

## 901: Taking Advantage of the Renesas USB API

**Description:** This lab is intended to help the user create and run USB applications using the free Renesas USB API's for Renesas MCUs

### Objectives

1. Understand USB basics
2. Hands on experience using Renesas USB APIs

### Lab Materials:

Please verify you have the following materials at your lab station.

Hardware:

- RSK1668 and add on LCD screen
- E10A Lite Debugger
- RSK 5V power supply
- Two USB cables

Software:

- H8/H8S C++ Compiler v 6.2
- Adobe pdf reader
- DT Tool

**Skill Level:** New to Renesas tools and MCUs, but has some basic C programming skills.

**Total Time to Complete Lab:** 100 minutes

### Lab Sections

- |          |   |                                   |
|----------|---|-----------------------------------|
| <b>1</b> | <b>Running the USB-Serial application</b>                                   | Time to complete task: 15 minutes |
| <b>2</b> | <b>Using the CDC API to transfer data from the board to the PC over USB</b> | Time to complete task: 30 minutes |
| <b>3</b> | <b>Run the HID program</b>  | Time to complete task: 15 minutes |
| <b>4</b> | <b>Create Report Descriptor for custom HID device</b>                       | Time to complete task: 40 minutes |

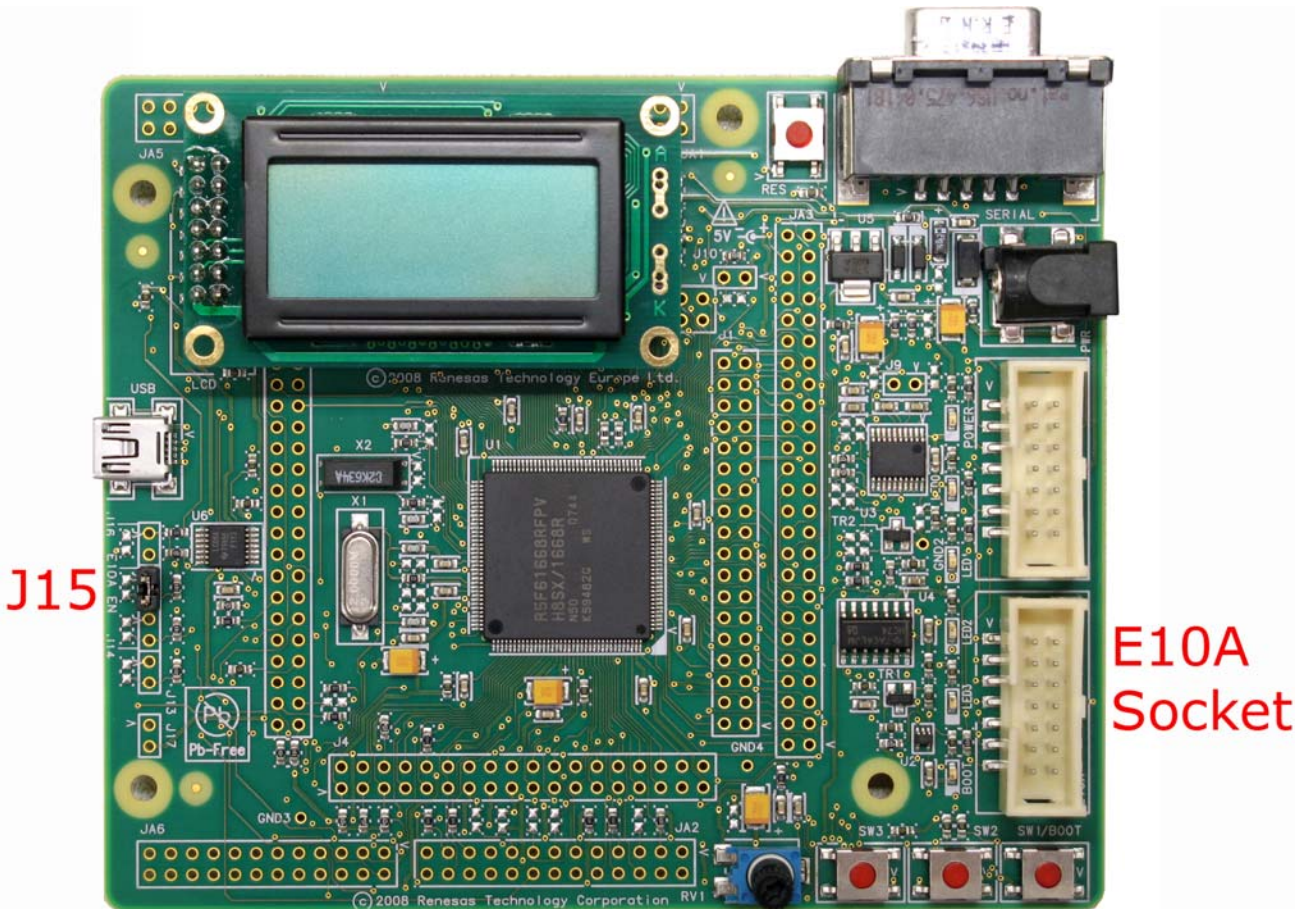
**1**

## Running the USB-Serial demo program.

Time to complete task: 15 minutes

### Overview:

In this section, we will run the sample program that implements a CDC class device. In the following sections, we will modify the program for additional functionality.



### Procedural Steps


1. On the RSK1668, ensure that jumper J15 is connected.
2. **Connect** the E10A to the PC using the provided USB cable.
3. **Connect** the E10A to the RSK1668 using the provided connector.
4. **Connect** the 5V supply to the power socket on the RSK1668.
5. **Browse** to and select the HEW workspace (\*.hws) file from the following location:  
 c:\Workspace\USB\CDC\_Demo
6. In the "Select Emulator Mode" dialogue, **select** the H8SX/1668RF as the device, mode as E10A-USB Emulator and click OK.
7. **Click OK** for the System Clock (18 MHz) and the ID code (0000E10A) dialogue windows.
8. **Build** the project within HEW by selecting the menu item: Build >> Build All
9. **Download** the code by double-clicking on the .abs file under the Download Modules project folder in the navigation pane.
10. **Reset** and run the program by selecting the menu item: [menu] Debug >> Reset Go.
11. With another USB cable, connect the USB port on the RSK1668 to the PC.
12. After successful enumeration, the H8SX/1668 USB will appear as a COM port on the PC and can be viewed in Windows Device Manager under COM ports.



To view Device Manager, right click on My Computer and select the Manage option. On the left-side tab, select Device Manager and view COM port under Ports on the right hand pane.

### Questions: (Write Answers on Sheet at the End of the Lab)

1.1) In Windows Device Manager, what COM port number does the CDC USB Demonstration device show up as?

13. **Configure** a terminal emulator (HyperTerminal) with the appropriate COM port (Answer to the above question) for 115200 Baud, 8-N-1. (There is a shortcut to Hyper terminal on the Desktop)
14. After connection, typing any character on the terminal emulator results in it being transmitted to the H8SX/1668 and displayed on the RSK LCD screen. The typed in character is echoed back to the terminal emulator as well.
15. When any of the three switches on the RSK are pressed they generate an Interrupt Request (IRQ): Switch 1 – IRQ2, Switch 2 – IRQ4, Switch 3 – IRQ7. The IRQ ISRs transmit a fixed text string to the host PC, which is displayed on the terminal emulator.
16. **Close** Hyper-terminal before stopping the program once you are done.
17. **Stop** the program by clicking on the Stop button. 
18. **Do not close** the workspace yet. We will be using it for the next section.



Use the ResponseCard to indicate that you are done with this section. Proceed to the next section only after instructed to do so.

**2**

## Using the CDC API to transfer data from the board to the PC over USB

Time to complete task: 30 minutes

### Overview:

Below is a pseudo-code summary of the main() routine in CDC\_Demo.c. Read this thoroughly and compare with individual lines of code in the main() routine before proceeding further.

Call Initialization routines.

Reset the "bulk data received" flag.

```

While (1)
{
    If Bulk data is received from the host computer
    {
        Display received data onto the H8SX/1668 LCD screen.

        Echo the received data back to the host PC.
        This is done by initializing the "pStart" and "pEnd" pointers defined
        in the structure BULKDataIN to the start and end addresses of the
        buffer that contains the data we want to send back to the PC.

        Call the Set_TransmitReady_Flag() function to set the flag indicating
        to the USB block in the MCU that data is ready to be transmitted. Once
        this flag is set, the USB block will transmit data held in the area
        between the pointers BULKDataIN.pStart and BULKDataIN.pEnd.

        Reset the flag indicating that data has been received from the host PC
    }
    Else if bulk data is not received from the host computer
    {
        Read 8 bit data from A2D register into a temporary variable.

        Convert A2D data to ASCII using the UL2A() function call. This function
        writes the converted ASCII value into the software buffer "adc_buffer".
        The function also returns the size of the data after it is converted
        from 8 bit binary to ASCII.
        Write "Newline" and "Carriage Return" characters into adc_buffer for a
        clearer display on hyper-terminal.
    }
}
    
```

In this section, the objective is to use the provided USB API and transfer data from the buffer "adc\_buffer" to a Hyper-terminal session on the PC. The firmware should only send this data to the PC when the user sends "Ctrl+z" (0x1a) from the Hyper-terminal session. In the main loop, while there is no USB activity, the firmware repeatedly reads the ADC and fills data into the buffer "adc\_buffer".

### Procedural Steps:

1. **Open** the file USBDescriptors.c.
2. **Locate** the descriptors defined in this file.

**Questions: (Write Answers on Sheet at the End of the Lab)**

2.1) How many endpoints are declared in the descriptor table? (Look for Endpoint Descriptors). Are these all the endpoints available on this device? (Refer to section 19.1 in the datasheet from the Datasheet folder in the "projects" panel)

2.2) What are the endpoint numbers for the Interrupt and Bulk endpoints? Are these numbers tied to the hardware or can they be modified? (Refer to the datasheet from the Datasheet folder in the project files panel)


3. **Locate** USB interrupt vectors in the file usb.c.

**Questions: (Write Answers on Sheet at the End of the Lab)**

2.4) What is the name of the data structure used to store bulk data from the host? (Look at the `RxDataBuff` structure typedef'd in USB.h).

2.5) What is the name of the data structure used to store the start and end addresses of bulk data to be sent to the host?

2.6) Which software function sets the bit (`EP2EMPTY`) that indicates to the USB block that Bulk data is ready to be transmitted?

4. **Modify** the main() routine in CDC\_demo.c so that characters are not echoed back onto the Hyper-terminal. This can be done by either not initializing the pointers for data to be transmitted, or by not indicating to the USB block that data is ready to be transmitted. (Always close hyper-terminal session before stopping the program).
5. **Modify** the main routine so that all the data from the buffer `adc_buffer` is sent to the hyper-terminal when "Ctrl+z" (`0x1a`) is sent from the host. (Hint: Parse the receive buffer "`BulkDataOUT.Data[ ]`" to look for the "Ctrl+z" (`0x1a`) character).
6. **Test** your code for the desired functionality.
7. **Click** on the **Disconnect** button.  to disconnect from the MCU.
8. **Close** Workspace.



Use the ResponseCard to indicate that you are done with this section. Proceed to the next section only after instructed to do so.

**3**
**Run the HID Demonstration program**

Time to complete task: 15 minutes

**Overview:**

Run the HID demo.

**Procedural Steps:**

1. On the RSK1668, ensure that jumper J15 is connected.
2. **Connect** the E10A to the PC using the provided USB cable.
3. **Connect** the E10A to the RSK1668 using the provided connector.
4. **Connect** the 5V supply to the power socket on the RSK1668.
5. **Browse** to and select the HEW workspace (\*.hws) file from the following location:  
     c:\Workspace\USB\HID\_Demo
6. In the "Select Emulator Mode" dialogue, **select** the H8SX/1668RF as the device, mode as E10A-USB Emulator and click OK.
7. **Click OK** for the System Clock (18 MHz) and the ID code (0000E10A) dialogue windows.
8. **Build** the project within HEW by selecting the menu item: Build >> Build All
9. **Download** the code by double-clicking on the .abs file under the Download Modules project folder in the navigation pane.
10. **Reset and run** the program by selecting the menu item: [menu] Debug >> Reset Go.
11. With another USB cable, connect the USB port on the RSK1668 to the PC.
12. **Run USB.exe** from  
     c:\Workspace\USB\HID\_Demo\USBDemo.exe
13. **Click** on the Connect button on the GUI. The button will gray out if the connection is successful.
14. **Click** on the LEDs to toggle them and depress the buttons on the board to toggle state.

**Questions: (Write Answers on Sheet at the End of the Lab)**

- 3.1) What is the size of the report descriptor and what is the name of the macro used to define the size of the report descriptor?
- 3.2) What is the size of the Input Report in bytes?
- 3.3) What kind of Usage Page is the Report Descriptor using?
- 3.4) In the report descriptor, how many bits are defined for the input report and how many for the output report?
- 3.5) Once a switch is depressed, describe the program flow in terms of functions executed and buffers modified. (Hint: Refer to Switch ISRs defined in interrupts.c)

15. **Close** the workspace after you are done.



Use the ResponseCard to indicate that you are done with this section. Proceed to the next section only after instructed to do so.

## 4

## Create Report Descriptor for custom HID device

Time to complete task: 40 minutes

### Overview:

In this section, we will modify the report descriptor from the HID demo program to create a custom HID device. The new report descriptor will allow us to control Windows Media player functions like Play, Pause, Scan, Volume etc on the PC using the RSK.

The main purpose of the Report Descriptor is to describe the format of the data that will be transferred between the host and the function. In order to create a new Report Descriptor we will use the DT tool.

The DT Tool is a GUI program downloadable from usb.org. The GUI lets the user select features that they need in a report descriptor and the tool creates the list of hex values that make up the descriptor. This hex list can then be inserted into the device firmware and used during the enumeration process.

### Procedural Steps:

1. Open the workspace (HWS file) under  
`c:\Workspace\USB\HID_Demo_Clean`
2. Connect to the device using the same steps as in session 3.

### Questions: (Write Answers on Sheet at the End of the Lab)

4.1) List the items that **must** be present in a report descriptor. Refer to page 25 in the document HID1\_11 in C:\workspace\usb\documents

3. Run the DT tool from `c:\Workspace\USB\DT\MSDEV\Projects\test\DT.exe`
4. Double click on USAGE\_PAGE under HID items and select Vendor Defined Page1 and compare the generated values with the first set of values in the current Report Descriptor. This step is to illustrate how the DT tool is used to create a report descriptor entry.
5. Clear the descriptor using the Clear Descriptor button on the DT tool. The last two steps were simply to demonstrate how the DT tool creates a report descriptor
6. To create a complete report descriptor, we have to create each element as specified in the order below. **DO NOT** clear the descriptor after each element.
  - a. Usage page: Consumer Devices
  - b. Usage : Consumer Control
  - c. Collection: Application
  - d. Logical Minimum: 0 (base 10)
  - e. Logical Maximum:1 (base 10)
  - f. Usage (Vol Up). **Usages for individual bits (this is where you set functions to be controlled by individual bits. Refer to page 79 in Hut1\_12.pdf (C\Workspace\USB\Documents) for a list of usages available)**
  - g. Usage (Vol Down)
  - h. Usage (SCN Next Track)
  - i. Report Size:1 bit
  - j. Report Count:3
  - k. Input: Data, Var, Abs, Pref (leave other options as default)
  - l. Usage (Mute)
  - m. Report Count:1
  - n. Input: Data, Var, Rel, Pref (leave other options as default)
  - o. Report Count:4
  - p. Input : Constant
  - q. End Collection

Note that the described input report byte looks like so:

0	0	0	0	Pause	SCN+	VOL-	VOL+
---	---	---	---	-------	------	------	------

7. Once the entire descriptor is defined in the DT tool, **replace** the report descriptor in the program with the values generated by the DT tool. Rather than copy pasting individual bytes from the DT tool, for convenience, the complete descriptor has been provided in the text file `ReportDescriptor.txt` in `c:\Workspace\USB\HID_Demo_Clean.`
8. Modify the report descriptor size variable (`HID_REPORT_DESCRIPTOR_SIZE`) to match the new descriptor size.
9. **Modify** the switch ISR's so that Switches 1, 2 and 3 can be used to control Volume up, Volume down, and Scan playlist respectively. In the ISR for each switch insert a line of code so that `g_InputReport.Data[g_ucSwitchPosition]` holds data with just the one bit set. For e.g. To set VOL+, set the `g_InputReport.Data[g_ucSwitchPosition]` to `0x01`, and for Scan Next set to `0x04` in the respective ISRs. This has to be done **before** the `SwitchHandler()` function is called.
10. **Modify** the Product ID in the Device Descriptor to a different value. For example `0x22, 0x00`. (The Device Descriptor can be found in the `usbdescriptors.c` file.)
11. **Compile** and run the program.
12. **Open** Windows Media Player, and test to see if the RSK switches control volume and playlist scanning.
13. Refer to page 79 in `Hut1_12.pdf` and try to set different usages in the Report Descriptor and control them over the RSK.



1. Independent bits have to be configured unlike previous example. Thus Report Size = 1 bit, and Report Count is the total number of bits defined.
2. Bits are assigned in the order that they are declared in the descriptor. To declare a bit, simply assign a usage. E.g.: `0x09, 0xB1`, assigns one bit for Pause control (Refer to page 79 in `Hut1_12.pdf` for a list of usages available). If this is the first usage declared under 'f' in above sequence, then bit 0 in the report descriptor will control Pause function.
3. Logical maximum and minimum are the max and min values that can be held in a Report Size.
4. Values that do not change or are not used in the byte have to be declared as constants

## Answer Page

### Section 1: Running the USB-Serial application

1.1) In Windows Device Manager, what COM port number does the CDC USB Demonstration device show up as?

---

---

### Section 2: Using the CDC API to transfer data from the board to the PC over USB

2.1) How many endpoints are declared in the descriptor table? Are these all the endpoints available on this device? (Refer to section 19.1 in the datasheet from the Datasheet folder in the "projects" panel)

---

---

2.2) What are the endpoint numbers for the Interrupt and Bulk endpoints? Are these numbers tied to the hardware or can they be modified? (Refer to the datasheet from the Datasheet folder in the project files panel)

---

---

2.4) What is the name of the data structure used to store bulk data from the host? (Look at the `RxDataBuff` structure typedef'd in `USB.h`).

---

---

2.5) What is the name of the data structure used to store the start and end addresses of bulk data to be sent to the host?

---

---

2.6) Which software function sets the bit (`EP2EMPTY`) that indicates to the USB block that Bulk data is ready to be transmitted?

---

---

### Section 3: Run the HID Demonstration program

3.1) What is the size of the report descriptor and what is the name of the macro used to define the size of the report descriptor?

---

---

3.2) What is the size of the Input Report in bytes?

---

---

3.3) What kind of Usage Page is the Report Descriptor using?

---

---

3.4) In the report descriptor, how many bits are defined for the input report and how many for the output report?

---

---

3.5) Once a switch is depressed, describe the program flow in terms of functions executed and buffers modified. (Hint: Refer to Switch ISRs defined in interrupts.c)

---

---

---

---

---

---

---

---

---

---

### Section 4: Create Report Descriptor for custom HID device

4.1) List the items that **must** be present in a report descriptor. Refer to page 25 in the document HID1\_11 in C:\workspace\usb\documents

---

---

---

---

---

---

---

---