KIOXIA

Technical Brief

Tech Brief Series: How to Validate a Managed Flash Device[1] Design

Part 3: Conducting a Protocol Analysis

One of the key steps for managed flash device developers when validating their UFS[2] or e-MMC[3] designs is to conduct a protocol analysis. A protocol analysis includes capturing, decoding and reviewing the command and data transactions that occur between the host controller and the managed flash device. Conducting this analysis ensures that the managed flash device initializes properly and behaves correctly under normal operating conditions. KIOXIA UFS and e-MMC devices follow JEDEC® standards (Table 1) and many of the sequences and protocols featured in this technical brief are explained in these standards.

Managed Flash Device	*Associated JEDEC Standard*
UFS 3.1	*JESD220E*
UFS 4.0	*JESD220F*
UFS 4.1	*JESD220G*
e-MMC 5.1	*JESD84-B51*

Table 1. JEDEC Standards

This technical brief is the third and final installment in the series, "How to Validate a Managed Flash Device Design," and presents the preparation and procedures for properly conducting a protocol analysis for managed flash devices. It follows the signal integrity analysis and power integrity analysis previously presented.

Introduction

A protocol analysis essentially checks the protocol exchanges that occur between the host controller and the managed flash device to ensure that commands, sequences and settings are within JEDEC specifications. Even though an operation or series of commands finishes successfully, it does not automatically mean that no protocol violations occurred. In some cases, these violations could produce failure symptoms that become evident later while addressing other commands or operations. Therefore, evaluating protocol exchanges for managed flash device sequences from initialization to normal operations is recommended. Section 1 covers UFS devices while Section 2 covers e-MMC devices.

Section 1: UFS Devices

There are six UFS device sequences recommended for protocol analysis. To analyze these six sequences, a protocol analyzer supporting the appropriate UFS device version is recommended.

#1 - Link Startup Sequence

In the flowchart depicting the link startup sequence (Figure 1), the protocol exchanges from the UFS device powers on until the fDeviceInit flag is reset. Once powered, the host controller can receive query responses that need to be analyzed to ensure that the UFS device initialized properly and behaved correctly under normal operating conditions.

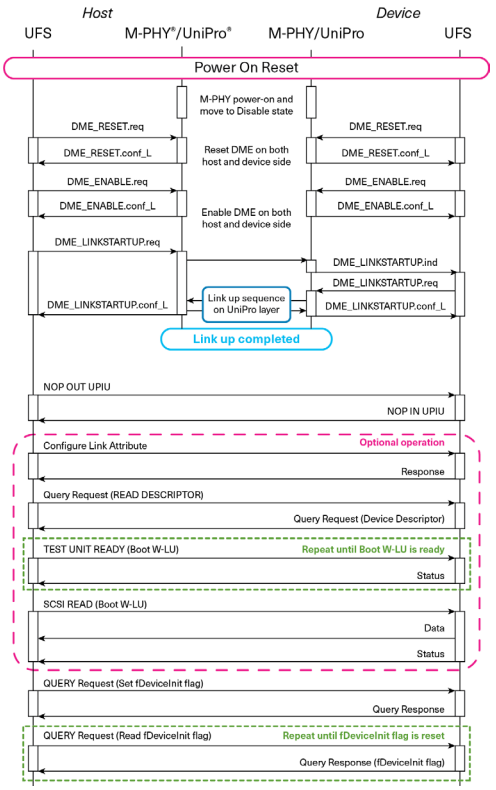

Figure 1: UFS link startup sequence flowchart (Source: Kioxia Corporation[4])

It is important to note that the host controller and UFS device also perform capability exchanges during the linkup stage of the startup sequence. It is during this sequence that the UFS device shares its capabilities locally with the host controller's peers. The values exchanged need to be checked to verify that both the host controller and UFS device are within specification. Refer to the applicable UFS version of the JEDEC® standard (Table 1) as well as related technical data sheets for more details covering the link startup sequence.

#2 - Boot Sequence

Boot sequences depend on the UFS device platform implementation so it's important to understand the intended sequence of the platform developer in advance. The objective for checking this sequence is to confirm alignment between the actual boot sequence and the associated protocols that were exchanged during the boot up process.

#3 - Power Mode Change Sequence

The UFS specifications referenced in Table 1 support both LS-Mode (low-speed gear) and HS-Mode (high-speed gear), to scale on performance as needed. Both modes offer further scaling by supporting several gear levels. For example, the UFS 4.1 specification supports PWM[5]-Gear 1 to 7 and HS-Gear 1 to 5. In these instances, the protocol exchanges for power mode change combinations used during mass production need to be checked. Once again, refer to the applicable UFS version of the JEDEC standard (Table 1) for more details covering the sequence when executing a power mode change.

#4 - Power Cycle Sequence

Before power is removed from the UFS device, it is important that all data in the volatile cache of the UFS controller is flashed into NAND memory, or else the data will be lost. Therefore, it is crucial to carry out the power cycling sequence properly.

The UFS power mode state machine (Figure 2) is an illustration that shows an overview of the various states that can lead to powering down a UFS device before completely removing its power. Once again, refer to the applicable UFS version of the JEDEC standard (Table 1) for more details covering the power down sequence.

It is also highly recommended to consult with the UFS device vendor and/or refer to technical data sheets and application notes on the recommended or prescribed sequences when carrying out a power cycle.

#5 - Normal Operation

Exchanged protocol between the host controller and UFS device during normal operation needs to be analyzed to ensure proper device initialization and correct behavior under normal operating conditions. The analysis is performed in different access patterns to confirm that the commands sent by the host controller are executed and the UFS device provides timely responses. Figure 3 is an example of protocol analysis packet data taken during read and write operations.

#6 - Specialized Features

Whether a JEDEC standardized feature or vendor specific feature, if the feature is to be used for a product that will go to mass production, it is highly recommended to do a protocol analysis to confirm that the driver commands and sequences are aligned with the JEDEC specification or the intended vendor implementation. Some examples of specialized features include device provisioning, RPMB access and Vendor Specific Functions (VSF).

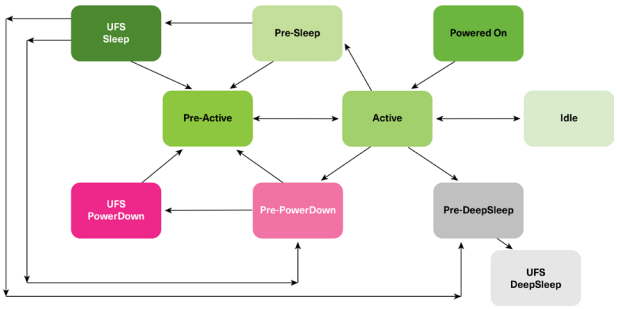

Figure 2: UFS power mode state machine

Time (s)	Index	Host	Device	Gear	TaskTag	LUN
1.894821312	10860	READ_10		HS_G4B	0x04	0x04
1.894835128	10861	READ_10		HS_G4B	0x05	0x04
1.894843072	10862	READ_10		HS_G4B	0x06	0x04
1.8948476	10863	READ_10		HS_G4B	0x07	0x04
1.894900792	10864		DATA_IN	HS_G4B	0x04	0x04
1.894902616	10865		RESPONSE	HS_G4B	0x04	0x04
1.894907448	10866		DATA_IN	HS_G4B	0x05	0x04
1.894909264	10867		DATA_IN	HS_G4B	0x05	0x04
1.894911088	10868		DATA_IN	HS_G4B	0x05	0x04
1.894912912	10869		DATA_IN	HS_G4B	0x05	0x04
1.894914736	10870		DATA_IN	HS_G4B	0x05	0x04
1.89491656	10871		DATA_IN	HS_G4B	0x05	0x04
1.894916616	10872	READ_10		HS_G4B	0x00	0x04
1.894920432	10873		DATA_IN	HS_G4B	0x05	0x04
⋮						
1.896105712	10918	WRITE_10		HS_G4B	0x02	0x04
1.896109728	10919		READY_TO_TRANSFER	HS_G4B	0x02	0x04
1.896110768	10920	DATA_OUT		HS_G4B	0x02	0x04
1.896248848	10921		RESPOSE	HS_G4B	0x02	0x04
1.896309184	10922	READ_10		HS_G4B	0x03	0x04
1.896364784	10923		DATA_IN	HS_G4B	0x03	0x04
1.896366608	10924		RESPONSE	HS_G4B	0x03	0x04
1.917077776	10925	READ_10		HS_G4B	0x04	0x04
1.917134744	10926		DATA_IN	HS_G4B	0x04	0x04
1.917141496	10927		DATA_IN	HS_G4B	0x04	0x04
1.917149272	10928		DATA_IN	HS_G4B	0x04	0x04
1.917155984	10929		DATA_IN	HS_G4B	0x04	0x04
1.917162728	10930		DATA_IN	HS_G4B	0x04	0x04
1.917169472	10931		DATA_IN	HS_G4B	0x04	0x04
1.917171296	10932		DATA_IN	HS_G4B	0x04	0x04
1.917173112	10933		DATA_IN	HS_G4B	0x04	0x04
1.917174936	10934		DATA_IN	HS_G4B	0x04	0x04

Figure 3: Read/write protocol analysis packet data

Section 2: e-MMC Devices

Like UFS devices, e-MMC devices also require analysis and validations to ensure that each device operates in accordance with device specifications and the system design, to pinpoint issues in hardware or software and to mitigate potential field failures. As such, there are two important command sequences to verify - the first requires analysis and validation of the Power On to Boot sequence during a device boot up, while the second requires analysis and validation of the Power On after Boot sequence once the device initializes.

Standard e-MMC power on sequences consists of a sequential series of commands which are issued by the host controller, typically starting with CMD0, followed by CMD1, CMD2, and so forth (Figure 4).

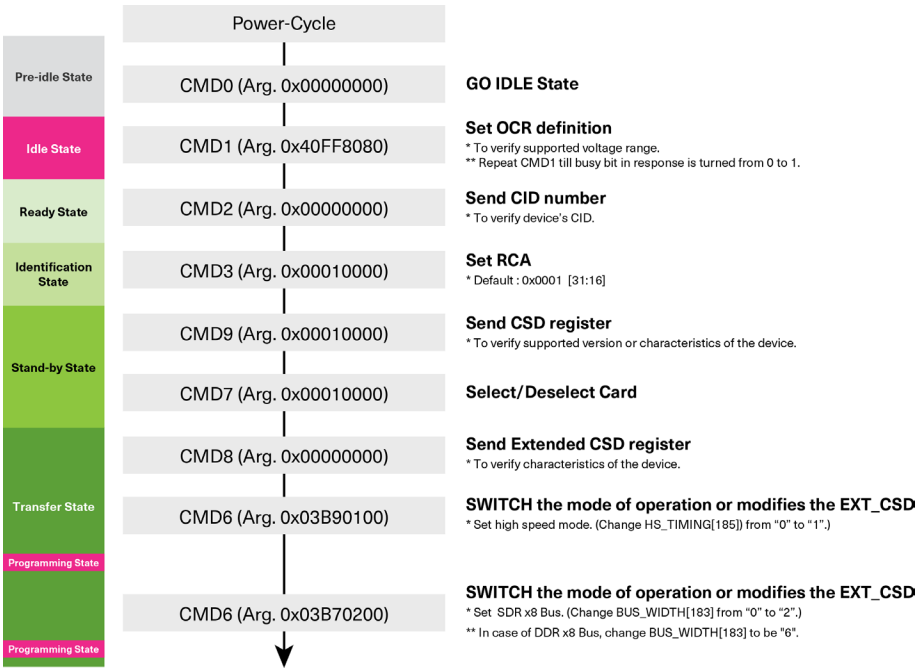

Figure 4: Typical e-MMC device initialization sequence (Source: Kioxia Corporation[6])

Each command has a specific purpose for exchanging information between the host controller and e-MMC device. For example, CMD0 resets the device to idle state, CMD1 asks the device to send its operating conditions register (OCR) contents, CMD2 asks the device to send all card identification device (CID) register numbers, CMD3 asks the device to send its relative card address (RCA) register and CMD8 and CMD9 asks the device to send its extended card specific data (CSD) and CSD register data, respectively.

To ensure that all commands are issued and in sequence, using an e-MMC protocol analyzer is recommended. In some cases, the host controller may send some commands in a different order than the process illustrated in Figure 4, which is acceptable behavior. If the e-MMC device reports an initialization error or timeout during a boot up, verifying the Power On to Boot sequence is an important step to debug the issue.

Once device initialization is verified, the next step is to verify the commands sent to the device after initialization. Typically, during this step is when standard e-MMC device operations occur such as configuring the device or reading data from the device after the operating system boots up. Additionally, the host controller may send several different commands that can vary. One example is a CMD6 switch command to configure the e-MMC device, such as setting the bus width to 8-bit mode or setting the bus speed to HS400. As a result of this activity, it is important to verify that the commands being sent by the host controller are expected (Figure 5).

```
[    2.452314] mmc0: new HS200 MMC card at address 0001
[    2.453102] mmc0: CMD6: switch EXT_CSD[183] to 0x2 (Bus Width: 8-bit)
[    2.454871] mmc0: CMD6: switch EXT_CSD[185] to 0x1 (HS_TIMING: HS200)
[    2.456214] mmc0: switching to 8-bit bus width
[    2.457092] mmc0: selected voltage range 2.7-3.6V
[    2.458314] mmc0: running at 200 MHz, HS200 SDR
```

Figure 5: Example of e-MMC switch commands in Linux®

Missing or unexpected commands may result in the e-MMC device not behaving as expected, or in some cases, there may be errors reported by the Linux® kernel. For example, when an e-MMC device operates in HS200/HS400 mode, a CMD21 tuning command is necessary to ensure the optimal data sampling point is used by the device. If the tuning command is missing, there may be intermittent timing issues or errors reported in the Linux kernel (see Figure 6). For guidance or any assistance with reviewing the e-MMC device command logs during validation, contact the sales department at KIOXIA America, Inc.

```
[   15.324812] mmc0: Switching to high speed HS200 mode
[   15.324850] mmc0: new HS200 MMC card at address 0001
[   15.325102] mmcblk0: mmc0:0001 SEM32G 29.1 GiB
[   15.325114] mmcblk0boot0: mmc0:0001 SEM32G partition 1 4.00 MiB
[   15.325123] mmcblk0boot1: mmc0:0001 SEM32G partition 2 4.00 MiB
[   15.328451] mmc0: error -84 whilst initialising MMC card
[   15.329102] mmc0: Tuning failed, falling back to default timing
[   15.329451] mmc0: unexpected CRC error sending read/write command
[   15.330142] blk_update_request: I/O error, dev mmcblk0, sector 0
[   15.330671] Buffer I/O error on dev mmcblk0, logical block 0, async page read
```

Figure 6: Example of missing tuning command and Linux kernel error log

Summary

Conducting a protocol analysis is a key step for UFS and e-MMC device developers when validating their designs to ensure proper device initialization and correct behavior under normal operating conditions. This analysis typically entails checking the protocol exchanges between the host controller and the managed flash device to ensure that commands, sequences and settings are within JEDEC® specifications. As it relates to UFS devices, there are six sequences that need to be analyzed: Link Startup Sequence; Boot Sequence; Power Mode Change Sequence; Power Cycle Sequence; Normal Operation; and Specialized Features. For e-MMC devices, the two key command sequences to verify include the Power On to Boot sequence during a device boot up and the Power On after Boot sequence once the device initializes.

This tech brief presents preparation and procedures to properly conduct a protocol analysis for UFS and e-MMC managed flash devices and concludes the tech brief series, “How to Validate a Managed Flash Device Design.” The first two installments featured conducting a signal integrity analysis and power integrity analysis, respectively. For further assistance in these analyses and validations, contact the sales department at KIOXIA America, Inc.

General information about KIOXIA memory products is available here.

FOOTNOTES:

[1] A managed flash device combines raw NAND flash memory and an intelligent controller in one integrated package, enabling internal memory management.

[2] Universal Flash Storage (UFS) devices are based on the UFS specification, of which, the v4.1 specification is the current release issued by JEDEC® and announced in January 2025.

[3] Embedded MultiMediaCard (e-MMC) is a specification developed by JEDEC for mobile applications. The current release is v5.1, published in February 2015.

[4] Used with permission from Kioxia Corporation, Application Note, UFS Memory Device JEDEC UFS Ver. 4.0, Revision 1.0, June 2022, Figure 5.

[5] PWM = Pulse Width Modulation.

[6] Used with permission from Kioxia Corporation, e-MMC Training presentation, April 2018, Slide 31.