

Renesas USB Device

R01AN0275EJ0110

Rev.1.10

Sep 01, 2011

USB Host Communication Device Class Driver (HCDC)

Introduction

This document is a manual describing use of the USB host communications device class driver (HCDC) with the USB basic firmware of Renesas USB Device.

Target Device

RX62N Group, R8A66597

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

Contents

1. Overview	2
2. Software Configuration.....	3
3. Communication Device Class (CDC).....	7
4. USB Host Communication Device Class Driver (HCDC).....	13
5. Host CDC Sample Application Program (APL)	31
6. Limitations	49

1. Overview

1.1 Overview

This document is a manual describing use of the USB host communication device class driver (HCDC) and communication port device sample driver with the USB basic firmware of Renesas USB Device.

1.2 Functions and Features

The USB Host communication device class driver conforms to the Abstract Control Model of the communication device class specification (CDC) and enables communication with a CDC peripheral device.

This class driver is intended to be used in combination with the μ ITRON version and nonOS version of the USB basic firmware, rev. 1.10, from Renesas Electronics.

1.3 Related Documents

1. Universal Serial Bus Revision 2.0 specification

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

2. USB Class Definitions for Communications Devices Revision 1.2

3. USB Communications Class Subclass Specification for PSTN Devices Revision 1.2

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

4. RX62N Group, RX621 Group User's Manual: Hardware (Document No.R01UH0033EJ)

5. R8A66597 Datasheet (Document No. REJ03F0229)

6. Renesas USB Device USB Basic Firmware Application Note (Document No.R01AN0512EJ)

7. RX600 Series USB Host Communication Device Class Driver Installation Guide (Document No. R01AN0530EJ)

- Renesas Electronics Website

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

- USB Devices Page

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

1.4 Terms and Abbreviations

USB	: Universal Serial Bus
USB-BASIC-F/W	: USB basic firmware of Renesas USB Device
H/W	: Renesas USB Device
HEW	: High-performance Embedded Workshop
RX62N-RSK	: Renesas Starter Kit + for RX62N
nonOS	: USB-BASIC-F/W for OS less system
μ ITRON	: USB-BASIC-F/W for μ ITRON system
HCD	: Host Control driver of Renesas USB Device (USB-BASIC-F/W)
MGR	: Peripheral device state manager of HCD
CDC	: Communications devices class
HCDC	: Host communication devices class
APL	: Sample application program
Task	: Processing unit
Scheduler Macro	: Macro for calling the scheduler function
SW1/SW2/SW3	: User switches on the RX62N-RSK (Caution 1)

(Caution 1) When RX62N-RSK used in conjunction with the R8A66597, SW1 is allocated to the port used by the interruption. Therefore please do not use SW1.

2. Software Configuration

2.1 Module Configuration

Table 2.1 lists the modules, and figure 2.1 shows the configuration of HCDC.

Table 2.1 Modules

Module	Description
APL	User application program. Created by the customer to match the system specifications.
HCDC	Sends requests from the APL for requests and data communication involving the CDC to the HCD.
MGR / HUB	Enumerates the connected devices and starts HCDC. Also performs device state management.
HCD	USB host H/W control driver.

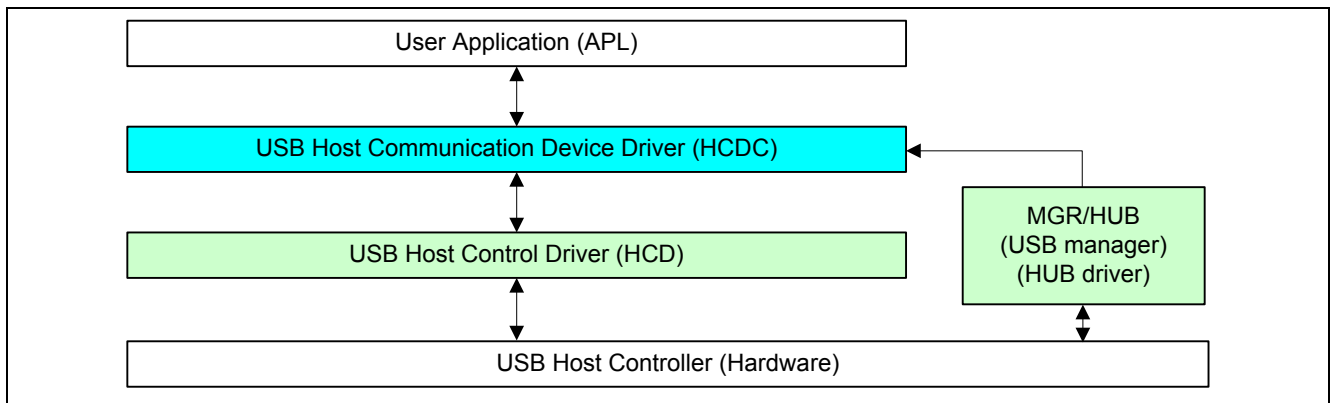


Figure 2.1 Software Configuration Diagram

2.2 File Configuration List

2.2.1 Folder Structure

The folder structure of the files supplied with the device class is shown below.

```

+---Workspace
  +---CDCFW                For USB0
  | +---HCDC                Host CDC driver
  | +---include            Host CDC header files
  +---CDC2FW              For USB1 (R8A66597 is not used.) :Structure is same to CDCFW
  | +---HCDC                Host CDC driver
  | +---include            Host CDC header files
  +---CDCCFW              USB0, 1 shared
  | +---include            CDC common header files
  +---RI600_4             ITRON folder (not included in nonOS version)
  | +---config_Hcdc        ITRON configuration file for HCDC: "r_usb_RX62N.cfg"
  +---HwResourceForUSB
  +---SmplMain
  | +---APL                Sample application
  +---USBSTDFW            For USB0
  +---USBCSTDFW           USB0, 1 shared
  | +---include            Common header file system resource file
  +---USB2STDFW           For USB1 (R8A66597 is used.) :Structure is same to USBSTDFW

```

Note: Some folders that are not mentioned in the description of the class driver are omitted from the listing above.

2.2.2 File Structure

Table 2.2 shows the file structure supplied with HCDC.

Table 2.2 File Structure

	File Name	Description	Note
SmplMain/APL	r_usb_cdata.c	Sample application data	Sample
SmplMain/APL	r_usb_cdata.h	Sample application data external reference	Sample
SmplMain/APL	r_usb_HCDC_apl.c	Sample application program	Sample
CDCCFW/include	r_usbc_HCDCdefine.h	CDC type definitions and macro definitions	
CDCFW/include	r_usb_HCDCextern.h	CDC prototype, external reference	
CDCFW/HCDC	r_usb_HCDCdefEp.c	CDC pipe information table	
CDCFW/HCDC	r_usb_HCDCdriver.c	Class checking and pipe information setting functions	
CDCFW/HCDC	r_usb_HCDCapi.c	CDC API functions	

2.3 System Resources

2.3.1 System Resource Definitions when Running μ ITRON Version

Table 2.3 lists the μ ITRON resources used by HCDC and the sample APL on the μ ITRON version.

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

See the USB Basic Firmware Application Note for more details on the definition method.

Table 2.3 μ ITRON Resource Definitions

	Name	Description
Task stack size: (512 bytes)	USB_HCDC_TSK	HCDC main task (usb_hcdc_Task) task ID: USB_HCDC_TSK Task priority: 6
	USB_HCDCSMP_TSK	APL main task (usb_hcdc_SampleApITask) task ID: USB_HCDCSMP_TSK Task priority: 7
Mailbox max. priority: 1	USB_HCDC_MBX	HCDC -> HCDC / APL -> HCDC mailbox ID
Waiting task queue: FIFO order	USB_HCDCSMP_MBX	HCDC -> APL mailbox ID
Message queue: FIFO order		
Fixed-length memory pool	USB_HCDC_MPL	HCDC memory pool ID
Block count: (10)	USB_HCDCSMP_MPL	APL memory pool ID
Block size: (64 bytes)		
Waiting task queue: FIFO order		
μ ITRON base timer	Hardware timer	1 ms

2.3.2 System Resource Definitions when Running NonOS Version

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

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

Table 2.4 List of Scheduler Registration IDs

	Name	Description
Scheduler registration task	USB_HCDC_TSK (default value: USBC_TID_4 = 4)	HCDC main task (usb_hcdc_Task) Task ID: USB_HCDC_TSK Task priority: USB_HCDC_PRI (default value: USBC_PRI_3 = 3)
	USB_HCDCSMP_TSK (default value: USBC_TID_5 = 5)	APL main task (usb_hcdc_SampleApiTask) Task ID: USB_HCDCSMP_TSK Task priority: USB_HCDCSMP_PRI (default value: USBC_PRI_3 = 3)
Mailbox ID	USB_HCDC_MBX (default value: USB_HCDC_TSK)	HCDC -> HCDC / APL -> HCDC mailbox ID
	USB_HCDCSMP_MBX (default value: USB_HCDCSMP_TSK)	HCDC -> APL mailbox ID
Memory pool ID	USB_HCDC_MPL (default value: USB_HCDC_TSK)	HCDC memory pool ID
	USB_HCDCSMP_MPL (default value: USB_HCDCSMP_TSK)	APL memory pool ID

3. Communication Device Class (CDC)

3.1 Basic Functions

This software conforms to the Abstract Control Model subclass of the communication device class specification. The main functions of HCDC are as follows.

1. Verifying connected devices
2. Making communication line settings
3. Acquiring the communication line state
4. Transferring data to and from the CDC peripheral device

3.2 Overview of Abstract Control Model

The Abstract Control Model subclass is a technology that bridges the gap between USB devices and earlier modems (employing RS-232C connections), enabling use of application programs designed for older modems.

3.3 Endpoint Specifications

The software uses the following endpoints.

Table 3.1 Endpoint Specifications

Transfer Method	Description
Control In/Out	Standard request, class request
Bulk In	Data transfer from device to host
Bulk Out	Data transfer from host to device
Interrupt In	State notification from device to host

3.4 Class Requests (Requests from Host to Device)

The software supports the following class requests.

Table 3.2 CDC Class Requests

Request	Code	Description
SendEncapsulatedCommand	0x00	Transmits AT commands, etc., defined by the protocol.
GetEncapsulatedResponse	0x01	Requests a response to a command transmitted by SendEncapsulatedCommand.
SetCommFeature	0x02	Enables or disables features such as device-specific 2-byte code and country setting.
GetCommFeature	0x03	Acquires the enabled/disabled state of features such as device-specific 2-byte code and country setting.
ClearCommFeature	0x04	Restores the default enabled/disabled settings of features such as device-specific 2-byte code and country setting.
SetLineCoding	0x20	Makes communication line settings (communication speed, data length, parity bit, and stop bit length).
GetLineCoding	0x21	Acquires the communication line setting state.
SetControlLineState	0x22	Makes communication line control signal (RTS, DTR) settings.
SendBreak	0x23	Transmits a break signal.

3.4.1 SendEncapsulatedCommand Data Format

The SendEncapsulatedCommand data format is shown below.

Table 3.3 SendEncapsulatedCommand Data Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	SEND_ENCAPSULATED_COMMAND (0x00)	0x0000	0x0000	Data length	Control protocol base command

Note: Items such as AT commands for modem control are set as Data, and wLength is set to match the length of the data.

3.4.2 GetEncapsulatedResponse Data Format

The GetEncapsulatedResponse data format is shown below.

Table 3.4 GetEncapsulatedResponse Data Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	GET_ENCAPSULATED_RESPONSE (0x01)	0x0000	0x0000	Data length	The data depends on the protocol.

Note: The response data to SendEncapsulatedCommand is set as Data, and wLength is set to match the length of the data.

3.4.3 SetCommFeature Data Format

The SetCommFeature data format is shown below.

Table 3.5 SetCommFeature Data Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	SET_COMM_FEATURE (0x02)	Feature Selector <small>Note</small>	0x0000	Data length	Status Either the country code or the Abstract Control Model idle setting/multiplexing setting for Feature Selector.

Note: Shown in Table 3.7 Feature selector Settings, Table 3.8 Status Format when ABSTRACT_STATE Selected.

3.4.4 GetCommFeature Data Format

The GetCommFeature data format is shown below.

Table 3.6 GetCommFeature Data Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	GET_COMM_FEATURE (0x03)	Feature Selector <small>Note</small>	0x0000	Data length	Status Either the country code or the Abstract Control Model idle setting/multiplexing setting for Feature Selector.

Note: Shown in Table 3.7 Feature selector Settings, Table 3.8 Status Format when ABSTRACT_STATE Selected.

Table 3.7 Feature Selector Settings

Feature Selector	Code	Targets	Length of Data	Description
RESERVED	00h	None	None	Reserved
ABSTRACT_STATE	01h	Interface	2	Selects the setting for Abstract Control Model idle state and signal multiplexing.
COUNTRY_SETTING	02h	Interface	2	Selects the country code in hexadecimal format, as defined by ISO 3166.

Table 3.8 Status Format when ABSTRACT_STATE Selected

Bit Position	Description
D15 to D2	Reserved
D1	Data multiplexing setting 1: Multiplexing of call management commands is enabled for the Data class. 0: Multiplexing is disabled.
D0	Idle setting 1: No endpoints of the target interface accept data from the host, and data is not supplied to the host. 0: Endpoints continue to accept data and it is supplied to the host.

3.4.5 ClearCommFeature Data Format

The ClearCommFeature data format is shown below.

Table 3.9 ClearCommFeature Data Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	CLEAR_COMM_FEATURE (0x4)	Feature Selector Note	0x0000	0x0000	None

Note: Shown in Table 3.7 Feature selector Settings.

3.4.6 SetLineCoding Data Format

The SetLineCoding data format is shown below.

Table 3.10 SetLineCoding Data Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	SET_LINE_CODING (0x20)	0x0000	0x0000	0x0007	Line Coding Structure See table 3.11, Line Coding Structure Format

Line Coding Structure Format is shown below.

Table 3.11 Line Coding Structure Format

Offset	Field	Size	Value	Description
0	dwDTERate	4	Number	Data terminal speed (bps)
4	bCharFormat	1	Number	Stop bits 0 - 1 stop bit 1 - 1.5 stop bits 2 - 2 stop bits
5	bParityType	1	Number	Parity 0 - None 1 - Odd 2 - Even 3 - Mask 4 - Space
6	bDataBits	1	Number	Data bits (5, 6, 7, 8)

3.4.7 GetLineCoding Data Format

The GetLineCoding data format is shown below.

Table 3.12 GetLineCoding Data Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0xA1	GET_LINE_CODING (0x21)	0 x0000	0x0000	0x0007	Line Coding Structure See table 3.11, Line Coding Structure Format

3.4.8 SetControlLineSTATE Data Format

The SetControlLineState data format is shown below.

Table 3.13 SetControlLineState Data Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	SET_CONTROL_ LINE_STATE (0x22)	Control Signal Bitmap See table 3.14, Control Signal Bitmap Format	0x0000	0x0000	None

Table 3.14 Control Signal Bitmap

Bit Position	Description
D15 to D2	Reserved (set to 0)
D1	DCE transmit function control 0 - RTS OFF 1 - RTS ON
D0	Notification of DTE ready state 0 - DTR OFF 1 - DTR ON

3.4.9 SendBreak Data Format

The SendBreak data format is shown below.

Table 3.15 SendBreak Data Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	SEND_BREAK (0x23)	Break signal output duration	0x0000	0x0000	None

3.5 Class Notifications (Notifications from Device to Host)

The class notifications supported and not supported by the software are shown below.

Table 3.16 CDC Class Notifications

Notification	Code	Description	Supported
NETWORK_CONNECTION	0x00	Notification of network connection state	No
RESPONSE_AVAILABLE	0x01	Response to GET_ENCAPSLATED_RESPONSE	Yes
SERIAL_STATE	0x20	Notification of serial line state	Yes

3.5.1 SerialState Data Format

The SerialState data format is shown below.

Table 3.17 SerialState Data Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0xA1	SERIAL_STATE (0x20)	0x0000	0x0000	0x0002	UART State bitmap See table 3.18, UART State bitmap Format

UART State bitmap Format is shown below.

Table 3.18 UART State bitmap Format

Bits	Field	Description
D15 to D7		Reserved
D6	bOverRun	Overflow error detected
D5	bParity	Parity error detected
D4	bFraming	Framing error detected
D3	bRingSignal	INCOMING signal (ring signal) detected
D2	bBreak	Break signal detected
D1	bTxCarrier	Data Set Ready: Line connected and ready for communication
D0	bRxCarrrier	Data Carrier Detect: Carrier detected on line

3.5.2 ResponseAvailable Data Format

The ResponseAvailable data format is shown below.

Table 3.19 ResponseAvailable Data Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0xA1	RESPONSE_AVAIL ABLE (0x01)	0x0000	0x0000	0x0000	None

4. USB Host Communication Device Class Driver (HCDC)

4.1 Basic Functions

This software conforms to the Abstract Control Model subclass of the communication device class specification.

The main functions of HCDC are as follows.

1. Sending requests to the CDC peripheral device for class requests
2. Transferring data to and from the CDC peripheral device
3. Receiving serial communication error information from the CDC peripheral device

4.2 HCDC Global Area Variables

Table 4.1 lists the global area variables of HCDC.

Table 4.1 Global Area Variables of HCDC

	Variable		Description
1	uint16_t	usb_ghcdc_tpl[]	Targeted Peripheral List (TPL)
2	uint16_t	usb_ghcdc_SendEncapsulatedCommand_Table[]	Class request setup table (SendEncapsulatedCommand)
3	uint16_t	usb_ghcdc_GetEncapsulatedResponse_Table[]	Class request setup table (GetEncapsulatedResponse)
4	uint16_t	usb_ghcdc_SetCommFeature_Table[]	Class request setup table (SetCommFeature)
5	uint16_t	usb_ghcdc_GetCommFeature_Table[]	Class request setup table (GetCommFeature)
6	uint16_t	usb_ghcdc_ClrCommFeature_Table[]	Class request setup table (ClrCommFeature)
7	uint16_t	usb_ghcdc_SetLineCoding_Table[]	Class request setup table (SetLineCoding)
8	uint16_t	usb_ghcdc_SetControlLineState_Table[]	Class request setup table (SetControlLineState)
9	uint16_t	usb_ghcdc_GetLineCoding_Table[]	Class request setup table (GetLineCoding)
10	uint16_t	usb_ghcdc_SendBreak_Table[]	Class request setup table (SendBreak)
11	uint16_t	usb_ghcdc_SerialState_Table[]	Data table for class notification SerialState receive data check
12	uint8_t	usb_ghcdc_ClassRequestData[]	Data area for class request (control transfer data stage)
13	uint8_t	usb_ghcdc_SerialState[]	Class notification SerialState receive area
14	USBC_UTR_t	usb_ghcdc_ControlSetupUtr	User transaction message (task-to-task communication) for class request
15	USBC_UTR_t	usb_ghcdc_st_utr	User transaction message (task-to-task communication) for class notification
16	uint16_t	usb_ghcdc_devaddr	Device address
17	uint16_t	usb_ghcdc_OutPipe	Pipe number (port 0) used for CDC data communication bulk out transfer
18	uint16_t	usb_ghcdc_InPipe	Pipe number (port 0) used for CDC data communication bulk in transfer

Variable		Description
19	uint16_t usb_ghcdc_StatInPipe	Pipe number (port 0) used for class notification SerialState receive (Interrupt in transfer)
20	uint16_t usb_ghcdc_OutPipe2	Pipe number (port 1) used for CDC data communication bulk out transfer
21	uint16_t usb_ghcdc_InPipe2	Pipe number (port 1) used for CDC data communication bulk in transfer
22	uint16_t usb_ghcdc_StatInPipe2	Pipe number (port 1) used for class notification SerialState receive (Interrupt in transfer)
23	uint16_t usb_ghcdc_InPipe_EpTblIndex	Offset to pipe information storage position used for bulk in transfer from beginning of pipe information table* ¹
24	uint16_t usb_ghcdc_OutPipe_EpTblIndex	Offset to pipe information storage position used for bulk out transfer from beginning of pipe information table* ¹
25	uint16_t usb_ghcdc_StatPipe_EpTblIndex	Offset to pipe information storage position used for interrupt in transfer from beginning of pipe information table* ¹
26	USBC_CB_t usb_ghcdc_TxCB	Address of APL call-back function called at data transmit end
27	USBC_CB_t usb_ghcdc_RxCB	Address of APL call-back function called at data receive end
28	USBC_HCDC_SERIAL_ST_CB_t usb_ghcdc_StatCB	Address of APL call-back function called when class notification SerialState received
29	USBC_CB_t usb_ghcdc_ClsReqCB	Address of APL call-back function called at class request communication end
30	USBC_UTR_t usb_ghcdc_Mess[]	Pipe-specific user transaction message
31	USBC_HCDC_ClassRequest_UTR_t usb_ghcdc_cls_req	Parameter area for class request
32	USBC_HCDC_LineCoding_t usb_ghcdc_line_coding_set_parm	Communication parameter area for class request SetLineCoding

Note1. Pipe information table: A table defining pipe settings customized for the class driver

4.3 HCDC Structure

4.3.1 HCDC Class Request Structure

Table 4.2 USBC_HCDC_LineCoding_t Structure

	Member	Description	Remarks
uint32_t	dwDTERate	Line speed	Unit: bps
uint8_t	bCharFormat	Stop bits setting	
uint8_t	bParityType	Parity setting	
uint8_t	bDataBits	Data bit length	

Table 4.3 USBC_HCDC_ControlLineState_t Structure

	Member	Description	Remarks
uint16_t	bDTR: 1	Line connected and ready for communication	Data Terminal Ready
uint16_t	bRTS: 1	Transmit request	Request to Send

Table 4.4 USBC_HCDC_SerialState_t Structure

	Member	Description	Remarks
uint16_t	bRxCarr: 1	Carrier detected on line	Data Carrier Detect
uint16_t	bTxCarr: 1	Line connected and ready for communication	Data Set Ready
uint16_t	bBreak: 1	break signal detected	
uint16_t	bRingSignal: 1	Incoming signal (Ring signal) detected	
uint16_t	bFraming: 1	Framing error detected	
uint16_t	bParity: 1	Parity error detected	
uint16_t	bOverRun: 1	Overrun error detected	

Table 4.5 USBC_HCDC_BreakDuration_t Structure

	Member	Description	Remarks
uint16_t	wTime_ms	Break output duration	Unit: msec

Table 4.6 USBC_HCDC_Encapsulated_t Structure

	Member	Description	Remarks
uint8_t	*p_data	Address for storing AT commands for transmission or reply data in response to AT command transmission	

Table 4.7 USBC_HCDC_AbstractState_t Structure

	Member	Description	Remarks
uint16_t	bDMS:1	Data Multiplexed State	
uint16_t	bIS:1	Idle Setting	

Table 4.8 USBC_HCDC_CountrySetting_t Structure

	Member	Description	Remarks
uint16_t	country_code	Country code	

4.3.2 CommFeature Function Selection Union

Table 4.9 USBC_HCDC_CommFeature_t Union

	Member	Description	Remarks
USBC_HCDC_AbstractState_t	abstractState	Abstract Control Model selection parameters	
USBC_HCDC_CountrySetting_t	countrySetting	Country Setting selection parameters	

4.3.3 Class Request Input Parameter Union

Table 4.10 USBC_HCDC_LineCoding_t Structure

	Member	Description	Remarks
USB_HCDC_LineCoding_t	*LineCoding	Address where LineCoding setting value is stored for LineCoding requests	
USB_HCDC_ControlLineState_t	ControlLineState	Line control setting value for SetControlLineState requests	
USB_HCDC_BreakDuration_t	BreakDuration	Break output duration for SendBreak requests	
USBC_HCDC_CommFeature_t	*CommFeature	Address for storing device-specific communication function settings	
USBC_HCDC_Encapsulated_t	Encapsulated	Address for storing AT commands for transmission or reply data in response to AT command transmission	

4.3.4 CDC Data Transfer API Input Parameter Structure

Table 4.11 USBC_HCDC_UTR_t Structure

	Member	Description	Remarks
void	*tranadr	Transfer data address	
uint32_t	tranlen	Transfer size	
USBC_CB_t	complete	Data transfer end call-back function	

4.3.5 Class Request API Input Parameter Structure

Table 4.12 USBC_HCDC_ClassRequest_UTR_t Structure

	Member	Description	Remarks
USBC_CDC_ABS_Req_t	bRequestCode	Class request classification	
USBC_CDC_ClassRequestParm_t	parm	Parameter setting value	For each class request
USBC_CB_t	complete	Class request processing end call-back function	

4.4 HCD Constant Definition

Table 4.13 HCD Constant Definition

	Constant Name	Value	Description
1	CDC class request code		
	USBC_HCDC_SEND_ENCAPSULATED_COMMAND	0x00	SendEncapsulatedCommand request code
	USBC_HCDC_GET_ENACAPSULATED_RESPONSE	0x01	GetEncapsulatedResponse request code
	USBC_HCDC_SET_COMM_FEATURE	0x02	SetCommFeature request code
	USBC_HCDC_GET_COMM_FEATURE	0x03	GetCommFeature request code
	USBC_HCDC_CLEAR_COMM_FEATURE	0x04	ClearCommFeature request code
	USBC_HCDC_SET_LINE_CODING	0x20	SetLineCoding request code
	USBC_HCDC_GET_LINE_CODING	0x21	GetLineCoding request code
	USBC_HCDC_SET_CONTROL_LINE_STATE	0x22	SetControlLineState request code
	USBC_HCDC_SEND_BREAK	0x23	SendBreak request code
2	Data bit length definition		
	USBC_HCDC_DATA_BIT_7	0x07	Data bit length: 7 bits
	USBC_HCDC_DATA_BIT_8	0x08	Data bit length: 8 bits
3	Stop bit length definition		
	USBC_HCDC_STOP_BIT_1	0x00	Stop bit length: 1 bit
	USBC_HCDC_STOP_BIT_1_5	0x01	Stop bit length: 1.5 bits
	USBC_HCDC_STOP_BIT_2	0x02	Stop bit length: 2 bits
4	Parity bit definition		
	USBC_HCDC_PARITY_BIT_NONE	0x00	No parity
	USBC_HCDC_PARITY_BIT_ODD	0x01	Odd parity
	USBC_HCDC_PARITY_BIT_EVEN	0x02	Even parity
5	Line speed definition		
	USBC_HCDC_SPEED_1200	1200	Line coding: 1200 bps
	USBC_HCDC_SPEED_2400	2400	Line coding: 2400 bps
	USBC_HCDC_SPEED_4800	4800	Line coding: 4800 bps
	USBC_HCDC_SPEED_9600	9600	Line coding: 9600 bps
	USBC_HCDC_SPEED_14400	14400	Line coding: 14400 bps
	USBC_HCDC_SPEED_19200	19200	Line coding: 19200 bps
	USBC_HCDC_SPEED_38400	38400	Line coding: 38400 bps
	USBC_HCDC_SPEED_57600	57600	Line coding: 57600 bps
	USBC_HCDC_SPEED_115200	115200	Line coding: 115200 bps
6	CDC driver definition		
	USBC_HCDC_SERIAL_STATE_MSG_LEN	10	SerialState notification message length (10 bytes)
	USBC_HCDC_LINE_CODING_STR_LEN	7	Line coding structure size
	USBC_HCDC_TR_DATA_SIZE	64	Area size for data transmitted/received by CDC driver
	USBC_HCDC_SETUP_TBL_SIZE	5	Setup packet table size (16 bits × 5)
	USBC_HCDC_SETUP_POS_REQ	0	Setup packet request field position
	USBC_HCDC_SETUP_POS_VAL	1	Setup packet wValue position
	USBC_HCDC_SETUP_POS_INDX	2	Setup packet wIndex position
	USBC_HCDC_SETUP_POS LENG	3	Setup packet wLength position

	Constant Name	Value	Description
7	Host CDC task notification command definition		
	USBC_HCDC_TCMD_SEND	0	USB transmit request
	USBC_HCDC_TCMD_RECEIVE	1	USB receive request
	USBC_HCDC_TCMD_CLASS_REQ	2	Class request communication request
	USBC_HCDC_TCMD_CLASS_NOTIFY	3	Class notification SerialState receive request
	USBC_HCDC_TCMD_SEND_CB	4	USB transmit end
	USBC_HCDC_TCMD_RECEIVE_CB	5	USB receive end
	USBC_HCDC_TCMD_CLASS_REQ_CB	6	Class request communication end
	USBC_HCDC_TCMD_CLASS_NOTIFY_CB	7	Class notification receive end

4.5 List of HCD Functions

Table 4.14 List of CDC Driver Functions

File Name	Function Name	Description	Note
r_usb_HCDCapi.c	R_usb_hcdc_Registration	CDC class registration function	
	R_usb_hcdc_Release	CDC class registration cancel function	
	R_usb_hcdc_Open	Host CDC task open function	
	R_usb_hcdc_Close	Host CDC task close function	
	R_usb_hcdc_Check	Descriptor check processing	
	R_usb_hcdc_SetPipeRegistration	Pipe setting processing	
	R_usb_hcdc_DataTrans	The data in the structure that transmitting and receiving, and transmitted to the PC	
	R_usb_hcdc_SendEncapsulatedCommand	SendEncapsulatedCommand request API	
	R_usb_hcdc_GetEncapsulatedResponse	GetEncapsulatedResponse request API	
	R_usb_hcdc_SetCommFeature	SetCommFeature request API	
	R_usb_hcdc_GetCommFeature	GetCommFeature request API	
	R_usb_hcdc_ClrCommFeature	ClrCommFeature request API	
	R_usb_hcdc_SetLineCoding	SetLineCoding request API	
	R_usb_hcdc_GetLineCoding	GetLineCoding request API	
	R_usb_hcdc_SendBreak	SendBreak request API	
	R_usb_hcdc_SetControlLineState	SetControlLineState request API	
	R_usb_hcdc_InTransfer	CDC data receive request	
	R_usb_hcdc_OutTransfer	CDC data transmit request	
	R_usb_hcdc_SerialStateTransfer	Line state acquisition request	
	r_usb_HCDCdriver.c	usb_hcdc_Task	Host CDC task
usb_hcdc_PipeInfo		Endpoint descriptor check processing	
usb_hcdc_ClassRequest		Request for CDC class request	
usb_hcdc_SerialStateNotification		SerialState receive setting processing	
usb_hcdc_InTransResult		Data receive end call-back	
usb_hcdc_OutTransResult		Data transmit end call-back	
usb_hcdc_StatusTransResult		Line state acquisition standby call-back function	
usb_hcdc_ClassRequestResult		Class request end call-back	
usb_hcdc_SerialStateTransResult		SerialState receive call-back	
usb_hcdc_ClassRequestProcess		Class request processing	

4.6 Description of HCDC Functions

The HCDC driver API specifications are described below.

Table 4.15 R_usb_hcdc_Registration()

Name	Register HCDC		
Call format	R_usb_hcdc_Registration		
Arguments	void	—	—
Return values	void	—	—
Description	Registers the host communication device class. The registration function changes to match the application program. The type for registration in HCD is USBC_HCDREG_t. (For information on types, see Renesas USB Device USB Basic Firmware Application Note.)		
	Variable Name	Description	
	ifclass	Registers the interface class code USBC_IFCLS_CDCC.	
	*tpl	Registers the target peripheral list.	
	*pipetbl	Registers the pipe information table address.	
	classinit	Registers the function that starts when the driver (HDCD) is registered. A dummy function is specified in the sample program.	
	classcheck	Registers the HCDC descriptor check processing function R_usb_hcdc_Check(). This checks the configuration and endpoint descriptors.	
	devconfig	Registers the host CDC open function usb_hcdc_SampleApiTaskOpen. This generates and starts the sample application task.	
	devdetach	Registers the host CDC close function usb_hcdc_SampleApiTaskClose. This forcibly terminates and erases the sample application.	
	devsuspend	Registers the function that starts when transitioning to the suspend state. A dummy function is specified in the sample program.	
devresume	Registers the function that starts when transitioning to the resume state. A dummy function is specified in the sample program.		
Notes			

Table 4.16 R_usb_hcdc_Release()

Name	Release HCDC		
Call format	R_usb_hcdc_Release		
Arguments	void	—	—
Return values	void	—	—
Description	Erases the host communication device class.		
Notes			

Table 4.17 R_usb_hcdc_Open()

Name	Open HCDC Task		
Call format	R_usb_hcdc_Open		
Arguments	void	—	—
Return values	void	—	—
Description	Starts the HCDC task.		
Notes			

Table 4.18 R_usb_hcdc_Close()

Name	Close HCDC Task		
Call format	R_usb_hcdc_Close		
Arguments	void	—	—
Return values	void	—	—
Description	Forces the HCDC task to end.		
Notes			

Table 4.19 R_usb_hcdc_Check()

Name	HCDC Class Check Processing		
Call format	R_usb_hcdc_Check		
Arguments	uint16_t	**table	Table elements Table[0] device descriptor Table[1] configuration descriptor Table[2] interface descriptor Table[3] descriptor check result Table[4] hub classification Table[5] port number Table[6] communication speed Table[7] device address
Return values	void	—	—
Description	This function is called by HCDC class checking. It checks the descriptor classifications of configuration descriptors, checks the pipe information table, and acquires string descriptors. USBC_DONE is returned if the configuration descriptor and pipe information table checks complete normally, and USBC_ERROR is returned if an error is detected.		
Notes			

Table 4.20 R_usb_hcdc_SetPipeRegistration()

Name	Register HCDC Pipes		
Call format	R_usb_hcdc_SetPipeRegistration		
Arguments	uint16_t	devadr	Device address
Return values	USBC_ER_t		Error code
Description	Makes hardware pipe configuration settings for the pipes used by the CDC device detected by usb_hcdc_PipeInfo. Settings are made for three pipes: bulk in, bulk out, and interrupt in. USBC_DONE is returned if settings for all the pipes complete normally, and USBC_ERROR is returned if one or more settings fail.		
Notes			

Table 4.21 R_usb_hcdc_DataTrans()

Name	HCDC Data Transfer		
Call format	R_usb_hcdc_DataTrans		
Arguments	uint16_t	pipe	Pipe number
	uint32_t	size	Data transfer size
	uint8_t	*table	Data transfer address
	USBC_CB_t	complete	Processing end notification call-back function
Return alues	void	-	-
Description	To send a message to the PCD		
Notes			

Table 4.22 R_usb_hcdc_SendEncapsulatedCommand()

Name	SendEncapsulatedCommand Class Request Processing		
Call format	R_usb_hcdc_SendEncapsulatedCommand		
Arguments	uint8_t	*p_data	AT command data
	uint16_t	length	AT command data length
	USBC_CB_t	complete	Processing end notification call-back function
Return values	USBC_ER_t		Error code
Description	Transmits an AT command to the CDC peripheral device.		
Notes			

Table 4.23 R_usb_hcdc_GetEncapsulatedResponse()

Name	GetEncapsulatedResponse Class Request Processing		
Call format	R_usb_hcdc_GetEncapsulatedResponse		
Arguments	uint8_t	*p_data	AT command data
	uint16_t	length	AT command data length
	USBC_CB_t	complete	Processing end notification call-back function
Return values	USBC_ER_t		Error code
Description	Requests a response to an AT command from the CDC peripheral device.		
Notes			

Table 4.24 R_usb_hcdc_SetCommFeature()

Name	SetCommFeature Class Request Processing		
Call format	R_usb_hcdc_SetCommFeature		
Arguments	uint16_t	selector	Feature Selector (ABSTRACT_STATE/COUNTRY_SETTING)
	USBC_HCDC_CommFeature_t	*p_commfeature	Feature Parameter
	USBC_CB_t	complete	Processing end notification call-back function
Return values	USBC_ER_t		Error code
Description	Makes device-specific communication function settings for the CDC peripheral device.		
Notes			

Table 4.25 R_usb_hcdc_GetCommFeature()

Name	GetCommFeature Class Request Processing		
Call format	R_usb_hcdc_GetCommFeature		
Arguments	uint16_t	selector	Feature Selector (ABSTRACT_STATE/COUNTRY_SETTING)
	USBC_HCDC_CommFeature_t	*p_commfeature	Feature Parameter
	USBC_CB_t	complete	Processing end notification call-back function
Return values	USBC_ER_t		Error code
Description	Makes device-specific communication function settings request for the CDC peripheral device.		
Notes			

Table 4.26 R_usb_hcdc_ClrCommFeature()

Name	ClrCommFeature Class Request Processing		
Call format	R_usb_hcdc_ClrCommFeature		
Arguments	uint16_t	selector	Feature Selector (ABSTRACT_STATE/COUNTRY_SETTING)
	USBC_CB_t	complete	Processing end notification call-back function
Return values	USBC_ER_t		Error code
Description	Makes device-specific communication function settings clear request for the CDC peripheral device.		
Notes			

Table 4.27 R_usb_hcdc_SetLineCoding()

Name	SetLineCoding Class Request Processing		
Call format	R_usb_hcdc_SetLineCoding		
Arguments	uint32_t	speed	Baud rate
	uint8_t	databits	Data bit length (7 bits or 8 bits)
	uint8_t	stopbit	Stop bits (1 bit, 1.5 bits, 2 bits)
	uint8_t	parity	Parity (NONE, ODD, EVEN)
Return values	USBC_CB_t	complete	Processing end notification call-back function
	USBC_ER_t		Error code
Description	Makes communication condition settings (communication speed, data length, number of stop bits, parity) for the CDC peripheral device.		
Notes	For supported baud rate settings, see "Line speed definition" in 4.4, HCDC Constant Definition.		

Table 4.28 R_usb_hcdc_GetLineCoding()

Name	GetLineCoding Class Request Processing		
Call format	R_usb_hcdc_GetLineCoding		
Arguments	USBC_CDC_LineCoding_t	*p_linecoding	Start address of communication condition storage area
	USBC_CB_t	complete	Processing end notification call-back function
Return values	USBC_ER_t		Error code
Description	Acquires the communication conditions (communication speed, data length, number of stop bits, parity) set for the CDC peripheral device.		
Notes			

Table 4.29 R_usb_hcdc_SendBreak()

Name	GetLineCoding Class Request Processing		
Call format	R_usb_hcdc_SendBreak		
Arguments	uint16_t	time_ms	Break signal output duration (ms)
	USBC_CB_t	complete	Processing end notification call-back function
Return values	USBC_ER_t		Error code
Description	Requests break signal output from the CDC peripheral device.		
Notes			

Table 4.30 R_usb_hcdc_SetControlLineState()

Name	SetControlLineState Class Request Processing		
Call format	R_usb_hcdc_SetControlLineState		
Arguments	uint16_t	dtr	Communication ready/not read setting
	uint16_t	rts	Transmit ready/not read setting
	USBC_CB_t	complete	Processing end notification call-back function
Return values	USBC_ER_t		Error code
Description	Makes control line settings for the CDC peripheral device.		
Notes			

Table 4.31 R_usb_hcdc_InTransfer()

Name	CDC Data Receive Processing		
Call format	R_usb_hcdc_InTransfer		
Arguments	USBC_CDC_UTR_t	*indata	Receive request parameters <ul style="list-style-type: none"> • Receive data storage address • Receive data size • Processing end notification call-back function
Return values	USBC_ER_t		Error code
Description	Receives data (bulk in transfer) from the CDC peripheral device. For information on the processing end notification call-back function type USBC_CB_t, see Renesas USB Device USB Basic Firmware Application Note.		
Notes			

Table 4.32 R_usb_hcdc_OutTransfer()

Name	CDC Data Transmit Processing		
Call format	R_usb_hcdc_OutTransfer		
Arguments	USBC_CDC_UTR_t	*outdata	Transmit request parameters <ul style="list-style-type: none"> • Transmit data storage address • Transmit data size • Processing end notification call-back function
Return values	USBC_ER_t		Error code
Description	Transmits data (bulk out transfer) to the CDC peripheral device. For information on the processing end notification call-back function type USBC_CB_t, see Renesas USB Device USB Basic Firmware Application Note.		
Notes			

Table 4.33 R_usb_hcdc_SerialStateTransfer()

Name	Register HCD Serial Port State Change Receive		
Call format	R_usb_hcdc_SerialStateTransfer		
Arguments	USBC_CDC_SERIAL_ST_CB_t	*serial_st_cb	Serial port state acquisition Call-back function
Return values	USBC_ER_t		Error code
Description	Registers the serial port state change receive function. When a change in the serial port state is detected, the registered call-back function is called to send notification of the serial port state detect value. To provide notification once again after notification of a serial port state change, it is necessary to call this API a second time.		
Notes			

Table 4.34 usb_hcdc_Task()

Name	HCDC Task		
Call format	usb_hcdc_Task		
Arguments	USBC_VP_INT_t	stacd	Task start code (not used)
Return values	void	—	—
Description	Receives messages addresses to the mailbox USB_HCDC_MBX and performs processing according to the message classification.		
	Message Classification (USBC_HCDC_TCMD_Omitted)	Processing by HCDC Task	
	SEND	<ul style="list-style-type: none"> Transmits data (bulk out transfer) to the CDC peripheral device. 	
	RECEIVE	<ul style="list-style-type: none"> Receives data (bulk in transfer) from the CDC peripheral device. 	
	CLASS_REQ	<ul style="list-style-type: none"> Sends a class request to the CDC peripheral device. 	
	CLASS_NOTIFY	<ul style="list-style-type: none"> Sends a class notification SerialState receive request to the CDC peripheral device. 	
	SEND_CB	<ul style="list-style-type: none"> Receives a transmit end for the CDC peripheral device and calls the transmit request source application call-back function. 	
	RECEIVE_CB	<ul style="list-style-type: none"> Receives a receive end for the CDC peripheral device and calls the receive request source application call-back function. 	
	CLASS_REQ_CB	<ul style="list-style-type: none"> Receives a class request communication end for the CDC peripheral device and calls the request class request source application call-back function. 	
CLASS_NOTIFY_CB	<ul style="list-style-type: none"> Receives a class notification SerialState receive end from the CDC peripheral device and calls the SerialState receive request source application call-back function. 		
Notes			

Table 4.35 usb_hcdc_PipeInfo()

Name	Check Pipe Information Table		
Call format	usb_hcdc_PipeInfo		
Arguments	uint8_t	*table	Interface descriptor address
	uint16_t	speed	Connection speed (USBC_HSCONNECT /FS /LS ...)
	uint16_t	length	Interface descriptor length
Return values	USBC_ER_t		Error code
Description	This function checks the endpoint descriptor included in the interface descriptors and the HCDC pipe setting table, and it detects the usable bulk in and bulk out endpoints as well as the interrupt in endpoint. USBC_DONE is returned if bulk in and bulk out endpoints are successfully detected, and USBC_ERROR is returned if not.		
Notes			

Table 4.36 usb_hcdc_ClassRequest()

Name	Request Class Request		
Call format	usb_hcdc_ClassRequest		
Arguments	void	—	—
Return values	void	—	—
Description	<ul style="list-style-type: none"> Transmits a request class request message to HCDC. The command set in the message is as follows. USBC_HCDC_TCMD_CLASS_REQ: Request class request 		
Notes			

Table 4.37 usb_hcdc_SerialStateNotification()

Name	Class Notification SerialState Receive Processing		
Call format	usb_hcdc_SerialStateNotification		
Arguments	USBC_UTR_t	*mess	Structure for USB communication
Return values	void	—	—
Description	<ul style="list-style-type: none"> Makes class notification SerialState receive settings for the peripheral device. The pipe set for interrupt in is used to make receive settings. 		
Notes			

Table 4.38 usb_hcdc_InTransResult()

Name	USB Data Receive End Call-Back		
Call format	usb_hcdc_InTransResult		
Arguments	USBC_UTR_t	*mess	Structure for USB communication
Return values	void	—	—
Description	<ul style="list-style-type: none"> This is the call-back function corresponding to USB data receive end. Notifies HCDC of a USB data receive end message. The command set in the message is as follows. USBC_HCDC_TCMD_RECEIVE_CB: USB data receive end 		
Notes			

Table 4.39 usb_hcdc_OutTransResult()

Name	USB Data Transmit End Call-Back		
Call format	usb_hcdc_OutTransResult		
Arguments	USBC_UTR_t	*mess	Structure for USB communication
Return values	void	—	—
Description	<ul style="list-style-type: none"> This is the call-back function corresponding to USB data transmit end. Notifies HCDC of a USB data transmit end message. The command set in the message is as follows. USBC_HCDC_TCMD_SEND_CB: USB data transmit end 		
Notes			

Table 4.40 usb_hcdc_SerialStateTransResult()

Name	Class notification SerialState Receive Call-Back		
Call format	usb_hcdc_SerialStateTransResult		
Arguments	USBC_UTR_t	*mess	Structure for USB communication
Return values	void	—	—
Description	<ul style="list-style-type: none"> This is the call-back function corresponding to class notification SerialState receive. Notifies HCDC of a class notification receive message. The command set in the message is as follows. USBC_HCDC_TCMD_CLASS_NOTIFY_CB: Class notification receive 		
Notes			

Table 4.41 usb_hcdc_ClassRequestResult()

Name	Class Request Communication End Call-Back		
Call format	usb_hcdc_ClassRequestResult		
Arguments	USBC_UTR_t	*mess	Structure for USB communication
Return values	void	—	—
Description	<ul style="list-style-type: none"> This is the call-back function corresponding to class request communication end. Notifies HCDC of a class request communication end message. The command set in the message is as follows. USBC_HCDC_TCMD_CLASS_REQ_CB: Class request communication end 		
Notes			

Table 4.42 usb_hcdc_SerialStateTransResult()

Name	Serial Port State Acquisition Call-Back Processing		
Call format	usb_hcdc_SerialStateTransResult		
Arguments	USBC_UTR_t	*mess	Transfer information
Return values	void	—	—
Description	<p>This function receives notification of the line state when a line state change is detected by the CDC peripheral device and calls the serial port state change receive function registered by R_usb_hcdc_SerialStateTransfer. If the notification data is in the correct signal format, the notification result USBC_OK is sent together with UART Statebitmap (USBC_CDC_SerialState_t type). If the notification data is not in the correct signal format, the notification USBC_ERROR is sent.</p>		
Notes			

Table 4.43 usb_hcdc_ClassRequestProcess()

Name	HCDC Class Request Processing		
Call format	usb_hcdc_ClassRequestProcess		
Arguments	void	—	—
Return values	USBC_ER_t		Error code
Description	Processes class requests for the CDC peripheral device. The class requests corresponding to the API and the associated parameter types are listed below.		
	Class Request		Associated Parameter Type
	USBC_HCDC_SET_LINE_CODING		USBC_HCDC_LineCoding_t
	USBC_HCDC_GET_LINE_CODING		USBC_HCDC_LineCoding_t
	USBC_HCDC_SET_CONTROL_LINE_STATE		USBC_HCDC_ControlLineState_t
	USBC_HCDC_SEND_BREAK		USBC_HCDC_BreakDuration_t
	USBC_HCDC_SEND_ENCAPSULATED_COMMAND		USBC_HCDC_Encapsulated_t
	USBC_HCDC_GET_ENACAPSULATED_RESPONSE		USBC_HCDC_Encapsulated_t
	USBC_HCDC_SET_COMM_FEATURE		USBC_HCDC_CommFeature_t
	USBC_HCDC_GET_COMM_FEATURE		USBC_HCDC_CommFeature_t
	USBC_HCDC_CLR_COMM_FEATURE		USBC_HCDC_CommFeature_t
	<ul style="list-style-type: none"> • SET_LINE_CODING makes the communication condition settings (communication speed, data length, number of stop bits, parity) passed to it by the USBC_HCDC_LineCoding_t type. • GET_LINE_CODING acquires the communication condition settings from the device and stores them in the area specified by the argument passed to it. • SET_CONTROL_LINE_STATE makes the control line settings (DTR, RTS) passed to it by the USBC_HCDC_ControlLineState_t type. • SEND_BREAK requests break signal output for a duration specified in msec. units. When the duration is set to 0xffff, break signal output does not stop until a SEND_BREAK request with a break signal output duration of 0 is transmitted. • SEND_ENCAPSULATED_COMMAND transmits an AT command. • GET_ENACAPSULATED_RESPONSE requests a response to an AT command. • SET_COMM_FEATURE requests device-specific communication function settings. • GET_COMM_FEATURE requests the current values of device-specific communication function settings. • CLR_COMM_FEATURE clears device-specific communication function settings. • For details of the processing end notification call-back function type USBC_CB_t, see Renesas USB Device USB Basic Firmware Application Note. 		
Notes	The class request parameter (USBC_HCDC_ClassRequest_UTR_t usb_ghcdc_cls_req) is a global variable.		

5. Host CDC Sample Application Program (APL)

The host CDC application program is described below.

5.1 Operating Environment

The figure below shows a sample operating environment for the software.

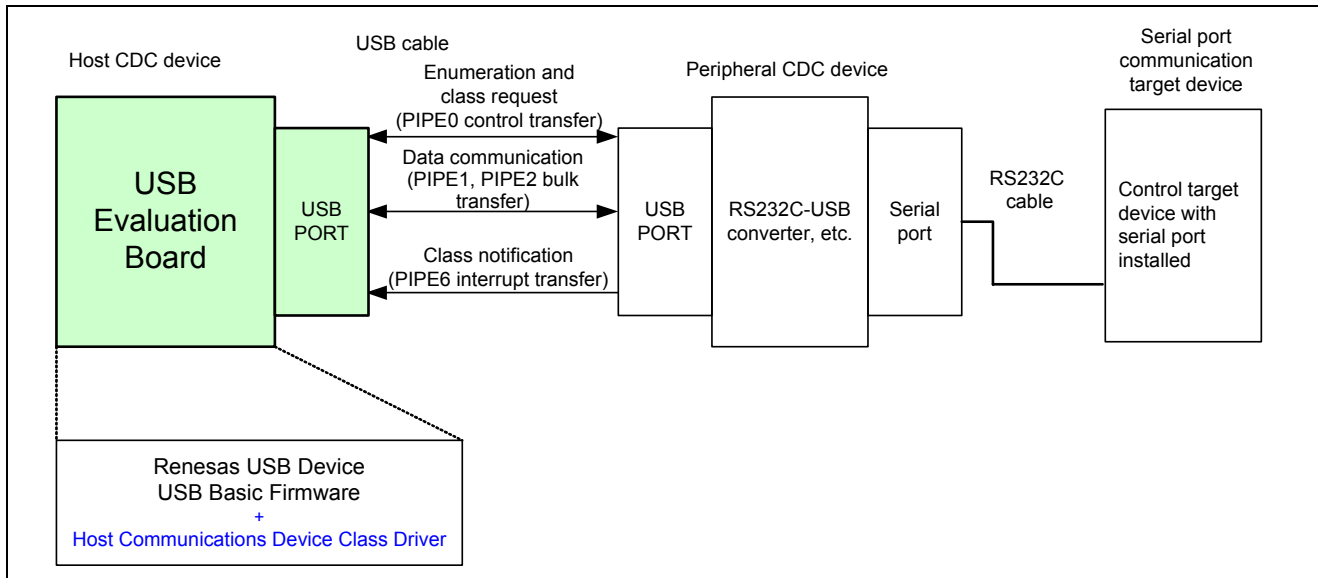


Figure 5.1 Example Operating Environment

5.2 Overview of Application Program Functions

The main functions of APL are as follows.

- (1) Communication device to set the conditions attached to the CDC.
 - (2) Performs the operation of the data sent from the CDC reception device, the operation loops back the received data.
 - (3) Make changes to the baud rate by using the SW2 and SW3 on the USB evaluation board. (Note)
- (Note) Please do not change transmission speed during a data transmission.

Specifications of the SW input are shown in Table 5.1.

Table 5.1 Operation of APL by SW input

Switch label	Operation Contents	Switch number used on RSK	
		RX62N-RSK	R8A66597 + RX62N-RSK
Baud Rate Selection SW	To select the communication speed (1200→2400→4800→....)	SW2	SW2
Baud Rate Setting SW	Chosen by Baud Rate Selection SW to set the baud rate speed	SW3	SW3

5.3 Description of Application Program Processing

- RTS and DTR settings are made by class request SET_CONTROL_LINE_STATE().
- Makes communication speed, number of data bits, number of stop bits, and parity bit settings by using class request SET_LINE_CODING.
- Acquires the communication setting values by using class request GET_LINE_CODING.
- Registers a call-back function that provides notifications of changes to the line state.
- Sends receive requests (bulk in transfer) to the CDC device and acquires the receive data.
- Transmits receive data to the CDC device by bulk out transfer (loopback).
- If receive data cannot be acquired from the CDC device, or if a receive request (bulk in transfer) is sent to the CDC device once again after completion of a loopback to the CDC device, the receive state is continuous.

5.4 Application Task (usb_hcdc_Task Function) Registration

For the μ ITRON version, register the application task in the configuration file.

For the nonOS version, see Renesas USB Device USB Basic Firmware Application Note for information on registering the application task.

Once the application task is registered as indicated above, the class checking function (table 4.19, R_usb_hcdc_Check()) is called when the CDC device is attached. If the processing result is OK, the open function (usb_hcdc_SampleApITaskOpen function) is called and the application task starts.

When the CDC device is detached, the close function is called.

The HCDC registration API (R_usb_hcdc_Registration function) is used to register the open function. For details of the HCDC registration API, see table 4.15. The HCDC registration API should be called in the initialization processing (usb_hcdc_MainInit function) of the main processing task (usb_cstd_MainTask function).

5.5 APL Global Area Variables

Table 5.2 lists the global area variables of APL

Table 5.2 Global Area Variables of APL

	Variable Name		Description
1	uint8_t	usb_ghcdc_rx_data[]	USB receive data storage area
2	uint16_t	usb_hcdc_test_SendBreak	Class request SendBreak transmit request flag
3	USBC_HCDC_LineCoding_t	usb_hcdc_LineCoding_SetParm	Communication setting parameter area for class request SetLineCoding
4	USBC_HCDC_LineCoding_t	usb_hcdc_LineCoding_GetParm	Communication setting acquisition area for class request GetLineCoding
5	uint32_t	usb_ghcdc_com_dterate	Baud rate selection Variable
6	uint32_t	usb_ghcdc_com_dterate_edit	Baud rate settings Variable
7	uint16_t	usb_ghcdc_sw_process_flg	Baud rate selection switch operation flags

5.6 APL Constant Definition

Table 5.3 APL Constant Definition

Constant Name	Value	Description
1	Host CDC application task notification command definition list	
USBC_HCDC_CMD_INIT	0x01	Initialize application
USBC_HCDC_CMD_SW_CHECK	0x02	Check switch
USBC_HCDC_CMD_RX_OK	0x10	Receive length for data receive 1 or greater
USBC_HCDC_CMD_RX_NG	0x11	Data not received
USBC_HCDC_CMD_TX_OK	0x20	Data transmit end
USBC_HCDC_CMD_TX_NG	0x21	Data transmit fail
USBC_HCDC_CMD_SET_CONTROL_ LINE_STATE	0x30	Class request SetControlLineState transmit end
USBC_HCDC_CMD_SET_LINE_CODING	0x40	Class request SetLineConfig transmit end
USBC_HCDC_CMD_SEND_BREAK	0x50	Class request SendBreak transmit end
USBC_HCDC_CMD_GET_LINE_ CODING	0x60	Class request GetLineCoding transmit and setting value acquisition end
USBC_HCDC_CMD_RCV_SERIAL_ STATE	0x70	Get control line state
USBC_HCDC_CMD_RCV_SERIAL_ STATE_NG	0x71	Control line state acquisition fail

5.7 List of APL Functions

Table 5.4 List of Functions of Sample Application

File Name	Function Name	Description	Remarks
r_usb_HCDC_apl.c	usb_hcdc_PrAplTitle	Host CDC application title display function	
	usb_hcdc_MainInit	Host CDC initialization function	
	usb_hcdc_MainLoop	Host CDC main loop	
	R_usb_hcdc_SampleAplTaskOpen	Host CDC sample application task open function	
	R_usb_hcdc_SampleAplTaskClose	Host CDC sample application task close function	
	R_usb_hcdc_SampleAplTask	Host CDC sample application processing task	
	usb_hcdc_AplInTransResult	Call-back function at in transaction end	
	usb_hcdc_AplOutTransResult	Call-back function at out transaction end	
	usb_hcdc_AplSetLine_CODING_Result	Call-back function at SetLineCoding transaction end	
	usb_hcdc_AplSetControlLineState_Result	Call-back function at SetControlLineState transaction end	
	usb_hcdc_AplSendBreak_Result	Call-back function at SendBreak transaction end	
	usb_hcdc_AplSerialStateReceive	Call-back function at line state notification	
	usb_hcdc_AplGetLine_CODING_Result	Call-back function at GetLineCoding transaction end	
	usb_hcdc_SampleAPL_Init	Function that transmits initialization request message to sample application	
	usb_hcdc_sw_process	Processing functions when pressed SW	
	usb_hcdc_sw_request	The function that send SW check request	
	usb_hcdc_get_line_coding_rcv_process	Processing when receiving the GetLineCoding	

5.8 Description of APL Functions

The functions of the application program are described below.

Table 5.5 usb_hcdc_MainInit()

Name	Initialization Processing		
Call format	usb_hcdc_MainInit		
Arguments	void	—	—
Return values	void	—	—
Description	<p>The initialization processing includes the following.</p> <ul style="list-style-type: none"> • Interrupt settings • DMA initialization settings • USB block operation start wait • Pin initialization processing • Software reset • Oscillation enable • MGR task start • HCD task start • Host CDC driver registration • HCDC task start • Hardware function settings (host/peripheral) 		
Notes			

Table 5.6 usb_hcdc_MainLoop()

Name	Main Loop Processing		
Call format	usb_hcdc_MainLoop		
Arguments	void	—	—
Return values	void	—	—
Description	<p>Waits for a task to complete within 100 msec. (uITRON version) MCD, MGR, and application task loop function (nonOS version only)</p>		
Notes			

Table 5.7 usb_hcdc_SampleApiTaskOpen()

Name	Open Processing		
Call format	usb_hcdc_SampleApiTaskOpen		
Arguments	uint16_t	devadr	Device address
	uint16_t	data2	Not used
Return values	void	—	—
Description	<p>The open processing includes the following.</p> <ul style="list-style-type: none"> • Pipe settings used by host CDC • Application task generation (uITRON version only) • Generation of mailbox used by application task (uITRON version only) • Generation of memory pool used by application task (uITRON version only) • Starting of application task operation (uITRON version only) • Issuing of class request R_usb_hcdc_SetControlLineState() (nonOS version only) 		
Notes			

Table 5.8 usb_hcdc_SampleApiTaskClose()

Name	Close Processing		
Call format	usb_hcdc_SampleApiTaskClose		
Arguments	uint16_t	data1	Not used
	uint16_t	data2	Not used
Return values	void	—	—
Description	<p>The close processing includes the following.</p> <ul style="list-style-type: none"> • Application task stop (μITRON version only) • Erasing of memory pool used by application task (μITRON version only) • Erasing of mailbox used by application task (μITRON version only) • Erasing of application task (μITRON version only) • Clearing of pipe data area 		
Notes			

Table 5.9 usb_hcdc_SampleApiTask()

Name	Application Task		
Call format	usb_hcdc_SampleApiTask		
Arguments	void	—	—
Return values	void	—	—
Description	<p>The host CDC application task processing includes the following.</p> <ul style="list-style-type: none"> • Application title display • Issuing of class request R_usb_hcdc_SetControlLineState() (μITRON version only) • Waiting for message receipt and performing processing according to the specified command after the message is received <p>For details of the processing, see figure 5.1, Outline of Application Task Processing Sequence, and table 5.21, Processing Details According to Command Classifications.</p>		
Notes			

Table 5.10 usb_hcdc_ApInTransResult()

Name	CDC Data Receive Processing (Call-Back Function)		
Call format	usb_hcdc_ApInTransResult		
Arguments	USBC_UTR_t	*mess	Structure for USB communication
Return values	void	—	—
Description	<ul style="list-style-type: none"> • This is the call-back function for the receive processing function R_usb_hcdc_InTransfer. • It is called at the end of receive processing and sends a notification message to the application task indicating that the receive operation has finished. • The commands specified by the message are as follows. USBC_HCDC_CMD_RX_OK: Receive OK (receive data length > 0) USBC_HCDC_CMD_RX_NG: No receive data (receive data length = 0) 		
Notes	<p>The message notified the application task acquires and sends the memory block from the memory pool.</p> <p>When it fails in memory block acquisition, it display error message, but do not perform the error processing.</p>		

Table 5.11 usb_hcdc_AplOutTransResult()

Name	CDC Data Transmit Processing Call-Back		
Call format	usb_hcdc_AplOutTransResult		
Arguments	USBC_UTR_t	*mess	Structure for USB communication
Return values	void	—	—
Description	<ul style="list-style-type: none"> This is the call-back function for the transmit processing function R_usb_hcdc_OutTransfer. It is called at the end of transmit processing and sends a notification message to the application task indicating that the transmit operation has finished. The command specified by the message is as follows. USBC_HCDC_CMD_TX_OK: Transmit end 		
Notes	<p>The message notified the application task acquires and sends the memory block from the memory pool.</p> <p>When it fails in memory block acquisition, it display error message, but do not perform the error processing.</p>		

Table 5.12 usb_hcdc_AplSetLine_CODING_Result()

Name	Class Request SetLineCoding Call-Back		
Call format	usb_hcdc_AplSetLine_CODING_Result		
Arguments	USBC_UTR_t	*mess	Structure for USB communication
Return values	void	—	—
Description	<ul style="list-style-type: none"> This is the call-back function corresponding to the class request R_usb_hcdc_SetLineCoding(). Sends a notification message to the application task indicating that the class request transfer operation has finished. The command specified by the message is as follows. USBC_HCDC_CMD_SET_LINE_CODING: Class request SetLineCoding end 		
Notes	<p>The message notified the application task acquires and sends the memory block from the memory pool.</p> <p>When it fails in memory block acquisition, it display error message, but do not perform the error processing.</p>		

Table 5.13 usb_hcdc_AplSetControlLineState_Result()

Name	Class Request SetControlLineState (Call-back function)		
Call format	usb_hcdc_AplSetControlLineState_Result		
Arguments	USBC_UTR_t	*mess	Structure for USB communication
Return values	void	—	—
Description	<ul style="list-style-type: none"> This is the call-back function corresponding to the class request R_usb_hcdc_SetControlLineState(). Sends a notification message to the application task indicating that the class request transfer operation has finished. The command specified by the message is as follows. USBC_HCDC_CMD_SET_CONTROL_LINE_STATE: Class request SetControlLineState end 		
Notes	<p>The message notified the application task acquires and sends the memory block from the memory pool.</p> <p>When it fails in memory block acquisition, it display error message, but do not perform the error processing.</p>		

Table 5.14 usb_hcdc_ApiGetLine_CODING_Result()

Name	Class Request GetLineCoding (Call-back function)		
Call format	usb_hcdc_ApiGetLine_CODING_Result		
Arguments	USBC_UTR_t	*mess	Structure for USB communication
Return values	void	—	—
Description	<ul style="list-style-type: none"> This is the call-back function corresponding to the class request R_usb_hcdc_GetLineCoding (). Sends a notification message to the application task indicating that the class request transfer operation has finished. The command specified by the message is as follows. USBC_HCDC_CMD_GET_LINE_CODING: Class request GetLineCoding end 		
Notes	<p>The message notified the application task acquires and sends the memory block from the memory pool.</p> <p>When it fails in memory block acquisition, it display error message, but do not perform the error processing.</p>		

Table 5.15 usb_hcdc_ApiSendBreak_Result()

Name	Class Request SendBreak (Call-back function)		
Call format	usb_hcdc_ApiSendBreak_Result		
Arguments	USBC_UTR_t	*mess	Structure for USB communication
Return values	void	—	—
Description	<ul style="list-style-type: none"> This is the call-back function corresponding to the class request R_usb_hcdc_SendBreak(). Sends a notification message to the application task indicating that the class request transfer operation has finished. The command specified by the message is as follows. USBC_HCDC_CMD_SEND_BREAK: Class request SendBreak end 		
Notes	<p>The message notified the application task acquires and sends the memory block from the memory pool.</p> <p>When it fails in memory block acquisition, it display error message, but do not perform the error processing.</p>		

Table 5.16 usb_hcdcSerialStateReceive()

Name	Line State Notification Receive Processing		
Call format	usb_hcdcSerialStateReceive		
Arguments	uint16_t	result	Line state notification receive result
	USBC_CDC_SerialState_t	SerialState	Line state bitmap
Return values	void	—	—
Description	<ul style="list-style-type: none"> This function provides notification of the line state when a change in the line state from the CDC peripheral device is detected. The line state notification receive result can be that the notification message format from the device correctly matches the class notification SerialState format or that the message format is incorrect. The serial line state is set in bit units in the line state bitmap. This function notifies the application task of the line state notification receive result and the line state bitmap. 		
Notes	<p>The message notified the application task acquires and sends the memory block from the memory pool.</p> <p>When it fails in memory block acquisition, it display error message, but do not perform the error processing.</p>		

Table 5.17 usb_hcdc_SampleAPL_Init()

Name	Initialization Processing		
Call format	usb_hcdc_SampleAPL_Init		
Arguments	void	—	—
Return values	void	—	—
Description	<ul style="list-style-type: none"> This function transmits an application initialization command to the sample application task. The command set in the message is as follows. USBC_HCDC_CMD_INIT: Application initialization command 		
Notes	<p>The message notified the application task acquires and sends the memory block from the memory pool.</p> <p>When it fails in memory block acquisition, it display error message, but do not perform the error processing.</p>		

Table 5.18 usb_hcdc_sw_process ()

Name	Processing when pressed SW		
Call format	usb_hcdc_sw_process		
Arguments	void	—	—
Return values	void	—	—
Description	<p>To be processed when switch was pressed.</p> <p>When pressed Baud Rate Selection SW Select baud rate. (1200 2400 4800 ... 57600 1200 ...)</p> <p>When pressed Baud Rate Setting SW Set the baud rate selected in the Baud Rate Selection SW.</p>		
Notes			

Table 5.19 usb_hcdc_sw_request ()

Name	Send SW check request		
Call format	usb_hcdc_sw_request		
Arguments	void	—	—
Return values	void	—	—
Description	Request message that processed when SW pressed, it to notify the Application Tasks. Command to set the message: USBC_HCDC_CMD_SW_CHECK		
Notes			

Table 5.20 usb_hcdc_get_line_coding_rcv_process ()

Name	Get Line Coding Receive Processing		
Call format	usb_hcdc_get_line_coding_rcv_process		
Arguments	void	—	—
Return values	void	—	—
Description	The serial port settings, output PRINT, and make the LCD display of communication speed.		
Notes			

5.9 Outline of Application Task Processing Sequence

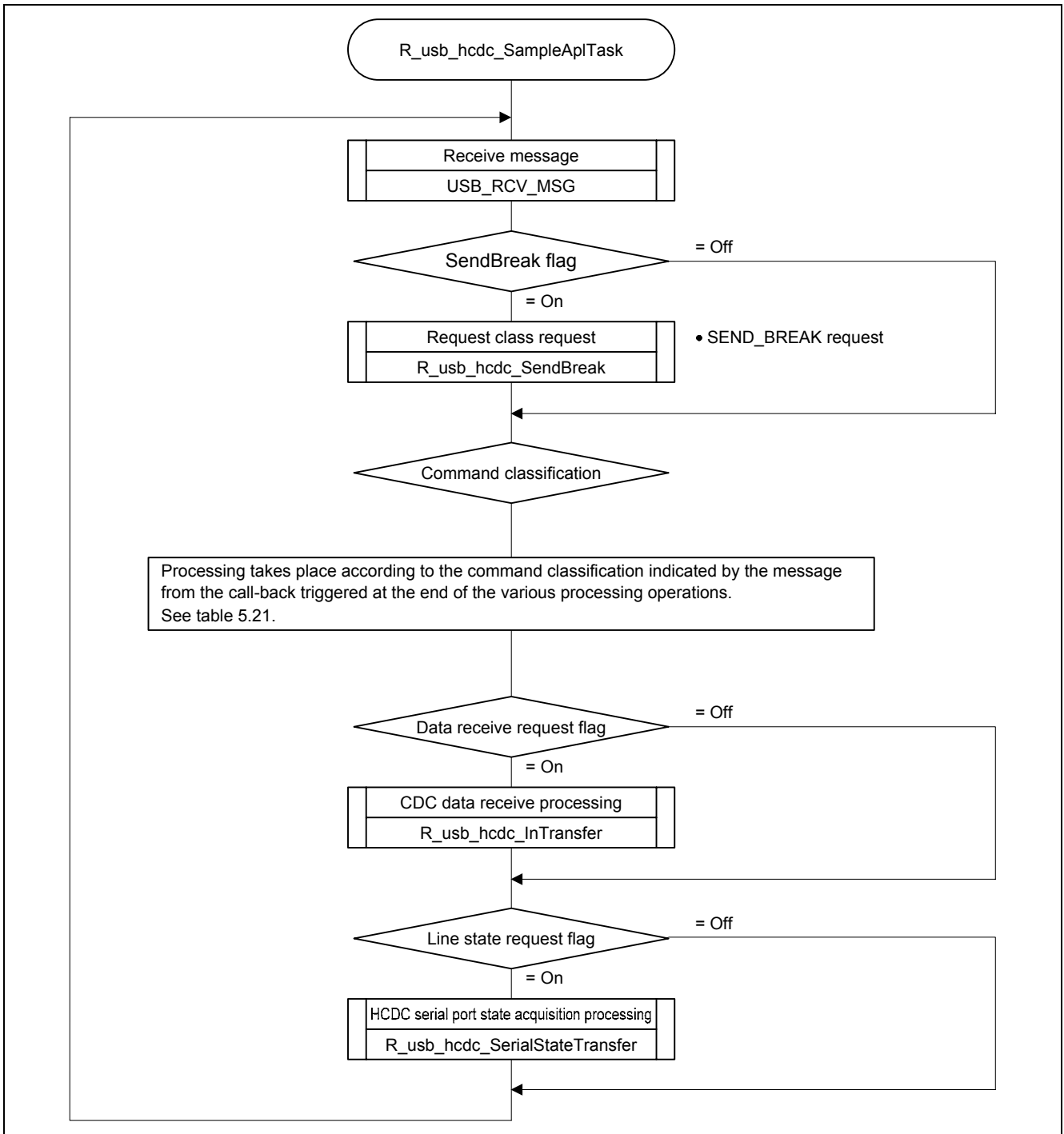


Figure 5.2 Outline of Application Task Processing Sequence

Table 5.21 Processing Details According to Command Classifications

Command Classification	Processing Details	Remarks
USBC_HCDC_CMD_INIT: Application initialization	<ul style="list-style-type: none"> Application title display Request for class request SetControlLineState (line state setting): R_usb_hcdc_SetControlLineState() 	
USBC_HCDC_CMD_RX_OK: Acquire receive data (receive length > 0)	<ul style="list-style-type: none"> Display receive data: SB_PRINTF1(); Loopback transmit: R_usb_hcdc_OutTransfer(); 	
USBC_HCDC_CMD_RX_NG: Do not acquire receive data (receive length = 0)	Set data receive request flag to ON	
USBC_CDC_CMD_TX_OK: Data transmit end		
USBC_CDC_CMD_TX_NG: Data transmit fail		
USBC_HCDC_CMD_SET_CONTROL_LINE_STATE: Class request SetControlLineState transfer end	Request class request SetLineCoding (communication condition settings): R_usb_hcdc_SetLineCoding();	
USBC_HCDC_CMD_SET_LINE_CODING: Class request SetLineCoding transfer end	Request class request GetLineCoding (acquire communication conditions): R_usb_hcdc_GetLineCoding();	
USBC_HCDC_CMD_GET_LINE_CODING: Class request GetLineCoding transfer end	<ul style="list-style-type: none"> Display acquired communication conditions Set data receive request flag to ON SetLineState request flag to ON 	
USBC_HCDC_CMD_SEND_BREAK: Class request SendBreak transfer end	Display SendBreak end message	
USBC_HCDC_CMD_RCV_SERIAL_STATE: Receive class notification SerialState	<ul style="list-style-type: none"> Display line state SetLineState request flag to ON 	
USBC_HCDC_CMD_RCV_SERIAL_STATE_NG: Receive class notification SerialState fail	SetLineState request flag to ON	
USBC_HCDC_CMD_SW_CHECK	<ul style="list-style-type: none"> Select baud rate by switch input Set baud rate Set serial port baud rate 	
Undefined command	Display received command	

5.10 Sequences

The operation sequences of the sample application program are described below.

5.10.1 Startup to CDC Device Attachment Sequence

The sequence from sample application program startup through completion of enumeration, application task startup, and completion of pipe hardware configuration is illustrated in figures 5.3 and 5.4.

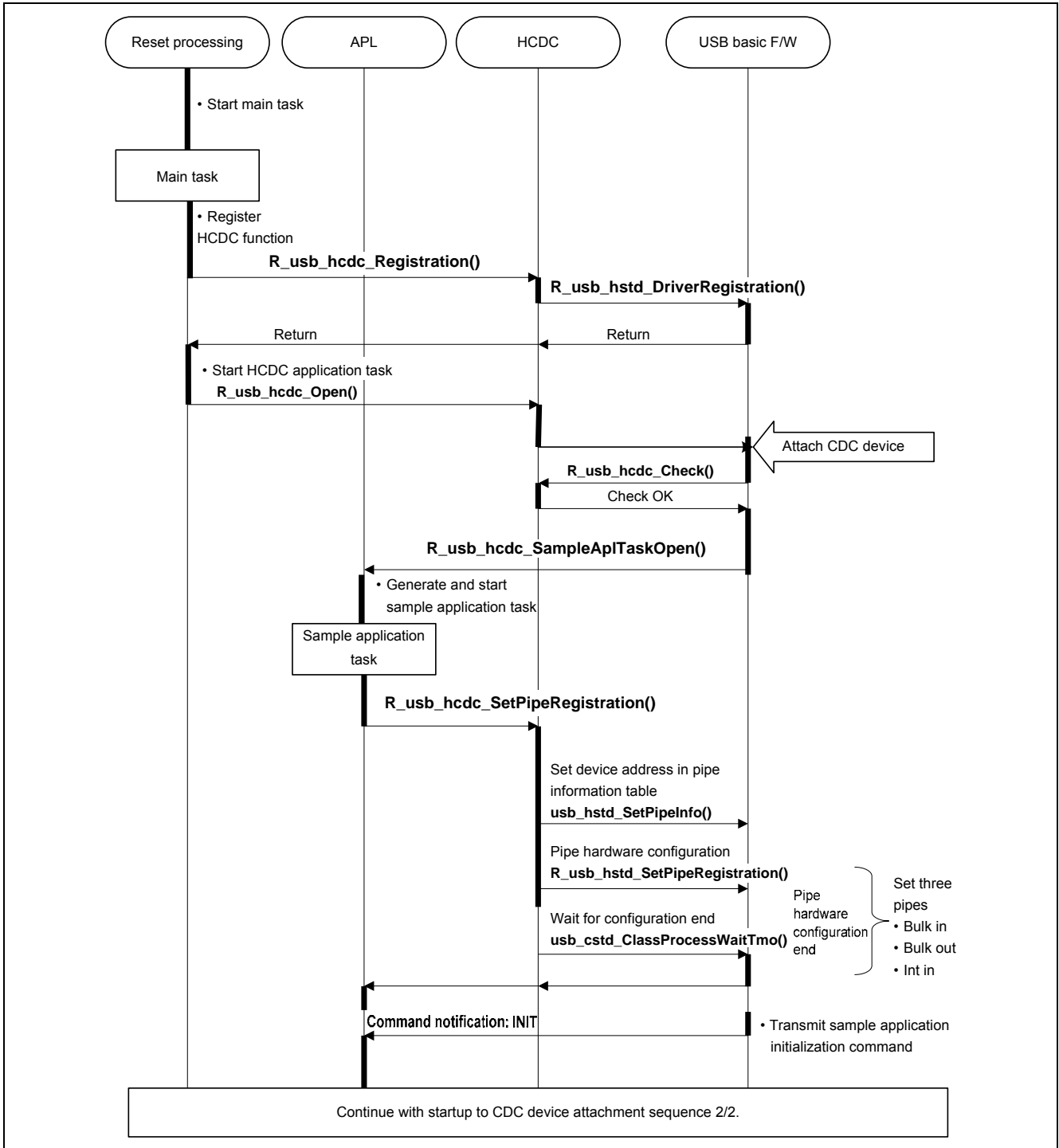


Figure 5.3 Startup to CDC Device Attachment Sequence (1/2)

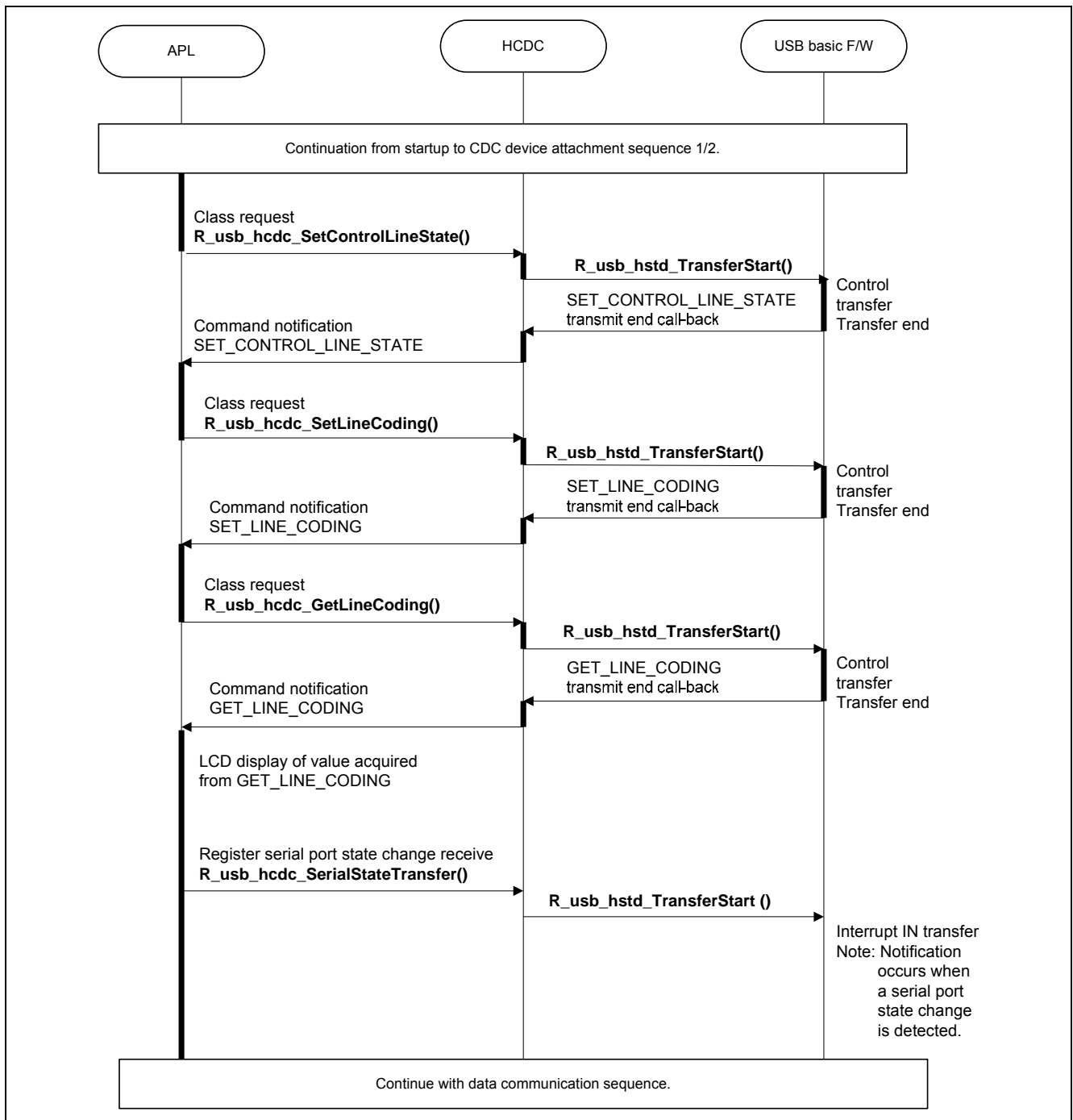


Figure 5.4 Startup to CDC Device Attachment Sequence (2/2)

5.10.2 Data Communication Sequence

The data communication sequence is illustrated in figure 5.5.

In the example below, receive data is present for the third data receive operation only.

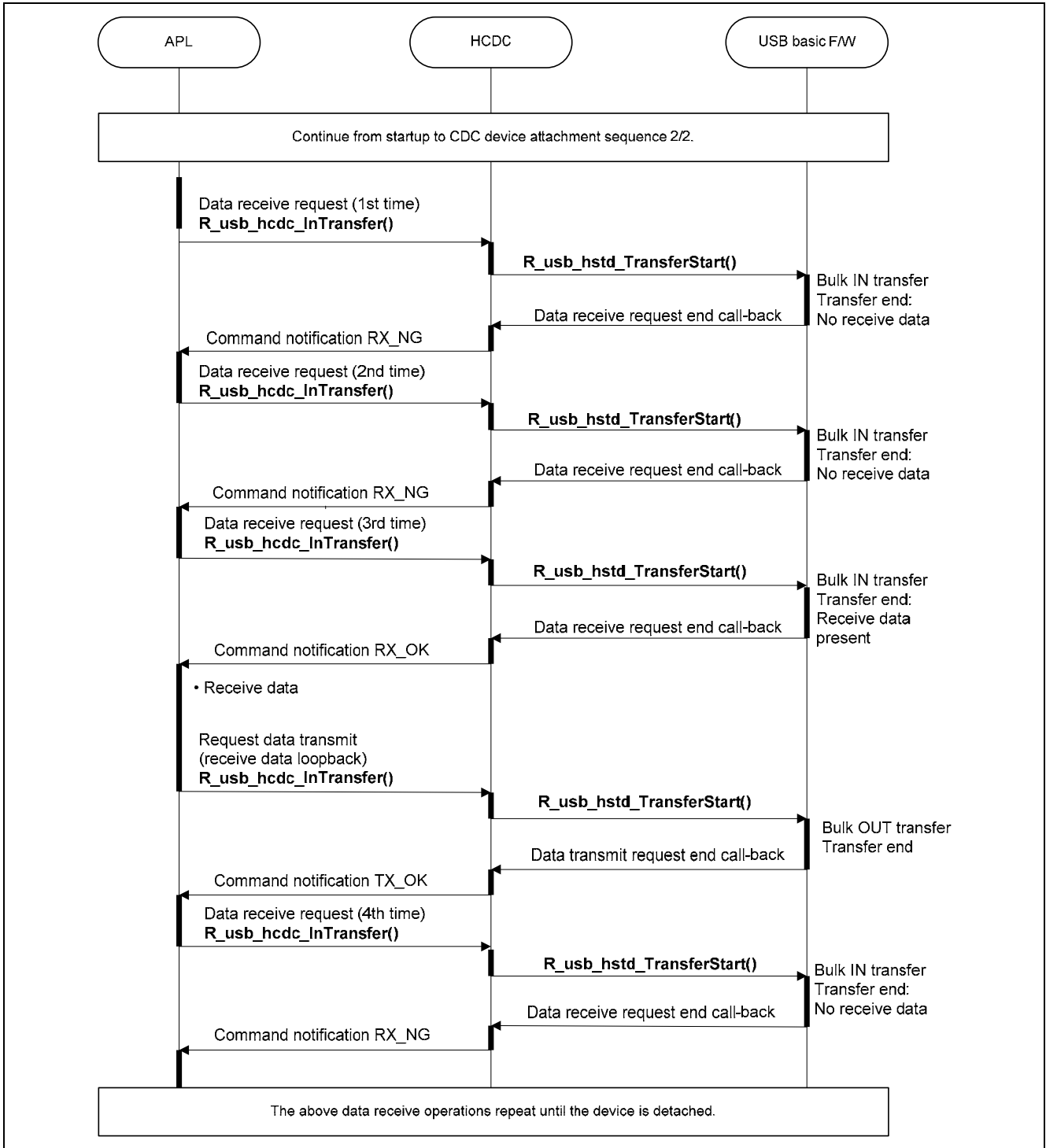


Figure 5.5 Data Communication Sequence

5.10.3 BREAK Signal Output

The BREAK signal output sequence is illustrated in figure 5.6.

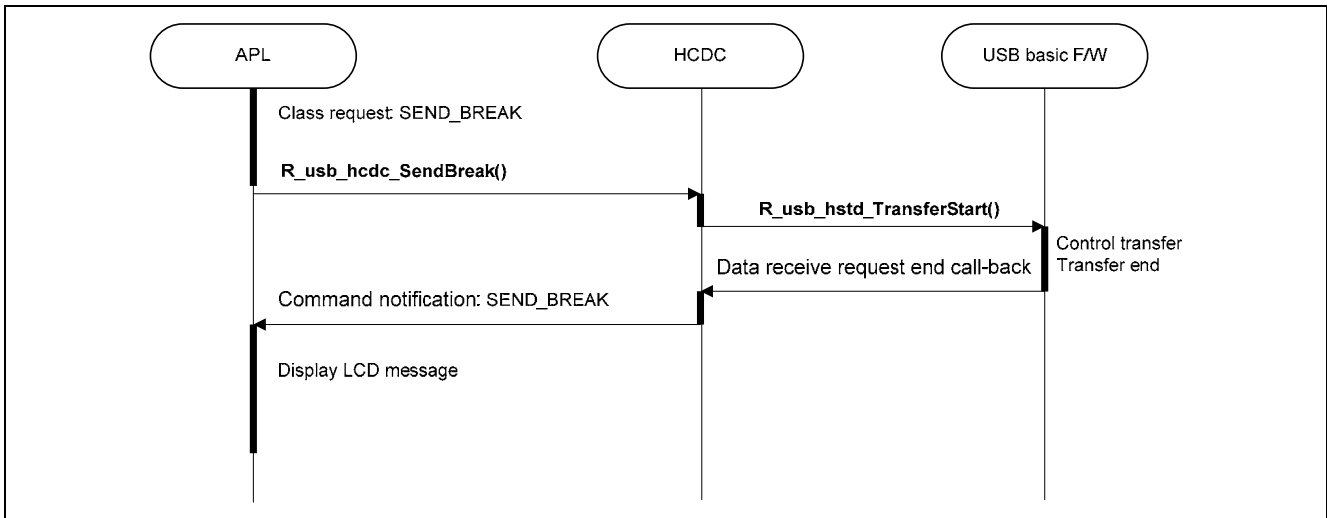


Figure 5.6 BREAK Signal Output Sequence

5.10.4 Serial Port State Change Notification

The serial port state change notification sequence is illustrated in figure 5.7.

Serial port state change receive must be registered before a serial port state change occurs.

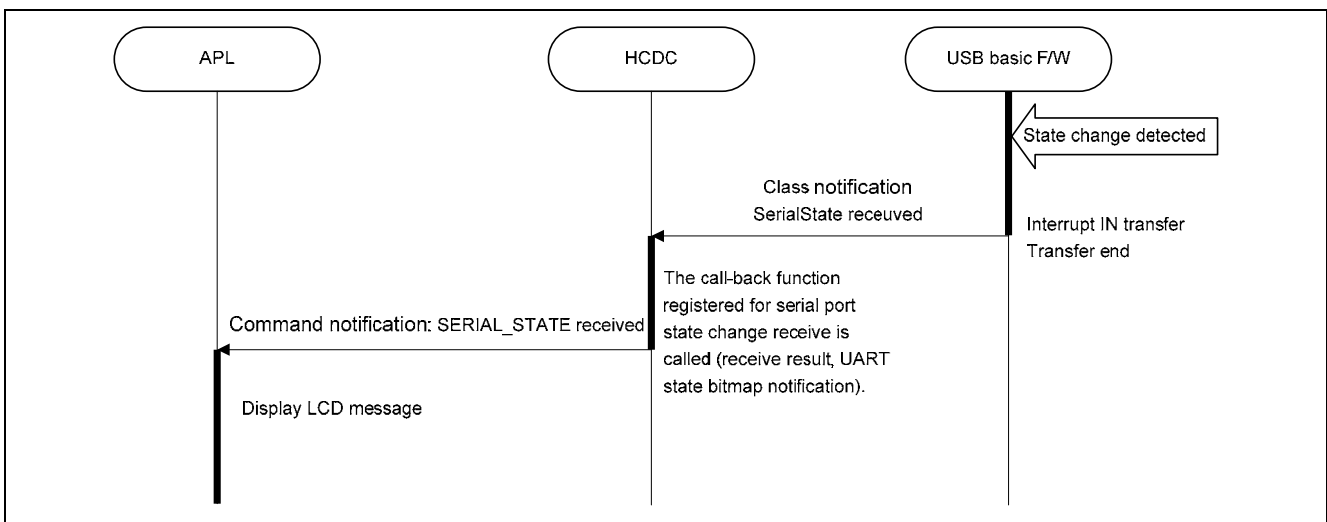


Figure 5.7 Serial Port State Change Notification Sequence

5.10.5 CDC Device Detach

The sequence when the CDC device is detached is illustrated in figure 5.8.

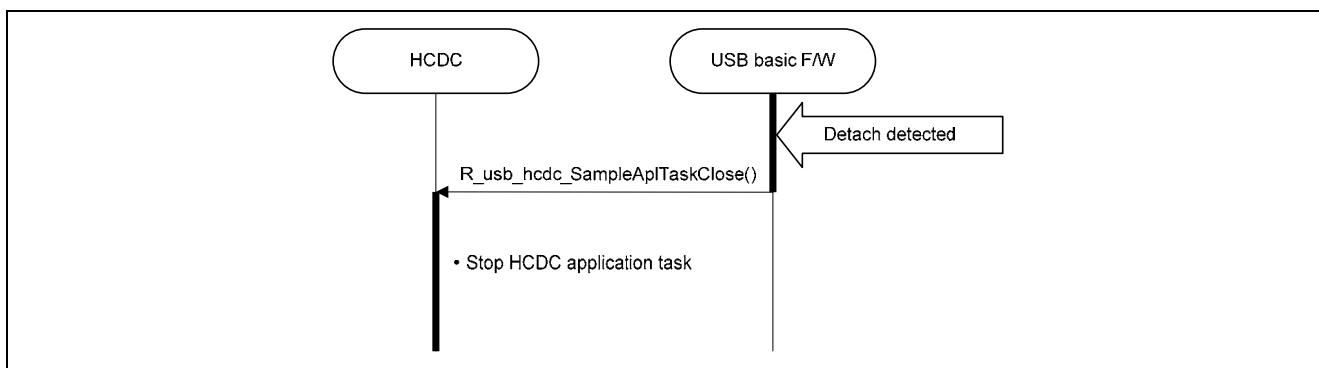


Figure 5.8 CDC Device Detach Sequence

6. Limitations

HDCD is subject to the following limitations.

1. The structures contain members of different types. (Depending on the compiler, this may cause address misalignment of structure members.)
2. Devices can connect to sample F/W this is the only one. Please do not connect two or more devices simultaneously.

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	Mar.22.11	—	First edition issued
1.10	July.11.07	—	Add Target Device RX630, R8A66597
		30	Add the information on RX630 and R8A66597(Hi-Speed USB) 5 Host CDC Sample Application Program (APL) <ul style="list-style-type: none">• Change the baud rate settings when pressing SW2• Set baud rate when pressing SW3

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins

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

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

2. Processing at Power-on

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

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

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

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

3. Prohibition of Access to Reserved Addresses

Access to reserved addresses is prohibited.

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

4. Clock Signals

After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

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

5. Differences between Products

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

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

Notice

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



SALES OFFICES

Renesas Electronics Corporation

<http://www.renesas.com>

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

Renesas Electronics America Inc.

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

Renesas Electronics Canada Limited

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

Renesas Electronics Europe Limited

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

Renesas Electronics Europe GmbH

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

Renesas Electronics (China) Co., Ltd.

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

Renesas Electronics (Shanghai) Co., Ltd.

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

Renesas Electronics Hong Kong Limited

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

Renesas Electronics Taiwan Co., Ltd.

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

Renesas Electronics Singapore Pte. Ltd.

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

Renesas Electronics Malaysia Sdn.Bhd.

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

Renesas Electronics Korea Co., Ltd.

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