|  |
| --- |
| ***UNI91K UFS***  ***ITEM List / API Function*** |
|  |

****

**Memory Division/R&D Group**

**Software team**

목차

[2. Source Tree 7](#_Toc481133937)

[3. Test Diagram 8](#_Toc481133944)

[4. How to make test item 9](#_Toc481133945)

[4.1. Test Item 작성 9](#_Toc481133946)

[5. Host Communication API Function 13](#_Toc481133947)

[5.1. ul\_sprintf() 13](#_Toc481133948)

[5.2. ul\_host\_printf() 13](#_Toc481133949)

[5.3. ul\_uart\_printf() 13](#_Toc481133950)

[5.4. ul\_uart\_fileprintf() 14](#_Toc481133951)

[5.5. ul\_uart\_getcommand() 14](#_Toc481133952)

[5.6. ul\_setitemblock\_startinfo() 14](#_Toc481133953)

[5.7. ul\_setitemblock\_endinfo() 15](#_Toc481133954)

[5.8. ul\_setitemblock\_errorinfo() 15](#_Toc481133955)

[5.9. ul\_setitemblock\_slewup\_setvs() 16](#_Toc481133956)

[5.10. ul\_setitemblock\_measurevoltage() 17](#_Toc481133957)

[5.11. ul\_setitemblock\_measurecurrent() 17](#_Toc481133958)

[5.12. ul\_setitemblock\_get\_voltage() 17](#_Toc481133959)

[5.13. ul\_setitemblock\_get\_current() 17](#_Toc481133960)

[5.14. ul\_setitemblock\_finish() 18](#_Toc481133961)

[5.15. ul\_setitemblock\_recipe() 18](#_Toc481133962)

[5.16. ul\_setitemblock\_temperature() 18](#_Toc481133963)

[5.17. ul\_setitemblock\_hostinfo() 18](#_Toc481133964)

[5.18. ul\_setitemblock\_changeuartlevel() 19](#_Toc481133965)

[6. UART API Function 19](#_Toc481133966)

[6.1. ul\_uart\_interrupt\_initialize() 19](#_Toc481133967)

[6.2. ul\_uart\_interrupt\_enable() 19](#_Toc481133968)

[6.3. ul\_uart\_interrupt\_disable() 20](#_Toc481133969)

[6.4. ul\_uart\_interrupt\_received\_ready() 20](#_Toc481133970)

[6.5. ul\_uart\_interrupt\_received\_end() 20](#_Toc481133971)

[6.6. ul\_uart\_interrupt\_received\_checkword\_end() 20](#_Toc481133972)

[6.7. ul\_uart\_interrupt\_send() 20](#_Toc481133973)

[6.8. ul\_uart\_display\_rxbuffer() 21](#_Toc481133974)

[6.9. ul\_uart\_set\_baudrate() 21](#_Toc481133975)

[6.10. ul\_uart\_realtime\_log() 21](#_Toc481133976)

[6.11. ul\_uart\_realtime\_enable() 21](#_Toc481133977)

[6.12. ul\_get\_firmwareinfo () 21](#_Toc481133978)

[6.13. ul\_serial\_flash\_tx() 22](#_Toc481133979)

[6.14. ul\_serial\_fwimage() 22](#_Toc481133980)

[7. UFS API Function 23](#_Toc481133981)

[7.1. ul\_ufs\_issue\_requestlist() 23](#_Toc481133982)

[7.2. ul\_ufs\_polling\_requestlist() 23](#_Toc481133983)

[7.3. ul\_ufs\_check\_response\_upiu() 23](#_Toc481133984)

[7.4. ul\_ufs\_check\_requestlist() 24](#_Toc481133985)

[7.5. ul\_ufs\_get\_prdtaddress() 25](#_Toc481133986)

[7.6. ul\_ufs\_build\_nop\_upiu() 25](#_Toc481133987)

[7.7. ul\_ufs\_build\_nop\_requestlist() 25](#_Toc481133988)

[7.8. ul\_scsi\_build\_inquiry() 26](#_Toc481133989)

[7.9. ul\_scsi\_build\_modeselect10() 27](#_Toc481133990)

[7.10. ul\_scsi\_build\_modesense10() 27](#_Toc481133991)

[7.11. ul\_scsi\_build\_read6() 28](#_Toc481133992)

[7.12. ul\_scsi\_build\_read10() 28](#_Toc481133993)

[7.13. ul\_scsi\_build\_read16() 29](#_Toc481133994)

[7.14. ul\_scsi\_build\_readcapacity10() 29](#_Toc481133995)

[7.15. ul\_scsi\_build\_readcapacity16() 29](#_Toc481133996)

[7.16. ul\_scsi\_build\_startstopunit() 30](#_Toc481133997)

[7.17. ul\_scsi\_build\_testunitready() 30](#_Toc481133998)

[7.18. ul\_scsi\_build\_reportlun() 30](#_Toc481133999)

[7.19. ul\_scsi\_build\_verify10() 31](#_Toc481134000)

[7.20. ul\_scsi\_build\_write6() 31](#_Toc481134001)

[7.21. ul\_scsi\_build\_write10() 32](#_Toc481134002)

[7.22. ul\_scsi\_build\_write16() 33](#_Toc481134003)

[7.23. ul\_scsi\_build\_requestsense() 34](#_Toc481134004)

[7.24. ul\_scsi\_build\_formatunit() 34](#_Toc481134005)

[7.25. ul\_scsi\_build\_prefetch10() 35](#_Toc481134006)

[7.26. ul\_scsi\_build\_prefetch16() 35](#_Toc481134007)

[7.27. ul\_scsi\_build\_securityprotocolin() 35](#_Toc481134008)

[7.28. ul\_scsi\_build\_securityprotocolout() 36](#_Toc481134009)

[7.29. ul\_scsi\_build\_senddiagnostic() 37](#_Toc481134010)

[7.30. ul\_scsi\_build\_synchronizecache10() 38](#_Toc481134011)

[7.31. ul\_scsi\_build\_synchronizecache16() 38](#_Toc481134012)

[7.32. ul\_scsi\_build\_unmap() 38](#_Toc481134013)

[7.33. ul\_scsi\_build\_readbuffer() 39](#_Toc481134014)

[7.34. ul\_scsi\_build\_writebuffer() 40](#_Toc481134015)

[7.35. ul\_ufs\_convert\_msb\_lsb\_char2char() 40](#_Toc481134016)

[7.36. ul\_ufs\_convert\_msb\_lsb\_int2char() 40](#_Toc481134017)

[7.37. ul\_ufs\_convert\_msb\_lsb\_longlong2char() 40](#_Toc481134018)

[7.38. ul\_ufs\_convert\_msb\_lsb\_char2longlong() 41](#_Toc481134019)

[7.39. ul\_ufs\_convert\_msb\_lsb\_char2int() 41](#_Toc481134020)

[7.40. ul\_ufs\_get\_prdtaddress() 41](#_Toc481134021)

[7.41. ul\_ufs\_build\_query\_upiu() 42](#_Toc481134022)

[7.42. ul\_ufs\_build\_query\_requestlist() 44](#_Toc481134023)

[7.43. ul\_ufs\_get\_query\_attr\_value() 44](#_Toc481134024)

[7.44. ul\_ufs\_get\_query\_data\_value() 45](#_Toc481134025)

[7.45. ul\_ufs\_build\_task\_management\_upiu() 46](#_Toc481134026)

[7.46. ul\_ufs\_issue\_task\_management\_list() 46](#_Toc481134027)

[7.47. ul\_ufs\_check\_taskmanagementlist() 46](#_Toc481134028)

[7.48. ul\_ufs\_send\_UICcmd() 47](#_Toc481134029)

[7.49. ul\_ufs\_get\_UICcmdMIBvalue() 48](#_Toc481134030)

[7.50. ul\_ufs\_display\_UICcmdresponsecode() 48](#_Toc481134031)

[7.51. ul\_ufs\_get\_UICcmdresponsecode() 48](#_Toc481134032)

# Source Tree

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | **비고** |
| **uni91k\_pcie** |  |
|  | **src** |  |
|  | **IntLib** |  |
|  | **periph.h** |  | **LIBRARY 형태로 제공**  **(libuni.a)** |
| **si5338.h** |
| **unicpld.h** |
| **unicommdef.h** |
| **unicommdef\_m.h** |
| **unidefine.h** |
| **unioplib.h** |
| **uniufslib.h** |
| **Testsript** |  |  | **\* Recipe mode을 사용하지 않을 경우, 해당 Script를 이용** |
|  | **Testsript** |  |
|  | **testscript.h** |
| **Script** |  |
|  | **testscipt.c** |
| bsp.c |  |  | **TEST ITEM이 구현된 파일 testitem.h testitem\_main.h testitem.c testitem\_main.c \* USER는 상기 파일로 TEST ITEM을 구성** |
| bsp.h |  |
| control.c(main.c) |  |
| includes.h |  |
| lscript.ld |  |
| platform.c |  |
| platform.h |  |
| platfrom\_config.h |  |
| **testitem.c** |  |
| **testitem.h** |  |
| **testitem\_main.c** |  |
| **testitem\_main.h** |  |
| ucos2\_cfg.h |  |

# Test Diagram

# How to make test item

## Test Item 작성

1. **Testitem.c : test item 함수작성(UL API함수를 이용하여 원하는 테스트 sequence를 작성)**

**int item\_user** (int testnum, char\* testname, int failcategory, int passcategory, int stop\_continue, int com\_timeout, int item\_timeout, **int user\_data,** int dut\_control)

{

unsigned int itestnum = testnum;

char\* ptestname = testname;

unsigned int ifailcategory = failcategory;

unsigned int ipasscategory = passcategory;

unsigned int istop\_continue = stop\_continue;

unsigned int icom\_timeout = com\_timeout;

unsigned int iitem\_timeout = item\_timeout;

**unsigned int iuser\_data = user\_data; // USER argument 추가**

..........

**// 1. Item start**

ul\_setitemblock\_startinfo(dut\_control, testnum, testname, judgetype, com\_timeout, BP\_NOUSE, polltimeout);

**// 2. Test sequence 추가**

ul\_host\_printf(dut\_control, "Hellow uni91k ....");

**// 3. Item end**

ul\_setitemblock\_endinfo(com\_timeout, OFF\_DEVICEDATA, dut\_control);

}

**1) testitem.c 파일에서 Test item 함수를 작성한다**

( test item 함수의 이름은 item\_xxx로 명명하는 것을 권장)

**2) 모든 test item함수에는 아래 argument들이 기본적으로 요구됨**

- int testnum : Test item의 test number

- char\* testname : Test item의 test name

- int failcategory : Test item의 fail category number

- int passcategory : Test item의 pass category number

- int stop\_continue : Test item의 stop/continue 설정

- int com\_timeout : OCP & Host 간의 통신 time out 설정

- int item\_timeout : Test item의 Test time out 설정

**3) test item 필요에 따라 argument 추가하기**

- **int user\_data** : 필요에 따라 argument 추가

1. **Testitem.h : test item 원형 함수 추가**

**int item\_user** (int testnum, char\* testname, int failcategory, int passcategory, int stop\_continue, int com\_timeout, int item\_timeout, , **int user\_data,** int dut\_control);

1. **Testitem\_main.c : Test script 또는 recipe를 통해 Test item들을 호출할 수 있도록 등록**

**int testitemRp(int dut\_control)**

**{**

testnum = stringtoint(test\_value[index++]); // 1. TEST\_NUMBER

testname = test\_value[index++]; // 2. TEST\_NAME

failcategory = stringtoint(test\_value[index++]); // 3. TEST\_FAIL\_CATAGORY

passcategory = stringtoint(test\_value[index++]); // 4. TEST\_PASS\_CATAGORY

stop\_continue = stringtostopcontinue(test\_value[index++]); // 5. stop\_continue

com\_timeout = stringtoint(test\_value[index++]); // 6. OCP\_WAIT\_TIMEOUT

item\_timeout = stringtoint(test\_value[index++]); // 7. TEST\_TIMEOUT

........................................

........................................

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// **item\_user** && Option Argument

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**else if(strcmp(test\_function," item\_user ") == 0)**

**{**

**int** **user\_data = stringtoint(test\_value[index++]);**

**//funtion call**

**iret=****item\_user(testnum, testname, failcategory, passcategory, stop\_continue, com\_timeout, tem\_timeout, user\_data, dut\_control);**

**}**

}

**1) argument parsing 함수**

- int stringtoint(char \*svalue) : 문자열을 int type형으로 parsing

- QWORD **stringtolong**(char \*svalue) : 문자열을 long type형으로 parsing

- int **stringtostopcontinue**(char \*svalue): 문자열(CONTINUE, STOP)을 int type형으로 parsing

- int **stringtojudgetype**(char \*svalue): 문자열(TIMEOUT, RFIS, AHCI)을 int type형으로 parsing

- int **stringtogentype**(char \*svalue): 문자열(GEN1, GEN2, GEN3)을 int type형으로 parsing

- int **stringtotransfertype**(char \*svalue): 문자열(DT\_PIO, DT\_DMA, DT\_FPDMA, DT\_FPDMA\_EXT)

을 int type형으로 parsing

- int **stringtotaddressingtype**(char \*svalue): 문자열(SEQ, RANDOM)을 int type형으로 parsing

- int **stringtodatatype**(char \*svalue): 문자열(HFIXED, HRND7, HRND15, HRND31, BIN0, BIN1, BIN2)

을 int type형으로 parsing

- int **stringtocompareenable**(char \*svalue): 문자열(CPN, CPE)을 int type형으로 parsing

- int **stringtoenabledisable**(char \*svalue)문자열(ENABLE, DISABLE)을 int type형으로 parsing

- int **stringtopmlmode**(char \*svalue): 문자열(PML11, PML12, AS\_PML11, AS\_PML12)

을 int type형으로 parsing

\* 필요에 따라 parsing을 위해 문자열을 추가 가능

1. **Recipe conf 파일 수정하기 : recipepara\_AHCINTB.conf**

[FUNCTION\_COUNT]

**3**

[FUNCTION\_NAME]

1 temperature

2 item\_poweron

**3 item\_user**

[FUNCTION\_PARAMETER\_COUNT]

1 2

2 13

**3 8**

[FUNCTION\_PARAMETER]

...............................................

**#item\_user**

**3** **01 UNSIGNED\_INT TEST\_NUMBER 10 NO\_USE**

**3** **02 STRING TEST\_NAME ITEM\_USER USE**

**3 03 UNSIGNED\_INT TEST\_FAILCATEGORY 1 USE**

**3 04 UNSIGNED\_INT TEST\_PASSCATEGORY 501 USE**

**3** **05 SELECT STOP\_CONTINUE STOP USE**

**3 06 UNSIGNED\_INT COM\_TIMEOUT (Sec.) 1000 NO\_USE**

**3** **07 UNSIGNED\_INT ITEM\_TIMEOUT(Min.) 5 USE**

**3 08 UNSIGNED\_INT USER\_DATA 100 USE**

[RANGED\_TYPE]

#item\_poweron

2 08 2.8~3.6

[SELECT\_TYPE]

**#item\_user**

**3 05 STOP**

**3 05 CONTINUE**

**1) FUNCTION\_COUNT]**

- Recipe에서 사용 가능한 Total Test item 개수

**2) [FUNCTION\_NAME]**

- Recipe에서 사용되는 고유 Test function ID

- Test Item function name

**3) [FUNCTION\_PARAMETER\_COUNT]**

- Test function ID

- Test Item에서 사용되는 argument 개수

**4) [FUNCTION\_PARAMETER]**

- Test function ID

- Recipe에서 사용되는 Test item argument의 ID

- Recipe에서 사용되는 Test item argument TYPE (**UNSIGNED\_INT, STRING,** **RANGED\_FLOAT,** **SELECT)**

\* 현재 총 4개의 type이로 구성됨

- 각 Test Item Argument에 대한 name

- 각 Test Item Argument에 대한 초기값

\* Recipe에서 Test Item 생성 시 자동 초기값들이 기입된다.

- Test Item Argument 사용 여부 설정 (USE, NO\_USE)

**4) [RANGED\_TYPE]**

- Test function ID

- Test item argument의 ID (Test item argument TYPE이 **RANGED\_FLOAT** 로 설정된 ID)

- Float 범위 설정

**5) [SELECT\_TYPE]**

- Test function ID

- Test item argument의 ID (Test item argument TYPE이 **SELECT** 로 설정된 ID)

- 문자열 enum 값 설정

1. **Recipe conf 파일을 ZONE conf파일로 copy하기**

**uni91k\zone1\UNI91K\_maint\conf\utility\recipepara\_AHCINTB.conf**

**uni91k\zone1\UNI91K\_maint\conf\utility\recipepara\_AHCINTB.conf**

# Host Communication API Function

Host API 함수는 Host PC로 부터 Data정보를 얻어오거나 Host PC에게 Data 정보를 제공할 때 사용되는 함수들을 의미한다.

## ul\_sprintf()

|  |  |
| --- | --- |
| Function | void ul\_sprintf(char \*buf, char \*fmt, ...) |
| Description | 표준 C언어의 sprintf()함수와 동일한 기능을 가짐 |
| Argument | 1) \*buf  2) \*fmt |
| Return | void |

## ul\_host\_printf()

|  |  |
| --- | --- |
| Function | void ul\_host\_printf(int dutid, char \*fmt, ...) |
| Description | User가 원하는 Data 또는 Log을 PMS창에 보여주고 동시에 특정 파일에 기록하는 함수  \* uni91k\TestResult 폴더 안에 DUTXX.rst 파일 이름으로 저장 |
| Argument | 1) dutid : Dut ID  2) \*fmt : print data |
| Return | void |

|  |
| --- |
| **Example Code** |
| char info[1024] ={0,};  ul\_sprintf(info, "TEST BLOCK START = %d\n", maxlba - istartlba);  ul\_host\_printf(dut\_control, info); |

## ul\_uart\_printf()

|  |  |
| --- | --- |
| Function | void ul\_uart\_printf(int dutid, char \*fmt, ...) |
| Description | UART Log을 PMS창에 보여주고 동시에 특정 파일에 기록하는 함수  \* uni91k\TestResult 폴더 안에 UARTLOG\_DUTXX.rst 파일 이름으로 저장 |
| Argument | 1) dutid : Dut ID  2) \*fmt : print data |
| Return | void |

## ul\_uart\_fileprintf()

|  |  |
| --- | --- |
| Function | void ul\_uart\_fileprintf (int dutid, char \*fmt, ...) |
| Description | UART Log들을 특정 파일에 기록하는 함수  \* uni91k\TestResult 폴더 안에 UARTLOG\_DUTXX.rst 이름으로 저장 |
| Argument | 1) dutid : Dut ID  2) \*fmt : print data |
| Return | void |

## ul\_uart\_getcommand()

|  |  |
| --- | --- |
| Function | void ul\_uart\_getcommand(int dutid, char \*uartcommand) |
| Description | UART Viewer GUI로 부터 입력된 UART Send(TX) data을 얻어옴 |
| Argument | 1) dutid : Dut ID  2) \*uartcommand : Uart Send(TX) Data을 얻어 온다  \* Uart Send Data는 Maxim 256 Byte까지 얻어옴 |
| Return | void |

|  |
| --- |
| **Example Code** |
| char buffer[256] = {0,};  char uartcommand[256] = {0,};  memset(buffer, 0x0, sizeof(buffer));  memset(uartcommand, 0x0, sizeof(uartcommand));  ul\_uart\_getcommand(dut, buffer);  ul\_sprintf(uartcommand,"%s\r",buffer);  ul\_uart\_interrupt\_send(uartcommand, dut); |

## ul\_setitemblock\_startinfo()

|  |  |
| --- | --- |
| Function | void ul\_setitemblock\_startinfo(int itestno, char\* testname, int failcategory, int passcategory,  int judgetype, int timeout\_cnt, int host\_poll\_timout\_max, int dutid) |
| Description | Host(PC)에게 Test Item 시작 정보 및 시작 시점을 알려준다  \* Host는 Test Item 시작 시점부터 Test Item 종료 시점까지 Test Item Time-out(host\_poll\_timout\_max) check  만약 Test Item Time-out이 발생되면 해당 Dut는 Fail처리 한다 |
| Argument | 1. int itestno : Test Number 2. char\* testname : Test Name 3. int failcategory : Test Fail Category Number (1~500 Number) 4. int passcategory : Test Pass Category Number (500 ~ 1000 Number) 5. int judgetype : Test Error 판단 조건   - TIMEOUT : Check Command Issue Time-out or Configuration Time-out  - SCT : Check Command SCT Error   1. int timeout\_cnt : communication time-out value between Host and PG (PGB) ( 1 = 50ms) 2. int host\_poll\_timout\_max : Test Item time-out value between starting and ending of Test-Item (1 = 1Min) 3. int dutid : Dut ID |
| Return | Host & PG간의 통신 결과를 리턴한다.  RET\_FAIL/ RET\_PASS |

## ul\_setitemblock\_endinfo()

|  |  |
| --- | --- |
| Function | int ul\_setitemblock\_endinfo(int timeout\_cnt, int dutid); |
| Description | Host(PC)에게 Test Item 종료 및 종료 시점을 알려준다  \* Host PC는 Test Item에 대한 Fail/Pass 결과를 처리하고 Test Item Time-out check를 정지한다. |
| Argument | 1. int timeout\_cnt : communication time-out value between Host and PG (PGB) ( 1 = 50ms) 2. int dutid : Dut ID |
| Return | Host & PG간의 통신 결과를 리턴한다.  RET\_FAIL/ RET\_PASS |

## ul\_setitemblock\_errorinfo()

|  |  |
| --- | --- |
| Function | void ul\_setitemblock\_errorinfo(int timeout\_cnt, QWORD lba\_error, unsigned int ahci\_error,  unsigned int rfis\_error, unsigned int rfis\_status, int error\_code, int category, int passfail, int mask, int dutid) |
| Description | Host(PC)에게 Test Item에 대한 Error 정보를 알려 준다. |
| Argument | 1. int timeout\_cnt : communication time-out value between Host and PG (PGB) ( 1 = 50ms) 2. QWORD lba\_error : Fail LBA 정보를 입력한다. 3. unsigned int ahci\_error : AHCI Error 정보를 입력한다(AHCI, SATA Device에서만 해당) 4. unsigned int rfis\_error : RFIS Error 정보를 입력한다 (AHCI, SATA Device에서만 해당)   또는 NVME SCT status code 정보를 입력한다. (NVME Device에서만 해당)   1. unsigned int rfis\_status : RFIS Status 정보를 입력한다 (AHCI, SATA Device에서만 해당)   또는 NVME SCT status code type 정보를 입력한다. (NVME Device에서만 해당)   1. int error\_code : Error Code를 입력한다   \* 아래 표를 참조   1. int category : Fail/Pass에 따라 설정된 Category Number를 입력한다. 2. int passfail : Fail/Pass 결과를 입력한다. (PF\_FAIL, PF\_PASS) 3. int mask : Fail/Pass에 따라 OCP DUT를 mask 시킬지 여부를 설정한다. (MASK\_ON, MASK\_OFF) 4. int dutid : Dut ID |
|  | Host & PG간의 통신 결과를 리턴한다.  RET\_FAIL/ RET\_PASS |

**Error Code**

|  |  |  |
| --- | --- | --- |
| Define | Error Code |  |
| ERR\_PHYREADY | 0x01 | - AHCI Device (PCIE AHCI, SATA)  : Phyready fail이 발생될 때 해당 Error code 사용 |
| ERR\_SIGNATURE | 0x02 | - PCIE에서 물리적인 Interface 이상으로 판단될 때 해당 Error code를 사용 |
| ERR\_CI\_TIMEOUT | 0x03 | - AHCI Device (PCIE AHCI, SATA) 또는 NVME Device  : Device로부터 일정 시간 동안 Command에 대한 Response가 없을 시 해당 Error code를 사용 |
| ERR\_COMPARE | 0x04 | DMA read verify 에러 발생 시 상기 Error code를 사용 |
| ERR\_NOTDEVICE | 0x05 | - PCIE Device (PCIE AHCI, NVME)  : PCIe Configuration fail이 발생될 때 해당 Error code 사용 |
| ERR\_RFIS  ERR\_STC | 0x06 | - AHCI Device (PCIE AHCI, SATA)  : RFIS Error가 발생 시 해당 Error code를 사용  - NVME Device  : STC Error가 발생 시 해당 Error code를 사용 |
| ERR\_SYSTEM | 0x07 | - AHCI Device (PCIE AHCI, SATA)  : PxSERR Error 값들이 존재할 때 해당 Error code를 사용 |
| ERR\_UNKNOWN | 0x08 | - 알 수 없는 Error Code인 경우에 사용 |
| ERR\_FW | 0x09 | - Device Firmware download Fail 시 해당 Error code를 사용 |
| ERR\_USER | 0x0A | - User program에 의한 Fail Error code로 사용 |

|  |
| --- |
| **Example Code** |
| ilba\_error = ul\_geterrorlbaaddress(dut\_control);  istatus\_code = ul\_getstatuscode(dut\_control);  istatus\_code\_type = ul\_getstatuscodetype(dut\_control);  if(ierror\_code == ERR\_SCT)  {  ul\_setitemblock\_errorinfo(icom\_timeout, ilba\_error, 0, istatus\_code, istatus\_code\_type, ERR\_SCT, ifailcategory, PF\_FAIL, MASK\_ON ,dut\_control);  }  else if(ierror\_code == ERR\_CI\_TIMEOUT)  {  ul\_setitemblock\_errorinfo(icom\_timeout, ilba\_error, 0, istatus\_code, istatus\_code\_type, ERR\_CI\_TIMEOUT, ifailcategory, PF\_FAIL, MASK\_ON ,dut\_control);  }  else if(ierror\_code == ERR\_PHYREADY)  {  ul\_setitemblock\_errorinfo(icom\_timeout, 0x0, 0x0, 0x0, 0x0, ERR\_PHYREADY, ifailcategory, PF\_FAIL, MASK\_ON ,dut\_control);  }  else if(ierror\_code != ERR\_NO)  {  ul\_setitemblock\_errorinfo(icom\_timeout, ilba\_error, 0, istatus\_code, istatus\_code\_type, ERR\_UNKNOWN, ifailcategory, PF\_FAIL, MASK\_ON ,dut\_control);  }  else // ERR\_NO  {  ul\_setitemblock\_errorinfo(icom\_timeout, ilba\_error, 0, istatus\_code, istatus\_code\_type, ERR\_NO, ipasscategory, PF\_PASS, MASK\_OFF ,dut\_control);  } |

## ul\_setitemblock\_slewup\_setvs()

|  |  |
| --- | --- |
| Function | float ul\_setitemblock\_slewup\_setvs(int timeout\_cnt, float startfvolt\_pps0, float startfvolt\_pps1,  float endfvolt\_pps0, float endfvolt\_pps1, float fvolt\_step, int time\_step, int dutid) |
| Description | Device Voltage(Power) Level을 설정한다 |
| Argument | 1. int timeout\_cnt : communication time-out value between Host and PG (PGB) ( 1 = 50ms) 2. float startfvolt\_pps0 : PPS0 자원의 Voltage 시작 값 (현재 설정 된 voltage 값을 의미) 3. float startfvolt\_pps1 : PPS1 자원의 Voltage 시작 값 (현재 설정 된 voltage 값을 의미) 4. float endfvolt\_pps0 : PPS0 자원의 Voltage 값 설정 (원하는 voltage 값 설정) 5. float endfvolt\_pps1 : PPS1 자원의 Voltage 값 설정 (원하는 voltage 값 설정), 6. float fvolt\_step : 단계별 voltage 증가 또는 감소 단위 7. int time\_step : 단계별 voltage 증가 또는 감소 시간 (mSec) 8. int dutid : Dut ID   \* NGFF Type인 경우 PPS0 argument만 사용  \* HHHL Type인 경우 PPS0, PPS1 argument들을 사용 |
| Return | Host & PG간의 통신 결과를 리턴한다.  RET\_FAIL/ RET\_PASS |

## ul\_setitemblock\_measurevoltage()

|  |  |
| --- | --- |
| Function | int ul\_setitemblock\_measurevoltage(int timeout\_cnt, int dutid) |
| Description | Device Voltage(power) 값을 측정하여 Return한다 |
| Argument | 1. int timeout\_cnt : communication time-out value between Host and PG (PGB) ( 1 = 50ms) 2. int dutid : Dut ID |
| Return | PPS0 자원의 Voltage 값을 리턴 한다 (mV) |

## ul\_setitemblock\_measurecurrent()

|  |  |
| --- | --- |
| Function | int ul\_setitemblock\_measurecurrent(int timeout\_cnt, int dutid) |
| Description | Device Current 값을 측정하여 Return한다 |
| Argument | 1. int timeout\_cnt : communication time-out value between Host and PG (PGB) ( 1 = 50ms) 2. int dutid : Dut ID |
| Return | PPS0 자원의 Current 값을 리턴 한다 (mA) |

## ul\_setitemblock\_get\_voltage()

|  |  |
| --- | --- |
| Function | int ul\_setitemblock\_get\_voltage(int dutid) |
| Description | Device Voltage(power) 값을 측정하여 Return한다  \* ul\_setitemblock\_measurevoltage 함수보다 보다 빠른 Voltage 측정이 가능 |
| Argument | 1. int dutid : Dut ID |
| Return | PPS0 자원의 Voltage 값을 리턴 한다 (mV) |

## ul\_setitemblock\_get\_current()

|  |  |
| --- | --- |
| Function | int ul\_setitemblock\_get\_current(int dutid) |
| Description | Device Current 값을 측정하여 Return한다  \* ul\_setitemblock\_measurecurrent 함수보다 보다 빠른 Current 측정이 가능 |
| Argument | 1. int dutid : Dut ID |
| Return | PPS0 자원의 Current 값을 리턴 한다 (mA) |

## ul\_setitemblock\_finish()

|  |  |
| --- | --- |
| Function | void ul\_setitemblock\_finish(int ibootload\_enable, int dutid) |
| Description | Host(PC)에게 Test Script 종료를 알려 준다  \* Host(PC)는 DUT에 대한 Test Script Finish 정보를 조사하고 모든 Dut가 finish 조건을 만족 할 경우 Test Cycle을 종료한다.  \* OCP(PG)에서는 Host에 대한 Finish Respone가 올 때까지 기다린다. |
| Argument | 1. int ibootload\_enable : 모든 Dut가 Finish 응답을 받을 경우 Boot load 모드로 전환할지에 대한 설정 2. int dutid : Dut ID |
| Return | NONE |

## ul\_setitemblock\_recipe()

|  |  |
| --- | --- |
| Function | int ul\_setitemblock\_recipe(char \*recipe\_data, int size, int timeout\_cnt, int dutid) |
| Description | Host(PC)로 부터 하나의 Test Item Script 정보를 얻어온다  \* OCP에서는 Test Item Script 문자열을 얻어서 Test Item Function, argument 정보들을 Parsing하여 사용한다. |
| Argument | 1. char \*recipe\_data : Test Item Script 문자열을 Return 받는다 2. char size : Return 받을 Test Item Script 문자열 크기 설정 3. int timeout\_cnt : communication time-out value between Host and PG (PGB) ( 1 = 50ms) 4. int dutid : Dut ID |
| Return | NONE |

## ul\_setitemblock\_temperature()

|  |  |
| --- | --- |
| Function | int ul\_setitemblock\_temperature(int dutid, float temp, int soaktime) |
| Description | Host(PC)에게 DUT에 대한 Temperature setup 정보를 알려 준다  \* Host(PC)는 모든 DUT에 대해 Temperature setup command를 check하고 모든 DUT에 대한 command가 들어 온 경우 Temperature 변경을 시작한다.  \* Host(pc)에서는 Temperature 변경이 완료 된 후 모든 DUT의 command에 대한 Response를 전달한다.  \* DUT에서는 Command Response가 올 때까지 기다린다. |
| Argument | 1. int dutid : Dut ID 2. float temp : Temperature 값 설정 3. int soaktime : soak time 설정 (1= Sec) |
| Return | Host & PG간의 통신 결과를 리턴한다.  RET\_FAIL/ RET\_PASS |

## ul\_setitemblock\_hostinfo()

|  |  |
| --- | --- |
| Function | int ul\_setitemblock\_hostinfo(int hostinfo\_code, char \*hostinfo\_data, int size, int timeout\_cnt, int dutid) |
| Description | Host(PC)로 부터 Lot setup 정보를 얻어 온다.  \* info code는 Host & OCP간에 미리 약속된 정보로서 info code에 따라 다양한 Lot setup 정보를 얻어온다. |
| Argument | 1. int hostinfo\_cod : Lot setup 정보를 얻기 위한 info code   e\_HOST\_INFO\_LOTID : Lot Setup 시 설정된 Lot Name 얻어옴  e\_HOST\_INFO\_MACHINEID : Lot Setup 시 설정된 Machine ID를 얻어옴  e\_HOST\_INFO\_STARTTIME : Lot Setup 설정 시간을 얻어옴  e\_HOST\_INFO\_STARTHOUT : Lot Setup 설정 시간을 얻어옴  e\_HOST\_INFO\_DEVICENAME : Lot Setup 시 설정된 Device Name을 얻어옴   1. int \*hostinfo\_data : Host로 부터 얻어온 DATA을 문자열로 리턴 2. int size : Host로 부터 얻어온 DATA 크기 3. int timeout\_cnt : communication time-out value between Host and PG (PGB) ( 1 = 50ms) 4. int dutid : Dut ID |
|  | Host & PG간의 통신 결과를 리턴한다.  RET\_FAIL/ RET\_PASS |

## ul\_setitemblock\_changeuartlevel()

|  |  |
| --- | --- |
| Function | int ul\_setitemblock\_changeuartlevel(int dutid, int iuartlevel, int test\_type) |
| Description | Host(PC)에게 DUT에 대한 Uart (voltage)Level Setup 정보를 알려 준다  \* Host(PC)는 모든 DUT에 대해 Uart Level Setup Command를 Check하고 모든 DUT에서 Command가 들어 온 경우 Uart Level 변경을 시작한다.  \* Host(pc)에서는 Uart Level 변경이 완료 된 후 모든 DUT의 Command에 대한 Response를 전달한다.  \* DUT에서는 Command Response가 올 때까지 기다린다. |
| Argument | 1. int dutid : Dut ID 2. int iuartlevel : UART Level 설정 값 (UART\_LEVEL1\_8V, UART\_LEVEL3\_3V) 3. int test\_type : UART Level Operation 설정 값 (TY\_BATCH, TY\_DUTBYDUT)   \* UART Level 변경은 DUT별 개별 제어가 불가능하다 |
| Return | Host & PG간의 통신 결과를 Return한다.  RET\_FAIL/ RET\_PASS |

# UART API Function

## ul\_uart\_interrupt\_initialize()

|  |  |
| --- | --- |
| Function | int ul\_uart\_interrupt\_initialize(XIntc \*int\_ctl) |
| Description | Uart 초기화 진행  ( Uart IP 초기화, Interrrupt Headler등록, Uart option 설정) |
| Argument | 1. XIntc \*int\_ctl : Interrupt controller address |
| Return | NONE |

## ul\_uart\_interrupt\_enable()

|  |  |
| --- | --- |
| Function | int ul\_uart\_interrupt\_enable(int dutid) |
| Description | Uart Interrupt Enable |
| Argument | 1. int dutid : Dut ID |
| Return | NONE |

## ul\_uart\_interrupt\_disable()

|  |  |
| --- | --- |
| Function | int ul\_uart\_interrupt\_disable(int dutid) |
| Description | Uart Interrupt Disable |
| Argument | 1. int dutid : Dut ID |
| Return | NONE |

## ul\_uart\_interrupt\_received\_ready()

|  |  |
| --- | --- |
| Function | int ul\_uart\_interrupt\_disable(int dutid) |
| Description | Uart Rx buffer를 초기화하고 Uart Rx data를 받을 준비를 한다. |
| Argument | 1. int dutid : Dut ID |
| Return | NONE |

## ul\_uart\_interrupt\_received\_end()

|  |  |
| --- | --- |
| Function | unsigned char \*ul\_uart\_interrupt\_received\_end(int sec, int \*recceived\_size, int dutid) |
| Description | 설정 시간(sec) 만큼 Uart Rx Data를 받고 Uart Rx data Buffer address와 Uart Rx size를 리턴한다. |
| Argument | 1. int sec : 설정 시간만큼 Uart Rx Data를 받는다 2. int \*recceived\_size : 설정 시간만큼 받은 RX data Size를 리턴한다. 3. int dutid : Dut ID |
| Return | Uart Rx buffer address를 리턴한다. |

## ul\_uart\_interrupt\_received\_checkword\_end()

|  |  |
| --- | --- |
| Function | ul\_uart\_interrupt\_received\_checkword\_end(int sec, unsigned char \*checkword, unsigned int checksize,  int dutid) |
| Description | 설정 시간(sec) 만큼 Uart Rx Data를 받고 Uart RX data에서 특정 단어(word)를 찾아 PASS/FAIL을 리턴한다. |
| Argument | 1. int sec : 설정 시간만큼 Uart Rx Data를 받는다 2. unsigned char \*checkword : Uart RxData에서 찾고자 하는 단어를 설정 3. unsigned int checksize : 특정 단어(word)의 length 4. int dutid : Dut ID |
| Return | RX Data에서 특정 단어를 찾을 경우 RET\_PASS 리턴  RX Data에서 특정 단어를 찾지 못할 경우 RET\_PASS 리턴 |

## ul\_uart\_interrupt\_send()

|  |  |
| --- | --- |
| Function | int ul\_uart\_interrupt\_send(char \*command, int dutid) |
| Description | Uart Tx Data를 전송한다 |
| Argument | 1. char \*command : Uart Tx data 입력 (Uart command) 2. int dutid : Dut ID |
| Return | Uart Tx Data 전송 과정에서 Error 발생 시 RET\_FAIL 리턴 (RET\_PASS, RET\_FAIL) |

## ul\_uart\_display\_rxbuffer()

|  |  |
| --- | --- |
| Function | int ul\_uart\_display\_rxbuffer(int dutid) |
| Description | Uart Rx Buffer에 저장된 내용을 PMS 또는 Uart Viewer GUI에 display하고  Uart Rx Buffer를 초기화 한다.  \* uni91k\TestResult 폴더 안에 DUTXX.rst 그리고 UARTLOG\_DUTXX.rst 이름으로 저장 |
| Argument | 1. int dutid : Dut ID |
| Return | RET\_PASS, RET\_FAIL |

## ul\_uart\_set\_baudrate()

|  |  |
| --- | --- |
| Function | int ul\_uart\_set\_baudrate(unsigned int baudrate, int dutid) |
| Description | Uart의 Baudrate 값들을 설정한다. (115200, 38400,9600) |
| Argument | 1. unsigned int baudrate : Uart의 baudrate 값을 설정 2. int dutid : Dut ID |
| Return | RET\_PASS, RET\_FAIL |

## ul\_uart\_realtime\_log()

|  |  |
| --- | --- |
| Function | int ul\_uart\_realtime\_log(int dutid) |
| Description | Uart Rx Buffer에 저장된 내용을 Uart Viewer GUI에 display하고  Uart Rx Buffer를 초기화 한다.  \* uni91k\TestResult 폴더 안에 UARTLOG\_DUTXX.rst 이름으로 저장 |
| Argument | 1. int dutid : Dut ID |
| Return | RET\_PASS, RET\_FAIL |

## ul\_uart\_realtime\_enable()

|  |  |
| --- | --- |
| Function | int ul\_uart\_realtime\_enable(unsigned char enable, int dutid) |
| Description | Real-time UART 기능 Enable/Disable 설정 |
| Argument | 1. unsigned char enable : Real-time UART 기능 ENABLE/DISABLE 설정 (true/false) 2. int dutid : Dut ID |
| Return | RET\_PASS, RET\_FAIL |

## ul\_get\_firmwareinfo ()

|  |  |
| --- | --- |
| Function | int ul\_get\_firmwareinfo() |
| Description | Sodimm(DRAM)에 저장된 Device Firmware 정보를 저장하고 UART TX,RX buffer를 할당한다. |
| Argument | NONE |
| Return | NONE |

// Vendor Spec

## ul\_serial\_flash\_tx()

|  |  |
| --- | --- |
| Function | int ul\_serial\_flash\_tx(int imageinfo\_index, u32 dutid) |
| Description | Sodimm(DRAM)에 저장된 Device Firmware binary를 읽어서 Uart Tx로 전송 |
| Argument | 1. int imageinfo : Sodimm에 저장된 Device Firmware image를 선택   SERIALFLASH\_UARTBOOT : Bootloader image  SERIALFLASH\_SFBOOT : Main firmware image   1. int dutid : Dut ID |
| Return | Firmware binary 전송 결과를 리턴 (RET\_PASS, RET\_FAIL) |

## ul\_serial\_fwimage()

|  |  |
| --- | --- |
| Function | int ul\_serial\_fwimage(int imageinfo\_index, char \*imagename, unsigned int dutid) |
| Description | Sodimm(DRAM)에 저장된 Device Firmware binary를 읽어서 Uart Tx 전송  ( Firmware image header에서 image name을 비교) |
| Argument | 1. int imageinfo : Sodimm에 저장된 Device Firmware image를 선택   SERIALFLASH\_UARTBOOT : Bootloader image  SERIALFLASH\_SFBOOT : Main firmware image   1. char \*imagename : Device Firmware binary name 2. int dutid : Dut ID |
| Return | Firmware binary 전송 결과를 리턴 (RET\_PASS, RET\_FAIL) |

# UFS API Function

## ul\_ufs\_writeHCIregister()

|  |  |
| --- | --- |
| Function | void ul\_ufs\_writeHCIregister(int dutid, unsigned int hci\_regoffset, unsigned int writevalue) |
| Description | UFS HCI Register에 Data를 Write하기 위해 사용된다. |
| Argument | 1. int dutid : Dut ID 2. unsigned int hci\_regoffset : HCI Register offset 3. unsigned int writevalue : Write Data |
| Return | NONE |

## ul\_ufs\_readHCIregister()

|  |  |
| --- | --- |
| Function | unsigned int ul\_ufs\_readHCIregister(int dutid, unsigned int hci\_regoffset) |
| Description | UFS HCI Register에 설정된 Data를 Read하기 위해 사용된다. |
| Argument | 1. int dutid : Dut ID 2. unsigned int hci\_regoffset : HCI Register offset |
| Return | UFS HCI에 설정된 Data를 리턴한다. |

## ul\_ufs\_get\_failmem\_address()

|  |  |
| --- | --- |
| Function | unsigned char \*ul\_ufs\_get\_failmem\_address(int dutid) |
| Description | 지정 DUT의 Fail Memory Start Address를 얻기 위해 사용된다. |
| Argument | 1. int dutid : Dut ID |
| Return | 지정한 DUT의 Fail Memory Start Address |

## ul\_ufs\_issue\_requestlist()

|  |  |
| --- | --- |
| Function | unsigned int ul\_ufs\_issue\_requestlist(int dutid, enum doorbell\_bit list\_bit) |
| Description | Transfer Request List를 Run 시키기 위해 사용된다. |
| Argument | 1. int dutid : Dut ID 2. enum doorbel\_bit list\_bit : Door Bell Register Bit 선택   DRB00=0x1<<0, DRB01=0x1<<1, DRB02=0x1<<2, DRB03=0x1<<3,  ...  DRB28=0x1<<28, DRB29=0x1<<29, DRB30=0x1<<30, DRB31=0x1<<31 |
| Return | UFS Host Controller가 Not Ready 시 RET\_FAIL을 리턴한다. (RET\_FAIL/RET\_PASS) |

## ul\_ufs\_polling\_requestlist()

|  |  |
| --- | --- |
| Function | unsigned int ul\_ufs\_polling\_requestlist(int dutid, unsigned int polling\_timeout\_msec,  enum doorbell\_bit list\_bit) |
| Description | Transfer Request List의 Complete 신호의 Pass/Fail 여부를 얻을 시 사용된다. |
| Argument | 1. int dutid : Dut ID 2. unsigned int polling\_timeout\_msec : Complete Interrupt Waiting Time 3. enum doorbel\_bit list\_bit : Door Bell Register Bit 선택   DRB00=0x1<<0, DRB01=0x1<<1, DRB02=0x1<<2, DRB03=0x1<<3,  ...  DRB28=0x1<<28, DRB29=0x1<<29, DRB30=0x1<<30, DRB31=0x1<<31 |
| Return | RET\_PASS(0) : Request List Complete Pass  RET\_TIMEOUT(-3) : Request List Complete Fail |

## ul\_ufs\_check\_response\_upiu()

|  |  |
| --- | --- |
| Function | enum requestlist\_complete ul\_ufs\_check\_response\_upiu(int dutid, int reqlist\_index) |
| Description | Transfer Request List의 Response의 Pass/Fail 여부를 얻을 시 사용된다. |
| Argument | 1. int dutid : Dut ID 2. int reqlist\_index : Slot Number |
| Return | REQ\_PASS(0) : request List Pass  REQ\_OSC\_ERROR(2) : OCS 값이 Success(0x0)이 아닐 경우 발생  REQ\_RSP\_TYPEERROR(3) : Response RX Transaction Type이 비정상적일 경우 발생  REQ\_QRY\_RSPEEOR(4) : Query를 Issue할 경우 Response가 Success(0x0)이 아닐 경우 발생  REQ\_CMD\_STATUSEEEOR(5) : SCSI Command를 Issue할 경우 Response가 Status Good(0x0)이 아닐 경우 발생 |

|  |
| --- |
| **Example Code - UFS SCSI Pseudo Code** |
| enum requestlist\_complete ireq\_ret = REQ\_PASS;  // Write10 Operation을 설정  ul\_scsi\_build\_write10 (dut\_control, 0x0, 0x0, 0x0, 0x0, 0x0, itrlength, iblksize, buffer, SLOT00);  // Transfer Request List Run  iret = ul\_ufs\_issue\_requestlist(dut\_control, DRB00);  // Check Request List Complete  iret = ul\_ufs\_polling\_requestlist(dut\_control, polling\_timeout\_msec, DRB00);  if (iret == RET\_PASS)  {  // Check Response UPIU  ireq\_ret = ul\_ufs\_check\_response\_upiu(dut\_control, icnt);  if (ireq\_ret != REQ\_PASS)  {  ul\_host\_printf(dut\_control, "Complete Request List = FAIL (%d) \n", ireq\_ret);  // Decode OCS Value ( User Function )  user\_decode\_ocs\_requestlist(dut\_control, SLOT00);  ierror\_code = ERR\_PHYREADY;  goto FAIL;  }  // Decode Response ( User Function )  user\_decode\_response(dut\_control, 0, iblksize, SLOT00);  } |

## ul\_ufs\_check\_requestlist()

|  |  |
| --- | --- |
| Function | enum requestlist\_complete ul\_ufs\_check\_requestlist(int dutid, unsigned int polling\_timeout\_msec,  int reqlist\_index) |
| Description | Transfer Request List의 Complete과 Response의 Pass/Fail 여부를 얻을 시 사용된다. |
| Argument | 1. int dutid : Dut ID 2. unsigned int polling\_timeout\_msec : Complete Interrupt Waiting Time 3. int reqlist\_index : Slot Number |
| Return | REQ\_PASS(0) : request List Pass  REQ\_TIMEOUT(1) : Complete Interrupt Flag가 On이 되지 않을 시 발생.  REQ\_OSC\_ERROR(2) : OCS 값이 Success(0x0)이 아닐 경우 발생  REQ\_RSP\_TYPEERROR(3) : Response RX Transaction Type이 비정상적일 경우 발생  REQ\_QRY\_RSPEEOR(4) : Query를 Issue할 경우 Response가 Success(0x0)이 아닐 경우 발생  REQ\_CMD\_STATUSEEEOR(5) : SCSI Command를 Issue할 경우 Response가 Status Good(0x0)이 아닐 경우 발생 |

|  |
| --- |
| **Example Code - UFS SCSI Pseudo Code** |
| enum requestlist\_complete ireq\_ret = REQ\_PASS;  // Write10 Operation을 설정  ul\_scsi\_build\_write10 (dut\_control, 0x0, 0x0, 0x0, 0x0, 0x0, itrlength, iblksize, buffer, SLOT00);  // Transfer Request List Run  iret = ul\_ufs\_issue\_requestlist(dut\_control, DRB00);  // Check Request List Response  ireq\_ret = ul\_ufs\_check\_requestlist(dut\_control, polling\_timeout\_msec, SLOT00);  if (ireq\_ret != REQ\_PASS)  {  ul\_host\_printf(dut\_control, "Complete Request List = FAIL (%d) \n", ireq\_ret);  // Decode OCS Value ( User Function )  user\_decode\_ocs\_requestlist(dut\_control, SLOT00);  ierror\_code = ERR\_PHYREADY;  goto FAIL;  }  // Decode Response ( User Function )  user\_decode\_response(dut\_control, 0, iblksize, SLOT00); |

## ul\_ufs\_get\_prdtaddress()

|  |  |
| --- | --- |
| Function | unsigned char \*ul\_ufs\_get\_prdtaddress(int dutid, int reqlist\_index) |
| Description | 지정 SLOT의 PRDT Start Address를 얻기 위해 사용된다. |
| Argument | 1. int dutid : Dut ID 2. int reqlist\_index : Slot Number |
| Return | 지정한 SLOT의 PRDT Start Address |

## ul\_ufs\_build\_nop\_upiu()

|  |  |
| --- | --- |
| Function | void ul\_ufs\_build\_nop\_upiu(int dutid, int reqlist\_index) |
| Description | NOP OUT UPIU를 이용 시 사용되며, Command UPIU 를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_ufs\_build\_nop\_requestlist()

|  |  |
| --- | --- |
| Function | void ul\_ufs\_build\_nop\_requestlist(int dutid, int reqlist\_index) |
| Description | NOP OUT UPIU를 이용 시 사용되며, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. int reqlist\_index : Slot Number |
| Return | NONE |

|  |
| --- |
| **Example Code - UFS NOP OUT/IN Pseudo Code** |
| enum requestlist\_complete ireq\_ret = REQ\_PASS;  // NOP OUT UPIU Operation을 설정  ul\_ufs\_build\_nop\_upiu(dut\_control, SLOT00);  // NOP OUT Transfer Request List을 설정  iret = ul\_ufs\_build\_nop\_requestlist (dut\_control, DRB00);  // Check Request List Response  ireq\_ret = ul\_ufs\_check\_requestlist(dut\_control, polling\_timeout\_msec, SLOT00);  if (ireq\_ret != REQ\_PASS)  {  ul\_host\_printf(dut\_control, "Complete Request List = FAIL (%d) \n", ireq\_ret);  // Decode OCS Value ( User Function )  user\_decode\_ocs\_requestlist(dut\_control, SLOT00);  ierror\_code = ERR\_PHYREADY;  goto FAIL;  }  // Decode Response ( User Function )  user\_decode\_response(dut\_control, 0, 0, SLOT00); |
|  |

## ul\_scsi\_build\_inquiry()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_inquiry(int dutid, char ilun, char epvd, char page\_code, unsigned int ilength,  int reqlist\_index) |
| Description | SCSI Command 중 Inquiry Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char epvd : EPVD 4. char page\_code : Page Code 5. unsigned int ilength : Allocation Length ( 내부에서 UPIU Length는 36byte 고정 ) 6. int reqlist\_index : Slot Number |
| Return | NONE |

|  |
| --- |
| **Example Code - UFS SCSI Inquiry Pseudo Code** |
| enum requestlist\_complete ireq\_ret = REQ\_PASS;  // Inquiry Operation을 설정  ul\_scsi\_build\_inquiry(dut\_control, lun, epvd, page\_code, ilength, SLOT00);  // Transfer Request List Run  iret = ul\_ufs\_issue\_requestlist(dut\_control, DRB00);  // Check Request List Response  ireq\_ret = ul\_ufs\_check\_requestlist(dut\_control, polling\_timeout\_msec, SLOT00);  if (ireq\_ret != REQ\_PASS)  {  ul\_host\_printf(dut\_control, "Complete Request List = FAIL (%d) \n", ireq\_ret);  // Decode OCS Value ( User Function )  user\_decode\_ocs\_requestlist(dut\_control, SLOT00);  ierror\_code = ERR\_PHYREADY;  goto FAIL;  }  // Decode Response ( User Function )  user\_decode\_response(dut\_control, 1, ilength, SLOT00); |

## ul\_scsi\_build\_modeselect10()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_modeselect10(int dutid, char ilun, char page\_format, char save\_page,  unsigned int ilength, char \*buffer, int reqlist\_index) |
| Description | SCSI Command 중 Mode Select10 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char page\_format : Page Format 4. char save\_page : Save Page 5. unsigned int ilength : Allocation Length 6. char \*buffer : Write Data 7. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_modesense10()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_modesense10(int dutid, char ilun, char page\_code, char page\_ctrl, char subpage\_code,  unsigned int ilength, int reqlist\_index) |
| Description | SCSI Command 중 Mode Sense10 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char page\_code : Page Code 4. char page\_ctrl : Page Control 5. char subpage\_code : Subpage Code 6. unsigned int ilength : Allocation Length ( 내부에서 UPIU Length는 16byte 고정 ) 7. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_read6()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_read6(int dutid, char ilun, unsigned int ilba, unsigned int trlength,  unsigned int blksize, char flag\_attr, int reqlist\_index) |
| Description | SCSI Command 중 Read6 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. unsigned int ilba : Logical Block Address 4. unsigned int trlength : Transfer Length 5. unsigned int blksize : Block Size 6. char flag\_attr : Command Flags.ATTR   CMD\_FLAG\_ATTR\_SIMPLE 0x00  CMD\_FLAG\_ATTR\_ORDERED 0x01  CMD\_FLAG\_ATTR\_HEAD\_OF\_QUEUE 0x02  CMD\_FLAG\_ATTR\_ACA 0x03   1. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_read10()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_read10(int dutid, char ilun, unsigned int ilba, char fua, char dpo, char grp\_num,  unsigned int trlength, unsigned int blksize, char flag\_attr, int reqlist\_index) |
| Description | SCSI Command 중 Read10 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. unsigned int ilba : Logical Block Address 4. char fua : FUA 5. char dpo : DPO 6. char grp\_num : Group Number 7. unsigned int trlength : Transfer Length 8. unsigned int blksize : Block Size 9. char flag\_attr : Command Flags.ATTR   CMD\_FLAG\_ATTR\_SIMPLE 0x00  CMD\_FLAG\_ATTR\_ORDERED 0x01  CMD\_FLAG\_ATTR\_HEAD\_OF\_QUEUE 0x02  CMD\_FLAG\_ATTR\_ACA 0x03   1. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_read16()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_read16(int dutid, char ilun, unsigned long long ilba, char fua, char dpo, char grp\_num,  unsigned int trlength, unsigned int blksize, char flag\_attr, int reqlist\_index) |
| Description | SCSI Command 중 Read16 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. unsigned long long ilba : Logical Block Address 4. char fua : FUA 5. char dpo : DPO 6. char grp\_num : Group Number 7. unsigned int trlength : Transfer Length 8. unsigned int blksize : Block Size 9. char flag\_attr : Command Flags.ATTR   CMD\_FLAG\_ATTR\_SIMPLE 0x00  CMD\_FLAG\_ATTR\_ORDERED 0x01  CMD\_FLAG\_ATTR\_HEAD\_OF\_QUEUE 0x02  CMD\_FLAG\_ATTR\_ACA 0x03   1. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_readcapacity10()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_readcapacity10(int dutid, char ilun, unsigned int ilength, int reqlist\_index) |
| Description | SCSI Command 중 Read Capacity10 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. unsigned int ilength : Allocation Length 4. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_readcapacity16()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_readcapacity16(int dutid, char ilun, unsigned int ilength, int reqlist\_index) |
| Description | SCSI Command 중 Read Capacity16 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. unsigned int ilength : Allocation Length 4. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_startstopunit()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_startstopunit(int dutid, char ilun, char immed, char start, char no\_flush, char pwr\_cond,  int reqlist\_index) |
| Description | SCSI Command 중 Start Stop Unit Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char immed : IMMED 4. char start : START 5. char no\_flush : NO\_FLUSH 6. char pwr\_cond : Power condition 7. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_testunitready()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_testunitready(int dutid, char ilun, int reqlist\_index) |
| Description | SCSI Command 중 Test Unit Ready Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_reportlun()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_reportlun(int dutid, char ilun, char select\_report, unsigned int ilength, int reqlist\_index) |
| Description | SCSI Command 중 Report LUN Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char select\_report : Select Report 4. unsigned int ilength : Allocation Length 5. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_verify10()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_verify10(int dutid, char ilun, unsigned int lba, unsigned int ilength, int reqlist\_index) |
| Description | SCSI Command 중 Verify10 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. unsigned int lba : Logical Block Address 4. unsigned int ilength : Allocation Length 5. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_write6()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_write6(int dutid, char ilun, unsigned int ilba, unsigned int trlength,  unsigned int blksize, char flag\_attr, int reqlist\_index) |
| Description | SCSI Command 중 Write6 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. unsigned int ilba : Logical Block Address 4. unsigned int trlength : Transfer Length 5. unsigend int blksize : Block Size 6. char flag\_attr : Command Flags.ATTR   CMD\_FLAG\_ATTR\_SIMPLE 0x00  CMD\_FLAG\_ATTR\_ORDERED 0x01  CMD\_FLAG\_ATTR\_HEAD\_OF\_QUEUE 0x02  CMD\_FLAG\_ATTR\_ACA 0x03   1. int reqlist\_index : Slot Number |
| Return | NONE |

|  |
| --- |
| **Example Code - UFS SCSI Write6 Pseudo Code** |
| enum requestlist\_complete ireq\_ret = REQ\_PASS;  unsigned int itrlength = 1;  unsigned int iblksize = 4096;  unsigned char \*prdtaddr = 0;  // 지정 SLOT의 PRDT Address Get 및 Data 입력  prdtaddr = ul\_ufs\_get\_prdtaddress(dut\_control, SLOT00);  memset( prdtaddr, 0x30, itrlength \* iblksize );  // Write6 Operation을 설정  ul\_scsi\_build\_write6(dut\_control, 0x0, 0x0, itrlength, iblksize, CMD\_FLAG\_ATTR\_SIMPLE, SLOT00);  // Transfer Request List Run  iret = ul\_ufs\_issue\_requestlist(dut\_control, DRB00);  // Check Request List Response  ireq\_ret = ul\_ufs\_check\_requestlist(dut\_control, polling\_timeout\_msec, SLOT00);  if (ireq\_ret != REQ\_PASS)  {  ul\_host\_printf(dut\_control, "Complete Request List = FAIL (%d) \n", ireq\_ret);  // Decode OCS Value ( User Function )  user\_decode\_ocs\_requestlist(dut\_control, SLOT00);  ierror\_code = ERR\_PHYREADY;  goto FAIL;  }  // Decode Response ( User Function )  user\_decode\_response(dut\_control, 0, itrlength \* iblksize, SLOT00); |

## ul\_scsi\_build\_write10()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_write10(int dutid, char ilun, unsigned int ilba, char fua, char dpo, char grp\_num,  unsigned int trlength, unsigned int blksize, char flag\_attr, int reqlist\_index) |
| Description | SCSI Command 중 Write10 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. unsigned int ilba : Logical Block Address 4. char fua : FUA 5. char dpo : DPO 6. char grp\_num : Group Number 7. unsigned int trlength : Transfer Length 8. unsigned int blksize : Block Size 9. char flag\_attr : Command Flags.ATTR   CMD\_FLAG\_ATTR\_SIMPLE 0x00  CMD\_FLAG\_ATTR\_ORDERED 0x01  CMD\_FLAG\_ATTR\_HEAD\_OF\_QUEUE 0x02  CMD\_FLAG\_ATTR\_ACA 0x03   1. int reqlist\_index : Slot Number |
| Return | NONE |

|  |
| --- |
| **Example Code - UFS SCSI Write10 Pseudo Code** |
| enum requestlist\_complete ireq\_ret = REQ\_PASS;  unsigned int itrlength = 1;  unsigned int iblksize = 4096;  unsigned char \*prdtaddr = 0;  // 지정 SLOT의 PRDT Address Get 및 Data 입력  prdtaddr = ul\_ufs\_get\_prdtaddress(dut\_control, SLOT00);  memset( prdtaddr, 0x30, itrlength \* iblksize );  // Write10 Operation을 설정  ul\_scsi\_build\_write10\_test(dut\_control, 0x0, 0x0, 0x0, 0x0, 0x0, itrlength, iblksize, CMD\_FLAG\_ATTR\_SIMPLE, SLOT00);  // Transfer Request List Run  iret = ul\_ufs\_issue\_requestlist(dut\_control, DRB00);  // Check Request List Response  ireq\_ret = ul\_ufs\_check\_requestlist(dut\_control, polling\_timeout\_msec, SLOT00);  if (ireq\_ret != REQ\_PASS)  {  ul\_host\_printf(dut\_control, "Complete Request List = FAIL (%d) \n", ireq\_ret);  // Decode OCS Value ( User Function )  user\_decode\_ocs\_requestlist(dut\_control, SLOT00);  ierror\_code = ERR\_PHYREADY;  goto FAIL;  }  // Decode Response ( User Function )  user\_decode\_response(dut\_control, 0, itrlength \* iblksize, SLOT00); |

## ul\_scsi\_build\_write16()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_write16(int dutid, char ilun, unsigned long long ilba, char fua, char dpo, char grp\_num,  unsigned int trlength, unsigned int blksize, char flag\_attr, int reqlist\_index) |
| Description | SCSI Command 중 Write16 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. unsigned long long ilba : Logical Block Address 4. char fua : FUA 5. char dpo : DPO 6. char grp\_num : Group Number 7. unsigned int trlength : Transfer Length 8. unsigned int blksize : Block Size 9. char flag\_attr : Command Flags.ATTR   CMD\_FLAG\_ATTR\_SIMPLE 0x00  CMD\_FLAG\_ATTR\_ORDERED 0x01  CMD\_FLAG\_ATTR\_HEAD\_OF\_QUEUE 0x02  CMD\_FLAG\_ATTR\_ACA 0x03   1. int reqlist\_index : Slot Number |
| Return | NONE |

|  |
| --- |
| **Example Code - UFS SCSI Write16 Pseudo Code** |
| enum requestlist\_complete ireq\_ret = REQ\_PASS;  unsigned int itrlength = 1;  unsigned int iblksize = 4096;  unsigned char \*prdtaddr = 0;  // 지정 SLOT의 PRDT Address Get 및 Data 입력  prdtaddr = ul\_ufs\_get\_prdtaddress(dut\_control, SLOT00);  memset( prdtaddr, 0x30, itrlength \* iblksize );  // Write16 Operation을 설정  ul\_scsi\_build\_write16\_test(dut\_control, 0x0, 0x0, 0x0, 0x0, 0x0, itrlength, iblksize, CMD\_FLAG\_ATTR\_SIMPLE, SLOT00);  // Transfer Request List Run  iret = ul\_ufs\_issue\_requestlist(dut\_control, DRB00);  // Check Request List Response  ireq\_ret = ul\_ufs\_check\_requestlist(dut\_control, polling\_timeout\_msec, SLOT00);  if (ireq\_ret != REQ\_PASS)  {  ul\_host\_printf(dut\_control, "Complete Request List = FAIL (%d) \n", ireq\_ret);  // Decode OCS Value ( User Function )  user\_decode\_ocs\_requestlist(dut\_control, SLOT00);  ierror\_code = ERR\_PHYREADY;  goto FAIL;  }  // Decode Response ( User Function )  user\_decode\_response(dut\_control, 0, itrlength \* iblksize, SLOT00); |

## ul\_scsi\_build\_requestsense()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_requestsense(int dutid, char ilun, unsigned int ilength, int reqlist\_index) |
| Description | SCSI Command 중 Request Sense Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. unsigned int ilength : Allocation Length 4. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_formatunit()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_formatunit(int dutid, char ilun, char cmplst, char longlist, unsigned int obsolete,  int reqlist\_index) |
| Description | SCSI Command 중 Format Unit Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char cmplst : CMPLST 4. char longlist : LONGLIST 5. unsigned int obsolete : OBSOLETE 6. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_prefetch10()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_prefetch10(int dutid, char ilun, char immed, unsigned int lba, unsigned int ilength,  int reqlist\_index) |
| Description | SCSI Command 중 Prefetch10 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char immed : IMMED 4. unsigned int lba : Logical Block Address 5. unsigned int ilength : Prefetch Length 6. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_prefetch16()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_prefetch16(int dutid, char ilun, char immed, unsigned long long lba,  unsigned int ilength, int reqlist\_index) |
| Description | SCSI Command 중 Prefetch16 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char immed : IMMED 4. unsigned long long lba : Logical Block Address 5. unsigned int ilength : Prefetch Length 6. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_securityprotocolin()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_securityprotocolin(int dutid, char ilun, char security\_prot,  unsigned int security\_prot\_spec, char inc\_512, unsigned int ilength, int reqlist\_index) |
| Description | RPMB Data를 얻기 위해 사용되고, Security Protocol In Format과 Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char security\_prot : Security Protocol 4. unsigned int security\_prot\_spec : Security Protocol Specific 5. char inc\_512 : INC\_512 6. unsigned int ilength : Allocation Length 7. int reqlist\_index : Slot Number |
| Return | NONE |

|  |
| --- |
| **Example Code - UFS SCSI Security Protocol In Pseudo Code** |
| enum requestlist\_complete ireq\_ret = REQ\_PASS;  unsigned char \*prdtaddr = 0;  // Security Protocol In Operation을 설정  ul\_scsi\_build\_securityprotocolin(dut\_control, 0xC4, 0xEC, 0x0001, 0x0, RPMB\_FRAME\_LENGTH, SLOT00);  // Transfer Request List Run  iret = ul\_ufs\_issue\_requestlist(dut\_control, DRB00);  // Check Request List Response  ireq\_ret = ul\_ufs\_check\_requestlist(dut\_control, polling\_timeout\_msec, SLOT00);  if (ireq\_ret != REQ\_PASS)  {  ul\_host\_printf(dut\_control, "Complete Request List = FAIL (%d) \n", ireq\_ret);  // Decode OCS Value ( User Function )  user\_decode\_ocs\_requestlist(dut\_control, SLOT00);  ierror\_code = ERR\_PHYREADY;  goto FAIL;  }  // Decode Response ( User Function )  user\_decode\_response(dut\_control, 1, RPMB\_FRAME\_LENGTH, SLOT00);  prdtaddr = ul\_ufs\_get\_prdtaddress(dut\_control, SLOT00);  // Display Security In Data ( User Function )  user\_display\_rpmb\_data( dut\_control, prdtaddr ); |

## ul\_scsi\_build\_securityprotocolout()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_securityprotocolout(int dutid, char ilun, char security\_prot,  unsigned int security\_prot\_spec, char inc\_512, unsigned int ilength, struct ufs\_rpmb\_data \*rpmb\_data,  int reqlist\_index) |
| Description | RPMB Data Out 시 사용되고, Security Protocol Out Format과 Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char security\_prot : Security Protocol 4. unsigned int security\_prot\_spec : Security Protocol Specific 5. char inc\_512 : INC\_512 6. unsigned int ilength : Transfer Length 7. struct ufs\_rpmb\_data \*rpmb\_data : RPMB Message Data Frame structure 8. int reqlist\_index : Slot Number |
| Return | NONE |

|  |
| --- |
| **Example Code - UFS SCSI Security Protocol Out Pseudo Code** |
| enum requestlist\_complete ireq\_ret = REQ\_PASS;  struct ufs\_rpmb\_data rpmb\_data;  // RPMB Message Data Frame structure 설정  memset( &rpmb\_data, 0, sizeof(struct ufs\_rpmb\_data) );  // Nonce  rpmb\_data.sznonce[0] = 0x0A; rpmb\_data.sznonce[1] = 0x1B;  ......  rpmb\_data.sznonce[14] = 0xE8; rpmb\_data.sznonce[15] = 0xF9;  // Block Count  rpmb\_data.iblkcount = 0x0001;  // Request/Response  rpmb\_data.ireqresp = AUTH\_DAT\_RD\_REQ;  // Security Protocol Out Operation을 설정  ul\_scsi\_build\_securityprotocolout(dut\_control, 0xC4, 0xEC, 0x0001, 0x0, RPMB\_FRAME\_LENGTH, &rpmb\_data, SLOT00);  // Transfer Request List Run  iret = ul\_ufs\_issue\_requestlist(dut\_control, DRB00);  // Check Request List Response  ireq\_ret = ul\_ufs\_check\_requestlist(dut\_control, polling\_timeout\_msec, SLOT00);  if (ireq\_ret != REQ\_PASS)  {  ul\_host\_printf(dut\_control, "Complete Request List = FAIL (%d) \n", ireq\_ret);  // Decode OCS Value ( User Function )  user\_decode\_ocs\_requestlist(dut\_control, SLOT00);  ierror\_code = ERR\_PHYREADY;  goto FAIL;  }  // Decode Response ( User Function )  user\_decode\_response(dut\_control, 1, RPMB\_FRAME\_LENGTH, SLOT00); |

## ul\_scsi\_build\_senddiagnostic()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_senddiagnostic(int dutid, char ilun, char unitoffl, char devoffl, char selftest,  char page\_format, char self\_test\_code, unsigned int ilength, int reqlist\_index) |
| Description | SCSI Command 중 Send Diagnostic Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char unitoffl : UNITOFFL 4. char devoffl : DEVOFFL 5. char selftest : SELFTEST 6. char page\_format : Page Format 7. char self\_test\_code : SELF-TEST CODE 8. unsigned int ilength : Parameter List Length 9. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_synchronizecache10()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_synchronizecache10(int dutid, char ilun, char immed, char sync\_nv, unsigned int lba,  unsigned int lb\_num, int reqlist\_index) |
| Description | SCSI Command 중 Synchronize Cache10 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char immed : IMMED 4. char sync\_nv : SYNC\_NV 5. unsigned int lba : Logical Block Address 6. unsigned int lb\_num : Number of Logical Blocks 7. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_synchronizecache16()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_synchronizecache16(int dutid, char ilun, char immed, char sync\_nv,  unsigned long long lba, unsigned int lb\_num, int reqlist\_index) |
| Description | SCSI Command 중 Synchronize Cache16 Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char immed : IMMED 4. char sync\_nv : SYNC\_NV 5. unsigned long long lba : Logical Block Address 6. unsigned int lb\_num : Number of Logical Blocks 7. int reqlist\_index : Slot Number |
|  | NONE |

## ul\_scsi\_build\_unmap()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_unmap(int dutid, char ilun, unsigned char anchor, unsigned int iparam\_len,  unsigned int idata\_len, unsigned int iblk\_desc\_data\_len, struct ufs\_unmap\_blk\_desc \*unmap\_blk\_desc,  int reqlist\_index) |
| Description | SCSI Command 중 Unmap Command 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. unsigned char anchor : ANCHOR 4. unsigned int iparam\_len : Parameter List Length 5. unsigned int idata\_len : Data Length 6. unsigned int iblk\_desc\_data\_len : Block Descriptor Data Length 7. struct ufs\_unmap\_blk\_desc \*unmap\_blk\_desc : Block Descriptor structure 8. int reqlist\_index : Slot Number |
|  | NONE |

|  |
| --- |
| **Example Code - UFS SCSI Unmap Pseudo Code** |
| enum requestlist\_complete ireq\_ret = REQ\_PASS;  struct ufs\_unmap\_blk\_desc unmap\_blk\_desc[1];  iparam\_len = 0x18;  idata\_len = 0x16;  iblk\_desc\_data\_len = 0x10;  // Unmap Block Descriptor Data structure 설정  memset( &unmap\_blk\_desc, 0, sizeof(unmap\_blk\_desc) );  unmap\_blk\_desc[0].ilba = imaxlba;  unmap\_blk\_desc[0].inumoflb = 0x02;  unmap\_blk\_desc[0].ireserved = 0;  // Unmap Operation을 설정  ul\_scsi\_build\_unmap(dut\_control, 0, 0x0, iparam\_len, idata\_len, iblk\_desc\_data\_len, unmap\_blk\_desc, SLOT00);  // Transfer Request List Run  iret = ul\_ufs\_issue\_requestlist(dut\_control, DRB00);  // Check Request List Response  ireq\_ret = ul\_ufs\_check\_requestlist(dut\_control, polling\_timeout\_msec, SLOT00);  if (ireq\_ret != REQ\_PASS)  {  ul\_host\_printf(dut\_control, "Complete Request List = FAIL (%d) \n", ireq\_ret);  // Decode OCS Value ( User Function )  user\_decode\_ocs\_requestlist(dut\_control, SLOT00);  ierror\_code = ERR\_PHYREADY;  goto FAIL;  }  // Decode Response ( User Function )  user\_decode\_response(dut\_control, 0, iparam\_len, SLOT00); |

## ul\_scsi\_build\_readbuffer()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_readbuffer(int dutid, char ilun, char mode, char buffer\_id, char buffer\_offset,  unsigned int ilength, int reqlist\_index) |
| Description | SCSI Command 중 Read Buffer 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char mode : Mode 4. char buffer\_id : Buffer ID 5. char buffer\_offset : Buffer Offset 6. unsigned int ilength : Allocation Length 7. int reqlist\_index : Slot Number |
|  | NONE |

## ul\_scsi\_build\_writebuffer()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_writebuffer(int dutid, char ilun, char mode, char buffer\_id, char buffer\_offset,  unsigned int ilength, char \*buffer, int reqlist\_index) |
| Description | SCSI Command 중 Write Buffer 이용시 사용되며, Command UPIU, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char mode : Mode 4. char buffer\_id : Buffer ID 5. char buffer\_offset : Buffer Offset 6. unsigned int ilength : Allocation Length 7. char \*buffer : Write Data 8. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_scsi\_build\_vender\_cmd()

|  |  |
| --- | --- |
| Function | void ul\_scsi\_build\_vender\_cmd(int dutid, char ilun, unsigned char \*cdb, int datadir, unsigned int ilength, int reqlist\_index) |
| Description | SCSI Command를 사용자가 직접 사용할 경우 사용한다. |
| Argument | 1. int dutid : Dut ID 2. char ilun : Logical Unit Number 3. char \*cdb : CDB Data 4. int datadir : Data Direction 5. unsigned int ilength : Allocation Length 6. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_ufs\_convert\_msb\_lsb\_char2char()

|  |  |
| --- | --- |
| Function | void ul\_ufs\_convert\_msb\_lsb\_char2char( unsigned char \*szdest, unsigned char \*szsrc, int ilength ) |
| Description | char형의 데이터를 MSB / LSB 를 반전시켜 char형으로 데이터를 얻는다. |
| Argument | 1. unsigned char \*szdest : Destination Data 2. unsigned char \*szsrc : Source Data 3. int ilength : MSB / LSB 반전 시킬 Length |
| Return | NONE |

## ul\_ufs\_convert\_msb\_lsb\_int2char()

|  |  |
| --- | --- |
| Function | void ul\_ufs\_convert\_msb\_lsb\_int2char( unsigned char \*szdest, unsigned int isrc, int ilength ) |
| Description | int형의 데이터를 MSB / LSB 를 반전시켜 char형으로 데이터를 얻는다. |
| Argument | 1. unsigned char \*szdest : Destination Data 2. unsigned int isrc : Source Data 3. int ilength : MSB / LSB 반전 시킬 Length |
| Return | NONE |

## ul\_ufs\_convert\_msb\_lsb\_longlong2char()

|  |  |
| --- | --- |
| Function | void ul\_ufs\_convert\_msb\_lsb\_longlong2char( unsigned char \*szdest, unsigned long long isrc, int ilength ) |
| Description | long long형의 데이터를 MSB / LSB 를 반전시켜 char형으로 데이터를 얻는다. |
| Argument | 1. unsigned char \*szdest : Destination Data 2. unsigned long long isrc : Source Data 3. int ilength : MSB / LSB 반전 시킬 Length |
| Return | NONE |

## ul\_ufs\_convert\_msb\_lsb\_char2longlong()

|  |  |
| --- | --- |
| Function | void ul\_ufs\_convert\_msb\_lsb\_char2longlong( unsigned long long \*idest, unsigned char \*szsrc,  int ilength ) |
| Description | char형의 데이터를 MSB / LSB 를 반전시켜 long long형으로 데이터를 얻는다. |
| Argument | 1. unsigned long long \*idest : Destination Data 2. unsigned char \*szsrc : Source Data 3. int ilength : MSB / LSB 반전 시킬 Length |
| Return | NONE |

## ul\_ufs\_convert\_msb\_lsb\_char2int()

|  |  |
| --- | --- |
| Function | void ul\_ufs\_convert\_msb\_lsb\_char2int( unsigned int \*idest, unsigned char \*szsrc, int ilength ) |
| Description | char형의 데이터를 MSB / LSB 를 반전시켜 int형으로 데이터를 얻는다. |
| Argument | 1. unsigned int \*idest : Destination Data 2. unsigned char \*szsrc : Source Data 3. int ilength : MSB / LSB 반전 시킬 Length |
| Return | NONE |

## ul\_ufs\_get\_prdtaddress()

|  |  |
| --- | --- |
| Function | unsigned char \*ul\_ufs\_get\_prdtaddress(int dutid, int reqlist\_index) |
| Description | 각 Slot의 PRDT Address를 얻기 위해 사용된다. |
| Argument | 1. int dutid : Dut ID 2. int reqlist\_index : Slot Number |
| Return | NONE |

|  |
| --- |
| **Example Code - UFS SCSI Inquiry Pseudo Code** |
| enum requestlist\_complete ireq\_ret = REQ\_PASS;  unsigned char \*prdtaddr = 0;  unsigned int imaxlba = 0;  // Read Capacity10 Operation을 설정  ul\_scsi\_build\_readcapacity10(dut\_control, 0, 8, SLOT00);  // Transfer Request List Run  iret = ul\_ufs\_issue\_requestlist(dut\_control, DRB00);  // Check Request List Response  ireq\_ret = ul\_ufs\_check\_requestlist(dut\_control, polling\_timeout\_msec, SLOT00);  if (ireq\_ret != REQ\_PASS)  {  ul\_host\_printf(dut\_control, "Complete Request List = FAIL (%d) \n", ireq\_ret);  // Decode OCS Value ( User Function )  user\_decode\_ocs\_requestlist(dut\_control, SLOT00);  ierror\_code = ERR\_PHYREADY;  goto FAIL;  }  // Decode Response ( User Function )  user\_decode\_response(dut\_control, 1, 8, SLOT00);  // 원하는 Slot의 PRDT Address를 얻는다.  prdtaddr = ul\_ufs\_get\_prdtaddress(dut\_control, SLOT00);  // MSB / LSB Data를 반전 시킨다.  ul\_ufs\_convert\_msb\_lsb\_char2int( &imaxlba, prdtaddr, 4 );  ul\_host\_printf(dut\_control, "Max LBA : 0x%x\n", imaxlba); |

## ul\_ufs\_build\_query\_upiu()

|  |  |
| --- | --- |
| Function | void ul\_ufs\_build\_query\_upiu(int dutid, int qry\_func, char qry\_opcode, char osf0, char osf1, char osf2, char osf3, char osf4, short osf5, int osf6, int osf7, char \*buf, int reqlist\_index) |
| Description | Query Request UPIU에 사용되며, Query UPIU를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. int qry\_func : Query Function   QUERY\_NOP\_READ 0x00  QUERY\_STD\_RD\_REQ 0x01  QUERY\_VENDOR\_RD\_START 0x40  QUERY\_VENDOR\_RD\_END 0x7F  QUERY\_NOP\_WRITE 0x80  QUERY\_STD\_WR\_REQ 0x81  QUERY\_VENDOR\_WR\_START 0xC0  QUERY\_VENDOR\_WR\_END 0xFF   1. char qry\_opcode : OPCODE   QUERY\_NOP 0x00  QUERY\_RD\_DESC 0x01  QUERY\_WR\_DESC 0x02  QUERY\_RD\_ATTRB 0x03  QUERY\_WR\_ATTRB 0x04  QUERY\_RD\_FLAG 0x05  QUERY\_SET\_FLAG 0x06  QUERY\_CLR\_FLAG 0x07  QUERY\_TOGGLE\_FLAG 0x08   1. char osf0 : OSF[0] ( Descriptor / Attribute / Flag IDN )   [ Descriptor IDN ]  DEVICE\_DESC\_IDN 0x00  CONFIG\_DESC\_IDN 0x01  UNIT\_DESC\_IDN 0x02  RPMB\_DESC\_IDN 0x03  INTERCONNECT\_DESC\_IDN 0x04  STRING\_DESC\_IDN 0x05  GEOMETRY\_DESC\_IDN 0x07  POWER\_DESC\_IDN 0x08  DEVICE\_HEALTH\_IDN 0x09  [ Attribute IDN ]  AttribBootLunEn 0x00  AttribCurrentPowerMode\_RO 0x02  AttribActiveICCLevel 0x03  bOutOfOrderDataEn 0x04  AttribBackgroundOpStatus\_RO 0x05  AttribPurgeStatus\_RO 0x06  AttribMaxDataInSize 0x07  AttribMaxDataOutSize 0x08  AttridDynCapNeeded\_RO 0x09  bRefClkFreq 0x0A  bConfigDescrLock 0x0B  bMaxNumOfRTT 0x0C  bExceptionEventControl 0x0D  bExceptionEventStatus 0x0E  dSecondsPassed 0x0F  wContextConf 0x10  dCorrPrgBlkNum 0x11  bDeviceFFUStatus 0x14  bPSAState 0x15  dPSADataSize 0x16  [ Flag IDN ]  FlagfDeviceInit 0x01  PermanentWPEn 0x02  PowerOnWPEn 0x03  BACKGROUND\_OPS\_EN 0x04  PURGE\_EN 0x06  PURGE\_ERR 0x07  fPhyResourceRemoval 0x08  fBusyRTC 0x09   1. char osf1 : OSF[1] ( Index ) 2. char osf2 : OSF[2] ( Selector ) 3. char osf3 : OSF[3] ( Reserved ) 4. char osf4 : OSF[4] ( Reserved ) 5. char osf5 : OSF[5] ( Descriptor Length ) 6. char osf6 : OSF[6] ( Write Attribute Value ) 7. char osf7 : OSF[7] ( Reserved ) 8. char \*buf : Write Buffer ( Write Descriptor ) 9. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_ufs\_build\_query\_requestlist()

|  |  |
| --- | --- |
| Function | void ul\_ufs\_build\_query\_requestlist(int dutid, int datadir, int reqlist\_index) |
| Description | Query Request UPIU에 사용되며, UTP Trasnfer Request Descriptor를 생성한다. |
| Argument | 1. int dutid : Dut ID 2. int datadir : Data Direction   DIR\_NO\_DATA 0  DIR\_WRITE 1  DIR\_READ 2   1. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_ufs\_get\_query\_attr\_value()

|  |  |
| --- | --- |
| Function | unsigned int ul\_ufs\_get\_query\_attr\_value(int dutid, int reqlist\_index); |
| Description | Query Response의 Attribute Value(OSF[6])를 얻기 위해 사용된다. |
| Argument | 1. int dutid : Dut ID 2. int reqlist\_index : Slot Number |
| Return | NONE |

|  |
| --- |
| **Example Code - UFS Query Pseudo Code** |
| enum requestlist\_complete ireq\_ret = REQ\_PASS;  unsigned int igetval = 0;  // Query Operation을 설정  ul\_ufs\_build\_query\_upiu(dut\_control, QUERY\_STD\_RD\_REQ, QUERY\_RD\_ATTRB, bConfigDescrLock, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, NULL, SLOT00);  // UTP Transfer Request List Build  ul\_ufs\_build\_query\_requestlist(dut\_control, DIR\_READ, SLOT00);  // Transfer Request List Run  iret = ul\_ufs\_issue\_requestlist(dut\_control, DRB00);  // Check Request List Response  ireq\_ret = ul\_ufs\_check\_requestlist(dut\_control, polling\_timeout\_msec, SLOT00);  if (ireq\_ret != REQ\_PASS)  {  ul\_host\_printf(dut\_control, "Complete Request List = FAIL (%d) \n", ireq\_ret);  // Decode OCS Value ( User Function )  user\_decode\_ocs\_requestlist(dut\_control, SLOT00);  ierror\_code = ERR\_PHYREADY;  goto FAIL;  }  // Decode Response ( User Function )  user\_decode\_response(dut\_control, 1, 8, SLOT00);  // Attribute Response Data를 얻는다.  igetval = ul\_ufs\_get\_query\_attr\_value(dut\_control, SLOT00); |

## ul\_ufs\_get\_query\_data\_value()

|  |  |
| --- | --- |
| Function | void ul\_ufs\_get\_query\_data\_value(int dutid, unsigned char \*szgetval, int reqlist\_index) |
| Description | Query Response의 Descriptor Data를 얻기 위해 사용된다. |
| Argument | 1. int dutid : Dut ID 2. unsigned char \*szgetval : Query Response Data ( Length : 256 ) 3. int reqlist\_index : Slot Number |
| Return | NONE |

|  |
| --- |
| **Example Code - UFS Query Pseudo Code** |
| enum requestlist\_complete ireq\_ret = REQ\_PASS;  unsigned char szgetval[256];  // Query Operation을 설정  ul\_ufs\_build\_query\_upiu(dut\_control, QUERY\_STD\_RD\_REQ, QUERY\_RD\_DESC, DEVICE\_DESC\_IDN, 0x0, 0x0, 0x0, 0x0, 0x1F, 0x0, 0x0, NULL, SLOT00);  // UTP Transfer Request List Build  ul\_ufs\_build\_query\_requestlist(dut\_control, DIR\_READ, SLOT00);  // Transfer Request List Run  iret = ul\_ufs\_issue\_requestlist(dut\_control, DRB00);  // Check Request List Response  ireq\_ret = ul\_ufs\_check\_requestlist(dut\_control, polling\_timeout\_msec, SLOT00);  if (ireq\_ret != REQ\_PASS)  {  ul\_host\_printf(dut\_control, "Complete Request List = FAIL (%d) \n", ireq\_ret);  // Decode OCS Value ( User Function )  user\_decode\_ocs\_requestlist(dut\_control, SLOT00);  ierror\_code = ERR\_PHYREADY;  goto FAIL;  }  // Decode Response ( User Function )  user\_decode\_response(dut\_control, 1, 8, SLOT00);  // Descriptor Response Data를 얻는다.  memset( szgetval, 0, sizeof(szgetval) );  ul\_ufs\_get\_query\_data\_value(dut\_control, szgetval, SLOT00); |

## ul\_ufs\_build\_task\_management\_upiu()

|  |  |
| --- | --- |
| Function | void ul\_ufs\_build\_task\_management\_upiu(int dutid, char iflags , char ilun, char task\_func,  int reqlist\_index) |
| Description | Task Management Request UPIU 생성 시 사용된다. |
| Argument | 1. int dutid : Dut ID 2. char iflgas : Task Flags 3. char ilun : Logical Unit Number 4. char task\_func : Task Management Function   TASK\_ABORT\_FUNC 0x01  TASK\_SET\_ABORT 0x02  TASK\_SET\_CLEAR 0x04  TASK\_LUN\_RESET 0x08  TASK\_QUERY 0x80  TASK\_SET\_QUERY 0x81   1. int reqlist\_index : Slot Number |
| Return | NONE |

## ul\_ufs\_issue\_task\_management\_list()

|  |  |
| --- | --- |
| Function | unsigned int ul\_ufs\_issue\_task\_management\_list(int dutid, enum doorbell\_bit list\_bit) |
| Description | Task Management Request List를 Run 시키기 위해 사용된다. |
| Argument | 1. int dutid : Dut ID 2. enum doorbel\_bit list\_bit : Door Bell Register Bit 선택   DRB00=0x1<<0, DRB01=0x1<<1, DRB02=0x1<<2, DRB03=0x1<<3,  DRB04=0x1<<4, DRB05=0x1<<5, DRB06=0x1<<6, DRB07=0x1<<7 |
| Return | UFS Host Controller가 Not Ready 시 RET\_FAIL을 리턴한다. (RET\_FAIL/RET\_PASS) |

## ul\_ufs\_check\_taskmanagementlist()

|  |  |
| --- | --- |
| Function | enum taskmanagementlist\_complete ul\_ufs\_check\_taskmanagementlist(int dutid,  unsigned int polling\_timeout\_msec, int reqlist\_index) |
| Description | Task Management Request List의 Complete과 Response의 Pass/Fail 여부를 얻을 시 사용된다. |
| Argument | 1. int dutid : Dut ID 2. unsigned int polling\_timeout\_msec : Complete Interrupt Waiting Time 3. int reqlist\_index : Slot Number |
| Return | TASKMNGT\_PASS(0) : Task Management Request List Pass  TASKMNGT\_TIMEOUT(1) : Complete Interrupt Flag가 On이 되지 않을 시 발생.  TASKMNGT\_OSC\_ERROR(2) : OCS 값이 Success(0x0)이 아닐 경우 발생  TASKMNGT\_RSP\_TYPEERROR(3) : Response RX Transaction Type이 비정상적일 경우 발생  TASKMNGT\_STATUSEEEOR(4) : Task Management Response가 Comeplete이 아닐 경우 발생 |

|  |
| --- |
| **Example Code - UFS Task Management List Pseudo Code** |
| enum taskmanagementlist\_complete ireq\_ret = TASKMNGT\_PASS;  // Task Management Operation을 설정  ul\_ufs\_build\_task\_management\_upiu(dut\_control, 0x0, 0x0, TASK\_ABORT\_FUNC, SLOT00);  // Task Management Request UPIU Run  iret = ul\_ufs\_issue\_task\_management\_list(dut\_control, DRB00);  // Check Task Management Request List Response  ireq\_ret = ul\_ufs\_check\_taskmanagementlist(dut\_control, 500, SLOT00)  if (ireq\_ret != TASKMNGT\_PASS)  {  ul\_host\_printf(dut\_control, "Complete Task Management List = FAIL (%d) \n", ireq\_ret);  // Decode OCS Value ( User Function )  user\_decode\_ocs\_taskmanagementlist(dut\_control, SLOT00);  ierror\_code = ERR\_PHYREADY;  goto FAIL;  }  // Decode Response ( User Function )  user\_decode\_taskmanagement\_response(dut\_control, SLOT00); |
|  |

## ul\_ufs\_send\_UICcmd()

|  |  |
| --- | --- |
| Function | enum uic\_cmd\_complete ul\_ufs\_send\_UICcmd(unsigned int dutid, unsigned int uiccmd,  unsigned int ucmdarg1, unsigned int ucmdarg2, unsigned int ucmdarg3, unsigned int msec\_timeout) |
| Description | UIC Command를 수행 시 사용된다. |
| Argument | 1. int dutid : Dut ID 2. unsigned int uiccmd : UIC Command   UIC\_DME\_GET 0x01  UIC\_DME\_SET 0x02  UIC\_DME\_PEER\_GET 0x03  UIC\_DME\_PEER\_SET 0x04  UIC\_DME\_POWERON 0x10  UIC\_DME\_POWEROFF 0x11  UIC\_DME\_ENABLE 0x12  UIC\_RESET 0x14  UIC\_DME\_ENDPOINTRESET 0x15  UIC\_LINKSTARTUP 0x16  UIC\_HIBERNATE\_ENTER 0x17  UIC\_HIBERNATE\_EXIT 0x18  UIC\_DME\_TEST\_MODE 0x1A   1. unsigned int ucmdarg1 : UIC Command Argument 1 2. unsigned int ucmdarg2 : UIC Command Argument 2 3. unsigned int ucmdarg3 : UIC Command Argument 3 4. unsigned int msec\_timeout : Complete Interrupt Waiting Time |
| Return | RET\_UIC\_PASS(0) : UIC Command Operation Pass  RET\_UIC\_TIMEOUT(1) : Complete Interrupt Flag가 On이 되지 않을 시 발생.  RET\_UIC\_CONFIGRESULTCODE(2) : Config Result Code가 SUCCESS(0x00)이 아닐 경우 발생.  RET\_UIC\_GENERRORCODE(3) : Generic Error Code가 SUCCESS(0x00)이 아닐 경우 발생. |

## ul\_ufs\_get\_UICcmdMIBvalue()

|  |  |
| --- | --- |
| Function | unsigned int ul\_ufs\_get\_UICcmdMIBvalue(unsigned int dutid) |
| Description | UIC Command의 MIB Value( UIC Command Argument 3 )를 리턴 받을 때 사용된다. |
| Argument | 1. int dutid : Dut ID |
| Return | UIC Command의 MIB Value ( UIC Command Argument 3 ) |

## ul\_ufs\_display\_UICcmdresponsecode()

|  |  |
| --- | --- |
| Function | unsigned int ul\_ufs\_display\_UICcmdresponsecode(unsigned int dutid) |
| Description | UIC Command의 Result Code를 리턴 받고, 결과를 Display할 때 사용된다. |
| Argument | 1. int dutid : Dut ID |
| Return | UIC Command Argument 2 Value ( Config Result Code / Generic Error Code ) |

## ul\_ufs\_get\_UICcmdresponsecode()

|  |  |
| --- | --- |
| Function | unsigned int ul\_ufs\_get\_UICcmdresponsecode(unsigned int dutid) |
| Description | UIC Command의 Result Code를 리턴 받을 때 사용된다. |
| Argument | 1. int dutid : Dut ID |
| Return | UIC Command Argument 2 Value ( Config Result Code / Generic Error Code ) |

|  |
| --- |
| **Example Code - UIC Command Pseudo Code** |
| enum uic\_cmd\_complete iret\_code = RET\_UIC\_PASS;  unsigned int msec\_timeout = 500; //msec  unsigned int uiccmd = UIC\_DME\_GET;  unsigned int iMIBvalue = 0;  unsigned int iUICResp = 0;  // UIC Command Send  iret\_code = ul\_ufs\_send\_UICcmd(dut\_control, uiccmd, (PA\_PWRMODE << 16 | GENSELECTORINDEX), 0, 0, msec\_timeout);  // Get MIB Value ( UIC Command Argument 3 )  iMIBvalue= ul\_ufs\_get\_UICcmdMIBvalue(dut\_control);  // Check UIC Command Response  if (iret\_code != RET\_UIC\_PASS )  {  iUICResp = ul\_ufs\_get\_UICcmdresponsecode (dut\_control);  goto FAIL;  } |

## ul\_ufs\_set\_powermode()

|  |  |
| --- | --- |
| Function | int ul\_ufs\_set\_powermode(int dutid, int gear, int lane, int hsseries, int pwrmode, int term) |
| Description | UFS GEAR, Lane, HS\_Series, Power Mode 변경 시 사용된다. |
| Argument | 1. int dutid : Dut ID 2. int gear : Gear 3. int lane : Lane 4. int hsseries : HS Series 5. int pwrmode : Power Mode 6. int term : Termination |
| Return | RET\_PASS(0) : Power Mode Change Pass  RET\_FAIL(1) : Power Mode Change Fail 시 발생 |

## ul\_ufs\_create\_mackey()

|  |  |
| --- | --- |
| Function | int ul\_ufs\_create\_mackey( int dutid, unsigned char \*pszresult, unsigned char \*pszinput, unsigned int ilength ) |
| Description | UFS RPMB에 사용되는 MAC/Key를 생성하기 위해 사용된다. |
| Argument | 1. int dutid : Dut ID 2. unsigned char \*pszresult : MAC/Key Output ( 32 byte ) 3. unsigned char \*pszinput : Intput Data 4. int ilength : Input Data Length |
| Return | RET\_PASS(0) : MAC/Key 생성 Pass  RET\_FAIL(1) : MAC/Key 생성 Fail 시 발생 |