Abstract

This document describes how to obtain the current time information while intermittently exiting software standby mode using the real-time clock (RTC) in the RX63N Group and RX631 Group.

Products

RX63N Group, 176-Pin and 177-Pin Packages, ROM Capacities: 768 Kbytes to 2 Mbytes
RX63N Group, 144-Pin and 145-Pin Packages, ROM Capacities: 768 Kbytes to 2 Mbytes
RX63N Group, 100-Pin Package, ROM Capacities: 768 Kbytes to 2 Mbytes
RX631 Group, 176-Pin and 177-Pin Packages, ROM Capacities: 256 Kbytes to 2 Mbytes
RX631 Group, 144-Pin and 145-Pin Packages, ROM Capacities: 256 Kbytes to 2 Mbytes
RX631 Group, 100-Pin Package, ROM Capacities: 256 Kbytes to 2 Mbytes

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.
## Contents

1. Specifications ..................................................................................................................................... 3
2. Operation Confirmation Conditions .................................................................................................... 4
3. Reference Application Note ................................................................................................................ 4
4. Hardware ............................................................................................................................................ 5
   4.1 Hardware Configuration ............................................................................................................... 5
   4.2 Pins Used ..................................................................................................................................... 5
4. Software ............................................................................................................................................. 6
   5.1 Operation Overview ..................................................................................................................... 7
   5.2 File Composition .......................................................................................................................... 8
   5.3 Option-Setting Memory ................................................................................................................ 8
   5.4 Constants ..................................................................................................................................... 8
   5.5 Structure/Union List ....................................................................................................................... 9
   5.6 Variable ....................................................................................................................................... 9
   5.7 Functions ..................................................................................................................................... 9
   5.8 Function Specifications .............................................................................................................. 10
   5.9 Flowcharts .................................................................................................................................. 13
      5.9.1 Main Processing .................................................................................................................... 13
      5.9.2 Port Initialization ................................................................................................................... 14
      5.9.3 Clock Initialization ............................................................................................................... 15
      5.9.4 Peripheral Function Initialization .......................................................................................... 16
      5.9.5 IRQ Initialization .................................................................................................................. 17
      5.9.6 RTC Initialization ................................................................................................................... 18
      5.9.7 Reading RTC Time Information ............................................................................................ 19
      5.9.8 Preparation to Enter Software Standby Mode ...................................................................... 19
      5.9.9 Exiting Software Standby Mode ........................................................................................... 20
      5.9.10 RTC Period Interrupt Handling .......................................................................................... 20
6. Sample Code .................................................................................................................................... 21
7. Reference Documents ...................................................................................................................... 21
1. Specifications

This document describes how to obtain the time information while intermittently exiting software standby mode using the RTC.

After a reset, if the reset processing is a cold start, the sub-clock oscillator and RTC are initialized. If the reset processing is a warm start, the RTC data is retained so initialization is not performed.

Next, monitor the input level of the interrupt request pin. If the input level is low, the MCU enters software standby mode.

Then when the periodic interrupt occurs every 0.5 seconds, the MCU exits software standby mode, time information is obtained, and the MCU enters software standby mode according to the input level of the interrupt request pin.

- RTC count source: Sub-clock
- VBATT pin: Connect to the VCC pin
- Intermittent period to exit software standby mode: 0.5 seconds

Table 1.1 lists the Peripheral Functions and Their Applications and Figure 1.1 shows the Operation Overview.

In this document, operating states other than software standby mode are referred to as normal mode.

<table>
<thead>
<tr>
<th>Peripheral Function</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTCa</td>
<td>Clock function or exiting from software standby mode</td>
</tr>
<tr>
<td>IRQ15</td>
<td>External input for entering software standby mode</td>
</tr>
</tbody>
</table>

**Table 1.1 Peripheral Functions and Their Applications**

**Figure 1.1 Operation Overview**
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>R5F563NBDDFC (RX63N Group)</td>
</tr>
</tbody>
</table>
| Operating frequencies         | - Main clock: 12 MHz  
- Sub-clock: 32.768 kHz  
- PLL clock: 192 MHz (main clock divided by 1 and multiplied by 16)  
- LOCO clock: 125 kHz  
- System clock (ICLK): 96 MHz (PLL clock divided by 2)  
- Peripheral module clock B (PCLKB): 48 MHz (PLL clock divided by 4) |
| Operating voltage             | 3.3 V                                                                                                                                     |
| Integrated development environment | Renesas Electronics Corporation  
High-performance Embedded Workshop Version 4.09.01                                                                                   |
| C compiler                    | Renesas Electronics Corporation  
C/C++ Compiler Package for RX Family V.1.02 Release 01                                                                                   |
| Compile options               | -cpu=rx600 -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" -debug -nologo  
The integrated development environment default settings are used.                                                                       |
| iof.h version                 | Version 1.50                                                                                                                              |
| Endian                        | Little endian                                                                                                                             |
| Operating mode                | Single-chip mode                                                                                                                          |
| Processor mode                | Supervisor mode                                                                                                                           |
| Sample code version           | Version 1.00                                                                                                                              |
| Device used                   | Renesas Starter Kit+ for RX63N (product part number: R0K50563NC000BE)                                                                       |

3. Reference Application Note

For additional information associated with this document, refer to the following application note.

- RX63N Group, RX631 Group Initial Setting Rev. 1.00 (R01AN1245EJ0100)

The initial setting functions in the reference application note are used in the sample code in this application note. The revision number of the reference application note is the one 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 Hardware Configuration

Figure 4.1 shows a Connection Example.

![Connection Example Diagram]

Figure 4.1 Connection Example

4.2 Pins Used

Table 4.1 lists the Pins Used and Their Functions.

This table assumes the 176-pin package is used. When using packages with less than 176 pins, select the pins appropriate to the package used.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P07/IRQ15</td>
<td>Input</td>
<td>Switch input for entering modes</td>
</tr>
<tr>
<td>P03</td>
<td>Output</td>
<td>LED output</td>
</tr>
</tbody>
</table>
5. Software

In normal mode, time information is updated and the level of the interrupt request pin is monitored. The RTC time information is stored in the time data storage area (global variable). The level of the interrupt request pin is monitored using the IR flag of the IRQ15 interrupt.

When the IR flag of the IRQ15 interrupt is 1, the RTC periodic interrupt (PRD interrupt) is enabled, and the MCU enters software standby mode. The MCU exits software standby mode by a PRD interrupt request, and the PRD interrupt is disabled.

After the MCU exits software standby mode, the RTC time information is updated, the level of the interrupt request pin is monitored, and the above processing is repeated.

Settings for the peripheral functions are listed below.

RTC
- Count source: Sub-clock
- Initial time setting: 00:00:00, Tuesday, January 1, 2013
- Time mode: 24-hour mode
- RTCOUT output: Disabled
- Error adjustment: Not used
- Time capture: Not used
- Interrupts used: PRD interrupt is generated in 0.5 sec. periods; Carry interrupt (CUP)

IRQ15 input pin
- Detection method: Low level is detected
- Digital filter: Enabled (sampling clock: PCLKB/8)
- Interrupt used: External pin interrupt (IRQ15 interrupt)
### 5.1 Operation Overview

(1) Initial setting

After a reset, if the RSTSR1.CWSF bit is 0 (cold start), the sub-clock oscillator and RTC are initialized, and the RSTSR1.CWSF bit is set to 1 (warm start). The LED is turned on, and the RTC time information is read. Also, the IR flag of the IRQ15 interrupt is monitored.

(2) Entering software standby mode

When the IR flag of the IRQ15 interrupt becomes 1, the PRD interrupt request is enabled, and the clock source for the system clock changes from the PLL clock to the LOCO clock. The LED is turned off, the WAIT instruction is executed, and the MCU enters software standby mode.

(3) Exiting software standby mode

The MCU exits software standby mode by the PRD interrupt request generated every 0.5 seconds. The LED is turned on in the PRD interrupt handling. The clock source for the system clock switches from the LOCO clock to the PLL clock, and the PRD interrupt request is disabled. After waiting 1/128th of a second, the RTC time information is read. If the IR flag of the IRQ15 interrupt is 1, perform the processing in step (3) again; if the IR flag is 0, read the RTC time information. Also, the IR flag of the IRQ15 interrupt is monitored.

(4) Warm start

After a reset, if the RSTSR1.CWSF bit is 1, the sub-clock oscillator and RTC are not initialized, and MCU operation continues.

Figure 5.1 shows the Timing Diagram.
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>
<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 FF8Bh 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>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW_STANDBY</td>
<td>IR(ICU,IRQ15)</td>
<td>IRQ15 interrupt status flag: Switch to enter software standby mode</td>
</tr>
<tr>
<td>SW_ON</td>
<td>1</td>
<td>Switch on</td>
</tr>
<tr>
<td>SW_OFF</td>
<td>0</td>
<td>Switch off</td>
</tr>
<tr>
<td>LED_RUN</td>
<td>PORT0.PODR.BIT.B3</td>
<td>P03 output data storage bit: LED</td>
</tr>
<tr>
<td>LED_ON</td>
<td>0</td>
<td>LED on</td>
</tr>
<tr>
<td>LED_OFF</td>
<td>1</td>
<td>LED off</td>
</tr>
<tr>
<td>LOOP_COUNT</td>
<td>96,000,000L/128</td>
<td>Loop counter: Wait at least 1/128th of a second (ICLK = 96 MHz)</td>
</tr>
</tbody>
</table>
5.5 Structure/Union List

Figure 5.2 shows the Structure/Union Used in the Sample Code.

```c
/* **** Time Data **** */
typedef struct {
    uint8_t second; /* Second */
    uint8_t minute; /* Minute */
    uint8_t hour; /* Hour */
    uint8_t dayweek; /* Day of the week */
    uint8_t day; /* Day */
    uint8_t month; /* Month */
    uint16_t year; /* Year */
} time_bcd_t;
```

Figure 5.2 Structure/Union Used in the Sample Code

5.6 Variable

Table 5.4 lists the Global Variable.

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_bcd_t</td>
<td>time</td>
<td>Time data storage area</td>
<td>rtc_time_read</td>
</tr>
</tbody>
</table>

5.7 Functions

Table 5.5 lists the Functions.

<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>irq_init</td>
<td>IRQ initialization</td>
</tr>
<tr>
<td>rtc_init</td>
<td>RTC initialization</td>
</tr>
<tr>
<td>rtc_time_read</td>
<td>Reading RTC time information</td>
</tr>
<tr>
<td>run_to_standby</td>
<td>Preparation to enter software standby mode</td>
</tr>
<tr>
<td>standby_to_run</td>
<td>Exiting software standby mode</td>
</tr>
<tr>
<td>Excep_RTC_PRD</td>
<td>RTC period interrupt handling</td>
</tr>
</tbody>
</table>
## 5.8 Function Specifications

The following tables list the sample code function specifications.

### main

<table>
<thead>
<tr>
<th>Outline</th>
<th>Main processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void main(void)</td>
</tr>
<tr>
<td>Description</td>
<td>After the initial setting, the LED is turned on, the RTC time information is read, and if switch input for entering software standby mode indicates on, the MCU enters software standby mode.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### port_init

<table>
<thead>
<tr>
<th>Outline</th>
<th>Port initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void port_init(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Initializes the ports.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### R_INIT_StopModule

<table>
<thead>
<tr>
<th>Outline</th>
<th>Stop processing for active peripheral functions after a reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_init_stop_module.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_INIT_StopModule(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Performs settings to enter the module-stop state.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td>Remark</td>
<td>Transition to the module-stop state is not performed in the sample code. For more information on this function, refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.00 application note.</td>
</tr>
</tbody>
</table>

### R_INIT_NonExistentPort

<table>
<thead>
<tr>
<th>Outline</th>
<th>Nonexistent port initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_init_non_existent_port.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_INIT_NonExistentPort(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Initializes port direction registers for ports that do not exist in products with less than 176 pins.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td>Remark</td>
<td>The number of pins in the sample code is set for the 176-pin package (PIN_SIZE=176). After this function is called, when writing in byte units to the PDR and 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 more information on this function, refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.00 application note.</td>
</tr>
</tbody>
</table>
### R_INIT_Clock

<table>
<thead>
<tr>
<th>Outline</th>
<th>Clock initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_init_clock.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_INIT_Clock(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Initializes clocks. Determines if the reset processing is a cold start or a warm start. The sub-clock is only initialized when the reset processing is a cold start.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td>Remark</td>
<td>In the sample code, the PLL clock is selected as the system clock, and the sub-clock is selected as the count source of the RTC (pattern D). For more information on this function, refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.00 application note.</td>
</tr>
</tbody>
</table>

### peripheral_init

<table>
<thead>
<tr>
<th>Outline</th>
<th>Peripheral function initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void peripheral_init(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Initializes the peripheral functions being used.</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>Initializes the IRQ.</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>Initializes the RTC.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### rtc_time_read

<table>
<thead>
<tr>
<th>Outline</th>
<th>Reading RTC time information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void rtc_time_read(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Reads the RTC time information and stores it in the time data storage area.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td><strong>run_to_standby</strong></td>
<td><strong>Outline</strong></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
</tr>
<tr>
<td><strong>Header</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
<td>void run_to_standby(void)</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Performs processing required before entering software standby mode.</td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Return Value</strong></td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>standby_to_run</strong></th>
<th><strong>Outline</strong></th>
<th>Exiting software standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
<td>void standby_to_run(void)</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Performs processing after exiting software standby mode.</td>
<td></td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td><strong>Return Value</strong></td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Excep_RTC_PRD</strong></th>
<th><strong>Outline</strong></th>
<th>RTC period interrupt handling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
<td>void Excep_RTC_PRD(void)</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Turns on the LED.</td>
<td></td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td><strong>Return Value</strong></td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
5.9 Flowcharts

5.9.1 Main Processing

Figure 5.3 shows the Main Processing.

![Flowchart of Main Processing](image)

**Figure 5.3 Main Processing**
5.9.2 Port Initialization

Figure 5.4 shows Port Initialization.

```
port_init

Set the port output data
- PORT0.PODR register
  B3 bit ← 1: P03: High output

Set the port direction
- PORT0.PDR register
  B3 bit ← 1: P03: Output
- PORT0.PDR register
  B7 bit ← 0: P07/IRQ15: Input

Set the port mode
- PORT0.PMR register
  B3 bit ← 0: P03: Use the pin as a general I/O pin
- PORT0.PMR register
  B7 bit ← 0: P07/IRQ15: Use the pin as a general I/O pin

return
```

Figure 5.4 Port Initialization
### 5.9.3 Clock Initialization

Figure 5.5 shows Clock Initialization.

**R_INIT_Clock**

Enable writing to registers

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 low power consumption.

Main clock oscillation setting
CGC_oscillation_main() (1)

Was reset processing
a cold start?

No

Read the RSTSR1 register
CWSF bit = 0: Cold start; CWSF bit = 1: Warm start

Processing when not using the sub-clock as the system clock
Function: no_use_subclk_as_sysclk

Yes

Sub-clock setting
CGC_subclk_as_RTC() (1)

Set the SOSTP bit to 1 so the sub-clock is not used as the system clock
SOSCCR register ← 01h
SOSTP bit = 1

Processing when using the sub-clock as the count source for the RTC, and not using the system clock (pattern D)

Wait processing
wait0_wait() (1)

Wait for two cycles of the sub-clock to elapse (approx. 61 µs)

PLL clock oscillation setting
CGC_oscillation_PLL() (1)

Set the internal clock division rate
SCKCR register ← 21C2 1211h
Bits PCKB[3:0] = 0010b: Peripheral module clock B (PCLKB) × 1/4
Bits PCKA[3:0] = 0001b: Peripheral module clock A (PCLKA) × 1/2
Bits BCK[3:0] = 0010b: System clock (ICLK) × 1/4
Bits FCK[3:0] = 0010b: FlashIF clock (FCLK) × 1/4
SCKCR2 register ← 0012h
Bits IEBCK[3:0] = 0010b: IEBUS clock (IECLK) × 1/4
Bits UCK[3:0] = 0001b: USB is not used

Set the BCLK pin output level
BCKCR register ← 00h
BCLKDIV bit = 0: BCLK

Switch the system clock
SCKCR3 register ← 0400h
Bits CKSEL[2:0] = 100b: PLL circuit

Turn the HOCO power supply off
HOCCCR register ← 01h
HCSTP bit = 1: HOCO is stopped
HOCCPCR register ← 01h
HOCCPCNT bit = 1: Turns the power supply of the HOCO off

Disable writing to registers
PRCR register ← A500h
PRC0 bit = 0: Disables writing to the registers related to the clock generation circuit
PRC1 bit = 0: Disables writing to the registers related to low power consumption

return

Note:
1. For more information on this function, refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.00 application note.

---

**Figure 5.5 Clock Initialization**
5.9.4 Peripheral Function Initialization

Figure 5.6 shows Peripheral Function Initialization.

```
peripheral_init

Enable writing to registers

PRCR register ← A502h
PRC1 bit = 1: Related registers are write enabled

Set the mode to enter into after the
WAIT instruction is executed

SBYCR register
SBBY bit ← 1
DPSBYCR register
DPSBY bit ← 0: Transition to software standby mode is
made after the WAIT instruction is executed

Disable writing to registers

PRCR register ← A500h
PRC1 bit = 0: Related registers are write disabled

Disable operations when entering
software standby mode

DMAST register
DMST bit ← 0: DMAC activation is disabled

DTCST register
DTCST bit ← 0: DTC module-stop

IER08 register ← 00h
 IEN0 bit = 0: IRQ0 interrupt request is disabled
 IEN1 bit = 0: IRQ1 interrupt request is disabled
 IEN2 bit = 0: IRQ2 interrupt request is disabled
 IEN3 bit = 0: IRQ3 interrupt request is disabled
 IEN4 bit = 0: IRQ4 interrupt request is disabled
 IEN5 bit = 0: IRQ5 interrupt request is disabled
 IEN6 bit = 0: IRQ6 interrupt request is disabled
 IEN7 bit = 0: IRQ7 interrupt request is disabled

IER09 register ← 00h
 IEN0 bit = 0: IRQ8 interrupt request is disabled
 IEN1 bit = 0: IRQ9 interrupt request is disabled
 IEN2 bit = 0: IRQ10 interrupt request is disabled
 IEN3 bit = 0: IRQ11 interrupt request is disabled
 IEN4 bit = 0: IRQ12 interrupt request is disabled
 IEN5 bit = 0: IRQ13 interrupt request is disabled
 IEN6 bit = 0: IRQ14 interrupt request is disabled
 IEN7 bit = 0: IRQ15 interrupt request is disabled

Disable the CUP interrupt request

IER07 register
IEN6 bit ← 0

Disable the PRD interrupt request

IER0B register
IEN5 bit ← 0

Set the PRD interrupt priority level

IPR093 register
Bits IPR[3:0] ← 0001b: Level 1

Is the reset processing
a cold start?

Yes
RTC initialization
rtc_init()

Enable a carry interrupt request (1) RCR1 register ← D2h
CIE bit = 1

Set the reset processing to warm start

RSTSR1 register
CWSF bit ← 1

IRQ initialization
irq_init()

return

No

Read the RSTSR1 register
CWSF bit = 0: Cold start; CWSF bit = 1: Warm start

Set the mode to enter into after the
WAIT instruction is executed

SBYCR register
SBBY bit ← 1
DPSBYCR register
DPSBY bit ← 0: Transition to software standby mode is
made after the WAIT instruction is executed

Set the reset processing to warm start

RSTSR1 register
CWSF bit ← 1

IEN7 bit = 0: IRQ15 interrupt request is disabled

Note:
1. After setting a value to the RCR1 register, read the register to confirm that the value set can be read.

Figure 5.6 Peripheral Function Initialization
```
5.9.5 IRQ Initialization

Figure 5.7 shows IRQ Initialization.

```
irq_init

Disable the IRQ15 interrupt request
   IER09 register IEN7 bit ← 0

Disable the IRQ15 digital filter
   IRQFLTE1 register FLTEN15 bit ← 0

Set the IRQ15 sampling clock
   IRQFTLC1 register FCLKSEL15 bit ← 1: PCLKB/8

Enable writing to the PFSWE bit
   PWPR register B0WI bit ← 0

Enable writing to the PFS register
   PWPR register PFSWE bit ← 1

Select the pin function
   P07PFS register ISEL bit ← 1: Used as IRQ15 input pin

Disable writing to the PFS register
   PWPR register PFSWE bit ← 0

Disable writing to the PFSWE bit
   PWPR register B0WI bit ← 1

Set the IRQ15 detection method
   IRQCR15 register ← 00h
   Bits IRQMD[1:0] = 00b: Low level

Enable the IRQ15 digital filter
   IRQFLTE1 register FLTEN15 bit ← 1

return
```

Figure 5.7 IRQ Initialization
5.9.6 RTC Initialization

Figure 5.8 shows RTC Initialization.

```
// 5.9.6 RTC Initialization

#define RTC_InitStruct (RTC_CNTL_CLKSOURCE_PLL, 0)

void rtc_init(void)
{
    RSECCNT = 0; // Set the date and time to Tuesday, January 1, 2013, 00:00:00
    RMINCNT = 0; // Bits SEC1[3:0] = 0000b: Ones place of seconds: 0
    RHRCNT = 0; // Bits SEC10[2:0] = 000b: Tens place of seconds: 0
    RWKCNT = 02; // Bits MIN1[3:0] = 0000b: Ones place of minutes: 0
    RDAYCNT = 01; // Bits MIN10[1:0] = 00b: Tens place of minutes: 0
    RMONCNT = 01; // Bits HR1[3:0] = 0001b: Ones place of hours: 1
    RYRCNT = 0013; // Bits HR10[3:0] = 0001b: Tens place of hours: 1
    RCR2 = 0; // Stop counters and the prescaler
    RCR1 = 0; // Disable interrupt requests
    RCR2 = RTC_InitStruct; // Operate counters and the prescaler
    return;
}
```

**Notes:**
1. After writing to the RCR1 register and the RCR2.START bit, read the register and the bit to confirm that the written values can be read.
2. After writing to the RCR2.HR24 bit, perform a dummy read three times.
5.9.7 Reading RTC Time Information

Figure 5.9 shows the Reading RTC Time Information.

```
5.9.7  Reading RTC Time Information
```

```
Figure 5.9 Reading RTC Time Information
```

---

5.9.8 Preparation to Enter Software Standby Mode

Figure 5.10 shows Preparation to Enter Software Standby Mode.

```
Figure 5.10 Preparation to Enter Software Standby Mode
```

---

Notes:
1. After writing to the RCR1 register and the SCKCR3 register, read the registers to confirm that the written values can be read.
2. After writing to the last register to be set, read the register to confirm that the written value can be read.
## 5.9.9 Exiting Software Standby Mode

Figure 5.11 shows Exiting Software Standby Mode.

```plaintext
<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standby_to_run</td>
<td></td>
</tr>
<tr>
<td>Enable writing to registers</td>
<td>PRCR register ← A501h</td>
</tr>
<tr>
<td></td>
<td>PRC0 bit = 1: Enables writing to related registers</td>
</tr>
<tr>
<td>Switch the clock source for the system clock</td>
<td>SCKCR3 register ← 0400h</td>
</tr>
<tr>
<td></td>
<td>Bits CKSEL[2:0] = 100b: PLL circuit</td>
</tr>
<tr>
<td>Disable writing to registers</td>
<td>PRCR register ← A500h</td>
</tr>
<tr>
<td></td>
<td>PRC0 bit = 0: Enables writing to related registers</td>
</tr>
<tr>
<td>Disable the RTC.PRD interrupt request</td>
<td>IER0B register</td>
</tr>
<tr>
<td></td>
<td>IEN5 bit ← 0</td>
</tr>
<tr>
<td>Disable the PRD interrupt request</td>
<td>RCR1 register ← D2h</td>
</tr>
<tr>
<td></td>
<td>PIE bit = 0: A periodic interrupt request is disabled.</td>
</tr>
<tr>
<td>Clear the RTC.PRD interrupt request</td>
<td>IR093 register</td>
</tr>
<tr>
<td></td>
<td>IR flag ← 0</td>
</tr>
<tr>
<td>Wait 1/128th of a second</td>
<td>After exiting software standby mode, wait 1/128th of a second and then read the RTC counter.</td>
</tr>
<tr>
<td>Reading RTC time information</td>
<td>rtc_time_read()</td>
</tr>
<tr>
<td>Enable the IRQ15 digital filter</td>
<td>IRQFLTE1 register</td>
</tr>
<tr>
<td></td>
<td>FLTEN15 bit ← 1</td>
</tr>
<tr>
<td>When exiting software standby mode</td>
<td>Registers IR064 to IR078</td>
</tr>
<tr>
<td></td>
<td>IR flag ← 0: IRQ0 to IRQ14 interrupt requests are disabled</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Note:
1. After writing to the RCR1 register and the SCKCR3 register, read the registers to confirm that the written values can be read.

Figure 5.11   Exiting Software Standby Mode

## 5.9.10 RTC Period Interrupt Handling

Figure 5.12 shows RTC Period Interrupt Handling.

```plaintext
<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Excep_RTC_PRD</td>
<td></td>
</tr>
<tr>
<td>Turn off the LED</td>
<td>PORT0.PODR register</td>
</tr>
<tr>
<td></td>
<td>B3 bit ← 0: P03: Low output</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 5.12   RTC Period Interrupt Handling
6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

7. Reference Documents

User’s Manual: Hardware
RX63N Group, RX631 Group User’s Manual: Hardware Rev.1.50 (R01UH0041EJ)
The latest version 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>June 14, 2013</td>
<td></td>
<td>First edition issued</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.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products
   Before changing from one product to another, i.e. to one with a different part number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different part numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different part numbers, implement a system-evaluation test for each of the products.