SH7216 Group

USB Multifunction Operation of USB Function Module

R01AN0294EJ0110
Rev.1.10
Mar 17, 2011

Introduction

This application note describes the use of the on-chip USB function module of the SH7216 through the example of a sample program that supports both the HID class and mass storage class.

The contents of this application note and the software are intended to illustrate the operation of the USB function module and are not guaranteed to be suitable for practical application.

Target Device

SH7216

Contents

1. Preface

1.1 Specifications

Using the USB function module of the SH7216, the USB Human Interface Device (HID) class is used to manipulate the mouse pointer on a PC and the USB mass storage class is used to manipulate local disk files.

1.2 Functions Used

- Interrupt controller (INTC)
- Multifunction timer pulse unit 2 (MTU2)
- Pin function controller (PFC)
- USB function module (USB)

1.3 Applicable Conditions

MCU	SH7216
Operating frequency	Internal clock: 200 MHz Bus clock: 50 MHz Peripheral clock: 50 MHz
Integrated development environment	Renesas Electronics High-performance Embedded Workshop, Ver. 4.07.00.007
C compiler	Renesas Electronics SuperH RISC engine Family C/C++ Compiler Package, Ver. 9.03, Release 02
Compile options	High-performance Embedded Workshop default settings (-cpu=sh2afpu -pic=1 -object="$(CONFIGDIR)¥$(FILELEAF).obj" -debug -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 –nologo)

1.4 Related Application Notes

- SH7216 Group Application Note: USB Function Module: USB Mass Storage Class (REJ06B0897)
- SH7216 Group Application Note: USB function module: USB HID Class (REJ06B0898)

2. Overview

The sample program supports use of the USB function module (USB) to perform control transfer, bulk transport, and interrupt transfer operations, and USB multifunction operations (HID class commands and mass storage class commands).

The features of the on-chip USB function module of the SH7216 are as follows.

- Automatic processing of USB protocol
- Automatic processing of USB standard commands for endpoint 0 (Some commands need to be processed through the firmware.)
- Transfer speed: Full speed
- Interrupt requests: Generation of interrupt signals needed for USB transmission and reception
- Clock: External input clock generated by USB oscillator (48 MHz)
- Low-power mode
- Integrated bus transceiver
- Endpoint configurations: Shown in table 1.

Table 1 Endpoint Configurations

Endpoint	Name	Transfer Type	Max. Packet Size	FIFO Buffer Capacity	DMA Transfer
Endpoint 0	EP0s	Setup	8 bytes	8 bytes	—
	EP0i	Control-in	16 bytes	16 bytes	—
	EP0o	Control-out	16 bytes	16 bytes	—
Endpoint 1	EP1	Bulk-in	64 bytes	64 × 2 (128) bytes	Possible
Endpoint 2	EP2	Bulk-out	64 bytes	64 × 2 (128) bytes	Possible
Endpoint 3	EP3	Interrupt-in	16 bytes	16 bytes	—
Endpoint 4	EP4	Bulk-in	64 bytes	64 × 2 (128) bytes	Possible
Endpoint 5	EP5	Bulk-out	64 bytes	64 × 2 (128) bytes	Possible
Endpoint 6	EP6	Interrupt-in	16 bytes	16 bytes	—
Endpoint 7	EP7	Bulk-in	64 bytes	64 bytes	—
Endpoint 8	EP8	Bulk-out	64 bytes	64 bytes	—
Endpoint 9	EP9	Interrupt-in	16 bytes	16 bytes	—

Figure 1 shows an example system configuration.

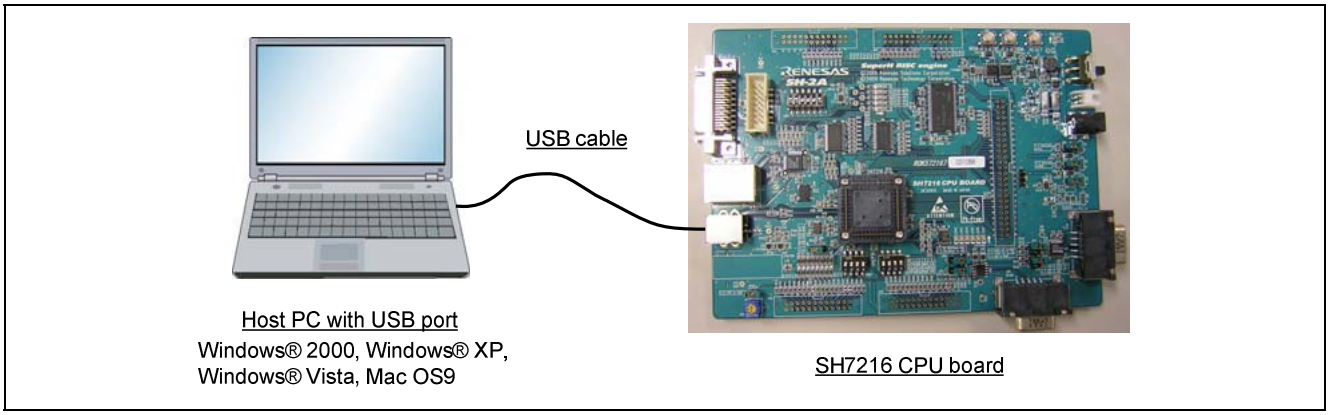


Figure 1 System Configuration Example

The system comprises the SH7216 CPU board from Renesas Electronics and a personal computer running the Windows® 2000, Windows® XP, Windows® Vista, or Mac OS9 operating system.

In the system, the host PC and SH7216 CPU board are connected by a USB cable. The firmware uses the HID class to automatically generate pseudo mouse data and uses the mass storage class to perform USB multifunction operations with a RAM disk.

The USB HID class device drivers and USB mass storage class (bulk-only transport) device drivers provided as standard components of the above operating systems may be used with the sample program.

The features of the system are as follows.

1. The sample program can be used to evaluate the USB module of the SH7216.
2. The sample program supports USB control transfer, interrupt transfer, and bulk transport.
3. The system can be debugged with the E10A (USB emulator).

Note: The SH7216 does not support isochronous transfer.

3. Overview of USB Multifunction Processing

The sample program performs USB multifunction processing using the USB mass storage class (bulk-only transport) and the USB human interface device (HID) class.

3.1 Overview of USB HID Class

The USB HID class is described below.

Use the description that follows as reference when developing USB HID class devices. For details on USB standards, see documents (4) and (5) listed in section 6, Reference Documents.

3.1.1 USB HID Class

The USB HID is a class of standards that apply to devices used by people to interact with personal computers. Examples of such devices include mouses, keyboards, and joysticks.

To notify the host PC that a function is of this class, the bInterfaceClass field of the Interface descriptor must have a value of H'03.

3.1.2 Subclass Code

Subclasses were originally devised to identify the specific protocols of different HID class devices, but since people use many different kinds of devices, subclass protocol definitions are not practical. The HID class therefore does not use subclasses to define most protocols. Instead, the Report descriptor is used to determine the protocol for HID class devices.

In the case of devices with BIOS support (boot devices), however, a simple method to identify the protocol is necessary. Therefore, subclasses are used to indicate HID class devices that support a predefined protocol (boot protocol) for mouse devices or keyboards (that is, devices that can be used as boot devices).

To notify the host PC that the device supports the boot protocol, the value of the bInterfaceSubClass field of the Interface descriptor must be H'01.

3.1.3 Protocol Code

When a device supports the boot protocol (subclass code other than 0), a protocol code is used to indicate the device type. The protocol code is H'01 for a keyboard and H'02 for a mouse. Specifying the device type by a protocol code indicates that the device can use the protocol for that device type.

To notify the host PC of the device type, the bInterfaceProtocol field of the Interface descriptor must have a value corresponding to the device type.

3.1.4 Descriptors for HID Class

HID class function devices need an HID descriptor, a Report descriptor, and a Physical descriptor (optional) in addition to descriptor information that other USB function devices need. Figure 2 shows the HID device descriptor configuration.

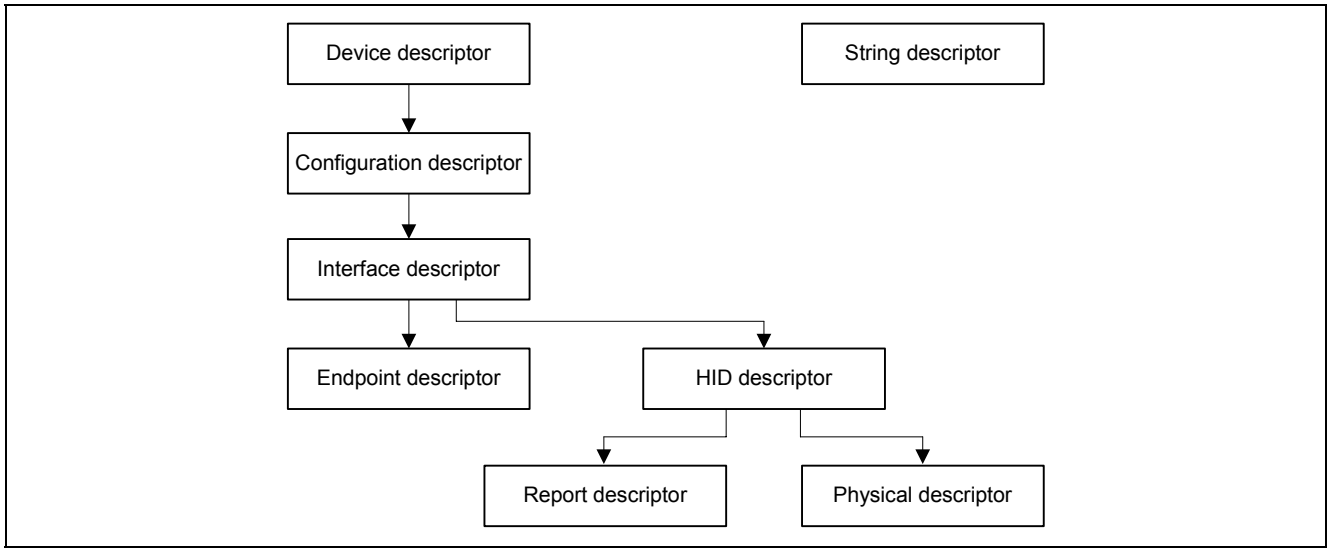


Figure 2 Descriptor Configuration

3.1.5 HID Descriptor

The purpose of the HID descriptor is to combine the Report descriptor and Physical descriptor (optional). Table 2 shows the format of the HID descriptor.

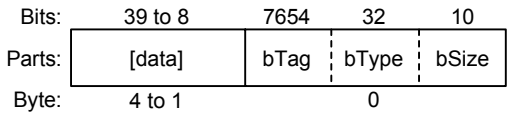
Table 2 HID Descriptor

Field	Size (Byte)	Description
bLength	1	Descriptor size (fixed at H'09)
bDescriptorType	1	Descriptor type (fixed at H'21)
bcdHID	2	HID version expressed in BCD
bCountryCode	1	Country ID for devices specific to a particular country (0 unless necessary)
bNumDescriptors	1	Number of class descriptors
bDescriptorType	1	Class descriptor type (H'22 for HIDREPORT)
wDescriptorLength	2	Report descriptor size

3.1.6 Report Descriptor

The Report descriptor specifies the format of data to be transferred between the host PC and the device. Unlike other descriptors, the Report descriptor has no standardized format, but the length and contents of the Report descriptor vary depending on the information the device is reporting or the number of data fields required for the device's report.

The Report descriptor consists of items that provide information about the device. There are two types, short and long items. The short item type is described below.



bType and bSize indicate the following meanings.

Item Type	
bType Value	Type
00	Main
01	Global
10	Local
11	Reserve

Item Size	
bSize Value	Data Size
00	0 byte
01	1 byte
10	2 bytes
11	4 bytes

Figure 3 Report Descriptor Items

An item consists of four fields: data, item tag, item type, and itemSize. The item uses these fields to indicate the variety of information.

There are three item types: Main, Global, and Local. The Main item type (for defining or grouping the data fields in a Report descriptor) has five types of item tags, the Global item type (for describing data) has 12, and the Local item type (for defining characteristics) has ten.

By combining these item tags, the Report descriptor specifies the format of the data to be transferred between the host PC and the device.

(1) Main Items

Table 3 lists the five item tags for the Main item type.

Table 3 Item Tags of Main Item Type

item tag	bTag	bType	bSize	Description
Input	1000	00	nn	Describes information about data provided by one or more physical controls.
Output	1001	00	nn	Defines the output data field.
Feature	1011	00	nn	Describes device configuration information that can be sent to the device.
Collection	1010	00	nn	Starts collecting relations between two or more data item tags (Input, Output, or Feature).
End Collection	1100	00	nn	Ends collecting relations between two or more data item tags (Input, Output, or Feature) in response to Collection.

(a) Input Item Tag

The Input item tag has eight parameters (data fields), which are set in one-bit units, as shown in table 4.

Table 4 Parameters of Input Item Tag

Bit	Value	Contents	Description
0	0	Data	The item reports data.
	1	Constant	The item reports a constant.
1	0	Array	The item reports an array data field.
	1	Variable	The item reports a variable.
2	0	Absolute	The item reports an absolute value.
	1	Relative	The item reports deviation relative to the last report.
3	0	No Wrap	The value reported by the item does not roll over.
	1	Wrap	The value reported by the item rolls over. (For example, if a dial outputs values from 0 to 10, 0 is output after 10 when dialing continues.)
4	0	Linear	The item reports the state of the target control in linear fashion.
	1	Non Linear	The item processes raw data and does not report the state of the target in linear fashion.
5	0	Preferred State	The item has a state to which it returns when it is not controlled by the user.
	1	No Preferred	The item does not have a state to which it returns when it is not controlled by the user.
6	0	No Null position	The item has a state in which it does not send meaningful data.
	1	Null state	The item does not have a state in which it does not send meaningful data.
7	0	Reserved	Reserved
8	0	Bit Field	The item issues a bit field.
	1	Buffered Bytes	The item issues a stream fixed at one-byte size.
9 to 31	0	Reserved	Reserved

(b) Output and Feature Item Tags

The Output and Feature item tags have nine parameters (data fields), which are the same as the Input item tag except for bit 7. Table 5 lists the parameters of the Output and Feature item tags.

Table 5 Parameters of Output and Feature Item Tags

Bit	Value	Contents	Description
1 to 6	—	—	Same as input item tag.
7	0	Non Volatile	The item value cannot change independent of host interaction.
	1	Volatile	The item value can change independent of host interaction.
8 to 31	—	—	Same as input item tag.

(c) Collection Item Tag

The Collection item tag has eight parameters (data fields), which are specified as one-byte values. Table 6 lists the parameters.

Table 6 Parameters of Collection Item Tag

Value	Contents	Description
H'00	Physical	Used for data items collected into one. This is used for devices that need to associate precise or sensed data with a single point. It does not indicate that the data comes from a single device such as a keyboard. It indicates that the data comes from different sensors, as in the case of a device that reports output from multiple sensor positions.
H'01	Application	Identifies a Usage only used at the application level. It indicates that the collection is a functionally subordinate group of an HID device or a complex device. The operating system uses the Usage associated with this collection to link to the application or driver that controls the device.
H'02	Logical	Used when data items compose a composite data structure.
H'03	Report	Defines a logical collection that includes all fields. A report ID is included in this collection. This enables an application to easily determine whether to support a certain function of the device.
H'04	Named Array	Used when data items compose a composite data structure and it is named.
H'05	Usage Switch	A logical collection that modifies the meaning of the Usage in which it is included. It identifies the Usage applied for logical collection to modify the purpose of the Usage being collected.
H'06	Usage Modifier	Modifies the meaning of the Usage attached to the collection in which it is included. The Usage typically defines a single operating mode for control. This enables the operating method of control to be expanded.
H'07 to H'7F	Reserved	Reserved
H'80to H'FF	Vendor-defined.	Defined by the vendor.

(2) Global Items

Table 7 shows 12 item tags for the Global item type.

Table 7 Item Tags of Global Item Type

Item Tag	bTag	bType	bSize	Description
Usage Page	0000	01	nn	A value specifying the current Usage Page. It defines the index to the item Usage.
Logical Minimum	0001	01	nn	The minimum value to be reported by a variable or array item. For example, a mouse that reports an X position value from 0 to 128 will have a minimum logical value of 0.
Logical Maximum	0010	01	nn	The maximum value to be reported by a variable or array item. For example, a mouse that reports an X position value from 0 to 128 will have a maximum logical value of 128.
Physical Minimum	0011	01	nn	The minimum value of physical range for a variable item.
Physical Maximum	0100	01	nn	The maximum value of physical range for a variable item.
Unit Exponent	0101	01	nn	The unit exponent in base 10.
Unit	0110	01	nn	The unit value.
Report Size	0111	01	nn	Unsigned value that specifies the report field size in bits.
Report ID	1000	01	nn	Unsigned value that specifies the report ID.
Report Count	1001	01	nn	Specifies the number of data fields for the item. An unsigned integer specifies how many fields can be included in the report for the particular item (that is, how many bits are added to the report).
Push	1010	01	nn	Places a copy of the Global item state table on the stack.
Pop	1011	01	nn	Replaces the item state table with the top data on the stack.

(3) Local Items

Table 8 shows 10 item tags for the Local item type.

Table 8 Item Tags of Local Item Type

Item Tag	bTag	bType	bSize	Description
Usage	0000	10	nn	A value specifying the current Usage. It defines the index to the item Usage.
Usage Minimum	0001	10	nn	Defines the start of Usage associated with an array or a bitmap.
Usage Maximum	0010	10	nn	Defines the end of Usage associated with an array or a bitmap.
Designator Index	0011	10	nn	Determines the body part used for control.
Designator Minimum	0100	10	nn	Defines the start index to the designator associated with an array or a bitmap.
Designator Maximum	0101	10	nn	Defines the end index to the designator associated with an array or a bitmap.
String Index	0111	10	nn	Index to the String descriptor, which enables the string to be associated with a particular item or control.
String Minimum	1000	10	nn	Specifies the first string index when associating a group of sequential strings to a control in an array or a bitmap.
String Maximum	1001	10	nn	Specifies the end string index when associating a group of sequential strings to a control in an array or a bitmap.
Delimiter	1010	10	nn	Defines the start or end of a set of Local items.

(4) Sample Report Descriptor

Figure 4 shows the Report descriptor of the sample program.

```
Usage Page (Generic Desktop),                              : 05 01
Usage (Mouse),                                             : 09 02
Collection (Application),                                  : A1 01
    Usage (Pointer),                                       : 09 01
    Collection (Physical),                                 : A1 00
        Usage Page (Buttons),                              : 05 09
        Usage Minimum (01),                                : 19 01
        Usage Maximum (03),                                : 29 03
        Logical Minimum (0),                               : 15 00
        Logical Maximum (1),                               : 25 01
        Report Count (3),                                  : 95 03
        Report Size (1),                                   : 75 01
        Input (Data, Variable, Absolute), ; 3 button bits  : 81 02
        Report Count (1),                                  : 95 01
        Report Size (5),                                   : 75 05
        Input (Constant), ; 5 bit padding                  : 81 01
        Usage Page (Generic Desktop),                      : 05 01
        Usage (X),                                         : 09 30
        Usage (Y),                                         : 09 31
        Usage (Wheel),                                     : 09 38
        Logical Minimum (-127),                            : 15 81
        Logical Maximum (127),                             : 25 7F
        Report Size (8),                                   : 75 08
        Report Count (3),                                  : 95 03
        Input (Data, Variable, Relative), ; 2 position bytes (X & Y) : 81 06
    End Collection,                                        : C0
End Collection                                             : C0
```

Figure 4 Report Descriptor

(5) Description of Report Descriptor

Table 9 shows the Report descriptor used by the sample program.

Table 9 Report descriptor

Item	Value (Hex)	Item Classification	Description
Usage Page (Generic esktop Control)	H'05 01	Global	A value specifying the Usage Page. H'01 indicates Generic Desktop Control.
Usage (Mouse)	H'09 02	Local	Index to the item Usage. H'02 indicates Mouse. The operating system links the device as a mouse to the active application or driver. The Usage type of Mouse is Collection Application.
Collection (Application)	H'A1 01	Main	Notifies the application of Pointer as a mouse.
Usage (Pointer)	H'09 01	Local	Index to the item Usage. H'01 indicates Pointer. The Usage type of Pointer is Collection Physical.
Collection (Physical)	H'A1 00	Main	Collects multiple sensor positions (button, X axis, Y axis, and rotary control) to one as a pointer.
Usage Page (Button)	H'05 09	Global	A value specifying the Usage Page. H'09 indicates Button.
Usage Minimum (1)	H'19 01	Local	Defines that the Usage associated with an array or a bitmap starts from 1.
Usage Maximum (3)	H'29 03	Local	Defines that the Usage associated with an array or a bitmap ends at 3.
Logical Minimum (0)	H'15 00	Global	The minimum value to be reported by the item is 0.
Logical Maximum (1)	H'25 01	Global	The maximum value to be reported by the item is 1.
Report Count (3)	H'95 03	Global	Indicates the number of data fields to be used for the item. This example indicates that three report fields are to be used.
Report Size (1)	H'75 01	Global	Indicates the report field size. This example indicates that one-bit fields are to be used.
Input (Data, Variable, Absolute)	H'81 02	Main	Indicates the type of input item. This example indicates that the input is variable data and reports an absolute value.
Report Count (1)	H'95 01	Global	Indicates the number of data fields to be used for the item. This example indicates that one report field is to be used.
Report Size (5)	H'75 05	Global	Indicates the report field size. This example indicates that five-bit fields are to be used.
Input (Constant)	H'81 01	Main	Indicates the number of data fields to be used for the item. This example indicates that the input reports a constant.
Usage Page (Generic Desktop Control)	H'05 01	Global	A value specifying the Usage Page. H'01 indicates Generic Desktop Control.
Usage (X)	H'09 30	Local	Index to the item Usage. H'30 indicates X. The controller reports X-direction values, and when the controller moves from left to right from the user's viewpoint, the value increases in linear fashion.

Item	Value (Hex)	Item Classification	Description
Usage (Y)	H'09 31	Local	Index to the item Usage. H'31 indicates Y. The controller reports Y-direction values, and when the controller moves from far to near from the user's viewpoint, the value increases in linear fashion.
Usage (Wheel)	H'09 38	Local	Index to the item Usage. H'38 indicates Wheel. Unlike a dial, this is a rotary control that generates a variable value when rotated. When the controller rotates toward the front (farther from the user), the value increases.
Logical Minimum (–127)	H'15 81	Global	The minimum value to be reported by the item is –127.
Logical Maximum (127)	H'25 7F	Global	The maximum value to be reported by the item is 127.
Report Size (8)	H'75 08	Global	Indicates the report field size. This example indicates that eight-bit fields are to be used.
Report Count (3)	H'95 03	Global	Indicates the number of data fields to be used for the item. This example indicates that three report fields are to be used.
Input (Data, Variable, Relative)	H'81 06	Main	Indicates the type of input item. This example indicates that the input is variable data and reports the change from the last input.
End Collection	H'C0	Main	Indicates the end of collection as a data set (physical).
End Collection	H'C0	Main	Indicates the end of collection as a data set (application).

3.1.7 Physical Descriptor

The Physical descriptor provides information about the human body (or a specific part of the human body) that is controlling the device. This descriptor is optional, and it is omitted in the sample program.

3.1.8 HID Data Transfer Format

HID data is transferred between the host PC and the USB function module mainly through interrupt transfers (control transfers are also available).

A boot device can use two types of protocols: report protocol and boot protocol. Other devices can only use the report protocol.

The format of data transfer used by the report protocol is described by a Report descriptor.

The format used by the boot protocol is prescribed in the USB standard.

The default protocol for a boot device is the report protocol, but a class command can select either the boot or report protocol. Figure 5 shows the report protocol format used by the sample program.

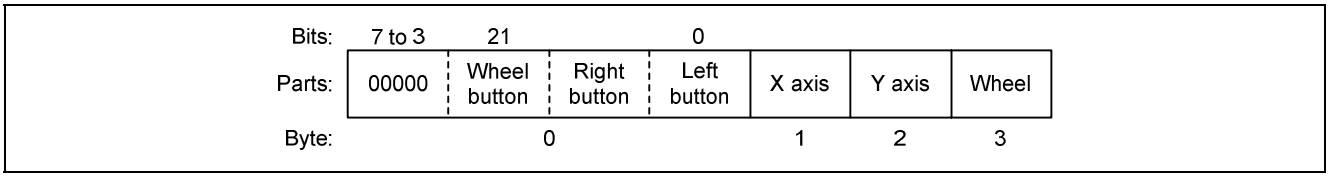


Figure 5 Report Protocol Format

3.1.9 Class Commands

Class commands are defined for each USB class. They use control transfer.

There are six commands for the USB HID class. Table 10 lists the class commands.

Table 10 Class Commands

bRequest Field Value	Command	Description of Command
H'01	GET_REPORT	Transfers HID data from the device to the host PC by using control transfer.
H'02	GET_IDLE	Returns the current value for the rate of time for which interrupt transfer stops.
H'03	GET_PROTOCOL	Reports the current active protocol (boot protocol or report protocol).
H'09	SET_REPORT	Transfers HID data from the host PC to the device by using control transfer.
H'0A	SET_IDLE	Specifies the rate of time for which interrupt transfer stops.
H'0B	SET_PROTOCOL	Specifies the active protocol (boot protocol or report protocol).

Notes: 1. All devices must support GET_REPORT.
2. Boot devices must support GET_PROTOCOL and SET_PROTOCOL.

When the GET_REPORT command is received, the function sends HID data to the host through the data stage of control transfer. The report type must be specified in the uppermost byte in the wValue field in the setup data and the report ID in the lowermost byte in the wValue field. A value of 0 is specified when no report ID is used.

When the GET_IDLE command is received, the function returns the duration for which interrupt transfer stops. The duration should be expressed as a time rate in 4 ms units. The host specifies the report ID in the lowermost byte in the wValue field in the setup data. If this value is 0, the time rates for all interrupt transfers of the target device are returned.

When the GET_PROTOCOL command is received, the function returns the current active protocol (boot protocol or report protocol) to the host through the data stage of control transfer. A value of 0 indicates the boot protocol and a value of 1 indicates the report protocol.

When the SET_REPORT command is received, the function receives HID data through the data stage of control transfer. However, the function may ignore the command from the host.

When the SET_IDLE command is received, the function stops interrupt transfer for the specified duration. The duration is specified by the uppermost byte of the wValue field in the setup data. The duration is expressed as a time rate in 4 ms units. The lowermost byte of the wValue field specifies the report ID. If this value is not 0, the transfer of the specified report ID is stopped. If this value is 0, all interrupt transfers of the target device are stopped.

When the SET_PROTOCOL command is received, the function specifies the protocol (boot protocol or report protocol) to be used from that time on. The protocol is specified by the wValue field in the setup data (a value of 0 indicates the boot protocol and a value of 1 indicates the report protocol). Note that the report protocol is the default protocol of the function.

3.2 Overview of USB Mass Storage Class (Bulk-Only Transport)

This section describes the USB mass storage class.

Use the description that follows as reference when developing systems related to USB storage. For details on USB standards, see documents (2) and (3) listed in section 6, Reference Documents.

3.2.1 USB Mass Storage Class

The USB mass storage class is a class of standards that apply to large-scale storage devices that are connected to a host PC and handle reading and writing of data.

To notify the host PC that a function is of this class, the bInterfaceClass field of the Interface descriptor must have a value of H'08. Furthermore, the USB mass storage class must tell the host the serial number using the String descriptor. Unicode 000000000001 is returned in the sample program.

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

The USB mass storage class has the following two data transport protocols.

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

As the name implies, USB mass storage class bulk-only transport is a data transfer protocol that only uses bulk transport.

USB mass storage class control/bulk/interrupt (CBI) transport is a data transfer protocol that uses control transfer, bulk transport, and interrupt transfer. CBI transport is further subdivided into a data transport protocol that uses interrupt transfer and one that does not use interrupt transfer.

The sample program provided here uses USB mass storage class bulk-only transport as the data transfer protocol.

When the host PC uses a device in order to load and save data, instructions (commands) are provided by the host PC to the function. The function then executes the commands sent to it to load and save data. The commands sent by the host PC to the function are defined in the form of subclass codes.

3.2.2 Subclass Codes

Subclass codes are values that indicate the command format sent from the host PC to a function by means of command transport. There are seven command formats. The command formats are listed in table 11.

Table 11 Subclass Codes

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

In order to tell the host PC the command format supported by the device, a subclass code value must be entered in the bINterface SubClass field of the Interface descriptor.

The sample program uses a sub-class code value of H'06, which indicates SCSI primary commands.

3.2.3 Bulk-Only Transport

Bulk-only transport uses only bulk transport to move data between the host PC and a function.

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

Bulk-only transport uses a predetermined combination of bulk-out and bulk-in transport to move data between the host and the function. Bulk-only transport always uses the combination of bulk transport methods shown in figure 2. These bulk transport methods have different meanings, and they are handled as stages (transports).

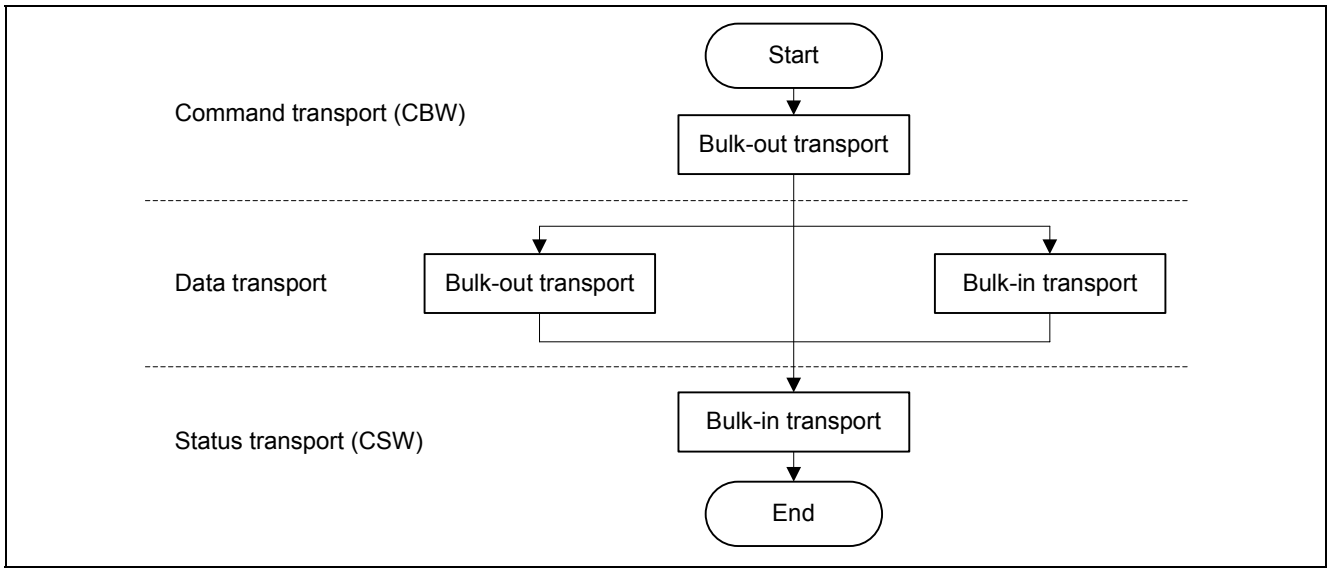


Figure 6 Correspondence between Transfer Methods and Transports

In order to tell the host PC that the bulk-only transport protocol is being used, a value of H'50 must be entered in the bInterfaceProtocol field of the Interface descriptor.

(1) Command Transport

In command transport, commands are sent from the host PC to the function using bulk-out transport. The command packet is defined as a command block wrapper (CBW), and bulk-only transport must always begin with a CBW.

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

The contents of the CBW are in the format shown in table 12.

Table 12 Command Transport Format

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

Each field is described below.

- dCBWSignature:
  This field identifies the data packet as a CBW. The value is H'43425355 (little endian).
- dCBWTag:
  This is the command block tag. It is used to connect the CBW with its corresponding CSW, and is specified by the host PC.
- dCBWDataTransferLength:
  This is the length of the data planned for transport. If this value is 0, no data transport exists.
- bmCBWFlags:
  If bit 7 of this field is 0, data is transported using bulk-out transport, and if it is 1, bulk-in transport is used. Bits 0 to 6 are fixed at 0.
- bCBWLUN:
  This is the logical unit number of the device sending the command block.
- bCBWCBLength:
  This indicates the number of valid bytes in the next field (CBWCB).
- CBWCB:
  This field stores the command block to be executed by the function. The command that the host PC wants to execute (a SCSI command in this sample program) is entered in this field.

(2) Status Transport

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

The status packet is defined as a command status wrapper (CBW), and bulk-only transport must always end with a CSW.

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

The contents of the CSW are in the format shown in table 13.

Table 13 Status Transport Format

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

Each field is described below.

- dCSWSignature:
  This field identifies the data packet as a CSW. The value is H'53425355 (little endian).
- dCSWTag:
  This is the command block tag. It ties the CBW to the CSW, and the same value is entered here as that of the dCBWTag field in the CBW.
- dCSWDataResidue:
  This reports the difference between the value of the dCBWDataTransferLength field of the CBW and the actual amount of data processed by the function.
- bCSWStatus:
  This indicates whether or not a command has been successfully executed. If the command was executed successfully, the function sets this field to H'00. A value other than 0 indicates that the command was not executed successfully, as follows: H'01 indicates a failed command, and H'02 indicates a phase error.

(3) Data Transport

Data transport is used to transfer data between the host PC and the function. For example, with the Read and Write commands, the actual data of the various storage sectors is sent using data transport.

Data transport is composed of multiple bus transactions.

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

(1) Data transport (bulk-out transport)
Bulk-out data transport works as follows.
This status is set if the value of bit 7 in the bmCBWFlags field of the CBW data is 0 and the value of the dCBWDataTransferLength field of the CBW data is not 0.
Here, the function receives data of the anticipated length as indicated by the dCBWDataTransferLength field in the CBW data. The data transferred at this point is needed when the SCSI command specified by the CBWCB field of the CBW data is executed.

(2) Data transport (bulk-in transport)
Bulk-in data transport works as follows.
This status is set if the value of bit 7 of the bmCBWFlags field of the CBW data is 1 and the value of the dCBWDataTransferLength field in the CBW data is not 0.
Here, data of the anticipated length as indicated by the dCBWDataTransferLength field of the CBW data is sent to the host PC. The data transferred at this point is the result produced when the SCSI command specified by the CBWCB field of the CBW data was executed.

3.2.4 Class Commands

Class commands are defined for each USB class. They use control transfer.

When USB mass storage class bulk-only transport is used as the data transfer protocol, there are two commands that must be supported. Table 14 lists the class commands.

Table 14 Class Commands

bRequest Field Value	Command	Description of Command
255 (H'FF)	Bulk-Only Mass Storage Reset	Resets the interface.
254 (H'FE)	Get Max LUN	Resets the interface. Checks the number of LUNs supported.

When the Bulk-Only Mass Storage Reset command is received, the function resets all of the interfaces used for USB mass storage class bulk-only transport.

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

3.2.5 Subclass Codes (SCSI Transparent Command Set)

The function must process the commands corresponding to the subclass commands in the CBWs sent to the function by the host PC.

Of the SCSI commands, the sample program supports the eleven commands shown in table 15. When a command is not supported, the CSW is used to inform the host PC that the command failed.

Table 15 Supported Commands

Operation Code	Command	Command Operation
H'00	TEST UNIT READY	Checks whether or not the media can be used.
H'03	REQUEST SENSE	If an error was generated by the previous command, this tells the host what kind of error occurred.
H'12	INQUIRY	Tells the host information about the drive.
H'1A	MODE SENSE (6)	Tells the host the drive status.
H'1B	STOP/START UNIT	Controls insertion/removal of media.
H'1E	PREVENT ALLOW MEDIUM REMOVAL	Disables/enables installing and removing media.
H'23	READ FORMAT CAPACITY	Tells the host the media format information.
H'25	READ CAPACITY	Tells the host the media sector information.
H'28	READ (10)	Reads the sector volume data from a specified sector.
H'2A	WRITE (10)	Writes the sector volume data to a specified sector.
H'2F	VERIFY (10)	Confirms whether or not the data in the media can be accessed.

4. Development Environment

This section describes at the development environment used to develop the system. The devices (tools) listed below are used for system development.

- SH7216 CPU board (product number R0K572167) manufactured by Renesas Electronics Corporation
- E10A-USB emulator manufactured by Renesas Technology Electronics Corporation
- E10A PC (Windows® 2000 or Windows® XP)
- USB host PC (Windows® 2000, Windows® XP, or Windows® Vista)
- USB cable
- High-performance Embedded Workshop 4 (HEW4) manufactured by Renesas Technology Electronics Corporation

4.1 Hardware Environment

Figure 7 shows the connections between the devices.

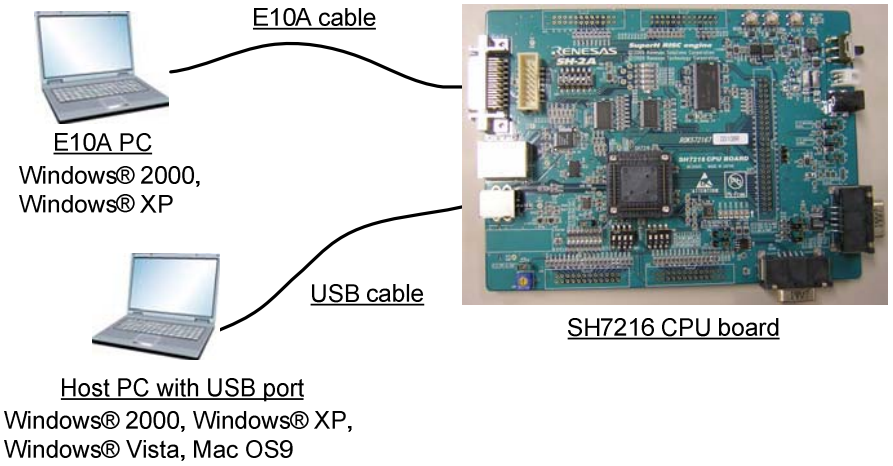


Figure 7 Device Connections

(1) SH7216 CPU board

Because the system uses the on-chip ROM and SDRAM, the SH7216 CPU board must be operated in MCU expansion mode 2 (both on-chip ROM and SDRAM enabled). Therefore, DIP switch SW1 on the SH7216 CPU board must be changed from the factory setting to the setting shown in table 16. Before powering on the board, ensure that the switches are set as shown. There is no need to change any other DIP switches.

Table 16 DIP Switch Settings

Factory Setting (Mode 6)	After Change (Mode 2)	DIP Switch Function
SW1-1 (FWE) OFF	SW1-1 (FWE) ON	On-chip flash memory write/erase protection
SW1-2 (MD1) OFF	SW1-2 (MD1) OFF	MD1 pin state
SW1-3 (MD0) ON	SW1-3 (MD0) ON	MD0 pin state

(2) USB host PC

A PC with a USB port and with Windows® 2000/Windows® XP/Windows® Vista or Mac OS9 installed is used as the USB host. The system uses the USB mass storage class (bulk-only transport) device drivers installed as a standard part of the above operating systems, and so there is no need to install new drivers.

(3) E10A PC

A PC with a USB port and with Windows® 2000/Windows® XP installed is used as the E10A PC. Connect the E10A-USB emulator to the USB connector on the E10A-USB PC, and then connect the E10A-USB emulator to the CPU board with the cable. After making connections, start the HEW4 and perform emulation.

4.2 Software Environment

Compile, link, and debug the source code with HEW4. To start HEW4, double-click **USB_MULTI.hws** in the sh7216_usb_multi folder.

4.2.1 Sample Program

The files required by the sample program are all stored in the sh7216_usb_multi folder. Once this folder is moved with its contents unchanged to a PC on which HEW4 is installed, the sample program can be used immediately.

The files contained in the folder are shown in figure 8.

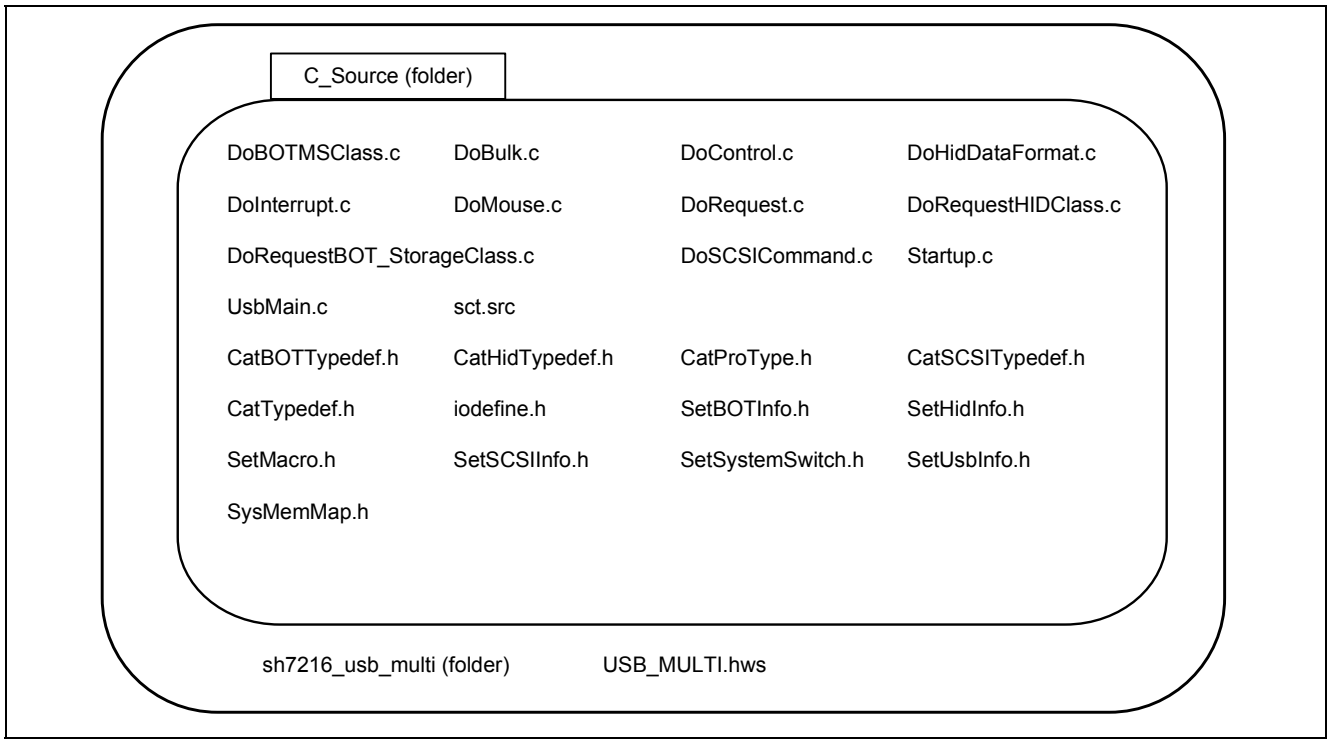


Figure 8 File Structure

4.2.2 Compiling and Linking

Compile the source code with HEW4.

4.3 Loading and Executing the Program

The procedure for loading and executing the program is described below.

4.3.1 Loading the Program

The procedure for loading the sample program into the SH7216 CPU board is as follows.

- Connect the E10A-USB to the E10A-USB PC, on which HEW4 is installed.
- Connect the E10A-USB to the SH7216 CPU board with the user cable.
- Power on the SH7216 CPU board.
- Execute **USB_MULTI.hws** in the sh7216_usb_multi folder.
- Select **Debug** > **Connection**.
- When prompted to select an emulator mode, select **SH7216 (R5F72167A)** or **E10A-USB Emulator**.
- Press the reset switch on the SH7216 CPU board, then click the **OK** button.
- When prompted to select an operating frequency, enter the frequency (12.50 MHz) of the installed crystal resonator.
- When prompted to enter an ID code, enter E10A.
- Select **Debug** > **Download** > **All Download Modules** to download the sample program to the SH7216 CPU board.

4.3.2 Executing the Program

Select **Debug** > **Execute after Reset** to run the sample program.

4.4 Executing USB Multifunction Processing

While the program is running, connect the series-B connector of the USB cable to the SH7216 CPU board and the series-A connector to the USB host PC.

The sample program uses USB multifunction processing to run a mouse pointer movement demonstration on the host PC by using the USB HID class and to display a USB mass storage device by using the USB mass storage class.

4.4.1 Mouse Pointer Movement Demonstration

After connections are made, the human interface devices and USB human interface devices are displayed in the Device Manager window by control transfer, and the host PC recognizes the SH7216 CPU board as a mouse device. Next, the sample program demonstrates movements of the host PC mouse pointer.

The SH7216 CPU board transmits mouse pointer movement data in response to interrupt-in transfers from the host PC. This causes the mouse pointer on the USB host PC to start moving automatically.

4.4.2 USB Mass Storage Device Display

After emulation using control transfer and bulk transport ends, the USB mass storage devices are displayed under USB Controllers in the Device Manager window, and the host PC recognizes the SH7216 CPU board as a storage device.

RENESAS EX RAM Disk USB Device is displayed under Disk Drives and mounted as a local disk in My Computer.

The next step is to format the local disk.

Select the local disk, right-click it with the mouse, and select Format from the contextual menu. When the Format Selection menu for the drive appears, enter the formatting settings. Confirm that FAT is selected as the file system, then click the Start button.

When the window to confirm the formatting operation appears, click the OK button.

A message window is displayed when formatting completes. Click OK to dismiss it.

When the Format Selection menu for the drive reappears, click the Close button to dismiss it.

Once the above steps are completed, the SH7216 CPU can be used as a RAM disk via the USB connection.

5. Overview of Sample Program

This section describes the features of the sample program and its structure. The sample program is firmware that runs on the SH7216 CPU board and performs operations using the USB mass storage and HID classes. USB data transfers are initiated by interrupts from the USB function module.

Of the interrupts from the on-chip modules of the SH7216, six are related to the USB function module: USI0, USI1, USBRXI0, USBTXI0, USBRXI1, and USBTXI1. The sample program uses only USI0 and USI1.

5.1 State Transition Diagram

Figure 9 is a diagram of state transitions in the sample program. The sample program includes transitions between four states, as shown in figure 9.

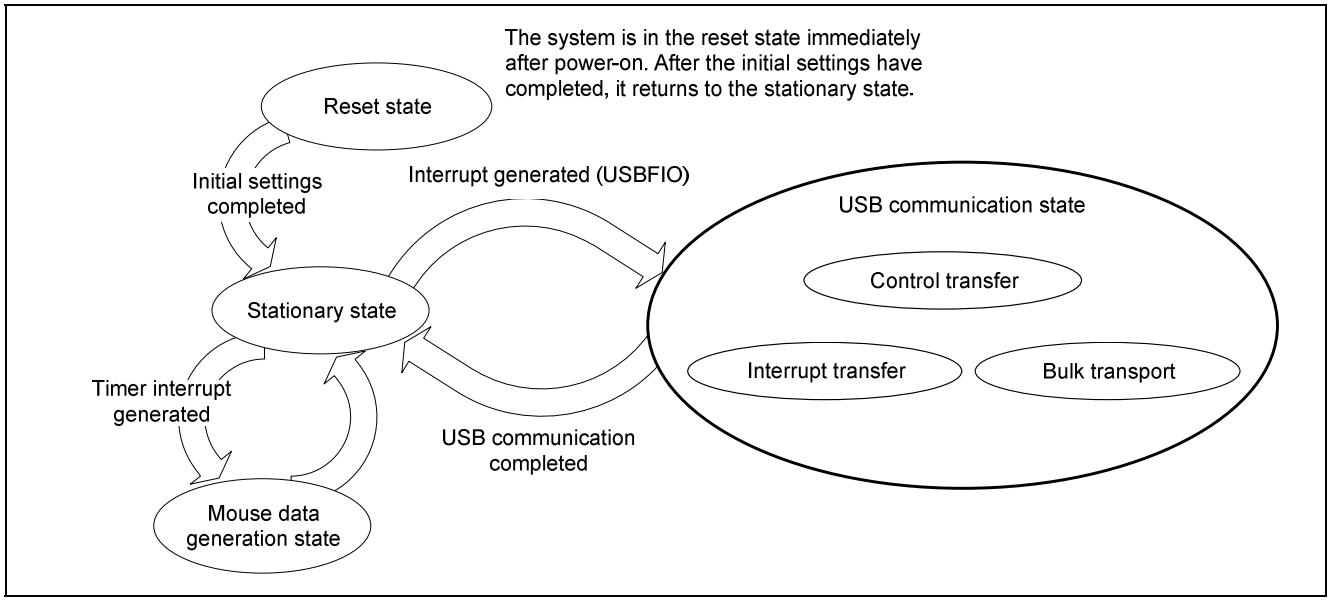


Figure 9 State Transition Diagram

- Reset state
  The system enters this state after a power-on reset or a manual reset. In the reset state, the SH7216 mainly performs initial settings.
- Stationary state
  When initial settings are completed, the system enters a stationary state in the main loop.
- USB communication state
  When an interrupt from the USB module occurs while in the stationary state, the system enters this state. In the USB communication state, data transfer is performed by a transfer method determined by the type of interrupt. The interrupts used in the sample program are indicated in interrupt flag registers 0, 1, 2, 3 and 4 (USBIFR0, USBIFR1, USBIFR2, USBIFR3, and USBIFR4). When an interrupt source occurs, the corresponding bit in USBIFR0, USBIFR1, USBIFR2, USBIFR3, or USBIFR4 is set to 1.
- Mouse data generation state
  When an overflow interrupt from 16-bit timer MTU2 occurs while in the stationary state, the system enters this state. In the mouse data generation state, mouse pointer movement data is generated automatically. A overflow interrupt occurs every 10 ms.

5.2 USB Communication State

The USB communication state can be subdivided into three states according to the transfer type. When an interrupt occurs, first there is a transition to the USB communication state, and then there is further branching to a transfer state according to the interrupt type.

5.2.1 Control Transfer

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

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

5.2.2 Interrupt Transfer

Interrupt transfer is a system that transfers data at fixed intervals in order to ensure the integrity of the data. The USB HID class uses interrupt transfer to transfer mouse data and keyboard data between the host PC and the function.

5.2.3 Bulk Transport

Bulk transport has no time limitations, so it is used to send large volumes of data with no errors. The data transport speed is not guaranteed, but the integrity of the data is assured. The USB mass storage class (bulk-only transport) uses bulk transport to transfer storage data between the host PC and the function.

Transport processing (such as reading or writing data) for the USB mass storage class (bulk-only transport) is carried out in a series of two or three stages. These stages are command transport (CBW), data transport, and status transport (CSW).

5.3 File Structure

Table 17 shows the file structure of the sample program. Each function is contained in a single file, according to transfer method or functionality.

Table 17 File Structure

File Name	Main Functionality
Startup.c	USB function initial settings
UsbMain.c	Determining interrupt sources Sending and receiving packets
DoRequest.c	Processing setup commands issued by the host PC
DoRequestHIDClass.c	Processing USB HID class commands
DoControl.c	Executing control transfer
DoInterrupt.c	Executing interrupt-in transfer
DoHidDataFormat.c	Formatting HID data to be transferred
DoMouse.c	Generating mouse data
DoRequestBOT_StorageClass.c	Processing USB mass storage class (bulk-only transport) class commands
DoBulk.c	Executing bulk transport
DoBOTMSClass.c	Executing USB mass storage class (bulk-only transport) transport
DoSCSICommand.c	Analyzing and processing SCSI commands
sct.src	Transferring initial values, etc., of variables from ROM to RAM
CatHidTypedef.h	Defining types and structures specific to the HID class
CatBOTTypedef.h	Defining structures used for bulk-only transport
CatProType.h	Prototype declarations
CatSCSITypedef.h	Defining structures used for SCSI, and making macro definitions to configure FAT information
CatTypedef.h	Defining basic structures used in USB firmware
SetBOTInfo.h	Initial settings of variables needed to support bulk-only transport
SetMacro.h	Macro definitions
SetHidInfo.h	Initial settings of variables needed to support HID class commands
SetSCSIInfo.h	Initial settings of variables needed to support SCSI commands
SetSystemSwitch.h	System operation settings
SetUsbInfo.h	Initial settings of variables used in USB firmware
SysMemMap.h	Defining memory map addresses
iodefine.h	Defining SH7216 registers

5.4 Purposes of Functions

Tables 18 to 29 show functions contained in each file and their purposes.

- Startup.c
  After a power-on reset or manual reset is carried out, the SetPowerOnSection of the Startup.c file is called. At this point, the SH7216 initial settings are entered and the RAM area used for bulk transport is cleared.

Table 18 Startup.c

Containing File	Function	Description
Startup.c	SetPowerOnSection	Initializes the module and memory, and jumps to the main loop.
	InitSDRAM	Makes initial settings for the SDRAM mounted on the SH7216 CPU board.
	_INITSCT	Copies variables that have initial settings to the RAM work area.
	InitMemory	Clears the RAM area used in bulk communication.
	InitSystem	Pull-up control of the USB bus.
	Set_EPInfoR	Writes endpoint information.

- UsbMain.c
  The main purposes of UsbMain.c are to determine interrupt sources by referencing the USB interrupt flag registers and to call functions according to the interrupt type. Also, packets are sent and received between the host controller and the function module.

Table 19 UsbMain.c

Containing File	Function	Description
UsbMain.c	BranchOfInt0	Performs a bus reset, determines the endpoint 0 interrupt source, and calls a function corresponding to the interrupt.
	BranchOfInt1	Determines the endpoint 1 to endpoint 9 interrupt sources and calls a function corresponding to each interrupt.
	GetPacket	Writes data transferred from the host controller to RAM.
	GetPacket4	Writes data transferred from the host controller to RAM in longwords (version with ring buffer support). (Not used by the sample program.)
	GetPacket4S	Writes data transferred from the host controller to RAM in longwords (high-speed version with no ring buffer support).
	PutPacket	Writes data for transfer to the host controller to the USB module.
	PutPacket4	Writes data for transfer to the host controller to the USB module in longwords (version with ring buffer support). (Not used by USB mass storage class.)
	PutPacket4S	Writes data for transfer to the host controller to the USB module in longwords (high-speed version with no ring buffer support).
	SetControlOutContents	Overwrites data with that sent from the host.
	SetUsbModule	Makes USB module initial settings.
	ActBusReset	Clears FIFO on receiving bus reset.
	ActBusVcc	Generates a USB cable connection interrupt. (Not used by the sample program.)
	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 reverse order.

- DoRequest.c
  During control transfer, commands sent from the host controller are decoded and the corresponding processing is performed. The sample program uses a vendor ID of H'045B (vendor: Renesas). When the customer develops a product, is it necessary to obtain a vendor ID from USB Implementers Forum, Inc. Because vendor commands are not used, DecVenderCommands does not perform any action. In order to use vendor commands, the customer must develop a program.

Table 20 DoRequest.c

Containing File	Function	Description
DoRequest.c	DecStandardCommands	Decodes command issued by host controller, and processes standard commands.
	DecVenderCommands	Processes vendor commands.

- DoRequestHIDClass.c
  Processing corresponding to HID class commands (GET_REPORT, GET_IDLE, GET_PROTOCOL, SET_REPORT, SET_IDLE, and SET_PROTOCOL) is performed, as follows.
  — The GET_REPORT command sends HID data from the device to the host PC through control transfer.
  — The GET_IDLE command returns the rate value of the duration for which interrupt transfer stops.
  — The GET_PROTOCOL command returns the current active protocol (boot protocol or report protocol).
  — The SET_REPORT command sends HID data from the host PC to the device through control transfer, but the sample program does not support out-direction transfers of HID data and only receives data.
  — The SET_IDLE command specifies the rate value of the duration for which interrupt transfer stops.
  — The SET_PROTOCOL command specifies the active protocol (boot protocol or report protocol).

Table 21 DoRequestHIDClass.c

Containing File	Function	Description
DoRequestHID Class.c	DecHIDClassCommands	Processes USB HID class commands.
	ActIdleCount	This function is called by an SOF interrupt, and calculates the duration for which interrupt transfer stops

- DoControl.c
  When control transfer interrupt SETUP TS occurs, ActControl obtains the command, and decoding is carried out by DecStandardCommands to determine the transfer direction. Next, when control transfer interrupt EP0o TS, EP0i TR, or EP0i TS occurs, ActControlInOut calls either ActControlIn or ActControlOut, depending on the transfer direction, and the data stage and status stage are carried out.

Table 22 DoControl.c

Containing File	Function	Description
DoControl.c	ActControl	Controls the setup stage of control transfer.
	ActControlIn	Controls the data stage and status stage of control-in transfers (transfers in which the data stage is in the in direction).
	ActControlOut	Controls the data stage and status stage of control-out transport (transport in which the data stage is in the out direction).
	ActControlInOut	Assigns the data stage and status stage of control transfers and to ActControlIn and ActControlOut, as appropriate.

- DoInterrupt.c
  On receiving the in-token of the interrupt transfer from the host PC, this function prepares next data to be sent as soon as the interrupt transfer buffer becomes empty.

Table 23 DoInterrupt.c

Containing File	Function	Description
DoInterrupt.c	ActInterruptIn	On receiving the in-token of the interrupt transfer, this function fetches data from the data transfer buffer as soon as the FIFO is empty to prepare for interrupt transfer

- DoHidDataFormat.c
  These functions prepare for transmission of HID data to the host PC.

Table 24 DoHidDataFormat.c

Containing File	Function	Description
DoHidDataFormat.c	ActMakeHidData	This function is a program interface for HID data communications. It calls ActInterruptIn if interrupt transfer stops after ActReportProtocol is called.
	ActReportProtocol	Arranges the data to be transferred according to the format specified by the Report descriptor, and writes the data to the transmit buffer.

- DoMouse.c
  This function uses a timer interrupt to generate data for mouse pointer movements.

Table 25 DoMouse.c

Containing File	Function	Description
DoMouse.c	MousePushedData Input2	This function is initiated by a timer interrupt and generates data for mouse pointer movements according to the elapsed time.

- DoRequestBOT_StorageClass.c
  This function carries out processing for USB mass storage class (bulk-only transport) commands (Bulk-Only Mass Storage Reset and Get Max LUN).
  The Bulk-Only Mass Storage Reset command resets all of the interfaces used by bulk-only transport.
  The Get Max LUN command returns the largest logical unit number used by peripheral devices. The sample program uses one logical unit, so a value of 0 is returned to the host.

Table 26 DoRequestBOT_StorageClass.c

Containing File	Function	Description
DoRequestBOT_ StorageClass.c	DecBOTClassCommands	Processes USB mass storage class (bulk-only transport) commands.

- DoBulk.c
  These functions carry out processing involving bulk transport. ActBulkInReady is not used by the USB mass storage class (bulk-only transport).

Table 27 DoBulk.c

Containing File	Function	Description
DoBulk.c	ActBulkOut	Performs bulk-out transport.
	ActBulkIn	Performs bulk-in transport.
	ActBulkInReady	Performs preparations for bulk-in transport.

- DoBOTMSClass.c
  DoBOTMSClass.c controls the two or three stages of USB mass storage class (bulk-only transport) and performs operation in accordance with the specifications.

Table 28 DoBOTMSClass.c

Containing File	Function	Description
DoBOTMSClass.c	ActBulkOnly	Divides bulk-only transport into separate stages.
	ActBulkOnlyCommand	Controls the CBW for bulk-only transport.
	ActBulkOnlyIn	Controls data transport and status transport (when the data stage is in the in direction).
	ActBulkOnlyOut	Controls data transport and status transport (when the data stage is in the out direction).

- DoSCSICommand.c
  These functions analyze SCSI commands sent from the host PC and prepare for the next data transport or status transport.

Table 29 DoSCSICommand.c

Containing File	Function	Description
DoSCSICommand.c	DecBotCmd	Processes SCSI commands sent from the host using bulk-only transport.
	SetBotCmdErr	Processes SCSI command errors.

5.4.1 Section Settings

Table 30 lists the section information for the SH7216 sample program.

Table 30 Section Information for SH7216 Sample Program

No.	Address	Section	Description
1	0x00000000	CStart	Vector Table
2	0x00000400	P	Program area
3		PURAM	CPG setting program area
4		C	Constant area
5		D	Initialized data area
6	0xFFF80000	RPURAM	CPG setting program area (allocated in RAM)
7	0xFFF80400	B	Uninitialized data area
8		R	Initialized data area (allocated in RAM)
9	0xFFF8FC00	S	Stack area

6. Reference Documents

- Software Manual
  SH-2A, SH2A-FPU User's Manual: Software (REJ09B0051)
  (The latest version can be downloaded from the Renesas Electronics Web site.)
- Hardware Manual
  SH7216 Group User's Manual: Hardware (REJ09B0543)
  (The latest version can be downloaded from the Renesas Electronics Web site.)
- SH7216 CPU Board User's Manual
  SH7216 CPU Board R0K572167C001BR User's Manual, Rev. 0.03
- USB Standard-Related
  (1) Universal Serial Bus Specification,Revision 2.0
  (2) Universal Serial Bus Mass Storage Class Specification Overview
  (3) Universal Serial Bus Mass Storage Class (Bulk-Only Transport)
  (4) Device Class Definition for Human Interface Devices (HID)
  (5) HID Usage Tables

  Website for USB Developers
  http://www.usb.org/developers

Website and Support

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

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Dec.14.10	—	First edition issued
1.10	Mar.17.11	—	Added read after FRQCR settings

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

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

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

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

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

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

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

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

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

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

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

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

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

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

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

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

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

© 2011 Renesas Electronics Corporation. All rights reserved.

Colophon 1.0