

Renesas MCU USB Solutions

R01AN1670EJ0100

Rev. 1.00

LibUSB - A Complete RX USB Function and PC Host Solution

Sep 17, 2013

Introduction

You don't necessarily have to follow a class specification to create a USB solution, though Renesas does support many USB classes. That is, in many fields it is not necessary to follow a certain USB class specification,

Certain professional domains mandate a certain class, and for a common device it can be more convenient for a user not to have to install a new driver on their PC. For example, a mouse, a keyboard are just expected to work when attached to a PC. Often though, a user must point to an INF-file, even if the driver exists on a PC, and so adding a new driver via the same media as that with which the INF-file is provided is not an additional obstacle.

The point is, that if a certain class must not be adhered to, where an application is unique, simple, etc, there is no reason to follow a complicated class specification. In this case one is free to use a so called "Vendor" class, and instead use a common "non-class" interface, namely "LibUSB". This can make development significantly faster. LibUSB will be shown in this solution as the easy choice for deploying a product.

LibUSB is a PC driver library that enables user space application programs to communicate with USB devices. For example for Windows, this is described in the "libusb-win32" download available at <https://sourceforge.net/apps/trac/libusb-win32>

Even if a class must be used, LibUSB can still have an important role to fill: It can be used to issue host-to-device Standard Control commands to test a USB function target's behavior. For example one can use it to iterate through enumeration step by step when developing target code.

Summary: LibUSB is a great tool for getting prototype targets up and running fast, to exchange data with a USB host, without having to follow the rigid specs of a class when this is not necessary. This document explains how to use the Peripheral LibUSB Demo on the RSK-RX63N, together with the included PC host application *Renesas_libusb_host*.

Target Device

The RX111 Group microcontrollers. These MCUs have a built in USB Full Speed or Low Speed controller.

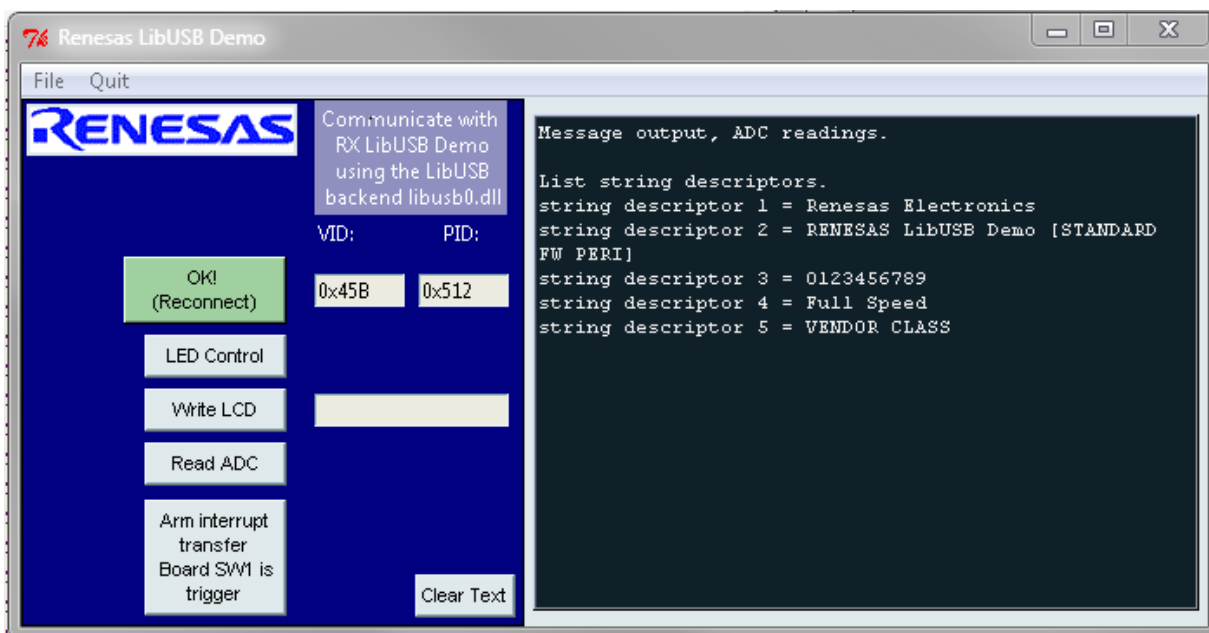


Figure 1. In this document, LibUSB is explained, and a sample host –target application is described.

Content

1. Overview	3
1.1 Functions and Features.....	3
1.2 Related Documents.....	3
1.3 Terms and Acronyms	3
2. Physical Setup.....	4
3. LibUSB Demo Description	4
3.1 Enumeration	4
3.2 Host - Function Demo	5
4. Software Setup.....	5
4.1 USB Function (RX board)	5
4.2 USB Host (PC)	5
4.2.1 Install the LibUSB PC driver.....	5
4.2.2 Run the host demo program.....	5
4.2.3 Windows 'Found New Hardware Wizard'	6
4.2.4 Windows Device Manager.....	8
4.3 Following the Demo in the Debugger.....	8
5. The Application Demo.....	8
5.1 Command Processing.....	8
5.2 Command Protocol Detail	10
6. Create your own Host Application.....	12
6.1 Python	12
6.2 .NET	12
7. Create Your Own Product INF-file	12
8. The RX USB Source Code.....	13
8.1 Software Configuration.....	13
8.2 Block Diagram	13
8.3 Application Level Processing & Code Changes vs. USB Basic FW	14
8.4 Pipe (Endpoint) Information Table	14
8.5 Descriptors	14
8.5.1 Device.....	15
8.5.2 Configuration	15
8.5.3 Interface.....	15
8.5.4 Endpoint	16
9. Using E2studio	17
9.1 Import the Project	17
9.1.1 New Workspace	17
9.1.2 Existing Workspace.....	17
9.2 Using the Renesas Debug Console	17
9.2.1 STUDIO Low Level Source Code	18
Website and Support.....	19

Revision Record	1
General Precautions in the Handling of MPU/MCU Products	2

1. Overview

This document shows how to use a standard PC LibUSB driver, together with an RX peripheral running a modified version of Renesas RX Basic FW. The modification consists of changing the EP numbers and types, and adding application level code to build a complete demonstration, the “Renesas LibUSB Demo”.

1.1 Functions and Features

The RX peripheral LibUSB Demo source code will communicate with a LibUSB host PC application “Renesas_libusb_host”.

1.2 Related Documents

1. Renesas USB Basic Firmware Application Note (*r01an0326ej0210_usb*).
2. RX111 Group Hardware Manual (*r01uh0365ej*).
3. RX111 Group Renesas Starter Kit (RSKRX111) User Manual.
4. Universal Serial Bus Revision 2.0 specification.
<http://www.usb.org/developers/docs>
5. Renesas Electronics USB.
http://www.renesas.com/applications/key_technology/connectivity/usb/index.jsp
6. Renesas USB Drivers.
http://www.renesas.com/products/tools/middleware_and_drivers/c_driver/usb_driver/index.jsp

1.3 Terms and Acronyms

Terms and acronyms used in this document:

EP	Endpoint
GUI	Guided User Interface.
LibUSB	An API for host (PC) applications. Includes PC driver to enable PC-application <=> USB function communication.
LibUSB backend	For Windows, the USB driver libusb0.dll, downloaded together with libusb-win32. Linux: LibUSB is built in, and is the default handler.
RSK	Renesas Starter Kit.
SW1	User switch one on the RX111 RSK, used for interrupt pipe demo.
USB	Universal Serial Bus
USB Basic FW	USB basic firmware for Renesas USB device (non-OS& RTOS)

2. Physical Setup

The peripheral LibUSB Demo source code allows the USB PC host application *Renesas_libusb_host* to manipulate the connected RSK board via USB.

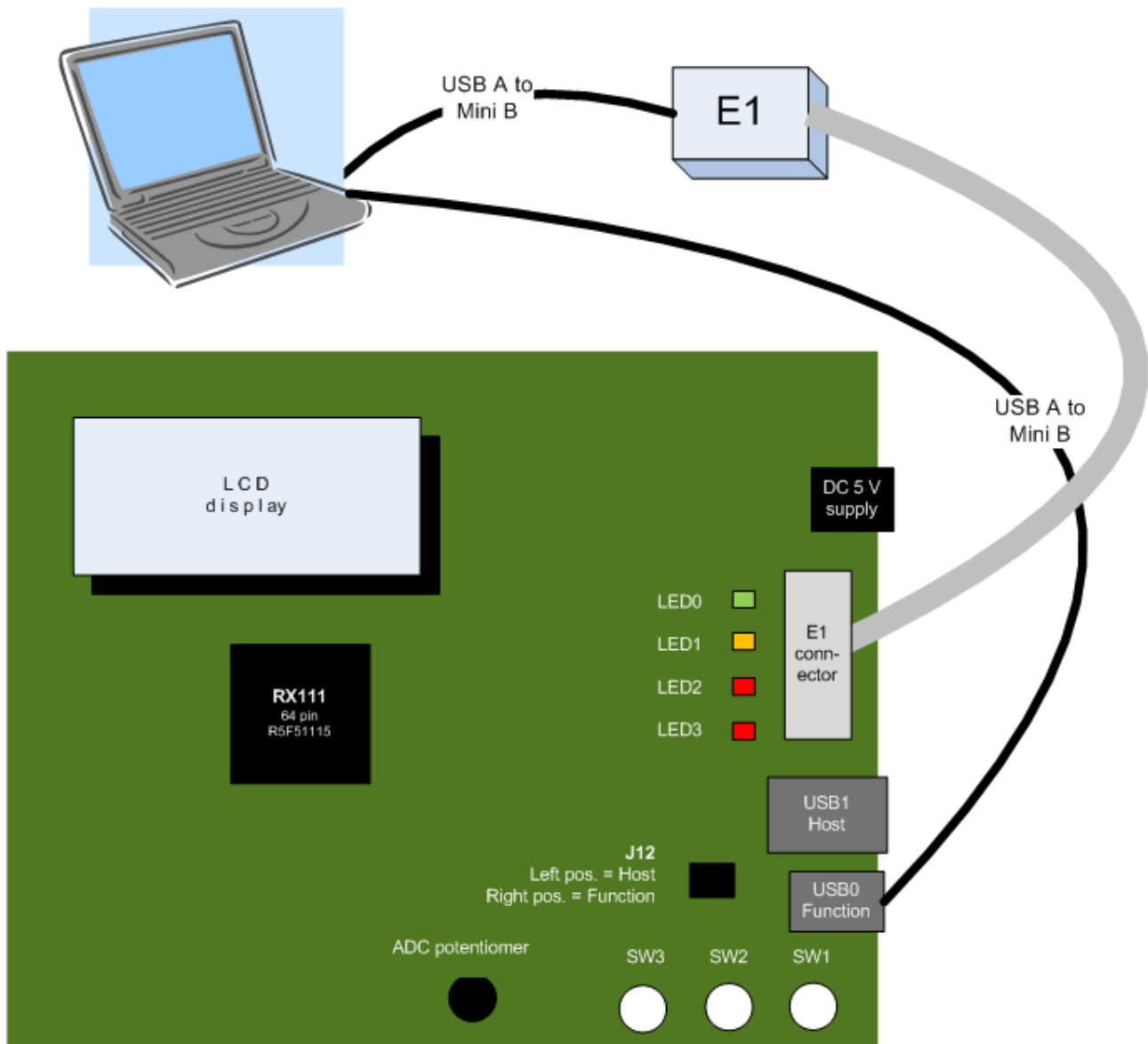


Figure 2. USB Host and target setup.

3. LibUSB Demo Description

3.1 Enumeration

Enumeration occurs when the target RX board is connected to the USB host (a PC normally). For the PC LibUSB driver to be associated with the RX board, the following must be true:

- The host LibUSB driver is installed on the PC. More on this below.
- The host application and the INF file used must have matching VID and PID numbers. By default in the firmware, VID is 0x45B, and PID is 0x512. *This must be changed for the peripheral by the customer when developing a product!*

3.2 Host - Function Demo

The PC USB host application *Renesas_libusb_host* can via the PC LibUSB backend now access the RX target application.

- LED demo: The RX board LED light pattern is changed when the LED command is sent from the host.
- LCD demo: The RX board LCD display is changed using user input from host when the LCD command is sent from the host.
- AD demo: The RX board AD value is sent to the host PC when the read ADC command is sent from the host. The value of the variable resistor connected to the AD as shown in the above figure is read and sent to the host PC.
- In interrupt demo is also included. When SW1 is pressed by the user, data is sent from board to host using an interrupt pipe.

4. Software Setup

This section will show how to get the LibUSB driver and host LibUSB Demo running from the host's perspective (the PC). Also, you will learn how to create your proprietary INF-file.

4.1 USB Function (RX board)

1. Import the downloaded e² studio project archive file or workspace folder structure into your e2studio workspace by carefully following the directions in chapter 9.1.
2. Create a debug session in e2studio as follows.
 - Highlight the LibUSB project in the Explorer pane in e2studio.
 - Select Run=>Debug Configurations.
 - Double-click on Renesas GDB HW Debugging.
 - Fill out the options for debugging. For example, if you are using the E1, click the Debugger tab, select 'E1', and fill out the sub-tabs. Make sure all settings correspond to your setup. Debug work RAM can be set at 0xc0 if you haven't changed the default demo. When you have looked over everything carefully, click Debug.
 - After connecting, download the object code and run it.
3. Connect an A-to-Mini-B USB cable between PC and RX target board.

4.2 USB Host (PC)

4.2.1 Install the LibUSB PC driver

4. Windows should show the "Found New HW Wizard". If not, in Windows Device Mgr, select 'Update Driver'.
5. Point to the *../Host_PC* folder which contains INF-files for the default demo.
You must later create your own INF file, for your product with your VID and PID. See 'Create Your Proprietary Product INF-file' below.
6. If *libusb_x.sys* is not already on your PC, point to the folder *../LibUSB_Demo/Host_PC/libusb-win32-bin-1.2.x.0/bin*
7. The Found New HW Wizard should copy the driver to e.g. *../Windows/system32*. If you are having any problems installing the driver, see section 4.2.3 "Windows 'Found New Hardware Wizard'".

4.2.2 Run the host demo program

1. Using the PC executable – Alt I

8. Run *../HostPC/host_exe/Renesas_libusb_host/dist/Renesas_libusb_host.exe*.
9. The host will manipulate in order the LEDs, the LCD, then read the ADC, in a loop.
10. If you prefer using a GUI, run
../LibUSB_HostPC/host_exe/Renesas_libusb_host_gui/dist/Renesas_libusb_host_gui.exe

- Connect with the target board using the “Connect” button with VID and PID set as in file `r_usb_vendor_descriptor.c`. Then, manipulate the LED, the display, and read the board ADC using the control buttons in the GUI.

2. Using Python – Alt II

To really understand what is happening, you should install Python and PyUSB. Read `START_HERE.txt` how to set up your PC.

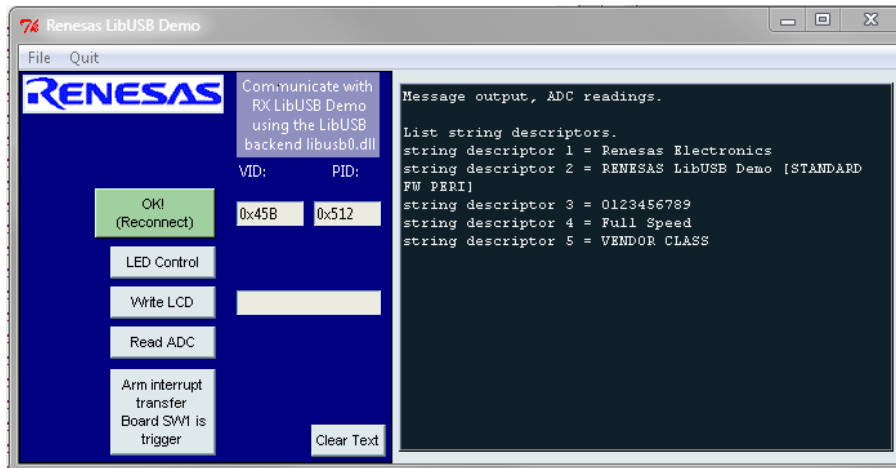


Figure 3. The commands to control the LED, the display, etc from the host PC user application, are implemented via the PC host “LibUSB backend”, that is, the driver `libusb.dll`.

4.2.3 Windows ‘Found New Hardware Wizard’

Do the following in detail to learn how this works, in order to understand how to implement it for your customers doing this for your new product. Or, if you are having any difficulty with the LibUSB driver.

- Start Windows Device Manager (type “device manager” into the Windows Run menu) to keep track of enumeration status. Keep it open.
- Restart the target by disconnecting and reconnecting the cable. We will follow the ‘Found New HW Wizard’. (The correct driver (`libusb0.sys`) will be found automatically by Windows if the vendor and product ids were used by Windows at some time in the past.)
- If the ‘Found New HW Wizard’ does not appear, right click on the Lib USB Demo device in Windows Device Manager and click on ‘Update Driver.’ Then follow these same steps.
- In the Windows ‘Found New HW wizard’, select “No, not this time” (Win7; Browse my computer...), then ‘Install from a specific location’ (Win7; Let me pick...).
- If you saved your INF-file to Window’s INF-directory and are given a choice of several INF-files, identify yours by matching manufacture name.
- To be completely specific, select ‘Don’t search, I will choose driver...’ then ‘Have disk’, to make sure we pick the right INF-file.
- The LibUSB driver for windows; `libusb0.sys` was probably already installed to folder `c:/Windows/system32` if you ran the INF Wizard, but if asked, select ‘Browse’ and point to the directory `.Host_PC/libusb-win32-bin-1.2.5.0/bin/x86`.
- Press OK and Finish.

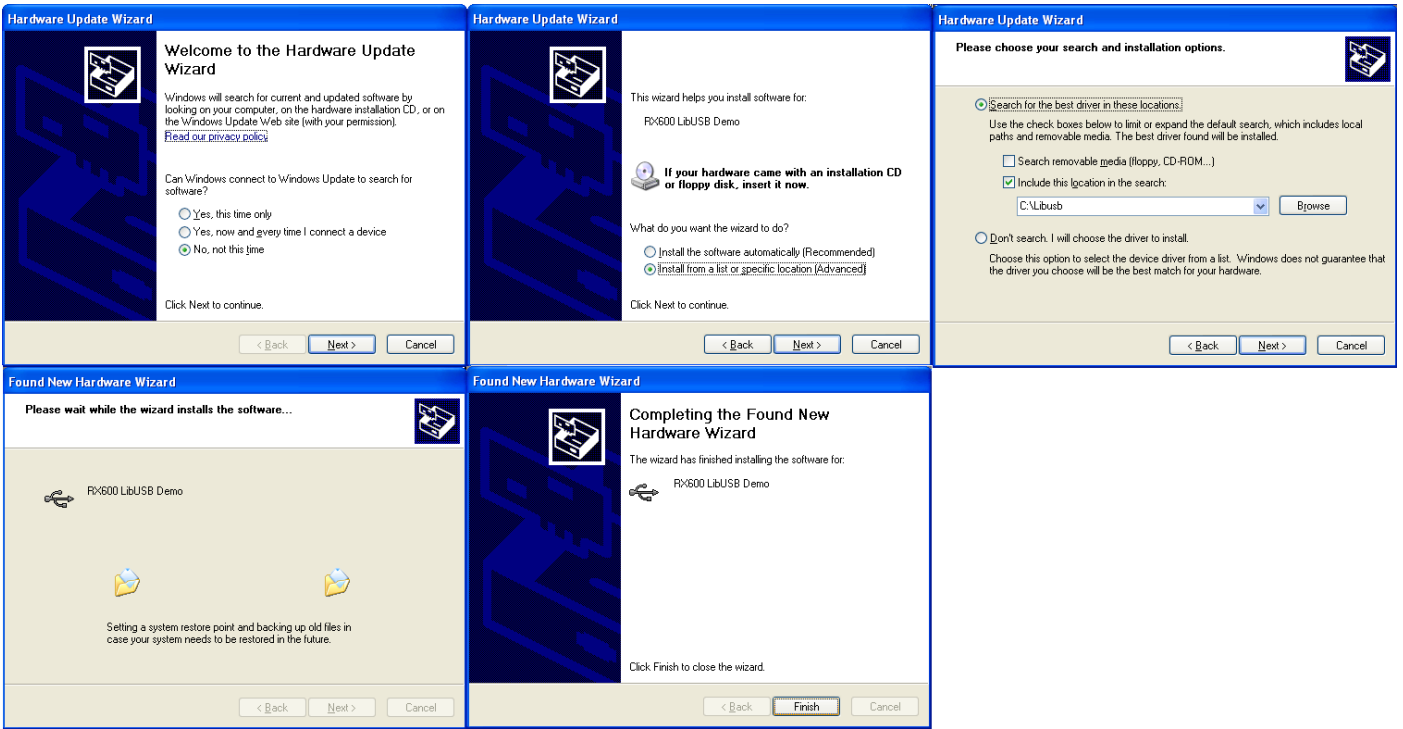
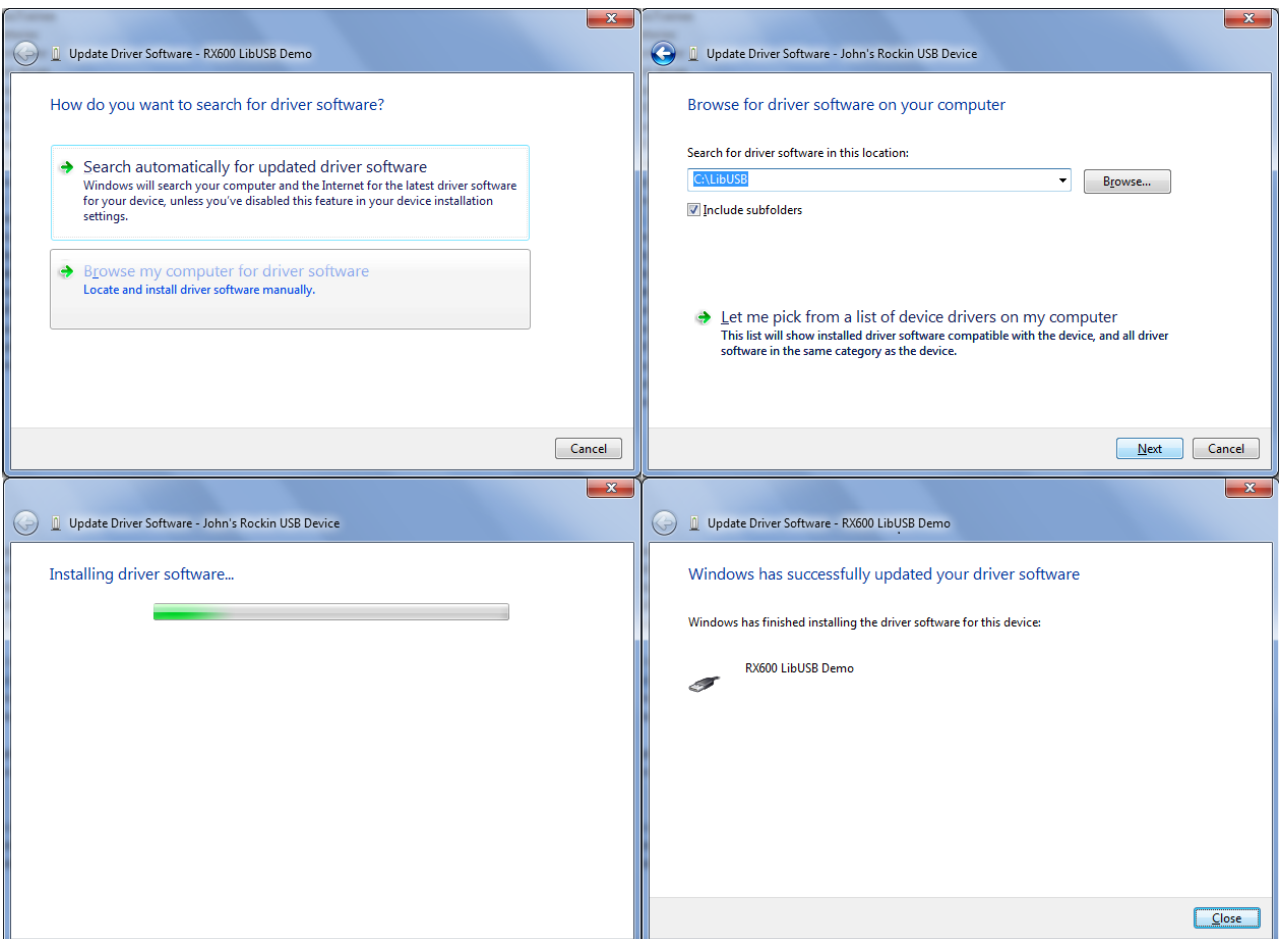


Figure 4. If something goes wrong, restart the target firmware, uninstall the driver by right-clicking on the device in Device Mgr, then redo the procedure.

Example above is for WinXP.

Below is for Windows7.



4.2.4 Windows Device Manager

9. Check that Windows has enumerated the RX properly. In Windows Device Manager you should see the RX LibUSB peripheral. See step 1 and Figure 5.

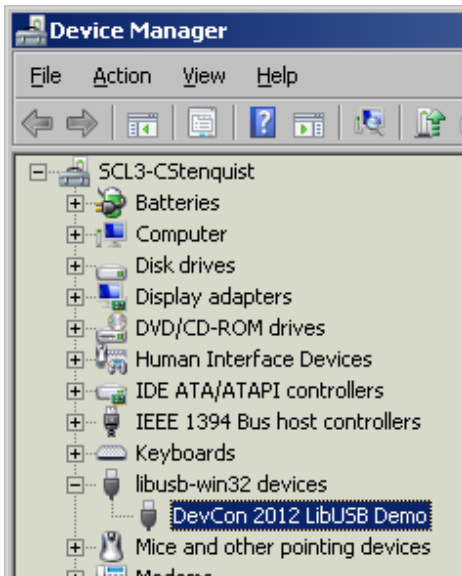


Figure 5. The RX LibUSB peripheral should show up in Device Manager, listed under the “libusb-win32 device” class.

4.3 Following the Demo in the Debugger

Set a breakpoint in file `r_usb_vendor_papl.c` at

```
usb_plibusb_LibUsbCmdChk();
```

When a command to a demo endpoint is received, the break will trigger and you can follow the demo on the target.

5. The Application Demo

Aside from setting VID and PID and the descriptors in `../src/SmplMain/APL/r_usb_vendor_descriptor.c`, The application layer programmer should mainly only have to deal with another file in the same directory; `r_usb_vendor_papl.c`. to create the desired application.

To understand fundamentals for using the underlying USB Basic FW, see document `r01an0326ej_usb_basic`.

5.1 Command Processing

The LibUSB application top level user control was described in chapter 3. Here follows what happens inside the above mentioned file `r_usb_vendor_papl`.

When the application receives PIPE1 (data whose first byte is a command (1, 2, or 4 in the demo), the demo application layer takes action depending on the command. As a user application programmer, you can see the action take place starting with execution of function `usb_plibusb_LibUsbCmdChk`.

The above procedure is illustrated in the next flowchart.

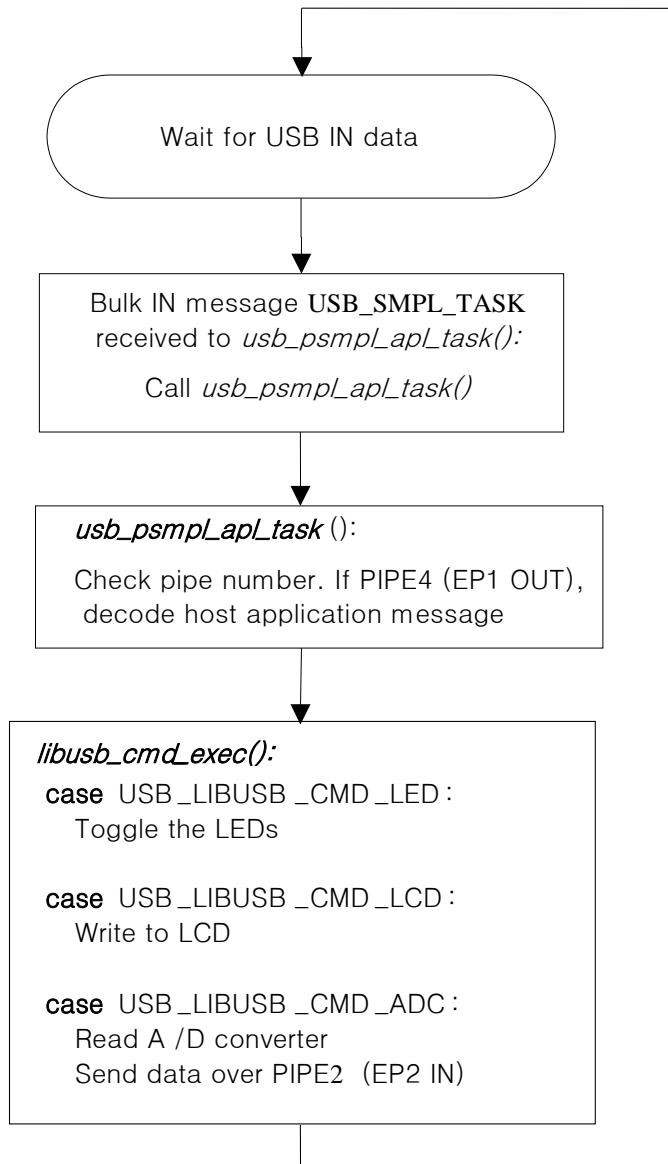


Figure 6. LibUSB application level processing flow in file *r_usb_vendor_papl.c*. For more explanation on how application level processing works, see 8.3.

5.2 Command Protocol Detail

The LibUSB Demo has three application level commands; "Toggle LED," "Read ADC," and "Set LCD." The first byte in the host PIPE1 transfer contains the command number. The second and subsequent bytes are not used. See the table below for details. For more info, see file `./src/SmplMain/APL/r_usb_vendor_papl.c`, function `usb_plibusb_LibUsbCmdChk`.

To easily see this protocol on the target, enable the Debug Console by following the steps in chapter 9.2 "Using the Renesas Debug Console".


Toggle LED

Host	EP and direction As seen from host	Function (RX)
First byte of host OUT transfer (command byte value) = 1	EP1 Bulk OUT ➔	Received in PIPE1. <i>Command processing. Changes pattern of LEDs.</i> Only the first byte is recognized as a command. The 2nd and subsequent bytes are not used.


Read ADC

Host	Direction	Function (RX)
First byte of host OUT transfer (command byte value) = 2	EP1 Bulk OUT ➔	Received in PIPE1. <i>Read AD Converter display on board LCD, but also send over USB back to host.</i> Only the first byte is recognized as a command. The 2nd and subsequent bytes are not used. Returns AD data (2-4 bytes).
Read ADC value.	EP2 Bulk IN ➔	The first byte has the following meaning: 0: OK. 1: NOT OK. The 3 bytes starting at the 2nd byte contain the AD value in little endian.

Set LCD

Host	Direction	Function (RX)
First byte of host OUT transfer. (command byte value) = 4	EP1 Bulk OUT 	Received in PIPE1. Command byte. Set LCD display.
Second byte		Line number of LCD. This can zero (0) or one (1) or higher, but a line number higher than what is available clears the display.
Bytes 3 and up		Text to display.

Read Switch

Host	Direction	Function (RX)
Receives test bytes; an increasing enumerated series.	EP3 Interrupt IN 	Sent in PIPE6.
		Sent when host asks for data in this endpoint AND user presses switch

6. Create your own Host Application

6.1 Python

See the included files *Renesas_pyusb.txt* and *Renesas_libusb_host.py* for instructions how to run Python and PyUSB. Once these are installed, the Python code for PC is easy to change. Run the code by right-clicking on *Renesas_Host_LibUSB.py* and selecting "Edit with Idle".

3. License

No license is needed to use Python.

4. Pyinstaller

To create a Windows executable, Pyinstaller was used. Pyinstaller can create one single executable file instead of as in the demo, where needed DLLs are included separately and referenced by the executable. See *Renesas_pyinstaller_info.txt*.

6.2 .NET

See the .NET library for your VB, C# etc application builder for information on how to interface with LibUSB.

7. Create Your Own Product INF-file

1. Change the VID and PID number in *../Workspace/SmplMain/APL/r_usb_vendor_descriptor.c*. Compile, connect, download, and run.
2. Run *../libusb-win32-bin-1.2.4.0/bin/inf-wizard.exe* and follow the GUI. The RX with your new VID and PID should show up. Enter desired Manufacturer and Device.
3. Enumeration: When Windows shows the "Found New HW Wizard", point to the above created INF file.
4. If *libusb0.sys* is needed, point to the folder *../LibUSB_Demo/Host_PC/libusb-win32-bin-1.2.6.0/bin*
5. The Found New HW Wizard should copy the driver to e.g. Windows/system32.

8. The RX USB Source Code

8.1 Software Configuration

The LibUSB Demo is implemented using the USB Basic Firmware. The Basic FW manual as noted in Related Documents needs to be studied e.g. in order to copy the right files to the *HwResources* folder for your RX MCU device. Here we will focus on the LibUSB specific changes made to the Basic FW package to implement the LibUSB Demo.

8.2 Block Diagram

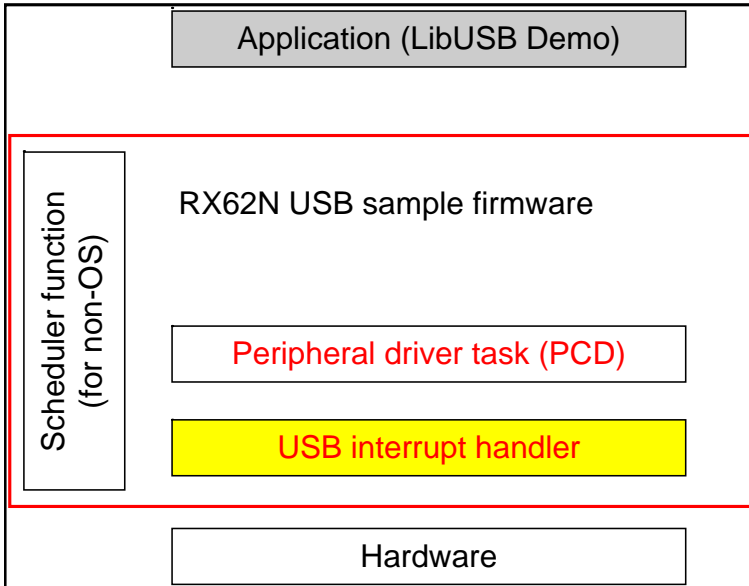


Figure 7. Software block diagram

8.3 Application Level Processing & Code Changes vs. USB Basic FW

The table below lists the most important functions that were modified for the LibUSB Demo. Search for “LibUSB” in the project to see other changes. Other functions, e.g. to read ADC and were also added.

File	Function / data	Outline
<i>../SmpMain/APL/r_usb_vendor_papl.c</i>	<code>usb_psmpl_transfer_result()</code>	Callback registered earlier with <code>usb_psmpl_transfer_start()</code> , one for each desired pipe. This callback is triggered when USB communication has taken place for a pipe. (Pipes must be set up in the Pipe Information Table; see below.) After triggering, <code>usb_psmpl_transfer_start()</code> must be called again to take care of the next data transfer.
"-	<code>usb_psmpl_application()</code>	The above callback sends message <code>USB_SMPL_TASK</code> to this function. It processes the data in the received pipe.
"-	<code>libusb_cmd_exec()</code>	Demonstrates usage of bulk OUT endpoint: Host writes to EP to control LED Host writes to EP to control LCD Demonstrates usage of bulk IN endpoint: Host reads EP to get ADC value
"-	<code>usb_vendor_int_in_handler ()</code>	Demonstrates usage of interrupt IN endpoint: Host reads switch press. (Not used in primary demo.)
<i>../SmpMain/APL/r_usb_vendor_descriptor.c</i>	<code>usb_gpvendor_smpl_eptbl1 []</code>	Pipe Information Table, and EP descriptors were changed. See 8.4.

8.4 Pipe (Endpoint) Information Table

The Pipe Information Table `usb_gpvendor_smpl_eptbl1`, in file `../Workspace/SmpMain/APL/r_usb_vendor_descriptor.c` is central in to how to set up your application. The P. I. T. is where endpoints and pipes are selected for an RX USB function. Pipes are paired with endpoints for the peripheral. (The host side selects its own pipes for these endpoints.). More information on the P.I.T is found in 1.2.1.

8.5 Descriptors

This section describes the default descriptors of the LibUSB Demo source code, which are in file `r_usb_vendor_descriptor.c`. The most important field to be changed is the VID (`idVendor` below), but of course any field is to be changed subject to the end product purpose.

8.5.1 Device

Field	Value	Bytes	Description
bLength	0x12	1	Descriptor size is 18 bytes.
bDescriptorType	0x01	2	DEVICE descriptor type.
bcdUSB	0x0200	2	USB specification version 2.00.
bDeviceClass	0xFF	1	Device Class is vendor-specific.
bDeviceSubClass	0xFF	1	Device Subclass is vendor-specific.
bDeviceProtocol	0xFF	1	Device Protocol is vendor-specific.
bMaxPacketSize0	0x40	1	Maximum packet size for endpoint zero is 64.
idVendor	0x45B	2	Vendor ID. <i>Customer must change this.</i> Obtain VID from http://www.usb.org
idProduct	0x512	2	Product ID. <i>Customer to manage PIDs.</i>
bcdDevice	0x0100	2	Device release number.
iManufacturer	0x01	1	Manufacturer index in string descriptor is 1.
iProduct	0x02	1	Product index in string descriptor is 2.
iSerialNumber	0x03	1	Serial number index in string descriptor is 3.
bNumConfigurations	0x01	1	Device has one possible configuration.

8.5.2 Configuration

Field	Value	Bytes	Description
bLength	0x09	1	Configuration section of descriptor is 9 bytes.
bDescriptorType	0x02	1	CONFIGURATION descriptor type.
wTotalLength	0x0046	2	Total length of data for this configuration.
bNumInterfaces	0x01	1	This configuration supports one interface only; Interface 0.
bConfigurationValue	0x01	1	Configuration 1 - the only configuration.
iConfiguration	0x03	1	String descriptor index describing this configuration (All string descriptors, starting with the first, belong to this configuration).
bmAttributes	0xE0	1	Configuration characteristics.
bMaxPower	0x32	1	Maximum power 100 mA.

8.5.3 Interface

Field	Value	Bytes	Description
bLength	0x09	1	Interface section of descriptor is 9 bytes.
bDescriptorType	0x04	1	INTERFACE descriptor type.
bInterfaceNumber	0x00	1	The number of this interface is 0.
bAlternateSetting	0x00	1	No alternate setting.
bNumEndpoints	0x04	1	Number of endpoints used by this interface.
bInterfaceClass	0xFF	1	Vendor-specific.
bInterfaceSubClass	0xFF	1	Vendor-specific.
bInterfaceProtocol	0xFF	1	Vendor-specific.
iInterface	0x05	1	String descriptor index describing the interface.

8.5.4 Endpoint

EP1

Field	Value	Bytes	Description
bLength	0x07	1	EP section of descriptor is 7 bytes.
bDescriptorType	0x05	1	ENDPOINT descriptor type.
bEndpointAddress	0x01	1	OUT endpoint number 1. Used for USB host LibUSB demo commands.
bmAttributes	0x02	1	Type – BULK (only).
wMaxPacketSize	0x0040	2	Maximum packet size for this endpoint is 64 bytes.
bInterval	0x01	1	How often the endpoint can send or receive data. An EP is serviced by the host at this rate. The device's endpoint NAKs if it has no data/is not ready. Although a poll/NAK is relatively lightweight, it can still waste bandwidth if the polling interval is much greater than is required by the application.) A value of 0 means the endpoint never NAKs.

EP2

Field	Value	Bytes	Description
bLength	0x07	1	EP section of descriptor is 7 bytes
bDescriptorType	0x05	1	ENDPOINT descriptor type
bEndpointAddress	0x82	1	IN endpoint number 2. Used for ADC data sent to host.
bmAttributes	0x02	1	Type – BULK (only).
wMaxPacketSize	0x0040	2	Maximum packet size for this endpoint is 64 bytes.
bInterval	0x01	1	See also explanation for EP1.

EP3

Field	Value	Bytes	Description
bLength	0x07	1	EP section of descriptor is 7 bytes
bDescriptorType	0x05	1	ENDPOINT descriptor type
bEndpointAddress	0x83	1	IN endpoint number 3. Used for switch interrupt EP demo.
bmAttributes	0x03	1	Type – INTERRUPT (only).
wMaxPacketSize	0x0010	2	Maximum packet size for this endpoint is 16 bytes.
bInterval	0x04	1	An interrupt IN EP is polled at the bInterval rate. (Bandwidth is reserved by host for EP). Set e.g. to 64 for a 64 ms latency, and 8 for a 8 ms latency. The lower the number the more often the endpoint is serviced. If no data is available from the Function (NAK), the available bandwidth will be available for control/bulk data. See also explanation for EP1.

EP4

Field	Value	Bytes	Description
bLength	0x07	1	EP section of descriptor is 7 bytes
bDescriptorType	0x05	1	ENDPOINT descriptor type
bEndpointAddress	0x84	1	OUT endpoint number 4. Not used in default demo.
bmAttributes	0x03	1	Type – INTERRUPT (only).
wMaxPacketSize	0x0010	2	Maximum packet size for this endpoint is 16 bytes.
Interval	0x04	1	How often the host will service the interrupt OUT EP and transmit data. See also explanation for EP3.

9. Using E2studio

9.1 Import the Project

With e² studio, a project should not be moved, but first exported then imported. Follow the directions below.

9.1.1 New Workspace

1. Create empty folder where you want workspace.
2. Start E2S, and point to that folder as E2S asks what workspace to open.
3. Click Workbench icon (bottom right in blue intro-screen).
4. Continue with next step below.

9.1.2 Existing Workspace

5. Select Import.
6. Select General => Existing Projects into workspace. ("Create new projects from an archive file or directory.")
7. Browse to
 - Archive zip-file, *or*
 - Root directory

For both, make sure checkbox "Copy project to workspace" is checked.

You have now imported this project into the workspace. You can go ahead and import other projects into the same workspace... Follow below for importing to existing workspace.

Build with *Cntrl+B*.

9.2 Using the Renesas Debug Console

The Renesas Debug Console means you have the ability to use *printf()* statements in C to send trace strings to the standard output. Standard output will in this case be the E1/E20 debug register. To use this feature, the following must be true.

1. *INIT_IOLIB()* must be called. This is sometimes commented out in *resetprog.c* to save object code space.
2. The code in 9.2.1 constitutes the *putchar* and *getchar* functions to reside in *lowlvl.src* so that the E1/E20 debug ports are used for I/O processing. Replace the existing code with this if is not already in *lowlvl.src*.
3. Include *<stdio.h>* in any files where you wish to use *printf*-statements.
4. In e² studio, add the Debug Console window by switching on both icons
 - “**I/O**” and
 - “**Pin Console**” as shown below.

Both must be on so that the print buffer in E1/E20 can be emptied, and not block.



See example usage of `printf()` in the function `libusb_cmd_exec()` in `r_usb_vendor_papl.c`.

9.2.1 STDIO Low Level Source Code

Use the following code in `lowlvl.src` to get printf statements to the E1/E20 Debug Console.

```

;-----
;
; FILE :lowlvl.src
; DATE :Wed, Jul 01, 2009
; DESCRIPTION :Program of Low level
; CPU TYPE :RX
;
;-----
                .GLB  _charput
                .GLB  _charget

FC2E0          .EQU  00084080h
FE2C0          .EQU  00084090h
DBGSTAT        .EQU  000840C0h
RXFL0EN        .EQU  00001000h
TXFL0EN        .EQU  00000100h

                .SECTION P, CODE

;-----
; _charput:
;-----
__charput:
                .STACK  _charput = 00000000h
__C2E0START:   MOV.L  #TXFL0EN, R3
                MOV.L  #DBGSTAT, R4
__TXLOOP:      MOV.L  [R4], R5
                AND   R3, R5
                BNZ   __TXLOOP
__WRITEFC2E0:  MOV.L  #FC2E0, R2
                MOV.L  R1, [R2]
__CHARPUTEXIT: RTS

;-----
; _charget:
;-----
__charget:
                .STACK  _charget = 00000000h
__E2CSTART:   MOV.L  #RXFL0EN, R3
                MOV.L  #DBGSTAT, R4
__RXLOOP:      MOV.L  [R4], R5
                AND   R3, R5
                BZ    __RXLOOP
__READFE2C0:  MOV.L  #FE2C0, R2
                MOV.L  [R2], R1
__CHARGETEXIT: RTS

;-----
; End of conditional code (section)
                .END

```

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 17, 2013	—	First edition for RX111 issued. Derived from r01an0492 – LibUSB for RX600.

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 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 a product with a different type 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.



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: +44-1628-651-700, Fax: +44-1628-651-804

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.
13F, No. 363, Fu Shing North Road, Taipei, 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 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

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