Introduction to UFS Subsystem

Mohammad Faiz Abbas Rizvi

ELC 2019

faiz_abbas@ti.com

About Me

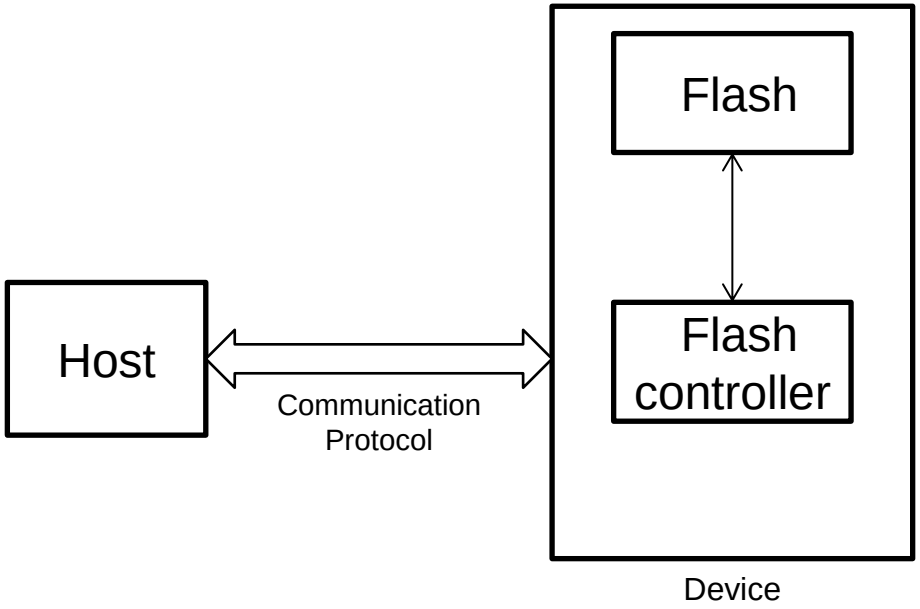
- Software Engineer at Texas Instruments India since 2017.
- Part of the Linux Team that works on supporting various TI SoCs in mainline kernel and u-boot.
- I work on supporting peripheral drivers on TI SoCs, mainly MMC, CAN and now UFS.
- This presentation is a result of my experience adding the UFS subsystem to U-boot.

What is Universal Flash Storage?

It is primarily a Managed Flash.

Managed Flash

- A non-volatile flash array combined with a memory controller
- The controller manages the memory – bad block management, ECC, wear leveling etc.
- The controller communicates with the outside world with a defined communication protocol like MMC, Sata, SPI, Hyperflash etc.



Generic Managed Flash

UFS Features

- It’s a high performance serial interface designed for low power devices.
- Marketed as a replacement for SD card and eMMC for smartphones, ultra portable PCs and other embedded devices.
- Theoretical bidirectional full duplex transfer speeds upto 1.45 GBps
- Can be a removable card or embedded on the board

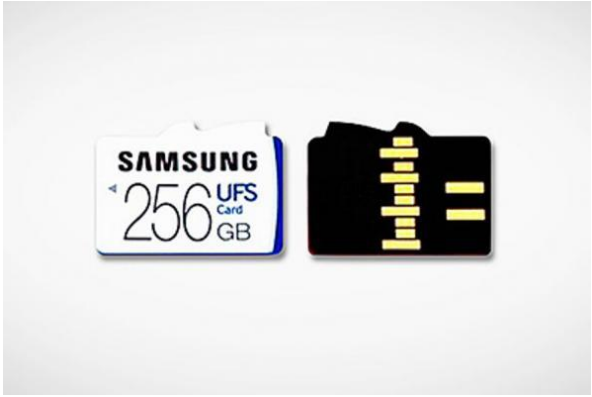


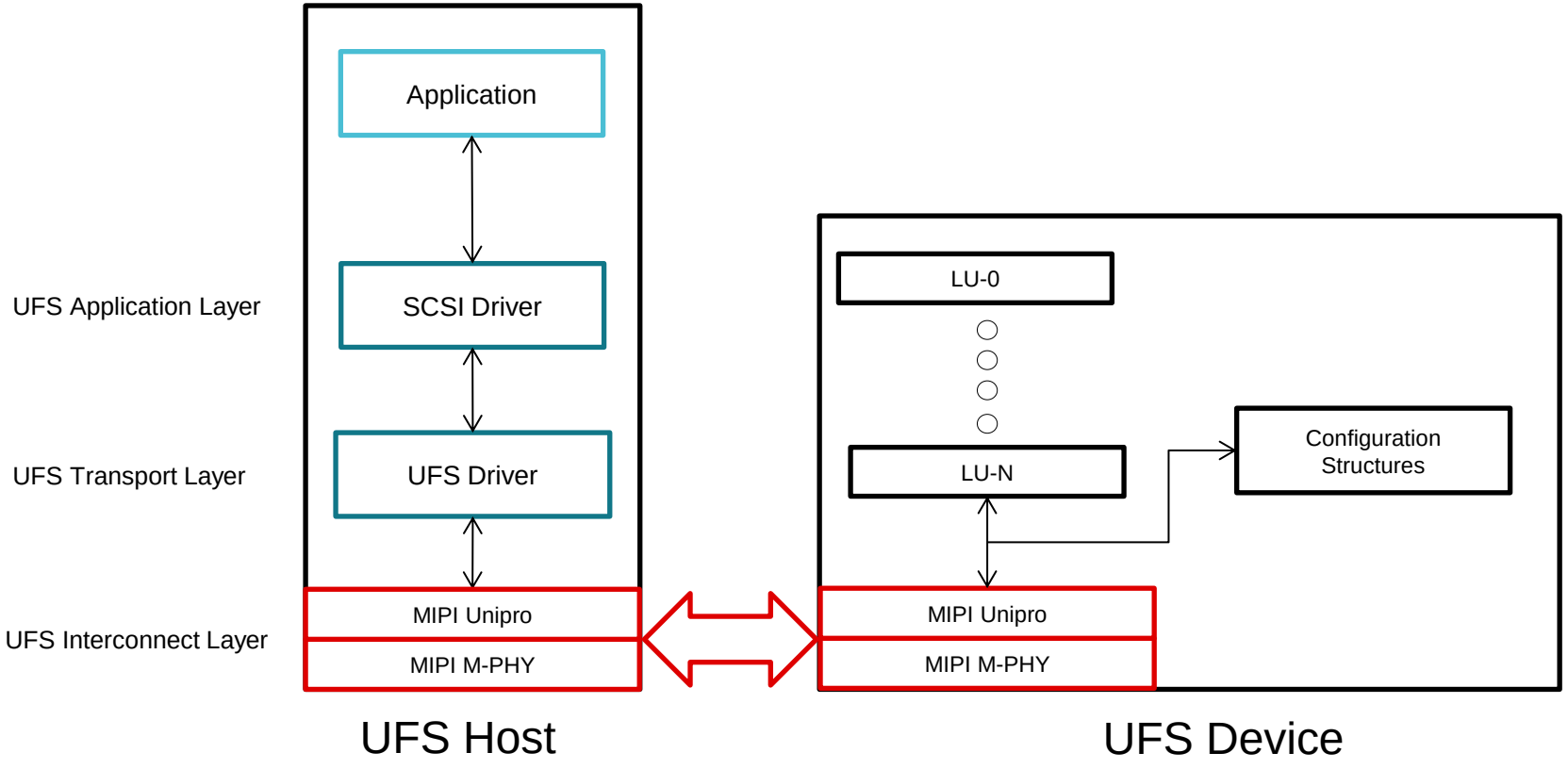
Image courtesy: dpreview.com

How is UFS better?

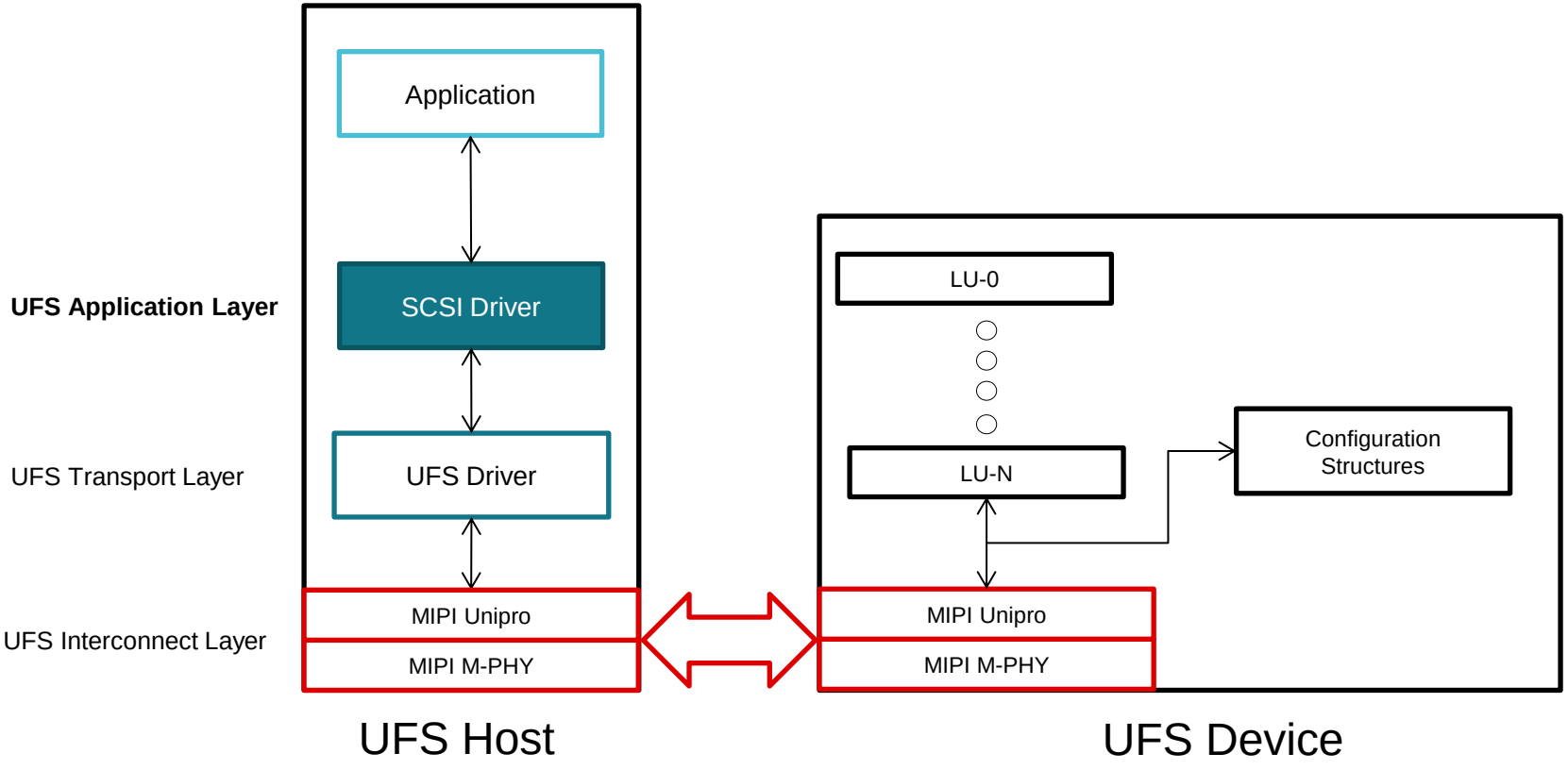
Property	eMMC 5.1	UFS
I/O speed	Upto 400 MBps	Upto 1.45 GBps
I/O Voltage	1.2V-3.3V	0.2V-0.4V differential
Random Read	13000 IOPS[1]	68000 IOPS[1]
Protocol	Half duplex	Full duplex
Theoretical Device Capacity	128 GB	16 TB

[1] androidcentral.com

UFS System Overview



UFS System Overview



UFS Application Layer

- Consists of the UFS SCSI command set based on SCSI Architecture Model (SAM)
- Transactions take place with fixed length Command Descriptor Blocks (CBD)
- Each Transaction follows the I_T_L_Q nexus meaning the CDB needs to identify an **I**nitiator communicating with a specific **L**UN in a **T**arget with a specific **Q**uery.

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation code = 03h							
1	LUN			Reserved				
2	Reserved							
3	Reserved							
4	Allocation length							
5	Control							

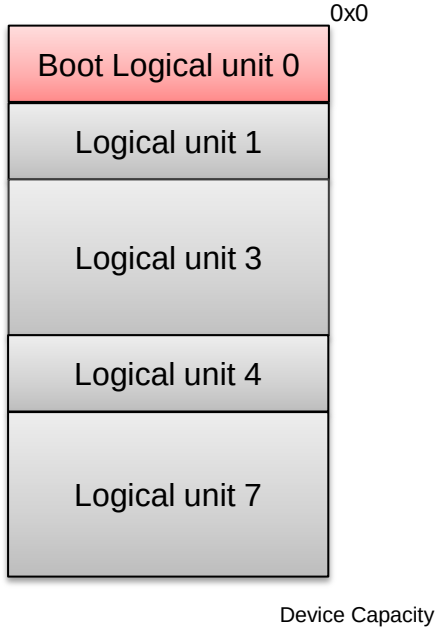
Example 6 byte CDB format

UFS SCSI Commands

- A command descriptor block can be of 6, 10 or 16 bytes of length
- Some important commands :
  - READ
  - WRITE
  - READ CAPACITY - get the size of a logical unit
  - REPORT LUNS - get list of all logical units
  - TEST UNIT READY - check if a logical unit is ready for accepting requests
  - START STOP UNIT – switch Power mode of the device
  - INQUIRY – More information about a logical unit

What is a Logical unit?

- Externally addressable
- Storage entity
- Internal Task Queue
- LUs inside a UFS device can be configured in variety of ways:
  - Amount of physical memory allocated to each LU
  - Write Protection
  - Boot
  - Memory Type (default, system code, non-persistent, enhanced)
  - Priority access
  - RPMB



Logical unit (continued)

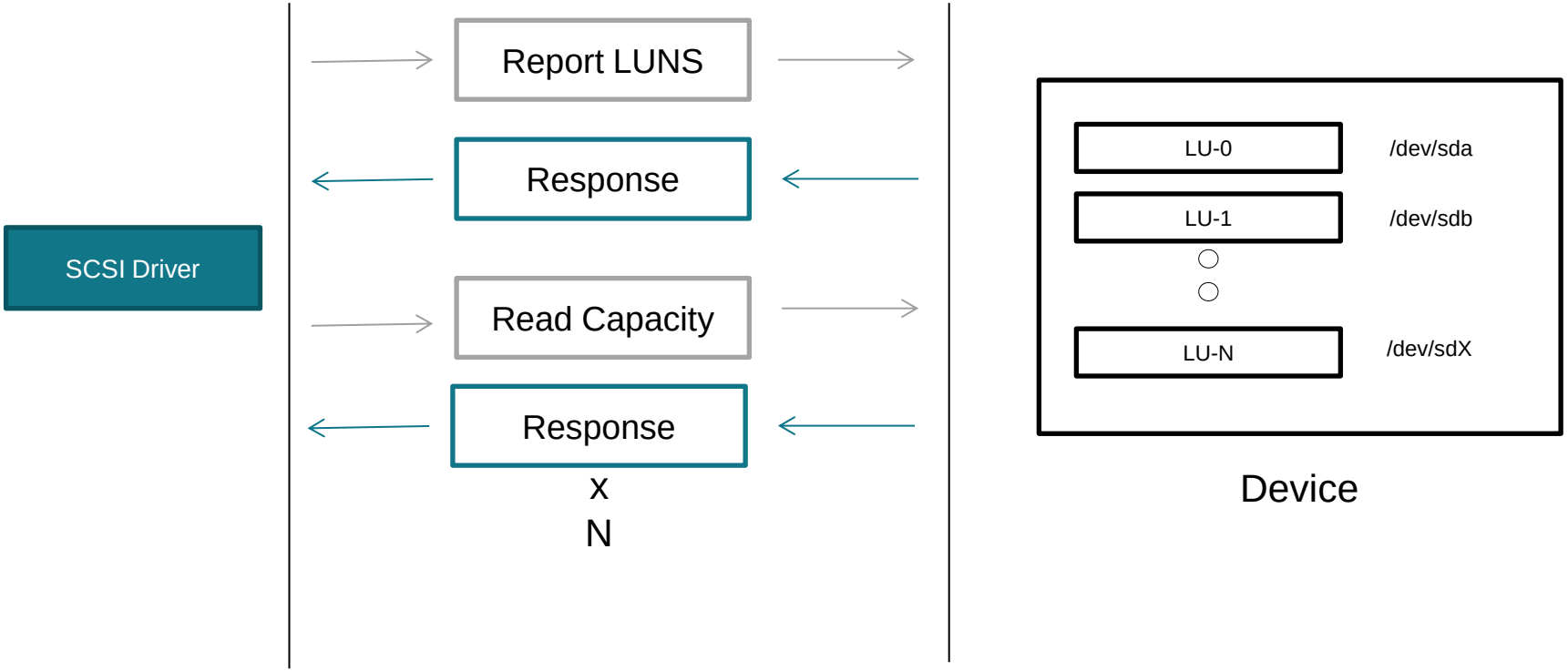
- A UFS device can have a maximum of 32 logical units in it.
- In addition, there might be 4 Well Known Logical Units (W-LUNS).

- REPORT LUNS
  - Target for REPORT LUNS SCSI Request
  - Returns number and configuration of all the LUNs on the device
- UFS Device
  - Target for INQUIRY SCSI Request
  - Configuration, Power Control, Formatting, flags, attributes
- Boot
  - Used for boot operation
  - Can be 1 or 2 boot partitions but only one active one
- RPMB –
  - Replay Protected memory block
  - Need security protocol to access data

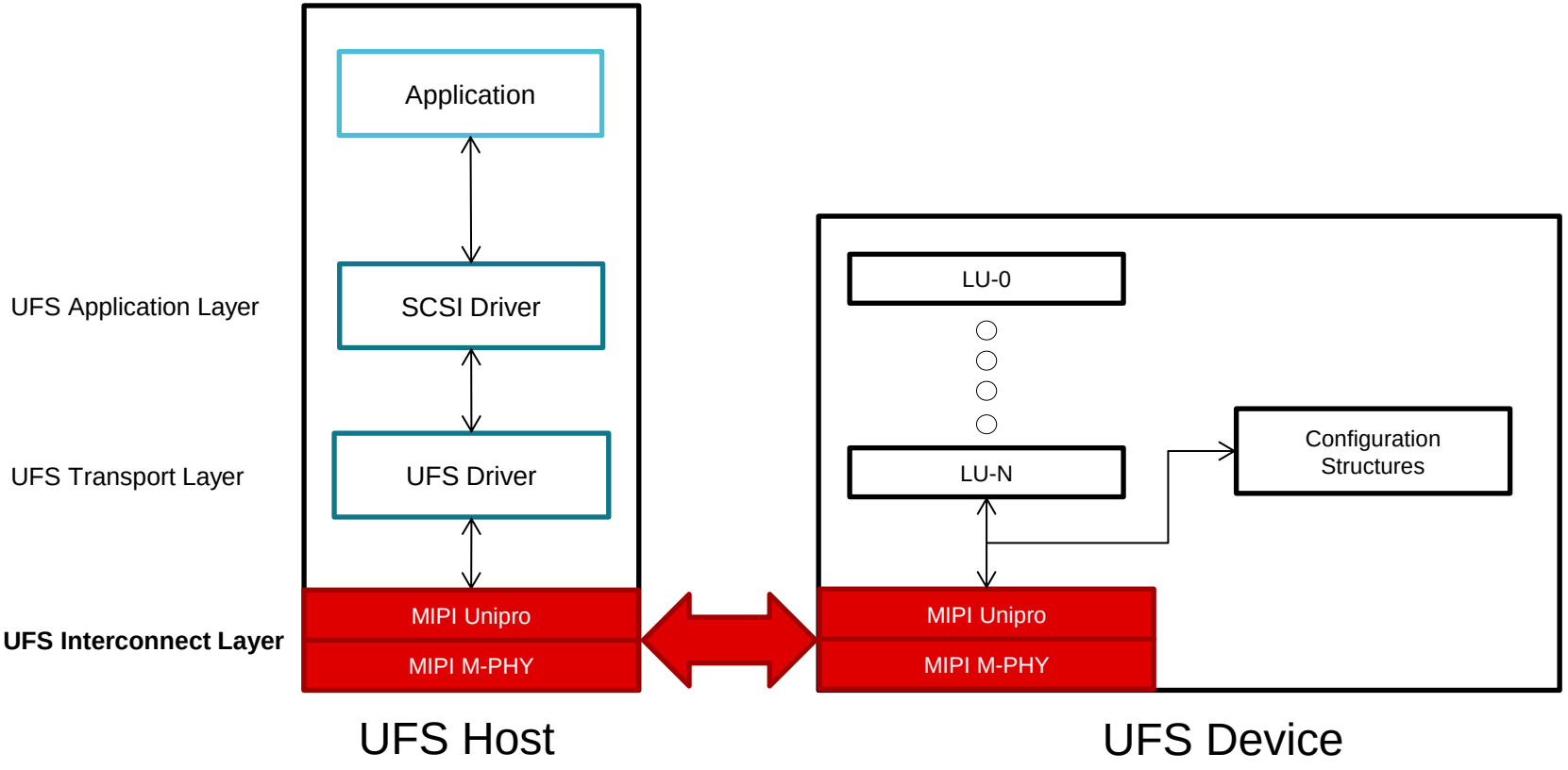
Well known logical unit	**LUN**
REPORT LUNS	0x81
UFS Device	D0h
Boot	B0h
RPMB	C4h

Well-known LUNS

Application Layer Transaction View



UFS System Overview

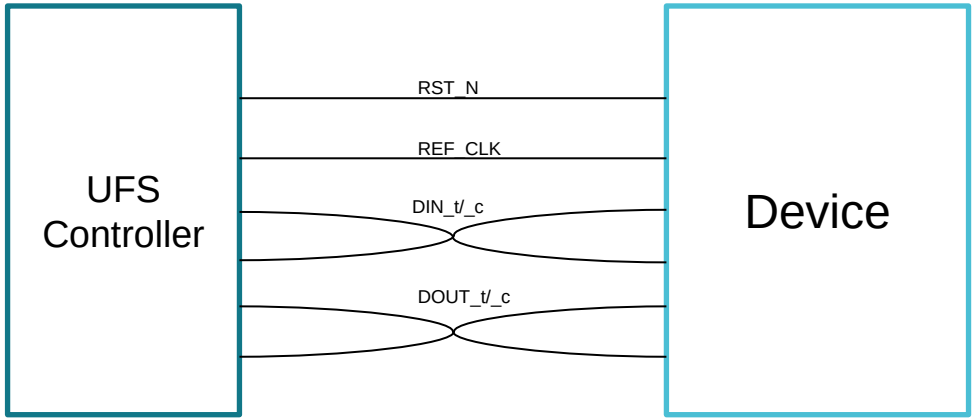


UFS Interconnect Layer

- MIPI-PHY standard defines the physical layer implementation.
- MIPI-Unipro standard defines the data link layer implementation.

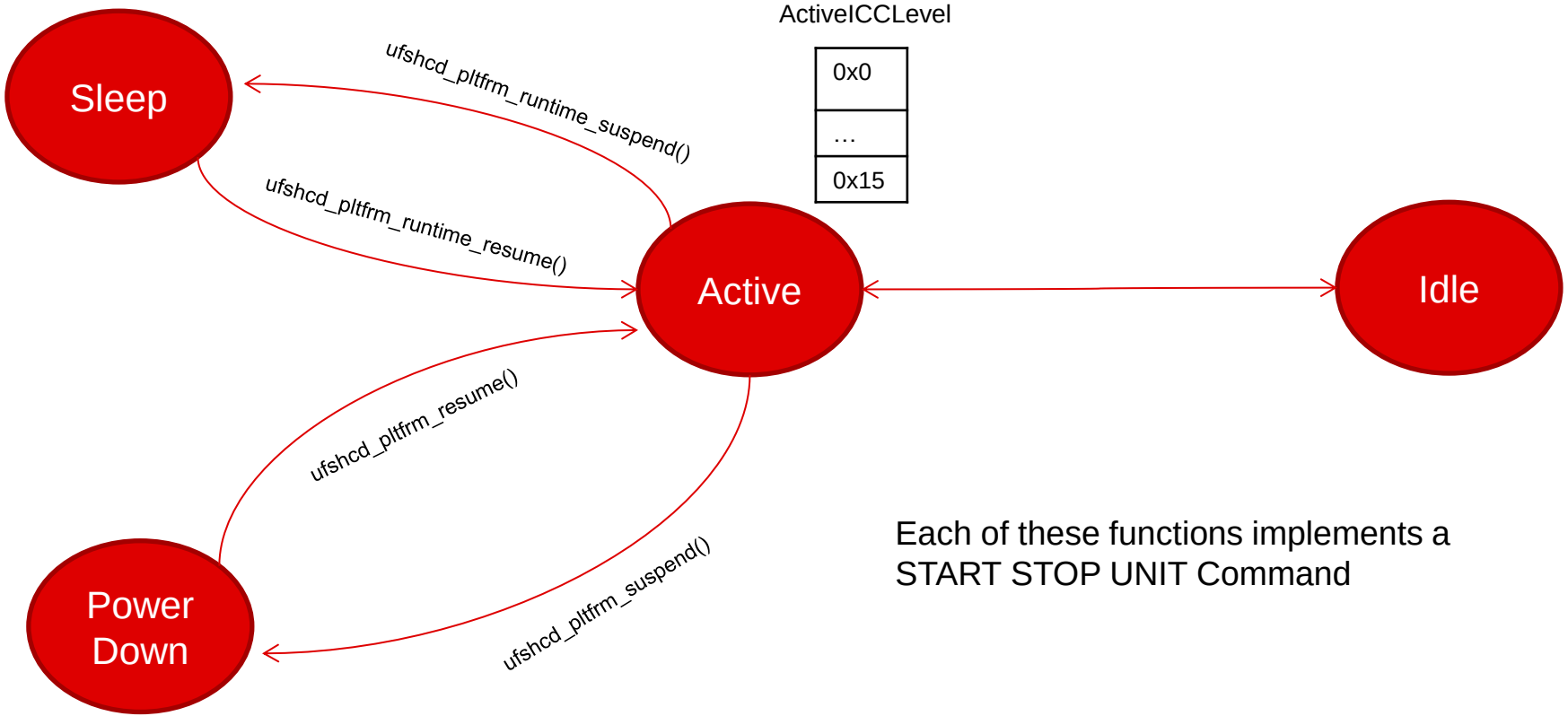
HS-GEAR	Maximum Datarate
HS-GEAR1	182 MBps
HS-GEAR2	364 MBps
HS-GEAR3	728 MBps
HS-GEAR4	1457 MBps

UFS data rates



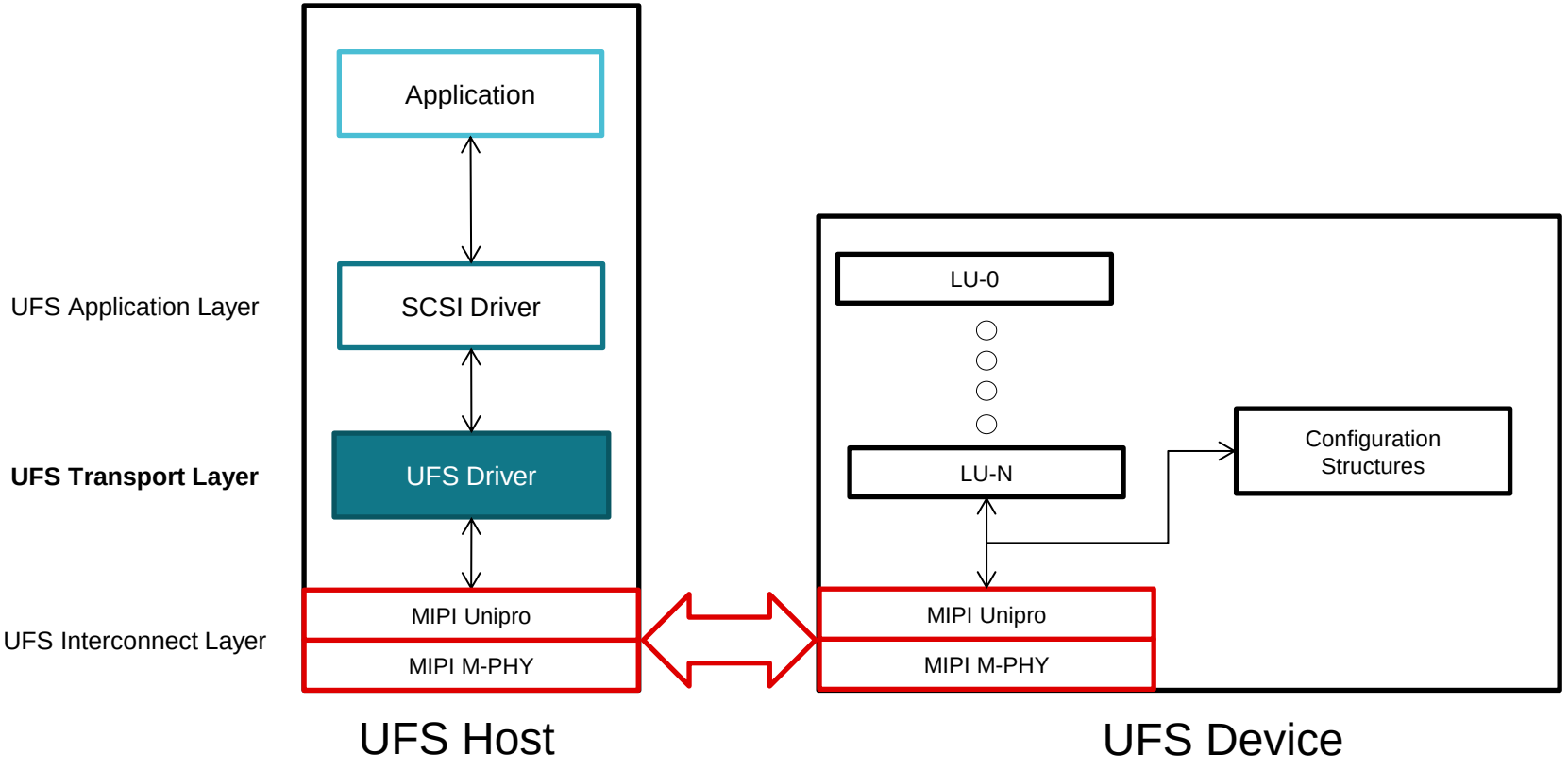
UFS signals

Simplified UFS Power Modes



Each of these functions implements a START STOP UNIT Command

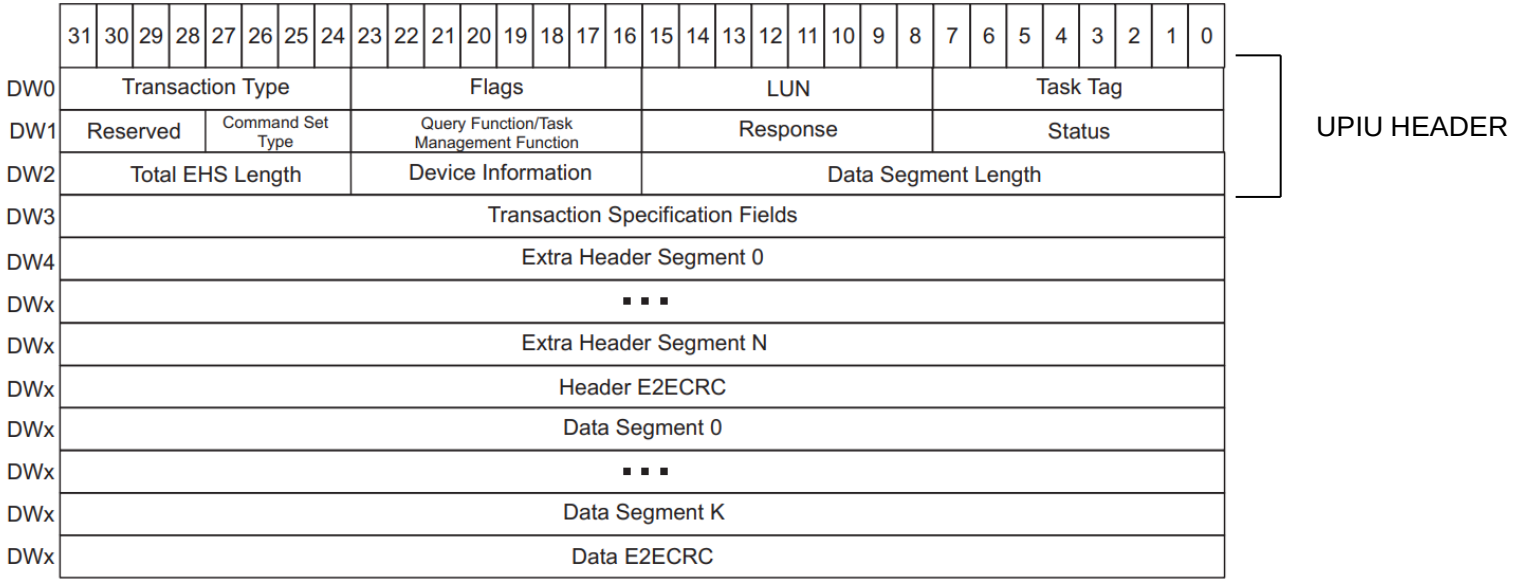
UFS System Overview



UFS Transport Layer

- Transactions consist of packets called UFS Protocol Information Units (UPIU).
- The host is the **Initiator** and the device is the **Target**.
- There are different types of UPIUs for handling SCSI commands, data operations, task management operations, query operations etc.
- Each transaction consists of:
  - One COMMAND UPIU
  - Zero or more DATA IN or DATA OUT UPIU
  - RESPONSE UPIU
- Each UPIU contains a 12 bytes header followed by optional fields depending on the type of UPIU.

UFS Transport Layer



General UPIU Structure

UPIU Structure

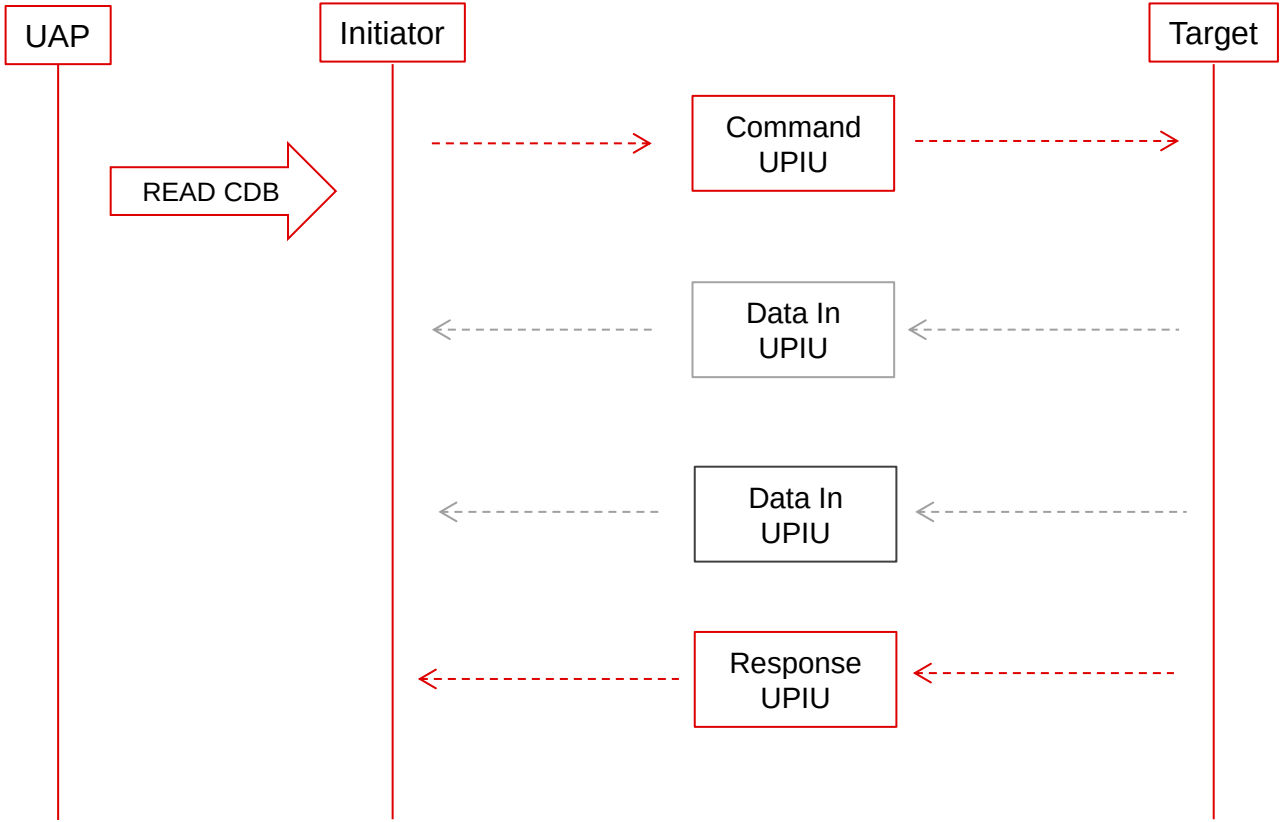
Transaction Type
Flags
LUN
Task Tag
Initiator ID
Command Set Type
Query/Task Manag. Fn
Response
Status
Total length
Device Info
Data Length

UPIU Header

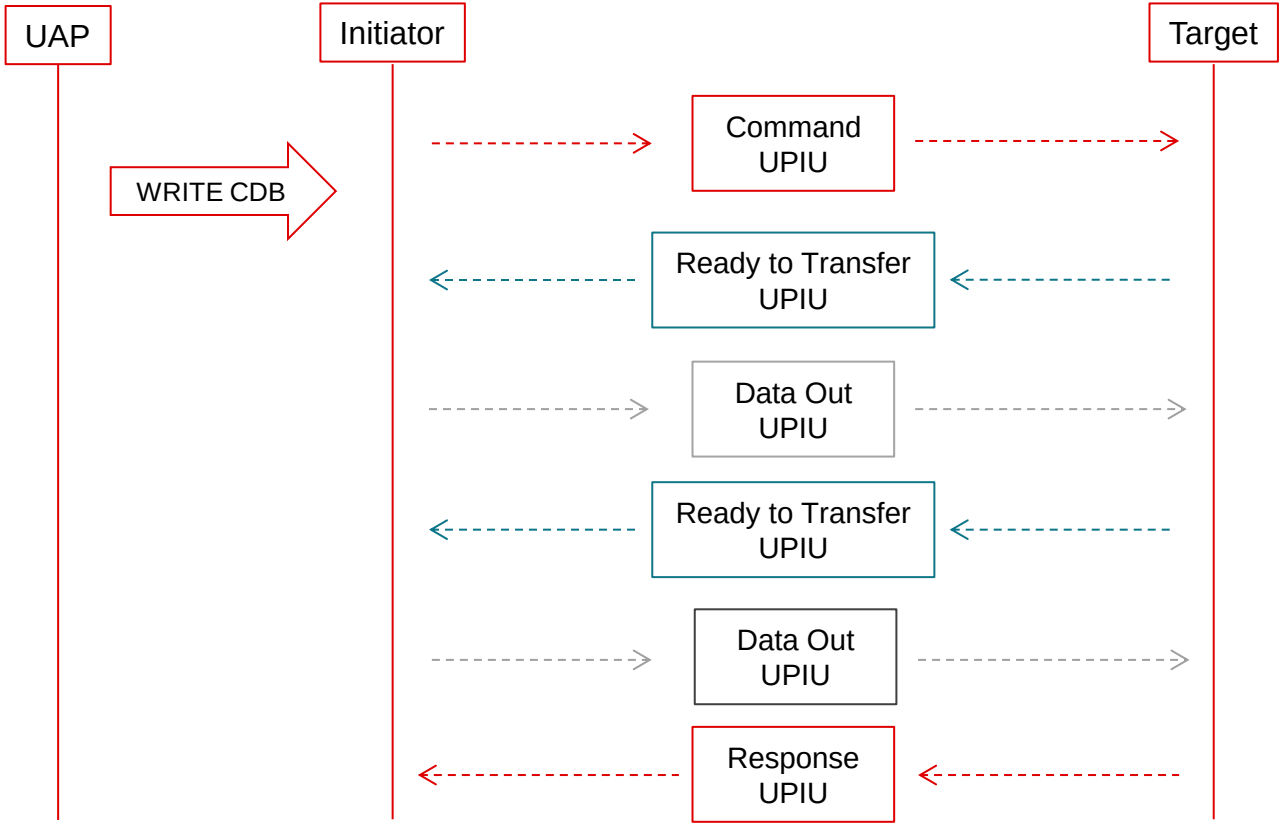
Initiator	**Target**
NOP OUT	NOP IN
Command	Response
Data Out	Data In
Task Manag. Req	Task Mang. Resp.
-	Ready to Transfer
Query Request	Query Response
-	Reject

Types of UPIUs

UTP Read Transaction

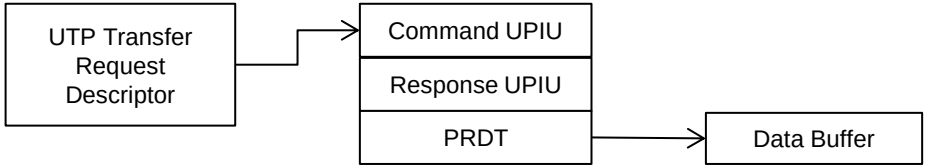


UTP Write Transaction



UTP Host Memory Configuration

```
struct utp_transfer_req_desc {

        /* DW 0-3 */
        struct request_desc_header header;

        /* DW 4-5*/
        __le32  command_desc_base_addr_lo;
        __le32  command_desc_base_addr_hi;

        /* DW 6 */
        __le16  response_upiu_length;
        __le16  response_upiu_offset;

        /* DW 7 */
        __le16  prd_table_length;
        __le16  prd_table_offset;
};
```



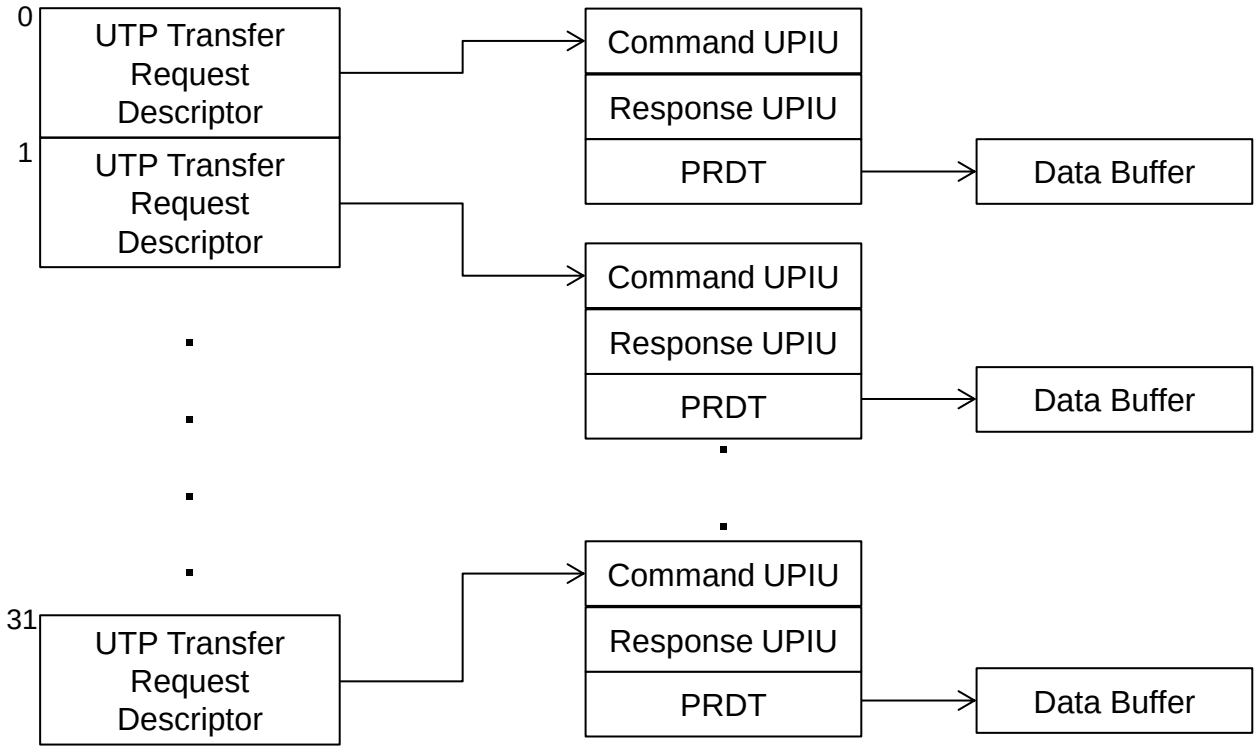
UTP Transfer Command Descriptor

```
struct utp_transfer_cmd_desc {
        u8 command_upiu[ALIGNED_UPIU_SIZE];
        u8 response_upiu[ALIGNED_UPIU_SIZE];
        struct ufshcd_sg_entry    prd_table[SG_ALL];
};
```

UFS Host Memory Configuration

Transaction Type
Flags
LUN
Task Tag
Initiator ID
Command Set Type
Query/Task Manag. Fn
Response
Status
Total length
Device Info
Data Length

UPIU Header



UFS Task Management

Transaction Type
Flags
LUN
Task Tag
Initiator ID
Command Set Type
Query/Task Manag. Fn
Response
Status
Total length
Device Info
Data Length

UPIU Header

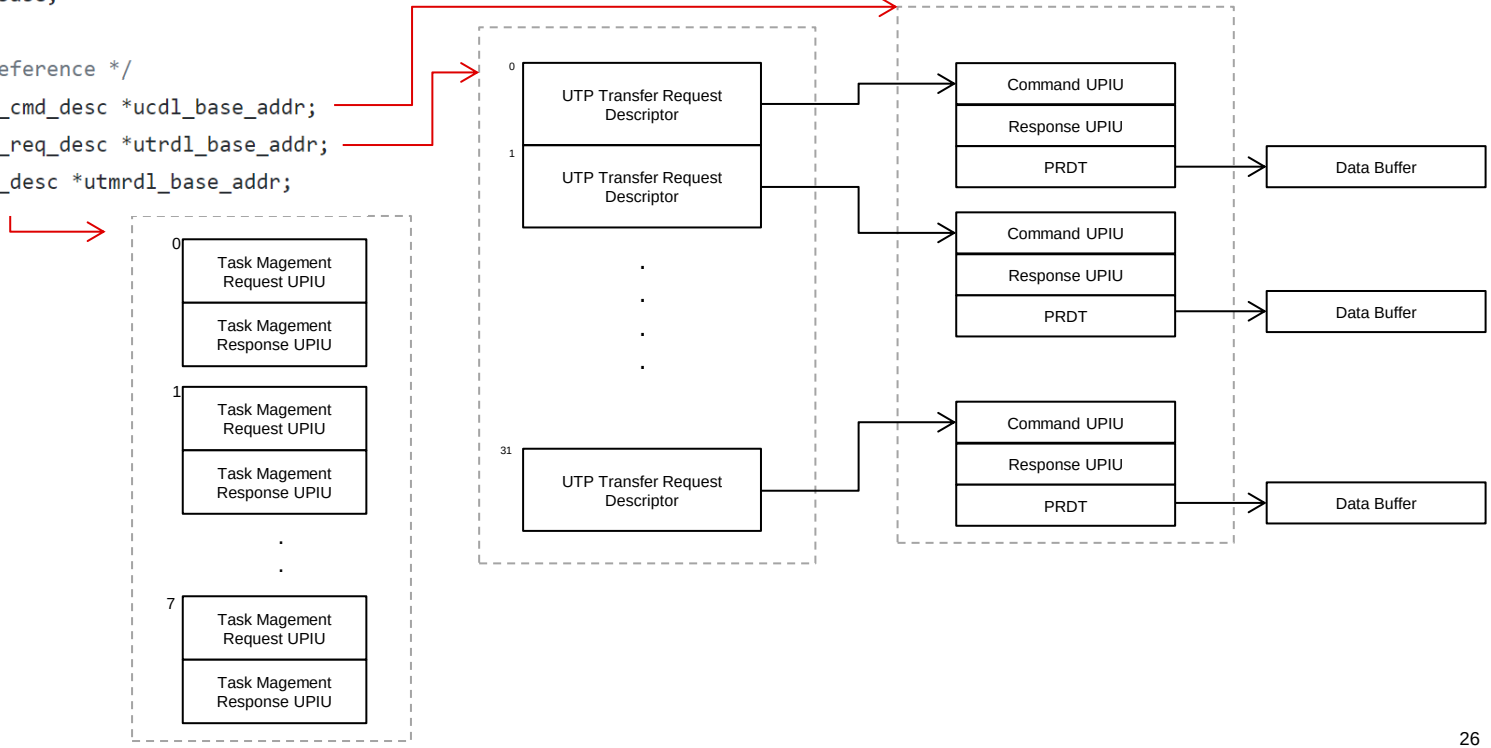
0	Task Management Request UPIU
	Task Management Response UPIU
1	Task Management Request UPIU
	Task Management Response UPIU
▪	
▪	
7	Task Management Request UPIU
	Task Management Response UPIU

```
struct utp_task_req_desc {
        /* DW 0-3 */
        struct request_desc_header header;

        /* DW 4-11 - Task request UPIU structure */
        struct utp_upiu_header  req_header;
        __be32                  input_param1;
        __be32                  input_param2;
        __be32                  input_param3;
        __be32                  __reserved1[2];

        /* DW 12-19 - Task Management Response UPIU structure */
        struct utp_upiu_header  rsp_header;
        __be32                  output_param1;
        __be32                  output_param2;
        __be32                  __reserved2[3];
};
```

```
        /* Task management functions */
        enum {
                UFS_ABORT_TASK          = 0x01,
                UFS_ABORT_TASK_SET      = 0x02,
                UFS_CLEAR_TASK_SET      = 0x04,
                UFS_LOGICAL_RESET       = 0x08,
                UFS_QUERY_TASK          = 0x80,
                UFS_QUERY_TASK_SET      = 0x81,
        };
```

UTP Host Memory Configuration

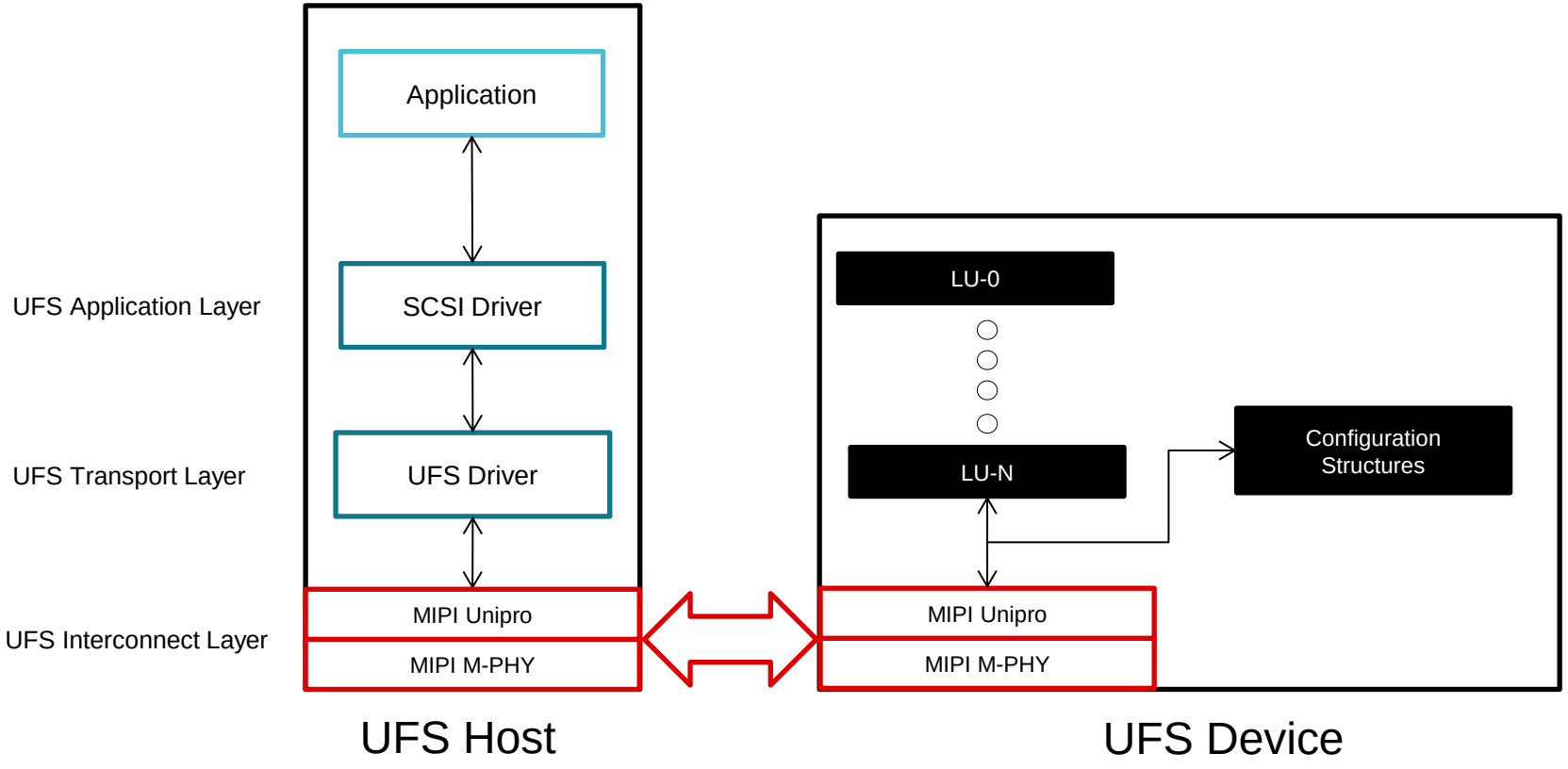
```
struct ufs_hba {
        void __iomem *mmio_base;

        /* Virtual memory reference */
        struct utp_transfer_cmd_desc *ucdl_base_addr;
        struct utp_transfer_req_desc *utrdl_base_addr;
        struct utp_task_req_desc *utmrdl_base_addr;
```



UFS Host Controller Interface

- Host Capabilities Registers: Version, Vendor ID, 64 bit support, number of slots etc.
- Operation and Runtime Registers: Interrupt handling (including aggregator), error status at each layer, host controller status etc.
- UTP Transfer Registers: Base addresses of UTPTRD table and UTPCMD table, doorbell registers, complete notifier etc.
- Task Management Registers: Base address of UTPTM table, doorbell, complete notifiers etc.
- UIC Command Registers: Interconnect layer command register and arguments.

UFS System Overview



Query Request

Transaction Type
Flags
LUN
Task Tag
Initiator ID
Command Set Type
Query/Task Manag. Fn
Response
Status
Total length
Device Info
Data Length

UPIU Header

- Query request UPIU is used to access information about configuration, enumeration, device descriptors, flags, attributes etc.

Query Function	Query OPCODE	Operation
Any value	00h	NOP
Std. Read	01h	Read Descriptor
Std. Write	02h	Write Descriptor
Std. Read	03h	Read Attribute
Std. Write	04h	Write Attribute
Std. Read	05h	Read Flag
Std. Write	06h	Set Flag
Std. Write	07h	Clear Flag
Std. Write	08h	Toggle Flag

Query functions

UFS Descriptors, Flags and Attributes

UFS Descriptor

- Independently addressable data structure describing something about the device
- Mostly read only except Configuration Descriptor

Query Function	Query OPCODE	Operation
Any value	00h	NOP
Std. Read	01h	Read Descriptor
Std. Write	02h	Write Descriptor

Query UPIU opcode

Descriptor IDN	Descriptor Type
00h	Device
01h	Configuration
02h	Unit
03h	-
04h	Interconnect
05h	String
06	-
07h	Geometry
08h	Power
09h	Device Health

Types of Descriptors

UFS Descriptors, Flags and Attributes

UFS Flags

- A single Boolean value that can be set On or Off. Can be set, cleared or toggled.

Query Function	Query OPCODE	Operation
Std. Read	05h	Read Flag
Std. Write	06h	Set Flag
Std. Write	07h	Clear Flag
Std. Write	08h	Toggle Flag

Query UPIU opcode

Flag IDN	Name
01h	fDeviceInit
02h	fPermanentWPEn
03h	fPowerOnWPEn
04h	fBackgroundOpsEn
05h	fDeviceLifeSpanModeEn
06h	fPurgeEnable
07h	fRefreshEnable
08h	fPhyResourceRemoval
09h	fBusyRTC
0Bh	fPermanentlyDisableFwUpdate

Types of Descriptors

UFS Descriptors, Flags and Attributes

UFS Attributes

- A single parameter that represents a specific range of numerical values that can be set or read.

Query Function	Query OPCODE	Operation
Std. Read	03h	Read Attribute
Std. Write	04h	Write Attribute

Query UPIU opcode

Attribute IDN	Name
00h	bBootLunEn
02h	bCurrentPowerMode
03h	bActiveICCLevel
04h	bOutOfOrderDataEn
05h	bBackgroundOpStatus
06h	bPurgeStatus
07h	bMaxDataInSize
08h	bMaxDataOutSize
09h	dDynCapNeeded
0Ah	bRefClkFreq

Types of Descriptors

Kernel Implementation

- All the drivers can be found under drivers/scsi/ufs/
- Documentation/scsi/ufs.txt – for information about device enumeration steps
- Documentation/devicetree/bindings/ufs/ - for device tree node implementation
- ufshcd_pltrfm_init() : Call this from your probe()

Kernel Implementation

```
[  279.213830] cdns-ufshcd 4e84000.ufs: ufshcd_populate_vreg: Unable to find vdd-hba-supply regulator, assuming enabled
[  279.224369] cdns-ufshcd 4e84000.ufs: ufshcd_populate_vreg: Unable to find vcc-supply regulator, assuming enabled
[  279.234554] cdns-ufshcd 4e84000.ufs: ufshcd_populate_vreg: Unable to find vccq-supply regulator, assuming enabled
[  279.244809] cdns-ufshcd 4e84000.ufs: ufshcd_populate_vreg: Unable to find vccq2-supply regulator, assuming enabled
[  279.255684] scsi host1: ufshcd
[  279.288184] cdns-ufshcd 4e84000.ufs: ufshcd_print_pwr_info:[RX, TX]: gear=[1, 1], lane[1, 1], pwr[SLOWAUTO_MODE, SLOWAUTO_MODE], rate = 0
[  279.325605] cdns-ufshcd 4e84000.ufs: ufshcd_print_pwr_info:[RX, TX]: gear=[3, 3], lane[2, 2], pwr[FAST MODE, FAST MODE], rate = 2
[  279.337271] cdns-ufshcd 4e84000.ufs: ufshcd_find_max_sup_active_icc_level: Regulator capability was not set, actvIccLevel=0
[  279.349037] scsi 1:0:0:49488: Well-known LUN    TOSHIBA  THGAF8G8T23BAILB 0300 PQ: 0 ANSI: 6
[  279.357988] scsi 1:0:0:49476: Well-known LUN    TOSHIBA  THGAF8G8T23BAILB 0300 PQ: 0 ANSI: 6
[  279.367150] cdns-ufshcd 4e84000.ufs: ufshcd_scsi_add_wlus: BOOT WLUN not found
[  279.375452] scsi 1:0:0:0: Direct-Access     TOSHIBA  THGAF8G8T23BAILB 0300 PQ: 0 ANSI: 6
[  279.383906] sd 1:0:0:0: Power-on or device reset occurred
[  279.390310] sd 1:0:0:0: [sda] 8192 4096-byte logical blocks: (33.6 MB/32.0 MiB)
[  279.397748] sd 1:0:0:0: [sda] Write Protect is off
[  279.398303] scsi 1:0:0:1: Direct-Access     TOSHIBA  THGAF8G8T23BAILB 0300 PQ: 0 ANSI: 6
[  279.410705] sd 1:0:0:0: [sda] Write cache: enabled, read cache: enabled, supports DPO and FUA
[  279.419347] sd 1:0:0:0: [sda] Optimal transfer size 65536 bytes
[  279.427141] sd 1:0:0:1: Power-on or device reset occurred
[  279.432820] sd 1:0:0:1: [sdb] 7808000 4096-byte logical blocks: (32.0 GB/29.8 GiB)
[  279.440598] sd 1:0:0:1: [sdb] Write Protect is off
[  279.445666] sd 1:0:0:1: [sdb] Write cache: enabled, read cache: enabled, supports DPO and FUA
[  279.454740] sd 1:0:0:1: [sdb] Optimal transfer size 65536 bytes
[  279.460876] sd 1:0:0:0: [sda] Attached SCSI disk
[  279.469225]  sdb: sdb1
[  279.473882] sd 1:0:0:1: [sdb] Attached SCSI disk
```

Configuring the device from user space

- UFS appears as a SCSI generic block device “/dev/bsg/ufs-bsg” that can accept an ioctl system call.
- User space can allocate *struct sg_io_v4* and send in the ioctl system call with request code SG_IO.

```
int ioctl(int fd, SG_IO, struct sg_io_v4 *)
```

ufs-utils

https://github.com/westerndigitalcorporation/ufs-utils

ufs-utils < desc | attr | fl | err_hist | uic >

- Examples:
  - Read Configuration descriptor

    ufs-utils desc –t 1 –p /dev/bsg/ufs-bsg

  - Set Background operations flag

    ufs-utils fl –t 4 –e –p /dev/bsg/ufs-bsg

U-boot Implementation

- Available in 2020.01 release (got merged this Thursday!).
- drivers/ufs/ufs.c
- cmd/ufs.c
  - Contains:
    - basic command to initialize the ufs device, detect all LUNs and register them as scsi devices.
  - Does not contain:
    - commands to access/configure descriptors/flags/attributes

Patches Welcome ☺

References

- Universal Flash Storage 3.0
- UFS Host Controller Interface (UFSHCI) version 3.0
- https://git.kernel.org
- J721E Technical Reference Manual: http://www.ti.com/lit/ug/spruil1/spruil1.pdf

Thank You