JEDEC STANDARD

Universal Flash Storage Host Controller Interface (UFSHCI), Unified Memory Extension

Version 1.1A

JESD223-1B
(Revision of JESD223-1A, March 2016)

MAY 2016

JEDEC SOLID STATE TECHNOLOGY ASSOCIATION

Published by
©JEDEC Solid State Technology Association 2016
3103 North 10th Street
Suite 240 South
Arlington, VA 22201-2107

PRICE: Contact JEDEC

UNIVERSAL FLASH STORAGE HOST CONTROLLER INTERFACE (UFSHCI), UNIFIED MEMORY EXTENSION, Version 1.1

Contents

Foreword

This Unified Memory Extension standard is an extension to the UFSHCI standard, JESD223.

Introduction

The UFSHCI standard defines the interface between the UFS driver and the UFS host controller. In addition to the register interface, it defines data structures inside the system memory, which are used to exchange data, control and status information. Furthermore the UFSHCI standard defines the protocol layer structure and abstract entities within these layers.

Unified Memory offers the possibility to move Device internal working memory into the system memory to reduce overall system cost and to improve Device performance.

The Unified Memory feature impacts Host and Device side. This standard, UFSHCI Unified Memory Extension, describes the requirements to implement Unified Memory functionality in a UFS Host Controller. It is based on the “UFS Unified Memory Extension” standard, which describes the general Unified Memory protocol.

UNIVERSAL FLASH STORAGE HOST CONTROLLER INTERFACE (UFSHCI), UNIFIED MEMORY EXTENSION, Version 1.1

(From JEDEC Board Ballot JCB-13-40 and JCB-15-61, under the cognizance of the JC-64.1 Subcommittee on Electrical Specifications and Command Protocols.)

1 Scope

This document provides a comprehensive definition of the requirements for implementation of a UFS Host Controller, which supports the optional Unified Memory extension.

2 Normative Reference

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

[MIPI-UniPro], *MIPI Alliance Specification for Unified Protocol (UniPro^{SM}), Version 1.6*

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

[UFSHCI], JEDEC JESD223C, *Universal Flash Storage Host Controller Interface (UFSHCI), Version 2.1*

[UFS], JEDEC JESD220C, *Universal Flash Storage (UFS), Version 2.1*

[UFS-UME], JEDEC JESD220-1A, *UFS Unified Memory Extension, Version 1.1*

3 Keywords, Abbreviations, Acronyms, and Conventions

3.1 Keywords

Several keywords are used to differentiate levels of requirements and options, as follow:

Can: A keyword used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

Expected: A keyword used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

Ignored: A keyword that describes bits, bytes, quadlets, or fields whose values are not checked by the recipient.

Mandatory: A keyword that indicates items required to be implemented as defined by this standard.

May: A keyword that indicates a course of action permissible within the limits of the standard (*may* equals *is permitted*).

Must: The use of the word *must* is deprecated and shall not be used when stating mandatory 61 requirements; *must* is used only to describe unavoidable situations.

Optional: A keyword that describes features which are not required to be implemented by this standard. However, if any optional feature defined by the standard is implemented, it shall be implemented as defined by the standard.

Reserved: A keyword used to describe objects—bits, bytes, and fields—or the code values assigned to these objects in cases where either the object or the code value is set aside for future standardization. Usage and interpretation may be specified by future extensions to this or other standards. A reserved object shall be zeroed or, upon development of a future standard, set to a value specified by such a standard. The recipient of a reserved object shall not check its value. The recipient of a defined object shall check its value and reject reserved code values.

Shall: A keyword that indicates a mandatory requirement strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*). Designers are required to implement all such mandatory requirements to assure interoperability with other products conforming to this standard.

Should: A keyword used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).

Will: The use of the word *will* is deprecated and shall not be used when stating mandatory requirements; *will* is only used in statements of fact.

3 Keywords, Abbreviations, Acronyms, and Conventions (cont'd)

3.2 Abbreviations

etc. - And so forth (Latin: et cetera)
e.g. - For example (Latin: exempli gratia)
i.e. - That is (Latin: id est)

3.3 Acronyms

BOM	Bill Of Material
DMA	Direct Memory Access
DRAM	Dynamic Random Access Memory
LUN	Logical Unit Number
NVM	Non-Volatile Memory
PC	Personal Computer
SAP	Service Access Point
SCSI	Small Computer System Interface
SRAM	Static Random Access Memory
TC	Traffic Class
UBMTP	UFS Bus Master Transport Protocol
UFS	Universal Flash Storage
UM	Unified Memory
UMA	Unified Memory Architecture
UME	Unified Memory Extension
UMPIU	Unified Memory Protocol Information Unit
UniPro	Unified Protocol.
UTP	UFS Transport Protocol

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:

HwInit	The default state is dependent on device and system configuration. The value is initialized at reset, either by an expansion ROM, or in the case of integrated devices, by a platform BIOS.
Impl Spec	Implementation Specific – the controller has the freedom to choose its implementation.
RO	Read Only
ROC	Read Only and Read to clear
RW	Read Write
R/W	Read Write. The value read may not be the last value written.
RWC	Read/Write '1' to clear
RWS	Read/Write '1' to set

4 Unified Memory Functional Requirements

4.1 Unified Memory Overview

Ever increasing demand for higher storage performance and lower cost in the consumer market put tight constraints on UFS Device and Host vendors. One way to improve both, performance and cost, is to use Unified Memory. This concept offers the option to move Device internal working memory into the Host memory, which is already available in large capacities in current and upcoming Smartphone, Tablet and portable computer generations. Furthermore the Unified Memory Architecture (UMA) concept is not new since it has been successfully used for many years in the PC and Workstation market.

High performance, in particular random read and write accesses, requires a huge amount of buffer and cache memory. This may be either implemented as SRAM on the UFS Device controller die or as a separate DRAM inside the UFS Device package. On-die SRAM increases the cost of the controller and DRAM dies in relatively small capacities will become unavailable or expensive in the future since they will become a non-mainstream niche product.

Moreover, due to UM, the SRAM size on the Device side may be reduced, which also reduces pellet cost, footprint, leakage and power consumption. Also making a dedicated DRAM die obsolete reduces the BOM of the UFS Device, increases the assembly yield, eliminates the need for an embedded DRAM controller and reduces the power consumption which is required for the DRAM refresh. One die less inside the UFS Device package offers the Device vendor the possibility to add another NVM die to increase the overall NVM capacity.

4.2 Unified Memory Layer Structure

The basic UFSHCI layer structure as shown in [UFSHCI] is solely based on the SCSI Architecture Model [SAM], which defines a strict client-server model that does not permit the UFS Device to issue requests to the UFS Host Controller. This is called “Device Bus Master” mode. Figure 1 depicts the extended UFS Host Controller Layer diagram, which adds “Device Bus Master” mode functionality to the Host Controller. Other future functionalities like e.g., UFSIO might use the “Device Bus Master” mode as well.

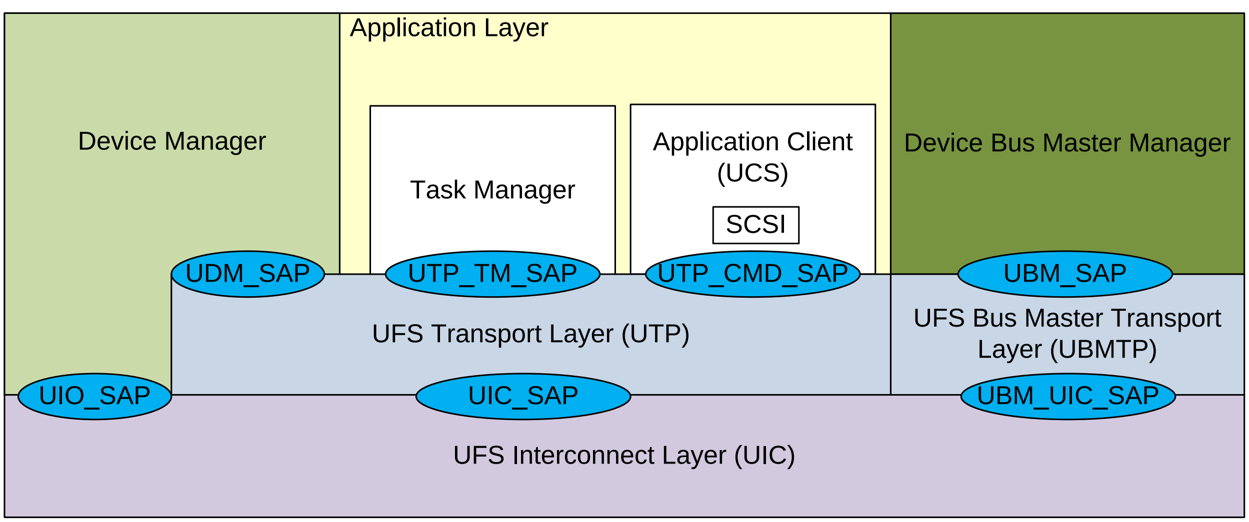

Figure 1 — Extended UFSHCI Layer Structure

4.2 Unified Memory Layer Structure (cont'd)

The following sections describe additional SAPs, Layers and Entities, that are required for UM operation, in detail.

4.2.1 UFS Bus Master Transport Layer (UBMTP)

The purpose of this layer is to interpret inbound UMPIU packets and assemble outbound UMPIU packets. In contrast to the UTP layer, the UBMTP layer supports requests from the UFS Device to the UFS Host Controller.

The UBMTP Layer is based on the UTP layer but optimized to the requirements of the UM operation (e.g., reduced header size to minimize load on the UniPro link).

4.2.2 Device Bus Master Manager

This entity is the counterpart of the same entity on the UFS Device side.

The "Device Bus Master Manager" contains a DMA Engine, which has direct access to the System Bus without going through the Application Layer. The benefits are that the UM operation is transparent to the Application Layer and hence the additional Software impact and load on the System Host is minimized. The DMA Engine is directly controlled by the UFS Device and hence shall be considered as a property of the Device.

In addition to this, the "Device Bus Master Manager" handles the UM protocol and sends responses to the UFS Device.

The host controller shall be able to support at least 32 outstanding UMPIU request up to a maximum that is defined by the MNOOUR field inside the UMACAP register. An outstanding request could be a COPY DATA, UM COPY DATA, ACCESS UM BUFFER or WRITE UM BUFFER UMPIU (see [UFS-UME]).

To realize the COPY DATA and UM COPY DATA UM commands, the DMA Engine shall be capable of performing Host Memory-to-Host Memory copy transactions.

4.2.3 UBM_SAP

This Service Access Point is used for communication between the "Device Bus Master Manager" and the "UFS Bus Master Transport Layer".

4.2 Unified Memory Layer Structure (cont'd)

4.2.4 UBM_UIC_SAP

This Service Access Point is used for communication between the "UFS Bus Master Transport Layer" and the "UFS InterConnect" Layer.

The packet format is UMPIU (See [UFS-UME]). The physical representations of UBM_UIC_SAP are UniPro Cport 1 (TC1) and Cport 2 (TC0).

The UFSHCI Unified Memory Extension functionality doesn't mandate any arbitration scheme between Cports.

According to the UniPro specification [UNIPRO], the application shall immediately consume data offered by a Cport to avoid backpressure on the UniPro link. Since this may have negative impact on other Cports of the same TC, the UFS Host Controller implementation shall contain sufficient buffer space to guarantee the requirement given by the UniPro specification.

4.3 Register Map

The following is an extension to the standard register map for UFSHCI.

	Start	End	Symbol	Description
Host Capabilities	00h	03h	CAP	Host Controller **Cap**abiities
	04h	07h	UMACAP	**U**nified **M**emory **A**rchitecture **C**apabilities
UMA	B0h	B3h	UMABA	**U**nified **M**emory **A**rea **B**ase **A**ddress
	B4h	B7h	UMABAU	**U**nified **M**emory **A**rea **B**ase **A**ddress **U**pper
	B8h	BBh	UMAOMAX	**U**nified **M**emory **A**rea **O**ffset **MAX**
	BCh	BFh	UMACONF	**U**nified **M**emory **A**rchitecture **Conf**iguration

4.3.1 Offset 00h: CAP – Controller Capabilities

Bit	Type	Reset	Description
27	RO	Impl Spec	**Device Bus Master Mode supported (DBMMS)**: Indicates whether the UFS host controller supports the device bus master mode, which is required to support the Unified Memory feature. The device bus master mode allows sending requests from the UFS device to the UFS host controller. The host controller responds to those requests.

4.3.2 Offset B0h: UMABA – Unified Memory Area Base Address

Bit	Type	Reset	Description
31:10	RW	3F_FFFFh	**Unified Memory Area Base Address (UMABA):** This register defines the lower 32bit of the Unified Memory Area Base Address. The address defined inside this field is a 1 KB aligned physical byte address inside the system memory. The register can be written only one time. After a power cycle or hardware reset event, the register is set to the default value NOTE This register value shall match the corresponding base address attribute value inside the device.
9:0	RO	Reserved	Reserved

4.3 Register Map (cont'd)

4.3.3 Offset B4h: UMABAU – Unified Memory Area Base Address Upper

Bit	Type	Reset	Description
31:0	RW	FFFF_FFFF h	**Unified Memory Area Base Address Upper (UMABAU):** This register defines the upper 32bit of the Unified Memory Area Base Address. The address defined inside this field is a physical byte address inside the system memory. The register can be written only one time. After a power cycle or hardware reset event, the register is set to the default value NOTE This register value has to match the corresponding base address attribute value inside the device.

4.3.4 Offset B8h: UMAOMAX – Unified Memory Area Offset MAX

Bit	Type	Reset	Description
31:10	RW	00_0000h	**Unified Memory Area Offset MAX (UMAOMAX):** This register defines the upper boundary of the Unified Memory area. In order to prohibit unrightful access to non UMA memory locations, the maximum offset from the Unified Memory Area Base Address (UMABA, UMABAU) is specified in this register. The sum of Unified Memory Area Base Address and UMAOMAX value minus 1 (zero-based value) shall be smaller or equal to the largest addressable physical system memory address. No wrap around is supported. The maximum offset in this register is given in bytes. Example: To define a 16MB UM Area the value 01_0000h needs to be programmed into the UMAOMAX register. UMAOMAX shall be 1KB aligned as indicated by bits 9:0 being read only. The UFS host controller shall evaluate equation (1) for every received COPY DATA, UM COPY DATA and ACCESS UM BUFFER UMPIU and equation (2) for every WRITE UM BUFFER UMPIU. (1) Target UM Area Offset + Data Length > UMAOMAX (2) Target UM Area Offset + Data Segment Length > UMAOMAX In case the result of the evaluation is TRUE, the UFS host controller shall disregard the request and directly respond by sending an ACKNOWLEDGE COPY or ACKNOWLEDGE UM BUFFER UPIU, depending on the original request. In this case, the "I" and the "E" flags shall be set to "1". The host controller does not trigger any operation on the system bus. The register can be written only one time. After a power cycle or hardware reset event, the register is set to the default value
9:0	RO	000h	Reserved

4.3.5 Offset BCh: UMACONF – Unified Memory Architecture Configuration

Bit	Type	Reset	Description
31:9	RO	Reserved	Reserved
8	RW	0h	**CportConfEn:** Cport Configuration Enable 0h Use of Cport s are defined as follows Cport 0 for regular UFS, Cports 1 and 2 for UM operations 1h Reserved for future Cport configurability
7:1	RO	Reserved	Reserved
0	RW	0h	**UMEn**: Unified Memory Enable 0h Unified Memory functionaly disabled. Host Controller shall ignore all UM requests from the UFS Device. 1h Unified Memory functionaly enabled

NOTE Other CportConfEn values than 0h are reserved in this version of the "UFSHCI Unified Memory Extension" because in a single-application UniPro system no configurability is required. Future versions of UniPro may allow multiple applications sharing one UniPro stack. In this case two or more applications may expect access to the same Cport. In this case a generic configuration protocol on UniPro level shall be used to dissolve this multi-assignment situation. Once a multi-application UniPro stack with a proper configuration protocol is available, other values than 0h may be defined to allow usage of a generic configuration protocol. By this, the UFS Host Controller Cport allocation stays flexible and future proof without the risk of incompatibility with a future generic configuration protocol.

4.3.6 04h: UMACAP - Unified Memory Architecture Capability

Bit	Type	Reset	Description
31:3	RO	Reserved	Reserved
2:0	RO	Impl Spec	**Maximum Number Of Outstanding UMPIU Requests (MNOOUR):** Indicates the maximum number of outstanding UMPIU requests that are supported by the UFS host controller implementation. This value should be written into the bMaxUMPIURequests attribute of the UFS Device. The maximum number of outstanding requests is defined by the following formula. 32 x (MNOOUR + 1)

4.4 Host Memory Map

Unified Memory means that a certain portion of the Host memory is dedicated to the UFS Device and shall be considered as its property. The remaining Host memory is usable by the Operating System. To avoid sophisticated scatter-gather list handling by the UFS Device, the Unified Memory region shall be a continuous chunk of memory. In order to guarantee that sufficient continuous system memory is available, the bootloader shall already reserve that memory and only report the remaining capacity to the Operating System kernel.

Two parameters define the Unified Memory area, a base address and a maximum offset from this base address. These parameters are defined by the UMABA, UMABAU and UMAOMAX registers.

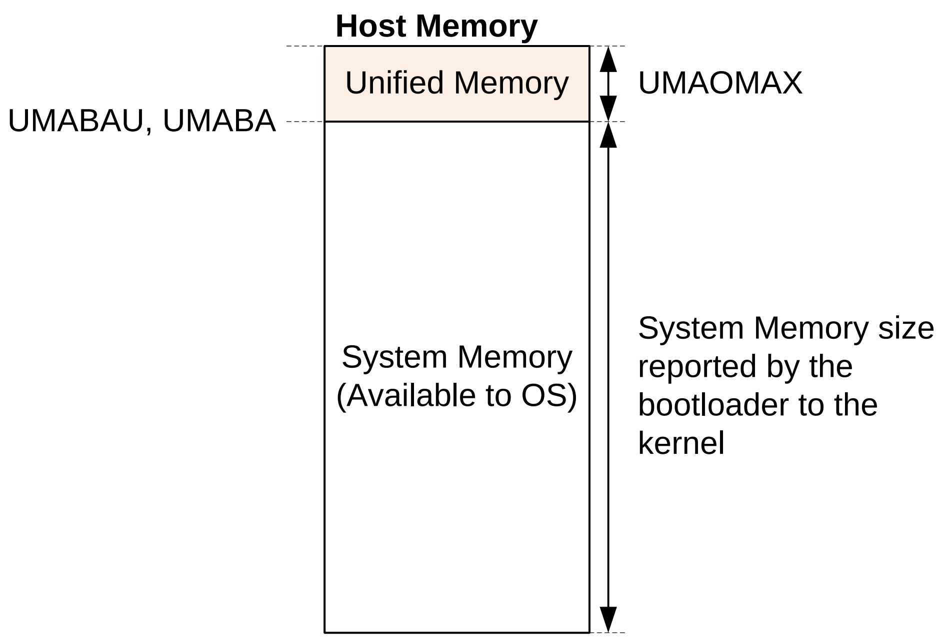

Figure 2 — System Memory Structure

4.5 UMPIU Processing

4.5.1 Outbound UMPIUs generated by Software

No outbound UMPIUs are generated by Software since the Unified Memory extension shall be as transparent as possible for the Software and hence it should not add additional load on the system host.

4.5.2 Outbound UMPIUs generated by Host Controller/UBMTP Engine

All outbound UMPIUs are generated by the UBMTP Engine inside the UFS Host Controller to keep the response latency as low as possible and to avoid putting additional load on the system host and system bus. Furthermore impact on existing Software is minimized.

Table 1 — Outbound UMPIU generated by UBMTP Engine

UMPIU Type	Relevant UMPIU Fields	UBMTP Engine actions on UMPIU fields
ACKNOWLEDGE COPY	Transaction Type	Set Transfer Type to the corresponding value (see [UFS-UME]).
	Flags	Set error status flags if required (see [UFS-UME]).
	UM ID	Set UM ID to the UM ID value present inside the original UM COPY DATA or COPY DATA request by the UFS Device.
UM DATA OUT	Transaction Type	Set Transfer Type to the corresponding value (see [UFS-UME]).
	Flags	Set error status flags if required (see [UFS-UME]).
	UM ID	Set UM ID to the UM ID value present inside the original ACCESS UM BUFFER read request by the UFS Device.
	Data Segment Length	Set to the same value as present in the “Data Length” field of the corresponding ACCESS UM BUFFER request.
	Data	Fetched from Unified Memory based on the DMA context information supplied in the corresponding ACCESS UM BUFFER request.
ACKNOWLEDGE UM BUFFER	Transaction Type	Set Transfer Type to the corresponding value (see [UFS-UME]).
	Flags	Set error status flags if required (see [UFS-UME]).
	UM ID	Set UM ID to the UM ID value present inside the original ACCESS UM BUFFER or WRITE UM BUFFER request by the UFS Device.

4.5.3 Inbound UMPIUs interpreted by Software

No inbound UMPIUs shall be interpreted by Software since the Unified Memory extension shall be as transparent as possible for the Software and hence it should not add additional load on the system host.

4.5 UMPIU Processing (cont'd)

4.5.4 Inbound UMPIUs interpreted by Host Controller/UBMTP 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 2 — Inbound UMPIUs interpreted by UBMTP Engine

UMPIU Type	Relevant UMPIU Fields	UBMTP Engine actions on UMPIU fields
COPY DATA	Transaction Type	Matched against the corresponding value (see [UFS-UME]).
	Flags	The "R" flag inside this field selects the direction (Source and destination) of the data copy between Unified Memory and System Memory (see [UFS-UME]).
	UM ID	Host Controller shall store UM ID since this value shall be used inside the UM ID field of the corresponding ACKNOWLEDGE COPY response to the Device.
	LUN, Task Tag	Host Controller shall use this LUN / Task Tag combination to identify the original SCSI command, which ultimately triggered the COPY DATA request from the UFS Device. Once the SCSI command has been identified, the Host Controller uses the PRD Table of that particular command to determine the source address of the data buffer inside the system memory. This is one parameter of the DMA context.
	Target UM Area Offset	The Host Controller shall verify that the "Target UM Area Offset" plus the "Data Length" is within the boundaries given by the UMAOMAX register (see 4.3.4). Furthermore the "Target UM Area Offset" plus the UM base address given by the registers UMABA and UMABAU determines the target address of the data copy operation. This is one parameter of the DMA context.
	Data Length	Length of the data to be copied. This is one parameter of the DMA context.
ACCESS UM BUFFER	Transaction Type	Matched against the corresponding value (see [UFS-UME]).
	Flags	This field defines whether the Host Controller shall wait for an incoming UM DATA IN UMPIU or whether the DMA Engine shall fetch data from the Unified Memory. Furthermore the Flags field defines via which Cport and TC that data shall be transmitted.
	UM ID	Host Controller shall store UM ID since this value shall be used inside the UM ID field of the corresponding ACKNOWLEDGE UM BUFFER response to the Device.
	Target UM Area Offset	The Host Controller shall verify that the "Target UM Area Offset" plus the "Data Length" is within the boundaries given by the UMAOMAX register (see 4.3.4). Furthermore the "Target UM Area Offset" plus the UM base address given by the registers UMABA and UMABAU determines the start address of the UM access operation. This is one parameter of the DMA context.
	Data Length	Length of the data to be transmitted. This is one parameter of the DMA context.
UM DATA IN	Transaction Type	Matched against the corresponding value (see [UFS-UME]).
	UM ID	Host Controller shall use this field to match the UM DATA IN UMPIU with the corresponding ACCESS UM BUFFER request.
	Data Segment Length	This field shall contain the same value as it is present inside the "Data Length" field inside the corresponding ACCESS UM BUFFER UMPIU. This value is used by the UMPIU parser inside the Host Controller to determine the length of the UM DATA IN UMPIU.

UMPIU Type	Relevant UMPIU Fields	UBMTP Engine actions on UMPIU fields
	Target UM Area Offset	The Host Controller shall verify that the “Target UM Area Offset” plus the “Data Segment Length” is within the boundaries given by the UMAOMAX register (see 4.3.4). Furthermore the “Target UM Area Offset” plus the UM base address given by the registers UMABA and UMABAU determines the start address of the UM access operation. This is one parameter of the DMA context.
	Data	DMA Engine takes this payload an writes it into the appropriate address inside the Unified Memory.
WRITE UM BUFFER	Transaction Type	Matched against the corresponding value (see [UFS-UME]).
	UM ID	Host Controller shall store UM ID since this value shall be used inside the UM ID field of the corresponding ACKNOWLEDGE UM BUFFER response to the Device.
	Data Segment Length	This value is used by the UMPIU parser inside the Host Controller to determine the length of the WRITE UM BUFFER UMPIU.
	Target UM Area Offset	The Host Controller shall verify that the “Target UM Area Offset” plus the “Data Length” is within the boundaries given by the UMAOMAX register (see 4.3.4). Furthermore the “Target UM Area Offset” plus the UM base address given by the registers UMABA and UMABAU determines the start address of the UM access operation. This is one parameter of the DMA context.
	Data	DMA Engine takes this payload an writes it into the appropriate address inside the Unified Memory.
UM COPY DATA	Transaction Type	Matched against the corresponding value (see [UFS-UME]).
	UM ID	Host Controller shall store UM ID since this value shall be used inside the UM ID field of the corresponding ACKNOWLEDGE COPY response to the Device.
	Source UM Area Offset	The Host Controller shall verify that the “Source UM Area Offset” plus the “Data Length” is within the boundaries given by the UMAOMAX register (see 4.3.4). Furthermore the “Source UM Area Offset” plus the UM base address given by the registers UMABA and UMABAU determines the source address of the data copy operation. This is one parameter of the DMA context.
	Target UM Area Offset	The Host Controller shall verify that the “Target UM Area Offset” plus the “Data Length” is within the boundaries given by the UMAOMAX register (see 4.3.4). Furthermore the “Target UM Area Offset” plus the UM base address given by the registers UMABA and UMABAU determines the target address of the data copy operation. This is one parameter of the DMA context.
	Data Length	Length of the data to be copied. This is one parameter of the DMA context.

4.6 Initialization Sequence for Unified Memory

Figure 3 provides details regarding the initialization sequence in order to configure and enable the Unified Memory functionality. It is closely related to the initialization and boot sequence diagram in JESD223B [UFS] but it also shows the interaction between the System Host and the Host Controller.

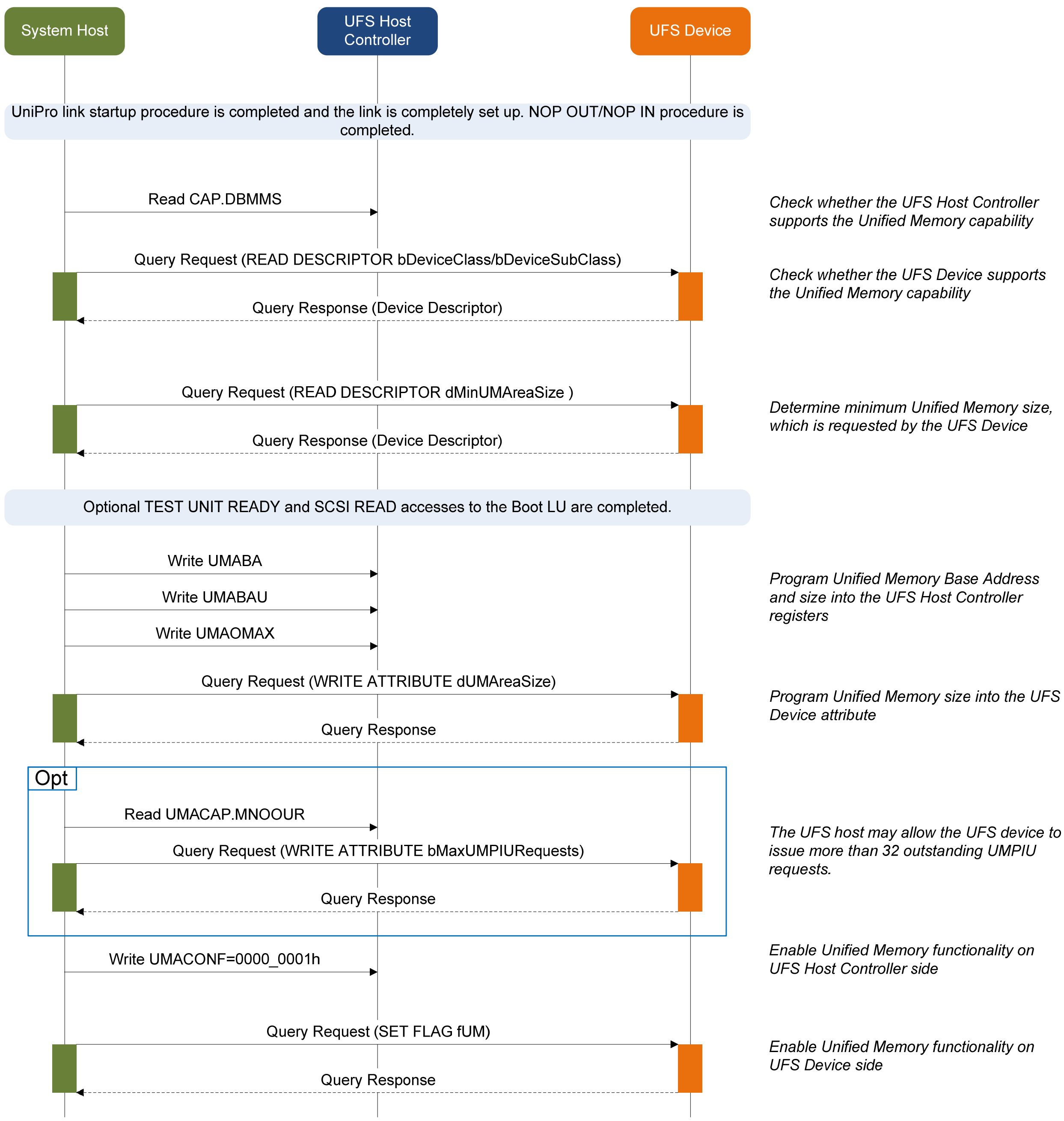

Figure 3 — Initialization Sequence Diagram for Unified Memory usage

5 Power Management

The power management mechanisms exist in the UFS Device Manager layer (UFS layer hereafter) and the UIC layer. In principle, the two layers are independent, but the UFS layer is responsible to the control of the UIC layer. Thus the UFS layer shall take care of the both power management mechanisms.

The UFS layer supports power management states called Sleep Mode and PowerDown Mode as shown in the UFS standard [UFS] to reduce the power consumption of the device

The UIC layer supports a power management state called HIBERNATE to reduce the power consumption of the UniPro link when there is no data to be sent. During HIBERNATE any remaining data inside UniPro buffers may be lost; hence HIBERNATE shall only be entered once it is guaranteed that there is no unsent data left inside the UniPro stack. To guarantee this, a handshaking mechanism between UFS host and UFS device is required. The UniPro standard does not provide a handshaking mechanism since it relies on the application layer (in this case the UFS layer) to take care for that. The “UFS Unified Memory Extension” standard [UFS-UME] defines two flags (fSuspendUM and fUMSuspended) that allow to suspend UM operation in order to make sure that there will not be any unsent UMPIU inside the UniPro stack left before an enter HIBERNATE request is issued.

Table 3 — Possible UM Operations and HIBERNATE in UFS Power Modes

Power Mode	Possible UM Operations	HIBERNATE	Involved Attribute/Flags	Note
Idle	None. Some background operation may start.	Manual/Auto (1)	fSuspendUM fUMSuspended bCurrentPowerMode(2)	May automatically transfer to Active.
Active	All UM operations for normal purposes.	Manual/Auto (1)	fSuspendUM fUMSuspended bCurrentPowerMode(2)	May automatically transfer to Idle.
Pre-Active	UM READ for recovering necessary context.	Not permitted	bCurrentPowerMode(2)	Transition state.
Sleep	None.	Manual	bCurrentPowerMode(2)	Data may reside in UM.
Pre-Sleep	UM WRITE to save necessary context.	Not permitted	bCurrentPowerMode(2)	Transition state.
PowerDown	None.	Manual	bCurrentPowerMode(2)	Data may not reside in UM.
Pre-PowerDown	UM READ to save all dirty data to non-volatile memory.	Not permitted	bCurrentPowerMode(2)	Transition state.

NOTE 1 Auto-HIBERNATE is not permitted if bInitPowerMode is set to 00h

NOTE 2 If IMMED field is cleared, the response to the START STOP UNIT (SSU) command returns after exiting from Pre-Sleep, Pre-PowerDown, or Pre-Active mode, and polling "bCurentPowerMode" can be avoided.

Table 3 shows possible UM operations, available HIBERNATE modes and involved Attribute and Flags.

5.1 Pre-Active

Software shall poll the attribute “bCurrentPowerMode” until Active mode has been reached. Requesting HIBERNATE is not permitted in this mode.

5.2 Pre-Sleep

Software shall poll the attribute “bCurrentPowerMode” until Sleep mode has been reached. Requesting HIBERNATE is not permitted in this mode. Auto-HIBERNATE shall be disabled before entering this mode.

5.3 Pre-PowerDown

Software shall poll the attribute “bCurrentPowerMode” until PowerDown mode has been reached. Requesting HIBERNATE is not permitted in this mode. Auto-HIBERNATE shall be disabled before entering this mode.

5.4 Sleep

Once the UFS device entered Sleep mode, it is assured that there is no outstanding UM operation. Software should request Manual-HIBERNATE to reduce power consumption. Usage of Auto-HIBERNATE is not permitted in this mode.

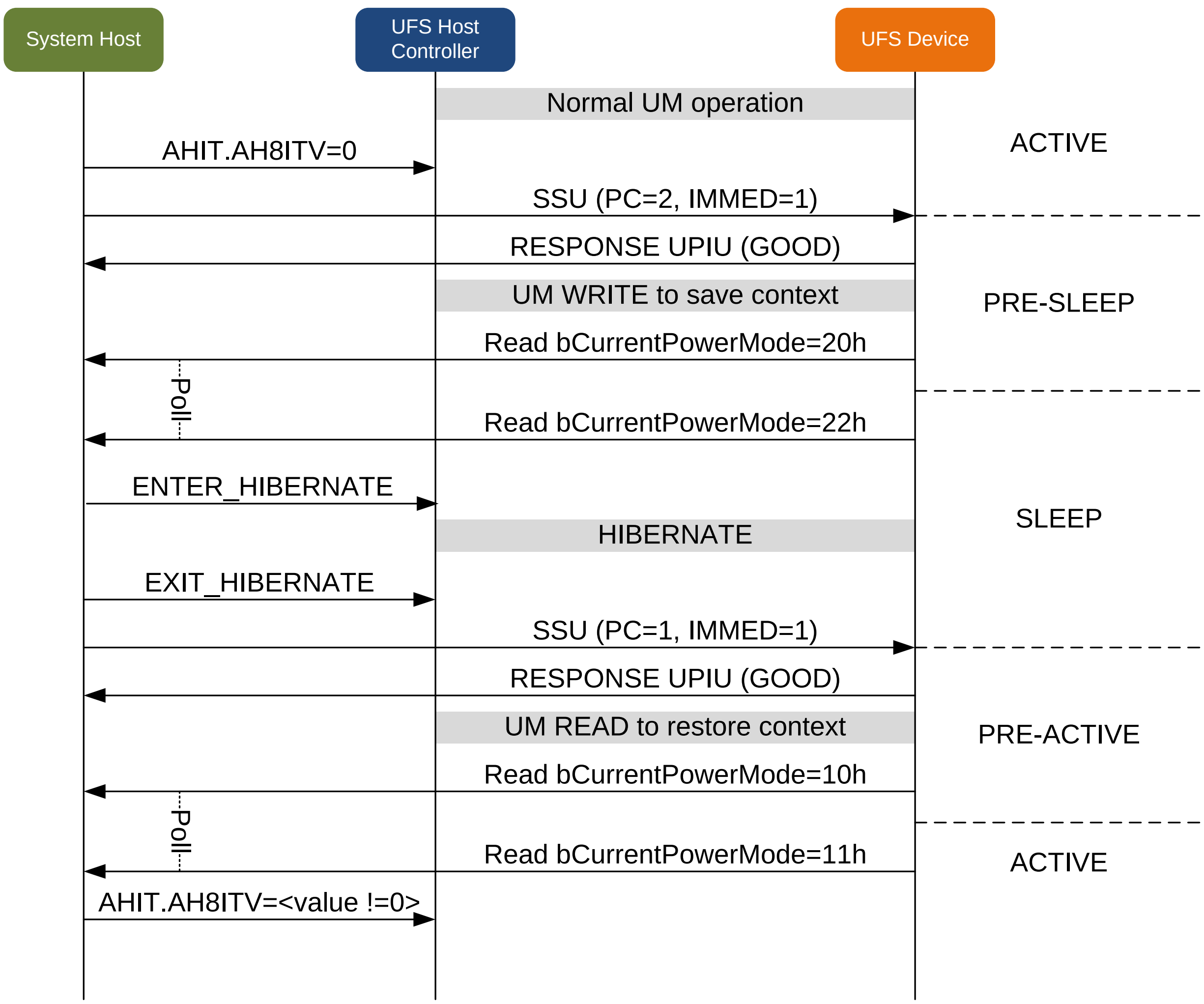

Figure 4 — Example flow from ACTIVE Device Power Mode to SLEEP and back to ACTIVE (IMMED=1)

5.4 Sleep (cont'd)

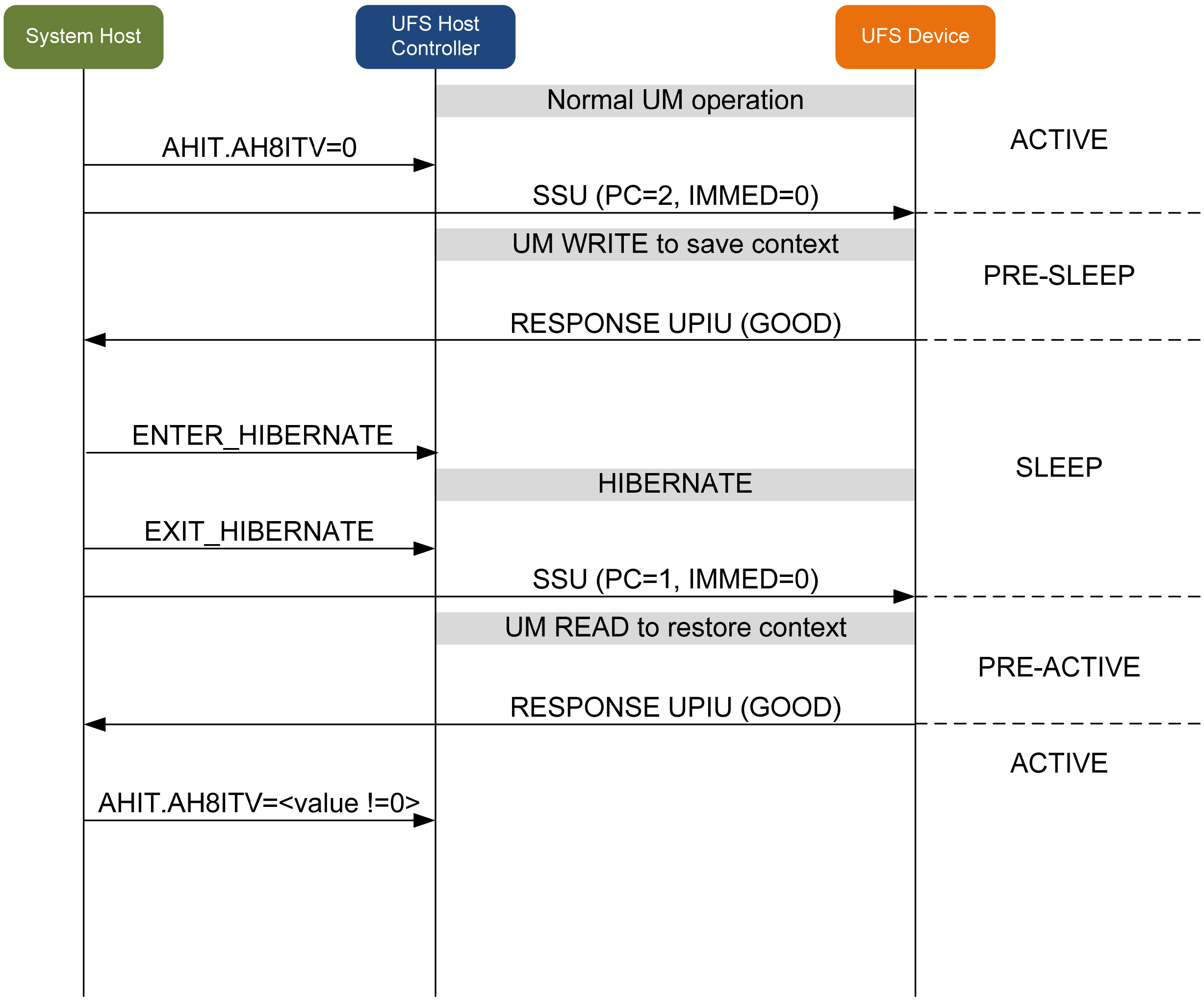

Figure 5 — Example flow from ACTIVE Device Power Mode to SLEEP and back to ACTIVE (IMMED=0)

5.5 PowerDown

Once the UFS device entered PowerDown mode, it is assured that there is no outstanding UM operation. Software may request Manual-HIBERNATE or shut off power supplies to reduce power consumption. Usage of Auto-HIBERNATE is not permitted in this mode.

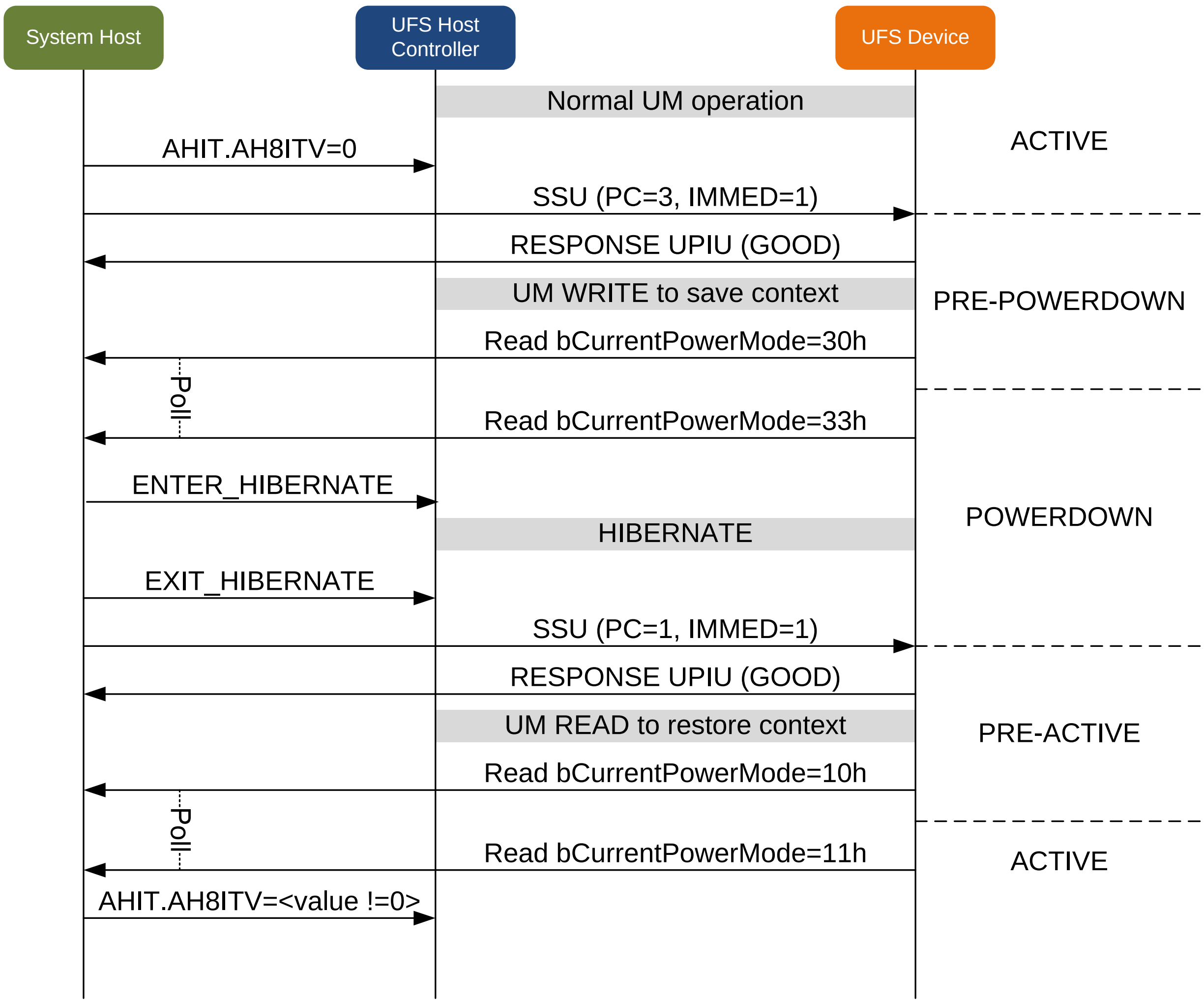

Figure 6 — Example flow from ACTIVE Device Power Mode to POWERDOWN and back to ACTIVE (IMMED=1)

5.5 Power down (cont'd)

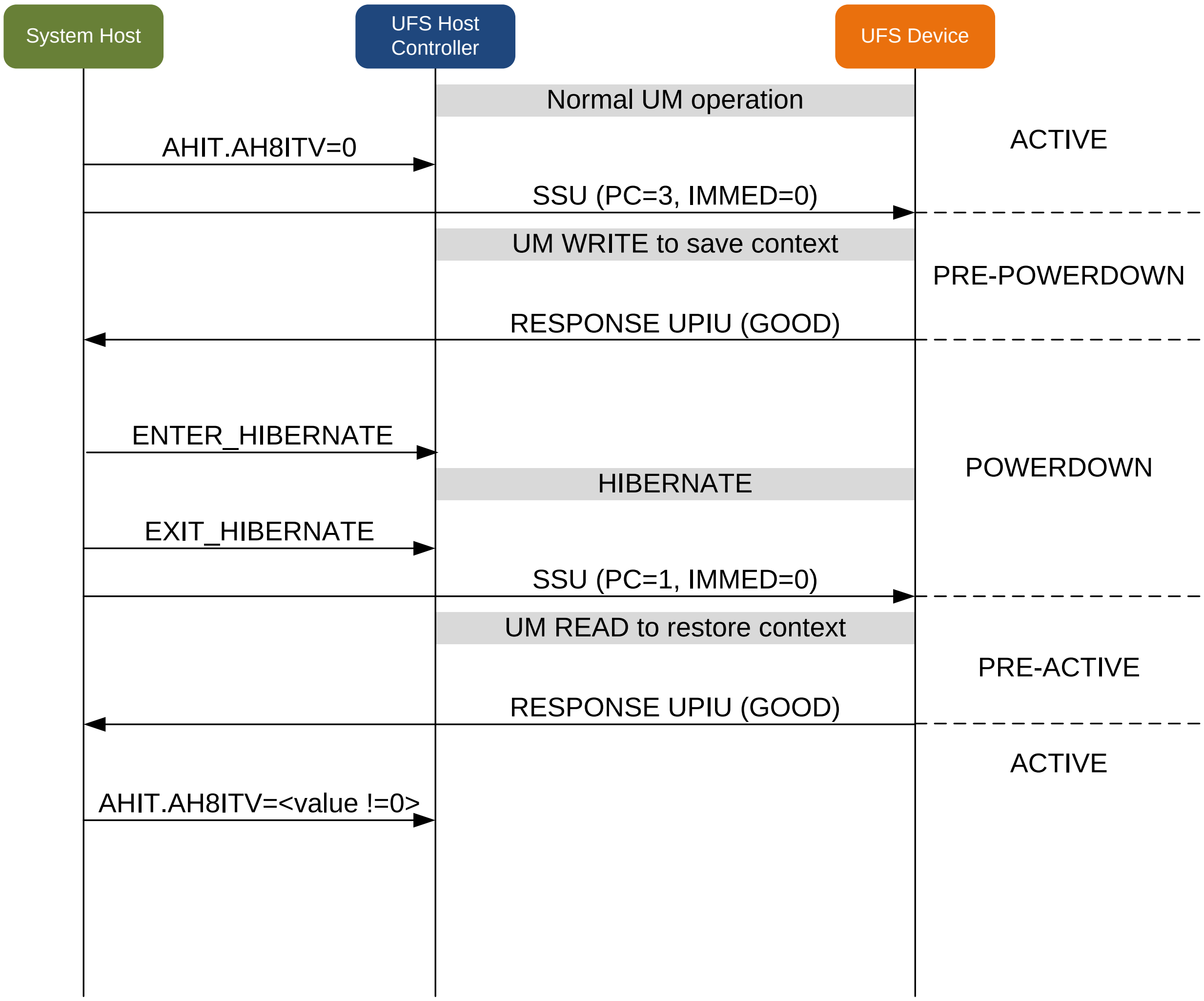

Figure 7 — Example flow from ACTIVE Device Power Mode to POWERDOWN and back to ACTIVE (IMMED=0)

5.6 Idle / Active

If the UFS device is either in Idle or Active mode, either Auto-HIBERNATE or Manual-HIBERNATE may be used with the exception that Auto-HIBERNATE is not permitted if bInitPowerMode is set to 00h.

Figure 8 and Figure 9 depict the UM specific pre-flow, which shall be performed prior to starting the enter HIBERNATE procedure. The waiting time between consecutive polling of the fUMSuspend flag is implementation specific. In case the Auto-HIBERNATE function is supported, the Auto-HIBERNATE Timer (AHIT) may be used for this purpose. If the fUMSuspended flag is still “0”, the AHIT may be reloaded and once expired another polling of the fUMSuspended flag may be triggered.

Failures within the pre-flow and post-flow shall be communicated to the system host via the HCS.UPMCRS field.

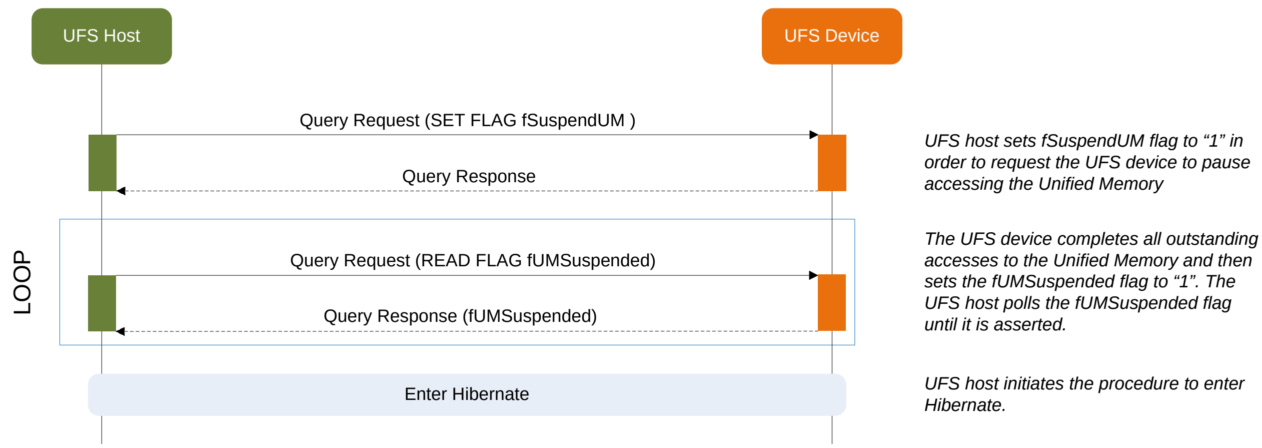

Figure 8 — UM specific pre-flow before entering HIBERNATE (Manual HIBERNATE)

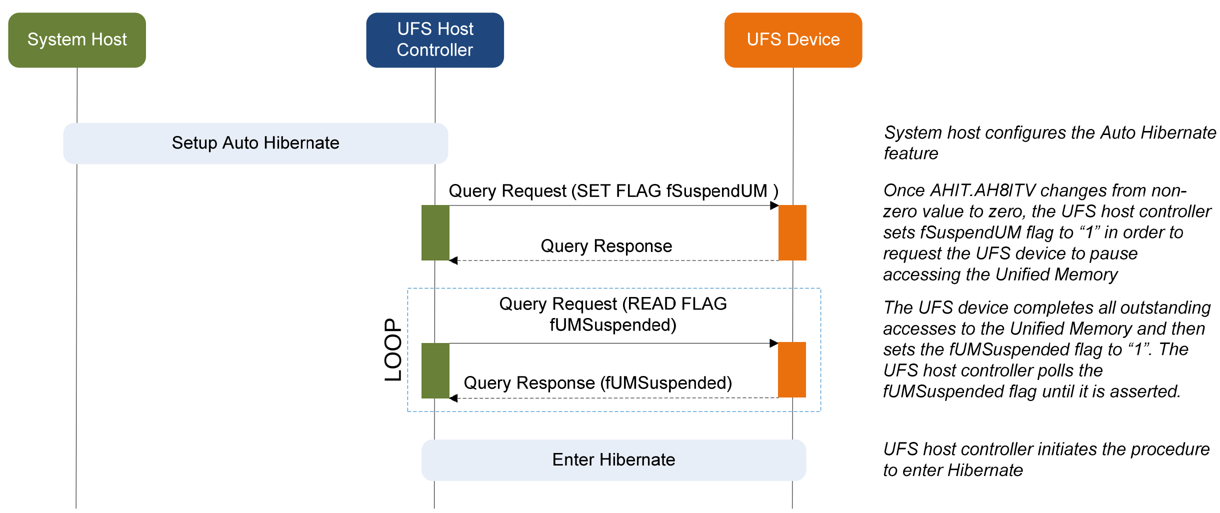

Figure 9 — UM specific pre-flow before entering HIBERNATE (Auto HIBERNATE)

5.6 Idle / Active (cont'd)

If the UniPro link is up and no longer in HIBERNATE, the UFS host should permit the UFS device to resume accessing the Unified Memory as depicted in Figure 10. In case of Auto-HIBERNATE, the UFS host controller handles the post flow autonomously as shown in Figure 11.

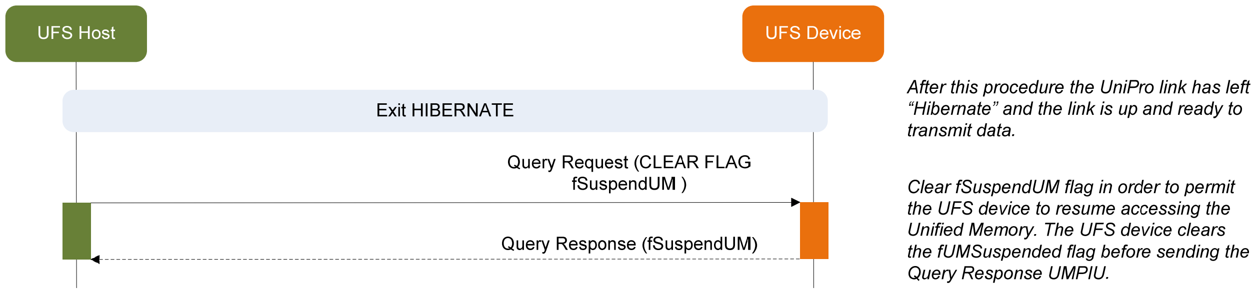

Figure 10 — UM specific post-flow after exiting HIBERNATE (Manual HIBERNATE)

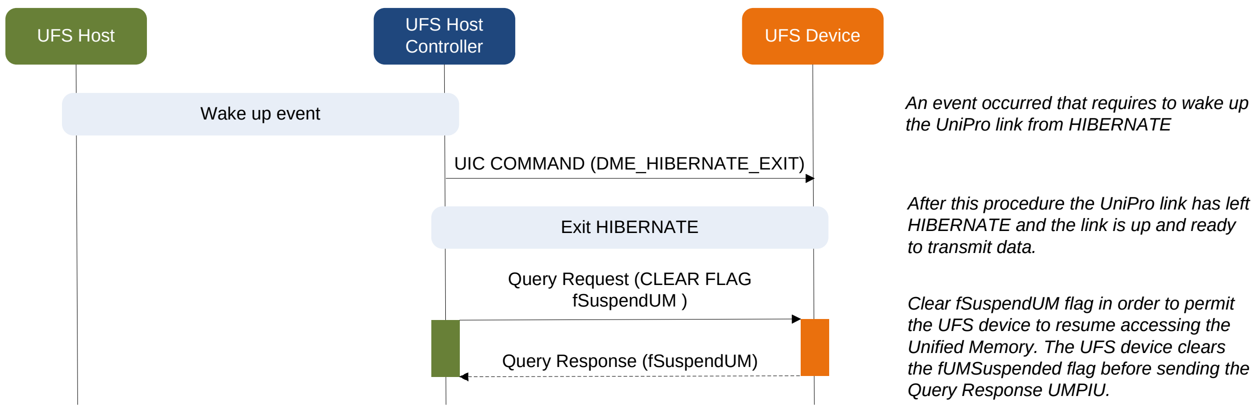

Figure 11 — UM specific post-flow after exiting HIBERNATE (Auto-HIBERNATE)

Annex A (informative) Differences between JESD223-1B and JESD223-1A

This table briefly describes most of the changes made to entries that appear in this standard, JESD223-1B, compared to its predecessor, JESD223-1A (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.

Clause	**Description of change**
4.3	Register map, added 04h under Host capabilities.
4.3	Register map, removed C0h under UMA
4.3.6	Changed title from C0h to 04h

A.1 Differences between JESD223-1A and JESD223-1 (September 2013)

Clause	**Description of change**
2	Updated normative references to reflect current versions.
3.3	Added UME
4.2.2	Revised paragraph 4
4.3	In Table for UMA, End Column corrected editorial error was E3h corrected to B3h
4.3	In Table for UMA, Added details for C0h (UMACAP)
4.3.2	In Table for Bit 31:10 under Description changed 1024 KB to 1 KB
4.3.4	In Table for Bit 31:10, revised description
4.3.4	In Table added details for Bit 9:0
4.3.6	New
4.5.4	Table 2, changed UPIUU to UMPIU, and added Flags
5	All New

Standard Improvement Form JEDEC JESD223-1B

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

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

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

Fax: 703.907.7583

1. I recommend changes to the following:
   - ☐ Requirement, clause number ____________
   - ☐ Test method number ____________ Clause number ____________

   The referenced clause number has proven to be:
   - ☐ Unclear ☐ Too Rigid ☐ In Error
   - ☐ Other ____________

2. Recommendations for correction:

   ____

   ____

   ____

   ____

3. Other suggestions for document improvement:

   ____

   ____

   ____

   ____

Submitted by

Name: ____________ Phone: ____________

Company: ____________ E-mail: ____________

Address: ____________

City/State/Zip: ____________ Date: ____________

JEDEC®