Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010 Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.



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; anticrime 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 majorityowned subsidiaries.
- (Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Regarding the change of names mentioned in the document, such as Hitachi Electric and Hitachi XX, to Renesas Technology Corp.

The semiconductor operations of Mitsubishi Electric and Hitachi were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Hitachi, Hitachi, Ltd., Hitachi Semiconductors, and other Hitachi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Renesas Technology Home Page: http://www.renesas.com

Renesas Technology Corp. Customer Support Dept. April 1, 2003



Cautions

Keep safety first in your circuit designs!

Renesas Technology Corporation puts the maximum effort into making semiconductor products better
and more reliable, but there is always the possibility that trouble may occur with them. Trouble with
semiconductors may lead to personal injury, fire or property damage.

Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

- These materials are intended as a reference to assist our customers in the selection of the Renesas
 Technology Corporation product best suited to the customer's application; they do not convey any
 license under any intellectual property rights, or any other rights, belonging to Renesas Technology
 Corporation or a third party.
- Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
- 3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
 - The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
 - Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
- 4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
- 5. Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
- 6. The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
- 7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
 - Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
- 8. Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.



H8S/2215 USB Function Module Mass Storage Class (Bulk-Only Transport) Application Notes Renesas 16-Bit Single-Chip Microcomputer

HD64F2215

Cautions

- 1. Hitachi neither warrants nor grants licenses of any rights of Hitachi's or any third party's patent, copyright, trademark, or other intellectual property rights for information contained in this document. Hitachi bears no responsibility for problems that may arise with third party's rights, including intellectual property rights, in connection with use of the information contained in this document.
- Products and product specifications may be subject to change without notice. Confirm that you have received the latest product standards or specifications before final design, purchase or use.
- 3. Hitachi makes every attempt to ensure that its products are of high quality and reliability. However, contact Hitachi's sales office before using the product in an application that demands especially high quality and reliability or where its failure or malfunction may directly threaten human life or cause risk of bodily injury, such as aerospace, aeronautics, nuclear power, combustion control, transportation, traffic, safety equipment or medical equipment for life support.
- 4. Design your application so that the product is used within the ranges guaranteed by Hitachi particularly for maximum rating, operating supply voltage range, heat radiation characteristics, installation conditions and other characteristics. Hitachi bears no responsibility for failure or damage when used beyond the guaranteed ranges. Even within the guaranteed ranges, consider normally foreseeable failure rates or failure modes in semiconductor devices and employ systemic measures such as fail-safes, so that the equipment incorporating Hitachi product does not cause bodily injury, fire or other consequential damage due to operation of the Hitachi product.
- 5. This product is not designed to be radiation resistant.
- 6. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without written approval from Hitachi.
- 7. Contact Hitachi's sales office for any questions regarding this document or Hitachi semiconductor products.

Preface

These application notes describe the Mass Storage Class (Bulk-Only Transport) firmware that uses the USB Function Module in the H8S/2215. They are provided to be used as a reference when the user creates USB Function Module firmware.

These application notes describe a system configuration example based on the USB Function Module, and do not guarantee the contents of the configuration.

In addition to these application notes, the manuals listed below are also available for reference when developing applications.

[Related manuals]

- Universal Serial Bus Specification Revision 1.1
- Universal Serial Bus Mass Storage Class Specification Overview Revision 1.1
- Universal Serial Bus Mass Storage Class(Bulk-Only Transport) Revision 1.0
- H8S/2215 USB Function Module Mass Storage Class (Bulk-Only Transport) Application Notes
- H8S/2215 Hardware Manual
- H8S/2215 Solution Engine CPU Board (MS2215CP01-C/S) Instruction Manual
- H8S/2215 Series TFP120 User System Interface Cable (HS2215ECN61H) User's Manual
- E6000 (HS2214EPI61H) Emulator User's Manual

[Caution] The sample programs described in these application notes do not include firmware related to interrupt transfer and isochronous transfer, which are USB transport types. When using these transfer types (see page 15-1 of the H8S/2215 Hardware Manual), the user needs to create the programs for them.

Also, the hardware specifications of the H8S/2215 and H8S/2215 Solution Engine, which will be necessary when developing the system described above, are described in these application notes, but more detailed information is available in the H8S/2215 Hardware Manual and the H8S/2215 Solution Engine Instruction Manual.

Contents

Secti	on 1	Overview	1					
Secti	on 2	Overview of the USB Mass Storage Class (Bulk-Only Transport)	3					
2.1		Mass Storage Class						
2.2	Sub-Class Code							
2.3		Only Transport						
2.3	2.3.1	Command Transport						
	2.3.1	Status Transport						
	2.3.2	Data Transport						
	2.3.4	Class Commands						
2.4		Transparent Command Set Sub-Class Code						
2.4	SCSI	Transparent Command Set Suo-Class Code	O					
Secti	on 3	Development Environment	11					
3.1	Hardy	vare Environment	11					
3.2	Softw	are Environment	12					
	3.2.1	Sample Program	12					
	3.2.2	Compiling and Linking	13					
3.3	Loadi	ng and Executing the Program	14					
	3.3.1	Loading the Program	15					
	3.3.2	Executing the Program	15					
3.4	Using	the RAM Disk	15					
Section	on 4	Overview of the Sample Program	17					
4.1		Transition Diagram						
4.2		Communication State						
1.2	4.2.1	Control Transfer						
	4.2.2	Bulk Transport						
4.3		tructure						
4.4		ses of Functions						
4.5		Disk						
4.6		tion of SCSI Commands That Are Supported						
4.7		ssing If an Error Occurs						
Secti		Sample Program Operation						
5.1		Loop						
5.2	Types	of Interrupts						
	5.2.1	Method of Branching to Different Transfer Processes						
5.3	USB (Operating Clock Stabilization Interrupt	44					
	5.3.1	EPINFO	45					

5.4	Interrupt on Cable Connection (VBUS)47					
5.5	Bus Re	eset Interrupt (BRST)	.48			
5.6	6 Control Transfers					
	5.6.1	Setup Stage	.50			
		Data Stage				
	5.6.3	Status Stage	.54			
5.7	Bulk Transfers					
	5.7.1	Command Transport	.57			
	5.7.2	Data Transport	.59			
	5.7.3	Status Transport	.63			
Secti	on 6	Analyzer Data	67			

Section 1 Overview

These application notes describe how to use the USB Function Module that is built into the H8/2215, and contain examples of firmware programs.

The features of the USB Function Module contained in the H8/2215 are listed below.

- An internal UDC (USB Device Controller) conforming to USB 1.1
- Automatic processing of USB controls
- Automatic processing of USB standard commands for end point 0 (some commands need to be processed through the firmware)
- Full-speed (12 Mbps) transfer supported
- Various interrupt signals needed for USB transmission and reception are generated.
- Internal system clock (16 MHz) multiplied by three or external input clock (48 MHz) can be selected as the USB operating clock by the USB clock selector in the clock pulse generator.
- An internal bus transceiver
- Endpoint configuration selectable

End Point Configurations

End Point Name	Name	Transfer Type	Max. Packet Size	FIFO Buffer Capacity	DMA Transfer
End point 0	EP0s	Setup	8 bytes	8 bytes	-
	EP0i	Control In	64 bytes	64 bytes	-
	EP0o	Control Out	64 bytes	64 bytes	-
End point (optional)	EPn	Interrupt (in)	64 bytes	64 bytes (variable)	-
End point (optional)	EPn	Bulk-in	64 bytes	64 x 2 (128 bytes)	Possible
End point (optional)	EPn	Bulk-out	64 bytes	64 x 2 (128 bytes)	Possible
End point (optional)	EPn	Isochronous (in)	128 bytes	128 x 2 (variable)	-
End point (optional)	EPn	Isochronous (out)	128 bytes	128 x 2 (variable)	-
End point (optional)	EPn	Bulk-in	64 bytes	64 x 2 (128 bytes)	Possible
End point (optional)	EPn	Bulk-out	64 bytes	64 x 2 (128 bytes)	Possible
End point (optional)	EPn	Interrupt (in)	64 bytes	64 bytes (variable)	-

Rev. 1.0, 04/02, page 1 of 80



Figure 1.1 shows an example of a system configuration.

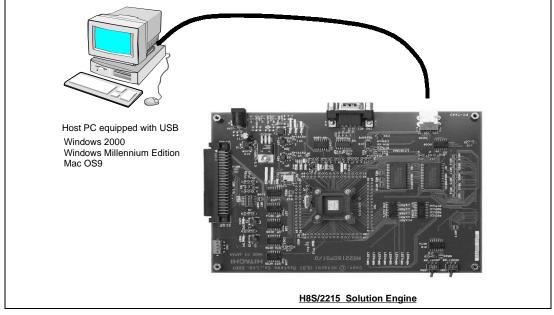


Figure 1.1 System Configuration Example

This system is configured of the H8/2215 Solution Engine made by Hitachi ULSI Systems Co., Ltd. (hereafter referred to as the MS2215CP) and a PC containing Windows 2000/Windows Millennium Edition or Mac OS9 operating system.

By connecting the host PC and the MS2215CP through USB, the SRAM in the MS2215CP can be accessed as a RAM disk, enabling data in the SRAM of the MS2215CP to be stored in and loaded from the host PC.

It is also possible to use the USB Mass Storage Class (Bulk-Only Transport) device driver that comes as an accessory with the operating systems listed above.

This system offers the following features.

- 1. The sample program can be used to evaluate the USB module of the H8/2215 quickly.
- 2. The sample program supports USB control transfer and bulk transport.
- 3. An E6000 can be used, enabling efficient debugging.
- 4. Additional programs can be created to support interrupt transfer and isochronous transfer. *

Note: * Interrupt transfer and isochronous transfer programs are not provided, and will need to be created by the user.

Rev. 1.0, 04/02, page 2 of 80

Section 2 Overview of the USB Mass Storage Class (Bulk-Only Transport)

This section describes the USB Mass Storage Class (Bulk-Only Transport).

We hope that it will provide a convenient reference for use when developing USB storage-related systems. For more detailed information on standards, please see the following:

- Universal Serial Bus Mass Storage Class Specification Overview Revision 1.1
- Universal Serial Bus Mass Storage Class Bulk-Only Transport Revision 1.0

2.1 USB Mass Storage Class

USB Mass Storage Class is a class of standards that apply to large-scale memory (storage) devices that are connected to a host PC and handle reading and writing of data.

<u>In order to let the PC know that a function is in this class, a value of 0x08 must be entered in the bInterfaceClass field of the Interface Descriptor.</u>

When transferring data between the host PC and the function, four transport methods defined by the USB are used (control transfer, bulk transport, interrupt transfer, and isochronous transfer). Protocol codes determine the transport method and how it is used.

There are two types of data transport protocols:

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

As its name indicates, USB Mass Storage Class Bulk-Only Transport is a data transport protocol that only uses bulk transport.

USB Mass Storage Class Control/Bulk/Interrupt (CBI) Transport is a data transport protocol that uses control transfer, bulk transport, and interrupt transfer. CBI Transport is further subdivided into a data transport protocol that uses interrupt transfer, and one that does not use interrupt transfer.

The sample programs provided here use USB Mass Storage Class Bulk-Only Transport as the data transport protocol.

When the host PC uses a device in order to load and save data, instructions (commands) are provided by the host PC to the function. The function then executes those commands to load and save data. The commands sent by the host PC to the function are defined in the form of sub-class code.

2.2 Sub-Class Code

Sub-class codes are values that indicate the command format sent from the host PC to a function by means of command transport. There are seven types of command formats, described in table 2.1.

Table 2.1

Sub-Class Code	Command Standards
0x01	Reduced Block Commands (RBC), T10/1240-D
0x02	Attachment Packet Interface (ATAPI) for CD-ROMs. SFF-8020i,
	Multi-Media Command Set 2 (MMC-2)
0x03	Attachment Packet Interface (ATAPI) for Tape. QIC-157
0x04	USB Mass Storage Class UFI Command Specification
0x05	Attachment Packet Interface (ATAPI) for Floppies. SFF-8070i
0x06	SCSI Primary Commands –2 (SPC-2), Revision 3 or later

In order to tell the host PC the command format supported by the device, a sub-class code value must be entered in the bINterfaceSubClass field of the Interface Descriptor.

The sample programs used here use a sub-class code value of 0x06, which indicates the SCSI Primary Commands.

2.3 Bulk-Only Transport

With Bulk-Only Transport, data is transferred between the host PC and a function using bulk data transport only.

Bulk transport can be divided into two types, depending on the direction in which the data is sent. If data is sent from the host controller to the function, bulk-out transport is used. If data is being sent to the host controller from the function, bulk-in transport is used.

With Bulk-Only Transport, a combination of bulk-out transport and bulk-in transport determined in advance is used to transfer data between the host and the function. Bulk-Only Transport always uses the combination of bulk transports shown in figure 2.1. These bulk transports have different meanings, but they are handled as stages (transports).

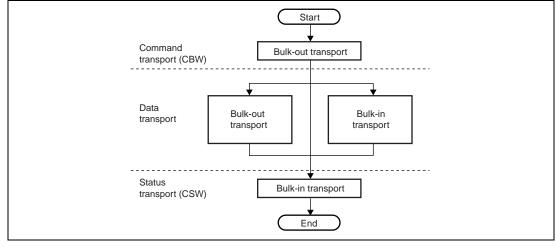


Figure 2.1 Relationship between Transfer Methods and Transports

In order to tell the host PC that the Bulk-Only Transport protocol is being used, a value of 0x50 must be entered in the bInterfaceProtocol field of the Interface Descriptor.

2.3.1 Command Transport

In command transport, commands are sent from the host PC to the function using bulk-out transport. This command packet is defined as the Command Block Wrapper (CBW), and Bulk-Only Transport must always begin with the CBW.

The CBW is sent from the host PC as a 31-byte packet, using bulk-out transport.

It is sent using the format shown in table 2.2.

Table 2.2

	7	6	5	4	3	2	1	0
00-03	dCBW:	Signature	!					
04-07	dCBW ⁻	dCBWTag						
08-0B	dCBWI	DataTran	sferLength					
0C	bmCB\	WFlags						
0D	Reserv	red (0)			bCB\	VLUN		
0E	Reserv	red (0)		bCB\	VCBLengt	h		
0F-1E	CBWC	В						

The fields are explained below.

dCBWSignature: This field identifies the data packet as a CBW.

The value is 43425355h (Little Endian).

dCBWTag: This is the command block tag. It is used to connect the CSW with

its corresponding CBW, and is specified by the host PC.

dCBWDataTransferLength: This is the length of the data planned for transport.

If this is 0, no data transport exists.

bmCBWFlags: If bit 7 of this field is 0, data is transported using bulk-out transport,

and if it is 1, bulk-in transport is used. Bits 0 to 6 are fixed at 0.

bCBWLUN: This is the Logical Unit Number of the device sending the

command block.

bCBWCBLength: This indicates the number of valid bytes for the next CBWCB field.

CBWCB: This field stores the command block to be executed by the function.

The command that the host PC wants to execute (the SCSI command in this sample program) is entered in this field.

2.3.2 Status Transport

Status transport is used to send the results of command execution from the function to the host PC, using bulk-in transport.

This status packet is defined by the Command Status Wrapper (CSW). Bulk-Only Transport must always end with the CSW.

The CSW is sent to the host as a 13-byte packet, using bulk-in transport.

It is sent in the format shown in table 2.3.

Table 2.3

	7	6	5	4	3	2	1	0
0-3	dCSWSignature							
4-7	dCSWTag							
8-B	dCSWDataResidue							
С				bC:	SWStatus			

The fields are explained below.

Rev. 1.0, 04/02, page 6 of 80



dCSWSignature: This field identifies the data packet as the CSW.

The value is 53425355h (Little Endian).

dCSWTag: This is the command block tag. It ties the CBW to the CSW, and the same

value is entered here as that of the dCBWTag field of the CBW.

dCSWDataResidue: This reports any differences in the value of the CBW

dCBWDataTransferLength and the actual amount of data processed by the

function.

bCSWStatus: This indicates whether or not a command has been successfully executed. If

the command was executed successfully, the function sets 0x00 in this field.

Any value other than 0 indicates that the command was not executed

successfully. Error values are as follows: 0x01 indicates a failed command,

and 0x02 indicates a phase error.

2.3.3 Data Transport

Data transport is used to transfer data between the host PC and the function. For example, with the Read/Write command (see section 4.6), the actual data of the various storage sectors is sent using data transport.

Data transport is configured of multiple bus transactions.

Data transfers carried out using data transport use either bulk-out or bulk-in transport. The bmCBWFlags field of the CBW data determines which type of transport is used.

Data transport (bulk-out transport)

This section explains how data is transferred when bulk-out data transport is used.

This status is set if bit 7 of the bmCBWFlags field of the CBW data is 0, and the dCBWDataTransferLength field of the CBW data is not 0.

Here, the function receives the anticipated length of the data indicated by the dCBWDataTransferLength field of the CBW data. The data transferred at this point is needed when the SCSI command specified by the CBWCB field of the CBW data is executed.

Data transport (bulk-in transport)

This section explains how data is transferred when bulk-in data transport is used.

This status is set if bit 7 of the bmCBWFlags field of the CBW data is 1, and the dCBWDataTransferLength field of the CBW data is not 0.

Here, the anticipated length of the data indicated by the dCBWDataTransferLength field of the CBW data is sent to the host PC. The data transferred at this point is the result produced when the SCSI command specified by the CBWCB field of the CBW data was executed.

2.3.4 Class Commands

Class commands are commands that are defined by the various USB classes. They use control transfer.

When USB Mass Storage Class Bulk-Only Transport is used as the data transport protocol, there are two types of commands that must be supported. The class commands are indicated in table 2.4.

Table 2.4 Class Commands

bRequest Field Value	Command	Meaning of Command
255 (0xFF)	Bulk-Only Mass Storage Reset	Resets the interface
254 (0xFE)	Get Max LUN	Checks the number of LUNs supported

When the Bulk-Only Mass Storage Reset command is received, the function resets all of the interfaces used in USB Mass Storage Class Bulk-Only Transport.

When the Get Max LUN command is received, the function returns the largest logical unit number that can be used. In the sample system used here, there is one logic unit, so a value of 0 will be returned to the host.

2.4 SCSI Transparent Command Set Sub-Class Code

The various commands must be processed in response to the sub-class commands in the CBW sent to the function by the host PC.

In this sample program, the nine SCSI commands shown in table 2.5 are supported. If a command is not supported, the CSW will be used to inform the host PC that the command failed.

Table 2.5 Supported Commands

Operation Code	Command Name	Command Operation
12	INQUIRY	Tells the host the drive information.
25	READ CAPACITY	Tells the host the media sector information.
28	READ(10)	Reads the specified sector volume data from a specified sector.
2A	WRITE(10)	Writes the specified sector volume data to a specified sector.
03	REQUEST SENSE	If an error occurred for the previous command, this tells the host what kind of error occurred.
1A	MODE SENSE(10)	Tells the host the drive status.
1E	PREVENT ALLOW MEDIUM REMOVAL	Inhibits/enables installing and removing media.
00	TEST UNIT READY	Checks whether or not a medium can be used.
2F	VERIFY(10)	Confirms whether or not the data in a medium can be accessed.

RENESAS

Section 3 Development Environment

This chapter looks at the development environment used to develop this system. The devices (tools) listed below were used when developing the system.

- H8S/2215 Solution Engine (hereafter called the MS2215CP; type number: MS2215CP01-C/S) manufactured by Hitachi ULSI Systems Co., Ltd.
- E6000 (HS2214EPI61H) Emulator manufactured by Hitachi, Ltd.
- H8S/2215 Series TFP120 User System Interface Cable (hereafter called the H8S/2215 user cable; type number: HS2215ECN61H) manufactured by Hitachi, Ltd.
- PC (Windows 95/98) equipped with an ISA, PCI, or PCMCIA slot
- PC (Windows 2000/Windows Millennium Edition or Mac OS9) to serve as the USB host
- USB cable
- Hitachi Debugging Interface (hereafter called the HDI) manufactured by Hitachi, Ltd.
- Hitachi Embedded Workshop (hereafter called the HEW) manufactured by Hitachi, Ltd.

3.1 Hardware Environment

Figure 3.1 shows device connections.

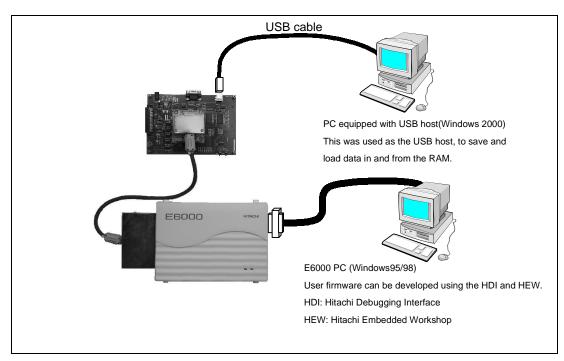


Figure 3.1 Device Connections

1. MS2215CP

Some jumper settings on the MS2215CP board must be changed from those at shipment. Before turning on the power, ensure that the jumpers are set as follows. There is no need to change any other jumpers.

Table 3.1 Jumper Settings

At Shipment	After Change	Jumper Function
J9 1-2: Closed	J9 2-3: Closed	Switches the EXTAL48 pin level

2. USB host PC

A PC with Windows 2000/Windows Millennium Edition or Mac OS9 installed, and with a USB port, is used as the USB host. This system uses USB Mass Storage Class (Bulk-Only Transport) device drivers installed as a standard part of the Windows 2000 system, and so there is no need to install new drivers.

3. E6000

The ISA is used for the communication interface between the E6000 PC and the E6000 emulator.

The E6000 I/F board should be inserted into an ISA slot and connected to the E6000 via an interface cable. Then, the E6000 should be connected to the MS2215CP via an H8S/2215 user cable. After connection, start the HDI and perform emulation.

3.2 Software Environment

A sample program, as well as the compiler and linker used, are explained.

3.2.1 Sample Program

Files required for the sample program are all stored in the H8S2215 folder. When this entire folder with its contents is moved to a PC on which HEW and HDI have been installed, the sample program can be used immediately. Files included in the folder are indicated in figure 3.2 below.

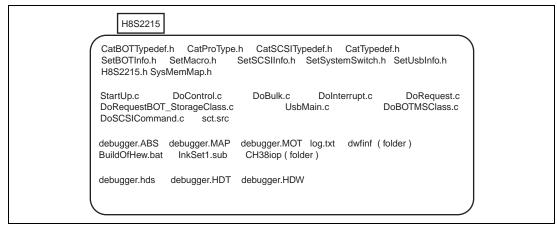


Figure 3.2 Files Included in the Folder

3.2.2 Compiling and Linking

The sample program is compiled and linked using the following software.

Hitachi Embedded Workshop Version 1.0 (release 9) (hereafter HEW)

When HEW is installed in C:\Hew, the procedure for compiling and linking the program is as follows.

First, a folder named Tmp should be created below the C:\Hew folder for use in compiling. (figure 3.3)

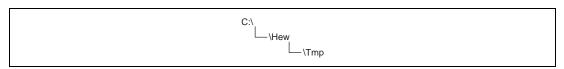


Figure 3.3 Creating a Working Folder

Next, the folder in which the sample program is stored (H8S/2215) should be copied to C:\Usr (or can be copied to any location, then "C:\Usr\h8s2215" written in the debugger.hds file should be modified to the path to the copied folder). In addition to the sample program, this folder contains a batch file named BildOfHew.bat. This batch file sets the path, specifies compile options, specifies a log file indicating the compile and linking results, and performs other operations. When BildOfHew.bat is executed, compiling and linking are performed. As a result, an executable file named debugger.ABS is created within the folder. At the same time, a map file named debugger.MAP and a log file named log.txt are created. The map file indicates the program size and variable addresses. The compile results (whether there are any errors etc.) are recorded in the log file. (figure 3.4)

Note: If HEW is installed in a folder other than C:\Hew, the compiler path setting and settings for environment variables used by the compiler in BildOfHew.bat, as well as the library settings in InkSet1.sub, must be changed. Here the compiler path setting should be changed to the path of ch38.exe, and the setting for the environment variable ch38 used by the compiler should be set to the folder of machine.h, and the setting of ch38tmp should specify the work folder for the compiler. The library setting should specify the path of c8s2ba.lib.

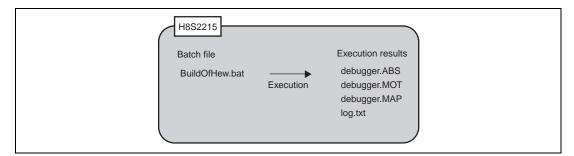


Figure 3.4 Compile Results

3.3 Loading and Executing the Program

Figure 3.5 shows the memory map for the sample program.

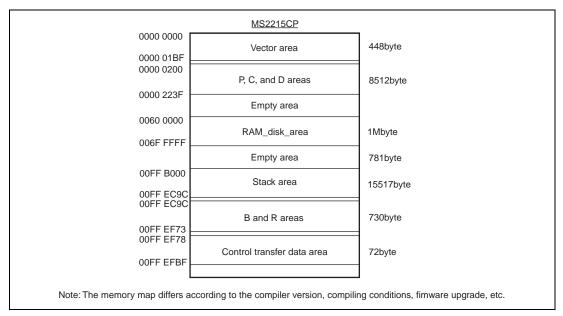


Figure 3.5 Memory Map

As shown in figure 3.5, this sample program allocates areas for vectors P, C, and D to the on-chip ROM area (E6000 emulation memory) in area 1, the stack, B, R, and control transfer areas to the on-chip RAM, and the RAM disk area to the SRAM. These memory allocations are specified by the InkSet1.sub file in the H8S2215 folder. When modifying the program allocation, this file must be modified.

3.3.1 Loading the Program

In order to load the sample program into the MS2215CP, the following procedure is used.

- Connect the E6000 PC in which the HDI has been installed to the E6000.
- Connect the E6000 to the MS2215CP through an H8S/2215 user cable.
- Turn on the power to the E6000 PC to start up the machine.
- Turn on the power to the MS2215CP.
- Execute debugger.hds in the H8S2215 folder.

Through the above procedure, the sample program can be loaded into the MS2215CP.

3.3.2 Executing the Program

In order to execute the program which was loaded in section 3.3.1 above, the program counter (PC) must be set appropriately.

Select Register Window from the View menu to open the Registers window. On double-clicking the numerical area of the register (PC) in the window, a dialog box appears, and the register value can be changed. Use this dialog box to set the PC to H'0000 0200.

After making the above settings, select Go from the Run menu to execute the program.

3.4 Using the RAM Disk

The following describes an example in which Windows 2000 is used.

After the program has been run, the Series B connector of the USB cable is inserted into the MS2215CP, and the Series A connector on the opposite side is connected to the USB host PC.

After the emulation used for control transfer and bulk transport has ended, USB Large-Size Storage Device is displayed under USB Controller in the Device Manager, and Hitachi EX RAM Disk USB Device is displayed under Disk Drive. As a result, the host PC recognizes the MS2215CP as the storage device, and the local disk is mounted in the microcomputer.

Next, the local disk needs to be formatted.

Select Local Disk and click with the right button of the mouse to display a floating menu. Select Format. A format selection window for the drive is displayed. Enter the necessary format settings. Check to make sure FAT has been selected for the file system, and click on the Start button.

A format confirmation window is displayed. Click on the OK button.

When the formatting has been completed, a message window is displayed. Click on the OK button.

The screen returns to the drive format selection window. Click on the Close button to exit the procedure.

The MS2215CP can now be used as the RAM disk for USB connection.

Section 4 Overview of the Sample Program

In this section, features of the sample program and its structure are explained. This sample program runs on the MS2215CP, which works as a RAM disk, and initiates USB transfers by means of interrupts from the USB function module. Of the interrupts from modules in the H8S/2215, there are three interrupts related to the USB function module: EXIRQ0, EXIRQ1, and IRQ6, but in this sample program, only EXIRQ0 is used.

Features of this program are as follows.

- Control transfer can be performed.
- Bulk-out transfer can be used to receive data from the host controller.
- Bulk-in transfer can be used to send data to the host controller.
- It operates as a RAM disk that supports SCSI commands.

4.1 State Transition Diagram

Figure 4.1 shows a state transition diagram for this sample program. In this sample program, as shown in figure 4.1, there are transitions between four states.

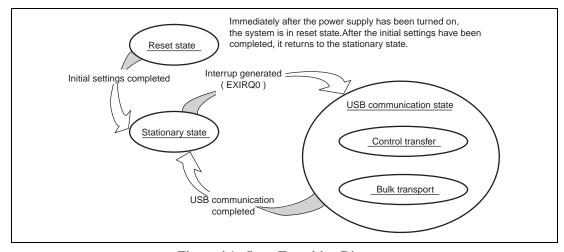


Figure 4.1 State Transition Diagram

Reset State

Upon power-on reset and manual reset, this state is entered. In the reset state, the H8S/2215 mainly performs initial settings.

• Stationary State

When initial settings are completed, a stationary state is entered in the main loop.

USB Communication State

In the stationary state, when an interrupt from the USB module occurs, this state is entered. In the USB communication state, data transfer is performed by a transfer method according to the type of interrupt. The interrupts used in this sample program are indicated by interrupt flag registers 0 to 3 (UIFR0 to UIFR3), and there are nine interrupt types in all. When an interrupt factor occurs, the corresponding bits in UIFR0 to UIFR3 are set to 1.

4.2 USB Communication State

The USB communication state can be further divided into two states according to the transfer type (see figure 4.2). When an interrupt occurs, first there is a transition to the USB communication state, and then there is further branching to a transfer state according to the interrupt type. The branching method is explained in section 5, Sample Program Operation.

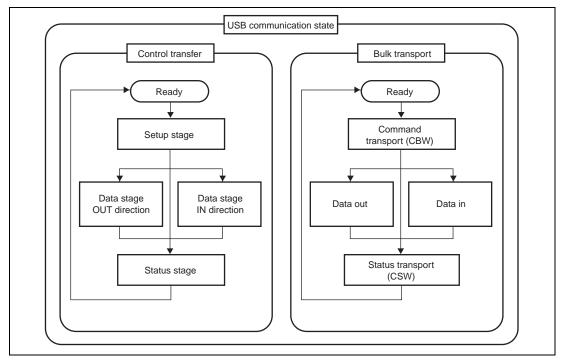


Figure 4.2 USB Communication State

4.2.1 Control Transfer

Control transfer is used mainly for functions such as obtaining device information and specifying device operating states. For this reason, when the function is connected to the host PC, control transfer is the first transport to be carried out.

Transport processing for control transfer is carried out in a series of two or three stages. These stages are a setup stage, a data stage, and a status stage.

4.2.2 Bulk Transport

Bulk transport has no time limitations, so it is used to send large volumes of data with no errors. The data transport speed is not guaranteed, but the data contents are guaranteed. With USB Mass Storage Class (Bulk-Only Transport), bulk transport is used to transfer storage data between the host PC and the function.

Transport processing for USB Mass Storage Class (Bulk-Only Transport) is carried out in a series of two or three stages. These stages are command transport (CBW), data transport, and status transport (CSW).

4.3 File Structure

This sample program consists of eight source files and eleven header files. The overall file structure is shown in table 4.1. Each function is arranged in one file by transfer method or function type. Figure 4.3 shows the layered configuration of these files.

Table 4.1 File Structure

File Name	Principle Role			
StartUp.c	Microcomputer default settings			
UsbMain.c	Judging the causes of interrupts			
OSDIVIAITI.C	Sending and receiving packets			
DoRequest.c	Processing setup commands issued by the host			
DoRequestBOT_StorageClass. c	Processing Mass Storage Class (Bulk-Only Transport) class commands			
DoControl.c	Executing control transfer			
DoBulk.c	Executing bulk transport			
DoBOTMSClass.c	Executing Mass Storage Class (Bulk-Only Transport)			
DoSCSICommand.c	Analyzing and processing SCSI commands			
h8s2215.h	Defining H8S/2215 registers			
SysMemMap.h	Defining MS2215CP memory map addresses			
CatProType.h	Prototype declarations			
CatTypedef.h	Defining the basic structures used in USB firmware			
CatBOTTypedef.h	Defining structures used for Bulk-Only Transport			
CatSCSITypedef.h	Defining structures used for SCSI			
SetUsbInfo.h	Default settings of variables needed to support USB			
SetBOTInfo.h	Default settings of variables needed to support Bulk-Only Transport			
SetSCSIInfo.h	Default settings of variables needed to support SCSI commands			
SetSystemSwitch.h	System operation settings			
SetMacro.h	Defining macros			

Target data file Operation: Interprets SCSI commands and carries out RAM disk operations Application Relevant files: DoSCSICommand.c layer CatSCSITypedef.h SetSCSIInfo.h Class file Operation: Carries out Mass Storage Class (Bulk-Only Transport) operations and supports class commands Class layer Relevant files: DoBOTMSClass.c CatBOTTypedef.h SetBOTInfo.h USB Vendor commands Bulk transport Standard commands Class commands device Operation: Carries out Operation: Carries out Operation: Carries out layer responses to standard responses to class commands bulk transport commands Relevant file: operations Relevant file: DoRequest.c DoRequestBOT_Storage Relevant file: DoBulk.c Class.c Control transfer Operation: Carries out control transfer operations Relevant file: DoControl.c USB common variables Operation: Carries out reception of packet data, transmission of packet data, Endian conversion, various types of settings, and other necessary operations regardless of transport method Relevant file: UsbMain.c CatTypedef.h SetUsbInfo.h USB bus USB hardware interface

Figure 4.3 Layered Configuration of the Storage Class (BOT) Firmware

4.4 Purposes of Functions

Table 4.2 shows functions contained in each file and their purposes.

Table 4.2-1 StartUp.c

File in Which Stored	Function Name	Purpose
	SetPowerOnSectio n	Sets BSC, terminals, and interrupt controller, calls initialization routines, and shifts to the main loop
StartUp.c	_INITSCT	Copies variables that have default settings to the RAM work area
	InitMemory	Clears the RAM area used in bulk communication
	InitSystem	Specifies the USB clock, system interrupts, and masks

When a power-on reset or manual reset is carried out, the SetPowerOnSection of the StartUp.c file is called. At this point, the H8S/2215 default settings are entered and the RAM area used for bulk transport is cleared.

Table 4.2-2 UsbMain.c

File in Which Stored	Function Name	Purpose
UsbMain.c	BranchOfInt	Discriminates interrupt factors, and calls function according to interrupt
	GetPacket	Writes data transferred from the host controller to RAM
	GetPacket4	Writes data transferred from the host controller to RAM in longwords. Ring buffer support version (not used by this sample program).
	GetPacket4S	Writes data transferred from the host controller to ROM in longwords
	PutPacket	Writes data for transfer to the host controller to the USB module
	PutPacket4	Writes data for transfer to the host controller to the USB module in longwords. Ring buffer support version (not used by this sample program).

File in Which Stored	Function Name	Purpose
UsbMain.c	PutPacket4S	Writes data for transfer to the host controller to the USB module in longwords. High-speed version.
	SetControlOutCont ents	Overwrites data with that sent from the host
	SetUsbModule	Sets USB module initial settings
	ActBusReset	Clears FIFO on receiving bus reset
	ActBusVcc	Pulls up D+ and controls USB module when the USB cable is connected or disconnected
	ConvRealn	Reads data of a specified byte length from a specified address
	ConvReflexn	Reads data of a specified byte length from specified addresses, in reverse order

In UsbMain.c, interrupt factors are discriminated by the USB interrupt flag register, and functions are called according to the interrupt type. Also, packets are sent and received between the host controller and function modules.

Table 4.2-3 DoRequest.c

File in Which Stored	Function Name	Purpose
DoRequest.c	DecStandardComm ands	Decodes command issued by host controller, and processes standard commands
	DecVenderComma nds	Processes vendor commands

During control transfer, commands sent from the host controller are decoded and processed. In this sample program, a vendor ID of 045B (vendor: Hitachi) is used. When the customer develops a product, the customer should obtain a vendor ID at the USB Implementers' Forum. Because vendor commands are not used, DecVenderCommands does not perform any action. In order to use a vendor command, the customer should develop a program.

Table 4.2-4 DoRequestBOT_StorageClass.c

File in Which Stored	Function Name	Purpose
DoRequestBOT_	DecBOTClass	Processes USB Mass Storage Class (Bulk-Only
StorageClass.c	Commands	Transport) commands

This function carries out processing according to the Mass Storage Class (Bulk-Only Transport) commands (Bulk-Only Mass Storage Reset and Get Max LUN).

The Bulk-Only Mass Storage Reset command resets all of the interfaces used in Bulk-Only Transport.

The Get Max LUN command returns the largest logical unit number used by peripheral devices. In this sample program, there is one logical unit, so a value of 0 is returned to the host.

Table 4.2-5 DoControl.c

File in Which Stored	Function Name	Purpose
DoControl.c	ActControl	Controls the setup stage of control transfer
	ActControlIn	Controls the data stage and status stage of control IN transport (transport in which the data stage is in the IN direction)
	ActControlOut	Controls the data stage and status stage of control OUT transport (transport in which the data stage is in the OUT direction)

When a control transfer interrupt (EP0oTS) is generated, ActControl obtains the command, and decoding is carried out by DecStandardCommands. Next, the data stage and status stage are carried out using either ActControlIn or ActControlOut, depending on the type of command.

Table 4.2-6 DoBulk.c

File in Which Stored	Function Name	Purpose
	ActBulkOut	Performs bulk-out transfer
DoBulk.c	ActBulkIn	Performs bulk-in transfer
	ActBulkInReady	Performs preparations for bulk-in transfer
		_

These functions carry out processing involving bulk transport. ActBulkInReady is not used in Mass Storage Class (Bulk-Only Transport).

Table 4.2-7 DoBOTMSClass.c

File in Which Stored	Function Name	Purpose
DoBOTMS Class.c	ActBulkOnly	Divides Bulk-Only Transport into separate stages
	ActBulkOnlyComm and	Controls CBW for Bulk-Only Transport
	ActBulkOnlyIn	Controls data transport and status transport (when the data stage is in the IN direction)
	ActBulkOnlyOut	Controls data transport and status transport (when the data stage is in the OUT direction)

With DoBOTMSClass.c, control of the two or three stages of the Mass Storage Class (Bulk-Only Transport) is carried out, and operation is carried out in accordance with the specifications.

Table 4.2-8 DoSCSICommand.c

File in Which Stored	Function Name	Purpose
DoSCSI	DecBotCmd	Processes SCSI commands sent from the host using Bulk-Only Transport
Command.c		

The DoSCSICommand.c function is used to analyze SCSI commands sent from the host PC and prepare for the next data transport or status transport.

Figure 4.4 shows the interrelationship between the functions explained in table 4.2. The upper-side functions can call the lower-side functions. Also, multiple functions can call the same function. In the stationary state, SetPowerOnSection calls other functions, and in the case of a transition to the USB communication state which occurs on an interrupt, BranchOfInt calls other functions. Figure 4.4 shows the hierarchical relation of functions; there is no order for function calling. For information on the order in which functions are called, please refer to the flow charts of section 5, Sample Program Operation.

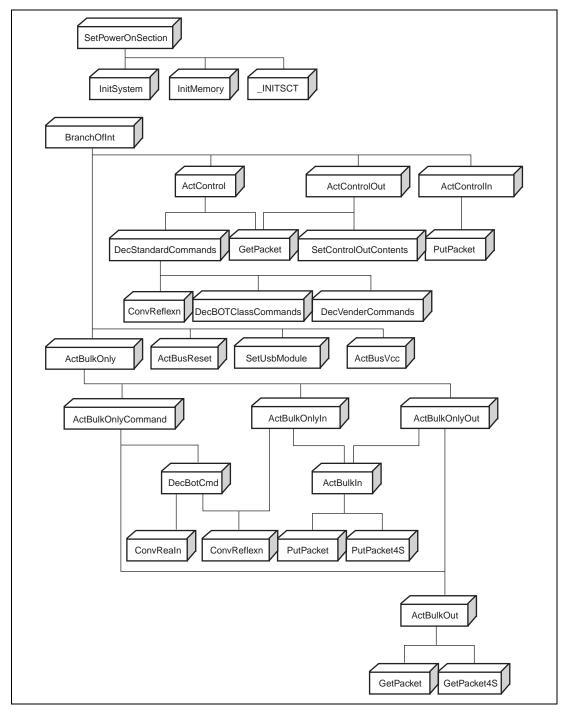


Figure 4.4 Interrelationship between Functions

4.5 RAM Disk

In the sample program provided here, the SRAM in the MS2215CP is selected as the disk device, and the host PC is notified that the MS2215CP (function) is a disk.

As shown in figure 4.5, the disk device of the function has a master boot block and a partition boot block. When the system is booted, an initialization routine is used to write the master boot block and the partition boot block to the RAM disk area on the SRAM.

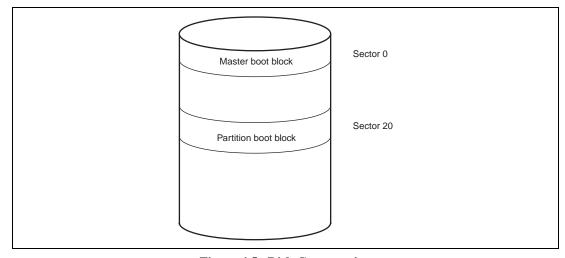


Figure 4.5 Disk Construction

SCSI commands are used to allow function access from the host PC (saving and loading data). In order to work with SCSI commands, the user needs to understand the construction shown in figure 4.5 and then write the operation.

4.6 Operation of SCSI Commands That Are Supported

Table 4.3 shows the SCSI commands that are supported by the sample program.

Table 4.3 SCSI Command Operations

Command Name	Transport Name	Operation Content				
	CBW	This decodes a command and recognizes it as an INQUIRY command. It then prepares to send the INQUIRY information (96 bytes) stored in the ROM.				
INQUIRY	Data	This sends the INQUIRY information to the host PC using bulk-in transport.				
	CSW	This sends the results of executing a command to the PC. If the data being sent is 96 bytes or less, the transmission will end successfully.				
READ	CBW	This decodes the command and recognizes it as a READ CAPACITY command. It then reads the number of bytes per sector, which is stored in the partition boot block on the disk device open on the SRAM, and the value stored for the total number of sectors on the disk, and prepares to send the READ CAPACITY information (8 bytes).				
CAPACITY	Data	This sends the READ CAPACITY information to the host PC using bulk-in transport.				
	CSW	This sends the results of the command execution to the host PC. The transmission is completed successfully as long as the data consists of 8 bytes or less.				
	CBW	This decodes the command and recognizes it as the READ (10) command. It then prepares to send the data for a specified read sector volume from the Disk device open on the SRAM.				
READ(10)	Data	This sends the data from the read sectors to the host PC using bulk-in transport.				
	CSW	This sends the results of executing the READ (10) command to the host computer. The transmission is completed successfully as long as the transmitted data is less than the number of bytes read.				

Command Name	Transport Name					
MDITE(40)	CBW	This decodes the command and recognizes it as the WRITE (10) command. It then prepares to receive the data of the specified sector volume from the specified write sector in the Disk device open on the SRAM.				
WRITE(10)	Data	This receives the write sector data from the host PC using bulk-out transport.				
	CSW	This notifies the host PC that the operation has been completed successfully.				
DE011507	CBW	This decodes the command and recognizes it as the REQUEST SENSE command. It then prepares to send the returned value (the results of executing the previous SCSI command).				
REQUEST SENSE	Data	This sends the returned value to the host PC using bulk-in transport.				
	CSW	This sends the results of the command execution to the host PC. The transmission is completed successfully as long as the data consists of 8 bytes or less.				
PREVENT ALLOW MEDIUM REMOVAL	CBW	This decodes the command and recognizes it as the PREVENT ALLOW MEDIUM REMOVAL command. It then prepares to notify the host PC that the operation has been successfully completed. The storage media used with the sample software is an SRAM that is permanently installed, and cannot be removed. For this reason, the returned value for this command always indicates that the operation has been successfully completed.				
	Data	Data transport does not exist for this command.				
	CSW	This notifies the host PC that the operation has been completed successfully.				
TEST UNIT READY	CBW	This decodes the command and recognizes it as the TEST UNIT READY command. It then prepares to notify the host PC that the operation has been successfully completed. The storage medium used with this sample software is an SRAM, and access is always enabled while the program is being executed. For this reason, the returned value for this command always indicates that the operation has been successfully completed.				
	Data	Data transport does not exist for this command.				
	CSW	This notifies the host PC that the operation has been completed successfully.				

Command Name	Transport Name	Operation Content			
VERIFY(10)	CBW	This decodes the command and recognizes it as the VERIFY(10) command. It then prepares to notify the host PC that the operation has been successfully completed. The storage medium used with this sample software is an SRAM, and access is always enabled while the program is being executed. For this reason, the returned value for this command always indicates that the operation has been successfully completed.			
	Data	Data transport does not exist for this command.			
	CSW	This notifies the host PC that the operation has been completed successfully.			
	CBW	This decodes the command and recognizes it as the MODE SENSE (10) command. It then prepares to send the mode parameters to the host PC.			
MODE		The sample software provides values for only the mode parameter header.			
SENSE(10)	Data	This sends the mode parameters to the host PC using bulk-in transport.			
	CSW	This sends the results of the command execution to the host PC. The transmission is completed successfully as long as the data consists of 8 bytes or less.			
	CBW	This decodes the command and, if it is an unsupported command, specifies INVALID FIELD IN CDB for the returned value of the REQUEST SENSE command. It then prepares to transport the data.			
Commands that are not supported	Data	If the host PC has requested data using bulk-in transport, this sends the same volume of data (0x00) as that requested by the host PC.			
		If the host PC has sent data using bulk-out transport, the number of bytes received are counted.			
		If there is no data transport, no operation is carried out.			
	CSW	This sends a command file to the host PC.			

4.7 Processing If an Error Occurs

The errors that may occur during a Mass Storage Class (Bulk-Only Transport) transmission between the host PC and function, and how the function operates when an error occurs are described below.

The Bulk-Only Transport standard defines the following two types of errors:

- Invalid CBW
- Operation expected by the host PC and operation planned by the function (operation specified by the SCSI command) do not match (10 cases)

The Bulk-Only Transport standard does not cover any other states.

There are 13 states for data transfer between the host PC and a function as shown in Tables 4.4 and 4.5. Cases 1, 6 and 12 are normal transport states.

Table 4.4 Data Transport States between Host PC and Function.

		What the Host PC Expects			
		No Data Transport	Data Reception from Function	Data Send to Function	
	No data transport	(1) Hn = Dn	(4) Hi > Dn	(9) Ho > Dn	
			(5) Hi > Di		
What	Data send to host PC	(2) Hn < Di	(6) Hi = Di	(10) Ho < > Di	
the function plans			(7) Hi < Di		
				(11) Ho > Do	
	Data reception from host	(3) Hn < Do	(8) Hi < > Do	(12) Ho = Do	
	PC			(13) Ho < Do	

Table 4.5 Explanation of Data Transport States between Host PC and Function

Case No.	Relation between Host PC and Function
1	The host PC expects no data transport and the function plans no data transport.
2	The host PC expects no data transport but the function plans to send data to the host PC
3	The host PC expects no data transport but the function plans to receive data from the host PC.
4	The host PC expects to receive data from the function but the function plans no data transport to the host PC.
5	The amount of data the function sends to the host PC is less than the amount of data the host PC expected to receive from the function.
6	The amount of data the function sends to the host PC is equal to the amount of data the host PC expected to receive from the function.
7	The amount of data the function sends to the host PC is greater than the amount of data the host PC expected to receive from the function.
8	The host PC expects to receive data from the function but the function plans to receive data from the host PC.
9	The host PC expects to send data to the function but the function plans no data transport to the host PC.
10	The host PC expects to send data to the function but the function plans to send data to the host PC.
11	The amount of data the function receives from the host PC is less than the amount of data the host PC expected to send to the function.
12	The amount of data the function receives from the host PC is equal to the amount of data the host PC expected to the function.
13	The amount of data the function receives from the host PC is greater than the amount of data the host PC expected to send to the function.

Table 4.6 shows sample error conditions that may be generated.

Table 4.6 Sample Error Conditions

Case No.	Relation between Host PC and Function
2	When a READ command is issued from the host PC, the amount of data to be transported in the USB data transport is 0 while the amount of data specified by the SCSI command is a value other than 0.
3	When a WRITE command is issued from the host PC, the amount of data to be transported in the USB data transport is 0 while the amount of data specified by the SCSI command is a value other than 0.
4	When a READ command is issued from the host PC, the amount of data to be transported in the USB data transport is 0 while the amount of data specified by the SCSI command is 0.
5	When a READ command is issued from the host PC, the amount of data specified by the SCSI command is less than the amount of data to be transported in the USB data transport.
7	When a READ command is issued from the host PC, the amount of data specified by the SCSI command is greater than the amount of data to be transported in the USB data transport.
8	Even though a WRITE command has been issued from the host PC, the host PC requests for data in the USB data transport.
9	When a WRITE command is issued from the host PC, the amount of data to be transported in the USB data transport is a value other than 0 while the amount of data specified by the SCSI command is 0.
10	Even though a READ command has been issued from the host PC, the host PC sends data in the USB data transport.
11	When a WRITE command is issued from the host PC, the amount of data specified by the SCSI command is less than the amount of data to be transported in the USB data transport.
13	When a WRITE command is issued from the host PC, the amount of data specified by the SCSI command is greater than the amount of data to be transported in the USB data transport.

Table 4.7 shows how a function operates when each error condition occurs.

Table 4.7 Function Operation for Each Error Condition

Case No.	Relation between Host PC and Function					
2, 3	Set 0x02 as the CSW status.					
4, 5	 The function adds data to become equal to the data length set in dCBWDataTransferLength and then sends data to the host PC. 					
	 Set the amount of data added in the data transport in dCBWDataResidue of CSW. 					
	Set 0x01 as the CSW status.					
7, 8	 The function sends data to the host PC up to the data length set in dCBWDataTransferLength. 					
	Set 0x02 as the CSW status.					
9, 11	 The function receives data from the host PC up to the data length set in dCBWDataTransferLength. 					
	 Set the difference between the amount of data received in the data transport and the amount of data processed by the function in dCBWDataResidue of CSW. 					
	Set 0x01 as the CSW status.					
10, 13	The function receives data from the host PC up to the data length set in dCBWDataTransferLength.					
	Set 0x02 as the CSW status.					

Figures 4.6 to 4.8 show the processing when a data transport error occurs.

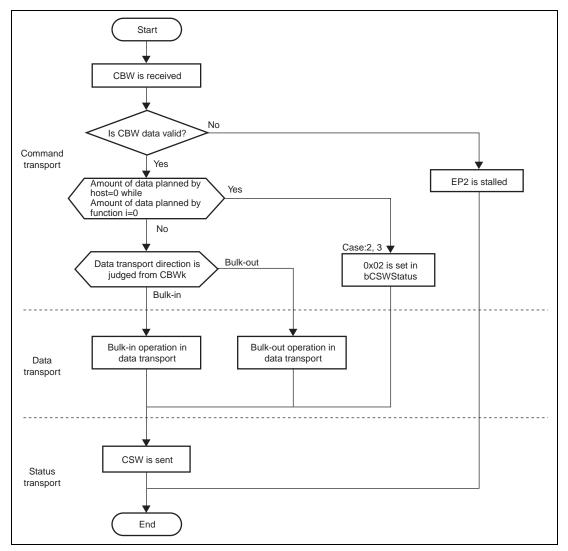


Figure 4.6 Error Processing Flow in Data Transport (1)

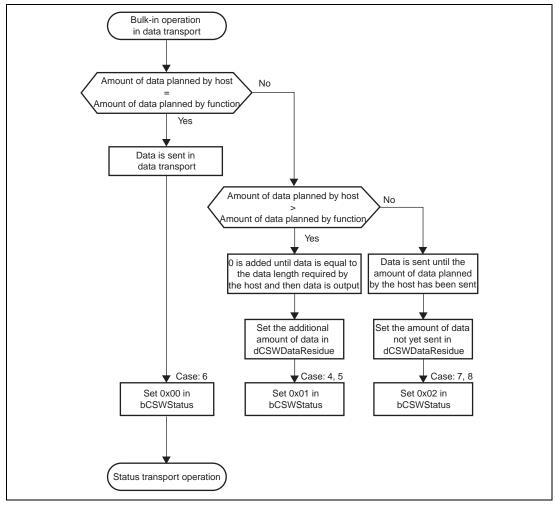


Figure 4.7 Error Processing Flow in Data Transport (2)

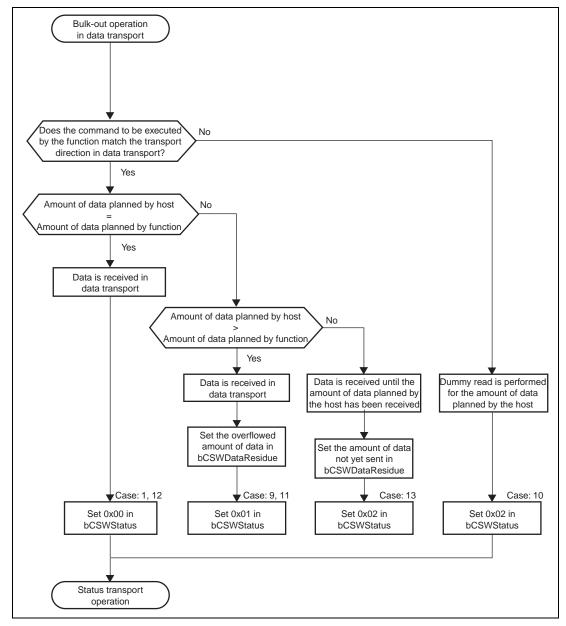


Figure 4.8 Error Processing Flow in Data Transport (3)

When a Mass Storage Class (Bulk-Only Transport) transmission is carried out, transport of the CBW initiates a series of data transfers, and when the CSW is transported to the host PC, a series of data transfers is processed. This status contains two items: dCSWStatus that indicates the transport result, and dCSWDataResidue that indicates the number of error bytes.

In this sample program, the following two fields are used to create these two items.

- dCBWDataTransferLength field of CBW packet
- dCSWDataTransferResidue field of CSW packet

The dCBWDataTransferLength field of the CBW packet is used as the variable in which the number of data bytes the host PC specifies to be handled in the data transport is entered.

The dCSWDataTrasferResidue field of the CSW packet is used as the variable in which the number of data bytes the function handles in the data transport is entered.

When the CBW transport has been completed, the number of data bytes planned to be handled in the data transport by the host PC and the function are stored in the dCBWDataTransferLength and dCSWDataTransferResidue fields, respectively.

Data is transferred in the data transport according to the flowcharts.

If data transport between the host PC and function has been processed without errors, the values in the dCBWDataTransferLength and dCSWDataTransferResidue fields are both subtracted by the number of bytes that have been transferred for every data transfer in the data transport. For other cases, the difference between the number of data bytes the host PC requires to be handled in the data transport and the number of data bytes the function has handled in the data transport is stored in the dCSWDataTransferResidue field of the CSW packet, and operation then moves to the status transport.

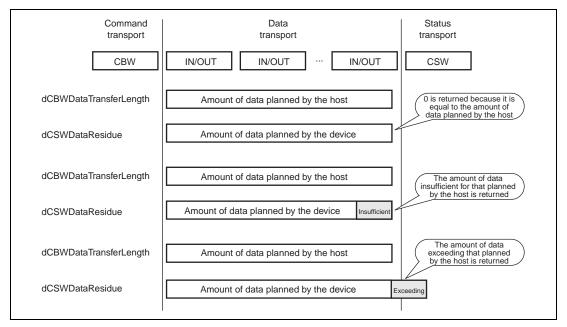


Figure 4.9 Each Stage in Bulk-Only Transport

Section 5 Sample Program Operation

In this chapter, the operation of the sample program is explained, relating it to the operation of the USB function module.

5.1 Main Loop

When the microcomputer is in the reset state, the internal state of the CPU and the registers of internal peripheral modules are initialized. Next, the function SetPowerOnSection in StartUp.c is called, and the CPU is initialized. Figure 5.1 is a flow chart for the SetPowerOnSection function operation.

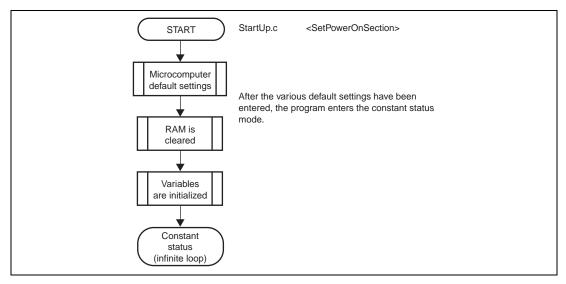


Figure 5.1 Main Loop

5.2 Types of Interrupts

As explained in section 4, the interrupts used in this sample program are indicated by the interrupt flag registers 0 to 3 (UIFR0 to UIFR3); there are a total of nine types of interrupts. When an interrupt factor occurs, the corresponding bits in the interrupt flag registers are set to 1, and an EXIRQ0 interrupt request is sent to the CPU. In the sample program, the interrupt flag registers are read as a result of this interrupt request, and the corresponding USB communication is performed. Figure 5.2 shows the interrupt flag registers and their relation to USB communication.

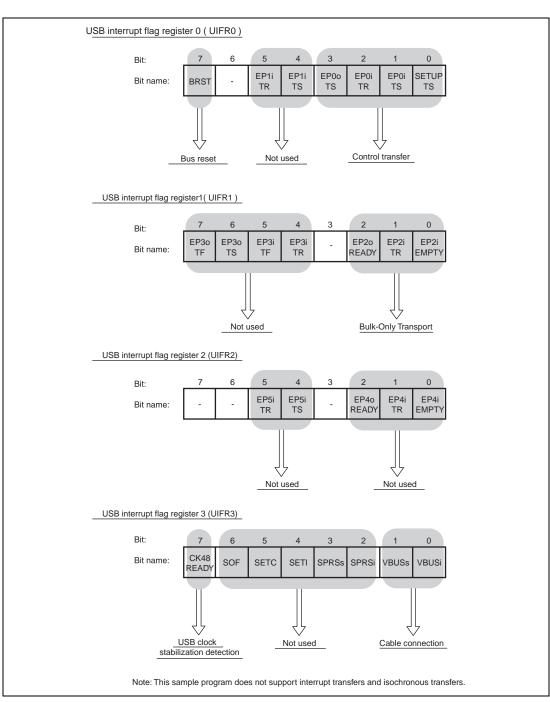


Figure 5.2 Types of Interrupt Flags

5.2.1 Method of Branching to Different Transfer Processes

In this sample program the transfer method is determined by the type of interrupt from the USB module. Branching to the different transfer methods is executed by BranchOfInt in UsbMain.c. Table 5.1 shows the relations between the types of interrupts and the functions called by BranchOfInt.

Table 5.1 Interrupt Types and Functions Called on Branching

Register Name	Bit Bit Name		Name of Function Called		
	7	BRST	ActBusReset		
	6	-	-		
	5	EP1i TR	-		
UIFR0	4	EP1i TS	-		
OIFKO	3	EP0o TS	ActControlIn•ActControlOut		
	2	EP0i TR	ActControlOut		
	1	EP0i TS	ActControlIn•ActControlOut		
	0	SETUP TS	ActControl		
	7	EP3o TF	-		
	6	EP3o TS	-		
	5	EP3i TF	-		
UIFR1	4	EP3i TR	-		
Oli ICI	3	-	-		
	2	EP2o READY	ActBulkOut		
	1	EP2i TR	ActBulkOut		
	0	EP2i EMPTY	ActBulkOut		
	7	CK48 READY	SetUSBModule		
	6	SOF	-		
	5	SETC	-		
UIFR3	4	SETI	-		
OIFKS	3	SPRSs	-		
	2	SPRSi	-		
	1	VBUSs	-		
	0	VBUSi	ActBusVcc		

The EP0iTS and EP0oTS interrupts are used both for control-in and control-out transfer. Hence in order to manage the direction and stage of control transfer, the sample program has three states: TRANS_IN, TRANS_OUT, and WAIT. For details, refer to section 5.4, Control Transfers.

In the H8S/2215 hardware manual, operation of the USB function module when an interrupt occurs, and a summary of operation on the application side are described. From the next section, details of application-side firmware are explained for each USB transfer method.

5.3 USB Operating Clock Stabilization Interrupt

This interrupt occurs when the USB operating clock (48 MHz) stabilization time is automatically counted after USB module stop is canceled. After receiving the interrupt, the sample program writes the endpoint configuration information to the USB endpoint information registers (UEPIR00_0 to 22_4), makes necessary interrupt settings, and waits for USB cable connection.

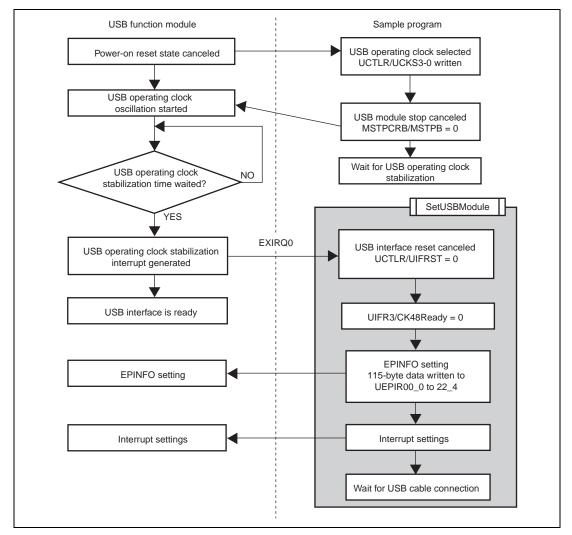


Figure 5.3 USB Operating Clock Stabilization Interrupt

5.3.1 EPINFO

In the USB function module in the H8S/2215, the endpoint configuration can be specified at initialization by software. The following transfer types can be specified:

Control transfer: One endpoint
Bulk-in transfer: Two endpoints
Bulk-out transfer: Two endpoints
Interrupt-in transfer: One endpoint
Isochronous-in transfer: One endpoint
Isochronous-out transfer: One endpoint

The endpoint number, interface number, alternate number, and maximum packet size can be specified for the above transfers (excluding control transfer) with the USB endpoint information registers (UEPIRs).

Table 5.2 shows transfer types and their corresponding UEPIRs.

Table 5.2 Transfer Types and UEPIRs

Transfer Type	Endpoints	Corresponding UEPIRs	
Control transfer	1	00	
Interrupt-in transfer	2	01 and 02	
Bulk-in transfer	2	02 and 20	
Bulk-out transfer	2	03 and 21	
Isochronous-in transfer	1	04, 06, 08, 10, 12, 14, 16, and 18	
Isochronous-out transfer	1	05, 07, 09, 11, 13, 15, 17, and 19	

The H8S/2215 Hardware Manual assumes that endpoint information is configured based on the Bluetooth standard. Figure 5.4 shows the comparison between the endpoint configuration used by this sample program and the endpoint numbers described in the H8S/2215 Hardware Manual.

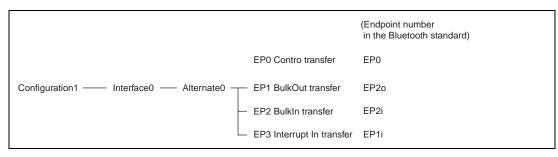


Figure 5.4 Endpoint Configuration in the Sample Program

Table 5.3 shows the UEPIR00_0 to 22_4 settings for the endpoint configuration shown in figure 5.4. Dummy data (0) must be written to the unused endpoints.

Table 5.3 UEPIR Settings

UEPIR	Set Value (Hexadecimal)	Transfer Type	EP No.	Interface No.	Alternate No.	Maximum Packet Size (Byte)
00	00_00_40_00_00	Control	0	0	0	64
01	34_1C_08_00_01	Interrupt In	3	0	0	8
02	24_14_40_00_02	BulkIn	2	0	0	64
03	14_10_40_00_03	BulkOut	1	0	0	64
04	04_1C_00_00_04	Isochronous In	0	0	0	0
05	04_08_00_00_05	Isochronous Out	0	0	0	0
06	04_1C_00_00_06	Isochronous In	0	0	0	0
07	04_08_00_00_07	Isochronous Out	0	0	0	0
08	04_1C_00_00_08	Isochronous In	0	0	0	0
09	04_08_00_00_09	Isochronous Out	0	0	0	0
10	04_1C_00_00_0A	Isochronous In	0	0	0	0
11	04_08_00_00_0B	Isochronous Out	0	0	0	0
12	04_1C_00_00_0C	Isochronous In	0	0	0	0
13	04_08_00_00_0D	Isochronous Out	0	0	0	0
14	04_1C_00_00_0E	Isochronous In	0	0	0	0
15	04_08_00_00_0F	Isochronous Out	0	0	0	0
16	04_1C_00_00_10	Isochronous In	0	0	0	0
17	04_08_00_00_11	Isochronous Out	0	0	0	0
18	04_1C_00_00_12	Isochronous In	0	0	0	0
19	04_08_00_00_13	Isochronous Out	0	0	0	0
20	04_14_00_00_14	BulkIn	0	0	0	0
21	04_10_00_00_15	BulkOut	0	0	0	0
22	04_10_00_00_16	Interrupt In	0	0	0	0

5.4 Interrupt on Cable Connection (VBUS)

This interrupt occurs when the cable of the USB function module is connected to the host controller. On the application side, after completion of initial microcomputer settings, a general-purpose output port is employed to pull-up the USB data bus D+. By means of this pull-up, the host controller recognizes that the device has been connected. (figure 5.5)

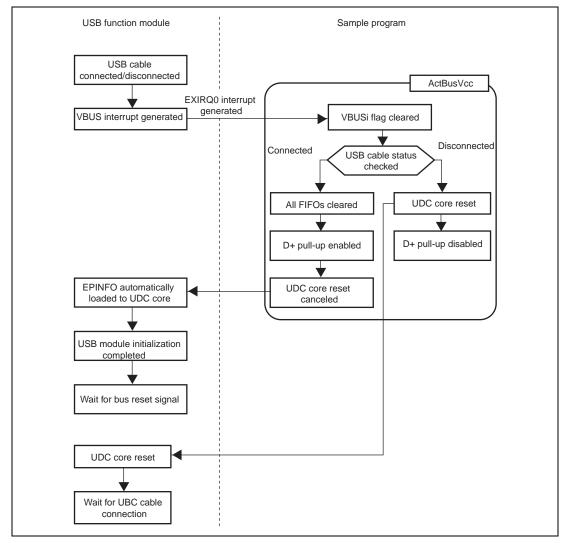


Figure 5.5 Interrupt on Cable Connection

5.5 Bus Reset Interrupt (BRST)

When the host controller detects that a device has been connected to the USB data bus, it outputs a bus reset signal. When receiving this bus reset signal, the USB function module generates a bus reset.

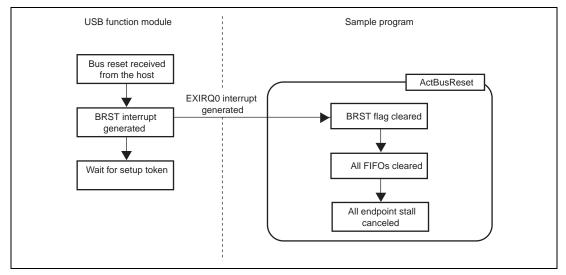


Figure 5.6 Bus Reset Interrupt

5.6 Control Transfers

In control transfers, bits 0 to 3 of the interrupt flag registers are used. Control transfers can be divided into two types according to the direction of data in the data stage. (figure 5.7) In the data stage, data transfers from the host controller to the USB function module are control-out transfers, and transfers in the opposite direction are control-in transfers.

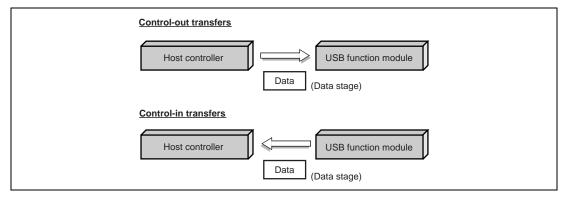


Figure 5.7 Control Transfers

Control transfers consist of three stages: setup, data (no data is possible), and status (figure 5.8). Further, the data stage consists of multiple bus transactions.

In control transfers, stage changes are recognized through the reversal of the data direction. Hence the same interrupt flag is used to call a function to perform control-in or control-out transfers (table 5.1). For this reason, the firmware must use states to manage the type of control transfer currently being performed, whether control-in or control-out, (figure 5.8) and must call the appropriate function. States in the data stage (TRANS_IN and TRANS_OUT) are determined by commands received in the setup stage.

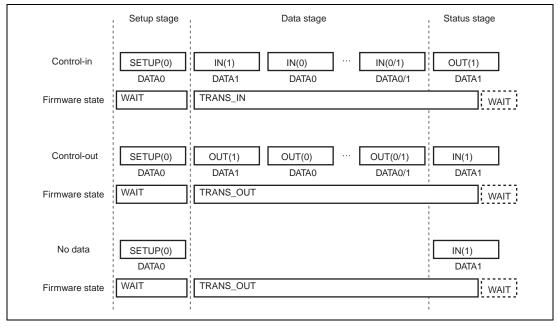


Figure 5.8 Status in Control Transfers

5.6.1 Setup Stage

In the setup stage, the host and function modules exchange commands. For both control-in and control-out transfer, the firmware goes into the WAIT state. Depending on the type of command issued, discrimination between control-in transfer and control-out transfer is performed, and the state of the firmware in the data stage (TRANS_IN or TRANS_OUT) is determined.

• Commands for control-in transfers: GetDescriptor (Standard command)

Get Max LUN (Class command)

• Commands for control-out transfers: Bulk-Only Mass Storage Reset (Class command)

Figure 5.9 shows operation of the sample program in the setup stage. The figure on the left shows operation of the USB function module.

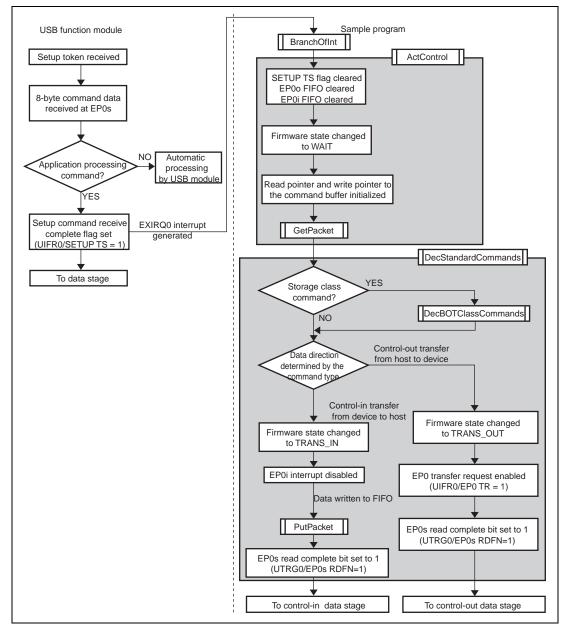


Figure 5.9 Setup Stage

5.6.2 Data Stage

In the data stage, the host and function module exchange data. The firmware state becomes TRANS_IN for control-in transfers, and TRANS_OUT for control-out transfers, according to the result of decoding of the command in the setup stage. Figures 5.10 and 5.11 show the operation of the sample program in the data stage of control transfer.

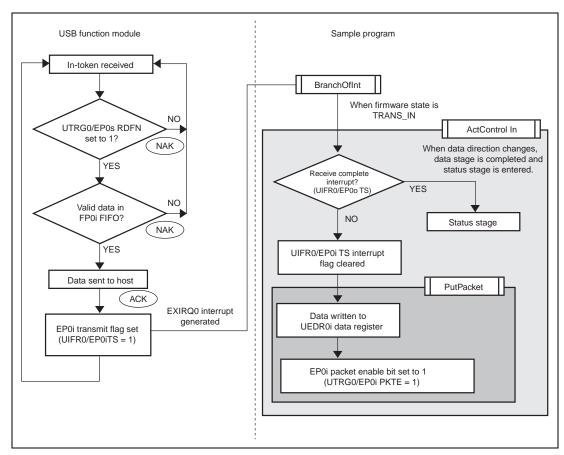


Figure 5.10 Data Stage (Control-In Transfer)

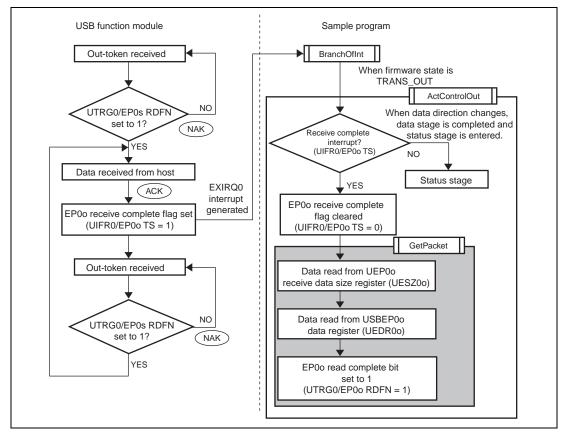


Figure 5.11 Data Stage (Control-Out Transfer)

5.6.3 Status Stage

The status stage begins with a token for the opposite direction from the data stage. That is, in control-in transfer, the status stage begins with an out-token from the host controller; in control-out transfer, it begins with an in-token from the host controller.

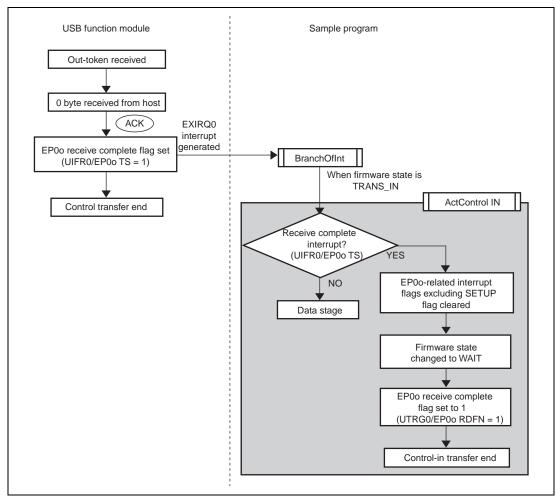


Figure 5.12 Status Stage (Control-In Transfer)

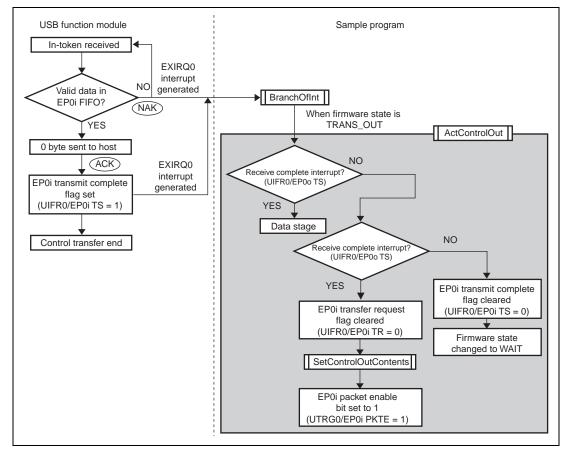


Figure 5.13 Status Stage (Control-Out Transfer)

5.7 Bulk Transfers

In bulk transfers, bits 0 to 2 of interrupt flag register 1 are used. Bulk transfers can also be divided into two types according to the direction of data transmission. (figure 5.14)

When data is transferred from the host controller to the USB function module, the transfer is called a bulk-out transfer; when data is transferred in the opposite direction, it is a bulk-in transfer.

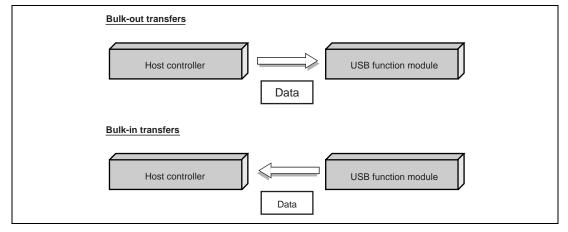


Figure 5.14 Bulk Transfers

The Bulk-Only Transport used in the USB Mass Storage Class consists of bulk-in and bulk-out transfers.

Bulk-Only Transfer comprises two or three stages (see figure 5.15): command transport (CBW), data transport (this is sometimes not included), and status transport (CSW). In addition, data transfer is made up of multiple bus transactions.

With Bulk-Only transport, the command transport (CBW) is done using bulk-out transfer, while the status transport (CSW) is done using bulk-in transfer. Either bulk-in transfer or bulk-out transfer may be used for data transport, depending on the direction in which the data is being sent.

Whether bulk-in or bulk-out transfer is used for data transport is determined by the CBW data received using command transport. In the firmware, whether bulk-in or bulk-out is used for data transport is controlled by states (TRANS_IN and TRANS_OUT) (see figure 5.15). The appropriate functions must be called by the firmware.

Additionally, the transition in stages from data transport to status transport is handled by data of a planned length being sent or received using data transport requested by the host PC. Consequently, the firmware manages the data length sent or received using data transport, and after the transition between stages, status transport must be used to send the data to the host PC.

If the CBW data received using command transport cannot be acknowledged as valid, the endpoint is stalled, and no bulk transfer is carried out.

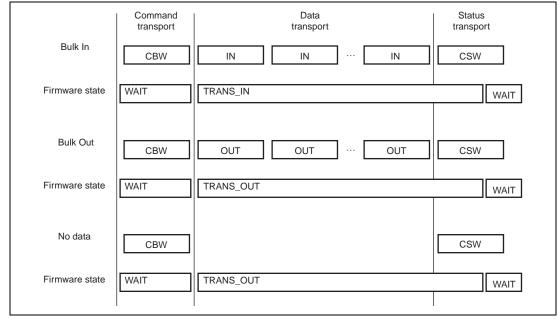


Figure 5.15 Various Stages in Bulk-Only Transport

5.7.1 Command Transport

With command transport, the CBW data is transferred from the host to the function.

At this point, the firmware is in the WAIT state. At the stage following reception of the CBW data, the five types of processing listed below are carried out.

- 1. The CBW data is stored from the EP1 data register to the work area.
- 2. A judgment is made as to whether the CBW data is valid.
- 3. The CSW data is prepared.
- 4. The contents of the CBW data are decoded, and if there is any data to be sent using data transport, the data is prepared. (Processing is carried out in the DecBotCmd function.)
- 5. A distinction is made as to whether the data transport is bulk-in or bulk-out, and the firmware state (TRANS_IN or TRANS_OUT) is determined.

Figure 5.16 shows the operation carried out by the sample program when command transport is used. The operation of the USB function module is shown at the left of the illustration.

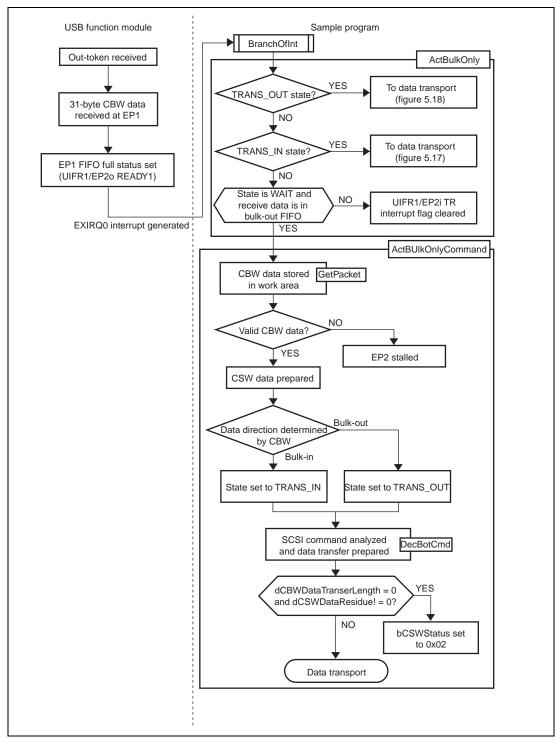


Figure 5.16 Command Transport

5.7.2 Data Transport

With data transport, data is sent and received between the host and the function.

At this point, the firmware is in either the TRANS_IN or TRANS_OUT state.

If the firmware state is TRANS_IN (bulk-in transport), the following three types of processing are carried out.

- 1. Data is sent from the function to the host.
- 2. If the length of the data sent by the function is shorter than the length planned by the host, 0 is added.
- 3. The information to be sent by the CSW is created.

Figure 5.17 shows the operations that take place when data transport (bulk-in transport) is carried out in the sample program. The operation of the USB function module is shown at the left side of the illustration.

In this sample software, if the length of the data sent by the function is shorter than the length of the data requested by the host, 0 is added after the data sent by the function, as noted in the Bulk-Only Transport of the USB Mass Storage Class, and after data of the length requested by the host has been sent, the number of 0 bytes added is reported, using status transport.

In order to carry out this operation, the following is used as global variables: the dCBWDataTransferLength of the CBW data, the dCSWDataResidue of the CSW data, and the bCSWStatus of the CSW data.

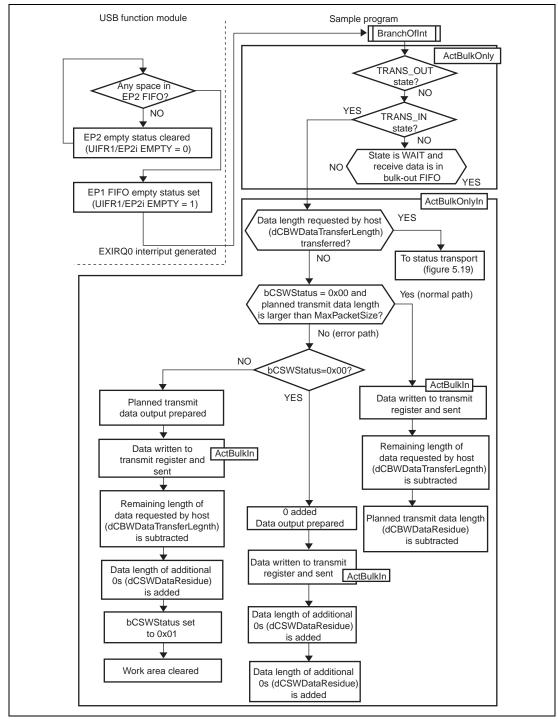


Figure 5.17 Data Transport (Bulk-In Transport)

Figure 5.18 shows the operations that take place when data transport (bulk-out transport) is carried out in the sample program. The operation of the USB function module is shown at the left side of the illustration.

If the firmware state is TRANS_OUT (bulk-out transport), the following three types of processing are carried out.

- 1. Data from the host is received in the function.
- 2. Data length is calculated.
- 3. The information to be sent by the CSW is created.

In this sample software, if the length of the data received by the function is shorter than the length of the data that the host planned to send, the missing length of data received by the function using data transport is reported using status transport, as noted in the Bulk-Only Transport of the USB Mass Storage Class.

In order to carry out this operation, the following is used as global variables: the dCBWDataTransferLength of the CBW data and the dCSWDataResidue of the CSW data.

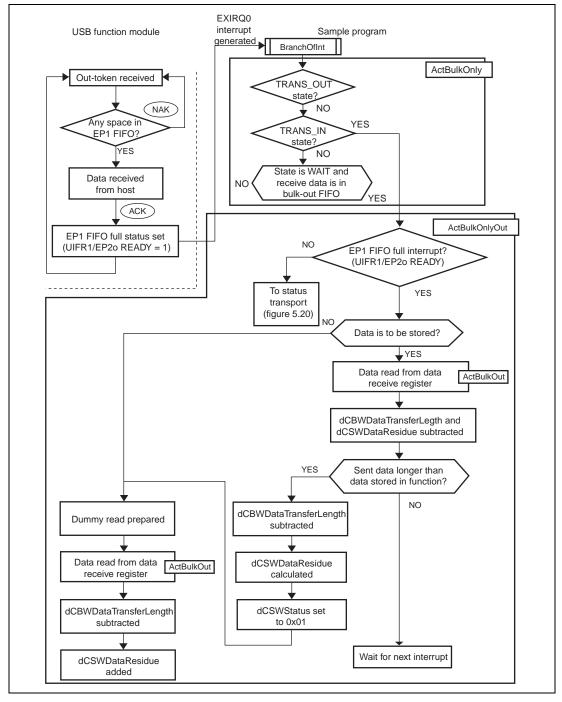


Figure 5.18 Data Transport (Bulk-Out Transport)

5.7.3 Status Transport

With status transport, data is sent from the function to the host.

At this point, the firmware is in either the TRANS_IN or TRANS_OUT state.

If the firmware state is TRANS_IN (bulk-in transport), the following four types of processing are carried out.

- 1. EP2 empty status interrupts are inhibited.
- 2. The system prepares to send the CSW data.
- 3. The CSW data is issued.
- 4. The firmware state is set to WAIT.

Figure 5.19 shows the operations that take place when status transport (data transport bulk-in transport) is carried out in the sample program. The operation of the USB function module is shown at the left side of the illustration.

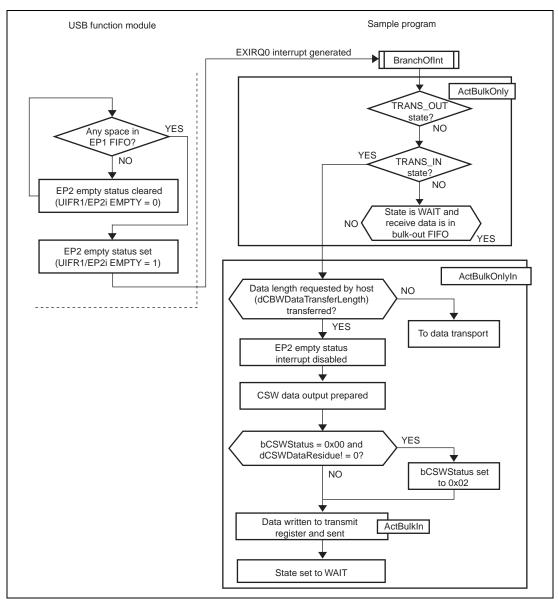


Figure 5.19 Status Transport (Data Transport Bulk-In Transport)

Figure 5.20 shows the operations that take place when status transport (data transport bulk-out transport) is carried out in the sample program. The operation of the USB function module is shown at the left side of the illustration.

If the firmware state is TRANS_OUT (bulk-out transport), the following four types of processing are carried out.

- 1. Preparation is made to send the CSW data.
- 2. The data is checked to see if any data is missing from the reception.
- 3. The CSW data is issued.
- 4. The firmware state is set to WAIT.

In this sample software, if the length of the data received by the function is shorter than the length of the data that the host planned to send, the missing length of data received by the function using data transport is reported using status transport, as noted in the Bulk-Only Transport of the USB Mass Storage Class. In order to do this, a check is made to see if there is any data missing that should have been received by the function, and if there is, the value of the bCSWStatus of the CSW data is set to 0x02 (phase error).

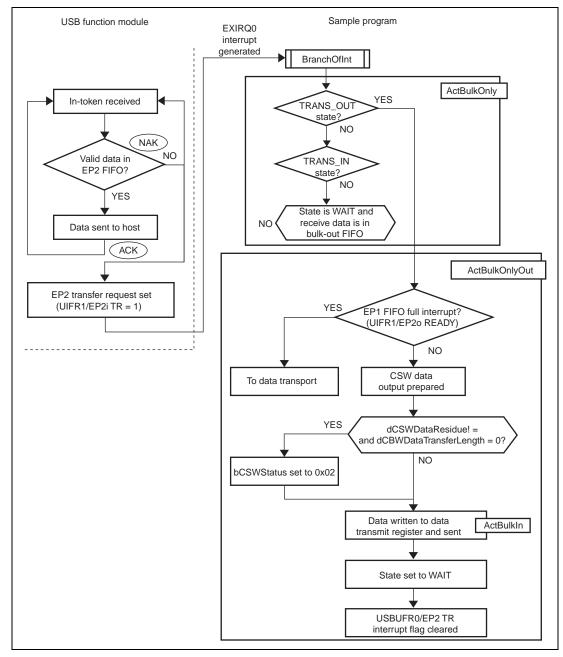


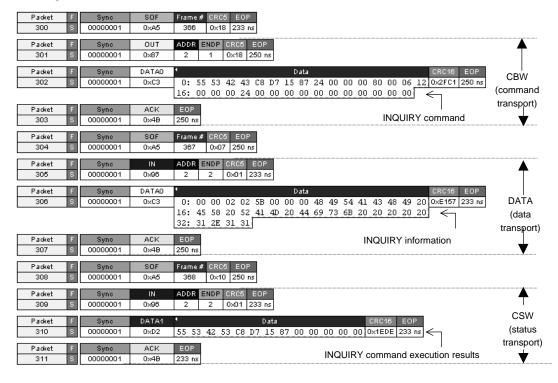
Figure 5.20 Status Transport (Data Transport Bulk-Out Transport)

Section 6 Analyzer Data

In this chapter, we look at how measurement is carried out with the USB Inspector, a USB protocol analyzer made by CATC (http://www.catc.com), using the USB function module in the H8S/2215, and at what happens to the data as it actually flows along the bus. For more detailed information on the packets, please see section 2.3.

Note: The Packet found in front of each packet is the packet number used when measuring.

INQUIRY command



READ FORMAT CAPACITIES command Packet F Sync SOF Frame # CRC6 EOP

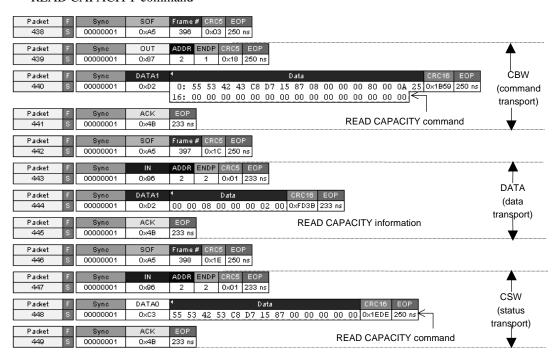
Packet 313	F	Sync 00000001	SOF 0xA5	370 0x0D 250 ns	
	_				
Packet 314	F	Sync 00000001	OUT 0x87	ADDR ENDP CRC5 EOP 2 1 0x18 250 ns	A
314	3	00000001	0,007		
Packet	F	Sync	DATA1	Data CRC16 EOP	CBW
315	S	00000001	0xD2	0: 55 53 42 43 C8 D7 15 87 FC 00 00 00 80 00 0A 23 0xB659 233 ns	(command
				16: 00 00 00 00 00 00 FC 00 00 00 00 00 00 00	transport)
Packet	F	Sync	ACK	EOP	,
316	S	00000001	0x4B	233 ns READ FORMAT CAPACITIES command	▼
Packet	F	Sync	SOF	Frame # CRC5 EOP	
317	s	00000001	0xA5	371 0x12 250 ns	
	=				
Packet	F	Sync	IN	ADDR ENDP CRC5 EOP	A
318	S	00000001	0::96	2 2 0x01 250 ns	
Packet	F	Sync	DATA0	↑ Data CRC16 EOP	
319	s	00000001	0xC3	0: 00 00 00 00 00 00 00 00 00 00 00 00 0	
				16: 00 00 00 00 00 00 00 00 00 00 00 00 00	
				32: 00 00 00 00 00 00 00 00 00 00 00 00 00	
				48: 00 00 00 00 00 00 00 00 00 00 00 00 00	
Packet	F	Sync	ACK	EOP	
320	S	00000001	0x4B	233 ns READ FORMAT CAPACITIES information	on
Packet	F	Sync	IN	ADDR ENDP CRC5 EOP	
321	S	00000001	0::96	2 2 0x01 233 ns	
	_				
Packet 322	F	Sync 00000001	DATA1 0xD2	O: 00 00 00 00 00 00 00 00 00 00 00 00 00	
322	9	00000001	UXD2	16: 00 00 00 00 00 00 00 00 00 00 00 00 00	
				1	
				32: 00 00 00 00 00 00 00 00 00 00 00 00 00	
Davist		C	0.01/	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	
Packet 323	F	Sync	ACK 0v4B	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	
323	s	00000001	0×4B	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	
323 Packet	S	00000001 Sync	0x4B IN	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	DATA
323	s	00000001	0×4B	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	
323 Packet	S F S	00000001 Sync	0x4B IN	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
323 Packet 324	S F S	00000001 Sync 00000001	0x48 IN 0x96	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	
323 Packet 324 Packet	F S	Sync 00000001 Sync 00000001	0x4B IN 0x96 NAK	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
323 Packet 324 Packet 325	S F S	00000001 Sync 00000001	0x48 IN 0x96 NAK 0x5A	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
323 Packet 324 Packet 325 Packet 326	F S F S	Sync 00000001 Sync 00000001 Sync 00000001	0x48 IN 0x96 NAK 0x5A IN 0x96	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
323 Packet 324 Packet 325 Packet 326 Packet	F S F	Sync 00000001 Sync 00000001 Sync 00000001 Sync	0x48 IN 0x96 NAK 0x5A IN 0x96 DATA0	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
323 Packet 324 Packet 325 Packet 326	F S F S	Sync 00000001 Sync 00000001 Sync 00000001	0x48 IN 0x96 NAK 0x5A IN 0x96	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
323 Packet 324 Packet 325 Packet 326 Packet	F S F	Sync 00000001 Sync 00000001 Sync 00000001 Sync	0x48 IN 0x96 NAK 0x5A IN 0x96 DATA0	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
323 Packet 324 Packet 325 Packet 326 Packet	F S F	Sync 00000001 Sync 00000001 Sync 00000001 Sync	0x48 IN 0x96 NAK 0x5A IN 0x96 DATA0	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
Packet 324 Packet 325 Packet 326 Packet 327	F S F S	00000001 Sync 00000001 Sync 00000001 Sync 00000001 Sync 00000001	0x4B IN 0x96 NAK 0x5A IN 0x96 DATA0 0xC3	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
323 Packet 324 Packet 325 Packet 326 Packet	F S F S	Sync 00000001 Sync 00000001 Sync 00000001 Sync	0x48 IN 0x96 NAK 0x5A IN 0x96 DATA0	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
Packet 324 Packet 325 Packet 326 Packet 327 Packet 327	F S	00000001 Sync 00000001 Sync 00000001 Sync 00000001 Sync 00000001 Sync 00000001	0x48 IN 0x96 NAK 0x5A IN 0x96 DATA0 0xC3	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
Packet 324 Packet 325 Packet 326 Packet 327 Packet 327 Packet 327	F S F S	00000001 Syne 0000001 Syne 0000001 Syne 0000001 Syne 0000001 Syne 0000001	0x48 IN 0x96 NAK 0x5A IN 0x96 DATA0 0xC3 ACK 0x48	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
Packet 324 Packet 325 Packet 326 Packet 327 Packet 327	F S	00000001 Sync 00000001 Sync 00000001 Sync 00000001 Sync 00000001 Sync 00000001	0x48 IN 0x96 NAK 0x5A IN 0x96 DATA0 0xC3	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
Packet 329 Packet 326 Packet 327 Packet 327 Packet 327 Packet 328 Packet 329 Packet	F S F S F S F S F S F S F S F F S F S F	00000001 Syne 0000001 Syne 0000001 Syne 0000001 Syne 0000001 Syne 0000001 Syne 0000001 Syne 0000001	0x48 IN 0x96 NAK 0x5A IN 0x96 DATA0 0xC3 ACK 0x48 IN 0x96 NAK	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
Packet 324 Packet 325 Packet 326 Packet 327 Packet 327 Packet 328 Packet 328	F S F S	00000001 Sync 00000001 Sync 00000001 Sync 00000001 Sync 00000001 Sync 00000001	0x48 N 0x96 NAK 0x5A N 0x96 DATA0 0xC3 ACK 0x4B N 0x96	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data
Packet 329 Packet 326 Packet 327 Packet 327 Packet 327 Packet 328 Packet 329 Packet	F S F S F S F S F S F S F S F F S F S F	00000001 Syne 0000001 Syne 0000001 Syne 0000001 Syne 0000001 Syne 0000001 Syne 0000001 Syne 0000001	0x48 IN 0x96 NAK 0x5A IN 0x96 DATA0 0xC3 ACK 0x48 IN 0x96 NAK	32: 00 00 00 00 00 00 00 00 00 00 00 00 00	(data

00000001

Packet

Packet	F	Sync	IN	ADDR ENDP CRC5 EOP	
333	s	00000001	0x96	2 2 0x01 233 ns	
Packet	F	Sync	DATA1	Data CRC16 EOP	
334	S	00000001	0xD2	0: 00 00 00 00 00 00 00 00 00 00 00 00 0	
				16: 00 00 00 00 00 00 00 00 00 00 00 00 00	
	F S	Sync 00000001	ACK 0x4B	EOP 250 ns	V
	F S	Sync 00000001	SOF 0xA5	Frame # CRC5 EOP 372 0x0C 233 ns	
	F S	Sync 00000001	IN 0x96	ADDR ENDP CRC5 EOP 2 0 x01 250 ns	1
Packet	F	Sync	DATAO	Data CRC16 EOP	csw
338	S	00000001	0xC3	8: FC 00 00 00 01	status
	F S	Sync 00000001	ACK 0x4B	EOP	nsport
	F S	Sync 00000001	SOF 0xA5	Frame # CRC5 EOP 373 0x13 233 ns	

• READ CAPACITY command



• REQUEST SENSE command

Packet	F	Sync	SOF	Frame # CRC5 EOP	
340	s	00000001	0xA5	373 0x13 233 ns	
Packet	F	Sync	OUT	ADDR ENDP CRC5 EOP	A
341	S	00000001	0x87	2 1 0x18 250 ns	<u> </u>
Packet	F	Sync	DATAO	↑ Data CRC16 EOP	CBW
342	s	00000001	0xC3	0: 55 53 42 43 C8 D7 15 87 12 00 00 00 80 00 0C 03 0x55555 233 ns	(command
				16: 00 00 00 12 00 00 00 00 00 00 00 00 00 00	transport)
Packet	F	Sync	ACK	EOP	ı
343	S	00000001	0×4B	233 ns REQUEST SENSE command	\perp
040	0	00000001	0,40	200 115	X
Packet	F	Sync	SOF	Frame # CRC5 EOP	
344	S	00000001	0xA5	374 0x11 233 ns	
Packet	F	Sync	IN	ADDR ENDP CRC5 EOP	X
345	S	00000001	0x96	2 2 0x01 250 ns	T
		· · · · · ·			
Destrok		0	DATA	00010 500	DATA
Packet 248	F	Sync 00000004	DATA1	Data CRC16 EOP	DATA
Packet 346	F	Sync 00000001	DATA1 0xD2	0: 70 00 00 00 00 00 00 00 00 00 00 00 00	DATA (data
346		_		Data Chelo Loi	
346 Packet	S	00000001 Sync	0xD2 ACK	0: 70 00 00 00 00 00 00 00 00 00 00 00 00	(data
346	S	00000001	0xD2	0: 70 00 00 00 00 00 00 00 00 00 00 00 00	(data
346 Packet	S	00000001 Sync	0xD2 ACK	0: 70 00 00 00 00 00 00 00 00 00 00 00 00	(data
346 Packet 347	S F S	00000001 Sync 00000001	0xD2 ACK 0x4B	0: 70 00 00 00 00 00 00 00 00 00 00 00 00	(data
Packet 347 Packet 348	F S	Sync 00000001 Sync 00000001	ACK 0x4B SOF 0xA5	0: 70 00 00 00 00 00 00 00 00 00 00 00 00	(data
Packet 347 Packet 348 Packet	F S	Sync 00000001 Sync 00000001 Sync 00000001 Sync 00000001	OxD2 ACK Ox4B SOF OxA5	0: 70 00 00 00 00 00 00 00 00 00 00 00 00	(data
Packet 347 Packet 348	F S	Sync 00000001 Sync 00000001	ACK 0x4B SOF 0xA5	0: 70 00 00 00 00 00 00 00 00 00 00 00 00	(data transport)
Packet 347 Packet 348 Packet	F S F	Sync 00000001 Sync 00000001 Sync 00000001 Sync 00000001	OxD2 ACK Ox4B SOF OxA5	0: 70 00 00 00 00 00 00 00 00 00 00 00 00	(data transport)
946 Packet 347 Packet 348 Packet 349	F S F S	Sync 00000001 Sync 00000001 Sync 00000001	0xD2 ACK 0x4B SOF 0xA5	0: 70 00 00 00 00 00 00 00 00 00 00 00 00	(data transport) CSW (status
346 Packet 347 Packet 348 Packet 349 Packet 350	F S F S	Sync 00000001 Sync 00000001 Sync 00000001 Sync 00000001	0xD2 ACK 0x4B SOF 0xA5 IN 0x96 DATA0 0xC3	0: 70 00 00 00 00 00 00 00 00 00 00 00 00	(data transport)
346 Packet 347 Packet 348 Packet 349 Packet	F S F	Sync 00000001 Sync 00000001 Sync 00000001 Sync	0xD2 ACK 0x4B SOF 0xA5 IN 0x96	0: 70 00 00 00 00 00 00 00 00 00 00 00 00	(data transport) CSW (status

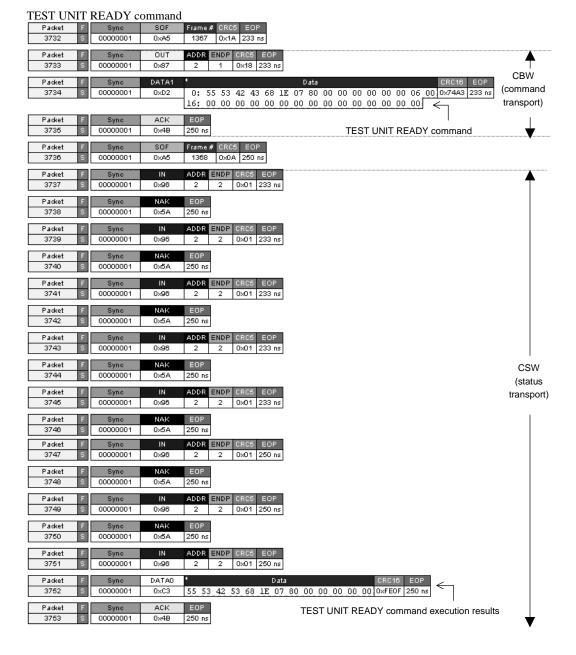
• READ(10) command

Г	Packet	F	Sync	SOF	Frame # CRC5 EOP	
	451	S	00000001	0xA5	400 0x16 233 ns	
Г	Packet	F	Sync	OUT	ADDR ENDP CRCS EOP	
	452	S	00000001	0x87	2 1 0x18 233 ns	Т
Г	Packet	F	Sync	DATAO	↑ Data CRC16 EOP	CBW
	453	S	00000001	0×C3	0: 55 53 42 43 C8 D7 15 87 00 02 00 00 80 00 0A 28 0x983D 233 ns	(command
					16: 00 00 00 00 00 00 00 01 00 00 00 00 00	transport)
	Packet	F	Sync	ACK	READ(10) command	<u> </u>
	454	S	00000001	0x4B	233 ns	V
	Packet	F	Sync	IN	ADDR ENDP CRC5 EOP	A
	455	S	00000001	0::96	2 2 0x01 233 ns	
	Packet	F	Sync	DATA1	Data CRC16 EOP	
	456	S	00000001	0×D2	0: 00 00 00 00 00 00 00 00 00 00 00 00 0	
					16: 00 00 00 00 00 00 00 00 00 00 00 00 00	
					48: 00 00 00 00 00 00 00 00 00 00 00 00 00	
	Packet	5	Sync	ACK	EOP DEAD(10): (#	
H	457	s	00000001	0x4B	250 ns READ(10) information	
_	Packet	F	Sync	IN	ADDR ENDP CRC5 EOP	
\vdash	458	S	00000001	0x96	2 2 0x01 250 ns	
-	Packet	-	Commo	DATAO	Data CRC16 EOP	
-	459	S	Sync 00000001	0xC3	0: 00 00 00 00 00 00 00 00 00 00 00 00 0	
_					16: 00 00 00 00 00 00 00 00 00 00 00 00 00	
					32: 00 00 00 00 00 00 00 00 00 00 00 00 00	
					48: 00 00 00 00 00 00 00 00 00 00 00 00 00	
	Packet	F	Sync	ACK	EOP	
L	460	S	00000001	0x4B	250 ns	DATA
	Packet	F	Sync	IN	ADDR ENDP CRC5 EOP	(data
	461	S	00000001	0x96	2 2 0x01 233 ns	transport)
	Packet	F	Sync	NAK	EOP	1
	462	S	00000001	0x5A	250 ns	

Packet F Sync IN	ADDR ENDP CRC5 EOP
463 S 00000001 0x96	2 2 0x01 233 ns
Packet F Sync DATA1	Data CRC16 EOP
464 S 00000001 0xD2	0: 00 00 00 00 00 00 00 00 00 00 00 00 0
	16: 00 00 00 00 00 00 00 00 00 00 00 00 00
	32: 00 00 00 00 00 00 00 00 00 00 00 00 00
	48: 00 00 00 00 00 00 00 00 00 00 00 00 00
Packet F Sync ACK	EOP .
465 S 00000001 0x4B	250 ns
Packet F Sync SOF	Frame # CRC5 EOP
466 S 00000001 0xA5	401 0x09 233 ns
Packet F Sync IN	ADDR ENDP CRC5 EOP
487 S 00000001 0x98	2 2 0x01 233 ns
Packet F Sync DATA0 468 S 00000001 0xC3	Data CRC16 EOP
468 S 00000001 0xC3	0: 00 00 00 00 00 00 00 00 00 00 00 00 0
	32: 00 00 00 00 00 00 00 00 00 00 00 00 00
	48: 00 00 00 00 00 00 00 00 00 00 00 00 00
Packet F Sync ACK	EOP (data
469 S 00000001 0x4B	250 ns transport)
Packet F Sync IN 470 S 00000001 0x96	ADDR ENDP CRC5 EOP 2 2 0x01 250 ns
470 8 00000001 0x96	2 2 0x01 250 ns
Packet F Sync DATA1	Data CRC16 EOP
471 S 00000001 0xD2	0: 00 00 00 00 00 00 00 00 00 00 00 00 0
	16: 00 00 00 00 00 00 00 00 00 00 00 00 00
	48: 00 00 00 00 00 00 00 00 00 00 00 00 00
Packet F Sync ACK 472 S 00000001 0x4B	EOP
472 S 00000001 0×4B	233 ns
Packet F Sync IN	ADDR ENDP CRC5 EOP
473 S 00000001 0x96	2 2 0x01 233 ns
Packet F Sync NAK	EOP
474 S 00000001 0x5A	

Packet F	- 7111	IN	ADDR ENDP CRC5 EOP	
475	0000000	0x96	2 2 0x01 233 ns	
Packet F	Sync	DATAO	Data CRC16 EOP	
476 S	0000000	0×C3	0: 00 00 00 00 00 00 00 00 00 00 00 00 0	5
			16: 00 00 00 00 00 00 00 00 00 00 00 00 00	
			32: 00 00 00 00 00 00 00 00 00 00 00 00 00	
			48: 00 00 00 00 00 00 00 00 00 00 00 00 00	
Packet F	Sync	ACK	EOP	
477 8	0000000	0x4B	250 ns	
Packet [Sync	IN	ADDR ENDP CRC5 EOP	
478	- 7111	_	2 2 0x01 250 ns	
Packet	-7	NAK	EOP	
479 S	0000000	0x5A	250 ns	
Packet F	Sync	IN	ADDR ENDP CRC5 EOP	DATA
480	00000000	0x96	2 2 0x01 233 ns	(data
Packet [Sync	NAK	EOP	ransport)
481 S	- /		250 ns	
Packet i	- 7	IN	ADDR ENDP CRC5 EOP	
482	00000000	0x96	2 2 0x01 233 ns	
Packet F	Sync	DATA1	Data CRC16 EOP	
483	00000000	0×D2	0: 00 00 00 00 00 00 00 00 00 00 00 00	3
			16: 00 00 00 00 00 00 00 00 00 00 00 00 00	
			32: 00 00 00 00 00 00 00 00 00 00 00 00 00	
			48: 00 00 00 00 00 00 00 00 00 00 00 00 00	
Packet F	Sync	ACK	EOP	
484 S	00000000	0x4B	250 ns	
Packet [Sync	IN	ADDR ENDP CRC5 EOP	
485 9			2 2 0x01 250 ns	
Packet F		NAK	EOP	
486	0000000	0x5A	250 ns	
Packet E		IN	ADDR ENDP CRC5 EOP	
487 8	00000000	0x96	2 2 0x01 250 ns	

					_																1
Packet	F S	ync	NAK	EOP																	
488	0000	00001	0x5A	250 r	s																
Packet	F S	yne	IN	ADDE	END	P CR	05 F	EOP													
	_	00001	0x96	2	2	0x0		50 ns													
100			0,00			10//	/ <u>- \</u>	30 113													_
Packet	F S	ync	DATA0	1						Data									CRC16	EOP	
490	0000	00001	0xC3	0:	00 0	0 01	00	00 00	20	00	00	00	00	08	00	00	00	00	0×D54E	3 250 n	5
				16:	00 0	0 00	00	00 00	00	00	00	00	00	00	00	00	00	00			
				32:	00 0	0 00	00	00 00	00	00	00	00	00	00	00	00	00	00			
				48:	00 0	0 00	00	00 00	00	00	00	00	00	00	00	00	55	AA			
Packet	F S	ync	ACK	EOP																	
	_	00001	0x4B	233 г																	\blacksquare
-101	000	50001	0,40	1200 .																	X
Packet	F S	ync	SOF	Fram	e#CF	RC5 I	EOP														
492	0000	00001	0xA5	402	0>	0B 2	33 ns]													
Packet	F Q	vne	IN	ADDE	END	P CR	^5 F	EOP													\blacktriangle
	_	00001	0x96	2	2	0×0		33 ns													Τ
180	0000	50001	0,50		1 -	10/	/ 1 Z \	30 113												(CSW
Packet	F S	ync	DATA1	1				Data							ORC1	16	EOF	•	<u> </u>	(status
494	0000	00001	0xD2	55_5	3 42	53_	C8 D	7 15	87	00_0	00_0	0 0	0_0	0 0	×1EI	DE 2	233 ı	ns	`	,	
Packet	- 0	va a	ACK	EOP							חר	۸۵/	40)					 :	l 		nsport)
	_	ync 00001	0x4B	233 r							KE	AD(10)	con	nina	arid	exe	cuti	on res	uits	\downarrow
495	0000	00001	0%48	233 F	S																X
Packet	F S	ync	SOF	Fram	e#CF	RC5 I	EOP	1													
496	S 0000	00001	0xA5	403	0:	(14 2	33 ns	1													



• VERIFY(10) command

•	V LIN	11. 1	(10) con	mana								
	Packet 6631	F	Sync 00000001	SOF 0xA5	Frame #	0x11 233 ns						
		=										
	Packet 6632	F	Sync 00000001	0UT 0x87	ADDR 2	ENDP CRC5 EOP 1 0x18 233 ns						•
		_				1 OX 10 200 115						CBW
	Packet 6633	F	Sync 00000001	DATA0 0xC3	0.5	5 53 42 43 48 14	Data . 07 80 00	00 00	00 00 0	O OA 2E	CRC16 EOP 0x89DC 250 ns	(command
	0000	_	00000001	0,000		0 00 00 00 20 00						transport)
F	acket	F	Sync	ACK	EOP						`	
	6634	S	00000001	0×4B	250 ns				\	/ERIFY(1	0) command	\forall
F	acket	F	Sync	SOF	Frame	CRC5 EOP						
	6635	s	00000001	0xA5	1874	0x13 233 ns						
F	acket	F	Sync	IN	ADDR	ENDP CRC5 EOP						.
	6636	s	00000001	0x96	2	2 0x01 250 ns						T
F	acket	F	Sync	NAK	EOP							
	6637	s	000001	0x5A	250 ns							
F	Packet	F	Sync	IN	ADDR	ENDP CRC5 EOP						
	6638	s	000001	0x96	2	2 0x01 250 ns						
F	acket	F	Sync	NAK	EOP							
	6639	s	000001	0x5A	250 ns							
F	acket	F	Sync	IN	ADDR	ENDP CRC5 EOP						
	6640	S	00000001	0x96	2	2 0x01 250 ns						
F	acket	F	Sync	NAK	EOP							
	6641	S	00000001	0x5A	250 ns							
F	acket	F	Sync	IN	ADDR	ENDP CRC5 EOP						
	6642	S	00000001	0x96	2	2 0x01 250 ns						CSW
F	acket	F	Sync	NAK	EOP							(status
	6643	S	000001	0x5A	250 ns							transport)
	acket	F	Sync	IN	ADDR							папорот
	6644	S	00000001	0x96	2	2 0x01 250 ns						1
	acket	F	Sync	NAK	EOP							
	6645	S	000001	0x5A	250 ns							
	Packet 8646	F	Sync 00000001	IN 0x96	ADDR 2	ENDP CRC5 EOP 2 0x01 250 ns						
		_				2 0x01 250 lis						
	Packet 6647	F	Sync 00000001	NAK 0x5A	EOP 250 ns							
		=										
	Packet 6648	F	Sync 00000001	IN 0x96	ADDR 2	ENDP CRC5 EOP 2 0x01 233 ns						
		=			1				050	FOR		
	Packet 6649	F	Sync 00000001	DATA1 0xD2	55 53	Data 42 53 48 14 07	80 00 00	00 00 0	0 0x6672		\leftarrow	
		_				22 00 40 14 07	00 00	23 00 0	=1	1-22 12	`	
	Packet 6650	F	Sync 00000001	ACK 0x4B	233 ns					VERIFY	10) command	\perp
											-,	▼

WRITE(10) command Sync Frame # CRC5 EOP 2650 00000001 0xA5622 0x06 | 233 ns ADDR ENDP CRC5 EOP Packet Sync OUT 2651 00000001 0x87 0x18 233 ns CBW DATA1 Packet Data CRC16 EOP Sync 2652 00000001 O: 55 53 42 43 88 C5 97 86 00 02 00 00 00 00 0A 2A 0x8E8D 250 ns (command $0 \times D2$ transport) Packet Sync ACK WRITE(10) command 00000001 0×4B 250 ns 2653 Frame # CRC5 EOP Packet SOF Sync 2654 00000001 0x19 233 ns $0 \times A5$ 623 Packet OUT ADDR ENDP CRC5 EOP Sync 2655 00000001 0x87 0x18 | 233 ns DATAO Packet Sync Data 00000001 2656 0xC3 Sync Packet ACK WRITE(10) information 2657 00000001 0×4B 250 ns Packet ADDR ENDP CRC5 EOP OUT Sync 0×18 250 ns 2658 000001 0x87 1 DATA1 Packet CRC16 EOP Sync Data 2659 00000001 0xD2 Packet ACK EOP Sync 2660 00000001 0×4B 250 ns OUT ADDR ENDP CRC5 EOP Packet Sync 2661 0x18 233 ns 00000001 0x871 Packet Sync DATA0 DATA 2662 00000001 0xC3 (data transport) Packet ACK EOP Sync 2663 00000001 0x4B 250 ns Packet Sync OUT ADDR ENDP CRC5 EOP 2664 00000001 0x87 1 0x18 | 250 ns Packet Sync DATA1 Data 2665 00000001 0xD2 Packet NAK Sync 2666 250 ns 00000001 0x5A Packet Sync OUT ADDR ENDP CRC5 EOP 2667 00000001 0x87 1 0x18 233 ns Packet DATA1 Sync CRC16 EOP 2668 00000001 0xD2 Packet ACK 2669 00000001 0x4B 250 ns

Packet F	Sync 00000001	OUT 0x87	ADDR ENDP CRC5 EOP 2 1 0x18 250 ns
Packet F 2671 S	Sync 00000001	DATA0 0xC3	0: 00 00 00 00 00 00 00 00 00 00 00 00 0
Packet F	Sync	ACK	32: 00 00 00 00 00 00 00 00 00 00 00 00 00
2672 S	00000001 Sync	0x4B OUT	ADDR ENDP CRC5 EOP
2673 S Packet F 2674 S	00000001 Sync 00000001	0x87 DATA1 0xD2	2
			16: 00 00 00 00 00 00 00 00 00 00 00 00 00
Packet F	Sync	ACK	EOP
2675 S	00000001	0x4B	250 ns
Packet F	Sync	OUT	ADDR ENDP CRC5 EOP
2676 S	00000001	0x87	
Packet F	Sync	DATA0	
2677 8	00000001	0xC3	0: 00 00 00 00 00 00 00 00 00 00 00 00 0
Packet F	Sync	NAK	EOP 250 ns
2678 S	00000001	0x5A	
Packet F	Sync	0UT	ADDR ENDP CRC5 EOP 2 1 0x18 233 ns
2679 S	00000001	0x87	
Packet F	Sync	DATAD	0: 00 00 00 00 00 00 00 00 00 00 00 00 0
2680 S	00000001	0xC3	
Packet F	Sync	ACK	EOP
2681 S	00000001	0x4B	250 ns
Packet F	Sync	0UT	ADDR ENDP CRC5 EOP 2 1 0x18 250 ns
2682 S	00000001	0x87	
Packet F	Sync	DATA1	Data CRC16 EOP
2683 S	00000001	0xD2	
Packet F	Sync	ACK	EOP 250 ns
2684 S	00000001	0x4B	
Packet F 2685 S	00000001	SOF 0xA5	Frame # CRC5 EOP 624 0x0E 233 ns
Packet F 2686 S	Sync 00000001	IN 0×96 NAK	ADDR ENDP CRC5 EOP 2 2 0x01 233 ns
Packet F 2687 S	Sync 00000001 Sync	0x5A IN	250 ns ADDR ENDP CRC5 EOP
2688 S Packet F 2689 S	00000001	0x96	2 2 0x01 233 ns
	Sync	NAK	EOP
	00000001	0x5A	250 ns

Packet	F	Sync	IN		ENDP	_									
2690	S	00000001	0x96	2	2	0x01	233 ns								
Packet	F	Sync	NAK	EOP	1										
2691	S	00000001	0x5A	250 ns											
D - d - d		0		4000	ENDO	ODOF	FOR								
Packet 2692	F	Sync 00000001	IN 0×96	ADDR 2	ENDP 2	0x01	EOP 233 ns								
2082	9	00000001	0,000	1 -		UXU I	233 118								
Packet	F	Sync	NAK	EOP											
2693	S	00000001	0x5A	250 ns											
Packet	F	Sync	IN	ADDR	ENDP	CRC5	EOP								
2694	s	00000001	0x96	2	2	0x01	250 ns								
	Ξ														
Packet	F	Sync	NAK	EOP											
2695	S	00000001	0x5A	250 ns											
Packet	F	Sync	IN	ADDR	ENDP	CRC5	EOP								CSW
2696	S	00000001	0x96	2	2	0x01	250 ns								
Packet	F	Sync	NAK	EOP	1										(status
2697	S	00000001	0x5A	250 ns	1										transport
2007	9	00000001	0,004	200 113	1										
Packet	F	Sync	IN	ADDR		CRC5	EOP								
2698	S	00000001	0x96	2	2	0x01	250 ns								
Packet	F	Sync	NAK	EOP	1										
2699	S	00000001	0x5A	250 ns											
	=	_													
Packet	F	Sync	IN			CRC5									
2700	S	00000001	0x96	2	2	0x01	250 ns								
Packet	F	Sync	DATA1	1			Dat	3			CRC16	EOP			
2701	S	00000001	0xD2	55 50	3 42	53 88	C5 97	86 00	00 0	00 00	0x5620	250 ns	$] \leftarrow$		
Packet	F	Sync	ACK	EOP	1										
2702	s	00000001	0x4B	250 ns	1				WR	ITE(10)	comma	and exe	ecution	results	\downarrow
2702	~			1200 113	1					, -,					•

H8S/2215, HD64F2215-USB Function Module Mass Storage Class (Bulk-Only Transport)

Publication Date: 1st Edition, April 2002

Published by: Customer Operation Division

Semiconductor & Integrated Circuits

Hitachi, Ltd.

Edited by: Technical Documentation Group

Hitachi Kodaira Semiconductor Co., Ltd.

Copyright ${\hbox{\o C}}$ Hitachi, Ltd., 2002. All rights reserved. Printed in Japan.