SH7286 Group

USB Function Module: USB Mass Storage Class

R01AN0063EJ0100
Rev. 1.00
Oct. 22, 2010

Summary

This application note describes how to use the SH7286 USB function module and shows an example to create the firmware which is compliant to the USB mass storage class specification.

This document and the sample program described are examples of the USB function module, and are therefore not guaranteed by Renesas.

Target Device

SH7286 MCU

Contents

1. Introduction

1.1 Specifications

This document describes how to use the SH7286 USB function module, and how to create the firmware which is compliant to the USB Mass Storage Class specification.

1.2 Modules Used

- Interrupt Controller (INTC)
- Pin Function Controller (PFC)
- USB Function Module (USB)

1.3 Applicable Conditions

MCU	SH7286
Operating Frequency	Internal clock: 100 MHz Bus clock: 50 MHz Peripheral clock: 50 MHz
Integrated Development Environment	Renesas Electronics High-performance Embedded Workshop Ver.4.04.01
C Compiler	Renesas Electronics SuperH RISC engine Family C/C++ compiler package Ver.9.01 Release 01
Compiler Options	Default setting in the High-performance Embedded Workshop (-cpu=sh2a -object="$(CONFIGDIR)¥$(FILELEAF).obj" -debug -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 -nologo)

1.4 Related Application Notes

For more information, refer to the following application notes:

- SH7285 Group Example of Initialization
- SH7285/SH7286 Group USB function module USB to Serial Conversion Application Note

1.5 About Active-low Pins (Signals)

The symbol "#" suffixed to the pin (or signal) names indicates that the pins (or signals) are active-low.

2. Applications

This sample program uses the USB function module (USB) to execute the control IN, control OUT, bulk IN, and bulk OUT transfers. This sample program also processes USB mass storage class commands.

The SH7286 USB function module includes the following features:

- USB protocol processed automatically
- USB standard request to endpoint 0 processed automatically (some requests need to be processed by the firmware)
- Data rate: Full-speed
- Interrupt request: Generates various interrupt signals required for USB communication
- Clocks:
  - External clock (48 MHz)
  - Internal clock (enabled only when EXTAL 12 MHz is selected)
- Power-down Mode
  When a USB cable is not connected, less power is consumed by stopping the UDC internal clock
- Endpoint configuration listed in Table 1

Table 1 Endpoint Configuration

Endpoint Number	Name	Transfer Mode	Maximum Packet Size	FIFO Buffer Capacity	DMA/DTC transfer
Endpoint 0	EP0s	Setup	8 bytes	8 bytes	–
	EP0i	Control IN	8 bytes	8 bytes	–
	EP0o	Control OUT	8 bytes	8 bytes	–
Endpoint 1	EP1	Bulk OUT	64 bytes	64 × 2 (128) bytes	Available
Endpoint 2	EP2	Bulk IN	64 bytes	64 × 2 (128) bytes	Available
Endpoint 3	EP3	Interrupt	8 bytes	8 bytes	–

Figure 1 shows the system configuration.

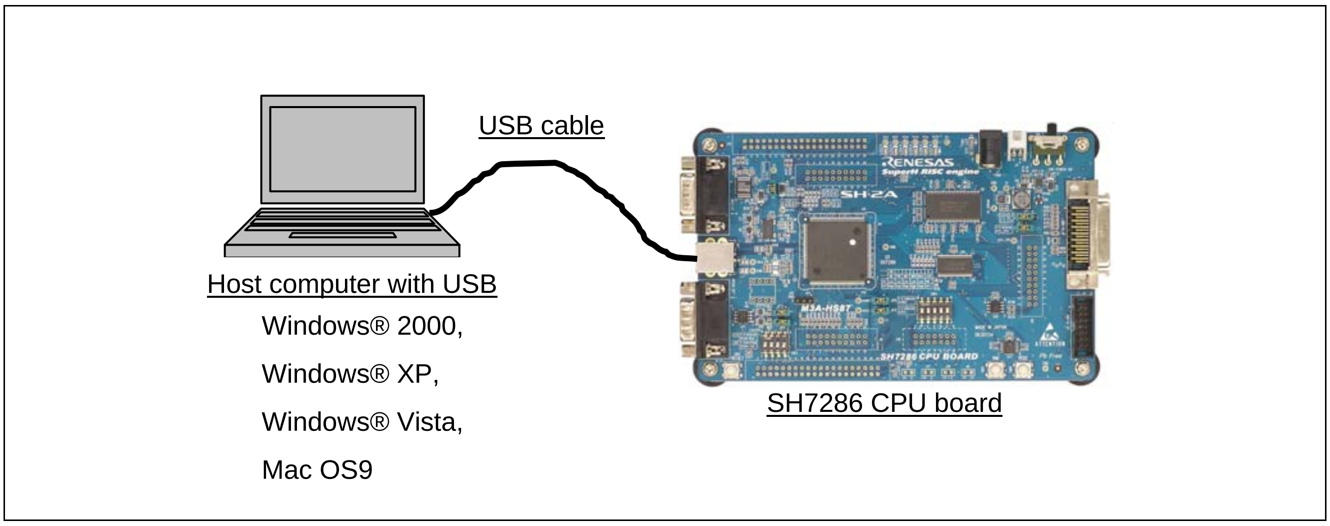

Figure 1 System Configuration

This system includes Renesas Electronics SH7286 CPU board with the SH7286 MCU, and the Windows® 2000, Windows® XP, Windows® Vista, or Mac OS9-based host computer.

The host computer and the SH7286 CPU board are connected with USB, an SDRAM on the SH7286 CPU board is used as the RAM disk to allow the host computer to write data to the SDRAM on the SH7286 CPU board, and to read data from the SDRAM.

The USB mass storage class (Bulk-only Transport) device driver can be used, which comes standard with the above-mentioned operating system.

This system includes the following features:

1. The sample program can be used to evaluate the SH7286 USB module
2. The sample program supports USB control transfer and bulk transfer
3. E10A-USB emulator can be used to debug the system
4. Create additional programs to support USB interrupt transfer

Note: Programs for USB interrupt transfer must be created by user. Note that the SH7286 does not support isochronous transfer.

3. USB Mass Storage Class (Bulk-Only Transport) Overview

This chapter describes the USB Mass Storage Class (Bulk-Only Transport). Use this guide as the reference when you develop the USB storage systems. For more information on the USB specifications, refer to (3) and (4) in 6 References.

3.1 USB Mass Storage Class

The USB Mass Storage Class is a USB protocol to read from or write data to a "mass storage" device which is connected to a host computer.

To notify the host computer that the module is a mass storage class-compliant function, set H'08 in the bInterfaceClass field of the Interface Descriptor. As a USB function module must notify the serial number to the host by the string descriptor in USB Mass Storage Class, this sample program returns 000000000001 by Unicode.

To transfer data between the host computer and a function module, use one of the four transfer types (Control transfer, bulk transfer, interrupt transfer, and isochronous transfer). The protocol code defines how to use these transfer types.

USB Mass Storage Class has two data transfer protocols:

- USB Mass Storage Class Bulk-Only Transport
- USB Mass Storage Class Control/Bulk/Interrupt (CBI) Transport

USB Mass Storage Class Bulk-Only Transport uses only Bulk transfer.

USB Mass Storage Class Control/Bulk/Interrupt (CBI) Transport uses control transfer, bulk transfer, and interrupt transfer. CBI Transport divided into two data protocols, one that uses interrupt transfer, and one that does not use interrupt transfer.

The sample program uses USB Mass Storage Class Bulk-Only Transport as its data transport protocol.

When the host computer uses a device to load or save data, the host computer transmits instructions (commands) to the USB function. The USB function can load or save data by executing the commands which are transmitted from the host computer. Commands from the host computer to the function are defined as Subclass codes.

3.2 Subclass Codes

Subclass codes indicate the command formats which are transmitted from the host computer to the function. There are 7 commands formats, and Subclass codes are defined as shown in Table 2.

Table 2 Subclass Codes

Subclass Codes	Command Block Specifications
H'01	Reduced Block Commands (RBC), T10/1240-D
H'02	Attachment Packet Interface (ATAPI) for CD-ROMS, SFF-8020i, Multi-Media Command Set 2 (MMC-2)
H'03	Attachment Packet Interface (ATAPI) for Tape, QIC-157
H'04	USB Mass Storage Class UFI Command Specification
H'05	Attachment Packet Interface (ATAPI) for Floppies, SFF-8070i
H'06	SCSI Primary Commands-2 (SPC-2), Revision 3 or later

To notify the command format supported by the mass storage device to the host computer, describe the Subclass code in the bInterfaceSubClass field of the Interface Descriptor.

This sample program uses H'06 SCSI Primary Commands.

3.3 Bulk-Only Transport

Bulk-Only Transport supports bulk transfer only; transfers data between the host computer and the function.

Bulk transfer has two different transfer types depending on the direction of the data transfer. Bulk IN transfer is to transfer data from the host controller to the function, and bulk OUT transfer is to transfer data from the function to the host controller.

Bulk-Only Transport specifies a combination of bulk OUT transfer and bulk IN transfer in advance to transfer data between the host and the function. Bulk-Only Transport always uses the combination of bulk transfers as shown in Figure 2. These bulk transfers have different roles, and are controlled as “stages (transports)”.

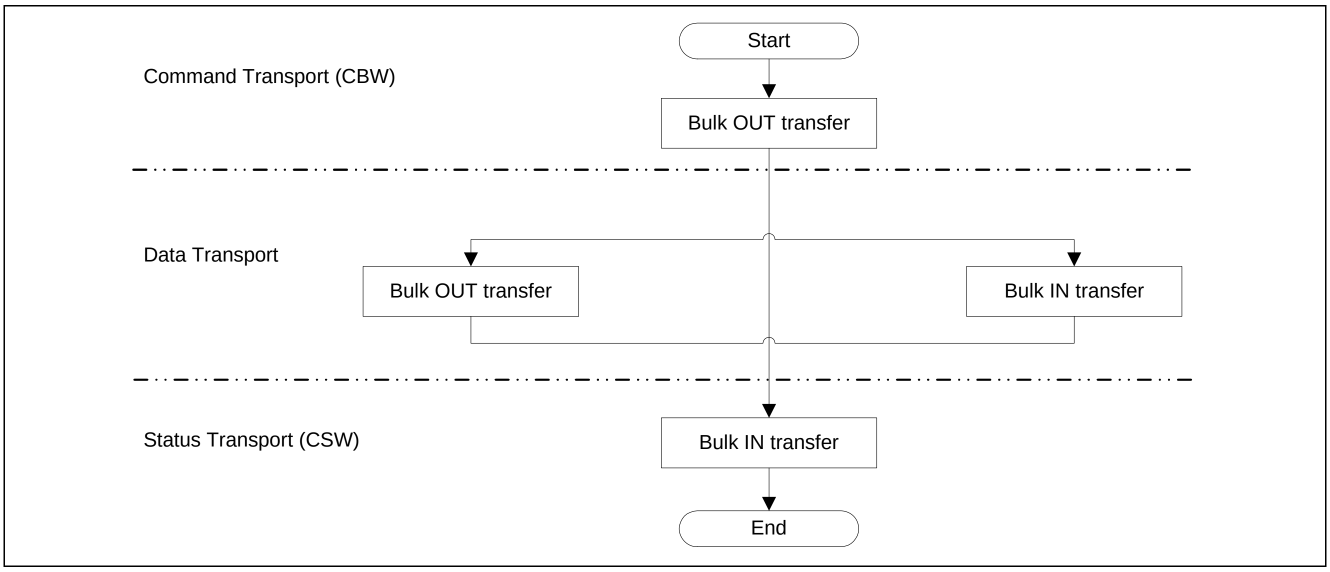

Figure 2 Relationship between the Transfer Type and Transport

To notify the host computer to use the Bulk-Only Transport protocol, describe H'50 in the bInterfaceProtocol field of the Interface descriptor.

3.3.1 Command Transport

In command transport, the host sends a command, which is defined as the Command Block Wrapper (CBW) to the function via the Bulk-Out transfer. The Bulk-Only Transport always starts with the CBW. The host sends the CBW with a 31-byte packet via the Bulk-Out transfer. The format of the CBW is listed in Table 3.

Table 3 Command Transport Format

	7	6	5	4	3	2	1	0
H'00 to H'03	dCBWSignature							
H'04 to H'07	dCBWTag							
H'08 to H'0B	dCBWDataTransferLength							
H'0C	bmCBWFlags							
H'0D	Reserved (0)				bCBWLUN			
H'0E	Reserved (0)			bCBWCBLength				
H'0F to H'1E	CBWCB							

- dCBWSignature:
  Signature that helps identify this data packet as a CBW. The signature field shall contain the value H'43425355 (little endian).
- dCBWTag:
  A Command Block Tag sent by the host. The dCSWTag is specified by the host, and it associates a CSW with the corresponding CBW.
- dCBWDataTransferLength:
  The number of bytes of data that the host expects to transfer on the data transport. If this field is zero, the data transport does not exist.
- bmCBWFlags:
  When bit 7 of this field is 0, data is transferred (data transport) via the bulk-OUT transfer, from the host to the function. When bit 7 is 1, data is transferred via the bulk-IN transfer, from the function to the host. Bits 6 to 0 are fixed to 0.
- bCBWLUN:
  The device Logical Unit Number (LUN) to which the command block is being sent.
- bCBWCBLength:
  The valid length of the CBWCB in bytes.
- CBWCB:
  The command block to be executed by the function. The CBWCB stores the command that the host computer expects to execute (SCSI commands in this sample program).

3.3.2 Status Transport

In status transport, the function sends the status of the execution of the command block to the host computer via bulk-IN transfer. The status packet is defined as the Command Status Wrapper (CSW). Bulk-Only Transport always ends with the CSW. The function sends the CSW with a 13-byte packet via the Bulk-IN transfer. The format of the CSW is listed in Table 4.

Table 4 Status Transport Format

	7	6	5	4	3	2	1	0
H'0 to H'3	dCSWSignature							
H'4 to H'7	dCSWTag							
H'8 to H'B	dCSWDataResidue							
H'C	bCSWStatus							

- dCSWSignature:
  Signature that helps identify this data packet as a CSW. The signature field contains the value H'53425355 (little endian).
- dCSWTag:
  A Command Block Tag. The function sets this field to the value received in the dCBWTag of the associated CBW.
- dCSWDataResidue:
  The function shall report in the dCSWDataResidue the difference between the amount of data expected as stated in the dCBWDataTransferLength, and the actual amount of data processed by the function.
- bCSWStatus:
  bCSWStatus indicates the success or failure of the command. The function sets this byte to H'00 if the command completed successfully. A non-zero value shall indicate a failure during command execution; H'01 indicates "Command failed", and H'02 indicates "Phase error".

3.3.3 Data Transport

Data transport defines the data transfer between the host and the function. When executing Read/Write command in data transport, the host sends the exact number of bytes of data in sectors to the function, and vice versa.

Data transport consists of multiple bus transactions.

Data transfer in Data transport is either Bulk-Out transfer or Bulk-IN transfer, which is specified by the bmCBWFlags of the CBW data.

(1) Data transport (Bulk-OUT transfer)
When bit 7 in bmCBWFlags field of the CBW data is 0, and dCBWDataTransferLength field of the CBW data is not 0, Data transport is specified as Bulk-OUT transfer.
The function receives the amount of data expected as stated in dCBWDataTransferLength field to the host computer, and receives the data which is required to execute the SCSI commands specified in CBWCB field of the CBW data.

(2) Data transport (Bulk-IN transfer)
When bit 7 in bmCBWFlags field of the CBW data is 1, and dCBWDataTransferLength field of the CBW data is not 0, Data transport is specified as Bulk-IN transfer.
The function sends the amount of data expected as stated in dCBWDataTransferLength field to the host computer, and sends the execution result of the SCSI commands specified in CBWCB field of the CBW data.

3.4 Class-specific Requests

Class-specific requests are defined as a group of devices by USB class specification. These requests are made using control transfer.

To use the USB Mass Storage Class Bulk-Only Transport as the data transfer protocol, the following commands listed in Table 5 must be supported.

Table 5 Class-specific Requests

bRequest Field	Request Name	Description
255 (H'FF)	Bulk-Only Mass Storage Reset	Resets the interface
254 (H'FE)	Get Max LUN	Determines the number of logical units

When the function receives the Bulk-Only Mass Storage Reset request, it resets all the interfaces used by the USB Mass Storage Class Bulk-Only Transport.

When the function receives the Get Max LUN request, it returns the maximum number of logical units available. As the number of logical units in this sample program is 1, the function returns "0" to the host.

3.5 Subclass Codes (SCSI Transparent Command Set)

The function must support subclass commands in CBW which are sent from the host computer, and handle commands.

This sample program supports 11 SCSI commands listed in Table 6. When the function receives commands which are not supported, it reports to the host computer "Command failed" using CSW.

Table 6 Supported Commands

Operation Code	Command Name	Description
H'00	TEST UNIT READY	Checks if a logical unit is ready
H'03	REQUEST SENSE	After an error occurred in the previous command block, it provides the information about the error to the host
H'12	INQUIRY	Reports the information about the drive to the host
H'1A	MODE SENSE (6)	Reports the state of the drive to the host
H'1B	START/STOP UNIT	Controls to attach or remove the media device
H'1E	PREVENT ALLOW MEDIUM REMOVAL	Prevents or allows the removal of media
H'23	READ FORMAT CAPACITIES	Reports the format information of the media
H'25	READ CAPACITY	Reports the information about sectors in the media to the host
H'28	READ (10)	Reads the specified number of sectors of data from the read sector specified
H'2A	WRITE (10)	Writes the specified number of sectors of data to the write sector specified
H'2F	VERIFY (10)	Checks if the data on the media can be accessed

4. Development Environment

Following devices (tools) are used to develop this system:

- SH7286 CPU board (Part number: M3A-HS87), Renesas Electronics
- E10A-USB Emulator, Renesas Electronics
- E10A-USB computer (Windows® 2000, Windows® XP)
- USB host computer (Windows® 2000, Windows® XP, Windows® Vista, Mac OS9)
- USB cable
- High-performance Embedded Workshop 4, Renesas Electronics

4.1 Hardware Requirements

Figure 3 shows the connection diagram.

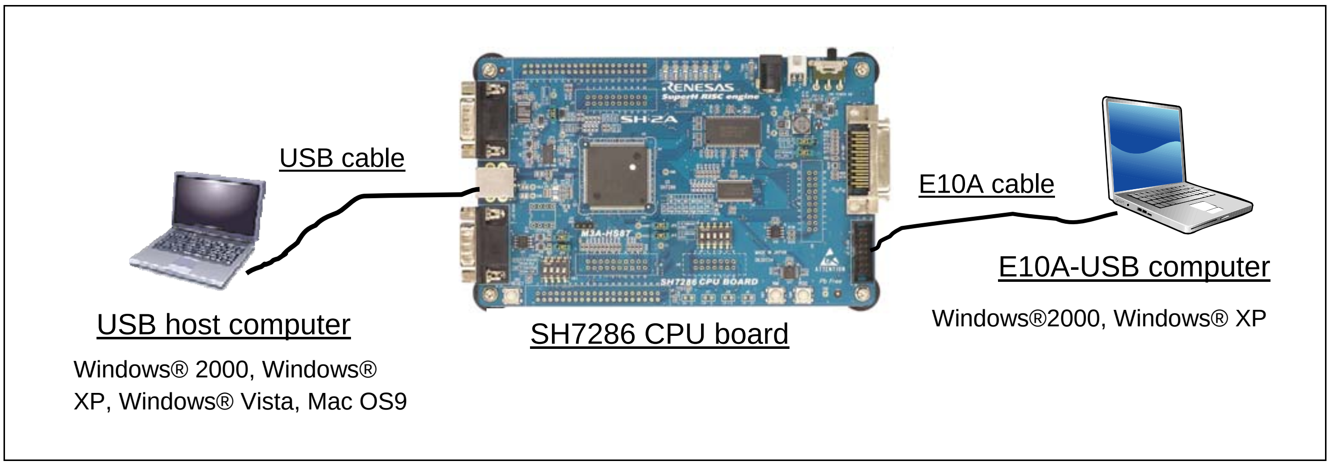
USB cable
E10A cable
E10A-USB computer
Windows®2000, Windows® XP
USB host computer
SH7286 CPU board
Windows® 2000, Windows® XP, Windows® Vista, Mac OS9

Figure 3 Device Connection Diagram

(1) SH7286 CPU board

This system uses the E10A-USB emulator. To use the E10A-USB emulator on the SH7286 CPU board, set the DIP switches (SW4) as listed in Table 7. Make sure to turn the power OFF before changing the SW4 setting.

Table 7 DIP Switches Setting (SW4)

Setting (Mode 6)	Description
SW4-1 (FWE): OFF	Write-/erase-protect the internal flash memory
SW4-2 (MD1): OFF	MD1 pin state
SW4-3 (MD0): ON	MD0 pin state

(2) USB host computer

Use Windows 2000-, Windows XP-, Windows Vista-, or Mac OS9-based computer with the USB port as the USB host computer. As this system uses the USB Mass Storage Class (Bulk-Only Transport) device driver which comes standard with the operating system, user does not have to install a driver newly.

(3) E10A-USB computer

Use Windows 2000-, Windows-XP-, Windows Vista-, or Mac OS9-based computer with the USB port as the E10A-USB computer. Use a USB cable to connect the E10A-USB emulator and the USB connector of the E10A-USB computer, and use an E10A cable to connect the E10A-USB emulator and the CPU board. After the connection is established, activate the High-performance Embedded Workshop 4 to emulate with the function.

4.2 Software Requirements

Use the High-performance Embedded Workshop 4 to compile, link, and debug the sample program. Double-click "sh7286_usb_msc.hws" to activate the High-performance Embedded Workshop 4.

4.2.1 Sample Program

All sample programs are stored in "sh7286_usb_msc" folder. Copy this folder to the computer where the High-performance Embedded Workshop 4 is installed to use the sample program.

Figure 4 shows files stored in "sh7286_usb_msc" folder.

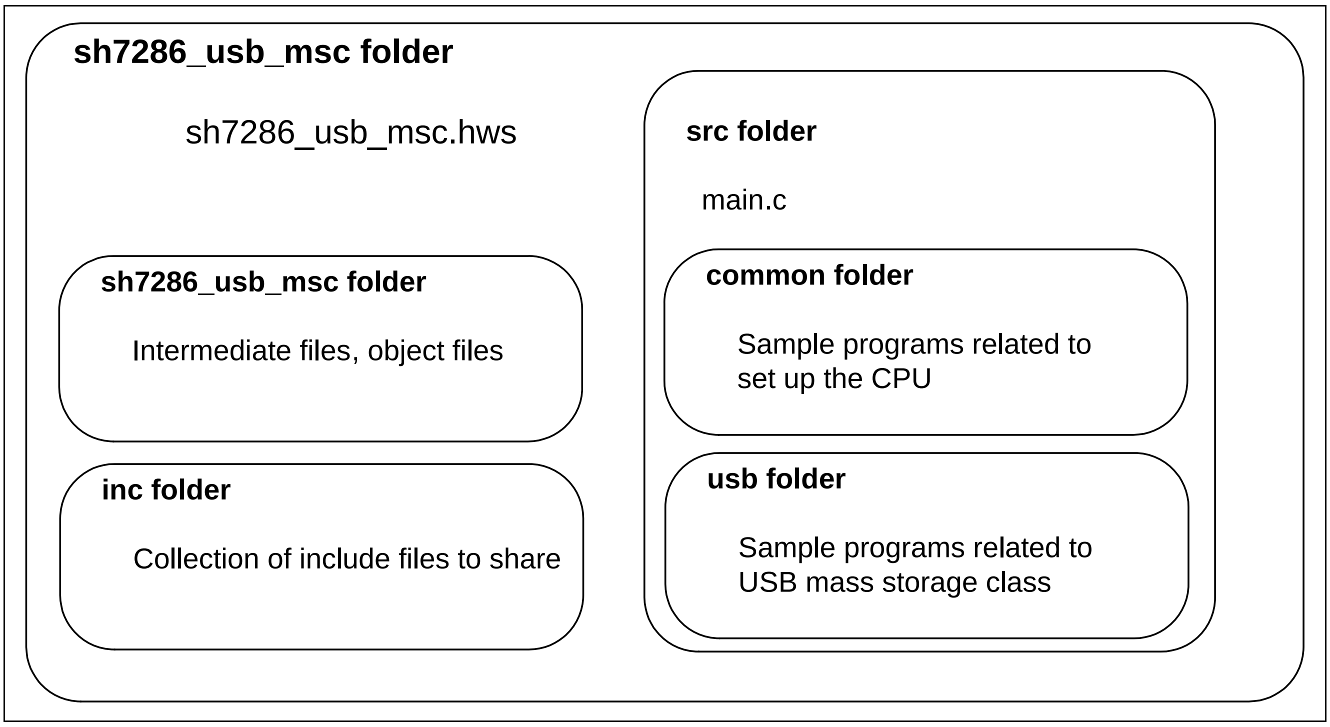
sh7286_usb_msc folder
sh7286_usb_msc.hws
sh7286_usb_msc folder
Intermediate files, object files
inc folder
Collection of include files to share
src folder
main.c
common folder
Sample programs related to set up the CPU
usb folder
Sample programs related to USB mass storage class

Figure 4 Collection of Files in the Folder

4.2.2 Compile and Link Source Codes

Use the High-performance Embedded Workshop 4 to compile the source codes.

4.3 How to Load and Execute the Program

This section describes how to load and execute the program.

4.3.1 Loading the Program

Follow the steps to load the sample program to the SH7286 CPU board:

(1) Connect the E10A-USB emulator to the E10A-USB computer where the High-performance Embedded Workshop 4.
(2) Use the user interface cable (E10A cable) to connect the E10A-USB emulator and the SH7286 CPU board.
(3) Turn ON the power of the SH7286 CPU board.
(4) Execute "sh7286_usb_msc.hws" in "sh7286_usb_msc" folder.
(5) Select [Connect] on the [Debug] menu. The [Select Emulator mode] dialog box will appear. Select "SH7286" as the device, and "E10A-USB Emulator" as the emulator mode.
(6) Press the reset button on the SH7286 CPU board, and click "OK".
(7) The [System Clock] dialog box will appear. Enter the clock frequency of the crystal units (10 MHz).
(8) The [ID Code] dialog box will appear. Enter "E10A".
(9) Select [Debug] menu -> [Download Modules] to load the program on the SH7286 CPU board.

4.3.2 Executing the Program

Select [Reset Go] on the [Debug] menu to execute the program.

4.4 How to Use the RAM Disk

This section gives an example to use the RAM disk on Windows XP-based computer.

When executing the program, insert a USB series-B connector to the SH7286 CPU board, series-A connector (on the other side) to the USB host computer.

After an enumeration using control transfer and bulk transfer, a USB mass storage device appears under the Universal Serial Bus controllers node, and RENESAS EX RAN Disk USB Device appears under the Disk drives node. Then, the host computer recognizes the SH7286 CPU board as a storage device to mount the Local Disk drive in My Computer.

Next, format the Local Disk drive on the computer.

Open My Computer and right-click the Local Disk drive, and then click "Format". "Format Local Disk" dialog box appears to set. Make sure that the file system is set as "FAT", and click "OK".

When the warning message appears, proceed with formatting by pressing "OK".

When formatting is completed, "Format completed" message will appear. Click "OK" to return to the "Format Local Disk" dialog box. Click "OK" to exit.

The SH7286 CPU board now can be used as the RAM disk which is connected on the USB.

4.5 Changing the RAM Disk Setting

This section describes how to change the RAM disk setting used in this sample program.

4.5.1 Removable Disk/Fixed Disk

This sample program uses the RAM disk as a removable disk. Comment out "#define REMOVABLE_DISK" in SetSystemSwitch.h file, and enable "#undef REMOVABLE_DISK" which is commented out to use the RAM disk as a fixed disk.

4.5.2 How to Modify the amount of RAM Disk

This sample program uses 16-MB SDRAM as a RAM disk. Alter the "SysMemMap.h" file to modify the amount of RAM disk. First, specify the total number of bytes of RAM disk (1) by "DISK_ALL_BYTE". Then, specify the beginning and the end of the RAM disk area by "RAM_DISK_S", and "RAM_DISK_E (2)", respectively.

Notes: 1. Specify the value bigger than 1.5 MB. As the FAT information consumes the RAM disk area, the actual amount of the RAM disk area will be reduced. This sample program configures FAT 12 information by up to 16-MB data, and FAT16 information by up to 2-GB data. Other FAT information must be created by user.
2. The RAM disk area specified between "RAM_DISK_S" and "RAM_DISK_E" must be bigger than the size specified by "DISK_ALL_BYTE".

5. Sample Program Overview

This chapter describes the features and the configuration of the sample program. This sample program operates on the SH7286 CPU board, and the SH7286 CPU board operates as a RAM disk. USB transfer is started by an interrupt from the USB function module.

5.1 State Transition Diagram

Figure 5 shows the state transition diagram of this sample program. This sample program transitions to three states as shown in Figure 5.

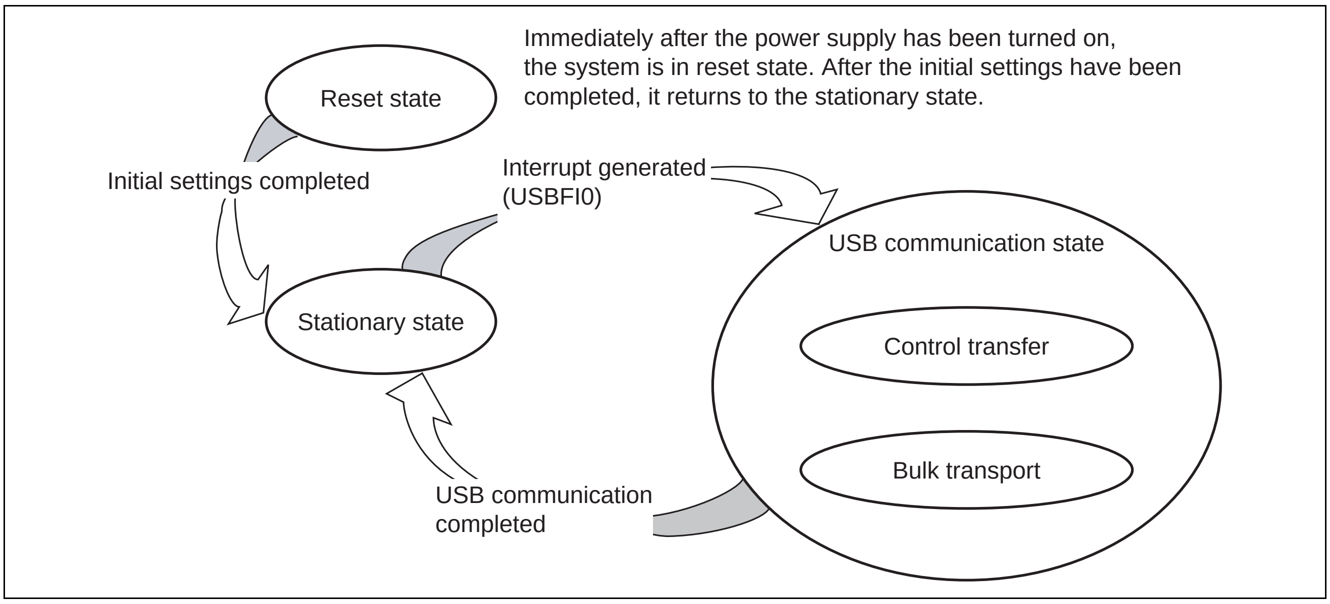
Reset state
Immediately after the power supply has been turned on, the system is in reset state. After the initial settings have been completed, it returns to the stationary state.
Initial settings completed
Interrupt generated (USBFI0)
USB communication state
Stationary state
Control transfer
Bulk transport
USB communication completed

Figure 5 State Transition Diagram

- Reset state
  The SH7286 enters reset state when it is reset at power-on, or manually reset. The sample program mainly configures the SH7286 in reset state.
- Stationary state
  When configuring the SH7286 is completed, the sample program stays in stationary state as the main loop.
- USB communication state
  The SH7286 transitions to USB communication state when the USB module generates an interrupt in stationary state. In this state, the SH7286 transfers data according to the type of interrupts. USB interrupt flag registers 0 and 1 (USBIFR 0, 1) specify interrupts used in this sample program. When an interrupt occurs, the corresponding bits to USBIFR0 and USBIFR1 are set to 1.

5.2 USB Communication State

The USB communication state is divided into two states according to the type of data transfer (See Figure 6). When an interrupt occurs, the SH7286 transitions to the USB communication state, and then jumps to each state according to the type of interrupts.

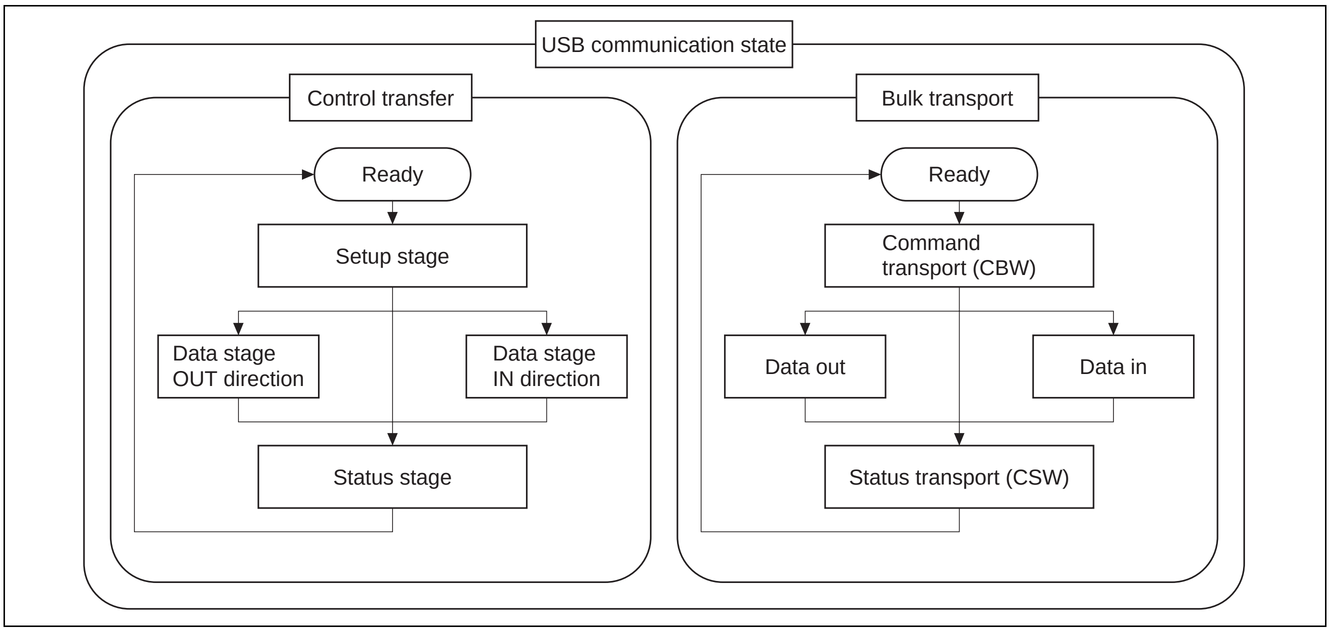

Figure 6 USB Communication State

5.2.1 Control Transfer

Control transfer is used to retrieve the device information and set the device state, which is executed immediately after the function is connected to the host.

Control transfer consists of two or three stages, Setup stage, Data stage, and Status stage.

5.2.2 Bulk Transfer

Bulk transfer is used to transfer large amount of data untimely but without any error. Transfer speed is not guaranteed, however, the delivery of the data is guaranteed. USB Mass Storage Class (Bulk-Only Transport) uses bulk transfer to send the data from the function to the host computer.

USB Mass Storage Class (Bulk-Only Transport) data transfer (read/write) is composed of two or three stages, Command transport (CBW), Data transport, and Status transport (CSW).

5.3 File Configuration

This sample program is composed of 15 source code files and 14 header files. Table 8 lists the file configuration. Functions are assembled in a file by transfer type or feature.

Table 8 File Configuration

File Name		Description
main.c		Configures the USB function
usb folder		
	UsbMain.c	Detects the interrupt source, sends or receives packets
	DoRequest.c	Handles the setup command issued by the host PC
	DoRequestBOT_StorageClass.c	Handles the USB Mass Storage Class (Bulk-Only Transport) class-specific commands
	DoControl.c	Executes control transfer
	DoBulk.c	Executes bulk transfer
	DoBOTMSClass.c	Executes the USB Mass Storage Class (Bulk-Only Transport) transport
	DoSCSICommand.c	Parses and processes SCSI commands
	CatBOTTypedef.h	Bulk-Only Transport structure definition
	CatProType.h	Prototype declaration
	CatSCSITypedef.h	SCSI structure definition, macro definition to create FAT information
	CatTypedef.h	Basic structure definition used by USB firmware
	SetBOTInfo.h	Configures variables to support Bulk-Only Transport
	SetMacro.h	Macro definition
	SetSCSIInfo.h	Configures variables to support SCSI commands
	SetSystemSwitch.h	Sets the system operation
	SetUsbInfo.h	Configures variables used by USB firmware
	SysMemMap.h	Defines the memory map address
common folder		
	bscsdram.c	Sets the SDRAM interface
	cpg.c	Sets the CPG
	dbsct.c	Defines initialized sections (B, R)
	hwsetup.c	Initializes the hardware
	intprg.c	Interrupt program
	resetprg.c	CPU initialization program
	stacksct.h	Defines the stack area
	vect.h	Declares the interrupt program
	vecttbl.c	Defines the interrupt handling vector table
inc folder		
	iodefine.h	SH7286 register definition
	typedefine.h	Common type definition

5.4 Functionality of Functions

Table 9 to Table 16 list functions stored in each file and those functionalities.

- main.c
  Upon power-on reset, resetprg.c CPU initialization program is executed to call main function in main.c. It initializes the SH7286, and clears the RAM area to use in Bulk transfer.

Table 9 main.c

File Name	Function Name	Description
main.c	main	Initializes the module and memory, pulls up the USB bus, and transitions to main loop

- UsbMain.c
  Usb.Main.c detects the interrupt sources mainly by the USB interrupt flag register, and calls functions according to the interrupt type. It also transmits/receives packets between the host controller and function module.

Table 10 Usb.Main.c

File Name	Function Name	Description
UsbMain.c	BranchOfInt	Detects the interrupt sources of the bus reset and endpoint 0, and calls the function according to the interrupt
	GetPacket	Writes data to RAM transferred from the host controller
	GetPacket4	Writes data transferred from the host controller to RAM in longwords (Ring buffer supported, not used in the USB Mass Storage Class)
	GetPacket4S	Writes data transferred from the host controller to RAM in longwords (Ring buffer not supported, high-speed)
	PutPacket	Writes data to USB module transferred to the host controller
	PutPacket4	Writes data to the USB module in longwords to transfer to the host controller (Ring buffer supported, not used in the USB Mass Storage Class)
	PutPacket4S	Writes data to the USB module in longwords to transfer to the host controller (Ring buffer not supported, high-speed)
	SetControlOutContents	Overwrites the existing data with the data transferred from the host
	SetUsbModule	Configures the USB module
	ActBusReset	Clears FIFOs when receiving the bus reset
	ActBusVcc	Handles USB cable attachment interrupt (This function is not used in this sample application)
	ConvRealn	Reads the specified bytes of data from the specified address
	ConvReflexn	Reads the specified bytes of data from the specified address in reverse

- DoRequest.c
  During Control transfer, it decodes commands from the host controller and handles the commands according to the type. This sample program sets H'045B (Vendor: Renesas) to the vendor ID. When developing the USB module, the user must get the vendor ID at the USB Implementers Forum. As the vendor command is not used in this sample, DecVendorCommands does not perform any action. Develop the program to use the vendor command.

Table 11 DoRequest.c

File Name	Function Name	Description
DoRequest.c	DecStandardCommands	Decodes commands issued by the host controller, and handles the standard commands
	DecVenderCommands	Handles the vendor command

- DoRequestBOT_StorageClass.c
  Handles the Bulk-Only Mass Storage Reset and Get Max LUN, the USB Mass Storage Class (Bulk-Only Transport commands.
  Bulk-Only Mass Storage Reset command resets all the interfaces used by the USB Mass Storage Class Bulk-Only Transport.
  Get Max LUN command returns the maximum number of logical units supported by peripherals. As the number of logical units in this sample program is 1, it returns 0 to the host.

Table 12 DoRequestBOT_StorageClass.c

File Name	Function Name	Description
DoRequestBOT_ StorageClass.c	DecBOTClassCommands	Handles the USB Mass Storage Class (Bulk-Only Transport)

- DoControl.c
  When the SETUP TS interrupt in Control transfer occurs, the ActControl retrieves the command, the DecStandardCommands decodes the command to detect its direction to transfer. When EP0oTS, EP0iTR, and EP0iTS interrupts in Control transfer occur, the Act ControlInOut calls either the ActControlIn or ActControlOut depending on its direction, and executes Data Stage and Status Stage.

Table 13 DoControl.c

File Name	Function Name	Description
DoControl.c	ActControl	Controls the Setup Stage in Control transfer
	ActControlIn	Controls the Data Stage and Status Stage in Control IN transfer (Data Stage direction: IN)
	ActControlOut	Controls the Data Stage and Status Stage in Control OUT transfer (Data Stage direction: OUT)
	ActControlInOut	Divides the Data Stage and Status Stage in control transfer into "ActControlIn" and "ActControlOut"

- DoBulk.c
  DoBulk.c handles Bulk transfer. The USB Mass Storage Class (Bulk-Only Transport) does not use the AcrBulkInReady.

Table 14 DoBulk.c

File Name	Funtion Name	Description
DoBulk.c	ActBulkOut	Executes Bulk OUT transfer
	ActBulkIn	Executes Bulk IN transfer
	ActBulkInReady	Prepares for bulk IN transfer

- DoBOTMSClass.c
  DoBOTMSClass.c controls two or three stages in the USB Mass Storage Class (Bulk-Only Transport), and behaves according to the specification.

Table 15 DoBOTMSClass.c

File Name	Function Name	Description
DoBOTMSClass.c	ActBulkOnly	Detects the current state and divides into stages
	ActBulkOnlyCommand	Controls CBW in the Bulk-Only Transport
	ActBulkOnlyIn	Controls Data transport and Status transport in Bulk-Only Transport (Data Stage direction: IN)
	ActBulkOnlyOut	Controls Data transport and Status transport in Bulk-Only Transport (Data Stage direction: OUT)

- DoSCSICommand.c

DoSCSICommand.c parses SCSI commands sent from the host computer, and prepares for the next Data transport or Status port.

Table 16 DoSCSICommand.c

File Name	Function Name	Description
DoSCSICommand.c	DecBotCmd	Handles the SCSI command sent from the host in Bulk-Only Transport
	SetBotCmdErr	Handles SCSI command error

5.5 RAM Disk

This sample program assumes the SDRAM on the SH7286 CPU board as a disk, and reports to the host computer that the SH7286 CPU board (function) is a disk.

The disk (function) includes the Master boot block and Partition boot block as shown in Figure 7. When turning ON the system, write the Master boot block and Partition boot block in the RAM disk area on the SDRAM using the initialization routine.

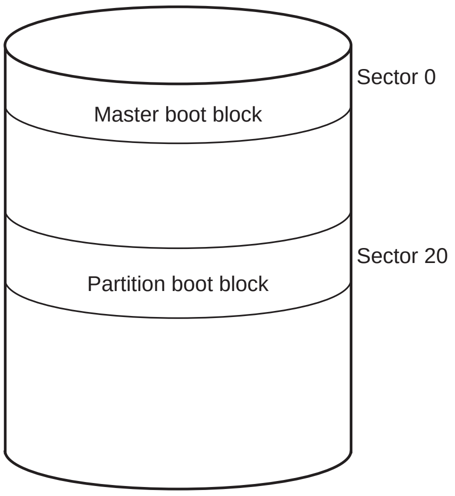

Figure 7 Disk Structure

Use the SCSI commands to access from the host computer to the function (store data, read data). User must understand the structure shown in Figure 7 to use SCSI commands.

5.6 Supported SCSI Commands

Table 17 to Table 19 describe SCSI commands supported by this sample program.

Table 17 SCSI Commands (1/3)

Command Name	Transport Name	Description
INQUIRY	CBW	Decodes the command, acknowledges it is the INQUIRY command, and prepares for sending the INQUIRY information (96-byte) stored in ROM
	Data	Sends the INQUIRY information to the host computer using Bulk IN transfer
	CSW	Sends the command completion result to the host computer. When the number of bytes of transmit data is equal to or less than 96 bytes, it sends "Command passed".
READ CAPACITY	CBW	Decodes the command, acknowledges it is the READ CAPACITY command, and prepares for sending the READ CAPACITY information (8-byte) which consists of the number of bytes per sector in a disk and total number of sectors in a disk. When the media cannot be accessed (LSB of unit_state[0] is 1), the function handles the transfer as with no data, and handles it according to "5.7 Error Handling Case (4)". Also, it sets the sense key of the REQUEST SENSE to "NOT READY".
	Data	Sends the READ CAPACITY information to the host computer using host computer using Bulk IN transfer. When the media cannot be accessed, it returns the same number of bytes of data (H'00) requested from the host.
	CSW	Sends the command completion result to the host computer. When the media cannot be accessed, it returns "Command failed (CSW status: H'01)
READ (10)	CBW	Decodes the command, acknowledges it is READ (10) command, and prepares for sending the specified number of bytes of data from the read sector in a disk which is executed on SDRAM. When the media cannot be accessed (LSB of unit_state[0] is 1), the function handles the transfer as with no data, and handles it according to "5.7 Error Handling Case (4)". Also, it sets the sense key of the REQUEST SENSE to "NOT READY".
	Data	Sends the data stored in the read sector to the host computer. When the media cannot be accessed, it returns the same number of bytes of data (H'00) requested from the host.
	CSW	Sends the READ (10) command execution result to the host computer. When the media cannot be accessed, it returns "Command failed (CSW status: H'01)"
WRITE (10)	CBW	Decodes the command, acknowledges it is the WRITE (10) command, and prepares for receiving the specified number of bytes of data from the write sector stored in the write sector in a disk which is executed on SDRAM. When the media cannot be accessed (LSB of unit_state[0] is 1), the function handles the transfer as with no data, and handles it according to "5.7 Error Handling Case (9)". Also, it sets the sense key of the REQUEST SENSE to "NOT READY".
	Data	Receives the data stored in the write sector from the write sector in Bulk OUT transfer data. When the media cannot be accessed, it executes "dummy read" to read the data sent from the host.
	CSW	Sends "Command passed" to the host computer. When the media cannot be accessed, it returns "Command failed (CSW status: H'01)".

Table 18 SCSI Commands (2/3)

Command Name	Transport Name	Description
REQUEST SENSE	CBW	Decodes the command, acknowledges it is the REQUEST SENSE command, and prepares for sending the value (the last SCSI command execution result)
	Data	Sends the value to the host computer using Bulk IN transfer
	CSW	Sends the command completion result to the host computer. When the number of bytes of transmit data is equal to or less than 18 bytes, it sends "Command passed".
PREVENT ALLOW MEDIUM REMOVAL	CBW	Decodes the command, acknowledges it is the PREVENT ALLOW MEDIUM REMOVAL command, and prepares for sending "Command passed". When the media cannot be accessed (LSB of unit_state[0] is 1), it sets the command as "Failed", and the sense key of the REQUEST SENSE to "NOT READY".
	Data	This command does not include the Data transport.
	CSW	Sends "Command passed" to the host computer. When the media cannot be accessed, it returns "Command failed (CSW status: H'01)".
TEST UNIT READY	CBW	Decodes the command, acknowledges it is the TEST UNIT READY command, and prepares for sending "Command passed". When the media cannot be accessed (LSB of unit_state[0] is 1), it sets the command as "Failed", and the sense key of the REQUEST SENSE to "NOT READY".
	Data	This command does not include the Data transport.
	CSW	Sends "Command passed" to the host computer. When the media cannot be accessed, it returns "Command failed (CSW status: H'01)".
VERIFY (10)	CBW	Decodes the command, acknowledges it is the VERIFY (10) command, and prepares for sending "Command passed" to the host computer. When the media cannot be accessed (LSB of the unit_state[0] is 1), it sets the command as "Failed", and the sense key of the REQUEST SENSE to "NOT READY".
	Data	This command does not include the Data transport.
	CSW	Sends "Command passed" to the host computer. When the media cannot be accessed, it returns "Command failed (CSW status: H'01).
START/STOP UNIT	CBW	Decodes the command, acknowledges it is the STOP/START UNIT command. When the command specifies to eject media or stop, it sets the LSB of the unit_state[0] to 1. Otherwise, it sets the LSB of the unit_state[0] to 0. When user wants to wake the media up from access-disabled state, set the LSB of unit_state[0] to 0.
	Data	This command does not include the Data transport.
	CSW	Sends "Command passed" to the host computer
MODE SENSE (6)	CBW	Decodes the command, acknowledges it is the MODE SENSE (6) command, and prepares for sending the requested MODE SENSE information
	Data	Sends the MODE SENSE information to the host computer using Bulk IN transfer
	CSW	Sends the command execution result to the host computer

Table 19 SCSI Commands (3/3)

Command Name	Transport Name	Description
READ FORMAT CAPACITY	CBW	Decodes the command, acknowledges it is the READ FORMAT CAPACITY command, and recognizes the disk which is executed on SDRAM is formatted. Then, it prepares for sending the READ FORMAT CAPACITY information (20-byte) which consists of the number of bytes per sector and total number of sectors in a disk. When the media cannot be accessed (LSB of unit_state[0] is 1), the function handles the transfer as with no data, and handles it according to "5.7 Error Handling Case (4)". Also, it sets the sense key of the REQUEST SENSE to "NOT READY".
	Data	Sends the READ FORMAT CAPACITY information to the host computer. When the media cannot be accessed, it returns the same number of bytes of data (H'00) requested from the host.
	CSW	Sends the command execution result to the host computer. When the media cannot be accessed, it returns "Command failed (CSW status: H'01)".
Other commands (Not supported)	CBW	Decodes the command. When the command is not supported, it sets the sense key of the REQUEST SENSE to "INVALID FIELD IN CDB", and prepares for the Data transport.
	Data	When the host computer requests to send data using Bulk IN transfer, it sends the same number of bytes of data (H'00) requested from the host. When the host computer sends data using Bulk OUT transfer, it calculates the number of bytes received. When the Data transport is not included, the command does nothing.
	CSW	Returns "Command failed (CSW status: H'01)" to the host computer

5.7 Error Handling

This section describes errors occur between the host computer and the function, and error handling by the function when transferring data in the USB Mass Storage Class (Bulk-Only Transport).

The USB Mass Storage Bulk-Only Transport defines the following error cases:

- CBW Not Valid
- Device intent does not meet the host's expectations (10 cases)

Other cases are not defined in the USB Mass Storage Bulk-Only Transport specification.

There are thirteen possible cases of data transfer between the host and function listed in Table 20 and Table 21. Cases (1), (6), and (12) are normal transfers.

Table 20 Host/Function Data Transfer Matrix

		Host		
		Host expects no data transfer	**Host expects to receive data from the function**	**Host expects to send data to the function**
Function	**Function intends to transfer no data**	(1) Hn = Dn	(4) Hi > Dn	(9) Ho > Dn
	Function intends to send data to the host	(2) Hn < Di	(5) Hi > Di	(10) Ho < > Di
			(6) Hi = Di	
			(7) Hi < Di	
	Function intends to receive data from the host	(3) Hn < Do	(8) Hi < > Do	(11) Ho > Do
				(12) Ho = Do
				(13) Ho < Do

Table 21 Host/Function Data Transfer Cases

CASE	**Relationship between Host and Function**
(1)	Host expects no data transfer, and function intends to transfer no data
(2)	Host expects no data transfer, and function intends to send data to the host
(3)	Host expects no data transfer, and function intends to receive data from the host
(4)	Host expects to receive data from the function, and function intends to transfer no data
(5)	The amount of data received from the function that the host expected is less than the amount of data that the function sends to the host
(6)	The amount of data received from the function that the host expected and the amount of data that the function sends to the host are the same
(7)	The amount of data received from the function that the host expected is more than the amount of data that the function sends to the host
(8)	Host expects to receive data from the function, and function intends to receive data from the host
(9)	Host expects to send data to the host, and function intends to transfer no data
(10)	Host expects to send data to the host, and function intends to send data to the host
(11)	The amount of data that the host expected to send to the function is less than the amount of data the function receives from the host
(12)	The amount of data that the host expected to send to the function and the amount of data that the function receives from the host are the same
(13)	The amount of data that the host expected to send to the function is more than the amount of data that the function receives from the host

Table 22 lists examples of possible errors.

Table 22 Error Cases

CASE	Status
(2)	When the host issues the READ command, the amount of data sent on the USB Data transport is 0, and the amount of data specified by the SCSI command is other than 0
(3)	When the host issues the WRITE command, the amount of data sent on the USB Data transport is 0, and the amount of data specified by the SCSI command is other than 0
(4)	When the host issues the READ command, the amount of data sent on the USB Data transport is 0, and the amount of data specified by the SCSI command is other than 0
(5)	When the host issues the READ command, the amount of data specified by the SCSI command is less than the data sent on the USB Data transport
(7)	When the host issues the READ command, the amount of data specified by the SCSI command is more than the data sent on the USB Data transport
(8)	When the host issues the WRITE command, it also requests to receive data on the USB Data transport
(9)	When the host issues the WRITE command, the amount of data sent on the USB Data transport is other than 0, and the amount of data specified by the SCSI command is 0
(10)	When the host issues the READ command, it also sends data on the USB Data transport
(11)	When the host issues the WRITE command, the amount of data specified by the SCSI command is less than the data sent on the USB Data transport
(13)	When the host issues the WRITE command, the amount of data specified by the SCSI command is more than the data sent on the USB Data transport

Table 23 lists how the function behaves toward the error cases.

Table 23 Function Behavior

CASE	Function behavior When an Error occurs
(2), (3)	• Sets H'02 to the CSW status
(4), (5)	• Adds data to make up the data length specified by the dCBWDataTransferLength, and sends the data to the host • Sets the amount of data added on the Data transport to the dCBWDataResidue in CSW • Sets H'00 to the CSW status
(7), (8)	• Sends data up to the data length specified by the dCBWDataTransferLength to the host • Sets H'02 to the CSW status
(9), (11)	• Receives the length of data indicated by the dCBWDataTransferLength • Sets the difference between the amount of data received on the Data transport and processed by the function to the dCBWDataResidue in CSW. • Sets H'01 to the CSW status
(10), (13)	• Receives the length of data indicated by the dCBWDataTransferLength • Sets H'02 to the CSW status

Figure 8 to Figure 10 show flow charts of error on data transfer.

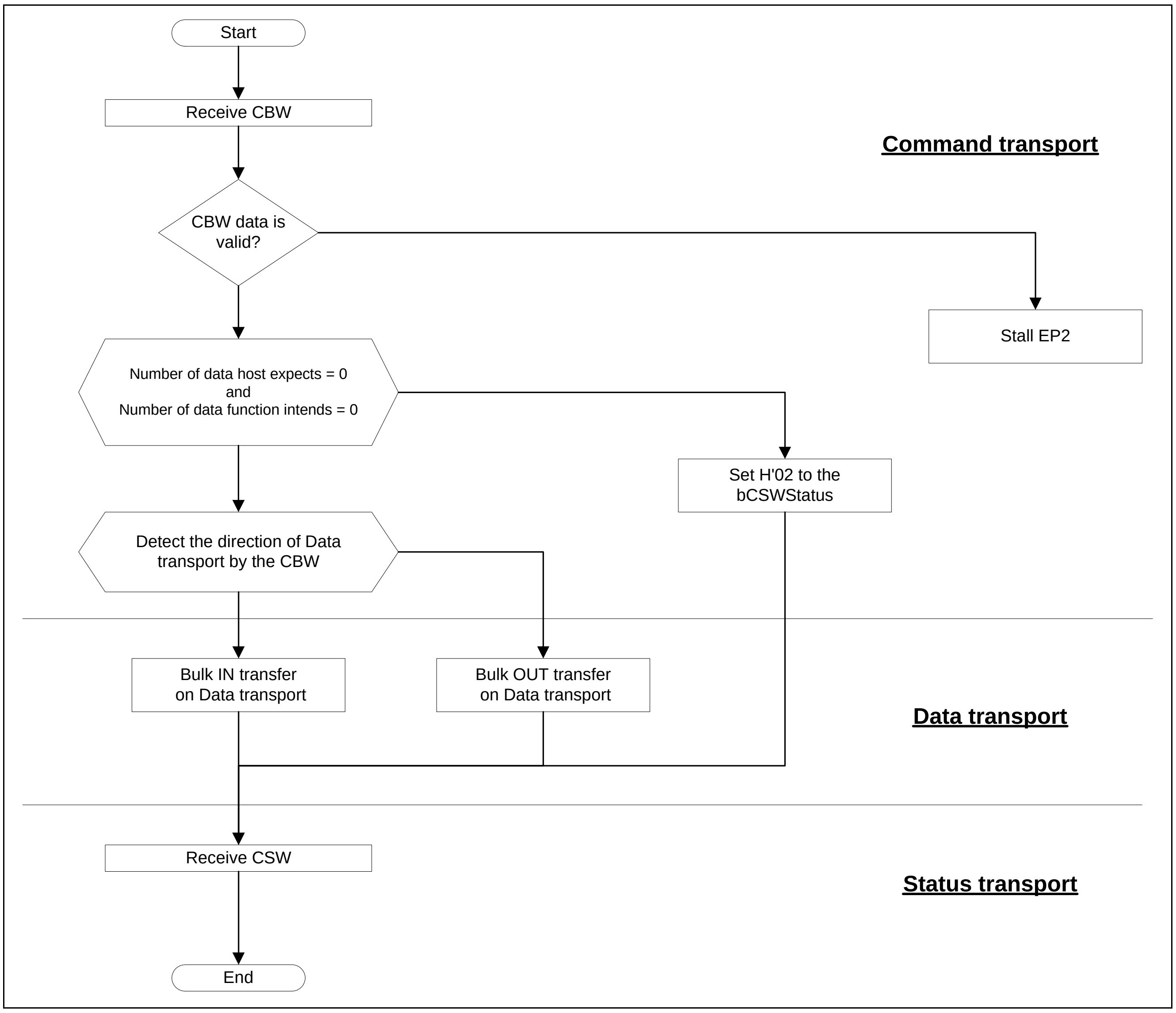
Start
Receive CBW
Command transport
CBW data is valid?
Stall EP2
Number of data host expects = 0 and Number of data function intends = 0
Set H'02 to the bCSWStatus
Detect the direction of Data transport by the CBW
Bulk IN transfer on Data transport
Bulk OUT transfer on Data transport
Data transport
Receive CSW
Status transport
End

Figure 8 Flow Chart for Error Handling on Data Transfer (1/3)

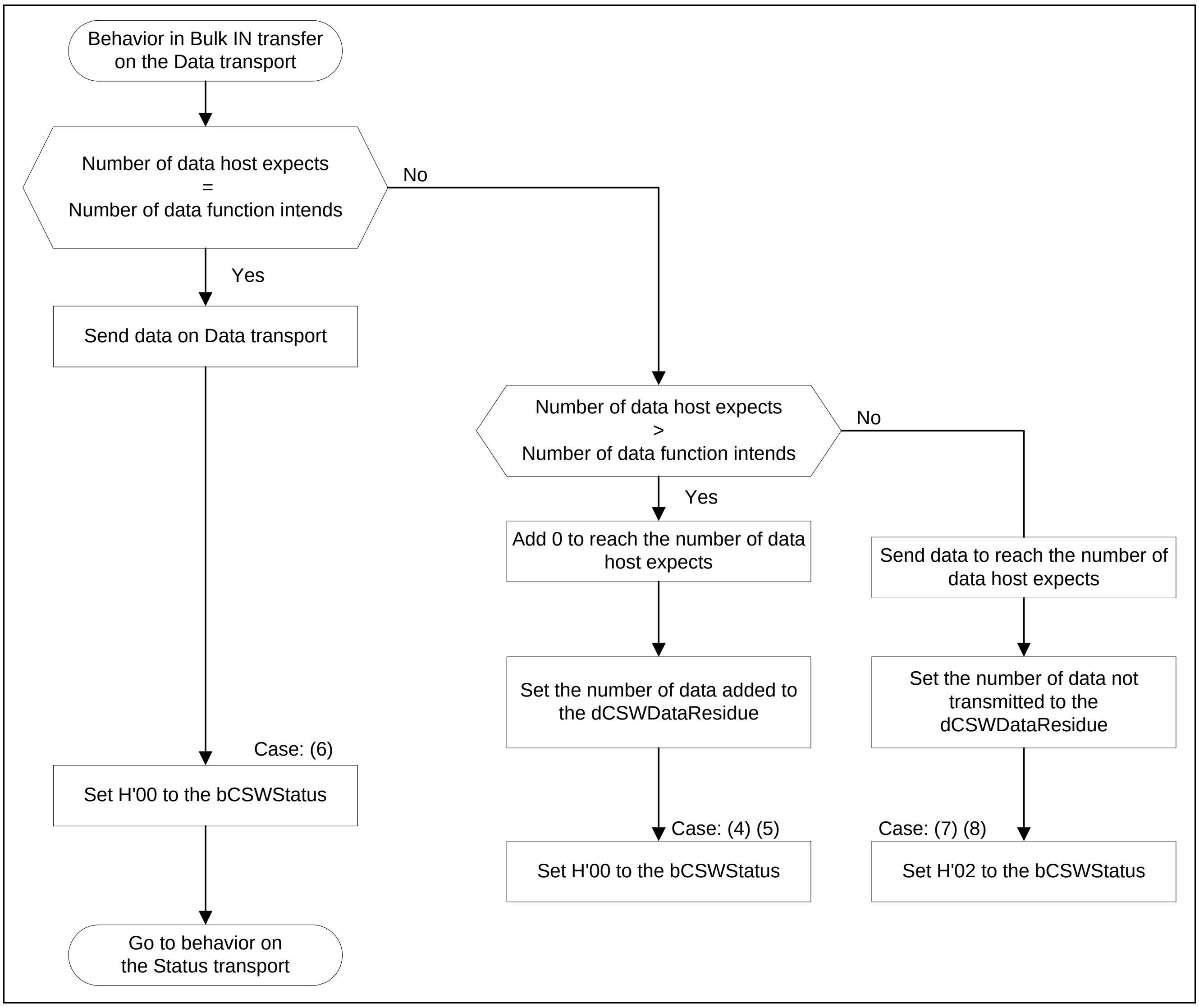
Behavior in Bulk IN transfer on the Data transport
Number of data host expects = Number of data function intends
No
Yes
Send data on Data transport
Number of data host expects > Number of data function intends
No
Yes
Add 0 to reach the number of data host expects
Send data to reach the number of data host expects
Set the number of data added to the dCSWDataResidue
Set the number of data not transmitted to the dCSWDataResidue
Case: (6)
Set H'00 to the bCSWStatus
Case: (4) (5)
Case: (7) (8)
Set H'00 to the bCSWStatus
Set H'02 to the bCSWStatus
Go to behavior on the Status transport

Figure 9 Flow Chart for Error Handling on Data Transfer (2/3)

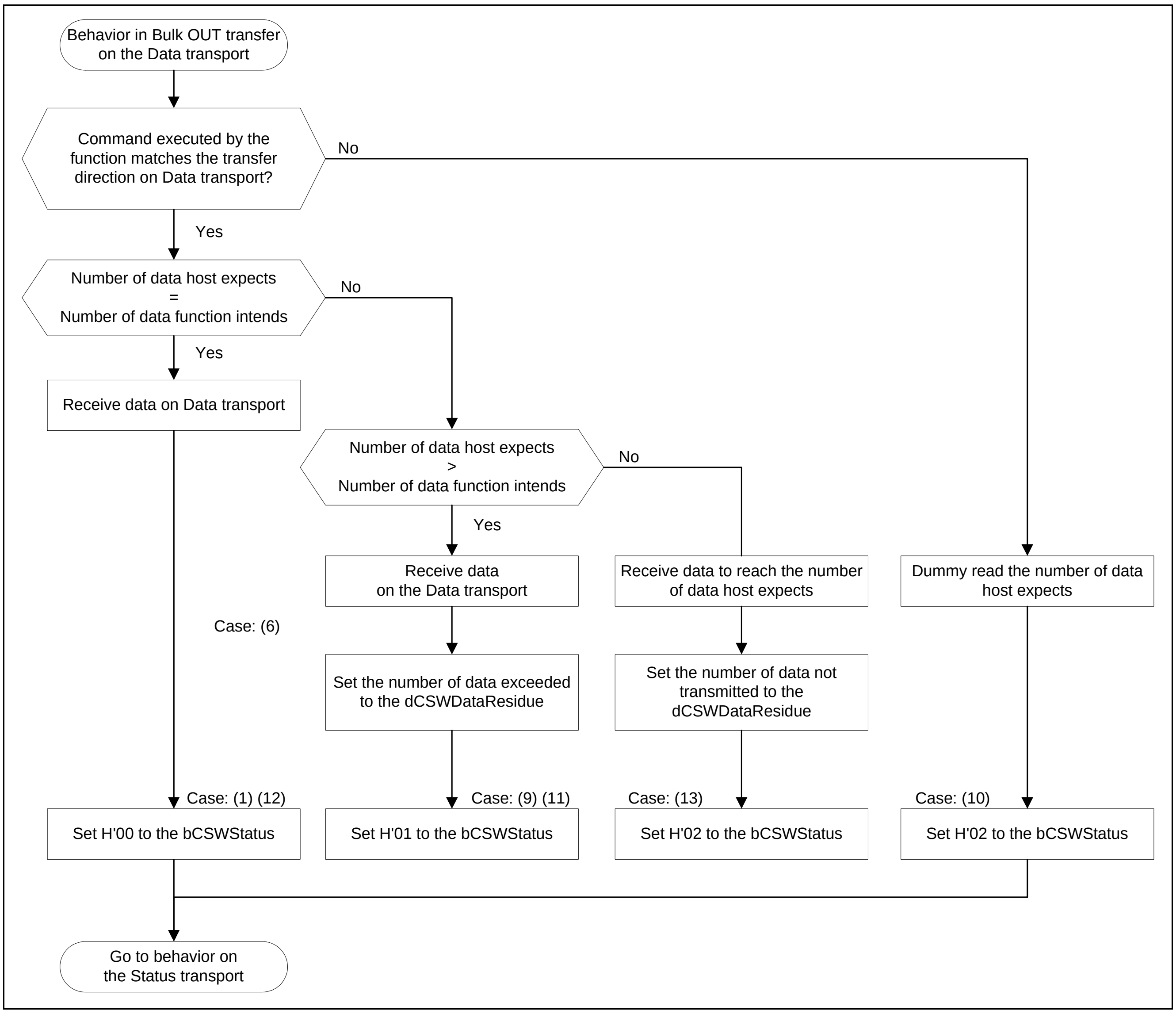
Behavior in Bulk OUT transfer on the Data transport
Command executed by the function matches the transfer direction on Data transport?
No
Yes
Number of data host expects = Number of data function intends
No
Yes
Receive data on Data transport
Number of data host expects > Number of data function intends
No
Yes
Receive data on the Data transport
Receive data to reach the number of data host expects
Dummy read the number of data host expects
Case: (6)
Set the number of data exceeded to the dCSWDataResidue
Set the number of data not transmitted to the dCSWDataResidue
Case: (1) (12)
Case: (9) (11)
Case: (13)
Case: (10)
Set H'00 to the bCSWStatus
Set H'01 to the bCSWStatus
Set H'02 to the bCSWStatus
Set H'02 to the bCSWStatus
Go to behavior on the Status transport

Figure 10 Flow Chart for Error Handling on Data Transfer (3/3)

When transferring data in the USB Mass Storage Class (Bulk-Only Transport), a series of data transfer starts on the CBW transport, and returns a series of the transfer result (status) to the host computer on the CSW transport. To return the status on the CSW transport, create values in two fields; bCSWStatus to indicate the success or failure of the command, and dCSWDataResidue to indicate the number of bytes of data transfer error.

This sample program uses the following fields;

— dCBWDataTransferLength field of the CBW packet
— dCSWDataTRansferResidue field of the CSW packet

dCBWDataTransferLength field is a variable to store the number of bytes of data specified by the host computer to process on the Data transport.

dCSWDataTransferResidue field is a variable to store the number of bytes of data processed by the function on the Data transport.

When the CBW transport is completed, dCBWDataTransferLength and dCSWDataTransferResidue fields store the number of bytes of data processed on the Data transport.

Data transfer on the Data transport is executed as shown in Figure 8 to Figure 10 flow charts.

When there is no error during the data transfer between the host and function, the sample program subtracts the number of bytes to transfer from fields dCBWDataTransferLength and dCSWDataTransferResidue every time the data transfer is executed on the Data transport. Otherwise, the sample program sets the "difference" between the number of bytes of data the host expects to process on the Data transport and the number of bytes of data processed by the function on the Data transport to the dCSWDataTransferResidue field of the CSW packet, and transitions to the Status transport.

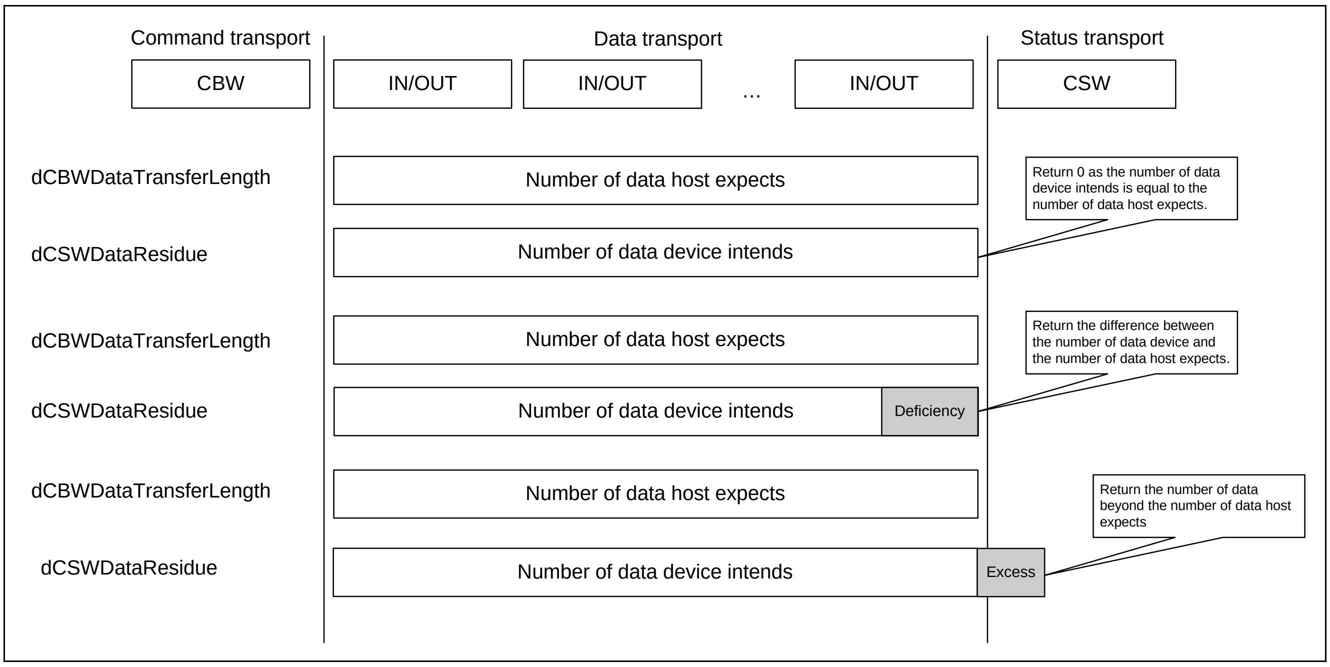
Command transport
Data transport
Status transport
CBW
IN/OUT
IN/OUT
...
IN/OUT
CSW
dCBWDataTransferLength
Number of data host expects
Return 0 as the number of data device intends is equal to the number of data host expects.
dCSWDataResidue
Number of data device intends
dCBWDataTransferLength
Number of data host expects
Return the difference between the number of data device and the number of data host expects.
dCSWDataResidue
Number of data device intends
Deficiency
dCBWDataTransferLength
Number of data host expects
Return the number of data beyond the number of data host expects
dCSWDataResidue
Number of data device intends
Excess

Figure 11 Stages on Bulk-Only Transport

6. References

- Software Manual
  (1) SH-2A, SH2A-FPU Software Manual Rev. 3.00
  (The latest version of software manual can be downloaded from the Renesas Electronics website.)
- Hardware Manual
  (2) SH7280 Group Hardware Manual Rev. 2.00
  (The latest version of hardware manual can be downloaded from the Renesas Electronics website.)
- USB specifications
  (3) Universal Serial Bus Specification, Revision 2.0
  (4) Universal Serial Bus Mass Storage Class Specification Overview
  (5) Universal Serial Bus Mass Storage Class (Bulk-Only Transport)
  — USB Implementers Forum website:
  http://www.usb.org/developers

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Oct.22.10	—	First edition issued

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.

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: +49-211-6503-0, 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, R.O.C.
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 Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141