User's Manual

RENESAS

Renesas Starter Kit for SH7216

USB Sample Code User's Manual

RENESAS SINGLE-CHIP MICROCOMPUTER
SH FAMILY

Renesas Electronics
www.renesas.com

Rev.1.00 2009.12

Table of Contents

Chapter 1. Preface

Cautions

Trademarks

Copyright

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

Glossary

ADC	Analog to Digital Converter
CDC	Communication Device Class
HAL	Hardware Abstraction Layer
HID	Human Interface Device
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MSC	Mass Storage Class
RSK	Renesas Starter Kit
USB	Universal Serial Bus

Chapter 2.Introduction

The RSK USB sample code provides a basis for a developer to add USB device functionality to a system. It includes sample applications for the three most common USB Device classes*:-

- Human Interface Device (HID)
- Communication Device Class - Abstract Control Model (CDC-ACM)
- Mass Storage Class (MSC)

In addition to the three defined USB classes a LibUSB sample is included. LibUSB is an open source project with the aim of providing a library that a user application can utilise to access a USB device regardless of operating system. A sample using a Microsoft Windows host is provided.

The embedded software is available as source written in ANSI C and does not require an operating system.

The host applications software is also available as source written for MS Windows using VisualC++.

This manual describes the USB sample code. The Quick Start Guide and Tutorial Manual provide details of the software installation and debugging environment.

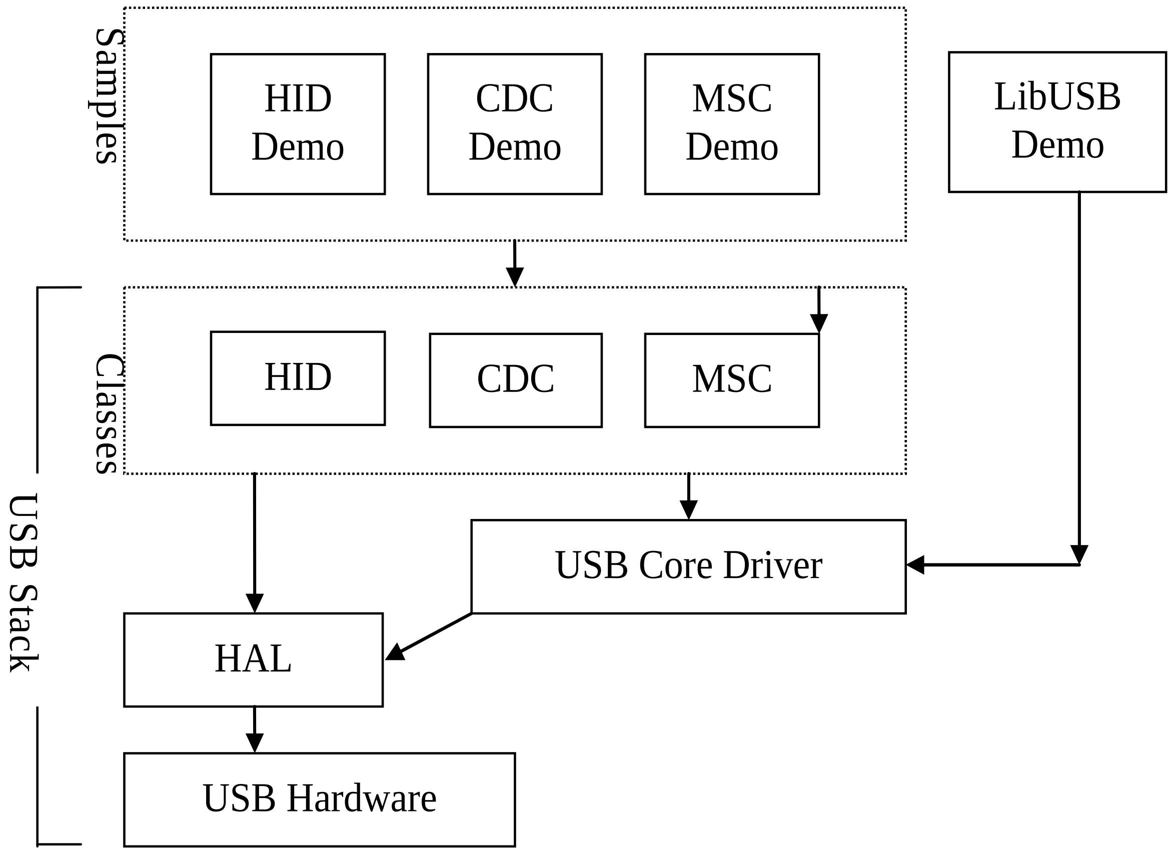


Figure 1 - Embedded SW, including Layers of USB Stack

* *See specific sections for details. Tested with a USB Host PC running MS Windows XP SP2. HID and LibUSB samples both include a PC Application.*

Chapter 3.Development Environment

3.1.Sample Code Configuration

The Sample code is provided as a project generator with the RSK. To create the sample code project follow the instructions in the RSK Quick Start Guide.

When created the sample code will contain the source for both the Target project and a Host project if applicable, including any configuration driver files.

3.2.Target Sample Code Options

When developing USB software it is useful to be able to get debug information out at runtime without stopping code from running such as when stepping in a debugger. All modules of the USB Stack software include debug messages that can be utilised in a system that supports printf. The sample applications all support printf and the output is viewable via the serial port of the RSK. To view the serial output the following settings are required:

Baud: 57600. Data: 8 Bit. Parity: None. Stop Bits: 1. Flow: None.

The level of debug message can be set using the #define DEBUG_LEVEL. This is described in the file usb_common.h in the USBStack directory. Note that a high level of debug messages can significantly slow down the system. To remove all debug messages from the build then #define RELEASE.

3.3.Host Application Software

To build the Microsoft Visual C++ Applications you will need to have the appropriate Windows Software Development Kit (SDK) and the Windows Driver Kit (WDK) installed.   These kits provide access to the library functions to access USB devices and are available directly from Microsoft.

We have provided the links to the current locations however we cannot guarantee the suitability or accuracy of these links. Both must be installed to be able to build the application, we suggest installing the SDK first.

Windows SDK
Windows Driver Kit (WDK)

Chapter 4.USB Stack (Target)

The USB software is implemented in the form of a USB stack comprising of three layers.

At the bottom of the stack is a hardware abstraction layer (HAL) that provides a hardware independent API for the upper layers.

In the middle is a core layer (USBCore) that handles standard device requests.

At the top of the stack are the USB Device Classes consisting of HID, CDC and MSC. This top layer includes an optional API layer called the Renesas USB Function API (R_USBF). This and the specific classes are all described in their own section below.

This modular design means this software can be still be used even if developing a proprietary USB interface. For example a proprietary module could be implemented by calling functions directly exposed by both the USBHAL API and USBCore API.

4.1.Hardware Abstraction Layer

The HAL is a hardware specific layer that provides a non-hardware specific API. The HAL supports the following transfer modes:

Control (Setup, Data IN/OUT, Status)

Bulk (IN and OUT)

Interrupt (IN)

Some HAL implementations may not be able to support all these modes but the SH7216 implementation does.

Here is a list of the functions that make up the USBHAL API.

Name	Description
USBHAL_Init	Initialise the HAL. Register callback functions. If using the USB Core layer then this is done automatically.
USBHAL_Config_Get	Get the current HAL configuration.
USBHAL_Config_Set	Set the current HAL configuration.
USBHAL_Control_ACK	Generate an ACK on the Control IN pipe. (Used following a setup packet)
USBHAL_Control_IN	Send data on the Control IN pipe. (Used following a setup packet)
USBHAL_Control_OUT	Receive data on the Control OUT pipe. (Used following a setup packet)
USBHAL_Bulk_IN	Send data on the Bulk IN pipe.
USBHAL_Bulk_OUT	Receive data on the Control OUT pipe.
USBHAL_Interrupt_IN	Send data on the Control IN pipe.
USBHAL_Reset	Reset this module. (Following an error).
USBHAL_Control_Stall	Stall the control pipe. (Used following a setup packet)
USBHAL_Bulk_IN_Stall	Stall the Bulk IN pipe.
USBHAL_Bulk_OUT_Stall	Stall the Bulk OUT pipe.
USBHAL_Interrupt_IN_Stall	Stall the Interrupt IN pipe.
USBHALInterruptHandler	The system must be setup so that this gets called when any USB Interrupt occurs.

The HAL module consists of the following files:-

usb_hal.c

usb_hal.h.

usb_common.h

4.2.USBCore

The USBCore layer handles standard USB requests common to all USB devices during the enumeration stage. This means that a developer can concentrate on any class or vendor specific implementation. The USBCore requires initialising with the descriptors specific to the device being implemented. It uses the USBHAL, which it initialises, to access the particular HW.

The following Get_Descriptor requests are handled:

- Device
- Configuration
- String
  - Language ID (Only a single language ID is supported and this is currently English)
  - Manufacturer
  - Product
  - Serial Number

The USBCDC API consists of a single function called ‘USBCORE_Init’. This initialises the USBCore and the HAL. In addition to passing device descriptors to this function it also requires call back functions for the following conditions:

- A Setup packet has been received that this layer can’t handle. This enables a higher layer to handle class or vender specific requests.
- A Control Data Out has completed following a setup packet that is being handled by the layer above.
- The USB cable has been connected or disconnected.
- An unhandled error has occurred.

The HAL module consists of the following files:-

usb_core.c

usb_core.h.

usb_common.h

4.3. Renesas USB Function API (R_USBF)

This is an optional API layer that can be used by an application rather than calling the USB Stack functions directly. It provides a generic API regardless of the USB class being used. The sample applications all use this API exclusively but as not all the functionality of the USB stack is available through this API it is realised that some applications may need to access the USB stack directly.

Here are the functions that make up the R_USBF API.

Note that the description here is a generic one, each class has its own specific implementation.

Name	Description
R_USBF_Open	Open the device.
R_USBF_Close	Close the device.
R_USBF_Read	Perform a read operation.
R_USBF_Write	Perform a write operation.
R_USBF_Control	Perform any other operation not supported by any of the above functions.

This API is accessible via a pointer to a R_USBF_API structure which must be obtained by a call to function:-

R_USBF_API* Get_API_R_USBF(R_USBF_CLASS_TYPE type)

File 'r_usbf.h' contains the definition of the common parts of the R_USBF interface. A particular class will have a file called 'r_usbf_xxx.h' which will contain any class specific extensions such as IOCTRL codes for the R_USBF_Control function. A particular class implementation of the interface will be in file 'r_usbf_xxx.c'.

4.4.Human Interface Device Class

The HID class as the name suggests is commonly used for things like keyboards, mice and joysticks where a human's action is causing the need for communication. However this does not need to be the case. The HID class is suitable for any device where the communication can be achieved by sending data in 'reports' of a predefined size where the data transfer rate is not critical.

The HID class has been supported by Microsoft Windows 98 onwards. Support is both at kernel level and user level. When a HID type device is plugged into a Windows PC it will be automatically recognised and Windows will load its own drivers for it, so there is no need to develop a custom Windows driver or even a Windows 'inf' file.

This implementation of the HID class supports a single IN report and a single OUT report. Both Interrupt IN and Control IN (via Get_Report) transfers are supported for sending a report to the host. Reports from the host must use Control OUT (via Set_Report).

Here are the functions that make up the USBHID API.

Name	Description
USBHID_Init	Initialise the HID module. Register a callback functions to be called when a report is received from the host. Provide the initial contents of a report to send to the host. Initialises the Core and HAL layers.
USBHID_ReportIN	Send a report to the host using Interrupt IN transfer.

The HID module consists of the following files:-

usb_hid.c

usb_hid.h.

usb_descriptors.c

usb_descriptors.h

usb_common.h

4.5.Communication Device Class

The CDC ACM allows a host to see a device as a standard serial (COM) port. This is particularly useful when working with legacy applications that use serial communications.   Bulk IN and Bulk OUT transfers are used for all non-setup data.

The CDC module utilises the USBCore layer for the processing of all standard requests. In addition it processes the following class requests.

GET_LINE_CODING

SET_LINE_CODING (As required by MS HyperTerminal)

SET_CONTROL_LINE_STATE

The CDC class is supported by MS Windows so there is no need to develop a custom Windows kernel driver. However a custom 'inf' file is required, the sample CDC application includes such a file. When a CDC ACM device is plugged into a Windows PC an additional (virtual) COM port will become available that applications can use just like a standard COM port.

Here are the functions that make up the USBCDC API.

Name	Description
USBCDC_Init	Initialise the CDC module. This also initialises the Core and HAL layers.
USBCDC_IsConnected	Returns the connected status of the device.
USBCDC_WriteString	A blocking function that sends a string to the host.
USBCDC_PutChar	A blocking function that sends a character to the host.
USBCDC_GetChar	A blocking function that gets a character from the host.
USBCDC_Write	A blocking function that sends a supplied data buffer to the host.
USBCDC_Write_Async	Starts an asynchronous write of a data buffer to the host. A call back is used to signal when the operation has completed.
USBCDC_Read	A blocking function that reads from the host into a supplied data buffer.
USBCDC_Read_Async	Starts an asynchronous read from the host into a data buffer. A call back is used to signal when the operation has completed.
USBCDC_Cancel	Cancel any asynchronous operations that are pending.

The CDC module consists of the following files:-

usb_cdc.c

usb_cdc.h.

usb_descriptors.c

usb_descriptors.h

usb_common.h

4.6.Mass Storage Class

The MSC class has become a very popular way for devices, such as cameras and USB Pens, to share data with PCs. The reason for the success is that when the device is plugged in to a host PC it appears to the PC as just another drive and therefore users can use familiar applications such as Windows Explorer to access the data. From Windows 2000 onwards the MSC class has been supported with no need for custom drivers or 'inf' files.

Bulk IN and Bulk OUT transfers are used for all non-setup data.

The MSC module utilises the USBCore layer for the processing of all standard requests. In addition it processes the following class requests.

- BULK_ONLY_MASS_STORAGE_RESET
- GET_MAX_LUN

In addition to supporting the standard USB protocol a MSC device must support a set of SCSI commands. All SCSI commands are sent packaged up in a Command Block Wrapper (CBW) within a Bulk OUT transfer. A data stage may follow in either direction and then to complete the SCSI command the device sends a status response in the form of a Command Status Wrapper (CSW). The following SCSI commands are supported:

- INQUIRY
- READ_CAPACITY10
- READ10
- REQUEST_SENSE
- TEST_UNIT_READY
- WRITE10
- VERIFY10
- PREVENT_ALLOW_MEDIUM_REMOVAL (Optional)
- MODE_SENSE6 (Optional, Limited support)

The USBMSC API consists of a single function called 'USBMSC_Init'. This initialises the MSC module and also the USBCore and HAL layers.

This implementation of the MSC class directly accesses a simple RAM Disk block driver that uses 24KB of the RSK's RAM. i.e. There is no separation between MSC class and MSC application. Hence when using a different memory this will need to be changed to access that rather than the sample RAM Disk.

The MSC module consists of the following files:-

usb_msc.c

usb_msc.h.

usb_msc_scsi.c

usb_msc_scsi.h

ram_disk.c

ram_disk.h

usb_descriptors.c

usb_descriptors.h

usb_common.h

Chapter 5.Applications

5.1.Introduction to Applications

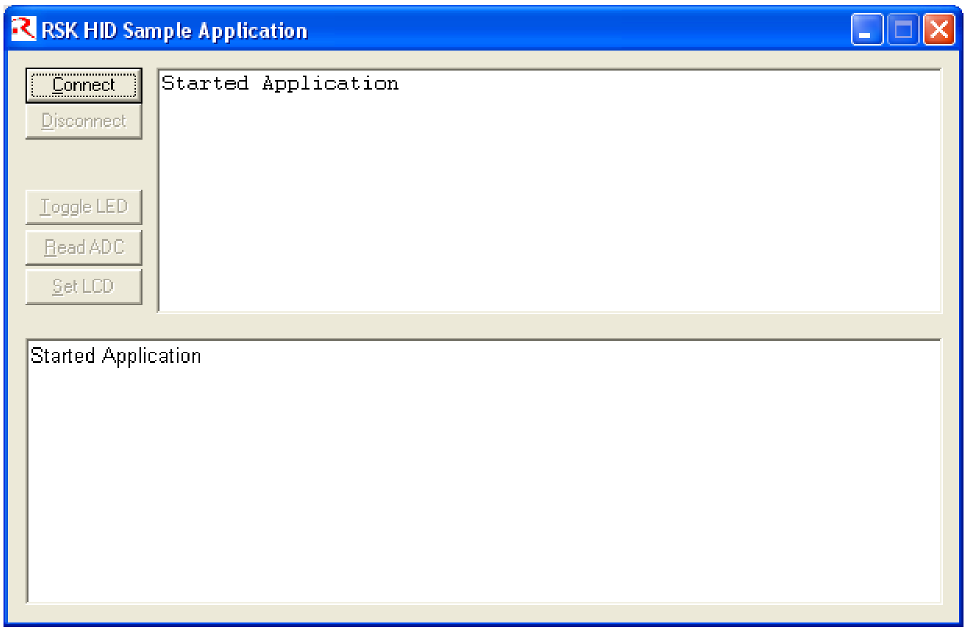
The following sections introduce the sample applications that can be used to demonstrate each of the USB solutions. The HID and LibUSB projects require specially written host applications that are supplied as both executables and as source. The CDC and MSC projects make use of standard Windows applications.

All the applications require that the RSK has been programmed with the appropriate sample code for the application. Details of how to program the RSK have been provided as part of the tutorial with the product. To obtain a digital copy of the manual please visit the Renesas web site at www.renesas.com/renesas_starter_kits and select your RSK from the list.

5.2.Human Interface Device Application

The HID host sample application is written for a Windows host PC and is called RSK_HID.

The pre-built executable has been provided with the project generator. Navigate to the release directory under the project and run RSK_HID.exe. The following window will be displayed:



```
Started Application
```

```
Started Application
```

Figure 2 - HID host PC application

Program the RSK with the HID application and run the code. Connect a USB cable between the PC and the RSK. The first time the device is connected to a specific USB port windows will detect the new device and automatically load the intrinsic HID class driver.

When windows has completed the enumeration process you need to make a connection from the application to the target. Click the “Connect” button and you will be asked to confirm the VID and PID of the device you wish to connect with. If you have not altered the firmware on the RSK to use your own VID and PID then the defaults will be correct. When a connection is successfully made information about the device will be displayed and the rest of the buttons will be enabled.

The “Toggle LED” button enables a LED on the RSK to be toggled on and off.

The “Read ADC” button will command the RSK to read its ADC and return the value back to the host where it will be displayed.

The “Set LCD” button allows the text of the LCD on the RSK to be changed.

To demonstrate that the RSK can also instigate communications you can press a switch on the RSK and this will be indicated back to the host resulting in a message being displayed on the dialog.

This demonstrates that Input and Output HID reports are being sent successfully between the RSK and the PC. The format of the reports is as follows:

Input Report:

- Byte 1
  - Bit 0 = LED status.
  - Bit 1 = ADC value valid indicator.
  - Bit 2 = Switch pressed indicator.
- Byte 2-5 = 32 bit, little endian ADC Value.

Output Report:

- Byte 1
  - Bit 0 = LED toggle request.
  - Bit 1 = ADC read request.
  - Bit 2 = LCD set request.
- Byte 2-17 = 16 ASCII Characters for LCD.

An input report is sent whenever a switch on the RSK is pressed or when the host requests a report.

An output report is sent whenever a user clicks on one of the dialog buttons.

The HID application functionality specific to USB consists of the following files:-

Target:

- usb_hid_app.c
- usb_hid_app.h

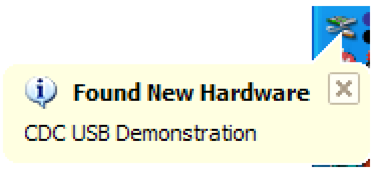
Host Application:

- \PC\RSK_HID\...

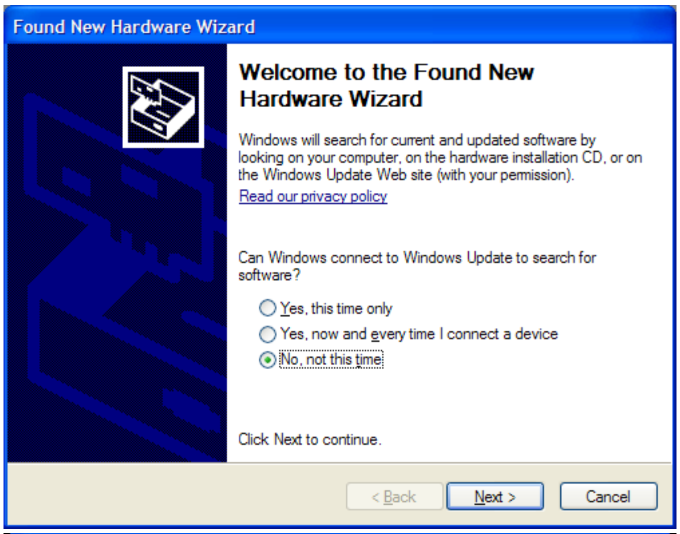
5.3.Communications Device Class Application

The CDC sample application demonstrates communication with a Windows PC using a standard terminal program. Windows provides a suitable application called HyperTerminal. Any other serial terminal program will be able to be used if available.

Program the RSK with the CDC application and run the code as described in the RSK tutorial manual. Connect a USB cable between the host PC and the RSK. The first time the device is connected, to a specific USB port, Windows will detect the new device and run the "Found New Hardware Wizard":


Found New Hardware
CDC USB Demonstration

Windows will present the following dialog where you should select "No, not this time".


Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

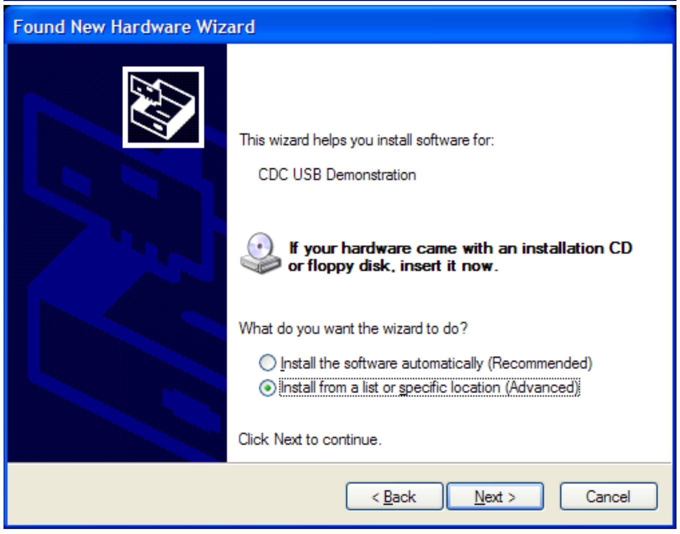
Read our privacy policy

Can Windows connect to Windows Update to search for software?

- Yes, this time only
- Yes, now and every time I connect a device
- [✓ No, not this time]

Click Next to continue.

In the following dialog select "Install from a list or specific location (Advanced)" to allow you to select the correct 'inf' file.


Found New Hardware Wizard

This wizard helps you install software for:

CDC USB Demonstration

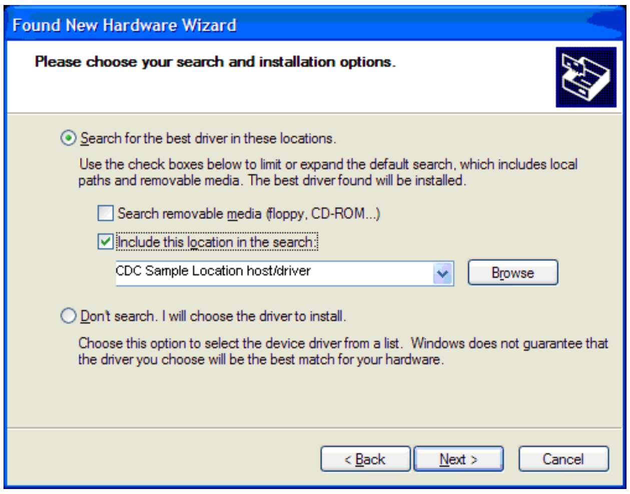
If your hardware came with an installation CD or floppy disk, insert it now.

What do you want the wizard to do?

- Install the software automatically (Recommended)
- [✓ Install from a list or specific location (Advanced)]

Click Next to continue.

Either type or browse to the location of the CDC project you have generated and built and then to the host/driver folder.


Found New Hardware Wizard

Please choose your search and installation options.

Search for the best driver in these locations.

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

☐ Search removable media (floppy, CD-ROM...)

☑ Include this location in the search:

CDC Sample Location host/driver

Browse

Don't search. I will choose the driver to install.

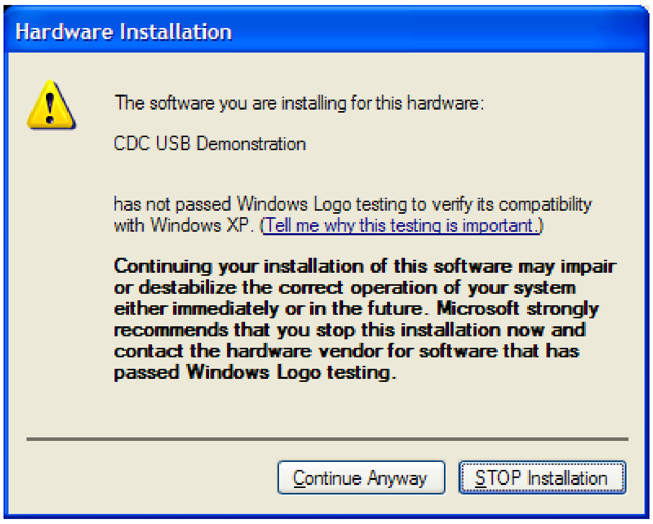
Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

< Back | Next > | Cancel

Press next to install the CDC support.

During the installation process a warning may be displayed as shown. Please choose "Continue Anyway" to install the driver.

Please review the Microsoft website for details of the Windows Logo Testing programme.


Hardware Installation

The software you are installing for this hardware:

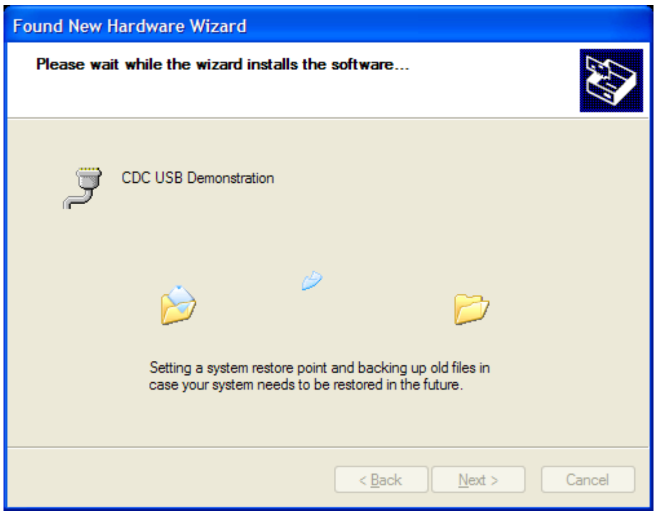
CDC USB Demonstration

has not passed Windows Logo testing to verify its compatibility with Windows XP. (Tell me why this testing is important.)

Continuing your installation of this software may impair or destabilize the correct operation of your system either immediately or in the future. Microsoft strongly recommends that you stop this installation now and contact the hardware vendor for software that has passed Windows Logo testing.

Continue Anyway | STOP Installation

Windows will then complete the installation of the CDC USB driver.


Found New Hardware Wizard

Please wait while the wizard installs the software...

CDC USB Demonstration

Setting a system restore point and backing up old files in case your system needs to be restored in the future.

< Back | Next > | Cancel

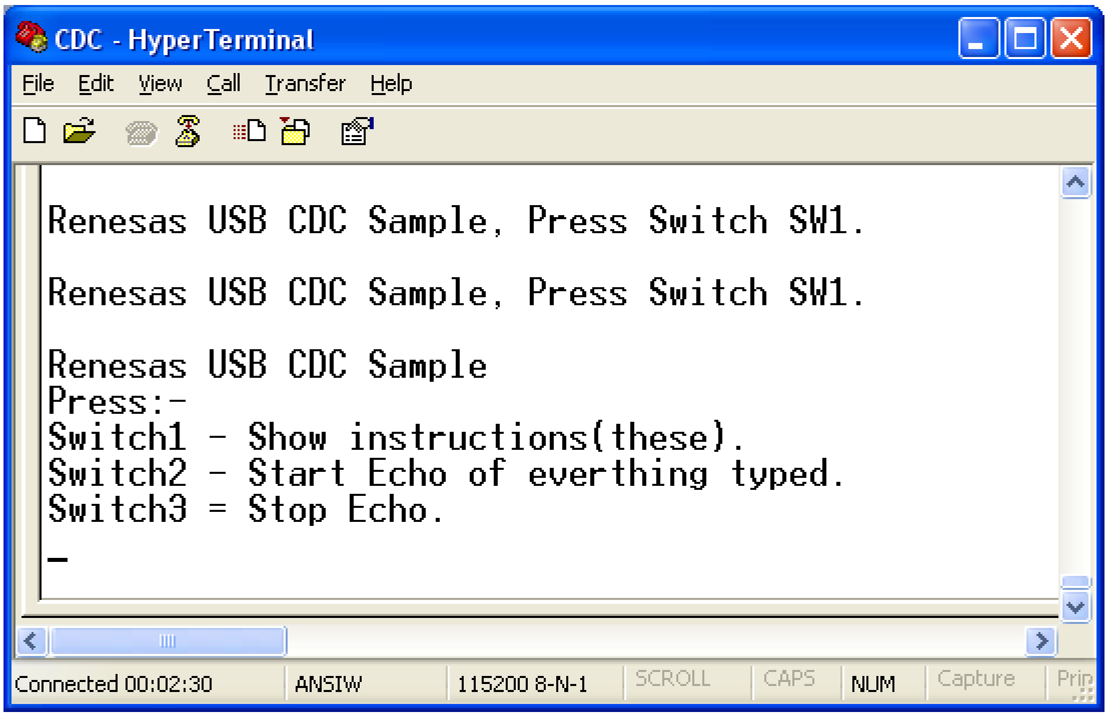
An additional COM port will become available to Window Applications. To be able to see the port that has been allocated you can open the Windows Device Manager window. To do this, go to the start menu and select run. In the dialog displayed type "devmgmt.msc". This will open the device manager. Expand the group of serial ports and the installed ports will be listed. When the Serial Terminal program connects to this COM port*, it will receive a repeating message from the RSK saying:

"Renesas USB CDC Sample, Press Switch SW1."

* Note that the configuration settings for such things as baud rate and parity are irrelevant for this virtual COM port.

Pressing SW1 on the RSK will stop this repeating message and will bring up the main menu as shown below.

To demonstrate two-way communication press SW2 to put the RSK into echo mode. In this mode anything typed on the Terminal will be read by the RSK and then echoed back to the terminal. Pressing SW3 cancels this echo mode.



```
Renesas USB CDC Sample, Press Switch SW1.

Renesas USB CDC Sample, Press Switch SW1.

Renesas USB CDC Sample
Press:-
Switch1 - Show instructions(these).
Switch2 - Start Echo of everthing typed.
Switch3 = Stop Echo.
_
```

Figure 3 - Serial communication dialog

The CDC application functionality specific to USB consists of the following files:-

Target:

usb_cdc_app.c

usb_cdc_app.h

Host:

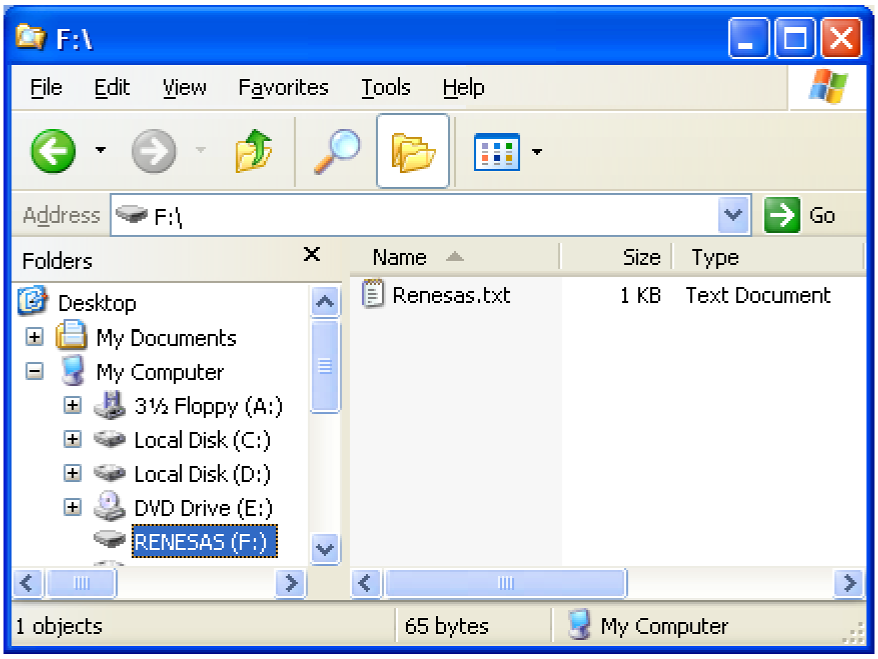
\PC\ CDC_Demo.inf

5.4.Mass Storage Class Demonstration

The MSC sample demonstrates how a host can view a MSC device as an external drive. There is no additional application for this as the MSC support is inherent in Windows XP.

Start the MSC sample application running on the RSK then connect the RSK to a Windows PC via a USB cable.

Using Windows Explorer, or similar, look to find the new drive that Windows will have mounted. This drive is viewing the contents of the sample RAM Disk on the RSK. It has been formatted with a FAT file system and given a volume name of “RENESAS”. The available space for data is 4KB. It includes one example file called Renesas.txt which can be opened edited and saved. As you would expect from a normal drive you can also copy files to it, although remember that this is a RAM Disk that will loose its contents when power is removed from the RSK.



Name	Size	Type
Renesas.txt	1 KB	Text Document

1 objects | 65 bytes | My Computer

Figure 4 - Windows Explorer showing new Disk Drive mapping

There is an option in file ‘ram_disk.c’ that will prevent the RAM disk from initialising itself with a file system. To select this comment out the #define of FORMAT_WITH_FAT_Example. In this case Windows will report that the drive is not formatted and give the user the option of formatting it.

5.5.LibUSB

The LibUSB sample application is functionally similar to the previous HID application. The difference is that this sample includes software for a Windows host PC called RSK_LibUSB. The intention of this open source library is to provide a platform independent operating system interface allowing a device to be used on multiple operating systems with a common code base.

The target RSK code is not dependant upon any external library code however it is written to support the LibUSB functionality.

To use the supplied host SW you will need to first download LibUSB-Win32 which is a port of the LibUSB code for Windows 32 bit environments. This can be obtained from the LibUSB32 web site:

http://libusb-win32.sourceforge.net/#installation

Follow the instructions for "Device Driver Installation" (not for the "Filter Driver Installation."), this will lead you to download a file "libusb-win32-device-bin-x.x.x.x.tar.gz". Once unzipped you will have the files you require – there is no installer to run.

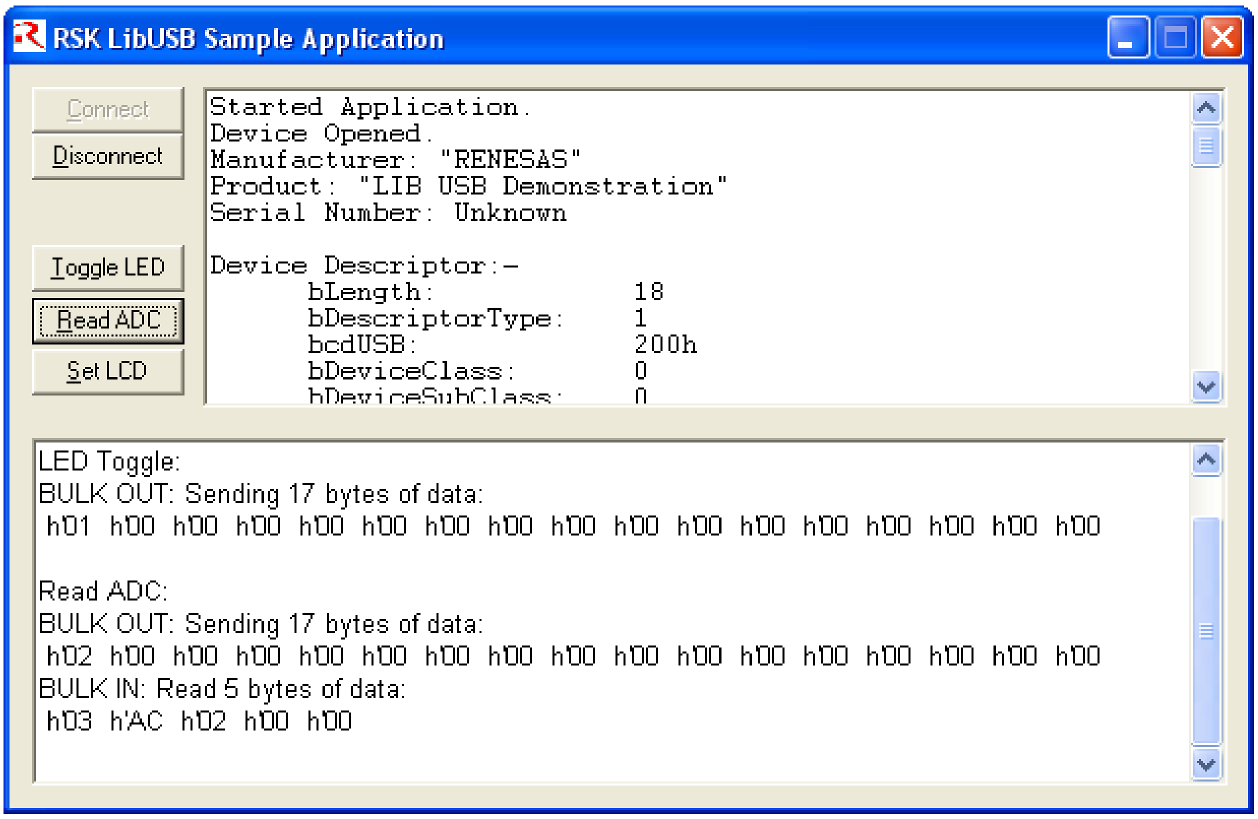
A sample inf file is provided in the LIBUSB sample project you have generated in the host/driver folder. Take a copy of this folder and copy it to the downloaded LibUSB folder "libusb-win32-device-bin-x.x.x.x\bin". (Note: Doing this saves the "Found New Hardware Wizard" having to ask where the driver files are.)

Program the RSK with the LibUSB sample code as described in the RSK tutorial manual. Then run the code. Connect a USB cable between the host PC and the RSK. The first time the device is connected to a specific USB port Windows will detect the new device and ask for the appropriate driver. Using the same procedure as described for the CDC project browse to the downloaded LibUSB folder where you have just copied the example inf file (libusb-win32-device-bin-x.x.x.x\bin). The LibUSB driver should now install.

Host SW:

This is supplied as a pre-built executable which is located in the release directory under the project and called RSK_LibUSB.exe. Alternatively you can build the supplied sorce code. To do this you will need the LibUSB files you downloaded - usb.h, LibUB.lib and LibUSB0.dll. Update the MS Visual C++ Project settings to point to these files as required.

Run the RSK_LibUSB.exe and the following Window will be displayed:



```
Started Application.
Device Opened.
Manufacturer: "RENESAS"
Product: "LIB USB Demonstration"
Serial Number: Unknown

Device Descriptor:-
        bLength:              18
        bDescriptorType:      1
        bcdUSB:               200h
        bDeviceClass:         0
        bDeviceSubClass:      0
```

```
LED Toggle:
BULK OUT: Sending 17 bytes of data:
 h'01  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00

Read ADC:
BULK OUT: Sending 17 bytes of data:
 h'02  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00  h'00
BULK IN: Read 5 bytes of data:
 h'03  h'AC  h'02  h'00  h'00
```

Figure 5 - LibUSB application window

Note:This is a screen shot after a connection has been made and the 'Set LCD' button and then the 'Read ADC' button have been used.

Program the RSK with the LibUSB sample code as described in the RSK tutorial manual. Then run the code. Connect a USB cable between the host PC and the RSK. The first time the device is connected to a specific USB port Windows will detect the new device and ask for the appropriate driver. This has been provided in a subdirectory of the sample code. Following the same process as described in the Communications Device Class Application above navigate to the LibUSB driver as described and install it.

It should now be possible to make a connection from the application. Click the "Connect" button and you will be asked to confirm the VID and PID of the device you wish to connect with. If you've not altered the firmware on the RSK to use your own VID and PID then the defaults will be correct. When a connection is successfully made information about the device will be displayed and the rest of the buttons will be enabled.

1. The "Toggle LED" button enables a LED on the RSK to be toggled on and off.
2. The "Read ADC" button will command the RSK to read its ADC and return the value back to the host where it will be displayed.
3. The "Set LCD" button allows the text of the LCD on the RSK to be changed.

To demonstrate that the RSK can also instigate communications you can press a switch on the RSK and this will be indicated back to the host resulting in a message being displayed on the dialog.

This demonstrates that data can be sent successfully between the RSK and the PC. A fixed sized format of data has been chosen for all messages, one for OUT and one for IN:

IN Message: (RSK to PC)

- Byte 1
  - Bit 0 = LED status.
  - Bit 1 = ADC value valid indicator.
  - Bit 2 = Switch pressed indicator.
  - Byte 2-5 = 32 bit, little endian ADC Value.

OUT Message: (PC to RSK)

- Byte 1
  - Bit 0 = LED toggle request.
  - Bit 1 = ADC read request.
  - Bit 2 = LCD set request.
- Byte 2-17 = 16 ASCII Characters for LCD.

An IN message is sent whenever a switch on the RSK is pressed, an Interrupt IN transfer is used for this. An IN message is also sent in response to certain OUT messages, a BULK IN transfer is used for this.

An OUT message is sent whenever a user clicks on one of the dialog buttons, a BULK OUT transfer is used for this.

The LibUSB application consists of the following files:-

Target:

- libusb_app.c
- libusb_app.h
- usbdescriptors.c

Host:

- \PC\RSK_LibUSB\...

Chapter 6.Additional Information

For details on how to use High-performance Embedded Workshop (HEW), refer to the HEW manual available on the CD or installed in the Manual Navigator.

For information about the SH7216 series microcontrollers refer to the SH7216 *Group Hardware Manual*

For information about the SH7216 assembly language, refer to the *SH Series Programming Manual*

For information about the E10A Emulator, please refer to the *SH Family E10A-USB Emulator User's Manual*

Further information available for this product can be found on the Renesas website at:

http://www.renesas.com/renesas_starter_kits

General information on Renesas Microcontrollers can be found on the following website.

Global: http://www.renesas.com/

For details of USB specifications and other USB information see http://www.usb.org/home

Renesas Starter Kit for SH7216
USB Sample Code User's Manual

Publication Date Rev.1.00 15.December.2009

Published by: Renesas Technology Europe Ltd.

Dukes Meadow, Millboard Road, Bourne End Buckinghamshire

SL8 5FH, United Kingdom

Renesas Starter Kit for SH7216
USB Sample Code User's Manual

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

REG10J0168-0100