RX63N Group, RX631 Group
True/False Determination of RTC Time Information on Cold Start

Abstract

This document describes the method to determine whether power is supplied from the VBATT while the voltage on the VCC pin is dropped when using the realtime clock (RTC) and the battery backup function together. The RTC clock settings and registers are used to determine whether the RTC continues operation or the RTC is initialized.

Products

- RX63N Group 177-pin and 176-pin packages with a ROM size between 768 KB and 2 MB
- RX63N Group 145-pin and 144-pin packages with a ROM size between 768 KB and 2 MB
- RX63N Group 100-pin package with a ROM size between 768 KB and 2 MB
- RX631 Group 177-pin and 176-pin packages with a ROM size between 256 KB and 2 MB
- RX631 Group 145-pin and 144-pin packages with a ROM size between 256 KB and 2 MB
- RX631 Group 100-pin package with a ROM size between 256 KB and 2 MB
- RX631 Group 64-pin package with a ROM size between 256 KB and 512 KB

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

After a reset, the software determines whether to continue the RTC count operation or initialize the RTC, then performs the operation according to the determination.

The battery backup function enables the RTC and the sub-clock oscillator to continue operating with power supplied by the VBATT even when the voltage on the VCC pin is dropped. The RTC must be initialized only when the voltage on the VCC pin continues being dropped and power supply from the VBATT is stopped. But after both of them hung a VCC pin and a VBATT pin on 0V once when the voltage of VBATT was lower than a guarantee area after changing to VBATT, please do a power-on reset.

The VBATT circuit is assumed that it takes 15 minutes for charging the connected capacitor at a maximum, and the fully charged capacitor can supply power for at least 10 days.

In this application note, the following are verified for determination:

- If the VCC voltage is recovered and the reset is released within the period the VBATT can supply power.
- If the charging time for the VBATT is enough.
- If the values stored in the RTC registers are correct after a reset.

The RTC time information displayed on the Debug LCD of the RSK is updated every second while the VCC voltage is retained.

Also the RTC time information (backup) stored in the E2 DataFlash is updated in the following timings:

- After a reset and 1 second elapses.
- After 15 minutes, which is time required to charge the capacitor connected to the VBATT.
- When the hour counter (RHRCNT) for the RTC time information is updated.

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

<table>
<thead>
<tr>
<th>Peripheral Function</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTCa</td>
<td>Clock counter</td>
</tr>
<tr>
<td>VBATT</td>
<td>Power supply to the RTC and the sub-clock oscillator.</td>
</tr>
<tr>
<td>E2 DataFlash</td>
<td>Backup for the RTC time information</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Table 2.1 Operation Confirmation Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item</strong></td>
</tr>
<tr>
<td>MCU used</td>
</tr>
</tbody>
</table>
| Operating frequencies | - Main clock: 12 MHz  
- Sub-clock: 32.768 kHz  
- PLL: 192 MHz (main clock divided by 1 and multiplied by 16)  
- LOCO: 125 kHz  
- System clock (ICLK): 96 MHz (PLL divided by 2)  
- Peripheral module clock B (PCLKB): 48 MHz (PLL divided by 4) |
| Operating voltage | 3.3 V, VBATT is V_BATT or greater |
| 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 default setting is used in the integrated development environment.) |
| iodefine.h version | Version 1.8 |
| Endian | Little endian |
| Operating mode | Single-chip mode |
| Processor mode | Supervisor mode |
| Sample code version | Version 1.01 |
| Board used | Renesas Starter Kit+ for RX63N (product part no.: R0K50563NC000BE) |

3. Reference Application Notes

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

- RX63N Group, RX631 Group Initial Setting Rev. 1.10 (R01AN1245EJ)
- RX600 & RX200 Series Simple Flash API for RX Rev.2.40 (R01AN0544EU)
- RX Family Using the Simple Flash API for RX without the r_bsp Module Rev.1.00 (R01AN1890EU)
- Renesas Starter Kit Sample Code for Hi-performance Embedded Workshop Rev.1.00 (R01AN1395EG)
- RX Family Coding Example of Wait Processing by Software Rev. 1.00 (R01AN1852EJ0100)

The initial setting functions, Debug LCD output functions, simple flash API functions, coding example of wait processing by software functions in the reference application notes are used in the sample code accompanying this application note. The revision numbers of the reference application notes are as of when these application notes were 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 the Connection Example.

![Connection Example Diagram]

In this application note, the maximum charging time is 15 minutes and the shortest discharging time (time to retain power) is 10 days.

Note that the charging and discharging time vary depending on the circuit structure of the VBATT pins. Make sure to use the charge and discharge time carefully evaluated and confirmed in the user system.

4.2 Pins Used

Table 4.1 lists the Pins Used and Their Functions.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P03</td>
<td>Output</td>
<td>LED0 output to indicate completion of the RTC initialization</td>
</tr>
<tr>
<td>P05</td>
<td>Output</td>
<td>LED1 output to indicate E2 DataFlash backup error</td>
</tr>
<tr>
<td>P87</td>
<td>Output</td>
<td>“Debug LCD Data bit 7” output</td>
</tr>
<tr>
<td>P86</td>
<td>Output</td>
<td>“Debug LCD Data bit 6” output</td>
</tr>
<tr>
<td>P85</td>
<td>Output</td>
<td>“Debug LCD Data bit 5” output</td>
</tr>
<tr>
<td>P84</td>
<td>Output</td>
<td>“Debug LCD Data bit 4” output</td>
</tr>
<tr>
<td>PJ5</td>
<td>Output</td>
<td>“Debug LCD Enable” output</td>
</tr>
<tr>
<td>PF5</td>
<td>Output</td>
<td>“Debug LCD Register select” output</td>
</tr>
</tbody>
</table>
5. Software

5.1 Operation Overview

In this application note, the RTC time information is validated using the items below as criteria. According to the validation, the RTC is determined to continue operation or be initialized. Figure 5.1 shows the Validation Procedure.

- RTC count source: Sub-clock operating
- RTC control register: Same value as the initial value
- RTC automatic adjustment setting: Same value as the initial value when this setting is enabled
- RTC time information: Values that are correct as the time information and within 10 days from the backup time information in the E2 DataFlash.
- RTC alarm information: Values that are correct as the time information and within 10 days from the backup time information in the E2 DataFlash.
- Charge detection in the VBATT circuit: Cold start and 15 minutes elapsed from the previous RTC initialization.

RTC initial setting values are as follows:

- Hours mode: 12-hour mode
- Initial time setting: 00:00:00, Tuesday, January 1, 2013
- Alarm setting: Disabled
- RTCOUT output: Output disabled
- Automatic adjustment: Not used
- Count source: Sub-clock for standard clock loads
- Time capture: Not used
- Interrupts: The periodic interrupt (PRD) is generated every second.
  The alarm interrupt and the carry interrupt are disabled.
5.2 Battery Backup Function

With the connected circuit in this application note, an output voltage of $V_{BATT}$ or greater on the VBATT pin can be retained for at least 10 days, and the capacitor can be fully charged within 15 minutes.

Note that the charging and discharging time vary depending on the connected circuit. Make sure to use the charging and discharging time carefully evaluated and confirmed in the user system.
5.3 Programming/Erasing the E2 DataFlash
In this application note, addresses 0010 0000h to 0010 003Fh (64 bytes) of the E2 DataFlash are used. 10 bytes of the RTC time information is overwritten and stored in addresses 0010 0000h to 0010 0009h and 0010 0020h to 0010 0029h alternately. Data is not retained for the other areas in the 64 bytes of the E2 DataFlash.

For details on the other Simple Flash API used for programming/erasing, refer to the RX600 & RX200 Series Simple Flash API for RX.

5.3.1 Erasing the E2 DataFlash
To erase the E2 DataFlash, the R_FlashEraseRange function of the Simple Flash API for RX is used. The erase error can be determined by the return value from the R_FlashEraseRange function or the register value in the FlashError function.

5.3.2 Programming the E2 DataFlash
To program the E2 DataFlash, the R_FlashWrite function of the Simple Flash API for RX is used. The program error can be determined by the return value from the R_FlashWrite function or the register value in the FlashError function.

5.3.3 Changes in the Simple Flash API
Programming and erasing of the E2 DataFlash is performed using the program in the flash memory (ROM). The main loop can be executed during programming or erasing.

In this application note, r_flash_api_rx_config.h is changed in the Simple Flash API.
Table 5.1 shows Changes in r_flash_api_rx_config.h.

<table>
<thead>
<tr>
<th>Changed Item</th>
<th>Description</th>
<th>Code After the Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Settings in the Simple Flash API</td>
<td>No programming performed in the ROM</td>
<td>// #define FLASH_API_RX_CFG_ENABLE_ROM_PROGRAMMING</td>
</tr>
<tr>
<td></td>
<td>Data to be programmed is stored in the RAM buffer.</td>
<td>#define FLASH_API_RX_CFG_FLASH_TO_FLASH</td>
</tr>
<tr>
<td></td>
<td>E2 DataFlash operation is executed in background.</td>
<td>#define FLASH_API_RX_CFG_DATA_FLASH_BGO</td>
</tr>
<tr>
<td></td>
<td>Flash API is not transferred to the RAM</td>
<td>// #define FLASH_API_RX_CFG_COPY_CODE_BY_API</td>
</tr>
</tbody>
</table>

Note: • When VCC power is cut off or reset during programming or erasing the E2 DataFlash, the RTC time information becomes incorrect. Thus the RTC is initialized after a reset.
5.4 File Composition

Table 5.2 and Table 5.3 list the files used in this application notes. Table 5.4 and Table 5.5 list the functions and setting values in the reference application notes. Table 5.6 lists files from the simple flash API module.

Files generated by the integrated development environment are not included in this table.

Table 5.2 Files Used in the Sample Code

<table>
<thead>
<tr>
<th>File Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.c</td>
<td>Main processing, port setting</td>
</tr>
<tr>
<td>r_init_clock_an1713.c</td>
<td>Clock initialization</td>
</tr>
<tr>
<td>r_init_clock_an1713.h</td>
<td>Header file for r_init_clock_an1713.c</td>
</tr>
<tr>
<td>rtc_func.c</td>
<td>RTC register determination, RTC initialization, and periodic interrupt handling</td>
</tr>
<tr>
<td>rtc_func.h</td>
<td>Header file for rtc_func.c</td>
</tr>
<tr>
<td>flash_write.c</td>
<td>Processing for backing up the RTC time information to the E2 DataFlash</td>
</tr>
</tbody>
</table>

Table 5.3 Standard Include Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>stdbool.h</td>
<td>Defines macros regarding the boolean type and value.</td>
</tr>
<tr>
<td>stdint.h</td>
<td>Defines macros declaring the integer type with the specified width.</td>
</tr>
<tr>
<td>machine.h</td>
<td>Defines types of built-in function for the RX Family.</td>
</tr>
</tbody>
</table>

Table 5.4 Functions and Setting Values in the RX63N Group, RX631 Group Initial Setting Application Note

<table>
<thead>
<tr>
<th>File Name</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_init_stop_module.c</td>
<td>R_INIT_StopModule()</td>
<td>−</td>
</tr>
<tr>
<td>r_init_stop_module.h</td>
<td>−</td>
<td>Module-stop state is canceled for DMAC/DTC, EXDMAC, RAM0, and RAM1.</td>
</tr>
<tr>
<td>r_init_non_existent_port.c</td>
<td>R_INIT_NonExistentPort()</td>
<td>−</td>
</tr>
<tr>
<td>r_init_non_existent_port.h</td>
<td>−</td>
<td>176 pin package is specified.</td>
</tr>
<tr>
<td>r_delay.c</td>
<td>R_DELAY_Us()</td>
<td>The waiting time is set.</td>
</tr>
<tr>
<td>r_delay.h</td>
<td>−</td>
<td>−</td>
</tr>
</tbody>
</table>

Table 5.5 Functions and Setting Values in the Sample Code Accompanying the Renesas Starter Kit for RX63N High-performance Embedded Workshop

<table>
<thead>
<tr>
<th>File Name</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lcd.c</td>
<td>- Init_LCD()</td>
<td>−</td>
</tr>
<tr>
<td></td>
<td>- Display_LCD()</td>
<td>−</td>
</tr>
<tr>
<td>lcd.h</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>rskr63ndef.h</td>
<td>−</td>
<td>−</td>
</tr>
</tbody>
</table>
Table 5.6 Simple Flash API Modules
(RX600 & RX200 Series Simple Flash API for RX Application Note)

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_flash_api_rx</td>
<td>RX Simple Flash API programs for RX600 and RX200 Series</td>
</tr>
<tr>
<td>u_bsp</td>
<td>Board support package for users (1)</td>
</tr>
</tbody>
</table>

Note:
1. The module is modified following the application note “Using the Simple Flash API for RX without the r_bsp Module”.

5.5 Option-Setting Memory

Table 5.7 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

Table 5.7 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 FF8Bh to FFFF FF88h</td>
<td>FFFF FFFBh</td>
<td>The voltage monitor 0 reset is enabled 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>
5.6 Constants

Table 5.8 lists the Constants Used in the Sample Code.

Table 5.8 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>RTC_INIT_RES</td>
<td>1</td>
<td>Reset with the RTC initialization required</td>
</tr>
<tr>
<td>RTC_RUN_RES</td>
<td>0</td>
<td>Reset with the RTC initialization not required</td>
</tr>
<tr>
<td>RTC_BACKUP_INIT</td>
<td>2</td>
<td>Backup of the RTC time information is requested. (Initialization source, backup request after 1 second)</td>
</tr>
<tr>
<td>RTC_BACKUP_ACK</td>
<td>1</td>
<td>Backup of the RTC time information is requested. (backup request after 15 minutes and every hour)</td>
</tr>
<tr>
<td>RTC_BACKUP_NACK</td>
<td>0</td>
<td>Backup of the RTC time information is not requested.</td>
</tr>
<tr>
<td>VBATT_FULL_CHARGE</td>
<td>1</td>
<td>VBATT circuit is fully charged.</td>
</tr>
<tr>
<td>VBATT_EMPTY</td>
<td>0</td>
<td>VBATT circuit is not charged enough.</td>
</tr>
<tr>
<td>MASK_DATA_SEC</td>
<td>0x01</td>
<td>Execution flag of second data verification, data for verifying the flag</td>
</tr>
<tr>
<td>MASK_DATA_MIN</td>
<td>0x02</td>
<td>Execution flag of minute data verification, data for verifying the flag</td>
</tr>
<tr>
<td>MASK_DATA_HR</td>
<td>0x04</td>
<td>Execution flag of hour data verification, data for verifying the flag</td>
</tr>
<tr>
<td>MASK_DATA_WK</td>
<td>0x08</td>
<td>Execution flag of day-of-week data verification, data for verifying the flag</td>
</tr>
<tr>
<td>MASK_DATA_DAY</td>
<td>0x10</td>
<td>Execution flag of day data verification, data for verifying the flag</td>
</tr>
<tr>
<td>MASK_DATA_MON</td>
<td>0x20</td>
<td>Execution flag of month data verification, data for verifying the flag</td>
</tr>
<tr>
<td>MASK_DATA_YR</td>
<td>0x40</td>
<td>Execution flag of year data verification, data for verifying the flag</td>
</tr>
<tr>
<td>ALL_DATA_CHECK</td>
<td>0x7F</td>
<td>Execution flag of all data verification</td>
</tr>
<tr>
<td>ADDRESS_BLOCK_DB0</td>
<td>0x100000</td>
<td>Start address of the DB0 block</td>
</tr>
<tr>
<td>ADDRESS_BLOCK_DB1</td>
<td>0x100020</td>
<td>Start address of the DB1 block</td>
</tr>
<tr>
<td>BUFF_SIZE</td>
<td>10</td>
<td>Sizes for storing the time information and programming</td>
</tr>
<tr>
<td>WRITE_BUSY</td>
<td>2</td>
<td>Programming mode (programming)</td>
</tr>
<tr>
<td>ERASE_BUSY</td>
<td>1</td>
<td>Programming mode (erasing)</td>
</tr>
<tr>
<td>WRITE_READY</td>
<td>0</td>
<td>Programming mode (preparing)</td>
</tr>
<tr>
<td>SELECT_SUB</td>
<td>PATTERN_D</td>
<td>Selection of the sub-clock setting pattern</td>
</tr>
</tbody>
</table>
5.7 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 */
    uint8_t     charge_VBATT;          /* VBATT charge info */
    uint8_t     write_cnt;             /* Backup counter */
} time_bcd_t;
```

Figure 5.2 Structure/Union Used in the Sample Code
### 5.8 Variables

Table 5.9 lists the Global Variables, Table 5.10 lists the static Variables, and Table 5.11 lists the const Variable.

#### Table 5.9 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>init_RTC</td>
<td>RTC initialization request flag</td>
<td>- main</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- enable_RTC_an1713</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- CGC_subclk_as_RTC_an1713</td>
</tr>
<tr>
<td></td>
<td>time_data</td>
<td>Buffer to obtain time information</td>
<td>- check_RTC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- flash_check</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- check_RTC_flash</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- check_RTC_after10days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- enable_RTC_an1713</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Excep_RTC_PRD</td>
</tr>
<tr>
<td></td>
<td>flash_write</td>
<td>Backup request flag for the RTC time information</td>
<td>- main</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- check_RTC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- enable_RTC_an1713</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Excep_RTC_PRD</td>
</tr>
<tr>
<td></td>
<td>write_counter</td>
<td>Backup counter</td>
<td>- main</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- check_RTC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- flash_check</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- check_RTC_flash</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- check_RTC_after10days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Excep_RTC_PRD</td>
</tr>
<tr>
<td></td>
<td>write_mode</td>
<td>Programming mode</td>
<td>- Excep_RTC_PRD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- flash_check</td>
</tr>
<tr>
<td></td>
<td>lcd_buffer[9]</td>
<td>Buffer for Debug LCD</td>
<td>- main</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- uint32_ToBCDString</td>
</tr>
<tr>
<td></td>
<td>flash_is_ready</td>
<td>Flash processing end flag</td>
<td>- time_backup</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- check_RTC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Excep_RTC_PRD</td>
</tr>
</tbody>
</table>

#### Table 5.10 static Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>charge_min_count</td>
<td>Charge time (minutes) counter for the VBAT circuit</td>
<td>- check_RTC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Excep_RTC_PRD</td>
</tr>
<tr>
<td>static</td>
<td>check_time_data</td>
<td>Time information check required/time information check not required</td>
<td>- check_RTC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Set_register_enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- flash_check</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- check_RTC_clock</td>
</tr>
<tr>
<td>static</td>
<td>prog_buff</td>
<td>Time information for backup</td>
<td>- time_backup</td>
</tr>
</tbody>
</table>

#### Table 5.11 const Variable

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>const static</td>
<td>backup_address[2] =</td>
<td>{ADDRESS_BLOCK_DB0, ADDRESS_BLOCK_DB1);</td>
<td>- check_RTC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- flash_check</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- check_RTC_flash</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- check_RTC_after10days</td>
</tr>
</tbody>
</table>
5.9 Functions

Table 5.12 lists the Functions.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
<th>File the Function is in</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>Main processing</td>
<td>main.c</td>
</tr>
<tr>
<td>port_init</td>
<td>Port output setting</td>
<td>main.c</td>
</tr>
<tr>
<td>check_RTC</td>
<td>Determination of RTC initialization requirement</td>
<td>rtc_func.c</td>
</tr>
<tr>
<td>Set_register_enable</td>
<td>Setting the validity of the alarm time information</td>
<td>rtc_func.c</td>
</tr>
<tr>
<td>flash_check</td>
<td>Determination of the backup time information for use</td>
<td>rtc_func.c</td>
</tr>
<tr>
<td>check_RTC_clock</td>
<td>Verification of the RTC time information value</td>
<td>rtc_func.c</td>
</tr>
<tr>
<td>check_RTC_flash</td>
<td>Comparison of the RTC time information with the backup time information</td>
<td>rtc_func.c</td>
</tr>
<tr>
<td>check_RTC_after10days</td>
<td>Verifying the range of the RTC time information</td>
<td>rtc_func.c</td>
</tr>
<tr>
<td>rtc_time_read</td>
<td>Reading the RTC time information</td>
<td>rtc_func.c</td>
</tr>
<tr>
<td>R_INIT_Clock_an1713</td>
<td>Clock initialization</td>
<td>r_init_clock_an1713.c</td>
</tr>
<tr>
<td>CGC_subclk_as_RTC_an1713</td>
<td>Sub-clock setting (sub-clock is used for the RTC count source and not</td>
<td>r_init_clock_an1713.c</td>
</tr>
<tr>
<td></td>
<td>used for the system clock)</td>
<td></td>
</tr>
<tr>
<td>enable_RTC_an1713</td>
<td>RTC initialization</td>
<td>rtc_func.c</td>
</tr>
<tr>
<td>Excep_RTC_PRD</td>
<td>RTC periodic interrupt handling</td>
<td>rtc_func.c</td>
</tr>
<tr>
<td>time_backup</td>
<td>Backup processing for the RTC time information</td>
<td>flash_write.c</td>
</tr>
</tbody>
</table>
5.10 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 initialization, if a backup request of the RTC time information is generated, the backup processing function is called.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>In the main loop processing, do not rewrite registers that are write protected while operating in E2 DataFlash P/E mode. When additional processing is added to the main loop, and if operating in E2 DataFlash P/E mode, any processing including the additional processing is executed in E2 DataFlash P/E mode.</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>static 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>

### check_RTC

<table>
<thead>
<tr>
<th>Outline</th>
<th>Determination of RTC initialization requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>rtc_func.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>uint8_t check_RTC(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Compares the RTC register value with the RTC initial setting value specified by the enable_RTC_an1713 function, performs processing and verification regarding the backup time information, determines cold/warm start, verifies the charge status of the VBATT circuit at a backup timing, and then determines whether to continue the RTC operation or initialize the RTC.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>RTC_INIT_RES: RTC time information is invalid. RTC_RUN_RES: RTC time information is valid.</td>
</tr>
</tbody>
</table>

### flash_check

<table>
<thead>
<tr>
<th>Outline</th>
<th>Determination of the backup time information for use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>rtc_func.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>static uint8_t flash_check(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Verifies the backup counters DB0 and DB1, and determines data to be used for the subsequent comparison processing. If the backup counter is invalid, reads the backup time information to determine the data to be used for the subsequent comparison processing.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>RTC_INIT_RES: RTC time information is invalid. RTC_RUN_RES: RTC time information is valid.</td>
</tr>
</tbody>
</table>
### Set_register_enable

**Outline**
Setting the validity of the alarm time information

**Header**
rtc_func.h

**Declaration**
static void Set_register_enable(void)

**Description**
Only when the ENB bit of the alarm time information is set to 1, the ENB bit of the RTC alarm register is read to verify the register value.

**Arguments**
None

**Return Value**
None

---

### check_RTC_clock

**Outline**
Verification of the RTC time information value

**Header**
rtc_func.h

**Declaration**
static unsigned char check_RTC_clock(void)

**Description**
Verifies that the register value of the RTC time information is correct.

**Arguments**
None

**Return Value**
RTC_INIT_RES: RTC time information is invalid.
RTC_RUN_RES: RTC time information is valid.

---

### check_RTC_flash

**Outline**
Comparison of the RTC time information with the backup time information

**Header**
rtc_func.h

**Declaration**
static unsigned char check_RTC_flash(void)

**Description**
Compares the RTC time information with the backup time information and verifies the RTC time information is newer than the backup time information.

**Arguments**
None

**Return Value**
RTC_INIT_RES: RTC time information is invalid.
RTC_RUN_RES: RTC time information is valid.

---

### check_RTC_after10days

**Outline**
Verifying the range of the RTC time information

**Header**
rtc_func.h

**Declaration**
static unsigned char check_RTC_after10days(void)

**Description**
Compares the RTC time information with the backup time information and verifies the RTC time information is within normal range (10 days after the backup time information).

**Arguments**
None

**Return Value**
RTC_INIT_RES: RTC time information is invalid.
RTC_RUN_RES: RTC time information is valid.

---

### rtc_time_read

**Outline**
Reading the RTC time information

**Header**
None

**Declaration**
static void rtc_time_read(void)

**Description**
Reads the RTC time information and stores it in the RAM area for the RTC time information.

**Arguments**
None

**Return Value**
None
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
<th>Header</th>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_INIT_Clock_an1713</td>
<td>Clock initialization</td>
<td>r_init_clock.h</td>
<td>void R_INIT_Clock_an1713(void)</td>
<td>Initializes the clock.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The sample code selects processing which uses PLL as the system clock and the sub-clock as the RTC count source. Refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.10 application note for details on this function.</td>
</tr>
<tr>
<td>CGC_subclk_as_RTC_an1713</td>
<td>Sub-clock setting</td>
<td>r_init_clock.h</td>
<td>void CGC_subclk_as_RTC_an1713(void)</td>
<td>Configures the setting when the sub-clock is used as the RTC count source and not used as the system clock.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Refer to pattern D in the RX63N Group, RX631 Group Initial Setting Rev. 1.10 application note for details on this function.</td>
</tr>
<tr>
<td>enable_RTC_an1713</td>
<td>RTC Initialization</td>
<td>None</td>
<td>static void enable_RTC_an1713(void)</td>
<td>Performs the initialization when the RTC is used (setting for clock provision and RTC software reset).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Excep_RTC_PRD</td>
<td>RTC periodic interrupt handling</td>
<td>rtc_func.h</td>
<td>static void Excep_RTC_PRD(void)</td>
<td>Reads the RTC time information in this interrupt handling and displays it on the Debug LCD. After a reset and time required to charge the VBATT circuit elapsed, updates the VBATT charge flag and the backup counter and sets the backup request. Also updates the backup counter every hour and sets the backup request.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Remarks**

- Arguments: None
- Return Value: None
- Remarks: Refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.10 application note for details on this function.
<table>
<thead>
<tr>
<th>time_backup</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outline</strong></td>
</tr>
<tr>
<td><strong>Header</strong></td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
</tr>
<tr>
<td>uint32_t time_data:</td>
</tr>
<tr>
<td>uint8_t write_block</td>
</tr>
<tr>
<td><strong>Return Value</strong></td>
</tr>
<tr>
<td><strong>Remarks</strong></td>
</tr>
</tbody>
</table>
Flowcharts

5.11.1 Main Processing

Figure 5.3 shows the Main Processing.

```
main
  Disable maskable interrupts
    I flag ← 0
  Port initialization
    port_init()
  Stop processing for active peripheral functions after a reset
    R_INIT_StopModule()
  Nonexistent port initialization
    R_INIT_NonExistentPort()

True/false determination of RTC time information and RTC initialization
  Determination of RTC initialization requirement
    check_RTC()
  Clock initialization
    R_INIT_Clock_an1713()

Debug LCD initialization
  Debug LCD initialization
    init_LCD()
  Debug LCD output
    Display_LCD()
    Displays "RTC" on the first line of the Debug LCD.
  Debug LCD output
    Display_LCD()
    Displays "00:00:00" on the second line of the Debug LCD.

Set warm start
  RSTSR1 register
    CWSF bit ← 1: Warm start
    I flag ← 1

Enable maskable interrupts

Backup request for RTC time information is generated?
  No
  Verifies whether the RTC time information backup request is generated in the RTC periodic interrupt.
  Yes

Backup processing for the RTC time information
  time_backup()
  Sets the RTC time information and sets bit 0 of the backup counter as the argument. The backup information is stored in DB0 or DB1 in the E2 DataFlash according to the argument.
```

Figure 5.3 Main Processing

5.11.2 Port Initialization

Figure 5.4 shows the Port Initialization.

```
port_init

Set the port output for LED0 and LED1
  PORT0.PODR register
    B3 bit ← 1: P03 (LED0): High level
    B5 bit ← 1: P05 (LED1): High level
  PORT0.PDR register
    B3 bit ← 1: P03 (LED0): Output
    B5 bit ← 1: P05 (LED1): Output

Set the Debug LCD data output
  PORT8.PODR register
    Bits B7 to B4 ← 0000b: P84 to P87 (DLCD): Outputs 0 as an initial value.
  PORT8.PDR register
    Bits B7 to B4 ← 1111b: P84 to P87 (DLCD): Output

Set the Debug LCD reset output and enable output
  PORTJ.PODR register
    B5 bit ← 0: PJ5 (DLCDRS): Outputs 0 as an initial value.
  PORTJ.PDR register
    B5 bit ← 1: PJ5 (DLCDRS): Output
  PORTF.PODR register
    B5 bit ← 0: PF5 (DLCDE): Outputs 0 as an initial value.
  PORTF.PDR register
    B5 bit ← 1: PF5 (DLCDE): Output
```

Figure 5.4 Port Initialization
5.11.3 Determination of RTC Initialization Requirement

Figure 5.5 and Figure 5.6 show the Determination of RTC Initialization Requirement.

Figure 5.5 Determination of RTC Initialization Requirement (1/2)
Compares the RTC alarm time registers (year, month, day, hour, minute, and second) with the backup time information when the alarm function is enabled, and verifies the information.

Verifies whether the VBATT circuit is fully charged at the previous backup timing. If the circuit is not fully charged, the RTC is initialized even if it is within 10 days from the previous backup. If the circuit is fully charged at the precious backup timing, the VBATT charge flag is reset (status: not enough charged) and the state is held for 15 minutes from this point.

**Figure 5.6 Determination of RTC Initialization Requirement (2/2)**
5.11.4 Setting the Validity of the Alarm Time Information

Figure 5.7 shows the Setting the Validity of the Alarm Time Information.

```
Set_register_enable

Clear the information for verification

Second data is valid?
  Yes
  The second data is to be verified

  No
  RSECAR.ENB: 1: This register value is compared with the RSECCNT value.

Minute data is valid?
  Yes
  The minute data is to be verified

  No
  RMINAR.ENB: 1: This register value is compared with the RMINCNT value.

Hour data is valid?
  Yes
  The hour data is to be verified

  No
  RHRAR.ENB: 1: This register value is compared with the RHRCNT value.

Day-of-week data is valid?
  Yes
  The day-of-week data is to be verified

  No
  RWKAR.ENB: 1: This register value is compared with the RWKCNT value.

Day data is valid?
  Yes
  The day data is to be verified

  No
  RDAYAR.ENB: This register value is compared with the RDAYCNT value.

Month data is valid?
  Yes
  The month data is to be verified

  No
  RMONAR.ENB: This register value is compared with the RMONCNT value.

Year data is valid?
  Yes
  The year data is to be verified

  No
  RYRAREN.ENB: The RYRAR register value is compared with the RYRCNT value.

return
```

Figure 5.7 Setting the Validity of the Alarm Time Information
5.11.5 Determination of the Backup Time Information for Use

Figure 5.8 shows the Determination of the Backup Time Information for Use.

![Flowchart: Determination of the Backup Time Information for Use]

**Figure 5.8 Determination of the Backup Time Information for Use**
5.11.6 Verification of the RTC Time Information Value

Figure 5.9 shows the Verification of the RTC Time Information Value.

![Flowchart Diagram]

**Figure 5.9 Verification of the RTC Time Information Value**
5.11.7 Comparison of the RTC Time Information with the Backup Time Information

Figure 5.10 shows the Comparison of the RTC Time Information with the Backup Time Information.

```
True/False Determination of RTC Time Information on Cold Start

check_RTC_flash

Determine the flash data for use

Backup year data < RTC year data\(^1\)?

Yes

Backup year data = RTC year data\(^1\)?

No

Backup month data < RTC month data\(^1\)?

Yes

Backup month data = RTC month data\(^1\)?

No

Backup day data < RTC day data\(^1\)?

Yes

No

return(RTC_RUN_RES)  return(RTC_INIT_RES)

Note:
1. Data in the RTC time information (in the RAM).
```

Figure 5.10 Comparison of the RTC Time Information with the Backup Time Information
5.11.8 Verifying the Range of the RTC Time Information

Figure 5.11 shows Verifying the Range of the RTC Time Information.

Note:
1. Data in the RTC time information (in the RAM).
5.11.9 Reading the RTC Time Information

Figure 5.12 shows the Reading the RTC Time Information.

![Diagram of RTC Time Information Reading]

**Figure 5.12 Reading the RTC Time Information**
5.11.10 Clock Initialization

Figure 5.13 shows the Clock Initialization.

Notes:
1. Select the sub-clock pattern according to the user system.
2. The PLL oscillation setting is not necessary when PLL is not used.
3. Set an appropriate value according to the user system.
### 5.11.11 Sub-Clock Oscillation Setting

Figure 5.14 shows the Sub-Clock Oscillation Setting.

![Sub-Clock Oscillation Setting Diagram]

- **CGC_subclk_as_RTC_an1713**
  - Yes: Sub-clock is used as the RTC count source?
  - No: Specify the RTC count source
  - No: RTC initialization requested?
  - Yes: Stop the sub-clock oscillator
  - Wait for 5 cycles of the sub-clock using CMT0
  - Set the sub-clock oscillation stabilization wait time using CMT0
  - Wait for the sub-clock oscillation stabilization wait time (2.6 sec.)
  - Initialization when using the RTC
  - Set the SOSTP bit to 1 since the sub-clock is not used as the system clock
  - SOSTP bit = 1

- **SOSCCR register**
  - 01h
  - SOSTP bit = 1: Sub-clock oscillator is stopped.

- **RCR3 register**
  - RTCEN bit = 0: Sub-clock oscillator is stopped.

- **RCR4 register**
  - RCKSEL bit = 0: The sub-clock is selected.

- **SOSCWTCR register**
  - 00h
  - SSTS[4:0] bits = 00000b: Wait for 2 cycles (approx. 61 μs).

- **SOSCCR register**
  - 00h
  - SOSTP bit = 0: Sub-clock oscillator is operating.

- **Processing when not using the sub-clock as the system clock**
  - (RTC count source is the sub-clock)
  - Function: no_use_subclk_as_sysclk

- **Notes:**
  1. Set the sub-clock oscillator wait control register to a value greater than or equal to the sub-clock oscillator start-up time recommended by the crystal/ceramic resonator manufacturer minus the minimum value of the sub-clock oscillation stabilization wait offset time (1.8 sec.).
  2. The time includes six clocks of the RTC count source.

---

**Figure 5.14 Sub-Clock Oscillation Setting**
5.11.12 RTC Initialization

Figure 5.15 shows the RTC Initialization.

- **RTC initialization requested?**
  - No
  - Set the RCR3 register
    - RCR3 register
      - RTCEN bit ← 1: Sub-clock oscillator is operating.
  - Stop counters and the prescaler
    - RCR2 register ← RCR2 register & 7Eh
      - START bit = 0: Counters and prescaler are stopped
  - RTC reset completed?
    - No
      - Reset the RCR3 register
        - RCR3 register ← 00h
          - RTCDV bit ← 110b: Standard driving ability for the clock
        - Set the date and time to Tuesday, January 1, 2013, 00:00:00
          - RSECCNT register ← 00h: 00 is set to the second counter.
            - RMINCNT register ← 00h: 00 is set to the minute counter.
            - RHRNCT register ← 00h: 00 is set to the hour counter.
            - RWKCNT register ← 02h: 2 (Tuesday) is set to the day-of-week counter.
            - RDAYCNT register ← 01h: 01 is set to the date counter.
            - RMONCNT register ← 01h: 01 is set to the month counter.
            - RYRCNT register ← 0013h: 13 is set to the year counter.
      - Specify the time error adjustment
        - RADJ register
          - ADJ(0) bits ← 00000000b: Adjustment value not set
          - PMADJ(4:0) bits ← 00b: Adjustment is not performed.
          - Set the RTC 1-second periodic interrupt
            - RCR1 register ← 44h
              - AIE bit = 0: An alarm interrupt request is disabled.
              - CIE bit = 0: A carry interrupt request is disabled.
              - PIE bit = 1: A periodic interrupt request is enabled.
              - PER[3:0] bits = 1110b: A periodic interrupt is generated every 1 second.
          - Set the automatic adjustment
            - RCR2 register ← 30h
              - START bit = 0: Year, month, day-of-week, date, hour, minute, second, and
                64-Hz counters, and prescaler are stopped.
              - RTOCE bit ← 0: RTCOUT output disabled.
              - AADJE bit = 1: Automatic adjustment is enabled.
              - AADJP bit = 1: The RADJ(ADJ)(5:0) setting value is adjusted from the count
                value of the prescaler every 10 seconds.
              - HR24 bit = 0: The RTC operates in 12-hour mode.
          - Turn on LED0
            - PORT0.PODR register
              - B3 bit ← 0: P03(LED0): Low level
          - Initialize the backup information
            - Time information (time_data) ← 00:00:00: Time information
            - VBAT register
              - Time information (time_data) ← 00:00:00: Time information
            - flash_write ← RTCC_BACKUP_INIT(2)
          - Specify the RTC periodic interrupt settings
            - IRHI3 register
              - IR flag ← 0: Interrupt request [IR[RTC,PRD]] is cleared.
            - IPRI3 register
              - [PR[RTC,PRD]] is level 1.
            - IER0B register
              - IEN bit ← 1: Interrupt request enable [IEN[RTC,PRD]] is enabled.
          - Start the RTC count operation
            - RCR2 register
              - START bit = 1: Year, month, day-of-week, date, hour, minute, second, and
                64-Hz counters, and prescaler operate normally.
              - Second data: time_data ← (unit32_t)(time.second & 0x00000007F)
              - Minute data: time_data ← time_data | (time.minute & 0x00000007F) << 8
              - Hour data: time_data ← time_data | (time.hour & 0x00000003F) << 16
            - Read the RTC time information
              - return

Figure 5.15 RTC Initialization
5.11.13  RTC Periodic Interrupt Handling

Figure 5.16 shows the RTC Periodic Interrupt Handling.

Figure 5.16  RTC Periodic Interrupt Handling
5.11.14 Backup Processing for the RTC Time Information

Figure 5.17 and Figure 5.18 show the Backup Processing for the RTC Time Information.

![Backup Processing Diagram](image-url)
Obtains 10-byte data for verifying.

Compares the programming data with data for verifying.

Yes

Transfer the time information (10 bytes) to a buffer for reading

No

Programming data = Data for verifying?

All data verified?

Yes

Turn on LED1

Port0.PODR register

B5 bit ← 0; P05 (LED1): Low level

No

Change the programming mode to ‘preparing’

Clear the RTC time information backup request

flash_write ← RTC_BACKUP_NACK(0)

return

Figure 5.18  Backup Processing for the RTC Time Information (2/2)
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.1.80 (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>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Apr. 1, 2014</td>
<td>First edition issued</td>
</tr>
<tr>
<td>1.01</td>
<td>Nov. 6, 2015</td>
<td>Page 3 The specification was corrected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Page 27 A figure was corrected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Program The set value of OFS0 register and OFS1 register was corrected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Program The waiting time to read time counter after a reset was added.</td>
</tr>
</tbody>
</table>

All trademarks and registered trademarks are the property of their respective owners.
**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. **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 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 Microprocessing unit or Microcontroller unit products 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.
Notice

1. Descriptions of circuits, software, and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability for any losses or damages occurring as a result of your noncompliance with applicable laws and regulations.

3. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.

4. Renesas Electronics endeavors to improve the quality and reliability of its products. Semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, the control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.

5. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics assumes no liability for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.

6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The recommended applications for each Renesas Electronics product depend on the product’s quality grade, as indicated below.

   “Standard”: Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment, and industrial robots etc.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, and safety equipment etc.

Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or your third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.

7. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.

8. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.

9. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for malfunctions or damages occurring as a result of noncompliance with applicable laws and regulations.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.