting of three types of registers:
  - **Host Controller Capability Registers.** These registers provide description of host controller capabilities. They include UFS standard version, the size of the command queue the host controller supported, and host controller identification data.
  - **MCQueue Capability Register.** These registers provide description of host controller capabilities specific for the Multi-Circular Queue.
  - **Runtime and Operation Registers.** These include support for the following:
    - **Interrupt configuration**. These registers provide an interface for host SW to enable/disable interrupt and status of the interrupts.
    - **Host controller status**. This register shows the status of the host controller and allows host software to initialize/deactivate the host controller.
    - **UTP Transfer request list management**. These registers provide an interface to UTP Transfer Request List. These registers are used for **QT** = 0 and are not used when **QT** = 1.
    - **UTP Task Management request lists management**. These registers provide an interface to UTP Transfer Request List.
    - **UIC Command Registers**. These registers provide an interface for UniPro configuration and control.
    - **MCQueue configuration register.** This register define how Multi-Circular Queue can be use & the associate attributes.
    - **MCQueue attribute registers.** These registers define attributes per circular queues.
    - **MCQ registers.** These registers define base address, head & tail pointers, priorities, Queue size, and CQ mapping to SQ.
  - **Vendor Specific Registers.** These registers are defined by vendors.
- **Host Memory Space**. This space includes data structures that provide description of the commands for execution and the data buffers which are a part of each command. The data structures and data buffers are application protocol specific (UTP).

4.3 Transfer Request Interface

A UFS Host System is composed of a number of hardware and software layers. Figure 3 illustrates a conceptual block diagram of the building block layers in a host system.

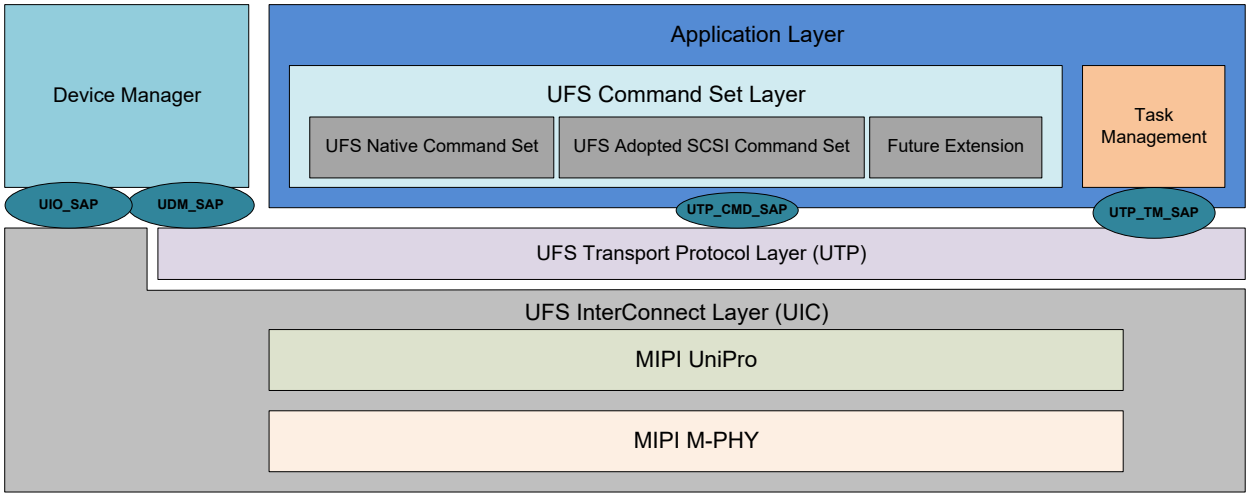


Figure 3 — A Conceptual Block Diagram of UFS Host System

This standard defines a set of registers and data structures that work in concert with UFS host SW to implement the four service access points (SAPs) as described in Figure 3:

UIO_SAP, UDM_SAP, UTP_CMD_SAP and UTP_TM_SAP.

Refer to the UFS Standard [UFS], for the definition of the service access points.

To manage the communication between host SW and the UFS devices attached, the host controller provides three independent interfaces that host software uses to send a transfer request.

- **UTP Transfer Request List**. This list is used by host software to implement UTP_CMD_SAP and UDM_SAP.
  - UTP_CMD_SAP includes support for the following command types:
    - All INCITS T10 draft standard functionality adopted by UFS.
    - Native UFS command set.
  - UDM_SAP includes support for the following command types:
    - Device management function via QUERY REQUEST UPIU/QUERY RESPONSE UPIU and NOP IN/NOP OUT UPIU.

The list consists of a data structure called UFS Transfer Request Descriptor (UTRD). UTRD describes a command to be executed and the data associated it. UFS host SW issues a command to the host controller by placing a UTRD on the List then rings the Host Controller doorbell for the list. Commands are dispatched for execution by the UFSHCI in the order that they are placed on the List even though they may be completed out of the order. The host controller acts on behalf of host processor to manage all the data transfer operations associated with the command. All commands could result in a Command Completion interrupt or status field of the UTRD for the command in the List being updated. UFS SW may add commands to the List while it is running. The host controller supports interrupt aggregation, such that a single command completion interrupt is generated for a pre-defined number of command completions.

4.3 Transfer Request Interface (cont'd)

- **UTP Task Management Request List**. This list is used by host software to implement UTP_TM_SAP. The List consists of a data structure called UFS Task Management Request Descriptor (UTMRD). UTMRD describes a task management function that host software wants the attached device to execute. All the Task Management Requests will be prioritized over the transfer requests listed in UTP Transfer Request List as described above. UFS host SW issues a task management function to the host controller by placing a UTMRD on the List then rings the Host Controller doorbell for the list. Functions are dispatched for execution by the UFSHCI in the order that they are placed on the List even though they may be completed out of the order. All task management functions could result in a Request Completion interrupt or status field of the UTMRD for the function in the List being updated. UFS SW may add task management function to the List while it is running. Interrupt aggregation is not supported for this list.

- **UIC Command Register**. This register set is used by host software to execute a UIC command directly.

4.4 Multi-Circular Queue for UFSHCI4.0

To improve performance, UFS4.0 HCI introduce the option to use Multi-Circular Queue. With the availability of this option, HCI can operate in legacy queue mode and be backward compatible with older UFS device. Or the HCI can enable MCQ for improve performance with UFS4.0 device or later.

The Host Controller Capabilities register, bit 29, indicate if legacy queue is supported. And MCQ Config Register, bit 0, will allow SW to select the appropriate Queue mode, legacy queue or MCQ.

MCQ register set includes MCQ capabilities register, MCQ configuration register, and MCQ definition.

Overview of the MCQ includes:

- Queues are core agnostic.
- Submission & completion queue definition include the base address & doorbell registers.
- Submission queue to include a Completion queue mapping. Mapping can be 1:1 or N:1.

4.4 Multi-Circular Queue for UFSHCI4.0 (cont'd)

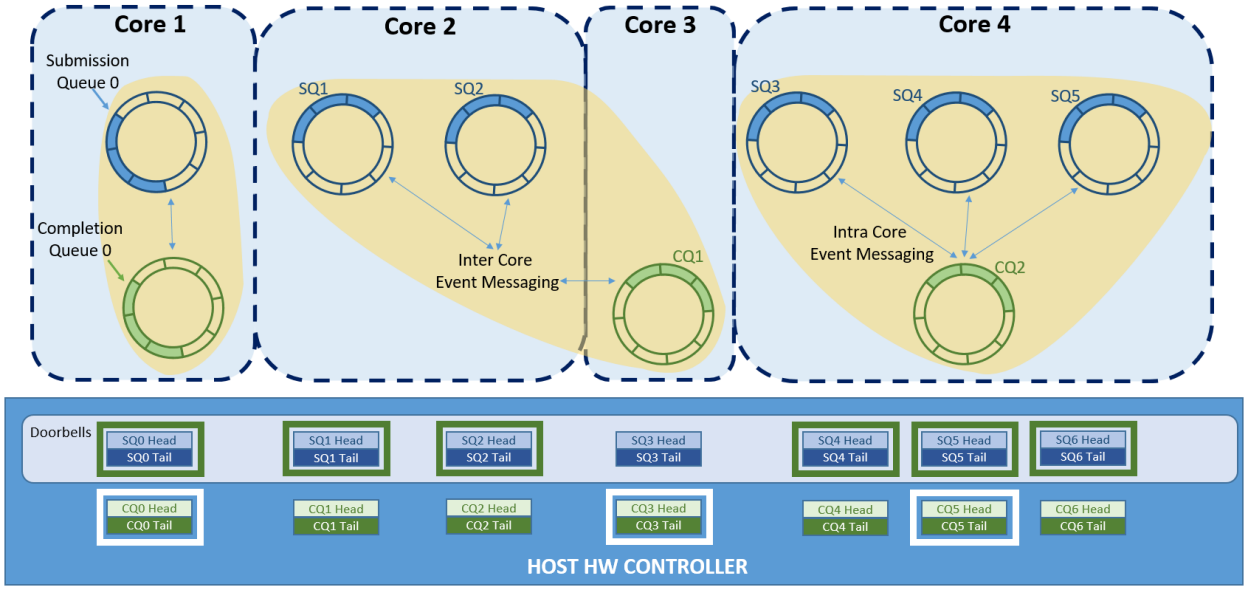


Figure 4 — Host Controller Multi-Circular Queue Example
(Head/Tail pointers of enabled queues are shown in bold box)

4.4.1 Multi-Circular Queue (MCQ) Definition

A Multi-Circular Queue (MCQ) is a communication mechanism for passing messages from a producer to a consumer. Circular queues are defined by:

- A queue entry size, appropriately sized for the largest message to be passed from the producer to the consumer.
- A message format such that the producer and consumer can encode the required message within the queue entry.
- A count of queue entries.
- A pointer to a host memory region sized to accommodate the count of queue entries.
- Doorbells indicate zero-based queue entry slots within the host memory region.
- Doorbells increment from lower to higher integers corresponding with queue entry slots within the host memory region.
    - A wrap condition occurs when a doorbell is incremented such that it is equal or greater to the count of queue entries.
    - A wrap condition resets the doorbell to zero, pointing that doorbell to the start of the host memory region.
- A head doorbell, written by the consumer, indicating one of the queue entry address offset to the start of host memory region.
    - Writes to the head doorbell indicate ~~to~~ the producer that one or more queue entries have been consumed, creating empty queue entry locations for reuse.

4.4.1 Multi-Circular Queue (MCQ) Definition (cont'd)

- A tail doorbell, written by the producer, indicating one of the queue entry address offset to the start of host memory region.
  - A producer writes to the queue entry referenced by the current tail doorbell and then increments the tail doorbell to the next empty queue entry address offset.
  - Writes to the tail doorbell indicate to the consumer that one or more queue entries have been added to the circular queue.
  - Queue entries may only be added when incrementing the tail doorbell would not make it equal to the head doorbell.
- When the head and tail doorbells are equal, the queue is empty.
  - Note that this definition means there will always be one empty queue entry.
- When the head and tail doorbells are not equal, the queue contains queue entries.

4.4.2 Submission Queue Definition

A Submission Queue (SQ) is a specialized Circular Queue for which the Host SW is the producer and the Host Controller is the consumer. The Host SW passes messages to the Host Controller indicating submission of new commands to be processed by the UFS device. Each SQ identifies the CQ which will receive its command completion notification once the HC has completed command processing, including servicing all UPIU traffic required to complete that command. Details on the SQ data structure can be found in 6.2.1.

4.4.3 Completion Queue Definition

A Completion Queue (CQ) is a specialized Circular Queue for which the Host Controller is the producer and the Host SW is the consumer. The Host Controller passes messages inside CQ Entries to the Host SW indicating completion of commands. Each CQ Entry identifies which submission queue the command originated in, and the unique identifier for that command, and command completion status. Details on the CQ data structure can be found in 6.2.2.

4.4.4 Multiple Queues

The HC shall publish the number of Completion Queues and Submission Queues supported MCQCAP.MAXQ. In order to support 1:1 topology, HC shall implement resources to support MAXQ number of Submission Queues and MAXQ number of Completion Queues. See 5.2.2 for details on the MCQCAP. In N:1 topology, number of active completion queues will be less than number of active submission queues.

Host software should provision queues by configuring the corresponding Submission Queue or Completion Queue configuration region of register space. When a Queue Attributes write indicates that the queue is active (CQATTR.CQEN or SQATTR.SQEN = 1b), the HC shall begin operation on that circular queue according to type and configuration described within the HCI registers.

4.4.5 Basic MCQ Flow

Before enabling a queue, it must be defined and configured.

- **Queues Configuration.**
  - Establish base address and doorbells
  - Establish SQ size, priority, CQ ID mapping
  - Establish CQ size
  - Enable queues
    - Enable mapped CQ
    - Enable SQ

Below is a basic MCQ processing flow during run time. This processing flow depends on Host Controller and Host SW implementation.

- **Command Submission**

1. Host SW writes an Entry to SQ
2. Host SW updates SQ doorbell tail pointer

- **Command Processing**

3. After fetching the Entry, Host Controller updates SQ doorbell head pointer
4. Host controller sends COMMAND UPIU to UFS device

- **Command Completion**

5. Host controller receives RESPONSE UPIU
6. Host controller updates CQ Entry to include RESPONSE UPIU, Status, and Submission Queue ID
7. Host controller updates CQ doorbell tail pointer. Interrupt & status generated
8. Host SW processes Completion Queue Entry
9. Host SW updates CQ doorbell head pointer

4.4.5 Basic MCQ Flow (cont'd)

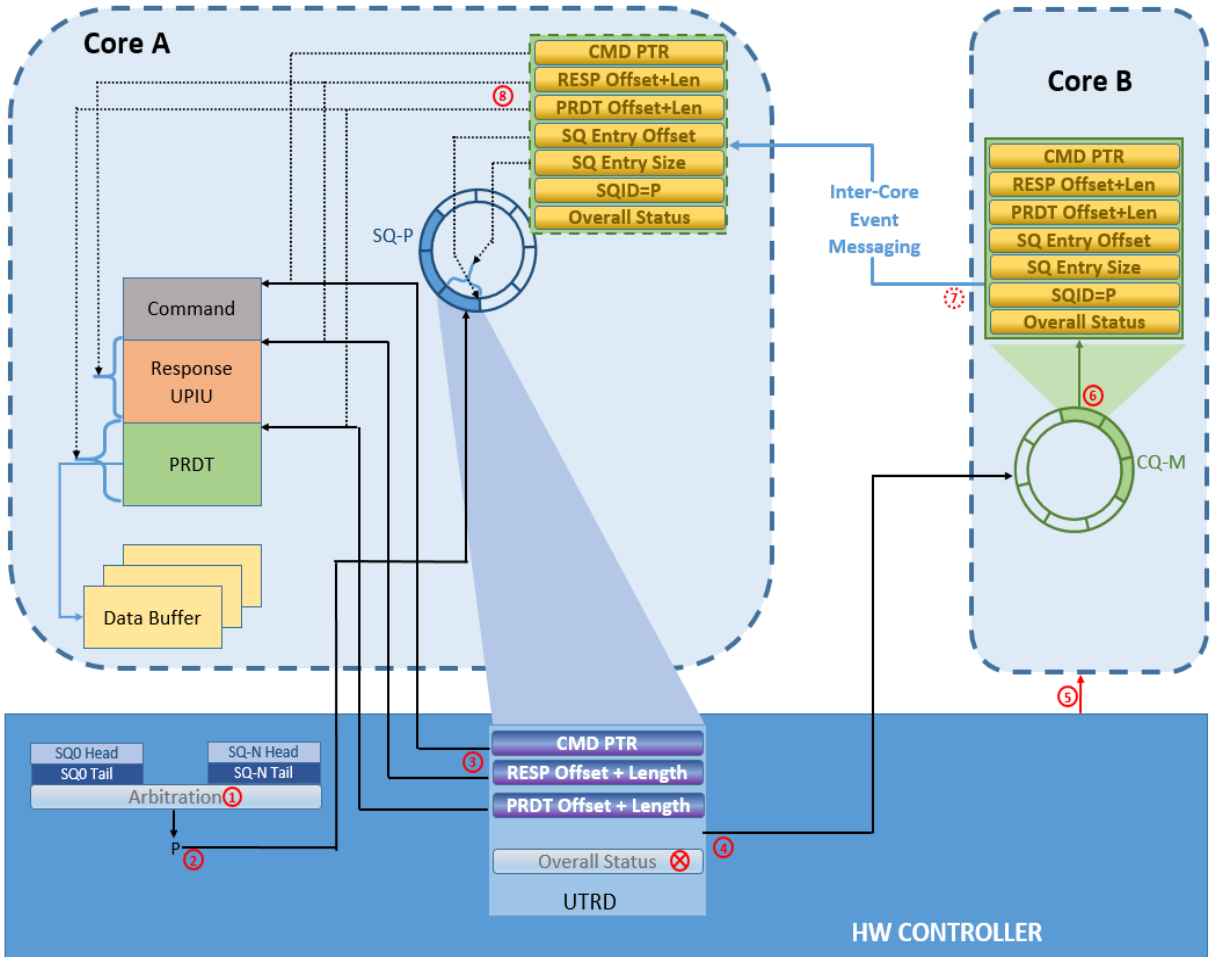


Figure 5 — Multi-Circular Queue Flow

Figure 6 on the following page describes how the multiple circular queue mode is working. Host software driver may refer this generic command processing flow to develop the software driver.

Host software checks the availability of SQ slot, puts a SQ entry in a SQ and updates the SQ tail index register of the SQ. HCI arbitrates SQs to choose a SQ entry from SQs depending on the Queue priority in SQ Attributes register, and move the chosen SQ entry from the SQ to the HCI Arbitrator's internal buffer. Then HCI updates SQ header index register since the SQ entry is moved to HCI Arbitrator's internal buffer. The command will be issued to the target device to execute, and UFS device will execute and return the response to the host.

After receiving the response from the device, HCI arbitrator checks the availability of the target CQ, and sends it to the target CQ with the CQ tail index update. And HCI arbitrator removes the corresponding command entry from the HCI internal buffer. Then HCI generates the interrupt to Host software to handle the CQ entry. Host software updates the CQ header index to remove any data structure for that CQ entry from the system memory.

4.4.5 Basic MCQ Flow (cont'd)

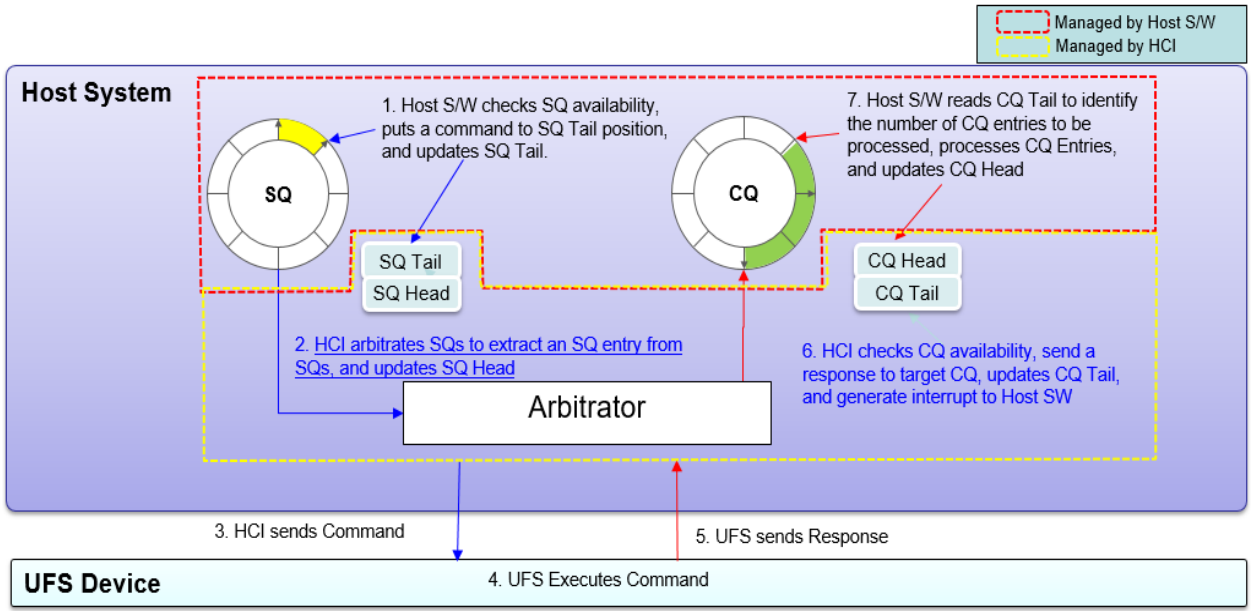


Figure 6 — Multi-Circular Queue Mode Operation

4.4.6 (Informative) Processing Abort in MCQ Mode: An Implementation Example

When host software issues an abort request, the command may be in one of following three cases.

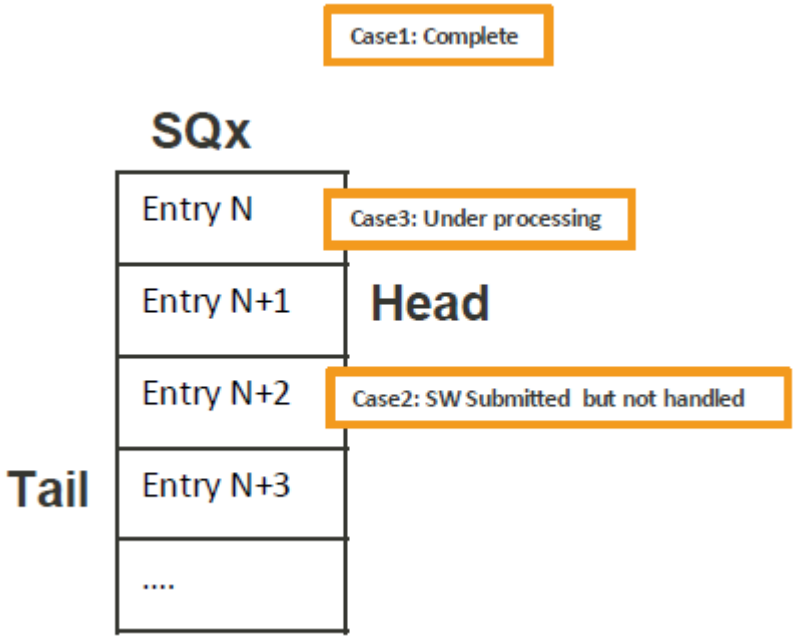


Figure 7 — Three Possible Command Cases when an Abort Request is Issued

As Figure 8 on the following page shows, depending on the status of the command, the host controller should clean up the resources related with the aborted command. The following is an exemplar implementation.

1) Host software sets SQRTCy.STOP as '1'. The host controller will stop the fetching command from the Submission Queue, and set the SQRTSy.STS bit as '1' to indicate that the SQ is stopped

2) Depending on the status of the command requested to be aborted,

   A. If the command is already completed, then go to step 3).

   B. If the command is in the Submission Queue and not issued to the device yet, the host controller will mark the command to be skipped in the Submission Queue. The host controller will post to the Completion Queue to update the OCS field with 'ABORTED'.

   C. If the command is issued to the device already but there is no response yet from the device, the host software driver issue the Abort task management function to the device for that command. Then the host driver set SQRTCy.ICU as '1' to initiate the clean up the hardware resources. The host controller will post to the Completion Queue to update the OCS field with 'ABORTED'. If host software driver receives the 'task not found' as the response of the associated task management function, then go to 3.

3) SW set SQRTCy. STOP as '0', host controller set the SQRTSy.STS bit as '0', then resume fetching the command from Submission Queue.

4.4.6 (Informative) Processing Abort in MCQ Mode: An Implementation Example (cont'd)

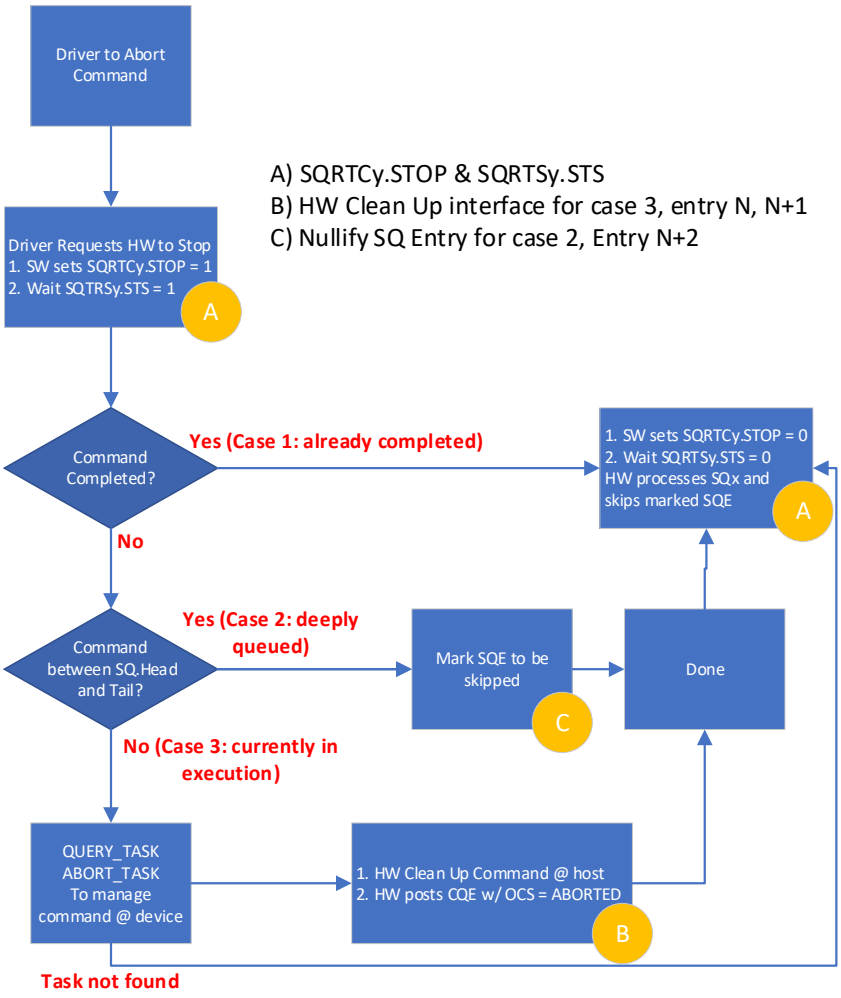


Figure 8 — Command Abort Flow in MCQ Mode

4.4.7 (Informative) Task Management in MCQ Mode

MCQ does not provide task management mechanism for each Submission Queue. Hence, even in MCQ mode, task management is to be performed using the existing 8-bits UTP Task Management Request List DoorBell Register **(UTMRLDBR)**.

Though other implementations of task management are possible in MCQ mode, this narrative presents a simple approach.

In MCQ, task management is handled entirely by SW. HW controller or device works same as legacy.

4.4.7 (Informative) Task Management in MCQ Mode (cont'd)

The legacy task management process becomes the central, common process in MCQ mode. All task managements are done by this central process. SQ specific processes, that requires management service for any task that they initiated, uses inter-process communication to pass on all relevant task information to the admin process, which in turn executes the task management operation and returns final status to the calling SQ process.

1. All SQ processes use piped or socket IPC to request task management process for any task management they need.
2. When SQ process calls a function of task management process via pipe or socket, task management process launches the task management request to HW controller.
3. Task management process maintains a map of {IID_Ext, IID} → socket
4. When task management response arrives, it uses {IID_Ext, IID} to dereference the pipe/socket, and returns the response to that pipe/socket, resulting task management response rerouting to correct SQ process

5 UFS Host Controller Register Interface

The host controller registers are memory mapped and exist in MMIO space. Registers access shall have a maximum size of 64-bits; 64-bit access shall not cross an 8-byte alignment boundary.

UFSHCI registers are used to control the operation of the Host Controller and also to read the status and interrupt information from the Host controller**.**

5.1 Register Map

The following is a standard register map for UFSHCI.

	Start	End	Symbol	Description
Host Capabilities	00h	03h	CAP	Host Controller **Cap**abilities
	04h	07h	MCQCAP	Multi-Circular Queue Capability Register
	08h	0Bh	VER	UFS **Ver**sion
	0Ch	0Fh	EXT_CAP	EXT_CAP – Extended Controller Capabilities
	10h	13h	HCPID	**H**ost **C**ontroller **I**dentification **D**escriptor – Product ID
	14h	17h	HCMID	**H**ost **C**ontroller **I**dentification **D**escriptor – Manufacturer ID
	18h	1Bh	AHIT	Auto-Hibernate Idle Timer
	1Ch	1Fh	Reserved	Reserved
Operation and Runtime	20h	23h	IS	**I**nterrupt **S**tatus
	24h	27h	IE	**I**nterrupt **E**nable
	28h	2Bh	Reserved	Reserved
	2Ch	2Fh	HCSEXT	Host Controller Status Extended
	30h	33h	HCS	**H**ost **C**ontroller **S**tatus
	34h	37h	HCE	**H**ost **C**ontroller **E**nable
	38h	3Bh	UECPA	Host **U**IC **E**rror **C**ode **P**HY **A**dapter Layer
	3Ch	3Fh	UECDL	Host **U**IC **E**rror **C**ode **D**ata Link Layer
	40h	43h	UECN	Host **U**IC **E**rror **C**ode **N**etwork Layer
	44h	47h	UECT	Host **U**IC **E**rror **C**ode **T**ransport Layer
	48h	4Bh	UECDME	Host **U**IC **E**rror **C**ode **DME**
	4Ch	4Fh	UTRIACR	**U**TP **T**ransfer **R**equest **I**nterrupt **A**ggregation **C**ontrol **R**egister
UTP Transfer	50h	53h	UTRLBA	**U**TP **T**ransfer **R**equest **L**ist **B**ase **A**ddress
	54h	57h	UTRLBAU	**U**TP **T**ransfer **R**equest **L**ist **B**ase **A**ddress **U**pper 32-Bits
	58h	5Bh	UTRLDBR	**U**TP **T**ransfer **R**equest **L**ist **D**oor**B**ell **R**egister
	5Ch	5Fh	UTRLCLR	**U**TP **T**ransfer **R**equest **L**ist **CL**ear **R**egister

5.1 Register Map (cont'd)

	Start	End	Symbol	Description
	60h	63h	UTRLRSR	**U**TP **T**ransfer **R**equest **R**un-**S**top **R**egister
	64h	67h	UTRLCNR	**U**TP **T**ransfer **R**equest **L**ist **C**ompletion **N**otification **R**egister
	68h	6Fh	Reserved	Reserved
UTP Task Managemeng	70h	73h	UTMRLBA	**U**TP **T**ask **M**anagement **R**equest **L**ist **B**ase **A**ddress
	74h	77h	UTMRLBAU	**U**TP **T**ask **M**anagement **R**equest **L**ist **B**ase **A**ddress **U**pper 32-Bits
	78h	7Bh	UTMRLDBR	**U**TP **T**ask **M**anagement **R**equest **L**ist **D**oor**B**ell **R**egister
	7Ch	7Fh	UTMRLCLR	**U**TP **T**ask **M**anagement **R**equest **L**ist **CL**ear **R**egister
	80h	83h	UTMRLRSR	**U**TP **T**ask **M**anagement **R**un-**S**top **R**egister
	84h	8Fh	Reserved	Reserved
UIC Command	90h	93h	UICCMD	**U**IC **C**om**m**an**d** **R**egister
	94h	97h	UCMDARG1	**U**IC **C**om**m**an**d** **Arg**ument 1
	98h	9Bh	UCMDARG2	**U**IC **C**om**m**an**d** **Arg**ument 2
	9Ch	9Fh	UCMDARG3	**U**IC **C**om**m**an**d** **Arg**ument 3
	A0h	AFh	Reserved	Reserved
UMA	B0h	BFh	Reserved	Reserved for Unified Memory Extension
Vendor Specific	C0h	FFh		Vendor Specific Registers
Crypto	100h	103h	CCAP	**C**rypto **Cap**ability
	104h	2FFh	x-CRYPTOCAP	Crypto Capability X
Config	300h	303h	Config	Global Configuration
	304h	307h	Reserved	Reserved
	308h	30Fh	Reserved	Reserved
MCQ Configuration	380h		MCQConfig	MCQ Config Register
	380h+04h		ESILBA	Event Specific Interrupt Lower Base Address
	380h+08h		ESIUBA	Event Specific Interrupt Upper Base Address
	380h+0Ch		ESIVB	Event Specific Interrupt Vector Base
			Reserved	Reserved

5.1 Register Map (cont'd)

	Start	End	Symbol	Description
Submission and Completion Queue related registers starts at address A = MCQCAP.QCFGPTR*200h [where y = 0..31]				
Submission Queue y	A+40h*y		SQATTRy	Submission Queue y Attributes
	A+40h*y+04		SQLBAy	Submission Queue y **L**ower **B**ase **A**ddress
	A+40h*y+08		SQUBAy	Submission Queue y **U**pper **B**ase **A**ddress
	A+40h*y+0C		SQDAOy	Submission Queue y Doorbell Address Offset
	A+40h*y+10		SQISAOy	Submission Queue y Interrupt Status Register Address offset
	A+40h*y+14		SQCFGy	Submission Queue y Configuration
	A+40h*y+18		Reserved	Reserved
	A+40h*y+1C		Reserved	Reserved
Completion Queue y	A+40h*y+20h		CQATTRy	Completion Queue y Attributes
	A+40h*y+24h		CQLBAy	Completion Queue y **L**ower **B**ase **A**ddress
	A+40h*y+28h		CQUBAy	Completion Queue y **U**pper **B**ase **A**ddress
	A+40h*y+2Ch		CQDAOy	Completion Queue y Doorbell Address Offset
	A+40h*y+30h		CQISAOy	Completion Queue y Interrupt Status Register Address offset
	A+40h*y+34h		CQCFGy	Completion Queue y Configuration
	A+40h*y+38h		Reserved	Reserved
	A+40h*y+3Ch		Reserved	Reserved
Following register are positioned configurably [y = 0..31]				
MCQ Operation & Runtime Registers	SQDAOy		SQHPy	Submission Queue y Head Pointer Doorbell
	SQDAOy+4		SQTPy	Submission Queue y Tail Pointer Doorbell
	SQDAOy+8h		SQRTCy	Submission Queue y Run Time Command
	SQDAOy+0Ch		SQCTIy	Submission Queue y Cleanup Task Information
	SQDAOy+10h		SQRTSy	Submission Queue y Run Time Status
	SQISAOy		SQISy	Submission Queue y Interrupt Status
	SQISAOy+4		SQIEy	Submission Queue y Interrupt Enable
	CQDAOy		CQHPy	Completion Queue y Head Pointer Doorbell
	CQDAOy+4		CQTPy	Completion Queue y Tail Pointer Doorbell
	CQISAOy		CQISy	Completion Queue y Interrupt Status
	CQISAOy+4		CQIEy	Completion Queue y Interrupt Enable
	CQISAOy+8		MCQIACRy	Interrupt Aggregation Control Register y

5.2 Host Controller Capabilities Registers

This sub-clause specifies the limits and capabilities of the host controller implementation. All Capability Registers are Read-Only (RO) or hardware Initialized. The offsets for these registers are all relative to the beginning of the host controller's MMIO address space.

5.2.1 Offset 00h: CAP – Controller Capabilities

Bit	Type	Reset	Description
31	RO	Impl Spec	**Event Specific Interrupt support (ESI):** When set, Controller supports Event Specific Interrupt topology, in addition to traditional interrupt scheme.
30	RO	Impl Spec	**MultiQueue Support (MCQS):** Indicates that Host controller can support Multiple Queues. • 0h: no support for MCQ • 1h: support for MCQ
29	RO	Impl Spec	**Legacy Single DoorBell Support (LSDBS):** Indicates the host controller support for legacy single doorbell mode with **U**TP **T**ransfer **R**equest **L**ist **B**ase **A**ddress located at 0x50h & its doorbell registers located at 0x58h. Legacy single doorbell mode allow the re-use of legacy SW. • 0h: legacy single doorbell support is available • 1h: indicate that legacy single doorbell support have been removed
28	RO	Impl Spec	**Crypto Support (CS):** Indicates whether the host controller supports cryptographic operations. • 0 – Host controller does not support cryptographic operations. • 1 – Host controller supports cryptographic operations.
27	RO		Reserved for Unified Memory Extension
26	RO	Impl Spec	**UIC DME_TEST_MODE command supported (UICDMETMS):** Indicates whether the host controller supports the UniPro DME_TEST_MODE.req SAP primitive.
25	RO	Impl Spec	**Out of order data delivery supported (OODDS):** Indicates whether the host controller supports out of order data delivery for UTP data transfer. When set to '1', the host controller shall support of out of order data delivery from the target device. When set to '0', the host controller will not support out of order data delivery from the target device. When OODDS is set to 1, Host Controller may fetch next PRDT Entry using Hint fields provided in previous DATA IN UPIU or RTT UPIU. When the current DATA IN UPIU or RTT UPIU does not match the previously cached Hint information, the host shall follow the information in the current DATA IN UPIU or RTT UPIU. Previously cached Hint information may be used for later DATA IN UPIU or RTT UPIU.

5.2.1 Offset 00h: CAP – Controller Capabilities (cont'd)

Bit	Type	Reset	Description
24	RO	Impl Spec	**64-bit addressing supported (64AS):** Indicates whether the host controller can access 64-bit data structures. When set to '1', the host controller shall make the 32-bit upper bits of the UTP Transfer Request List Base Address Upper 32-bit and UTP Task Management Request List Base Address upper 32-bit, the PRD Base, and each PRD entry read/write. When cleared to '0', these are read-only and treated as '0' by the host controller.
23	RO	Impl Spec	**Auto-Hibernation Support (AUTOH8):** Indicates whether the host controller supports auto-hibernation. • 0 – Host controller does not support auto-hibernation • 1 – Host controller supports auto-hibernation.
22	RO	Impl Spec	**EHS Length in UTRD Supported (EHSLUTRDS):** Indicates whether the host controller supports EHS Length field in UTRD. • 0 – Host controller takes EHS length from CMD UPIU, and SW driver use EHS Length field in CMD UPIU. • 1 – HW controller takes EHS length from UTRD, and SW driver use EHS Length field in UTRD. NOTE Recommend Host controllers move to taking EHS length from UTRD, and in UFS-5, it will be mandatory.
21:19	RO	0h	Reserved
18:16	RO	Impl Spec	**Number of UTP Task Management Request Slots (NUTMRS):** 0's based value indicating the number of slots provided by the UTP Task Management Request List. A minimum of 1 and maximum of 8 slots may be supported. **0's based value** **0:** 1 slot **1:** 2 slots **…** **7:** 8 slots
15:08	RO	Impl Spec	**Number of outstanding READY TO TRANSFER (RTT) requests supported (NORTT):** Indicates the maximum number of outstanding RTTs which are supported by the host controller.'0' based value indication the maximum number of RTTs that can be outstanding on the host at a particular instance. A minimum of 2 RTTs shall be supported and maximum is implementation specific.

5.2.1 Offset 00h: CAP – Controller Capabilities (cont'd)

Bit	Type	Reset	Description
07:00	RO	Impl Spec	**Number of UTP Transfer Request Slots (NUTRS):** For Legacy Single Doorbell mode, this indicates the number of slots provided by the UTP Transfer Request List. A minimum of 1 and maximum of 32 slots may be supported. For MCQ mode, this field specifies how many active transfer tasks the Host HW controller is capable of managing in parallel. The minimum of 1 and maximum of 256 slots may be supported. **0's based value** **0:** 1 slot **1:** 2 slots **…** **255:** 256 slots

5.2.2 Offset 04h: MCQCAP – Multi-Circular Queue Capability Register

This register indicates the capability of Multi-Circular Queue for the host controller. Relevant only when CAP.MCQS is set.

Bit	Type	Reset	Description
31:24	RO	Impl Specific	**MaxInterruptAggregationGroup (MIAG):** Maximum total number of interrupt Aggregation groups. In this version of standard, the value of MIAG shall not exceed 32.
23:16	RO	Impl Spec	**Queue Configuration Pointer (QCFGPTR):** An offset pointer to the base of the SQ/CQ Configuration Array, in 512 B units. This offset points to SQATTR0 register. Value shall be selected to ensure SQ/CQ configuration array does not overlap with x-CRYPTOCFG. ADDR (SQATTR0) = *UFS_HCI_BASE* + Q*CFGPTR**200h
15:11	RO	0h	**Reserved**
10	RO	Impl Specific	**Extended IID Support (EIS):** When set, controller supports 8 bit IID, consisting of {4-bit EXT_IID, 4-bit IID}
09	RO	1h	**Round Robin priority arbitration support (RRP):** When set, Controller supports simple round robin Arbitration Scheme
08	RO	Impl Spec	**Strict priority arbitration support (SP):** When set, Controller supports Strict priority Arbitration Scheme
07:00	RO	Impl Spec 0 ≤ Value ≤ 31	**Maximum number of Queues (MAXQ):** Maximum number of Queues this controller can support. In this version of specification, maximum value is 31. NOTE To support 1:1 topology, the Host HW controller must support HW resources for MAXQ number of Completion Queues too. Host SW may use less number of completion queues for N:1 topology..**:** 1 Queue. **0:** 1 Queue **1:** 2 Queues **…** **31:** 32 Queues **32-255** : reserved

5.2.3 Offset 08h: VER – UFS Version

This register indicates the major version, minor version and version suffix of the UFSHCI standard that the controller implementation supports. The lower two bytes represent the major version number, minor version number and the version suffix. Example: Version 3.12 would be represented as 0000_0312h.

Bit	Type	Reset	Description
31:16	RO	0000h	Reserved
15:08	RO	Impl Spec	**Major Version Number (MJR):** Major version in BCD format.
07:04	RO	Impl Spec	**Minor Version Number (MNR):** Minor version in BCD format.
03:00	RO	Impl Spec	**Version Suffix (VS):** Version suffix in BCD format

5.2.4 Offset 0Ch: EXT_CAP – Extended Controller Capabilities

Bit	Type	Reset	Description
31:16	RO	0h	Reserved
15:00	RO	Impl Spec	**wHostHintCacheSize**: This field specifies the maximum total number of hints for 4 kB data units in the host Hint Cache This formula gives the total possible hints for the controller: 32 x 2^wHostHintCacheSize. For example, if wHostHintCacheSize = 2, the maximum total number of outstanding Hint Data Count fields is 128 units, with each unit capable of containing a hint for 4 kB of data to be transferred. If the device sends more hints than indicated by wHostHintCacheSize, the host may replace one of the outstanding hints with the newly received hint within its capacity.

5.2.5 Offset 10h: HCPID – Host Controller Identification Descriptor – Product ID

This register indicates the product identification information for host controller.

Bit	Type	Reset	Description
31:00	RO	Impl spec	**Product ID (PID):** Product ID that host controller manufacturer assigns for the host controller. This is vendor specific.

5.2.6 Offset 14h: HCMID – Host Controller Identification Descriptor – Manufacturer ID

This register provides manufacturer identification information for host controller manufacturer. The Manufacturer ID is defined by JEDEC in Standard Manufacturer's identification code [JEP106]. The Manufacturer ID consists of two parts: Manufacturer Identification Code and Bank Index.

Bit	Type	Reset	Description
31:16	RO	0	Reserved
15:08	RO	Impl spec	**Bank Index (BI):** This field contains an index value of the bank that contains the Manufacturer Identification Code. The BI value shall be equal to the number of the continuation fields that precede the MIC as specified by [JEP106].
07:00	RO	Impl spec	**Manufacturer Identification Code (MIC):** Manufacturer Identification code as defined by JEDEC in Standard Manufacturer's identification code [JEP106].

5.2.7 Offset 18h: AHIT – Auto-Hibernate Idle Timer

UFS utilizes components of the UniPro and INCITS T10 draft standards as its power management framework. To improve power efficiency, UFS Host Controller may support a mechanism called auto-hibernation.

Auto-hibernate allows the host controller to put UniPro link into Hibernate state autonomously. Host register **CAP.AUTOH8** provides a method for software to detect support of this feature. **AHIT.AH8ITV** provides a method for software to directly control of this feature.

Bit	Type	Reset	Description
31:13	RO	0	Reserved
12:10	RW	Impl Spec	**Timer scale (TS):** • 000 – Value times 1 us • 001 – Value times 10 us • 010 – Value times 100 us • 011 – Value times 1 ms • 100 – Value times 10 ms • 101 – Value times 100 ms • 110 - 111 – reserved
09:00	RW	0	**Auto-Hibern8 Idle Timer Value (AH8ITV)**: This is the timer that UFS subsystem must be idle before UFS host controller may put UniPro link into Hibernate state autonomously. The idle timer value is multiplied by the indicated timer scale to yield an absolute timer value. The idle timer starts decrement when all of the following conditions are satisfied: • UTMRLDBR='0' • No UIC command is outstanding • If the queue type is in legacy single doorbell mode, UTRLDBR='0' • But if the queue type is in MCQ mode, all queued task are completed The idle timer shall continue decrement until it reaches zero or it stopped as a result of software accesses one of the host controller interface registers. When idle timer changes a non-zero to zero, host controller shall put UniPro link into Hibernate state. Host controller reloads this value each time the UniPro link transitions out of the Hibernate state. Software writes "0" to disable Auto-Hibernate Idle Timer. Any non-zero value will enable Auto-Hibernate idle timer. UFS host controller shall put UniPro link out of Hibernate state when the link communication is required. The mechanism to decide when the UniPro link needs to become active is host controller specific implementation, and is transparent to the software.

5.3 Operation and Runtime Registers

This sub-clause defines the operation and runtime registers exposed by the host controller.

5.3.1 Offset 20h: IS – Interrupt Status

This register indicates pending interrupts that require service.

Bit	Type	Reset	Description
31:22	RO	0h	Reserved
21	RO	0	**MCQ Interrupt Aggregation Event Status (IAGES):** This bit is transparent and becomes '1' when all of the following conditions are met • Controller is operating in MCQ mode (**Config.QT =** 1) • ESI is not enabled (**Config.ESIE = 0)** • At least one interrupt aggregation group has triggered, which means it has satisfied either counter or timer condition When in MCQ mode, and ESI is not used, SW can use traditional interrupt approach. When this bit is set, interrupt routine needs to scan all interrupt aggregation groups to determine which IAG has caused this interrupt. When none of the AIGs are triggered, this bit is automatically cleared.
20	RO	0	**MCQ CQ Event Status (CQES):** This bit is transparent and becomes '1' when all of the following conditions are met • Controller is operating in MCQ mode (**Config.QT =** 1) • ESI is not enabled (**Config.ESIE = 0)** • CQES set only for Events in Queues that do not have interrupt aggregation enabled or the Events that do not belong to MCQIACRy.IACTH counter operation criteria. • At least one bit in CQISy is set and associated bit in CQIEy is set. y = 0..31 When in MCQ mode, and ESI is not used, SW can use traditional interrupt approach. When this bit is set, interrupt routine needs to scan all CQISy registers to determine which event for which CQ has caused this interrupt. To clear this bit, all CQISy[y = 0..31] bits must be cleared by interrupt routine.
19	RO	0	**MCQ SQ Event Status (SQES):** This bit is transparent and becomes '1' when all of the following conditions are met • Controller is operating in MCQ mode (**Config.QT = 1**) • ESI is not enabled (**Config.ESIE = 0)** • SQES set only for Events in Queues that do not have interrupt aggregation enabled or the Events that do not belong to MCQIACRy.IACTH counter operation criteria • At least one bit in SQISy is set and associated bit in SQIEy is set. y = 0..31 When in MCQ mode, and ESI is not used, SW can use traditional interrupt approach. When this bit is set, interrupt routine needs to scan all SQISy registers to determine which event for which SQ has caused this interrupt. To clear this bit, all SQISy[y = 0..31] bits must be cleared by interrupt routine.

5.3.1 Offset 20h: IS – Interrupt Status (cont'd)

Bit	Type	Reset	Description
18	RWC	0	**Crypto Engine Fatal Error Status (CEFES):** Indicates that the host controller's encryption/decryption hardware has encountered an error from which it cannot recover. When the error occurs, the host controller is stopped and both **UTRLRSR** and **UTMRLRSR** will be cleared to "0" by host controller. If the error occurs, host SW should reset the host controller.
17	RWC	0	**System Bus Fatal Error Status (SBFES):** Indicates that the host controller encountered a system bus error that it cannot recover from, such as a bad software pointer. When the error occurs, the host controller shall perform the steps defined in 8.2.1. Host software shall reset the device and the host controller whenever this error occurs.
16	RWC	0	**Host Controller Fatal Error Status (HCFES)**: Indicates that the host controller encountered a fatal error that it cannot recover from. When the error occurs, the host controller is stopped and both **UTRLRSR** and **UTMRLRSR** will be cleared to "0" by host controller. If the error occurs, host SW should reset the host controller.
15:13	RO		Reserved.
12	RWC	0	**UTP Error Status (UTPES):** Indicates that the host controller encountered an error at UTP layer that it cannot recover from. When the error occurs, the host controller will update UTP error code field within Host Controller Status register. It is up to host software to decide how to handle the error condition.
11	RWC	0	**Device Fatal Error Status (DFES):** Indicates that the host controller encountered a fatal error from device that it cannot recover. When the error occurs, the host controller shall perform the steps defined in 8.2.6. If the error occurs, host SW should reset the device only or the device and host controller.
10	RWC	0	**UIC Command Completion Status (UCCS):** This bit is set to '1' by the host controller upon completion of a UIC command.
09	RWC	0	**UTP Task Management Request Completion Status (UTMRCS):** This bit is set to '1' by the host controller upon completion of a task management function whose UTMRD.I bit is set.
08	RWC	0	**UIC Link Startup Status (ULSS):** indication that Link start-up process has been initiated by the remote end of the Link. This bit corresponds to the UniPro DME_LINKSTARTUP.ind SAP primitive.
07	RWC	0	**UIC Link Lost Status (ULLS):** This indicates a condition where remote end is trying to re-establish a link and the link is lost. This bit corresponds to the UniPro DME_LINKLOST.ind SAP primitive.
06	RWC	0	**UIC Hibernate Enter Status (UHES):** When the hibernate entering process is initiated by host software, this field indicates that UniPro hibernate entering process has been completed. If the process was successful, the Link state is changed to the Hibernate state. Register HCS.UPMCRS indicates the status of the hibernation entering process. When the hibernate entering process is initiated by auto-hibernation as defined in **AHIT**, this field indicates that error is detected during hibernate entering process. Register **HCS.UPMCRS** indicates the error status. This bit corresponds to the UniPro DME_HIBERNATE_ENTER.ind SAP primitive.

5.3.1 Offset 20h: IS – Interrupt Status (cont'd)

Bit	Type	Reset	Description
05	RWC	0	**UIC Hibernate Exit Status (UHXS):** When the hibernate exiting process is initiated by host software, this field indicates that the Link has exited UniPro Hibernate state. Register **HCS.UPMCRS** indicates the status of the hibernation exiting process. When the hibernate exiting process is initiated by auto-hibernation as defined in **AHIT**, this field indicates that error is detected during hibernate exiting process. Register **HCS.UPMCRS** indicates the error status. This bit corresponds to the UniPro DME_HIBERNATE_EXIT.ind SAP primitive.
04	RWC	0	**UIC Power Mode Status (UPMS):** This field indicates a completion of either the power mode change or the TX Equalization Training. **HCS.UPMCRS** indicates the completion status of a UIC layer request: 1. For power mode change, **HCS.UPMCRS** corresponds to the UniPro DME_POWERMODE.ind SAP primitive. 2. For TX Equalization Training, **HCS.UPMCRS** corresponds to the UniPro DME_EQTR.ind SAP primitive.
03	RWC	0	**UIC Test Mode Status (UTMS):** Indicate that the peer UniPro stack has been set to a given UniPro test mode. This bit corresponds to the UniPro DME_TEST_MODE.ind SAP primitive.
02	RWC	0	**UIC Error (UE):** Indicate that a layer in the UniPro stack has encountered an error condition. Register **UECPA/UECDL/UECN/UECT/UECDME** contains the error code for the condition. This bit corresponds to the UniPro DME_ERROR.ind SAP primitive. See 8.1.2, UIC Error.
01	RWC	0	**UIC DME_ENDPOINTRESET Indication(UDEPRI):** Indicate that the attached device has issued an *DME_ENDPOINTRESET* indication which is not allowed.
00	RWC	0	**UTP Transfer Request Completion Status (UTRCS):** When host controller is operating in Legacy Single Doorbell mode (CONFIG.QT=0), this bit is set to '1' by the host controller upon one of the following: • Completion of a UTP transfer request with its UTRD Interrupt bit set to '1'. • Interrupt caused by the UTR interrupt aggregation logic. • Overall command Status (OCS) of the completed command is not equal to "SUCCESS" even if its UTRD Interrupt bit set to '0'.

5.3.2 Offset 24h: IE – Interrupt Enable

This register enables and disables the reporting of the corresponding interrupt to host software. When a bit is set (‘1’) and the corresponding interrupt condition is active, then an interrupt is generated. Interrupt sources that are disabled (‘0’) are still indicated in the IS register. This register is symmetrical with the IS register.

Bit	Type	Reset	Description
31:22	RO	0h	Reserved
21	RW	0	MCQ Interrupt Aggregation Event Enable **(IAGEE**): When set and IS.IAGES is set, the controller shall generate an interrupt.
20	RW	0	**MCQ CQ Event Enable (CQEE**): When set and IS.CQES is set, the controller shall generate an interrupt.
19	RW	0	**MCQ SQ Event Enable (SQEE**): When set and IS.SQES is set, the controller shall generate an interrupt.
18	RW	0	**Crypto Engine Fatal Error Enable (CEFEE):** When set and **IS.CEFES** is set, the controller shall generate an interrupt.
17	RW	0	**System Bus Fatal Error Enable (SBFEE):** When set and **IS.SBFES** is set, the controller shall generate an interrupt.
16	RW	0	**Host Controller Fatal Error Enable (HCFEE):** When set and **IS.HCFES** is set, the controller shall generate an interrupt.
15:13	RO	0	Reserved
12	RW	0	**UTP Error Enable (UTPEE):** When set and IS.UTPES is set, the controller shall generate an interrupt.
11	RW	0	**Device Fatal Error Enable (DFEE):** When set and **IS.DFES** is set, the host controller shall generate an interrupt.
10	RW	0	**UIC COMMAND Completion Enable (UCCE):** When set and **IS.UCCS** is set, the host controller shall generate an interrupt.
09	RW	0	**UTP Task Management Request Completion Enable (UTMRCE):** When set and **IS.UTMRCS** is set, the host controller shall generate an interrupt.
08	RW	0	**UIC Link Startup Status Enable (ULSSE):** When set and **IS.ULSS** is set, the controller shall generate an interrupt.
07	RW	0	**UIC Link Lost Status Enable (ULLSE):** When set and **IS.ULLS** is set, the controller shall generate an interrupt.
06	RW	0	**UIC Hibernate Enter Status Enable (UHESE):** When set and **IS.UHES** is set, the controller shall generate an interrupt.
05	RW	0	**UIC Hibernate Exit Status Enable (UHXSE):** When set and **IS.UHXS** is set, the controller shall generate an interrupt.

5.3.2 Offset 24h: IE – Interrupt Enable (cont'd)

Bit	Type	Reset	Description
04	RW	0	**UIC Power Mode Status Enable (UPMSE):** When set and **IS.UPMS** is set, the controller shall generate an interrupt.
03	RW	0	**UIC Test Mode Status Enable (UTMSE):** When set and **IS.UTMS** is set, the controller shall generate an interrupt.
02	RW	0	**UIC Error Enable (UEE):** When set and **IS.UEE** is set, the controller shall generate an interrupt.
01	RW	0	**UIC DME_ENDPOINTRESET (UDEPRIE):** When set and **IS. UDEPRI** is set, the controller shall generate an interrupt.
00	RW	0	**UTP Transfer Request Completion Enable (UTRCE):** When set and **IS.UTRCS** is set, the host controller shall generate an interrupt, when host controller is operating in Legacy Single Doorbell mode (**Config.QT = 0**).

5.3.3 Offset 2Ch: HCSEXT – Host Controller Status Extended

The following fields are provided by host controller for an error condition detected within UTP layer. They are valid only when UTPES is set. They are automatically reset by host controller when UTPES is cleared. In case multiple UTP errors occur before UTPES is cleared, the first UTP error is kept in the following fields.

Bit	Type	Reset	Description
31:08	RO	0	Reserved
07:04	RO	0	**EXT_IID of UTP Error (EXT_IIDUTPE):** The EXT_IID of the command that a UTP error occurs during execution of the command.
03:00	RO	0	**IID of UTP Error (IIDUTPE):** The ID of the command that a UTP error occurs during execution of the command.

5.3.4 Offset 30h: HCS – Host Controller Status

The following fields are provided by host controller for an error condition detected within UTP layer. They are valid only when UTPES is set. They are automatically reset by host controller when UTPES is cleared. In case multiple UTP errors occur before UTPES is cleared, the first UTP error is kept in the following fields.

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:12</td><td>RO</td><td>0</td><td>Bit[31:24] – Target LUN of UTP error (TLUNUTPE): The LUN of the command that a UTP error occurs during execution of the command.
Bit[23:16] – Task Tag of UTP error (TTAGUTPE): The Task Tag of the command that a UTP error occurs during execution of the command.
Bit[15:12] - UTP Error Code (UTPEC): Indicate that the error code of a UTP layer error.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>Reject UPIU has invalid EXT_IID, IID, Task Tag or LUN. This error occurs when UFS host receives Reject UPIU from UFS device, but there is no pending Transfer Request nor pending Task Management Request with matching EXT_IID, IID, Task Tag or LUN.</td></tr>
<tr><td>1h</td><td>Invalid UPIU type.</td></tr>
<tr><td>2h</td><td>Inbound UPIU associated with Transfer Request has invalid EXT_IID, IID, Task Tag or LUN. This error occurs when UFS host receives inbound UPIU from UFS device that is associated with Transfer Request, but there is no pending Transfer Request with matching Task Tag or LUN.</td></tr>
<tr><td>3h</td><td>Inbound UPIU associated with Task Management Request has invalid EXT_IID, IID, Task Tag or LUN. This error occurs when UFS host receives inbound UPIU from UFS device that is associated with Task Management Request, but there is no pending Task Management Request with matching EXT_IID, IID, Task Tag or LUN.</td></tr>
<tr><td>4h - Fh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>11</td><td>RO</td><td>0</td><td>Reserved</td></tr>
</table>

5.3.4 Offset 30h: HCS – Host Controller Status (cont'd)

<table>
<tr><td>10:08</td><td>RO</td><td>0</td><td>UIC Power Mode Change Request Status (UPMCRS): Indicates the completion status of a UIC layer request for either power mode change or TX Equalization Training.

<table><tr><th>Value</th><th>Description</th></tr><tr><td>0h</td><td>PWR_OK. The request was accepted.</td></tr><tr><td>1h</td><td>PWR_LOCAL. The local request was successfully applied.</td></tr><tr><td>2h</td><td>PWR_REMOTE.The remote request was successfully applied.</td></tr><tr><td>3h</td><td>PWR_BUSY.The request was aborted due to concurrent requests.</td></tr><tr><td>4h</td><td>PWR_ERROR_CAP.The request was rejected because the requested configuration exceeded the Link's capabilities.</td></tr><tr><td>5h</td><td>PWR_FATAL_ERROR. The request was aborted due to a communication problem. The Link may be inoperable.</td></tr><tr><td>6h - 7h</td><td>Reserved</td></tr></table></td></tr>
<tr><td>07:04</td><td>RO</td><td>0</td><td>Reserved.</td></tr>
<tr><td>03</td><td>RO</td><td>0</td><td>UIC COMMAND Ready (UCRDY): This field indicates whether the host controller is ready to process UIC COMMAND. Host software shall only set the UICCMD if HCS.UCRDY is set to '1'.</td></tr>
<tr><td>02</td><td>RO</td><td>0</td><td>UTP Task Management Request List Ready (UTMRLRDY): This field is set to '1' when the host controlleris ready to Task Management requests. This field is cleared to '0' by host controller when one of the following conditions occur:

• The device presence is not detected;
• UTP Task Management Request List is full;
• There is an error with host controller or device that is not command specific.

Host software shall only set the UTMRLRSR register if HCS.UTMRLRDY is set to '1'.</td></tr>
<tr><td>01</td><td>RO</td><td>0</td><td>UTP Transfer Request List Ready (UTRLRDY): This field indicates whether the host controller is ready to process UTP Transfer Request.

This field is cleared to '0' by host controller when one of the following conditions occur:

• The device presence is not detected;
• UTP Transfer Request List is full;
• There is an error with host controller or device that is not command specific.

Host software shall only set the UTRLRSR bit to '1' if HCS. UTRLRDY is set to '1'.</td></tr>
<tr><td>00</td><td>RO</td><td>0</td><td>Device Present (DP): This field is set to '1' after host controller receive 'SUCCESS' return code on the response of the DME_LINKSTARTUP UIC CMD during host controller initialization when an UFS device is detected at physical link that is attached to the controller. This field is cleared to '0' when no UFS device is detected or host controller unable to communicate with the attached device successfully.</td></tr>
</table>

5.3.5 Offset 34h: HCE – Host Controller Enable

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:02</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>01</td><td>RW</td><td>0</td><td>Crypto General Enable (CGE): Enable/Disable bit for Crypto Engine
<table><tr><th>Bit Value</th><th>Description</th></tr><tr><td>0</td><td>Disable cryptographic operations for all transactions</td></tr><tr><td>1</td><td>Enable cryptographic operations for transactions where UTRD.CE = 1</td></tr></table></td></tr>
<tr><td>00</td><td>RW</td><td>0</td><td>Host Controller Enable (HCE): When HCE is ‘0’ and software writes ‘1’, the host controller hardware shall execute the step 2) described in 7.1.1 of this standard, including reset of the host UTP and UIC layers. When the initialization process is completed, host controller will set the register to ‘1’. When HCE is ‘1’ and software writes ‘0’, host controller will disable the host controller hardware and the device attached. Host controller will clear the register to ‘0’ after completing disable operation. Software shall wait until HCE = ‘0’ to conclude host controller disable. Mechanisms which enable reactivation of the host controller by software (e.g. HCE) shall be available.
Writing ‘0’ when HCE = ‘0’, and writing ‘1’ when HCE = ‘1’ shall have no effect.
<table><tr><th>Bit Value</th><th>Description</th></tr><tr><td>0</td><td>Disable</td></tr><tr><td>1</td><td>Enable</td></tr></table></td></tr>
</table>

5.3.6 Offset 38h: UECPA – Host UIC Error Code PHY Adapter Layer

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:31</td><td>ROC</td><td>0</td><td>UIC PHY AdapterA Layer Error (ERR): Indicates whether an error was generated by the PHY Adapter Layer</td></tr>
<tr><td>30:05</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>04:00</td><td>ROC</td><td>0</td><td>UIC PHY Adapter Layer Error Code (EC): error code generated when IS.UE and UECPA.ERR are set to ‘1’.
<table><tr><th>Bit</th><th>Description</th></tr><tr><td>00</td><td>Error reported by the M-PHY layer: PHY error on Lane 0.</td></tr><tr><td>01</td><td>Error reported by the M-PHY layer: PHY error on Lane 1</td></tr><tr><td>02</td><td>Error reported by the M-PHY layer: PHY error on Lane 2</td></tr><tr><td>03</td><td>Error reported by the M-PHY layer: PHY error on Lane 3</td></tr><tr><td>04</td><td>Generic PHY Adapter Error. This should be the LINERESET indication. Categorized as "ERROR" (Not FATAL). SW is informed that M-PHY has been reset and all M-PHY Attributes (that are not handled by UniPro) need to be restored in order to keep the link optimized.</td></tr></table></td></tr>
</table>

5.3.7 Offset 3Ch: UECDL – Host UIC Error Code Data Link Layer

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:31</td><td>ROC</td><td>0</td><td>UIC Data Link Layer Error (ERR): Indicates whether an error was generated by the Data Link Layer</td></tr>
<tr><td>30:16</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>15:00</td><td>ROC</td><td>0</td><td>UIC Data Link Layer Error Code (EC): error code generated when IS.UE and UECDL.ERR are set to ‘1’. Refer to UniPro Specification for the definition of the error codes.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>00</td><td>NAC_RECEIVED</td></tr>
<tr><td>01</td><td>TCx_REPLAY_TIMER_EXPIRED</td></tr>
<tr><td>02</td><td>AFCx_REQUEST_TIMER_EXPIRED</td></tr>
<tr><td>03</td><td>FCx_PROTECTION_TIMER_EXPIRED</td></tr>
<tr><td>04</td><td>CRC_ERROR</td></tr>
<tr><td>05</td><td>RX_BUFFER_OVERFLOW</td></tr>
<tr><td>06</td><td>MAX_FRAME_LENGTH_EXCEEDED</td></tr>
<tr><td>07</td><td>WRONG_SEQUENCE_NUMBER</td></tr>
<tr><td>08</td><td>AFC_FRAME_SYNTAX_ERROR</td></tr>
<tr><td>09</td><td>NAC_FRAME_SYNTAX_ERROR</td></tr>
<tr><td>10</td><td>EOF_SYNTAX_ERROR</td></tr>
<tr><td>11</td><td>FRAME_SYNTAX_ERROR</td></tr>
<tr><td>12</td><td>BAD_CTRL_SYMBOL_TYPE</td></tr>
<tr><td>13</td><td>PA_INIT_ERROR</td></tr>
<tr><td>14</td><td>PA_ERROR_IND_RECEIVED</td></tr>
<tr><td>15</td><td>PA_INIT</td></tr>
</table></td></tr>
</table>

5.3.8 Offset 40h: UECN – Host UIC Error Code Network Layer

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:31</td><td>ROC</td><td>0</td><td>UIC Network Layer Error (ERR): Indicates whether an error was generated by the Network Layer</td></tr>
<tr><td>30:03</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>02:00</td><td>ROC</td><td>0</td><td>UIC Network Layer Error Code (EC): error code generated when IS.UE and UECN.ERR are set to ‘1’. Refer to UniPro Specification [MIPI-UniPro] for the definition of the error codes.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>00</td><td>UNSUPPORTED_HEADER_TYPE</td></tr>
<tr><td>01</td><td>BAD_DEVICEID_ENC</td></tr>
<tr><td>02</td><td>LHDR_TRAP_PACKET_DROPPING</td></tr>
</table></td></tr>
</table>

5.3.9 Offset 44h: UECT – Host UIC Error Code Transport Layer

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:31</td><td>ROC</td><td>0</td><td>UIC Transport Layer Error (ERR): Indicates whether an error was generated by the Transport Layer</td></tr>
<tr><td>30:07</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>06:00</td><td>ROC</td><td>0</td><td>UIC Transport Layer Error Code (EC): error code generated when IS.UE and UECT.ERR are set to ‘1’. Refer to UniPro Specification for the definition of the error codes.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>00</td><td>UNSUPPORTED_HEADER_TYPE</td></tr>
<tr><td>01</td><td>UNKNOWN_CPORTID</td></tr>
<tr><td>02</td><td>NO_CONNECTION_RX</td></tr>
<tr><td>03</td><td>CONTROLLED_SEGMENT_DROPPING</td></tr>
<tr><td>04</td><td>BAD_TC</td></tr>
<tr><td>05</td><td>E2E_CREDIT_OVERFLOW</td></tr>
<tr><td>06</td><td>SAFETY_VALVE_DROPPING</td></tr>
</table></td></tr>
</table>

5.3.10 Offset 48h: UECDME – Host UIC Error Code

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:31</td><td>ROC</td><td>0</td><td>UIC DME Error (ERR): Indicates whether an error was generated by the DME</td></tr>
<tr><td>30:04</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>03:00</td><td>ROC</td><td>0</td><td>UIC DME Error Code (EC): error code generated when IS.UE and UECDME.ERR are set to ‘1’.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>00</td><td>Generic DME error.</td></tr>
<tr><td>01</td><td>QoS from TX is detected. This bit corresponds to the UniPro DME_QoS.ind(TX) SAP primitive.</td></tr>
<tr><td>02</td><td>QoS from RX is detected. This bit corresponds to the UniPro DME_QoS.ind(RX) SAP primitive.</td></tr>
<tr><td>03</td><td>QoS from PA_INIT is detected. This bit corresponds to the UniPro DME_QoS.ind(PA_INIT) SAP primitive.</td></tr>
</table></td></tr>
</table>

5.3.11 Offset 4Ch: UTRIACR – UTP Transfer Request Interrupt Aggregation Control Register

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31</td><td>RW</td><td>0</td><td>Interrupt Aggregation Enable/Disable (IAEN): When set to ‘0’ by host software, command reponses are neither counted nor timed. Interrupts are still triggerred by responses to Interrupt Commands.
When set to ‘1’, the interrupt aggregation mechanism is enabled and aggregation-based interrupts are generated.
<table>
<tr><th>Bit Value</th><th>Description</th></tr>
<tr><td>0</td><td>Disable</td></tr>
<tr><td>1</td><td>Enable</td></tr>
</table></td></tr>
<tr><td>30:25</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>24</td><td>WO</td><td>0</td><td>Interrupt aggregation parameter write enable (IAPWEN): When host SW writes ‘1’, the values in IACTH and IATOVAL are updated with the contents written at the same cycle.
When host SW writes ‘0’, the values in IACTH and IATOVAL are not updated.
NOTE Write operations to IACTH and IATOVAL are only allowed when no commands are outstanding.</td></tr>
<tr><td>23:21</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>20</td><td>RO</td><td>0</td><td>Interrupt aggregation status bit (IASB): This bit indicates to Host SW whether any responses have been received and counted towards interrupt aggregation (i.e., IASB is set iff IA counter > 0).
<table>
<tr><th>Bit Value</th><th>Description</th></tr>
<tr><td>0</td><td>No commands has been received since last counter reset (IA counter = 0)</td></tr>
<tr><td>1</td><td>At least one command has been received and counted (IA counter > 0)</td></tr>
</table></td></tr>
<tr><td>19:17</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>16</td><td>WO</td><td>0</td><td>Counter and Timer Reset (CTR): When host SW writes ‘1’, the interrupt aggregation timer and counter are reset.
It is recommended that host software use this field to reset the timer and counter every time it services newly received UTP responses.</td></tr>
<tr><td>15:13</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>12:8</td><td>RW</td><td>0</td><td>Interrupt aggregation counter threshold (IACTH): Host SW uses this field to configure the number of responses that are required to generate an interrupt.
Counter Operation: As UTP responses are received by the host controller, they are counted. The counter is reset by software during the interrupt service routine. It increments with every response to a Regular Transfer Request Command received at the host controller. The counter stops counting when it reaches the value configured in IACTH, and sets the IS.UTRCS bit.
The maximum allowed value is 31
NOTE 1 When IACTH is 0, responses are not counted, and counting-based interrupts are not generated.
NOTE 2 QUERY RESPONSE UPIUs and NOP IN UPIUs shall not be counted by the Interrupt Aggregation logic.
In order to write to this field, the IAPWEN bit must be set at the same write operation.</td></tr>
</table>

5.3.11 Offset 4Ch: UTRIACR – UTP Transfer Request Interrupt Aggregation Control Register (cont'd)

7:0	RW	0	**Interrupt Aggregation Timeout Value (IATOVAL):** Host SW uses this field to configure the maximum time allowed between a response arrival to the host controller and the generation of an interrupt. **Timer Operation:** The timer is reset by software during the interrupt service routine. It starts running when the host controller receives the first response to a Regular Transfer Request Command, after the timer was reset. The timer stops when it reaches the value configured in **IATOVAL** field, and **IS.UTRCS** bit is set. NOTE 1 When **IATOVAL** is 0, the timer is not running, and timer-based interrupts are not generated. NOTE 2 QUERY RESPONSE UPIUs and NOP IN UPIUs shall not be counted by the Interrupt Aggregation logic. The Time units in this field are 40 us. Therefore, writing 0x01 represents a time-out value of 40 us, and writing 0xFF represents a time-out value of 10.2 ms

5.4 UTP Transfer Request Registers

5.4.1 Offset 50h: UTRLBA – UTP Transfer Request List Base Address

Bit	Type	Reset	Description
31:10	RW	Impl Spec	**UTP Transfer Request List Base Address (UTRLBA):** Indicates the 32-bit base physical address for the UTP Transfer Request list. This base is used when fetching commands for execution. The structure pointed to by this address range is 1 KB in length. This address shall be 1 KB aligned as indicated by bits 09:00 being read only.
09:00	RO	0	Reserved

5.4.2 Offset 54h: UTRLBAU – UTP Transfer Request List Base Address Upper 32-bits

Bit	Type	Reset	Description
31:00	RW	Impl Spec	**UTP Transfer Request List Base Address Upper (UTRLBAU):** Indicates the upper 32-bits for the UTP Transfer Request list base physical address. This base is used when fetching commands for execution.

5.4.3 Offset 58h: UTRLDBR – UTP Transfer Request List DoorBell Register

Bit	Type	Reset	Description
31:0	RWS	0	**UTP Transfer Request List DoorBell Register (UTRLDBR):** This field is bit significant. Each bit corresponds to a slot in the UTP Transfer Request List, where bit 0 corresponds to request slot 0. A bit in this field is set to '1' by host software to indicate to the host controller that a transfer request has been built in system memory for the associated transfer request slot and may be ready for execution. The host software indicates no change to request slots by setting the associated bits in this field to '0'. Bits in this field shall only be set to '1' by host software when **UTRLRSR** is set to '1'. When a transfer request is completed (with success or error), the corresponding bit is cleared to '0' by the host controller. The host controller always process transfer requests in-order according to the order submitted to the list. In case of multiple commands with single doorbell register ringing (batch mode), The dispatch order for these transfer requests by host controller will base on their index in the List. A transfer request with lower index value will be executed before a transfer request with higher index value. This field is also cleared when **UTRLRSR** is written from a '1' to a '0' by host software.

5.4.4 Offset 5Ch: UTRLCLR – UTP Transfer Request List CLear Register

Bit	Type	Reset	Description
31:0	WO	0	**UTP Transfer Request List CLear Register (UTRLCLR):** This field is bit significant. Each bit corresponds to a slot in the UTP Transfer Request List, where bit 0 corresponds to request slot 0. A bit in this field is set to '0' by host software to indicate to the host controller that a transfer request slot is cleared. The host controller shall free up any resources associated to the request slot immediately, and shall set the associated bit in UTRLDBR to '0'. The host software indicates no change to request slots by setting the associated bits in this field to '1'. Bits in this field shall only be set '1' or '0' by host software when **UTRLRSR** is set to '1'. The host software shall use this field only when a UTP Transfer Request is expected to not be completed, e.g., when the host software receives a "FUNCTION COMPLETE" Task Management response which means a Transfer Request was aborted.

5.4.5 Offset 60h: UTRLRSR – UTP Transfer Request List Run Stop Register

Bit	Type	Reset	Description
31:01	RO	0	Reserved
0	RW	0	**UTP Transfer Request List Run-Stop Register (UTRLRSR):** When set to '1', the host controller may process the list. Host controller starts processing the list at entry '0'. The host controller continues process the list as long as this bit is set to a '1'. When cleared to '0', the host controller shall continue to complete all the outstanding transfer requests in the list and then stop. This bit shall only be set to '1' when **HCS.UTRLRDY** is set to '1'.

5.4.6 Offset 64h: UTRLCNR – UTP Transfer Request List Completion Notification Register

Bit	Type	Reset	Description
31:0	RWC	0	**UTP Transfer Request List Completion Notification Register (UTRLCNR):** This field is bit significant. Each bit corresponds to a slot in the UTP Transfer Request List, where bit 0 corresponds to request slot 0. A bit in this field is set to '1' by the host controller when a transfer request from the associated transfer request slot has completed (with success or error). The host controller sets the bit at the same time it clears the bit with the same index in UTRLDBR. Host software is expected to clear the bit, by writing '1' to it, after processing the completed task. Clearing a bit in this register shall have no effect on the hardware, other than changing the value of this register. The host controller shall clear this register when **UTRLRSR** is written from a '0' to a '1' by host software.

5.5 UTP Task Management Registers

5.5.1 Offset 70h: UTMRLBA – UTP Task Management Request List Base Address

Bit	Type	Reset	Description
31:10	RW	Impl Spec	**UTP Task Management Request List Base Address (UTMRLBA):** Indicates the 32-bit base physical address for the list. This base is used when fetching Task Management Functions for execution. The structure pointed to by this address range is 640 Bytes in length. This address shall be 1 KB aligned as indicated by bits 09:00 being read only.
09:00	RO	0	Reserved

5.5.2 Offset 74h: UTMRLBAU – UTP Task Management Request List Base Address Upper 32-bits

Bit	Type	Reset	Description
31:00	RW	Impl Spec	**UTP Task Management Request List Base Address (UTMRLBAU):** Indicates the upper 32-bits for the list base physical address. This base is used when fetching task management functions for execution.

5.5.3 Offset 78h: UTMRLDBR – UTP Task Management Request List DoorBell Register

Bit	Type	Reset	Description
31:08	RO	0	**Reserved**
07:0	RWS	0	**UTP Task Management Request List DoorBell Register (UTMRLDBR):** This field is bit significant. Each bit corresponds to a slot in the task management request List, where bit 0 corresponds to slot 0. A bit in this field is set by host software to indicate to the host controller that a task management request has been built in system memory for the associated task management request slot, and may be ready for execution. The host software indicates no change to request slots by setting the associated bits in this field to '0'. Bits in this field shall only be set to '1' by host software when **UTMRLRSR** is set to '1'. When a task management request is completed (with success or error), the corresponding bit is cleared to '0' by the host controller. The host controller always processes task management requests in-order according to the order submitted to the list. In case of multiple requests with single doorbell register ringing (batch mode), The dispatch order for these requests by host controller will base on their index in the List. A task management with lower index value will be executed before a task management request with higher index value. This field is also cleared when **UTMRLRSR** is written from a '1' to a '0' by host software.

5.5.4 Offset 7Ch: UTMRLCLR – UTP Task Management Request List CLear Register

Bit	Type	Reset	Description
31:08	RO	0	**Reserved**
07:0	WO	0	**UTP Task Management List CLear Register (UTMRLCLR):** This field is bit significant. Each bit corresponds to a slot in the task management request List, where bit 0 corresponds to slot 0. A bit in this field is set to '0' by host software to indicate to the host controller that a task management request slot is cleared. The host controller shall free up any resources associated to the task management request slot immediately, and shall set the associated bit in UTMRLDBR to '0'. The host software indicates no change to task management request slots by setting the associated bits in this field to '1'. Bits in this field shall only be set '1' or '0' by host software when **UTRLRSR** is set to '1'. The host software shall use this field only when a UTP Task Management Request is expected to not be completed, e.g., in case of a system bus error, such as an invalid UTMRD.

5.5.5 Offset 80h: UTMRLRSR – UTP Task Management Request List Run Stop Register

Bit	Type	Reset	Description
31:01	RO	0	Reserved
0	RW	0	**UTP Task Management Request List Run-Stop Register (UTMRLRSR):** When set to '1', the host controller may process the list. Host controller starts processing the list at entry '0'. The host controller continues process the list as long as this bit is set to a '1'. When cleared to '0', the host controller shall continue to complete all the outstanding task management requests in the list and then stop. This bit shall only be set to '1' when **HCS.UTMRLRDY** is set to '1'.

5.6 UIC Command Registers

5.6.1 Offset 90h: UICCMD – UIC Command

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:08</td><td>RO</td><td>0</td><td>Reserved.</td></tr>
<tr><td>07:00</td><td>RW</td><td>0h</td><td>Command Opcode (CMDOP): Indicate the Opcode of a UIC Command to be dispatched to local UIC layer. When this register is set, the host controller shall take the values of UICCMDARGx as the corresponding parameters (input and output) that are a part of the UIC Command.
<table>
<tr><th>Opcode</th><th>O/M</th><th>UIC Command</th></tr>
<tr><td colspan="3">Configuration</td></tr>
<tr><td>01h</td><td>M</td><td>DME_GET</td></tr>
<tr><td>02h</td><td>M</td><td>DME_SET</td></tr>
<tr><td>03h</td><td>M</td><td>DME_PEER_GET</td></tr>
<tr><td>04h</td><td>M</td><td>DME_PEER_SET</td></tr>
<tr><td>05h – 0Fh</td><td></td><td>Reserved</td></tr>
<tr><td colspan="3">Control</td></tr>
<tr><td>10h</td><td>O</td><td>DME_POWERON</td></tr>
<tr><td>11h</td><td>O</td><td>DME_POWEROFF</td></tr>
<tr><td>12h</td><td>M</td><td>DME_ENABLE</td></tr>
<tr><td>13h</td><td></td><td>Reserved</td></tr>
<tr><td>14h</td><td>M</td><td>DME_RESET</td></tr>
<tr><td>15h</td><td>M</td><td>DME_ENDPOINTRESET</td></tr>
<tr><td>16h</td><td>M</td><td>DME_LINKSTARTUP</td></tr>
<tr><td>17h</td><td>M</td><td>DME_HIBERNATE_ENTER</td></tr>
<tr><td>18h</td><td>M</td><td>DME_HIBERNATE_EXIT</td></tr>
<tr><td>19h</td><td></td><td>Reserved</td></tr>
<tr><td>1Ah</td><td>O</td><td>DME_TEST_MODE</td></tr>
<tr><td>1Bh - FFh</td><td></td><td>Reserved</td></tr>
</table>
O/M: O = Optional, M = Mandatory.</td></tr>
</table>

5.6.2 Offset 94h: UICCMDARG1 – UIC Command Argument 1

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:00</td><td>RW</td><td>0</td><td>
Argument 1 (ARG1): This register contains the value for 1st argument of the UIC command if applicable. The content of this field varies with the UIC Command (UICCMD).
<table>
<tr><th rowspan="2">UIC Command</th><th colspan="4">Value</th></tr>
<tr><th>Bit[31:24]</th><th>Bit[23:16]</th><th>Bit[15:08]</th><th>Bit[07:00]</th></tr>
<tr><td>DME_GET</td><td colspan="2">MIBattribute</td><td colspan="2">GenSelectorIndex</td></tr>
<tr><td>DME_SET</td><td colspan="2">MIBattribute</td><td colspan="2">GenSelectorIndex</td></tr>
<tr><td>DME_PEER_GET</td><td colspan="2">MIBattribute</td><td colspan="2">GenSelectorIndex</td></tr>
<tr><td>DME_PEER_SET</td><td colspan="2">MIBattribute</td><td colspan="2">GenSelectorIndex</td></tr>
<tr><td>DME_POWERON</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_POWEROFF</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_ENABLE</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_RESET</td><td colspan="2">Reserved</td><td>ResetMode</td><td>ResetLevel</td></tr>
<tr><td>DME_ENDPOINTRESET</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_LINKSTARTUP</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_HIBERNATE_ENTER</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_HIBERNATE_EXIT</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_TEST_MODE</td><td colspan="4">Reserved</td></tr>
</table>
</td></tr>
</table>

MIBattribute: Indicates the ID of the attribute of the requested. See MIPI UniPro Specification for the details of the MIBattribute parameter.

GenSelectorIndex: Indicates the targeted M-PHY data lane or CPort or Test Feature when relevant. See MIPI UniPro Specification for the details of the GenSelectorIndex parameter.

Layer	Valid Range
L1	0 to 2*PA_MaxDataLanes – 1
L4 / CPort	0 to T_NumCPorts – 1
L4 / Test Feature	0 to T_NumTestFeatures – 1

ResetMode: Indicates the link startup mode. See MIPI UniPro Specification for the details of the ResetMode parameter.

Value	Definition
00h	LS Mode
01h	HS Mode
02h-FFh	Reserved

ResetLevel: Indicates the reset type. See MIPI UniPro Specification for the details of the ResetLevel parameter.

Value	Definition
00h	Cold Reset
01h	Warm Reset
02h-FFh	Reserved

5.6.3 Offset 98h: UICCMDARG2 – UIC Command Argument 2

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:00</td><td>RW</td><td>0</td><td>Argument 2 (ARG2): This register contains the value for 2nd argument of the UIC command if applicable. The content of this field vary with UIC Command.
<table>
<tr><th rowspan="2">UIC Command</th><th colspan="4">Value</th></tr>
<tr><th>Bit[31:24]</th><th>Bit[23:16]</th><th>Bit[15:08]</th><th>Bit[07:00]</th></tr>
<tr><td>DME_GET</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>ConfigResultCode</td></tr>
<tr><td>DME_SET</td><td>Reserved</td><td>AttrSetType</td><td>Reserved</td><td>ConfigResultCode</td></tr>
<tr><td>DME_PEER_GET</td><td>Reserved</td><td>Reserved</td><td>Reserved</td><td>ConfigResultCode</td></tr>
<tr><td>DME_PEER_SET</td><td>Reserved</td><td>AttrSetType</td><td>Reserved</td><td>ConfigResultCode</td></tr>
<tr><td>DME_POWERON</td><td colspan="3">Reserved</td><td>GenericErrorCode</td></tr>
<tr><td>DME_POWEROFF</td><td colspan="3">Reserved</td><td>GenericErrorCode</td></tr>
<tr><td>DME_ENABLE</td><td colspan="3">Reserved</td><td>GenericErrorCode</td></tr>
<tr><td>DME_RESET</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_ENDPOINTRESET</td><td colspan="3">Reserved</td><td>GenericErrorCode</td></tr>
<tr><td>DME_LINKSTARTUP</td><td colspan="3">Reserved</td><td>GenericErrorCode</td></tr>
<tr><td>DME_HIBERNATE_ENTER</td><td colspan="3">Reserved</td><td>GenericErrorCode</td></tr>
<tr><td>DME_HIBERNATE_EXIT</td><td colspan="3">Reserved</td><td>GenericErrorCode</td></tr>
<tr><td>DME_TEST_MODE</td><td colspan="3">Reserved</td><td>GenericErrorCode</td></tr>
</table></td></tr>
</table>

AttrSetType: Indicates whether the attribute value (AttrSet = NORMAL) or the attribute non-volatile reset value (STATIC) setting is requested. See MIPI UniPro Specification for the details of the AttrSetType parameter.

ConfigResultCode: Indicates the result of the UIC configuration command request. It is valid after host controller has set the **IS.UCCS** bit to '1'. See MIPI UniPro Specification for the details of the ConfigResultCode parameter.

Value	Definition
00h	SUCCESS
01h	INVALID_MIB_ATTRIBUTE
02h	INVALID_MIB_ATTRIBUTE_VALUE
03h	READ_ONLY_MIB_ATTRIBUTE
04h	WRITE_ONLY_MIB_ATTRIBUTE
05h	BAD_INDEX
06h	LOCKED_MIB_ATTRIBUTE
07h	BAD_TEST_FEATURE_INDEX
08h	PEER_COMMUNICATION_FAILURE
09h	BUSY
0Ah	DME_FAILURE
0Bh-FFh	Reserved

GenericErrorCode: Indicates the result of the UIC control command request. It is valid after host controller has set the **IS.UCCS** bit to '1'. See MIPI UniPro Specification for the details of the GenericErrorCode parameter.

Value	Definition
0h	SUCCESS
1h	FAILURE
2h-FFh	Reserved

5.6.4 Offset 9Ch: UICCMDARG3 – UIC Command Argument 3

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:00</td><td>RW</td><td>0</td><td>Argument 3 (ARG3): This register contains the value for 3rd argument of the UIC command if applicable. The content of this field vary with UIC Command.
<table>
<tr><th rowspan="2">UIC Command</th><th colspan="4">Value</th></tr>
<tr><th>Bit[31:24]</th><th>Bit[23:16]</th><th>Bit[15:08]</th><th>Bit[07:00]</th></tr>
<tr><td>DME_GET</td><td colspan="4">MIBvalue_R</td></tr>
<tr><td>DME_SET</td><td colspan="4">MIBvalue_W</td></tr>
<tr><td>DME_PEER_GET</td><td colspan="4">MIBvalue_R</td></tr>
<tr><td>DME_PEER_SET</td><td colspan="4">MIBvalue_W</td></tr>
<tr><td>DME_POWERON</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_POWEROFF</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_ENABLE</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_RESET</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_ENDPOINTRESET</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_LINKSTARTUP</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_HIBERNATE_ENTER</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_HIBERNATE_EXIT</td><td colspan="4">Reserved</td></tr>
<tr><td>DME_TEST_MODE</td><td colspan="4">Reserved</td></tr>
</table></td></tr>
</table>

MIBvalue_R: Indicates the value of the attribute as returned by the UIC command returned. It is valid after host controller has set the **IS.UCCS** bit to '1'. See MIPI UniPro Specification for the details of the MIBvalue parameter.

MIBvalue_W: Indicates the value of the attribute to be set. See MIPI UniPro Specification for details of the MIBvalue parameter.

5.6.5 Attributes for Local L2 Timers

The UniPro specification [MIPI-UniPro] defines DME attributes necessary for performing the UIC power mode change using the DME_SET primitives only. These attributes are accessible via DME_SET command as any other UIC-defined attributes. In this sub-clause, these attributes are collectively named DME LocalL2TimerData attributes.

The DME LocalL2TimerData and PA_PWRModeUserData attributes and their behavior are equivalent to the LocalL2TimerData and RemoteL2TimerData parameters in the DME_POWERMODE.req primitive, and define the local and remote Layer 2 timer values for the next UIC power mode, respectively. Neither the DME LocalL2TimerData attributes, nor the PA_PWRModeUserData attributes have an immediate effect on the UIC stack. They are solely used to store the local and remote L2 timer values for the next UIC power mode. These values are used by UIC during the UIC power mode change triggered by setting the PA_PWRMode attribute, and are only committed to UIC if the UIC power mode change is successful. If the UIC power mode change fails, the previous L2 timer values are still used.

As a result, using the DME LocalL2TimerData, PA_PWRModeUserData and PA_PWRMode attributes for changing the UIC power mode has an identical behavior to changing the UIC power mode using the DME_POWERMODE.req primitive. These attributes are a way to implement the DME_POWERMODE.req primitive. In case of the UFS application, it is mandatory to implement Annex H, as specified by [MIPI-UniPro].

5.7 Vendor Specific Registers

5.7.1 Offset C0h to FFh: VS – Vendor Specific

This block of registers is reserved for vendor specific.

5.8 Crypto Registers

5.8.1 Offset 100h: CCAP – Crypto Capability

The CCAP register provides information about the capabilities of the host controller's cryptographic hardware. This register is valid only in controllers supporting cryptographic operations (CAP.CS = 1). If CAP.CS = 0, this register is reserved.

Bit	Type	Reset	Description
31:24	RO	Impl Spec	**Configuration Array Pointer (CFGPTR):** An offset pointer to the base of the Configuration Array (x-CRYPTOCFG registers), in 256 B units. CFGPTR value shall be larger than 04h, so that it does not conflict with the x-CRYPTOCAP array The address for entry *x* of the x-CRYPTOCFG array is calculated as follows: ADDR (x-CRYPTOCFG) = *UFS_HCI_BASE* + *CFGPTR**100h + *x**80h
23:16	RO	0	Reserved.
15:08	RO	Impl Spec	**Configuration Count (CFGC):** The maximum number of configurations supported by the host controller. The actual number of configurations is equal to (CFGC+1). The minimum number of configurations supported is 1 (CFGC = 00h). The maximum number of configurations supported is 256 (CFGC = FFh).
07:07	RO	0	Reserved
06:00	RO	Impl Spec	**Crypto Capabilities (CC):** The number of crypto capabilities that the host controller provides. The values allowed are between 1 and 127. 1: 1 Capability 2: 2 Capabilities … 127: 127 Capabilities

5.8.2 x-CRYPTOCAP – Crypto Capability X

A Crypto Capability defines a set of properties associated with a crypto algorithm.

Crypto Capabilities are organized as a continuous register array, starting at offset 104h. Each entry of the x-CRYPTOCAP array provides information of one Crypto Capability.

The entry for Crypto Capability *i* is located in offset 104h+*i**4h from the UFS HCI base address. Crypto Capability #0 shall be located in offset 104h. Crypto Capability #126 (if implemented) shall be located in offset 2FCh.

When the host controller implements a number of Crypto Capabilities as declared in CCAP.CC field, these Crypto Capabilities shall be organized in entries 0 through *CC*-1 of x-CRYPTOCAP array. Entries beyond CCAP.CC-1 are not valid and shall be ignored by software.

Each entry of the x-CRYPTOCAP array shall have the following register mapping.

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:24</td><td>RO</td><td>0</td><td>Reserved.</td></tr>
<tr><td>23:16</td><td>RO</td><td>Impl Spec</td><td>Key Size (KS): Specifies Key Size in bits used by this algorithm if ALGID represents an encryption algorithm. Ignored if ALGID selects a hashing algorithm.
<table>
<tr><th>KS</th><th>Key Size in Bits</th></tr>
<tr><td>00h</td><td>Reserved</td></tr>
<tr><td>01h</td><td>128 bits</td></tr>
<tr><td>02h</td><td>192 bits</td></tr>
<tr><td>03h</td><td>256 bits</td></tr>
<tr><td>04h</td><td>512 bits</td></tr>
<tr><td>05h-FFh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>15:08</td><td>RO</td><td>Impl Spec</td><td>Supported Data Unit Size Bitmask (SDUSB): Specifies the data unit sizes supported by the capability, in bitmask encoding.
When bit j in this field (j = 0..7) is set, data unit size of 512*2^j bytes is supported.
Bit 0 indicates 512B, bit 1 indicates 1 KB, ..., bit 7 indicates 64 KB.
One or more bits in this field may be set. For example, if sizes 1 KB, 4 KB, and 16 KB are supported by the capability, then SDUSB = 00101010b (= 2Ah).</td></tr>
<tr><td>07:00</td><td>RO</td><td>Impl Spec</td><td>Algorithm ID (ALGID): The identification code of the crypto algorithm according to the following table.
<table>
<tr><th>ID code</th><th>Algorithm</th></tr>
<tr><td>00h</td><td>AES-XTS</td></tr>
<tr><td>01h</td><td>Microsoft Bitlocker™ AES-CBC</td></tr>
<tr><td>02h</td><td>AES-ECB</td></tr>
<tr><td>03h</td><td>ESSIV-AES-CBC</td></tr>
<tr><td>04h</td><td>SHA-256 [SHA]</td></tr>
<tr><td>05h</td><td>SHA-512 [SHA]</td></tr>
<tr><td>06h-7Fh</td><td>Reserved</td></tr>
<tr><td>80h-FFh</td><td>Vendor specific</td></tr>
</table></td></tr>
</table>

5.8.3 x-CRYPTOCFG – Crypto Configuration X

Crypto Configurations enable host software to instantiate a Crypto Capability with a key. Crypto Configurations are organized as a continuous register array, starting at the offset declared in CCAP.CFGPTR. Each entry of the x-CRYPTOCFG array contains information of one Crypto Configuration. The size of each entry is 128 Bytes (1024 b).

The entry for Crypto Configuration *x* is located in address *UFS_HCI_BASE* + CCAP.*CFGPTR**100h + *x**80h. Crypto Configuration #0 shall be located in address *UFS_HCI_BASE* + CCAP.*CFGPTR**100h.

When the host controller implements a number of Crypto Configurations as declared in CCAP.CFGC field, these Crypto Configurations shall be organized in entries 0 through *CFGC* of x-CRYPTOCFG array. Entries beyond *CFGC* are not valid and shall be ignored by software.

The layout of each entry in the x-CRYPTOCFG array is shown in Figure 9. All entries in x-CRYPTOCFG array shall have the following register mapping.

Bit	Type	Reset	Description
1023:576	RO	0	Reserved.
575:560	RW	Impl Spec	**Vendor-Specific Bits (VSB):** This field is used by software to enable host-specific features associated with the Crypto Configuration.
559:552	RO	0	Reserved.
551:544	RO	0	**Reserved for Multi-Host Related Functions**
543	RW	0	**Configuration Enable (CFGE):** This field is used by software to enable/disable a Crypto Configuration usage • 0b – Configuration Disabled. Transactions using this Crypto Configuration (UTRD.CCI field) shall be terminated with error by host controller (OCS = INVALID_CRYPTO_CONFIG) • 1b – Configuration Enabled. Transactions using this Crypto Configuration (UTRD.CCI) field can be executed
542:528	RO	0	Reserved.
527:520	RW	0	**Crypto Capability Index (CAPIDX):** Specifies the index of the Crypto Capability to be used for this configuration. Values allowed are between 0 and CCAP.CC-1
519:512	RW	0	**Data Unit Size (DUSIZE):** Size of data unit used with this configuration, encoded in one-hot encoding, analogous to bitmask used in CRYPTOCAP.SDUSB field. When bit *j* in this field (*j* = 0..7) is set, a data unit size of $512*2^j$ bytes is selected. Bit *j* may be set only if the same bit is also set in the SDUSB field of the capability referenced in CAPIDX field.
511:000	WO	Impl Spec	**Crypto Key (CRYPTOKEY):** Specifies the key to be used for this configuration. The specific key layout is defined according to the Key Size and Algorithm specified in the Crypto Capability with index value specified in CAPIDX. When configuring CRYPTOKEY field software shall write the entire key from DW0 to DW15, sequentially, in one atomic set of operations. The unused regions of CRYPTOKEY according to the selected Key Size and Algorithm shall be written with zeros by Software.

5.8.3 x-CRYPTOCFG – Crypto Configuration X (cont'd)

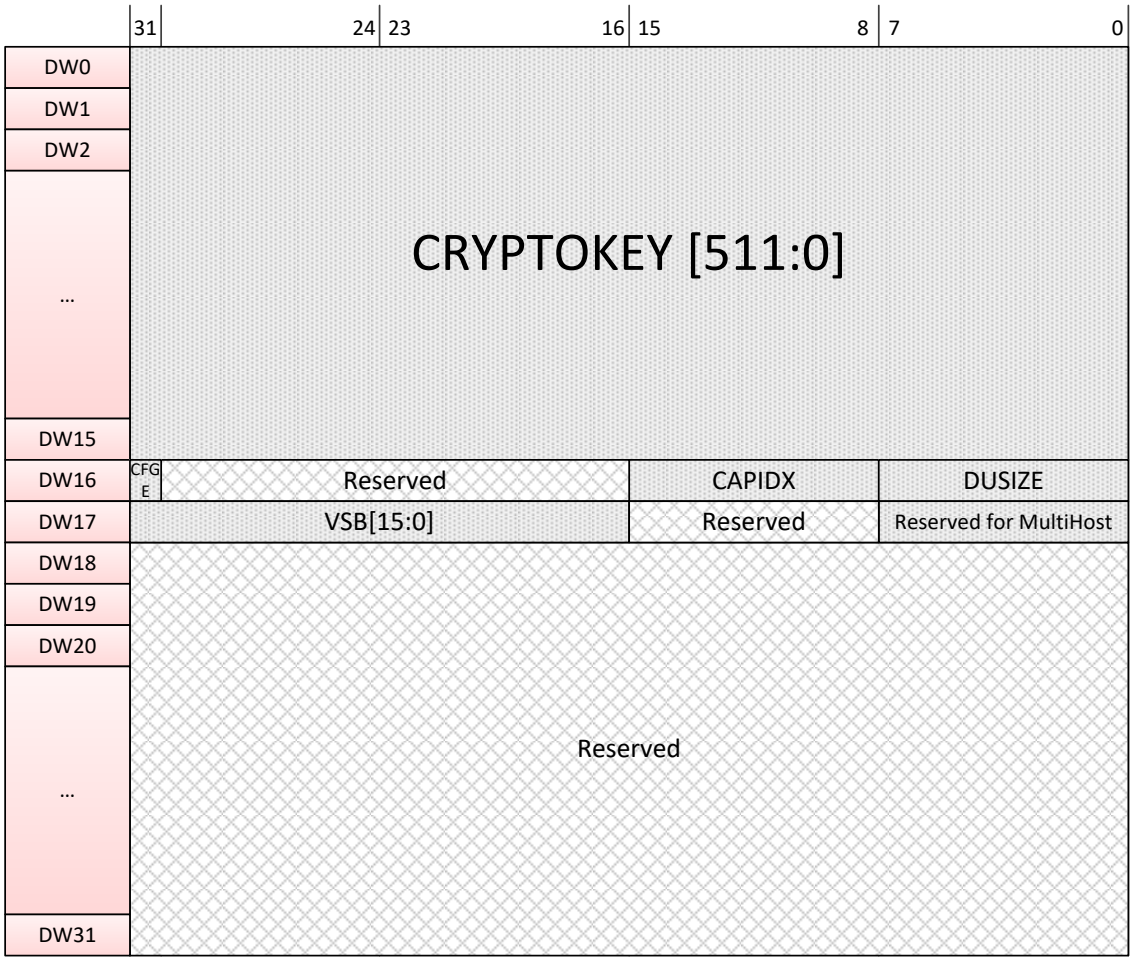


Figure 9 — x-CRYPTOCFG Array Entry Layout

5.9 Config Register

5.9.1 Offset 300h – Global Config Register

This register provides the Global configuration. Host software is expected to ensure that there is no pending task before setting Global Config Registers.

Bit	Type	Reset	Description
31:3	RO	0000h	**Reserved**
2	RW	0h	**2DW PRDT Format Enable (2DWPRDTEN).** In OoO mode, usage of 2DW format of PRDT entry is mandatory. When set to 1, indicates that 2DW entry format is being used. When set to 0, indicates that legacy 4DW entry format is being used. NOTE 1 • When Host SW enables OoO in the device in any direction, it must use 2DW mode and set Config.2DWPRDTEN to 1. • When Host SW does not enable OoO in any direction, SW has freedom to use 2DW or 4DW format and program Config.2DWPRDTEN accordingly. NOTE 2 • When 2DWPRDTEN = 0, the LDBC and CDS fields of the UTP Transfer Request Descriptor (UTRD) are ignored. The DBC fields of the PRDT entry is used. This retains legacy operation. • When 2DWPRDTEN = 1, the LDBC and CDS fields of the UTP Transfer Request Descriptor must be populated. There is no DBC fields in 2DW PRDT entry format.
1	RW	0h	**Event Specific Interrupt Enable (ESIE).** Enables Event Specific Interrupt • 0h – Event Specific Interrupt is disabled • 1h - Event Specific Interrupt is enabled
0	RW	0h	**Queue type (QT).** Selects the Queue mode for Host controller operation • 00h – Legacy Single Doorbell (UTRLDBR Doorbell Mode) • 01h – Select Multi-Circular, Multi Doorbell Queue Mode

5.9.2 Offset 380h – MCQ Config Register (MCQConfig)

This register provides the Multi-Circular Queue configuration. Arbitration scheme is defined for all active or valid SQs.

Bit	Type	Reset	Description
31:17	RO	0000h	**Reserved**
16:8	RW	1Fh	**MaxActiveCommand (MAC).** The Host Controller shall not send more than the maximum number of active commands to the device at any time. The Host Controller may have a separate internal limitation independent of this value. Host SW is responsible for setting this value after discovering the device queue depth capability. The default value is 32 commands, allowing operation prior to Host SW initialization. This is a zero-based value.
7:2		00h	**Reserved**
1:0	RW	0h	**Arbitration scheme (AS).** Select the arbitration scheme to be use for the Multi-Circular Queue. • 00h – Round Robin priority arbitration • 01h – Strict priority arbitration • Others - Reserved

5.9.3 Interrupt Topology

Host controller can support interrupting the processor (or multiple processor cores) over either physical interrupt signal lines, or through event specific interrupt (ESI) topology.

This specification provides Interrupt Status (IS) and Interrupt Enable (IE) registers for single processing core, as well as SQ/CQ based Interrupt Status (SQISy/CQISy) and Interrupt Enable (SQIEy/CQIEy) registers for interrupting multiple processing cores. In this kind of interrupting, after getting interrupted, processing core needs to read the interrupt status register (IS/SQISy/CQISy) to determine the reason for interruption.

In ESI, interrupts are communicated to a central interrupt controller via event indexes and the interrupt is routed and distributed to processing cores by the interrupt controller. Based on the architecture (ARM, x86, etc.) of the system where UFS HW Host controller is being deployed, there are multiple interrupt controller architecture available with multiple interrupt distribution schemes and these are out of scope for this specification. However, all of them is based on a message being written into a register inside interrupt controller. The message provides all information for the interrupt controller to distribute multiple events to multiple processors.

This specification uses a generic event vector-based approach as described below. It is up to the Host controller to map this scheme to specific interrupt distribution scheme that the system employs.

System provides an address and a base Event vector to the Host controller. Whenever an Event needs to be raised, Host controller adds the base Event vector up to the Event vector to be raised as the interrupt vector and writes the 32-bits interrupt vector to that address. This 32-bits interrupt vector encodes multiple events, mapped to multiple simultaneous interrupts, to be distributed to multiple processing cores.

5.9.3 Interrupt Topology (cont'd)

Since this vector is interpretable by the interrupt controller of the system, which UFS Host HW controller is part of, the encoding is out of scope of this specification.

For MCQ, ESI is recommended. However, a system can use traditional interrupt approach if it does not have message-based interrupt routing and distribution capability.

Legacy Single Doorbell, that currently uses traditional interrupt approach, may use ESI too. This is implementation specific.

5.9.3.1 Offset 380h+04h – Event Specific Interrupt Lower Base Address (ESILBA)

Bit	Type	Reset	Description
31:00	RW	Impl Spec	Specifies the lower 32-bits of the system address where Interrupt vector shall be written when ESI is enabled.

5.9.3.2 Offset 380h+08h – Event Specific Interrupt Upper Base Address (ESIUBA)

Bit	Type	Reset	Description
31:00	RW	Impl Spec	Specifies the upper 32-bits of the system address where Interrupt vector shall be written when ESI is enabled.

5.9.3.3 Offset 380h+0Ch – Event Specific Interrupt Vector Base (ESIVB)

Bit	Type	Reset	Description
31:00	RW	Impl Spec	Specifies the Event vector base, i.e., ESIVB is added up to Event vector to be raised when ESI is enabled.

5.9.4 Submission Queues (SQ) Configuration Registers

The following registers are per queue definition, including queue size, priority, completion queue mapping to submission queue, base address, address offset (relative to HCI base address) of head and tail pointer

8DWs are reserved for defining each submission Queue. Hence, a total 1024 B address space is reserved for defining 32 submission queues. A controller implements registers for only MCQCap.MAXQ number of Submission Queues. Other addresses will remain reserved and will return value DEADBEEFh when read.

The Submission queue number (y) is its own ID.

5.9.4.1 Offset MCQCAP.QCFGPTR*200h+40h*y – Submission Queue y Attributes (SQATTRy)

Bit	Type	Reset	Description
31:31	RW	Impl Spec	**Submission Queue y Enable (SQEN).** Enable/Disable control for Submission Queue y, why y = 0..31. Setting a 1 will enable Submission Queue y for use in MCQ mode, while setting a 0 will disable Submission Queue y for use in MCQ mode. NOTE 1 Host SW is expected to ensure that this SQ is empty before disabling this SQ. Once the Host SW disables this SQ, the host controller will not be responsible for any commands pending in this SQ. NOTE 2 Setting this bit of SQy to 0 may not ensure immediate disabling of SQy fetching. This behavior is implementation specific. If Host HW controller needs time to disable this SQy, it shall ensure that SQEN value becomes 0 only after SQy gets disabled. After setting this bit to 0, Host SW driver shall poll this bit till the value actually reflects value 0.
30:28	RW	Impl Spec	**Priority level for the Queue (SQPL):** 0 to 7 is valid priority value, with lower value having higher priority.
27:24	RW	0000h	Reserved
23:16	RW	Impl Spec	**Completion Queue ID (CQID):** Indicates completion Queue ID that is mapped to this Submission Queue SW shall ensure this CQ ID value is valid, and the CQ is adequately configured and enabled. Otherwise, Host controller's behavior is undefined.
15:0	RW	Impl Spec	**Submission Queue Size (SIZE):** Specifies the depth of this submission queue in terms of DWords Each SQ entry is 8 Dwords in size. Each SQ is composed of slots which may each contain a queue entry. **SIZE** is a zero-based value constructed to create a count of slots. Therefore, **SIZE** shall indicate a multiple of 8 Dwords, equal to the following formula: $$\textbf{SIZE} = (SlotCount \times 8) - 1$$ Example values for **SIZE**: **15: 16 Dwords (2 queue slots)** **23: 24 Dwords (3 queue slots)** **…** **65535: 65536 Dwords (8192 queue slots)** If **SIZE** is less than 15, then queue entries cannot be pushed into the SQ and host sofware shall disable the queue by setting SQEN = 0.

5.9.4.2 Offset MCQCAP.QCFGPTR*200h+40h*y+04h – Submission Queue y Lower Base Address (SQLBAy)

Bit	Type	Reset	Description
31:10	RW	Impl Spec	**Submission Queue y Lower Base Address (SQLBAy):** Indicates the lower 32-bit base physical address for the Submission Queue y . This base is used when fetching commands for execution. This address shall be 1 KB aligned, this field is multiplied by 1024.
09:00	RO	0	Reserved

5.9.4.3 Offset MCQCAP.QCFGPTR*200h+40h*y+08h – Submission Queue y Upper Base Address (SQUBAy)

Bit	Type	Reset	Description
31:00	RW	Impl Spec	**Submission Queue y Upper Base Address (SQUBAy):** Indicates the upper 32-bits for the Submission Queue y base physical address. This base is used when fetching commands for execution.

5.9.4.4 Offset MCQCAP.QCFGPTR*200h + 40h*y + 0Ch – Submission Queue y Doorbell Address Offset (SQDAOy)

Bit	Type	Reset	Description
31:0	RW Or RO (Impl Spec)	Impl Spec	Specifies the address offset to the MCQ Operation & Runtime Head and Tail Doorbell Pointer registers for this submission queue. Offset is from the base address of the HCI. The SQ Head Pointer (SQHPy) is located in address *UFS_HCI_BASE + SQDAOy*. The SQ Tail Pointer (SQTPy) is located in address *UFS_HCI_BASE + SQDAOy + 4*. NOTE The type of this register is implementation specific. Refer to the data sheet.

5.9.4.5 Offset MCQCAP.QCFGPTR*200h + 40h*y + 10h – Submission Queue y Interrupt Status Register Address Offset (SQISAOy)

Bit	Type	Reset	Description
31:0	RW Or RO (Impl Spec)	Impl Spec	Specifies the address offset to the MCQ Interrupt registers for this submission queue. Offset is from the base address of the HCI. NOTE The type of this register is implementation specific. Refer to the data sheet.

5.9.4.6 Offset MCQCAP.QCFGPTR*200h + 40h*y + 14h – Submission Queue y Configuration Register (SQCFGy)

Bit	Type	Reset	Description
31:09	RO	0	Reserved

08:08	RW	0	**Interrupt Aggregation Group Valid (IAGVLD):** Association with the interrupt aggregation group ID is valid.
			NOTE:
			• If 0, events from this SQ is not subjected to interrupt aggregation.
			• If 1,
			  ○ If associated IAG is disabled, events from this SQ is not subjected to interrupt aggregation.
			  ○ If associated IAG is enabled, events from this SQ is subjected to configuration of the IAG

07:05	RO	0	Reserved
04:00	RW	0	**Interrupt Aggregation Group (IAG):** The interrupt aggregation group ID associated with this SQ.

5.9.4.7 Offset MCQCAP.QCFGPTR*200h+40h*y + 18h

Bit	Type	Reset	Description
31:0	RO	0	Reserved

5.9.4.8 Offset MCQCAP.QCFGPTR*200h+40h*y + 1Ch

Bit	Type	Reset	Description
31:0	RO	0	Reserved

5.9.5 Completion Queues (CQ) Configuration Registers

The following registers are per queue definition, including queue size, base address, address offset (relative to HCI Base address) of head & tail pointer

8DWs are reserved for defining each completion Queue. Hence, total 1024B address space is reserved for defining 32 completion queues. A controller implements registers for only MCQCap.MAXQ number of Completion Queues. Other addresses will remain reserved and will return value DEADBEEFh when read.

The Completion queue number (y) is its own ID.

5.9.5.1 Offset MCQCAP.QCFGPTR*200h + 40h*y + 20h – CQ y Attribute (CQATTRy) (y = Completion Q number

Bit	Type	Reset	Description
31:31	RW	Impl Spec	**Completion Queue y Enable (CQEN):** Enable/Disable control for Completion Queue y, where y = 0..31. Setting a 1 will enable Completion Queue y for use in MCQ mode, while setting a 0 will disable Completion Queue y for use in MCQ mode. NOTE 1 Host SW is expected to ensure that this CQ is empty before disabling this CQ. Once the Host SW disables this CQ, the host controller will not be responsible for any CQ entry pending in this CQ. NOTE 2 Setting this bit of CQy to 0 may not ensure immediate disabling of push to CQy. This behavior is implementation specific. If Host HW controller needs time to disable this CQy, it shall ensure that CQEN value becomes 0 only after push to CQy gets disabled. After setting this bit to 0, Host SW driver shall poll this bit till the value actually reflects value 0.
30:16	RW	0000h	**Reserved**
15:0	RW	Impl Spec	**Completion Queue Size (SIZE):** Specifies the depth of this completion queue in terms of DWords. Each CQ entry is 8 Dwords in size. Each CQ is composed of slots which may each contain a queue entry. **SIZE** is a zero-based value constructed to create the a count of slots. Therefore, **SIZE** shall indicate a multiple of 8 DWords, equal to the following formula: $\mathbf{SIZE} = (SlotCount \times 8) - 1$ Example values for **SIZE**: **15: 16 Dwords (2 queue slots)** **23: 24 Dwords (3 queue slots)** **…** **65535: 65536 Dwords (8192 queue slots)** If **SIZE** is less than 15, then queue entries cannot be pushed into the CQ and host sofware shall disable the queue by setting CQEN = 0.

5.9.5.2 Offset MCQCAP.QCFGPTR*200h + 40h*y + 24h – CQ y Lower Base Address (CQLBAy)

Bit	Type	Reset	Description
31:10	RW	Impl Spec	**Completion Queue y Lower Base Address (CQLBAy):** Indicates the lower 32-bit base physical address for the Submission Queue y . This base is used when fetching commands for execution. This address shall be 1 KB aligned, this field is multiplied by 1024.
09:00	RO	0	Reserved

5.9.5.3 Offset MCQCAP.QCFGPTR*200h + 40h*y + 28h – CQ y Upper Base Address (CQUBAy)

Bit	Type	Reset	Description
31:00	RW	Impl Spec	**Completion Queue y Upper Base Address (CQUBAy):** Indicates the upper 32-bits for the Completion Queue y base physical address. This base is used when fetching commands for execution.

5.9.5.4 Offset MCQCAP.QCFGPTR*200h + 40h*y + 2Ch – CQ y Doorbell Address Offset (CQDAOy)

Bit	Type	Reset	Description
31:0	RW Or RO (Impl Spec)	Impl Spec	**Completion Queue y Doorbell Address Offset (CQDAOy):** Specifies the address offset to the MCQ Operation & Runtime Head and Tail Doorbell Pointer registers for this completion queue. Offset is from the base address of the HCI. The CQ Head Pointer (CQHPy) is located in address *UFS_HCI_BASE + CQDAOy*. The CQ Tail Pointer (CQTPy) is located in address *UFS_HCI_BASE + CQDAOy + 4.* NOTE the type of this register is implementation specific. Refer to the data sheet.

5.9.5.5 Offset MCQCAP.QCFGPTR*200h + 40h*y + 30h - CQ y Interrupt Status Register Address Offset (CQISAOy)

Bit	Type	Reset	Description
31:0	RW Or RO (Impl Spec)	Impl Spec	**Completion Queue y Interrupt Status Register Address Offset (CQISAOy):** Specifies the address offset to the MCQ Interrupt registers for this completion queue. Offset is from the base address of the HCI. NOTE The type of this register is implementation specific. Refer to the data sheet.

5.9.5.6 Offset MCQCAP.QCFGPTR*200h + 40h*y + 34h - Completion Queue y Configuration Register (CQCFGy)

Bit	Type	Reset	Description
31:09	RO	0	Reserved
08:08	RW	0	**Interrupt Aggregation Group Valid (IAGVLD):** Association with the interrupt aggregation group ID is valid. NOTE: • If 0, events from this CQ is not subjected to interrupt aggregation. • If 1, o If associated IAG is disabled, events from this CQ is not subjected to interrupt aggregation. o If associated IAG is enabled, events from this CQ is subjected to configuration of the IAG.
07:05	RO	0	Reserved
04:00	RW	0	**Interrupt Aggregation Group (IAG):** The interrupt aggregation group ID associated with this CQ.

5.9.5.7 Offset MCQCAP.QCFGPTR*200h + + 40h*y + 38h

Bit	Type	Reset	Description
31:0	RO	0	Reserved

5.9.5.8 Offset MCQCAP.QCFGPTR*200h + 40h*y + 3Ch)

Bit	Type	Reset	Description
31:0	RO	0	Reserved

5.9.6 Operation and Runtime Registers - Submission Queues and Completion Queues

The following registers are accessed during operation mode by the processor core that own the specific submission queue. Note that addresses of these registers are not fixed relative to base address of HCI. Address of these registers shall be configured (through SQDAO, CQDAO, SQISAO, CQISAO) at Init Phase by Admin processor, to position these registers in the memory mapped address range in which the owner processor core has access permission.

To support variety of topology, the specification allows associated SQ and CQs to be owned by different processing cores, which means that, for the same Transfer Request two different processor core might need to be interrupted. The interrupting status registers for each processing cores might be positioned in the memory address region owned by those processing cores.

If any fatal or error condition is encountered by HW controller that requires admin level action (for example, resetting the HW controller), the main Interrupt Status (IS) Register shall be used to interrupt the Admin processor core. The communication between admin processing core to other participating processing cores for any error or fatal condition is out of scope of this HCI.

y = 0..31

5.9.6.1 Offset SQDAOy – SQ y Head Pointer (SQHPy)

Bit	Type	Reset	Description
31:18	RO	0	Reserved
17:0	RW	Impl Spec	**Submission Queue y Head Pointer:** This register holds the 32-bits offset of system memory mapped address of the head entry of this Submission Queue, relative to {SQUBAy, SQLBAy}. The content of this register is primarily updated by Host HW controller when it fetches an entry from this Submission Queue, and read by Host Driver SW either by polling or upon interrupt, to determine amount of free space available in this Submission queue. It is read only by Host SW when Queue enable bit in SQATTR is “1”. The pointer is DWord aligned, bit 0-1 are zero. The size of SQ is specified by SQATTRy.SIZE, which is 16 bits wide and in terms of DW. Hence this field is 18 bits wide.

5.9.6.2 Offset SQDAOy + 04h – SQ y Tail Pointer (SQTPy)

Bit	Type		Reset	Description
31:18	RO		0	Reserved
17:0	RW		Impl Spec	**Submission Queue y Tail Pointer:** This register holds the 32-bits offset of system memory mapped address of the Tail entry of this Submission Queue, relative to {SQUBAy, SQLBAy}. The content of this register is primarily updated by Host Driver SW when a new entry is pushed into this Submission queue and read by Host HW controller to determine if this Submission queue is empty. The pointer is DWord aligned, bit 0-1 are zero. The size of SQ is specified by SQATTRy.SIZE, which is 16 bits wide and in terms of DW. Hence this field is 18 bits wide. NOTE A write operation to tail pointer indicates that SW has enqueued new SQ entry. Host starts calculating number of SQ entries only when Tail pointer is written by SW driver. Hence, Host driver shall make write operation to the tail pointer register of SQy whenever it wants to conveny HW controller about the newly added entry/entries in SQy.

5.9.6.3 Offset SQDAOy + 08h – SQ y Run Time Command (SQRTCy)

SQRTCy and SQRTSy are used for submission queue specific management by owner process of the submission queue. Currently 2 such management functions are defined.

Similar to Legacy Single Doorbell run/stop function that is controlled by UTRLRSR, Running/Stopping of a submission queue is done by SQRTCy.STOP. The status of whether a SQ is running or stopped is available in SQRTSy.STS.

Cleanup of host HW resources for an aborted TR task of a SQ is initiated by SQRTCy.ICU. The nexus of the task {Initiator ID, LUN, Task Tag} is specified in SQCTIy. The status of the HW cleanup is available in SQRTSy.CUS and SQRTSy.RTC. The host controller shall free up any resources associated to the nexus, and shall post a Completion Queue entry with OCS = ABORTED if clean up successful.

NOTE The Initiator ID is comprised of two fields in the UPIU, IID as the least significant nibble and IID_EXT as the most significant nibble.

5.9.6.3 Offset SQDAOy + 08h – SQ y Run Time Command (SQRTCy) (cont'd)

Bit	Type	Reset	Description
31:2	R	0	**Reserved**
1	RW	0	**Initiate Clean Up (ICU) :** By writing '1' to this field, SW initiates the host controller HW resource cleanup associated with a task with nexus {Initiator ID-LUN-Task Tag } in SQCTIy register. When SW sets this bit to '1', host controller shall immediately clear SQRTSy.CUS and SQRTSy.RTC, before initiating the cleanup process. Host controller shall self-clear this field to '0' when all resources for the corresponding task is cleaned up. The status of the cleanup and return code are available in SQRTSy.CUS and SQRTSy.RTC respectively. NOTE The Initiator ID is comprised of two fields in the UPIU, IID as the least significant nibble and IID_EXT as the most significant nibble.
0	RW	0	**Submission Queue Stop (STOP):** Host software sets this bit as '1' to stop the SQ fetch, and clears to '0' to resume fetching of the SQ entry. When this value is changed from '0' to '1' by host software, the host controller stops the fetching entries from Submission Queue. When host controller successfully stops SQ fetching, it shall set SQISy.STS to '1' to indicate that SQ fetching is stopped and shall raise interrupt for this SQ if SQIEy.SIE is set. Host contrtoller will continue to execute all tasks that had been de-queued from the SQ before stopping it. When this value is changed from '1' to '0' by host software, the host controller shall resume fetching entries from Submission Queue. When host controller successfully resumes SQ fetching, it shall set SQISy.STS to '0' to indicate that SQ fetching is resumed and shall raise interrupt for this SQ if SQIEy.SIE is set.

5.9.6.4 Offset SQDAOy + 0Ch – SQ y Cleanup Task Information (SQCTIy)

Bit	Type	Reset	Description
31:24	R	0	Reserved
23:20	RW	0	EXT_IID
19:16	RW	0	IID
15:8	RW	0	LUN
7:0	RW	0	Task Tag

5.9.6.5 Offset SQDAOy + 10h – SQ y Run Time Status (SQRTSy)

Bit	Type	Reset	Description
31:8	RO	0	Reserved
7:4	RO	0	**CleanUp Command Return Code (RTC)** : Host controller sets this return code to provide more details of the cleanup process. It is valid only when CUS is 1. 0 : Success 1 : Fail – Task Not found 2 : Fail – SQ not stopped 3 : Fail – SQ is disabled Others : Reserved
3:2	RO	0	Reserved
1	RO	0h	**SQ CleanUp Status (CUS)** : Indicates whether HW cleanup process has been completed or not. 0: The resource clean is not completed yet. 1: The resource clean completed. This field is useful for polling and is equivalent to SQISy.SUS Note that the final result of the cleanup process is available in RTC field.
0	RO	0h	**SQ Stop Status (STS) :** Indicates whether the fetching of SQ entry is stopped or running. 0: Running the fetching of the SQ Entry 1: Stopped the fetching of the SQ Entry

5.9.6.6 Offset SQISAOy – SQ y Interrupt Status (SQISy)

This register indicates pending interrupts for this SQ that requires service.

Bit	Type	Reset	Description
31:4	RO	0h	Reserved
3	RWC	0h	**CQ Disable Status (CDS) :** Indicates that mapped CQ is not enabled and SQ is enabled. CQ Entry can not be posted.
2	RWC	0h	**SQ CleanUp Status (SUS)** : Indicates that HW resource clean up command is complete. Upon this interrupt, ISR should check SQRTSy.RTC for final results.
1	RWC	0h	**SQ Stop Status (SSS) :** Indicates that the SQ entry stopping/running command is complete. Upon this interrupt, ISR should check SQRTSy.STS for more details.
0	RWC	0	**Head Entry Fetch Status (HEFS):** Indicates that the Head entry of the SQy has been fetched by Host controller. Upon receiving this interrupt, Host SW reads SQHPy to determine amount of free space in the SQy. The following events are used for (1) setting this bit to '1' irrespective of IAG association of this queue, and (2) towards counting in associated IAG if IAG is active for this queue: Head Entry of this queue is fetched by host controller

5.9.6.7 Offset SQISAOy + 04h – SQ y Interrupt Enable (SQIEy)

This register enables and disables the reporting of the corresponding interrupt to host software. When a bit is set (‘1’) and the corresponding interrupt condition is active, then an interrupt is generated to the processing core that owns this Submission Queue. Interrupt sources that are disabled (‘0’) are still indicated in the IS register. This register is symmetrical with the SQISy register.

Bit	Type	Reset	Description
31:4	RO	0h	Reserved
3	RW	0h	**CQ Disable Interrupt Enable (CDIE)** When set and SQISy.CDS is set, the controller shall generate an interrupt.
2	RW	0h	**SQ CleanUp Interrupt Enable (SCIE)** When set and SQISy.SCS is set, the controller shall generate an interrupt.
1	RW	0h	**SQ Stop Interrupt Enable (SSIE)** When set and SQISy.SSS is set, the controller shall generate an interrupt.
0	RW	0	**Head Entry Fetch Interrupt Enable (HEFIE):** When set and SQISy.HEFS is set, the controller shall generate an interrupt.

5.9.6.8 Offset CQDAOy – CQ y Head Pointer (CQHPy)

Bit	Type	Reset	Description
31:18	RO	0	Reserved
17:0	RW	Impl Spec	**Completion Queue y Head Pointer:** This register holds the 32-bits offset of system memory mapped address of the head entry of this Completion Queue, relative to {CQUBAy, CQLBAy}. The content of this register is primarily updated by Host Driver SW when it fetches an entry from this Completion Queue, and read by Host HW Controller, to determine amount of free space available in this Completion queue. The pointer is DWord aligned, bit 0-1 are zero. The size of CQ is specified by CQATTRy.SIZE, which is 16 bits wide and in terms of DW. Hence this field is 18 bits wide.

5.9.6.9 Offset CQDAOy + 04h – CQ y Tail Pointer (CQTPy)

Bit	Type	Reset	Description
31:18	RO	0	Reserved
17:0	RW	Impl Spec	**Completion Queue y Tail Pointer:** This register holds the 32-bits offset of system memory mapped address of the Tail entry of this Completion Queue, relative to { CQUBAy, CQLBAy}. The content of this register is primarily updated by Host Controller when a new entry is pushed into this Completion queue and read by Host Driver SW to determine if this Completion queue is empty. It is read only by Host SW when Queue enable bit in CQATTR is “1”. The pointer is DWord aligned, bit 0-1 are zero. The size of CQ is specified by CQATTRy.SIZE, which is 16 bits wide and in terms of DW. Hence this field is 18 bits wide.

5.9.6.10 Offset CQISAOy – CQ y Interrupt Status (CQISy)

This register indicates pending interrupts for this CQ that requires service.

Bit	Type	Reset	Description
31:1	RO	0h	Reserved
00	RWC	0	**Tail Entry Push Status (TEPS):** Indicates that a new Completion entry has been pushed at the tail of this CQ by Host controller. Upon receiving this interrupt, Host SW reads CQTPy to determine if this completion Queue is empty. The following events are used for (1) setting this bit to '1' irrespective of IAG association of this queue, and (2) towards counting in associated IAG if IAG is active for this queue: • Overall command Status (OCS) of the completed command is equal to "SUCCESS" with its UTRD Interrupt bit set to '1'. • Overall command Status (OCS) of the completed command is not equal to "SUCCESS" irrespective of its UTRD Interrupt bit value.

5.9.6.11 Offset CQISAOy + 04h – CQ y Interrupt Enable (CQIEy)

This register enables and disables the reporting of the corresponding interrupt to host software. When a bit is set ('1') and the corresponding interrupt condition is active, then an interrupt is generated to the processor core that own this Completion Queue. Interrupt sources that are disabled ('0') are still indicated in the IS register. This register is symmetrical with the CQISy register.

Bit	Type	Reset	Description
31:1	RO	0h	Reserved
0	RW	0	**Tail Entry Push Interrupt Enable (TEPIE):** When set and **CQISy.TEPS** is set, the controller shall generate an interrupt.

5.9.6.12 Offset CQISAOy + 08h – MCQ Interrupt Aggregation Control Register y (MCQIACRy)

NOTE For this register, y = 0..MCQCAP.MIAG – 1 where MCQCAP.MIAG ≤ CQCAP.MAXQ

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31</td><td>RW</td><td>0</td><td>Interrupt Aggregation Enable/Disable (IAEN): When set to ‘0’ by host software, command reponses are neither counted nor timed. Interrupts are still triggerred by responses to Interrupt Commands.
When set to ‘1’, the interrupt aggregation mechanism is enabled and aggregation-based interrupts are generated.
<table>
<tr><th>Bit Value</th><th>Description</th></tr>
<tr><td>0</td><td>Disable</td></tr>
<tr><td>1</td><td>Enable</td></tr>
</table></td></tr>
<tr><td>30:25</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>24</td><td>WO</td><td>0</td><td>Interrupt aggregation parameter write enable (IAPWEN): When host SW writes ‘1’, the values in IACTH and IATOVAL are updated with the contents written at the same cycle.
When host SW writes ‘0’, the values in IACTH and IATOVAL are not updated.
NOTE Write operations to IACTH and IATOVAL are only allowed when no commands are outstanding.</td></tr>
<tr><td>23:21</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>20</td><td>RO</td><td>0</td><td>Interrupt aggregation status bit (IASB):
This bit indicates to Host SW whether any responses have been received and counted towards interrupt aggregation (i.e., IASB is set iff IA counter > 0).
<table>
<tr><th>Bit Value</th><th>Description</th></tr>
<tr><td>0</td><td>No commands has been received since last counter reset (IA counter = 0)</td></tr>
<tr><td>1</td><td>At least one command has been received and counted (IA counter > 0)</td></tr>
</table></td></tr>
<tr><td>19:17</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>16</td><td>WO</td><td>0</td><td>Counter and Timer Reset (CTR): When host SW writes ‘1’, the interrupt aggregation timer and counter are reset.
It is recommended that host software use this field to reset the timer and counter every time it services newly received UTP responses.</td></tr>
<tr><td>15:13</td><td>RO</td><td>0</td><td>Reserved</td></tr>
</table>

5.9.6.12 Offset CQISAOy + 08h – MCQ Interrupt Aggregation Control Register y (MCQIACRy) (cont'd)

Bit	Type	Reset	Description
12:8	RW	0	**Interrupt aggregation counter threshold (IACTH):** Host SW uses this field to configure the number of responses that are required to generate an interrupt. **Counter Operation:** As events (refer to description of SQIS.HEFS and CQIS.TEPS) from queues associated with this IAG are received by the host controller, they are counted in a counter that is incremented with each such event. The counter is reset by software during the interrupt service routine. The counter stops counting when it reaches the value configured in **IACTH**, and sets the **IS.IAGES** bit. The maximum allowed value is 31 NOTE 1 When **IACTH** is 0, responses are not counted, and counting-based interrupts are not generated. NOTE 2 QUERY RESPONSE UPIUs and NOP IN UPIUs shall not be counted by the Interrupt Aggregation logic. In order to write to this field, the IAPWEN bit must be set at the same write operation.
7:0	RW	0	**Interrupt aggregation timeout value (IATOVAL):** Host SW uses this field to configure the maximum time allowed between a response arrival to the host controller and the generation of an interrupt. **Timer Operation:** The timer is reset by software during the interrupt service routine. It starts running when the host controller first recieves any of the events (refer to description of SQIS.HEFS and CQIS.TEPS) from queues associated with this IAG after the timer was reset. The timer stops when it reaches the value configured in **IATOVAL**, and the **IS.IAGES** bit is set. NOTE 1 When **IATOVAL** is 0, the timer is not running, and timer-based interrupts are not generated. NOTE 2 QUERY RESPONSE UPIUs and NOP IN UPIUs shall not be counted by the Interrupt Aggregation logic. The Time units in this field are 40 us. Therefore, writing 0x01 represents a time-out value of 40 us, and writing 0xFF represents a time-out value of 10.2 ms.

6 Data Structures

Most communications between host software and the UFS subsystems are via system memory descriptors. These descriptors describe commands to be executed, and data transfer operations that are part of those commands. This clause defines these descriptors. The data structure definitions in this clause support a 32-bit or 64-bit memory buffer address space. They are all in little endian format except as noted.

6.1 Legacy Single Doorbell Mode - UTP Transfer Request List

The interface consists of UFS Transfer Request Descriptors that are managed in a list. The list is an array that consists of up to 32 UFS Transfer Request Descriptors (**UTRD**). The base of the List structure is pointed by a 64-bit pointer specified in the **UTRLBA/ UTRLBAU** registers. Except UTP Task Management, all UTP command types (SCSI/UFS commands and Device Management) utilize the same UTRD structure.

6.1.1 UTP Transfer Request Descriptor

This table defines Transfer Request Descriptor for UTP commands. The data structure supports a 32-bit or 64-bit memory buffer address space.

	31 30 29 28	27	26 25	24	23	22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
DW 0	Command Type	R	DD	I	CE	Reserved	Total EHS Length	Crypto Config. Index (CCI)
DW 1	Data Unit Number Lower 32 bits (DUNL)							
DW 2	Last DATA Byte Count (LDBC)						Common Data Size (CDS)	Overall Command Status
DW 3	Data Unit Number Upper 32 bits (DUNU)							
DW 4	UTP Command Descriptor Base Address							Reserved
DW 5	UTP Command Descriptor Base Address Upper 32 bits							
DW 6	Response UPIU Offset						Response UPIU Length	
DW 7	PRDT Offset						PRDT Length	

Figure 10 — UTP Transfer Request Descriptor

6.1.1 UTP Transfer Request Descriptor (cont'd)

The following tables provide the description of the data structure.

<table>
<tr><th></th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="4">DW0</td><td>31:28</td><td>Command Type (CT): CT = 1h. Type of the command to be transferred.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>0h</td><td>Reserved</td></tr>
<tr><td>1h</td><td>UFS Storage</td></tr>
<tr><td>2h-Eh</td><td>Reserved</td></tr>
<tr><td>Fh</td><td>Nullified UTRD (valid only in MCQ Mode):
Host controller skips this transfer request, reply CQ entry to Host SW with OCS = ABORTED
NOTE This value is valid only when Config.QT is 01h – Select Multi-Circular Queue Mode</td></tr>
</table></td></tr>
<tr><td>27</td><td>Reserved</td></tr>
<tr><td>26:25</td><td>Data Direction (DD): This field indicates the direction of a data transfer as part of this command. When set to '01b', indicates that the transfer is from system memory to the target device. The PRDT is used as the memory block descriptions for the source. When set to '10b', indicates that the transfer is from the target device to system memory. The PRDT is used the memory block descriptions for the destination. This field is cleared to '00b' when there is no data transfer.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>00b</td><td>No data transfer. PRDT shall have 0 entry.</td></tr>
<tr><td>01b</td><td>From system memory to target device</td></tr>
<tr><td>10b</td><td>From target device to system memory</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>24</td><td>Interrupt (I): This field indicates the type of command with regard to interrupt generation.
<table>
<tr><th>Bit</th><th>Definition</th><th>Description</th></tr>
<tr><td>0b</td><td>Regular Command</td><td>Hardware shall count the completion of this command towards interrupt aggregation. Impact on IS.UTRCS is described in Interrupt Aggregation 5.9.6.12.</td></tr>
<tr><td>1b</td><td>Interrupt Command</td><td>Hardware shall set IS.UTRCS to '1' on completion of this command. The completion is not counted towards interrupt aggregation.</td></tr>
</table>
NOTE 1 QUERY RESPONSE UPIUs and NOP IN UPIUs are not counted towards interrupt aggregation by the host controller hardware. If an interrupt is required upon the completion of a Query Request or NOP transaction, UTRD.I bit shall be set.
NOTE 2 For MCQ mode, see CQISy.TEPS.</td></tr>
</table>

DW0 (cont'd)

<table>
<tr><th></th><th>Bit</th><th>Description</th></tr>
<tr><td rowspan="4">DW0</td><td>23</td><td>Crypto Enable (CE):
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>0b</td><td>Disable cryptographic operations for this transaction.</td></tr>
<tr><td>1b</td><td>Enable cryptographic operations for this transaction.
Incoming payload is either decrypted or used to generate a cryptographic hash, depending on the algorithm specified by CCI if the command is SCSI READ operation;
Outgoing payload is encrypted if the command is SCSI WRITE operation.
The transfer request shall be terminated with an OCS status of ABORTED for a SCSI WRITE operation in conjunction with a crypto hashing algorithm as configured by CCI.
UFS host controller takes no action for all other commands.</td></tr>
</table>
NOTE This field is valid only in controllers supporting cryptographic operations (CAP.CS = 1).
NOTE: Encryption algorithms are applicable to both READ and WRITE requests, whereas hashing algorithms are applicable only to READ requests.
If CAP.CS = 0, this field is reserved.</td></tr>
<tr><td>22:16</td><td>Reserved</td></tr>
<tr><td>15:08</td><td>Total EHS Length (TEL): This field represents the size in 32-bytes units of all Extra Header Segments contained within the UPIU. Total EHS length align with Device spec, the valid value of this field should be one of the following, 0,1,2,3.
NOTE This field is valid when CAP.EHSLUTRDS is 1.</td></tr>
<tr><td>07:00</td><td>Crypto Configuration Index (CCI): The index of Crypto Configuration to be used with this transaction. The values allowed are between 0 and CCAP.CFGC
When UTRD.CE is 0, this field is reserved.
NOTE This field is valid only in controllers supporting cryptographic operations (CAP.CS = 1).
If CAP.CS = 0, this field is reserved.</td></tr>
</table>

	Bit	Description
DW1	31:00	**Data Unit Number Lower 32 bits (DUNL)**: Contains bits [31:00] of the 64-bit DUN cryptographic parameter which is used by some algorithms for key generation. NOTE This field is valid only in controllers supporting cryptographic operations (CAP.CS = 1). If CAP.CS = 0, this field is reserved.

<table>
<tr><th></th><th>Bit</th><th>Initialization Value</th><th>Description</th></tr>
<tr><td rowspan="3">DW2</td><td>31:16</td><td></td><td>Last Data Byte Count (LDBC): Indicates the size of the data block of the last PRDT Entry. A maximum of length of 256 KB may exist for this entry. Dword granularity. A value of ‘0’ indicates 4B, a value of ‘1’ indicates 8B, etc.</td></tr>
<tr><td>15:08</td><td></td><td>Common Data Size (CDS): This value indicates the PRDT Entry size for all PRDT Entries, except for the last PRDT entry. A maximum length of 256 KB may exist for any entry. Granularity of this field is 4096 Bytes. Only values in range of 1-64 are valid. A value of ‘3’ indicates that a Data Buffer Pointed by a single PRDT entry is 12 KB.</td></tr>
<tr><td>07:00</td><td>0Fh</td><td>Overall Command Status (OCS): Contains the command status of the associated command. The command status field is valid after host controller has cleared the corresponding UTRLDBR bit to zero.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>00h</td><td>SUCCESS</td></tr>
<tr><td>01h</td><td>INVALID_COMMAND_TABLE_ATTRIBUTES</td></tr>
<tr><td>02h</td><td>INVALID_PRDT_ATTRIBUTES</td></tr>
<tr><td>03h</td><td>MISMATCH_DATA_BUFFER_SIZE</td></tr>
<tr><td>04h</td><td>MISMATCH_RESPONSE_UPIU_SIZE</td></tr>
<tr><td>05h</td><td>COMMUNICATION_FAILURE within UIC layers</td></tr>
<tr><td>06h</td><td>ABORTED</td></tr>
<tr><td>07h</td><td>FATAL ERROR within host controller that is not covered by the error conditions described above in this table.</td></tr>
<tr><td>08h</td><td>DEVICE FATAL ERROR: A fatal error within the device</td></tr>
<tr><td>09h</td><td>INVALID_CRYPTO_CONFIGURATION</td></tr>
<tr><td>0Ah</td><td>GENERAL_CRYPTO_ERROR</td></tr>
<tr><td>0Bh-0Eh</td><td>Reserved</td></tr>
<tr><td>0Fh</td><td>INVALID_OCS_VALUE</td></tr>
<tr><td>10h-FFh</td><td>Reserved</td></tr>
</table></td></tr>
</table>

	Bit	Description
DW3	31:00	**Data Unit Number Upper 32 bits (DUNU)**: Contains bits [63:32] of the 64-bit DUN cryptographic parameter which is used by some algorithms for key generation. NOTE This field is valid only in controllers supporting cryptographic operations (CAP.CS = 1). If CAP.CS = 0, this field is reserved.

	Bit	Description
DW4	31:07	**UTP Command Descriptor Base Address (UCDBA):** Indicates the lower 32-bits of the physical address of the command descriptor, which contains the Command, Status, and PRD Table. This address shall be aligned to a 128-byte address, indicated by bits 06:00 being reserved.
	06:00	Reserved

	Bit	Description
DW5	31:00	**UTP Command Descriptor Base Address Upper 32-bits (UCDBAU):** This is the upper 32-bits of the Command Descriptor Base.

	Bit	Description
DW6	31:16	**Response UPIU Offset (RUO):** This field contains the Dword offset of the Response UPIU within the Command Descriptor. The Response UPIU may be located at a 64-bit aligned boundary right after Command UPIU in the Command Descriptor (Bit 16 is always 0).
	15:00	**Response UPIU Length (RUL): T**his field contains the length of the Response UPIU in Dword. The use and format of the Response UPIU depends on command type for this command.

	Bit	Description
DW7	31:16	**PRDT Offset (PRDTO):** This field contains the Dword offset for the Physical Region Description Table within the Command Descriptor. The Physical Region Description Table may be located at a 64-bit aligned boundary right after Response UPIU in the Command Descriptor (Bit 16 is always 0). This field is valid only when field **PRDTL** > 0.
	15:00	**PRDT Length (PRDTL):** This field contains the count of the entries in PRDT. A ‘0’ means that PRDT is empty. If this field is ‘0’, then no data transfer shall occur with the command. The use and format of the PRDT depends on command type for this command. For non-data transfer requests and Device Management function, this field must set to ‘0’. Only the commands that require data transfer operation could have non-zero value. The rule is that if a command requires at least one Data-in UPIU or Data-Out UPIU in its sequence, then non-empty PRDT is required.

6.1.2 UTP Command Descriptor

UTRD contains a pointer for a data structure called UTP Command Descriptor (UCD). The data structure consists of the UPIU for the command, the offset and length of the Sense data buffer associated with the command, the offset and length for PRDT (scatter-gather list that is a part of the command). The format of the UTRD is defined as in 6.2.1.

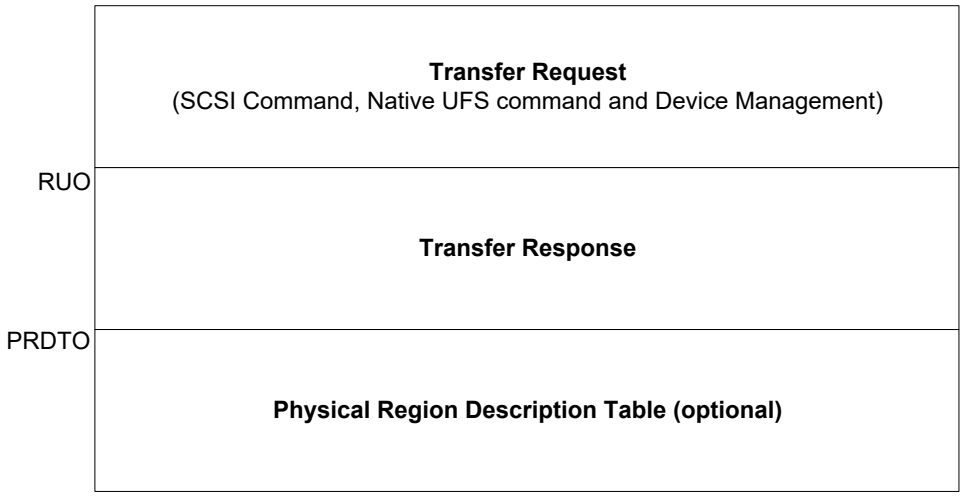


Figure 11 — UTP Command Descriptor (UCD)

NOTE Both Command UPIU (Transfer Request) and Response UPIU (Transfer Response) are in big endian format and PRDT is in little endian format.

The Transfer Request region in the UCD provides the description of the requested command: NOP Out, Command, or Query Request.

The Transfer Response region in the UCD will store the content of the incoming UPIU that completes the Transfer Request: NOP In, Response, or Query Response.

PRD Table supports scatter/gather operations for the command. PRDT is required only for a subset of SCSI commands that requires data transfer operation. UFSHCI supports three command types: SCSI, Native UFS Commands, and Device Management Functions. Refer to [UFS] for the definition of UTP Command UPIU and UTP Response UPIU.

6.1.2 UTP Command Descriptor (cont'd)

To provide description of data buffers that are associated with a UTP Transfer Request, this standard provides a data structure called Physical Region Description Table. For the UTP Transfer Request that does not require data transfer operation, this table is empty.

There are 2 different formats for PRDT Entry, refer to 2DWPRDTEN in config register for details.

- 4DW format: When 2DWPRDTEN = 0, 4DW format is used.
- 2DW format: When 2DWPRDTEN = 1, 2DW format is used.

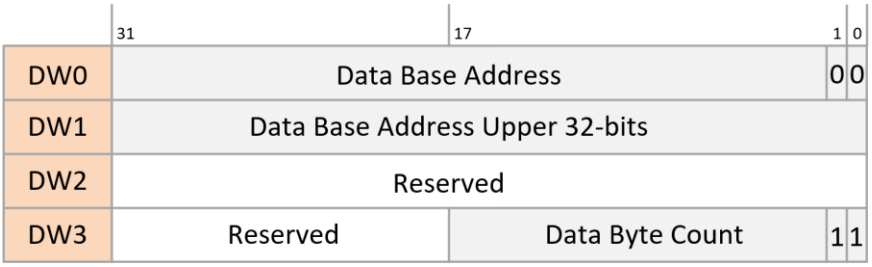


Figure 12 — Data Structure for Normal Physical Region Descriptor (4DW format)

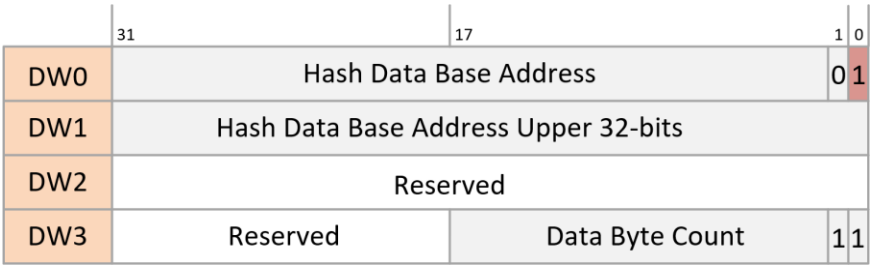


Figure 12-X — Data Structure for Hash Value Physical Region Descriptor (4DW format)

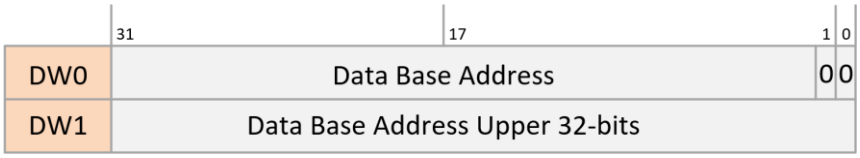


Figure 13 — Data Structure for Physical Region Descriptor (2DW Format)

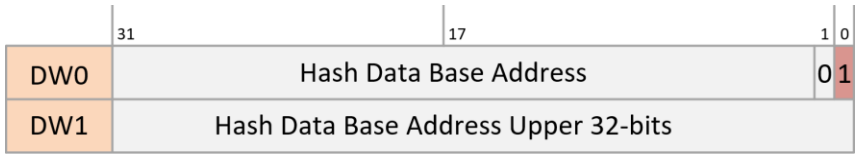


Figure 13-X — Data Structure for Hash Value Physical Region Descriptor (2DW Format)

6.1.2 UTP Command Descriptor (cont'd)

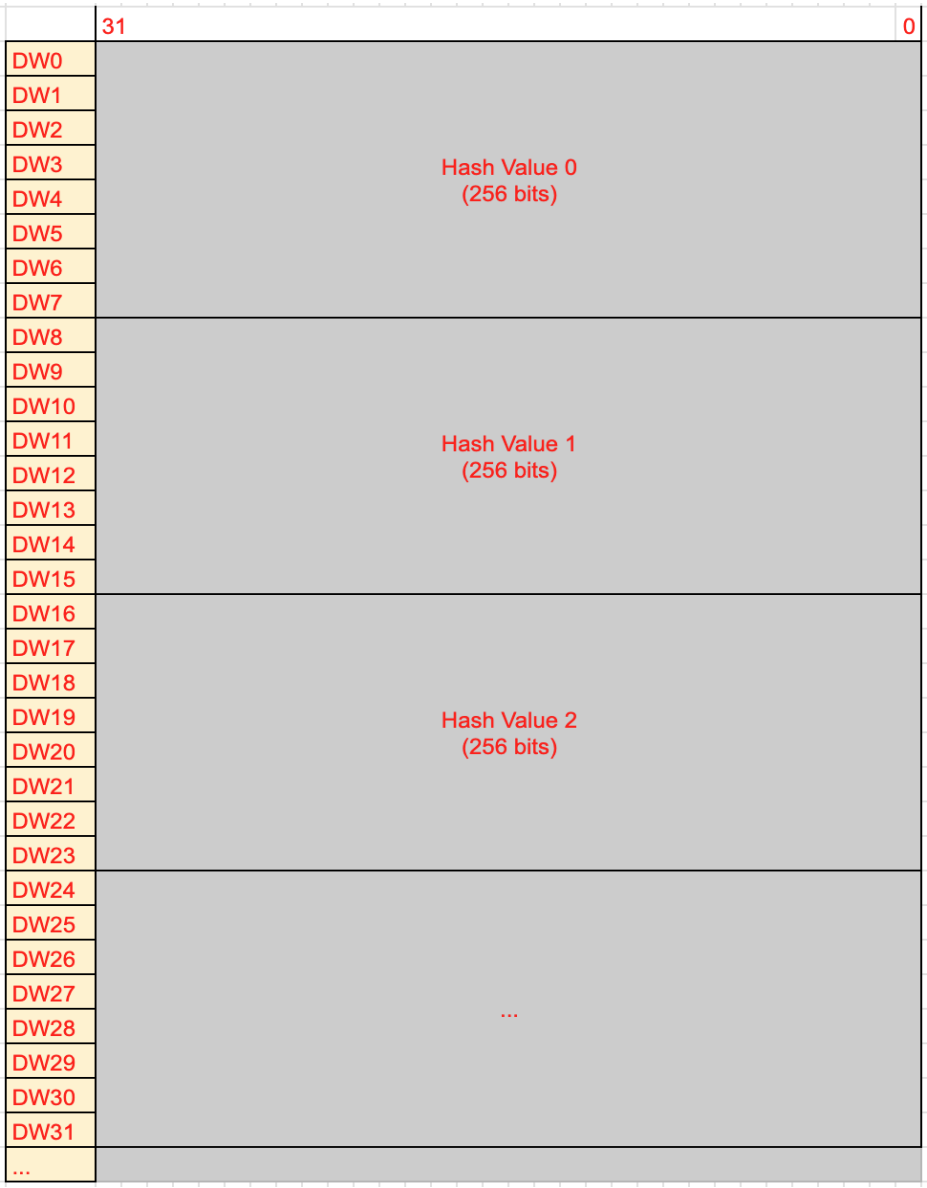


Figure 12-Y — Hash Values

A hash value PRDT entry, described in Figures 12-X and 13-X points to a hash value array, described in Figure 12-Y.

If cryptographic hashing is enabled via the x-CRYPTOCFG register and a READ command's UTRD.CCI field points to that enabled algorithm, the cryptographic hardware shall generate hash values for each data unit. The size of each data unit is defined by the DUSIZE field within the x-CRYPTOCFG register. Figure 12-Y depicts the layout of the hash value block, assuming a hash output of 256 bits per data unit. Within a hash data block, the hash values shall be stored contiguously. The location of the hash block shall be determined by the hash value PRDT entry. See also 6.1.2, UTP Command Descriptor.

6.1.2 UTP Command Descriptor (cont'd)

A breakdown of each field in a PRD table is shown below. Note that DW2 and DW3 are present only in 4DW format.

The 4DW PRDT entry can be used to describe either a data block or a hash data block. For the purpose of this standard, a Normal Physical Region Descriptor Entry (NPRDE) describes a data block and a Hash Physical Region Descriptor Entry (HPRDE) describes a hash data block. Within the PRDT, HPRDEs shall precede corresponding NPRDEs. In case of multiple HPRDEs, the first HPRDE shall precede its associated NPRDEs, and subsequent HPRDEs shall each precede their respective NPRDEs, resulting in an interleaved arrangement of HPRDEs and NPRDEs. The data block described by the NPRDEs associated with a given HPRDE shall be used to generate the hash values described by that HPRDE.

In the event of any of the following conditions, the UFS host controller shall fail the request and report an error status (OCS = INVALID_PRDT_ATTRIBUTES):

- **Hashing Enabled, Missing HPRDEs:** The x-CRYPTOCFG register and UTRD.CCI indicates that hashing is enabled for the request, but the PRDT contains no HPRDEs.
- **Hashing Disabled, HPRDEs Present:** The x-CRYPTOCFG register and UTRD.CCI indicates that hashing is disabled for the request, but the PRDT contains one or more HPRDEs.
- **Incorrect HPRDE Count:** The number of HPRDEs in the PRDT is either more or less than the number required to describe the hash values for the corresponding data blocks. This could involve having extra HPRDEs or missing HPRDEs, regardless of their position in the PRDT.
- **HPRDE Hash Data Size Mismatch**: The hash data byte count (HDBC) specified within an HPRDE does not match the size of its associated data block.
- **Invalid HPRDE/NPRDE Arrangement:** The HPRDEs and NPRDEs are not arranged in the required interleaved pattern, where each HPRDE is immediately followed by its corresponding NPRDEs.

	Bit	Description
DW0	31:02	**Data Base Address (DBA) / Hash Data Base Address (HDBA):** Indicates the lower 32-bits of the physical address of the data block or the hash data block. The block shall be Dword aligned.
	01	Reserved
	00	**PRDT Entry Type (PET)**: Indicates the PRDT entry type 0b: Indicates a normal PRDT entry, which contains a data buffer address 1b: Indicates a hash value PRDT entry, which contains the address holding hash values.

	Bit	Description
DW1	31:00	**Data Base Address Upper 32-bits (DBAU) / Hash Data Base Address Upper 32-bits (HDBAU):** This is the upper 32-bits of the data block or the hash data block physical address.

	Bit	Description
DW2	31:00	Reserved

6.1.2 UTP Command Descriptor (cont'd)

	Bit	Description
DW3	31:18	Reserved
	17:00	**Data Byte Count (DBC) / Hash Data Byte Count (HDBC):** A '0' based value that indicates the length, in bytes, of the data block or the hash data block. A maximum of length of 256 KB may exist for any entry. Bits 1:0 of this field shall be 11b to indicate Dword granularity. A value of '3' indicates 4 bytes, '7' indicates 8 bytes, etc.

NOTE: A single HPRDE can point to a hash data block of up to 256 KiB. If the hash values are 256 bits (as is the case for SHA-256), then a single hash data block holds up to 8192 hash values. This is enough for up to 32 MiB of data if the data unit size is 4 KiB.

6.2 Multi Queue (MCQ) Mode – Submission Queue (SQ) & Completion Queue (CQ)

In Multi Queue (MCQ) mode, UTRL is not used. Instead, the Submission Queue & Completion Queue are constructed in system memory by Host Driver SW, with Head and Tail doorbell pointers stored in the Host HW controller. Submission Queue [y]'s base address is referenced by a 64-bit pointer found in its corresponding **SQLBAy/SQUBAy** registers. Completion Queue [y]'s base address is referenced by a 64-bit pointer found in its corresponding **CQLBAy/CQUBAy** registers.

6.2.1 Submission Queue (SQ)

UTRD is pushed or copied into SQ.

6.2.2 Completion Queue (CQ)

This table defines Completion Queue within a MCQ definition. The data structure supports a 32-bit or 64-bit memory buffer address space.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DW0	UTP Command Descriptor Base Address																									R	R	Submission Queue ID				
DW1	UTP Command Descriptor Base Address Upper 32 bits																															
DW2	Response UPIU Offset																Response UPIU Length															
DW3	PRDT Offset																PRDT Length															
DW4	Reserved																Extended Error Code									Overall Status							
DW5	Reserved												EXT_IID				IID						LUN								Task Tag		
DW6	Reserved																															
DW7	Reserved																															

Figure 14 — Completion Queue Entry

6.2.2.1 Completion Queue Entry

The following tables provide the description of the data structure.

	Bit	Description
DW0	31:07	**UTP Command Descriptor Base Address (UCDLBA):** Copied from SQ Entry.
	06:05	Reserved
	04:00	**Submission Queue ID (SQID):** ID of the SQ from where the TR request (UTRD) was fetched

	Bit	Description
DW1	31:00	**UTP Command Descriptor Upper Address 32-bits (UCDUBA):** Copied from SQ Entry

	Bit	Description
DW2	31:16	**Response UPIU Offset (RUO):** Copied from SQ Entry
	15:00	**Response UPIU Length (RUL):** Copied from SQ Entry

	Bit	Description
DW3	**31:16**	**PRDT Offset (PO):** Copied from SQ Entry
	15:00	**PRDT Length (PL):** Copied from SQ Entry

	Bit	Description
DW4	31:16	**Reserved**
	15:08	**Extended Error Code (EEC):** For each OCS error value, EEC holds finer granularity error code. 00h : No extra information 01h : Duplicate Task Others : Reserved
	07:00	**Overall Status (OCS):** In MCQ, controller provides OCS in this field in CQ Entry, and not in UTRD.OCS field. The definition remains same as that of Legacy Single Doorbell mode.

	Bit	Description
DW5	31:24	**Reserved**
	23:20	**EXT_IID:** Copied from SQ Entry
	19:16	**IID:** Copied from SQ Entry
	15:08	**LUN:** Copied from SQ Entry
	07:00	**Task Tag:** Copied from SQ Entry

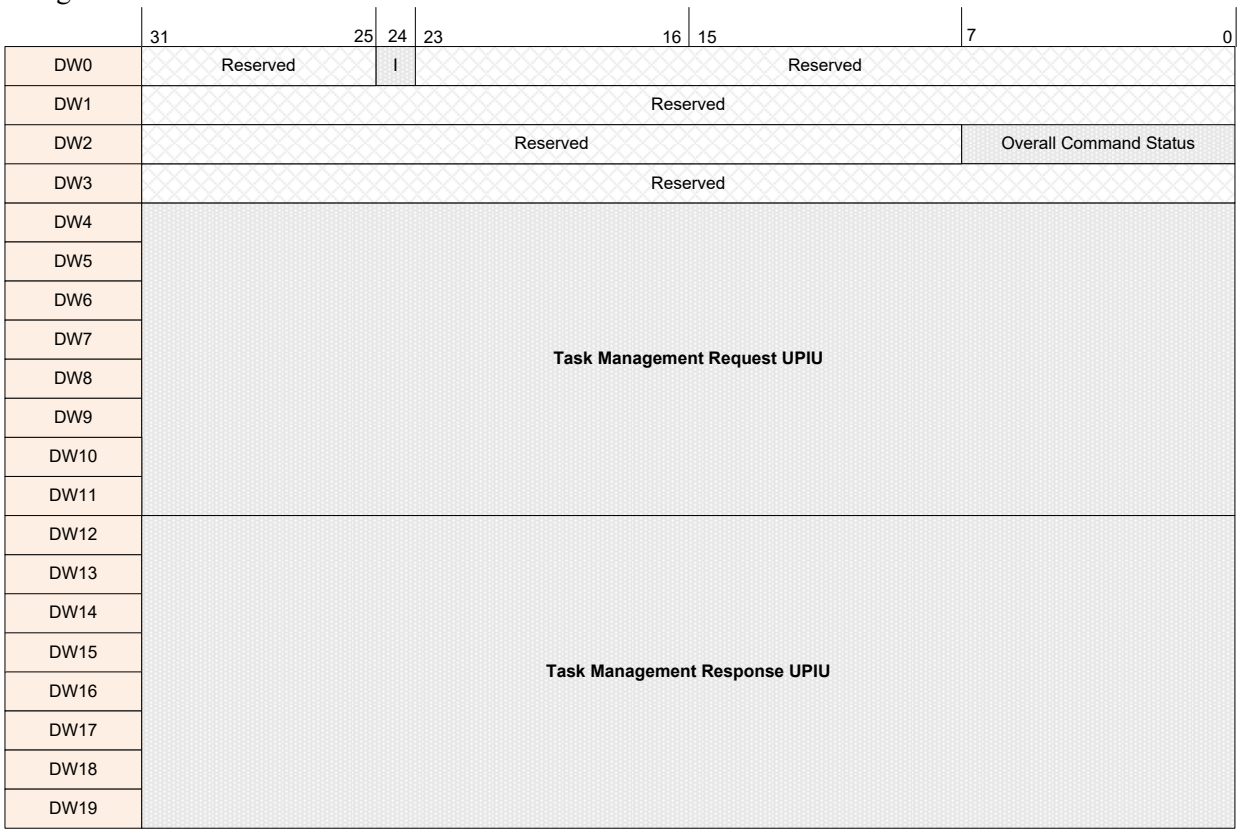
	Bit	Description
DW6-7	31:00	**Reserved**

6.3 UTP Task Management Request List

The list consists of a list of UTP Task Management Request Descriptors. The Maximum of the list size is 8. Host software is responsible to process the Task Management Response UPIU directly

6.3.1 UTP Task Management Request Descriptor

Host controller acts as a pass through for UTP Task Management. The format of the UTMRD is defined as in Figure 15.



NOTE The first 4 DWORD are in little endian format. But both Task Management Request UPIU and Task Management Response UPIU are in big endian format.

Figure 15 — UTP Task Management Request Descriptor.

The following table provides the description of the 1st DWORD (DW0) of the data structure.

Bit	Description
31:25	Reserved.
24	**Interrupt (I):** When set to '1', hardware shall set **IS.UTMRCS** to '1' on completion of this command. When cleared to '0', hardware shall not set **IS.UTMRCS** to '1' on completion of this command.
23:00	Reserved

6.3.1 UTP Task Management Request Descriptor (cont'd)

The following table provides the description of the 3rd DWORD (DW2) of the data structure.

<table>
<tr><th>Bit</th><th>Initialization Value</th><th>Description</th></tr>
<tr><td>31:08</td><td>0</td><td>Reserved.</td></tr>
<tr><td>07:00</td><td>Fh</td><td>Overall Command Status (OCS): Contains the status of the Task Management Request. The status field is valid after host controller has cleared the corresponding UTMRLDBR bit to zero.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>SUCCESS</td></tr>
<tr><td>1h</td><td>INVALID_TASK MANAGEMENT FUNCTION_ATTRIBUTES</td></tr>
<tr><td>2h</td><td>MISMATCH_TASK_MANAGEMENT_REQUEST_SIZE</td></tr>
<tr><td>3h</td><td>MISMATCH_TASK_MANAGEMENT_RESPONSE_SIZE</td></tr>
<tr><td>4h</td><td>PEER_COMMUNICATION_FAILURE</td></tr>
<tr><td>5h</td><td>ABORTED</td></tr>
<tr><td>6h</td><td>FATAL ERROR</td></tr>
<tr><td>7h</td><td>DEVICE FATAL ERROR: A fatal error within the device</td></tr>
<tr><td>8h-Eh</td><td>Reserved</td></tr>
<tr><td>Fh</td><td>INVALID_OCS_VALUE</td></tr>
</table></td></tr>
</table>

Both Task Management Request UPIU and Task Management Response UPIU are 32-Byte in length. Refer to [UFS] for the definition of Task Management Request UPIU and Task Management Response UPIU.

6.4 Key Organization for Cryptographic Algorithms

The organization of keys in the CRYPTOKEY field of CRYPTOCFG entries is dependent upon the algorithm and the key size. This sub-clause provides information about algorithm-specific organization of the CRYPTOKEY field. Additional information about the algorithms can be found in clause 9.

6.4.1 AES-XTS

The AES-XTS algorithm uses two keys, and allows three possible key sizes: 128 bits, 192 bits and 256 bits, corresponding to the key sizes defined for the AES block cipher. 512 bits of key material shall be supplied in any key size mode. The key material organizations for AES128-XTS, AES192-XTS, and AES256-XTS are shown in Figure 16, Figure 17, and Figure 18, respectively.

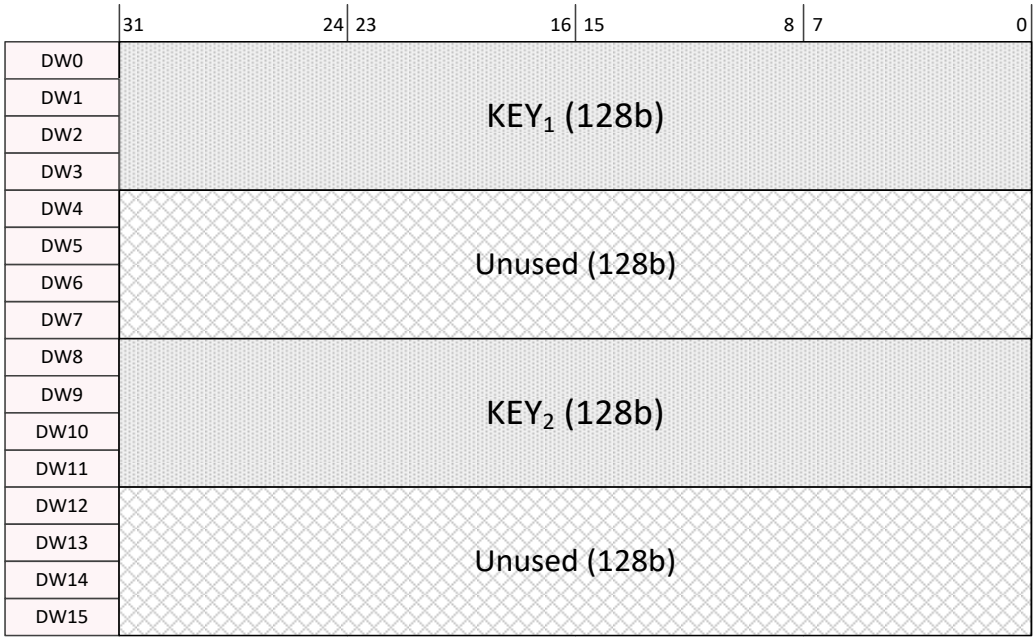


Figure 16 — AES128-XTS Key Layout

6.4.1 AES-XTS (cont'd)

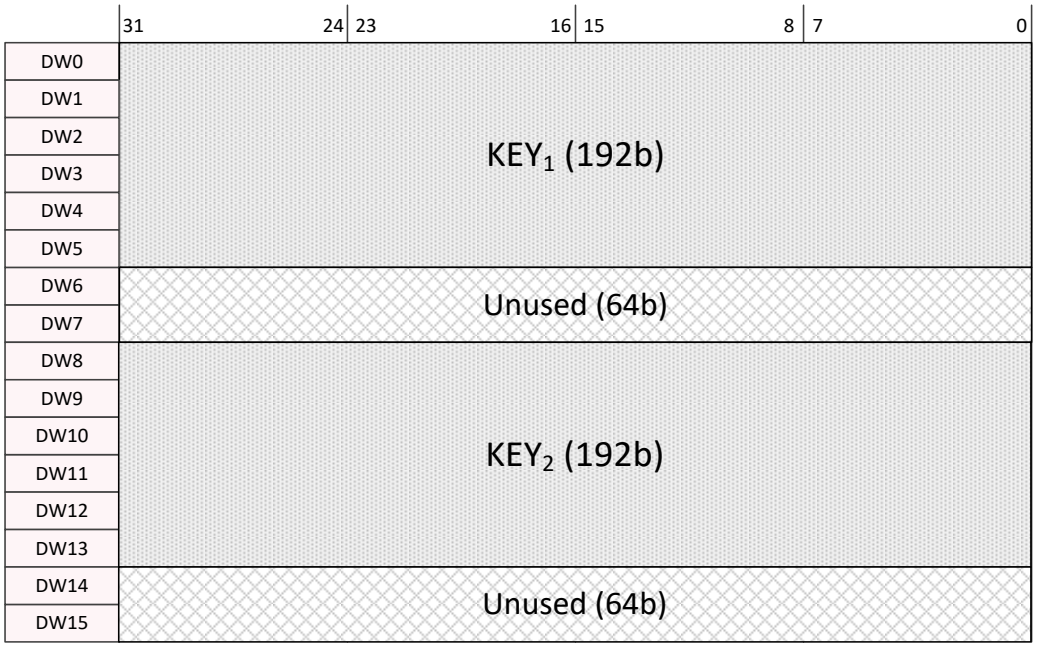


Figure 17 — AES192-XTS Key Layout

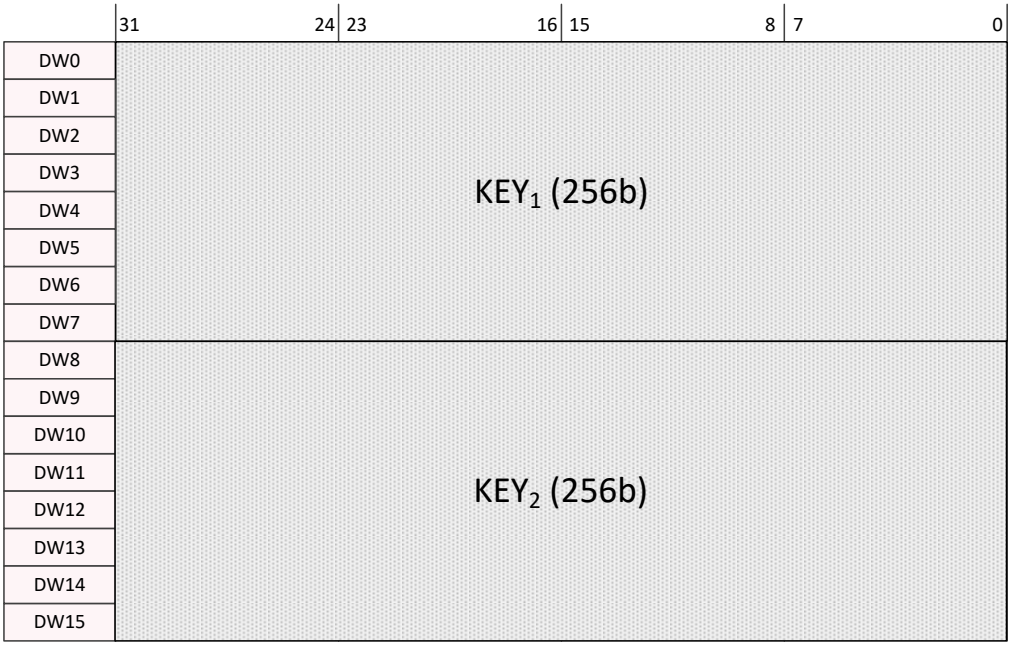


Figure 18 — AES256-XTS Key Layout

6.4.2 Microsoft Bitlocker™ AES-CBC

Microsoft BitLocker™ key infrastructure provides a 512-bit key *K*. For 128-bit AES-CBC, bits 0 through 127 of key *K* are used as the AES key (as shown in Figure 19). For 256-bit AES-CBC, bits 0 through 255 of *K* are used as the AES key (as shown in Figure 20). The unused bits are ignored.

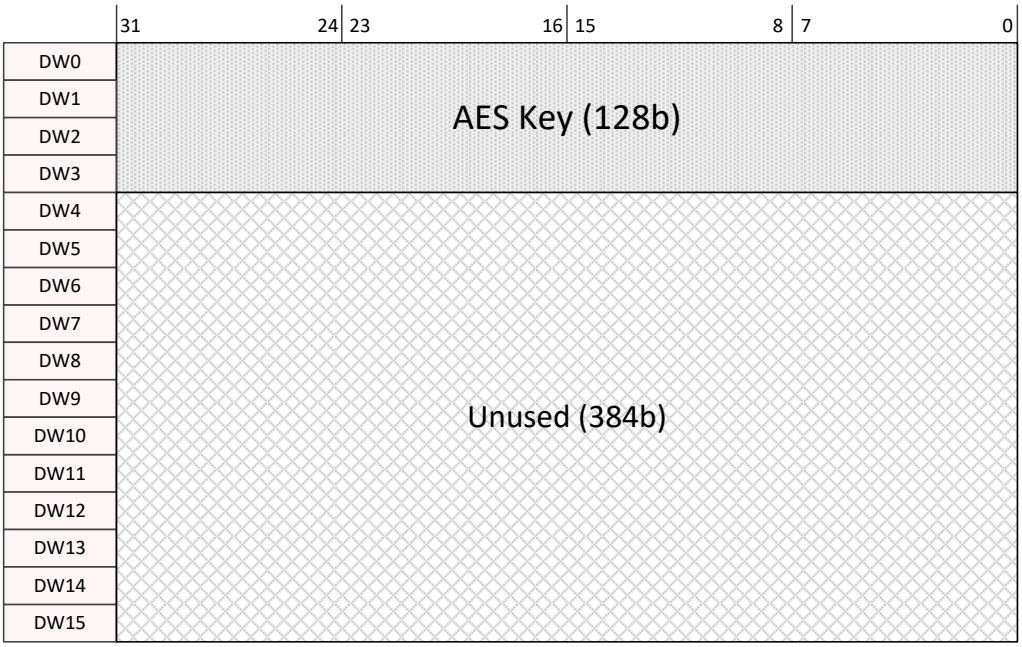


Figure 19 — AES128-CBC Key Layout

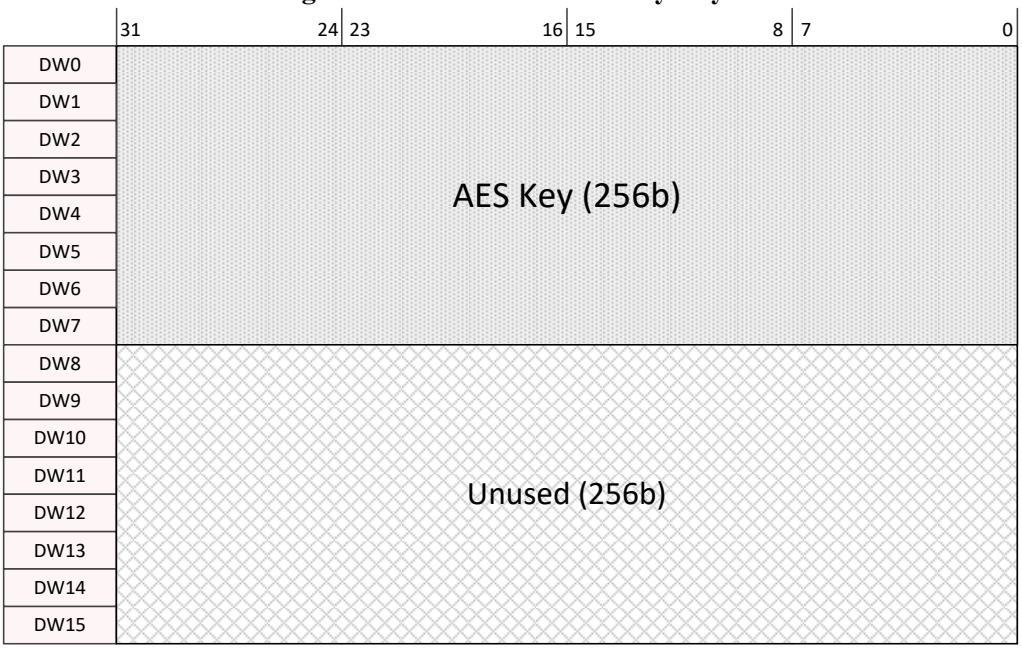


Figure 20 — AES256-CBC Key Layout

6.4.3 AES-ECB

The AES-ECB algorithm allows two possible key sizes: 128 bits, and 256 bits, corresponding to the key sizes defined for the AES block cipher. The key material organizations for AES128-ECB and AES256-ECB are shown in Figure 21 and Figure 22, respectively.

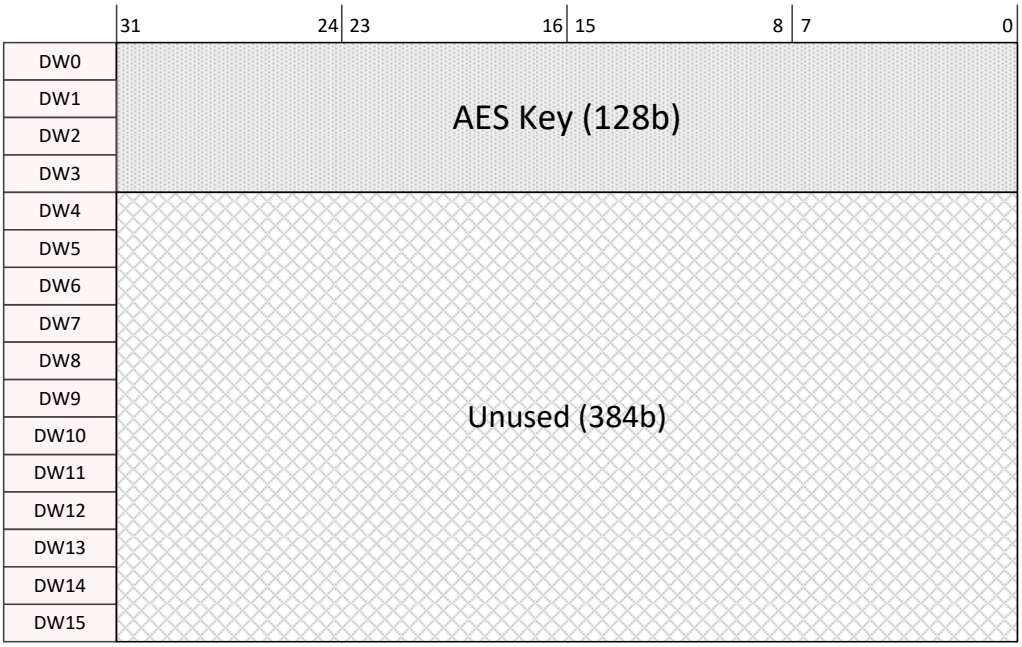


Figure 21 — AES128-ECB Key Layout

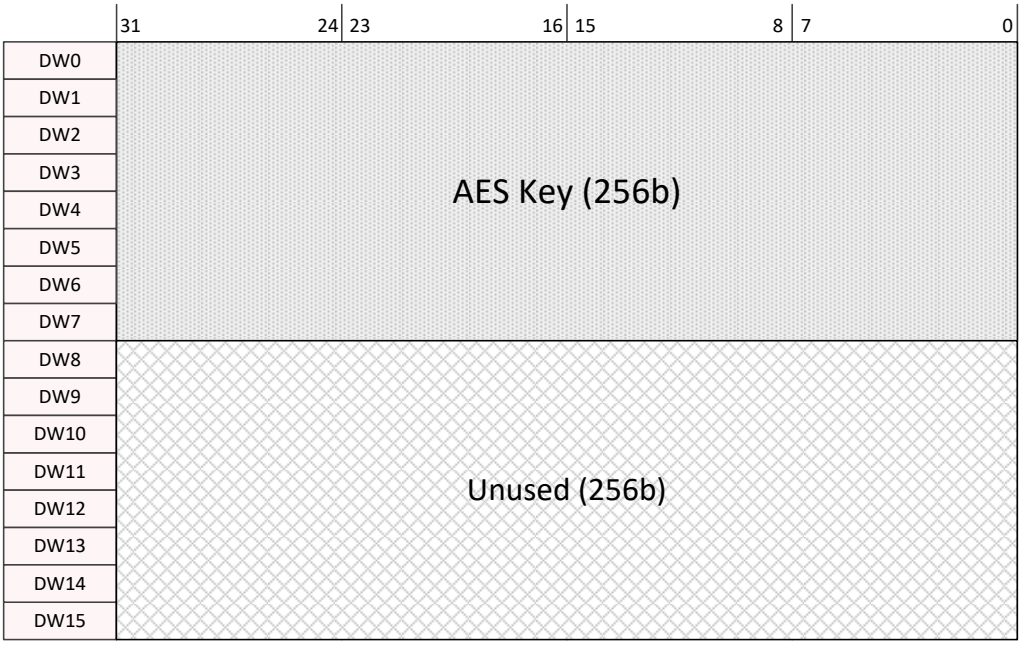


Figure 22 — AES256-ECB Key Layout

6.4.4 ESSIV-AES-CBC

ESSIV-AES-CBC key infrastructure provides a 512-bit key *K*. For 128-bit AES-CBC, bits 0 through 127 of key *K* are used as the AES key (same as shown in Figure 19). For 256-bit AES-CBC, bits 0 through 255 of K are used as the AES key (same as shown in Figure 20). The unused bits are ignored.

7 Theory of Operation

While the register interface provides the concise description of the software interface to UFSHCI, the implementation and interpretation of these various bits is not always clear from the definition of the bit(s). This clause is a supplement to the register definitions, and provides narrative and interpretation guidance for the behaviors of the bits. The software operation of the UFS HCI is divided into three categories: Host Controller Configuration and Control, Data Transfer Operation, and Task Management. These three categories are discussion in detail in the following sub-clauses.

7.1 Host Controller Configuration and Control

7.1.1 Host Controller Initialization

When the host controller comes out of power up reset, all MMIO registers will be in their power-on default state, and the link will be inactive. Following is a sequence of the operations that host software would perform to initialize the host controller:

1) The first step in starting the controller is properly programming system bus interface. Because this operation is specific to the system bus used by the controller implementation, the documentation for the specific system bus should be followed. At the conclusion of this programming, the controller should be ready to transfer data on the system bus.
2) Write a 1 to the **HCE** register in order to enable the host controller. This triggers an autonomous basic initialization of the local UIC layer. The initialization sequence shall consist of a DME_RESET and a DME_ENABLE command. Further commands, such as DME_SET commands may be added, depending on the implementation needs. During the basic initialization sequence, the **HCE** is read as '0'.
3) Wait until **HCE** is read as '1' before continuing. This indicates that the basic initialization sequence is completed.
4) Additional commands, such as DME_SET commands may be sent from the system host to the UFS host controller to provide configuration flexibility.
5) Optionally set **IE.UCCE** to 1 in order to enable the **IS.UCCS** interrupt
6) Sent DME_LINKSTARTUP command to start the link startup procedure.
7) Completion of the DME_LINKSTARTUP command sets the **IS.UCCS** bit and may flag an interrupt to the system host if the **IE.UCCE** is set. This interrupt will be flagged independently from the *GenericErrorCode*.
8) In case the *GenericErrorCode* of the completed DME_LINKSTARTUP command is SUCCESS, the **HCS.DP** is set in addition to the **IS.UCCS** bit .
9) Check value of **HCS.DP** and make sure that there is a device attached to the Link. If presence of a device is detected, go to step 10); otherwise, resend the DME_LINKSTARTUP command after **IS.ULSS** has been set to 1 (Go to step 6)). **IS.ULSS** equal 1 indicates that the UFS Device is ready for a link startup.
10) Enable additional interrupts by programming the **IE** register.

7.1.1 Host Controller Initialization (cont'd)

11) Initialize the Interrupt Aggregation Control Register (**UTRIACR**) with the desired values for the threshold (IACTH) and timeout (IATOVAL).

    For example, write value 0x81010664 to initialize with the following parameters: bits 31 (enable), 24 (IAPWEN), and 16 (timer/counter reset) are set, IACTH = 6, IATOVAL = 0x64 (= 4.0 ms).

    NOTE UTRIACR initialization may be executed at any time when the Run/Stop register (**UTRLRSR**) is not enabled or when no requests are outstanding.

12) Complete the host controller configuration via UIC command interface if required.

13) Allocate and initialize UTP Task Management Request List.

14) Program the *UTP Task Management Request List Base Address* and *UTP Task Management Request List Base Address* with a 64-bit address pointed to the starting address of the *UTP Task Management Request List* created at the previous step.

15) Allocate and initialize UTP Transfer Request List.

16) Program the *UTP Transfer Request List Base Address* and *UTP Transfer Request List Base Address* with a 64-bit address pointed to the starting address of the *UTP Transfer Request List* created at the previous step.

17) Enable the *UTP Task Management Request List* by setting the *UTP Task Management Request List Run-Stop Register (***UTMRLRSR***)* to '1'. This operation allows the host controller to begin accepting UTP Task Management Request via the UTP Task Management Request DoorBell mechanism.

18) Enable the *UTP Transfer Request List* by setting the *UTP Transfer Request List Run-Stop Register* (**UTRLRSR**) to '1'. This operation allows the host controller to begin accepting UTP Transfer Request via the UTP Transfer Request DoorBell mechanism.

19) bMaxNumOfRTT will be set as the minimum value of bDeviceRTTCap and NORTT. At this point, the host controller is up and running.

7.1.1 Host Controller Initialization (cont'd)

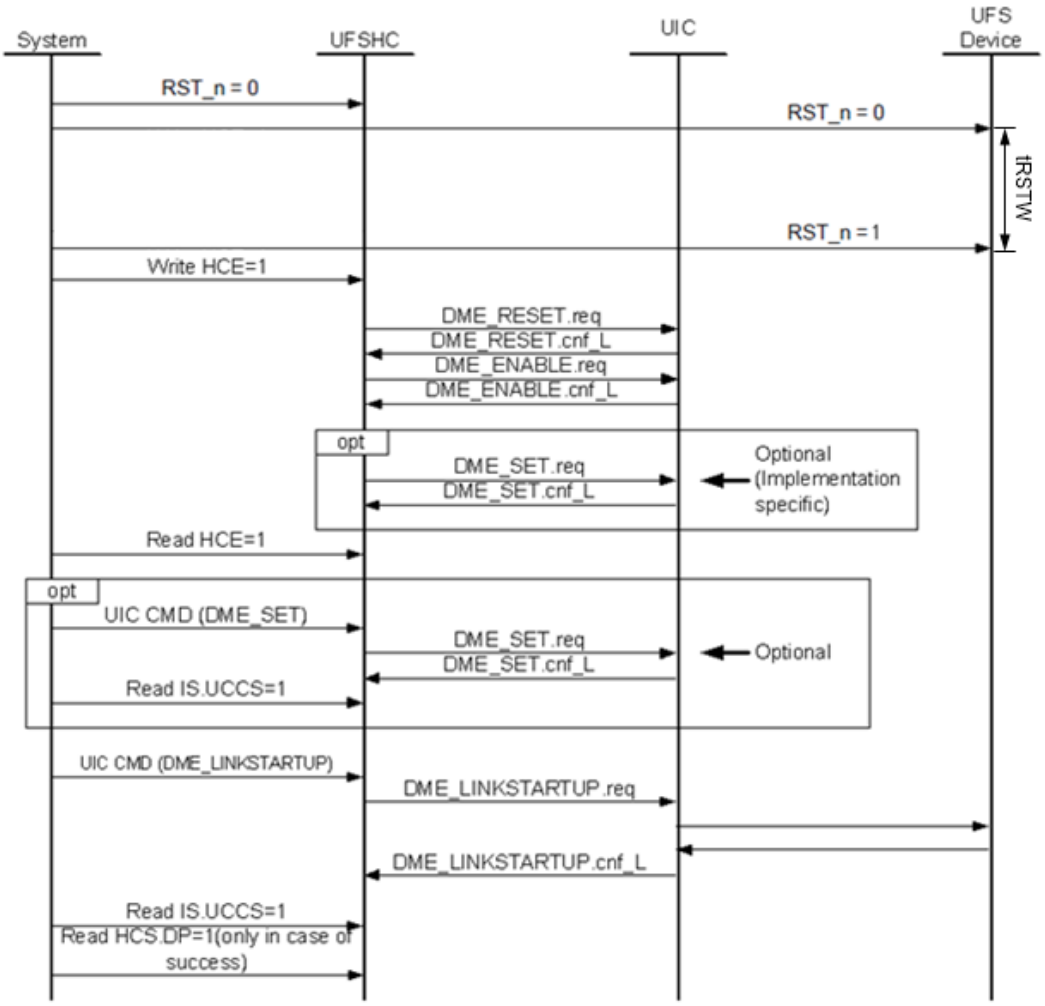


Figure 23 — Host Controller Link Startup Sequence

7.1.2 Configuration and Control

Once the host controller is out of reset, host software may use UIC Command Registers to configure and control the link and the attached device. Host software is responsible to configuring the UFS Interconnect stack and the attached device. When programming UIC command registers, host software shall set the register **UICCMD** only after all the UIC command argument registers (**UICCMDARG1**, **UICCMDARG2** and **UICCMDARG3**) are set. There are two options to check the status of the completion of execution of a UIC command:

- Via interrupt mechanism. Before a UIC command is sent to the host controller for execution, software enables UIC command completion interrupt by setting *UIC COMMAND Completion Enable* register **IE.UCCE**. Once the command execution is completed, the host controller shall generate an interrupt and set the register to *UIC COMMAND Completion Status* register '1'.
- Software Pulling. After a UIC command is sent to the host controller for execution, software keeps pulling the register *UIC COMMAND Completion Status* until it is returned '1'.

Once the command is completed, software can check the return/status code if applicable to the command.

The description of the attributes associated with host controller configuration and control are provided in MIPI UniPro specification [MIPI-UniPro]. Referred to the MIPI UniPro specification [MIPI-UniPro] for the definition of the attributes.

7.1.3 CRYPTOCFG Configuration Procedure

To configure an entry in the CRYPTOCFG array, it is recommended that software follows the procedure below:

1) Select an entry x-CRYPTOCFG in the CRYPTOCFG array.
2) If the entry is not enabled, i.e. x-CRYPTOCFG.CFGE = 0, skip to step 5).
3) Verify that no pending transactions reference x-CRYPTOCFG in their CCI field, i.e., UTRD.CCI $\neq x$ for all pending transactions. After said verification, proceed to step 4).
4) Clear x-CRYPTOCFG.CFGE bit by writing 0000h to DW16 of x-CRYPTOCFG. CAPIDX and DUSIZE fields are cleared in this operation as well.
5) Write the cryptographic key to x-CRYPTOCFG.CRYPTOKEY field according to the following rules:
   A. The key is organized according to the algorithm-specific layout listed in 6.4. Unused regions of CRYPTOKEY should be written with zeros.
   B. The key is written in little-endian format: Byte 0 of the key to CRYPTOKEY[0], byte 1 to CRYPTOKEY[1], byte 15 to CRYPTOKEY[15], etc.
   C. The contents of CRYPTOKEY should be written from DW0 to DW15, sequentially, in one atomic set of operations.
6) Optionally write DW17 of x-CRYPTOCFG.
7) Write DW16 of x-CRYPTOCFG with CAPIDX, DUSIZE, and CFGE = 1.

Only after the completion of the procedure above, software may use the new configuration, by issuing transactions with UTRD.CCI = x.

7.2 Data Transfer Operation

After the host controller reset and configuration is completed, software can utilize the *UTP Transfer Request List* (UTRL) to pass UTP commands to the UFS device connected to the link. The UTRL is a list buffer located in system memory that is used to pass commands from software to the device. Software is responsible for choosing a UTRL size based on the value of **CAP.NUTRS.** In general, the software should choose the 32 entries option unless the system capabilities dictate a smaller memory footprint.

7.2.1 Basic Steps when Building a UTP Transfer Request

When host software needs to send a UTP command to host controller, it utilizes *UTP Transfer Request List*. The following is the steps for host software to build a UTP Transfer Request.

1) Find an empty transfer request slot by reading the **UTRLDBR**. An empty transfer request slot has its respective bit cleared to '0' in the **UTRLDBR**.
2) Host software builds a UTRD at the empty slot.
   A. Program field **CT** (command type) to indicate the command type: SCSI, native UFS command or device management function.
   B. Program field **DD** (data direction) that contains the direction of the data operations that are a part of the command if any.
   C. **I** (interrupt) bit set if software requests to mark the command as an Interrupt Command (**IS.UTRCS** to be set on command completion). The bit is cleared if software requests to mark the command as a Regular Command.
   D. Initialize **OCS** with 'Fh'.
   E. Allocate and initialize a UCD.
   F. Program field *Command UPIU* in UCD with a UTP command excluding task management function.
   G. Initialize field *Response UPIU* in UCD with '0'.
   H. Fill PRD table with the pointers and sizes for all the data buffers associated with the data transfer of the command if required.
3) Program field **UCDBA** and **UCDBAU** with the starting address of UCD.
4) Program field **RUO** with the offset (from the starting address of UCD) of Response UPIU within UCD.
5) Program field **RUL** with the length of *Response UPIU*.
6) Program field **PRDTO** with the offset (from the starting address of UCD) of PRDT within UCD if required.
7) Program field **PRDTL** with the length of PRDT if required. Repeat the step 1) to step 7) for each command to be sent to host controller for execution.
8) Check register **UTRLRSR** and make sure it is read '1' before continuing.
9) Set UTP Transfer Request Interrupt Aggregation Control Register (**UTRIACR**) enable bit to '1' to enable the interrupt.
10) Set Counter and Timer Reset (**CTR**) bit to '1' to reset the counter and timer associated with the interrupt.

7.2.1 Basic Steps when Building a UTP Transfer Request (cont'd)

11) Program field Interrupt aggregation counter threshold (**IACTH**) with the number of command completions that are required to generate an interrupt.
12) Program field Interrupt aggregation timeout value (**IATOVAL**) with the maximum time allowed between a response arrival to the host controller and the generation of an interrupt.
13) Set **UTRLDBR** to ring the doorbell register to indicate to the host controller that one or more transfer requests are ready to be sent to the attached device. Host software shall only write a '1' to the bit position that corresponding to the new command; All other bit positions within UTRLDBR should be written with a '0', which indicates no change to their current values.

7.2.2 UPIU Processing

7.2.2.1 Outbound UPIUs Generated by Software

Except for the DATA OUT UPIU all other UFS defined outbound UPIUs have to be composed and stored in the Host's memory by software. The Host controller then uses DMA to fetch the outbound UPIUs from the memory and dispatches them to the UFS Device using the local UniPro stack. The EXT_IID, IID, LUN and the Task Tag fields of outbound UPIUs will be analyzed by the UTP Engine in order to enable matching future corresponding inbound UPIUs. Note that for QUERY REQUEST and NOP OUT UPIUs the LUN field is reserved and will not be used for matching. The specific UTP Engine behavior is detailed in Table 1.

Table 1 — Outbound UPIUs Generated by Software

UPIU Type	Relevant UPIU Fields	UTP Engine actions on UPIU fields
Command	EXT_IID[(1)], IID, LUN, Task Tag	To match future incoming Response
Task Management Request		To match future incoming Task Management Response
Query Request	Task Tag	To match future incoming Query Response
NOP Out		To match future incoming NOP In
NOTE 1 EXT_IID is used when MCQCAP.EIS is '1', dExtendedUFSFeaturesSupport.EXT_IID in the device is '1', and bEXTIIDen in the UFS device is enabled by the host.		

7.2.2.2 Outbound UPIU Generated by Host Controller/UTP Engine

Only DATA OUT UPIU s are automatically generated by the UTP Engine without any involvement of software. A DATA OUT UPIU is created in response to incoming Ready To Transfer (RTT) UPIU from the Device which in turn resulted from a prior transmission of Command UPIU towards the Device. The UTP engine uses information from the corresponding RTT UPIU and from the PRD Table associated with the original Command UPIU. The specific UTP Engine behavior is detailed in Table 2.

7.2.2.2 Outbound UPIU Generated by Host Controller/UTP Engine (cont'd)

Table 2 — Outbound UPIUs Generated by UTP Engine

UPIU Type	Relevant UPIU Fields	UTP Engine actions on UPIU fields
Data Out	Transaction Type	Set to xx00 0010b
	EXT_IID(1), IID, LUN, Task Tag	Used to identify the corresponding Ready To Transfer UPIU.
	Total EHS Length	Always set to '0'
	Data Segment Length	Same value as Data Transfer Count (see below)
	Data Buffer Offset, Data Transfer Count	Filled with Data Segment Offset and Data Buffer Count information from the corresponding to Ready To Transfer UPIU
	Data Segment	Fetched from PRDT buffers based on the DMA context information supplied in the corresponding RTT UPIU
NOTE 1 EXT_IID is used when MCQCAP.EIS is '1', dExtendedUFSFeaturesSupport.EXT_IID in the device is '1', and bEXTIIDen in the UFS device is enabled by the host.		

7.2.2.3 Inbound UPIUs Interpreted by Software

The UTP Engine shall analyze the EXT_IID, IID, Task Tag and -in case of transaction types where applicable- LUN fields of all UPIUs the UFS Host receives so that they can be matched to UPIUs previously sent from the UFS Host towards the UFS Device (Request/Response matching)

UPIUs received from the UFS Device with EXT_IID, IID, Task Tag and -where applicable- LUN fields matching those of a UPIU previously sent by the UFS Host will be transferred into the respective Descriptor in Host memory. Only Ready To Transfer (RTT) and DATA IN UPIUs are handled differently and shall be analyzed further by the UTP Engine to enable autonomous data transfer operations as described in 7.2.2.4.

For the Host Controller the reception of an inbound UPIU of the types listed in Table 3 will finally close a UTP transaction that was started when the corresponding outbound UPIU was sent. Ultimately, it results in clearing the corresponding bit in the doorbell register UTRLDBR for NOP, Transfer Request or Query transactions and UTMRLDBR for Task Management transactions. Further analysis of the content of the listed inbound UPIUs will be only done by software and is transparent to the Host Controller.

Table 3 — Inbound UPIUs Consumed by Software

UPIU Type	Relevant UPIU Fields	UTP Engine actions on UPIU fields
Response	EXT_IID(1), IID, LUN, Task Tag	Identify corresponding Command
Task Management Response		Identify corresponding Task Management Request
Query Response	Task Tag	Identify corresponding Query Request
NOP In		Identify corresponding NOP Out
NOTE 1 EXT_IID is used when MCQCAP.EIS is '1', dExtendedUFSFeaturesSupport.EXT_IID in the device is '1', and bEXTIIDen in the UFS device is enabled by the host.		

7.2.2.4 Inbound UPIUs # by Host Controller/UTP Engine

The Data In and Ready To Transfer UPIUs are handled entirely by the Host Controller/UTP Engine and software is not involved when processing them. DATA IN UPIUs carry data retrieved from the UFS Device and their header information is parsed to allow the Host Controller to transfer the contained data to the correct location in Host Memory.

Table 4 — Inbound DATA IN UPIU Handled by UTP Engine

UPIU Type	Relevant UPIU Fields	UTP Engine actions on UPIU fields
Data In	Transaction Type	Matched against xx10 0010b
	EXT_IID(1), IID, LUN, Task Tag	Used to identify the corresponding Command which created the current transaction this DATA IN UPIU belongs to.
	Data Segment Length	Same value as Data Buffer Count value (see below).
	Data Buffer Offset, Data Transfer Count	Used to identify the correct Host memory location and DMA write transfer length (in conjunction with PRD Table info)
	Data Segment	Data to be transferred to Host memory by the Host Controller.
NOTE 1 EXT_IID is used when MCQCAP.EIS is '1', dExtendedUFSFeaturesSupport.EXT_IID in the device is '1', and bEXTIIDen in the UFS device is enabled by the host.		

Ready To Transfer UPIUs are analyzed by the Host Controller/UTP Engine to extract the necessary information provided by the UFS Device about the next DATA OUT UPIU the UTP Engine is expected to generate.

Table 5 — Inbound RTT UPIU Handled by UTP Engine

UPIU Type	Relevant UPIU Fields	UTP Engine actions on UPIU fields
Ready To Transfer	Transaction Type	Matched against xx11 0001b.
	EXT_IID(1), IID, LUN, Task Tag	Used to identify the corresponding Command which created the current transaction this RTT UPIU belongs to.
	Total EHS Length	Always '0'
	Data Segment Length	Always '0'
	Data Buffer Offset, Data Transfer Count	Used to identify the correct Host memory location and DMA read transfer length (in conjunction with PRD Table info) to compose the requested DATA OUT UPIU.
NOTE 1 EXT_IID is used when MCQCAP.EIS is '1', dExtendedUFSFeaturesSupport.EXT_IID in the device is '1', and bEXTIIDen in the UFS device is enabled by the host.		

7.2.3 Processing UTP Transfer Request Completion

When a UTP Transfer Request Completion is received in the host controller, it is handled as follows:
If the completion is for a Regular Command (UTRD.I = 0):

1. The IA interrupt counter is incremented.
2. If the counter was 0 before incrementing, the IA timer starts running.

IS.UTRCS bit is set when at least one of the following four conditions is met:

1. The UTRD.I bit is set (Interrupt Command)
2. The counter, after incrementing, reaches the value configured in IACTH
3. The IA timer reaches the value configured in IATOVAL (this event may occur at any time, not necessarily coupled with request completion).
4. Overall command Status (OCS) of the completed command is not equal to “SUCCESS”.

An interrupt is generated by the write operation if the completion interrupt is not masked (disabled) by the IE.UTRCE bit.

Host software processes the interrupt generated by host controller for command completion. In the interrupt service routine, host software checks register **IS** to determine if there is an interrupt pending. If IS.UTRCS bit is set, indicating that one or more UTP Transfer Requests (TR) have completed, the following procedure is recommended:

1) If there were errors, noted in the **IS** register, host software performs error recovery actions.
2) In the case of IS.UTRCS interrupt, host software clears the interrupt and then may use one of two methods to determine which UTP TRs have completed:

   Method A: Read the **UTRLDBR** register and compare the current value to the list of commands previously issued by host software that are still outstanding. For any TR which is outstanding, a value of 0 in bit *i* (where *i* is the UTRL slot through which the TR is issued) of UTRLDBR means that the TR has completed. **UTRLDBR** is a volatile register; software should only use its value to determine commands that have completed, not to determine which commands have previously been issued.

   Method B: Read the **UTRLCNR** register. For any TR, a value of 1 in bit *i* (where *i* is the UTRL slot through which the TR is issued) of UTRLCNR means that the TR has completed.
3) For every TR *i* whose completion is detected, software repeats the following actions:

   A. Processes the request completion as required by higher OS layers (e.g. file system).

   B. Clears bit *i* of UTRLCNR, by writing ‘1’ to it.

   C. Marks slot *i* as available for reuse (software only).
4) After processing all previously detected TRs, software may reset and restart Interrupt Aggregation mechanism by writing 80010000h to the UTRIACR register.
5) Software determines if new TRs have completed since step 2), by repeating one of the two methods described in step 2). If new TRs have completed, software repeats the sequence from step 3).

7.3 Task Management Function

UTMRL is used to send a UTP Task Management Function to the attached device. Host controller shall place higher priority on request for task management function over a UTP Transfer Request. When a task management request is submitted, host controller is required to suspend any active UTP transfer requests and switch to dispatch the task management to the attached device. The granularity of the context switching should happen at the transfer of the boundary of UPIU.

7.3.1 Basic Steps When Building a UTP Task Management Request

When host software needs to send a UTP Task Management function to host controller, it utilizes *UTP Task Management Request List*. The following is the list of steps for host software to build a UTP Task Management Request.

1) Find an empty transfer request slot by reading the **UTMRLDBR**. An empty transfer request slot has its respective bit cleared to '0' in the **UTMRLDBR**.
2) Host software builds a UTMRD at the empty slot.
3) **I** (interrupt) bit set if software requests **IS.UTMRCE** to be set on command completion.
4) Set **OCS** to 'Fh'.
5) Program field *Task Management Request UPIU*.
6) Initialize field *Task Management Response UPIU* with '0'.
7) Repeat step 1) to step 7) for each task management function to be sent to host controller for execution.
8) Check register **UTMRLRSR** and make sure it is read '1' before continuing.
9) Set *UTP Task Management Request Completion Enable* (UTMRCE) enable bit to '1' to enable the interrupt.
10) Set **UTMRLDBR** to ring the doorbell register to indicate to the host controller that multiple transfer requests are ready to be sent to the attached device. Host software shall only write a '1' to the bit position that corresponding to the new command; All other bit positions within **UTMRLDBR** should be written with a '0', which indicates no change to their current values.

7.3.2 Processing UTP Task Management Completion

Host software processes the interrupt generated by host controller for command completion. In the interrupt service routine, host software checks **IS** to determine if there is an interrupt pending. If the UFSHCI has an interrupt pending:

1. Host software determines the cause of the interrupt by reading the **IS** register. If the IS.UTMRCS bit is set this indicates that a UTP Task Management Request has completed.
2. Host software clears appropriate bits in the **IS** register corresponding to the cause of the interrupt.
3. Host software reads the **UTMRLDBR** register, and compares the current value to the list of commands previously issued by host software that are still outstanding. Host software completes with success any outstanding command whose corresponding bit has been cleared in the respective register. **UTMRLDBR** is a volatile register; software should only use its value to determine commands that have completed, not to determine which commands have previously been issued.
4. If there were errors, noted in the **IS** register, host software performs error recovery actions.

7.4 UIC Power Mode Change

The UIC power mode change is performed using the DME_SET command to set UIC attributes. The UIC power mode change is an atomic operation, which means that these attributes need to be set according to certain rules as described subsequently in this sub-clause.

One or more of the following attributes can be set as necessary:

- PA_ActiveTxDataLanes
- PA_ActiveRxDataLanes
- PA_TxGear
- PA_RxGear
- PA_TxTermination
- PA_RxTermination
- PA_HSSeries
- PA_PWRModeUserData
- PA_TxHsAdaptType
- DME_ Local_FC0ProtectionTimeOutVal
- DME_ Local_TC0ReplayTimeOutVal
- DME_ Local_ AFC0ReqTimeOutVal
- DME_ Local_FC1ProtectionTimeOutVal
- DME_ Local_TC1ReplayTimeOutVal
- DME_ Local_ AFC1ReqTimeOutVal
- PA_PWRMode

Currently, UFS uses TC0 only. Therefore, setting the following values are not needed:

- DME_ Local_FC1ProtectionTimeOutVal
- DME_ Local_TC1ReplayTimeOutVal
- DME_ Local_ AFC1ReqTimeOutVal

Setting the PA_PWRMode attribute triggers the UIC power mode change. Therefore, PA_PWRMode shall be the last attribute to be set. The other attributes can be set in any order before PA_PWRMode.

The UIC power mode change operation completes when IS.UPMS is set to ‘1’. If the IE.UPMSE is set, the UIC power mode change completion is also signaled by an interrupt. The status of the UIC power mode change is given by the value of HCS.UPMCRS. If HCS.UPMCRS is set to PWR_LOCAL, the UIC power mode change has completed with success. If HCS.UPMCRS is set to PWR_ERROR_CAP or PWR_FATAL_ERROR, the UIC power mode change has failed. HCS.UPMCRS cannot be set to PWR_REMOTE or PWR_BUSY, as these are associated to UIC power mode change requests from the UFS Device, which are not permitted [UFS].

To guarantee the atomicity of the UIC power mode change operation, between the PA_PWRMode attribute is set with DME_SET and the Host Controller Interface sets IS.UPMS, the software shall not set the attributes listed above. Otherwise, the behavior is undefined as the Host Controller Interface is not required to prevent these attributes from being set during the UIC power mode change.

7.4 UIC Power Mode Change (cont'd)

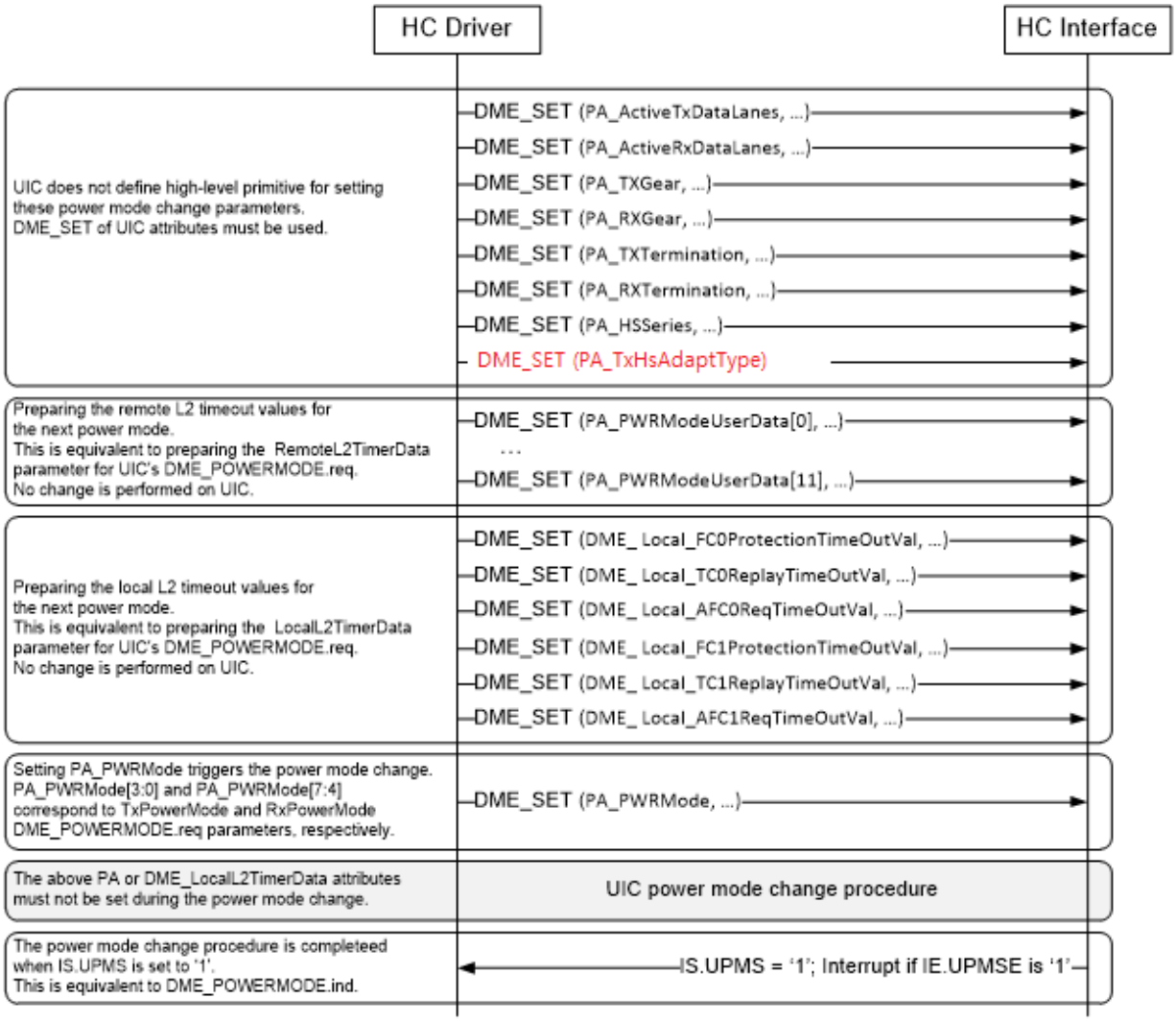


Figure 24 — UIC Power Mode Change

7.4.1 Adapt

The use of Adapt isn't mandatory but the specification provides some guidelines on its use.

The HCI should perform an Initial Adapt in the following cases if the link is running at HS-G4 or higher:

- If DME_RESET is initiated.
- If an unused line is activated for HS-G4 or higher.
- If UECDME.EC is triggered with bit 3 set to '1'.
- If a change between Rate A and Rate B in HS-G4 or higher is performed.

7.4.1 Adapt (cont'd)

The HCI should perform a Refresh Adapt in the following cases:

- If UECDME.EC is triggered with bit 1 set to '1'.
- If UECDME.EC is triggered with bit 2 set to '1'.

The Adapt is always performed in conjunction with Power Mode Change. During the Power Mode Change (and hence for the whole duration of the Adapt sequence) there will be no data traffic on the link.

7.5 UFSHCI Internal Rules

7.5.1 Command Processing Order

UFSHCI processes three types of commands, which shall be processed using the following priority (in order of priority, highest priority first):

- UIC Commands issued via the UIC Command Registers.
- Task Management Requests issued via UTP Task Management Request List.
- Transfer Requests issued via the UTP Transfer Request List.

Software shall issue UIC Commands one at a time. For Task Management Requests and Transfer Requests, software may issue multiple commands at a time, and may issue new commands before previous commands have completed. When software sets the corresponding doorbell register, the Task Management Requests and Transfer Requests automatically get a time stamp with their issue time. The commands within a command list (Task Management List or Transfer Request List) shall be processed in the order of their time stamps, starting from the oldest time stamp. In the case multiple commands from the same list have the same time stamp, they shall be processed in the order of their command list index, starting from the lowest index.

The UPIUs associated with Task Management Requests and Transfer Requests are sequentialized due to the use of a single UIC CPort. Therefore, the UPIUs associated with these Requests are transmitted one at a time.

In addition to the Request UPIUs provided by software (NOP Out, Command, Task Management Request and Query Request), UFS HCI also generates DATA OUT UPIUs. The order of the DATA OUT UPIUs and their interleaving with the Request UPIUs is implementation dependent, and is therefore not specified.

7.5.1 Command Processing Order (cont'd)

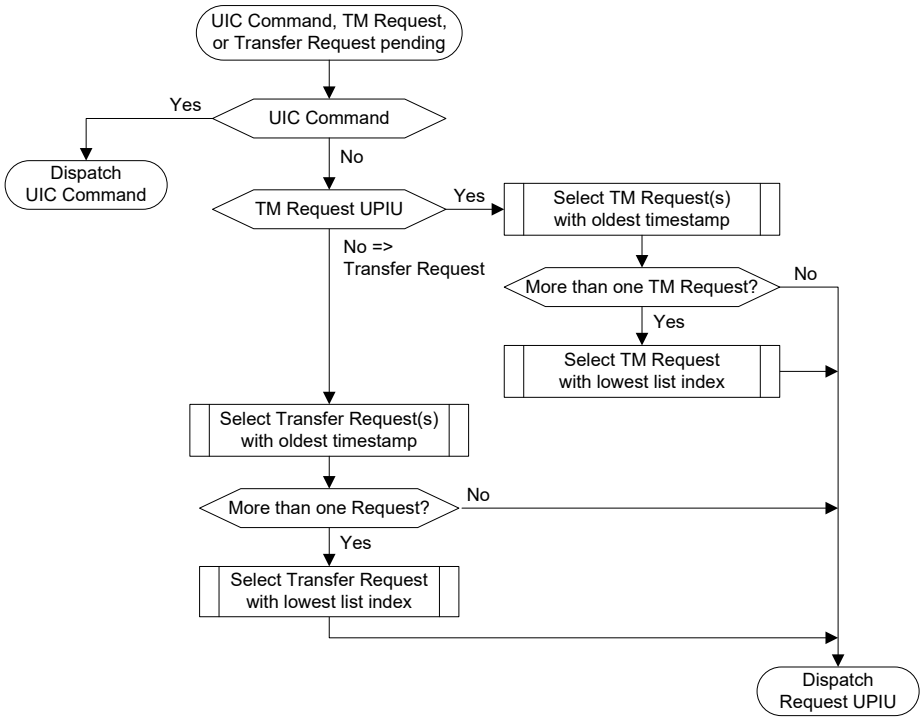


Figure 25 — Command Processing Order

7.5.2 RTT Processing Rules

The host controller shall process the pending RTTs in-order they were received from the device.

7.5.3 Data Unit Processing Order for Cryptographic Operations

The byte order in the Data Unit input of Crypto Engine and the byte order for Data Segment of UPIU shall be maintained as shown in Figure 26 for both encryption and decryption.

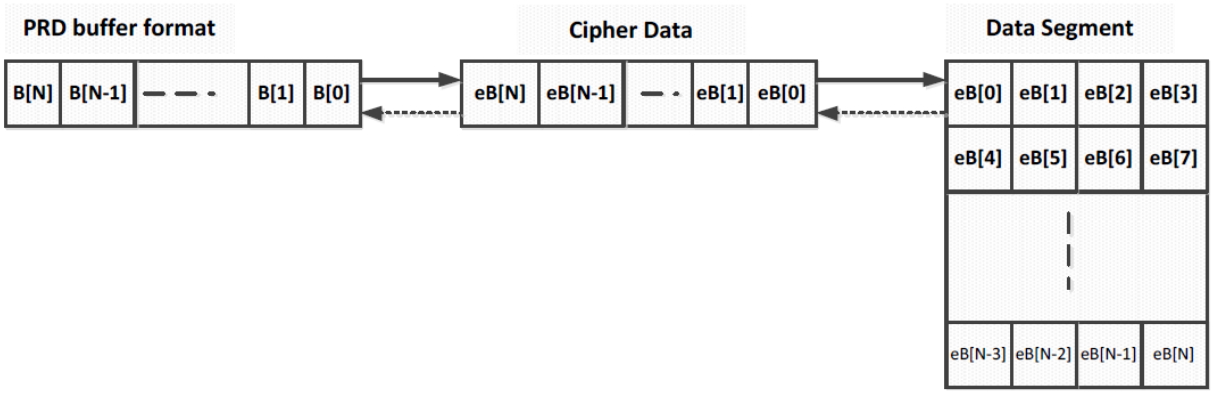


Figure 26 — Byte Order for Data Unit Processing in Cryptographic Operations

7.6 TX Equalization Configuration

To support HS-G6, MIPI UniPro3.0 introduced new attributes, Equalization setting and Pre-Coding Enable (PA_TxEQGnSetting, PA_PreCodeEn). It is Host Software's responsibility to configure these attributes for both host and device before initiating Power Mode Change to HS-G6.

MIPI UniPro3.0 also introduced TX Equalization Training (EQTR) to identify optimal TX Equalization settings for use by both Host's and Device's UniPro. TX EQTR shall be initiated from the most reliable High-Speed Gear (HS-G1) targeting High-Speed Gears (HS-G4 to HS-G6).

The TX EQTR procedure is defined for only one set of TX Equalization setting. Therefore, Host Software can identify the optimum TX Equalization setting after running TX EQTR procedure over multiple sets of TX Equalization setting. All the following pre-conditions shall be met before starting the Link EQTR procedure:

- UFS device supports Link EQTR, i.e., UniPro Version (in UniPro PA_RemoteVerInfo attribute) is greater than or equal to 7.
- All connected Lanes are activated.
- The target Series (A or B) is used.

The TX EQTR procedure is atomic, which means that these attributes need to be set according to certain rules as described subsequently in this sub-clause.

One or more of the following attributes shall be set for TX EQTR procedure as necessary:

- PA_TxEQTRSetting
- PA_PeerTxEQTRSetting
- PA_TxAdaptLength_EQTR
- PA_EQTR_Gear

A TX EQTR procedure is triggered by setting the UFS Host PA_EQTR_Gear attribute. Therefore, PA_EQTR_Gear shall be the last attribute to be set. The other attributes can be set in any order before PA_EQTR_Gear.

A TX EQTR procedure is completed when IS.UPMS is set to '1'. If the IE.UPMSE is set, a TX EQTR procedure completion is also signaled by an interrupt. HCS.UPMCRS indicates the status of the TX EQTR. If HCS.UPMCRS is set to PWR_LOCAL, the TX EQTR has completed with success. If HCS.UPMCRS is set to PWR_ERROR_CAP or PWR_FATAL_ERROR, the TX EQTR has failed. HCS.UPMCRS cannot be set to PWR_REMOTE or PWR_BUSY as a result of Link EQTR, since these are associated to TX EQTR requests from the UFS Device, which are not permitted in JESD220H.

After setting PA_EQTR_Gear, the Host Software should wait until IS.UPMS is set to 1 before setting any PA Layer Attribute and before initiating DME_PEER_SET.req and DME_PEER_GET.req. Otherwise, the behavior is undefined as the Host Controller Interface is not required to prevent these attributes from being set during the TX EQTR procedure.

7.6 TX Equalization Configuration (cont'd)

Following is a sequence of the operations that host software would perform to conduct TX EQTR and configure TX Equalization and Pre-Coding settings for target high speed Gear **n**:

1) Determine the TX Adapt Length for TX EQTR by setting PA_TxAdaptLength_EQTR attribute. The same TX Adapt Length is used for both UFS Host and UFS Device. Note that Host Software only needs to set UFS Host PA_TxAdaptLength_EQTR attribute, it is sent to the UFS Device when TX EQTR is activated. Select the minimum Adapt Length for the TX EQTR procedure based on the following conditions:
   a. If the target High-Speed Gear **n** is HS-G4 or HS-G5:
      - PA_TxAdaptLength_EQTR[7:0] ≥ Max (10 µs, RX_HS_Gn_ADAPT_INITIAL_Capability, PA_PeerRxHsGnAdaptInitial)
      - PA_TxAdaptLength_EQTR[7:0] shall be shorter than PACP_REQUEST_TIMER value of 10 ms
      - PA_TxAdaptLength_EQTR[15:8] is not relevant for HS-G4 and HS-G5. This field is set to 255 (reserved value).
   b. If the target High-Speed Gear **n** is HS-G6:
      - PA_TxAdaptLength_EQTR ≥ 10 µs
      - PA_TxAdaptLength_EQTR[7:0] ≥ Max (RX_HS_G6_ADAPT_INITIAL_Capability, PA_PeerRxHsG6AdaptInitialL0L3)
      - PA_TxAdaptLength_EQTR[15:8] ≥ Max (RX_HS_G6_ADAPT_INITIAL_L0_L1_L2_L3_Capability, PA_PeerRxHsG6AdaptInitialL0L1L2L3)
      - PA_TxAdaptLength_EQTR shall be shorter than PACP_REQUEST_TIMER value of 10 ms.
2) Determine TX Equalization setting combination count **M**.
   a. Get UFS Host Equalization capabilities from UFS Host TX_HS_PreShoot_Setting_Capability and TX_HS_DeEmphasis_Setting_Capability attributes.
   b. Get UFS Device Equalization capabilities from UFS Host PA_PeerTxEQCap attribute.
3) Apply TX Equalization setting combination **k** (**k ≤ M**) for TX EQTR procedure by setting UFS Host PA_TxEQTRSetting and PA_PeerTxEQTRSetting attributes for all active lanes.
4) Trigger TX EQTR procedure by setting the UFS Host PA_EQTR_Gear attribute to target High-Speed Gear **n** and wait for its completion.
5) For each active lane, read RX_FOM attributes in both UFS Host and UFS Device.
6) Repeat step 3) to step 5) to find out the TX Equalization setting combination which yields the highest Figure of Merit (FOM) values in both UFS Host and UFS Device.
7) Apply the TX Equalization setting combination in step 6) by setting PA_TxEQG**n**Setting attributes in both UFS Host and UFS Device, where **n** is the target High-Speed Gear **n**.

7.6 TX Equalization Configuration (cont'd)

8) Configure PA_PreCodeEn for both UFS Host and UFS Device. Host Software can skip this step if the target High-Speed Gear is HS-G5 or lower.

   a. Check UFS Host RX_FOM[7] for each active lane; only for the lanes on which UFS Host needs to enable Pre-Coding (RX_FOM[7] is set), Host Software should set PreCodeEn for RX in UFS Host PA_PreCodeEn attribute and PreCodeEn for TX in UFS Device PA_PreCodeEn attribute.

   b. Check UFS Device RX_FOM[7] for each active lane; only for the lanes on which UFS Device receiver needs to enable Pre-Coding (RX_FOM[7] is set), Host Software should set PreCodeEn for RX in UFS Device PA_PreCodeEn attribute and PreCodeEn for TX in UFS Host PA_PreCodeEn attribute.

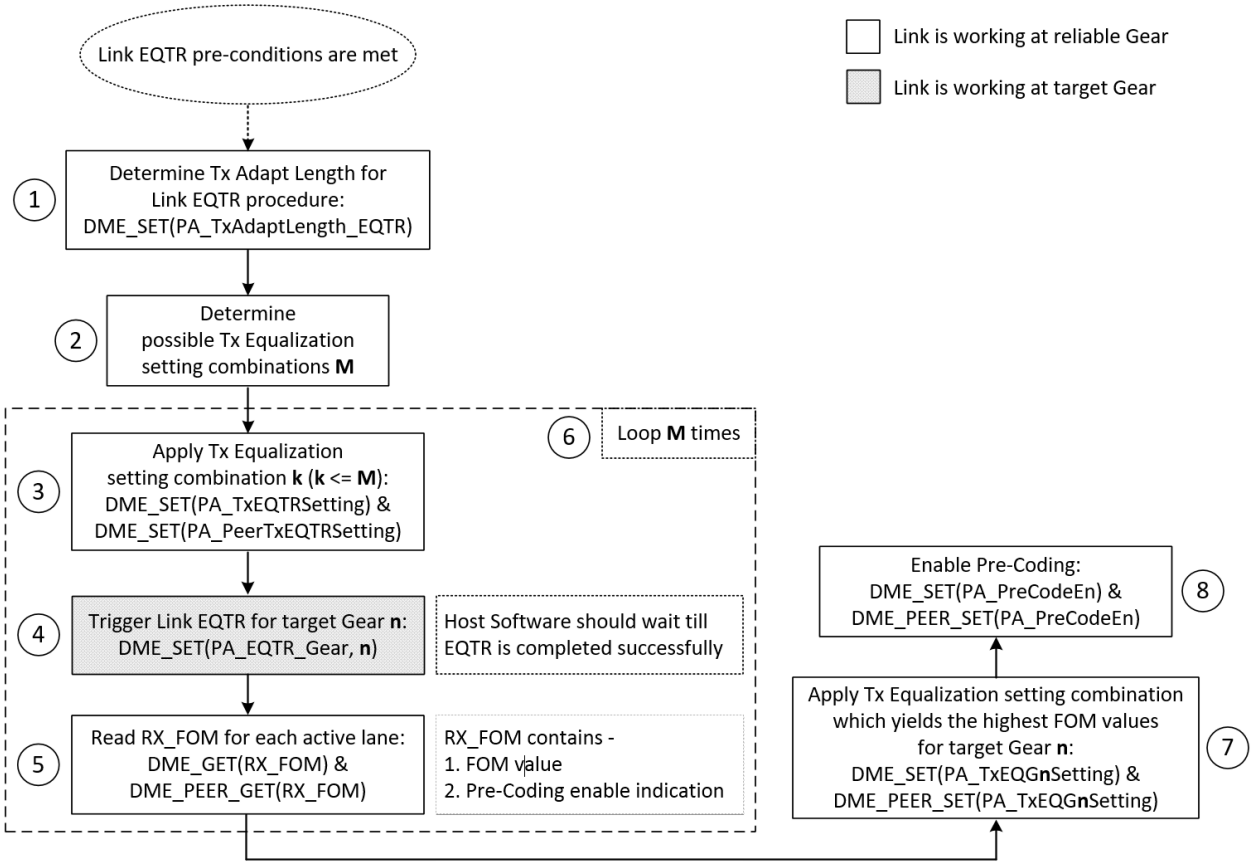


Figure 27 — Link EQTR and Tx Equalization Configuration Flow

8 Error Reporting and Handling

This clause describes the various errors that UFS host controller can report and how they are handled.

NOTE When device is reset using Power-on reset or Hardware reset the power-on write protection bit in the device gets cleared.

8.1 Error Types

8.1.1 System Bus Error

There are several sources of errors that could occur on the system bus:

- Bad memory pointers. This occurs when host software provides an invalid memory pointer.
- Bad operation. This occurs when host controller tries to write to read-only memory.
- Protection violation. This occurs when host controller tries to read/write to protected memory without proper privilege.

All these errors are not recoverable. When one of them occurs, host controller shall stop accepting new requests and report the error by setting register **IS.SBFES**. Host controller optionally generates an interrupt if **IE.SBFEE** is set.

8.1.2 UIC Error

UIC errors occur within UIC layers of host controller. When a UIC error occurs, host controller shall abort the request and report the error by setting register **IS.UE**. Host controller optionally generates an interrupt if **IE.UEE** is set. Host software shall check all of the following registers for the cause of the errors:

- **UECPA** for Host UIC Error Code within PHY Adapter Layer.
- **UECDL** for Host UIC Error Code within Data Link Layer.
- **UECN** for Host UIC Error Code within Network Layer.
- **UECT** for Host UIC Error Code within Transport Layer.
- **UECDME** for Host UIC Error Code within DME subcomponent.

Refer to 5.3.6 through 5.3.10 in 5.3 for the error codes of each UIC error register.

8.1.3 UIC Command Error

UIC Command errors are indicated by a non-zero ConfigResultCode or GenericErrorCode field in the **UICCMDARG2** register after the UIC Command has completed.

These errors are generally recoverable by software.

8.1.4 UTP Error

There are two mechanisms that host controller uses to report a UTP error:

- *UTP Transfer Request*. When a UTP transfer request completes (success or failure as indicated by **IS.UTRCS**), host software shall check field **OCS** of UTRD for the request just completed. The field contains the status code for the request. Host software may need to check Transfer Response UPIU's Response, and possibly Sense Data to find out more details for the cause of the error. The host controller does not halt for non-fatal error conditions.
- *UTP Task Management Request*. When a UTP task management request completes (success or failure as indicated by **IS.UTMRCS**), host software shall check OCS of UTMRD for request status. Host software may need to check Task Management Response UPIU to find out more details for the cause of the error. The host controller does not halt for non-fatal error conditions.

8.1.5 Host Controller Fatal Error

Within host controller errors could occur outside of system bus and UIC subcomponent. These errors are reported as host controller fatal errors. When a host controller fatal error is detected, the host controller shall set the **IS.HCFES** bit. If **IE.HCFEE** is set, the host controller shall generate an interrupt.

8.1.6 Device Error

Device errors are the fatal error conditions that prevent the device from completing any request. They apply the entire device. When a fatal device error is detected, the host controller shall set the **IS.DFES** bit. If **IE.DFEE** is set, the host controller shall generate an interrupt. Before setting the **IS.DFES** bit, the host controller shall perform the following steps:

1) Clear UTRLRSR.
2) Clear UTMRLRSR.
3) Clear **HCS.UTRLRDY.**
4) Clear **HCS.UTMRLRDY**.
5) Abort all outstanding UTP transfer requests.
6) For each of the outstanding request, update the **OCS** field of the UTRD corresponding request with DEVICE FATAL ERROR to indicate the reason for the UTP Transfer Request abortion.
7) Abort all outstanding UTP task management requests. For each of the outstanding request, update the **OCS** field of the UTMRD corresponding request with DEVICE FATAL ERROR to indicate the reason for the UTP Task Management Request abortion.

It is recommended to recover the error in the following order:

1) Send DME_ENDPOINT_RESET command to the device.
2) Perform a host controller and device power cycle or hardware reset.

8.1.7 Hibernate Enter/Exit Error

Hibernate Enter/Exit errors occur when the host controller encounters error during UniPro hibernate entering/exiting process. The errors may occur either by manual-hibernate entering/exiting requested explicitly by host software through UIC command or auto-hibernate entering/exiting triggered by **AHIT** register.

When a hibernate enter/exit error occurs, the host controller shall set the **IS.UHES** (error during hibernate entering) bit or **IS.UHXS** (error during hibernate exiting) bit. The host controller shall also set **HCS.UPMCRS** register with PWR_BUSY, PWR_ERROR_CAP, or PWR_FATAL_ERROR, depending on the error status.

8.2 Error Handling

8.2.1 System Bus Error Handling

System Bus Errors are serious errors and cannot be recovered. Whenever System Bus Errors occur, host controller shall use the following procedure to recover:

1) Stop receiving new request by clearing **UTRLRSR** and **UTMRLRSR** bits to ‘0’.
2) Assert **IS.SBFES.**

Host software shall use following order to recover from system bus error:

1) Send DME_ENDPOINT_RESET command to the device.
2) Reset the host controller by setting register HCE to ‘0’ and re-initializing host by setting register HCE to ‘1’.

These steps ensure that Host Controller and Device are brought into a known state after encountering System Bus error. Optionally, Host and Device may undergo power cycle if systems are not able to recover with the above procedure.

The standard does not define ways for the System to recover from such errors as it is out of scope for this specification.

8.2.2 UIC Error Handling

The following are the rules for handling UIC errors:

- Except for PA_INIT_ERROR, all other errors indicated in the UECPA and UECDL registers are non-fatal error. UIC recovers by itself.
- Errors indicated in the UECN, UECT and UECDME registers need software intervention. UIC doesn't need to be reset.
- PA_INIT_ERROR indicated in the UECDL register is fatal should lead to a UIC reset as shown below.

When a fatal UIC error occurs, host software shall follow the steps below to recover:

1) Reads **UTRLDBR** to determine which requests are completed.
2) Checks **OCS** field of the *UTRD* for each request completed to determine if there was an error with that request, and if so, the host software shall ignore the data in the system memory locations for requests that complete with such error conditions.
3) Save to a list of outstanding and failed request so that they can be optionally re-issue them after host controller reset.
4) Reset the controller by setting register HCE to ‘0’.
5) Wait until HCE is read as ‘0’.
6) Re-initialize the host controller by setting register HCE to ‘1’.
7) Optionally host software can re-issue the saved requests to the host controller.

8.2.3 UIC Command Error Handling

In case the UIC Command Error is caused by DME_GET, DME_SET, DME_PEER_GET or DME_PEER_SET, the ConfigResultCode field in the **UICCMDARG2** register carries the cause of the error. This is either an incorrect set of UIC Command parameters or UIC state when UIC cannot be configured. In the case of DME_PEER_GET and DME_PEER_SET, the UIC link may also be in a state where it cannot transfer the UIC configuration command to the Device.

The host software should reissue the UIC Command with a correct set of parameters when the UIC is in a state where it can accept UIC Commands.

8.2.4 UTP Error Handling

8.2.4.1 UTP Transfer Request Error Handling

The following is a flow that host software shall use to recover from an error within UTP Transfer Request List:

1) Reads **UTRLDBR** to determine which requests are still outstanding.
2) Checks **OCS** field of the *UTRD* for each request completed to determine if there was an error with that request, and if so the error condition.
3) Clear **IS.UTRCS** to '0'.
4) Host software then either completes the request that had the error and requests still outstanding with error to higher level software, or re-issues these requests to the host controller.

8.2.4.2 UTP Task Management Request Error Handling

The following is a flow that host software shall use to recover from an error within UTP Task Management Request List:

1) Reads **UTMRLDBR** to determine which requests are still outstanding.
2) Checks **OCS** field of the *UTMRD* for each request completed to determine if there was an error with that request, and if so the error condition.
3) Clear **IS.UTMRCS** to '0'.
4) Host software then either completes the request that had the error and requests still outstanding with error to higher level software, or re-issues these requests to the host controller.

8.2.5 Host Controller Error Handling

Host Controller Errors are fatal errors. When this condition occurs, host software should reset by setting register HCE to '0', wait until HCE is read as '0', and then re-initialize the host controller by setting register HCE to '1'.

8.2.6 Device Error Handling

Device Errors are fatal errors. When this condition occurs, host software shall follow the same procedure for UIC error handling as described in 8.2.2, except that in addition to resetting UIC, the host software shall reset the device too.

8.2.7 Hibernate Enter/Exit Error Handling

Hibernate Enter/Exit Error occurs when the UniPro link is broken. When this condition occurs, host software should reset the host controller by setting register **HCE** to '0', re-initialize the host controller by setting register **HCE** to '1', and then start link startup sequence as shown in Figure 23.

9 (Informative) Encryption Engine Details

The algorithms described in this clause are defined in their respective standard specifications, which are beyond the scope of this document. The usage of each algorithm as it is referred to in this clause may be constrained compared to the definition in its standard specification. This clause specifies the limitations and constrains applicable to cryptographic operations in a UFS host controller.

In addition, the terminology used in this specification might differ from the algorithm specifications. This clause clarifies the terms used, and maps the usage of the transaction parameters to each specific algorithm.

9.1 AES-XTS

9.1.1 Overview

The AES-XTS algorithm is defined in IEEE standard 1619-2007 for cryptographic protection of data on block-oriented storage devices. It is a tweakable block cipher that acts on data units of 128 bits or more, using AES as the underlying block cipher.

Although the AES-XTS specification specifies method for encrypting data unit sizes that are not multiple of 128 bits, UFS host encryption only operates on data unit sizes that are multiple of 128 bits. Therefore, only the portion of the algorithm that is relevant to the processing of data units multiple of 128 bits, and is referred to as XEX, is used.

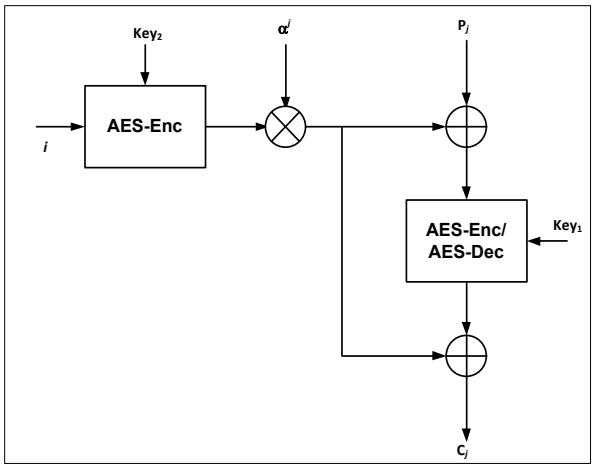


Figure 27 — AES-XTS Encryption

Figure 27 illustrates an overview of the encryption of a unit of data using the AES-XTS algorithm for data unit sizes that is multiple of 128 bits, AES-XEX.

9.1.2 Data Unit Size

The AES-XTS algorithm encrypts a data unit at a time. The Supported Data Unit Size Bitmask (SDUSB) field in the Crypto Capability specifies the data unit sizes permitted to be used when performing the encryption. The Data Unit Size (DUSIZE) field in the Crypto Configuration is used for selecting one of the permitted data unit sizes.

For example, to select a data unit size of 4096 bytes, bit 3 of DUSIZE field is set, and DUSIZE = 08h.

9.1.3 Tweak

One of the inputs to the encryption of a data unit is the tweak, "***i***" in the diagram above. This is a 128-bit value that is unique for a given data unit. In UFS host encryption engine, the Data Unit Number (DUN) of the data unit encrypted is used for generating the tweak as follows:

i = Data Unit Number (DUN) || 00000… (128 bits)

For the first data unit of a transaction, the Data Unit Number is initialized with the value of the DUN field of the UTRD. For subsequent data units, the offset Data Unit Number is recalculated at the start of every data unit.

For a different method of forming the tweak, a different Algorithm ID shall be specified and the formulation, as well as the method to calculate DUN, shall be clearly defined.

9.2 Microsoft Bitlocker™ AES-CBC

9.2.1 Background

Microsoft Bitlocker™ AES-CBC refers to the following encryption algorithms, all developed by Microsoft Corporation:

- 128-bit AES-CBC
- 256-bit AES-CBC
- 128-bit AES-CBC + Elephant diffuser
- 256-bit AES-CBC + Elephant diffuser

Elephant Diffuser algorithms were used in previous versions of Windows OS and may be supported for backward compatibility and decryption of existing volumes. It is recommended that new encryption of an unencrypted volume use the AES-CBC algorithms without Elephant Diffusers.

Complete information about the two AES-CBC algorithms is provided in the paper "*AES-CBC + Elephant diffuser; A Disk Encryption Algorithm for Windows Vista by Niels Ferguson, August 2006*" available on http://download.microsoft.com. Information about AES-CBC without Elephant Diffusers is provided in the subsequent sub-clauses.

9.2.2 Overview

The Microsoft Bitlocker™ AES-CBC algorithm takes the following inputs:

- Key *K*. This key is always 512 bits long.
- Sector size *S*. Supported sectors sizes are 512, 1024, 2048, and 4096 bytes.[1]
- Sector offset *O*.
- Sector data, containing *S* bytes.

Figure 28 illustrates the encryption process using Microsoft Bitlocker™ AES-CBC.

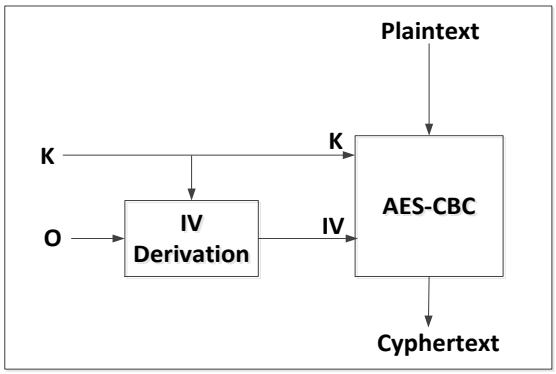


Figure 28 — Microsoft Bitlocker™ AES-CBC Encryption

9.2.3 Sector (Data Unit) Size S

The Microsoft Bitlocker™ algorithm refers to data units as sectors, encrypting a sector (data unit) at a time. The sector sizes supported by Microsoft Bitlocker™ specification are 512, 1024, 2048, and 4096 bytes.

The Supported Data Unit Size Bitmask (SDUSB) field in the Crypto Capability specifies the data unit sizes permitted to be used when performing the encryption. The Data Unit Size (DUSIZE) field in the Crypto Configuration is used for selecting one of the permitted data unit sizes.

For example, to select a data unit size of 4096 bytes, bit 3 of DUSIZE field is set, and DUSIZE = 08h.

[1] Microsoft Bitlocker™ terminology uses the term “sector” where other algorithms use “data unit”. These terms have identical meanings and are used interchangeably

9.2.4 Sector Offset *O*

The sector offset *O* is the byte offset of the start of the sector within the volume that is being encrypted. For example, on a disk with 4096-byte sectors, the 3rd sector will have offset 12288. The offset *O* is a 64-bit number.

For the first sector of a transaction, the offset *O* is initialized with the value of the DUN field of the UTRD. For subsequent sectors, the offset *O* is recalculated at the start of every sector.

9.2.5 Sector Initialization Vector (IV)

A new IV is computed for every sector. The IV is 16 bytes (128 bits) long. It is constructed by creating a 128-bit value, writing the sector offset *O* into bits 63:00, and setting bits 127:64 to zeros, as illustrated in Figure 29. The constructed 128-bit value is then encrypted using AES with the key derived from *K*. The result of this encryption is the IV for the sector.

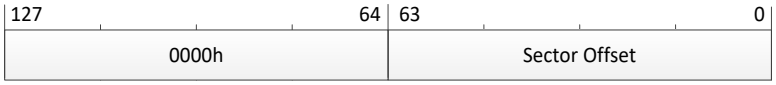


Figure 29 — IV Derivation from Sector Offset

9.2.6 Encryption / Decryption

The sector data is then encrypted using AES in CBC mode, with the computed IV. The AES key is the key derived from *K*. As the sector size is always a multiple of 16, there is no block padding.

Decryption is the obvious inverse. The same algorithm is used to compute the IV, and then the encrypted sector data is decrypted with AES-CBC and the computed IV.

9.3 AES-ECB

9.3.1 Overview

In AES-ECB mode, the plaintext is divided into a number of 128-bit blocks. Each block is then encrypted or decrypted in the AES encryption/decryption module. In AES-ECB mode, the only inputs to the encryption/decryption module are the plaintext data and the key, as shown in Figure 30.

The data unit number is not used.

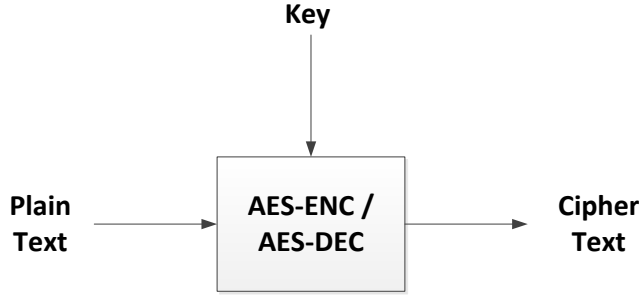


Figure 30 — AES-ECB Encryption

9.4 ESSIV-AES-CBC

9.4.1 Background

ESSIV-AES-CBC refers to the following encryption algorithms:

- 128-bit AES-CBC
- 256-bit AES-CBC

9.4.2 Data Unit Size

Supported data unit sizes are 512, 1024, 2048, and 4096 bytes.

9.4.3 Sector Number (SN)

In ESSIV-AES-CBC, a 64-bit Sector Number (SN) is generated from the LBA field, extracted from the Command Descriptor Block (CDB) clause of the COMMAND UPIU, as follows:

- READ6/WRITE6 (16b LBA): the LBA is written into bits 15:00; bits 63:16 are filled with zeros.
- READ10/WRITE10 (32b LBA): the LBA is written into bits 31:00; bits 63:32 are filled with zeros.
- READ16/WRITE16 (64b LBA): the LBA is written into bits 63:00.

9.4.4 Initialization Vector (IV)

The IV is computed for every sector. The IV is 16 bytes (128 bits) long. It is constructed by creating a 128-bit value, writing the SN into bits 63:00, and setting bits 127:64 to zeros. The constructed 128-bit value is then encrypted using AES with the key derived from *K* on which SHA-256 hash is computed. The result of this encryption is the IV for the sector, as follows:

$$(SN) = AES(SN)\ where\ s = SHA256(key)$$

9.4.5 Encryption / Decryption

The data unit is encrypted using AES in CBC mode, with the computed IV. The AES key is the key derived from K. As the data unit is always a multiple of 16, there is no block padding. Decryption is the obvious inverse. The same algorithm is used to compute the IV, and then the encrypted data unit is decrypted with AES-CBC and the computed IV.

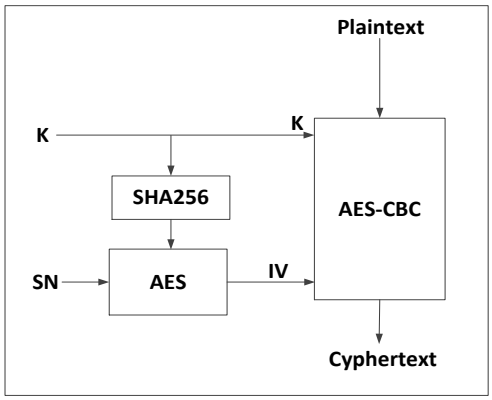


Figure 31 — ESSIV-AES-CBC Encryption

9.5 Inline Hashing

This feature defines a hardware-based inline hashing solution to address the problems of CPU consumption and scheduling overhead. The feature includes the following components:

- Cryptographic Hash Engine: The hash calculation is performed by a cryptographic hash engine located in the UFS host controller. This engine calculates cryptographic hash values such as SHA-256, which creates a 256-bit (32 byte) hash for each data block. The host controller advertises supported data block sizes through x-CRYPTOCAP, and the current data block size can be configured using x-CRYPTOCFG.

- Capability & Configuration: The feature adds cryptographic hashing algorithms to the x-CRYPTOCAP register. It also specifies the hash algorithm and chunk size using x-CRYPTOCFG. Specifically, the supported hash algorithms are SHA-256 and SHA-512.

- Locating hash values: The proposal uses a PRDT entry to pass either a normal data address or a hash value address. This approach keeps the size of the UTP transfer request descriptor data structure unchanged and requires the least hardware change to the UFS Host Controller design.

Annex A (Informative) Differences between Revisions

This annex briefly describes most of the changes made to entries that appear in this standard, JESD223G, compared to its predecessor, JESD223F (December 2024). If the change to a concept involves any words added or deleted (excluding deletion of accidentally repeated words), it is included. Some punctuation changes are not included.

A.1 Differences between JESD223G and JESD223F (December 2024)

- Upgraded MIPI UniPro standard [MIPI-UniPro] from version 2.0 to 3.0 (see clause 2)
- Upgraded MIPI M-PHY standard [MIPI-M-PHY] from version 5.0 to 6.0 (see 2)
- Upgraded UFS standard [UFS] from version 4.1 to 5.0 (see 2)
- Added support for HS-G6 speed (see 7.6)
- Added theory of operation for TX Equalization Configuration (see 7.6)
- Extended IS.UPMS to indicate a completion of Power Mode Change or TX Equalization Training (see 7.6)
- Extended HCS.UPMCRS to reflect a completion status of Power Mode Change or TX Equalization Training (see 7.6)
- Added Event Specific Interrupt Vector Base (ESIVB) (see 5.1 and 5.9.3)
- Added new Crypto algorithms SHA-256 and SHA-512 to support Inline Hashing (see 5.8.2 and 9.5)
- Updated 4DW PRDT format and 2DW PRDT format for Inline Hashing usage (see 9.5)

A.2 Differences between JESD223F and JESD223E (August 2022)

- UFS normative reference updated to Version 4.1 (see clause 2)
- UniPro 2.0's publication date updated (see clause 2)
- JEDEC JEP106 revision and publication date updated (see clause 2)
- Completion Queue Entry updated with I_T_L_Q information (see 6.2.2)
- Crypto Capability index range reduced to 127 total entries (see 5.8.1)
- Interrupt language refined for MCQ mode
- MCQ mode register numbers made consistent
- Clarification of minimum Submission and Completion Queue sizes (see 5.9.5.1)
- EHS length restriction reference to UFS standard's allowed length (see 5.2.1)
- Minor editorial clarifications were added, typographic errors were corrected, and minor formatting changes were made to comply with *Style Manual for Standards and Other Publications of JEDEC, JM7A*, in addition to what is summarized above.

A.3 Differences between JESD223E and JESD223D (January 2022)

- Upgraded MIPI UniPro standard [MIPI-UniPro] from version 1.8 to 2.0
- Upgraded MIPI M-PHY standard [MIPI-M-PHY] from version 4.0 to 5.0
- Added support of HS-G5 speed
- Added Multiple Circular Queue (MCQ) mode and its behavior definition
- Added MCQ registers (including MCQ configuration registers, SQ & CQ registers, MCQ operation & Runtime registers)
- Added data structures (including CQ entry) for MCQ mode
- Added NUTRS register definition for MCQ mode
- Revised OODDS register definition for hint information in OOO
- Added wHostHintCacheSize register for hint information in OOO
- Added AH8ITV register definition for MCQ mode
- Added EXT_IID to extend IID
- Added ResetMode for UICCMDARG1 to indicate link startup mode
- Revised the allowed value for Crypto Capabilities (CC) from 255 to 127
- Added Data structure for Physical Region Description Table (2DW Format)
- Revised UTP Transfer Request Descriptor format by adding LDBC and CDS field in DW2.
- Revised the description for Total EHS Length to allow non-zero value for COMMAND and RESPONSE UPIU in UTP Engine actions.

A.4 Differences between JESD223D and JESD223C (March 2016)

- Upgraded MIPI UniPro standard [MIPI-UniPro] from version 1.6 to 1.8.
- Upgraded MIPI M-PHY standard [MIPI-M-PHY] from version 3.0 to 4.1.
- Added support of HS-G4 speed with Adapt.
- Added support of QoS.
- Revised UTPEC register definition for more error status.

A.5 Differences between JESD223C and JESD223B (September 2013)

- Added UTP Transfer Request List Completion Notification Register
- Added cryptographic operation support
- Revised DFES error handling and SBFES error handling
- Revised HCE register behavior
- Revised AH8ITV definition and AH8 error handling
- Revised SBFES error handling
- Improved task completion sequence

A.6 Differences between JESD223B and JESD223A (June 2012)

- All references are revised to be consistence with UFS standard reference convention [UFS].
- Upgraded MIPI UniPro standard [MIPI-UniPro] from version 1.41 to 1.6.
- Upgraded MIPI MPHY standard [MIPI-M-PHY] from version 2.0 to 3.0.
- Added Auto-Hibernate support.
- Added Register for Unified Memory Extension
- Revised definition of version registers
- Revised definition of Manufacturer ID and Product ID
- Revised definition of Host Controller Enable register
- Added clarification to UTP Transfer Request Interrupt Aggregation Control Register
- Added RTT processing Rules.
- System bus error handling flow is revised.

This page intentionally left blank.

STANDARD IMPROVEMENT FORM **JEDEC** **JESD223G**

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 10th Street North
Suite 240S
Arlington, VA 22201

Email: angies@jedec.org

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®