

# **JEDEC STANDARD**

---

## **Universal Flash Storage Host Controller Interface (UFSHCI)**

**Version 3.0**

---

**JESD223D**

(Revision of JESD223C, March 2016)

**JANUARY 2018**

---

**JEDEC SOLID STATE TECHNOLOGY ASSOCIATION**



## NOTICE

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

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

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

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

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

Inquiries, comments, and suggestions relative to the content of this JEDEC standard or publication should be addressed to JEDEC at the address below, or refer to [www.jedec.org](http://www.jedec.org) under Standards and Documents for alternative contact information.

Published by  
©JEDEC Solid State Technology Association 2018  
3103 North 10th Street  
Suite 240 South  
Arlington, VA 22201-2107

JEDEC retains the copyright on this material. By downloading this file the individual agrees not to charge for or resell the resulting material.

**PRICE: Contact JEDEC**

Printed in the U.S.A.  
All rights reserved

PLEASE!

DON'T VIOLATE  
THE  
LAW!

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

For information, contact:

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

or refer to [www.jedec.org](http://www.jedec.org) under Standards-Documents/Copyright Information.



# UNIVERSAL FLASH STORAGE HOST CONTROLLER INTERFACE (UFSHCI)

## Contents

<b>1</b>	<b>Scope .....</b>	<b>1</b>
<b>2</b>	<b>Normative Reference .....</b>	<b>1</b>
<b>3</b>	<b>Acronyms, Terms and Definitions, Keywords, and Conventions.....</b>	<b>2</b>
<b>4</b>	<b>Architectural Overview .....</b>	<b>5</b>
4.1	Outside of Scope .....	5
4.2	Interface Architecture .....	6
4.3	Transfer Request Interface.....	8
4.4	Limitations.....	9
<b>5</b>	<b>UFS Host Controller Register Interface .....</b>	<b>9</b>
5.1	Register Map.....	10
5.2	Host Controller Capabilities Registers.....	11
5.2.1	Offset 00h: CAP – Controller Capabilities .....	11
5.2.2	Offset 08h: VER – UFS Version .....	12
5.2.3	Offset 10h: HCPID – Host Controller Identification Descriptor – Product ID.....	12
5.2.4	Offset 14h: HCMID – Host Controller Identification Descriptor – Manufacturer ID .....	12
5.2.5	Offset 18h: AHIT – Auto-Hibernate Idle Timer .....	13
5.3	Operation and Runtime Registers .....	14
5.3.1	Offset 20h: IS – Interrupt Status .....	14
5.3.2	Offset 24h: IE – Interrupt Enable.....	16
5.3.3	Offset 30h: HCS – Host Controller Status .....	17
5.3.4	Offset 34h: HCE – Host Controller Enable .....	18
5.3.5	Offset 38h: UECPA – Host UIC Error Code PHY Adapter Layer .....	19
5.3.6	Offset 3Ch: UECDL – Host UIC Error Code Data Link Layer .....	19
5.3.7	Offset 40h: UECN – Host UIC Error Code Network Layer .....	20
5.3.8	Offset 44h: UECT – Host UIC Error Code Transport Layer .....	20
5.3.9	Offset 48h: UECDME – Host UIC Error Code .....	20
5.3.10	Offset 4Ch: UTRIACR – UTP Transfer Request Interrupt Aggregation Control Register .....	21
5.4	UTP Transfer Request Registers.....	22
5.4.1	Offset 50h: UTRLBA – UTP Transfer Request List Base Address.....	22
5.4.2	Offset 54h: UTRLBAU – UTP Transfer Request List Base Address Upper 32-bits.....	22
5.4.3	Offset 58h: UTRLDBR – UTP Transfer Request List Door Bell Register.....	23
5.4.4	Offset 5Ch: UTRLCLR – UTP Transfer Request List CLeaR Register .....	23
5.4.5	Offset 60h: UTRLRSR – UTP Transfer Request List Run Stop Register .....	23
5.4.6	Offset 64h: UTRLCNR – UTP Transfer Request List Completion Notification Register .....	24
5.5	UTP Task Management Registers .....	25
5.5.1	Offset 70h: UTMRLBA – UTP Task Management Request List Base Address.....	25
5.5.2	Offset 74h: UTMRLBAU – UTP Task Management Request List Base Address Upper 32-bits ..	25
5.5.3	Offset 78h: UTMRLDBR – UTP Task Management Request List Door Bell Register .....	25
5.5.4	Offset 7Ch: UTMRLCLR – UTP Task Management Request List CLeaR Register.....	26
5.5.5	Offset 80h: UTMRLRSR – UTP Task Management Request List Run Stop Register.....	26
5.6	UIC Command Registers .....	27
5.6.1	Offset 90h: UICCMD – UIC Command .....	27
5.6.2	Offset 94h: UICCMDARG1 – UIC Command Argument 1 .....	28
5.6.3	Offset 98h: UICCMDARG2 – UIC Command Argument 2 .....	29
5.6.4	Offset 9Ch: UICCMDARG3 – UIC Command Argument 3.....	30
5.6.5	Attributes for Local L2 Timers .....	30

## Contents (cont'd)

5.7	Vendor Specific Registers.....	31
5.7.1	Offset C0h to FFh: VS – Vendor Specific .....	31
5.8	Crypto Registers .....	31
5.8.1	Offset 100h: CCAP – Crypto Capability .....	31
5.8.2	x-CRYPTOCAP – Crypto Capability X .....	32
5.8.3	x-CRYPTOCFG – Crypto Configuration X .....	33
<b>6</b>	<b>Data structures.....</b>	<b>35</b>
6.1	UTP Transfer Request List .....	35
6.1.1	UTP Transfer Request Descriptor.....	35
6.1.2	UTP Command Descriptor .....	39
6.2	UTP Task Management Request List .....	41
6.2.1	UTP Task Management Request Descriptor.....	41
6.3	Key Organization for Cryptographic Algorithms .....	43
6.3.1	AES-XTS .....	43
6.3.2	Bitlocker-AES-CBC .....	45
6.3.3	AES-ECB.....	46
6.3.4	ESSIV-AES-CBC .....	47
<b>7</b>	<b>Theory of Operation .....</b>	<b>47</b>
7.1	Host Controller Configuration and Control .....	48
7.1.1	Host Controller Initialization .....	48
7.1.2	Configuration and control .....	50
7.1.3	CRYPTOCFG Configuration Procedure .....	50
7.2	Data Transfer Operation .....	51
7.2.1	Basic Steps when Building a UTP Transfer Request.....	52
7.2.2	UPIU Processing.....	53
7.2.3	Processing UTP Transfer Request Completion .....	55
7.3	Task Management Function.....	57
7.3.1	Basic Steps when Building a UTP Task Management Request.....	57
7.3.2	Processing UTP Task Management Completion .....	57
7.4	UIC Power Mode Change.....	58
7.5	UFSHCI Internal Rules.....	59
7.5.1	Command Processing Order .....	60
7.5.2	RTT Processing Rules.....	61
7.5.3	Data Unit Processing Order for Cryptographic operations .....	61
<b>8</b>	<b>error reporting and handling.....</b>	<b>62</b>
8.1	Error Types .....	62
8.1.1	System Bus Error .....	62
8.1.2	UIC Error .....	62
8.1.3	UIC Command Error.....	62
8.1.4	UTP Error .....	63
8.1.5	Host controller Fatal Error.....	63
8.1.6	Device Error.....	63
8.1.7	Hibernate Enter/Exit Error.....	64
8.2	Error Handling .....	64
8.2.1	System Bus Error Handling .....	64
8.2.2	UIC Error Handling .....	65
8.2.3	UIC Command Error Handling.....	65
8.2.4	UTP Error Handling.....	66

## Contents (cont'd)

8.2.5	Host Controller Error Handling .....	66
8.2.6	Device Error Handling .....	66
8.2.7	Hibernate Enter/Exit Error Handling .....	66
<b>9</b>	<b>Encryption ENGINE DETAILS (Informative) .....</b>	<b>67</b>
9.1	AES-XTS .....	67
9.1.1	Overview .....	67
9.1.2	Data Unit Size .....	67
9.1.3	Tweak68 .....	
9.2	Bitlocker AES-CBC .....	68
9.2.1	Background .....	68
9.2.2	Overview .....	69
9.2.3	Sector (Data Unit) Size $S$ .....	69
9.2.4	Sector Offset $O$ .....	69
9.2.5	Sector Initialization Vector (IV) .....	70
9.2.6	Encryption / Decryption .....	70
9.3	AES-ECB .....	70
9.3.1	Overview .....	70
9.4	ESSIV-AES-CBC .....	70
9.4.1	Background .....	70
9.4.2	Data Unit Size .....	70
9.4.3	Sector Number (SN) .....	71
9.4.4	Initialization Vector (IV) .....	71
9.4.5	Encryption / Decryption .....	71
<b>Figures</b>		
Figure 1	— UFS Architecture Overview .....	5
Figure 2	— General architecture of UFS Host Controller Interface .....	6
Figure 3	— A conceptual block diagram of UFS host system .....	8
Figure 4	— x-CRYPTOCFG Array Entry Layout .....	34
Figure 5	— UTP Transfer Request Descriptor .....	35
Figure 6	— UTP Command Descriptor (UCD) .....	39
Figure 7	— Data structure for Physical Region Description Table .....	40
Figure 8	— UTP Task Management Request Descriptor .....	41
Figure 9	— AES128-XTS Key Layout .....	43
Figure 10	— AES192-XTS Key Layout .....	44
Figure 11	— AES256-XTS Key Layout .....	44
Figure 12	— AES128-CBC Key Layout .....	45
Figure 13	— AES256-CBC Key Layout .....	45
Figure 14	— AES128-ECB Key Layout .....	46
Figure 15	— AES256-ECB Key Layout .....	46
Figure 16	— Host controller link startup sequence .....	49
Figure 17	— UIC Power mode change .....	59
Figure 18	— Command processing order .....	60
Figure 19	— Byte Order For Data Unit Processing in Cryptographic operations .....	61
Figure 20	— AES-XTS Encryption .....	67
Figure 21	— Bitlocker AES-CBC Encryption .....	69
Figure 22	— IV Derivation from Sector Offset .....	70
Figure 23	— AES-ECB Encryption .....	70
Figure 24	— ESSIV-AES-CBC Encryption .....	71





## UNIVERSAL FLASH STORAGE HOST CONTROLLER INTERFACE (UFSHCI)

(From JEDEC Board ballot JCB-17-42, formulated under the cognizance of the JC-64.1 Subcommittee on Electrical Specifications and Command Protocols (Item 205.06).)

---

### 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 Reference

---

The following normative documents contain provisions that, through reference in this text, constitute provisions of this standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated. For undated references, the latest edition of the normative document referred to applies.

[MIPI-M-PHY], *MIPI Alliance Specification for M-PHY<sup>SM</sup>, Version 4.1*

[MIPI-UniPro], *MIPI Alliance Specification for Unified Protocol (UniPro<sup>SM</sup>), Version 1.8*

[MIPI-DDB], *MIPI Alliance Specification for Device Descriptor Block (DDB), Version*

[SAM], *INCITS T10 draft standard: SCSI Architecture Model – 5 (SAM–5), Revision 05, 19 May 2010*

[SPC], *INCITS T10 draft standard: SCSI Primary Commands – 4 (SPC-4), Revision 27, 11 October 2010*

[SBC], *INCITS T10 draft standard: SCSI Block Commands – 3 (SBC–3), Revision 24, 05 August 2010*

[UFS], JEDEC JESDxxxx, *Universal Flash Storage (UFS 2.1)*

[JEP], JEDEC JEP106, *Standard Manufacturer's Identification Code*

---

### 3 Acronyms, Terms and Definitions, Keywords, and Conventions

---

#### 3.1 Acronyms

<b>DID</b>	Device ID
<b>GB</b>	Gigabyte
<b>HCI</b>	Host Controller Interface
<b>KB</b>	Kilobyte
<b>LUN</b>	Logical Unit Number
<b>MIPI</b>	Mobile Industry Processor Interface
<b>MB</b>	Megabyte
<b>NA</b>	Not applicable
<b>PRDT</b>	Physical Region Description Table
<b>UCD</b>	UTP Command Descriptor
<b>UFS</b>	Universal Flash Storage
<b>UPIU</b>	UFS Protocol Information Unit
<b>UTP</b>	UFS Transport Protocol
<b>UTRD</b>	UTP Transfer Request Descriptor
<b>UTMRD</b>	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):** 1024 or  $2^{10}$  bytes.

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

**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 bit 0.

### 3.2 Terms and Definitions (cont'd)

**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:

<b>Expected</b>	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.
<b>Ignored</b>	A keyword that describes bits, bytes, quad lets, or fields whose values are not checked by the recipient.
<b>Mandatory</b>	A keyword that indicates items required to be implemented as defined by this standard.
<b>May</b>	A keyword that indicates flexibility of choice with no implied preference.
<b>Optional</b>	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)

<b>Reserved</b>	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.
<b>Shall</b>	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 sections that follow.

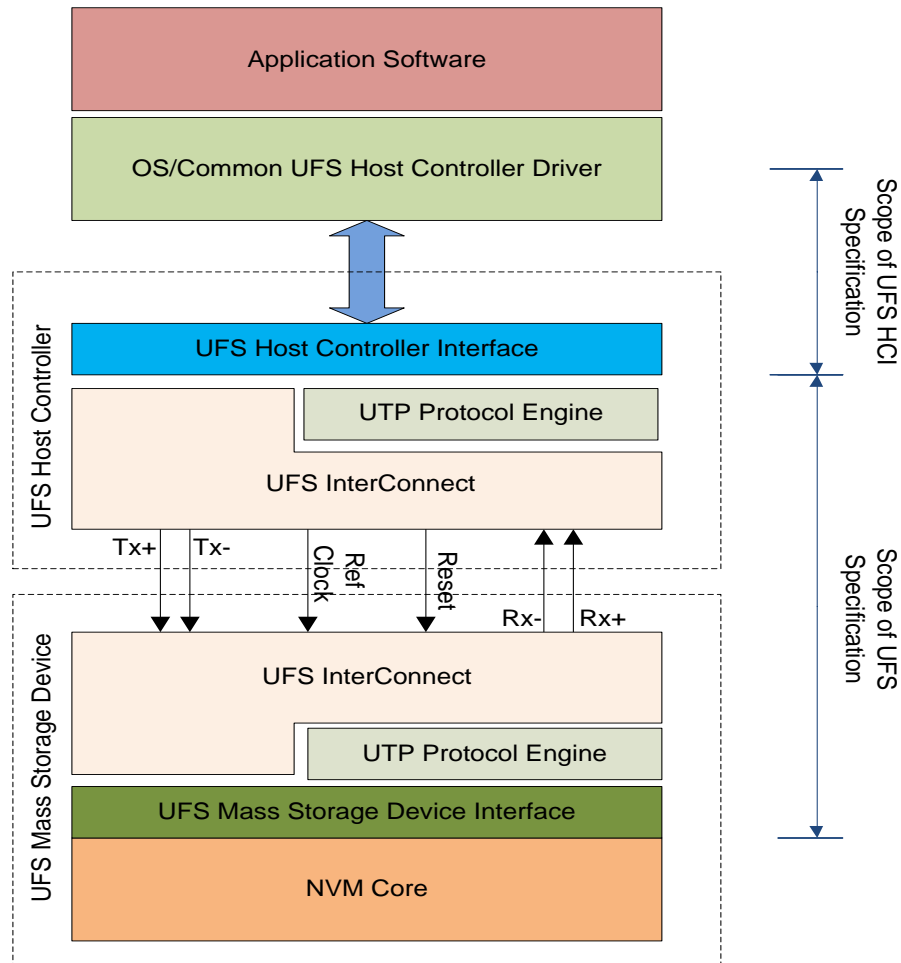
Hardware shall return '0' for all bits and registers that are marked as reserved, and host software shall write all reserved bits and registers with the value of '0'.

Inside the register section, the following abbreviations are used:

<b>HwInit</b>	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.
<b>Impl Spec</b>	Implementation Specific – the controller has the freedom to choose its implementation.
<b>RO</b>	Read Only
<b>ROC</b>	Read Only and Read to clear
<b>RW</b>	Read Write
<b>R/W</b>	Read Write. The value read may not be the last value written.
<b>RWC</b>	Read/Write '1' to clear
<b>RWS</b>	Read/Write '1' to set
<b>WO</b>	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 [UFS], 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.

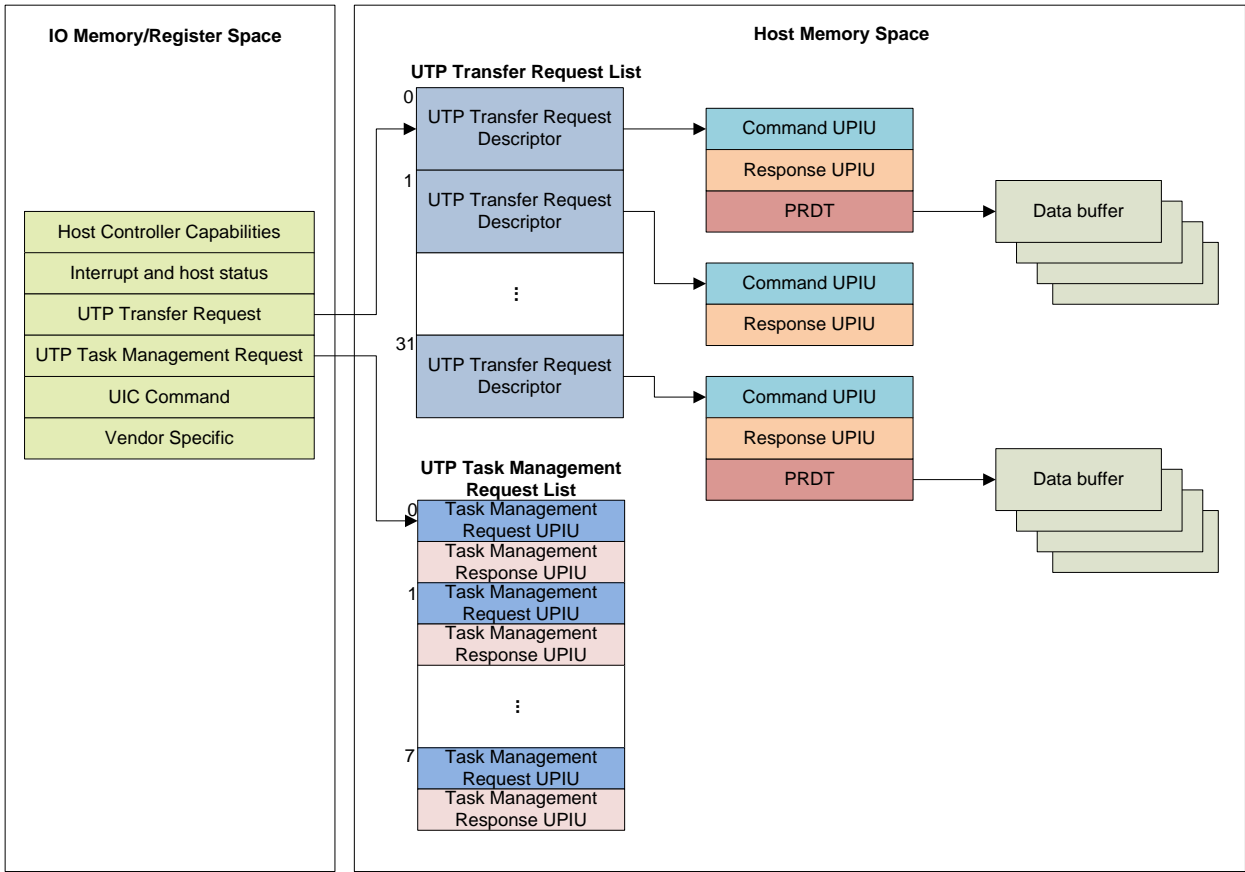


Figure 2 — General architecture of UFS Host Controller Interface.

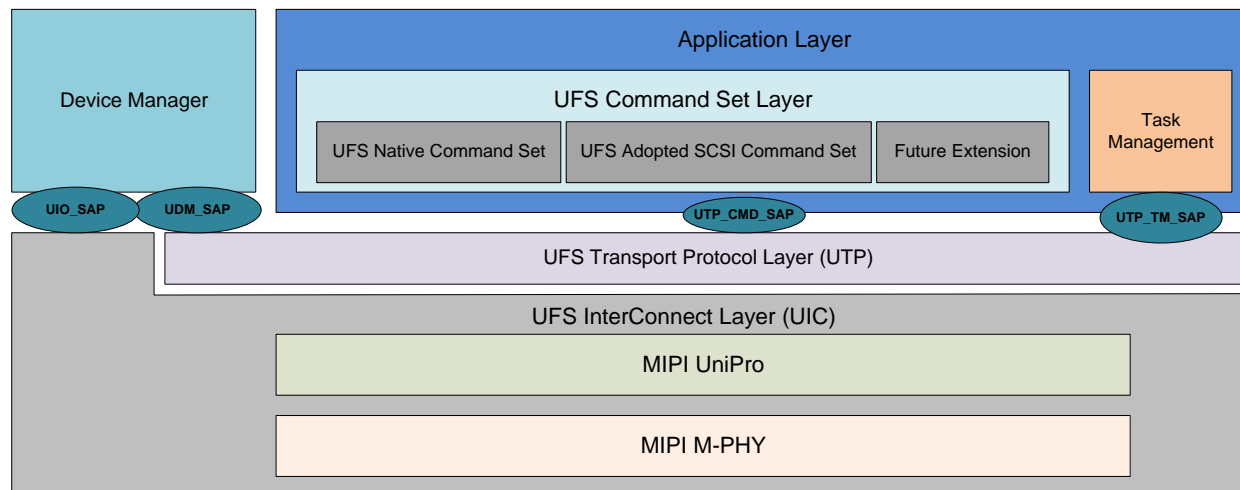
## 4.2 Interface Architecture (cont'd)

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.
  - **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.
    - **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.
  - **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 the Figure 3: UIO\_SAP, UDM\_SAP, UTP\_CMD\_SAP and UTP\_TM\_SAP. Refer to [UFS], UFS Standard 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 Limitations

The phase collapse feature, currently specified in [UFS], UFS Standard, will be removed from future UFS standards. As phase collapse also adds complexity to UFS Host Controller, the phase collapse feature is not supported by UFS HCI.

---

## 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 UFSHCL.

	Start	End	Symbol	Description
Host Capabilities	00h	03h	CAP	Host Controller <b>Cap</b> abilities
	04h	07h	Reserved	Reserved
	08h	0Bh	VER	UFS <b>V</b> ersion
	0Ch	0Fh	Reserved	Reserved
	10h	13h	HCDDID	<b>H</b> ost <b>C</b> ontroller <b>I</b> dentification <b>D</b> escriptor – Device ID and Device Class
	14h	17h	HCPMID	<b>H</b> ost <b>C</b> ontroller <b>I</b> dentification <b>D</b> escriptor – Product ID and Manufacturer ID
	18h	1Bh	AHIT	Auto-Hibernate Idle Timer
	1Ch	1Fh	Reserved	Reserved
Operation and Runtime	20h	23h	IS	<b>I</b> nterrupt <b>S</b> tatus
	24h	27h	IE	<b>I</b> nterrupt <b>E</b> nable
	28h	2Fh	Reserved	Reserved
	30h	33h	HCS	<b>H</b> ost <b>C</b> ontroller <b>S</b> tatus
	34h	37h	HCE	<b>H</b> ost <b>C</b> ontroller <b>E</b> nable
	38h	3Bh	UECPA	Host <b>U</b> IC <b>E</b> rror <b>C</b> ode <b>P</b> HY <b>A</b> dapter <b>L</b> ayer
	3Ch	3Fh	UECDL	Host <b>U</b> IC <b>E</b> rror <b>C</b> ode <b>D</b> ata <b>L</b> ink <b>L</b> ayer
	40h	43h	UECN	Host <b>U</b> IC <b>E</b> rror <b>C</b> ode <b>N</b> etwork <b>L</b> ayer
	44h	47h	UECT	Host <b>U</b> IC <b>E</b> rror <b>C</b> ode <b>T</b> ransport <b>L</b> ayer
	48h	4Bh	UECDME	Host <b>U</b> IC <b>E</b> rror <b>C</b> ode <b>D</b> ME
	4Ch	4Fh	UTRIACR	<b>U</b> TP <b>T</b> ransfer <b>R</b> equest <b>I</b> nterrupt <b>A</b> ggregation <b>C</b> ontrol <b>R</b> egister
UTP Transfer	50h	53h	UTRLBA	<b>U</b> TP <b>T</b> ransfer <b>R</b> equest <b>L</b> ist <b>B</b> ase <b>A</b> ddress
	54h	57h	UTRLBAU	<b>U</b> TP <b>T</b> ransfer <b>R</b> equest <b>L</b> ist <b>B</b> ase <b>A</b> ddress <b>U</b> pper 32-Bits
	58h	5Bh	UTRLDBR	<b>U</b> TP <b>T</b> ransfer <b>R</b> equest <b>L</b> ist <b>D</b> oor <b>B</b> ell <b>R</b> egister
	5Ch	5Fh	UTRLCLR	<b>U</b> TP <b>T</b> ransfer <b>R</b> equest <b>L</b> ist <b>C</b> lear <b>R</b> egister
	60h	63h	UTRLRSR	<b>U</b> TP <b>T</b> ransfer <b>R</b> equest <b>R</b> un- <b>S</b> top <b>R</b> egister
	64h	67h	UTRLCNR	<b>U</b> TP <b>T</b> ransfer <b>R</b> equest <b>L</b> ist <b>C</b> ompletion <b>N</b> otification <b>R</b> egister
	68h	6Fh	Reserved	Reserved
UTP Task Management	70h	73h	UTMRLBA	<b>U</b> TP <b>T</b> ask <b>M</b> anagement <b>R</b> equest <b>L</b> ist <b>B</b> ase <b>A</b> ddress
	74h	77h	UTMRLBAU	<b>U</b> TP <b>T</b> ask <b>M</b> anagement <b>R</b> equest <b>L</b> ist <b>B</b> ase <b>A</b> ddress <b>U</b> pper 32-Bits
	78h	7Bh	UTMRLDBR	<b>U</b> TP <b>T</b> ask <b>M</b> anagement <b>R</b> equest <b>L</b> ist <b>D</b> oor <b>B</b> ell <b>R</b> egister
	7Ch	7Fh	UTMRLCLR	<b>U</b> TP <b>T</b> ask <b>M</b> anagement <b>R</b> equest <b>L</b> ist <b>C</b> lear <b>R</b> egister
	80h	83h	UTMRLRSR	<b>U</b> TP <b>T</b> ask <b>M</b> anagement <b>R</b> un- <b>S</b> top <b>R</b> egister
	84h	8Fh	Reserved	Reserved
UIC Command	90h	93h	UICCMD	<b>U</b> IC <b>C</b> ommand <b>R</b> egister
	94h	97h	UCMDARG1	<b>U</b> IC <b>C</b> ommand <b>A</b> rgument 1
	98h	9Bh	UCMDARG2	<b>U</b> IC <b>C</b> ommand <b>A</b> rgument 2
	9Ch	9Fh	UCMDARG3	<b>U</b> IC <b>C</b> ommand <b>A</b> rgument 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	<b>C</b> rypto <b>C</b> apability

## 5.2 Host Controller Capabilities Registers

This section 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
31:29	RO	0	Reserved.
28	RO	Impl Spec	<b>Crypto Support (CS):</b> Indicates whether the host controller supports cryptographic operations. <ul style="list-style-type: none"> <li>0 – Host controller does not support cryptographic operations.</li> <li>1 – Host controller supports cryptographic operations.</li> </ul>
27	RO		Reserved for Unified Memory Extension
26	RO	Impl Spec	<b>UIC DME_TEST_MODE command supported (UICDMETMS):</b> Indicates whether the host controller supports the UniPro DME_TEST_MODE.req SAP primitive.
25	RO	Impl Spec	<b>Out of order data delivery supported (OODDS):</b> Indicates whether the host controller supports out of order data delivery for UTP data transfer. When set to '1', the host controller shall support 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.
24	RO	Impl Spec	<b>64-bit addressing supported (64AS):</b> 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	<b>Auto-Hibernation Support (AUTOH8):</b> Indicates whether the host controller supports auto-hibernation. <ul style="list-style-type: none"> <li>0 – Host controller does not support auto-hibernation</li> <li>1 – Host controller supports auto-hibernation.</li> </ul>
22:19	RO	0h	Reserved
18:16	RO	Impl Spec	<b>Number of UTP Task Management Request Slots (NUTMRS):</b> 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.
15:08	RO	Impl Spec	<b>Number of outstanding READY TO TRANSFER (RTT) requests supported (NORTT):</b> 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.
07:05	RO	0h	Reserved
04:00	RO	Impl Spec	<b>Number of UTP Transfer Request Slots (NUTRS):</b> 0's based value indicating the number of slots provided by the UTP Transfer Request List. A minimum of 1 and maximum of 32 slots may be supported.

## 5.2 Host Controller Capabilities Registers (cont'd)

### 5.2.2 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. Valid versions of the standard are: 3.0.

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

### 5.2.3 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	<b>Product ID (PID):</b> Product ID that host controller manufacturer assigns for the host controller. This is vendor specific.

### 5.2.4 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 [JEDEC-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	<b>Bank Index (BI):</b> 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 [JEDEC-JEP106].
07:00	RO	Impl spec	<b>Manufacturer Identification Code (MIC):</b> Manufacturer Identification code as defined by JEDEC in Standard Manufacturer's identification code [JEDEC-JEP106].

## 5.2 Host Controller Capabilities Registers (cont'd)

### 5.2.5 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	<b>Timer scale(TS):</b> <ul style="list-style-type: none"> <li>• 000 – Value times 1 us</li> <li>• 001 – Value times 10 us</li> <li>• 010 – Value times 100 us</li> <li>• 011 – Value times 1 ms</li> <li>• 100 – Value times 10 ms</li> <li>• 101 – Value times 100 ms</li> <li>• 110 - 111 – reserved</li> </ul>
09:00	RW	0	<p><b>Auto-Hibern8 Idle Timer Value (AH8ITV):</b> 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:</p> <ul style="list-style-type: none"> <li>• UTRLDBR='0'</li> <li>• UTMRLDBR='0'</li> <li>• No UIC command is outstanding</li> </ul> <p>The idle timer shall continue decrement until it reaches zero or it is stopped as result of software access to one of host controller interface registers. When idle timer changes a non-zero to zero, host controller shall put UniPro link into Hibernate state.</p> <p>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.</p> <p>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.</p>

### 5.3 Operation and Runtime Registers

This section 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
31:19	RO	0h	Reserved
18	RWC	0	<b>Crypto Engine Fatal Error Status (CEFES):</b> 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 <b>UTRLRSR</b> and <b>UTMRLRSR</b> will be cleared to "0" by host controller. If the error occurs, host SW should reset the host controller.
17	RWC	0	<b>System Bus Fatal Error Status (SBFES):</b> 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	<b>Host Controller Fatal Error Status (HCFES):</b> 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 <b>UTRLRSR</b> and <b>UTMRLRSR</b> 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	<b>UTP Error Status (UTPES):</b> 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	<b>Device Fatal Error Status (DFES):</b> 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	<b>UIC Command Completion Status (UCCS):</b> This bit is set to '1' by the host controller upon completion of a UIC command.
09	RWC	0	<b>UTP Task Management Request Completion Status (UTMRCS):</b> 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	<b>UIC Link Startup Status (ULSS):</b> 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	<b>UIC Link Lost Status (ULLS):</b> 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	<b>UIC Hibernate Enter Status (UHES):</b> 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 <b>HCS.UPMCRS</b> indicates the status of the hibernation entering process. When the hibernate entering process is initiated by auto-hibernation as defined in <b>AHIT</b> , this field indicates that error is detected during hibernate entering process. Register <b>HCS.UPMCRS</b> indicates the error status. This bit corresponds to the UniPro DME_HIBERNATE_ENTER.ind SAP primitive.
05	RWC	0	<b>UIC Hibernate Exit Status (UHXS):</b> When the hibernate exiting process is initiated by host software, this field indicates that the Link has exited UniPro Hibernate state. Register <b>HCS.UPMCRS</b> indicates the status of the hibernation exiting process. When the hibernate exiting process is initiated by auto-hibernation as defined in <b>AHIT</b> , this field indicates that error is detected during hibernate exiting process. Register <b>HCS.UPMCRS</b> indicates the error status. This bit corresponds to the UniPro DME_HIBERNATE_EXIT.ind SAP primitive.

### 5.3 Operation and Runtime Registers (cont'd)

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

Bit	Type	Reset	Description
04	RWC	0	<b>UIC Power Mode Status (UPMS):</b> indicate that the <i>Unipro/PA/DL</i> part of the power mode change has been completed. Register <b>HCS.UPMCRS</b> contains the power mode change status. This bit corresponds to the UniPro DME_POWERMODE.ind SAP primitive..
03	RWC	0	<b>UIC Test Mode Status (UTMS):</b> 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	<b>UIC Error (UE):</b> Indicate that a layer in the UniPro stack has encountered an error condition. Register <b>UECPA/UECDL/UECN/UECT/UECDME</b> 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	<b>UIC DME_ENDPOINTRESET Indication(UDEPRI):</b> Indicate that the attached device has issued an <i>DME_ENDPOINTRESET</i> indication which is not allowed.
00	RWC	0	<b>UTP Transfer Request Completion Status (UTRCS):</b> This bit is set to '1' by the host controller upon one of the following: <ul style="list-style-type: none"> <li>• Completion of a UTP transfer request with its UTRD Interrupt bit set to '1'.</li> <li>• Interrupt caused by the UTR interrupt aggregation logic.</li> <li>• Overall command Status (OCS) of the completed command is not equal to "SUCCESS".</li> </ul>

### 5.3 Operation and Runtime Registers (cont'd)

#### 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
31:19	RO	0h	Reserved
18	RW	0	<b>Crypto Engine Fatal Error Enable (CEFEE):</b> When set and <b>IS.CEFES</b> is set, the controller shall generate an interrupt.
17	RW	0	<b>System Bus Fatal Error Enable (SBFEE):</b> When set and <b>IS.SBFES</b> is set, the controller shall generate an interrupt.
16	RW	0	<b>Host Controller Fatal Error Enable (HCFEE):</b> When set and <b>IS.HCFES</b> is set, the controller shall generate an interrupt.
15:13	RO	0	Reserved
12	RW	0	<b>UTP Error Enable (UTPEE):</b> When set and <b>IS.UTPES</b> is set, the controller shall generate an interrupt.
11	RW	0	<b>Device Fatal Error Enable (DFEE):</b> When set and <b>IS.DFES</b> is set, the host controller shall generate an interrupt.
10	RW	0	<b>UIC COMMAND Completion Enable (UCCE):</b> When set and <b>IS.UCCS</b> is set, the host controller shall generate an interrupt.
09	RW	0	<b>UTP Task Management Request Completion Enable (UTMRCE):</b> When set and <b>IS.UTMRCS</b> is set, the host controller shall generate an interrupt.
08	RW	0	<b>UIC Link Startup Status Enable (ULSSE):</b> When set and <b>IS.ULSS</b> is set, the controller shall generate an interrupt.
07	RW	0	<b>UIC Link Lost Status Enable (ULLSE):</b> When set and <b>IS.ULLS</b> is set, the controller shall generate an interrupt.
06	RW	0	<b>UIC Hibernate Enter Status Enable (UHESE):</b> When set and <b>IS.UHES</b> is set, the controller shall generate an interrupt.
05	RW	0	<b>UIC Hibernate Exit Status Enable (UHXSE):</b> When set and <b>IS.UHXS</b> is set, the controller shall generate an interrupt.
04	RW	0	<b>UIC Power Mode Status Enable (UPMSE):</b> When set and <b>IS.UPMS</b> is set, the controller shall generate an interrupt.
03	RW	0	<b>UIC Test Mode Status Enable (UTMSE):</b> When set and <b>IS.UTMS</b> is set, the controller shall generate an interrupt.
02	RW	0	<b>UIC Error Enable (UEE):</b> When set and <b>IS.UEE</b> is set, the controller shall generate an interrupt.
01	RW	0	<b>UIC DME_ENDPOINTRESET (UDEPRIE):</b> When set and <b>IS. UDEPRI</b> is set, the controller shall generate an interrupt.
00	RW	0	<b>UTP Transfer Request Completion Enable (UTRCE):</b> When set and <b>IS.UTRCS</b> is set, the host controller shall generate an interrupt.



### 5.3 Operation and Runtime Registers (cont'd)

#### 5.3.3 Offset 30h: HCS – Host Controller Status

Bit	Type	Reset	Description																
31:12	RO	0	<b>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.</b>																
			<b>Bit[31:24] – Target LUN of UTP error (TLUNUTPE):</b> The LUN of the command that a UTP error occurs during execution of the command.																
			<b>Bit[23:16] – Task Tag of UTP error (TTAGUTPE):</b> The Task Tag of the command that a UTP error occurs during execution of the command.																
			<b>Bit[15:12] - UTP Error Code (UTPEC):</b> 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 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 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 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 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 Task Tag or LUN.</td></tr><tr><td>4h - Fh</td><td>Reserved</td></tr></table>	Value	Description	0h	Reject UPIU has invalid 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 Task Tag or LUN.	1h	Invalid UPIU type.	2h	Inbound UPIU associated with Transfer Request has invalid 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.	3h	Inbound UPIU associated with Task Management Request has invalid 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 Task Tag or LUN.	4h - Fh	Reserved				
			Value	Description															
			0h	Reject UPIU has invalid 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 Task Tag or LUN.															
1h	Invalid UPIU type.																		
2h	Inbound UPIU associated with Transfer Request has invalid 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.																		
3h	Inbound UPIU associated with Task Management Request has invalid 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 Task Tag or LUN.																		
4h - Fh	Reserved																		
11	RO	0	Reserved																
10:08	RO	0	<b>UIC Power Mode Change Request Status (UPMCRS):</b> Indicate that the status of a UIC layer request for power mode change.																
			<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>	Value	Description	0h	PWR_OK. The request was accepted.	1h	PWR_LOCAL. The local request was successfully applied.	2h	PWR_REMOTE. The remote request was successfully applied.	3h	PWR_BUSY. The request was aborted due to concurrent requests.	4h	PWR_ERROR_CAP. The request was rejected because the requested configuration exceeded the Link’s capabilities.	5h	PWR_FATAL_ERROR. The request was aborted due to a communication problem. The Link may be inoperable.	6h - 7h	Reserved
			Value	Description															
			0h	PWR_OK. The request was accepted.															
			1h	PWR_LOCAL. The local request was successfully applied.															
			2h	PWR_REMOTE. The remote request was successfully applied.															
			3h	PWR_BUSY. The request was aborted due to concurrent requests.															
4h	PWR_ERROR_CAP. The request was rejected because the requested configuration exceeded the Link’s capabilities.																		
5h	PWR_FATAL_ERROR. The request was aborted due to a communication problem. The Link may be inoperable.																		
6h - 7h	Reserved																		
07:04	RO	0	Reserved.																
03	RO	0	<b>UIC COMMAND Ready (UCRDY):</b> This field indicates whether the host controller is ready to process UIC COMMAND. Host software shall only set the <b>UICCMD</b> if <b>HCS.UCRDY</b> is set to ‘1’.																

### 5.3 Operation and Runtime Registers (cont'd)

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

Bit	Type	Reset	Description
02	RO	0	<p><b>UTP Task Management Request List Ready (UTMRLRDY):</b> This field is set to '1' when the host controller is ready to Task Management requests. This field is cleared to '0' by host controller when one of the following conditions occur:</p> <ul style="list-style-type: none"> <li>The device presence is not detected;</li> <li>UTP Task Management Request List is full;</li> <li>There is an error with host controller or device that is not command specific.</li> </ul> <p>Host software shall only set the <b>UTMRLRSR</b> register if <b>HCS.UTMRLRDY</b> is set to '1'.</p>
01	RO	0	<p><b>UTP Transfer Request List Ready (UTRLRDY):</b> 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:</p> <ul style="list-style-type: none"> <li>The device presence is not detected;</li> <li>UTP Transfer Request List is full;</li> <li>There is an error with host controller or device that is not command specific.</li> </ul> <p>Host software shall only set the <b>UTRLRSR</b> bit to '1' if <b>HCS.UTRLRDY</b> is set to '1'.</p>
00	RO	0	<p><b>Device Present (DP):</b> 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.</p>

#### 5.3.4 Offset 34h: HCE – Host Controller Enable

Bit	Type	Reset	Description						
31:02	RO	0	Reserved						
01	RW	0	<b>Crypto General Enable (CGE):</b> 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>	Bit Value	Description	0	Disable cryptographic operations for all transactions	1	Enable cryptographic operations for transactions where UTRD.CE=1
			Bit Value	Description					
0	Disable cryptographic operations for all transactions								
1	Enable cryptographic operations for transactions where UTRD.CE=1								
00	RW	0	<b>Host Controller Enable (HCE):</b> 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>	Bit Value	Description	0	Disable	1	Enable
Bit Value	Description								
0	Disable								
1	Enable								

### 5.3 Operation and Runtime Registers (cont'd)

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

Bit	Type	Reset	Description	
31:31	ROC	0	<b>UIC PHY AdapterA Layer Error (ERR):</b> Indicates whether an error was generated by the PHY Adapter Layer	
30:05	RO	0	Reserved	
04:00	ROC	0	<b>UIC PHY Adapter Layer Error Code (EC):</b> error code generated when <b>IS.UE</b> and <b>UECPA.ERR</b> are set to ‘1’.	
			Bit	Description
			00	Error reported by the M-PHY layer: PHY error on Lane 0.
			01	Error reported by the M-PHY layer: PHY error on Lane 1
			02	Error reported by the M-PHY layer: PHY error on Lane 2
			03	Error reported by the M-PHY layer: PHY error on Lane 3
04	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.			

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

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

### 5.3 Operation and Runtime Registers (cont'd)

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

Bit	Type	Reset	Description								
31:31	ROC	0	<b>UIC Network Layer Error (ERR):</b> Indicates whether an error was generated by the Network Layer								
30:03	RO	0	Reserved								
02:00	ROC	0	<b>UIC Network Layer Error Code (EC):</b> error code generated when <b>IS.UE</b> and <b>UECN.ERR</b> 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>BAD_DEVICEID_ENC</td></tr><tr><td>02</td><td>LHDR TRAP PACKET DROPPING</td></tr></table>	Bit	Description	00	UNSUPPORTED_HEADER_TYPE	01	BAD_DEVICEID_ENC	02	LHDR TRAP PACKET DROPPING
			Bit	Description							
			00	UNSUPPORTED_HEADER_TYPE							
			01	BAD_DEVICEID_ENC							
02	LHDR TRAP PACKET DROPPING										

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

Bit	Type	Reset	Description																
31:31	ROC	0	<b>UIC Transport Layer Error (ERR):</b> Indicates whether an error was generated by the Transport Layer																
30:07	RO	0	Reserved																
06:00	ROC	0	<b>UIC Transport Layer Error Code (EC):</b> error code generated when <b>IS.UE</b> and <b>UECT.ERR</b> 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>	Bit	Description	00	UNSUPPORTED_HEADER_TYPE	01	UNKNOWN_CPORTID	02	NO_CONNECTION_RX	03	CONTROLLED_SEGMENT_DROPPING	04	BAD_TC	05	E2E_CREDIT_OVERFLOW	06	SAFETY_VALVE_DROPPING
			Bit	Description															
			00	UNSUPPORTED_HEADER_TYPE															
			01	UNKNOWN_CPORTID															
			02	NO_CONNECTION_RX															
			03	CONTROLLED_SEGMENT_DROPPING															
			04	BAD_TC															
			05	E2E_CREDIT_OVERFLOW															
06	SAFETY_VALVE_DROPPING																		

#### 5.3.9 Offset 48h: UECDME – Host UIC Error Code

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

### 5.3 Operation and Runtime Registers (cont'd)

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

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

### 5.3 Operation and Runtime Registers (cont'd)

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

Bit	Type	Reset	Description
7:0	RW	0	<p><b>Interrupt aggregation timeout value (IATOTAL):</b> 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.</p> <p><b>Timer Operation:</b> 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 <b>IATOTAL</b> field, and <b>IS.UTRCS</b> bit is set.</p> <p>NOTE 1 When <b>IATOTAL</b> is 0, the timer is not running, and timer-based interrupts are not generated.</p> <p>NOTE 2 QUERY RESPONSE UPIUs and NOP IN UPIUs shall not be counted by the Interrupt Aggregation logic.</p> <p>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</p>

### 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	<b>UTP Transfer Request List Base Address (UTRLBA):</b> 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 1KB in length. This address shall be 1KB 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	<b>UTP Transfer Request List Base Address Upper (UTRLBAU):</b> 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 UTP Transfer Request Registers (cont'd)

### 5.4.3 Offset 58h: UTRLDBR – UTP Transfer Request List Door Bell Register

Bit	Type	Reset	Description
31:0	RWS	0	<p><b>UTP Transfer Request List Door bell Register(UTRLDBR):</b> 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 <b>UTRLRSR</b> is set to '1'.</p> <p>When a transfer request is completed (with success or error), the corresponding bit is cleared to '0' by the host controller.</p> <p>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.</p> <p>This field is also cleared when <b>UTRLRSR</b> is written from a '1' to a '0' by host software.</p>

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

Bit	Type	Reset	Description
31:0	WO	0	<p><b>UTP Transfer Request List CLear Register(UTRLCLR):</b> 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 <b>UTRLRSR</b> is set to '1'.</p> <p>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.</p>

### 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	<p><b>UTP Transfer Request List Run-Stop Register (UTRLRSR):</b> 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.</p> <p>This bit shall only be set to '1' when <b>HCS.UTRLRDY</b> is set to '1'.</p>

## 5.4 UTP Transfer Request Registers (cont'd)

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

Bit	Type	Reset	Description
31:0	RWC	0	<p><b>UTP Transfer Request List Completion Notification Register(UTRLCNR):</b> 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.</p> <p>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.</p> <p>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.</p> <p>The host controller shall clear this register when <b>UTRLRSR</b> is written from a '0' to a '1' by host software.</p>



## 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	<b>UTP Task Management Request List Base Address (UTMRLBA):</b> 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 1KB 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	<b>UTP Task Management Request List Base Address (UTMRLBAU):</b> 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 Door Bell Register

Bit	Type	Reset	Description
31:08	RO	0	<b>Reserved</b>
07:0	RWS	0	<p><b>UTP Task Management Request List Door bell Register(UTMRLDBR):</b> 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 <b>UTMRLRSR</b> is set to ‘1’.</p> <p>When a task management request is completed (with success or error), the corresponding bit is cleared to ‘0’ by the host controller.</p> <p>The host controller always process task management request in-order according to the order submitted to the list. In case of multiple requests with single door bell 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.</p> <p>This field is also cleared when <b>UTMRLRSR</b> is written from a ‘1’ to a ‘0’ by host software..</p>

## 5.5 UTP Task Management Registers (cont'd)

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

Bit	Type	Reset	Description
31:08	RO	0	<b>Reserved</b>
07:0	WO	0	<p><b>UTP Task Management List CLear Register(UTMRLCLR):</b> 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 <b>UTRLRSR</b> is set to '1'.</p> <p>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.</p>

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

Bit	Type	Reset	Description
31:01	RO	0	<b>Reserved</b>
0	RW	0	<p><b>UTP Task Management Request List Run-Stop Register (UTMRLRSR):</b> 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.</p> <p>This bit shall only be set to '1' when <b>HCS.UTMRLRDY</b> is set to '1'.</p>

## 5.6 UIC Command Registers

### 5.6.1 Offset 90h: UICCMD – UIC Command

Bit	Type	Reset	Description																																																											
31:08	RO	0	Reserved.																																																											
07:00	RW	0h	<b>Command Opcode(CMDOP):</b> 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 <b>UICCMDARGx</b> as the corresponding parameters (input and output) that are a part of the UIC Command.																																																											
			Opcode	O/M	UIC Command	Configuration			01h	M	DME_GET	02h	M	DME_SET	03h	M	DME_PEER_GET	04h	M	DME_PEER_SET	05h – 0Fh		Reserved	Control			10h	O	DME_POWERON	11h	O	DME_POWEROFF	12h	M	DME_ENABLE	13h		Reserved	14h	M	DME_RESET	15h	M	DME_ENDPOINTRESET	16h	M	DME_LINKSTARTUP	17h	M	DME_HIBERNATE_ENTER	18h	M	DME_HIBERNATE_EXIT	19h		Reserved	1Ah	O	DME_TEST_MODE	1Bh - FFh		Reserved
			Opcode	O/M	UIC Command																																																									
			Configuration																																																											
			01h	M	DME_GET																																																									
			02h	M	DME_SET																																																									
			03h	M	DME_PEER_GET																																																									
			04h	M	DME_PEER_SET																																																									
			05h – 0Fh		Reserved																																																									
			Control																																																											
			10h	O	DME_POWERON																																																									
			11h	O	DME_POWEROFF																																																									
			12h	M	DME_ENABLE																																																									
			13h		Reserved																																																									
			14h	M	DME_RESET																																																									
			15h	M	DME_ENDPOINTRESET																																																									
			16h	M	DME_LINKSTARTUP																																																									
			17h	M	DME_HIBERNATE_ENTER																																																									
			18h	M	DME_HIBERNATE_EXIT																																																									
			19h		Reserved																																																									
			1Ah	O	DME_TEST_MODE																																																									
			1Bh - FFh		Reserved																																																									
			O/M: O = Optional, M = Mandatory.																																																											

## 5.6 UIC Command Registers (cont'd)

### 5.6.2 Offset 94h: UICCMDARG1 – UIC Command Argument 1

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

**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

**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 UIC Command Registers (cont'd)

### 5.6.3 Offset 98h: UICCMDARG2 – UIC Command Argument 2

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

**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 UIC Command Registers (cont'd)

### 5.6.4 Offset 9Ch: UICCMDARG3 – UIC Command Argument 3

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

**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 section, 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 in fact a way to implement the DME\_POWERMODE.req primitive.

In case of the UFS application it is mandatory to implement the informative 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	<b>Configuration Array Pointer (CFGPTR):</b> An offset pointer to the base of the Configuration Array (x-CRYPTOCFG registers), in 256B 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	<b>Configuration Count (CFGC):</b> 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).
07:00	RO	Impl Spec	<b>Crypto Capabilities (CC):</b> The number of crypto capabilities that the host controller provides. The values allowed are between 1 and 255.

## 5.8 Crypto Registers (cont'd)

### 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 #254 (if implemented) shall be located in offset 4FCh.

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.

Bit	Type	Reset	Description														
31:24	RO	0	Reserved.														
23:16	RO	Impl Spec	<b>Key Size (KS):</b> Specifies Key Size in bits used by this 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>	KS	Key Size in Bits	00h	Reserved	01h	128 bits	02h	192 bits	03h	256 bits	04h	512 bits	05h-FFh	Reserved
KS	Key Size in Bits																
00h	Reserved																
01h	128 bits																
02h	192 bits																
03h	256 bits																
04h	512 bits																
05h-FFh	Reserved																
15:08	RO	Impl Spec	<b>Supported Data Unit Size Bitmask (SDUSB):</b> 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 1KB, ..., bit 7 indicates 64 KB. One or more bits in this field may be set. For example, if sizes 1KB, 4KB, and 16KB are supported by the capability, then SDUSB=00101010b (=2Ah).														
07:00	RO	Impl Spec	<b>Algorithm ID (ALGID):</b> 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-7Fh</td><td>Reserved</td></tr><tr><td>80h-FFh</td><td>Vendor specific</td></tr></table>	ID code	Algorithm	00h	AES-XTS	01h	Microsoft Bitlocker™ AES-CBC	02h	AES-ECB	03h	ESSIV-AES-CBC	04h-7Fh	Reserved	80h-FFh	Vendor specific
ID code	Algorithm																
00h	AES-XTS																
01h	Microsoft Bitlocker™ AES-CBC																
02h	AES-ECB																
03h	ESSIV-AES-CBC																
04h-7Fh	Reserved																
80h-FFh	Vendor specific																



## 5.8 Crypto Registers (cont'd)

### 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 illustrated in Figure 4. 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	<b>Vendor-Specific Bits (VSB):</b> 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	<b>Reserved for Multi-Host Related Functions</b>
543	RW	0	<b>Configuration Enable (CFGE):</b> This field is used by software to enable/disable a Crypto Configuration usage <ul style="list-style-type: none"> <li>0b – Configuration Disabled. Transactions using this Crypto Configuration (UTRD.CCI field) shall be terminated with error by host controller (OCS=INVALID_CRYPTO_CONFIG)</li> <li>1b – Configuration Enabled. Transactions using this Crypto Configuration (UTRD.CCI) field can be executed</li> </ul>
542:528	RO	0	Reserved.
527:520	RW	0	<b>Crypto Capability Index (CAPIDX):</b> 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	<b>Data Unit Size (DUSIZE):</b> 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	<b>Crypto Key (CRYPTOKEY):</b> 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 Crypto Registers (cont'd)

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

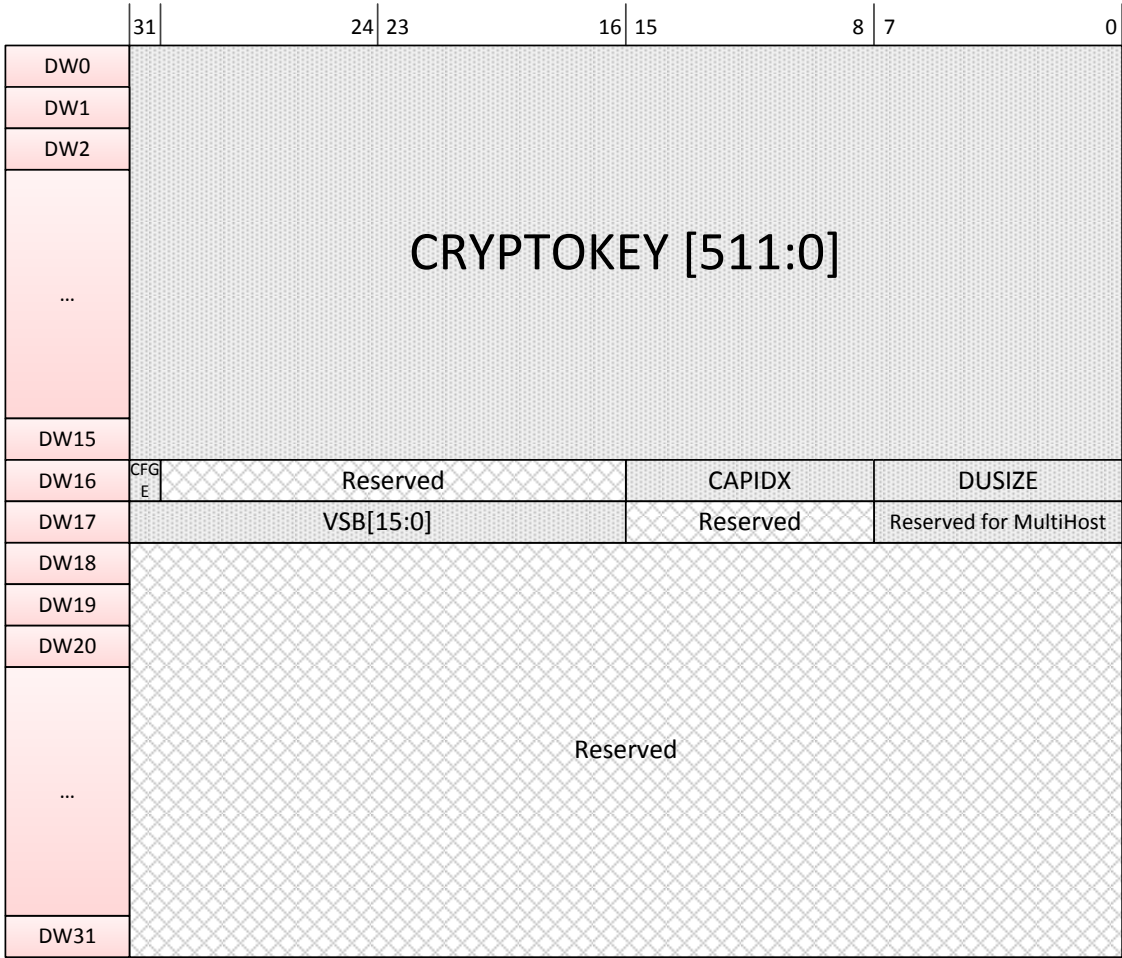


Figure 4 — x-CRYPTOCFG Array Entry Layout

## 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 section defines these descriptors. The data structure definitions in this section support a 32-bit or 64-bit memory buffer address space. They are all in little endian format except as noted.

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

	31	27	26	25	24	23	22	16	15	8	7	6	0
DW0	Command Type (01h)	R	DD	I	C E	Reserved						Crypto Config. Index (CCI)	
DW1	Data Unit Number Lower 32 bits (DUNL)												
DW2	Reserved										Overall Command Status		
DW3	Data Unit Number Upper 32 bits (DUNU)												
DW4	UTP Command Descriptor Base Address										Reserved		
DW5	UTP Command Descriptor Base Address Upper 32-bits												
DW6	Response UPIU Offset								Response UPIU Length				
DW7	PRDT Offset								PRDT Length				

**Figure 5 — UTP Transfer Request Descriptor**

## 6.1 UTP Transfer Request List (cont'd)

### 6.1.1 UTP Transfer Request Descriptor (cont'd)

The following section provides the description of the data structure.

DW0	Bit	Description	
	31:28	<b>Command Type (CT):</b> CT=1h. Type of the command to be transferred.	
		Bits	Definition
		0h	Reserved
		1h	UFS Storage
	2h-Fh	Reserved	
	27	Reserved	
	26:25	<b>Data Direction (DD):</b> 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.	
		Bits	Definition
		00b	No data transfer. PRDT shall have 0 entry.
		01b	From system memory to target device
		10b	From target device to system memory
11b	Reserved		
24	<b>Interrupt (I):</b> This field indicates the type of command with regard to interrupt generation.		
	Bit	Definition	Description
	0b	Regular Command	Hardware shall count the completion of this command towards interrupt aggregation. Impact on <b>IS.UTRCS</b> is described in Interrupt Aggregation section.
	1b	Interrupt Command	Hardware shall set <b>IS.UTRCS</b> to ‘1’ on completion of this command. The completion is not counted towards interrupt aggregation.
NOTE 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.			
23	<b>Crypto Enable (CE):</b>		
	Bit	Definition	
	0b	Disable cryptographic operations for this transaction.	
1b	Enable cryptographic operations for this transaction. Incoming payload is decrypted if the command is SCSI READ operation; Outgoing payload is encrypted if the command is SCSI WRITE operation. UFS host controller takes no action for all other commands.		
	<b>Note:</b> This field is valid only in controllers supporting cryptographic operations (CAP.CS = 1). If CAP.CS=0, this field is reserved.		
22:08	Reserved		
07:00	<b>Crypto Configuration Index (CCI):</b> 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.		
	<b>Note:</b> This field is valid only in controllers supporting cryptographic operations (CAP.CS = 1). If CAP.CS=0, this field is reserved.		

## 6.1 UTP Transfer Request List (cont'd)

### 6.1.1 UTP Transfer Request Descriptor (cont'd)

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

DW2	Bit	Initialization Value	Description																													
	31:08	0	Reserved																													
	07:00	0Fh	<b>Overall Command Status (OCS):</b> Contains the command status of the associated command. The command status field is valid after host controller has cleared the corresponding <b>UTRLDBR</b> bit to zero.																													
			Value	Description	00h	SUCCESS	01h	INVALID_COMMAND_TABLE_ATTRIBUTES	02h	INVALID_PRDT_ATTRIBUTES	03h	MISMATCH_DATA_BUFFER_SIZE	04h	MISMATCH_RESPONSE_UPIU_SIZE	05h	COMMUNICATION_FAILURE within UIC layers	06h	ABORTED	07h	FATAL ERROR within host controller that is not covered by the error conditions described above in this table.	08h	DEVICE FATAL ERROR: A fatal error within the device	09h	INVALID_CRYPTO_CONFIGURATION	0Ah	GENERAL_CRYPTO_ERROR	0Bh-0Eh	Reserved	0Fh	INVALID_OCS_VALUE	10h-FFh	Reserved
			Value	Description																												
			00h	SUCCESS																												
			01h	INVALID_COMMAND_TABLE_ATTRIBUTES																												
			02h	INVALID_PRDT_ATTRIBUTES																												
			03h	MISMATCH_DATA_BUFFER_SIZE																												
			04h	MISMATCH_RESPONSE_UPIU_SIZE																												
05h			COMMUNICATION_FAILURE within UIC layers																													
06h			ABORTED																													
07h	FATAL ERROR within host controller that is not covered by the error conditions described above in this table.																															
08h	DEVICE FATAL ERROR: A fatal error within the device																															
09h	INVALID_CRYPTO_CONFIGURATION																															
0Ah	GENERAL_CRYPTO_ERROR																															
0Bh-0Eh	Reserved																															
0Fh	INVALID_OCS_VALUE																															
10h-FFh	Reserved																															

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

DW4	Bit	Description
	31:07	<b>UTP Command Descriptor Base Address (UCDBA):</b> 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

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

## 6.1 UTP Transfer Request List (cont'd)

### 6.1.1 UTP Transfer Request Descriptor (cont'd)

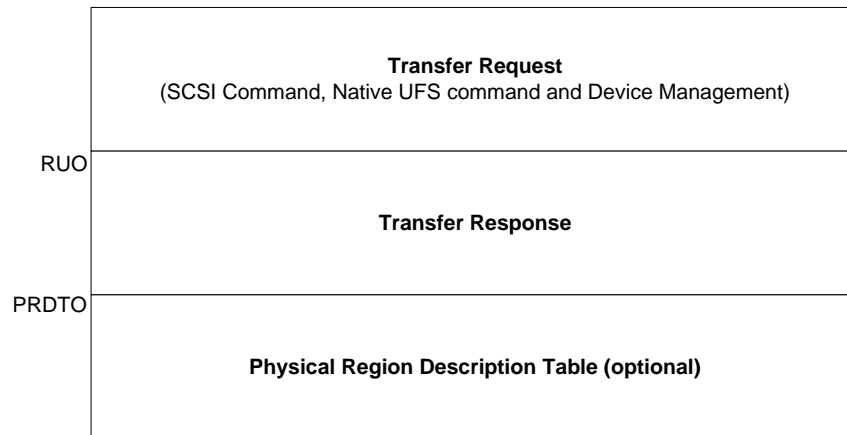
DW6	Bit	Description
	31:16	<b>Response UPIU Offset (RUO):</b> This field contains the Dword offset of the Response UPIU within the Command Descriptor. The Response UPIU may be located at a 64-bit aligned boundary right after Command UPIU in the Command Descriptor (Bit 16 is always 0).
	15:00	<b>Response UPIU Length (RUL):</b> This 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.

DW7	Bit	Description
	31:16	<b>PRDT Offset (PRDTO):</b> 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 <b>PRDTL</b> > 0.
	15:00	<b>PRDT Length (PRDTL):</b> 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 UTP Transfer Request List (cont'd)

### 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 6 — 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. Given that phase collapse is not supported, the Command UPIU's Data Segment Length field shall be 0. The NOP UPIU and Query Request UPIU do not have restrictions.

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. Since phase collapse feature will be removed from [UFS], UFSHCI will not support that feature. Consequently, receiving a DATA IN UPIU with Status phase collapse from a UFS 1.0-compliance device will result in an unspecified behavior.

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 UTP Transfer Request List (cont'd)

### 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.

	31	17	1	0
DW0	Data Base Address			0 0
DW1	Data Base Address Upper 32-bits			
DW2	Reserved			
DW3	Reserved		Data Byte Count	
			1	1

**Figure 7 — Data structure for Physical Region Description Table**

A breakdown of each field in a PRD table is shown below.

DW0	Bit	Description
	31:02	<b>Data Base Address (DBA):</b> Indicates the lower 32-bits of the physical address of the data block. The block shall be Dword aligned, indicated by bits 01:00 being reserved.
	01:00	Reserved

DW1	Bit	Description
	31:00	<b>Data Base Address Upper 32-bits (DBAU):</b> This is the upper 32-bits of the data block physical address.

DW2	Bit	Description
	31:00	Reserved

DW3	Bit	Description
	31:18	Reserved
	17:00	<b>Data Byte Count (DBC):</b> A '0' based value that indicates the length, in bytes, of the data block. A maximum of length of 256KB 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.

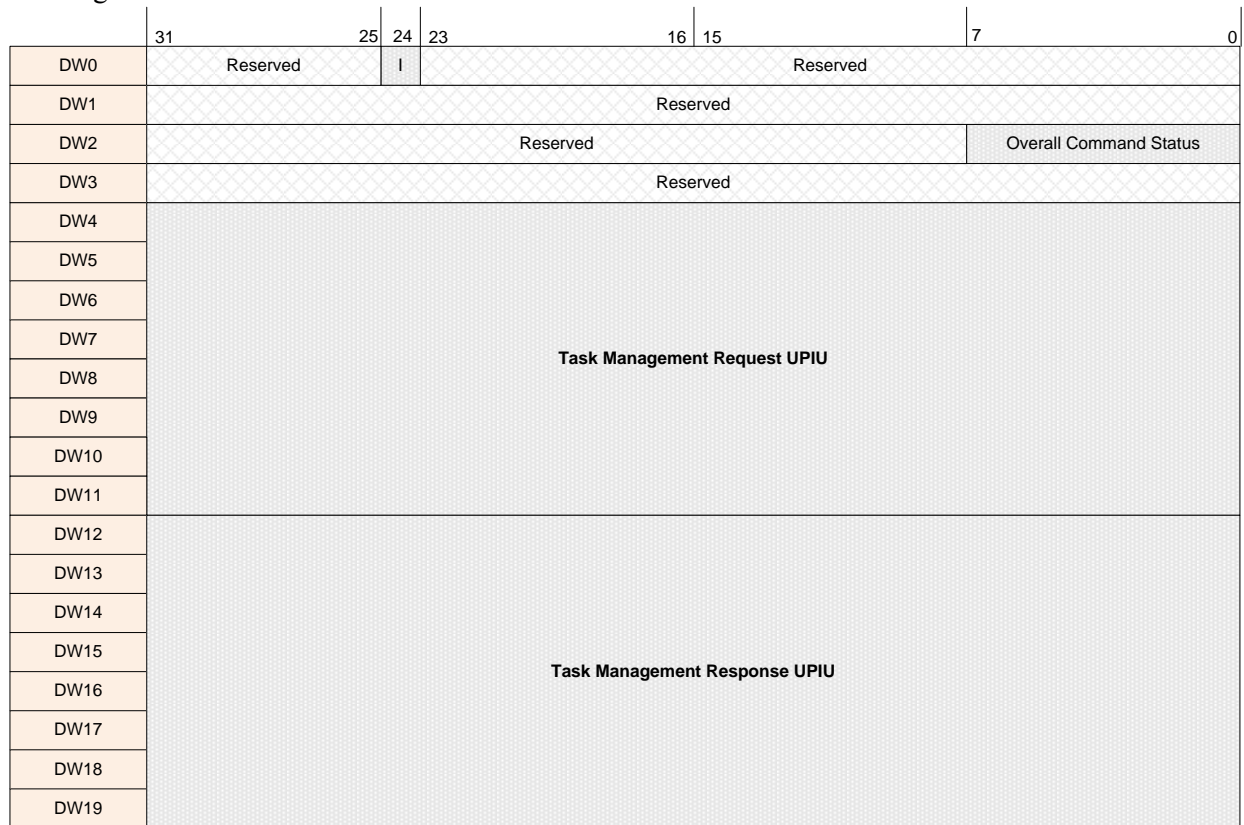


## 6.2 UTP Task Management Request List

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.

### 6.2.1 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 8.



**Figure 8 — UTP Task Management Request Descriptor.**

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.

The following section provides the description of the 1<sup>st</sup> DWORD (DW0) of the data structure.

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

## 6.2 UTP Task Management Request List (cont'd)

### 6.2.1 UTP Task Management Request Descriptor (cont'd)

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

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

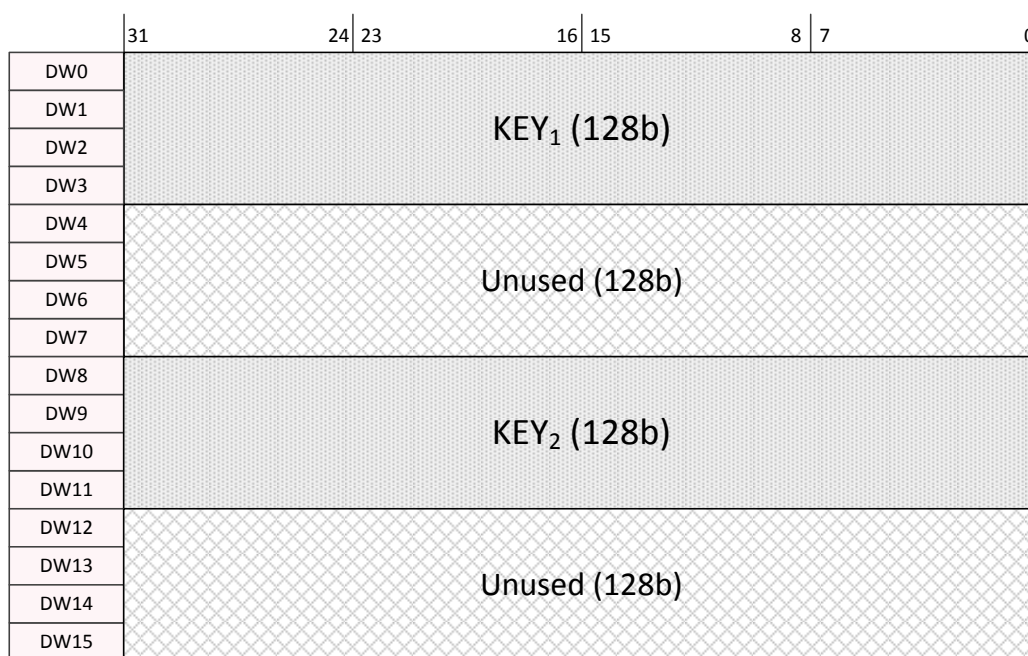
Both Task Management Request UPIU and Task Management Response UPIU are 32-Byte in length. Refer to [UFS] for the definition of Task Management Request UPIU and UTP Task Management Response UPIU.

## 6.3 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 subsection provides information about algorithm-specific organization of the CRYPTOKEY field. Additional information about the algorithms in Section 9.

### 6.3.1 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 9, Figure 10, and Figure 11, respectively.



**Figure 9 — AES128-XTS Key Layout**

6.3 Key Organization for Cryptographic Algorithms (cont'd)

6.3.1 AES-XTS (cont'd)

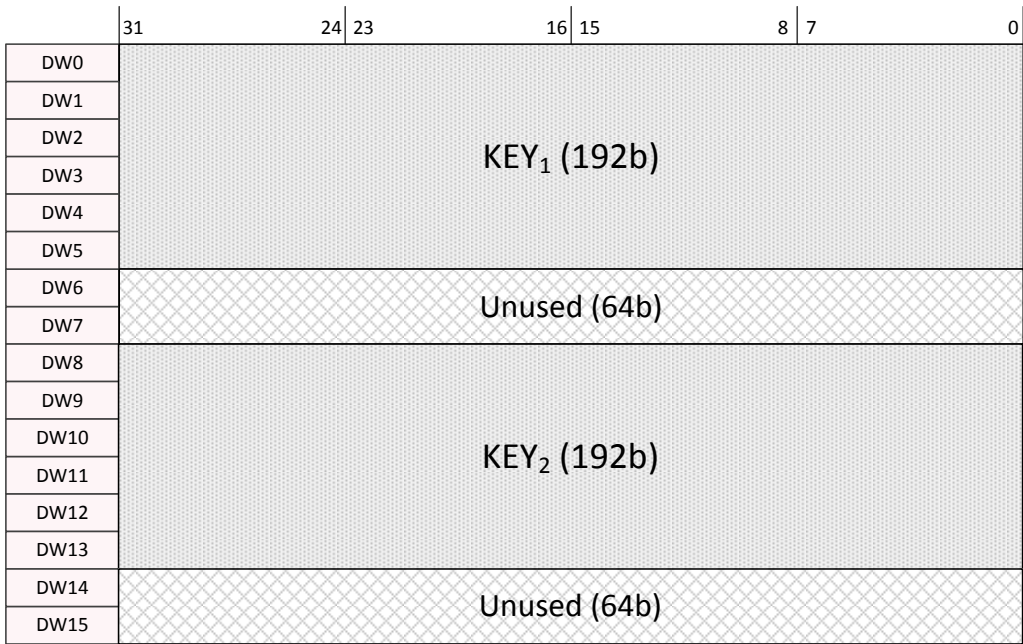


Figure 10 — AES192-XTS Key Layout

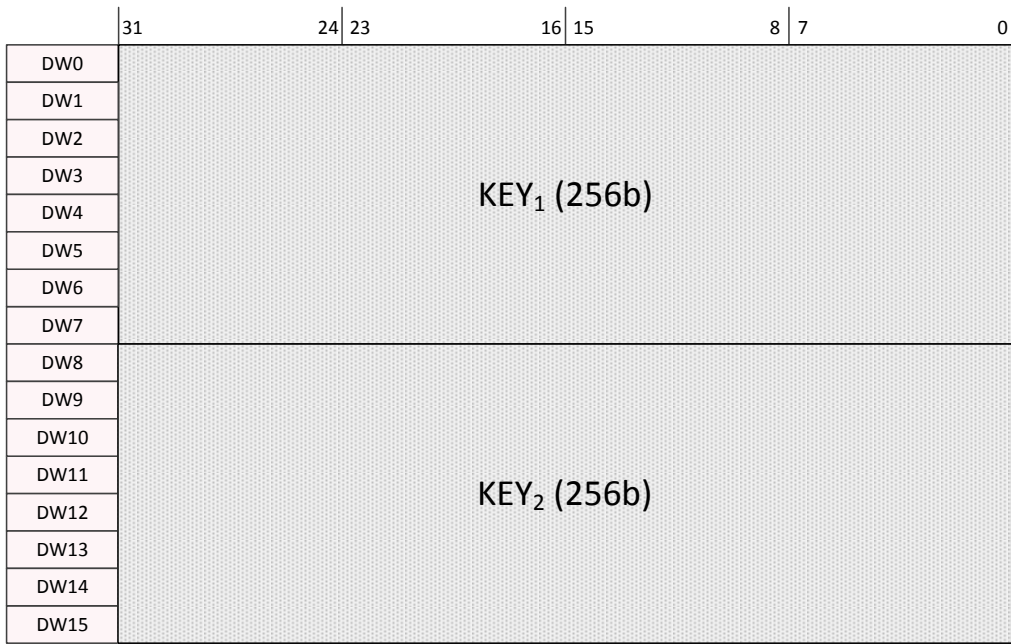
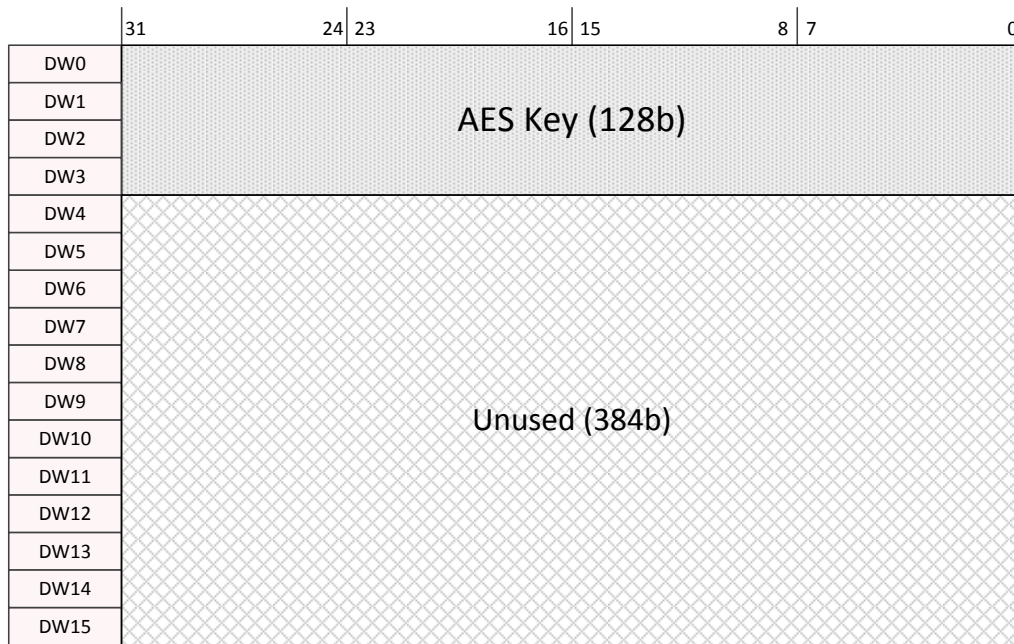


Figure 11 — AES256-XTS Key Layout

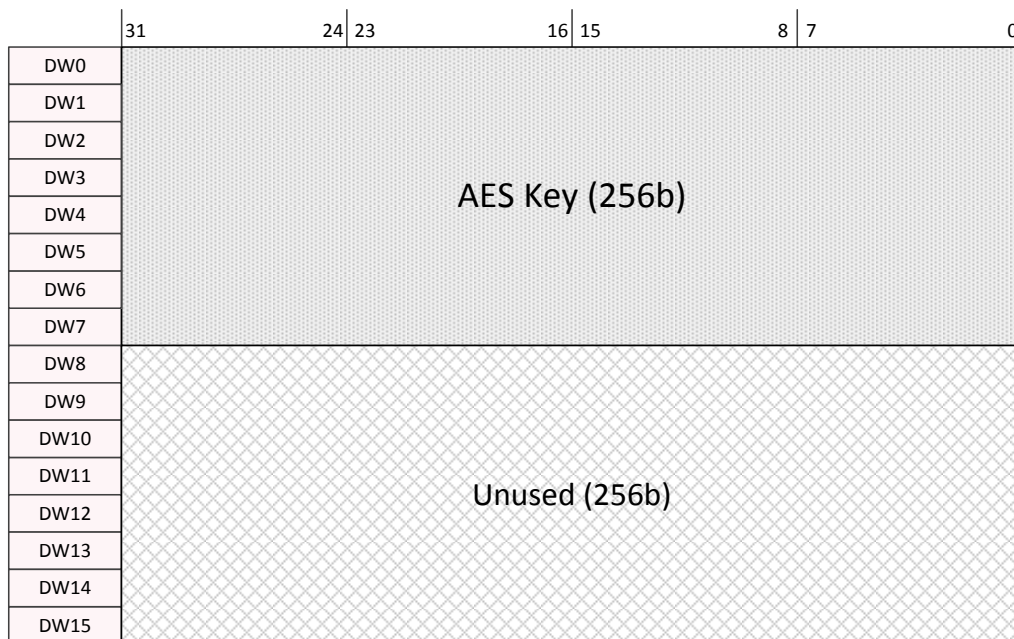
### 6.3 Key Organization for Cryptographic Algorithms (cont'd)

#### 6.3.2 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 12). For 256-bit AES-CBC, bits 0 through 255 of  $K$  are used as the AES key (as shown in Figure 13). The unused bits are ignored.



**Figure 12 — AES128-CBC Key Layout**



**Figure 13 — AES256-CBC Key Layout**

6.3 Key Organization for Cryptographic Algorithms (cont'd)

6.3.3 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 14 and Figure 15, respectively.

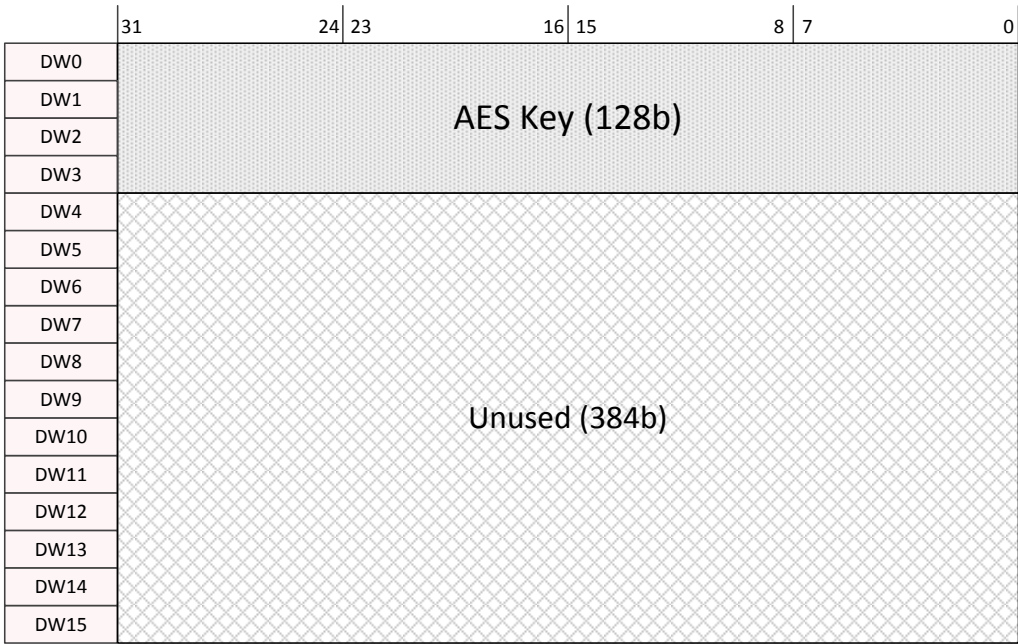


Figure 14 — AES128-ECB Key Layout

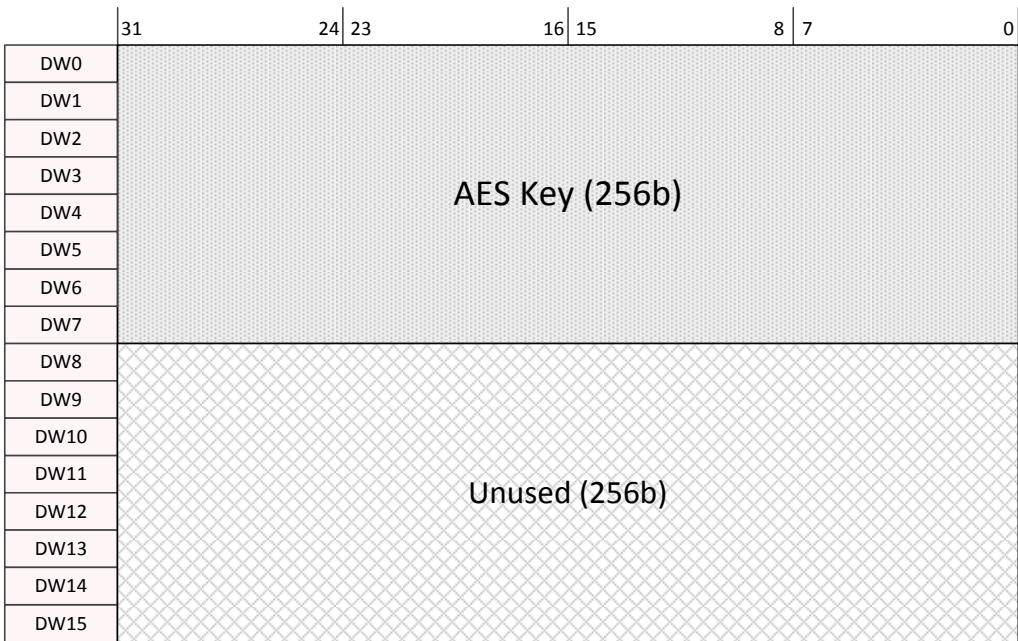


Figure 15 — AES256-ECB Key Layout

### **6.3 Key Organization for Cryptographic Algorithms (cont'd)**

#### **6.3.4 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 12). For 256-bit AES-CBC, bits 0 through 255 of K are used as the AES key (same as shown in Figure 13). 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 section 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 discussed in detail in the following sections.

## 7.1 Host Controller Configuration and Control

### 7.1.1 Host Controller Initialization

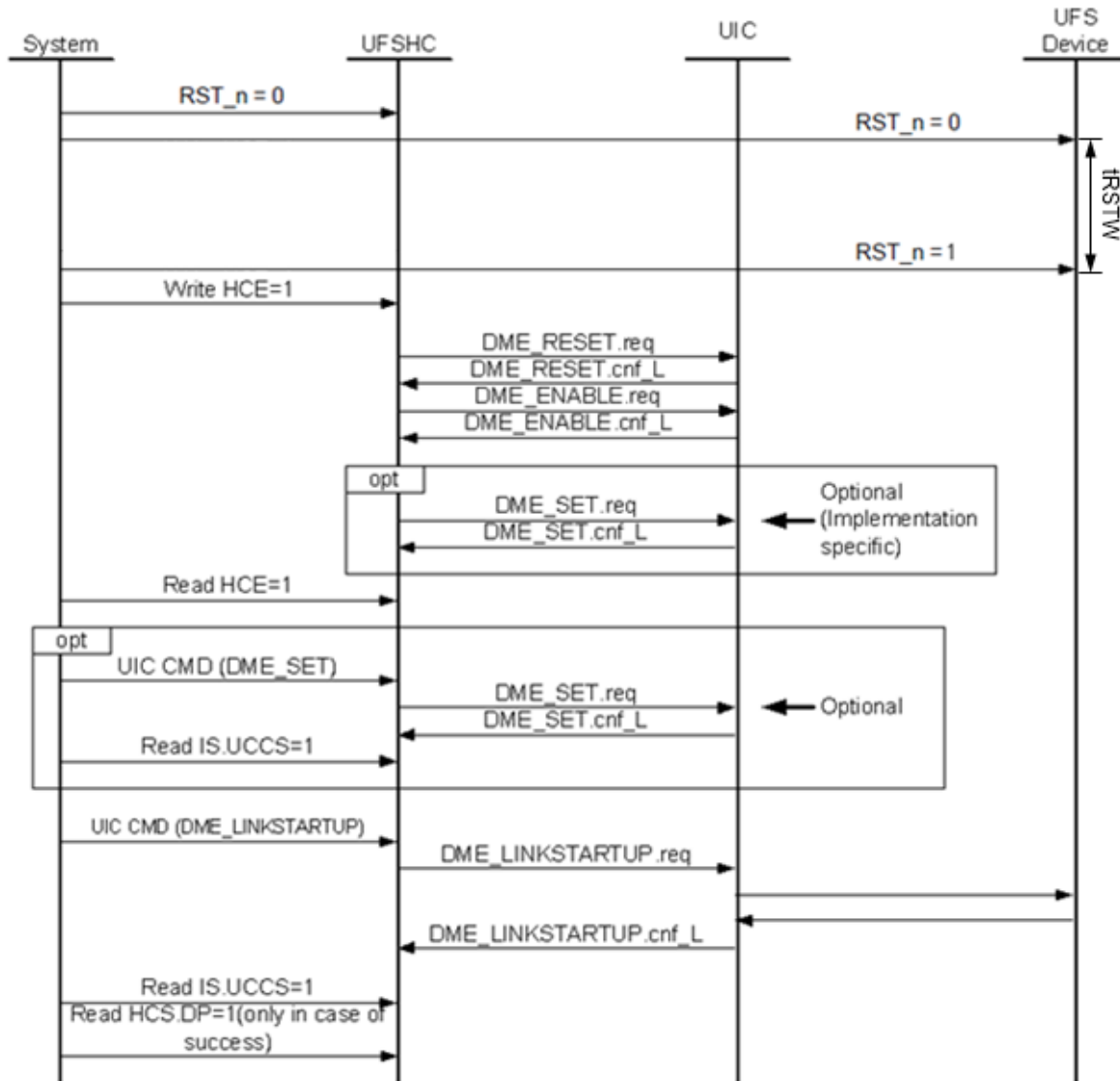
When the host controller comes out of power up reset, all MMIO registers will be in their power-on default state, and the link will be inactive. Following is a sequence of the operations that host software would perform to initialize the host controller:

- 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.
- 11) Initialize the Interrupt Aggregation Control Register (**UTRIACR**) with the desired values for the threshold (IAC<sub>TH</sub>) and timeout (IAT<sub>OV</sub>AL).  
For example, write value 0x81010664 to initialize with the following parameters: bits 31 (enable), 24 (IAP<sub>W</sub>EN), and 16 (timer/counter reset) are set, IAC<sub>TH</sub> = 6, IAT<sub>OV</sub>AL = 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 step 13.
- 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 step 15.
- 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 Door Bell 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 Door Bell mechanism.



## 7.1 Host Controller Configuration and Control (cont'd)

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



### Figure 16 — Host controller link startup sequence

## 7.1 Host Controller Configuration and Control (cont'd)

### 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 6.3. 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 Data Transfer Operation (cont'd)

### 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.
  1. Program field **CT** (command type) to indicate the command type: SCSI, native UFS command or device management function.
  2. Program field **DD** (data direction) that contains the direction of the data operations that are a part of the command if any.
  3. **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.
  4. Initialize **OCS** with 'Fh'.
  5. Allocate and initialize a UCD.
  6. Program field *Command UPIU* in UCD with a UTP command excluding task management function.
  7. Initialize field *Response UPIU* in UCD with '0'.
  8. 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.
9. Check register **UTRLRSR** and make sure it is read '1' before continuing.
10. Set *UTP Transfer Request Interrupt Aggregation Control Register* (UTRIACR) enable bit to '1' to enable the interrupt.
11. Set *Counter and Timer Reset* (CTR) bit to '1' to reset the counter and timer associated with the interrupt.
12. Program field *Interrupt aggregation counter threshold* (IACTH) with the number of command completions that are required to generate an interrupt.
13. 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.
14. 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 Data Transfer Operation (cont'd)

### 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. Only the 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	LUN, Task Tag	To match future incoming Response
Task Management Request		To match future incoming TM Response
Query Request	Task Tag	To match future incoming Query Response
NOP Out		To match future incoming NOP In

#### 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.

**Table 2 — Outbound UPIU generated by UTP Engine**

UPIU Type	Relevant UPIU Fields	UTP Engine actions on UPIU fields
Data Out	Transaction Type	Set to xx00 0010b
	LUN, Task Tag	Filled with LUN and Task Tag information from the corresponding to 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	Fetches from PRDT buffers based on the DMA context information supplied in the corresponding RTT UPIU

## 7.2.2 UPIU Processing (cont'd)

### 7.2.2.3 Inbound UPIUs interpreted by Software

The UTP Engine shall analyze the 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 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 Door Bell 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	LUN, Task Tag	Identify corresponding Command
Task Management Response		Identify corresponding TM Request
Query Response	Task Tag	Identify corresponding Query Request
NOP In		Identify corresponding NOP Out

### 7.2.2.4 Inbound UPIUs interpreted 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
	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.

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.

## 7.2.2 UPIU Processing (cont'd)

### 7.2.2.4 Inbound UPIUs interpreted by Host Controller/UTP Engine (cont'd)

**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.
	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.

## 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):

- The IA interrupt counter is incremented.
- 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:

- The UTRD.I bit is set (Interrupt Command)
- The counter, after incrementing, reaches the value configured in IACTH
- The IA timer reaches the value configured in IATOVAL (this event may occur at any time, not necessarily coupled with request completion).
- 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.

## 7.2 Data Transfer Operation (cont'd)

### 7.2.3 Processing UTP Transfer Request Completion (cont'd)

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:
  - 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.
  - 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 steps:
  - 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 **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 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 the 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 section.

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:

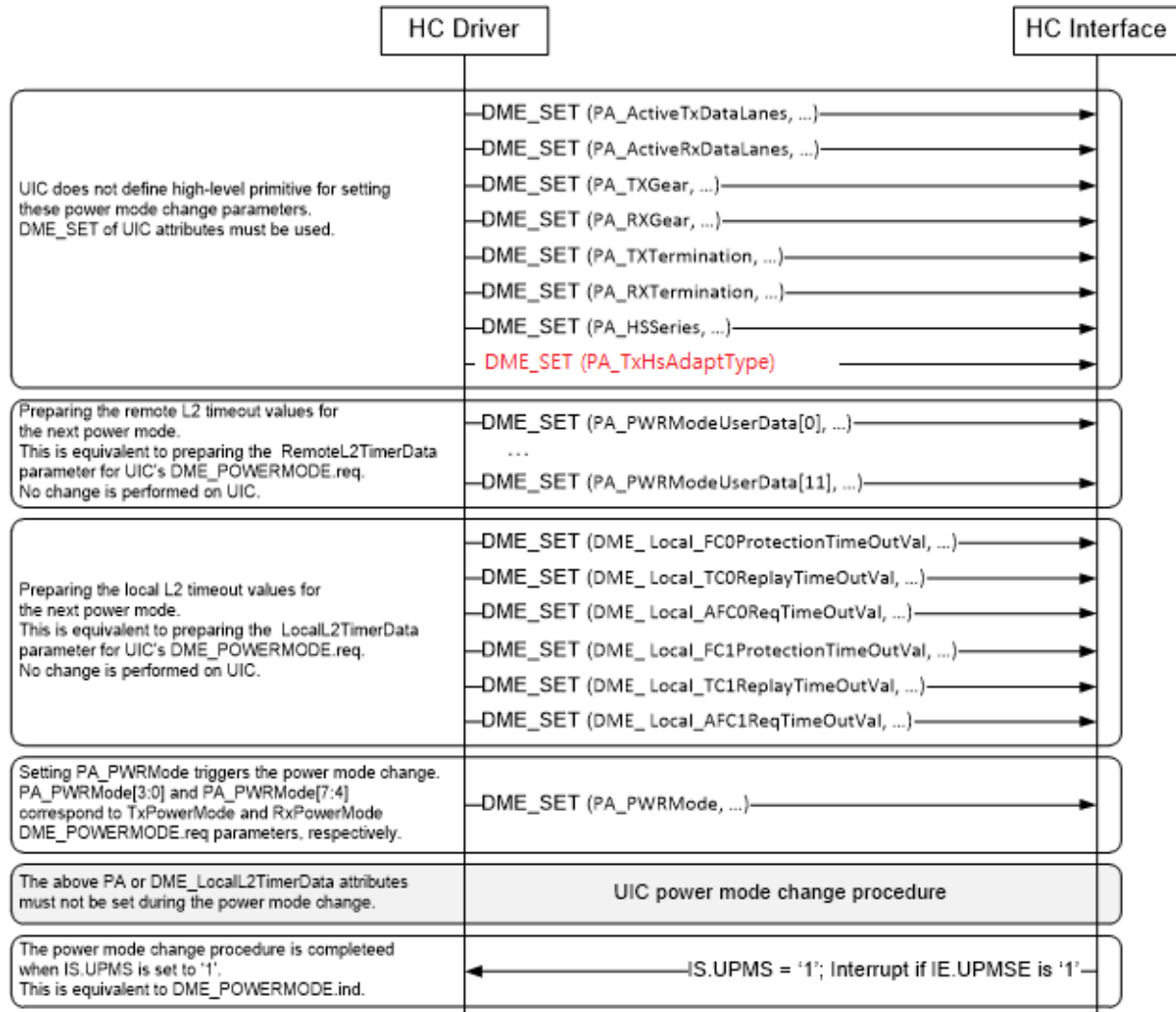
- DME\_Local\_FC1ProtectionTimeOutVal
- DME\_Local\_TC1ReplayTimeOutVal
- DME\_Local\_AFC1ReqTimeOutVal

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 in JESD220B.

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 17 — 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 speed:

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

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

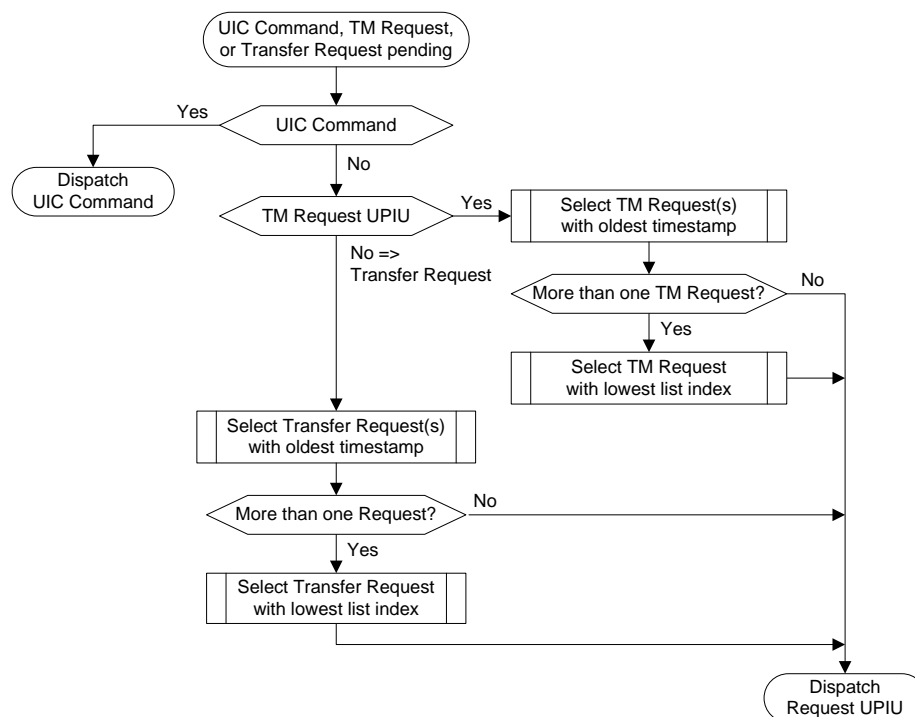
UFSHCI processes three types of commands, which shall be processed using the following priority (in order of priority, highest priority first):

- 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 Door Bell 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.



**Figure 18 — Command processing order**

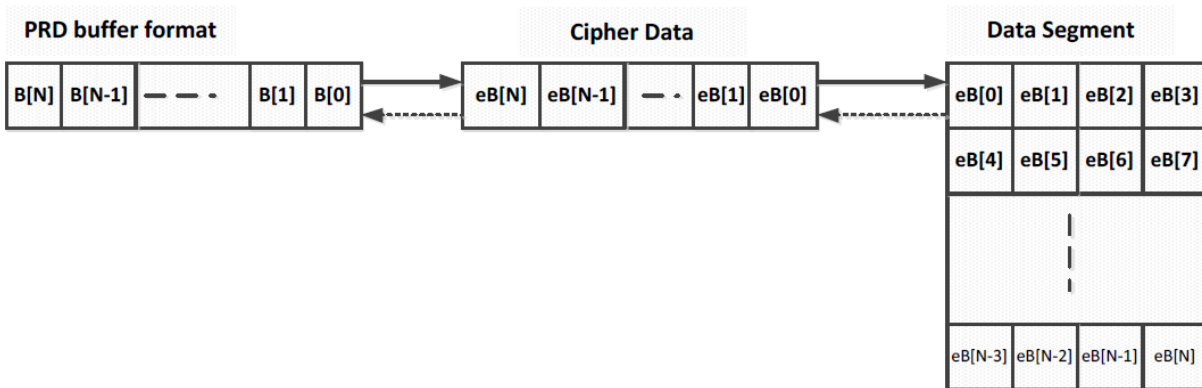
## 7.5 UFSHCI Internal Rules (cont'd)

### 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 19 for both encryption and decryption.



**Figure 19 — Byte Order For Data Unit Processing in Cryptographic operations**

---

## 8 Error reporting and handling

---

This section 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 provide 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 5.3.5 through 5.3.9 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 Error Types (cont'd)

### 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 Error Handling (cont'd)

### 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 16.

## 9 Encryption Engine Details (Informative)

The algorithms described in this section 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 section may be constrained compared to the definition in its standard specification. This section specifies the limitations and constraints applicable to cryptographic operations in a UFS host controller.

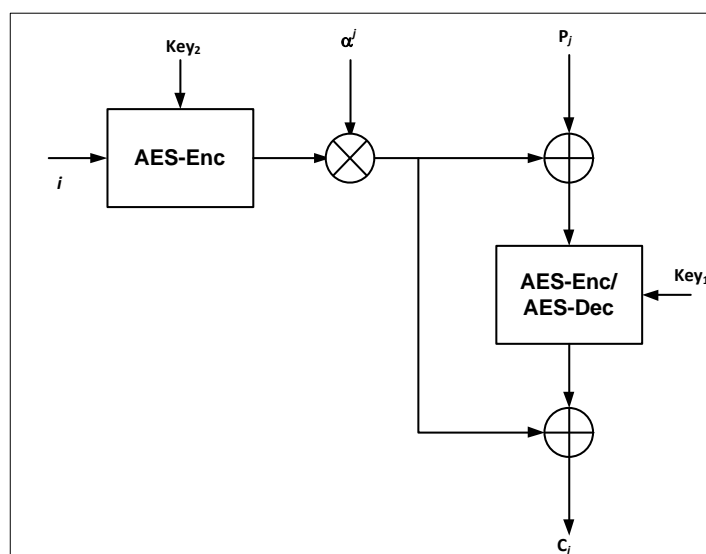
In addition, the terminology used in this specification might differ from the algorithm specifications. This section clarifies the terms used, and maps the usage of the transaction parameters to each specific algorithm.

### 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 20 — AES-XTS Encryption**

The diagram in Figure 20 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, “ $t$ ” 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:

$$t = \text{Data Unit Number (DUN)} \parallel 00000\dots (128 \text{ bits})$$

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.

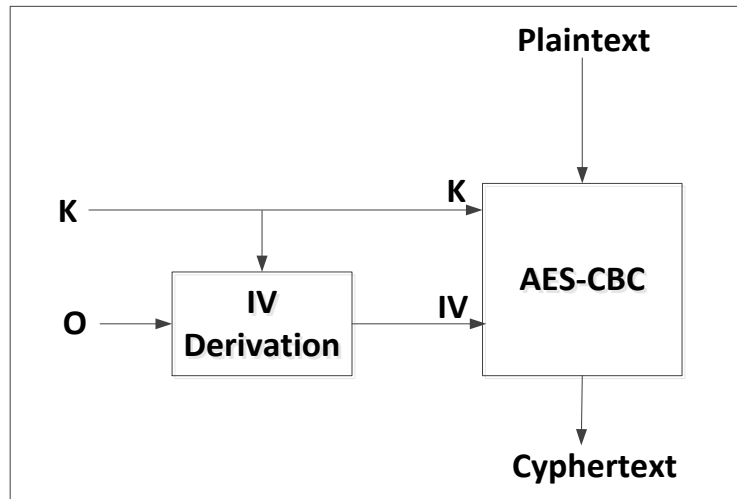
Complete information about the two AES-CBC algorithms is provided in the paper “*AES-CBC + Elephant diffuser; A Disk Encryption Algorithm for Windows Vista by Niels Ferguson, August 2006*” available on <http://download.microsoft.com>. Information about AES-CBC without Elephant Diffusers is provided in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

### 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.<sup>1</sup>
- Sector offset  $O$ .
- Sector data, containing  $S$  bytes.

Figure 21 illustrates the encryption process using Microsoft Bitlocker™ AES-CBC.



**Figure 21 — 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.

### 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.

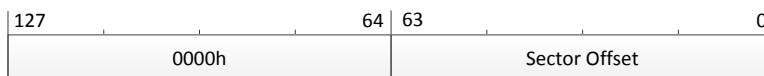
For the first sector of a transaction, the offset  $O$  is initialized with the value of the DUN field of the UTRD. For subsequent sectors, the offset  $O$  is recalculated at the start of every sector.

---

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

### 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 22. 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 22 — 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.

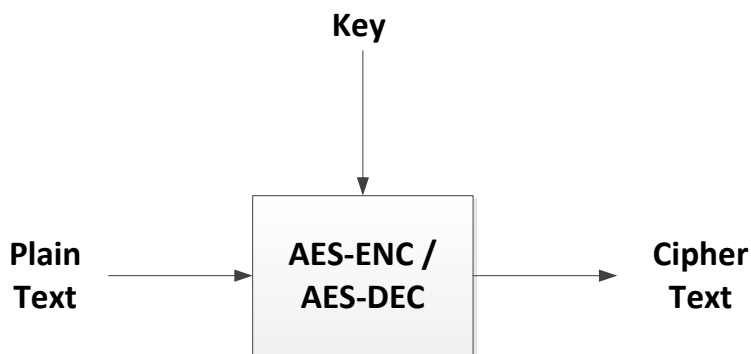
Decryption is the obvious inverse. The same algorithm is used to compute the IV, and then the encrypted sector data is decrypted with AES-CBC and the computed IV.

## 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 23.

The data unit number is not used.



**Figure 23 — 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) section 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:

$$(SN) = AES_s(SN) \text{ where } s = SHA256(key)$$

### 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.

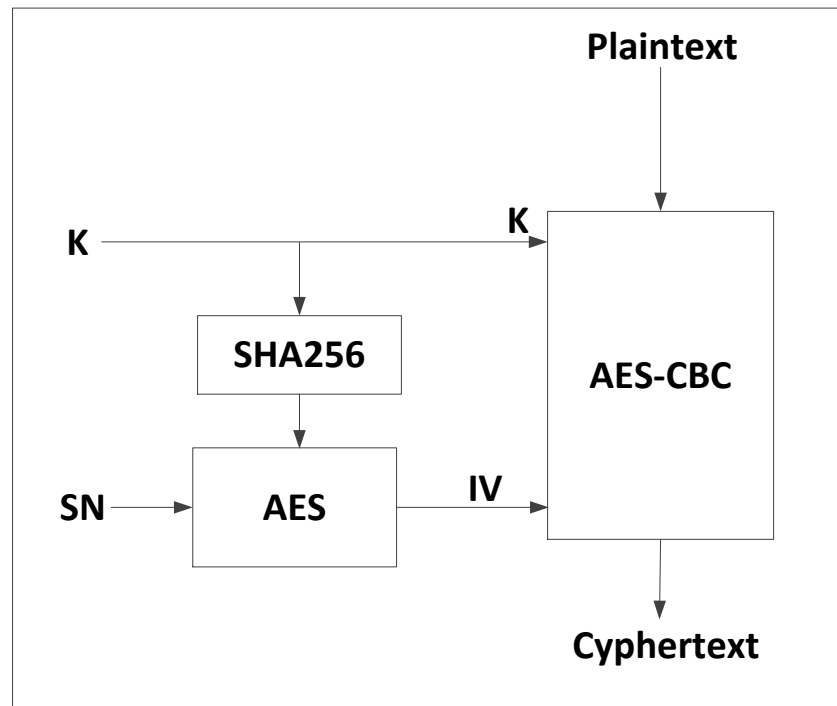


Figure 24 — ESSIV-AES-CBC Encryption

---

**Annex A (informative) Differences between JESD223D and JESD223C**

---

This annex briefly describes most of the changes made to entries that appear in this standard, JESD223D, compared to its predecessor, JESD223C (March 2016). 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.

**Description of change**

- 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.1 Differences between JESD223C and JESD223B (September 2013)****Description of change:**

- 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.2 Differences between JESD223B and JESD223A (June 2012)****Description of change:**

- All references are revised to be consistent with UFS standard reference convention [UFS].
- Upgraded MIPI UniPro standard [MIPI-UNIPRO] from version 1.41 to 1.6.
- Upgraded MIPI M-PHY 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.





---

**STANDARD IMPROVEMENT FORM****JEDEC JESD223D**

---

The purpose of this form is to provide the Technical Committees of JEDEC with input from the industry regarding usage of the subject standard. Individuals or companies are invited to submit comments to JEDEC. All comments will be collected and dispersed to the appropriate committee(s).

If you can provide input, please complete this form and return to:

JEDEC  
Attn: Publications Department  
3103 North 10<sup>th</sup> Street  
Suite 240 South  
Arlington, VA 22201-2107

Fax: 703.907.7583

---

1. I recommend changes to the following:

☐ Requirement, clause number \_\_\_\_\_

☐ Test method number \_\_\_\_\_ Clause number \_\_\_\_\_

The referenced clause number has proven to be:

☐ Unclear ☐ Too Rigid ☐ In Error

☐ Other \_\_\_\_\_

---

2. Recommendations for correction:

---

---

---

---

---

3. Other suggestions for document improvement:

---

---

---

---

---

Submitted by

Name: \_\_\_\_\_

Phone: \_\_\_\_\_

Company: \_\_\_\_\_

E-mail: \_\_\_\_\_

Address: \_\_\_\_\_

City/State/Zip: \_\_\_\_\_

Date: \_\_\_\_\_

---

