RX63N Group, RX631 Group

Clock Synchronous SCIc Communication Using the DMAC

Abstract

This application note describes a method for clock synchronous serial communication using the RX63N and RX631’s DMA controller (DMAC) and serial communication interface (SCI).

Products

- RX63N Group, 177- and 176-pin versions, ROM capacity: 768 KB to 2 MB
- RX63N Group, 145- and 144-pin versions, ROM capacity: 768 KB to 2 MB
- RX63N Group, 100-pin version, ROM capacity: 768 KB to 2 MB
- RX631 Group, 177- and 176-pin versions, ROM capacity: 256 KB to 2 MB
- RX631 Group, 145- and 144-pin versions, ROM capacity: 256 KB to 2 MB
- RX631 Group, 100-pin version, ROM capacity: 256 KB to 2 MB

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

5. Software ............................................................................................................................................. 6
   5.1 Operational Overview .................................................................................................................. 7
      5.1.1 Transmit Operation ............................................................................................................... 7
      5.1.2 Reception Operation ............................................................................................................. 9
   5.2 File Composition ........................................................................................................................ 11
   5.3 Option-Setting Memory .............................................................................................................. 11
   5.4 Constants ................................................................................................................................... 12
   5.5 Variable ...................................................................................................................................... 12
   5.6 Functions .................................................................................................................................... 13
   5.7 Function Specifications .............................................................................................................. 14
   5.8 Flowcharts .................................................................................................................................. 17
      5.8.1 Main Processing ................................................................................................................. 17
      5.8.2 Port Initialization ............................................................................................................... 18
      5.8.3 Peripheral Function Initialization ......................................................................................... 19
      5.8.4 SCI9 Initialization .............................................................................................................. 20
      5.8.5 DMAC0 Initialization ........................................................................................................... 21
      5.8.6 DMAC1 Initialization ........................................................................................................... 22
      5.8.7 SCI9 Transmission and Reception Start ............................................................................ 23
      5.8.8 DMAC0 Transfer Complete Interrupt Handler ................................................................. 24
      5.8.9 DMAC1 Transfer Complete Interrupt Handler .................................................................... 24
      5.8.10 SCI9 Transmission Complete Interrupt Handler ............................................................ 25
      5.8.11 Group 12 Interrupt Handler (SCI9 Reception Error Interrupt) ....................................... 25

6. Sample Code .................................................................................................................................... 26

7. Reference Documents ......................................................................................................................... 26
1. Specifications

This sample program performs clock synchronous serial transmission and reception using the SCI module. The transmit data is stored in advance in a RAM transmit data storage area and is transferred using a DMAC. The receive data is transferred to the RAM receive data storage area using a DMAC. Serial communication is started when a falling edge is detected on the interrupt request pin (IRQ15).

- Transfer rate: 38,400 bps
- Communication format: 8 bits, LSB first
- Clock input/output: Clock output (master)
- Transmission/reception operation: Transmission and reception can be performed at the same time.

Table 1.1 lists the peripheral function used 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>SCI channel 9 (SCI9)</td>
<td>Clock synchronous serial transmission and reception</td>
</tr>
<tr>
<td>DMACA channel 0 (DMAC0)</td>
<td>Transfer of SCI receive data to RAM</td>
</tr>
<tr>
<td>DMACA channel 1 (DMAC1)</td>
<td>Transfer of RAM transmit data to the SCI module</td>
</tr>
<tr>
<td>IRQ15</td>
<td>Start trigger for serial transmission or reception</td>
</tr>
</tbody>
</table>

![Figure 1.1 Block Diagram](image-url)
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>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller used</td>
<td>R5F563NBDDFC (RX63N Group)</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>Main clock: 12 MHz&lt;br&gt;P LL: 192 MHz (Main clock divided by 1 and multiplied by 16)&lt;br&gt;System clock (ICLK): 96 MHz (PLL divided by 2)&lt;br&gt;Peripheral module clock B (PCLKB): 48 MHz (PLL divided by 4)</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>3.3 V</td>
</tr>
<tr>
<td>Integrated development environment</td>
<td>Renesas Electronics Corporation&lt;br&gt;High-performance Embedded Workshop Version 4.09.01</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>Compiler options&lt;br&gt;-cpu=rx600 -output=object=&quot;$(CONFIGDIR)$ (FILELEAF).obj&quot; -debug -nologo (The integrated development environment default settings are used.)</td>
</tr>
<tr>
<td>iodefine.h version</td>
<td>Version 1.50</td>
</tr>
<tr>
<td>Endian order</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 RX63N (Product number: R0K50563NC000BE)</td>
</tr>
</tbody>
</table>

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_RX63N)

The initialization functions from the above application note are used by the sample code in this application note. The revision number shown is the one used when this application note was written.

If there is a more recent version, use the latest version. Check the Renesas Electronics Corporation web site to verify and download the latest version.
4. Hardware

4.1 Hardware Configuration

Figure 4.1 shows a connection example.

![Connection Example Diagram](image)

**Figure 4.1 Connection Example**

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>P07/IRQ15</td>
<td>Input</td>
<td>Transmit/receive start switch input</td>
</tr>
<tr>
<td>PB5/SCK9</td>
<td>Output</td>
<td>SCI9 clock output</td>
</tr>
<tr>
<td>PB6/RXD9</td>
<td>Input</td>
<td>SCI9 receive data input</td>
</tr>
<tr>
<td>PB7/TXD9</td>
<td>Output</td>
<td>SCI9 transmit data output</td>
</tr>
</tbody>
</table>
5. Software

This sample code implements automatic SCI transmission and reception operations by using the DMAC. When the transmit/receive start switch is pressed a SCI transmit/receive operation is started.

If transmission is enabled and a TXI9 interrupt request occurs, it functions as a DMAC1 transfer request. DMAC1 transfers data in the transmit data storage area to the TDR register and that data is transmitted.

If a reception completes, an RXI9 interrupt request is generated and it functions as a DMAC0 transfer request. DMAC0 is used to transfer the receive data to the receive data storage area.

When the transfer of transmit data has been performed 256 times, a DMAC1 interrupt is generated. This disables the TXI9 interrupt and enables the TEI9 interrupt.

When the transfer of receive data has been performed 256 times, a DMAC0 interrupt is generated. This disables the RXI9 interrupt and sets the reception complete flag to 1. If the transmission complete flag is 1 at this time, SCI transmit/receive operations are disabled.

When 256 bytes of transmissions and 256 bytes of receptions have been completed, a TEI9 interrupt is generated. This disables the TEI9 interrupt and sets the transmission complete flag to 1. If the reception complete flag is 1 at this time, SCI transmit/receive operations are disabled.

The settings of the peripheral functions are listed below.

SC19

- Communication mode: Clock synchronous
- SCK9 pin: Internal clock output (master)
- Clock source: PCLKB/4
- Transfer rate: 38,400 bps (BRR register setting = (PCLKB/(8×2×38,400 bps)) -1)
- Transmit operation: Enabled
- Receive operation: Enabled
- Data transfer direction: LSB first
- Interrupts used: Transmission complete interrupt (TEI9)
  Transmit data empty interrupt (TXI9)
  Receive data full interrupt (RXI9)
  Receive error interrupt (ERI9)

DMAC0

- Start factor: RXI9 interrupt request
  The IR flag for the RXI9 interrupt is cleared to 0 at the start of transfer.
- Transfer source address: SC19.RDR register
- Transfer source address update mode: Fixed address
- Transfer destination address: RAM (start address of the receive data storage area)
- Transfer destination address update mode: Increment
- Transfer mode: Normal mode
- Data transfer size: 8 bits
- Transfer count: 256 transfers
- Interrupts used: Transfer complete interrupt (DMAC0I)
DMAC1

- Start factor: TXI9 interrupt request
  - The IR flag for the TXI9 interrupt is cleared to 0 at the start of transfer.
- Transfer source address: RAM (start address of the transmit data storage area)
- Transfer source address update mode: Increment
- Transfer destination address: SC19.TDR register
- Transfer destination address update mode: Fixed address
- Transfer mode: Normal mode
- Data transfer size: 8 bits
- Transfer count: 256 transfers
- Interrupts used: Transfer complete interrupt (DMAC1I)

IRQ15 input pin

- Detection method: Falling edge detection
- Digital filter: Enabled (sampling clock: PCLKB/8)
- Interrupts used: None

5.1 Operational Overview

5.1.1 Transmit Operation

(1) Initialization
  - After initialization, the sample code waits for a transmit/receive start switch input.

(2) Transmit/receive start switch input detection
  - When a transmit/receive start switch input is detected, the IRQ15 interrupt IR flag is set to 0. After verifying that transmission and reception have completed by checking the transmission complete flag and the reception complete flag, the transmission complete flag is set to 0 (transmission in progress). The DMAC1 transfer source address and transfer count are set and DMA transfers are enabled.
  - The SC19.SCR.TEIE, TIE, RIE, TE, and RE bits are all set to 1 at the same time to enable transmission and reception operations. The TXI9 interrupt IR flag is set to 1 by the SC19.TCR.TIE and TE bits being set to 1 at the same time.

(3) Data transfer start
  - When the TXI9 interrupt is enabled, DMAC1 is started and the TXI9 interrupt IR flag is set to 0. The first byte of transmit data is transferred from the RAM transmit data storage area to the SC19.TSR register.

(4) Data transmission start
  - Data is transferred from the SC19.TDR register to the SC19.TSR register, the TXI9 interrupt IR flag is set to 1, and the first byte of transmit data is output from the TXD9 pin. DMAC1 is started by the TXI9 interrupt request and the second byte of transmit data is transferred.

(5) DMAC1I interrupt
  - When the 256th data transfer completes, a DMAC1I interrupt request is generated. The TXI9 interrupt is disabled and the TEI9 interrupt is enabled during DMAC1I interrupt handling.

(6) TEI9 interrupt
  - Since the SC19.TDR register is not updated when the last bit of the 256th byte is transmitted, a TEI9 interrupt request is generated. During TEI9 interrupt handling, the TEI9 interrupt is disabled and the transmission complete flag is set to 1 (transmission complete). If the reception complete flag is 1 (reception complete), transmission and reception are disabled.
  - Execution is then repeated from step (2) above.

Figure 5.1 shows the timing chart for the transmission operation.
Figure 5.1 Transmission Operation Timing Chart
5.1.2 Reception Operation

(1) Initialization
After initialization, the sample code waits for a transmit/receive start switch input.

(2) Transmit/receive start switch input detection
When a transmit/receive start switch input is detected, the IRQ15 interrupt IR flag is set to 0. After verifying that transmission and reception have completed by checking the transmission complete flag and the reception complete flag, the reception complete flag is set to 0 (reception in progress). The DMAC0 transfer destination address and transfer count are set and DMA transfers are enabled.

The SCI9.SCR.TEIE, TIE, RIE, TE, and RE bits are all set to 1 at the same time to enable transmission and reception operations and the RXI9 interrupt.

(3) Data reception completion
When reception of the first byte of data completes, the data is transferred from the SCI9.RSR register to the SCI9.RDR register and the RXI9 interrupt IR flag is set to 1.

(4) Data transfer start
DMAC0 is started by the RXI9 interrupt request and the RXI9 interrupt IR flag is cleared to 0. The first byte of receive data is transferred from the SCI9.RDR register to the RAM receive data storage area.

(5) DMAC0I interrupt
When the 256th data transfer completes, a DMAC0I interrupt request is generated. During DMAC0I interrupt handling, the RXI9 interrupt is disabled and the reception complete flag is set to 1 (reception complete). If the transmission complete flag is 1 (transmission complete), transmission and reception are disabled. Execution is then repeated from step (2) above.

Figure 5.2 shows the timing chart for the transmission operation.
<table>
<thead>
<tr>
<th>Event/State</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit/receive start switch input pin</td>
<td>“H”</td>
<td>Set to 0 by the acceptance of the interrupt</td>
</tr>
<tr>
<td></td>
<td>“L”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“1”</td>
<td></td>
</tr>
<tr>
<td>IRQ15 interrupt IR flag</td>
<td>“0”</td>
<td></td>
</tr>
<tr>
<td>Reception complete flag</td>
<td>“1”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“0”</td>
<td></td>
</tr>
<tr>
<td>SCI9.SCR.RE bit</td>
<td>“0”</td>
<td></td>
</tr>
<tr>
<td>SCI9.SCR.RIE bit</td>
<td>“0”</td>
<td></td>
</tr>
<tr>
<td>SCI9.SCR.RE bit</td>
<td>“1”</td>
<td></td>
</tr>
<tr>
<td>DMAC0.DMCNT. DTE bit</td>
<td>“0”</td>
<td>Set to 0 after 256 DMA transfers complete</td>
</tr>
<tr>
<td>DMAC0I interrupt IR flag</td>
<td>“0”</td>
<td>Set to 0 by the acceptance of the interrupt</td>
</tr>
<tr>
<td>RXI9 interrupt IEN bit</td>
<td>“1”</td>
<td></td>
</tr>
<tr>
<td>RXI9 interrupt IR flag</td>
<td>“0”</td>
<td>Set to 0 when a DMA transfer starts</td>
</tr>
<tr>
<td>SCI9.RDR register</td>
<td>00h</td>
<td>data0</td>
</tr>
<tr>
<td>SCI9.RSR register</td>
<td>Undefined</td>
<td>data0</td>
</tr>
</tbody>
</table>

**Figure 5.2 Reception Operation Timing Chart**

Transmit/receive start switch input pin: Switching between high ("H") and low ("L").

RIQ15 interrupt IR flag: Set to 0 when an interrupt occurs.

Reception complete flag: Set to 1 after 256 DMA transfers complete.

SCI9.SCR.RIE bit: Set to 1 when a receive interrupt occurs.

SCI9.SCR.RE bit: Set to 0 when a DMA transfer starts.

DMAC0I interrupt IR flag: Set to 0 by the acceptance of the interrupt.

RXI9 interrupt IEN bit: Set to 1 when an interrupt occurs.

RXI9 interrupt IR flag: Set to 0 when a DMA transfer starts.

SCI9.RDR register: Shows the DMA transfer status.

SCI9.RSR register: Shows the status of the RSR register.

Receive data input to the RXD9 pin: Data transferred to RAM.
Notes on Embedding the Sample Code in the User System

When embedding the sample code from this application note in the actual system, note the following:

- When operating in slave mode, if the interrupts used in this application are forced to wait for extended periods due to, for example, the handling of other interrupts, this code may not operate correctly.

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 5.1 Files Used in the Sample Code

<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>Stops peripheral modules that are operating</td>
<td></td>
</tr>
<tr>
<td></td>
<td>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>Initialization for ports that do not exist</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>Initialization for clock</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>Stops IWDT after a reset&lt;br&gt;Stops WDT after a reset</td>
</tr>
<tr>
<td>OFS1</td>
<td>FFFF FF8Bh to FFFF FF88h</td>
<td>FFFF FFFFh</td>
<td>Disables voltage monitoring 0 resets after a reset&lt;br&gt;Disables HOCO oscillation 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>MASTER</td>
<td>00h</td>
<td>Set value of the SCI9.SCR.CKE[1:0] bits: Internal clock (master)</td>
</tr>
<tr>
<td>SLAVE</td>
<td>02h</td>
<td>Set value of the SCI9.SCR.CKE[1:0] bits: External clock (slave)</td>
</tr>
<tr>
<td>SCI_CLK</td>
<td>MASTER</td>
<td>Set value of the SCI9.SCR.CKE[1:0] bits: Master mode selected</td>
</tr>
<tr>
<td>BUF_SIZE</td>
<td>256</td>
<td>Size of the transmission and reception data storage areas</td>
</tr>
<tr>
<td>DMAC_CNT</td>
<td>BUF_SIZE</td>
<td>DMAC transfer count</td>
</tr>
<tr>
<td>SW_ON</td>
<td>1</td>
<td>Switch input on state</td>
</tr>
<tr>
<td>SW_OFF</td>
<td>0</td>
<td>Switch input off state</td>
</tr>
</tbody>
</table>

5.5 Variable

Table 5.4 lists the global variables.

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>rcv_end_flag</td>
<td>Reception complete flag</td>
<td>main</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Reception in progress</td>
<td>Excep_DMAC_DMAC0I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Reception complete</td>
<td>Excep_SCI9_TEI9</td>
</tr>
<tr>
<td>unsigned char</td>
<td>trn_end_flag</td>
<td>Transmission complete flag</td>
<td>main</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Transmission in progress</td>
<td>Excep_DMAC_DMAC0I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Transmission complete</td>
<td>Excep_SCI9_TEI9</td>
</tr>
<tr>
<td>unsigned char</td>
<td>rcvbuf[BUF_SIZE]</td>
<td>Receive data storage area</td>
<td>dmac0_init</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sci9_start</td>
</tr>
<tr>
<td>unsigned char</td>
<td>trnbuf[BUF_SIZE]</td>
<td>Transmit data storage area</td>
<td>main</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dmac1_init</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sci9_start</td>
</tr>
</tbody>
</table>
## 5.6 Functions

Table 5.5 lists the 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>sci9_init</td>
<td>SCI9 initialization</td>
</tr>
<tr>
<td>dmac0_init</td>
<td>DMAC0 initialization</td>
</tr>
<tr>
<td>dmac1_init</td>
<td>DMAC1 initialization</td>
</tr>
<tr>
<td>sci9_start</td>
<td>Starts SCI9 transmission and reception</td>
</tr>
<tr>
<td>Excep_DMAC_DMAC0I</td>
<td>DMAC0 transfer complete interrupt handler</td>
</tr>
<tr>
<td>Excep_DMAC_DMAC1I</td>
<td>DMAC1 transfer complete interrupt handler</td>
</tr>
<tr>
<td>Excep_SCI9_TEI9</td>
<td>SCI9 transmission complete interrupt handler</td>
</tr>
<tr>
<td>Excep_ICU_GROUP12</td>
<td>Group 12 interrupt handler (SCI9 receive error interrupt)</td>
</tr>
</tbody>
</table>
5.7 Function Specifications
The following tables list the sample code function specifications.

<table>
<thead>
<tr>
<th>Function</th>
<th>Outline</th>
<th>Header</th>
<th>Declaration</th>
<th>Description</th>
<th>Arguments</th>
<th>Return Value</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>Main processing</td>
<td>None</td>
<td>void main(void)</td>
<td>After initialization, this function starts SCI9 transmission and reception operations when a transmit/receive start switch input is detected.</td>
<td>None</td>
<td>None</td>
<td>Transition to the module-stop state is not performed in the sample code. Refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.00 application note for details of this function.</td>
</tr>
<tr>
<td>port_init</td>
<td>Port initialization</td>
<td>None</td>
<td>void port_init(void)</td>
<td>Initializes the ports.</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>R_INIT_StopModule</td>
<td>Stop processing for active peripheral functions after a reset</td>
<td>r_init_stop_module.h</td>
<td>void R_INIT_StopModule(void)</td>
<td>Configures the setting to enter the module-stop state.</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>R_INIT_NonExistentPort</td>
<td>Nonexistent port initialization</td>
<td>r_init_non_existent_port.h</td>
<td>void R_INIT_NonExistentPort(void)</td>
<td>Initializes port direction registers for ports that does not exist in products with less than 176 pins.</td>
<td>None</td>
<td>None</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 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. Refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.00 application note for details of this function.</td>
</tr>
</tbody>
</table>
### 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 which uses PLL as the system clock without using the sub-clock. Refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.00 application note for details of this function.

### peripheral_init

**Outline**  
Peripheral function initialization

**Header**  
None

**Declaration**  
void peripheral_init(void)

**Description**  
Initializes the peripheral functions used.

**Arguments**  
None

**Return Value**  
None

### sci9_init

**Outline**  
SCI9 initialization

**Header**  
None

**Declaration**  
void sci9_init(void)

**Description**  
Initializes SCI9.

**Arguments**  
None

**Return Value**  
None

### dmac0_init

**Outline**  
DMAC0 initialization

**Header**  
None

**Declaration**  
void dmac0_init(void)

**Description**  
Initializes DMAC0.

**Arguments**  
None

**Return Value**  
None

### dmac1_init

**Outline**  
DMAC1 initialization

**Header**  
None

**Declaration**  
void dmac1_init(void)

**Description**  
Initializes DMAC1.

**Arguments**  
None

**Return Value**  
None
sci9_start

Outline  Starts SCI9 transmission and reception
Header   None
Declaration  void sci9_start(void)
Description  Starts the SCI9 transmission and reception operation.
Arguments  None
Return Value  None

Excep_DMAC_DMA0I

Outline  DMAC0 transfer complete interrupt handler
Header   None
Declaration  void Excep_DMAC_DMA0I(void)
Description  Disables the RXI9 interrupt and sets the reception complete flag. If the transmission complete flag is 1, it disables SCI9 transmission and reception.
Arguments  None
Return Value  None

Excep_DMAC_DMA1I

Outline  DMAC1 transfer complete interrupt handler
Header   None
Declaration  void Excep_DMAC_DMA1I(void)
Description  Disables the TXI9 interrupt and enables the TEI9 interrupt.
Arguments  None
Return Value  None

Excep_SCI9_TEI9

Outline  SCI9 transmission complete interrupt handler
Header   None
Declaration  void Excep_SCI9_TEI9(void)
Description  Disables the TEI9 interrupt and sets the transmission complete flag. If the reception complete flag is 1, it disables SCI9 transmission and reception.
Arguments  None
Return Value  None

Excep_ICU_GROUP12

Outline  Group 12 interrupt handler (SCI9 receive error interrupt)
Header   None
Declaration  void Excep_ICU_GROUP12 (void)
Description  Handles group 12 interrupts (SCI9 receive error interrupt).
Arguments  None
Return Value  None
Remarks  This sample code does not perform the SCI9 receive error handling; it simply executes an infinite loop. Add a program if required.
5.8 Flowcharts

5.8.1 Main Processing

Figure 5.3 shows the flowchart for the main processing.

```
main

<table>
<thead>
<tr>
<th>Disable maskable interrupts</th>
<th>I flag ← 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize ports</td>
<td></td>
</tr>
<tr>
<td>port_init()</td>
<td></td>
</tr>
<tr>
<td>After a reset, stop operating peripheral functions</td>
<td></td>
</tr>
<tr>
<td>R_INIT_StopModule()</td>
<td></td>
</tr>
<tr>
<td>Initialize nonexistent ports</td>
<td></td>
</tr>
<tr>
<td>R_INIT_NonExistenPort()</td>
<td></td>
</tr>
<tr>
<td>Initialize clocks</td>
<td></td>
</tr>
<tr>
<td>R_INIT_Clock()</td>
<td></td>
</tr>
<tr>
<td>Initialize peripheral functions</td>
<td></td>
</tr>
<tr>
<td>peripheral_init()</td>
<td></td>
</tr>
<tr>
<td>Enable maskable interrupts</td>
<td>I flag ← 1</td>
</tr>
<tr>
<td>Store transmit data in RAM</td>
<td>trnbuf[0] to trnbuf[255] ← 00h to FFh</td>
</tr>
<tr>
<td>Set transmission and reception complete flags</td>
<td>trn_end_flag ← 1</td>
</tr>
<tr>
<td>rcv_end_flag ← 1</td>
<td>:Transmission complete</td>
</tr>
<tr>
<td></td>
<td>:Reception complete</td>
</tr>
<tr>
<td>No Transmit/receive start switch input? Yes</td>
<td></td>
</tr>
<tr>
<td>Clear IRQ15 interrupt request</td>
<td>IR flag ← 0</td>
</tr>
<tr>
<td>Reads the IR079 register IR flag</td>
<td></td>
</tr>
<tr>
<td>:0 :No IRQ15 interrupt request</td>
<td></td>
</tr>
<tr>
<td>:1 :There is an IRQ15 interrupt request</td>
<td></td>
</tr>
<tr>
<td>Wait for transmission and reception complete</td>
<td></td>
</tr>
<tr>
<td>Reads the tm_end_flag tm_end_flag ← 0</td>
<td></td>
</tr>
<tr>
<td>:0 :Transmission in progress</td>
<td></td>
</tr>
<tr>
<td>:1 :Transmission complete</td>
<td></td>
</tr>
<tr>
<td>Reads the rcv_end_flag rcv_end_flag ← 0</td>
<td></td>
</tr>
<tr>
<td>:0 :Reception in progress</td>
<td></td>
</tr>
<tr>
<td>:1 :Reception complete</td>
<td></td>
</tr>
<tr>
<td>Clear transmission and reception complete flags</td>
<td></td>
</tr>
<tr>
<td>Start SCI9 transmission and reception sci9_start()</td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 5.3 Main Processing
5.8.2 Port Initialization

Figure 5.4 shows the flowchart for the port initialization.

![Port Initialization Flowchart]

**Figure 5.4 Port Initialization**

- **port_init**
- **Set port output data**
  - PORTB.PODR register
    - B7 bit ← 1 : PB7/TXD9 : High-level output
  - PORT0.PDR register
    - B7 bit ← 0 : P07/IRQ15 : Input
  - PORTB.PDR register
    - B5 bit ← 0 : PB5/SCK9 : Input
    - B6 bit ← 0 : PB6/RXD9 : Input
    - B7 bit ← 1 : PB7/TXD9 : Output
  - PORT3.PMR register
    - B2 bit ← 0 : P07/IRQ15 : Used as a general-purpose I/O port
  - PORTB.PMR register
    - B5 bit ← 0 : PB5/SCK9 : Used as a general-purpose I/O port
    - B6 bit ← 0 : PB6/RXD9 : Used as a general-purpose I/O port
    - B7 bit ← 0 : PB7/TXD9 : Used as a general-purpose I/O port
- **Set port direction**
- **Set port mode**
- **return**
5.8.3 Peripheral Function Initialization

Figure 5.5 shows the flowchart for the peripheral function initialization.

```
peripheral_init

Enable writing to related registers

PRCR register ← A502h
PRC1 bit = 1

Clear module stop state

MSTPCRC register
MSTPC26 bit ← 0
:Clears the SCI9 module stop state.

Clear module stop state

MSTPCRA register
MSTPA28 bit ← 0
:Clears the DMAC module stop state.

Disable writing to related registers

PRCR register ← A500h
PRC1 bit = 0

Initialize SCI9
sci9_init()

Initialize DMAC0
dmac0_init()

Initialize DMAC1
dmac1_init()

Enable DMAC start

DMAST register ← 01h
DMST bit = 1

Disable IRQ15 interrupt request

IER09 register
IEN7 bit ← 0

Disable IRQ15 digital filter

IRQFLTE1 register
FLTEN15 bit ← 0

Set up IRQ15 sampling clock

IRQFTC1 register
FCLKSEL15[1:0] bits ← 01b
: PCLKB/8

Set IRQ15 port

PWPR register
B0WI bit ← 0
: Enables writing the PFSWE bit.
PWPR register
PFSWE bit ← 1
: Enables writing the PFS register.
P07PFS register
ISEL bit ← 1
: Sets up use as the IRQ15 input pin.
PWPR register
PWPR register
B0WI bit ← 0
: Disables writing the PFS register.
PFSWE bit ← 1
: Disables writing the PFSWE bit.

Set IRQ15 detection method

IRQCR15 register ← 04h
IRQMD[1:0] bits = 01b
:Falling edge

Clear IRQ15 interrupt request

IR079 register
IR flag ← 0

Enable IRQ15 digital filter

IRQFLTE1 register
FLTEN15 bit ← 1

return
```

Figure 5.5 Peripheral Function Initialization
### 5.8.4 SCI9 Initialization

Figure 5.6 shows the flowchart for the SCI9 initialization.

#### SCI9 Initialization

1. **Disable SCI9 interrupt requests**
   - `IER0E` register
   - `IEN2` bit ← 0: Disables ICU.GROUP12 (SCI9.ERI9) interrupt requests.
   - `GEN12` register
   - `EN9` bit ← 0: Disables SCI9.ERI9 (GROUP12) interrupt requests.
   - `IER1E` register
   - `IEN1` bit ← 0: Disables SCI9.RXI9 interrupt requests.
   - `SCI9.SCR` register
   - `IEN2` bit ← 0: Disables SCI9.TXI9 interrupt requests.
   - `IEN3` bit ← 0: Disables SCI9.TEI9 interrupt requests.

2. **Disable transmission, reception, and interrupt requests**
   - `SCI9.SCR` register ← 00h
   - `TEIE` bit = 0: Disables TEI interrupt requests.
   - `RE` bit = 0: Disables serial reception operation.
   - `TE` bit = 0: Disables serial transmission operation.
   - `RIE` bit = 0: Disables RXI and ERI interrupt requests.
   - `TIE` bit = 0: Disables TXI interrupt requests.

3. **Set up I/O port functions**
   - `SCI9.SCR` register
   - `CKE[1:0]` bits ← SCK_CLK: Internal clock (The SCK9 pin is used as the clock output pin)
   - `SCI9.SIMR1` register
   - `IICM` bit ← 0: Serial interface mode
   - `SCI9.SPMR` register
   - `CKPH` bit ← 0: No clock phase delay
   - `CKPOL` bit ← 0: No clock polarity inversion
   - `SCI9.SMR` register ← 81h
   - `CKS[1:0]` bits = 01b: PCLKB/4
   - `CM` bit = 1: Operates in clock synchronous mode.
   - `SCI9.SCMR` register ← F2h
   - `SMIF` bit = 0: Serial communications interface mode
   - `SINV` bit = 0: No transmit or receive data inversion
   - `SDIR` bit = 0: LSB first

4. **Set bit rate**
   - `SCI9.BRR` register ← 77h: 77.125 = (48 MHz / (8 × 2 × 38400 bps)) - 1

5. **Set interrupt priority level**
   - `IPR241` register
   - `IPR[3:0]` bits ← 0001b: Sets the SCI9.RXI9, TXI9, and TEI9 interrupts to priority level 1.
   - `IPR114` register
   - `IPR[3:0]` bits ← 0001b: Sets the ICU.GROUP12 (SCI9.ERI9) interrupts to priority level 1.

---

**Figure 5.6 SCI9 Initialization**
5.8.5 DMAC0 Initialization

Figure 5.7 shows the flowchart for the DMAC0 initialization.

```
Figure 5.7   DMAC0 Initialization
```

```
dmac0_init

Disable DMA transfers
DMAC0.DMCNT register ← 00h
DTE bit = 1

IER18 register
IEN6 bit ← 0

DMAC0.DMAMD register ← 0080h
DARA[4:0] bits = 00000b: Does not set up an extended repeat area.
DM[1:0] bits = 10b: Increment
SARA[4:0] bits = 00000b: Does not set up an extended repeat area.
SM[1:0] bits = 00b: Fixed address

Set addressing mode
DMAC0.DMRSR0 register ← 241
DMRS[7:0] bits = 11011100b: Sets the DMAC0 start request (RXI9) vector number.

DMAC0.DMAMD register ← 0080h
DARA[4:0] bits = 00000b: Does not set up an extended repeat area.
DM[1:0] bits = 10b: Increment
SARA[4:0] bits = 00000b: Does not set up an extended repeat area.
SM[1:0] bits = 00b: Fixed address

Set up start factor
DMAC0.DMCSL register ← 00h
DISEL bit = 0: Clears to 0 the interrupt flag that was the start factor at transfer start.

Set transfer mode
DMAC0.DMTMD register ← 0001h
DCTG[1:0] bits = 01b: Peripheral module interrupts
SZ[1:0] bits = 00b: 8-bit transfers
MD[1:0] bits = 00b: Normal transfers

Set transfer source address
DMAC0.DMSAR register ← The address of the SCI9.RDR register

Set transfer destination address
DMAC0.DMDAR register ← The address of the rcvbuf[0]

Set transfer count
DMAC0.DMCRA register ← DMAC_CNT

Set interrupt priority level
IPR198 register
IPR[3:0] bits ← 0001b: Sets the DMAC0i interrupts to priority level 1.

Enable transfer complete interrupt
DMAC0.DMINT register ← 10h
DTIE bit = 1

Enable DMAC0i interrupt requests
IER18 register
IEN6 bit ← 1

Enable DMA transfers
DMAC0.DMCNT register ← 01h
DTE bit = 1

return
```
### 5.8.6 DMAC1 Initialization

Figure 5.8 shows the flowchart for the DMAC1 initialization.

![Flowchart for DMAC1 Initialization](image)

**Figure 5.8 DMAC1 Initialization**

- **dmac1_init**
  - Disable DMAC1 interrupt requests
    - IER18 register
      - IEN7 bit \(\leftarrow 0\)
  - Disable DMA transfers
    - DMAC1.DMCNT register \(\leftarrow 00h\)
      - DTE bit = 0
  - Set up start factor
    - DMRS1 register \(\leftarrow 242\)
      - DMRS[7:0] bits = 11011101b: Sets the DMAC1 start request (TXI9) vector number.
    - DMAC1.DMAMD register \(\leftarrow 8000h\)
      - DARA[4:0] bits = 00000b: Does not set up an extended repeat area.
      - DM[1:0] bits = 00b: Fixed address
      - SARA[4:0] bits = 00000b: Does not set up an extended repeat area.
      - SM[1:0] bits = 10b: Increment
  - Set addressing mode
    - DMAC1.DMTMD register \(\leftarrow 0001h\)
      - DCTG[1:0] bits = 01b: Peripheral module interrupts
      - SZ[1:0] bits = 00b: 8-bit transfers
      - MD[1:0] bits = 00b: Normal transfers
  - Set up clear as the start factor
    - DMAC1.DMCSL register \(\leftarrow 00h\)
      - DISEL bit = 0: Clears to 0 the interrupt flag that was the start factor at transfer start.
  - Set transfer source address
    - DMAC1.DMSAR register \(\leftarrow\) The address of the trnbuf[0]
  - Set transfer destination address
    - DMAC1.DMDAR register \(\leftarrow\) The address of the SCI9.TDR register
  - Set transfer count
    - DMAC1.DMCRA register \(\leftarrow\) DMAC_CNT
  - Set interrupt priority level
    - IPR199 register
      - IPR[3:0] bits \(\leftarrow 0001b\): Sets the DMAC1 interrupts to priority level 1.
  - Enable transfer complete interrupt
    - DMAC1.DMINT register \(\leftarrow 10h\)
      - DTIE bit = 1
  - Enable DMAC1 interrupt requests
    - IER18 register
      - IEN7 bit \(\leftarrow 1\)
  - Enable DMA transfers
    - DMAC1.DMCNT register \(\leftarrow 01h\)
      - DTE bit = 1
  - return
5.8.7 SCI9 Transmission and Reception Start

Figure 5.9 shows the flowchart for the SCI9 transmission and reception start.

![Flowchart of SCI9 Transmission and Reception Start](image)

- **sci9_start**
- Set DMAC0 transfer destination address
- DMAC0.DMDAR register ← The address of the rcvbuf[0]
- Set DMAC0 transfer count
- DMAC0.DMCRA register ← DMAC_CNT
- Enable DMA transfers
- DMAC0.DMCNT register ← 01h
  - DTE bit = 1
- Set DMAC1 transfer source address
- DMAC1.DMSAR register ← The address of the trnbuf[0]
- Set DMAC1 transfer count
- DMAC1.DMCRA register ← DMAC_CNT
- Enable DMA transfers
- DMAC1.DMCNT register ← 01h
  - DTE bit = 1
- Start SCI9 transmission and reception
  - SCI9.SCR register ← SCI9.SCR register | F4h
    - TEIE bit = 1: Enables TEI interrupt requests.
    - RE bit = 1: Enables serial reception operation.
    - TE bit = 1: Enables serial transmission operation.
    - RIE bit = 1: Enables RXI and ERI interrupt requests.
    - TIE bit = 1: Enables TXI interrupt requests.
- Set up I/O port functions
- PORTB.PMR register
  - B7 bit ← 1: PB7/TXD9: Used as a peripheral function pin.
- Enable ERI9 interrupt request
- GEN12 register
  - EN9 bit ← 1: Enables SCI9.ERI9 (GROUP12) interrupt requests.
- Enable GROUP12 interrupt request
- IER0E register
  - IEN2 bit ← 1: Enables ICU.GROUP12 (SCI9.ERI9) interrupt requests.
- Enable RXI9 interrupt request
- IER1E register
  - IEN1 bit ← 1: Enables SCI9.RXI9 interrupt requests.
- Enable TXI9 interrupt request
- IER1E register
- return

Figure 5.9 SCI9 Transmission and Reception Start
5.8.8 DMAC0 Transfer Complete Interrupt Handler
Figure 5.10 shows the flowchart for the DMAC0 transfer complete interrupt handler.

![Flowchart for DMAC0 Transfer Complete Interrupt Handler]

Note: 1. After writing a value to the RIE bit, verify that the written value can be read.

5.8.9 DMAC1 Transfer Complete Interrupt Handler
Figure 5.11 shows the flowchart for the DMAC1 transfer complete interrupt handler.

![Flowchart for DMAC1 Transfer Complete Interrupt Handler]

Note: 1. After writing a value to the TIE bit, verify that the written value can be read.
5.8.10 SCI9 Transmission Complete Interrupt Handler

Figure 5.12 shows the flowchart for the SCI9 transmission complete interrupt handler.

\[
\text{Excep\_SCI9\_TEI9}
\]

Check the source of the interrupt request

An interrupt occurred

No interrupt occurred

Reads the SCI9.SCR register.

TEIE bit

\(0\) : Enables TEI interrupt request

\(1\) : Disables TEI interrupt request

Reads the SCI9.SSR register.

TEND flag

\(0\) : Character transmission in progress

\(1\) : Character transmission completed

Disable TEI9 interrupt requests

IER1E register

IEN3 bit \(\leftarrow 0\) : Disables SCI9.TEI9 interrupt requests.

Disable interrupt requests*1

SCI9.SCR register

TEIE bit \(\leftarrow 0\) : Disables TEI interrupt requests.

Set transmission complete flag

tm_end_flag \(\leftarrow 1\) : Transmission completed.

Check the source of the interrupt request

IR = 1

Check IR flag

IR = 0

Reads the IR243 register.

IR flag

\(0\) : No interrupt request occurred

\(1\) : An interrupt request occurred

Set I/O port functions

PORTB.PMR register

B7 bit \(\leftarrow 0\) : Used as the PB7/TXD9 general-purpose I/O port.

Disable SCI9 transmission and reception

SCI9.SCR register \(\leftarrow\) SCI9.SCR register \& 03h

TEIE bit = 0 : Disables TEI interrupt requests.

RE bit = 0 : Disables serial reception operation.

TE bit = 0 : Disables serial transmission operation.

RIE bit = 0 : Disables RXI and ERI interrupt requests.

TIE bit = 0 : Disables TXI interrupt requests.

Figure 5.12 SCI9 Transmission Complete Interrupt Handler

5.8.11 Group 12 Interrupt Handler (SCI9 Reception Error Interrupt)

Figure 5.13 shows the flowchart for the Group 12 interrupt handler (SCI9 reception error interrupt).

\[
\text{Excep\_ICU\_GROUP12}
\]

Handle SCI9 reception error

This sample code does not perform the SCI9 receive error handling; it simply executes an infinite loop. Add a program if required.

Figure 5.13 Group 12 Interrupt Handler (SCI9 Reception Error Interrupt)
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/
<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Apr. 05, 2013</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 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.
**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 whatsoever for any damages incurred by you resulting from errors or omissions from the information included herein.

3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.

5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High-Quality". The recommended applications for each Renesas Electronics product depends on the products 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 implantation 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 third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.

6. 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.

7. Although 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 the 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, fire 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.

8. 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 damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

9. 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.

10. 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 regulations and follow the procedures required by such laws and regulations.

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

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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