To our customers,

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; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Application Note

32

SH7727 Group USB Host Module

Application Note (Advanced)

Renesas 32 bit RISC Microcomputer
SuperH™ RISC engine Family/
SH7700 Series

Renesas Electronics
www.renesas.com

Rev.1.00 2003.4

Cautions

Keep safety first in your circuit designs!

1. 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

1. 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.
2. 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.

Preface

This application note describes the firmware that uses the USB function module and USB host module in the SH7727. This is provided to be used as a reference when the user creates the USB function controller firmware and USB host controller firmware.

This application note and the described software are application examples of the USB function module and USB host module, and their contents and operation are not guaranteed.

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

[Related manuals]

- Universal Serial Bus Specification Revision 1.1
- Open Host Controller Interface Specification for USB Preliminary Release Revision 1.0a
- Open Universal Serial Bus Driver Interface (Open USBDI) Specification Revision 1.0
- Universal Serial Bus Mass Storage Class Specification Overview Revision 1.1
- Universal Serial Bus Mass Storage Class (Bulk-Only Transport) Revision 1.0
- Device Class Definition for Human Interface Devices (HID) Version 1.1
- SH7727 Hardware Manual
- SH7727 Solution Engine (MS7727SE01) Instruction Manual
- SH7727 E10A Emulator User's Manual
- SH7727 USB Host Controller Application Note

[Caution] The sample programs described in this application note do not include firmware related to isochronous transfer of USB transfer types. When using the transfer type, the user needs to create the program for it.

Also, the hardware specifications of the SH7727 and SH7727 Solution Engine, which will be necessary when developing the system described above, are described in this application note, but more detailed information is available in the SH7727 Hardware Manual and the SH7727 Solution Engine Instruction Manual.

Contents

Section 1 Overview

This application note describes how to use the USB function module (USBF) and USB host module (USBH) that are incorporated in the SH7727, and contains examples of firmware programs.

The features of the USB function module incorporated in the SH7727 are listed below.

- Includes UDC (USB Device Controller) conforming to USB 1.1
- Automatic processing of USB protocol
- Automatic processing of USB standard commands for endpoint 0 (some commands need to be processed by the firmware)
- Full-speed (12 Mbps) transfer supported
- Various interrupt signals for USB transmission and reception are generated.
- Internal system clock generated by the EXCPG or external input (48 MHz) can be selected.
- Power-down mode supported
- Includes bus transceiver

Endpoint Configurations

Endpoint Name	Abbr.	Transfer Type	Max. Packet Size	FIFO Buffer Capacity (bytes)	DMA Transfer
Endpoint 0	EP0s	Setup	8 bytes	8 bytes	—
	EP0i	Control-In	8 bytes	8 bytes	—
	EP0o	Control-Out	8 bytes	8 bytes	—
Endpoint 1	EP1	Bulk-OUT	64 bytes	64 x 2 (128 bytes)	Possible
Endpoint 2	EP2	Bulk-IN	64 bytes	64 x 2 (128 bytes)	Possible
Endpoint 3	EP3	Interrupt	8 bytes	8 bytes	—

The features of the USB host module incorporated in the SH7727 are listed below.

- Conforming to USB 1.1
- Conforming to Open Host Controller Interface (OHCI) 1.0 register set
- Root hub function
- Low-speed (1.5 Mbps) and full-speed (12 Mbps) transfer supported
- Detection of overcurrent supported
- Maximum 127 endpoints
- User memory area connected to the SH7727 can be used for transfer data and descriptor storage

Figure 1.1 shows an example of a system configuration.

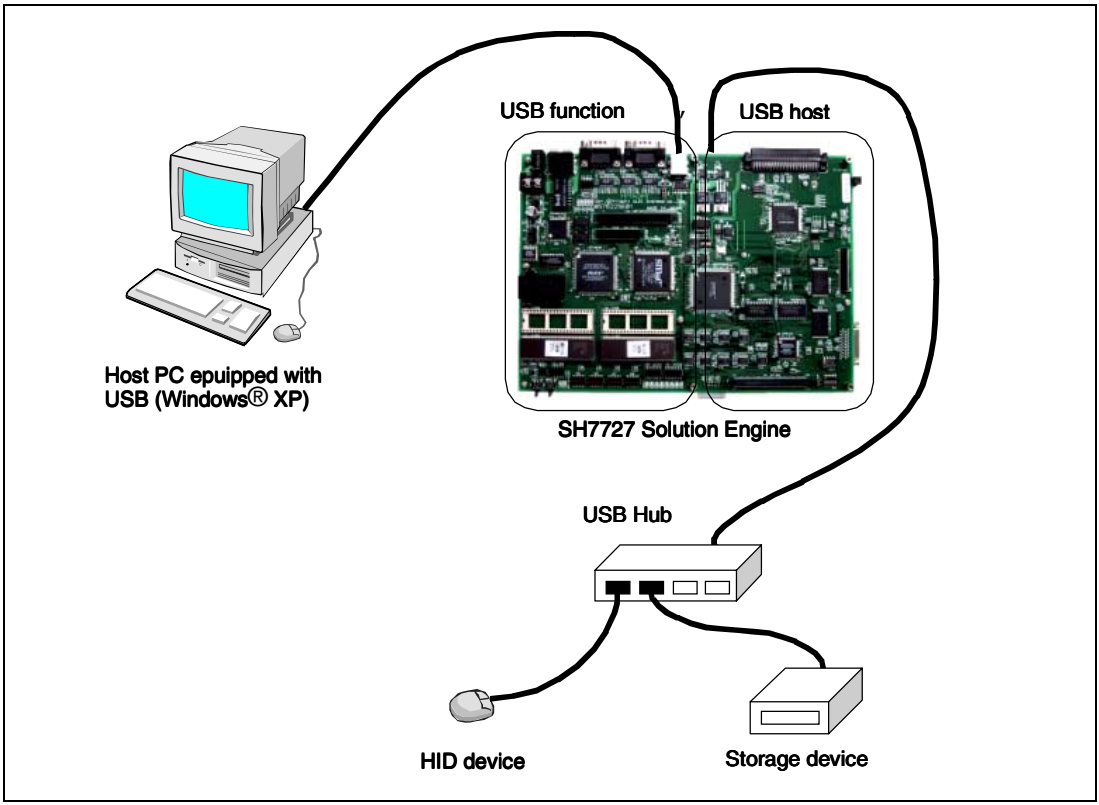

Figure 1.1 System Configuration Example

This system is configured with the SH7727 Solution Engine made by Hitachi ULSI Systems Co., Ltd. (hereafter referred to as the SH7727SE), a PC running on Windows® XP (hereafter referred to as the host PC), USB hub, and USB function devices (USB Mass Storage Class and HID class).

In this application note, data transfer between the host PC and USB function devices through USB is described using the system in which the host PC and USB function devices are connected via the SH7727SE and USB hub.

The SH7727 in the SH7727SE includes the USB host module and controls the USB hub and function devices as the USB host (hereafter referred to as the SH7727 host).

The SH7727 also includes the USB function module, therefore, the SH7727SE operates as the function device to the host PC (here after referred to as the SH 7727 function).

A USB function device has descriptor information which indicates the type, characteristics, attribute of a device. The SH7727SE generates new descriptor information of a device with

multiple interfaces (multifunction device) based on descriptor information of the USB function device connected to the SH7727 host.

Then, by connecting to the host PC using descriptor information which is generated by the SH7727 host, the SH7727 function transfers data between the host PC as a multifunction device.

At this time, the SH7727SE serves as a bridge between the host PC and USB function devices and enables to store and load data from the host PC to USB function devices.

For operation of this application, the USB host, USB function, and timer modules incorporated in the SH7727 are used. It is also possible to use the device driver that comes as an accessory with the operating system.

This system offers the following features.

1. The sample program can be used to evaluate the USB host and USB function modules of the SH7727 quickly.
2. The debugging support function is provided in the sample program. Settings in the source program enables to output error detected by the host module via serial interface.
3. The sample program supports control, bulk, and interrupt transfers.
4. The E10A (PC card-type emulator) can be used, enabling efficient debugging.

Section 2 Outline of USB Hub

This section describes the USB hub.

This is provided to be used as a reference when the user creates the USB host related system. For details of the specification, refer to the Universal Serial Bus Specification revision 1.1.

2.1 Role of Hub

A USB hub is a device that is located between the host and a function device to increase the number of nodes and provide the characteristic functions; strict rules of the data transfer method, power supply from the USB ports, the plug-and-play architecture, and up to 127 function devices connectable. A USB hub also has the following functions.

- Signal transmission/disruption
- Realizing the plug and play architecture
- Power management

To perform these functions correctly, the host and hubs are required to communicate in decided method and the method is defined as the Hub Class.

When the USB host confirms that the bDeviceClass field in the device descriptor and the bInterfaceClass field in the interface descriptor are set to 0x09, it realizes that the connected devices are the Hub Class.

2.2 Hub Descriptor

USB devices have the descriptor information which indicates the type, characteristics, and attribution of the device.

The standard USB device has the descriptors of device, configuration, interface, and endpoint. In addition to these descriptors, the hub device has the Hub class descriptor information (Hub Descriptor). The hub descriptor is obtained by using the class command. The table 2.1 shows the hub descriptor.

Table 2.1 Hub Descriptor Format

Field	Size (byte)	Descriptions
bDescLength	0x01	Descriptor size
bDescriptorType	0x01	Hub descriptor value: 0x29
bNbrPorts	0x01	The number of ports supported by this hub
wHubCharacteristics	0x02	Hub characteristics Bits 1 and 0: Port power control mode 00: Controls all ports 01: Controls ports individually 1X: Reserved Bit 2: Identifies compound 0: Hub is not a part of compound device 1: Hub is a part of compound device Bits 4 and 3: Over-current Protection Mode 00: Over-current Protection by all ports 01: Over-current Protection by individual port 1X: No Over-current Protection Bits 15 to 5: Reserved
bPwrOn2PwrGood	0x01	Wait time between the power supply to a port and start of access. In 2 ms intervals.
bHubContrCurrent	0x01	Maximum current requirements of a Hub. Unit is mA.
DeviceRemovable	0x01	Indicates whether a port has a removable function module. Bit 0: Reserved Bits 1 to n: Ports 1 to n (up to 255) 0: Removable 1: Not removable
PortPwrCtrlMask	The number of ports	This field is used for a hub supporting the USB 1.0 but not used for a hub supporting the USB 1.1. This field for the USB 1.1 exists only for compatibility.

2.3 Hub Class-Specific Requests

A class command is a command which is defined for each USB class definition. A class command uses control transfers. There are nine types of requests that can be sent to a USB hub. Table 2.2 lists the requests.

Table 2.2 List of Commands

Requests	Command Descriptions	Value of BmRequestType Field	Value of BRequest Field
ClearHubFeature	Reset a value reported by the hub status	0x20	0x01
ClearPortFeature	Reset a value reported by the port status	0x23	0x01
GetBusState	Return the bus state	0xA3	0x02
GetHubDescriptor	Obtain a hub descriptor	0xA0	0x06
GetHubStatus	Return the current hub state and changed hub state from previous recognition	0xA0	0x00
GetPortStatus	Return the current port state and changed port state from previous recognition	0xA3	0x00
SetHubDescriptor	Overwrite a hub descriptor	0x20	0x07
SetHubFeature	Recognize the changed state from the previous hub state	0x20	0x03
SetPortFeature	Recognize the changed state from the previous port state	0x23	0x03

2.4 Host-Hub-Function Connection State

A hub connects the upstream host and downstream devices by using including ports.

A hub has a port for upstream and two or more ports for downstream, and controls the ports to gather and report hub state information to the host. Figure 2.1 shows the connection between the host, hub, and function devices. Necessary things to control a hub (ports) is described below.

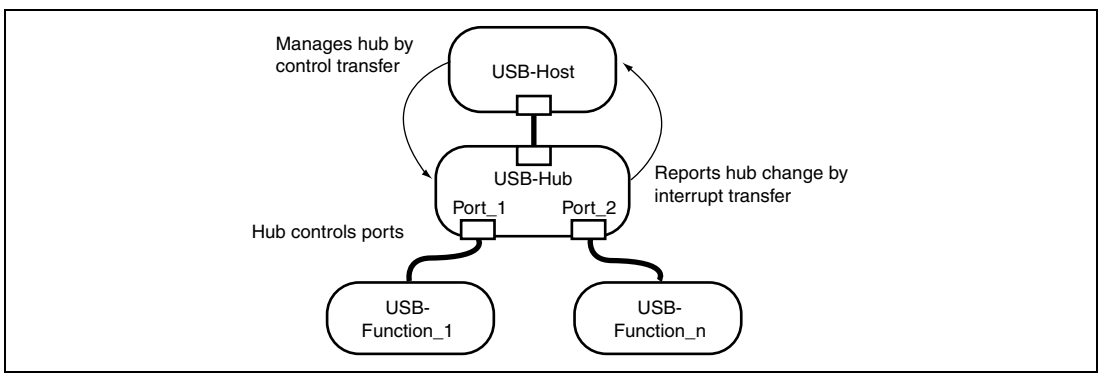

Figure 2.1 Host—Hub—Function Connection Diagram

2.5 Port State Transition

Ports included in a hub have seven states: NotConfigured, Powered-off, Disconnected, Disabled, Resetting, Enabled, and Suspend. These states changes are caused by a setting command from the host, defined time, and the operation of connected device.

To set a port state, first confirm the port state, next, set port state. Finally, after recognizing the port change, the port state is changed.

Figure 2.2 shows the state transition.

To confirm the port state, GetPortStatus is used.

For the transition caused by the host setting, use a class command SetPortFeature.

For the recognition of the port transition, the ClearPortFeature is used.

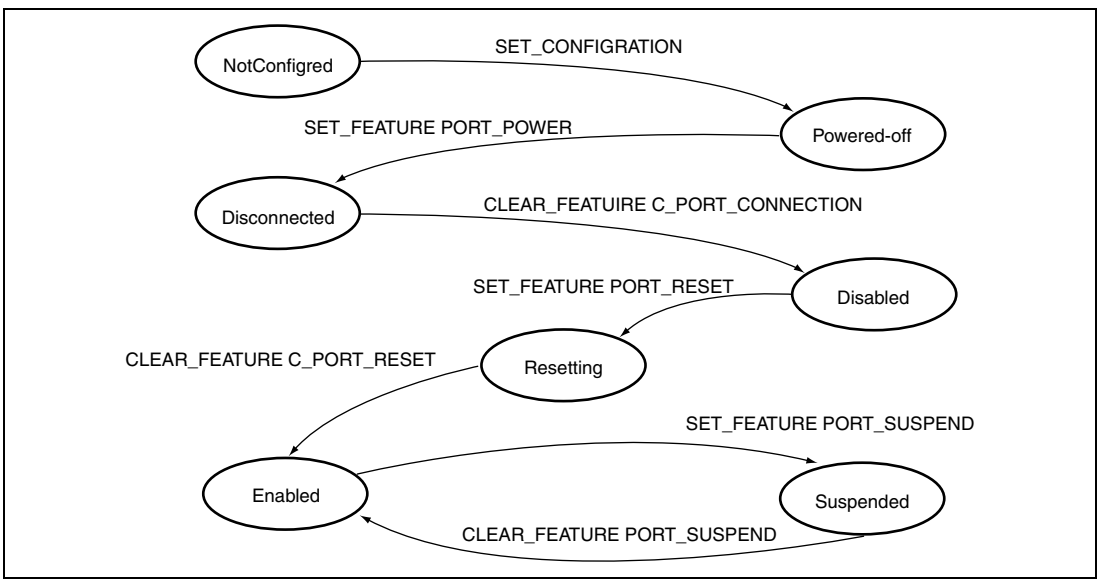

Figure 2.2 Port State Transition State

- NotConfigured
  A hub is unconfigured and ports are undefined
- Powered-off
  Power supply to the ports are stopped
- Disconnected
  Though power supply to the ports is started, no device is connected to the ports.

- Disabled

  Though function devices are connected to ports, a bus has not been reset yet.
  Ports do not transfer packets to the connected function devices.
- Resetting

  Under bus reset.
- Enabled

  The state after a bus reset. Ports carry out the packet transfer between the connected host and function devices.
- Suspend

  The state is suspended.

2.5.1 Setting Port State

To set a port state, check the port state and then set the port state.

To set a port sate, use class commands: SetPortFeature and ClearPortFeature. SetPortFeature is used for setting a port sate and ClearPortFeature is used for recognizing a change of port.

When the state of port is set by SetPortFeature or a change of port state is reported, check the port state by GetPortStatus. The change of port state is defined by checking the change using ClearPortFeature.

2.5.2 Checking Port State

The current port state can be checked by using a class command GetPortStatus. GetPortStatus returns data in 4-byte unit. Table 2.3 shows the formats.

Table 2.3 GetPortStatus Command Return Formats

Bytes	Bit	Descriptions
00 and 01	0	Current Connect Status: (PORT_CONNECTION) Indicates whether a function device is currently connected to this port. 0: A function module is not connected to this port 1: A function module is connected to this port
	1	Port Enabled/Disabled: (PORT_ENABLE) Indicates a port is enabled/disabled. 0: Port is disabled 1: Port is enabled
	2	Suspend: (PORT_SUSPEND) Indicates whether or not a port is suspended. 0: A port is not suspended. 1: A port is suspended.
	3	Over-current Indicator: (PORT_OVER_CURRENT) Indicates whether a port is overcurrented. 0: A port is not overcurrented. 1: A port is overcurrented.
	4	Reset: (PORT_RESET) Indicates whether a port is carrying out a bus reset. 0: A port is not carrying out a bus reset. 1: A port is carrying out a bus reset.
	5 to 7	Reserved
	8	Port Power: (PORT_POWER) Indicates a power supply state of port. 0: A port is not supplied the power. 1: A port is supplied the power.
	9	Low Speed Device Attached: (PORT_LOW_SPEED) Indicates whether a function device connected to a port is a low-speed device. This bit is enabled only when a function device is connected to a port. 0: A connected function device is a full-speed device. 1: A connected function device is a low-speed device.
	10 to 15	Reserved

Bytes	Bit	Descriptions
02 and 03	0	Connect Status Change: (C_PORT_CONNECTION) Indicates whether a port connection state has changed. 0: A port connection state has not changed 1: A port connection state has changed
	1	Port Enable/Disable Change: (C_PORT_ENABLE) Indicates whether a port state has changed. 0: A port state has not changed. 1: A port state has changed.
	2	Suspend Change: (C_PORT_SUSPEND). Indicates whether a port suspend state has changed. 0: A port suspend state has not changed. 1: A port suspend state has changed.
	3	Over-Current Indicator Change: (C_PORT_OVER_CURRENT) Indicates whether a port over-current state has changed. 0: A port over-current state has not changed. 1: A port over-current state has changed.
	4	Reset Change: (C_PORT_RESET) Indicates whether a port bus reset state has changed. 0: A port bus reset state has not changed. 1: A port bus reset state has changed.
	5 to 15	Reserved

2.5.3 Power Supply from Port

The power supply from the host to each port is turned on by a class command after hubs have been configured. As a result, bus-powered function devices can be supplied the power. A port checks the power supply. When the extraordinary amount of current is supplied, over-current is detected and the power supply is stopped.

When a function module is connected, the value of power supply is up to 100 mA for 1 port. Though the function devices are high-power bus-powered devices, the devices should be operated with 100 mA until they have been configured. After the host recognizes the required value by descriptor information and function devices are configured, required power is supplied.

2.5.4 Port Device Detection

The connection/disconnection of devices can be detected by checking the electric potential of the D+ pin and D− pin of every port. When a device is not connected to a port, the voltage level of the D+ pin and D− pin is VIL (0.8 V at the maximum) or less because of a 15 kΩ pull-down resistor.

When a function device is connected, the pin levels of the D+ pin and D− pin changes to VIH (2.0 V at the minimum) or more by the included 15kΩ pull-up resistor. When the state is maintained for 2.5 μs or more, the connection/disconnection of devices can be detected.

When the connection has been detected, it is reported to the host by an interrupt transfer. The packet format of interrupt transfer is a bitmap format. A 1 is written to the changed port, while a 0 is written to the unchanged port then it is reported to the host.

Bit	**n**	**…**	**2**	**1**	**0**
Change generated part	Change of port n	…	Change of port 2	Change of port 1	Change of hub

When the D+ pin is pulled up, the function module is recognized as a full-speed device. When the D− pin is pulled up, a function module is recognized as a low-speed device.

2.5.5 From Power Supply to Packet Transmission

Some timings are specified between start of power supply to a port after configuration and first transmission of packets to the connected device. Packets should be sent according to the specified timing. Figure 2.3 shows the specified time and changes of a port.

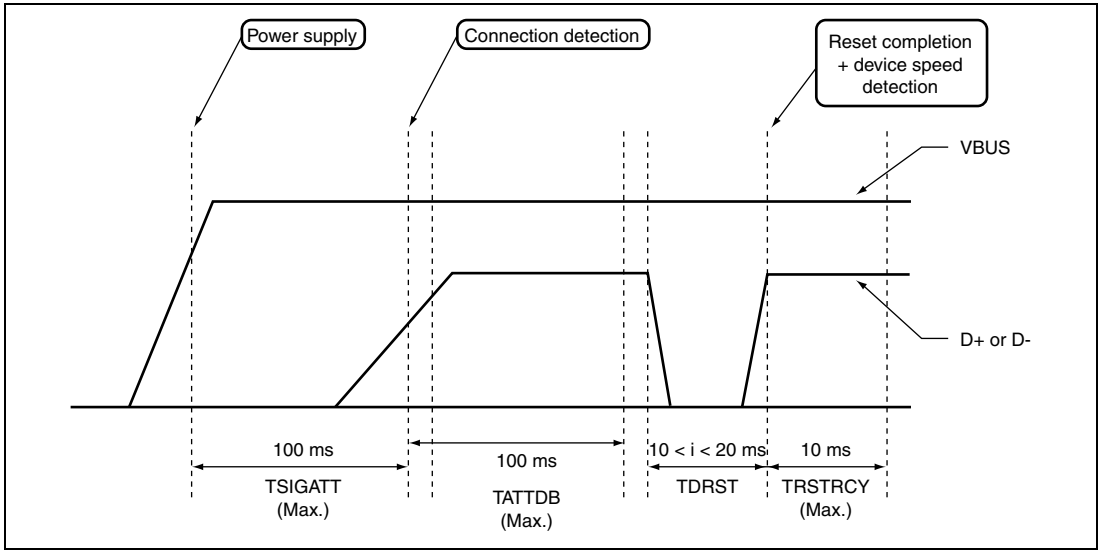

Figure 2.3 Timing of Port Change

2.5.6 Port State Transition

The host needs to recognize the current port state and indicate the next state to shift to by cooperating with hubs. Figure 2.4 shows the flow of the state transition of the host, hub, port, and function device in each procedure stage; unconfigured state of hub, start of power supplying from port, connecting bus-powered function device, activating function device by bus power, checking and recognizing that a function device is connected to a port.

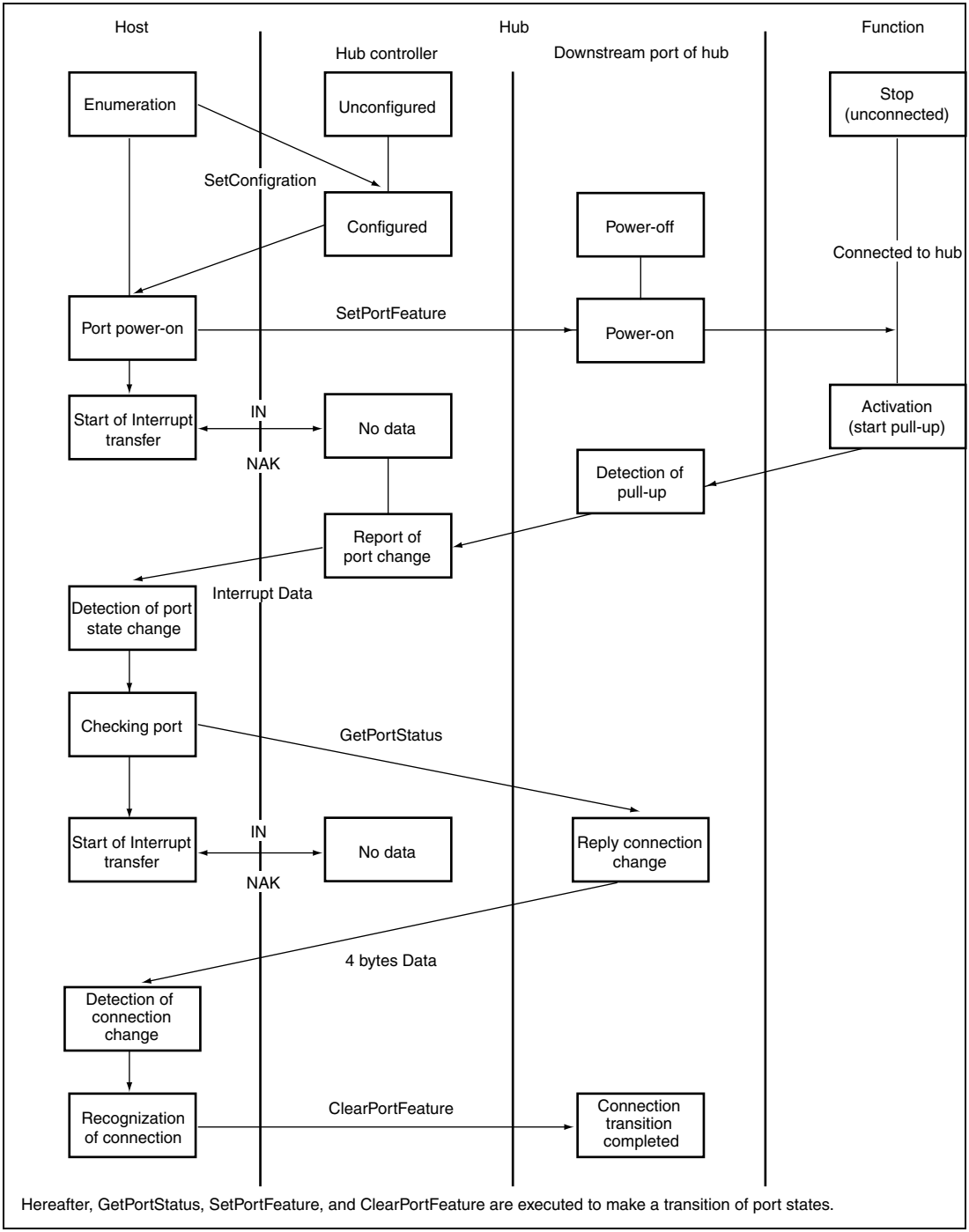

Figure 2.4 State Transition Diagram of Host, Hub, Port, and Function Device

Section 3 Development Environment

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

- SH7727 Solution Engine (type number: MS7727SE01, hereafter referred to as SH7727SE) manufactured by Hitachi ULSI Systems Co., Ltd.
- SH7727 E10A Emulator manufactured by Renesas Technology Corp.
- PC (Windows® 95/98) equipped with a PCMCIA slot
- PC (Windows® XP) to serve as the USB host
- USB cable
- USB hub
- High-Performance Debugging Interface (hereafter called HDI) manufactured by Renesas Technology Corp.
- High-Performance Embedded Workshop (hereafter called HEW) manufactured by Renesas Technology Corp.

3.1 Hardware Environment

Figure 3.1 shows device connections.

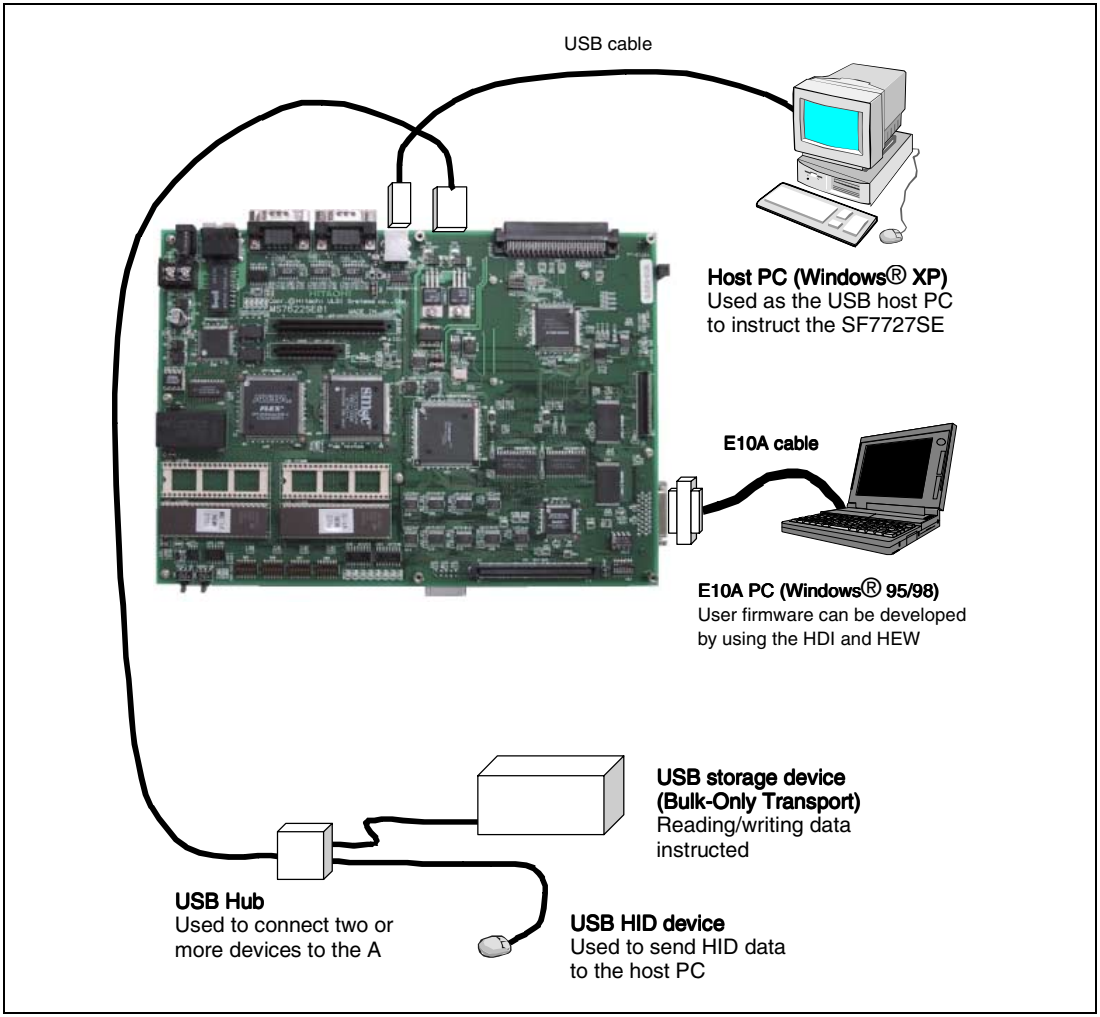

Figure 3.1 Device Connections

1. SH7727SE

   The DIP switch settings on the SH7727SE board must be changed from those at shipment. Before turning on the power, ensure that the DIP switches are set as shown in table 3.1. There is no need to change any other DIP switches.

Table 3.1 DIP Switch Settings

Switch	At Time of Shipment	After Change	DIP Switch Function
Baseboard SW1-6	Off	On	Select endian mode
Baseboard SW1-8	Off	On	Select E10A emulator mode
Baseboard JP14	1 – 2 short	2 – 3 short	Select USB D+ pull-up control signal
Baseboard JP15	1 – 2 short	2 – 3 short	Select USB VBUS signal detection port

2. USB host PC

   A PC with Windows® XP installed, and with a USB port, is used as the USB host. This system uses USB Mass Storage Class (Bulk-Only Transport) and HID Class device drivers, installed as a standard part of the Windows® XP, so that there is no need to install new drivers.

3. E10A PC

   The E10A should be inserted into a PC card slot and connected to the SH7727 via an interface cable. After connection, starts the HDI and perform emulation.

4. USB hub

   The USB Hub is used to connect two or more devices to the A connector of the SH7727SE.

5. USB storage device

   The USB storage device should be connected to the SH7727SE via the USB hub. The device stores and reads data between the USB host through the SH7727SE.

6. USB HDI device

   The USB HDI device should be connected to the SH7727SE via the USB hub. The device inputs data to the USB host PC through the SH7727SE.

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 SH7727 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 shown in figure 3.2.

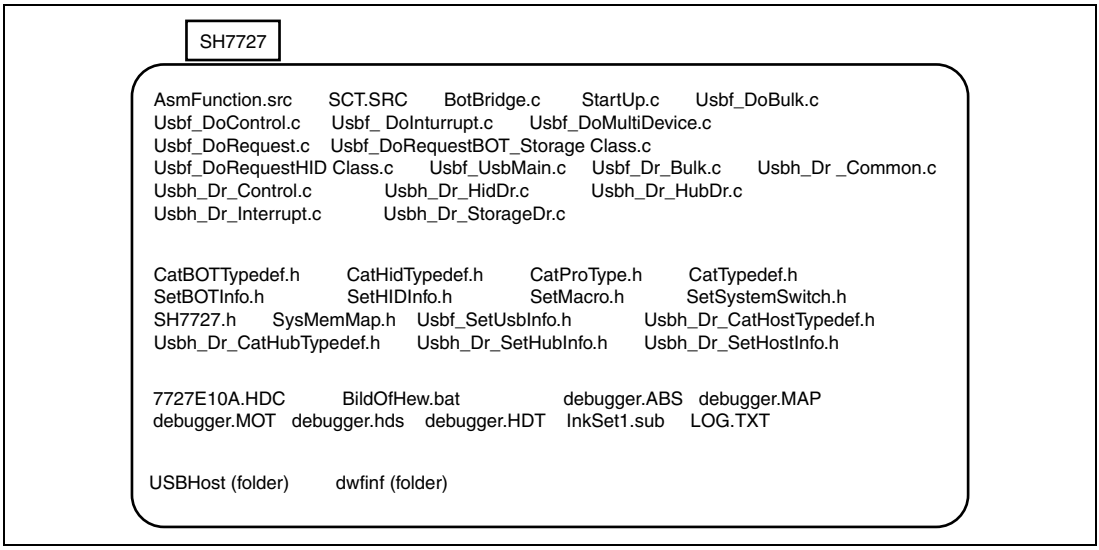

Figure 3.2 Files Included in SH7727 Folder

3.2.2 Compiling and Linking

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

High-Performance Embedded Workshop Version 1.0 (release 9) (hereafter called 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).

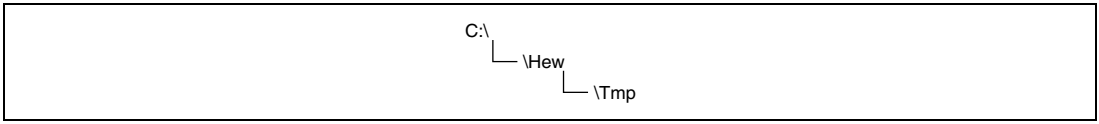

Figure 3.3 Creating a Working Folder

Next, the folder in which the sample program is stored (SH7727) should be copied to any drive. In addition to the sample program, this folder contains a batch file named BuildOfHew.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 BuildOfHew.bat is executed, compiling and linking are performed. As a result, a file named debugger.ABS, which is an executable file, 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 to a folder other than C:\Hew, the compiler path setting and settings for environment variables used by the compiler in BuildOfHew.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 shc.exe, and the setting for the environment variable shc_lib used by the compiler should be set to the folder of shc.exe, the shc_inc setting should be changed to the folder of machine.h, and the setting of shc_tmp should specify the working folder for the compiler. The path of shcpic.lib should be specified for the library.

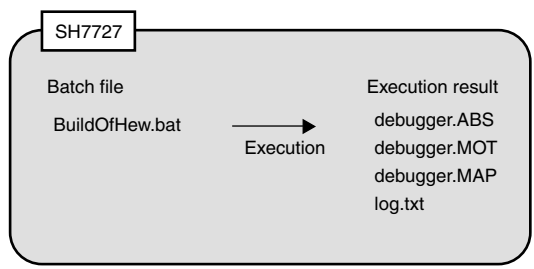

Figure 3.4 Compile Results

3.3 Loading and Executing the Program

Figure 3.6 shows the memory map for the sample program.

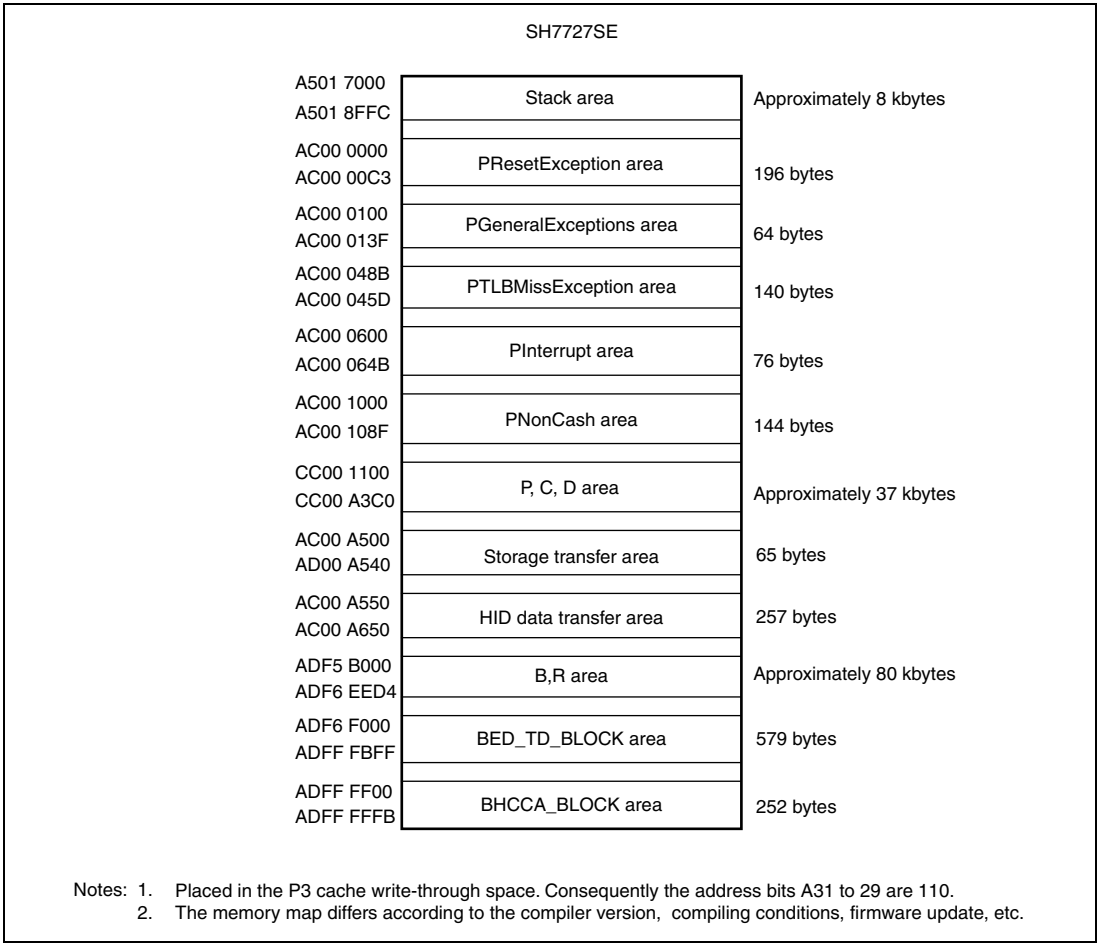

Figure 3.5 Memory Map

As shown in figure 3.5, this sample program allocates areas of PResetExecution, PGeneralException, PTLBMissException, PInterrupt, PNonCash, P, C, D, R, and B to the SDRAM and area of Stack to the on-chip memory. In order to use the break and other functions of the E10A, the program must be placed in RAM in this way. These memory allocations are specified by the InkSet1.sub file in the SH7727 folder. When incorporating the program in ROM by writing it to flash memory or some other media, this file must be modified.

3.3.1 Loading the Program

In order to load the sample program into the SDRAM of the SH7727SE, the following procedure is used.

- Insert the E10A in which the HDI has been installed into the E10A PC, connect the E10A to the SH7727SE via a user cable
- Turn on the power to the E10A PC for start up.
- Initiate the HDI.
- Turn on the power to the SH7727SE.
- A dialog (figure 3.6) is displayed on the PC screen; turn the SH7727SE reset switch (SW1) on, and after resetting the CPU, click the OK button or press the Enter key.
- Select CommandLine in the View menu to open a window (figure 3.7), click the BatchFile button on the upper left, and specify the 7727E10A.hdc file in the SH7727 folder. As a result, the BSC is set and access to the SDRAM is enabled.
- Select LoadProgram... from the File menu; in the Load Program dialog box, specify debugger.ABS in the SH7727 folder.

Through the above procedure, the sample program can be loaded into the SDRAM of the SH7727SE.

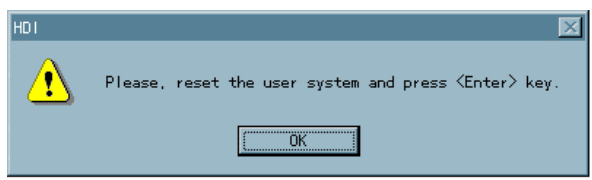

Figure 3.6 Reset Request Dialog

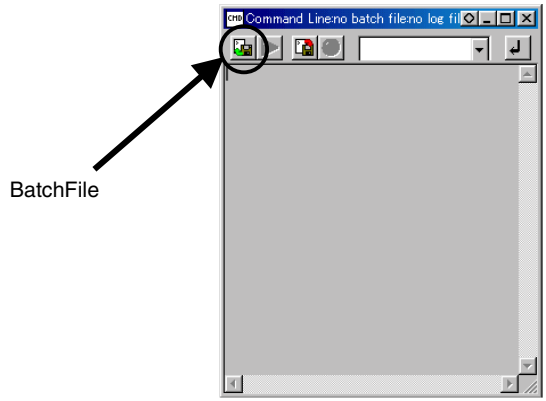

Figure 3.7 Command Line Input

3.3.2 Execution

Select Go from the Run menu bar to execute the program.

3.4 Multi Function Device

An example using Windows® XP is explained below.

Insert the series A connector of the USB cable to the SH7727SE with a program in execution, and connect the series B connector at the other end of the cable to the USB hub. Then connect a storage device and mouse to the ports of the USB hub.

Next, insert the series B connector of the USB cable to the SH7727SE, and connect the series A connector at the other end of the cable to the USB host PC.

After the SH7727SE has been recognized as a peripheral function device by executing the control transfer and bulk transfer, a mass-storage device is appeared below the USB controller which is on the Device Manager screen. As a result, the host PC recognizes the SH7727SE as a multi function device which consists of a storage and mouse, and the local disk is mounted on the main computer and the USB mouse can be used.

Through the process explained above, the SH7727SE can be used as the USB connected multi function device.

Section 4 Overview of Sample Program

Features of the sample program and its structure are explained in this section. This sample program runs on the SH7727SE using the USB host and USB function modules. Therefore, the SH7727SE as the USB host module controls the function device and the USB hub which is connected to the USB series A connector on it, and also serves as the USB function to the host PC.

Of the on-chip peripheral modules of the SH7727E, the sample program uses the on-chip timer (one channel) other than the USB host and USB function modules.

The USB transfer by the USB function module is started by an interrupt requested from the USB function module.

The USB transfer by the USB host module is started when the completion of the process requested to the USB host module in the steady routine is recognized.

Of the interrupts requested from the on-chip peripheral modules in the SH7727, there are two interrupts related to the USB function module; the USB function moduleI0 and the USB function moduleI1. In this sample program only the USB function moduleI0 is used. The interrupt related to the USB host module is the USB host module.

Features of this sample program are as follows.

- Control transfer can be performed by using the USB host module
- Bulk-OUT transfer can be used to transmit data to a device by using the USB host module
- Bulk-IN transfer can be used to receive data from a device by using the USB host module
- Interrupt transfer can be used to receive data from a device by using the USB host module
- Data can be transmitted/receiver from the host PC by using the USB function module
- Data between the host PC and device is relayed by the SH7727SE by cooperating the USB host module and the USB function module

4.1 Entire Structure of Sample Program

The entire structure of this sample program is shown in figure 4.1. The numbers 1 to 8 indicate data transfers.

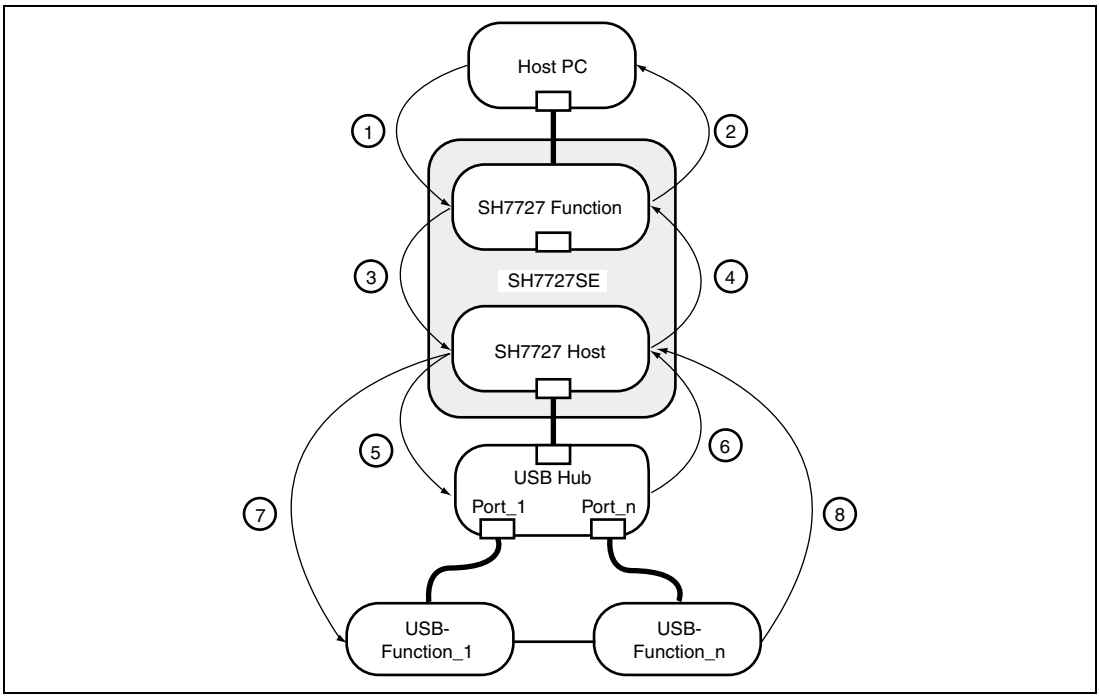

Figure 4.1 Entire Figure

The layered structure of sample program is shown in figure 4.2.

Data transfers 1 and 2 shown in figure 4.1 are performed in the Device layer of the SH7727SE.

Data transfers 3 and 4 shown in figure 4.1 are performed in the Host-Function link layer of the SH7727 SE.

Data transfers 5 to 8 shown in figure 4.1 are performed in the USB system software layer of the SH7727SE.

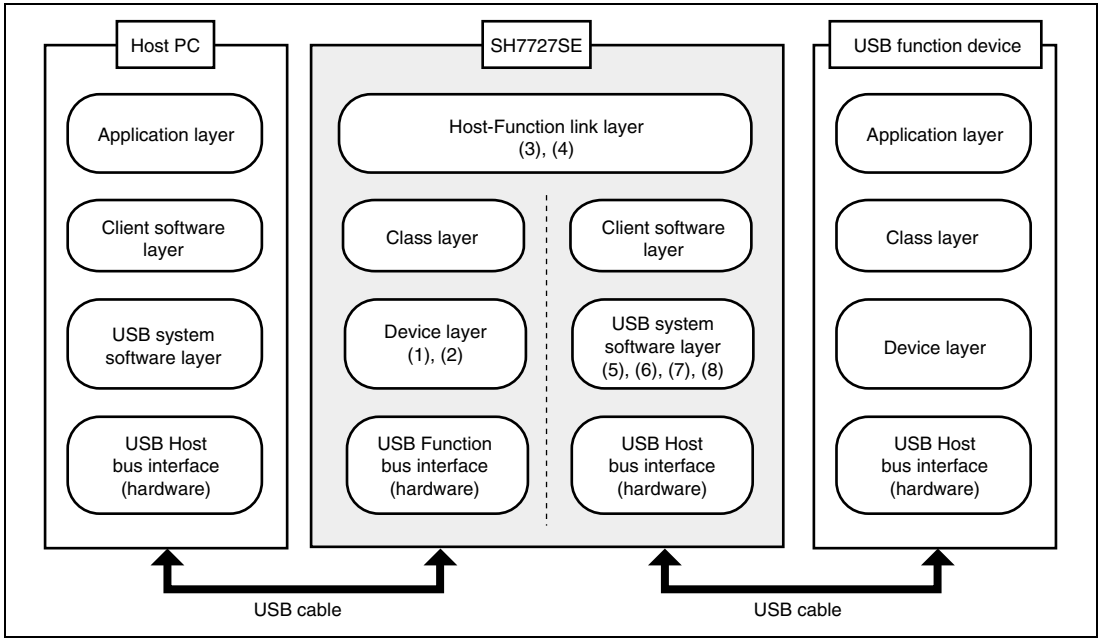

Figure 4.2 Layer Structure of System

This sample program can be categorized into three major programs.

- Program of the SH7727 function which performs data transfer with the host PC.
- Program of the SH7727 host which controls the USB hub and USB function device to perform data transfer between them.
- Link program which connects the program of the SH7727 function and program of the SH7727 host to perform data transfer between them.

Above three programs are shown in figure 4.2.

The program of the SH7727 function corresponds to the Class layer and Device layer.

The program of the SH7727 host corresponds to the Client software layer. According to the devices to control, the layer is divided into the Hub driver, HID driver, and storage driver.

The link program corresponds to the Host-Function link layer.

4.2 State Transition Diagram

Figure 4.3 shows a state transition diagram of this sample program. As is shown in figure 4.3, there are transitions between six states.

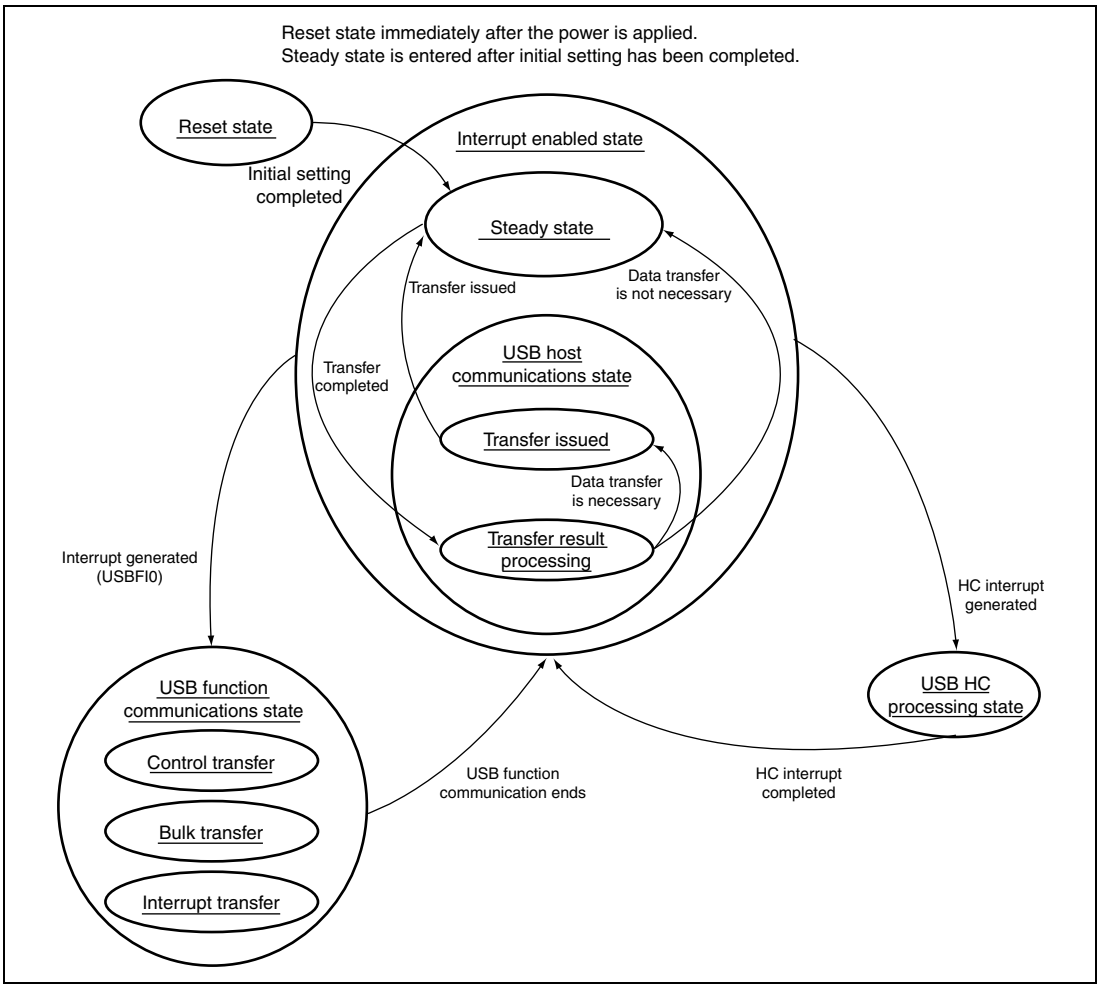

Figure 4.3 State Transition Diagram

- Reset State

  This state is entered by power-on reset and manual reset. In the reset state, the initial settings of the SH7727 mainly performs.

- Interrupt Enabled State

  Interrupts are enabled in this state. This state includes the steady state and USB host communications state.

- Steady State

  When initial settings are completed, the steady state is entered in the main loop.
- USB Host Communications State

  This state is entered when the USB hub and USB function devices are connected to the SH7727 host.
- USB HC Processing State

  This state is entered when an interrupt is occurred from the USB host module in the interrupt enabled state.
- USB Function Communication State

  This state is entered when an interrupt is occurred from the USB function module in the interrupt enabled state. In the USB function 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 the interrupt flag register 0 (USBIFR0) and there are eight types in all. When an interrupt is occurred, the corresponding bit in USBIFR0 is set to 1.

4.3 USB Function Communication State

The USB function communication state is categorized into three states according to the transfer type (see figure 4.4). When an interrupt occurs, first the USB function communications state is entered, and then a further branch to a transfer state according to the type of interrupt is performed. The brunching process is explained in section 5, Sample Program Operation.

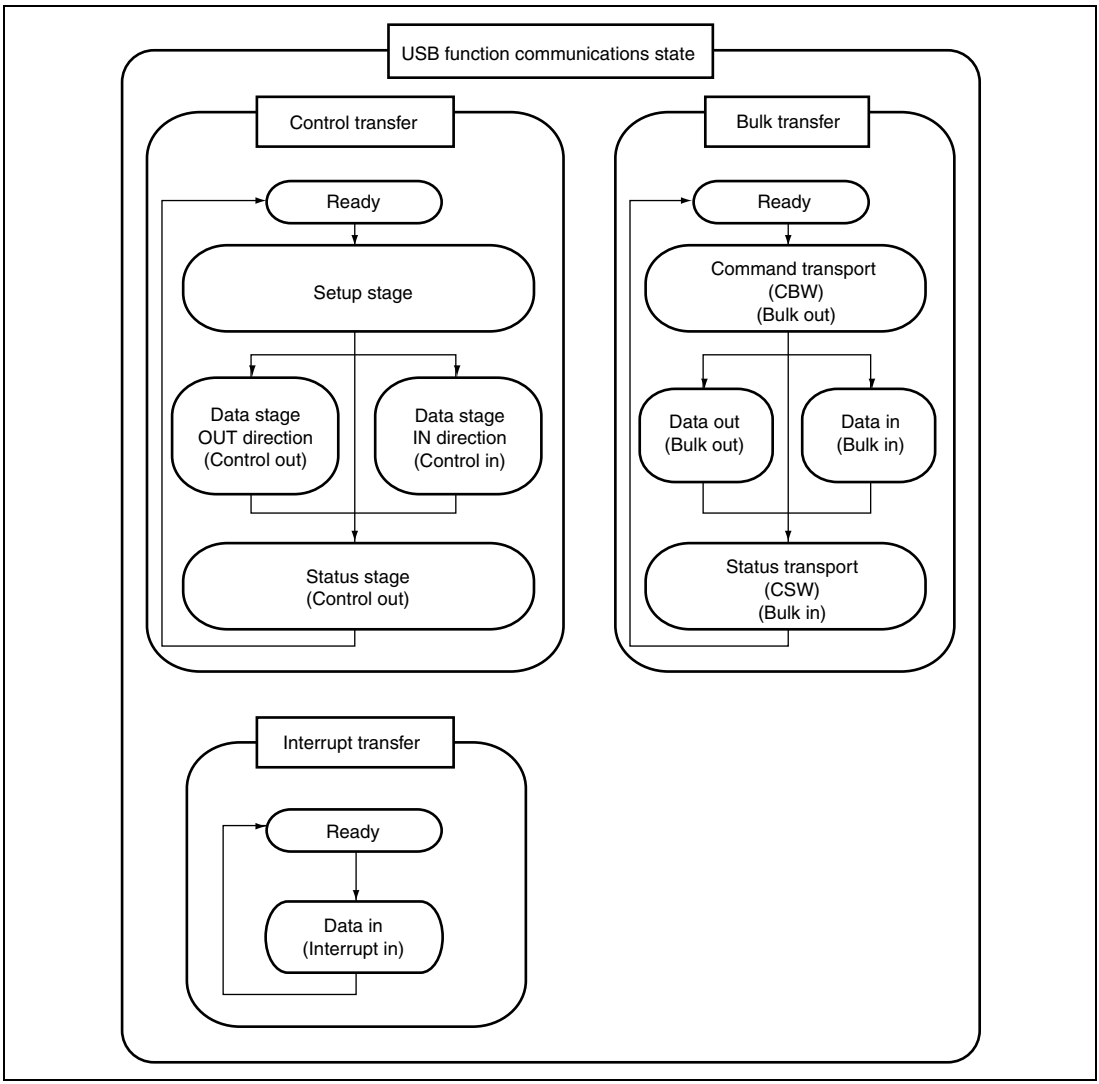

Figure 4.4 USB Function Communications State

4.3.1 Control Transfers

In this sample program, descriptor information and class command are returned to the host PC. Descriptor information to return is created based on descriptor information which is obtained from the function device connected to the SH7727 host.

4.3.2 Bulk Transfers

In this sample program, through following two processes, data between the host PC and a storage function device is transferred using bulk transfer.

1. Packets for a storage device are received from the host PC. Then the packets are sent to the link program.
2. Packets received from a storage device are sent to the host PC via the link program.

4.3.3 Interrupt Transfers

In this sample program, interrupt transfer is used for HID device data transfer. If there is data received from the HID device connected to the SH7727 host in the transfer buffer, it is transmitted by the IN request from the host PC. If there is no data in the transfer buffer, a NAK is transmitted by hardware. The transmit data is written from the HID device to the transfer buffer by using the SH7727 host program.

4.4 USB Host Communication State

The USB host communication state is categorized into two sates: state in which transfer is issued and the another state in which transfer result is processed. Transfer is a data transfer to the USB hub and USB function device performed by the SH7727 host. In this sample program, a control transfer completes when the processing of the Setup and Status stages are completed. The bulk and interrupt transfers complete when one packet is completely transferred.

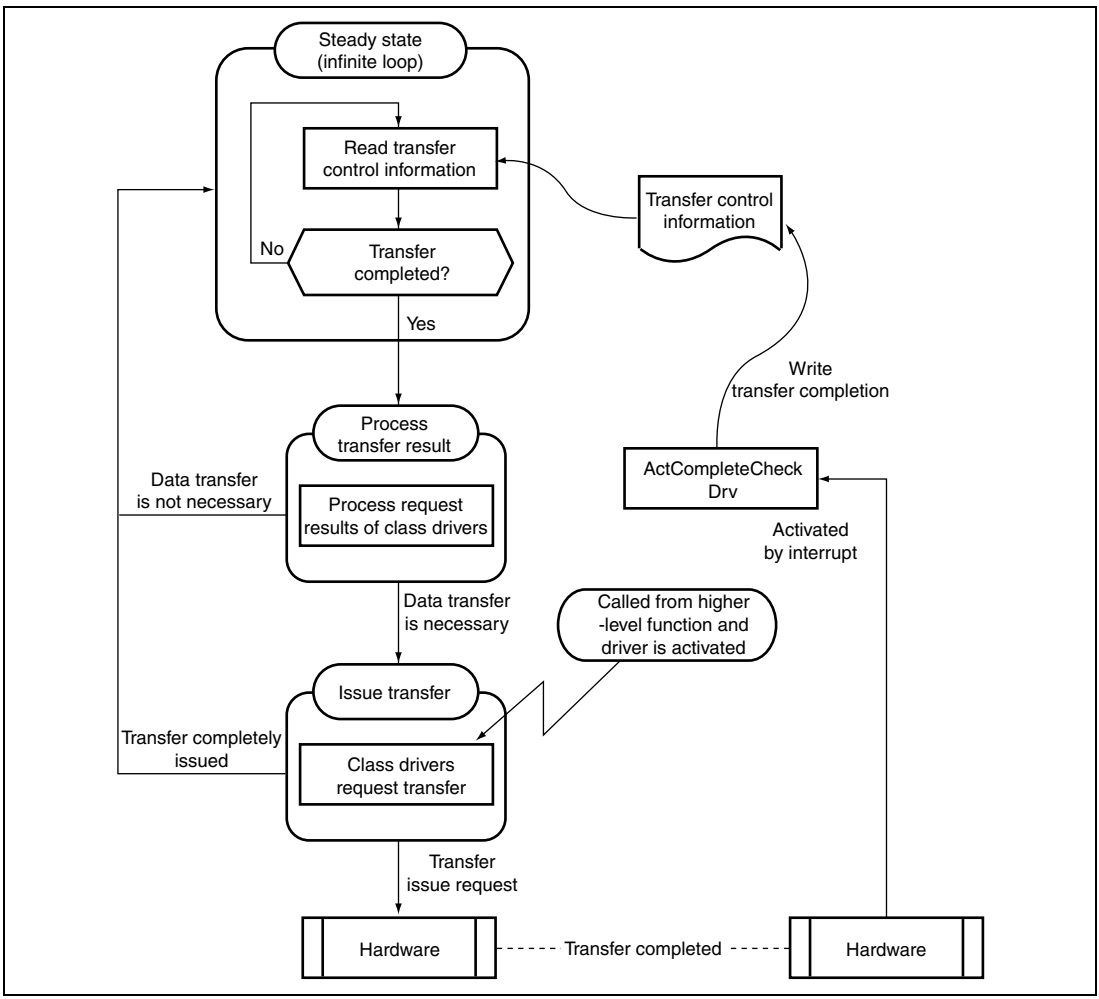

Figure 4.5 USB Host Communication State

In a state for issuing transfer, data transfer between the SH7727 host and devices is carried out by using the following transfer methods: control transfer, bulk transfer, and interrupt transfer.

In a state for processing result, process is carried out based on a result of data transfer with a device.

Each driver performs data transfer with a device by two states and three transfer methods as necessary.

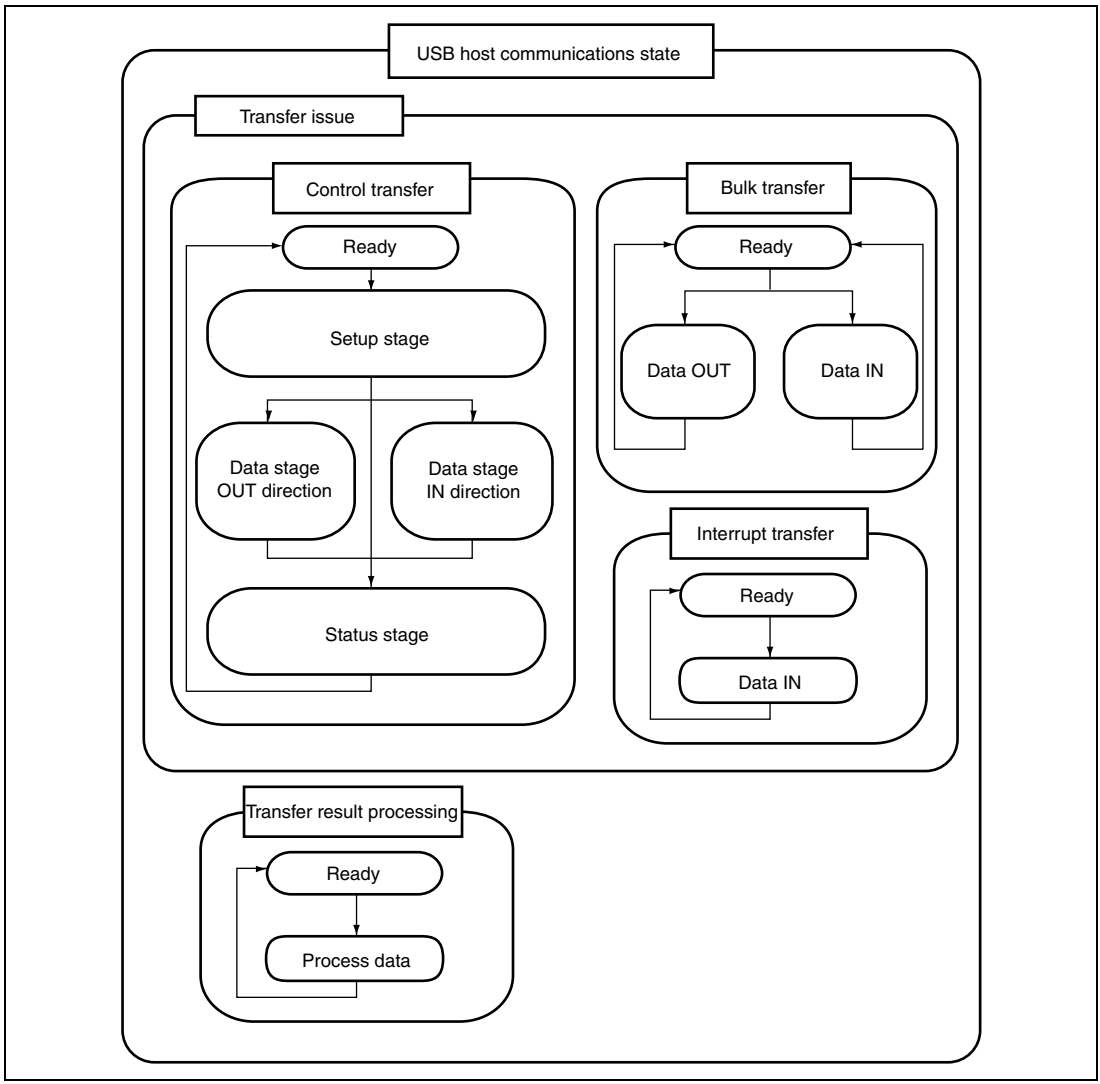

Figure 4.6 USB Host Communications State

4.4.1 Transfer Issue

In a state for issuing a transfer, data transfer between devices and the SH7727 host is performed by using the following transfer methods: control transfer, bulk transfer, and interrupt transfer. The USB transfer must be started from the host, so that data transfer with a device is performed in this state.

Which transfer is performed by using a transfer method changes according to the USB hub and USB function device connected to the SH7727 host. However, for obtaining connected device information, device address allocation to the connected device, and enabling connected device usage, GET_DESCRIPTOR, SET_ADDRESS, and SET_CONFIGURATION are used respectively. These three processes are indispensable.

4.4.2 Control Transfer

A control transfer is mainly used to obtain device information and set device operating state. Therefore, a control transfer is the first transfer to be carried out when a device is connected to the SH7727SE.

A series of control transfer process is carried out through two or three stages. The stages of controller transfer are classified into the Setup stage, Data stage, and Status stage.

4.4.3 Bulk Transfers

A bulk transfer is used to transfer large amounts of data without any error. The transfer is performed with no time limitation. Though data transfer speed is not guaranteed, the contents of data are guaranteed.

In this sample program, communication between the SH7727SE and a storage device is made by using the bulk transfer.

4.4.4 Interrupt Transfer

An interrupt transfer performs data transfer at least once within a specified interval. Though the data size handled at one transfer is small, the worst transfer rate and contents of data are guaranteed. Communication between the SH7727SE and a hub or HID devices are made by using the interrupt transfer.

4.4.5 Transfer Result Processing

This state is entered when a transfer which is generated from the host to a device has completed in the transfer issue state (in the case of a control transfer when the Status stage ends, while in the case of bulk and interrupt transfers when one transaction completed).

In this state, checking the result of the data transfer to a device, obtaining data from a device, and activating next process are carried out by using the three transfer methods: control transfers, bulk transfers, and interrupt transfers).

4.5 File Structure

This sample program consists of 18 source files and 15 header files. The overall file structure is shown in table 4.1. Several functions are arranged in one file depending on its transfer method or function type. Figure 4.7 shows the layered configuration of these files. The relationships among files are shown as layered configuration in figure 4.8 and figure 4.9.

Table 4.1 File Configuration

File Name	Description
StartUp.c	Microcomputer initial setttings
Usbf_UsbMain.c	Decides the interrupt source and transmits/receives packets
Usbf_DoControl.c	Executes control transfer
Usbf_DoBulk.c	Executes bulk transfer
Usbf_DoInterrupt.c	Executes interrupt transfer
Usbf_DoRequest.c	Processes setup command issued by the host
Usbf_DoRequestBOT_StorageClass.c	Processes class command of Mass Storage Class (Bulk-Only Transport)
Usbf_DoRequestHIDClass.c	Processes HID class command
BOTBridge.c	Bridges Mass Storage Class (Bulk-Only Transport) data between the USB host and USB function modules
Usbf_DoMultiDevice.c	Creates descriptor information
Usbh_Dr_Common.c	Driver layer common function
Usbh_Dr_Control.c	Requests control transfer by the USB host
Usbh_Dr_Bulk.c	Requests bulk transfer by the USB host
Usbh_Dr_Interrupt.c	Requests interrupt transfer by the USB host
Usbh_Dr_HubDr.c	Processes Hub control
Usbh_Dr_HidDr.c	Processes HID device control
Usbh_Dr_StorageDr.c	Processes storage device control
AsmFunction.src	Sets stack
CatHidTypedef.h	Sets type necessary for HID class
CatBOTTypedef.h	Sets type necessary for BOT class
Usbh_CatHubTypedef.h	Sets type necessary for Hub class
CatProType.h	Global variable and function prototype decralation
CatTypedef.h	Basic structure definition used by USB firmware

File Name	Description
Usbh_CatHostTypedef.h	Sets type necessary for USB host
SetBOTInfo.h	Initial setting of variables necessary for BOT class control
SetHIDInfo.h	Initial setting of variables necessary for HID class control
Usbh_SetHostInfo.h	Initial setting of variables necessary for the USB host control
Usbh_Dr_SetHubInfo.h	Initial setting of variables necessary for USB hub control
SetMacro.h	Macro definition
SetSystemSwitch.h	Sets system operation
Usbf_SetUsbInfo.h	Initial setting of variables necessary for USB
SysMemMap.h	Address definition of HS7727SE memory map
SH7727.h	Sh7727 register definition

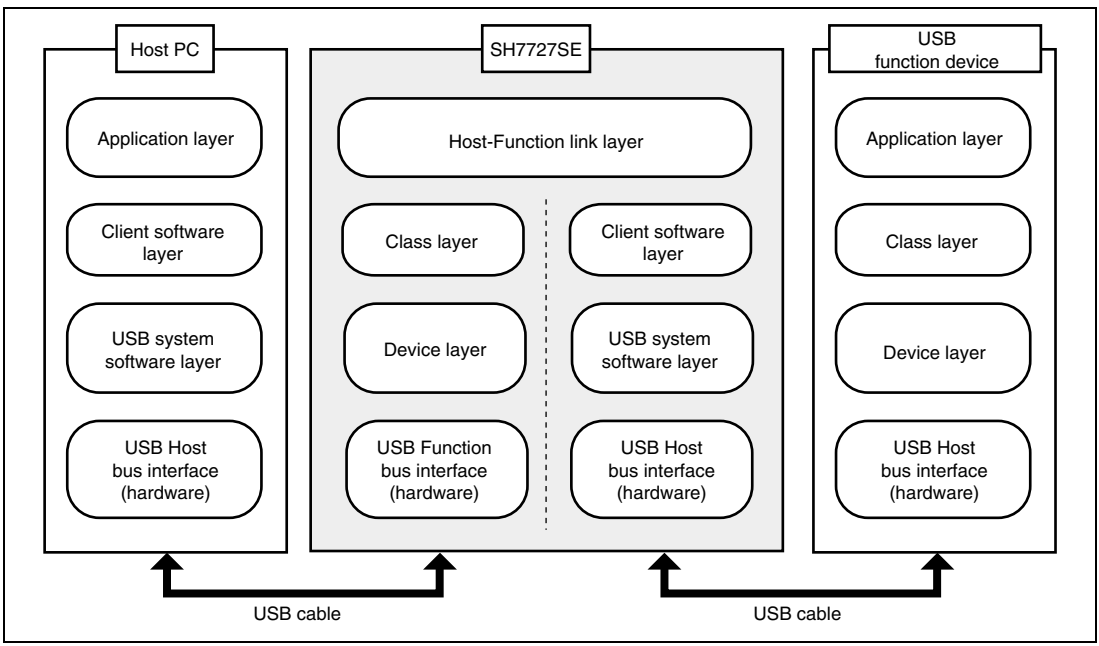

Figure 4.7 Layered Configuration of System

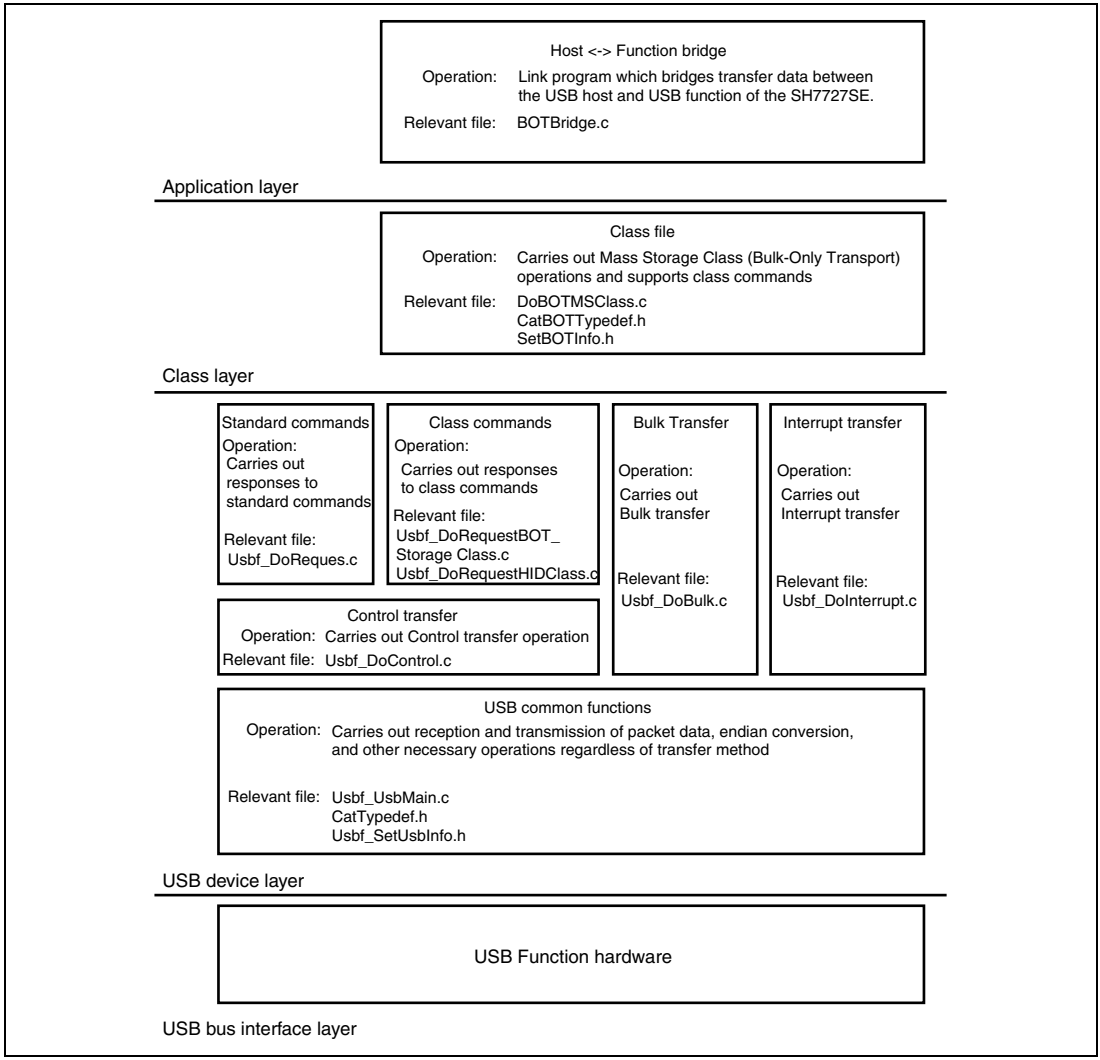

Figure 4.8 Layered Configuration of SH7727SE Function

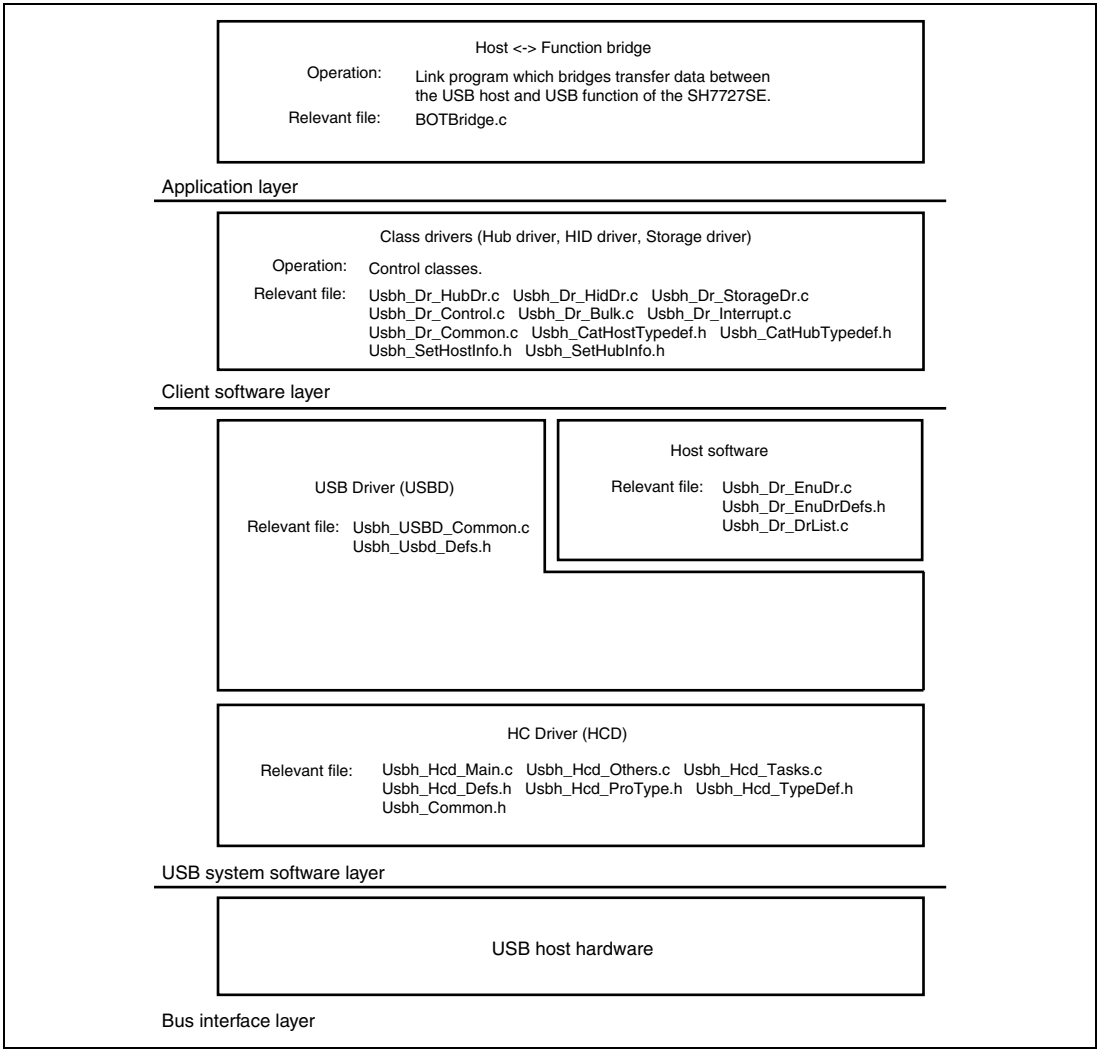

Figure 4.9 Layered Configuration of SH7727SE

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
StartUp.c	CallReseException	Operates for reset exception and calls the following function to be executed
	CallGeneralException	Calls function for the general exception other than TLB miss occurrence
	CallTLBMissException	Calls function for the TLB miss occurrence
	CallInterrupt	Calls function for an interrupt request
	SetPowerOnSection	Initializes modules and memory, and jumps to main loop
	_INITSCT	Copies variables that have initial settings to the RAM work area
	InitMemory	Clears RAM area used in bulk communicatuions
	InitSystem	Pull-up control of the USB bus

When a power-on reset or manual reset is carried out, SetPowerOnSection in StartUp.c is called. At this point, the SH7727 initial settings are carried out and the RAM area used for the bulk transfers is cleared.

Table 4.2-2 Usbf_UsbMain.c

File in Which Stored	Function Name	Purpose
Usbf_UsbMain.c	BranchOfInt	Decides interrupt sources, and calls function according to the interrupt
	GetPacket	Write data transferred from the host module of the host PC to RAM
	GetPacket4	Write data transferred from the host module of the host PC to RAM in longwords. Ring buffer supported version (not used in this sample program).
	GetPacket4S	Write data transferred from the host module of the host PC to RAM in longwords. High performance version (not used in this sample program).
	PutPacket	Write data to transfer to the host module of the host PC to USB module
	PutPacket4	Write data to transfer to the host module of the host PC to USB module in longwords. Ring buffer supported version (not used in this sample program).

File in Which Stored	Function Name	Purpose
Usbf_UsbMain.c	PutPacket4S	Write data to transfer to the host module of the host PC to USB module in longwords. High performance version (not used in this sample program)
	SetControlOutContents	Overwrite data with data transmitted from the host
	SetUsbModule	Initial settings of USB module
	ActBusReset	Clears FIFO on bus reset reception
	ConvRealn	Reads data of a specified byte length from a specified address
	ConvReflexn	Reads data of a specified byte length from a specified address in descending order.

In Usbf_UsbMain.c, interrupt sources are decided by the USB interrupt flag register, and functions are called according to the interrupt type. Also, packets are transmitted and received between the host controller of the host PC and the USB function module.

Table 4.2-3 Usbf_DoControl.c

File in Which Stored	Function Name	Purpose
Usbf_DoControl.c	ActControl	Controls the Setup stage of control transfer
	ActControlIn	Controls the Data stage and Status stage of control-IN transfer (transfer in which Data stage is in the IN direction)
	ActControlOut	Controls the Data stage and Status stage of control-OUT transfer (transfer in which Data stage is in the OUT direction)
	ActControlInOut	Allocates the Data stage and Status stage of control transfer to ActControlIn and ActControlOut

When a control transfer interrupt (SETUP TS) is occurred, ActControl obtains the command, and DecStandardCommands decodes the command to decide command transfer direction. After that, when a control transfer interrupts (EP0o TS, EP0i TR, and EP0i TS) is occurred, according to the transfer direction of the command, ActControlInOut calls ActControlIn or ActControlOut, and the Data stage and Status stage is carried out.

Table 4.2-4 Usbf_DoBulk.c

File in Which Stored	Function Name	Purpose
Usbf_DoBulk.c	ActBulkOut	Performs bulk-OUT transfer
	ActBulkIn	Performs bulk-IN transfer
	ActBulkInReady	Prepares for bulk-IN transfer

These functions carry out processes of bulk transfers. ActBulkInReady is no used in Mass Storage Class (Bulk-Only transport).

Table 4.2-5 Usbf_DoInterrupt.c

File in Which Stored	Function Name	Purpose
Usbf_DoInterrupt.c	ActInterruptIn	Decodes a command issued by the host module and process standard commands.

This function carries out processes of interrupt transfers.

Table 4.2-6 Usbf_DoRequest.c

File in Which Stored	Function Name	Purpose
Usbf_DoRequest.c	DecStandardCommands	Performs interrupt-IN transfer.
	DecVenderCommands	Processes a vendor command

During control transfer, commands sent from the host module of the host PC are decoded and processed. In this sample program, a vendor ID of 045B (vendor: Renesas) is used. When developing a product, the user 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 4.2-7 Usbf_DoRequestBOT_StorageClass.c

File in Which Stored	Function Name	Purpose
Usbf_DoRequestBOT_StorageClass.c	DecBOTClassCommands	Processes USB Mass Storage Class (Bulk-Only Transport) commands

During control transfer, the Mass Storage Class (Bulk-Only Transport) commands sent from the host module of the host PC is decoded and processed.

Table 4.2-8 Usbf_DoRequestHIDClass.c

File in Which Stored	Function Name	Purpose
Usbf_DoRequestHIDClass.c	DecHIDClass Commands	Processes HID class commands

During control transfer, the HID class commands sent from the host module of the host PC are decoded and processed.

Table 4.2-9 BOTBridge.c

File in Which Stored	Function Name	Purpose
BOTBridge.c	ActBulkOnly	Allocates Bulk-Only Transport by states and interrupt sources
	ActBulkOnly Command	Processes when CBW is sent from the host PC. Reads CBW data, and sends CBW to storage device.
	ActChangStateCBW	Function which is used to receive CBW transmission completion. Changes state.
	ActBulkOnlyIn	Controls data transport (transfer in which data stage is in the IN direction) and status transport
	ActBulkOnlyOut	Controls data transport (transfer in which data stage is in the OUT direction) and status transport
	ActCallBulkIn	Function which is used to receive response reception completion of bulk-In transfer request. When STALL is returned, this function changes state and issues Clear Feature.
	ActNop	Returns to called source function without processing anything
	ActBulkOnlyStallIn	When state is STALL, controls data transport (transfer in which data stage is in the IN direction) and status transport
	ActStallAfterCSW	During data transport, sends CSW to storage device
	ActFreeBKOUT	Function which is used to receive bulk-OUT transmission completion

BOTBridge.c bridges data of Mass storage class (Bulk-Only Transport) between the SH 7727 host and SH7727 function.

Table 4.2-10 Usbf_DoMultiDevice.c

File in Which Stored	Function Name	Purpose
Usbf_DoMultiDevice.c	BuildDescriptorHid	Creates descriptor information when HID class device is connected
	BuildDescriptor	Creates descriptor information when device of Mass Storage Class Bulk-Only Transport class is connected

In Usbf_DoMultiDevice.c, Descriptor information is created when the SH7727SE is connected to the host PC.

Table 4.2-11 Usbf_Dr_Common.c

File in Which Stored	Function Name	Purpose
Usbh_Dr_Common.c	ActCompleteCheck ClassDrv	Detects completion of processes which are requested to driver by class driver
	ActCompleteCheck Drv	Detection function for JOB which is requested by class driver
	EntryMemory	Sets memory area to use
	FreeUpMemory	Releases used memory area.
	EntryJobMemory	Entries necessary area for request which is used to request to the host driver.
	errorHost	Back to this function on USB host error operation

In Usbh_Dr_Common.c, common functions between each class driver are gathered.

Table 4.2-12 Usbh_Dr_Control.c

File in Which Stored	Function Name	Purpose
Usbh_Dr_Control.c	SendControl	Requests control transfer to specified device

In Usbh_Dr_Control.c, control transfer request is executed to the specified device address.

Table 4.2-13 Usbh_Dr_Bulk.c

File in Which Stored	Function Name	Purpose
Usbh_Dr_Bulk.c	SendBulk	Requests bulk transfer to specified device

In Usbh_Dr_Bulk .c, bulk-transfer request is executed to the specified device address.

Table 4.2-14 Usbh_Dr_Interrupt.c

File in Which Stored	Function Name	Purpose
Usbh_Dr_Interrupt.c	SendInterrupt	Requests interrupt transfer to specified device

In Usbh_Dr_Interrupt, interrupt transfer request is executed to the specified device address.

Table 4.2-15 Usbh_Dr_HubDr.c

File in Which Stored	Function Name	Purpose
Usbh_Dr_HubDr.c	OpenHubInterface	Carries out configuration. Performs control transfer to gather information of device, interface, and endpoint, then sets interface.
	WriteHubInterface	Cooperates with OpenHubInterface function to register information of device, interface, and endpoint
	ActChangPortState	Receives port change notification from hubs
	ActAfterJob	Activated by timer interrupt, and calls following function to activate.
	ActCallNewDevice	Starts bus enumeration of newly connected device
	ActCheckHubPort	Checks port state of hub
	ActContorlHubPort	Sets necessary setting according to port state
	ActCheckPortState	Checks specified port state (issues GET_STATUS For Port)
	ActSendSetFeature	Issues Set Feature
	ActSendControllear Feature	Issues Clear Feature
	ActCount	Sets timer unit to count for specified time (ms)
	ActFindHubPort	Returns to which downstream port of hub specified device address is connected

In Usbh_Dr_HubDr.c, connected hubs are controlled.

Table 4.2-16 Usbh_Dr_HidDr.c

File in Which Stored	Function Name	Purpose
Usbh_Dr_HidDr.c	OpenHidInterface	Carries out configuration. Performs control transfer to gather information of device, interface, and endpoint, then sets interface.
	WriteHidInterface	Cooperates with OpenHubInterface function to register information of device, interface, and endpoint, then begins interrupt transfer
	ActReceiveHidData	Receives data which is transmitted by interrupt transfer

In Usbh_Dr_HidDr.c, connected HID device is controlled, and data which has been transmitted by the interrupt transfer is sent to the SH7727 function.

Table 4.2-17 Usbh_Dr_StorageDr.c

File in Which Stored	Function Name	Purpose
Usbh_Dr_StorageDr.c	OpenMscBotInterface	Carries out configuration. Performs control transfer to gather information of device, interface, and endpoint, then sets interface.
	WriteMscBotInterface	Cooperates with OpenHubInterface function to register information of device, interface, and endpoint, then begins interrupt transfer
	ActReceiveHidData	Receives data which is transmitted by interrupt transfer

Figure 4.10 to 4.14 show 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 steady state, CallResetException calls other functions, and in the case of a transition to the USB communication state which occurs on an interrupt, interrupt function CallInterrupt calls BranchOfInt, and BranchOfInt calls other functions. Figure 4.10 to 4.14 show 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.

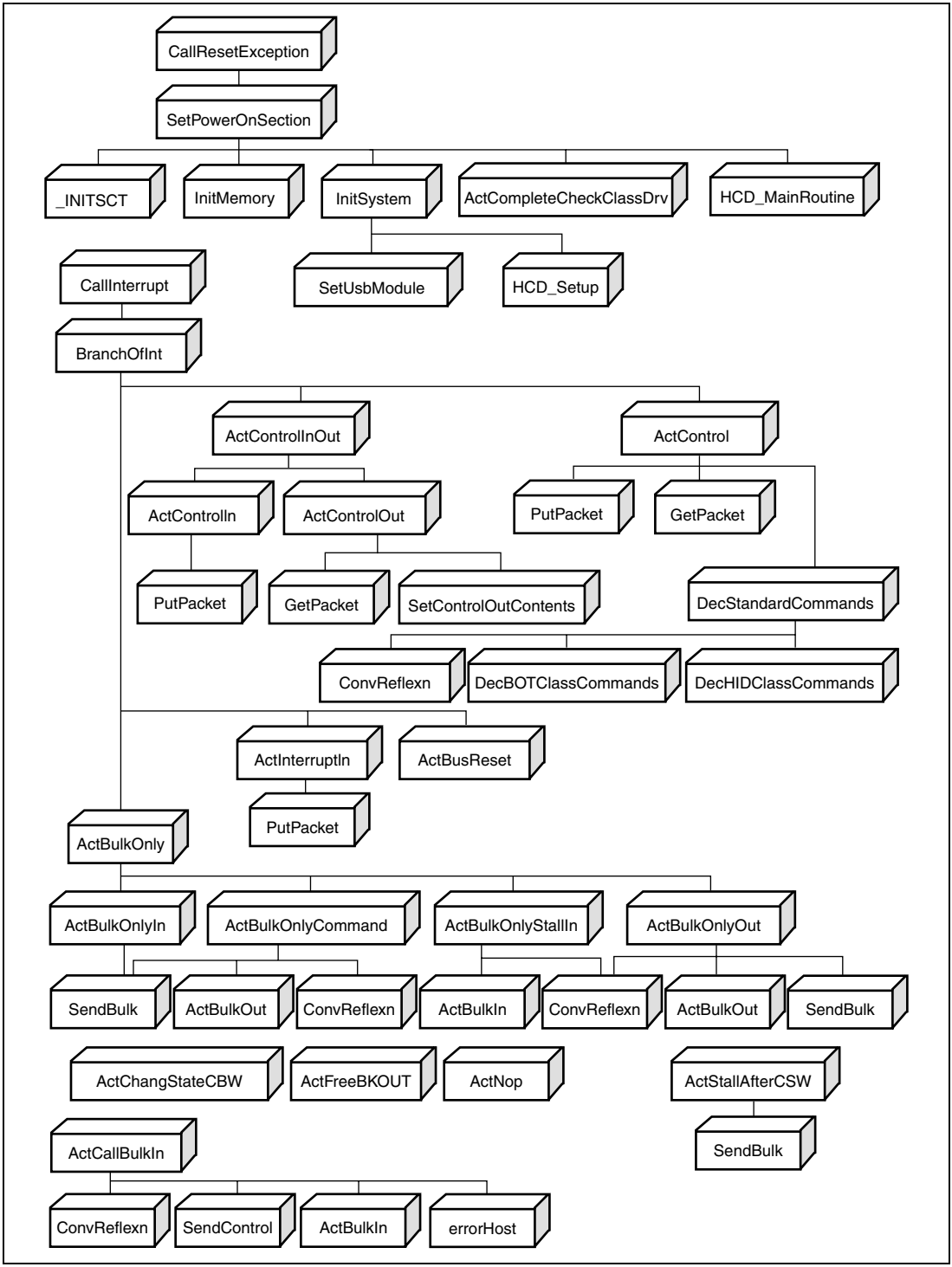

Figure 4.10 Interrelationship between Functions

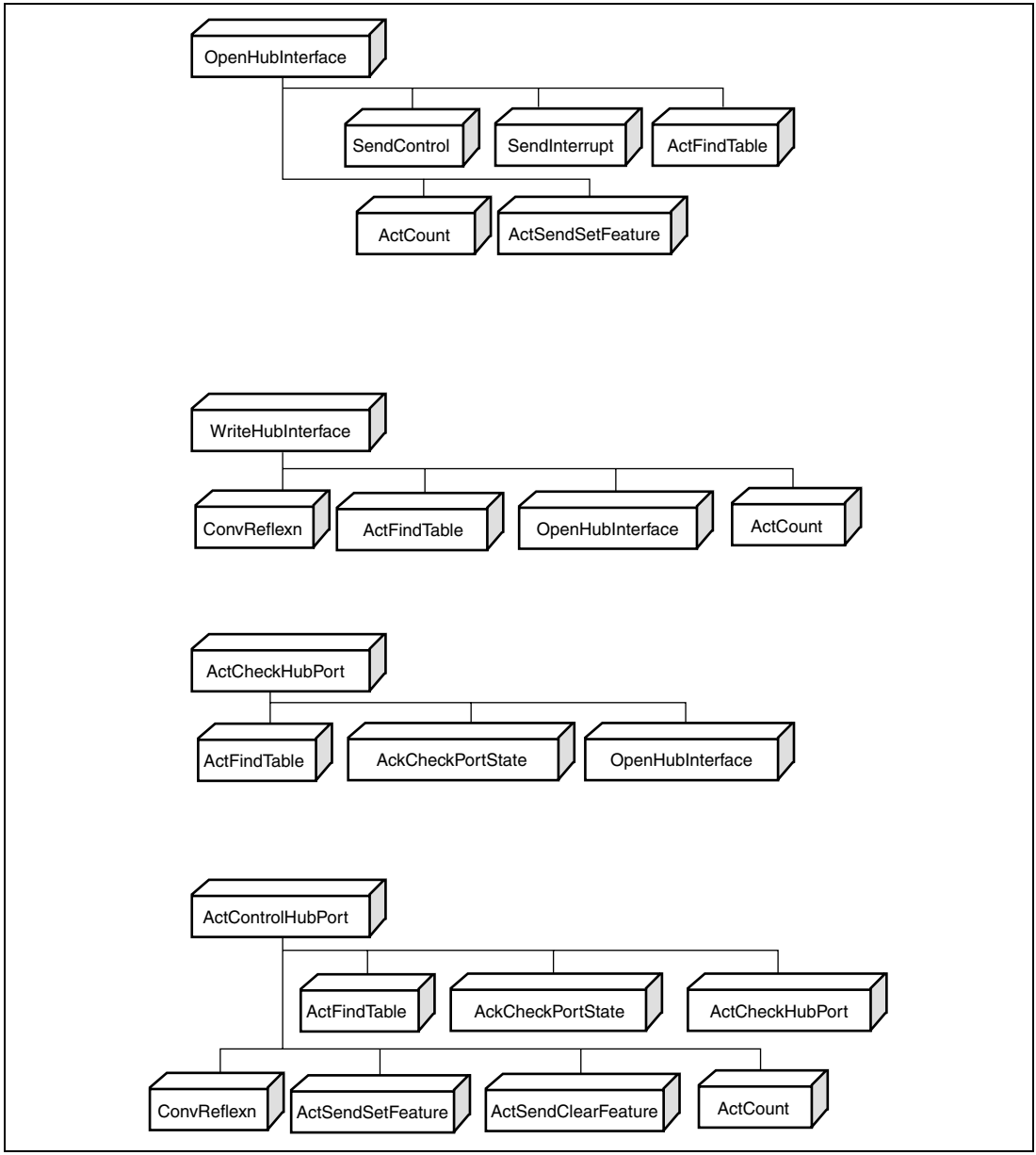

Figure 4.11 Interrelationship between Functions

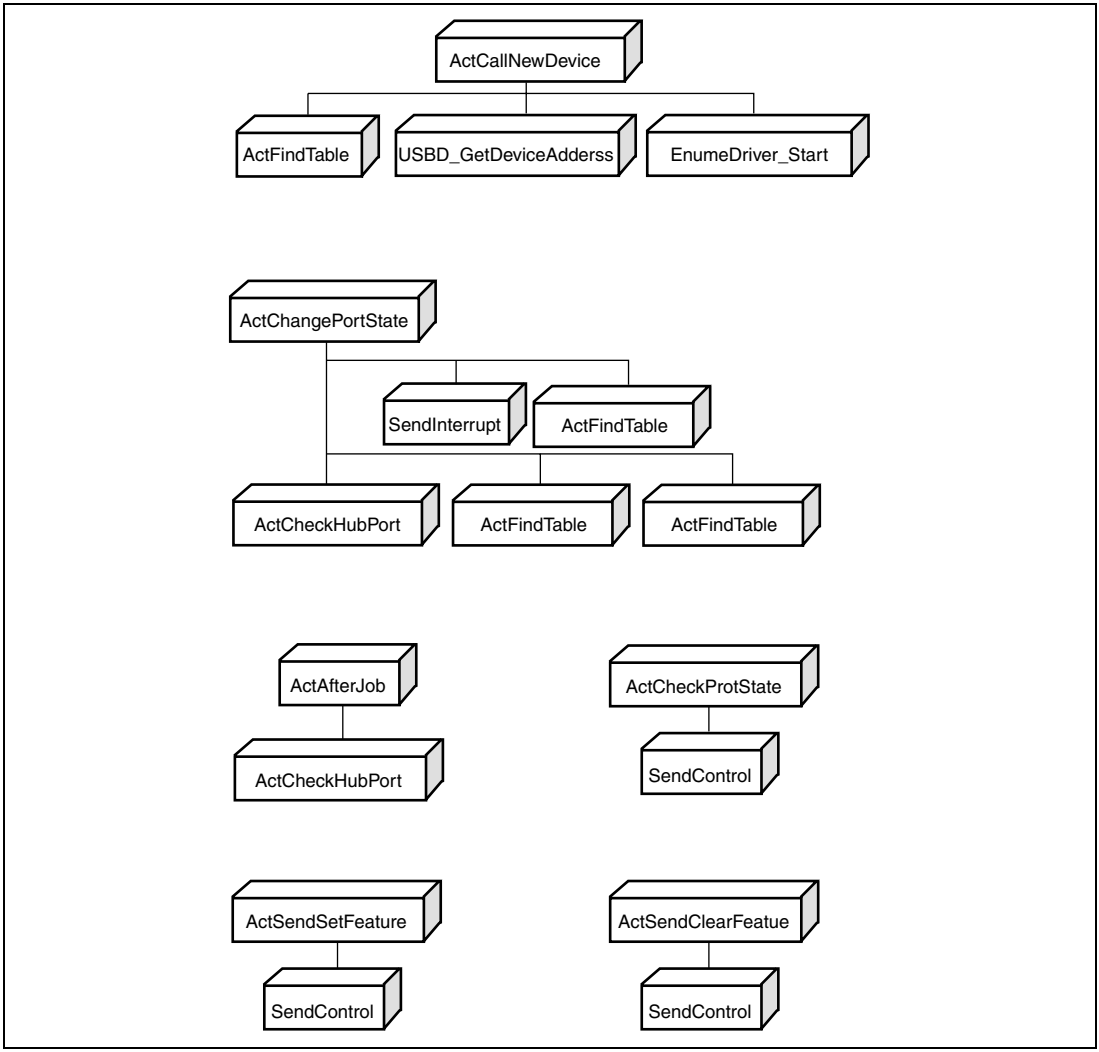

Figure 4.12 Interrelationship between Functions

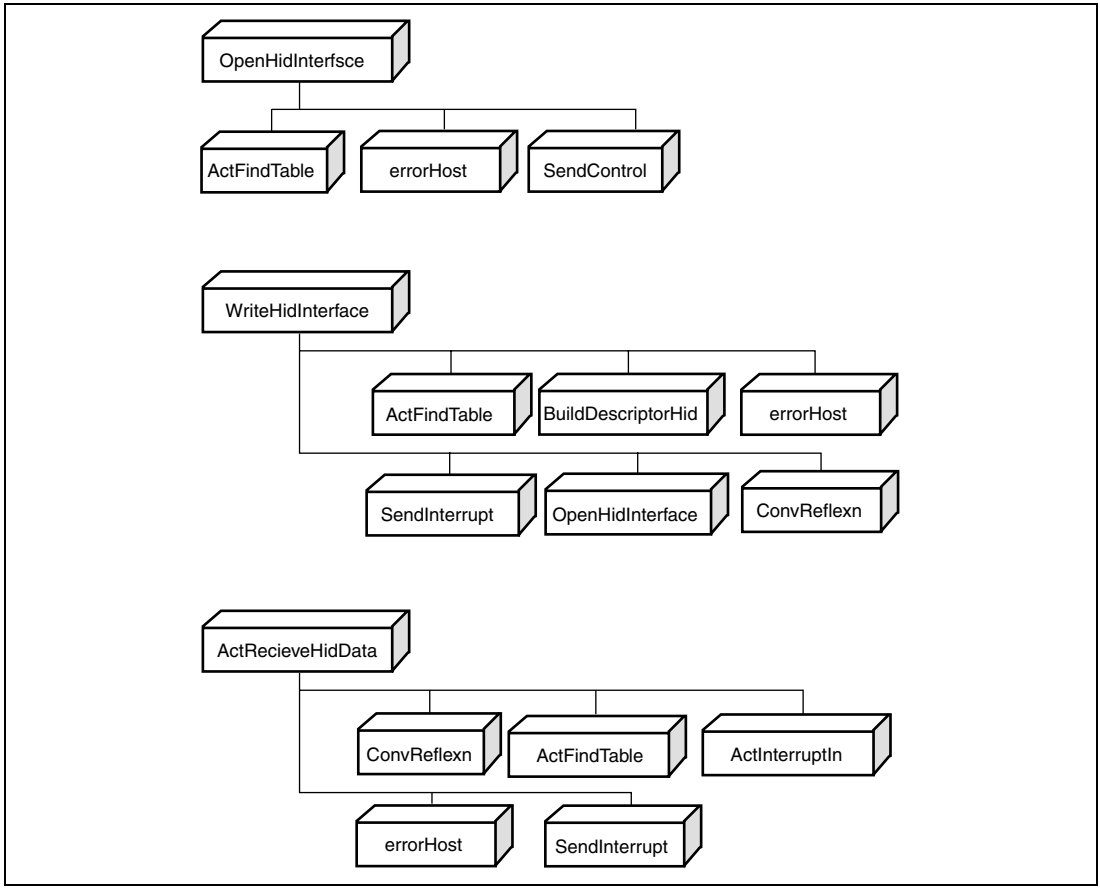

Figure 4.13 Interrelationship between Functions

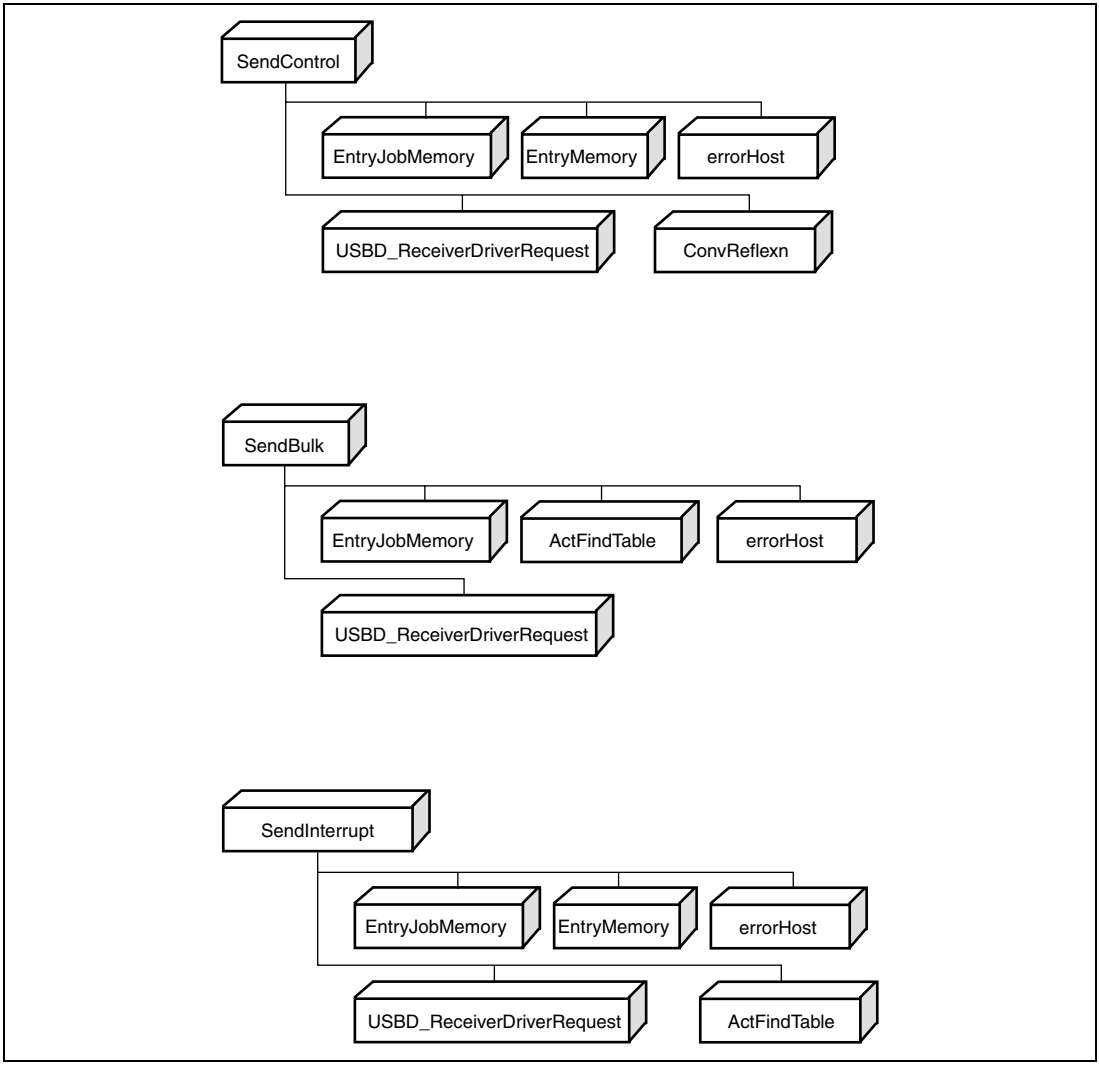

Figure 4.14 Interrelationship between Functions

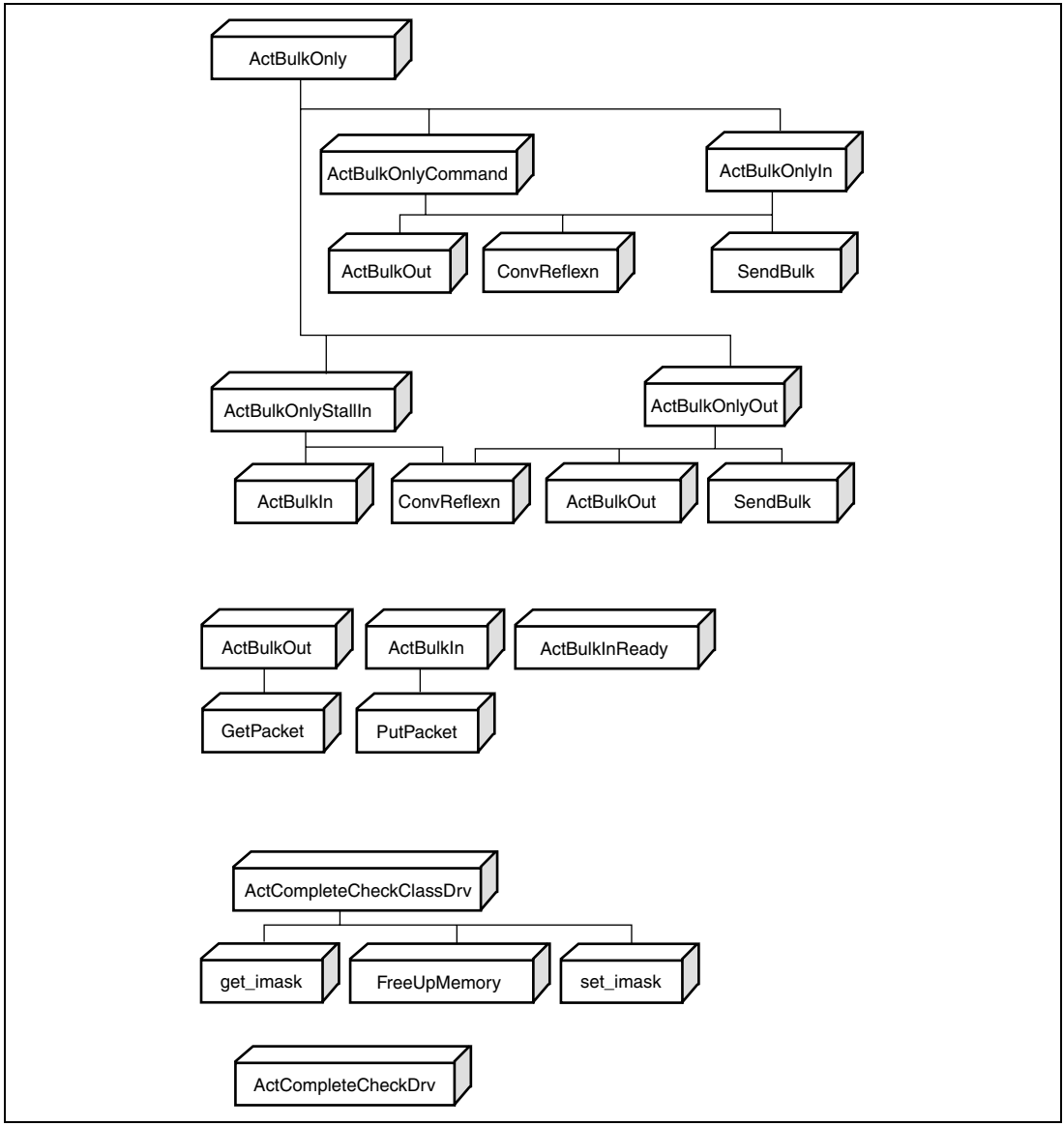

Figure 4.15 Interrelationship between Functions

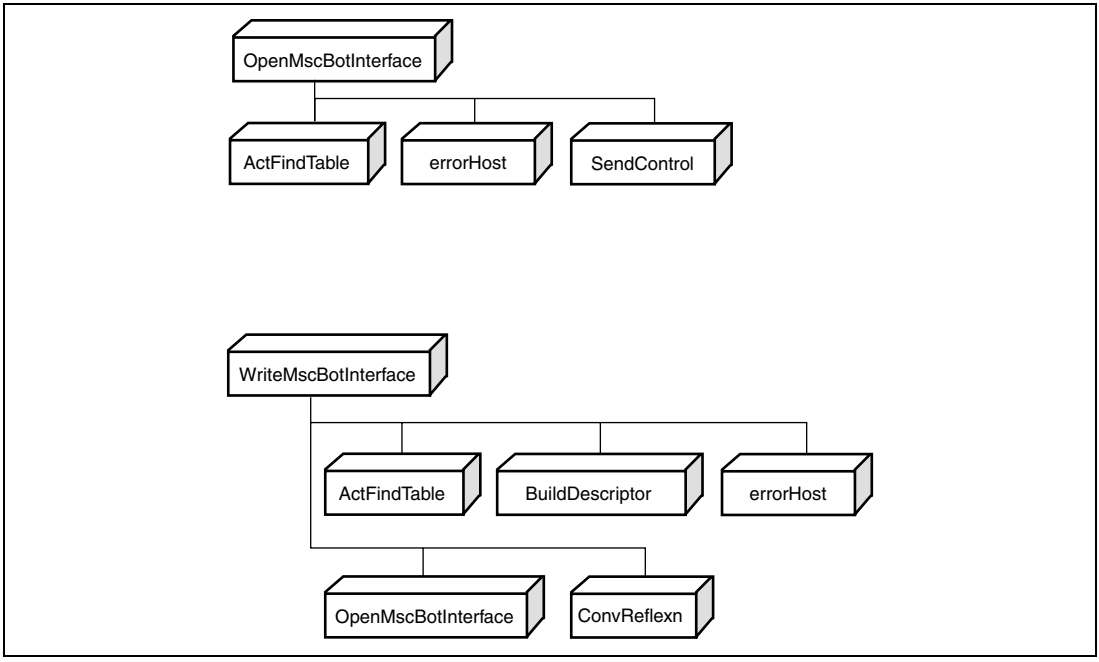

Figure 4.16 Interrelationship between Functions

4.6 Argument Types

In this sample program, other than such as char contained as a standard part of C-language, argument types depending on the function's purpose are defined and used. Types are defined in Usbh_Dr_CatHostTypedef.h.

The argument types for typical functions are listed below.

- The DriverRequestType argument type is for the USBD_ReceiveDriverRequest function which belongs to USBD in the USB system software layer.
- The ClassDriverRequestType argument type is for the SendBulk function which prepares bulk transfers and the SendInterrupt function which prepares interrupt transfers.
- The ClassDriverRequestType and SetupDataType argument types are for the SendControl function which prepares control transfers.

Defined function types and elements are shown below.

Table 4.3-1 DriverRequestType Structure Members

	Member Type	Member Name	Description
1	unsigned char	jobNum	Transfer control number
2	unsigned char	sendDeviceNum	Device number
3	unsigned char	sendEPsNum	Endpoint number
4	unsigned char*	Buffer	Pointer to data transfer area
5	unsigned long	BufferLength	Data transfer size
6	unsigned long	direction	Data transfer direction
7	unsigned long	Status	Transfer completed state
8	unsigned char	Setup[8]	Setup packet data
9	unsigned char	Ep_Type	Endpoint type
10	unsigned short	Ep_Mps	Maximum packet size of endpoint
11	unsigned char	Ep_Speed	Speed of endpoint
12	unsigned char	*Function (unsigned char)	Calling function on transfer completion

A class driver uses above structure as argument types to the USBD_ReceiveDriverRequest function in the USB system software layer.

Table 4.3-2 ClassDriverRequestType Structure Member

	Member Type	Member Name	Description
1	DeviceListType*	deviceList	Pointer to device information which are managed by the host drivers of each class
2	unsigned char	deviceAddress	Device address number of transfer destination device
3	unsigned char	interfaceNum	Interface number of transfer destination device
4	unsigned char	sendEPNum	Endpoint number of transfer destination device
5	unsigned char	direction	Data transfer direction
6	unsigned short	bufferLength	Data transfer size
7	unsigned char*	buffer	Pointer to the data transfer area
8	void	*CallOn	Calling function on transfer completion

Used as argument types to the functions SendControl, SendBulk, and SendInterrupt that generate transfers.

Table 4.3-3 DeviceListType Structure Member

	Member Type	Member Name	Description
1	unsigned char	dAddres	Device address number
2	unsigned char	dSpeed	Device speed
3	unsigned char	dReady	Device preparation
4	unsigned char	dIfNum	The number of interfaces contained in a device
5	IFListType*	dIfInfoPtr	Pointer to interface information of device

This structure is a structure of device information which host drivers for each class use to control devices.

This structure is also used as types to device information contained in the ClassDriverRequestType structure.

Table 4.3-4 IFListType Structure Member

	Member Type	Member Name	Description
1	unsigned char	ifConfigNum	Structure number
2	unsigned char	ifNum	Interface number
3	unsigned char	ifAltemateSetNum	Alternate setting number
4	unsigned char	ifHaveEpNum	The number of endpoints which can be handled by this interface
5	unsigned char	ifClass	Interface class number
6	unsigned char	ifSubClass	Subclass number
7	unsigned char	ifProtocol	Using protocol number
8	EPListType*	ifEpInfoPtr	Pointer to endpoint information which can be handled by this interface

This structure is the structure of interface information which host drivers for each class use to control devices.

This structure is also used as types to the interface information included in the DeviceListType structure.

Table 4.3-5 EPListType Structure Member

	Member Type	Member Name	Description
1	unsigned char	epAddress	Endpoint number and transfer direction
2	unsigned char	epAttributes	Transfer method
3	unsigned char	epMaxPacketSize	Maximum packet size
4	unsigned char	epInterval	Minimum access cycle to endpoint

This structure is a structure of endpoint information which host drivers for each class use to control devices.

This structure is also used as types to endpoint information included in the IFListType structure.

Table 4.3-6 SetupDataType Structure Member

	Member Type	Member Name	Description
1	unsigned char	ByteVal[0]	Stores BmRequest
2	unsigned char	ByteVal[1]	Stores bRequest
3	unsigned char	ByteVal[2]	Stores wValue
4	unsigned char	ByteVal[3]	Stores wValue
5	unsigned char	ByteVal[4]	Stores wIndex
6	unsigned char	ByteVal[5]	Stores wIndex
7	unsigned char	ByteVal[6]	Stores wLength
8	unsigned char	ByteVal[7]	Srores wLength

This structure is used as argument types to the SendControl function. This structure is used to handle 8-byte setup packet data which follows the setup to be transferred in the control transfer. The values of this structure should be stored in big endian format.

4.7 Multifunction

When multiple interfaces are equipped in this sample program, that is informed to the host. When the host received the information, it makes access to endpoints of this function on need.

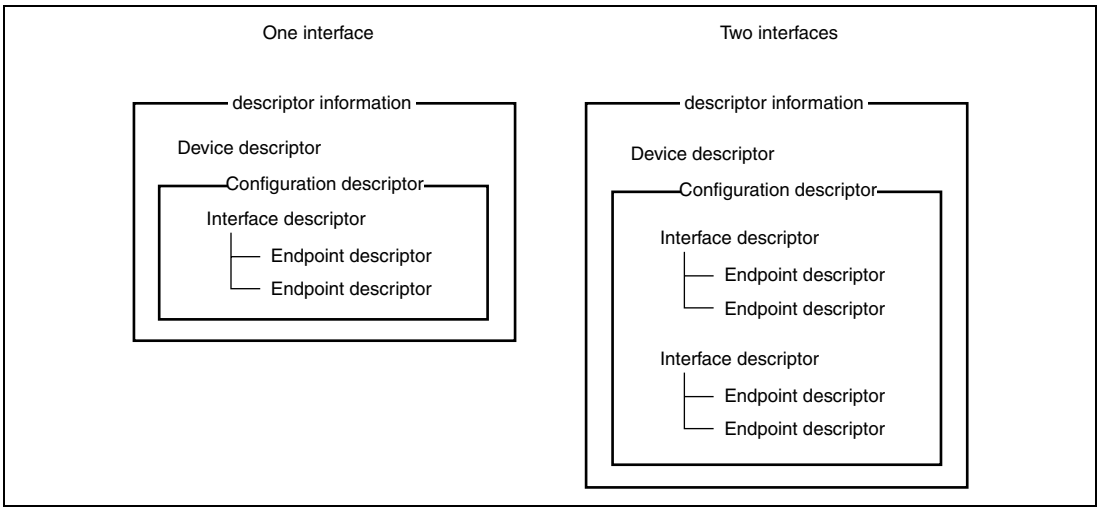

Figure 4.17 Descriptor Information Structure

4.7.1 Descriptor

Descriptor information is made on the basis of descriptor information of function devices (storage and HID class) that are connected to the SH7727 host. When a connected function device is a hub, descriptor information is not made.

When a function device other than a hub connection to the host is detected, the descriptor information with one interface is made. When the another connection is detected, descriptor information with two interfaces can be generated by adding interface descriptor information to existing descriptor information.

If a connected function device has been disconnected, created descriptor information is not deleted. Making new descriptor information is needed. Turn on a reset or NMI before connecting the SH7727SE to the host PC.

4.8 Device Driver

In this sample program, the USB hub and USB function devices are connected to the SH7727 host in the SH7727SE. When the connection is detected, the SH7727 host needs to control the USB hub and USB function devices by carrying out a data transfer. The control condition differs according to the USB class. In this sample program, three types of drivers, that is, hub driver, HID driver, and storage driver are included to control each class.

4.8.1 Hub Driver Operation

The hub driver carries out obtaining descriptor information, controlling a hub class device by using the obtained descriptor information, setting configuration, setting lower-ports, managing and supporting lower-ports.

As a restriction, the hub driver does not manage or control the current of downstream ports.

Note that maximum four hubs can be connected.

4.8.2 HID Driver Operation

The HID driver carries out obtaining descriptor information, controlling a HID class device by using the obtained descriptor information, setting configuration, and data reception from the HID device.

As a restriction, one device can be connected to the HID driver

4.8.3 Storage Driver Operation

The storage driver carries out obtaining descriptor information, controlling a storage device by using the obtained descriptor information, and setting configuration.

As a restriction, one storage driver can be connected.

4.9 Cooperation of Host and Function

In this sample program, the SH7727 host and the SH7727 function transfer data by cooperating each other and realize data transmission/reception between the host PC and USB function devices.

Connect the USB function device to the SH7727 host and the host PC to the SH7727 function. When the host PC detects the connection of the SH7727 function, it performs bus enumeration.

Then the host PC transfers data to each endpoints depending on descriptor information returned from the SH7727SE.

4.9.1 HID Class Cooperation

In the HID class, the SH7727 host and SH7727 function establish a ring buffer in the memory of the SH7727SE and perform sending and receiving data.

After configuration completes, the SH7727 host sends an IN token by the interrupt transfer to the endpoint returned by the HID device. If the HID device has data to send, it sends the data to the SH7727 host. If the HID has no data, it returns a NAK to the SH7727 host. After receiving data, the SH7727 host writes data to the ring buffer, and then resumes the interrupt transfer.

The host PC sends an IN token to the endpoint 3 of the SH7727 function. If there is transfer data in the ring buffer, the SH7727 function sends the data to the host PC. If there is no data in the ring buffer, the function sends a NAK to the host PC.

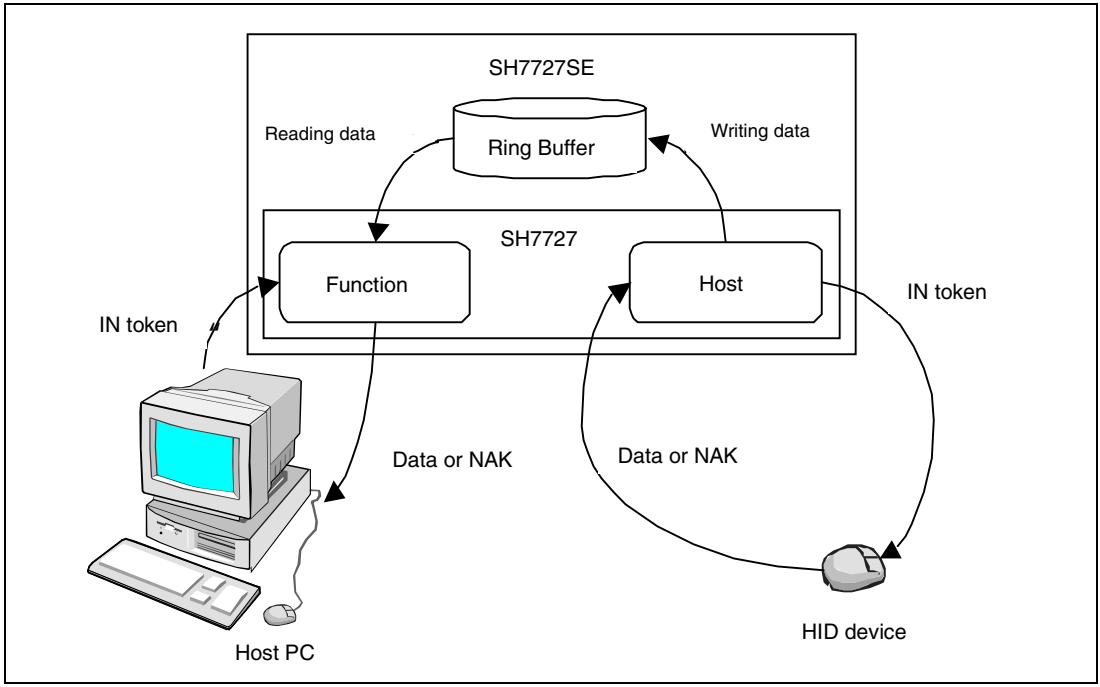

Figure 4.18 HID Class Cooperation Diagram

4.9.2 Storage Class Cooperation

In the storage class, the SH7727 host and SH7727 function establish a transfer buffer in the memory of the SH7727SE, and perform sending and receiving data. Sending/receiving data is performed in 1 packet unit. According to the transfer procedure of the Bulk-Only Transport in USB Mass Storage Class, a processing flow in the storage class is explained below.

When the SH7727 function receives CBW which is issued by the host PC, it occurs a receive interrupt and activates the USB receive interrupt routine. The interrupt routine performs following two processes sequentially.

1. CBW is sent from the SH7727 function to the transfer buffer.
2. CBW is transferred from the transfer buffer to the storage device by the SH7727 host.

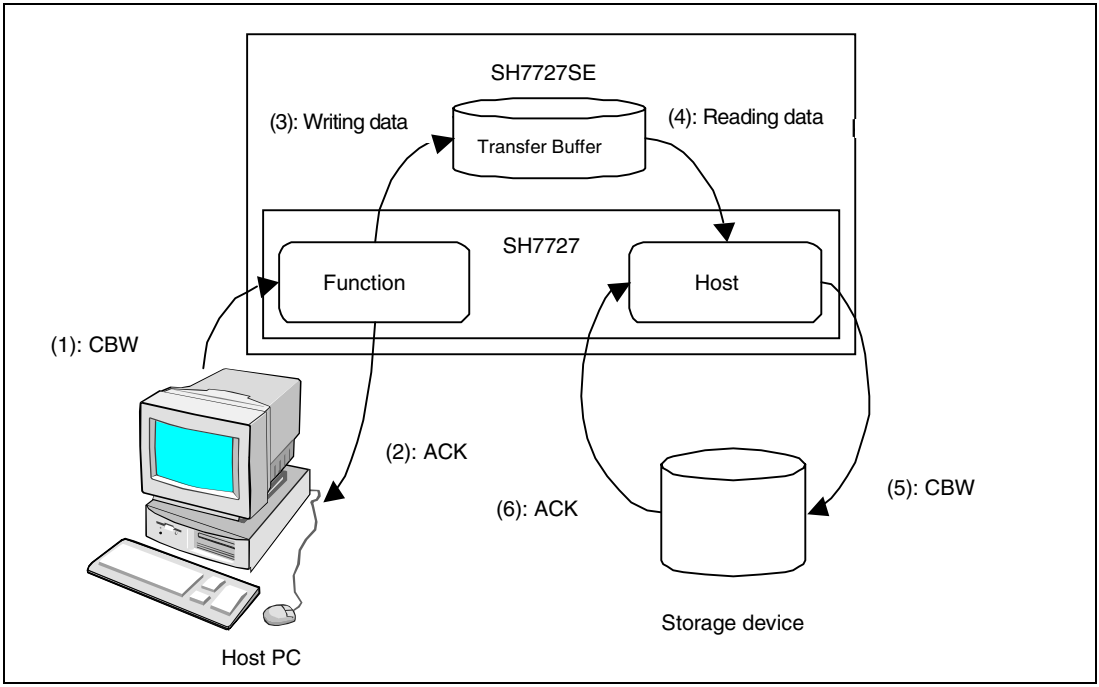

Figure 4.19-1 Storage Class Cooperation Diagram

When CBW transfer has been completed, processing data transport or CSW is begun.

When data transport is the in transfer, the host PC issues an IN token to the SH7727 function to require data. However, as the SH7727 function has no data to send to the host PC, it sends back a NAK to the host PC. At this time, an interrupt is occurred and process is done by the interrupt processing routine. Three procedures are explained below.

1. The SH7727 host transfers data from the storage device to the transfer buffer to store it in the transfer buffer.
2. The SH7727 function transfers data which is stored in the transfer buffer to the host PC.
3. The interrupt enable bit is modified.

In the SH7727 function, there are two FIFO buffer that are used for the data transfer (bulk-IN). When the buffer leaves space, an interrupt occurs and the interrupt is processed sequentially by the interrupt processing routine. The interrupt continues until CSW is transmitted.

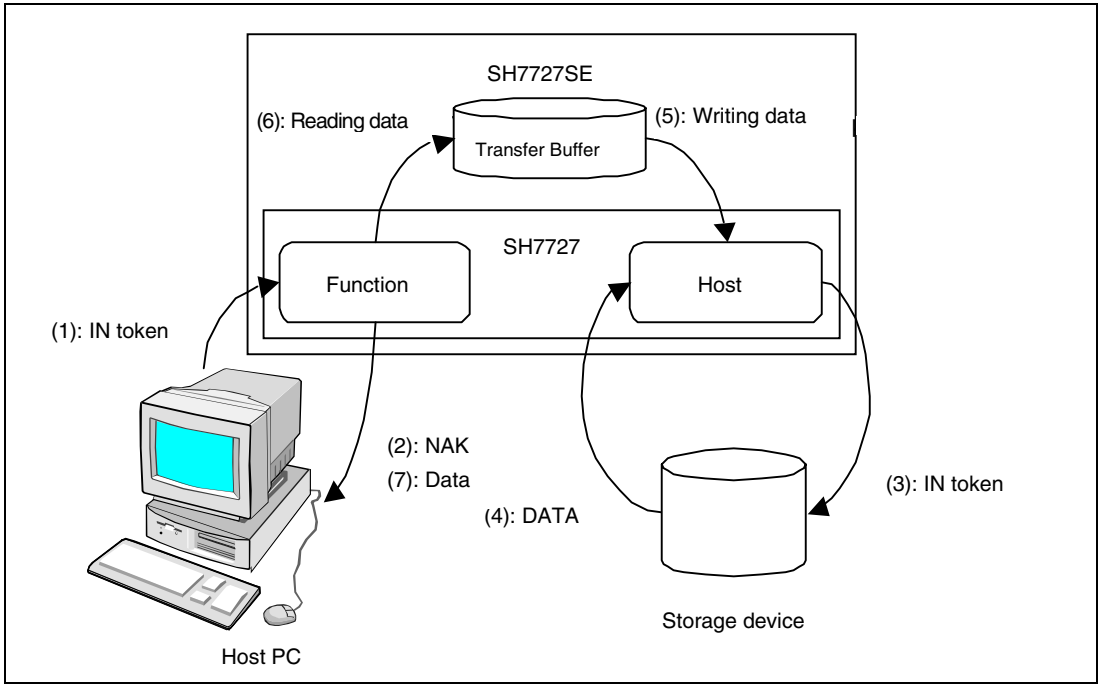

Figure 4.19-2 Storage Class Cooperation Diagram

When the data transport is out transfer, the host PC issues an OUT token and a following transfer data to the SH7727 function. As there is no receive data in the SH7727 function, the SH7727 function receives the data from the host PC and sends back an ACK to the host PC. At this time, as an interrupt occurs, following two procedures are processed sequentially by the interrupt processing routine.

1. Receive data is sent from the SH7727 function to the transfer buffer and stored in the buffer.
2. Data which is stored in the transfer buffer is transferred to the storage device by the SH7727 host.

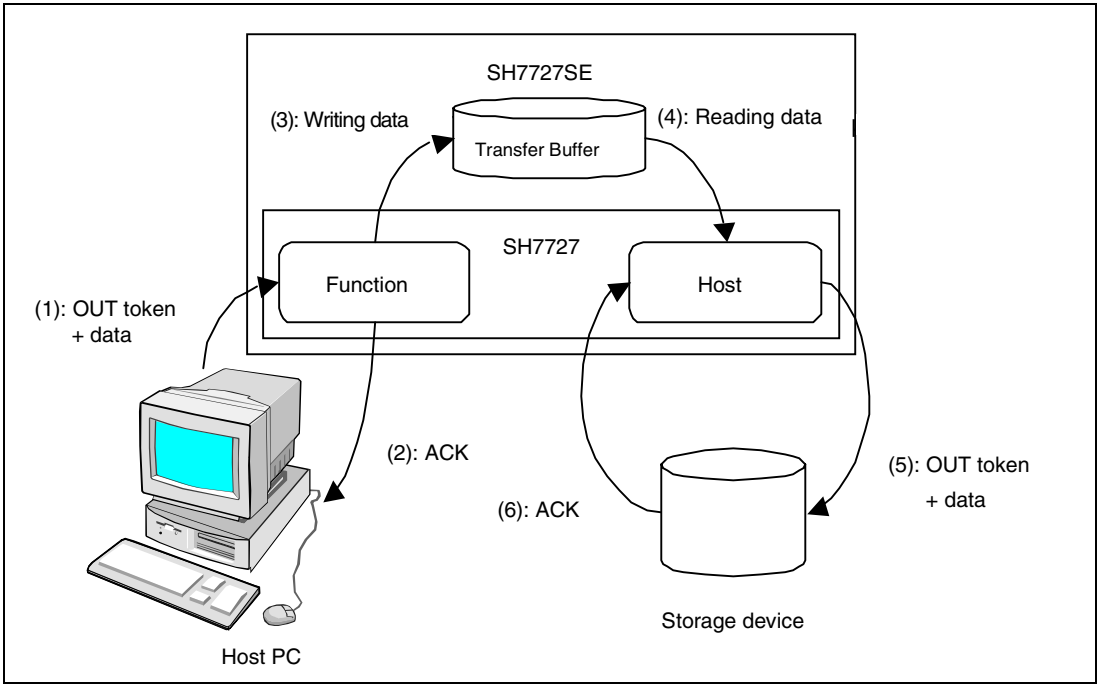

Figure 4.19-3 Storage Class Cooperation Diagram

When there is no transport or the OUT direction data transport has been completed, the host PC issues an IN token to the SH7727 function and requires CSW. However, as the SH7727 function has no transfer data to the host PC, the SH7727 function sends back a NAK to the host PC. At this time, as an interrupt occurs, following two procedures are processed sequentially by interrupt processing routine.

1. CSW is sent from the function to the transfer buffer and stored in the buffer by the SH7727 host.
2. CSW which is in the transfer buffer is transferred to the host PC by the SH7727 function.

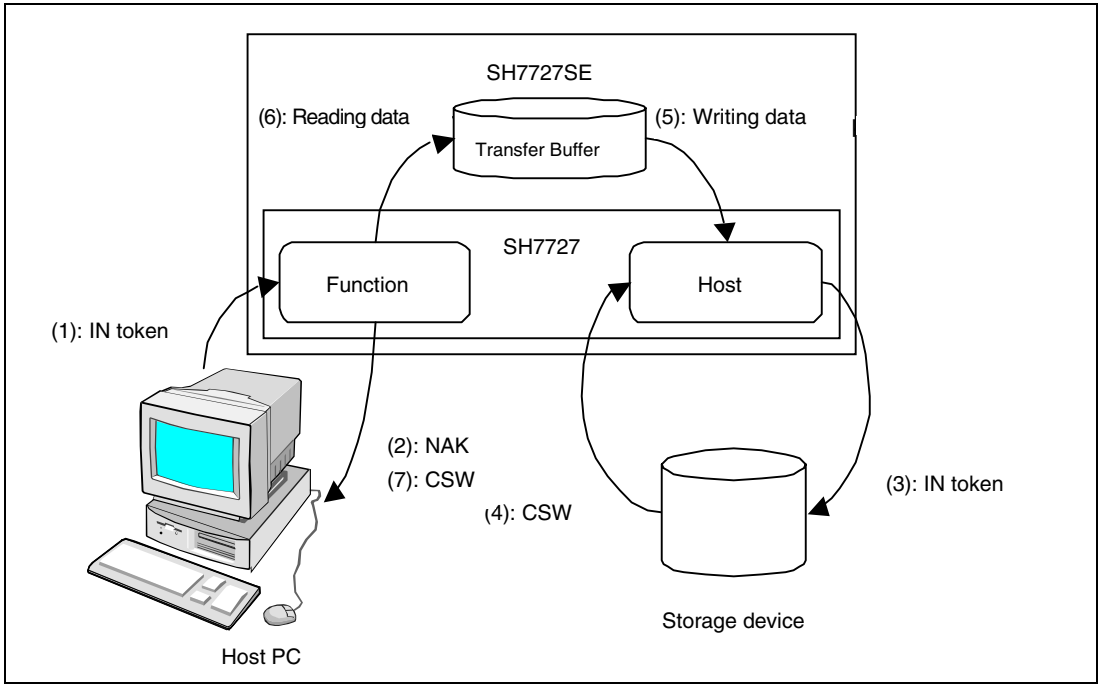

Figure 4.19-4 Storage Class Cooperation Diagram

When the data transport is an IN transfer and the SH7727 host requests the storage device transfer data and the storage device sends back a STALL, individual process is done to the host PC and storage device.

For the host PC, following two processes are done.

1. A 0x00 data with bytes as the same number of the data required in the CBW is transferred to the host PC.
2. CSW is made based on CBW and sent to the host PC with status code 0x01.

In the SH7727 function, there are two FIFO buffer that are used for the data transfer (bulk-IN). When the buffer leaves space, an interrupt occurs and the interrupt is processed sequentially by the interrupt processing routine. The interrupt continues until CSW is transmitted.

For the storage device, following two processes are done.

1. Clear Feature is issued to clear stall state..
2. The host issues an IN token to the storage device and receives CSW from the device.

The SH7727 host should control the storage device and prepare for the next CBW.

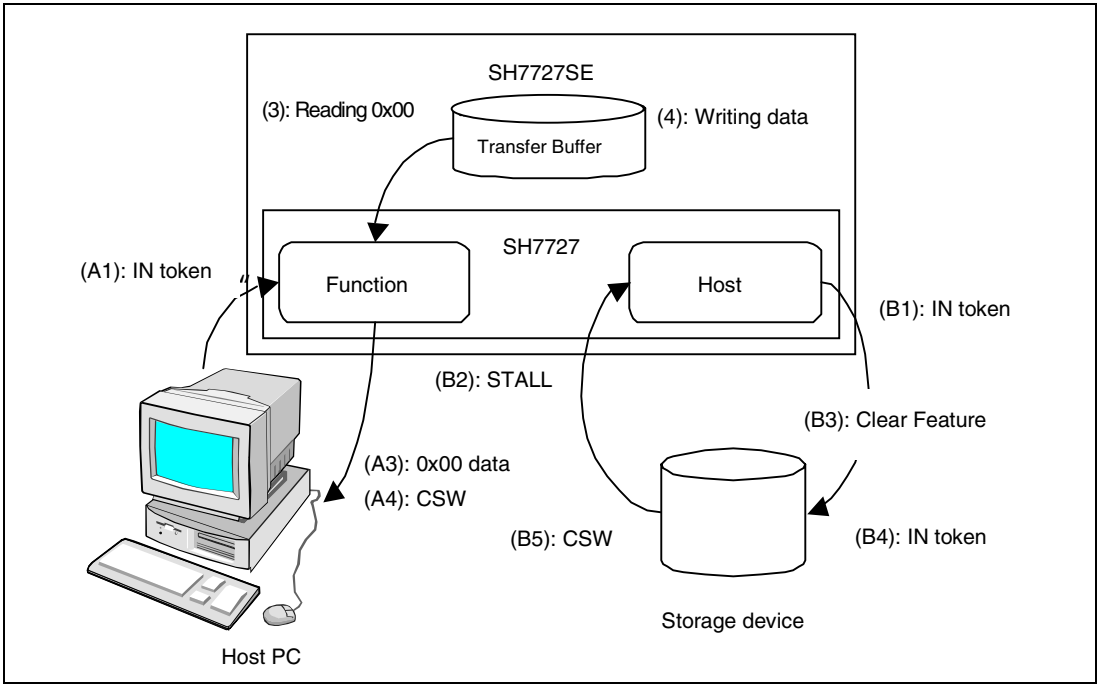

Figure 4.19-5 Storage Class Cooperation Diagram

Section 5 Sample Program Operation

In this section, the operation of the sample program is explained, relating it to the operation of the USB function and USB host modules in the SH7727.

5.1 Main Loop

When the microcomputer is in the reset state, the internal state of the CPU and the registers of on-chip peripheral modules are initialized. Next, the reset interrupt function CallResetException is called, and the reset exception handling is executed, then the function SetPowerOnSection is called. Figure 5.1 is a flow chart from the interrupt occurrence to the steady state.

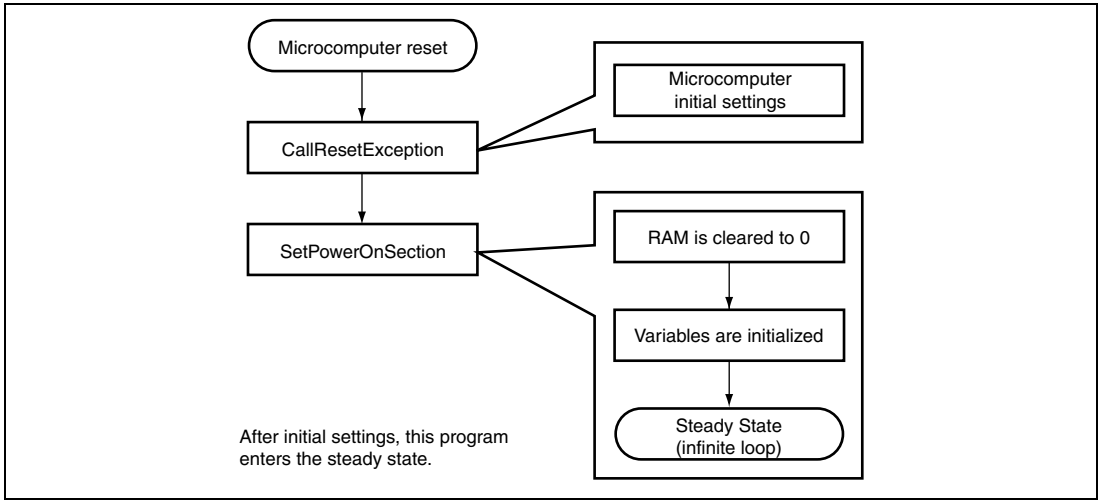

Figure 5.1 Main Loop

5.2 Types of Interrupts

The interrupts used in this sample program are indicated by the interrupt flag registers 0 and 1 (USBIFR0 and USBIFR1), and the HcInterruptStatus register; there are ten types of interrupts.

When a USB interrupt occurs, the corresponding bit in the interrupt flag register is set to 1 and a USBFI0 interrupt request is sent to the CPU. In the sample program, when the interrupt occurs, the CPU reads the interrupt flag register to perform the corresponding USB communications. Figure 5.2 shows correspondence between the interrupt flag registers and USB communications.

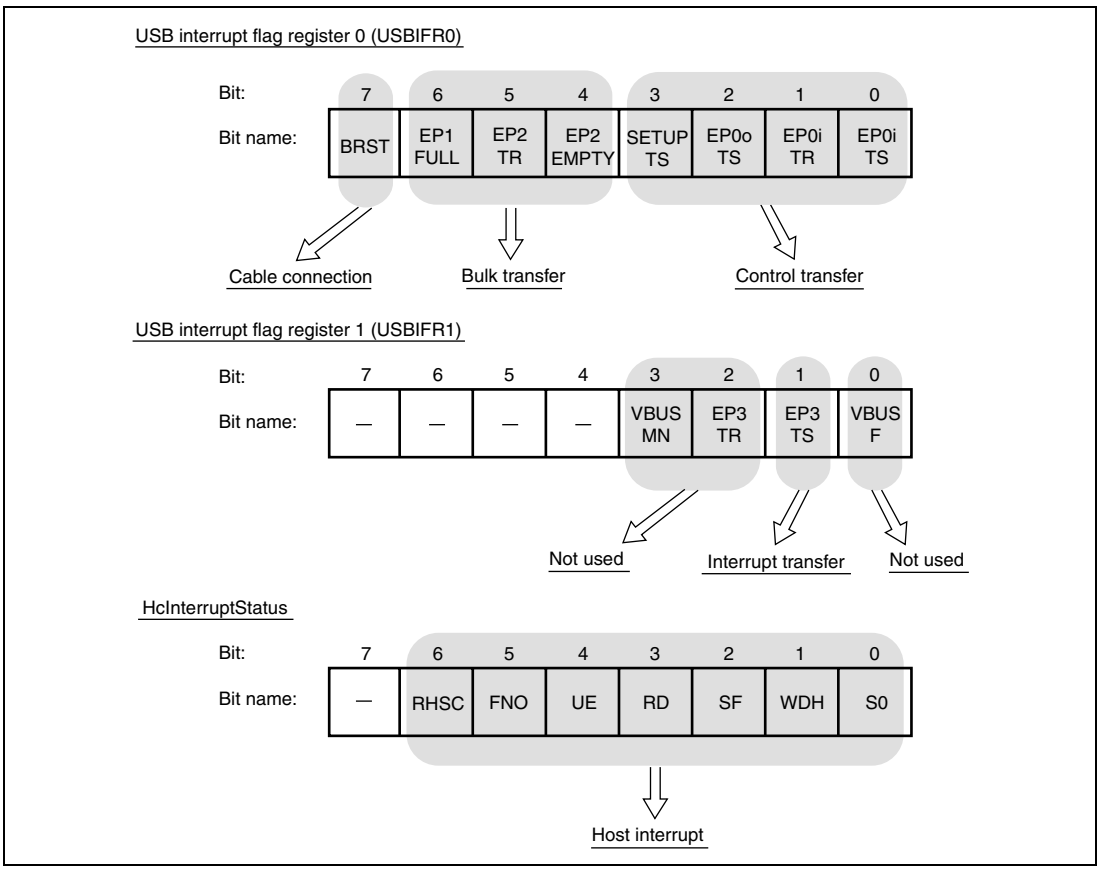

Figure 5.2 Types of Interrupt Flags

5.2.1 Branching to Transfer Function

In this sample program, the transfer method is decided by the type of interrupt from the USB module. Branching according to the decided transfer method is executed by BranchOfInt of UsbMain.c. Table 5.1 shows the relationship between interrupt types and functions called by BranchOfInt.

Table 5.1 Interrupt Types and Called Functions

Register Name	Bit	Bit name	Called Function
USBIFR0	0	EP0i TS	ActControlInOut
	1	EP0i TR	ActControlInOut
	2	EP0o TS	ActControlInOut
	3	SETUP TS	ActControl
	4	EP2 EMPTY	ActBulkIn
	5	EP2 TR	ActBulkInReady
	6	EP1 FULL	ActBulkOut
	7	BRST	ActBusReset
USBIFR1	1	EP3 TS	ActInterruptIn

The EP0i and EP0o TS interrupts are used for both control-IN and control-OUT transfers. Therefore, to manage the direction and stage of control transfer, the sample program has three states: TRANS_IN, TRANS_OUT, and WAIT. For details, see section 5.4, SH7727 Function Control Transfer.

5.3 Interrupt on Cable Connection (BRST)

These interrupts occur when the USB function module is connected to the host PC by using the USB cable. After completion of initializing the microcomputer, the software pulls up the USB data bus D+ by using a general output-only port. By means of this pull-up, the host PC recognizes that the device has been connected (figure 5.3)

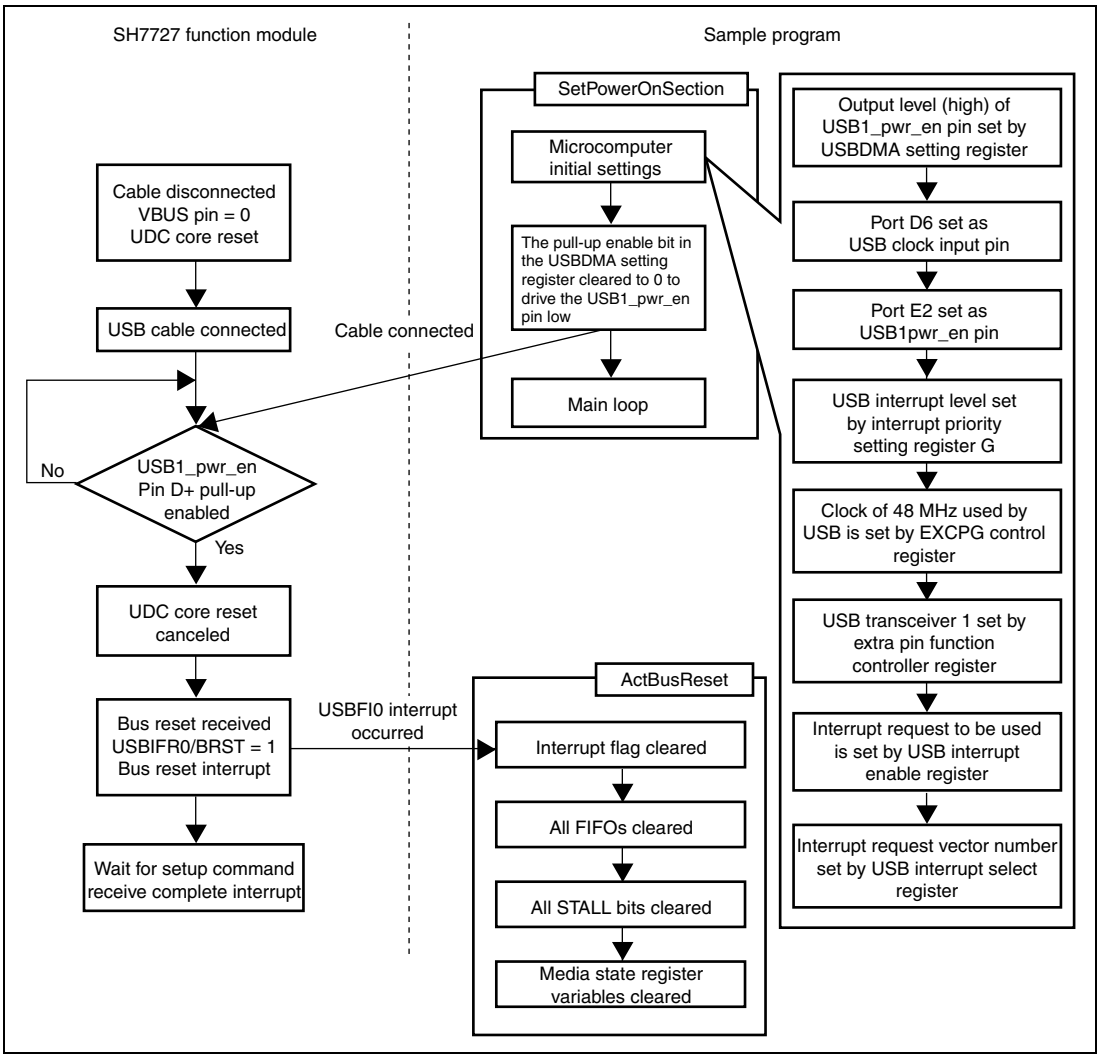

Figure 5.3 Interrupt on Cable Connection

5.4 SH7727 Function Control Transfer

For control transfers, bits 0 to 3 in the interrupt flag registers are used. Control transfers are divided into two types according to the direction of data in the Data stage (figure 5.4). In the Data stage, data transfer from the host PC to the SH7727 function is control-OUT transfer and transfer in the opposite direction is control-IN transfer.

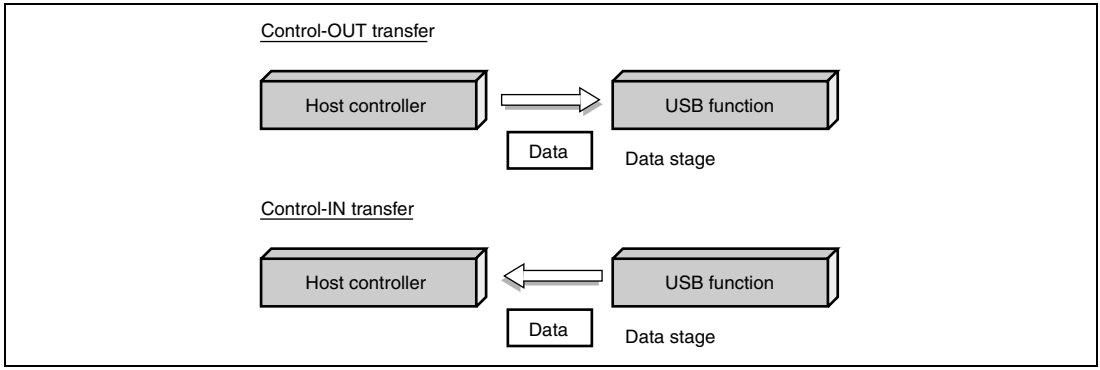

Figure 5.4 Control Transfer

Control transfers consist of three stages: Setup, Data (no data is possible), and Status (figure 5.5). 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 (see 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 5.5) 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.

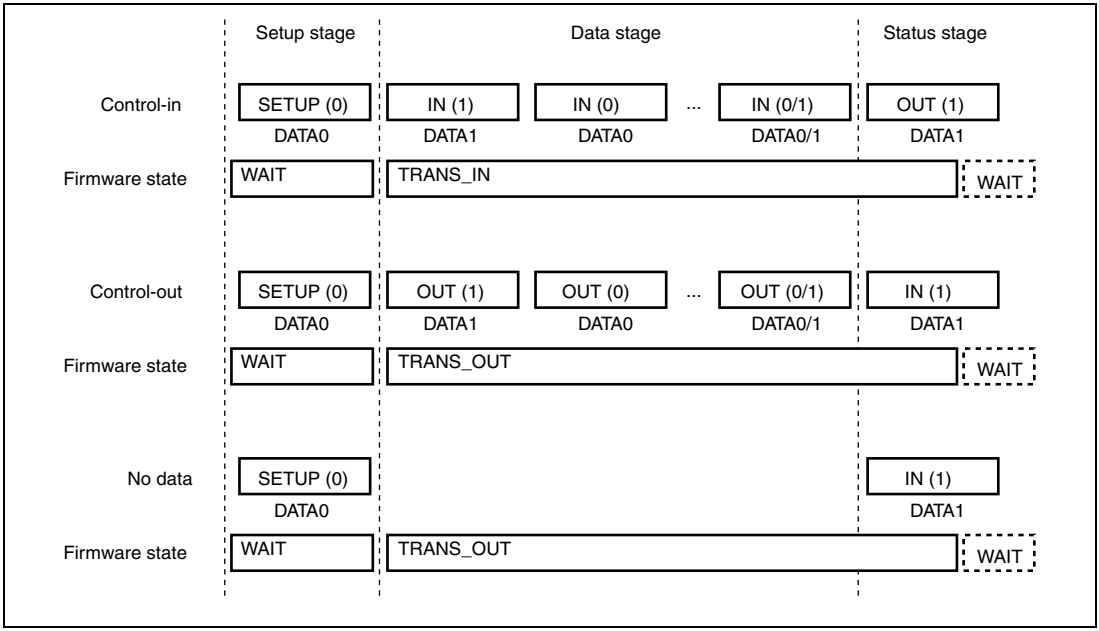

Figure 5.5 Each Stage in Control Transfers

5.4.1 Setup Stage

In the Setup stage, the host PC and SH7727 function exchange commands. For both control-IN and control-OUT transfers, the firmware goes into the WAIT state. Depending on the type of command issued, distinction between control-IN and control-OUT transfers is performed, and the state of the firmware in the Data stage (TRANS_IN or TRANS_OUT) is decided.

Command for TRANS_IN GetDescriptor (Standard command)

Figure 5.6-1 shows operation of the sample program in the Setup stage. The figure on the left shows the operation of the USB function module.

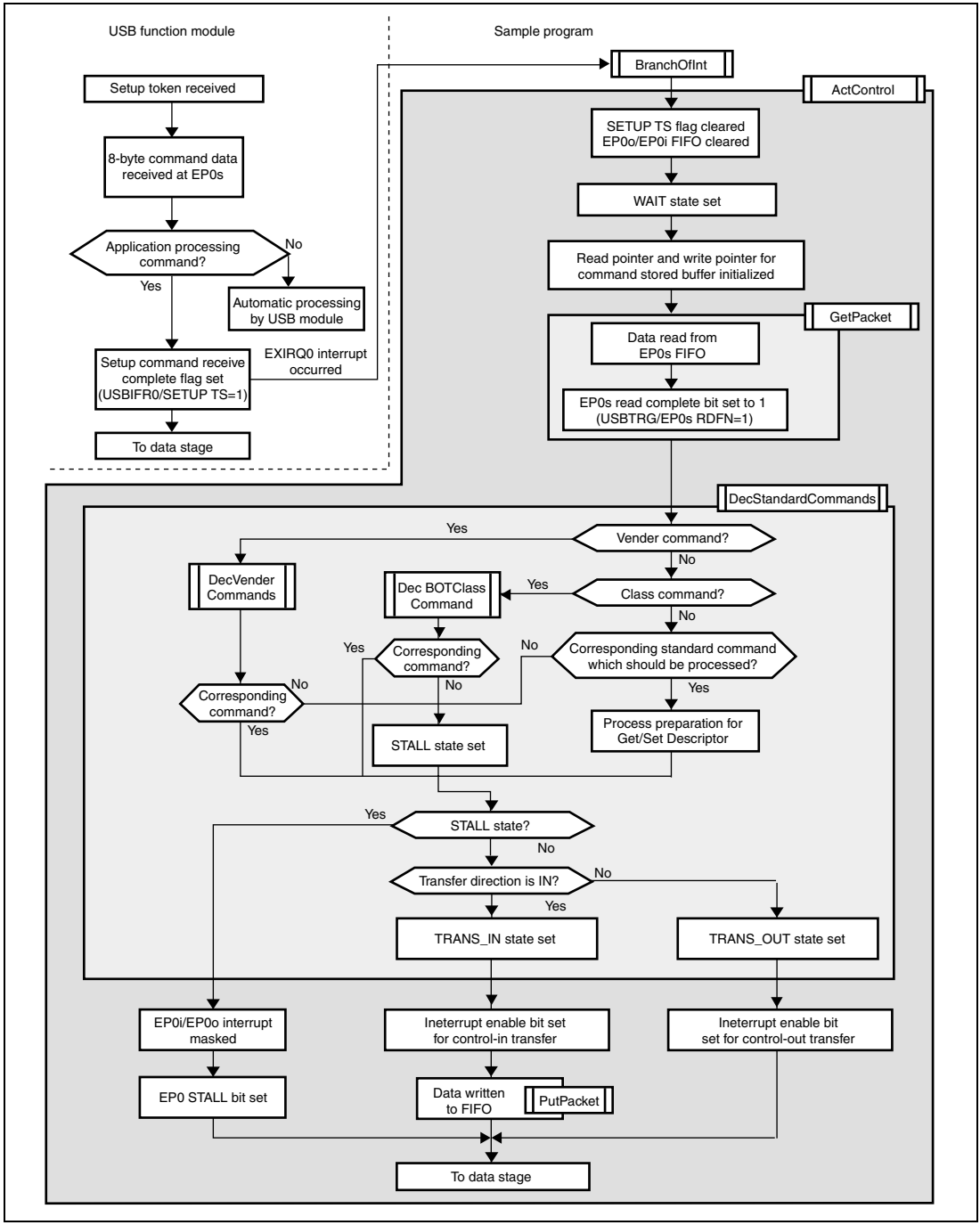

Figure 5.6-1 Setup Stage

5.4.2 Data Stage

In the Data stage, the host PC and SH7727 function exchange data. The firmware state becomes TRANS_IN for control-IN transfer, and TRANS_OUT for control-OUT transfer, according to the result of decoding of the command in the Setup stage. Figures 5.6-2 and 5.6-3 show the operation of the sample program in the Data stage of control transfer

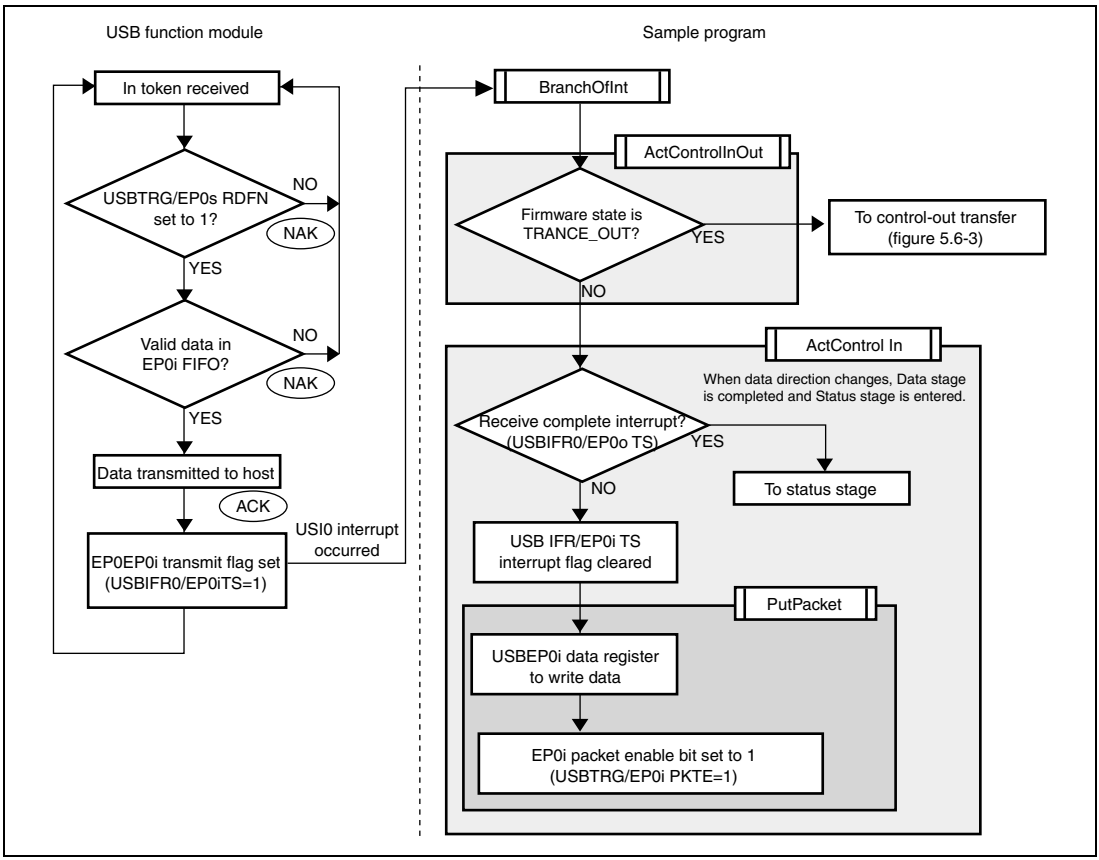

Figure 5.6-2 Data Stage (Control-IN Transfer)

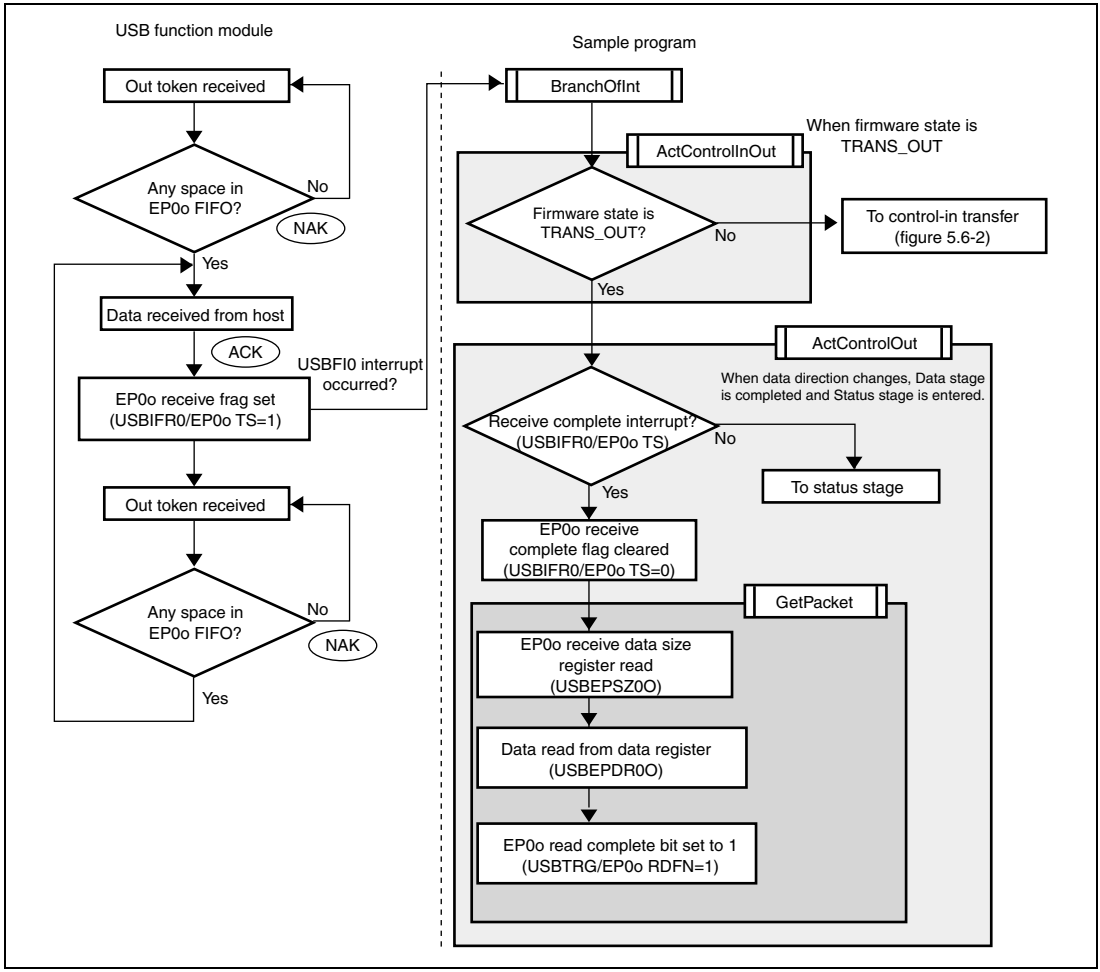

Figure 5.6-3 Data Stage (Control-OUT Transfer)

5.4.3 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 PC; in control-OUT transfer, it begins with an in-token from the host PC.

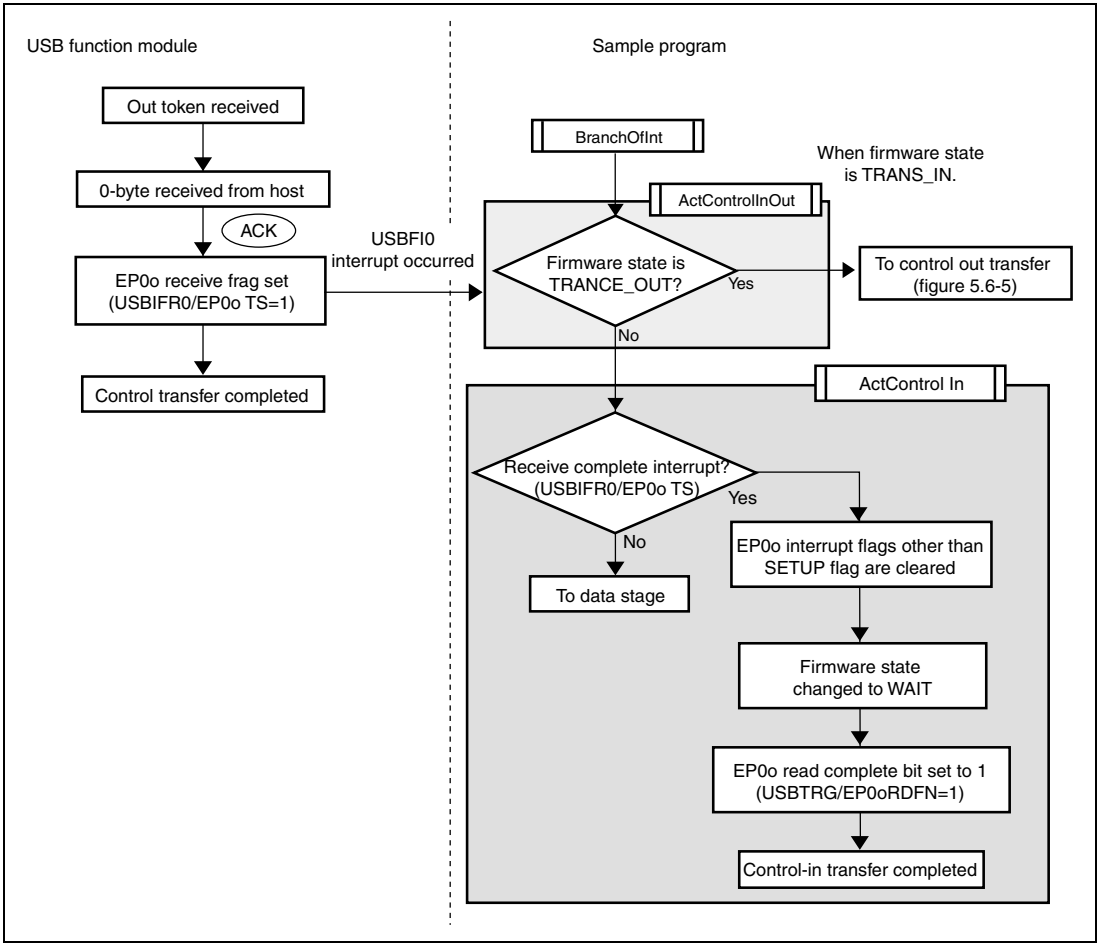

Figure 5.6-4 Status Stage (Control-IN Transfer)

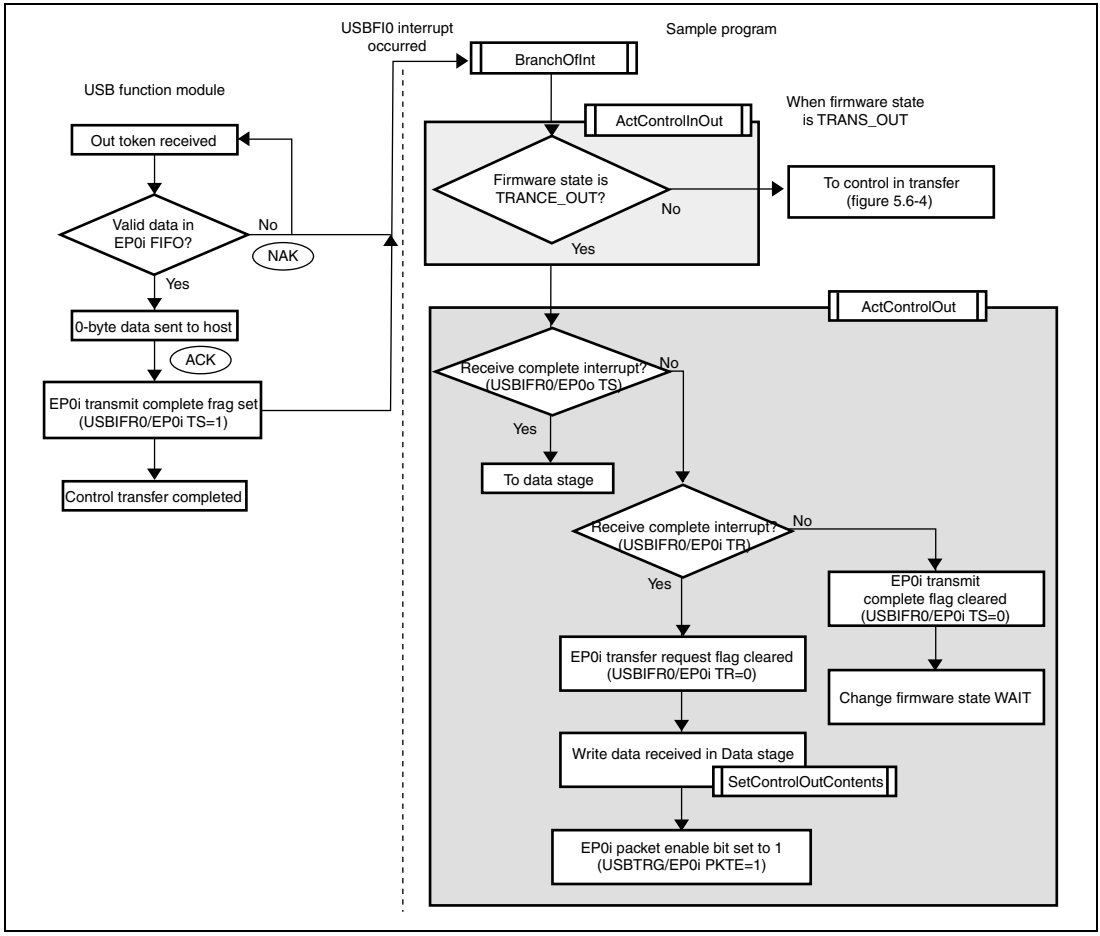

Figure 5.6-5 Status Stage (Control-OUT Transfer)

5.5 Bulk Transfer

In bulk transfers, bits 4 to 6 of the interrupt flag register are used. Bulk transfers can also be divided into two types according to the direction of data transmission. (Figure 5.7)

When data is transferred from the host PC to the SH7727 function, the transfer is called a bulk-OUT transfer; when data is transferred in the opposite direction, it is a bulk-IN transfer.

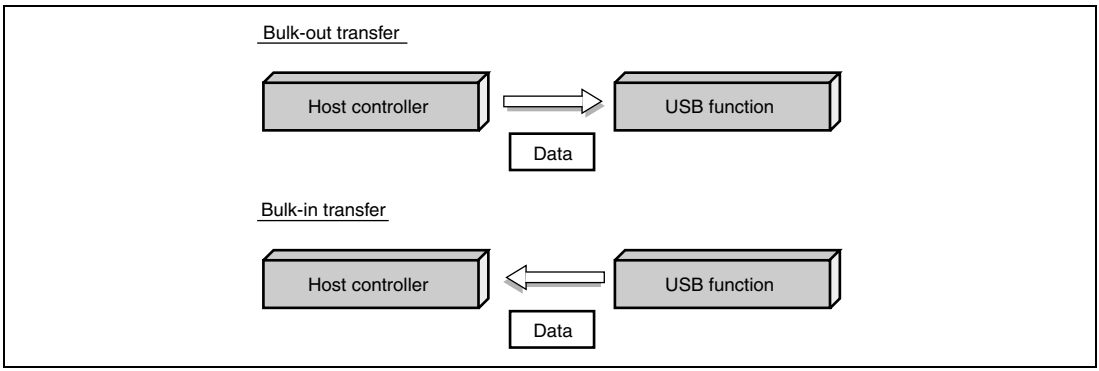

Figure 5.7 Bulk Transfer

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

Bulk-Only Transport comprises two or three stages (figure 5.8): command transport (CBW), data transport (this is sometimes not included), and status transport (CSW). In addition, data transport 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 sent 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 5.8). The appropriate variables must be loaded 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.

In this sample program, the CBW and data transport received from the host PC by using a bulk-OUT transfers and retained in the common memory. In a bulk-IN transfer, when the CSW and data transport to transfer are in the common memory, they are sent to the host PC. When the CSW

and data transport are not in the common memory, the USB function module (hardware) sends a NAK to the host PC.

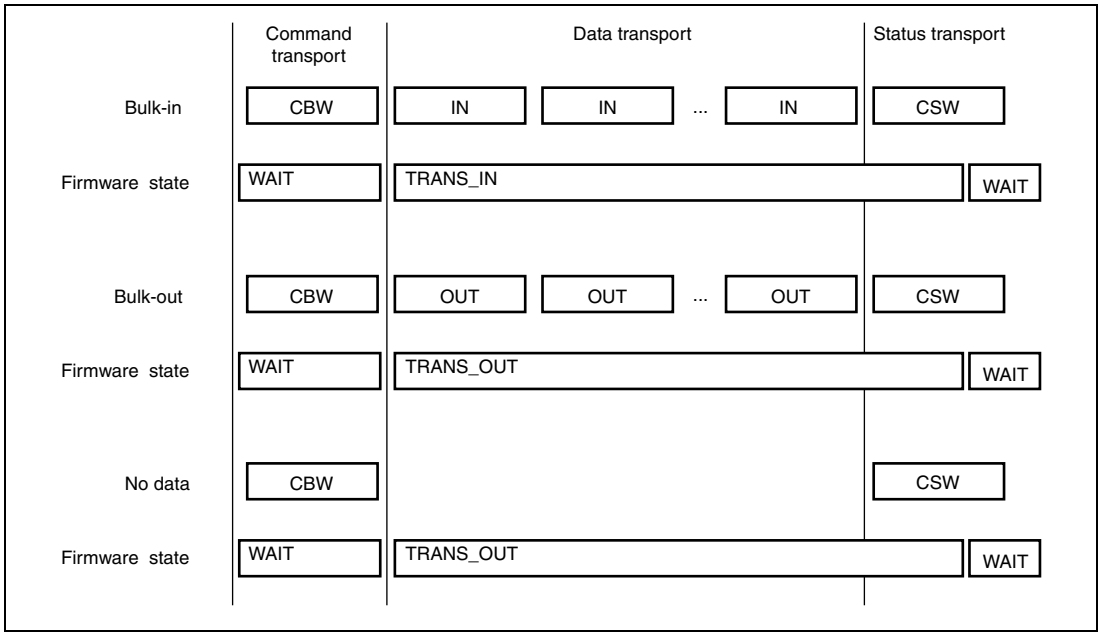

Figure 5.8 Each Stage in Bulk-Only Transport

5.5.1 Bulk-Out Transfer

In bulk-OUT transfer, data is transferred from the host PC to the SH7727 function. Bulk-OUT transfer is used in two cases: data reception and data transport reception.

When the CBW data is received, the state of firmware is WAIT. When data transport is received, the state of firmware is TRANS_OUT.

5.5.2 Bulk-In Transfer

In bulk-IN transfer, transfer data to the host PC is sent from the SH7727 function. Bulk-IN transfer is used in three cases: data transport transmission, CSW transmission, and when the link program transmits data of 0x00 with the same number of data requested in the CBW, to the host PC substitutes for the storage device.

In the data transport transmission stage, state of the firmware is TRANS_IN. In the CSW data transmission, the state of firmware is TRANS_IN or TRANS_OUT. When the link program transmits data of 0x00 substitutes for the storage device, the state of firmware is STALL.

5.6 Interrupt Transfer

For interrupt transfers, bit 1 in the interrupt flag register1 (USBIFR1) is used. In the SH7727, data transfer in interrupt transfer is executed in one direction (figure 5.9).

In the direction, data is transferred from the SH7727 function to the host PC; that is, the interrupt-IN transfer.

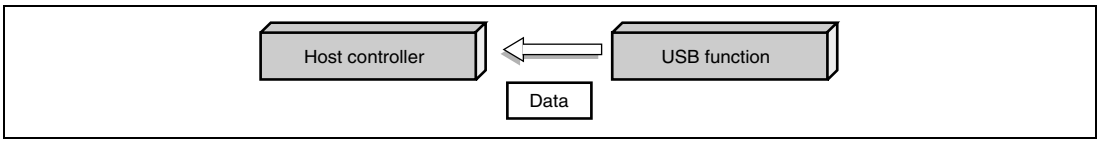

Figure 5.9 Interrupt Transfer

In interrupt transfers, the host PC receives data from the SH7727 function. The firmware is WAIT state or TRANS_IN state. Figure 5.10 shows the operation of the sample program of the interrupt transfer. On the left side of the figure shows the operation of the USB function module.

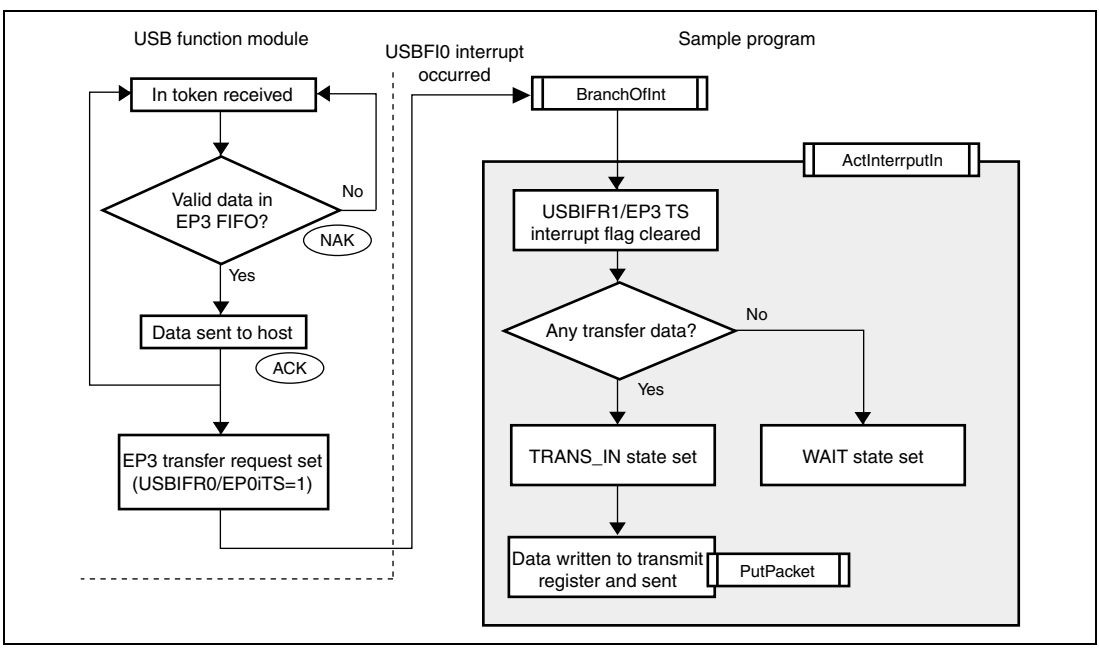

Figure 5.10 Interrupt-In Transfer

5.7 Hub Control by SH7727 Host

A hub driver has three roles; activating connected hub, resetting the USB hub connected to a downstream port or the USB function device, and managing downstream ports of a hub.

Figure 5.11 shows the hub drivers.

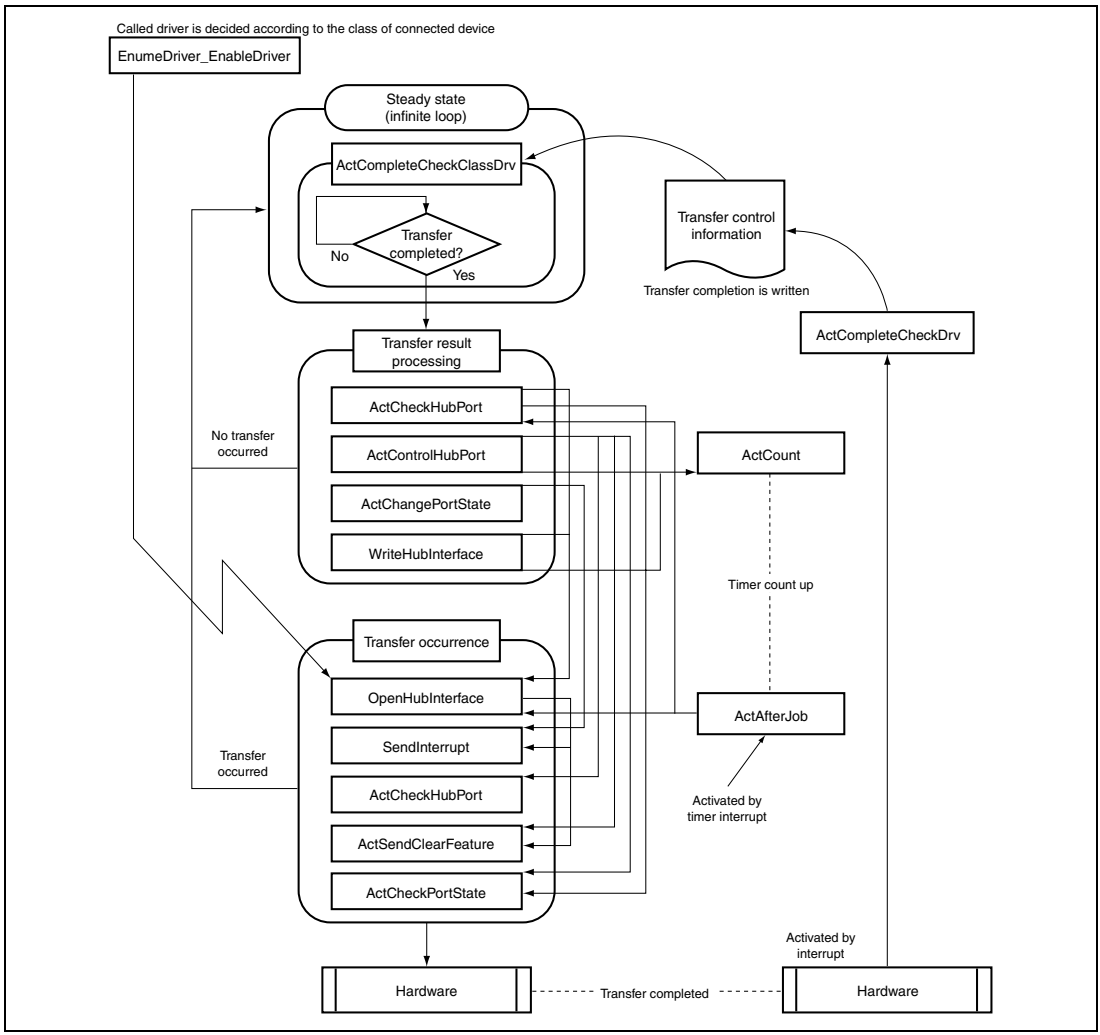

Figure 5.11 Hub Drivers

When setting a device address by functions in the USBD layer has been completed, the OpenHubInterface function is called to start activation processing by a hub driver.

Figure 5.12-1 shows transfers performed between the SH7727 host and a hub; from obtaining descriptor information to completion of activation processing.

Figure 5.12-2 shows transfers performed between the SH7727 host and a hub. The transfers are performed to report the change of the hub state after completion of activation processing by an interrupt transfer, to check the state of a downstream port by receiving the report, and to operate a down stream port.

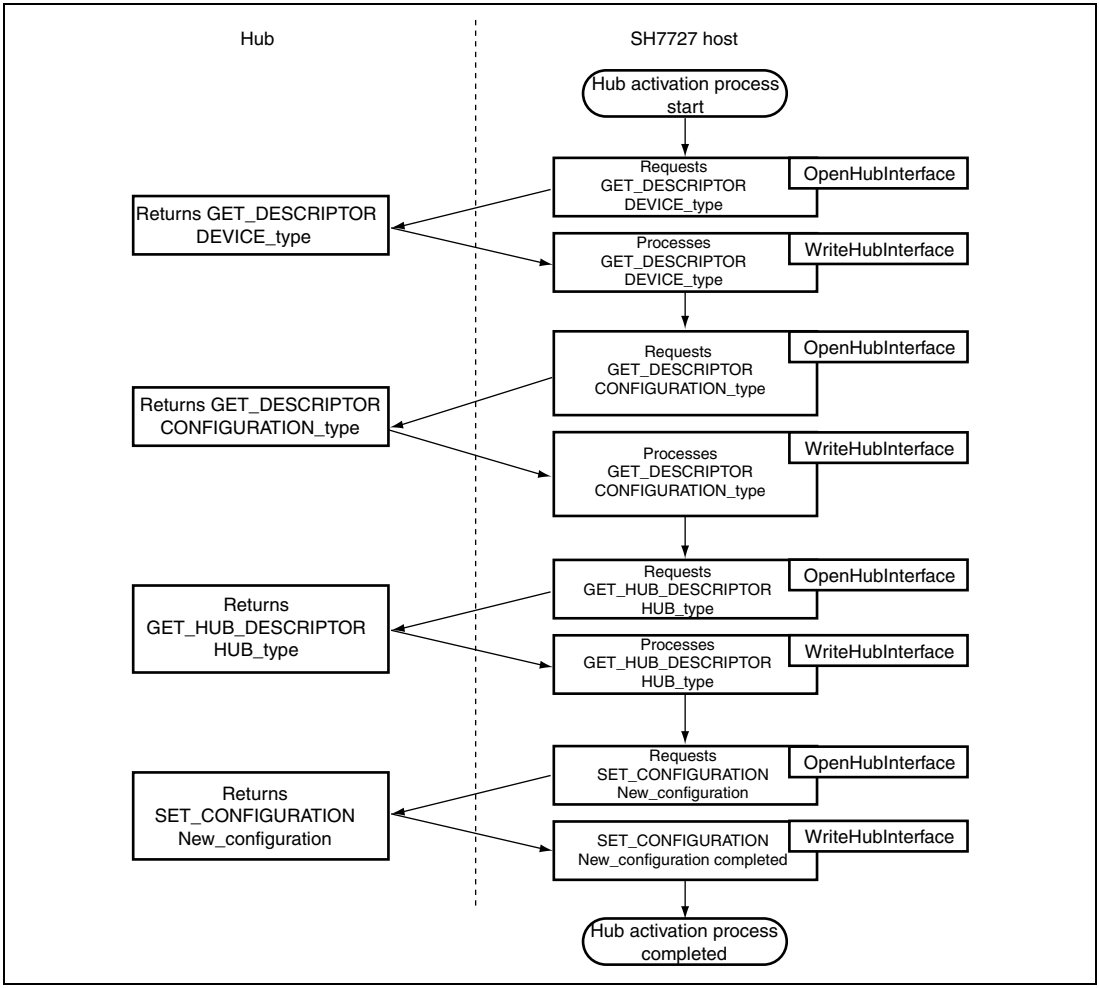

Figure 5.12-1 Flow of SH7727 Host—Hub Transfer 1

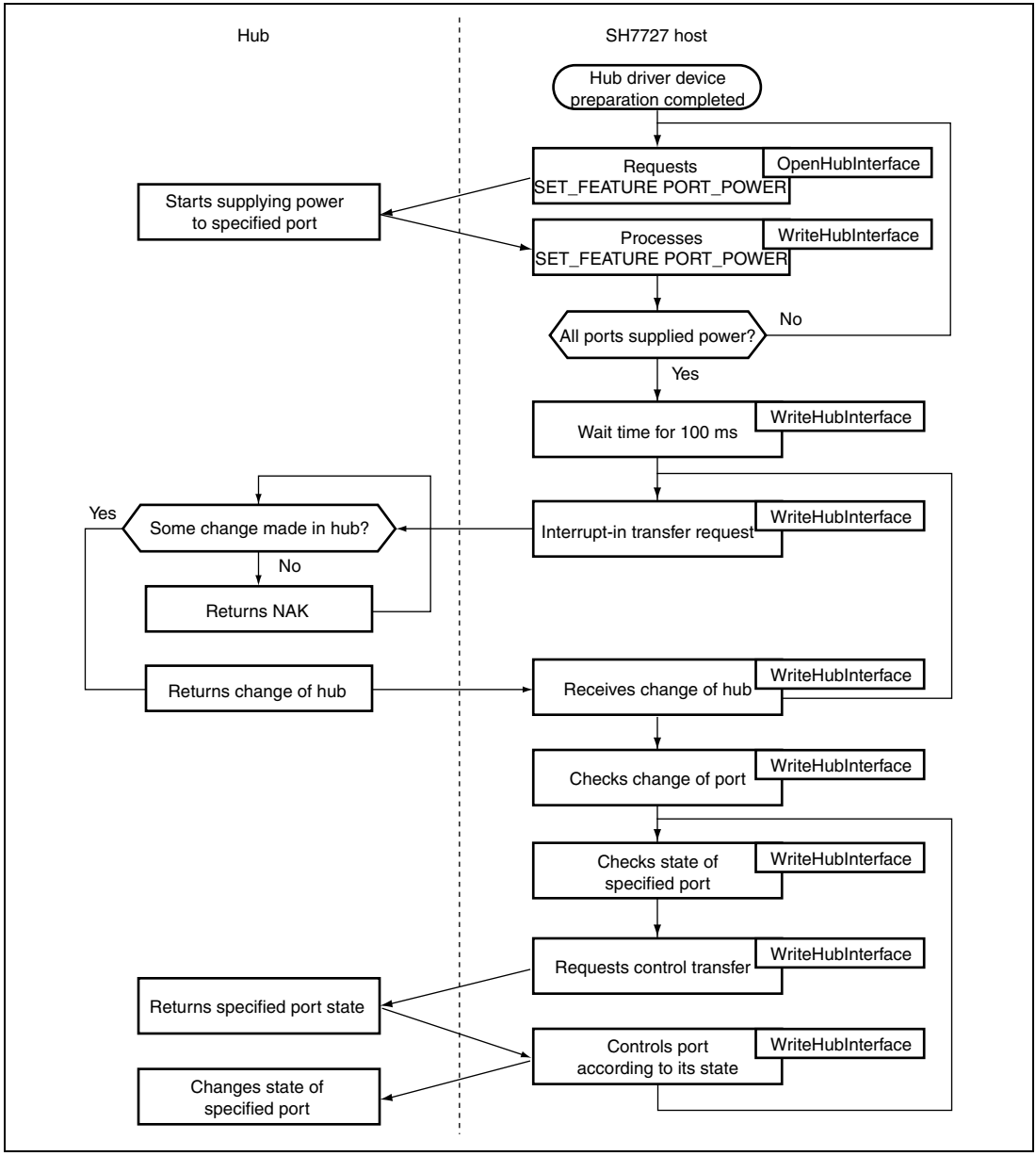

Figure 5.12-2 Flow of SH7727 Host—Hub Transfer 2

Next, five flowcharts of functions that issue transfers are shown below.

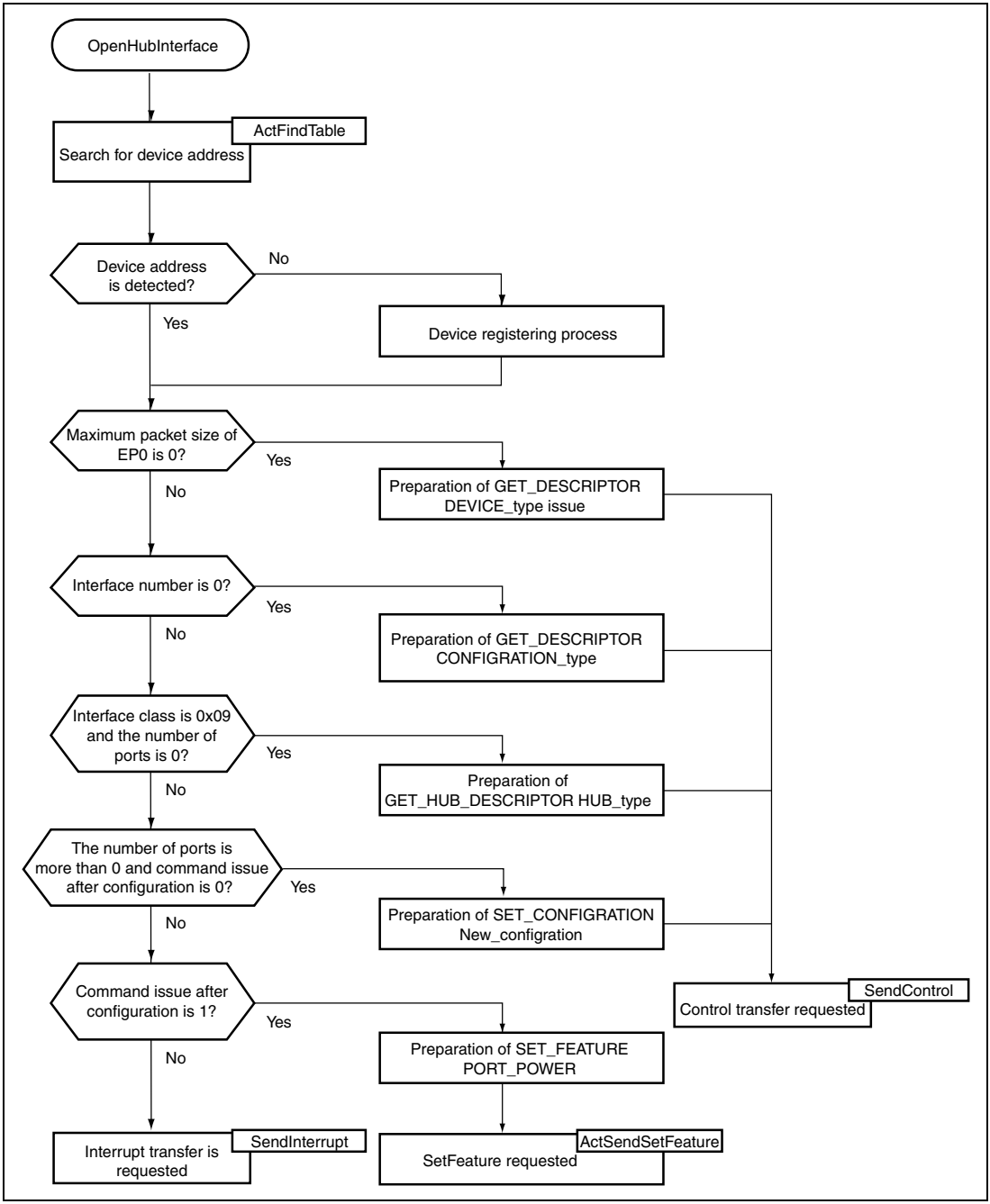

Figure 5.13-1 Transfer Issue Functions (1)

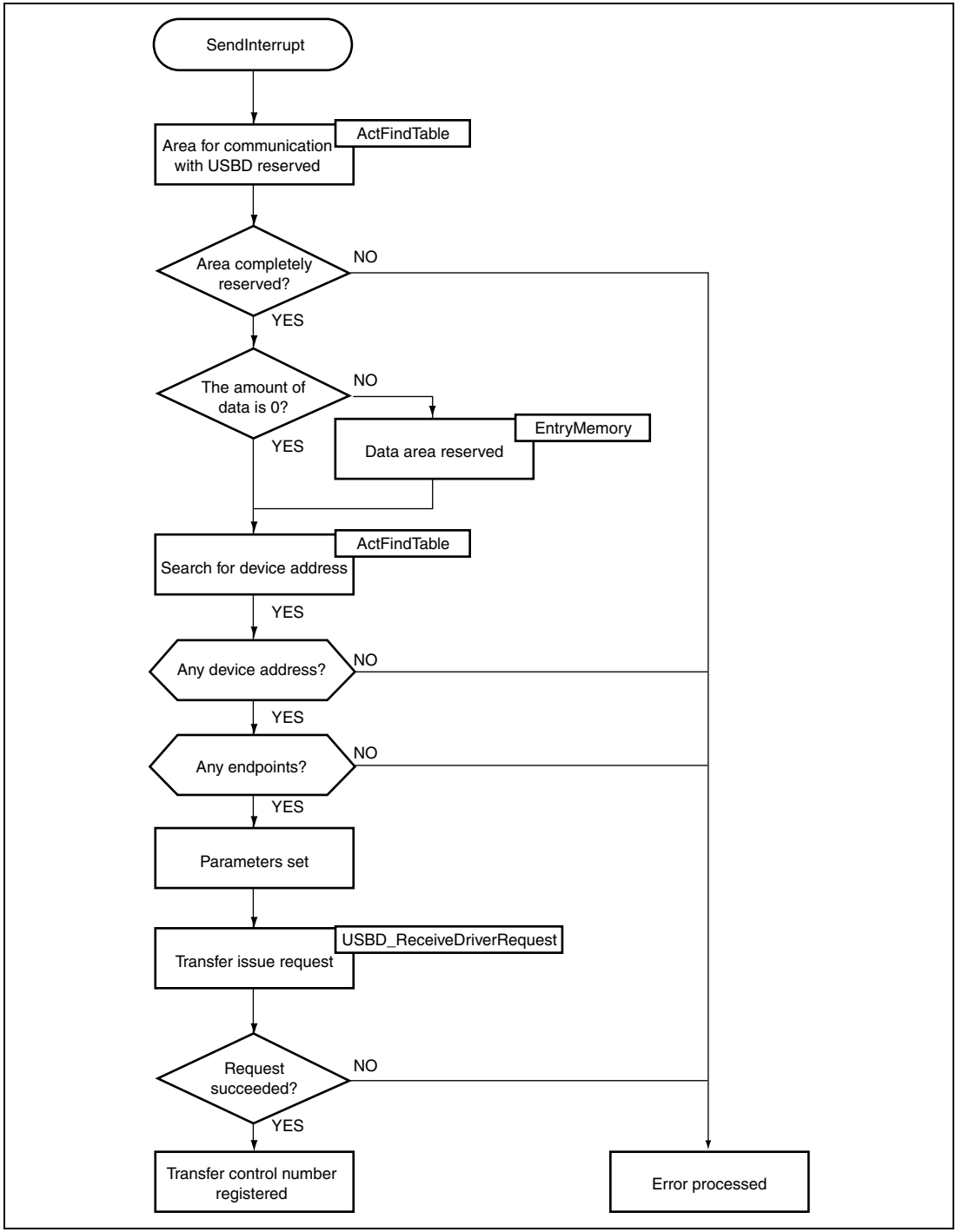

Figure 5.13-2 Transfer Issue Functions (2)

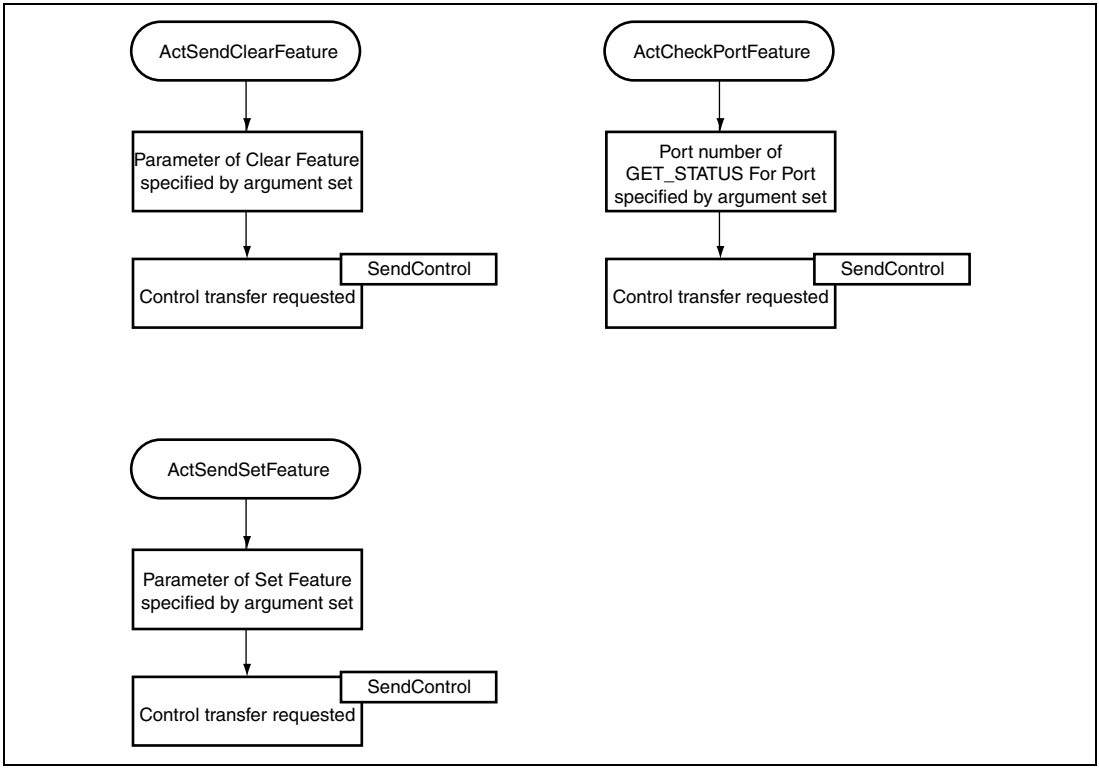

Figure 5.13-3 Transfer Issue Functions (3)

Four flowcharts of transfer result processing functions that start processing after the completion of transfers are shown below.

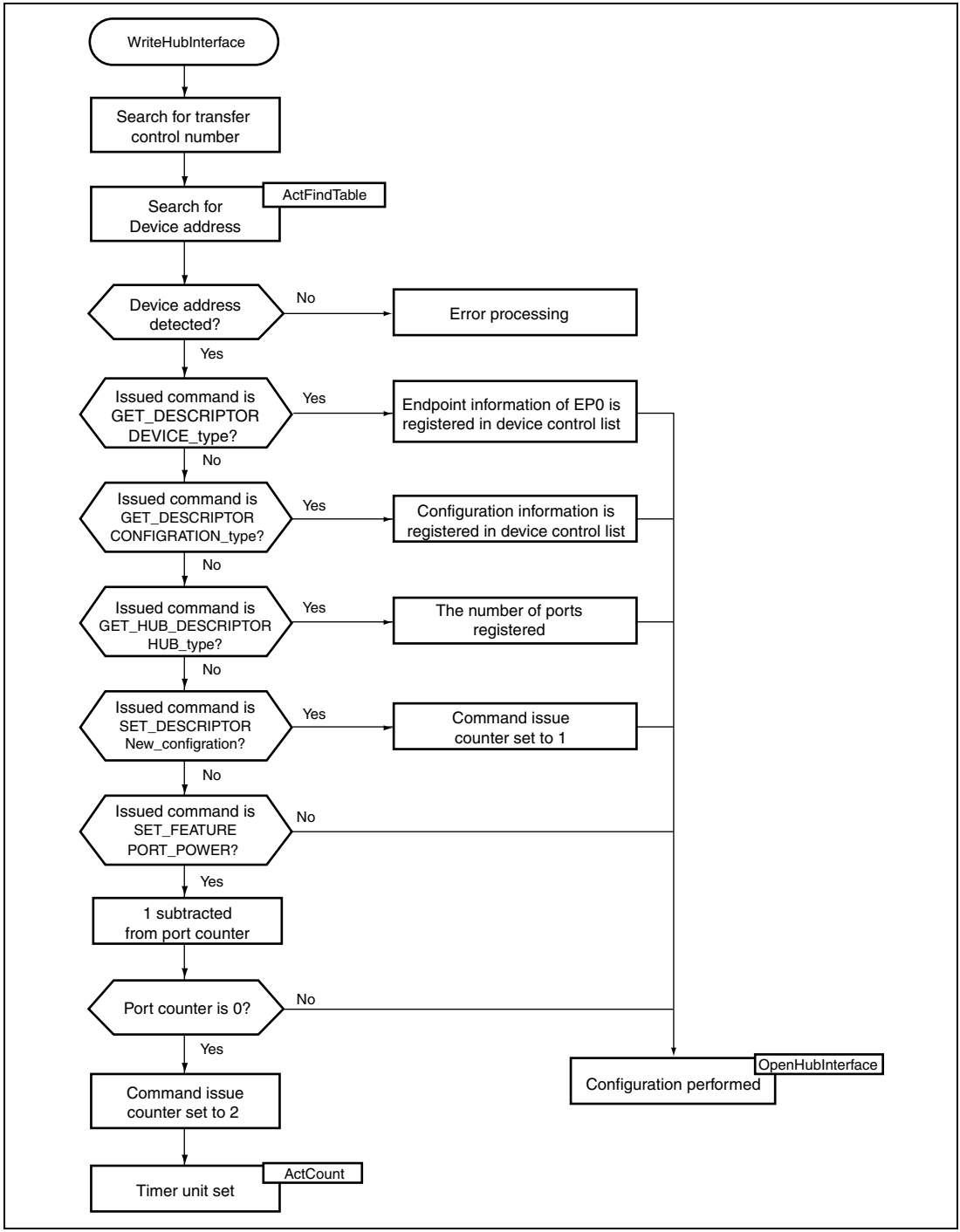

Figure 5.14-1 Transfer Result Processing (1)

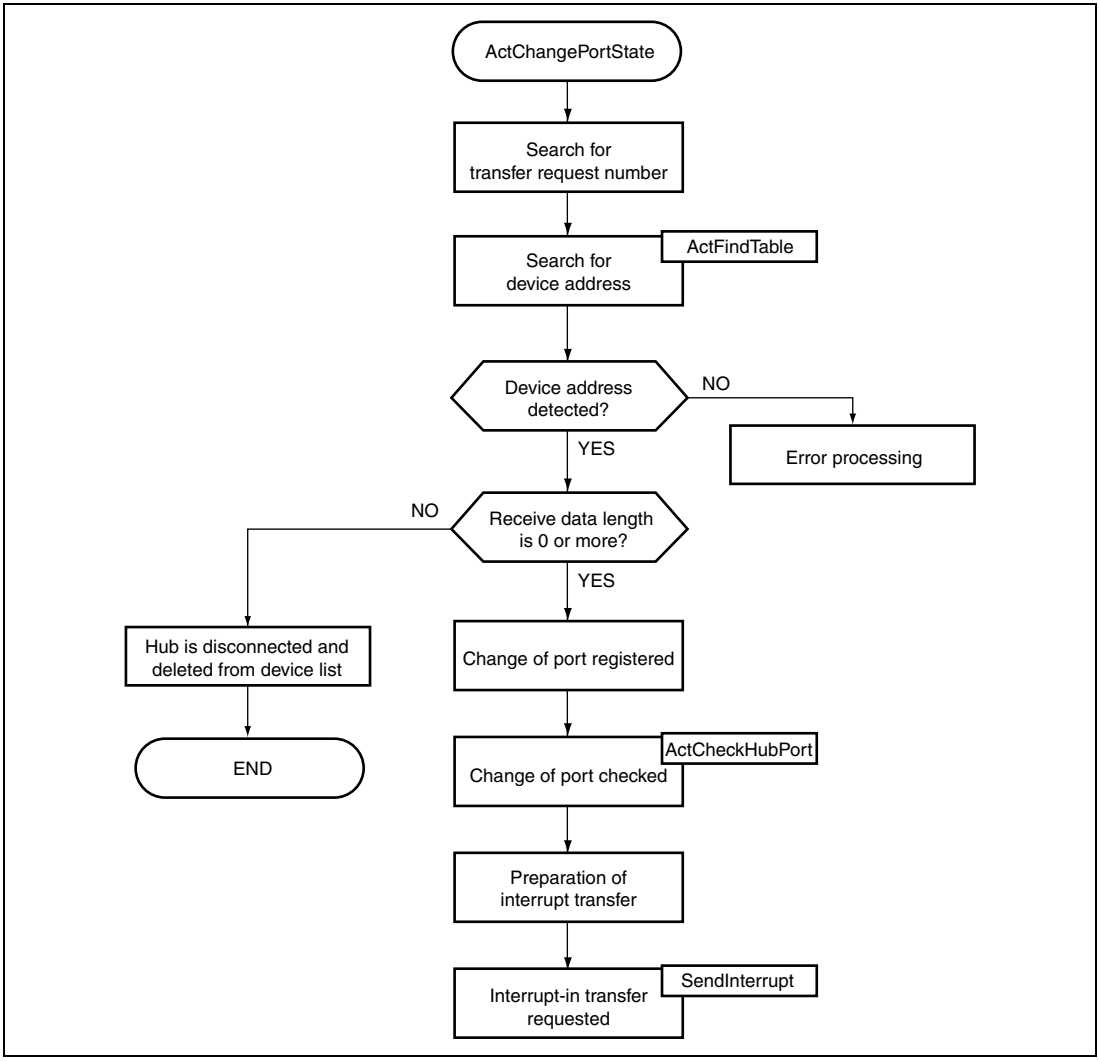

Figure 5.14-2 Transfer Result Processing (2)

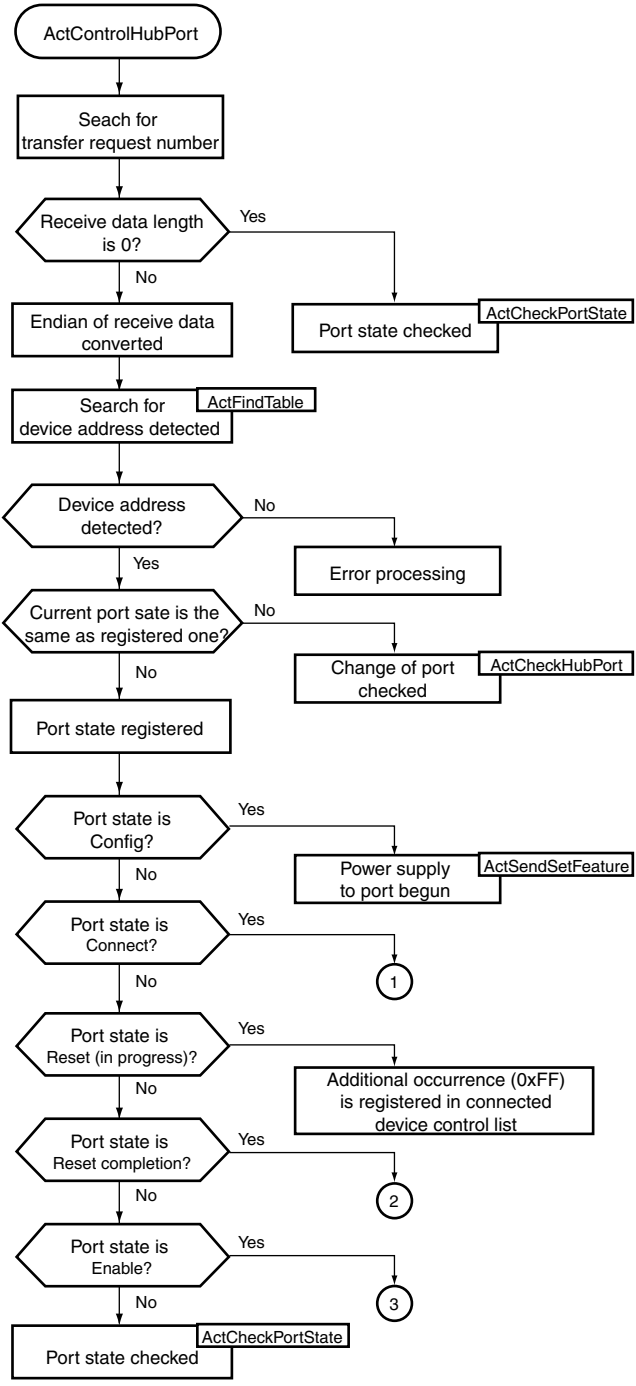

Figure 5.14-3 Transfer Result Processing (3)

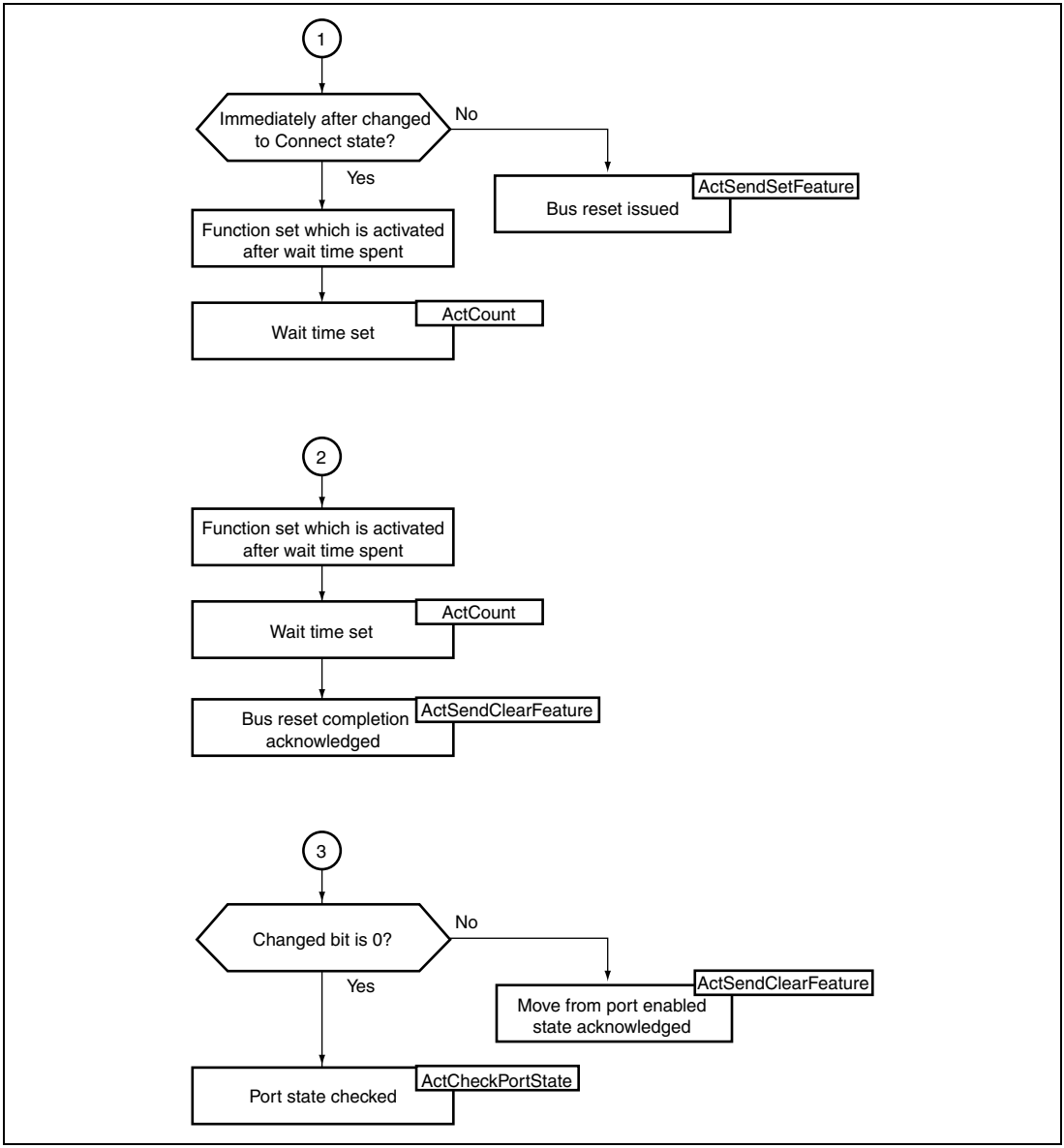

Figure 5.14-4 Transfer Result Processing (4)

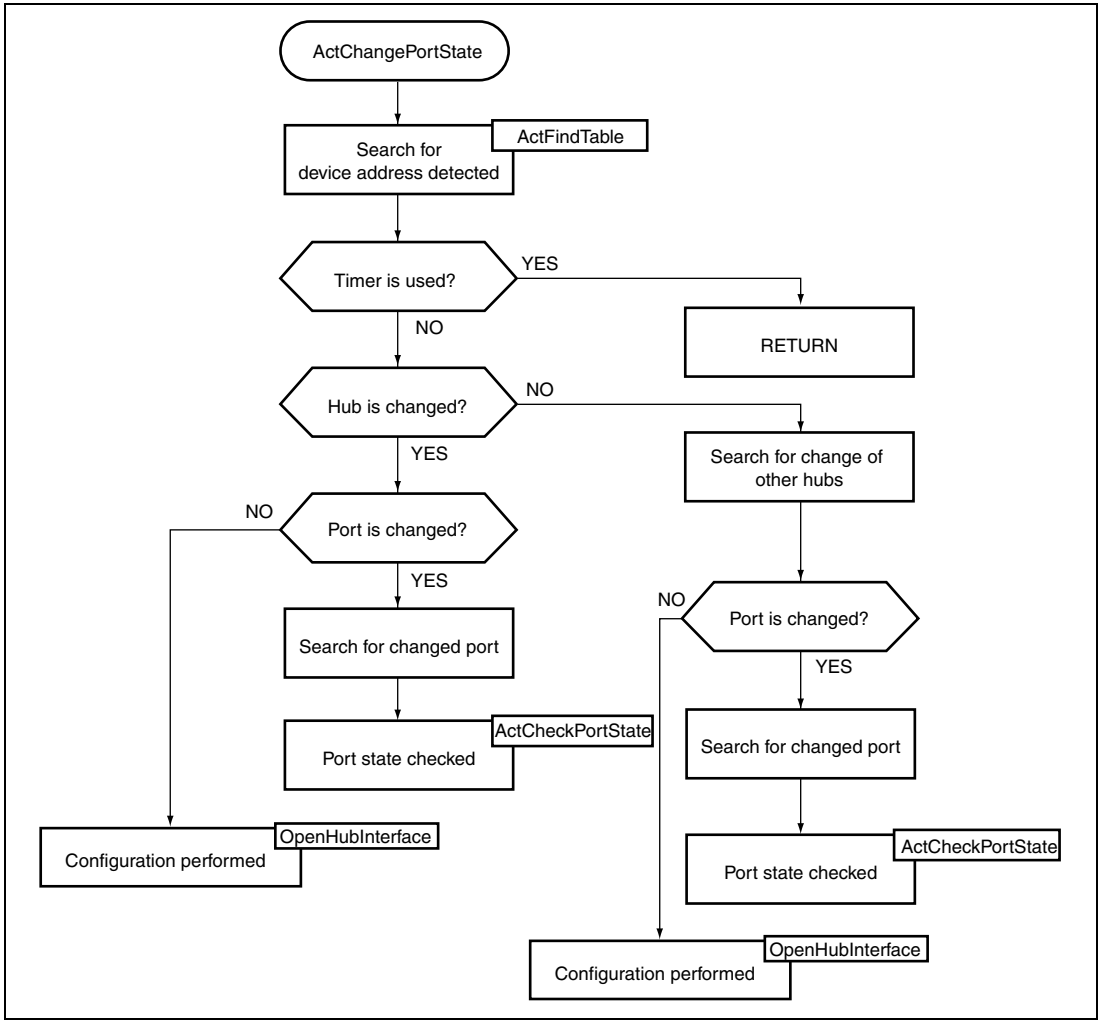

Figure 5.14-5 Transfer Result Processing (5)

5.8 HID Control by SH7727 Host

HID drivers perform activating the HID devices and transferring HID data.

Figure 5.15 shows the HID drivers.

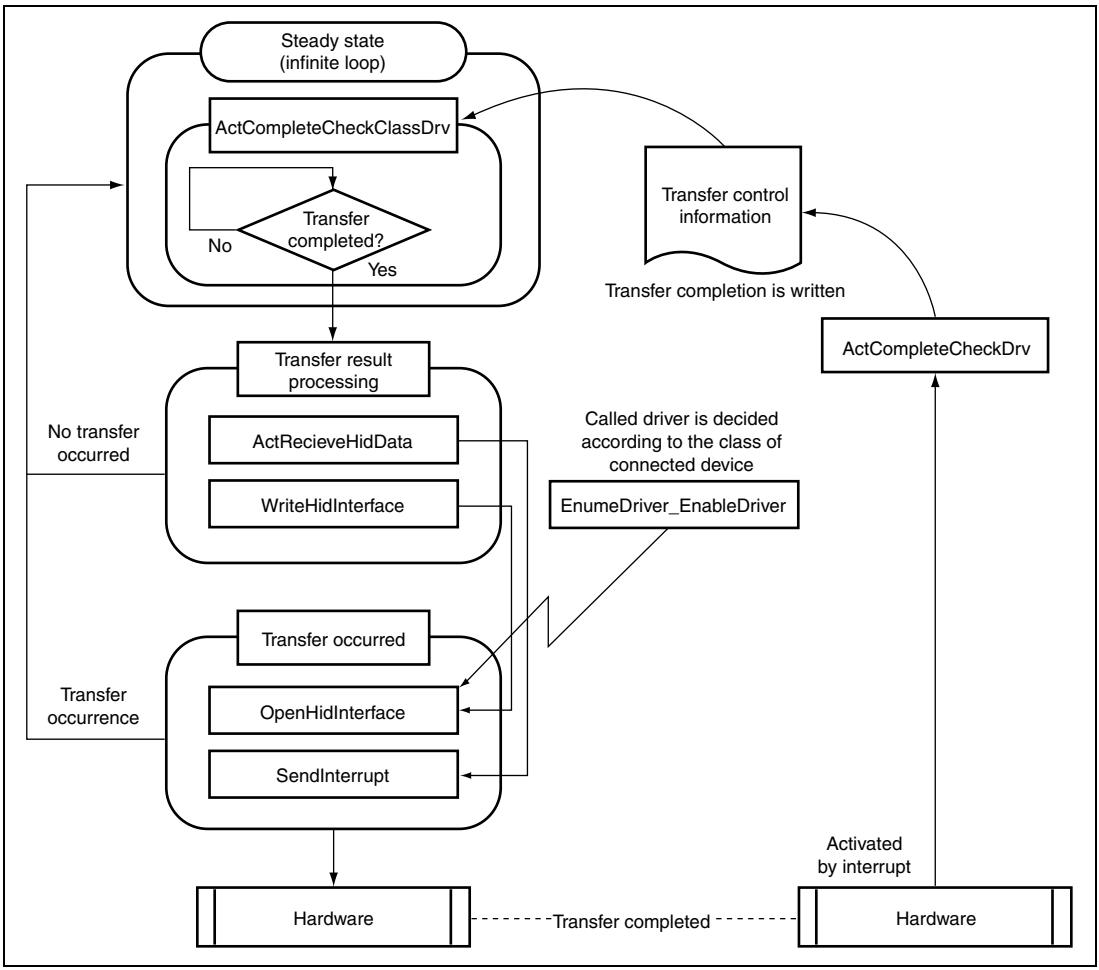

Figure 5.15 HID Drivers

When the setting of a device address has been completed by functions in the USBD layer, the OpenHidInterface function is called to start activation processing by a HID driver.

Figure 5.16-1 shows transfers performed between the SH7727 host and a HID device from obtaining descriptor information to the completion of activation processing.

Figure 5.16-2 shows HID data transfers performed between the SH7727 host and a HID device in interrupt transfers.

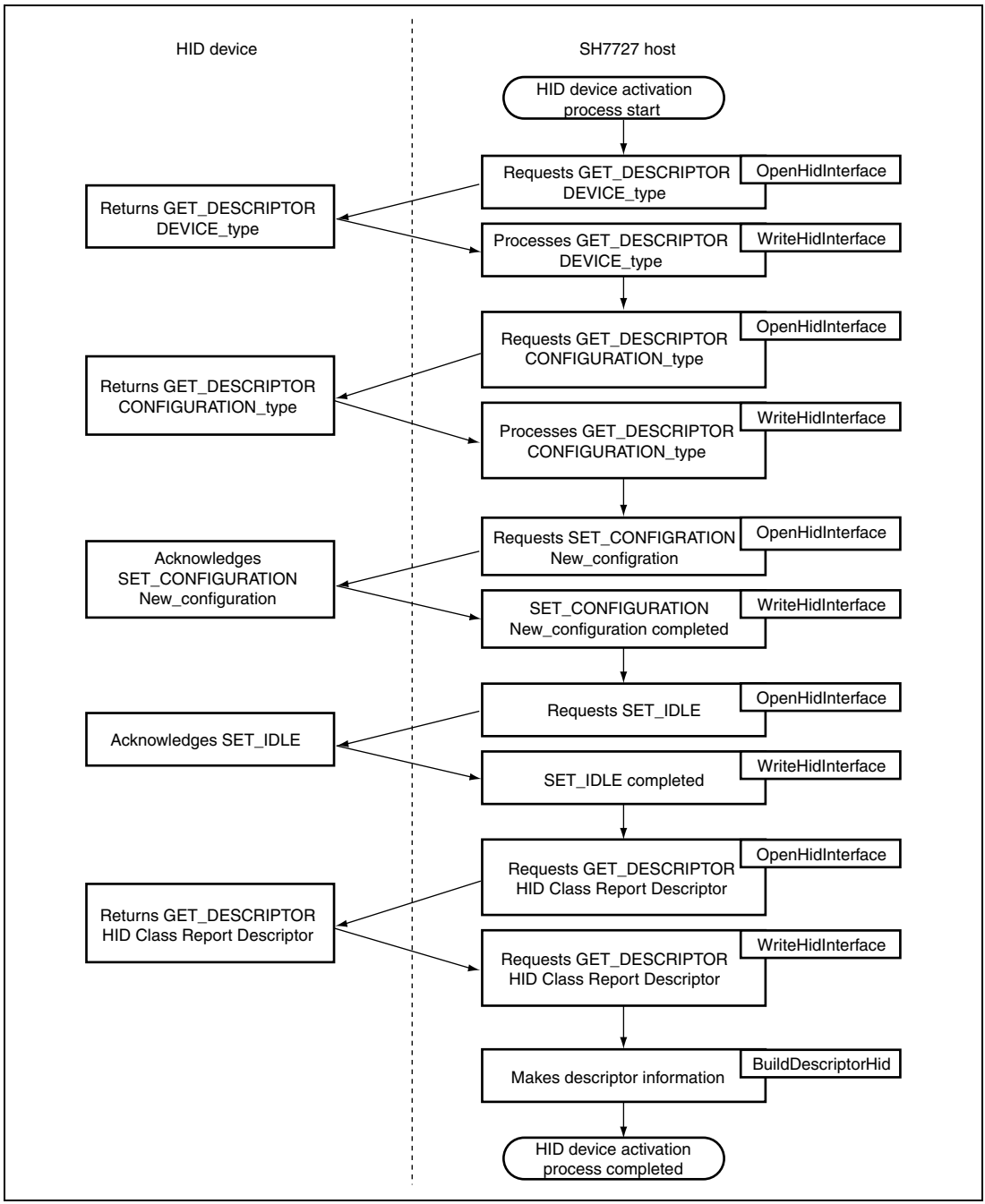

Figure 5.16-1 Flow of SH7727 Host—HID device transfer 1

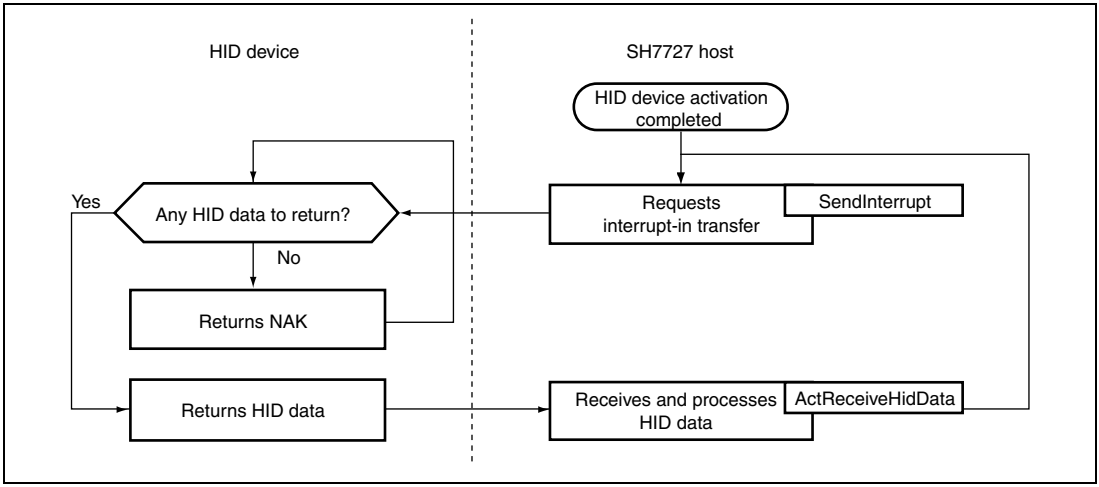

Figure 5.16-2 Flow of SH7727 Host—HID device transfer 2

Two flowcharts of functions that issue transfers are shown below.

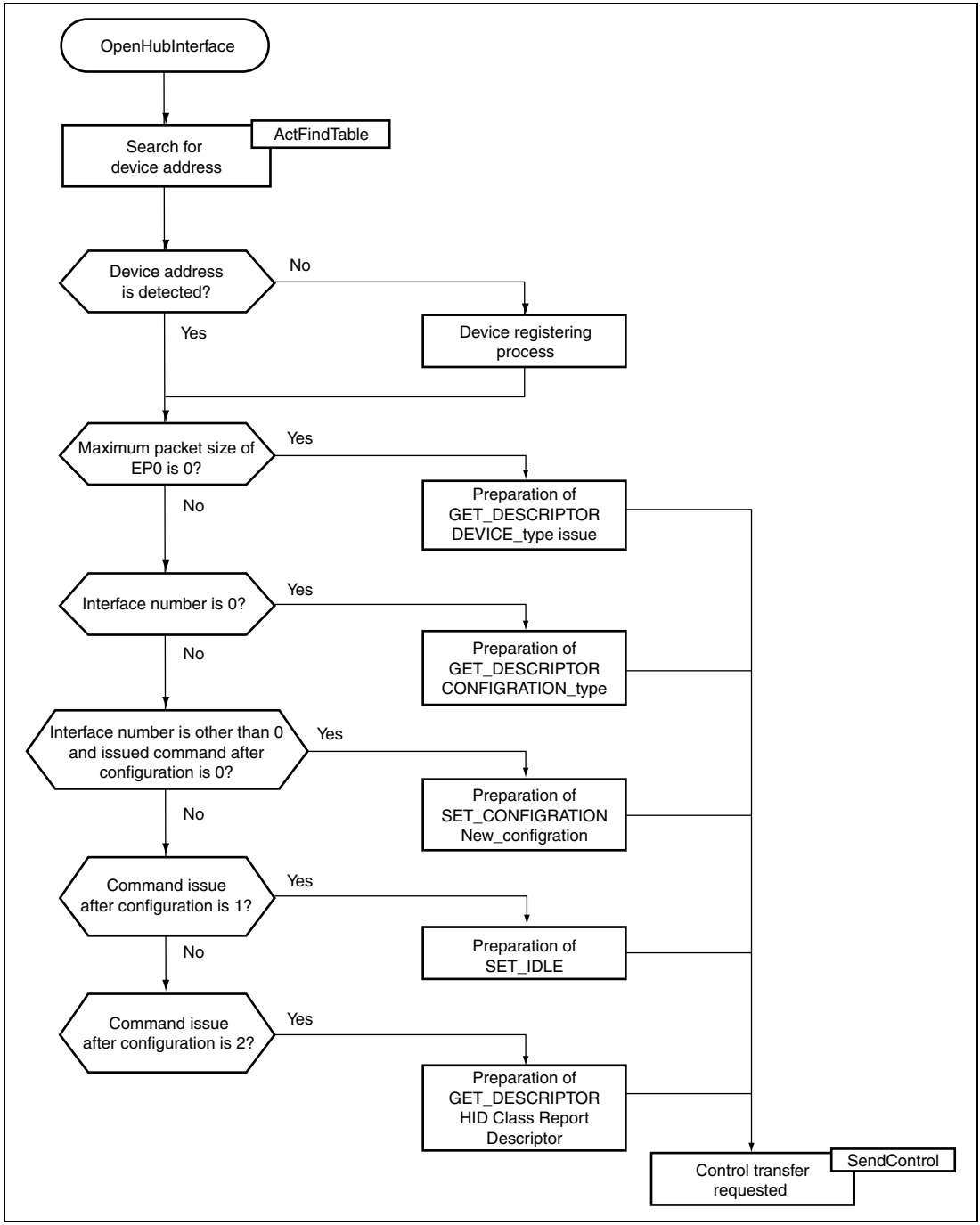

Figure 5.17-1 Transfer Issue Functions (1)

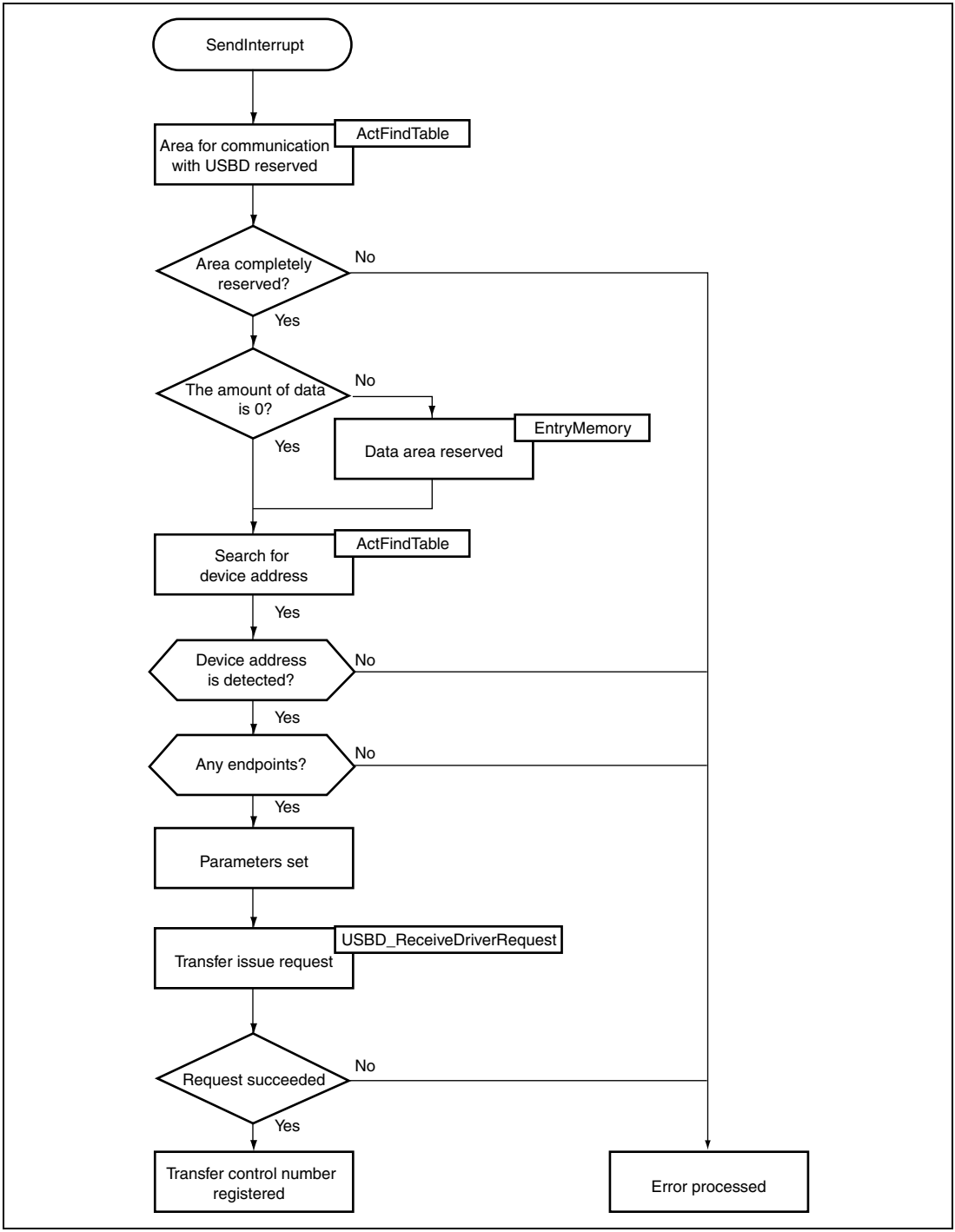

Figure 5.17-2 Transfer Issue Functions (2)

Two flowcharts of transfer result processing functions that start processing after the completion of transfers are shown below.

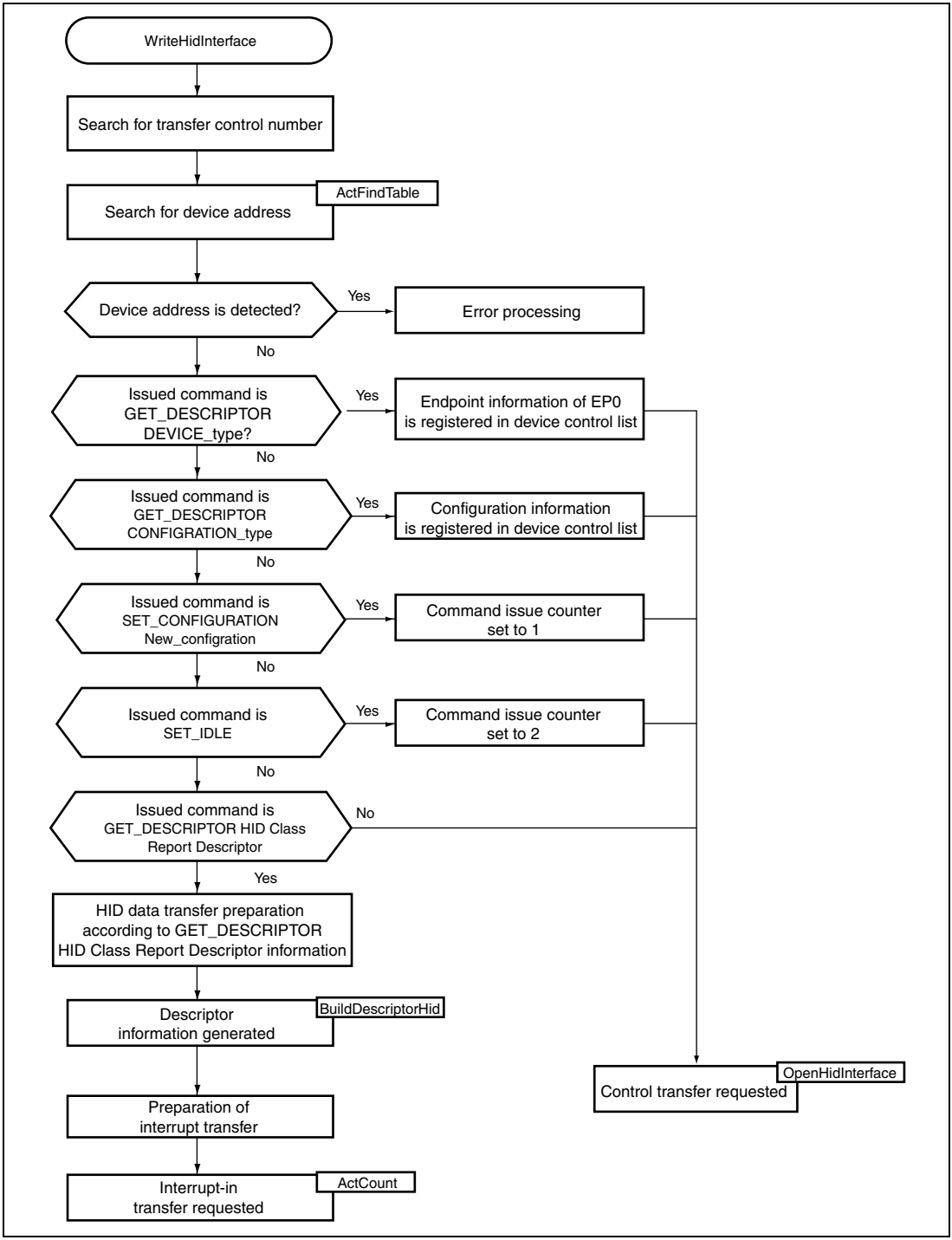

Figure 5.18-1 Transfer Result Processing (1)

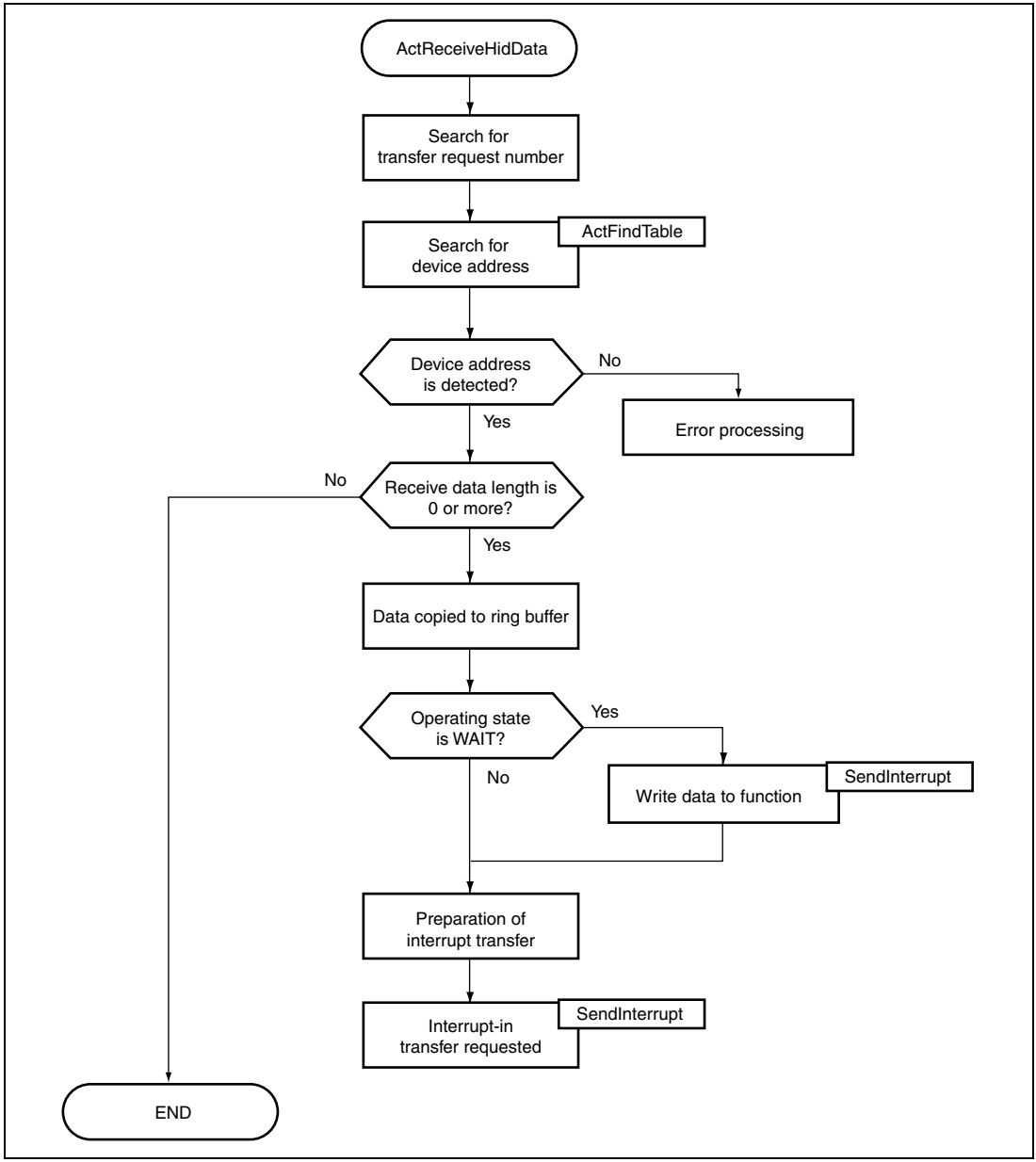

Figure 5.18-2 Transfer Result Processing (2)

5.9 Storage Control by SH7727 Host

A storage driver performs activating the connected storage device. In actual storage data transfers a link program connects the host PC and a storage device.

Figure 5.19 shows the storage drivers.

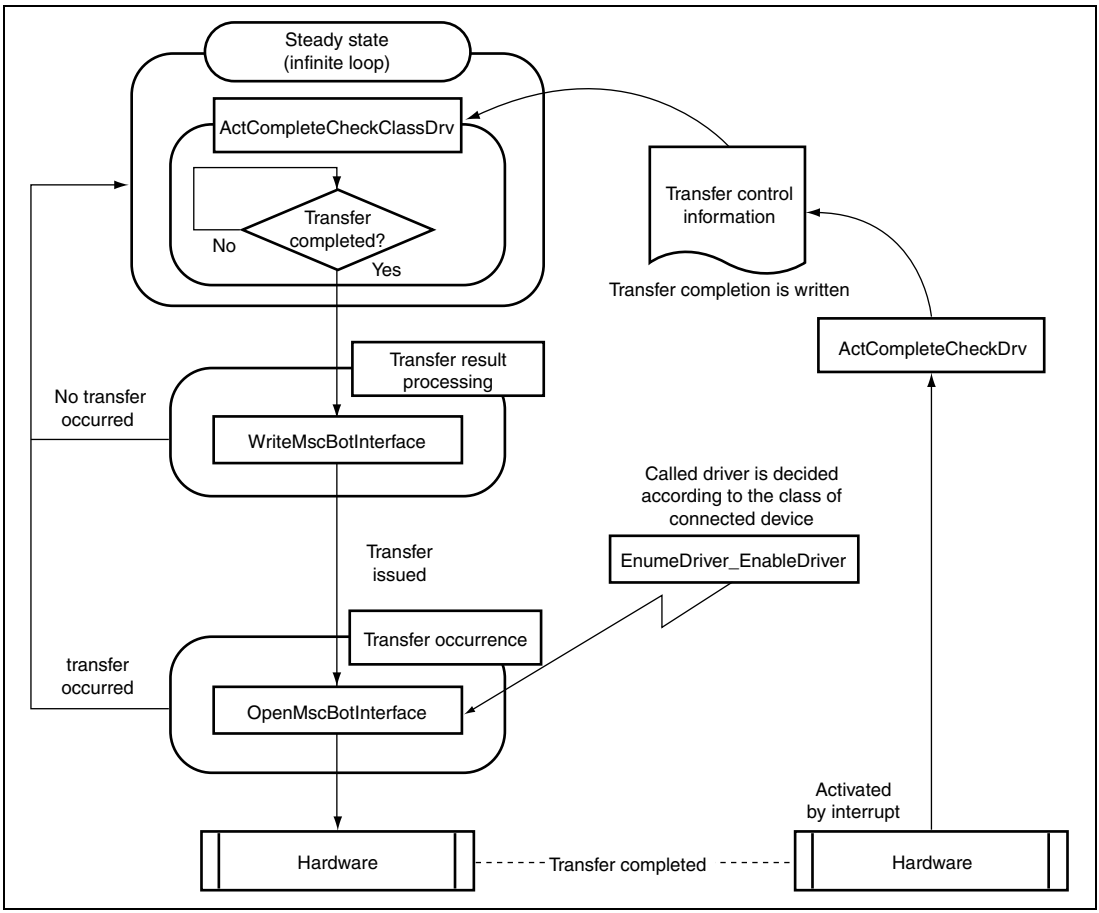

Figure 5.19 Storage Drivers

When the setting of the device address has been completed by functions in the USBD layer, the OpenMscInterface function is called to start activation processing by a storage driver.

Figure 5.20 shows transfers performed between the USB host and a storage device from obtaining descriptor information to completion of activation processing.

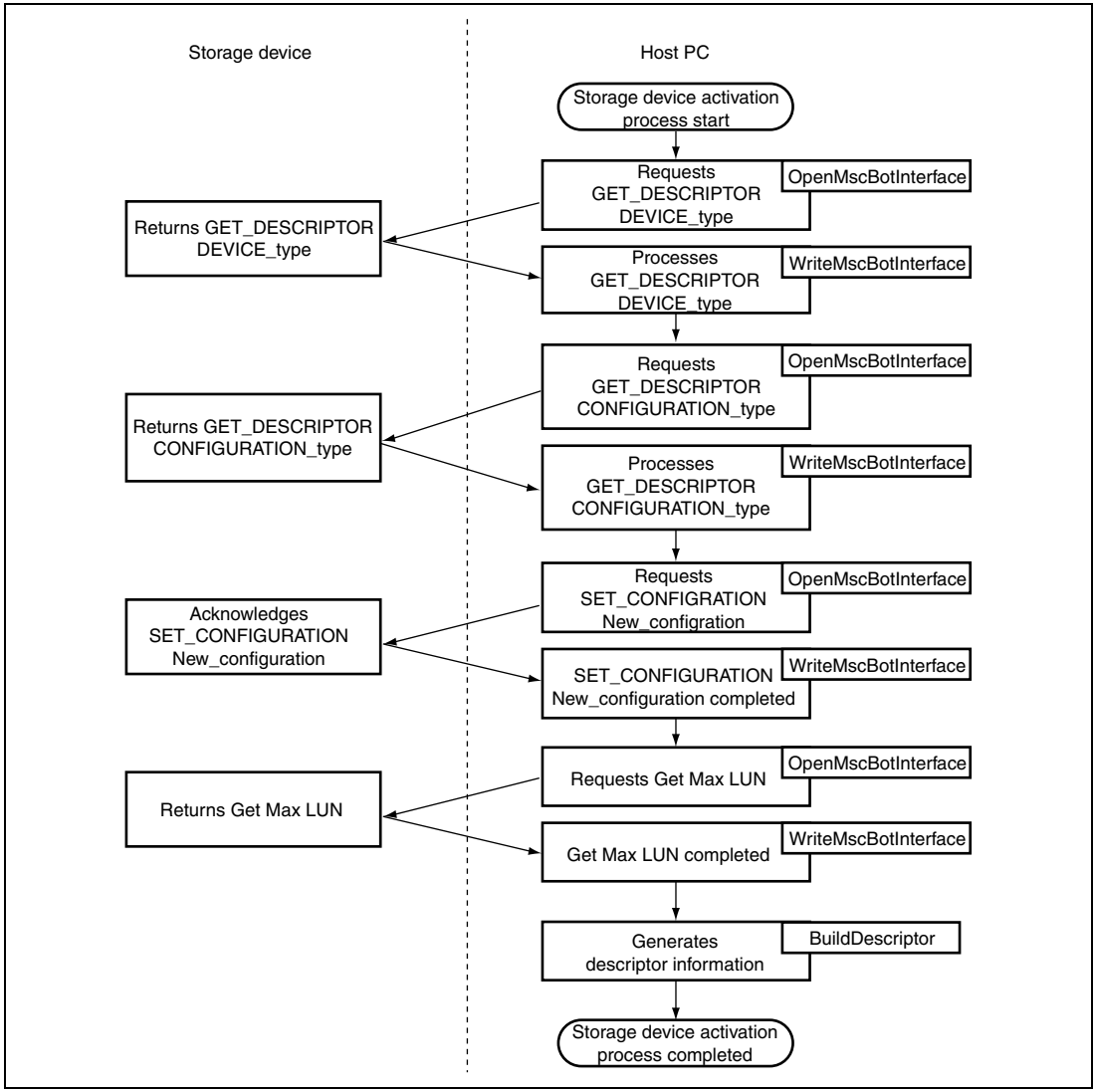

Figure 5.20 Flow of SH7727 Host - Storage device transfer

Next, a flowchart of functions that issue transfers is shown below.

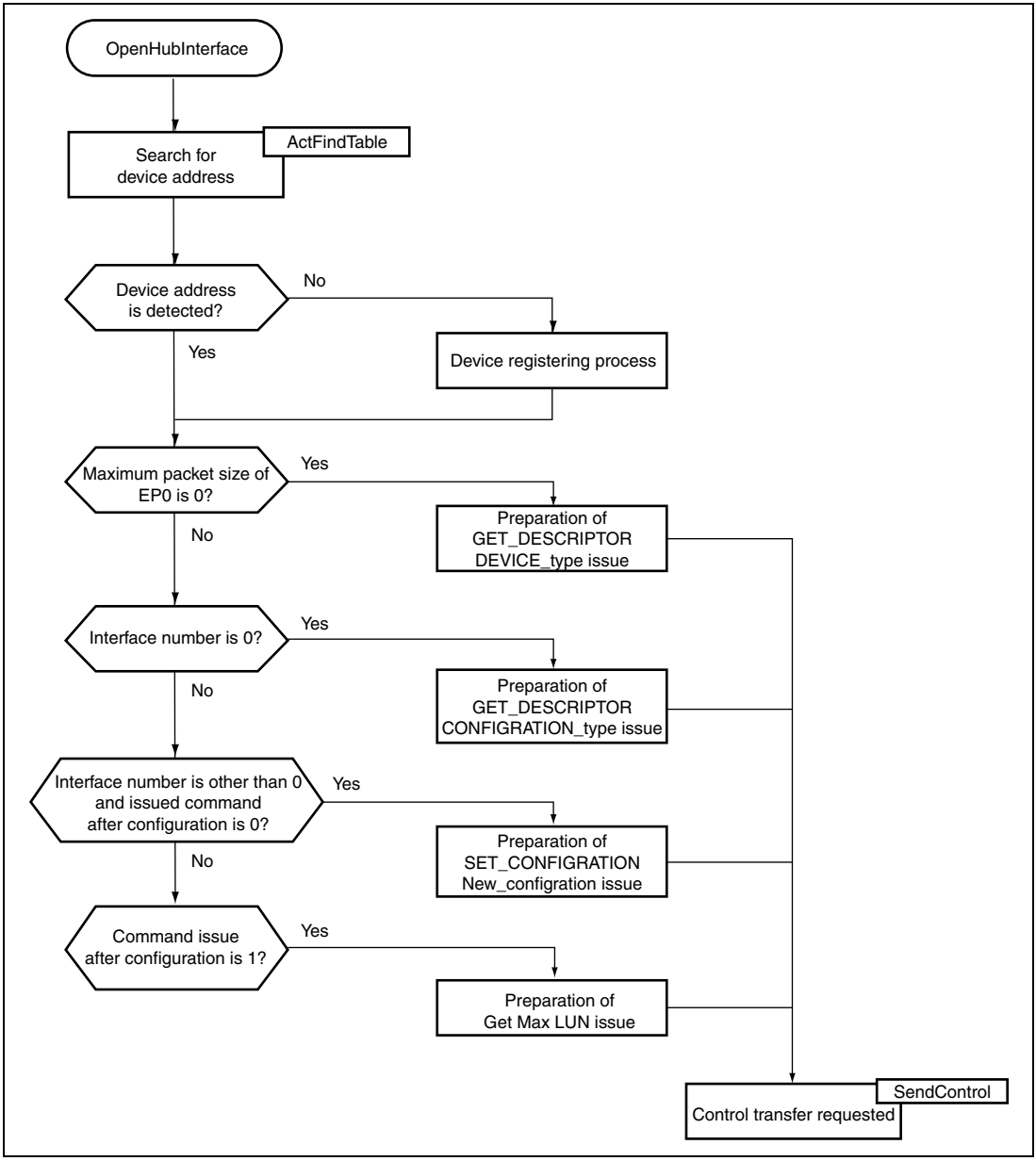

Figure 5.21 Transfer Issue Functions

A flowchart of transfer result processing functions that starts processing after the completion of transfers is shown below.

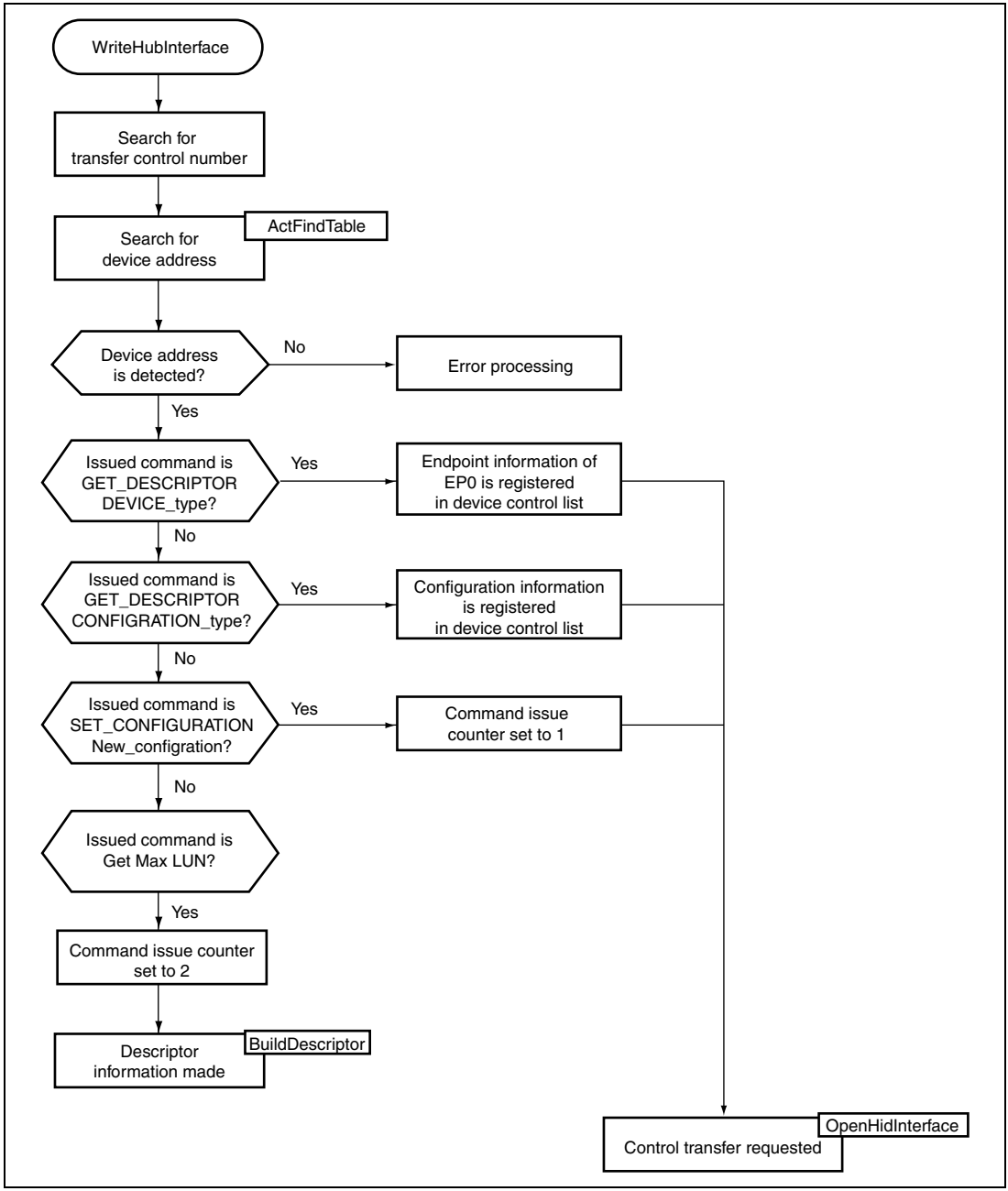

Figure 5.22 Transfer Result Processing

5.10 Link Operation between SH7727 Host and SH7727 Function

This sample program is categorized into major three programs: link program, program of the SH7727 host, and program of the SH7727 function.

- The program of the SH7727 host controls connected USB hub and USB function devices and transmits/receives data
- The program of the SH7727 function transmits/receives data between the host PC and the function module
- The link program controls transmitting/receiving data between the SH7727 host and SH7727 function when the storage data is transferred between the host PC and the storage device.

The program of the SH7727 function communicates only with the host PC. Though the transfer data from the SH7727 function are various, they are classified according to the each transfer method. Therefore, details of the firmware are shown in USB transfer methods.

The controls of the program of the SH7727 host are various according to the connected USB hubs and USB function devices, therefore, details of the firmware control are shown in connected devices.

The link program is used to handle the storage device. The program connects the HS7727 host and SH7727 function and controls data transfer and devices. Figure 5.23 shows the entire figure of the link program.

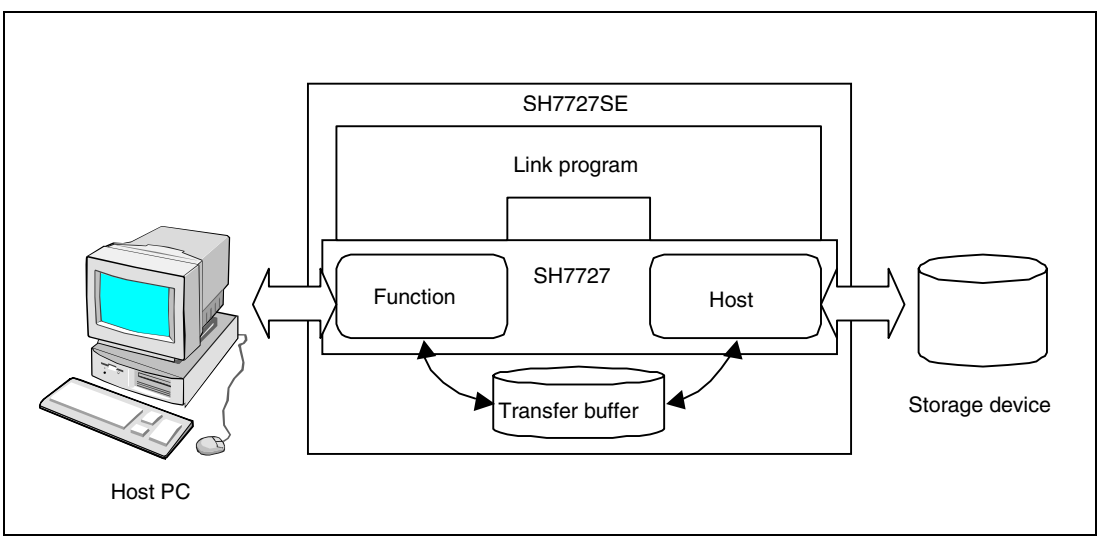

Figure 5.23 Link Program

Next, flowcharts of functions that are activated by the interrupts from the SH7727 function.

The link program is activated by an interrupt from the SH7727 function, and controls the SH7727 function and SH7727 host. In other words, a single interrupt source is used to control two different modules. Data packets are sent from the SH7727 function to the SH7727 host or from the SH7727 host to the SH7727 function in normal data transfer. However, when the storage device returns a STALL, the flow of data packets and operations of the SH7727 function and the SH7727 host are different.

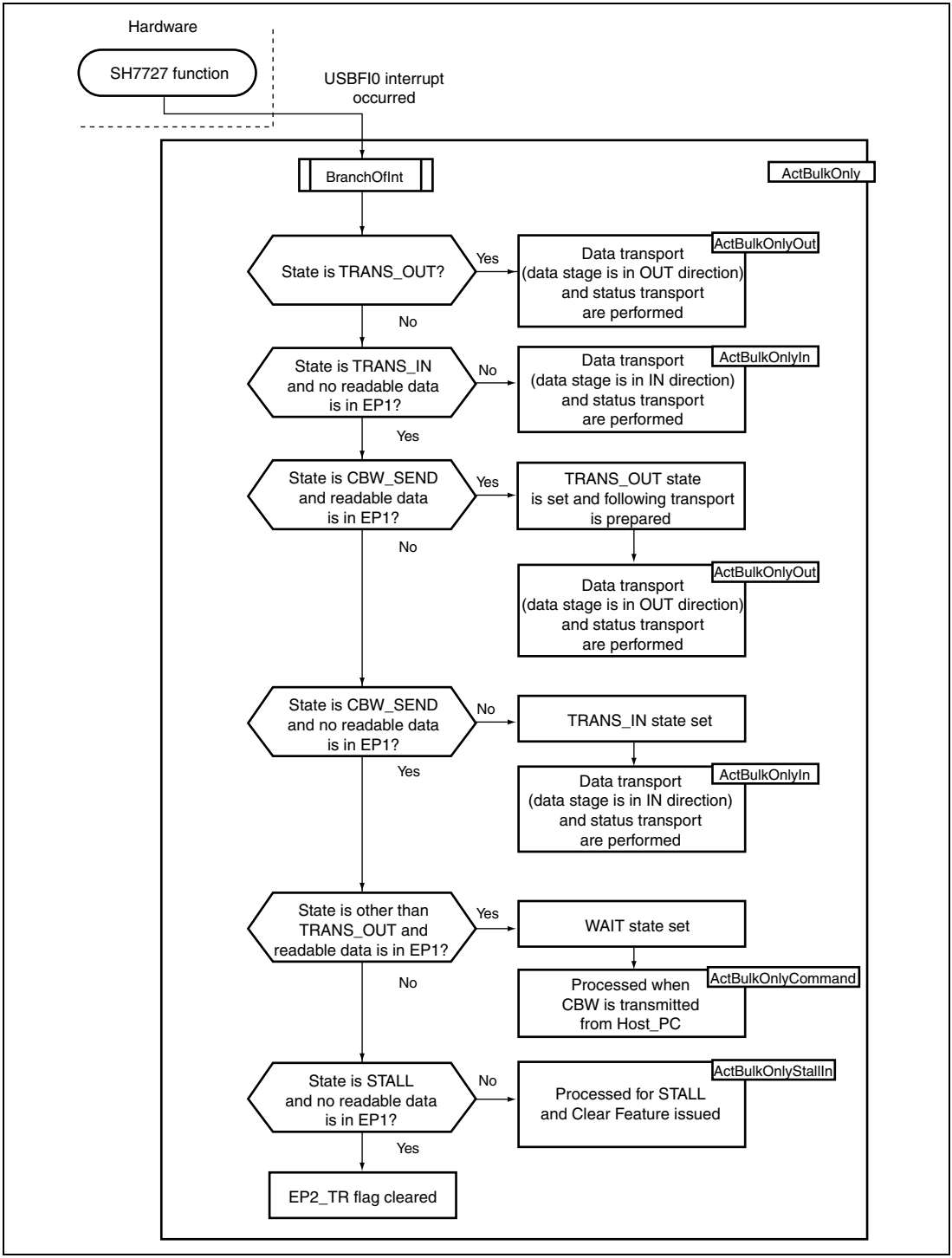

Figure 5.24-1 Link Operation Functions between SH7727 Host and SH7727 Function

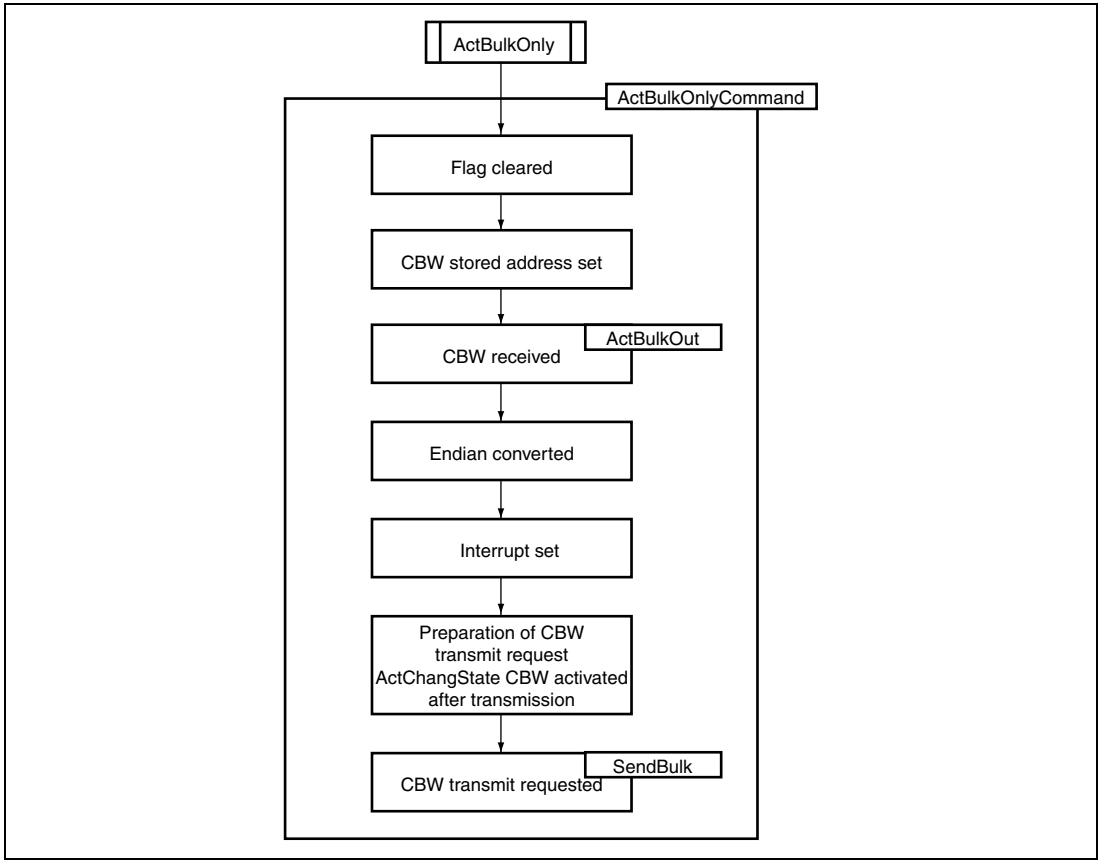

Figure 5.24-2 Functions on CBW Transmission from Host PC

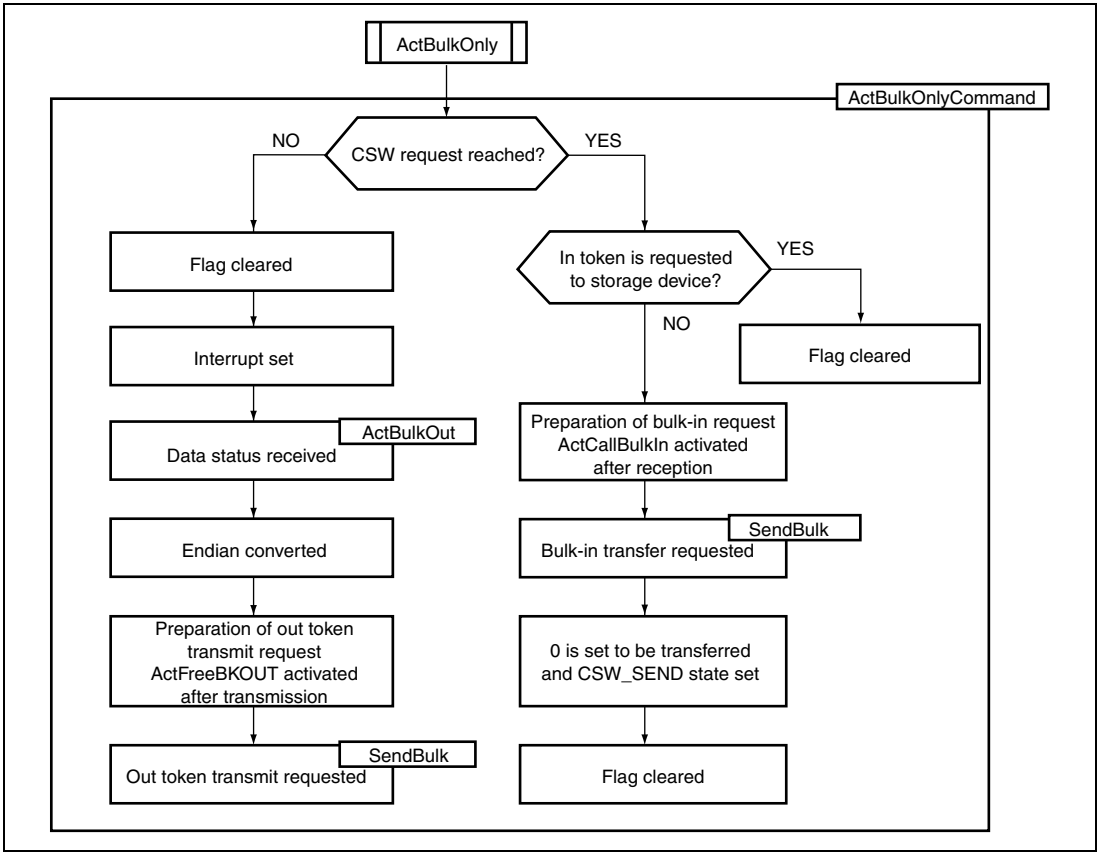

Figure 5.24-3 Control Functions for Data Transport (Data Stage is in Out Direction) and Status Transport Port

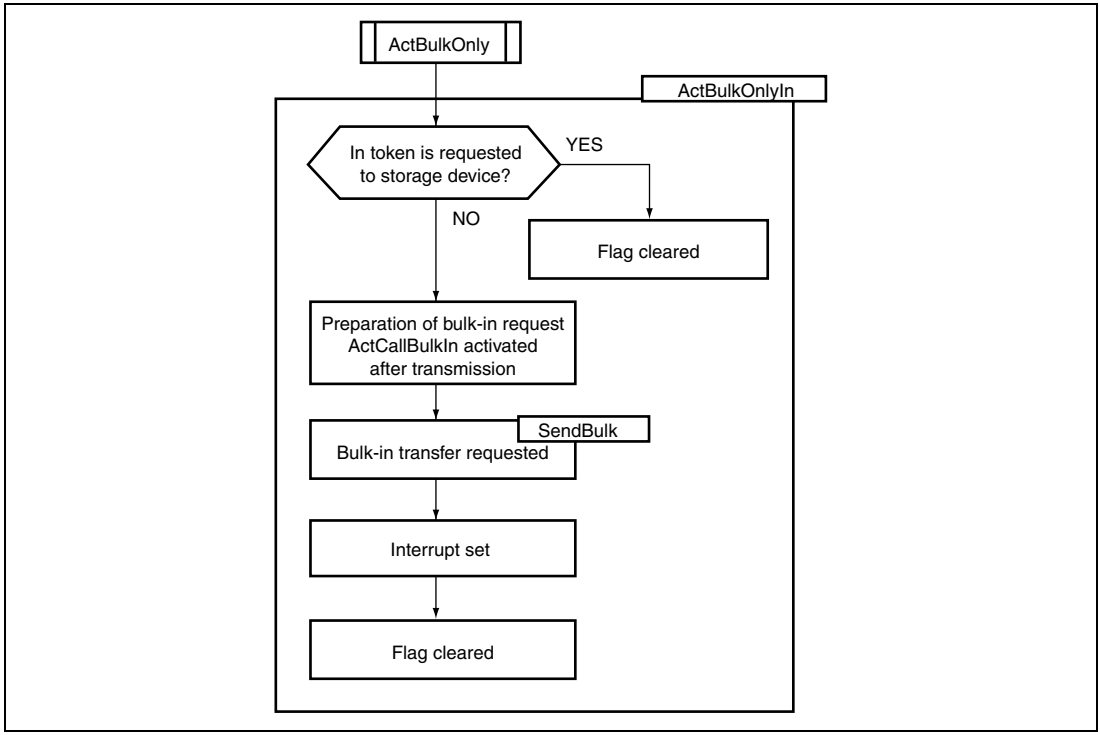

Figure 5.24-4 Control Functions for Data Transport (Data Stage is in IN Direction) and Status Transport Port

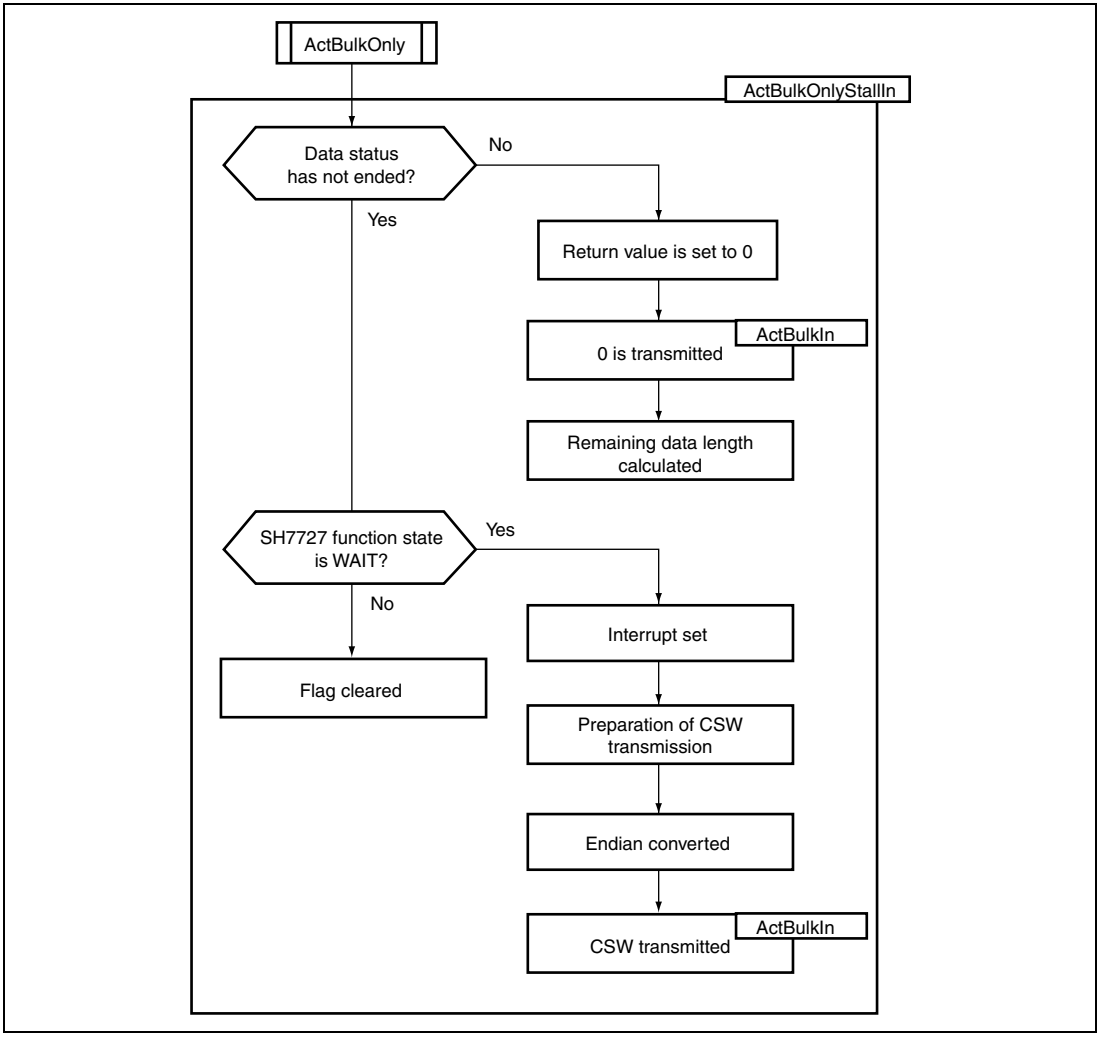

Figure 5.24-5 Functions that Process STALL

The flowcharts of functions that are initiated on the completion of the processing of the SH7727 host.

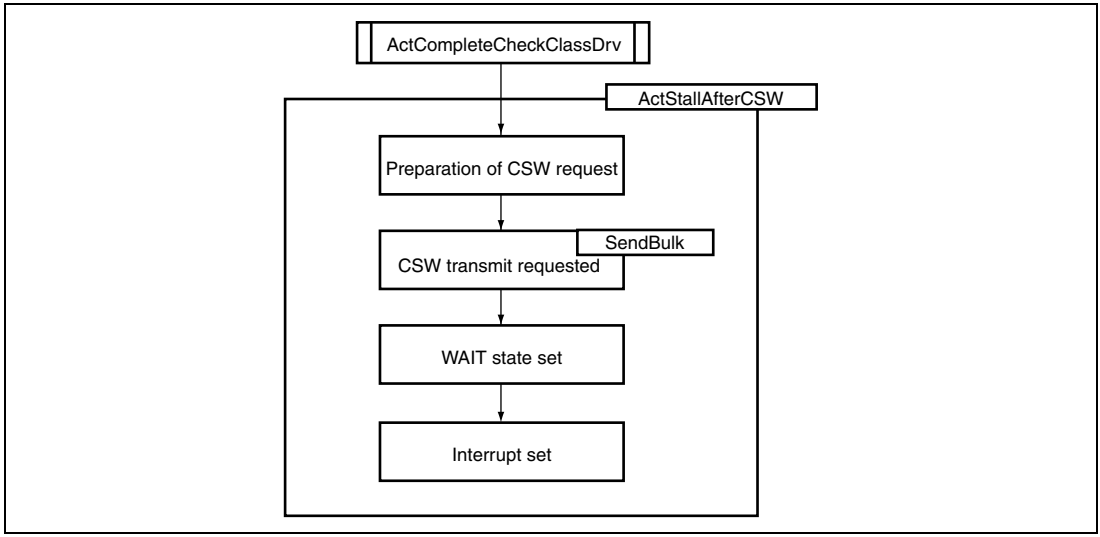

Figure 5.25-1 Functions for Process after Clear Feature Completed

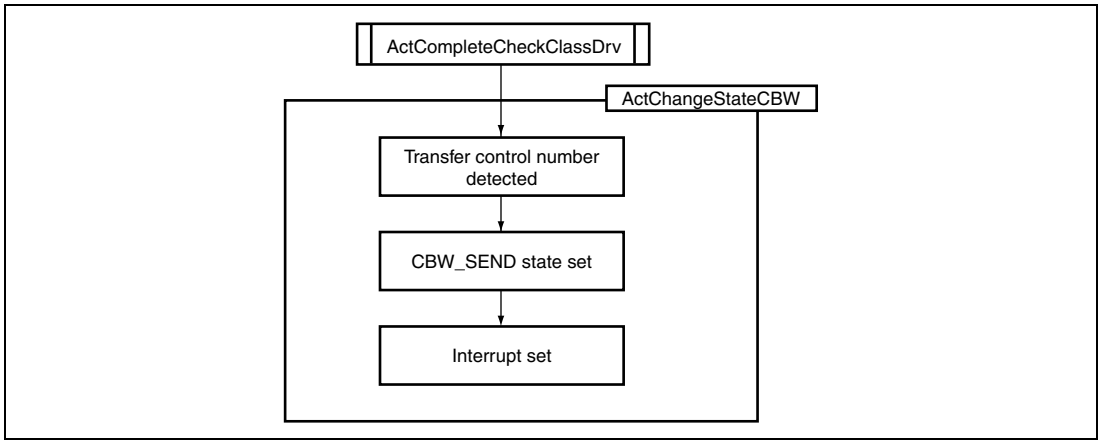

Figure 5.25-2 Functions for Process after CBW Transmit Completed

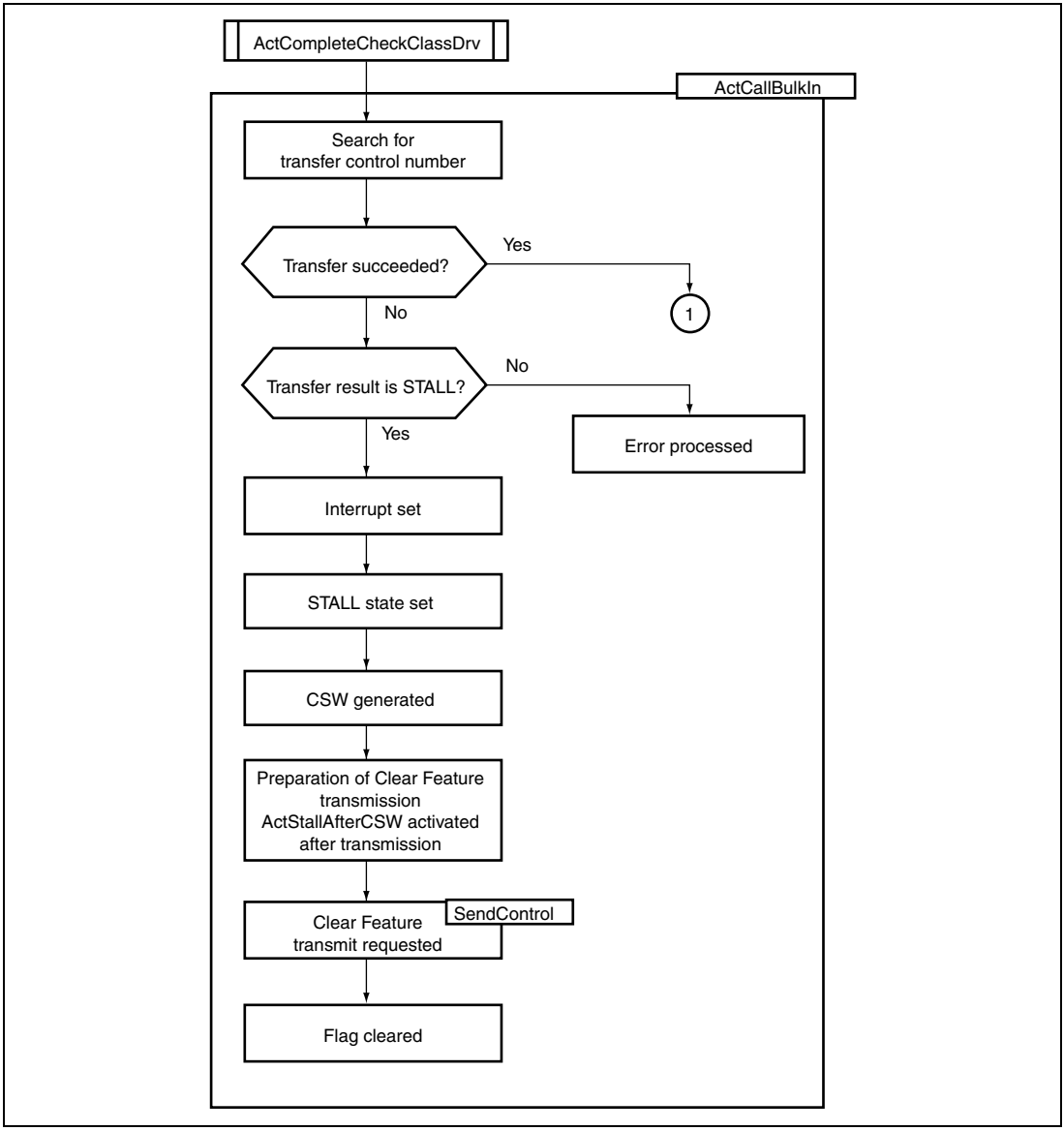

Figure 5.25-3 Functions for Process after Bulk-IN Transfer Completed (1)

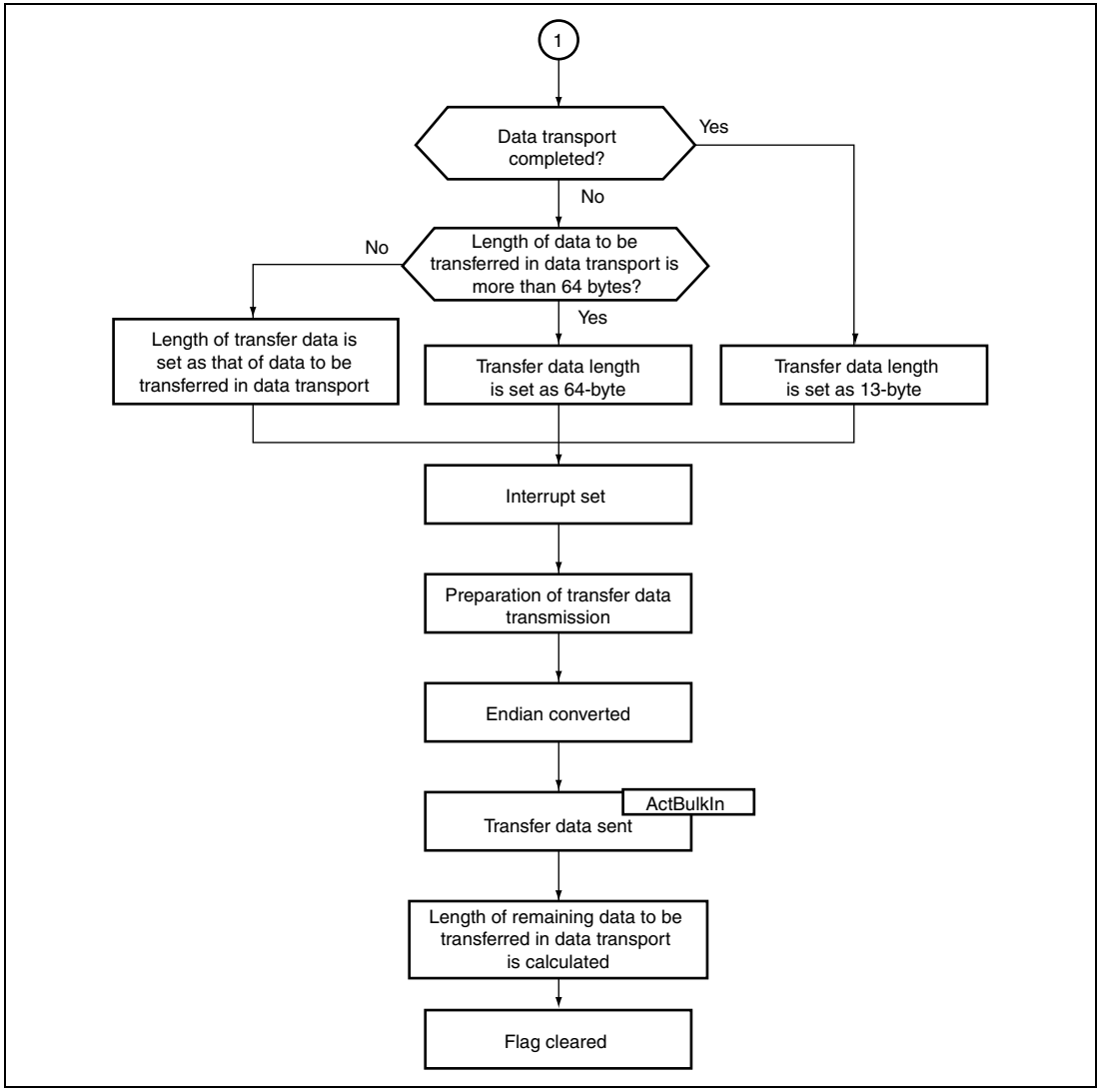

Figure 5.25-4 Functions for Process after Bulk-IN Transfer Completed (2)

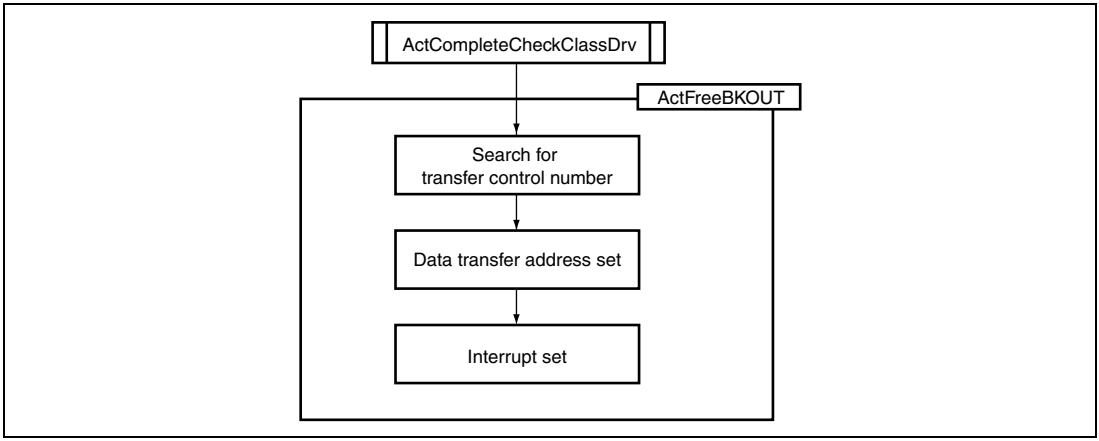

Figure 5.25-5 Functions for Process after Data Transport (Data Stage is in Out Direction) Completed

SH7727 USB Host Module Application Note (Advanced)

Publication Date: Rev.1.00, April 15, 2003
Published by: Sales Strategic Planning Div.
Renesas Technology Corp.
Edited by: Technical Documentation & Information Department
Renesas Kodaira Semiconductor Co., Ltd.

SH7727 Group USB Host Module Application Note (Advanced)

Renesas Electronics Corporation
1753, Shimonumabe, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8668 Japan

REJ05B0016-0100Z