

---

# Renesas USB Device

R01AN0514EJ0110

Rev.1.10

## USB Peripheral Mass Storage Class Driver

---

Sep 01, 2011

### Introduction

This document is a manual describing use of the USB peripheral mass storage class driver with Renesas USB Device.

### Target Device

RX62N Group, RX630 Group, R8A66597

This application note also applies to other microcontrollers in the RX 600 Series that have the same USB module as the RX62N Group microcontrollers. When using this code in an end product or other application, its operation must be tested and evaluated thoroughly.

### Contents

1. Overview .....	2
2. Software Configuration.....	4
3. Peripheral Device Class Driver (PDCD).....	8
4. USB Peripheral Mass Storage Class Driver (PMSCD).....	13
5. Peripheral Mass Storage Device Driver (PMSDD) .....	33
6. Limitations .....	40

## 1. Overview

### 1.1 Overview

This document is a manual describing use of the USB peripheral mass storage class driver and the storage device sample driver with Renesas USB Device.

### 1.2 Functions and Features

The USB peripheral mass storage class driver comprises a USB mass storage class bulk-only transport (BOT) protocol. When combined with a USB peripheral control driver and storage device driver, it enables communication with a USB host as a BOT-compatible storage device.

### 1.3 Related Documents

1. USB Revision 2.0 Specification
2. USB Mass Storage Class Specification Overview Revision 1.1
3. USB Mass Storage Class Bulk-Only Transport Revision 1.0

[<http://www.usb.org/developers/docs/>]

4. RX62N Group, RX621 Group User's Manual: Hardware (Document No. R01UH0033EJ)
5. RX630 Group User's Manual: Hardware (Document No. R01UH0040EJ)
6. R8A66597 Data Sheet(Document No.REJ03F0229RJJ03F)
7. Renesas USB Device USB Basic Firmware Application Note (Document No. R01AN0512EJ)
8. RX600 Series USB Peripheral Mass Storage Class Driver (PMSC) installation guide (Document No. R01AN0529EJ)

Renesas Electronics Website

[[http:// www.renesas.com/](http://www.renesas.com/)]

USB Devices Page

[<http://www.renesas.com/prod/usb/>]

### 1.4 Terms and Abbreviations

USB:	Universal Serial Bus
USB-BASIC-F/W:	USB basic firmware for Renesas USB Device
H/W:	Renesas USB Device
RX62N-RSK:	Renesas Starter Kit+ for RX62N
RX630-RSK:	Renesas Starter Kit for RX630
nonOS:	USB-BASIC-F/W for OS less system
μITRON:	USB-BASIC-F/W for μITRON system
PCD:	Peripheral control driver of Renesas USB Device (USB-BASIC-F/W)
PMSCD:	Peripheral mass storage USB class driver (PMSCF + PCI + DDI)
PMSDD:	Sample peripheral mass storage device driver (sample ATAPI driver)
PMSCF:	Peripheral mass storage class function
PCI:	PCD interface
DDI:	Device driver interface
PDCD:	Peripheral device class driver (PMSCD + PMSDD)
BOT:	USB mass storage class bulk only transport

APL: : Application program

Task: : A processing unit

Scheduler macro : A macro used to call the scheduler function (NonOS version)

\*\*\*\*(): Function call

\*\*\*\*(capital letter) command: Storage device control command

## 2. Software Configuration

### 2.1 Module Configuration

As shown in figure 2.1, PDCD comprises two layers: PMSCD and PMSDD.

PMSCD comprises three layer: PCD interface function group (PCI) and PMSDD interface function group (DDI) and BOT protocol control and data sends and receives (PMSCF).

In the case of uTRON, PMSCD and PMSDD run on top of uTRON as tasks.

PMSCD uses the BOT protocol to communicate with the host via PCD.

PMSDD analyzes and executes storage commands received from PMSCD. PMSDD accesses media data via the media driver.

Figure 2.1 shows the configuration of the modules related to PDCD. Table 2.1 provides an overview of the individual modules.

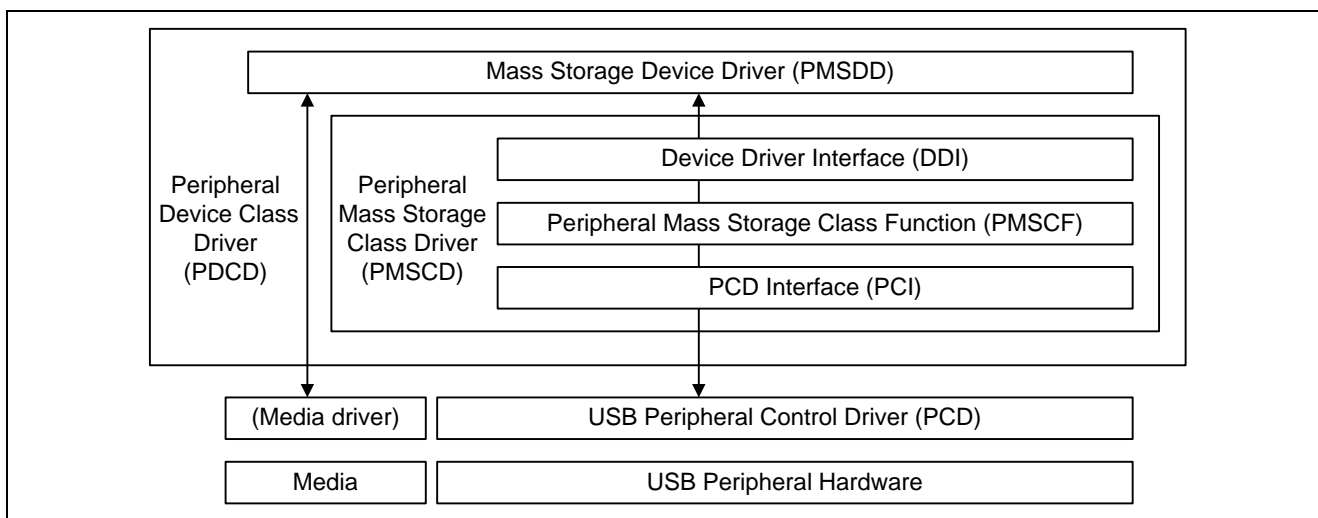


Figure 2.1 Software Configuration Diagram

Table 2.1 Overview of Module Functions

Module	Description
PMSDD	A mass storage device driver. It processes storage commands from PMSCD and accesses the media via the media driver. It should be created (modified) by the customer to match the system specifications.
DDI	PMSDD-PMSCF interface functions.
PMSCF	This is the core component of PMSCD. It controls BOT protocol data and responds to class requests. It also transfers storage commands and data to and from PMSDD.
PCI	PMSCF-PCD interface functions.
PCD	USB peripheral hardware control driver.
Media driver	Storage device control driver (including PMSDD in the sample).

## 2.2 File Structure Listing

### 2.2.1 Folder Structure

The following shows the folder structure for the files provided in this device class.

```

+---Workspace
  +---MSCCFW          USB0, 1 shared
    | +---include      PHID header file
  +---MSCFW          For USB0
    | +---PMSC        PMSC Driver
    | +---MEDIA       MEDIA Driver
    | +---include      PMSC header file
  +---MSC2FW         For USB1 (RX630 and R8A66597 are not used.) :Structure is same to HIDFW
+---RI600_4         uITRON folder (not included in nonOS version)
  | +--- config_Pmsc  uITRON configuration file for PMSC: "r_usb_RX62N.cfg"/"r_usb_RX630.cfg"
  +---SmplMain
    | +---APL         Sample Application
  +---USBSTDFW       For USB0
  +---USBCSTDFW      USB0, 1 shared
    | +---include      Common header file system resource
  +---USB2STDFW      For USB1 (RX630 and R8A66597 are not used.) :Structure is same to USBSTDFW
  +---HwResourceForUSB

```

Note: Some folders that are not used in this application have been eliminated from this class driver description.

Table 2.2 shows the file structure supplied with PDCD.

**Table 2.2 File Structure**

File Name	Description	Note
MEDIA/r_usb_ATAPImemory.c	FAT (16) data	Sample
MEDIA/r_usb_ATAPIdriver.c	Device driver (PMSDD/media driver)	Sample
include/r_usb_cATAPIdefine.h	Device driver header file	Sample
PMSC/r_usb_PMSCddi.c	PMSDD interface functions (DDI)	
PMSC/r_usb_PMSCdriver.c	USB class driver (PMSCF)	
PMSC/r_usb_PMSCpci.c	PCD interface functions (PCI)	
PMSC/r_usb_PMSrequest.c	PCD interface functions (class requests)	
include/r_usb_pMSCdefine.h	PMSCD header file	
PMSC/r_usb_PMSCdescriptor.c	Mass storage class descriptor	Sample
include/r_usb_cMSCdefine.h	PDCD(PMSCD+PMSDD) common header file	
include / r_usb_pMSCextern.h	External reference header file	

## 2.3 System Resources

### 2.3.1 System Resource Definitions for $\mu$ ITRON

Table 2.3 shows the  $\mu$ ITRON resources used by PDCD on the  $\mu$ ITRON version.

These resources are defined in the `r_usb_RX62N.cfg` file or `r_usb_RX630.cfg` file.

For details on how to define, refer to the Renesas USB Device USB Basic Firmware Application note.

**Table 2.3**  $\mu$ ITRON Resources

	Name	Description
Task Stack size: USB_TSK_STK (512)	USB_PMSC_TSK	PMSCD main task (usb_pmsc_Task) Task_ID: USB_PMSC_TSK Task priority: 5
	USB_PFLSH_TSK	PMSDD main task (usb_pmsc_SmpAtapiTask) Task_ID: USB_PFLSH_TSK Task priority: 5
Mailbox max Priority: 1 Waiting task queue: FIFO order	USB_PMSC_MBX	PDCD -> PMSCD / PMSDD -> PMSCD mailbox ID
Message queue: FIFO order	USB_PFLSH_MBX	PMSCD -> PMSDD mailbox ID
Memory pool Block count: USB_BLK_CNT (10)	USB_PMSC_MPL	PMSCD memory pool ID
Block size: USB_BLK_SIZ (64) Waiting task queue: FIFO order	USB_PFLSH_MPL	PMSDD memory pool ID
OS base timer	Hardware timer	1 ms

### 2.3.2 System Resource Definitions for nonOS

Shows Table 2.4 lists the ID and priority definitions used to register PMSC in the scheduler.

These are defined in the `r_usbc_cKernelId.h` header file.

For details on how to define, refer to the Renesas USB Device USB Basic Firmware Application note.

**Table 2.4 List of Scheduler Registration IDs**

	Name	Description
Scheduler registration task	USB_PMSC_TSK (default value: USBC_TID_1)	PMSCD main task (usb_pmsc_Task) Task_ID: USB_PMSC_TSK Task priority: USB_PMSC_PRI (default value: USBC_PRI_1)
	USB_PFLSH_TSK (default value: USBC_TID_2)	PMSDD main task (usb_pmsc_SmpAtapiTask) Task_ID: USB_PFLSH_TSK Task priority: USB_PFLSH_PRI (default value: USBC_PRI_2)
Mailbox ID	USB_PMSC_MBX (default value: USB_PMSC_TSK)	PDCD -> PMSCD / PMSDD -> PMSCD mailbox ID
	USB_PFLSH_MBX (default value: USB_PFLSH_TSK)	PMSCD -> PMSDD mailbox ID
memory pool ID	USB_PMSC_MPL (default value: USB_PMSC_TSK)	PMSCD memory pool ID
	USB_PFLSH_MPL (default value: USB_PFLSH_TSK)	PMSDD memory pool ID

### 2.3.3 USB Interrupt Handler Definition

This application uses the USB interrupt handler. For details on how to define the handler, refer to the Renesas USB Device USB Basic Firmware Application note.

**Table 2.5 USB Interrupt Handler**

	Name	Description
USB Interrupt Handler	usb_cstd_UsbIntHand USB	Interrupt handler

### 3. Peripheral Device Class Driver (PDCD)

#### 3.1 Basic Functions

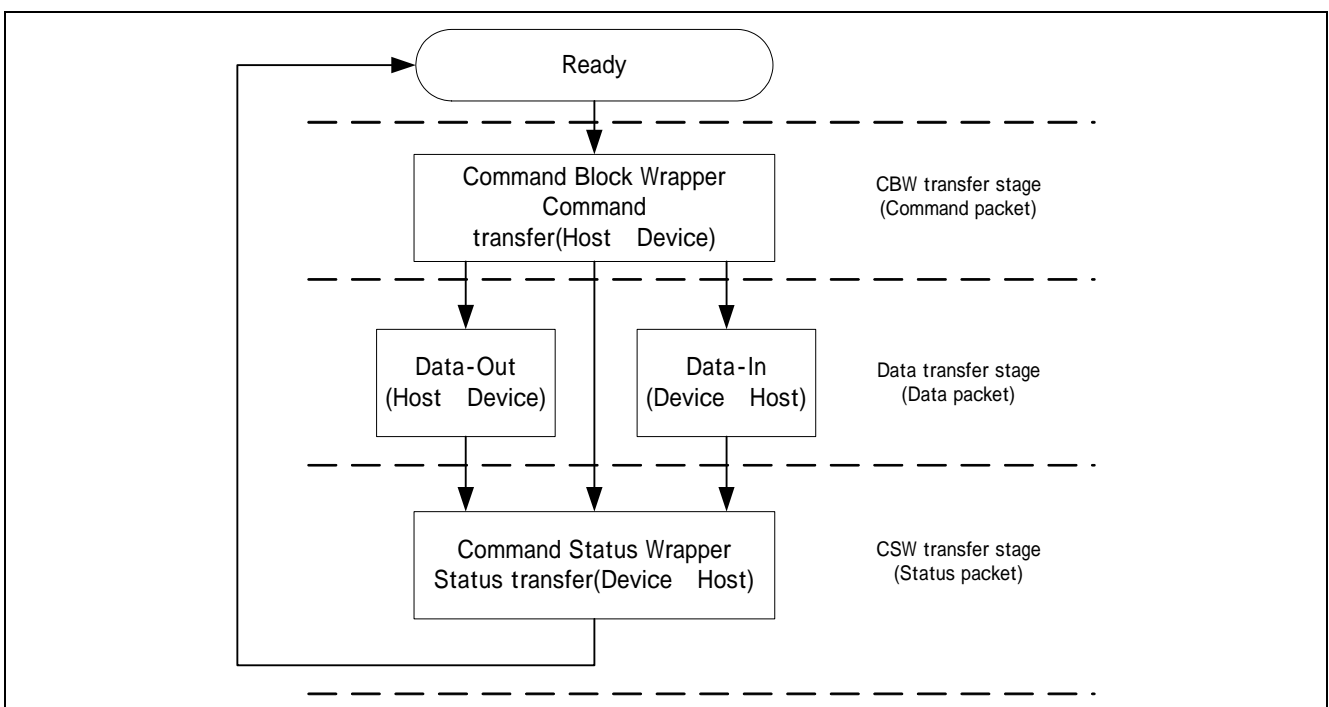
The functions of PDCD are as follows.

1. Storage command control using BOT protocol
2. Responding to class requests from the host
3. Responding to storage commands from the host

#### 3.2 BOT Protocol Overview

BOT(Bulk-Only Transport) is a transfer protocol that, manages command, data, and status (result of the command processing) by only 2 Endpoint(bulk In and bulk Out).

The Data format of command and status are Command Block Wrapper(CBW) and Command Status Wrapper(CSW).



**Figure 3.1 BOT protocol Over view**

When PMSCD receives a command block wrapper (CBW) from the host, it first verifies the validity of the CBW. If the CBW is valid, PMSCD notifies PMSDD of the storage command contained in the CBW and requests analysis of the command. PMSCD then performs processing based on the information from the analysis by PMSDD (command validity, data transfer direction and size) and the information contained in the wrapper (data communication direction and size).

- For BOT detail, refer to the [Universal Serial Bus Mass Storage Class Bulk-Only Transport - Revision 1.0] of the USB Implementers Forum issue.

### 3.2.1 Sequence of Storage Command for No Transmit/Receive Data

Figure 3.2 shows the sequence of storage command when no data transfers occurs. In the CBW transfer stage, PMSCD executes CBW receive request to PCD, registers callback function. When PCD receives CBW, PCD executes callback function. In the CBW transfer stage PMSCD verifies the validity of the CBW and transfers the storage command (CBWCB) to PMSDD when callback receives. PMSCD then compares the analysis result from PMSDD with the information contained in the CBW and sends a storage command execution request to PMSDD. PMSDD executes storage command processing and returns the result to PMSCD as a callback. Based on the execution result, PMSCD creates a command status wrapper (CSW) and transmits it to the host via PCD.

For the PCD operation detail, refer to "USB basic firmware Application note". And for PMSCD sequence detail, refer to "Figure 4.1 Command Sequence When No Transmit/Receive Data Present".

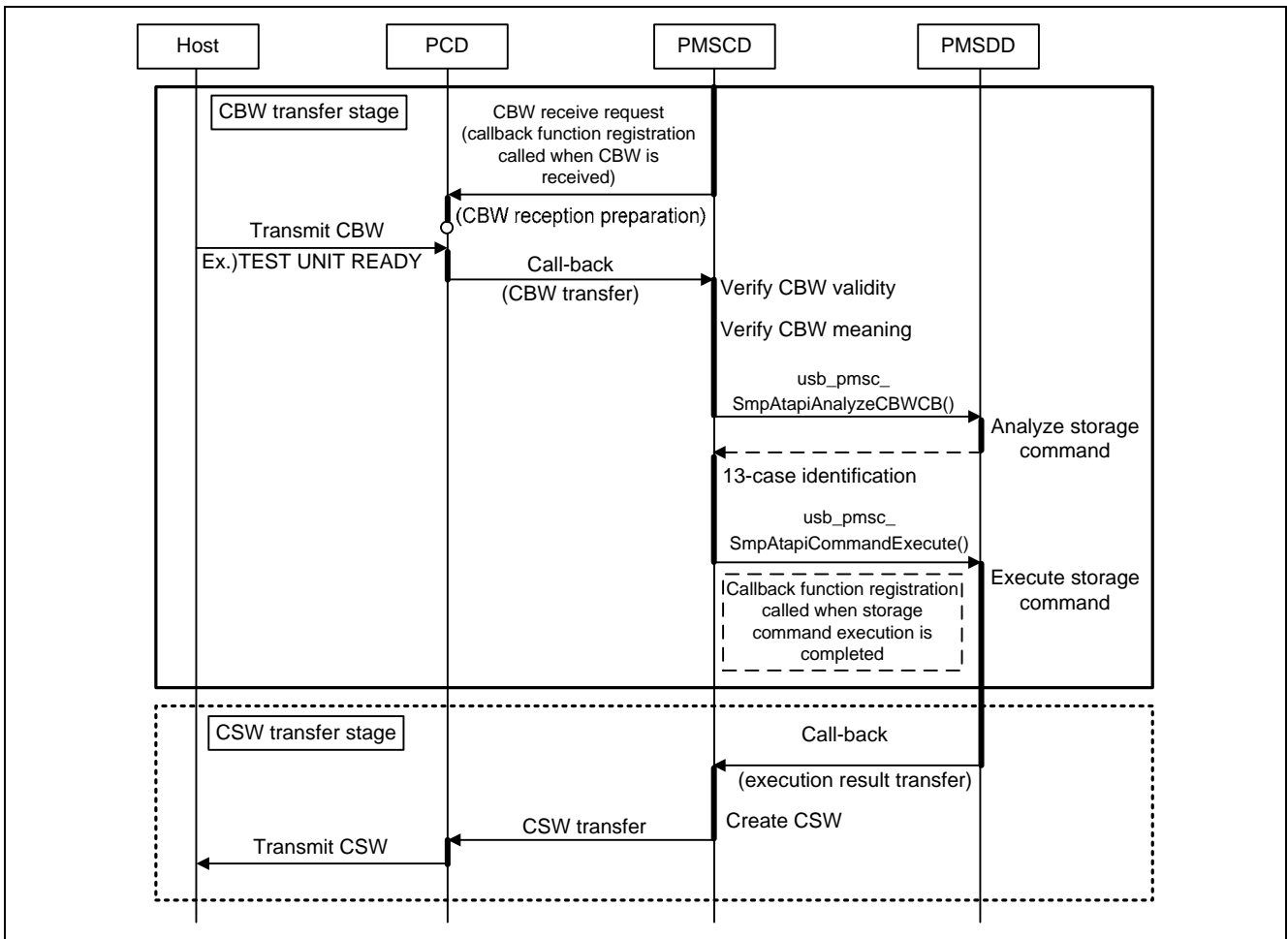


Figure 3.2 Sequence of Storage Command for No Transmit/Receive Data

### 3.2.2 Sequence of Storage Command for Transmit (IN) Data

Figure 3.3 shows the sequence of storage command when there is transmit (IN) data from the peripheral side. In the CBW transfer stage, PMSCD executes CBW receive request to PCD, when PCD receives CBW, PCD executes callback function set up at the time of a CBW receive request. In the CBW transfer stage PMSCD verifies the validity of the CBW and transfers the storage command (CBWCB) to PMSDD when callback receives. PMSDD checks the data transmit command, and returns the result to PMSCD. PMSCD then compares the analysis result from PMSDD with the information contained in the CBW and sends a storage command execution request to PMSDD. Based on the execution result, PMSCD notifies PCD of the data storage area and data size, and data communication with the host takes place. When it receives transmit end notification from PCD, PMSCD once again sends a common continuation request to PMSDD, and data transmission is repeated. When it receives a command processing end result from PMSDD, PMSCD creates a command status wrapper (CSW) and transmits it to the host via PCD.

For the PCD operation detail, refer to "USB basic firmware Application note". And for PMSCD sequence detail, refer to "Figure 4.2 Command Sequence When Transmit (IN) Data Present".

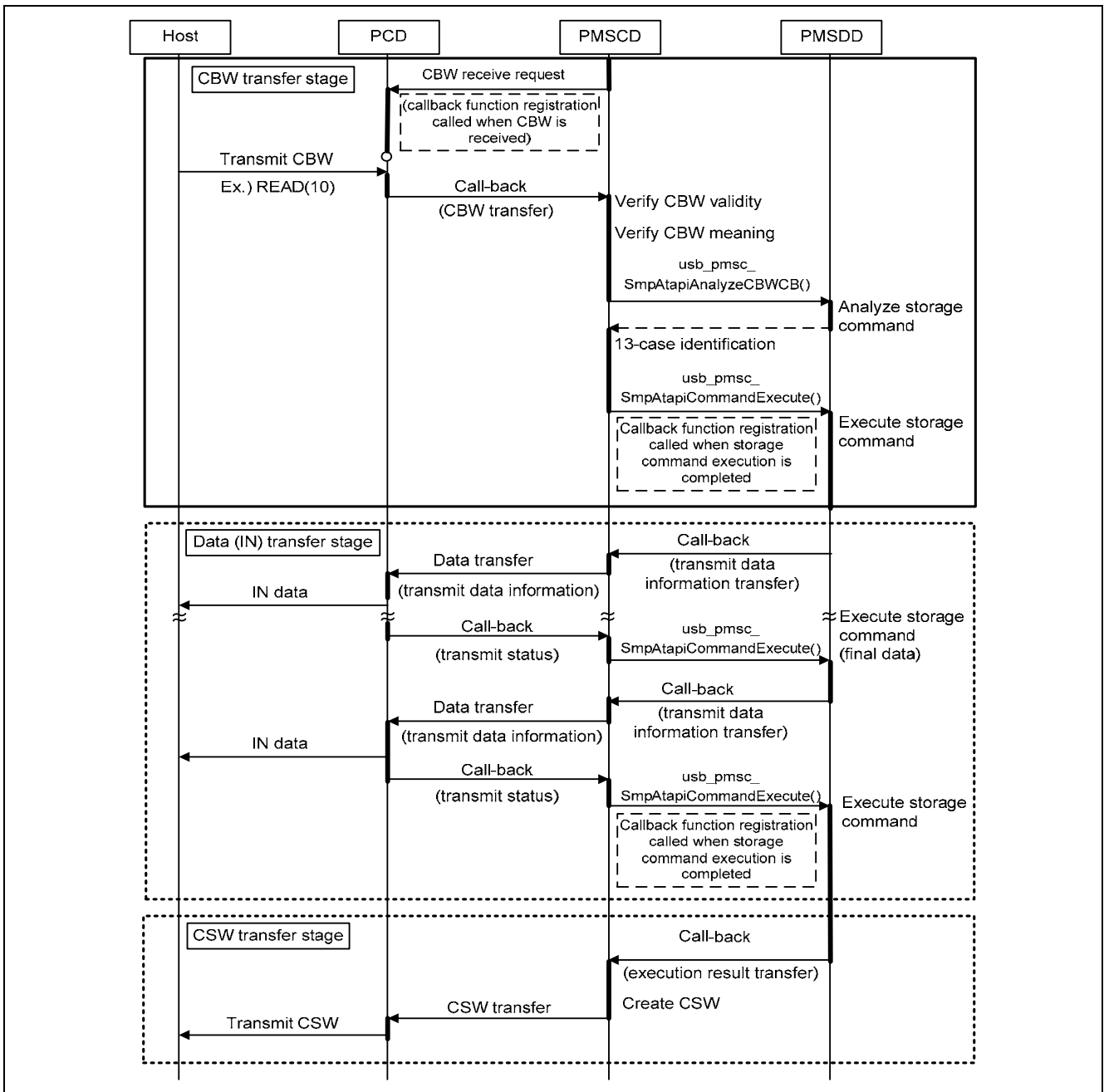


Figure 3.3 Sequence of Storage Command for Transmit (IN) Data

### 3.2.3 Sequence of Storage Command for Receive (OUT) Data

Figure 3.4 shows the sequence of storage command when there is transmit (OUT) data from the peripheral side. In the CBW transfer stage, PMSCD executes CBW receive request to PCD, when PCD receives CBW, PCD executes callback function set up at the time of a CBW receive request. In the CBW transfer stage PMSCD verifies the validity of the CBW and transfers the storage command (CBWCB) to PMSDD when callback receives. PMSDD checks the data transmit command, and returns the result to PMSCD. PMSCD then compares the analysis result from PMSDD with the information contained in the CBW and sends a storage command execution request to PMSDD. Based on the execution result, PMSCD notifies PCD of the data storage area and data size, and data communication with the host takes place. When it receives transmit end notification from PCD, PMSCD once again sends a common continuation request to PMSDD, and data transmission is repeated. When it receives a command processing end result from PMSDD, PMSCD creates a command status wrapper (CSW) and transmits it to the host via PCD.

For the PCD operation detail, refer to "USB basic firmware Application note". And for PMSCD sequence detail, refer to "Figure 4.3 Command Sequence When Receive (OUT) Data Present".

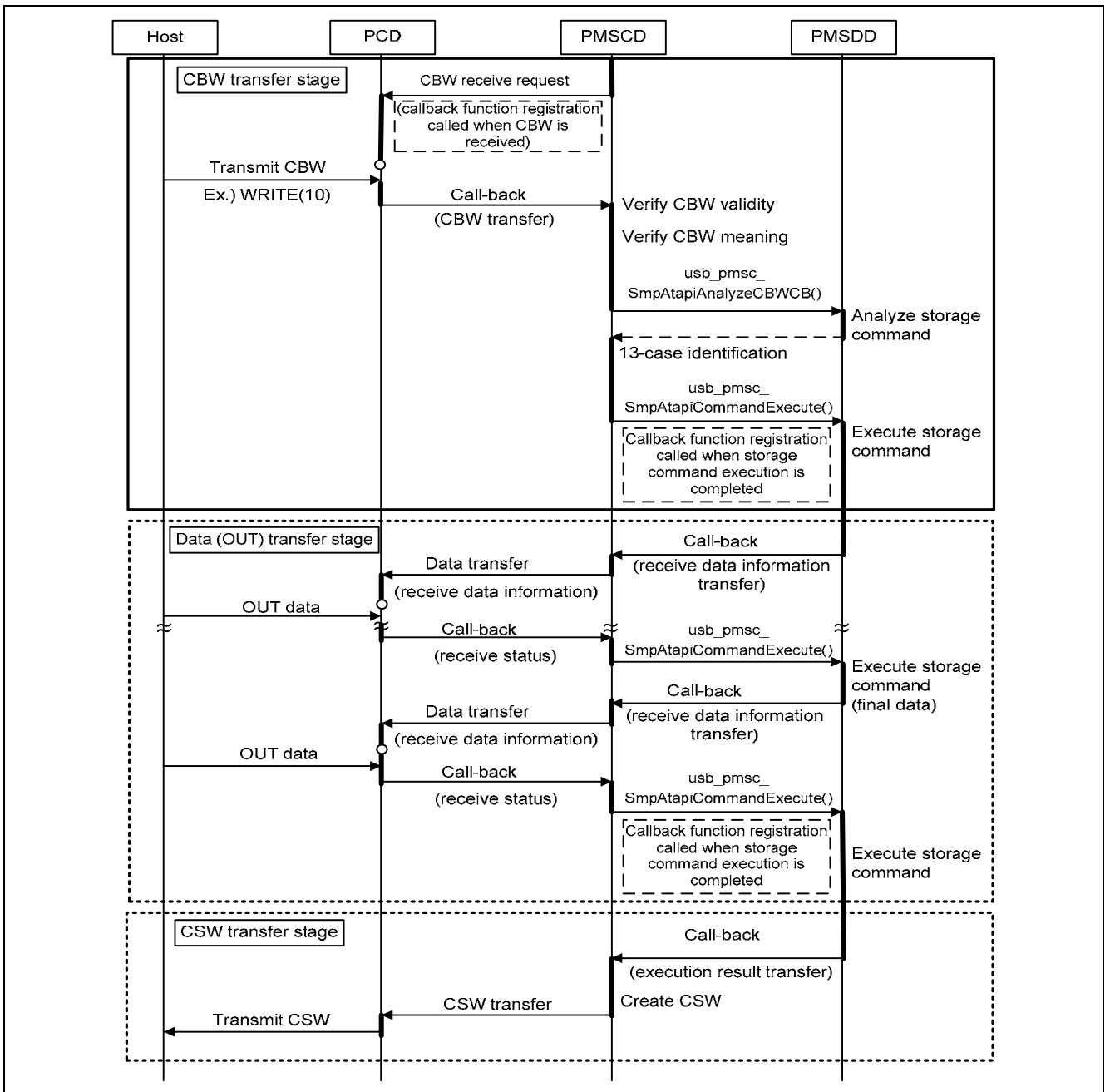


Figure 3.4 Sequence of Storage Command for Receive (OUT) Data

### 3.2.4 Access Sequence for a Class Request

Figure 3.5 shows the sequence when a mass storage class request is received. When PCD receives a class request in the control transfer setup stage, it sends a request received notification to PMSCD. PMSCD executes the control transfer data stage and notifies PCD of data stage end by means of a callback function. PCD executes the status stage and ends the control transfer.

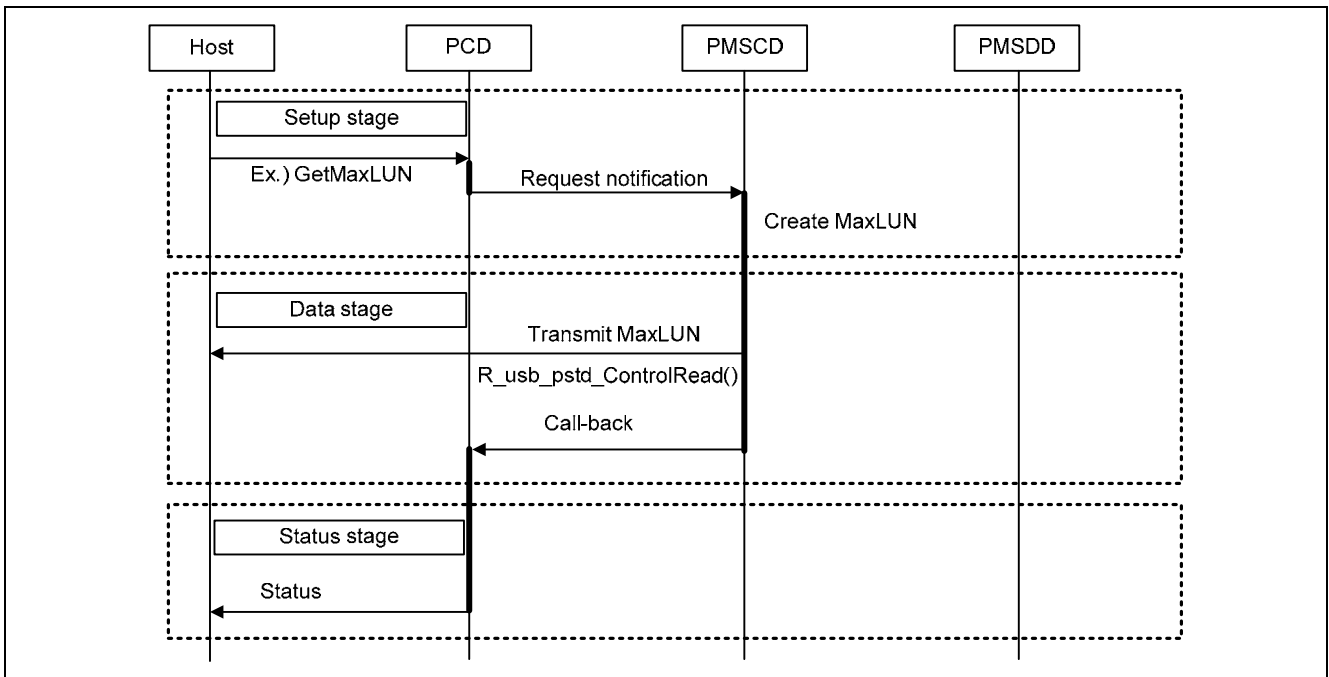


Figure 3.5 Sequence for Class Request

## 4. USB Peripheral Mass Storage Class Driver (PMSCD)

### 4.1 Basic Functions

The basic functions of PMSCD are as follows.

1. Storage command control using BOT protocol
2. Transferring storage commands and data information to and from PCD and PMSDD.
3. Responding to mass storage class requests

### 4.2 Overview of Functions

As shown in figure 2.1, PMSCD comprises three layers: PMSCF, which performs BOT protocol control and data transmission/reception; a group of functions (DDI) for interfacing with PMSDD; and a group of functions (PCI) for interfacing with PCD. The main functions of these layers are as follows.

1. The functions of PMSCF are as follows.
  - USB mass storage class BOT protocol control
  - CBW analysis, data transmission/reception, and CSW creation in coordination with PMSDD/PCD
  - Responding to class requests (MassStorageReset, GetMaxLUN )
2. The functions of PCI are as follows.
  - Processing of tasks, message boxes, and memory pools during configuration and detach
  - Receiving class requests
  - Clearing STALL states and setting related callback functions
  - Setting structures and callback functions for PCD transmit/receive data
3. The functions of DDI are as follows.
  - Transferring data information and execution results during PMSDD execution

### 4.3 List of PMSCF Global Area Variables

Table 4.1 lists the global area variables of PMSCF. Table 4.2 lists the global area variables of PCI.

**Table 4.1 List of PMSCF Global Area Variables**

Variable Name		Description
1	uint8_t usb_gpmisc_Seq	PMSCD operating state (See table 4.46 for details.)
2	uint32_t usb_gpmisc_Tag	CBW tag data (copy of usb_gpmisc_Cbw.dCBWTag)
3	uint32_t usb_gpmisc_Dtl	Transmit/receive data byte length specified by the host (copy of usb_gpmisc_Cbw.dCBWDTL_xx)
4	USBC_MSC_CBW_t usb_gpmisc_Cbw	CBW receive structure (See table 4.3 for details.)
5	USBC_MSC_CSW_t usb_gpmisc_Csw	CSW transmit structure (See table 4.4 for details.)
6	USBC_PMSC_CBM_t usb_gpmisc_Messag e	Storage command analysis result from PMSDD (See table 4.5 for details.)
7	uint16_t usb_gpmisc_Outpipe	Pipe No. of OUT pipe
8	uint16_t usb_gpmisc_Inpipe	Pipe No. of IN pipe

**Table 4.2 List of PCI Global Area Variables**

Variable Name		Description
1	uint8_t usb_gpmisc_CbwCbLength	Storage command length (See bInterfaceClass and bInterfaceSubClass configuration descriptors.)
2	USBC_UTR_t usb_gpmisc_Mess	Message structure from PMSCD to PCD (See table 4.6 for details.)
3	uint8_t usb_gpmisc_InterfaceSubClass [7]	Subclass storage command length definition (Stipulated for SFF-8070i only.)

## 4.4 PMSCD Structures

Tables 4.3 to 4.6 show the main structures used by PMSCF, PCI, and DDI.

**Table 4.3 USBC\_MSC\_CBW\_t Structure**

	Member	Description	Remarks
uint32_t	dCBWSignature	CBW Signature	0x55534243: USBC
uint32_t	dCBWTag	CBW Tag	Tag corresponding to CSW
uint8_t	dCBWDTL_Lo	CBW DataTransfer Length	Data length of transmit/receive data
uint8_t	dCBWDTL_ML		
uint8_t	dCBWDTL_MH		
uint8_t	dCBWDTL_Hi		
uint8_t	bmCBWFlags	CBW Direction	Data transmit/receive direction
uint8_t	bCBWLUN	Logical Unit Number	Unit number
uint8_t	bCBWCBLength	CBWCB Length	Command length
uint8_t	CBWCB[16]	CBWCB	Command block

**Table 4.4 USBC\_MSC\_CSW\_t Structure**

	Member	Description	Remarks
uint32_t	dCSWSignature	CSW Signature	0x55534253: USBS
uint32_t	dCSWTag	CSW Tag	Tag corresponding to CBW
uint8_t	dCSWDataResidue_Lo	CSW DataResidue	Data length used
uint8_t	dCSWDataResidue_ML		
uint8_t	dCSWDataResidue_MH		
uint8_t	dCSWDataResidue_Hi		
uint8_t	bCSWStatus	CSW Status	Command status

**Table 4.5 USBC\_PMSC\_CBM\_t Structure**

	Member	Description	Remarks
uint32_t	ar_rst	PMSDD storage command analysis result (data direction)	Analysis result of usb_pm_sc_SmpAtapi AnalyzeCbwCb function
uint32_t	ul_size	PMSDD storage command analysis result (data size)	Analysis result of usb_pm_sc_SmpAtapi AnalyzeCbwCb function

**Table 4.6 USBC\_UTR\_t Structure**

	Variable	Description	Remarks
USBC_MH_t	msghead	Message header for SI7000/4	uITRON information area
uint16_t	msginfo	Message Info for firmware	HCD information area
uint16_t	keyword	Port number, pipe number, etc.	HCD information area
void	*tranadr	Transfer data Start address	Transmit/receive data area
uint32_t	tranlen	Transfer data length	Transfer data size
uint16_t	*setup	Setup packet (for control only)	Setup packet address
uint16_t	status	Transfer status	HCD operation result
uint16_t	pipectr	Status of PIPECTR	SQMON state during pipe switching
USBC_CB_t	complete	Callback Function Address	Callback run function
uint8_t	errcnt	Error count	Error information area used by HCD
uint8_t	segment	Segment information	Continuous communication determination

## 4.5 PMSCD Constant Definitions

Tables 4.7 and 4.8 list the PMSCD constant definitions.

**Table 4.7 List of PMSCD Constant Definitions (Host/Peripheral Common Definitions)**

	Description	Definition Name	Value
1	BOT wrapper		
	CBW data length	USBC_MSC_CBWLENGTH	31
	CBW signature	USBC_MSC_CBW_SIGNATURE	0x55534243
	CSW signature	USBC_MSC_CSW_SIGNATURE	0x55534253
	CBWCB data length (fixed length for ATAPI specification)	USBC_MSC_CBWCB_LENGTH	12
	CSW data length	USBC_MSC_CSW_LENGTH	13
2	CSW status		
	Storage command OK	USBC_MSC_CSW_OK	0x00
	Storage command FAIL	USBC_MSC_CSW_NG	0x01
	Phase error	USBC_MSC_CSW_PHASE_ERR	0x02
3	Descriptor check		
	USB subclass code	USBC_ATAPI	0x05
		USBC_SCSI	0x06
	USB protocol code	USBC_BOTP	0x50
	Endpoint total	USBC_TOTALEP	0x02
4	Storage command		
		USBC_ATAPI_TEST_UNIT_READY	0x00
		USBC_ATAPI_REQUEST_SENSE	0x03
		USBC_ATAPI_FORMAT_UNIT	0x04
		USBC_ATAPI_INQUIRY	0x12
		USBC_ATAPI_MODE_SELECT6	0x15
		USBC_ATAPI_MODE_SENSE6	0x1A
		USBC_ATAPI_START_STOP_UNIT	0x1B
		USBC_ATAPI_PREVENT_ALLOW	0x1E
		USBC_ATAPI_READ_FORMAT_CAPACITY	0x23
		USBC_ATAPI_READ_CAPACITY	0x25
		USBC_ATAPI_READ10	0x28
		USBC_ATAPI_WRITE10	0x2A
		USBC_ATAPI_SEEK	0x2B
		USBC_ATAPI_WRITE_AND_VERIFY	0x2E
		USBC_ATAPI_VERIFY10	0x2F
		USBC_ATAPI_MODE_SELECT10	0x55
		USBC_ATAPI_MODE_SENSE10	0x5A

Table 4.8 List of PMSCD Constant Definitions (Definitions for Dedicated Peripheral Macros)

	Description	Definition Name	Value
1	PCD operation definition		
	Clear pipe stall	USBC_PMSC_PIPE_STALL_CLEAR	0xFC00
2	PMSCD execution state definitions (usb_gpmc_Seq variable)		
	CBW receive state	USBC_PMSC_PCBWRCV	0x00
	Transmit data present state	USBC_PMSC_PDASND	0x01
	Receive data present state	USBC_PMSC_PDARCV	0x02
	CSW transmit state	USBC_PMSC_PCSWSND	0x03
	Error0 processing state	USBC_PMSC_PERROR0	0x04
	Error1 processing state	USBC_PMSC_PERROR1	0x05
	Error2 processing state	USBC_PMSC_PERROR2	0x06
	Error3 processing state	USBC_PMSC_PERROR3	0x07
	Error4 processing state	USBC_PMSC_PERROR4	0x08
	Error5 processing state	USBC_PMSC_PERROR5	0x09
	Command check state	USBC_PMSC_PCHECK	0xFF
3	Message transfer direction definitions		
	PCD → PMSCD	USBC_PMSC_USB2PMSC	0x00
	PMSDD → PMSCD	USBC_PMSC_PFLASH2PMSC	0x01
4	PMSDD execution result notification definitions		
	Command execution end and success	USBC_PMSC_CMD_COMPLETE	0x00
	Command execution end and failure	USBC_PMSC_CMD_FAILED	0x01
	Command execution in progress (repeated execution)	USBC_PMSC_CMD_CONTINUE	0x02
	Internal error during command execution (debug target)	USBC_PMSC_CMD_ERROR	0x03
5	Subclass count definition		
	Subclass count max. value	USBC_PMSC_MAX_SUBCLASS_RANGE	6

## 4.6 List of PMSCD Functions

Table 4.9 lists the PMSCF functions and describes what they do. Tables 4.10 and 4.11 list the PCI and DDI functions and describe their operation.

**Table 4.9 List of PMSCF Functions**

	Function Name	Description
1	usb_pmsc_Task	PMSCD task
2	usb_pmsc_CheckValid	Checks CBW validity.
3	usb_pmsc_CheckMeaning	Checks CBW meaning
4	usb_pmsc_CheckCase13	Performs BOT protocol 13-case identification.
5	usb_pmsc_TransferMatrix	Determines actual processing state from identified 13-case.
6	usb_pmsc_CommandCheck	Creates 13-case identification information from PMSDD storage command analysis result.
7	usb_pmsc_UsbExecute	Determines PMSDD storage command processing result (data transfer continuation determination).
8	usb_pmsc_CswSet	Sets CSW information.
9	usb_pmsc_Error0	Performs processing when CBW validity error occurs.
10	usb_pmsc_Error1	(1) Performs processing when CBW meaning error occurs. (2) Performs BOT protocol case 4 processing. (3) Performs processing when command not supported (IN pipe stall) occurs.
11	usb_pmsc_Error2	(1) Performs BOT protocol case 9 and case 11 processing. (2) Performs processing when command not supported (OUT pipe stall) occurs.
12	usb_pmsc_Error3	Performs BOT protocol case 2, case 3, case 7, and case 8 processing.
13	usb_pmsc_Error4	(1) Performs BOT protocol case 10 and case 13 processing. (2) Performs BOT protocol internal device error processing.
14	usb_pmsc_Error5	Performs processing when command not supported (no pipe stalled) occurs.
15	usb_pmsc_MassStorageReset	Responds to mass storage class MassStorageReset request.
16	usb_pmsc_GetMaxLun	Responds to mass storage class GetMaxLUN request.

Table 4.10 List of PCI Functions

	Function Name	Description
1	R_usb_pmsc_Open	Generates and starts task, mailbox, and memory pool.
2	R_usb_pmsc_Close	Releases task, mailbox, and memory pool.
3	R_usb_pmsc_MassStorageReset	Notifies of MassStorageReset request.
4	R_usb_pmsc_GetMaxLun	Notifies of GetMaxLUN request.
5	R_usb_pmsc_SetConfig	Processes Set Configuration request (enables mass storage command reception).
6	R_usb_pmsc_SetInterface	Processes Set Interface request
7	R_usb_pmsc_DescriptorChange	Updates descriptor data at HS/FS switching.
8	R_usb_pmsc_DataTrans	Sets and sends to PCD data transmit/receive structure.
9	R_usb_pmsc_TransResult	Callback at PCD data transmit/receive end.
10	R_usb_pmsc_StallClearResult	Callback at stall clear execution.
11	R_usb_pmsc_ProcessWaitTmo	Callback function at processing end with timeout.
12	R_usb_pmsc_ControlTrans0	Class request (Dummy function)
13	R_usb_pmsc_ControlTrans1	Receives class request (GetMaxLUN request).
14	R_usb_pmsc_ControlTrans2	Class request (Dummy function)
15	R_usb_pmsc_ControlTrans3	Receives class request (Mass Storage Reset request).
16	R_usb_pmsc_ControlTrans4	Class request (Dummy function)
17	R_usb_pmsc_ControlTrans5	Class request (Dummy function)

Table 4.11 List of DDI Functions

	Function Name	Description
1	usb_pmsc_AtapiTransResult	Callback at storage command execution by PMSDD.

## 4.7 Description of PMSCD Functions

### 4.7.1 PMSCF Functions

Table 4.12 usb\_pmsc\_Task()

Name	PMSCD Task		
Call format	void usb_pmsc_Task(void)		
Arguments	void		
Return values	void		
Description	PMSCD task. Controls BOT protocol.		
Notes	See 4.8 for details.		

Table 4.13 usb\_pmsc\_CheckValid()

Name	Check CBW Validity		
Call format	uint8_t usb_pmsc_CheckValid(uint32_t length)		
Arguments	uint32_t	length	
Return values	uint8_t		USBC_PMSC_PCHECK: Normal end USBC_PMSC_PERROR0: Error end
Description	a) Copies tag information from CBW to PMSCD global variables. b) Checks whether or not dCBWSignature of CBW is valid. c) Checks whether or not CBW data length is correct (were 31 bytes received?).		
Notes			

**Table 4.14** usb\_pmsc\_CheckMeaning()

Name	Check CBW Meaning		
Call format	uint8_t usb_pmsc_CheckMeaning(uint8_t seq)		
Arguments	uint8_t	seq	usb_pmscCheckValid function execution result
Return values	uint8_t		USBC_PMSC_PCHECK: Normal end USBC_PMSC_PERROR1: Error end
Description	1) Checks whether or not reserved bits in CBW are cleared to 0. a) Bit 7 in bmCBWFlags b) Bits 7 to 4 in bCBWLUN c) Bits 7 to 5 in bCBWCBLength 2) Checks bCBWLUN value in CBW. 3) Checks whether or not bCBWCBLength value is equal to or less than value determined by subclass.		
Notes	The bCBWLUN is set to 0. On the SFF-8070i, the subclass determines an bCBWCBLength value of 12.		

**Table 4.15** usb\_pmsc\_CheckCase13()

Name	BOT Protocol 13-Case Identification		
Call format	uint8_t usb_pmsc_CheckCase13(uint32_t ul_size, uint8_t *uc_case)		
Arguments	uint32_t	ul_size	Transfer data size determined by PMSDD
	uint8_t	*uc_case	Transfer data direction determined by PMSDD
Return values	uint8_t		13-case identification result
Description	Performs BOT protocol 13-case identification.		
Notes			

**Table 4.16** usb\_pmsc\_TransferMatrix()

Name	Select PMSCD Processing (usb_gpmsc_Seq)		
Call format	uint8_t usb_pmsc_TransferMatrix(uint8_t uc_pmsc_Case)		
Arguments	uint8_t	uc_pmsc_Case	13-case identification result.
Return values	uint8_t		PMSCD processing contents
Description	Selects the PMSCD processing (usb_gpmscSeq). 1) USBC_PMSC_PCSWSND (BOT protocol case 1) 2) USBC_PMSC_PDASND (BOT protocol case 5/case 6) 3) USBC_PMSC_PDARCV (BOT protocol case 12) 4) USBC_PMSC_PERROR1 (BOT protocol case 4) 5) USBC_PMSC_PERROR2 (BOT protocol case 9/case 11) 6) USBC_PMSC_PERROR3 (BOT protocol case 2/case 3/case 7/case 8) 7) USBC_PMSC_PERROR4 (BOT protocol case 10/case 13) 8) USBC_PMSC_PERROR4 (Other than the above)		
Notes			

**Table 4.17 usb\_pmsc\_CommandCheck()**

Name	Create BOT Protocol 13-Case Identification Information		
Call format	uint8_t usb_pmsc_CommandCheck(uint8_t seq)		
Arguments	uint8_t	seq	usb_pmsc_CheckMeaning function execution result
Return values	uint16_t		PMSCD processing contents
Description	1) Creates device-side information for BOT protocol 13-case identification. USBC_ATAPI_NO_DATA → USBC_MSC_DNXX USBC_ATAPI_SND_DATAS → USBC_MSC_DIXX USBC_ATAPI_RCV_DATAS → USBC_MSC_DOXX USBC_ATAPI_NOT_SUPPORT: CBW size: 0 → USBC_PMSC_PERROR5 CBW direction: IN → USBC_PMSC_PERROR1 CBW direction: OUT → USBC_PMSC_PERROR2 2) Determines PMSCD processing contents by using usb_pmsc_CheckCase13() or usb_pmsc_TransferMatrix().		
Notes			

**Table 4.18 usb\_pmsc\_UsbExecute()**

Name	Control PCD		
Call format	void usb_pmsc_UsbExecute(USBC_UTR_t *mess)		
Arguments	USBC_UTR_t	*mess	PMSDD execution result
Return values	void		
Description	Starts PCD based on the PMSDD storage command processing result. USBC_PMSC_CMD_COMPLETE → Create and transmit CSW (status OK) USBC_PMSC_CMD_FAILED → Create and transmit CSW (status FAIL) USBC_PMSC_CMD_CONTINUE: USBC_PMSC_PDATARCV → Data transfer start (OUT pipe) USBC_PMSC_PDATASND → Data transfer start (IN pipe) USBC_PMSC_CMD_ERROR:		
Notes			

**Table 4.19 usb\_pmsc\_CswSet()**

Name	Set CSW Information		
Call format	void usb_pmsc_CswSet(uint8_t ar_resp, uint32_t ul_size)		
Arguments	uint8_t	ar_resp	CSW status
	uint32_t	ul_size	Actual data length transmitted/received
Return values	void		
Description	Creates a CSW structure.		
Notes			

**Table 4.20 usb\_pmsc\_Error0()**

Name	Error Case 0 Processing		
Call format	void usb_pmsc_Error0(void)		
Arguments	void		
Return values	void		
Description	CBW receive state		
Notes			

**Table 4.21 usb\_pmsc\_Error1()**

Name	Error Case1 Processing		
Call format	void usb_pmsc_Error1(void)		
Arguments	void		
Return values	void		
Description	1) IN pipe stalled 2) Status FAIL, so transmit CSW		
Notes			

**Table 4.22 usb\_pmsc\_Error2()**

Name	Error Case2 Processing		
Call format	void usb_pmsc_Error2(void)		
Arguments	void		
Return values	void		
Description	1) OUT pipe stalled 2) Status FAIL, so transmit CSW		
Notes			

**Table 4.23 usb\_pmsc\_Error3()**

Name	Error Case 3 Processing		
Call format	void usb_pmsc_Error3(void)		
Arguments	void		
Return values	void		
Description	1) IN pipe stalled 2) Status FAIL, so transmit CSW		
Notes			

**Table 4.24 usb\_pmsc\_Error4()**

Name	Error Case 4 Processing		
Call format	void usb_pmsc_Error4(void)		
Arguments	void		
Return values	void		
Description	1) IN/OUT pipes stalled 2) Status FAIL, so transmit CSW		
Notes			

**Table 4.25 usb\_pmsc\_Error5()**

Name	Error Case 5 Processing		
Call format	void usb_pmsc_Error5(void)		
Arguments	void		
Return values	void		
Description	1) Status FAIL, so transmit CSW		
Notes			

Table 4.26 usb\_pmsc\_MassStrageReset()

Name	Notify of Mass Storage Reset Request		
Call format	void usb_pmsc_MassStrageReset(uint16_t value, uint16_t index, uint16_t length)		
Arguments	uint16_t	value	wValue of request
	uint16_t	index	wIndex of request
	uint16_t	length	wLength of request
Return values	void		
Description	Notifies PMSCD of a MassStorageReset request. 1) Sets PIPE0 to BUF. 2) Notifies of request reception. 3) Waits for PMSCD processing end.		
Notes			

Table 4.27 usb\_pmsc\_GetMaxLun()

Name	Notify of GetMaxLUN Request		
Call format	void usb_pmsc_GetMaxLUN (uint16_t value, uint16_t index, uint16_t length)		
Arguments	uint16_t	value	wValue of request
	uint16_t	index	wIndex of request
	uint16_t	length	wLength of request
Return values	void		
Description	Notifies PMSCD of a GetMaxLUN request. 1) Notifies of request reception. 2) Waits for PMSCD processing end.		
Notes			

## 4.7.2 PCI Functions

Table 4.28 R\_usb\_pmsc\_Open()

Name	Start Task		
Call format	USBC_ER_t R_usb_pmsc_Open(void)		
Arguments	void		
Return values	USBC_ER_t		
Description	Creates a task, mailbox, and memory pool, and starts the task. Sets the subclass command length based on the descriptor table.		
Notes			

Table 4.29 R\_usb\_pmsc\_Close()

Name	Release Task		
Call format	USBC_ER_t R_usb_pmsc_Close(void)		
Arguments	void		
Return values	USBC_ER_t		
Description	Ends a task, and releases the mailbox and memory pool.		
Notes			

Table 4.30 R\_usb\_pmsc\_MassStorageReset()

Name	Notify of MassStorageReset		
Call format	void R_usb_pmsc_MassStorageReset(USBC_CB_INFO_t complete)		
Arguments	USBC_CB_INFO_t	complete	Callback function at end processing
Return values	void		
Description	Notifies the PMSCD task that a MassStorageReset request was received.		
Notes			

Table 4.31 R\_usb\_pmsc\_GetMaxLun()

Name	Notify of GetMaxLUN		
Call format	void R_usb_pmsc_GetMaxLun(USBC_CB_INFO_t complete)		
Arguments	USBC_CB_INFO_t	complete	Callback function at end processing
Return values	void		
Description	Notifies the PMSCD task that a GetMaxLUN request was received.		
Notes			

Table 4.32 R\_usb\_pmsc\_SetConfig()

Name	SetConfiguration Request Processing		
Call format	void R_usb_pmsc_SetConfig(void)		
Arguments	void		
Return values	void		
Description	Initializes the internal status to enable CBW reception.		
Notes			

**Table 4.33 R\_usb\_pmsc\_SetInterface()**

Name	SetInterface Request Processing		
Call format	void R_usb_pmsc_SetInterface (uint16_t data1, uint16_t data2)		
Arguments	uint16_t	data1	
	uint16_t	data2	
Return values	void		
Description	Initializes the internal status to enable CBW reception.		
Notes			

**Table 4.34 R\_usb\_pmsc\_DescriptorChange()**

Name	Hi-Speed/Full-Speed Descriptor Table Switching		
Call format	void R_usb_pmsc_DescriptorChange(uint16_t mode , uint16_t data2)		
Arguments	uint16_t	mode	Reset handshake result
	uint16_t	data2	
Return values	void		
Description	Switches the descriptor table.		
Notes			

**Table 4.35 R\_usb\_pmsc\_DataTrans()**

Name	Issue Message for Data Transfer to PCD		
Call format	void R_usb_pmsc_DataTrans(uint16_t pipe, uint32_t size, uint8_t *table, USBC_CB_t complete)		
Arguments	uint16_t	pipe	Pipe number
	uint32_t	size	Data transfer size
	uint8_t	*table	Transfer data address
	USBC_CB_t	complete	Callback function
Return values	void		
Description	Issues a message to PCD.		
Notes			

**Table 4.36 R\_usb\_pmsc\_TransResult()**

Name	Callback Function at Data Transmit/Receive End		
Call format	void R_usb_pmsc_TransResult(USBC_UTR_t *mess)		
Arguments	USBC_UTR_t	*mess	Structure for data transfer
Return values	void		
Description	Notifies the PMSCD task by message of the data transmit/receive result.		
Notes			

**Table 4.37 R\_usb\_pmsc\_StallClearResult()**

Name	Callback Function at Stall Clear Processing End		
Call format	void R_usb_pmsc_StallClearResult(uint16_t data, uint16_t dummy)		
Arguments	uint16_t	data	Status
	uint16_t	dummy	Not used
Return values	void		
Description	Notifies the PMSCD task by message of stall clear end.		
Notes	Gets the message area (USB_PGET_BLK) and transmits a message.		

**Table 4.38 R\_usb\_pmsc\_ProcessWaitTmo()**

Name	Wait for PCD Task Processing End		
Call format	void R_usb_pmsc_ProcessWaitTmo(uint16_t tmo)		
Arguments	uint16_t	tmo	Timeout duration
Return values	void		
Description	Waits for the end of processing of a PCD task with timeout.		
Notes			

**Table 4.39 R\_usb\_pmsc\_ControlTrans0**

Name	Receive Class Request		
Call format	void R_usb_pmsc_ControlTrans0(USBC_REQUEST_t *req)		
Arguments	USBC_REQUEST_t	*req	Structure for class request
Return values	void		
Description	Dummy function for registration to PCD		
Notes			

**Table 4.40 R\_usb\_pmsc\_ControlTrans1**

Name	Receive Class Request		
Call format	void R_usb_pmsc_ControlTrans1(USBC_REQUEST_t *req)		
Arguments	USBC_REQUEST_t	*req	Structure for class request
Return values	void		
Description	Calls the usb_pmsc_GetMaxLun() function.		
Notes			

**Table 4.41 R\_usb\_pmsc\_ControlTrans2**

Name	Receive Class Request		
Call format	void R_usb_pmsc_ControlTrans2(USBC_REQUEST_t *req)		
Arguments	USBC_REQUEST_t	*req	Structure for class request
Return values	void		
Description	Dummy function for registration to PCD		
Notes			

**Table 4.42 R\_usb\_pmsc\_ControlTrans3**

Name	Receive Class Request		
Call format	void R_usb_pmsc_ControlTrans3(USBC_REQUEST_t *req)		
Arguments	USBC_REQUEST_t	*req	Structure for class request
Return values	void		
Description	Calls the usb_pmsc_MassStrageReset() function.		
Notes			

**Table 4.43 R\_usb\_pmsc\_ControlTrans4**

Name	Receive Class Request		
Call format	void R_usb_pmsc_ControlTrans4(USBC_REQUEST_t *req)		
Arguments	USBC_REQUEST_t	*req	Structure for class request
Return values	void		
Description	Dummy function for registration to PCD		
Notes			

Table 4.44 R\_usb\_pmsc\_ControlTrans5

Name	Receive Class Request		
Call format	void R_usb_pmsc_ControlTrans5(USBC_REQUEST_t *req)		
Arguments	USBC_REQUEST_t	*req	Structure for class request
Return values	void		
Description	Dummy function for registration to PCD		
Notes			

### 4.7.3 DDI Function

Table 4.45 usb\_pmsc\_AtapiTransResult()

Name	Callback Function at Storage Command Execution		
Call format	void usb_pmsc_AtapiTransResult(USBC_UTR_t *mess)		
Arguments	USBC_UTR_t	*mess	Structure for data transfer
Return values	void		
Description	This is a callback function for notifying by message of the execution result of a storage command. It is used as the callback function of usb_pmsc_SmpAtapiCommandExecute.		
Notes			

### 4.8 PMSCD Task Description

Table 4.46 lists the types of messages received by PMSCD. The operation sequence is as follows.

- \*0 When the task starts, PMSCD enables CBW reception (CBWRCV) by PCD.
- \*1 PMSCD receives a data (CBW) receive message from PCD.  
If the CBW was received correctly, data communication by PCD (DATARCV/DATASND/CSWSND) is enabled and usb\_gpmsc\_Seq is updated.
- \*2/4 PMSCD receives a data receive (transmit) message from PCD.  
The receive (transmit) data is transferred to PMSDD, and PMSCD waits for PMSDD processing to finish.
- \*3/5 PMSCD receives a data processing end message from PMSDD.  
If data communication will continue, data communication by PCD (DATARCV/DATASND) is enabled.  
If data communication will end, CSW transmission by PCD (CSWSND) is enabled and usb\_gpmsc\_Seq is updated.
- \*6 PMSCD receives a data (CSW) transmit message from PCD.  
If the CBW was transmitted correctly, CBW reception by PCD (CBWRCV) is enabled and usb\_gpmsc\_Seq is updated.

**Table 4.46 PMSCD Task Message Status Types**

usb_gpmsc_Seq	From	Message Status(mess->status)	Processing	usb_gpmsc_Seq	
USBC_PMSC_PCBWRCV (waiting to receive CBW)	PCD	USBC_DATA_OK	Analyze receive CBW data from Host.	USBC_PMSC_PDARCV /	
		USBC_DATA_SHT			
		USBC_DATA_OVR	Determine command.	USBC_PMSC_PDASND /	
		USBC_DATA_TMO	Do nothing.	USBC_PMSC_PCSWSND	
		USBC_DATA_STOP	Detect detach.		
		USBC_DATA_ERR	CBW receive error		
USBC_PMSC_PDARCV (data reception (OUT) in progress)	PCD	USBC_DATA_OK	Notify PMSDD of result.		
		USBC_DATA_SHT			
		USBC_DATA_OVR			
		USBC_DATA_TMO	Timeout error		
		USBC_DATA_STOP	Detect detach.		
		USBC_DATA_ERR	Data receive error		
	PMSDD	*3	USBC_PMSC_CMD_COMPLETE	Transmit OK CSW.	USBC_PMSC_PCSWSND
			USBC_PMSC_CMD_FAILED	Transmit FAIL CSW.	USBC_PMSC_PCSWSND
			USBC_PMSC_CMD_CONTINUE	Continue receiving data.	USBC_PMSC_PDARCV
			USBC_PMSC_CMD_ERROR	Error	
USBC_PMSC_PDASND (data transmission (IN) in progress)	PCD	USBC_DATA_NONE	Notify PMSDD of result.		
		USBC_DATA_TMO	Timeout error		
		USBC_DATA_STOP	Detect detach.		
		USBC_DATA_SHT	Data transmit error		
	PMSDD	*5	USBC_PMSC_CMD_COMPLETE	Transmit OK CSW.	USBC_PMSC_PCSWSND
			USBC_PMSC_CMD_FAILED	Transmit FAIL CSW.	USBC_PMSC_PCSWSND
			USBC_PMSC_CMD_CONTINUE	Continue transmitting data.	USBC_PMSC_PDASND
			USBC_PMSC_CMD_ERROR	Error	
USBC_PMSC_PCSWSND	PCD	USBC_DATA_NONE	Wait to receive CBW.	USBC_PMSC_PCBWRCV	
		USBC_DATA_TMO	Timeout error		

(CSW transmission in progress)	*6	USBC_DATA_STOP	Detect detach.
		USBC_DATA_SHT	CSW transmit error
		USBC_DATA_OVR DATA_ERR	
		USBC_DATA_OK	

\*1 to \*6 Match up to \*1 to 6 of " Figure 4.1 Command Sequence When No Transmit/Receive Data Present  
"to " Figure 4.3 Command Sequence When Receive (OUT) Data Present"

4.8.1 Storage Command With No Transmit/Receive Data

Figure 4.1 shows the sequence that occurs when a storage command with no transmit or receive data is received. PMSCD notifies PMSDD of the storage command. PMSDD executes the storage command and returns the processing result to PMSCD. PMSCD creates a CSW and transmits it to the host by using the `usb_pmesc_UsbExecute()` function.

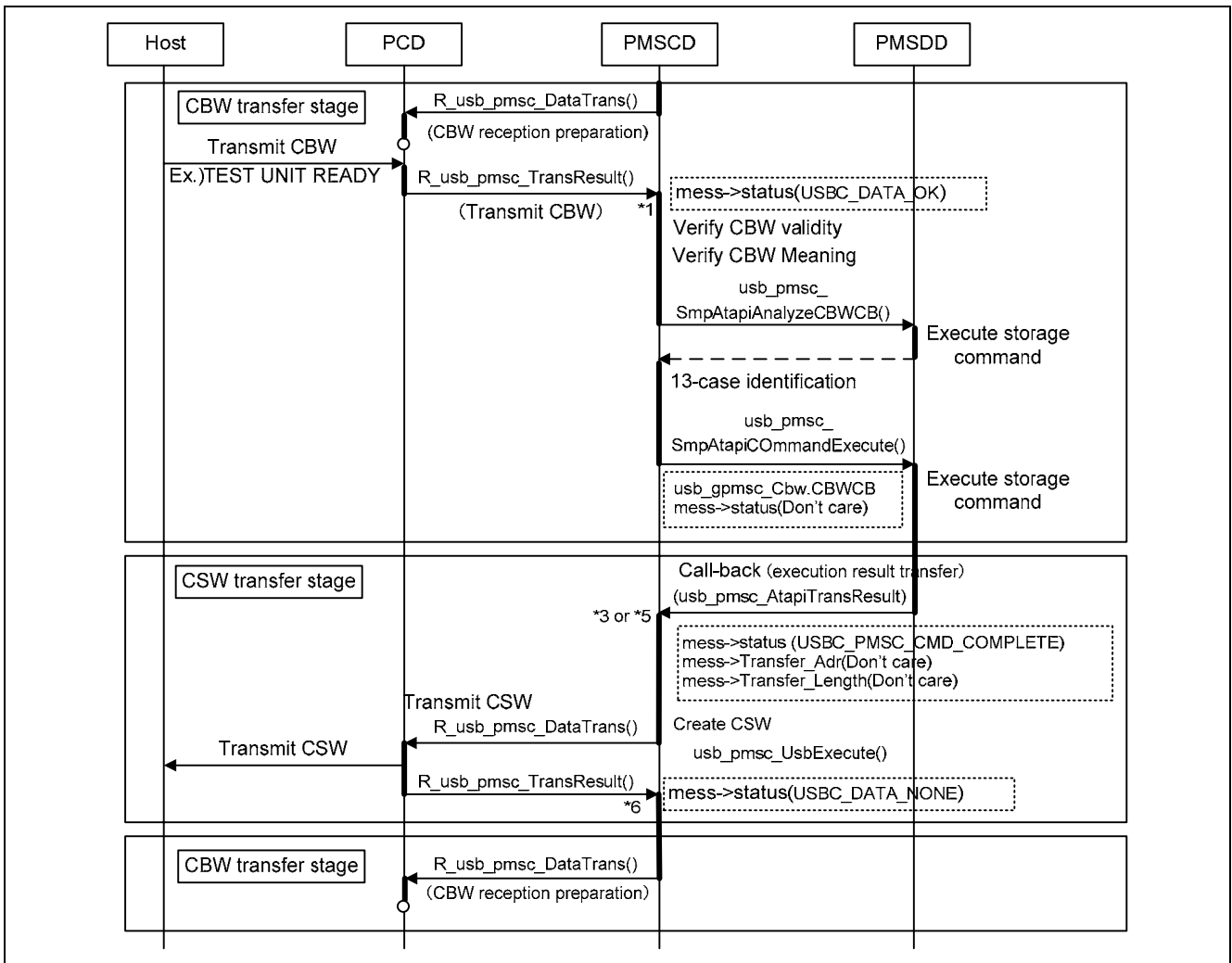


Figure 4.1 Command Sequence When No Transmit/Receive Data Present

4.8.2 Storage Command with Transmit (IN) Data

Figure 4.2 shows the sequence that occurs when a storage command with transmit data is received. PMSCD notifies PMSDD of the storage command. PMSDD returns the transmit data storage area and size to PMSCD. PMSCD notifies PCD of the transmit data and transmits the data to the host by using the `usb_pmesc_UsbExecute()` function. When PMSCD receives a transmit end notification from PCD, PMSCD sends a transfer end notification to PMSDD. If there is additional transmit data, the address and size are returned in the same manner as when a CBW is received, and data is transmitted to the host. When the end of the transmit data is reached, PMSDD returns the processing result to PMSCD. PMSCD creates a CSW and transmits it to the host by using the `usb_pmesc_UsbExecute()` function.

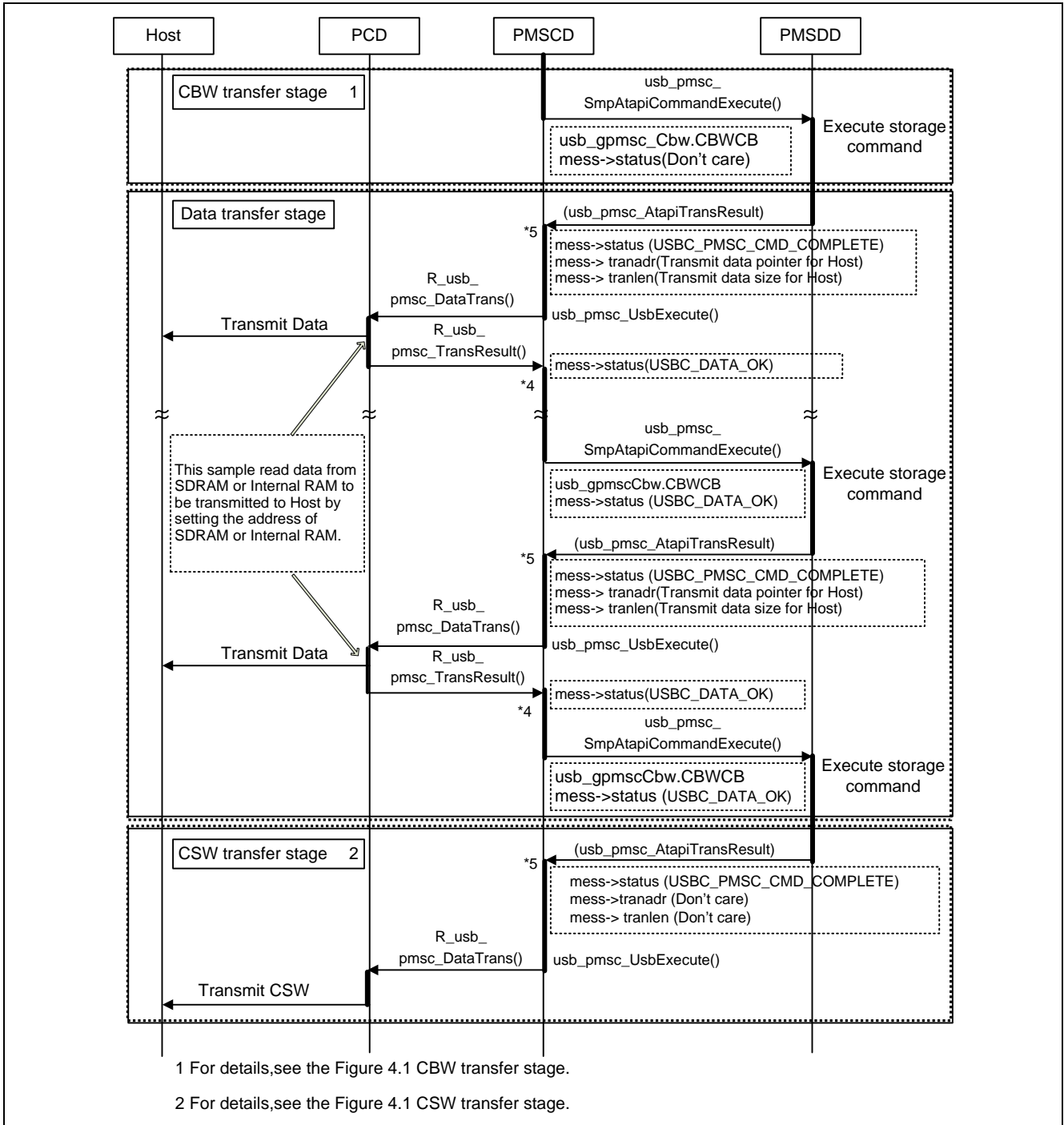


Figure 4.2 Command Sequence When Transmit (IN) Data Present

4.8.3 Storage Command with Receive (OUT) Data

Figure 4.3 shows the sequence that occurs when a storage command with receive data is received. PMSCD notifies PMSDD of the storage command. PMSDD returns the receive data storage area and size to PMSCD. PMSCD notifies PCD of the receive area and receives the data from the host by using the usb\_pmesc\_UsbExecute() function. When PMSCD receives a receive end notification from PCD, PMSCD sends a transfer end notification to PMSDD. If there is additional receive data, the address and size are returned in the same manner as when a CBW is received, and data is received from the host. When the end of the receive data is reached, PMSDD returns the processing result to PMSCD. PMSCD creates a CSW and transmits it to the host by using the usb\_pmesc\_UsbExecute() function.

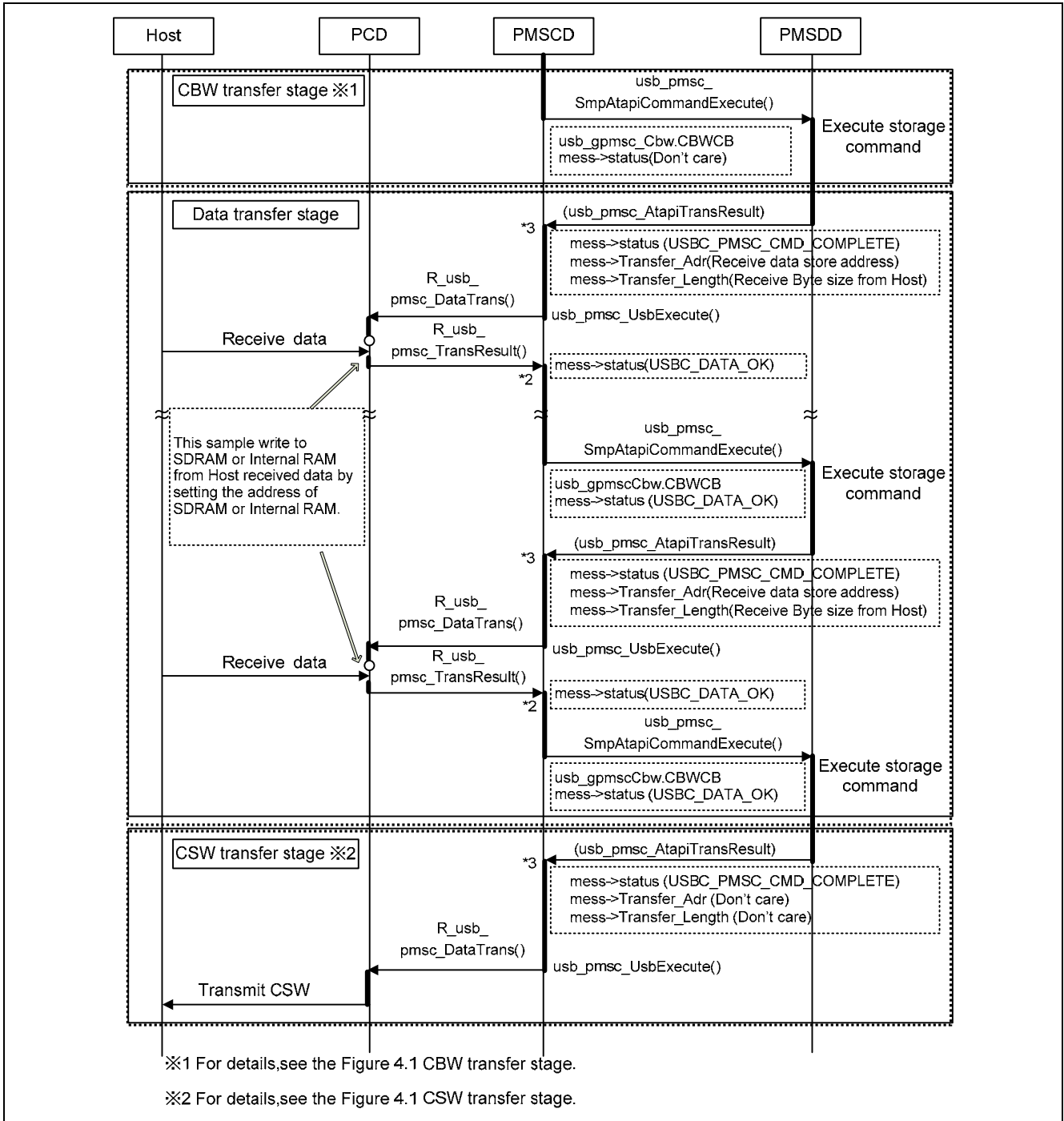


Figure 4.3 Command Sequence When Receive (OUT) Data Present

#### 4.9 PMSCD Registration

PMSCD must be registered with PCD to function as a device class driver.

Use the PeripheralRegistration() function to register PMSCD, using the sample functions as reference.

For details, refer to the Renesas USB Device USB Basic Firmware Application note.

#### 4.10 User Definition Tables

It is necessary to create a descriptor table and pipe information table for use by PCD.

Refer to the sample r\_usb\_PMSCdescriptor.c and r\_usb\_PMSCdefEp.h files when creating these tables.

For details, refer to the Renesas USB Device USB Basic Firmware Application note.

## 5. Peripheral Mass Storage Device Driver (PMSDD)

### 5.1 Basic Functions

The main functions of PMSDD are as follows.

1. Analysis of storage commands received from PMSCD
2. Execution of storage commands received from PMSCD
3. Notifying PMSCD of communication data information and execution results related to storage command execution

### 5.2 Overview of Functions

PMSDD analyzes storage commands it is notified of by PMSCD and performs command processing.

PMSDD has the following features.

1. Support for SFF-8070i (ATAPI).
2. Responds to the following storage commands.
  - READ10
  - INQUIRY
  - REQUEST\_SENSE
  - MODE\_SENSE10
  - READ\_FORMAT\_CAPACITY
  - READ\_CAPACITY
  - WRITE10
  - WRITE\_AND\_VERIFY
  - MODE\_SELECT10
  - FORMAT\_UNIT
  - TEST\_UNIT\_READY
  - START\_STOP\_UNIT
  - SEEK
  - VERIFY10
3. Divides the data transfer when the transfer data length exceeds the block count (user-specified).
4. A master boot recorder (FAT16) sample table is provided.

### 5.3 PMSDD Global Area

Table 5.1 lists details of the PMSDD global area.

**Table 5.1 PMSDD Global Area**

	Type	Variable Name	Description
1	USBC_PMSC_CDB_t	*usb_gpmc_AtapiCbwcb	Storage command sent from PMSCD
2	USBC_UTR_t	usb_gpmc_AtapiMess	Message sent from PMSCD
3	uint16_t	usb_gpmc_AtapiDataSize[]	Size of storage command response data
4	uint16_t	usb_gpmc_AtapiDataIdx[]	Index of storage command response data
5	uint8_t	usb_gpmc_AtapiReqIdx[]	Opcode of storage command
6	uint8_t	usb_gpmc_AtapiRdDataTbl[]	Storage command response data
7	uint8_t	usb_gpmc_AtapiSetupFlg	Command execution flag
8	uint32_t	usb_gpmc_AtapiFlashSize	Transfer data size
9	void	*usb_gpmc_AtapiFlashAdr	Transfer data start address
10	uint32_t	usb_gpmc_AtapiRealData	Actual total transfer data size

## 5.4 PMSDD Structure

USB\_PMSC\_CDB\_t is the storage command structure. The format of a storage command(SFF-8070i) differs depending on the command category, so a union is used. Four patterns sort out from ten kinds command types details are shown in table 5.2.

**Table 5.2 USBC\_PMSC\_CDB\_t Structure**

Union Member	Type	Structure Member	Bit Count	Command Category
s_usb_ptn0	uint8_t	uc_OpCode		Command determination (common)
	uint8_t	b_LUN	3	
	s_LUN	b_reserved	5	
	uint8_t	uc_data		
s_usb_ptn12	uint8_t	uc_OpCode		INQUIRY / REQUEST_SENSE
	uint8_t	b_LUN	3	
	s_LUN	b_reserved4	4	
		b_immed	1	
	uint8_t	uc_rsv2[2]		
	uint8_t	uc_Allocation		
	uint8_t	uc_rsv1[1]		
s_usb_ptn378	uint8_t	uc_OpCode		Not used (FORMAT UNIT)
	uint8_t	b_LUN	3	
	s_LUN	b_FmtData	1	
		b_CmpList	1	
		b_Defect	3	
	uint8_t	ul_LBA0		
	uint8_t	ul_LBA1		
	uint8_t	ul_LBA2		
	uint8_t	ul_LBA3		
	uint8_t	uc_rsv6[6]		
	s_usb_ptn4569	uint8_t	uc_OpCode	
uint8_t		b_LUN	3	
s_LUN		b_1	1	
		b_reserved2	2	
		b_ByteChk	1	
		b_SP	1	
uint8_t		ul_LogicalBlock0		
uint8_t		ul_LogicalBlock1		
uint8_t		ul_LogicalBlock2		
uint8_t		ul_LogicalBlock3		
uint8_t		uc_rsv1[1]		
uint8_t		us_Length_Hi		
uint8_t		us_Length_Lo		
uint8_t		uc_rsv3[3]		

## 5.5 PMSDD Constant Definitions

Table 5.3 lists the constant definitions of PMSDD.

**Table 5.3 PMSDD Global Area**

	Description	Definition	Value
1	Storage command analysis results		
	Command with no transfer data	USBC_ATAPI_NO_DATA	0x21
	Command with 1 byte of transmit data	USBC_ATAPI_A_SND_DATA	0x22
	Command with 1 byte of receive data	USBC_ATAPI_A_RCV_DATA	0x23
	Command with transmit data	USBC_ATAPI_SND_DATAS	0x24
	Command with receive data	USBC_ATAPI_RCV_DATAS	0x25
	Unsupported command	USBC_ATAPI_NOT_SUPPORT	0x26
2	Storage command codes		
		USBC_ATAPI_TEST_UNIT_READY	0x00
		USBC_ATAPI_REQUEST_SENSE	0x03
		USBC_ATAPI_FORMAT_UNIT	0x04
		USBC_ATAPI_INQUIRY	0x12
		USBC_ATAPI_START_STOP_UNIT	0x1B
		USBC_ATAPI_PREVENT_ALLOW_MEDIUM_REMOVAL	0x1E
		USBC_ATAPI_READ_FORMAT_CAPACITY	0x23
		USBC_ATAPI_READ_CAPACITY	0x25
		USBC_ATAPI_READ10	0x28
		USBC_ATAPI_WRITE10	0x2A
		USBC_ATAPI_SEEK	0x2B
		USBC_ATAPI_WRITE_AND_VERIFY	0x2E
		USBC_ATAPI_VERIFY10	0x2F
		USBC_ATAPI_MODE_SELECT	0x55
		USBC_ATAPI_MODE_SENSE	0x5A
3	Max. block count <sup>note1</sup>	USBC_ATAPI_BLK_MAXNUM	32768(16MB/512)
4	Logical unit size <sup>note1</sup>	USBC_ATAPI_BLOCK_UNIT	512
5	Data buffer size <sup>note1</sup>	USBC_ATAPI_TRANSFER_UNIT	256*USBC_ATAPI_BLOCK_UNIT

Note1. Change the above as appropriate to match the system under development.

## 5.6 List of PMSDD Functions

Table 5.4 lists the functions of PMSDD.

**Table 5.4 List of PMSDD Functions**

	Function Name	Description
1	usb_pmsc_SmpAtapiAnalyzeCbwCb	Analyzes storage command.
2	usb_pmsc_SmpAtapiTask	Main task of PMSDD
3	usb_pmsc_SmpAtapiGetReadData	Returns transmit data storage address and data size.
4	usb_pmsc_SmpAtapiGetReadMemory	Read data address and data size
5	usb_pmsc_SmpAtapiGetWriteMemory	Write data address and data size
6	usb_pmsc_SmpAtapiInitMedia	Initialization at PMSDD start
7	usb_pmsc_SmpAtapiCloseMedia	Processing at PMSDD end
8	usb_pmsc_SmpAtapiCommandExecute	Transmits message from PMSCD to PMSDD main task.

## 5.7 Description of PMSDD Functions

Tables 5.5 to 5.12 list details of functions used by PMSDD.

**Table 5.5 usb\_pmsc\_SmpAtapiAnalyzeCbwCb()**

Name	Analyze Storage Command		
Call format	void usb_pmsc_SmpAtapiAnalyzeCbwCb(uint8_t *cbwcb)		
Arguments	uint8_t	*cbwcb	Storage command portion of CBW
Return values	void		
Description	<p>Analyzes a storage command. The analysis result is returned as a global variable (usb_gpmsc_Message).</p> <ol style="list-style-type: none"> <li>Transfer data direction (usb_gpmsc_Message. ar_rst) <ul style="list-style-type: none"> <li>USBC_ATAPI_NO_DATA: No transfer data</li> <li>USBC_ATAPI_SND_DATAS: Transmit data present</li> <li>USBC_ATAPI_RCV_DATAS: Receive data present</li> <li>USBC_ATAPI_NOT_SUPPORT: Unsupported command</li> </ul> </li> <li>Transfer data size (usb_gpmsc_Message. ul_size) <ul style="list-style-type: none"> <li>The transfer data size is set in byte units.</li> </ul> </li> </ol>		
Notes	<p>The format of a storage command differs depending on the command category. PMSDD uses the USBC_MSC_CBW_t structure as a standard format. This also takes into account the possibility that the transfer data direction of the storage command may have a data size of 0.</p>		

**Table 5.6** usb\_pmsc\_SmpAtapiTask()

Name	PMSDD Task		
Call format	void usb_pmsc_SmpAtapiTask(void)		
Arguments	void		
Return values	void		
Description	<p>This function receives the storage command and data transfer results from PMSCD, and executes the storage command. It reports command execution result to PMSCD by means of a callback function.</p> <p>1) Command execution result (status)  USBC_PMSC_CMD_COMPLETE (execution end (success)) → PMSCD issues CSW (OK).  USBC_PMSC_CMD_FAILED (execution end (failure)) → PMSCD issues CSW (FAIL).  USBC_PMSC_CMD_CONTINUE (execution continue) → PMSCD implements data transfer.  USBC_PMSC_CMD_ERROR PMSDD (internal error) → Debug target</p> <p>2) For USBC_CMD_CONTINUE, transfer data information  (Transfer_Adr/Transfer_Length)  The data area is set in Transfer_Adr and the data size in Transfer_Length.</p>		
Notes			

**Table 5.7** usb\_pmsc\_SmpAtapiGetReadData()

Name	Data Transmit Command Response		
Call format	void usb_pmsc_SmpAtapiGetReadData(uint32_t *size , uint8_t **buff)		
Arguments	uint32_t	*size	Read data size pointer
	uint8_t	**buff	Read data start address pointer
Return values	void		
Description	<p>Sets the data area and size in response to the following commands.</p> <p>1) READ10  2) READ_FORMAT_CAPACITY  3) READ_CAPACITY  4) MODE_SENSE  5) REQUEST_SENSE  6) INQUIRY</p>		
Notes	For commands other than READ10, the response is set in usb_gpmsc_AtapiRdDataTbl[].		

**Table 5.8** usb\_pmsc\_SmpAtapiGetReadMemory()

Name	Storage Data Read Area Information Response		
Call format	void usb_pmsc_SmpAtapiGetReadMemory(uint32_t *size , uint8_t **buff)		
Arguments	uint32_t	*size	Read data size pointer
	uint8_t	**buff	Read data start address pointer
Return values	void		
Description	Returns the data area and data size for the READ10 storage command.		
Notes			

**Table 5.9** usb\_pmsc\_SmpAtapiGetWriteMemory()

Name	Storage Data Write Area Information Response		
Call format	void usb_pmsc_SmpAtapiGetWriteMemory(uint32_t *size , uint8_t **buff)		
Arguments	uint32_t	*size	Write data size pointer
	uint8_t	**buff	Write data start address pointer
Return values	void		
Description	Returns the data area and data size for the WRITE10 and WRITE_AND_VERIFY storage commands.		
Notes	No VERIFY processing is performed for the WRITE_AND_VERIFY command.		

**Table 5.10** usb\_pmsc\_SmpAtapiInitMedia()

Name	Start PMSDD		
Call format	USBC_ER_t usb_pmsc_SmpAtapiInitMedia(void)		
Arguments	USBC_ER_t		
Return values	void		
Description	Creates the PMSDD task, mailbox, and memory pool, and starts the task.		
Notes			

**Table 5.11** usb\_pmsc\_SmpAtapiCloseMedia()

Name	End PMSDD		
Call format	USBC_ER_t usb_pmsc_SmpAtapiCloseMedia(void)		
Arguments	USBC_ER_t		
Return values	void		
Description	Ends PMSDD task, and releases the mailbox and memory pool.		
Notes			

**Table 5.12** usb\_pmsc\_SmpAtapiCommandExecute()

Name	Notify PMSDD of Storage Command		
Call format	void usb_pmsc_SmpAtapiCommandExecute(uint8_t *cbw, uint16_t status, USBC_CB_t complete)		
Arguments	uint8_t	*cbw	CBW
	uint16_t	status	Data transfer result
	USBC_CB_t	complete	Callback function processing end
Return values	void		
Description	PMSCD notifies PMSDD of the command execution.		
Notes			

## 5.8 PMSDD Task Description

PMSDD receives a storage command and data transfer result from PMSCD, and executes the storage command. Table 5.13 lists the command processing of PMSDD. When the transfer data size exceeds USB\_ATAPI\_TRANSFER\_UNIT, the data is divided into smaller units and transferred. In addition, the transmit data is created from the response data tables (\*1 usb\_gpmsc\_AtapiDataSize[], usb\_gpmsc\_AtapiDataIndx[], usb\_gpmsc\_AtapiReqIndx[], usb\_gpmsc\_AtapiRdDataTbl[]) for data transmit commands that do not involve memory access.

\*1 The response data table is composed by the value of storage command set SFF-8070i, and the index for the table reference is decided according to the command (Refer to Uc\_OpCode in Table 5.2 USBC\_PMSC\_CDB\_t structure) provided in the subclass.

**Table 5.13 Processing Examples for Each Storage Command**

Storage command	Corresponding Function	Description
READ10	usb_pmsc_SmpAtapiGetReadMemory()	Gets start address and size.
INQUIRY	usb_pmsc_SmpAtapiGetReadData()	Selects from array usb_gpmsc_AtapiRdDataTbl.
REQUEST_SENSE	usb_pmsc_SmpAtapiGetReadData()	Selects from array usb_gpmsc_AtapiRdDataTbl.
MODE_SENSE	usb_pmsc_SmpAtapiGetReadData()	Selects from array usb_gpmsc_AtapiRdDataTbl.
READ_FORMAT_CAPACITY	usb_pmsc_SmpAtapiGetReadData()	Selects from array usb_gpmsc_AtapiRdDataTbl.
READ_CAPACITY	usb_pmsc_SmpAtapiGetReadData()	Selects from array usb_gpmsc_AtapiRdDataTbl.
WRITE10	usb_pmsc_SmpAtapiGetWriteMemory()	Gets start address and size.
WRITE_AND_VERIFY	usb_pmsc_SmpAtapiGetWriteMemory()	Gets start address and size.
MODE_SELECT10	usb_pmsc_SmpAtapiGetWriteMemory()	Gets start address and size.
FORMAT_UNIT	usb_pmsc_SmpAtapiGetWriteMemory()	Gets start address and size.
TEST_UNIT_READY	usb_pmsc_SmpAtapiTask()	Status = USBC_PMSC_CMD_COMPLETE
START_STOP_UNIT	usb_pmsc_SmpAtapiTask()	Status = USBC_PMSC_CMD_COMPLETE
SEEK10	usb_pmsc_SmpAtapiTask()	Status = USBC_PMSC_CMD_COMPLETE
VERIFY10	usb_pmsc_SmpAtapiTask()	Status = USBC_PMSC_CMD_COMPLETE
PREVENT_ALLOW_MEDIUM_REMOVAL	usb_pmsc_SmpAtapiTask()	Status = USBC_PMSC_CMD_FAILED
else	usb_pmsc_SmpAtapiTask()	Status = USBC_PMSC_CMD_ERROR

## 6. Limitations

The following limitations apply to PMSC.

1. Structures are composed of members of different types.  
(Depending on the compiler, the address alignment of the structure members may be shifted.)
2. The media area of the sample as configured as FAT16.

**Website and Support**

Renesas Electronics Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/inquiry>

All trademarks and registered trademarks are the property of their respective owners.

## Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Dec.10.2010	—	First edition issued
1.10	Sep.1.2011	—	Add Target device R8A66597 and RX630
		2	Swap 1.3 Related Documents and 1.4 Terms and Abbreviations
		3	Update 2.1 Module Configuration
		5	Add 2.4.2 System Resource Definitions for NonOS Add 2.4.3 USB Interrupt Handler Definition
		6	Update BOT Protocol Overview Add Figure 3.1 BOT protocol Overview
		7	Update 3.2.1 Sequence of Storage Command for No Transmit/Receive Data Update Figure 3.1 Sequence of Storage Command for No Transmit/Receive Data
		8	Update 3.2.2 Sequence of Storage Command for Transmit (IN) Data Update Figure 3.2 Sequence of Storage Command for Transmit (IN) Data
		9	Update 3.2.3 Sequence of Storage Command for Receive (OUT) Data Update Figure 3.3 Sequence of Storage Command for Receive (OUT) Data
		27	Move 4.8.1 Storage Command With No Transmit/Receive Data ( 4.8.3 4.8.1) Update Figure 4.1 Command Sequence When No Transmit/Receive Data Present
		28	Move 4.8.2 Storage Command with Transmit (IN) Data ( 4.8.1 4.8.2) Update Figure 4.2 Command Sequence When Transmit (IN) Data Present
		29	Move 4.8.3 Storage Command with Receive (OUT) Data ( 4.8.2 4.8.3) Update Figure 4.3 Command Sequence When Receive (OUT) Data Present
		32	Update PMSDD Structure
		37	Update PMSDD Task Description

## General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

### 1. Handling of Unused Pins

Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

- The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

### 2. Processing at Power-on

The state of the product is undefined at the moment when power is supplied.

- The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.

In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.

In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

### 3. Prohibition of Access to Reserved Addresses

Access to reserved addresses is prohibited.

- The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

### 4. Clock Signals

After applying a reset, only release the reset line after the operating clock signal has become stable.

When switching the clock signal during program execution, wait until the target clock signal has stabilized.

- When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

### 5. Differences between Products

Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.

- The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

## Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
  2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
  3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
  4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
  5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
  6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
  7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.  
"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.  
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.  
"Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
  8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
  9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
  10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
  11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
  12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.
- (Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.  
(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.



### SALES OFFICES

Renesas Electronics Corporation

<http://www.renesas.com>

Refer to "<http://www.renesas.com/>" for the latest and detailed information.

#### Renesas Electronics America Inc.

2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.  
Tel: +1-408-588-6000, Fax: +1-408-588-6130

#### Renesas Electronics Canada Limited

1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada  
Tel: +1-905-898-5441, Fax: +1-905-898-3220

#### Renesas Electronics Europe Limited

Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K  
Tel: +44-1628-585-100, Fax: +44-1628-585-900

#### Renesas Electronics Europe GmbH

Arcadiastrasse 10, 40472 Düsseldorf, Germany  
Tel: +49-211-65030, Fax: +49-211-6503-1327

#### Renesas Electronics (China) Co., Ltd.

7th Floor, Quantum Plaza, No.27 ZhichunLu Haidian District, Beijing 100083, P.R.China  
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

#### Renesas Electronics (Shanghai) Co., Ltd.

Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China  
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

#### Renesas Electronics Hong Kong Limited

Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong  
Tel: +852-2886-9318, Fax: +852-2886-9022/9044

#### Renesas Electronics Taiwan Co., Ltd.

7F, No. 363 Fu Shing North Road Taipei, Taiwan  
Tel: +886-2-8175-9600, Fax: +886-2-8175-9670

#### Renesas Electronics Singapore Pte. Ltd.

1 HarbourFront Avenue, #06-10, Keppel Bay Tower, Singapore 098632  
Tel: +65-6213-0200, Fax: +65-6278-8001

#### Renesas Electronics Malaysia Sdn.Bhd.

Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia  
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

#### Renesas Electronics Korea Co., Ltd.

11F., Samik Laviel' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea  
Tel: +82-2-558-3737, Fax: +82-2-558-5141