RX210, RX21A, and RX220 Groups

Transition to Low Power Consumption Modes Using the Low Power Consumption Function

Abstract

This document describes transition to low power consumption modes using the low power consumption function in the RX210, RX21A, and RX220 Groups.

Products

- RX210, RX21A, and RX220 Groups

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.
RX210, RX21A, and RX220 Groups

Transition to Low Power Consumption Modes
Using the Low Power Consumption Function

Contents

1. Specifications ..................................................................................................................................... 3
2. Operation Confirmation Conditions .................................................................................................... 5
3. Reference Application Note ................................................................................................................ 5
4. Hardware ............................................................................................................................................. 6
4.1 Pins Used ..................................................................................................................................... 6
5. Software ............................................................................................................................................. 7
5.1 Operation Overview ..................................................................................................................... 8
  5.1.1 Sleep Mode ........................................................................................................................... 8
  5.1.2 Software Standby Mode ....................................................................................................... 8
  5.1.3 Deep Software Standby Mode .............................................................................................. 9
  5.1.4 Entering and Exiting a Low Power Consumption Mode Using the IRQ ................................ 10
  5.1.5 Entering and Exiting a Low Power Consumption Mode Using the RTC ....................... 10
  5.1.6 Entering and Exiting a Low Power Consumption Mode Using the LVD ....................... 11
5.2 File Composition ........................................................................................................................ 11
5.3 Option-Setting Memory .............................................................................................................. 12
5.4 Constants ................................................................................................................................... 12
5.5 Variables .................................................................................................................................... 12
5.6 Functions .................................................................................................................................... 13
5.7 Function Specifications .............................................................................................................. 13
5.8 Flowcharts .................................................................................................................................. 17
  5.8.1 Main Processing ................................................................................................................. 17
  5.8.2 Port Initialization ............................................................................................................... 18
  5.8.3 Peripheral Function Initialization ....................................................................................... 18
  5.8.4 Transition to Sleep Mode .................................................................................................... 19
  5.8.5 Transition to Software Standby Mode ................................................................................ 20
  5.8.6 Transition to Deep Software Standby Mode ....................................................................... 21
  5.8.7 IRQ Initialization .................................................................................................................. 24
  5.8.8 LVD Initialization ................................................................................................................. 25
  5.8.9 RTC Initialization ................................................................................................................. 27
  5.8.10 IRQ1 Interrupt Handling ..................................................................................................... 28
  5.8.11 LVD1 Interrupt Handling .................................................................................................. 28
  5.8.12 LVD2 Interrupt Handling .................................................................................................. 29
  5.8.13 RTC.PRD Interrupt Handling .......................................................................................... 29
6. Applying This Application Note to the RX21A or RX220 Group ....................................................... 30
7. Sample Code ..................................................................................................................................... 31
8. Reference Documents ...................................................................................................................... 31
1. Specifications

The sample code performs processing to enter and exit a low power consumption mode by specifying a source for transition and exit. The MCU enters and exits a low power consumption mode when the specified source occurs.

- Low power consumption modes:
  Selectable from sleep mode, software standby mode, and deep software standby mode
- Source to enter or exit low power consumption mode: Selectable from IRQ1, LVD, and RTC
- Clock source: Selectable (1)
- Operating power control mode: Selectable (1)
- Sleep mode return clock source switching function: Not used

Note:
1. The clock source and operating power control mode are selected in r_init_clock.h. This application note uses HOCO as the clock source and middle-speed operating mode 1A for the operating power control mode. Refer to RX210 Group Initial Setting Rev. 2.00 application note for details.

Table 1.1 lists the Peripheral Functions and Their Applications and Figure 1.1 shows the Block Diagram.

### Table 1.1   Peripheral Functions and Their Applications

<table>
<thead>
<tr>
<th>Peripheral Function</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low power consumption function</td>
<td>Reduces power consumption.</td>
</tr>
<tr>
<td>External pin interrupt (IRQ)</td>
<td>Enters low power consumption mode.</td>
</tr>
<tr>
<td></td>
<td>Exits low power consumption mode.</td>
</tr>
<tr>
<td>Voltage detection circuit (LVD)</td>
<td>Enters low power consumption mode.</td>
</tr>
<tr>
<td></td>
<td>Exits low power consumption mode.</td>
</tr>
<tr>
<td>Realtime clock (RTC)</td>
<td>Enters low power consumption mode.</td>
</tr>
<tr>
<td></td>
<td>Exits low power consumption mode.</td>
</tr>
</tbody>
</table>
Using the Low Power Consumption Function

Figure 1.1   Block Diagram
2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

<table>
<thead>
<tr>
<th>Item</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU used</td>
<td>R5F52108ADFP (RX210 Group)</td>
</tr>
<tr>
<td>Operating frequencies</td>
<td>- HOCO: 32 MHz&lt;br&gt;- Sub-clock: 32.768 kHz&lt;br&gt;- System clock (ICLK): 32 MHz (HOCO divided by 1)&lt;sup&gt;(1)&lt;/sup&gt;&lt;br&gt;- Peripheral module clock B (PCLKB): 16 MHz (HOCO divided by 2)</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>1.8 to 5.0 V &lt;sup&gt;(1)&lt;/sup&gt;</td>
</tr>
<tr>
<td>Integrated development</td>
<td>Renesas Electronics Corporation&lt;br&gt;High-performance Embedded Workshop Version 4.09.01</td>
</tr>
<tr>
<td>environment</td>
<td></td>
</tr>
<tr>
<td>C compiler</td>
<td>Renesas Electronics Corporation&lt;br&gt;C/C++ Compiler Package for RX Family V.1.02 Release 01</td>
</tr>
<tr>
<td></td>
<td>Compile options&lt;br&gt;-cpu=rx200 -output=obj=&quot;$(CONFIGDIR)/$(FILELEAF).obj&quot; -debug -nologo&lt;br&gt;(The default setting is used in the integrated development environment.)</td>
</tr>
<tr>
<td>iodefine.h version</td>
<td>Version 1.2A</td>
</tr>
<tr>
<td>Endian</td>
<td>Little endian</td>
</tr>
<tr>
<td>Operating mode</td>
<td>Single-chip mode</td>
</tr>
<tr>
<td>Processor mode</td>
<td>Supervisor mode</td>
</tr>
<tr>
<td>Sample code version</td>
<td>Version 1.00</td>
</tr>
<tr>
<td>Board used</td>
<td>Renesas Starter Kit for RX210 (product part no.: R0K505210C000BE)</td>
</tr>
</tbody>
</table>

Note:
1. In this application note, middle-speed operating mode 1A is selected for the operating power control mode. Therefore when the operating voltage used is less than 1.8 V, set ICLK to less than or equal to 20 MHz.

3. Reference Application Notes

For additional information associated with this document, refer to the following application notes.
- RX210 Group Initial Setting Rev. 2.00 (R01AN1002EJ)
- RX21A Group Initial Setting Rev. 1.10 (R01AN1486EJ)
- RX220 Group Initial Setting Rev. 1.10 (R01AN1494EJ)

The initial setting functions in the reference application notes are used in the sample code in this application note. The revision numbers of the reference application notes are current as of when this application note was made. However the latest version is always recommended. Visit the Renesas Electronics Corporation website to check and download the latest version.
4. Hardware

4.1 Pins Used

Table 4.1 lists the Pins Used and Their Functions.

The pins described here are for 100-pin products. When the product with less than 100-pin is used, select appropriate pins for the product used.

Table 4.1 Pins Used and Their Functions

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P31/IRQ1</td>
<td>Input</td>
<td>SW1 input (for entering or exiting low power consumption mode)</td>
</tr>
<tr>
<td>P14</td>
<td>Output</td>
<td>LED0 output (turned on after the initial setting has been performed)</td>
</tr>
<tr>
<td>P15</td>
<td>Output</td>
<td>LED1 output (turned on before entering low power consumption mode)</td>
</tr>
<tr>
<td>P16</td>
<td>Output</td>
<td>LED2 output (turned on after exiting low power consumption mode)</td>
</tr>
</tbody>
</table>
5. Software

When the source for entering a low power consumption mode occurs, the MCU enters a low power consumption mode. When in a low power consumption mode, if the source for exit occurs, the MCU exits the mode it is in.

Sleep mode, software standby mode, or deep software standby mode can be selected as the low power consumption mode. The IRQ, LVD, or RTC can be used for the source to enter or exit a low power consumption mode.

Settings for the peripheral functions are as follows:

**IRQ**
- Detection method: Falling edge on the IRQ1 pin
- Digital filter: Disabled
- Interrupt priority level: Level 15

**LVD**
- Condition for LVD1 detection: VCC passed upward through Vdet1 (4.15 V)
- Condition for LVD2 detection: VCC passed downward through Vdet2 (3.40 V)
- Processing when LVD1 is detected: Voltage monitoring 1 interrupt (maskable)
- Processing when LVD2 is detected: Voltage monitoring 2 interrupt (maskable)
- Digital filter: Disabled
- Interrupt priority level: Level 15

**RTC**
- Initial date and time setting: 2013-01-01 (Tue.) 00:00:00
- Time mode: 24-hour mode
- Interrupt: Periodic interrupt (PRD) is used and generated every 2 seconds.
- Interrupt priority level: Level 15
5.1 Operation Overview

5.1.1 Sleep Mode
After the initial setting has been performed, turns on LED0 and waits until the source for transition to sleep mode occurs. When the source occurs, turns off LED0, turns on LED1, and enters sleep mode. When the source for exit occurs during sleep mode, exits sleep mode, turns off LED1, and turns on LED2.

Figure 5.1 shows the Operation Overview when Entering and Exiting Sleep Mode.

5.1.2 Software Standby Mode
After the initial setting has been performed, turns on LED0 and waits until the source for transition to software standby mode occurs. When the source occurs, turns off LED0, turns on LED1, and enters software standby mode. When the source for exit occurs during software standby mode, exits software standby mode, turns off LED1, and turns on LED2.

Figure 5.2 shows the Operation Overview when Entering and Exiting Software Standby Mode.
5.1.3 Deep Software Standby Mode

After the initial setting has been performed, turns on LED0 and waits until the source for transition to deep software standby mode occurs. When the source occurs, turns off LED0, turns on LED1, and enters deep software standby mode. When the source for exit occurs during deep software standby mode, exits deep software standby mode, and performs a reset. After the reset, performs the initial setting and turns on LED2.

Figure 5.3 shows the Operation Overview when Entering and Exiting Deep Software Standby Mode.

---

Figure 5.3  Operation Overview when Entering and Exiting Deep Software Standby Mode

Note:
1. LED1 is turned on only when the reset is executed due to exiting deep software standby mode (deep software standby reset).
5.1.4 Entering and Exiting a Low Power Consumption Mode Using the IRQ

When using the IRQ for the source to enter and exit a low power consumption mode, the MCU enters or exits a low power consumption mode by the IRQ interrupt request generation.

The IRQ interrupt request generated during a wait period for transition to a low power consumption mode becomes the source for transition, and the IRQ interrupt request generated during the low power consumption mode becomes the source for exit.

Figure 5.4 shows the Timing for Entering and Exiting a Low Power Consumption Mode Using the IRQ.

5.1.5 Entering and Exiting a Low Power Consumption Mode Using the RTC

When using the RTC for the source to enter and exit a low power consumption mode, the MCU enters or exits a low power consumption mode by the RTC.PRD interrupt request generated every 2 seconds.

The RTC.PRD interrupt request generated during a wait period for transition to a low power consumption mode becomes the source for transition, and the RTC.PRD interrupt request generated during the low power consumption mode becomes the source for exit.

Figure 5.5 shows the Timing of Entering and Exiting a Low Power Consumption Mode Using the RTC.
5.1.6 Entering and Exiting a Low Power Consumption Mode Using the LVD

When using the LVD for the source to enter and exit a low power consumption mode, the MCU enters a low power consumption mode by the LVD2 interrupt request generation and exits the low power consumption mode by the LVD1 interrupt request generation. The LVD2 interrupt request is generated when ‘VCC < Vdet2’ is detected. The LVD1 interrupt request is generated when ‘VCC ≥ Vdet1’ is detected.

Figure 5.6 shows the Timing of Entering and Exiting a Low Power Consumption Mode Using the LVD.

![Figure 5.6 Timing of Entering and Exiting a Low Power Consumption Mode Using the LVD](image)

5.2 File Composition

Table 5.1 lists the Files Used in the Sample Code. Files generated by the integrated development environment are not included in this table.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Outline</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.c</td>
<td>Main processing</td>
<td></td>
</tr>
<tr>
<td>r_init_stop_module.c</td>
<td>Stop processing for active peripheral functions after a reset</td>
<td></td>
</tr>
<tr>
<td>r_init_stop_module.h</td>
<td>Header file for r_init_stop_module.c</td>
<td></td>
</tr>
<tr>
<td>r_init_non_existent_port.c</td>
<td>Nonexistent port initialization</td>
<td></td>
</tr>
<tr>
<td>r_init_non_existent_port.h</td>
<td>Header file for r_init_non_existent_port.c</td>
<td></td>
</tr>
<tr>
<td>r_init_clock.c</td>
<td>Clock initialization</td>
<td></td>
</tr>
<tr>
<td>r_init_clock.h</td>
<td>Header file for r_init_clock.c</td>
<td></td>
</tr>
</tbody>
</table>
5.3 Option-Setting Memory

Table 5.2 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

Table 5.2 Option-Setting Memory Configured in the Sample Code

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFS0</td>
<td>FFFF FF8Fh to FFFF FF8Ch</td>
<td>FFFF FFFFh</td>
<td>The IWDT is stopped after a reset. The WDT is stopped after a reset.</td>
</tr>
<tr>
<td>OFS1</td>
<td>FFFF FF88h to FFFF FF88h</td>
<td>FFFF FFFFh</td>
<td>The voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.</td>
</tr>
<tr>
<td>MDES</td>
<td>FFFF FF83h to FFFF FF80h</td>
<td>FFFF FFFFh</td>
<td>Little endian</td>
</tr>
</tbody>
</table>

5.4 Constants

Table 5.3 lists the Constants Used in the Sample Code.

Table 5.3 Constants Used in the Sample Code

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>L_IRQ</td>
<td>0</td>
<td>Source to enter and exit a low power consumption mode: IRQ</td>
</tr>
<tr>
<td>L_LVD</td>
<td>1</td>
<td>Source to enter and exit a low power consumption mode: LVD</td>
</tr>
<tr>
<td>L_RTC</td>
<td>2</td>
<td>Source to enter and exit a low power consumption mode: RTC</td>
</tr>
<tr>
<td>L_SOURCE</td>
<td>L_IRQ</td>
<td>Selection of the source to enter and exit a low power consumption mode: IRQ</td>
</tr>
<tr>
<td>L_SLEEP</td>
<td>0</td>
<td>Low power consumption mode: Sleep mode</td>
</tr>
<tr>
<td>L_SOFT_STANDBY</td>
<td>1</td>
<td>Low power consumption mode: Software standby mode</td>
</tr>
<tr>
<td>L_DEEP_STANDBY</td>
<td>2</td>
<td>Low power consumption mode: Deep software standby mode</td>
</tr>
<tr>
<td>L_MODE</td>
<td>L_SLEEP</td>
<td>Selection of low power consumption mode: Sleep mode</td>
</tr>
<tr>
<td>WAIT_tdEA</td>
<td>480</td>
<td>td(E-A) wait time (max. 15 µs) Wait time ÷ ICLK (32 MHz) cycles = 15 ÷ 0.03125 = 480</td>
</tr>
</tbody>
</table>

5.5 Variables

Table 5.4 lists the Global Variables.

Table 5.4 Global Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td>initial_end</td>
<td>Initial setting end flag 0: Processing 1: Completed</td>
<td>sleep_mode software_standby_mode deep_standby_mode Excep_ICU_IRQ1 Excep_LVD_LVD2 Excep_RTC_PRD</td>
</tr>
<tr>
<td>uint8_t</td>
<td>enable_low_power</td>
<td>Enable flag for transition to a low power consumption mode 0: Transition disabled 1: Transition enabled</td>
<td>sleep_mode software_standby_mode deep_standby_mode Excep_ICU_IRQ1 Excep_LVD_LVD2 Excep_RTC_PRD</td>
</tr>
</tbody>
</table>
5.6 Functions

Table 5.5 lists the Functions Used in the Sample Code.

Table 5.5 Functions Used in the Sample Code

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>Main processing</td>
</tr>
<tr>
<td>port_init</td>
<td>Port initialization</td>
</tr>
<tr>
<td>R_INIT_StopModule</td>
<td>Stop processing for active peripheral functions after a reset</td>
</tr>
<tr>
<td>R_INIT_NonExistentPort</td>
<td>Nonexistent port initialization</td>
</tr>
<tr>
<td>R_INIT_Clock</td>
<td>Clock initialization</td>
</tr>
<tr>
<td>peripheral_init</td>
<td>Peripheral function initialization</td>
</tr>
<tr>
<td>sleep_mode</td>
<td>Transition to sleep mode</td>
</tr>
<tr>
<td>software_standby_mode</td>
<td>Transition to software standby mode</td>
</tr>
<tr>
<td>deep_standby_mode</td>
<td>Transition to deep software standby mode</td>
</tr>
<tr>
<td>irq_init</td>
<td>IRQ initialization</td>
</tr>
<tr>
<td>lvd_init</td>
<td>LVD initialization</td>
</tr>
<tr>
<td>rtc_init</td>
<td>RTC initialization</td>
</tr>
<tr>
<td>Excep_ICU_IRQ1</td>
<td>IRQ1 interrupt handling</td>
</tr>
<tr>
<td>Excep_LVD_LVD1</td>
<td>LVD1 interrupt handling</td>
</tr>
<tr>
<td>Excep_LVD_LVD2</td>
<td>LVD2 interrupt handling</td>
</tr>
<tr>
<td>Excep_RTC_PRD</td>
<td>RTC.PRD interrupt handling</td>
</tr>
</tbody>
</table>

5.7 Function Specifications

The following tables list the sample code function specifications.

| main                           | Outline: Main processing
|--------------------------------| Header: None
| Declaration: void main(void)  | Description: Enters a low power consumption mode after the initial setting has been performed.
| Arguments: None                | Return Value: None

| port_init                     | Outline: Port initialization
|--------------------------------| Header: None
| Declaration: void port_init(void) | Description: Initializes ports.
| Arguments: None                | Return Value: None
### R_INIT_StopModule

**Outline**
Stop processing for active peripheral functions after a reset

**Header**
r_init_stop_module.h

**Declaration**
void R_INIT_StopModule(void)

**Description**
Configures the setting to enter the module-stop state.

**Arguments**
None

**Return Value**
None

**Remarks**
Transition to the module-stop state is not performed in the sample code. For details on this function, refer to the Initial Setting application note for the product used.

### R_INIT_NonExistentPort

**Outline**
Nonexistent port initialization

**Header**
r_init_non_existent_port.h

**Declaration**
void R_INIT_NonExistentPort(void)

**Description**
Initializes port direction registers for ports that do not exist in products with less than 100 pins.

**Arguments**
None

**Return Value**
None

**Remarks**
The number of pins in the sample code is set for the 100-pin package (PIN_SIZE=100). After this function is called, when writing in byte units to the PDR registers or PODR registers which have nonexistent ports, set the corresponding bits for nonexistent ports as follows: set the I/O select bits in the PDR registers to 1 and set the output data store bits in the PODR registers to 0.

For details on this function, refer to the Initial Setting application note for the product used.

### R_INIT_Clock

**Outline**
Clock initialization

**Header**
r_init_clock.h

**Declaration**
void R_INIT_Clock(void)

**Description**
Initializes the clock.

**Arguments**
None

**Return Value**
None

**Remarks**
The sample code selects processing with the following settings:
- System clock: HOCO
- Operating power control mode: Middle-speed operating mode 1A
- Main clock and PLL: Not used

For details on this function, refer to the Initial Setting application note for the product used.

### peripheral_init

**Outline**
Peripheral function initialization

**Header**
None

**Declaration**
void peripheral_init(void)

**Description**
Initializes peripheral functions used.

**Arguments**
None

**Return Value**
None
### sleep_mode

<table>
<thead>
<tr>
<th>Outline</th>
<th>Transition to sleep mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void sleep_mode(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Configures settings to enter sleep mode.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### software_standby_mode

<table>
<thead>
<tr>
<th>Outline</th>
<th>Transition to software standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void software_standby_mode(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Configures settings to enter software standby mode.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### deep_standby_mode

<table>
<thead>
<tr>
<th>Outline</th>
<th>Transition to deep software standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void deep_standby_mode(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Configures settings to enter deep software standby mode.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### irq_init

<table>
<thead>
<tr>
<th>Outline</th>
<th>IRQ initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void irq_init(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Performs the IRQ initialization.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### lvd_init

<table>
<thead>
<tr>
<th>Outline</th>
<th>LVD initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void lvd_init(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Performs the LVD initialization.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### rtc_init

<table>
<thead>
<tr>
<th>Outline</th>
<th>RTC initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void rtc_init(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Performs the RTC initialization.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>
### Excep_ICU_IRQ1

<table>
<thead>
<tr>
<th>Outline</th>
<th>IRQ1 interrupt handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void Excep_ICU_IRQ1(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Performs the IRQ1 interrupt handling.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### Excep_LVD_LVD1

<table>
<thead>
<tr>
<th>Outline</th>
<th>LVD1 interrupt handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void Excep_LVD_LVD1(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Performs the LVD1 interrupt handling.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### Excep_LVD_LVD2

<table>
<thead>
<tr>
<th>Outline</th>
<th>LVD2 interrupt handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void Excep_LVD_LVD2(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Performs the LVD2 interrupt handling.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### Excep_RTC_PRD

<table>
<thead>
<tr>
<th>Outline</th>
<th>RTC.PRD interrupt handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void Excep_RTC_PRD(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Performs the RTC.PRD interrupt handling.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>
5.8 Flowcharts

5.8.1 Main Processing

Figure 5.7 shows the Main Processing.

```
main

Disable maskable interrupts  \( I \) flag &lt;= 0

Port initialization
  \texttt{port\_init()}

Stop processing for active peripheral functions after a reset
  \texttt{R\_INIT\_StopModule()}

Nonexistent port initialization
  \texttt{R\_INIT\_NonExistentPort()}

Clock Initialization
  \texttt{R\_INIT\_Clock()}

Peripheral function initialization
  \texttt{peripheral\_init()}

Enable maskable interrupts  \( I \) flag &lt;= 1

\textbf{When L\_SLEEP (sleep mode) is set to L\_MODE (low power consumption mode):}

  Transition to sleep mode
    \texttt{sleep\_mode()}

\textbf{When L\_SOFT\_STANDBY (software standby mode) is set to L\_MODE:}

  Transition to software standby mode
    \texttt{software\_standby\_mode()}

\textbf{When L\_DEEP\_STANDBY (deep software standby mode) is set to L\_MODE:}

  Transition to deep software standby mode
    \texttt{deep\_standby\_mode()}
```

Figure 5.7 Main Processing
5.8.2 Port Initialization
Figure 5.8 shows the Port Initialization.

```
port_init

Turn off LEDs

PORT1.PODR register
B4 bit ← 1: LED0: High
B5 bit ← 1: LED1: High
B6 bit ← 1: LED2: High

PORT1.PDR register
B4 bit ← 1: LED0: Output
B5 bit ← 1: LED1: Output
B6 bit ← 1: LED2: Output

return
```

Figure 5.8 Port Initialization

5.8.3 Peripheral Function Initialization
Figure 5.9 shows the Peripheral Function Initialization.

```
peripheral_init

When L_IRQ (IRQ) is set to the L_SOURCE (source to enter and exit low power consumption mode)

IRQ initialization
irq_init()

When L_LVD (LVD) is set to the L_SOURCE

LVD initialization
lvd_init()

When L_RTC (RTC) is set to the L_SOURCE

RTC initialization
rtc_init()

return
```

Figure 5.9 Peripheral Function Initialization
5.8.4 Transition to Sleep Mode

Figure 5.10 shows the Transition to Sleep Mode.

Figure 5.10 Transition to Sleep Mode

Note:
1. Read the register written immediately before the WAIT instruction is executed to confirm the written value can be read correctly.
5.8.5 Transition to Software Standby Mode

Figure 5.11 shows the Transition to Software Standby Mode.

- **software_standby_mode**

  - **Turn on LED0**
    - PORT1.PODR register
    - B4 bit ← 0: LED0: Low

  - **Set the initial setting end flag**
    - initial_end ← 1

  - **Wait period for transition to software standby mode**
    - I flag ← 0

  - **Disable maskable interrupts**
    - PRCR register ← A503h
    - PRC0 bit = 1: Enables writing to the registers related to the clock generation circuit.
    - PRC1 bit = 1: Enables writing to the registers related to the low power consumption function.

  - **Enable write protection**
    - PRCR register ← A500h
    - PRC0 bit = 0: Enables writing to the registers related to the clock generation circuit.
    - PRC1 bit = 0: Enables writing to the registers related to the low power consumption function.

  - **Set the transition destination after the WAIT instruction is executed**
    - SBYCR register
    - SSBY bit ← 1: Transition to software standby mode is made after the WAIT instruction is executed.
    - DPSBYCR register
    - DPSBY bit ← 0: Transition to software standby mode is made after the WAIT instruction is executed.

  - **Disable the oscillation stop detection function**
    - OSTDCR register
    - OSTDE bit ← 0: Oscillation stop detection function is disabled.

  - **Enable write protection**
    - PRCR register ← A500h
    - PRC0 bit = 1: Enables writing to the registers related to the clock generation circuit.
    - PRC1 bit = 1: Enables writing to the registers related to the low power consumption function.

  - **Disable the DMAC and DTC**
    - DMAST register
    - DMST bit ← 0: DMAC activation is disabled.
    - DTCST register
    - DTCST bit ← 0: DTC module stop

  - **Turn off LED0 and turn on LED1**
    - PORT1.PODR register
    - B4 bit ← 1: LED0: High
    - B5 bit ← 0: LED1: Low

  - **Execute the WAIT instruction**
    - Enters software standby mode.

  - **Turn off LED1 and turn on LED2**
    - PORT1.PODR register
    - B5 bit ← 1: LED1: High
    - B6 bit ← 0: LED2: Low

**Note:**
1. Read the register written immediately before the WAIT instruction is executed to confirm the written value can be read correctly.

---

**Figure 5.11 Transition to Software Standby Mode**
5.8.6 Transition to Deep Software Standby Mode

Figure 5.12 to Figure 5.14 show the Transition to Deep Software Standby Mode.

![Diagram of Transition to Deep Software Standby Mode]

- Reads RSTSR0 register.
- DPSRSTF flag: 0: Deep software standby mode cancellation not requested by an interrupt.
- 1: Deep software standby mode cancellation requested by an interrupt.

- PORT1.PODR register
  - B4 bit ← 0: LED0: Low
  - initial_end ← 1

- DMAST register
  - DMST bit ← 0: DMAC activation is disabled.
  - DTCST bit ← 0: DTC module stop

- Disable the DMAC and DTC
- Disable write protection
- Disable the oscillation stop detection function
- Set the transition destination after the WAIT instruction is executed
- Configure pin states while in deep software standby mode and pin states after exiting
- Turning off LED0 and turning on LED1
- DPSBYCR register
  - IPT bit ← 0: LED0: High
  - B5 bit ← 0: LED1: Low

- SBYCR register
  - OPE bit ← 1: In deep software standby mode, the address bus and bus control signals retain the output state.
  - DPSBYCR register
    - IKEEP bit ← 1: The I/O port state is retained even after deep software standby mode is canceled. Then, writing 0 to the IKEEP bit cancels the I/O port retention.
RX210, RX21A, and RX220 Groups  Transition to Low Power Consumption Modes
Using the Low Power Consumption Function

Figure 5.13  Transition to Deep Software Standby Mode (2/3)

1. After reading registers DPSIFR0 and DPSIFR2, write 00h to these registers.
2. Read the register written immediately before the WAIT instruction is executed to confirm the written value can be read correctly.
Determine the exit request \(^{(1)}\)

Read the DPSIFR0 register
Read the DPSIFR2 register

Turn off LED0 and LED1, and turn on LED2

PORT1.PODR register
B4 bit ← 1: LED0: High
B5 bit ← 1: LED1: High
B6 bit ← 0: LED2: Low

PRCR register ← A500h
PRC1 bit = 1: Enables writing to the registers related to the low power consumption function.

B

Disable write protection

Cancel the I/O port retention

DPSBYCR register
IOKEEP bit ← 0

Enable write protection

PRCR register ← A500h
PRC1 bit = 0: Disables writing to the registers related to the low power consumption function.

Processing according to the exit request \(^{(1)}\)

return

Note:
1. Processing for exit requests is not performed in the sample code. Add a program as required.

Figure 5.14 Transition to Deep Software Standby Mode (3/3)
5.8.7 IRQ Initialization

Figure 5.15 shows the IRQ Initialization.

```
irq_init

Disable the IRQ1 interrupt request

Enable writing to the PFSWE bit

Enable writing to the PFS register

Select the pin function

Disable writing to the PFS register

Disable writing to the PFSWE bit

Set the IRQ1 detection method

Clear the IRQ1 interrupt request

Set the IRQ1 interrupt priority level

Enable the IRQ1 interrupt request

return
```

IER08 register
IEN1 bit ← 0

PWPR register
B0WI bit ← 0

PWPR register
PFSWE bit ← 1

P31PFS register
ISEL bit ← 1: P31: Used as IRQ1 input pin

PWPR register
PFSWE bit ← 0

PWPR register
B0WI bit ← 1

IRQCRI register
IRQMD[1:0] bits ← 01b: Falling edge

I0R65 register
IR flag ← 0

IPR065 register
IPR[3:0] bits ← 1111b: Level 15

IER08 register
IEN1 bit ← 1

Figure 5.15 IRQ Initialization
5.8.8 LVD Initialization

Figure 5.16 and Figure 5.17 show the LVD Initialization.

![LVD Initialization Diagram]

Figure 5.16  LVD Initialization (1/2)
Clear the voltage change detection flags of voltage monitoring 1 and 2

\[ \text{LVD1SR register} \quad \text{LVD1DET flag} \leftarrow 0 \]
\[ \text{LVD2SR register} \quad \text{LVD2DET flag} \leftarrow 0 \]

Disable write protection

\[ \text{PRCR register} \quad \text{PRC3 bit} = 0: \text{Disables writing to registers related to the LVD.} \]

Enable interrupt requests

\[ \text{IER0B register} \quad \text{IEN0 bit} \leftarrow 1: \text{LVD1 interrupt request is enabled.} \]
\[ \text{IEN1 bit} \leftarrow 1: \text{LVD2 interrupt request is enabled.} \]

Set the interrupt priority level

\[ \text{IPR088 register} \quad \text{IPR[3:0] bits} \leftarrow 1111b: \text{LVD1 interrupt priority level is set to level 15.} \]
\[ \text{IPR089 register} \quad \text{IPR[3:0] bits} \leftarrow 1111b: \text{LVD2 interrupt priority level is set to level 15.} \]

Wait for two or more PCLKB cycles

Holds two or more PCLKB cycles by reading the LVD1CR0 register.

Enable outputting comparison results of voltage monitoring 1 and 2 circuits

\[ \text{LVD1CR0 register} \quad \text{LVD1CMPE bit} = 1: \text{Voltage monitoring 1 circuit comparison results output enabled} \]
\[ \text{LVD2CR0 register} \quad \text{LVD2CMPE bit} = 1: \text{Voltage monitoring 2 circuit comparison results output enabled} \]

Clear interrupt requests

\[ \text{IR088 register} \quad \text{IR flag} \leftarrow 0: \text{LVD1 interrupt request is not generated.} \]
\[ \text{IR089 register} \quad \text{IR flag} \leftarrow 0: \text{LVD2 interrupt request is not generated.} \]

Figure 5.17  LVD Initialization (2/2)
5.8.9 RTC Initialization

Figure 5.18 shows the RTC Initialization.

---

**Figure 5.18 RTC Initialization**

```
rtc_init

Disable RTC interrupt requests

IER07 register
IRE7 bit ← 0: RTC.CUP interrupt is disabled.
IER0B register
IRE4 bit ← 0: RTC.ALM interrupt is disabled.
IRE5 bit ← 0: RTC.PRD interrupt is disabled.

Stop RTC counters and prescaler (1)

RCR2 register ← RCR2 register & 7Eh
START bit ← 0: RTC counters and prescaler are stopped.

Set the initial date and time setting to
2013-01-01 (Tue.) 00:00:00

RSECCNT register ← 00h
SEC1[3:0] bits = 0000b: Ones place of seconds: 0
SEC10[2:0] bits = 000b: Tens place of seconds: 0
RMINCNT register ← 00h
MIN1[3:0] bits = 0000b: Ones place of minutes: 0
MIN10[2:0] bits = 000b: Tens place of minutes: 0
RHRCNT register ← 00h
HR[3:0] bits = 0000b: Ones place of hours: 0
HR10[1:0] bits = 00b: Tens place of hours: 0
RWKCNT register ← 02h
DAYW[2:0] bits = 010b: Day-of-week: Tuesday
RDAYCNT register ← 01h
DATE[3:0] bits = 0001b: Ones place of days: 1
DATE10[1:0] bits = 00b: Tens place of days: 0
RMONCNT register ← 01h
MON1[3:0] bits = 0001b: Ones place of months: 1
MON10 bit = 0: Tens place of months: 0
RYRCNT register ← 0013h
YR[3:0] bits = 0011b: Ones place of years: 3
YR10[3:0] bits = 0001b: Tens place of years: 1

RCR2 register
HR24 bit ← 1: The RTC operates in 24-hour mode.

RCR1 register ← 00h
AIE bit = 0: An alarm interrupt request is disabled.
CIE bit = 0: A carry interrupt request is disabled.
PIE bit = 0: A periodic interrupt request is disabled.

IR062 register
IR flag ← 0: No carry interrupt request is generated.
IR092 register
IR flag ← 0: No alarm interrupt request is generated.
IR093 register
IR flag ← 0: No periodic interrupt request is generated.

IPR093 register
IPR[3:0] bits ← 1111b: Interrupt priority level for the RTC.PRD interrupt is set to level 15.

RCR1 register ← F4h
PIE bit = 1: A periodic interrupt request is enabled.
PES[3:0] bits = 1111b: A periodic interrupt is generated every 2 seconds.

IER0B register
IEN5 bit ← 1

RCR2 register
START bit ← 1

Notes:
1. After writing to the RCR1 register and RCR2.START bit, confirm that the written value can be read correctly.
2. After writing to the RCR2.HR24 bit, dummy read the register three times.
```
5.8.10 IRQ1 Interrupt Handling

Figure 5.19 shows the IRQ1 Interrupt Handling.

```
Excep_ICU_IRQ1

Has the initial setting completed?

Yes

Set the enable flag for transition to low power consumption mode

Clear the enable flag for transition to low power consumption mode

No

return
```

Figure 5.19 IRQ1 Interrupt Handling

5.8.11 LVD1 Interrupt Handling

Figure 5.20 shows the LVD1 Interrupt Handling.

```
Excep_LVD_LVD1

PRCR register ← A508h
PRC3 bit = 1: Enable writing to registers related to the LVD.

Disable write protection

LVD1CR0 register ← 06h
LVD1RIE bit = 0

Disable the voltage monitoring 1 interrupt

Clear the voltage monitoring 1 voltage change detection flag

LVD1SR register
LVD1DET flag ← 0

Wait for two or more PCLKB cycles

Holds two or more PCLKB cycles by reading the LVD1CR0 register.

Enable the voltage monitoring 1 interrupt

LVD1CR0 register ← 07h
LVD1RIE bit = 1

Enable write protection

PRCR register ← A500h
PRC3 bit = 0: Disable writing to registers related to the LVD.

return
```

Figure 5.20 LVD1 Interrupt Handling
5.8.12  LVD2 Interrupt Handling

Figure 5.21 shows the LVD2 Interrupt Handling.

![Flowchart of LVD2 Interrupt Handling]

- **Excep_LVD_LVD2**
- **Disable write protection**
  - PRCR register ← A508h
  - PRC3 bit = 1: Enable writing to registers related to the LVD.
- **Disable the voltage monitoring 2 interrupt**
  - LVD2CR0 register ← 06h
  - LVD2RIE bit = 0
- **Clear the voltage monitoring 2 voltage change detection flag**
  - LVD2SR register
  - LVD2DET flag ← 0
- **Wait for two or more PCLKB cycles**
  - Holds two or more PCLKB cycles by reading the LVD2CR0 register.
- **Enable the voltage monitoring 2 interrupt**
  - LVD2CR0 register ← 07h
  - LVD2RIE bit = 1
- **Enable write protection**
  - PRCR register ← A500h
  - PRC3 bit = 0: Disable writing to registers related to the LVD.

Has the initial setting completed?
- **No**
- **Set the enable flag for transition to low power consumption mode**
- **Clear the enable flag for transition to low power consumption mode**

**return**

Figure 5.21  LVD2 Interrupt Handling

5.8.13  RTC.PRD Interrupt Handling

Figure 5.22 shows the RTC.PRD Interrupt Handling.

![Flowchart of RTC.PRD Interrupt Handling]

- **Excep_RTC_PRD**
- **Has the initial setting completed?**
  - **No**
  - **Set the enable flag for transition to low power consumption mode**
  - **Clear the enable flag for transition to low power consumption mode**

**return**

Figure 5.22  RTC.PRD Interrupt Handling
6. Applying This Application Note to the RX21A or RX220 Group

The sample code accompanying this application note has been confirmed to operate with the RX210 Group. To make the sample code operate with the RX21A or RX220 Group, use this application note in conjunction with the Initial Setting application note for each group.

For details on using this application note with the RX21A and RX220 Groups, refer to “5. Applying the RX210 Group Application Note to the RX21A Group” in the RX21A Group Initial Setting application note, and “4. Applying the RX210 Group Application Note to the RX220 Group” in the RX220 Group Initial Setting application note.

Note: • The r_init_clock.h file will be overwritten when applying the Initial Setting for the product used. Make the settings in the overwritten file be same as the original settings in the r_init_clock.h file accompanying this application note.
7. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

8. Reference Documents

User’s Manual: Hardware
- RX210 Group User's Manual: Hardware Rev.1.50 (R01UH0037EJ)
- RX21A Group User's Manual: Hardware Rev.1.00 (R01UH0251EJ)
- RX220 Group User's Manual: Hardware Rev.1.10 (R01UH0292EJ)
  The latest versions can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
  The latest information can be downloaded from the Renesas Electronics website.

User’s Manual: Development Tools
- RX Family C/C++ Compiler Package V.1.01 User’s Manual Rev.1.00 (R20UT0570EJ)
  The latest version can be downloaded from the Renesas Electronics website.

Website and Support

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

Inquiries
  http://www.renesas.com/contact/
## REVISION HISTORY

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Aug. 1, 2013</td>
<td>—</td>
<td>First edition issued</td>
</tr>
<tr>
<td>1.01</td>
<td>July 1, 2014</td>
<td>1</td>
<td>Products: Added the RX21A and RX220 Groups.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>3. Reference Application Notes: Added the Initial Setting application notes for the RX21A and RX220 Groups.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>Modified the description of reference application note in the following functions: R_INIT_StopModule, R_INIT_NonExistentPort, and R_INIT_Clock.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>6. Applying This Application Note to the RX21A or RX220 Group: Added.</td>
</tr>
</tbody>
</table>

All trademarks and registered trademarks are the property of their respective owners.
# 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.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Handling of Unused Pins</td>
<td></td>
</tr>
</tbody>
</table>
Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.  
- The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual. |
| 2. Processing at Power-on |   
The state of the product is undefined at the moment when power is supplied.  
- The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.  
In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.  
In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified. |
| 3. Prohibition of Access to Reserved Addresses |   
Access to reserved addresses is prohibited.  
- The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed. |
| 4. Clock Signals |   
After applying a reset, only release the reset line after the operating clock signal has become stable.  
When switching the clock signal during program execution, wait until the target clock signal has stabilized.  
- When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal.  
Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable. |
| 5. Differences between Products |   
Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.  
- The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product. |