R01AN3429EJ0110
Rev.1.10
Sep 30, 2016

RZ/A1H Group

USB Peripheral Communications Device Class Driver (PCDC)

Introduction

This application note describes USB Peripheral Communication Device Class Driver (PCDC). This module performs hardware control of USB communication. It is referred to below as the USB-BASIC-F/W.

Target Device

RZ/A1H Group

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Related Documents

1. Universal Serial Bus Revision 2.0 specification
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. RZ/A1H Group, RZ/A1M Group User's Manual: Hardware (Document No.R01UH0403EJ)
5. RZ/A1H Group USB Host and Peripheral Interface Driver (Document No.R01AN3291EJ)
6. RZ/A1H Group Downloading Program to NOR Flash Memory Using ARM® Development Studio 5 (DS-5™) Semihosting Function (for GENMAI) (Document No.R01AN1957EJ)
7. RZ/A1H Group I/O definition header file (Document No.R01AN1860EJ)
8. RZ/A1H Group Example of Initialization (for GENMAI) (Document No.R01AN1864EJ)

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

USB Devices Page
http://www.renesas.com/prod/usb/

Contents

1. Overview

The USB PCDC, when used in combination with the USB-BASIC-F/W, operates as a USB peripheral communications device class driver (PCDC). The PCDC conforms to the abstract control model of the USB communication device class specification (CDC) and enables communication with a USB host.
This module supports the following functions.

- Data transfer to and from a USB host
- Response to CDC class requests
- Provision of communication device class notification transmit service

1.1 Please be sure to read

It is recommended to use the APIs described in the document (Document No: R01AN3291EJ) when creating an application program using this driver.
That document is located in the "reference_documents" folder within the package.
[Note]

a. The document (Document No: R01AN3291EJ) also provides how to create an application program using the APIs described above.
b. If the APIs described in the document (Document No: R01AN3293JJ) are used, there is no need to use the API described in “6.2 PCDC API Functions” of this document of this document.

1.2 Operation Confirmation Conditions

The operation of the USB Driver module has been confirmed under the conditions listed in Table 1.1.

Table 1.1 Operation Confirmation Conditions

Item	Description
MCU	RZ/A1H
Operating frequency (Note)	CPU clock (Iφ): 400 MHz
	Image-processing clock (Gφ): 266.37 MHz
	Internal bus clock (Bφ): 133.33 MHz
	Peripheral clock 1 (P1φ): 66.67 MHz
	Peripheral clock 0 (P0φ): 33.33 MHz
Operating voltage	Power supply voltage (I/O): 3.3 V
	Power supply voltage (internal): 1.8 V
Integrated development environment	ARM Integrated Development Environment
	• ARM Development Studio (DS-5™) Version 5.16
	IAR Integrated Development Environment
	• IAR Embedded Workbench for ARM Version 7.40
Compiler	ARM C/C++ Compiler/Linker/Assembler Ver.5.03 [Build 102]
	KPIT GNUARM-RZ v14.01
	IAR C/C++ Compiler for ARM 7.40
Operating mode	Boot mode 0
	(CS0-space 16-bit booting)
Communication setting of terminal software	Communication speed: 115200 bps
	Data length: 8 bits
	Parity: None
	Stop bit length: 1 bit
	Flow control: None
Board	GENMAI board
	R7S72100 CPU board (RTK772100BC00000BR)
Device (Functions used on the board)	Serial interface (D-sub 9-pin connector)
	USB1 connector, USB2 connector

1.3 Limitations

This module is subject to the following restrictions.

1. Structures are composed of members of different types (Depending on the compiler, the address alignment of the structure members may be shifted).

Terms and Abbreviations

ACM	:	Abstract Control Model. This is the USB interface subclass used for virtual COM ports, based in the old V.250 (AT) command standard. See PSTN below.
APL	:	Application program
CDC	:	Communications Devices Class
CDCC	:	Communications Devices Class Communications Interface Class
CDCD	:	Communications Devices Class Data Class Interface
CPD	:	Serial Communication Port Driver
cstd	:	Prefix of function and file for Host & Peripheral USB-BASIC-FW
non-OS	:	USB basic firmware for OS less system
PCD	:	Peripheral control driver of USB-BASIC-F/W
PCDC	:	Communications Devices Class for peripheral
PCDCD	:	Peripheral Communications Devices Class Driver
PP	:	Pre-processed definition
pstd	:	Prefix of function and file for Peripheral USB-BASIC-FW
PSTN	:	Public Switched Telephone Network, contains the ACM (above) standard.
Scheduler	:	Used to schedule functions, like a simplified OS.
Scheduler Macro	:	Used to call a scheduler function (non-OS)
SCI	:	Serial Communication Interface
Task	:	Processing unit
USB	:	Universal Serial Bus
USB-BASIC-FW	:	USB basic firmware for Renesas USB device (non-OS)

2. Software Configuration

Figure 2-1 shows the configuration of the modules related to PCDC.

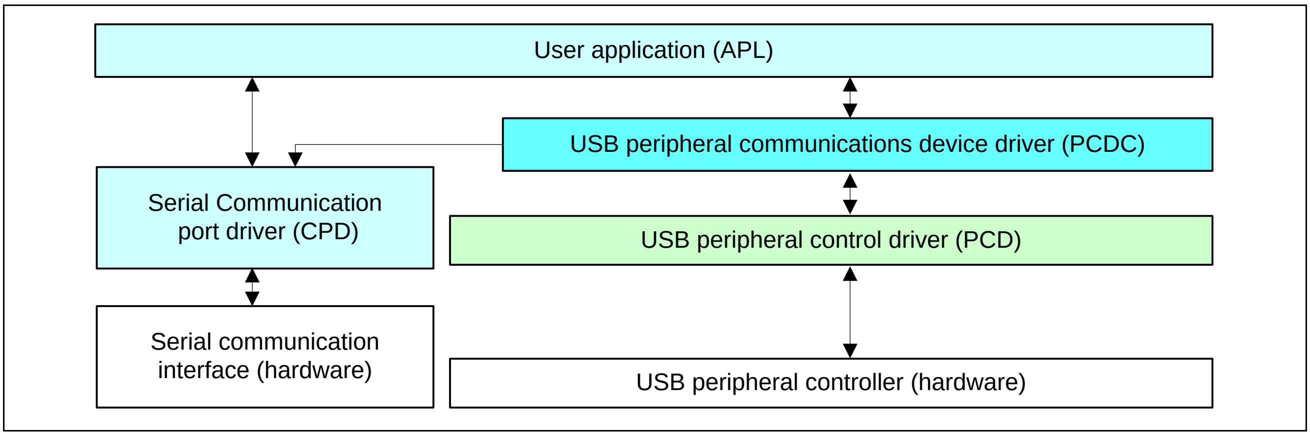

Figure 2-1 Source Code Block Diagram

Table 2.1 Modules

Module	Description
APL	User application program.
PCDC	Sends requests from the APL for requests and data communication involving the CDC to the PCD.
PCD	USB peripheral hardware control driver. (Basic USB FW.)
CPD	Serial port control driver

3. System Resources

The resource which PCDC uses is showed in Table 3.1 - Table 3.3.

Table 3.1 Task Information

Function	Task ID	Priority	Description
usb_pcdc_Task	USB_PCDC_TSK	USB_PRI_3	PCDC Task

Table 3.2 Mailbox Information

Mailbox	Mailbox ID	Queue	Description
USB_PCDC_MBX	USB_PCDC_TSK	FIFO order	for PCDC

Table 3.3 Memory Pool Information

Fixed Memorypool	Queue	Memory Block(*)	Description
USB_PCDC_MPL	FIFO order	40byte	for PCDC

Note: The maximum number of memory blocks for the entire system is defined in USB_BLKMAX.
The default value is 20.

4. Compile Setting

In order to use this module, it is necessary to set the USB-BASIC-F/W FIT module as a peripheral. Refer to USB Basic Firmware application note (Document No. R01AN3291JEJ) for information on USB-BASIC-F/W settings.
Please modify r_usb_pcdc_config.h when User sets the module configuration option.
The following table shows the option name and the setting value.

Configuration options in r_usb_pcdc_config.h	
USB_PCDC_USE_PIPE_IN USB_PCDC_USE_PIPE_OUT	Specifies the pipe number which is used at the data transfer. (Specifies any one from USB_PIPE1 to USB_PIPE5)
USB_PCDC_USE_PIPE_STATUS	Specifies the pipe number which is used at the class notification. (Specifies any one from USB_PIPE6 to USB_PIPE9)
USB_UART_ENABLE	Enables this definition when using UART.

5. CDC, PSTN, and ACM (Abstract Control Model)

5.1 Basic Functions

This software conforms to the Abstract Control of the CDC PSTN Subclass. See.5.2 below.
The main functions of the PCDC firmware:

1. Respond to functional inquiries from the USB Host
2. Respond to class requests from the USB Host
3. Data communication with the USB Host
4. Notify the USB Host of serial communication errors

5.2 Abstract Control Model Overview

The Abstract Control Model subclass of CDC 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. The class requests and class notifications supported are listed below.

5.2.1 Class Requests (Host to Peripheral)

Table 5.1 shows CDC class requests, and whether they are supported.

Table 5.1 CDC class requests

Request	Code	Description	Supported
SendEncapsulatedCommand	0x00	Transmits AT commands, etc., defined by the protocol.	No
GetEncapsulatedResponse	0x01	Requests a response to a command transmitted by SendEncapsulatedCommand.	No
SetCommFeature	0x02	Enables or disables features such as device-specific 2-byte code and country setting.	No
GetCommFeature	0x03	Acquires the enabled/disabled state of features such as device-specific 2-byte code and country setting.	No
ClearCommFeature	0x04	Restores the default enabled/disabled settings of features such as device-specific 2-byte code and country setting.	No
SetLineCoding	0x20	Makes communication line settings (communication speed, data length, parity bit, and stop bit length).	Yes
GetLineCoding	0x21	Acquires the communication line setting state.	Yes
SetControlLineState	0x22	Makes communi d f r g cation line control signal (RTS, DTR) settings.	Yes
SendBreak	0x23	Transmits a break signal.	No

For details concerning the Abstract Control Model requests, refer to Table 11, “Requests - Abstract Control Model” in “USB Communications Class Subclass Specification for PSTN Devices”, Revision 1.2.

5.2.2 Data Format of Class Requests

The data format of the class requests supported by the class driver software is described below.

(1). SetLineCoding

This is the class request the host transmits to perform the UART line setting.
The SetLineCoding data format is shown below.

Table 5.2 SetLineCoding Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	SET_LINE_CODING (0x20)	0x00	0x00	0x07	Line Coding Structure See Table 5.3 Line Coding Structure Format

Table 5.3 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
6	BdataBits	1	Number	Data bits (5, 6, 7, 8)

The following shows the setting that this S/W supports.

DwDTERate: 1200bps/2400bps/4800bps/9600bps/14400bps/19200bps/38400bps/57600bps/115200bps
BcharFormat: 1Stop bit/ 2 Stop bit
BparityType: None/Odd/Even
BdataBits: 7bit/8bit

(2). GetLineCoding

This is the class request the host transmits to request the UART line state.
The GetLineCoding data format is shown below.

Table 5.4 SetLineCoding Format

bmRequestType	bRequest	wValue	WIndex	wLength	Data
0xA1	GET_LINE_CODING (0x21)	0x00	0x00	0x07	Line Coding Structure See Table 5.3 Line Coding Structure Format

(3). SetControlLineState

This is a class request that the host sends to set up the signal for flow controls of UART.
This software does not support RTS/DTR control.
The SET_CONTROL_LINE_STATE data format is shown below.

Table 5.5 SET_CONTROL_LINE_STATE Format

bmRequestType	bRequest	WValue	wIndex	wLength	Data
0x21	SET_CONTROL_ LINE_STATE (0x22)	Control Signal Bitmap See Table 5.6 Control Signal Bitmap	0x00	0x00	None

Table 5.6 Control Signal Bitmap

Bit Position	Description	
D15 to D2	Reserved (reset 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

5.2.3 Class Notifications (Peripheral to Host)

Whether or not a class notification is supported is shown in Table 5.7.

Table 5.7 CDC Class Notifications

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

(1). Serial State

The host is notified of the serial state when a change in the UART port state is detected.
This software supports the detection of overrun, parity and framing errors. A state notification is performed when a change from normal state to error is detected. However, notification is not continually transmitted when an error is continually detected.
The SerialState data format is shown below.

Table 5.8 SerialState Format

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0xA1	SERIAL_STATE (0x20)	0x00	0x00	0x02	UART State bitmap See Table 5.9 UART state bitmap format

Table 5.9 UART state bitmap format

Bits	Field	Description	Supported
D15～D7		Reserved	-
D6	bOverRun	Overrun error detected	Yes
D5	bParity	Parity error detected	Yes
D4	bFraming	Framing error detected	Yes
D3	bRingSignal	INCOMING signal (ring signal) detected	No
D2	bBreak	Break signal detected	No
D1	bTxCarrier	Data Set Ready: Line connected and ready for communication	No
D0	bRxCarrier	Data Carrier Detect: Carrier detected on line	No

5.3 PC Virtual COM-port Usage

The CDC device can be used as a virtual COM port when operating in Windows OS.
Use a PC running Windows OS, and connect an RSK board. After USB enumeration, the CDC class requests *GetLineCoding* and *SetControlLineState* are executed by the target, and the CDC device is registered in Windows Device Manager as a virtual COM device.
Registering the CDC device as a virtual COM-port in Windows Device Manager enables data communication with the CDC device via a terminal app such as “HyperTerminal” which comes standard with Windows OS. When changing settings of the serial port in the Windows terminal application, the UART setting is propagated to the firmware via the class *request SetLineCoding*.
Data input (or file transmission) from the terminal app window is transmitted to the evaluation board using endpoint 2 (EP2); data from the evaluation board side is transmitted to the PC using EP1.
When the last packet of data received is the maximum packet size, and the terminal determines that there is continuous data, the received data may not be displayed in the terminal. If the received data is smaller than the maximum packet size, the data received up to that point is displayed in the terminal.
(The maximum packet size for Full-Speed is 64 bytes.)
The received data is outputed on the terminal when the data less than Maximum packet size is received.

6. USB Peripheral Communication Device Class Driver (PCDC)

6.1 Basic Functions

The basic functions of PCDC are as follows.

1. Provides data transmission and reception services to the USB host.
2. Responds to CDC class requests.
3. Provides a CDC notification transmission service.

6.2 PCDC API Functions

Table 6.1 shows all the PCDC API functions.

[Note]

If you want to use the API, which is described in USB Host and Peripheral Interface Driver (Document No: R01AN3291EJ), in the application program, you do not need to use the following API.

Table 6.1 PCDC API Functions

Function	Description
R_usb_pcdc_SendData	Sends a data transmit request message to PCDC task.
R_usb_pcdc_ReceiveData	Sends a data receive request message to PCDC task.
R_usb_pcdc_SerialStateNotification	Sends a serial status message class notification to the PCDC task.
R_usb_pcdc_usr_ctrl_trans_function	Control transfer processing for CDC
R_usb_pcdc_task	PCDC task
R_usb_pcdc_driver_start	PCDC driver start processing.

6.2.1 R_usb_pcdc_SendData

Transfer USB data

Format

void R_usb_pcdc_SendData (USB_UTR_t* ptr, uint8_t* buf, uint32_t size, USB_CB_t complete)

Argument

ptr	Pointer to a USB Transfer Structure
buf	Pointer to buffer containing data to transmit
size	Transfer size
complete	Process completion callback function

Return Value

—

Description

This function transfers the specified USB data of the specified size from the address specified in the Transmit Data Address (buf).
When the transmission is done, the call-back function 'complete' is called.

Note

1. Please set the following members of the USB_UTR_t structure before calling the function.

USB_REGADR_t	ipp	: USB register base address
uint16_t	ip	: USB IP Number

2. Specify the area other than the auto variable (stack) area to the 2nd argument.
3. The USB transmit process results are found via the USB_UTR_t pointer in the call-back function's arguments.
4. See "USB Communication Structure" (USB_UTR_t) in the USB Basic Firmware application note.

Example

```
void usb_apl_task( void )
{
  USB_UTR_t  *ptr;

  ptr = (USB_UTR_t *)&utr;
  ptr->ip = USB_PERI_USBIP_NUM;                  /* USB IP number set */
  ptr->ipp = R_usb_cstd_GetUsbIpAdr( ptr->ip );  /* USB IP base address set */

  R_usb_pcdc_SendData(ptr, send_data, size, (USB_CB_t)&usb_complete);
}

/* Callback function */
void  usb_complete( USB_UTR_t *mess, uint16_t data1, uint16_t data2 )
{
  /* Processing at the time of the completion of USB transmitting */
}
```

6.2.2 R_usb_pcdc_ReceiveData

Issue a data receive request to USB driver

Format

void R_usb_pcdc_ReceiveData (USB_UTR_t *ptr, uint8_t *buf, uint32_t size, USB_CB_t complete)

Argument

ptr	Pointer to a USB transfer structure
buf	Pointer to transmission data buffer address
size	Transfer size
complete	Process completion callback function

Return Value

—

Description

This function requests USB data reception of the USB driver.
When the data of the size specified by 3rd argument is received or the data of less than max packet size is received from USB, callback function is called.
The received data is stored in the area that is specified by the second argument.

Note

1. Please set the following members of the USB_UTR_t structure before calling the function.

USB_REGADR_t	ipp	: USB register base address
uint16_t	ip	: USB IP Number

2. Specify the area other than the auto variable (stack) area to the 2nd argument.
3. When the received data is n times of the maximum packet size and less than the specified size in the argument (*size*), it is considered that the data transfer is not ended and a callback function (*complete*) is not generated.
4. The USB transmit process results are found via the USB_UTR_t pointer in the call-back function's arguments.
5. See "USB Communication Structure" (USB_UTR_t) in the USB Basic Firmware application note.

Example

```
void usb_smp_task( void )
{
                :
  g_utr.ip = USB_PERI_USBIP_NUM;                  /* USB IP number set */
  g_utr.ipp = R_usb_cstd_GetUsbIpAdr( ptr->ip ); /* USB IP base address set */

  R_usb_pcdc_ReceiveData(&g_utr, receive_data, size,(USB_CB_t)&usb_complete);
}

/* Callback function */
void  usb_complete( USB_UTR_t *mess, uint16_t data1, uint16_t data2 )
{
  /* Processing at the time of the completion of USB reception */
}
```

6.2.3 R_usb_pcdc_SerialStateNotification

Transmit SerialState class notification to host

Format

void R_usb_pcdc_SerialStateNotification (USB_UTR_t *ptr, uint16_t serial_state, USB_CB_t complete)

Argument

*ptr	Pointer to a USB Transfer Structure
serial_state	Serial status
complete	Process completion callback function

Return Value

—

Description

The CDC class notification “Serial State” is transmitted to the USB host.
Serial State transmission uses an interrupt pipe (EP3 in the code).
A callback function is called after completing the transmission.

Note

1. Refer to “Table 5.9 UART state bitmap format” for the bit pattern of Serial Status.
2. Please set the following members of the USB_UTR_t structure before calling the function.

USB_REGADR_t	ipp	: USB register base address
uint16_t	ip	: USB IP Number

3. The USB transmit process results are found via the USB_UTR_t pointer in the call-back function’s arguments.
4. See "USB Communication Structure" (USB_UTR_t) in the USB Basic Firmware application note.

Example

```
{
  USB_UTR_t  utr;
  USB_UTR_t  *ptr;
  uint16_t   state;       /* Serial state */

  ptr = (USB_UTR_t *)&utr;
  ptr->ip = USB_PERI_USBIP_NUM;                      /* USB IP number set */
  ptr->ipp = R_usb_cstd_GetUsbIpAdr( ptr->ip );  /* USB IP base address set */

   state = 0x0020;        /* D5 : Parity error */
  R_usb_pcdc_SerialStateNotification(ptr, state, (USB_CB_t)&usb_complete);
}

/* Callback function */
void  usb_complete( USB_UTR_t *mess, uint16_t data1, uint16_t data2 )
{
  /* Processing at the time of the completion of serial State transmitting */
}
```

6.2.4 R_usb_pcdc_usr_ctrl_trans_function

Control transfer processing for CDC

Format

void	R_usb_pcdc_usr_ctrl_trans_function(USB_UTR_t *ptr, USB_REQUEST_t *preq, uint16_t ctsq)

Argument

*ptr	Pointer to a USB Transfer Structure		
*preq	Pointer to a Class request message		
ctsq	Control transfer stage information		
		USB_CS_IDST	Idle or setup stage
		USB_CS_RDDS	Control read data stage
		USB_CS_WRDS	Control write data stage
		USB_CS_WRND	Control write no data status stage
		USB_CS_RDSS	Control read status stage
		USB_CS_WRSS	Control write status stage
		USB_CS_SQER	Sequence error

Return Value

—

Description

When the request type is a CDC class request, this function calls the processing that corresponds to the control transmit stage.
Register this API to the member "*ctrltrans*" in USB_PCDREG_t structure as the call-back function to be called at the time of CDC control transfer. This callback must be registered earlier during device class "driver registration".

Note

—

Example

```
void usb_apl_task( void )
{
  USB_PCDREG_t  driver;

        :
  /* Control Transfer */
  driver.ctrltrans = (USB_CB_TRN_t)&R_usb_pcdc_usr_ctrl_trans_function;
  R_usb_pstd_DriverRegistration(ptr, &driver);
        :
}
```

6.2.5 R_usb_pcdc_Task

The PCDC task

Format

void R_usb_pcdc_Task(USB_VP_INT_t stacd)

Argument

stacd Task start code (Not used)

Return Value

—

Description

This is the PCDC task which processes requests by the application and notifies the application of the results.

Note

1. Call this function from the user application during initialization.
2. In non-OS operations, the function is registered to be scheduled by the scheduler.

Example

```
void usb_apl_task_switch(void)
{
  while( 1 )
  {
    /* Scheduler */
    R_usb_cstd_Scheduler();

    if( USB_FLGSET == R_usb_cstd_CheckSchedule() )
    {
       R_usb_pstd_PcdTask((USB_VP_INT)0);        /* PCD Task */
       /* Peripheral Communications Devices Class Task */
       R_usb_pcdc_Task(0);
       /* Peripheral Communications Class Application Task */
       usb_pcdc_main_task(0);
    }
  }
}
```

6.2.6 R_usb_pcdc_driver_start

Start PCDC Driver

Format

void R_usb_pcdc_driver_start(USB_UTR_t *ptr)

Argument

*ptr Pointer to USB Transfer Structure

Return Value

— —

Description

This function sets the priority of the PCDC driver task.
Sending and receiving of task messages is enabled.

Note

1. Call this function from the user application during initialization.
2. Set the following members of USB_UTR_t structure before calling this function..

USB_REGADR_t	ipp	: USB register base address
uint16_t	ip	: USB IP Number

Example

```
void usb_apl( void )
{
  USB_UTR_t  *ptr;
    :
  ptr->ip = USB_PERI_USBIP_NUM;   /* USB IP No */
  ptr->ipp = R_usb_cstd_GetUsbIpAdr( ptr->ip );  /* USB IP base address */
    :
  R_usb_pcdc_driver_start(ptr); /*Peripheral Class Driver Task Start Setting*/
    :
}
```

7. Sample Application

7.1 Application Specifications

The main functions of the PCDC sample application (hereafter APL) are as follows.

1. Loopback mode (Echo mode)
   Transmits data received from the USB host back to the USB host.

7.2 Application Processing

The APL comprises two parts: initial setting and main loop. The following gives the processing summary for each part.

7.2.1 Initial Setting

In the initial setting part, the initial setting of the USB controller and the initialization of the application program are performed.

7.2.2 Main Loop

In loop-back mode, loop-back processing in which data sent by the USB host is received and then transmitted unmodified back to the USB host takes place as part of the main routine. An overview of the processing of the main loop is presented below.

1. When the R_USB_GetEvent function is called after enumeration with the USB host finishes, USB_STS_CONFIGURED is set as the return value. When the APL confirms USB_STS_CONFIGURED, it calls the R_USB_Read function to make a data receive request for data sent by the USB host.
2. When the R_USB_GetEvent function is called after reception of data from the USB host has finished, USB_STS_READ_COMPLETE is set as the return value. When the APL confirms USB_STS_READ_COMPLETE, it calls the R_USB_Write function to make a data transmit request to transmit the received data to the USB host.
3. When the R_USB_GetEvent function is called after transmission of data to the USB host finishes, USB_STS_WRITE_COMPLETE is set as the return value. When the APL confirms USB_STS_CONFIGURED, it calls the R_USB_Read function to make a data receive request for data sent by the USB host.
4. The processing in steps 2 and 3, above, is repeated.

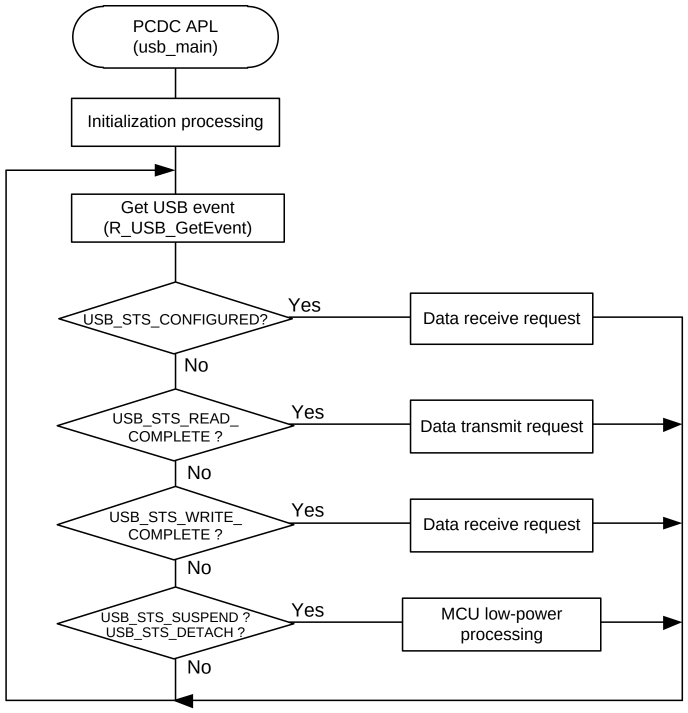

Figure 7-1 Main loop

8. Setup

8.1 Hardware

Figure 8-1 shows an example operating environment for the PCDC. Refer to the associated instruction manuals for details on setting up the evaluation board and using the emulator, etc.

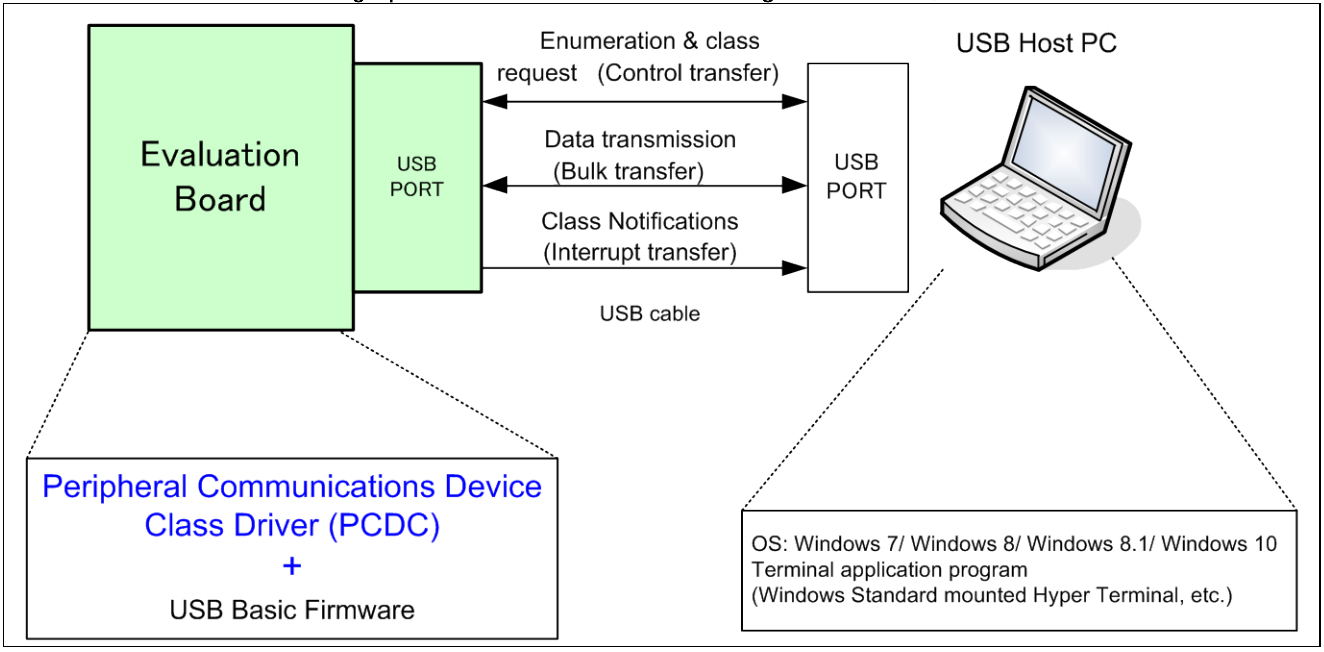

Figure 8-1 Example Operating Environment

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	Sep 01, 2016	—	First edition issued
1.10	Sep 30, 2016	—	Since the USB-BASIC-F / W has been revised, the version up
		21	“8. Setup” add

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 document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance 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 a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. 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.
2. 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.
3. 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.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "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 etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.
6. 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.
7. 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 systems manufactured by you.
8. 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.
9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
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.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

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

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, 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 India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

© 2016 Renesas Electronics Corporation. All rights reserved.

Colophon 5.0