# RX Family

R01AN2296EJ0130
Rev.1.30
Jul 31, 2024

**Sample Program using USB Peripheral Communication Device Class Driver (PCDC) for USB Mini Firmware to communicate via USB with USB Host Firmware Integration Technology**

## Introduction

This document describes the following sample firmware: USB Peripheral Communication Devices Class Driver using Firmware Integration Technology. The sample firmware is referred to below as the PCDC.

When developing an actual software, be sure to use the "USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology Application Note" (Document number: R01AN2166) together with the user's manual for each MCU (Hardware). In addition, also refer to the " USB Peripheral Communication Device Class Driver (PCDC) for USB Mini Firmware Firmware Integration Technology Application Note" (Document number: R01AN2170), if necessary. "USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology Application Note" (Document number: R01AN2166) is located in the "reference_documents" folder within the package.

## Target Device

RX111 Group
RX113 Group
RX231 Group
RX23W Group
RX261 Group

The operation of this program has been confirmed using the Renesas Starter Kits (RSK), the Renesas Solution Starter Kit (RSSK) or EK.

## Contents

# 1. Introduction

## 1.1 Functions

The PCDC conforms to the Abstract Control Model of the USB communication device class (CDC) specification and implements communication with the USB host PC.

The PCDC provides the following functionalities:

- Implements USB-serial conversion functionality and USB loopback communication functionality (echo mode).
- It is recognized as a communication class (virtual COM) when connected to a USB host.
- Supports communication by designating a virtual COM port in the terminal software.

## 1.2 FIT Module Configuration

The PCDC comprises the following FIT modules and a sample application:

**Table 1-1 FIT Module Configuration**

| FIT Module | Folder Name |
|---|---|
| RX Family Board Support Package Module Using Firmware Integration Technology | r_bsp |
| RX Family USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology | r_usb_basic_mini |
| RX Family USB Peripheral Communications Devices Class Driver(CDC) for USB Mini Firmware using Firmware Integration Technology | r_usb_pcdc_mini |
| RX Family DTC Module Using Firmware Integration Technology | r_dtc_rx |
| RX Family DMA Controller DMACA Control Module Firmware Integration Technology | r_dmaca_rx |
| RX Family BYTEQ Module Using Firmware Integration Technology | r_byteq |
| RX Family SCI Multi-Mode Module Using Firmware Integration Technology | r_sci_rx |
| RX Family LPC (Low Power Consumption) Module Firmware Integration Technology | r_lpc_rx |

Refer to the related documentation for details of each FIT module. Note that the latest versions of the FIT modules used by the sample firmware are available for download from the following website:

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

## 1.3 Note

This driver is not guaranteed to provide USB communication operation. The customer should verify operation when utilizing it in a system and confirm the ability to connect to a variety of different types of devices.

## 1.4 Operating Confirmation Environment

The operating confirmation environment for the PCDC is described below:

Table 1-2 Operation Confirmation Environment

| Item | Contents |
|---|---|
| C compiler | Renesas Electronics C/C++ compiler for RX Family<br>(The option "-lang=C99" is added to the default setting of IDE) |
| | GCC for Renesas RX<br>(The option "-std=gnu99" is added to the default setting of IDE) |
| | IAR C/C++ Compiler for Renesas RX |
| Real-Time OS | FreeRTOS<br>RI600V4 |
| Endian | Little Endian, Big Endian |
| USB Driver Revision Number | Rev.1.30 |
| Using Board | Renesas Starter Kit for RX111<br>Renesas Starter Kit for RX113<br>Renesas Starter Kit for RX231<br>Renesas Solution Starter Kit for RX231<br>EK-RX261 |
| Host Environment | The operation of this USB Driver module connected to the following OSes has been confirmed.<br>1. Windows® 10 |

# 2. Software Configuration

## 2.1 Module Configuration

The PCDC has a USB-serial converter mode and a USB loopback mode (echo mode). In USB-serial converter mode it uses the serial communication interface (SCI). LCD display and low-power control processing are implemented as a sample application.

Figure 2-1 shows the module configuration of the PCDC, and Table 2-1 lists the functions of the modules.

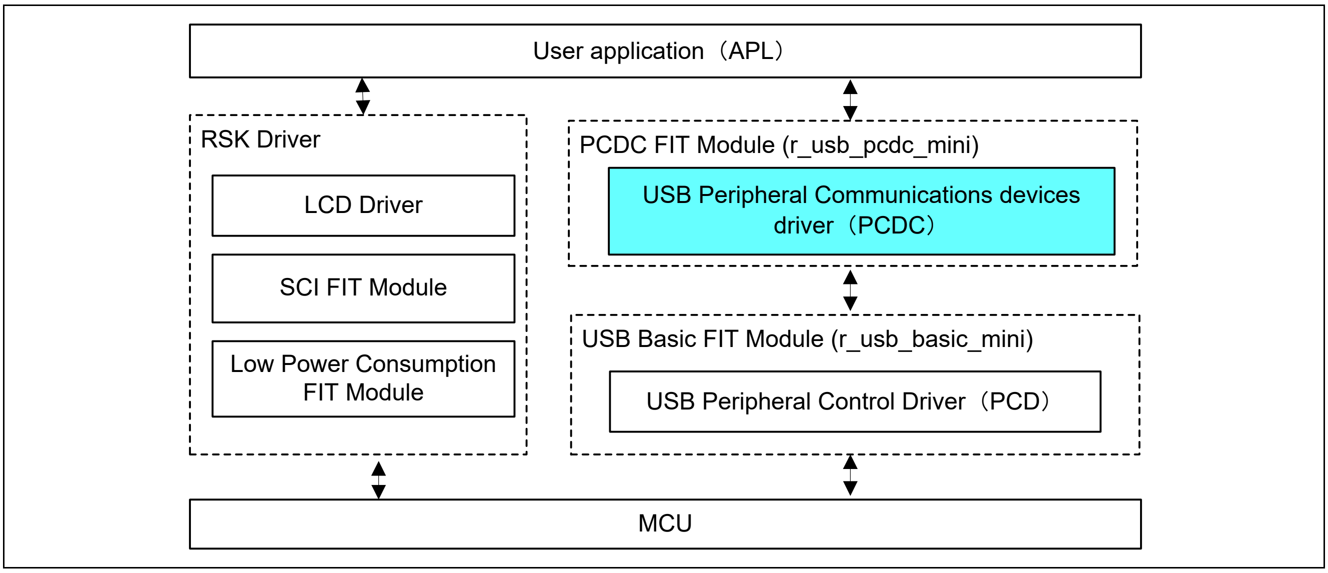



**Figure 2-1 Module Configuration**

**Table 2-1 Functions of Modules**

| Module Name | Function |
|---|---|
| APL | Sample application program |
| RSK driver | Sample application for using the peripheral functions of the RSK. |
| PCDC (r_usb_pcdc_mini) | CDC class driver<br>• Interprets requests from the USB host.<br>• Provides data transfer services between the APL and the USB host, via the PCD. |
| PCD (r_usb_basic_mini) | USB peripheral hardware control driver |

# 3. Setup

## 3.1 Hardware

### 3.1.1 Example Operating Environment

Figure 3-1 and Figure 3-2 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.

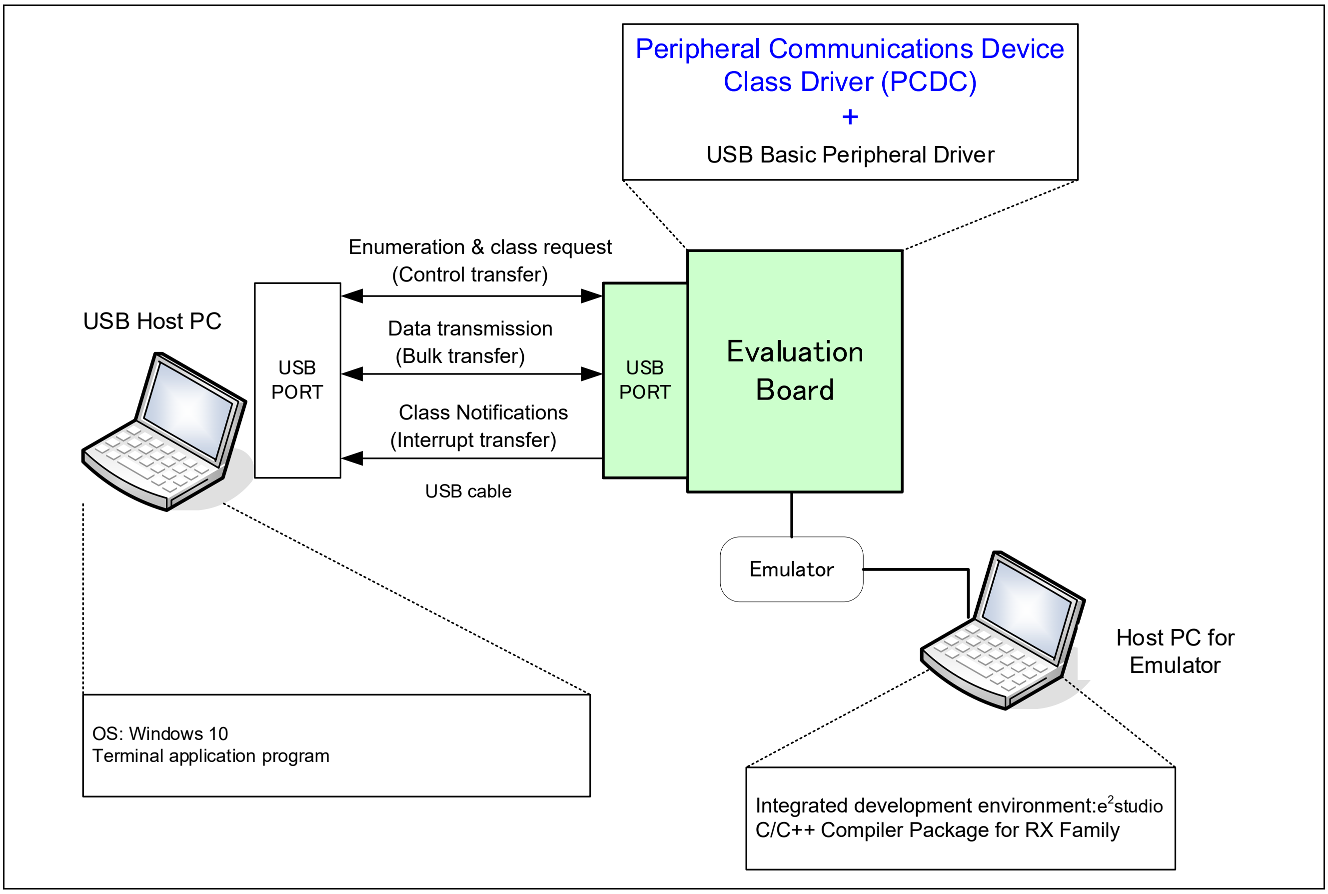



**Figure 3-1 Example Operating Environment (Echo Mode)**

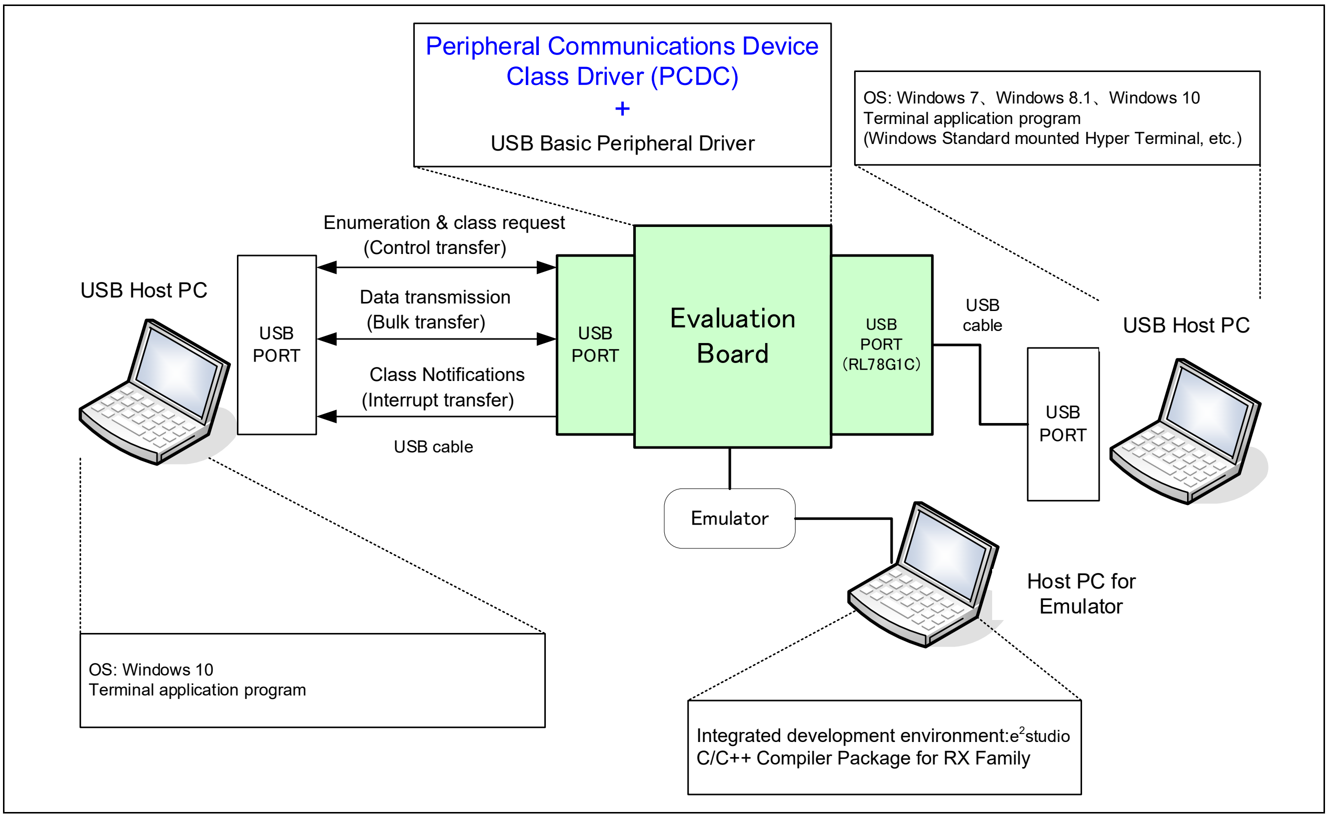



**Figure 3-2 Example Operating Environment (USB-Serial Converter Mode)**

Table 3-1 shows the evaluation board on which operation has been confirmed.

**Table 3-1 Evaluation Board on which PCDC Operation Has Been Verified**

| MCU | Evaluation Board |
|---|---|
| RX111 | RSKRX111 |
| RX113 | RSKRX113 |
| RX231 | RSKRX231 |
| RX23W | RSSKRX23W |
| RX261 | EK-RX261 |

### 3.1.2 RSK / RSSK / EK Setting

It is necessary to set RSK/RSSK to operate in the peripheral mode. Please refer to the following.

Table 3-2 **Jumper Setting**

| RSK / RSSK | Jumper Setting |
|---|---|
| RSKRX111 | J12: Shorted Pin2-3 |
| RSKRX113 | J12: Shorted Pin2-3 |
| RSKRX231 | J15: Shorted Pin2-3 |
| RSSKRX23W | J5: Shorted Pin1-2 |
| EK-RX261 | J18: Shorted Pin2-3, J19: Shorted Pin2-3 |

Note:

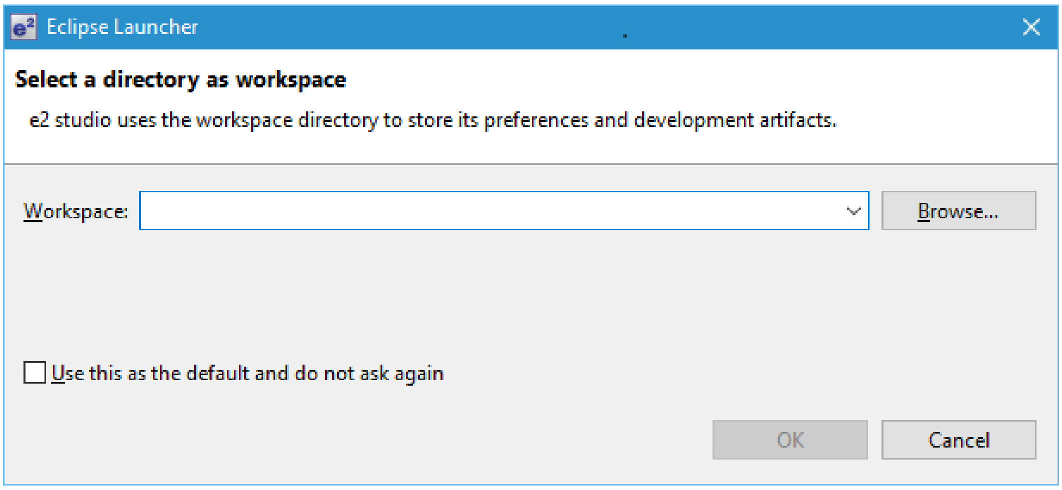
For the detail of RSK/RSSK/EK setting, refer to the user's manual of RSK/RSSK/EK.

## 3.2 Software

1) Setup e$^{2}$ studio

a) Start e$^{2}$ studio

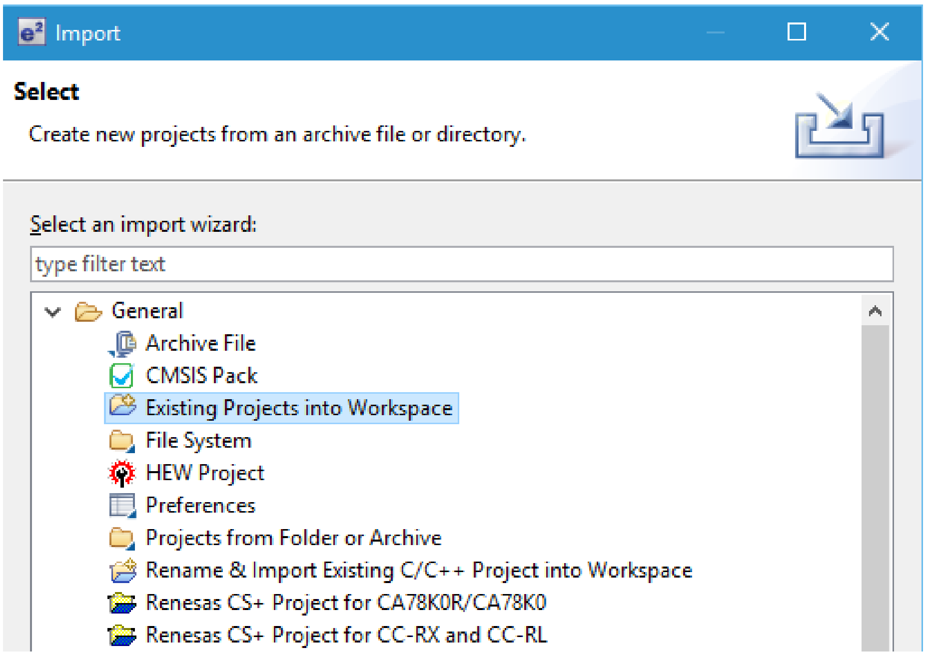
b) If you start up e$^{2}$ studio at first, the following dialog is displayed. Specify the folder to store the project in this dialog.



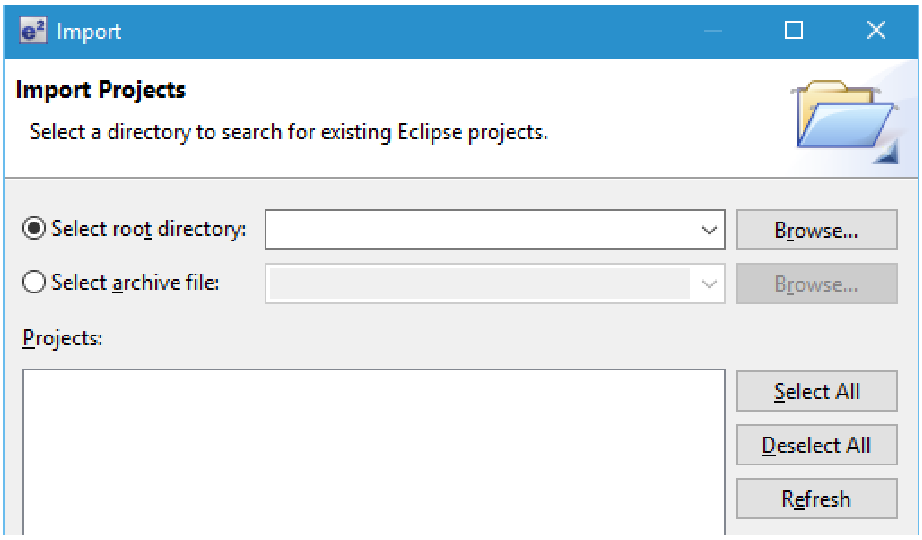

2) Import the project to the workspace

a) Select [File] > [Import]

b) Select [General] => [Existing Projects into Workspace]



c) Select the root directory of the project, that is, the folder containing the ".cproject" file.




3) Click "Finish".

You have now imported the project into the workspace. Note that you can import other projects into the same workspace.

4) Generate the binary target program by clicking the "Build" button.

5) Connect the target board to the debug tool and download the executable. The target is run by clicking the "Run" button.

# 4. Sample Application

## 4.1 Application Specifications

The main functions of the APL are as follows:

1) Echo mode (Loopback mode)（Note1）

Transmits data received from the USB host back to the USB host.

2) USB-serial converter mode（Note1）(Note2) (Note3) (Note4)

Transmits data received from the USB host back to a COM port and data received from the COM port to the USB host. Note that if a COM port error (parity error, framing error, or overrun error) occurs, the APL sends a “SerialState” class notification to the USB host.

3) Low-power functionality

This functionality transitions the MCU to low-power mode according to the status of the USB.

a. The APL transitions the MCU to sleep mode when the USB is suspended.

b. When the USB is detached (disconnected), the APL transitions the MCU to software standby mode.

（Note1） The selection of Echo mode or USB-serial converter mode is made by setting the "OPERATION_MODE" macro definition in the file “*r_usb_pcdc_apl_config.h.*”

（Note2） The COM port is connected to RL78G1C on the RSK/RSSK, so G1CUSB0 is used for USB-USB communication.

（Note3） In USB-serial converter mode, the following setting is required to RSK/RSSK.

a. The setting to change the channel number from SCI1 to SCI2 is required when using RSKRX111. Please set the follows to RSK.

a) Removes the option link resistance (0 ohm) on R224 and R116.

b) Mounts the option link resistance (0 ohm) on R37 and R38.

b. The setting to change the channel number from SCI1 to SCI6 is required when using RSKRX113. Please set the follows to RSK.

a) Removes the option link resistance (0 ohm) on R100 and R121.

b) Mounts the option link resistance (0 ohm) on R99 and R106.

c. The change is not required to RSK when using RSKRX231.

d. Please short Pin1-2 in J1(jumper) when using RSSKRX23W.

（Note4） To select USB-serial converter mode, the following FIT is required.

a) RX Family Serial Communication Interface Firmware Integration Technology

b) RX Family BYTEQ Module Firmware Integration Technology

## 4.2 Application Processing (for Non-OS)

The APL comprises two parts: initial settings and main loop. An overview of the processing in these two parts is provided below.

### 4.2.1 Initial Setting

Initial settings consist of MCU pin settings, USB driver settings, and initial settings to the USB controller.

### 4.2.2 Main Loop (Echo mode)

In Echo 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 completes, *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 enumeration with the USB host completes, the USB host sends an CDC class request to the CDC device. After it receives the CDC class request, the CDC device calls the *R_USB_GetEvent* function and the return value is set to *USB_STS_REQUST*. When the APL confirms *USB_STS_REQUEST*, it analyzes the received class request and performs processing corresponding to it.
3. When the *R_USB_GetEvent* function is called after reception of data from the USB host has completed, *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.
4. When the *R_USB_GetEvent* function is called after transmission of data to the USB host completes, *USB_STS_WRITE_COMPLETE* is set as the return value. When the APL confirms *USB_STS_WRITE_COMPLETE*, it calls the *R_USB_Read* function to make a data receive request for data sent by the USB host.
5. The processing in steps 3 and 4, above, is repeated.
6. When it confirms reception of a suspend signal from the USB host or DETACH, the APL performs processing to transition the CDC device (RSK/RSSK) to low-power mode. For information on low-power mode, refer to **4.4, MCU Low power consumption processing**. Note that confirmation of reception of a suspend signal or DETACH is performed by referencing the return value (*USB_STS_SUSPEND* or *USB_STS_DETACH*) of the *R_USB_GetEvent* function.

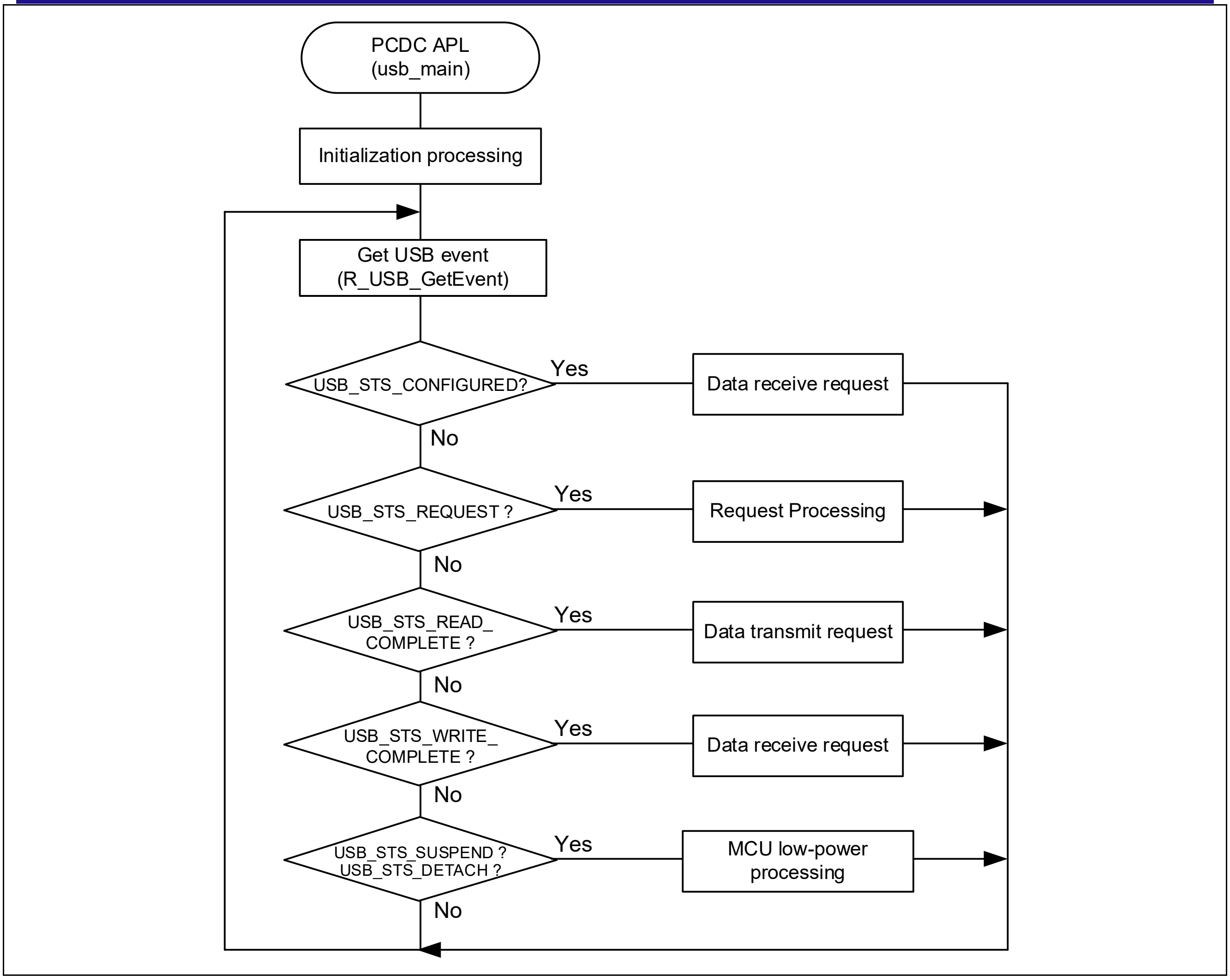



Figure 4-1 Main Loop processing (Loopback mode)

### 4.2.3 Main Loop (USB-serial converter mode)

The processing performed in USB serial convert mode is described below.

a. Reception of data from the USB host, and transmission of the received data to the COM port.

b. Transmission to the USB host of data received from the COM port.

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 completes, *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 Bulk data sent by the USB host.
2. When enumeration with the USB host completes, the USB host sends an CDC class request to the CDC device. After it receives the CDC class request, the CDC device calls the *R_USB_GetEvent* function and the return value is set to *USB_STS_REQUST*. When the APL confirms *USB_STS_REQUEST*, it analyzes the received class request and performs processing corresponding to it.
3. When the *R_USB_GetEvent* function is called after processing of the class request mentioned in 2, above, completes, the return value is set to *USB_STS_REQUST_COMPLETE*. The APL performs processing to make request information settings, etc.
4. When the *R_USB_GetEvent* function is called after reception of Bulk data by the USB host has completed, *USB_STS_READ_COMPLETE* is set as the return value. When the APL confirms *USB_STS_READ_COMPLETE*, it sets the reception size of the bulk data to the global variable. This global variable is referenced in step 6, below.
5. When the *R_USB_GetEvent* function is called after sending of USB data (refer to step 6, below) to the USB host has completed, *USB_STS_WRITE_COMPLETE* is set as the return value. When the APL confirms *USB_STS_WRITE_COMPLETE*, it references member type in the *usb_ctrl_t* structure to identify the device class on which data transmission has completed. It then sets the completion flag corresponding to the device class on which data transmission has completed. This flag is referenced in step 6, below.
6. When the *R_USB_GetEvent* function is called when no USB event has occurred, *USB_STS_NONE* is set as the return value. The processing of *USB_STS_NONE* consists of the transmission processing described below. Note that before the transmission processing described below, the global variable and the flags set in steps 4 and 5 above are referenced to determine if transmission is possible.
   (1). SCI transmission processing to send bulk data received from the USB host to the COM port
   (2). If a SCI error (parity error, framing error, overrun error, etc.) has been detected, class notification (serial state) transmit request processing to notify the USB host
   (3). Data transmit request processing to send the data received from the COM port to the USB host
7. If reception of a suspend signal from the USB host or DETACH is confirmed while steps 4 to 6 to, above, are repeating, the APL performs processing to transition the CDC device (RSK/RSSK) to low-power mode. For information on low-power mode, refer to **4.4, MCU Low power consumption processing**. Note that confirmation of reception of a suspend signal or DETACH is performed by referencing the return value (*USB_STS_SUSPEND* or *USB_STS_DETACH*) of the *R_USB_GetEvent* function.

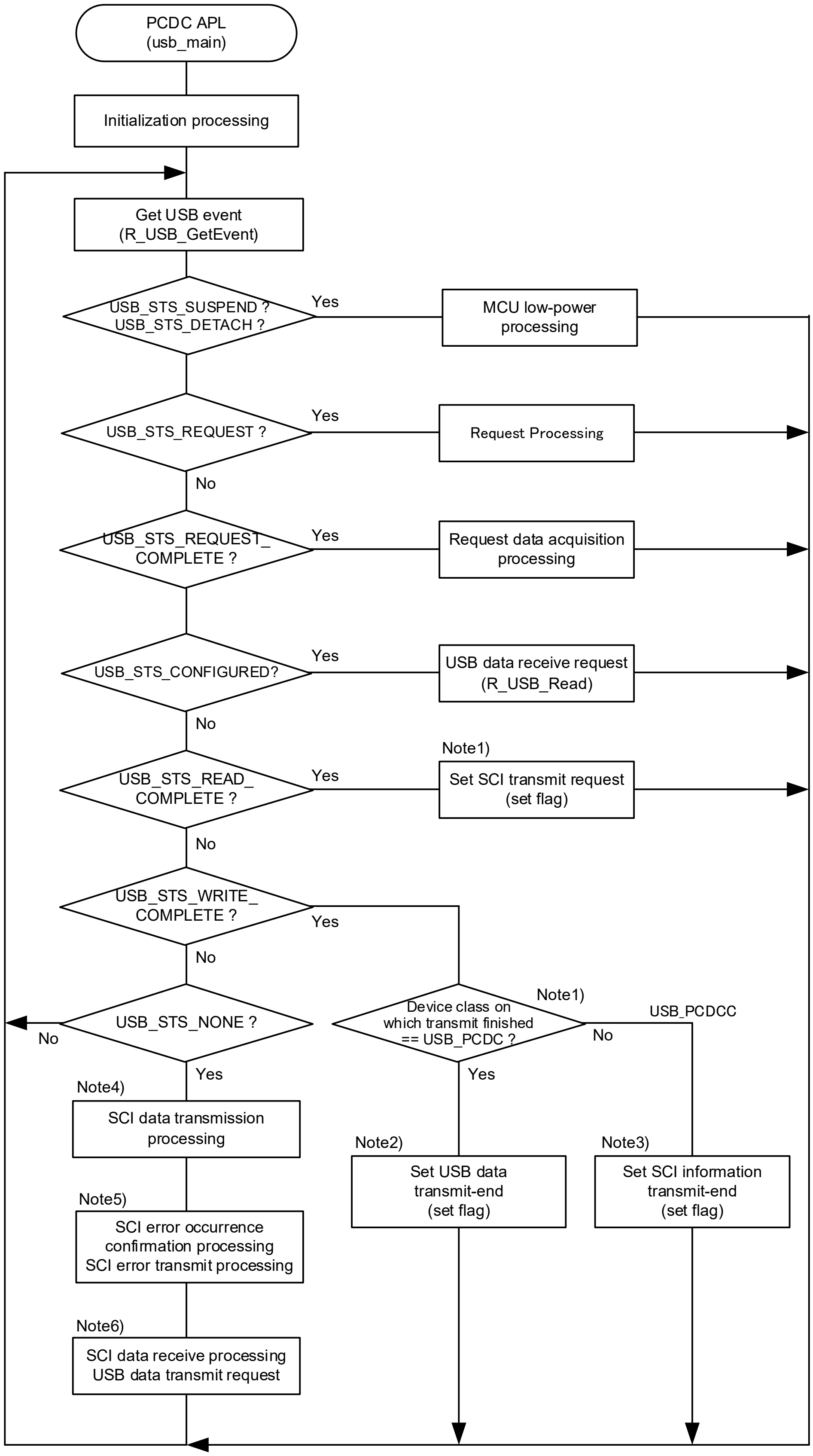



**Figure 4-2 Main Loop processing (USB-serial converter mode mode)**

(Note1) Sets the Bulk data reception size to the global variable. This global variable is referenced in the processing described in (Note4).

(Note2) Sets the USB data transmit-end flag. This flag is referenced in the processing described in (Note6).

(Note3) Sets the SCI information (serial state) transmit-end flag. This flag is referenced in the processing described in (Note5).

(Note4) References the global variable in (Note1) and performs SCI data transmit processing. If SCI data transmission completes normally, the SCI transmit request flag is cleared. If SCI data transmission fails, the SCI transmit request flag it not cleared so that SCI data transmit processing will be retried.

(Note5) Performs processing to check the SCI status for the occurrence of an error. If an error has occurred, the *R_USB_Write* function is used to send error information to the USB host.

(Note6) References the flag mentioned in (Note2) to confirm whether USB transmission is possible. If transmission is possible, the USB_Write function is used to send the received SCI data to the USB host. Note that confirmation of reception of data from the COM port is not performed until transmission of data to the USB host completes.

## 4.3 Application Processing (for RTOS)

The APL comprises two parts: initial settings and main loop. An overview of the processing in these two parts is provided below.

### 4.3.1 Initial Setting

Initial settings consist of MCU pin settings, USB driver settings, and initial settings to the USB controller.

### 4.3.2 Main Loop (Echo mode)

In Echo 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 performed by the loop is shown below.

1. When a USB-related event has completed, the USB driver calls the callback function (*usb_apl_callback*). In the callback function (*usb_apl_callback*), the application task (APL) is notified of the USB completion event using the real-time OS functionality.
2. In APL, information regarding the USB completion event was notified from the callback function is retrieved using the real-time OS functionality.
3. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_CONFIGURED*, APL performs a data reception request to receive data transmitted from the USB Host by calling the *R_USB_Read* function.
4. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_REQUEST*, APL performs processing in response to the received request.
5. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_READ_COMPLETE*, APL performs a data transmission request to send USB Host the reception data by calling the *R_USB_Write* function.
6. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_WRITE_COMPLETE*, APL performs a data reception request to receive the data sent from USB Host by calling the *R_USB_Read* function.
7. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_SUSPEND* or *USB_STS_DETACH*, APL performs processing to transition the CDC device (RSK) to low-power mode. For information on low-power mode, refer to **4.4, MCU Low power consumption processing**.

An overview of the processing performed by the APL is shown below:

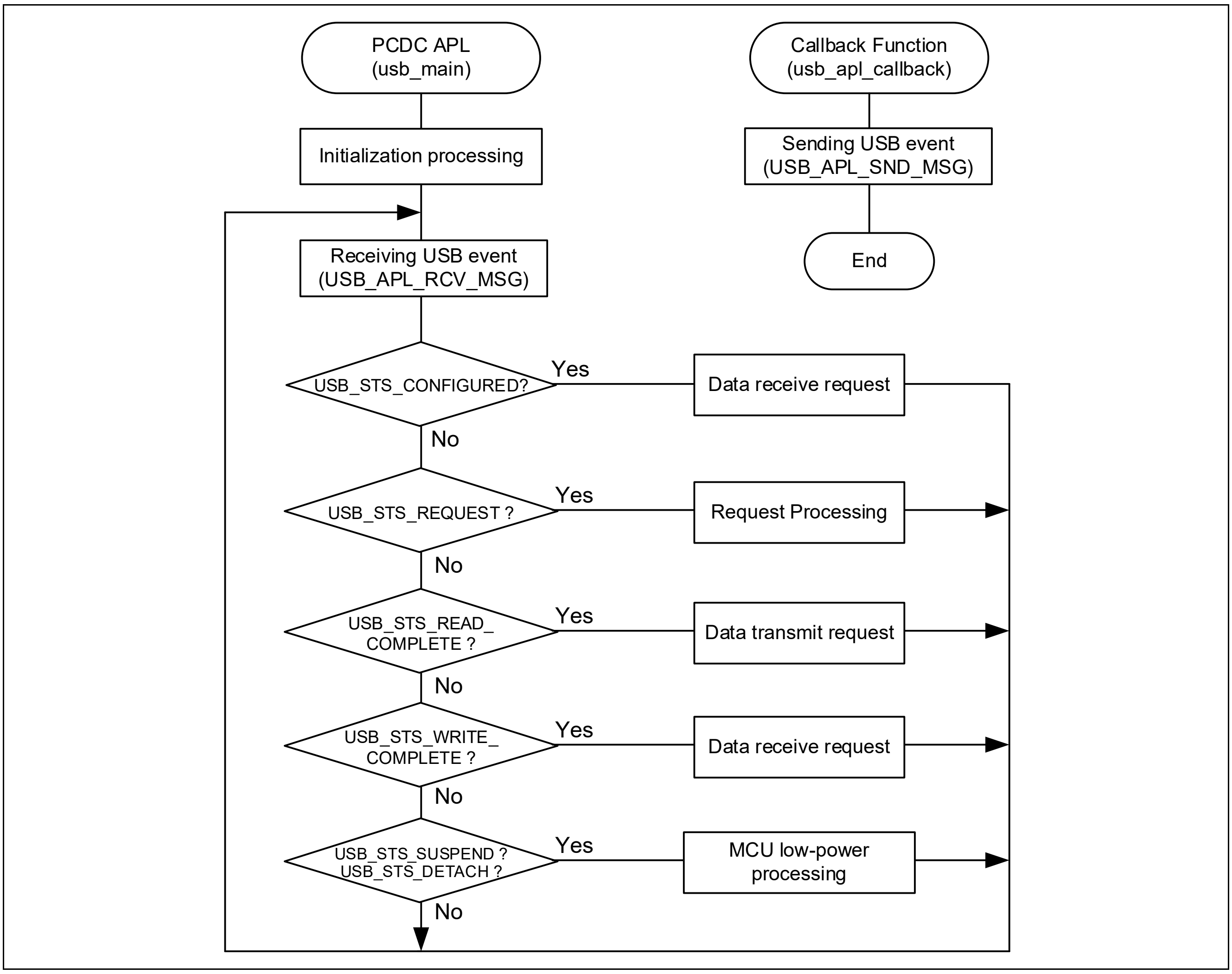



Figure 4-3 Main Loop processing (Echo mode)

### 4.3.3 Main Loop (USB-serial converter mode)

The processing performed in USB serial convert-back mode is described below.

a. Reception of data from the USB host, and transmission of the received data to the COM port.

b. Transmission to the USB host of data received from the COM port.

An overview of the processing of the main loop is presented below.

1. When a USB-related event has completed, the USB driver calls the callback function (*usb_apl_callback*). In the callback function (*usb_apl_callback*), the application task (APL) is notified of the USB completion event using the real-time OS functionality.
2. In APL, information regarding the USB completion event was notified from the callback function is retrieved using the real-time OS functionality.
3. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_CONFIGURED*, APL performs a data reception request to receive data transmitted from the USB Host by calling the *R_USB_Read* function.
4. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_REQUEST*, APL performs processing in response to the received request.
5. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_REQUEST_COMPLETE*, APL performs processing to make request information settings, etc.
6. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_READ_COMPLETE*, APL sets the reception data size to a gloval variable.
7. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_WRITE_COMPLETE*, APL references member *type* in the *usb_ctrl_t* structure to identify the device class on which data transmission has completed. It then sets the completion flag corresponding to the device class on which data transmission has completed. This flag is referenced in step 9, below.
8. If the USB completion event (the *event* member of the *usb_ctrl_t* structure) retrieved in step 2 above is *USB_STS_SUSPEND* or *USB_STS_DETACH*, APL performs processing to transition the CDC device (RSK) to low-power mode. For information on low-power mode, refer to **4.4, MCU Low power consumption processing**.
9. The following transmission processing is performed after completing the above processing. Note that before the transmission processing described below, the flags set in steps 5 and 7 above are referenced to determine if transmission is possible.

   (1). SCI transmission processing to send bulk data received from the USB host to the COM port and the data reception processing from USB Host.

   (2). If a SCI error (parity error, framing error, overrun error, etc.) has been detected, class notification (serial state) transmit request processing to notify the USB host

   (3). Data transmit request processing to send the data received from the COM port to the USB host

An overview of the processing performed by the APL is shown below:

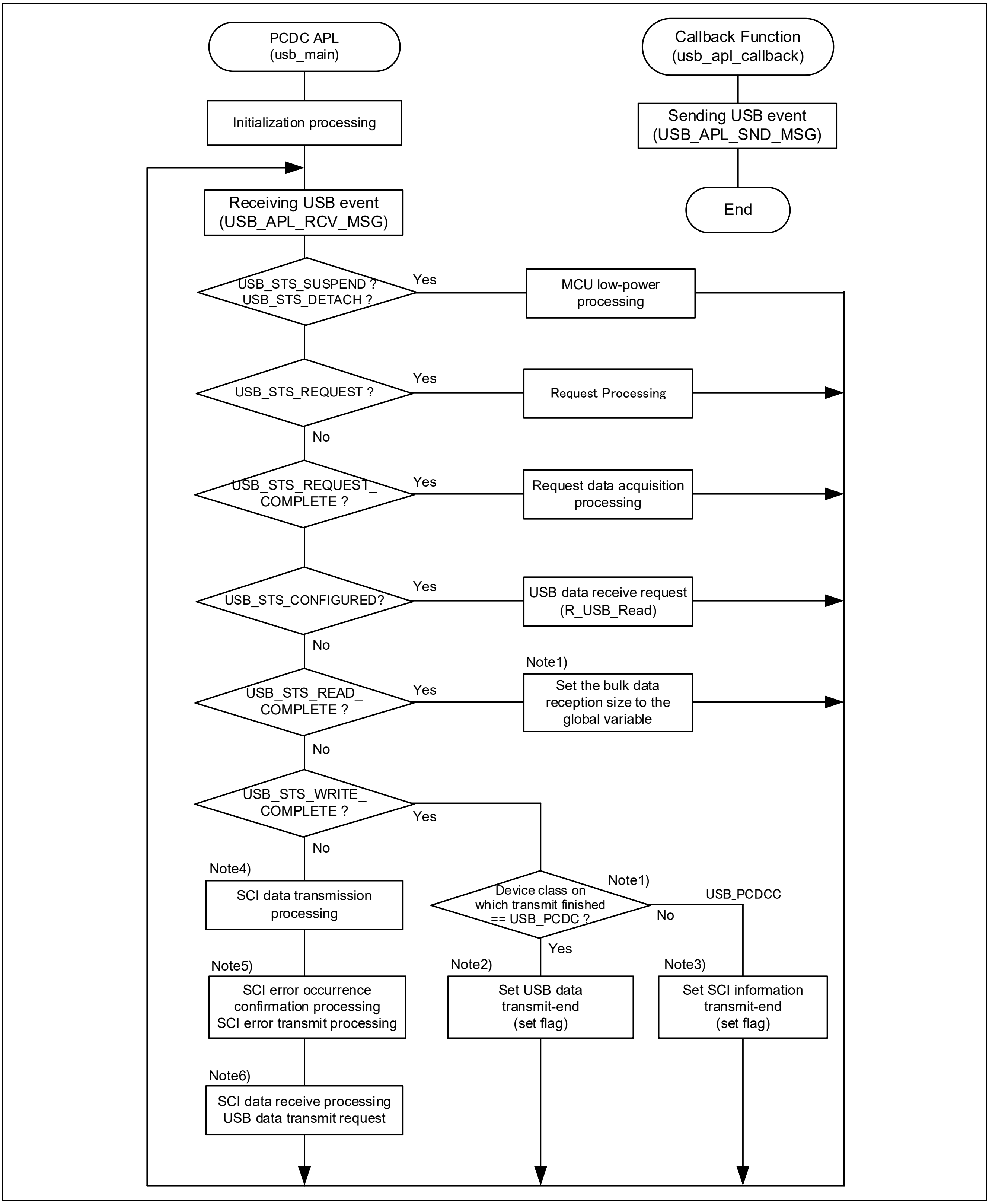



**Figure 4-4 Main Loop processing (USB-serial converter mode mode)**

(Note1) The reception data size is set to a global variable. This variable is refered in (Note4).

(Note2) Sets the USB data transmit-end flag. This flag is referenced in the processing described in (Note6).

(Note3) Sets the SCI information (serial state) transmit-end flag. This flag is referenced in the processing described in (Note5).

(Note4) References the global variable in (Note1) and performs SCI data transmit processing. If SCI data transmission completes normally, the SCI transmit request flag is cleared. If SCI data transmission fails, the SCI transmit request flag it not cleared so that SCI data transmit processing will be retried.

(Note5) Performs processing to check the SCI status for the occurrence of an error. If an error has occurred, the *R_USB_Write* function is used to send error information to the USB host.

(Note6) References the flag mentioned in (Note2) to confirm whether USB transmission is possible. If transmission is possible, the USB_Write function is used to send the received SCI data to the USB host. Note that confirmation of reception of data from the COM port is not performed until transmission of data to the USB host completes.

## 4.4 MCU Low power consumption processing

MCU low-power processing occurs when the conditions in Table 4-1 or Table 4-2 are met, causing a transition to low-power mode.

### 1. Non-OS

**Table 4-1 Conditions for Transition to Low-Power Mode**

| Transition Condition | | Transition Status |
|---|---|---|
| VBUS | USB State | |
| OFF | — | Software standby mode |
| ON | Suspend Configured | Sleep mode |
| ON | Other than Suspend Configured | Normal mode (program running) |

(1). When the CDC device (RSK/RSSK) detaches from the USB host (VBUS OFF), the APL performs processing to transition the MCU to software standby mode. Recovery from software standby mode occurs when the CDC device (RSK/RSSK) attaches to the USB host.

(2). When a suspend signal sent by the USB host is received while the CDC device (RSK/RSSK) is connected to the USB host, the APL performs processing to transition the MCU to sleep mode. Note that recovery from sleep mode occurs when a resume signal is received from the USB host.

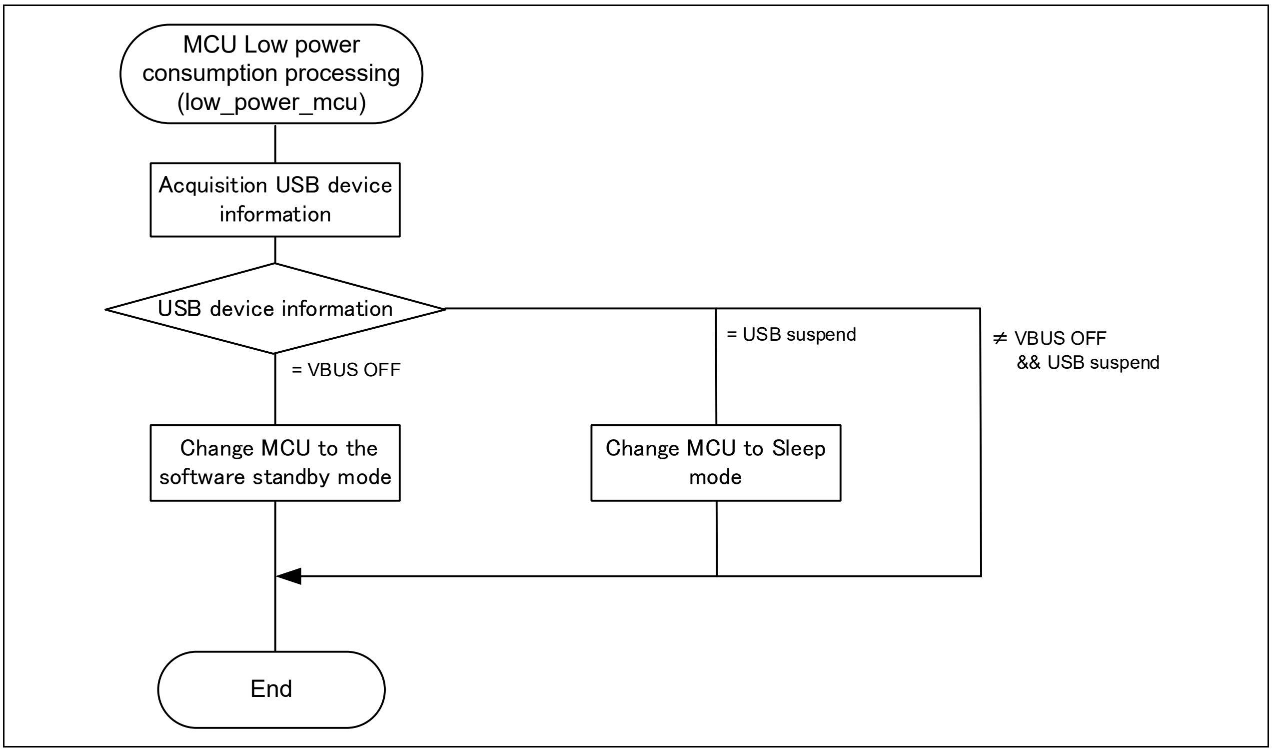



**Figure 4-5 Flowchart of MCU Low Power Consumption Processsing**

## 2. RTOS (FreeRTOS only)

**Table 4-2 Conditions for Transition to Low-Power Mode**

| Transition Condition | | Transition Status |
|---|---|---|
| VBUS | USB State | |
| OFF | — | Software standby mode |
| ON | Suspend Configured | Software standby mode |
| ON | Other than Suspend Configured | Normal mode (program running) |

(1). When the CDC device (RSK) detaches from the USB host (VBUS OFF), the APL performs processing to transition the MCU to software standby mode. Recovery from software standby mode occurs when the CDC device (RSK) attaches to the USB host.

(2). When a suspend signal sent by the USB host is received while the CDC device (RSK) is connected to the USB host, the APL performs processing to transition the MCU to software standby mode. Note that recovery from softwre standby mode occurs when a resume signal is received from the USB host.

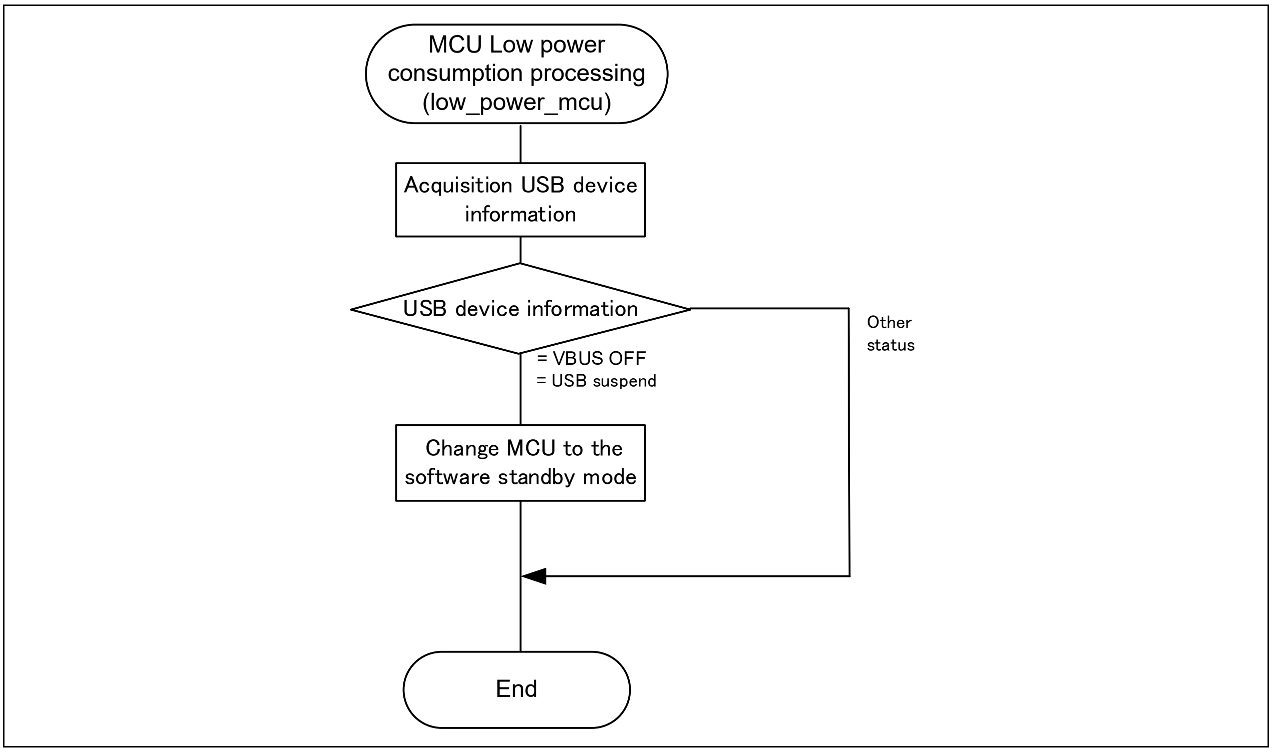



**Figure 4-6 Flowchart of MCU Low Power Consumption Processsing**

## 4.5 Configuration File for the application program (r_usb_pcdc_apl_config.h)

Make settings for the definitions listed below.

1. OPERATION_MODE Definition

   Specify one of the following settings for the *OPERATION_MODE* definition.

   ```
   #define    OPERATION_MODE    USB_ECHO    // Echo mode
   #define    OPERATION_MODE    USB_UART    // USB-serial conversion mode
   ```

2. Low-Power Function Definition

   Specify whether or not the low-power function will be used. If the low-power function will be used, speify *USB_APL_ENABLE* to *USB_SUPPORT_LPW* definition.

   ```
   #define    USE_SUPPORT_LPW    USB_APL_DISABLE    // No use the low-power function
   #define    USB_SUPPORT_LPW    USB_APL_ENABLE     // Use the low-power function
   ```

3. USB_SUPPORT_RTOS Definition

   Please specify *USB_APL_ENABLE* to *USB_SUPPORT_RTOS* definition when using the real-time OS.

   ```
   #define    USB_SUPPORT_RTOS    USB_APL_DISABLE    // No use the real-time OS
   #define    USB_SUPPORT_RTOS    USB_APL_ENABLE     // Use the real-time OS
   ```

4. Note

   The above configuration settings apply to the application program. USB driver configuration settings (*r_usb_basic_mini_config.h*) are required in addition to the above settings. For information on USB driver configuration settings, refer to the application note *USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) using Firmware Integration Technology* (Document number. R01AN2166).

## 4.6 Descriptor

The PCDC's descriptor information is contained in *r_usb_pcdc_descriptor.c*. Also, please be sure to use your vendor ID.

# 5. Class Driver Overview

## 5.1 Class Request（Request from Host to Device）

Table 5-1 lists the class requests supported by the PCDC.

**Table 5-1** Supported CDC Class Requests

| Request | Coce | Description |
|---|---|---|
| SetLineCoding | 0x21 | Sets transmission line coding (transmission speed, data length, parity bit, stop bit length) |
| GetLineCoding | 0x22 | Notify transmission line coding state. |
| SetControlLineState | 0x23 | Sets control signals RTS and DTR for transmission line. |

## 5.2 Data Format

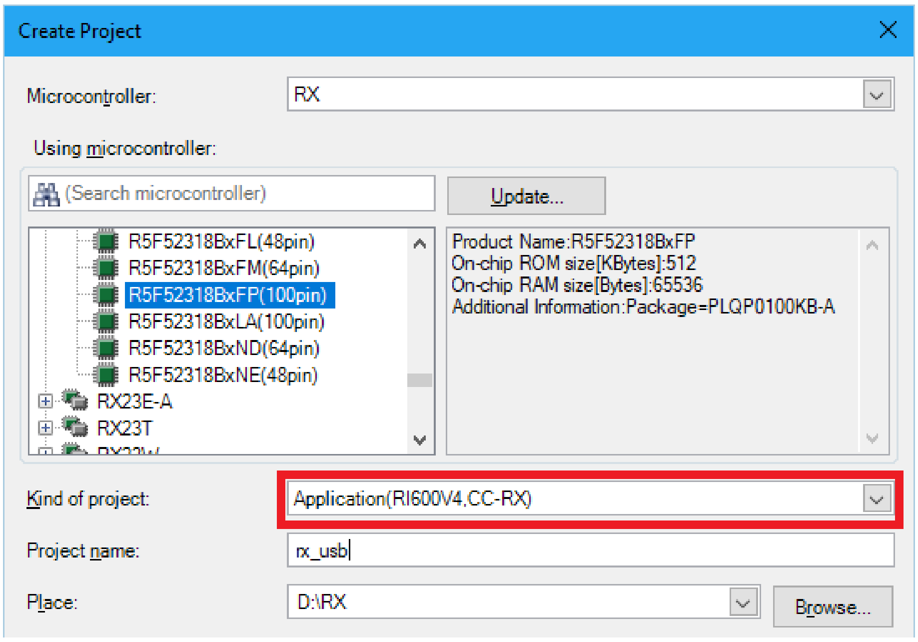
The CDC data class has no specified format. Data may be transferred in any format.

# 6. Using RI600V4 project with CS+

The RI600V4 project in the package does not support CS+. The user needs to create a project for CS+ according to the following procedure when using RI600V4 project on CS+.

## 6.1 New Project Creation

Select "Application(RI600V4, CC-RX) for the Kind of project.

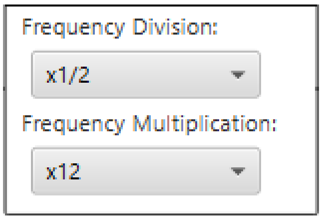



## 6.2 Launch Smart Configurator

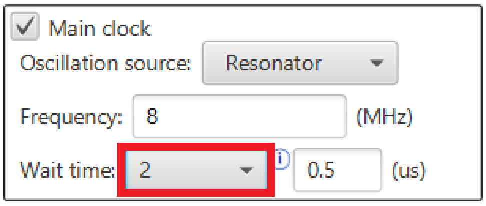
### 1. Clock Setting (Select "Clocks" tab)

(1). Set the related clock so that "48MHz" is set to UCLK (USB clock).

The following is a setting example when using the oscillator(8MHz).




(2). Set the minimum value to the wait time for the main clock.



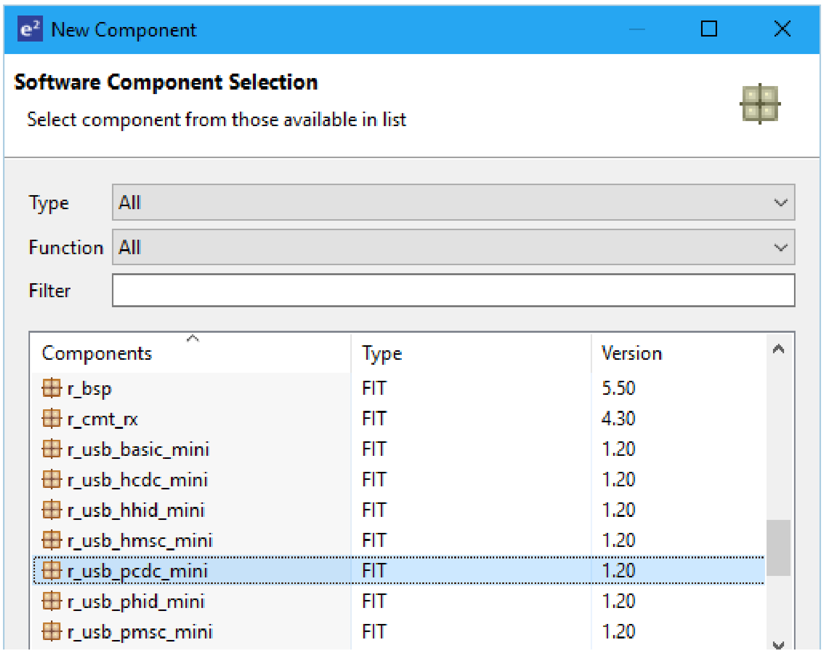

### 2. Component Setting (Select "Components" tab)

(1). Import the USB FIT module

Select the *r_usb_pcdc_mini* module and press the "Finish" button. The *r_usb_basic_mini* module is imported at the same time.

Note:

Select the *r_dtc_rx* / *r_dmaca_rx* module when using the DTC/DMA.

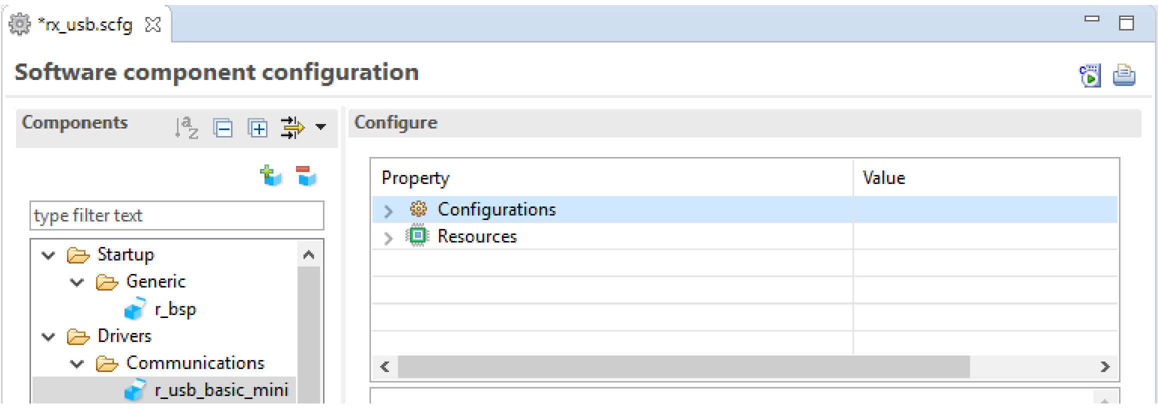



(2). Configuration Setting

a. r_bsp

Change the heap size when using DTC transfer. For the setting value, refer to the documentation for DTC FIT module.

b. r_usb_basic_mini

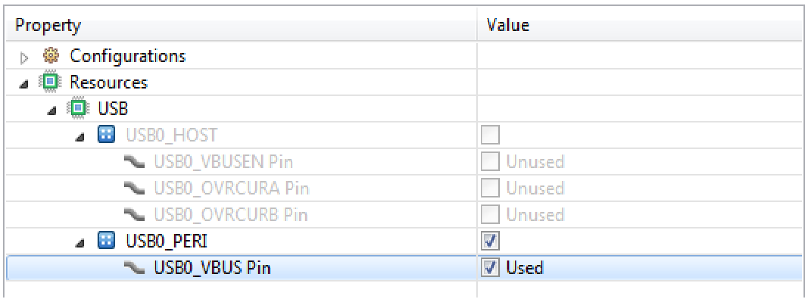



(a). Configurations

Set each item according to the user system.
For the detail of each item, refer to chapter "Configuration" in *USB Basic Mini Host and Peripheral Driver Firmware Integration Technology* application note (Document number: R01AN2166).

(b). Resources

Check the check box for USBx_VBUS pin.

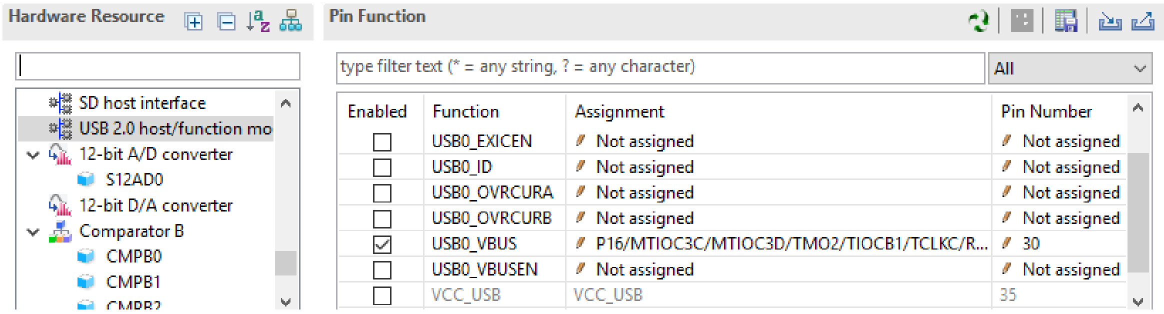



c. r_usb_pcdc_mini

Refer to chapter "Configuration" in *USB Peripheral Communications Devices Class Driver (PCDC) for USB Mini Firmware Firmware Integration Technology* application note (Document number: R01AN2170).

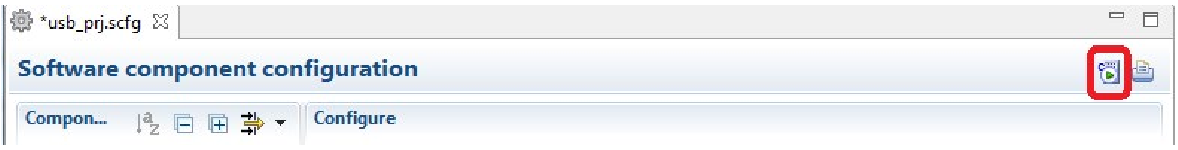
## 3. Pin Setting (Select "Pins" tab)

Select the port for USB pin match the user system.



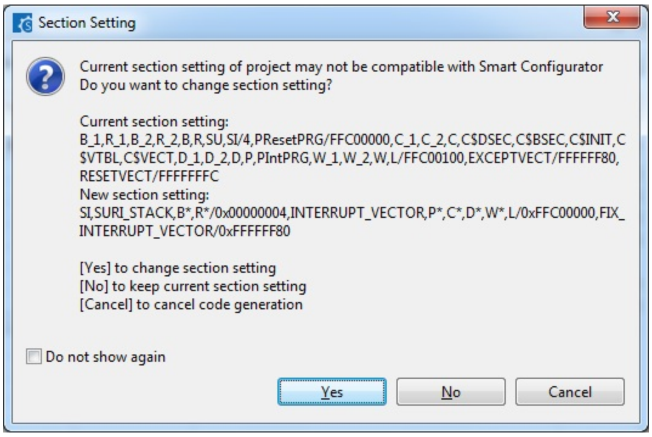
## 4. Generate Code

The Smart Configurator genrates source codes for USB FIT module and USB pin setting in "*<ProjectDir>¥src¥smc_gen*" folder by by clicking on the [ (Generate Code)] button.



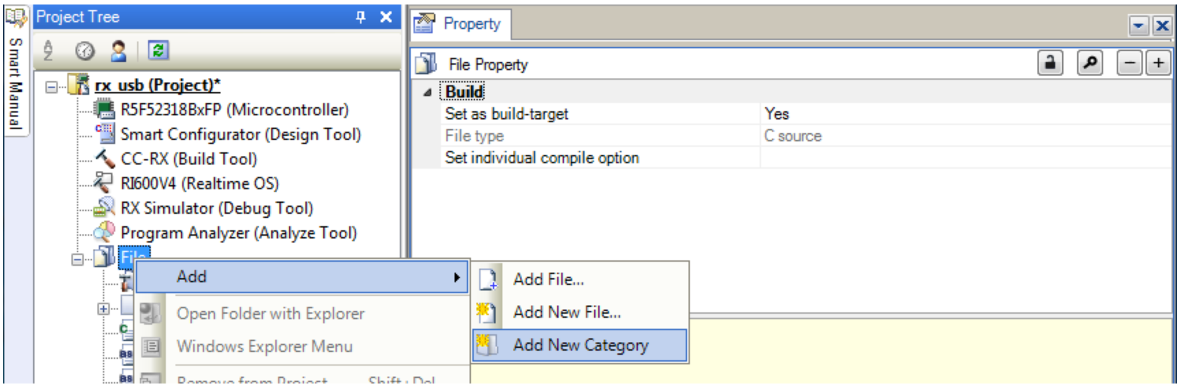

Note:

Select "Yes" if the following dialog box is displayed.



## 6.3 Add the application program and the configuration file

1. Copy the *demo_src* folder in this package to the *"<ProjectDir>¥src"* folder.
2. Copy the RI600V4 configuration file (.cfg file) to *"<ProjectDir>"* folder.
3. Select "File" in the "Project Tree" and click the right button. Select [Add] → [Add New Category] and create the category to store the application program. Then select [Add File] and register the application program and the configuration file which are copied at the above 2.




Note:

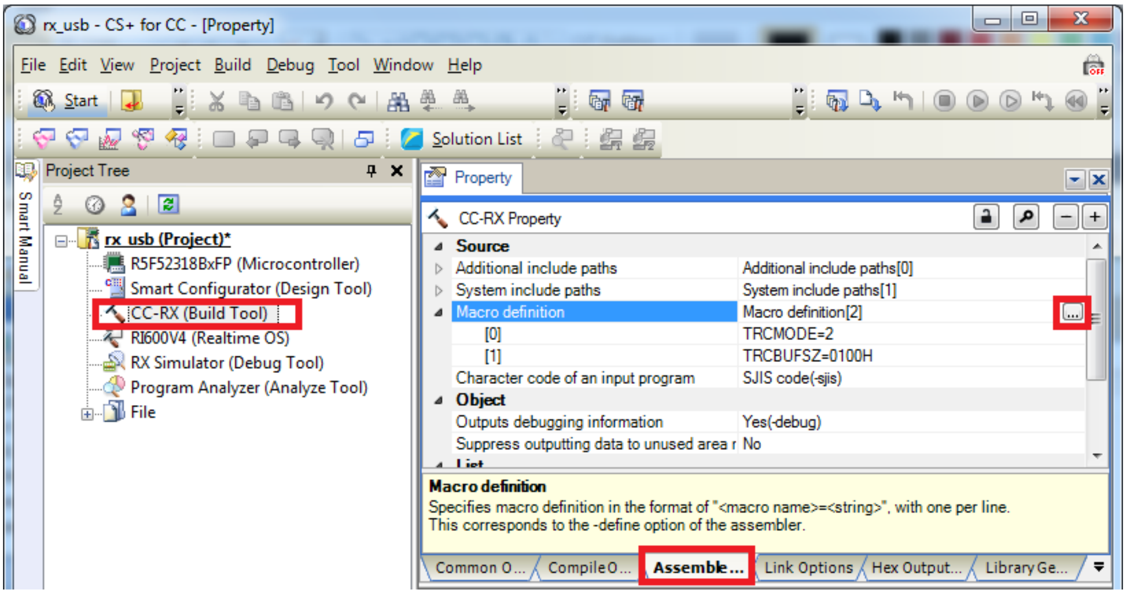
Remove the "task.c" file and "sample.cfg" created in "<*ProjectDir*>" folder by CS+.

## 6.4 Remote Macro Definition

Remove these macros since the following macros is defined in the new created project.

Select [CC-RX(Build Tool)] → [Assemble Options] tab, remove the following macros.

1. TRCMODE = 2
2. TRCBUFSZ = 0100H




## 6.5 Build Execution

Excecute the build and generate the binary target program.

## 7. Using the e² studio project with CS+

The PCDC contains a project only for e² studio. When you use the PCDC with CS+, import the project to CS+ by following procedures.

[Note]

1. Uncheck the checkbox Backup the project composition files after conversion in Project Convert Settings window.
2. The following method is not supported when using RI600V4. Refer to chapter **6, Using RI600V4 project with CS+**.

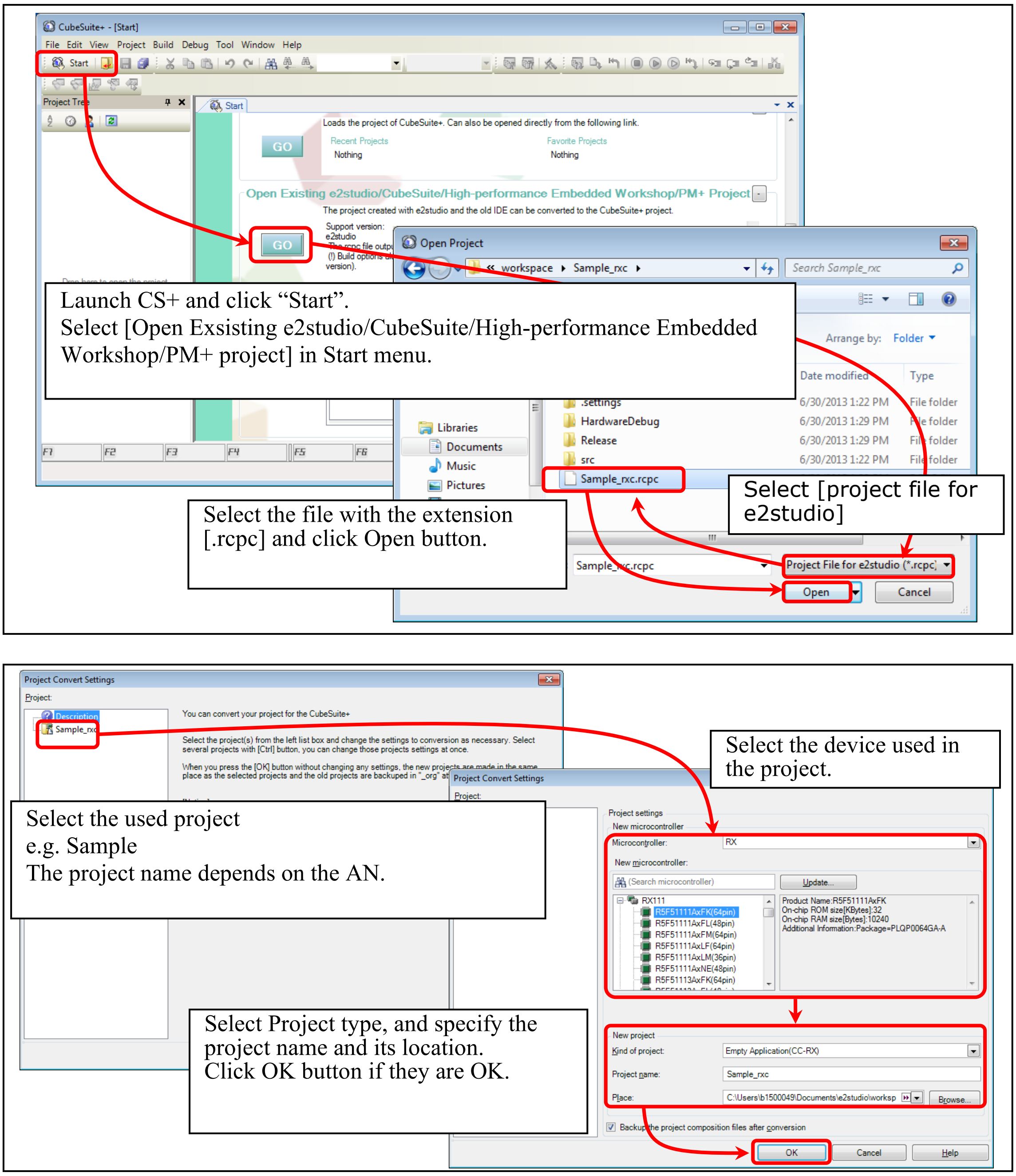



**Figure 7-1 Using the e² studio project with CS+**

## Website and Support

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

Inquiries
http://www.renesas.com/inquiry/

## Revision Record

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Dec 1, 2014 | — | First Edition Issued. |
| 1.01 | Jun 1, 2015 | — | RX231 is added in Target Device. |
| 1.02 | Dec 28, 2015 | — | Checked the operation with Windows® 10. |
| 1.10 | Nov 30, 2018 | — | 1. The following chapter has beed added.<br>(1). 3.1.2 RSK/RSSK Setting<br>2. The following chapter has beed changed.<br>(1). 4. Sample Application |
| 1.12 | Jun 30, 2019 | — | RX23W is added in Target Device. |
| 1.20 | Jun 1, 2020 | — | Supported the real-time OS. |
| 1.30 | Jul 31, 2024 | — | RX261 is added in Target Device. |
| | | | |
| | | | |
| | | | |

# General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit 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. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. 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 the 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.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of 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.

## Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

## Trademarks



## Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.