RX210, RX21A, and RX220 Groups
Asynchronous SCIc Transmission/Reception Using DTCa

Abstract
This application note describes how to perform asynchronous transmission/reception using the serial communications interface (SCI) with the data transfer controller (DTC) in the RX210, RX21A, and RX220 Groups.

Products
RX210, RX21A, and RX220 Groups

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

1. Specifications ............................................................................................................................ 3
2. Operation Confirmation Conditions ............................................................................................. 4
3. Reference Application Notes ....................................................................................................... 4
4. Hardware ..................................................................................................................................... 5
   4.1 Hardware Configuration ........................................................................................................ 5
   4.2 Pins Used ........................................................................................................................... 5
5. Software ...................................................................................................................................... 5
   5.1 Operation Overview ............................................................................................................. 7
      5.1.1 Transmit Operation ...................................................................................................... 7
      5.1.2 Receive Operation ...................................................................................................... 9
   5.2 Section Composition ........................................................................................................... 11
   5.3 File Composition ............................................................................................................... 11
   5.4 Option-Setting Memory ..................................................................................................... 11
   5.5 Constants .......................................................................................................................... 11
   5.6 Structure/Union List .......................................................................................................... 12
   5.7 Variables .......................................................................................................................... 13
   5.8 Functions .......................................................................................................................... 13
   5.9 Function Specifications ...................................................................................................... 14
      5.10 Flowcharts .................................................................................................................... 17
         5.10.1 Main Processing ...................................................................................................... 17
         5.10.2 Port Initialization ................................................................................................... 18
         5.10.3 Peripheral Function Initialization ........................................................................... 18
         5.10.4 SCI Initialization ..................................................................................................... 19
         5.10.5 DTC Initialization .................................................................................................. 20
         5.10.6 IRQ Initialization .................................................................................................... 21
         5.10.7 SCI Transmission/Reception Start Processing ...................................................... 22
         5.10.8 SCI Receive Data Full Interrupt Handling ............................................................ 23
         5.10.9 SCI Transmit Data Empty Interrupt Handling ....................................................... 23
         5.10.10 SCI Transmit End Interrupt Handling ................................................................. 24
         5.10.11 SCI Receive Error Interrupt Handling ................................................................. 24
6. Applying This Application Note to the RX21A or RX220 Group ................................................. 25
7. Sample Code ............................................................................................................................. 26
8. Reference Documents ............................................................................................................... 26
1. Specifications

This document describes performing asynchronous serial communication using the SCI.

Transmit data is prestored in the transmit data storage area in the RAM and transmitted using the DTC. Receive data is stored in the RAM's receive data storage area using the DTC.

Serial transmission/reception starts when a falling edge is detected on the IRQ1 interrupt request pin.

- Bit rate: 38400 bps
- Data length: 8-bit, LSB first
- Stop bit: 1 bit
- Parity: None
- Hardware flow control: None

Table 1.1 lists the Peripheral Functions and Their Applications and Figure 1.1 shows a 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 1 (SCI1)</td>
<td>Asynchronous serial transmission/reception</td>
</tr>
<tr>
<td>DTCa (DTC)</td>
<td>Transfer data received by SCI1 to the RAM</td>
</tr>
<tr>
<td></td>
<td>Transfer transmit data in the RAM to SCI1</td>
</tr>
<tr>
<td>IRQ1</td>
<td>Start trigger for serial transmission/reception</td>
</tr>
</tbody>
</table>

**Figure 1.1 Block Diagram**
2. Operation Confirmation Conditions

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU used</td>
<td>R5F52108ADFP (RX210 Group)</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>- Main clock: 20 MHz</td>
</tr>
<tr>
<td></td>
<td>- PLL: 100 MHz (main clock divided by 2 and multiplied by 10)</td>
</tr>
<tr>
<td></td>
<td>- System clock (ICLK): 50 MHz (PLL divided by 2)</td>
</tr>
<tr>
<td></td>
<td>- Peripheral module clock B (PCLKB): 25 MHz (PLL divided by 4)</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>5.0 V</td>
</tr>
<tr>
<td>Integrated development</td>
<td>Renesas Electronics Corporation</td>
</tr>
<tr>
<td>environment</td>
<td>High-performance Embedded Workshop Version 4.09.01</td>
</tr>
<tr>
<td>C compiler</td>
<td>Renesas Electronics Corporation</td>
</tr>
<tr>
<td></td>
<td>C/C++ Compiler Package for RX Family V.1.02 Release 01</td>
</tr>
<tr>
<td>Compile options</td>
<td>-cpu=rx200 -output=obj=&quot;$(CONFIGDIR)$$(FILELEAF).obj&quot; -debug -nologo</td>
</tr>
<tr>
<td></td>
<td>The integrated development environment default settings are used.</td>
</tr>
<tr>
<td>iodefille.h version</td>
<td>Version 1.2A</td>
</tr>
<tr>
<td>Endian</td>
<td>Little endian</td>
</tr>
<tr>
<td>Operating mode</td>
<td>Single-chip mode</td>
</tr>
<tr>
<td>Processor mode</td>
<td>Supervisor mode</td>
</tr>
<tr>
<td>Sample code version</td>
<td>Version 1.00</td>
</tr>
<tr>
<td>Board used</td>
<td>Renesas Starter Kit for RX210 (product part number: R0K505210C000BE)</td>
</tr>
</tbody>
</table>

3. Reference Application Notes

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

- RX210 Group Initial Setting Rev. 2.00 (R01AN1002EJ)
- RX21A Group Initial Setting Rev. 1.10 (R01AN1486EJ)
- RX220 Group Initial Setting Rev. 1.10 (R01AN1494EJ)

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

4.1 Hardware Configuration

Figure 4.1 shows a Connection Example.

![Connection Example Diagram](image)

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>P31/IRQ1</td>
<td>Input</td>
<td>Switch input to start transmission and reception</td>
</tr>
<tr>
<td>P15/RXD1</td>
<td>Input</td>
<td>Receive data input to SCI1</td>
</tr>
<tr>
<td>P16/TXD1</td>
<td>Output</td>
<td>Transmit data output from SCI1</td>
</tr>
</tbody>
</table>

5. Software

In the sample code, DTC is used for automatically processing SCI1 data transmission and reception. SCI1 data transmission and reception are started by pressing a switch.

When data transmission is enabled, a TXI1 interrupt request is generated, and this becomes the DTC activation source. The DTC is used to transmit data from the transmit data storage area to the TDR register, and then the data is transmitted from a pin.

When data reception is completed, an RXI1 interrupt request is generated, and this becomes the DTC activation source. The DTC is used to transfer receive data to the receive data storage area.

After transmit data has been transferred 256 times, a TXI1 interrupt occurs. At this point, the TXI1 interrupt is disabled and the TEI1 interrupt is enabled.

After receive data has been transferred 256 times, an RXI1 interrupt occurs. At this point, SCI1 reception and the RXI1 interrupt are disabled, and the receive end flag becomes 1.

After 256 bytes of data have been transmitted, a TEI1 interrupt occurs. At this point, SCI1 transmission and the TEI1 interrupt are disabled, and the transmit end flag becomes 1.
Settings for the peripheral functions used are listed below.

**SCI**
- Serial communication mode: Asynchronous mode
- Transfer rate: 38400 bps
- Clock source: PCLKB
- Data length: 8 bits
- Stop bit: 1 bit
- Parity: None
- Data transfer direction: LSB first
- Interrupts used: Receive error interrupt (ERI1), receive data full interrupt (RXI1), transmit data empty interrupt (TXI1), transmit end interrupt (TEI1)

**DTC**
- Activation source: TXI1 or RXI1 interrupt request
- DTC address mode: Full-address mode

**Setting of DTC transfer triggered by the TXI1 interrupt request**
- Transfer mode: Normal transfer
- Transfer source addressing mode: Increment the SAR register after transfer
- Transfer source address: RAM (start address of the transmit data storage area)
- Transfer destination addressing mode: Address in the DAR register is fixed
- Transfer destination address: SCI1.TDR register
- Data transfer size: 8 bits
- Number of transfer operations: 256
- Chain transfer: Disabled
- Interrupts: When data transfers have been completed for the specified number of times, an interrupt request to the CPU is generated

**Setting of DTC transfer triggered by the RXI1 interrupt request**
- Transfer mode: Normal transfer
- Transfer source addressing mode: Address in the SAR register is fixed
- Transfer source address: SCI1.RDR register
- Transfer destination addressing mode: Increment the DAR register after transfer
- Transfer destination address: RAM (start address of the receive data storage area)
- Data transfer size: 8 bits
- Number of transfer operations: 256
- Chain transfer: Disabled
- Interrupts: When data transfers have been completed for the specified number of times, an interrupt request to the CPU is generated

**INT1 input pin**
- Detection method: Falling edge
- Digital filter: Disabled
- Interrupts: Not used
5.1 Operation Overview

5.1.1 Transmit Operation

1. Initialization
   After initialization, wait for input from a switch to start transmission/reception.

2. Detecting input from a switch to start transmission/reception
   When input from a switch to start transmission/reception is detected, set the IR flag for the IRQ1 interrupt to 0. Read the transmit end flag and receive end flag. If transmission and reception have both ended, then set the transmit end flag to 0 (transmitting). Set the transfer source address and the number of transfer operations for the DTC, and enable DTC activation. Set the SCI1.SCR.TEIE, TIE, RIE, TE, and RE bits to 1 at the same time to enable transmission and reception. By setting the SCI1.SCR.TIE and TE bits to 1 at the same time, the IR flag for the TXI1 interrupt becomes 1.

3. Starting data transfer
   After the TXI1 interrupt is enabled, the DTC is activated and the IR flag for the TXI1 interrupt becomes 0. The first byte of transmit data is transferred from the transmit data storage area in the RAM to the SCI1.TDR register.

4. Starting data transmission
   The data is transferred from the SCI1.TDR register to the SCI1.TSR register, the IR flag for the TXI1 interrupt becomes 1, and the first byte of transmit data is output from the TXD1 pin. The DTC is activated by a TXI1 interrupt request, and the second byte of transmit data is transferred.

5. TXI1 interrupt
   After the 256th data transfer has ended, the CPU accepts a TXI1 interrupt request. In the TXI1 interrupt handling, disable the TXI1 interrupt and enable the TEI1 interrupt.

6. TEI1 interrupt
   When the last bit of the 256th byte is transmitted, the SCI1.TDR register is not updated, so a TEI1 interrupt request is generated. In the TEI1 interrupt handling, disable transmission and the TEI1 interrupt, and set the transmit end flag to 1 (transmission ended). Operation is repeated from step 2 above.
Figure 5.1 shows the Timing Diagram of the Transmit Operation.

**Figure 5.1 Timing Diagram of the Transmit Operation**

```
<table>
<thead>
<tr>
<th>Event</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmission/reception start switch input pin</td>
<td>High, Low</td>
</tr>
<tr>
<td>IR flag for the IRQ1 interrupt</td>
<td>0, 1</td>
</tr>
<tr>
<td>Transmit end flag</td>
<td>0, 1</td>
</tr>
<tr>
<td>SCI1.SCR.TE bit</td>
<td>0, 1</td>
</tr>
<tr>
<td>SCI1.SCR.TIE bit</td>
<td>0, 1</td>
</tr>
<tr>
<td>SCI1.SCR.TIE bit</td>
<td>0, 1</td>
</tr>
<tr>
<td>SCI1.TDR register</td>
<td>FFh, 00h, 01h, 02h, FEh, FFh</td>
</tr>
<tr>
<td>SCI1.TSR register</td>
<td>Undefined, 00h, 01h, FEh, FFh</td>
</tr>
<tr>
<td>Transmit data output from the TXD1 pin</td>
<td>00h, 01h, FEh, FFh</td>
</tr>
<tr>
<td>RAM → TDR (DTC transfer)</td>
<td>FFh</td>
</tr>
<tr>
<td>TDR → TSR (automatic transfer)</td>
<td></td>
</tr>
<tr>
<td>Does not become 1 because the SCI1.SCR.TIE bit is 0</td>
<td></td>
</tr>
<tr>
<td>SCI1.TDR register</td>
<td>FFh</td>
</tr>
<tr>
<td>SCI1.TSR register</td>
<td>FEh</td>
</tr>
<tr>
<td>Transmit data output from the TXD1 pin</td>
<td>FFh</td>
</tr>
<tr>
<td>256 bytes transmitted</td>
<td></td>
</tr>
</tbody>
</table>

Becomes 0 at completion of 256th DTC transfer
Becomes 0 when an interrupt is accepted
Become 0 when DTC transfer starts
Does not become 1 because the SCI1.SCR.TIE bit is 0
```
5.1.2 Receive Operation

1. Initialization
   After initialization, wait for input from a switch to start transmission/reception.

2. Detecting input from a switch to start transmission/reception
   When input from a switch to start transmission/reception is detected, set the IR flag for the IRQ1 interrupt to 0.
   Read the transmit end flag and receive end flag. If transmission and reception have both ended, then set the receive
   end flag to 0 (receiving). Set the transfer destination address and the number of transfer operations for the DTC, and
   enable DTC activation. Set the SCI1.SCR.TEIE, TIE, RIE, TE, and RE bits to 1 at the same time to enable transmission and reception, and enable the RXI1 interrupt.

3. Data reception end
   After the first byte of data is received, transfer the data from the SCI1.RSR register to the SCI1.RDR register, and
   the IR flag for the RXI1 interrupt becomes 1.

4. Starting data transfer
   The DTC is activated by an RXI1 interrupt request, and the IR flag for the RXI1 interrupt becomes 0. Then transfer the first byte of receive data from the SCI1.RDR register to the RAM's receive data storage area.

5. RXI1 interrupt
   After the 256th data transfer has ended, the CPU accepts an RXI1 interrupt request. In the RXI1 interrupt handling, disable reception and the RXI1 interrupt, and set the receive end flag to 1 (reception ended). Operation is repeated from step 2 above.
Figure 5.2 shows the Timing Diagram of the Receive Operation.

![Timing Diagram of the Receive Operation](image_url)

**Figure 5.2 Timing Diagram of the Receive Operation**

Notes on Implementing the Sample Code Into the User System

Note that the following may occur when implementing the sample code accompanying this application note into the user system.

- When user defined interrupt handling makes the interrupts defined in the sample code wait for a lengthy amount of time, the sample code may operate erroneously.
5.2 Section Composition

Table 5.1 lists the section information that is changed in the sample code.

For details on adding/changing or deleting sections, refer to the latest RX Family, C/C++ Compiler Package User’s Manual.

Table 5.1 Section Information Changed in the Sample Code

<table>
<thead>
<tr>
<th>Section Name</th>
<th>Change</th>
<th>Address</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTC_SECTION</td>
<td>Addition</td>
<td>0000 3000h</td>
<td>DTC vector table</td>
</tr>
</tbody>
</table>

5.3 File Composition

Table 5.2 lists the Files Used in the Sample Code. Files generated by the integrated development environment should not be listed 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</td>
</tr>
<tr>
<td>r_init_stop_module.c</td>
<td>Stop processing for active peripheral functions after a reset</td>
</tr>
<tr>
<td>r_init_stop_module.h</td>
<td>Header file for r_init_stop_module.c</td>
</tr>
<tr>
<td>r_init_non_existent_port_init.c</td>
<td>Nonexistent port initialization</td>
</tr>
<tr>
<td>r_init_non_existent_port_init.h</td>
<td>Header file for r_init_non_existent_port_init.c</td>
</tr>
<tr>
<td>r_init_clock.c</td>
<td>Clock initialization</td>
</tr>
<tr>
<td>r_init_clock.h</td>
<td>Header file for r_init_clock.c</td>
</tr>
</tbody>
</table>

5.4 Option-Setting Memory

Table 5.3 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

Table 5.3 Option-Setting Memory Configured in the Sample Code

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

5.5 Constants

Table 5.4 lists the Constants Used in the Sample Code.

Table 5.4 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>BUF_SIZE</td>
<td>256</td>
<td>Size of the transmit data or receive data storage area</td>
</tr>
<tr>
<td>DTC_CNT</td>
<td>BUF_SIZE</td>
<td>Number of DTC transfers</td>
</tr>
</tbody>
</table>
5.6 Structure/Union List

Figure 5.3 shows the Structure/Union Used in the Sample Code.

```c
/* **** DTC transfer information **** */
#pragma bit_order left /* Bit field order: The bit field members are allocated from upper bits */
#pragma unpack /* Boundary alignment value for structure members: Alignment by member type */
struct st_dtc_full
{
    union
    {
        unsigned long LONG;
        struct
        {
            unsigned long MRA_MD :2;
            unsigned long MRA_SZ :2;
            unsigned long MRA_SM :2;
            unsigned long :2;
            unsigned long MRB_CHNE :1;
            unsigned long MRB_CHNS :1;
            unsigned long MRB_DISEL :1;
            unsigned long MRB_DTS :1;
            unsigned long MRB_DM :2;
            unsigned long :2;
            unsigned long :16;
        } BIT;
    } MR;
    void * SAR;
    void * DAR;
    struct
    {
        unsigned long CRA:16;
        unsigned long CRB:16;
    } CR;
};
#pragma packoption /* End of specification for the boundary alignment value for structure members */
#pragma bit_order /* End of specification for the bit field order */
```

Figure 5.3 Structure/Union Used in the Sample Code
5.7 Variables
Table 5.5 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>trn_end_flag</td>
<td>Transmit end flag</td>
<td>main, Excep_SCI1_TEI1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Transmitting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Transmission ended</td>
<td></td>
</tr>
<tr>
<td>unsigned char</td>
<td>rcv_end_flag</td>
<td>Receive end flag</td>
<td>main, Excep_SCI1_RXI1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Receiving</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Reception ended</td>
<td></td>
</tr>
<tr>
<td>unsigned char</td>
<td>trnbuf[BUF_SIZE]</td>
<td>Transmit data storage area</td>
<td>main, dtc_init, sci1_start</td>
</tr>
<tr>
<td>unsigned char</td>
<td>rcvbuf[BUF_SIZE]</td>
<td>Receive data storage area</td>
<td>dtc_init, sci1_start</td>
</tr>
<tr>
<td>struct st_dtc_full</td>
<td>dtc_info_rxi1</td>
<td>DTC transfer information for RXI1</td>
<td>dtc_init, sci1_start</td>
</tr>
<tr>
<td>struct st_dtc_full</td>
<td>dtc_info_txi1</td>
<td>DTC transfer information for TXI1</td>
<td>dtc_init, sci1_start</td>
</tr>
<tr>
<td>void *</td>
<td>dtc_vect_table[256]</td>
<td>DTC vector table</td>
<td>dtc_init</td>
</tr>
</tbody>
</table>

5.8 Functions
Table 5.6 lists the Functions.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>Main processing</td>
</tr>
<tr>
<td>port_init</td>
<td>Port initialization</td>
</tr>
<tr>
<td>R_INIT_StopModule</td>
<td>Stop processing for active peripheral functions after a reset</td>
</tr>
<tr>
<td>R_INIT_NonExistentPort</td>
<td>Nonexistent port initialization</td>
</tr>
<tr>
<td>R_INIT_Clock</td>
<td>Clock initialization</td>
</tr>
<tr>
<td>peripheral_init</td>
<td>Peripheral function initialization</td>
</tr>
<tr>
<td>sci1_init</td>
<td>SCI1 initialization</td>
</tr>
<tr>
<td>dtc_init</td>
<td>DTC initialization</td>
</tr>
<tr>
<td>irq_init</td>
<td>IRQ initialization</td>
</tr>
<tr>
<td>sci1_start</td>
<td>SCI1 transmission/reception start processing</td>
</tr>
<tr>
<td>Excep_SCI1_RXI1</td>
<td>SCI1 receive data full interrupt handling</td>
</tr>
<tr>
<td>Excep_SCI1_TXI1</td>
<td>SCI1 transmit data empty interrupt handling</td>
</tr>
<tr>
<td>Excep_SCI1_TEI1</td>
<td>SCI1 transmit end interrupt handling</td>
</tr>
<tr>
<td>Excep_SCI1_ERI1</td>
<td>SCI1 receive error interrupt handling</td>
</tr>
</tbody>
</table>
5.9 Function Specifications

The following tables list the sample code function specifications.

### main

<table>
<thead>
<tr>
<th>Overview</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, SCI1 transmission/reception starts when the transmit/receive start switch input is detected.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return values</td>
<td>None</td>
</tr>
</tbody>
</table>

### port_init

<table>
<thead>
<tr>
<th>Overview</th>
<th>Port initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void port_init(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Ports are initialized.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return values</td>
<td>None</td>
</tr>
</tbody>
</table>

### R_INIT_StopModule

<table>
<thead>
<tr>
<th>Overview</th>
<th>Stop processing for active peripheral functions after a reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_init_stop_module.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_INIT_StopModule(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Performs settings to enter the module stop state.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return values</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>Transition to the module stop state is not performed in the sample code. For details on this function, refer to the Initial Setting application note for the product used.</td>
</tr>
</tbody>
</table>

### R_INIT_NonExistentPort

<table>
<thead>
<tr>
<th>Overview</th>
<th>Nonexistent port initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>non_existent_port_init.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_INIT_NonExistentPort (void)</td>
</tr>
<tr>
<td>Description</td>
<td>Initializes port direction registers for ports that do not exist in products with less than 100 pins.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return values</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>The number of pins in the sample code is set for the 100-pin package (PIN_SIZE=100). After this function is called, when writing in byte units to the PDR and PODR registers which have nonexistent ports, set the corresponding bits for nonexistent ports as follows: set the I/O select bits in the PDR registers to 1 and set the output data store bits in the PODR registers to 0. For details on this function, refer to the Initial Setting application note for the product used.</td>
</tr>
</tbody>
</table>
### R_INIT_Clock

<table>
<thead>
<tr>
<th>Overview</th>
<th>Clock initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_init_clock.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_INIT_Clock(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Initializes clocks.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return values</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>In the sample code, processing is performed so the system clock is used as the PLL clock, and the sub-clock is not used. For details on this function, refer to the Initial Setting application note for the product used.</td>
</tr>
</tbody>
</table>

### peripheral_init

<table>
<thead>
<tr>
<th>Overview</th>
<th>Peripheral function initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void peripheral_init (void)</td>
</tr>
<tr>
<td>Description</td>
<td>Initializes peripherals functions that are used.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return values</td>
<td>None</td>
</tr>
</tbody>
</table>

### sci1_init

<table>
<thead>
<tr>
<th>Overview</th>
<th>SCI1 initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void sci1_init(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Initializes SCI1.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return values</td>
<td>None</td>
</tr>
</tbody>
</table>

### dtc_init

<table>
<thead>
<tr>
<th>Overview</th>
<th>DTC initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void dtc_init(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Initializes the DTC.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return values</td>
<td>None</td>
</tr>
</tbody>
</table>
irq_init

Overview    IRQ initialization
Header      None
Declaration  void irq_init(void)
Description  Initializes IRQ1.
Arguments    None
Return values None

sci1_start

Overview    SCI1 transmission/reception start processing
Header      None
Declaration  void sci1_start(void)
Description  Starts SCI1 transmission/reception.
Arguments    None
Return values None

Excep_SCI1_RXI1

Overview    SCI1 receive data full interrupt handling
Header      None
Declaration  void Excep_SCI1_RXI1(void)
Description  Reception is disabled, the RXI1 interrupt is disabled, and the receive end flag is set.
Arguments    None
Return values None

Excep_SCI1_TXI1

Overview    SCI1 transmit data empty interrupt handling
Header      None
Declaration  void Excep_SCI1_TXI1(void)
Description  Disables the TXI1 interrupt and enables the TEI1 interrupt.
Arguments    None
Return values None

Excep_SCI1_TEI1

Overview    SCI1 transmit end interrupt handling
Header      None
Declaration  void Excep_SCI1_TEI1(void)
Description  Transmission is disabled, the TEI1 interrupt is disabled, and the transmit end flag is set.
Arguments    None
Return values None

Excep_SCI1_ERI1

Overview    SCI1 receive error interrupt handling
Header      None
Declaration  void Excep_SCI1_ERI1(void)
Description  Performs SCI1 reception error processing.
Arguments    None
Return values None
Remarks      SCI1 reception error processing is not performed in the sample code (infinite loop). Add processing as needed.
5.10 Flowcharts

5.10.1 Main Processing

Figure 5.4 shows the Main Processing.

![Flowchart of Main Processing](image-url)

---

**Figure 5.4 Main Processing**
5.10.2 Port Initialization

Figure 5.5 shows the Port Initialization.

```
port_init
|
Set port output data
|
PORT1.PODR register
B6 bit ← 1: P16/TXD1: High output
|
Set the port direction
|
PORT1.PDR register
B5 bit ← 0: P15/RXD1: Input
B6 bit ← 1: P16/TXD1: Output
PORT3.PDR register
B1 bit ← 0: P31/IRQ1: Input
|
Set the port mode
|
PORT1.PMR register
B5 bit ← 0: P15/RXD1: Use pin as general I/O port
B6 bit ← 0: P16/TXD1: Use pin as general I/O port
PORT3.PMR register
B1 bit ← 0: P31/IRQ1: Use pin as general I/O port
|
return
```

Figure 5.5 Port Initialization

5.10.3 Peripheral Function Initialization

Figure 5.6 shows the Peripheral Function Initialization.

```
peripheral_init
|
Enable writing to associated registers
|
PRCR register ← A502h
PRC1 bit = 1
|
Cancel the module stop state
|
MSTPCRB register
MSTPB30 bit ← 0: SCI1 module stop state is canceled
|
Cancel the module stop state
|
MSTPCRA register
MSTPA28 bit ← 0: DTC module stop state is canceled
|
Disable writing to associated registers
|
SCI1 initialization
sci1_init()
|
DTC initialization
dtc_init()
|
IRQ initialization
irq_init()
|
return
```

Figure 5.6 Peripheral Function Initialization
5.10.4 SCI1 Initialization

Figure 5.7 shows the SCI1 Initialization.

```
sci1_init

Disable SCI1 interrupt requests
IER1B register
IEN2 bit ← 0: SCI1.ERI1 interrupt request is disabled
IEN3 bit ← 0: SCI1.RXI1 interrupt request is disabled
IEN4 bit ← 0: SCI1.TXI1 interrupt request is disabled
IEN5 bit ← 0: SCI1.TEI1 interrupt request is disabled

Disable transmission, reception,
and the interrupt request (1)
SCI1.SCR register ← 00h
TEIE bit = 0: TEI interrupt request is disabled
RE bit = 0: Serial reception is disabled
TE bit = 0: Serial transmission is disabled
RIE bit = 0: RXI and ERI interrupt requests are disabled
TIE bit = 0: TXI interrupt request is disabled

Set the I/O port functions
PWPR register
B0WI bit ← 0: Writing to the PFSWE bit is enabled
PWPR register
PFSWE bit ← 1: Writing to the PFS register is enabled
P15PFS register ← 0Ah
Bits PSEL[3:0] = 1010b: P15 pin functions as RXD1
P16PFS register ← 0Ah
Bits PSEL[3:0] = 1010b: P16 pin functions as TXD1
PWPR register
PFSWE bit ← 0: Writing to the PFS register is disabled
PWPR register
B0WI bit ← 1: Writing to the PFSWE bit is disabled
PORT1.PMR register
B5 bit ← 1: P15/RXD1 pin used as I/O port for peripheral functions

Select the clock
SCI1.SMR register ← 00h
Bits CKS[1:0] = 00b: PCLK clock
MP bit = 0: Multi-processor communications function is disabled
STOP bit = 0: 1 stop bit
PE bit = 0: Parity bit addition/checking is not performed
CHR bit = 0: Selects 8 bits as the data length
CM bit = 0: Asynchronous mode
SCI1.SCMR register ← F2h
SMIF bit = 0: Serial communications interface mode
SINV bit = 0: TDR contents are transmitted as they are.
SDIR bit = 0: Transfer with LSB-first

Set the bit rate
SCI1.BRR register ← 19: 19.34 \times \left( \frac{25 MHz}{64 \times 2^{-1} \times 38400 bps} \right) - 1

Set the interrupt priority level
IPR218 register
Bits IPR[3:0] ← 0001b: Interrupt priority level for SCI1.RXI1, TXI1, and TEI1 is level 1

Clear the interrupt request
IR219 register
IR flag ← 0: SCI1.RXI1 interrupt request not generated
IR220 register
IR flag ← 0: SCI1.TXI1 interrupt request not generated

return
```

Note:
1. After writing to the SCR register, read the register to confirm that the correct value was written.

Figure 5.7 SCI1 Initialization
5.10.5 DTC Initialization

Figure 5.8 shows the DTC Initialization.

- **dtc_init**
  - DTCST register ← 00h
    - DTCSR bit = 0: DTC module stop
  - DTCCCR register
    - RRS bit ← 0: Transfer data read is not skipped
  - Set DTC address mode
    - DTCADMOD register
      - SHORT bit ← 0: Full-address mode
  - Set the RX1 DTC transfer data
    - dtc_info_rx1.MR.LONG ← 0000 0000h: Initialized by 0
    - dtc_info_rx1.MR.BIT.MRA_MD ← 00b: Normal transfer mode
    - dtc_info_rx1.MR.BIT.MRA_SZ ← 00b: Byte transfer
    - dtc_info_rx1.MR.BIT.MRA_SM ← 00b: Address in the SAR register is fixed
    - dtc_info_rx1.MR.BIT.MRB_DM ← 0: An interrupt request to the CPU is generated when specified data transfer is completed
    - dtc_info_rx1.MR.BIT.MRB_CHNE ← 0: Chain transfer is disabled
    - dtc_info_rx1.MR.BIT.MRB_DISEL ← 0: An interrupt request to the CPU is generated when specified data transfer is completed
    - dtc_info_rx1.SAR ← SCI1.RDR register address
    - dtc_info_rx1.DAR ← rcvbuf address
    - dtc_info_rx1.CR.CRA ← DTC_CNT: DTC transfer count
    - dtc_info_rx1.CR.CRB ← 0000h: Not used for normal transfer mode

  - Set the TX1 DTC transfer data
    - dtc_info_tx1.MR.LONG ← 0000 0000h: Initialized by 0
    - dtc_info_tx1.MR.BIT.MRA_MD ← 00b: Normal transfer mode
    - dtc_info_tx1.MR.BIT.MRA_SZ ← 00b: Byte transfer
    - dtc_info_tx1.MR.BIT.MRA_SM ← 10b: SAR value is incremented after data transfer
    - dtc_info_tx1.MR.BIT.MRB_CHNE ← 0: Chain transfer is disabled
    - dtc_info_tx1.MR.BIT.MRB_DISEL ← 0: An interrupt request to the CPU is generated when specified data transfer is completed
    - dtc_info_tx1.MR.BIT.MRB_DM ← 00b: Address in the DAR register is fixed
    - dtc_info_tx1.SAR ← trnbuf address
    - dtc_info_tx1.DAR ← SCI1.TDR register address
    - dtc_info_tx1.CR.CRA ← DTC_CNT: DTC transfer count
    - dtc_info_tx1.CR.CRB ← 0000h: Not used for repeat transfer mode

  - Set the DTC transfer data address to the DTC vector
    - dtc_vect_table[219] ← dtc_info_rx1 address
    - dtc_vect_table[220] ← dtc_info_tx1 address

  - Set the base address
    - DTCVBR register ← dtc_vect_table address

**Figure 5.8 DTC Initialization**
5.10.6 IRQ Initialization

Figure 5.9 shows the IRQ Initialization.

```plaintext
    irq_init

  IER08 register
    IEN1 bit ← 0

  Set the IRQ1 port
    PWPR register
      BWI bit ← 0: Writing to the PFSWE bit is enabled
      PWPR register
      PFSWE bit ← 1: Writing to the PFS register is enabled
    PWPR register
    PWPR register
      PWPR register
    BWI bit ← 0: Writing to the PFSWE bit is disabled
    PWPR register

  Set the IRQ1 detection method
    IRQCR1 register ← 04h
    Bits IRQMD[1:0] = 01b: Falling edge

  Clear IRQ1 interrupt request
    IR065 register
    IR flag ← 0

  return
```

Figure 5.9 IRQ Initialization
5.10.7 SCI1 Transmission/Reception Start Processing

Figure 5.10 shows the SCI1 Transmission/Reception Start Processing.

```
sci1_start

Stop the DTC module
DTCST register ← 00h
DTCST bit ← 0: DTC module stop

Disable transfer data read skip
DTCCR register
RRS bit ← 0: Transfer data read is not skipped

Reset the RXI1 DTC transfer data
dtc_info_rxi1.DAR ← rcvbuf address
dtc_info_rxi1.CR.CRA ← DTC_CNT

Reset the TXI1 DTC transfer data
dtc_info_txi1.SAR ← trnbuf address
dtc_info_txi1.CR.CRA ← DTC_CNT

Enable transfer data read skip
DTCCR register
RRS bit ← 1: Transfer data read is skipped when vector numbers match

Enable DTC activation by an RXI1 and TXI1 interrupt request
DTCE219 register
DTCE bit ← 1
DTCE220 register
DTCE bit ← 1

Set the DTC module operation
DTCST register ← 01h
DTCST bit ← 1: DTC module start

Start SCI1 transmission/reception
SCI1.SCR register ← SCI1.SCR | F4h
RE bit ← 1: Serial reception is enabled
TE bit ← 1: Serial transmission is enabled
RIE bit ← 1: RXI and ERI interrupt requests are enabled
TIE bit ← 1: TXI interrupt request is enabled

Set the I/O port function
PORT1.PMR register
B6 bit ← 1: P16/TXD1 used as I/O port for peripheral functions

Enable the ERI1 interrupt request
IER1B register
IEN2 bit ← 1: SCI1.ERI1 interrupt request is enabled

Enable the RXI1 interrupt request
IER1B register
IEN3 bit ← 1: SCI1.RXI1 interrupt request is enabled

Enable the TXI1 interrupt request
IER1B register
IEN4 bit ← 1: SCI1.TXI1 interrupt request is enabled

return
```

Figure 5.10 SCI1 Transmission/Reception Start Processing
5.10.8 SCI1 Receive Data Full Interrupt Handling

Figure 5.11 shows the SCI1 Receive Data Full Interrupt Handling.

```plaintext
Excep_SCI1_RXI1

SCI1.SCR register
RE bit ← 0

Disable RXI1 interrupt request
IER1B register
IEN3 bit ← 0: SCI1.RXI1 interrupt request disabled

Disable ERI1 interrupt request
IER1B register
IEN2 bit ← 0: SCI1.ERI1 interrupt request disabled

Disable interrupt requests
SCI1.SCR register
RIE bit ← 0: RXI and ERI1 interrupt requests are disabled

Clear RXI1 interrupt request
IR219 register
IR flag ← 0: No SCI1.RXI1 interrupt request is generated

Set the receive end flag
rcv_end_flag ← 1: Reception ended

return
```

Note:
1. After writing to the RE and RIE bits, read the register to confirm that the correct value was written.

Figure 5.11 SCI1 Receive Data Full Interrupt Handling

5.10.9 SCI1 Transmit Data Empty Interrupt Handling

Figure 5.12 shows the SCI1 Transmit Data Empty Interrupt Handling.

```plaintext
Excep_SCI1_TXI1

IER1B register
IEN4 bit ← 0: SCI1.TXI1 interrupt request is disabled

Disable interrupt request
SCI1.SCR register
TIE bit ← 0: TXI interrupt request is disabled

Clear TXI1 interrupt request
IR220 register
IR flag ← 0: SCI1.TXI1 interrupt request is not generated

Enable TEI1 interrupt request
IER1B register
IEN5 bit ← 1: SCI1.TEI1 interrupt request is enabled

return
```

Note:
1. After setting the TIE bit, read the TIE bit and confirm it has the value set.

Figure 5.12 SCI1 Transmit Data Empty Interrupt Handling
5.10.10 SCI1 Transmit End Interrupt Handling

Figure 5.13 shows the SCI1 Transmit End Interrupt Handling.

- **Excep_SC11_TEI1**
  - Confirm source of interrupt request generation
  - Interrupt occurred
    - Read the SCI1.SCR register
      - TEIE bit = 0: TEI interrupt request is disabled
      - = 1: TEI interrupt request is enabled
    - Read the SCI1.SSR register
      - TEND flag = 0: A character is being transmitted
      - = 1: Character transfer has been completed
  - Interrupt did not occur
    - Set the I/O port function
      - PORT1.PMR register
        - B6 bit ← 0: P16/TXD1 pin used as general I/O port
    - Disable serial transmission (1)
      - SCI1.SCR register
        - TE bit ← 0: Serial transmission is disabled
    - Disable TEI1 interrupt request
      - IER1B register
        - IEN5 bit ← 0: SCI1.TEI1 interrupt request disabled
    - Disable interrupt request (1)
      - SCI1.SCR register ← SCI1.SCR register & 7Bh
        - TEIE bit = 0: TEI interrupt request is disabled
        - TIE bit = 0: TXI interrupt request is disabled
    - Set the transmit end flag
      - trn_end_flag ← 1: Transmission ended

- **IR flag = 1**
  - Read IR flag
    - IR flag = 0: No interrupt request is generated
    - = 1: An interrupt request is generated
  - **IR flag = 0**
    - **return**

**Note:**
1. After writing to the SCR register, read the register and confirm that the read value is the written value.

5.10.11 SCI1 Receive Error Interrupt Handling

Figure 5.14 shows the SCI1 Receive Error Interrupt Handling.

- **Excep_SC11_ERI1**
  - SCI1 reception error handling
    - SCI1 reception error handling is not performed in the sample code and enters an infinite loop. Add this function to the program where necessary.
  - **return**

**Figure 5.14 SCI1 Receive Error Interrupt Handling**
6. Applying This Application Note to the RX21A or RX220 Group

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

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

Note: • When using the RX21A Group, SCI0 and SCI12 are not available. Use SCI1, SCI5, SCI6, SCI8, or SCI9.
    When using the RX220 Group, SCI0, SCI8, and SCI12 are not available. Use SCI1, SCI5, SCI6, or SCI9.
7. Sample Code
Sample code can be downloaded from the Renesas Electronics website.

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

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

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

Website and Support
Renesas Electronics website
   http://www.renesas.com

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

**RX210, RX21A, and RX220 Groups Application Note**  
Asynchronous SCIc Transmission/Reception Using DTCa

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>June 14, 2013</td>
<td>—</td>
<td>First edition issued</td>
</tr>
<tr>
<td>1.01</td>
<td>July 1, 2014</td>
<td>1</td>
<td>Products: Added the RX21A and RX220 Groups.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>3. Reference Application Notes: Added the Initial Setting application notes for the RX21A and RX220 Groups.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14,15</td>
<td>Modified the description of reference application note in the following functions: R_INIT_StopModule, R_INIT_NonExistentPort, and R_INIT_Clock.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>6. Applying This Application Note to the RX21A or RX220 Group: Added.</td>
</tr>
</tbody>
</table>

All trademarks and registered trademarks are the property of their respective owners.
### General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

<table>
<thead>
<tr>
<th>1. Handling of Unused Pins</th>
<th>2. Processing at Power-on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.</td>
<td>The state of the product is undefined at the moment when power is supplied.</td>
</tr>
<tr>
<td>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.</td>
<td>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.</td>
</tr>
<tr>
<td>3. Prohibition of Access to Reserved Addresses</td>
<td>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.</td>
</tr>
<tr>
<td>Access to reserved addresses is prohibited.</td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>4. Clock Signals</td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>5. Differences between Products</td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>
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 in 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 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 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 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 ratings, 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 environmental 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 in accordance with 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.

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 products 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 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.

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.

Notice

"Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

© 2014 Renesas Electronics Corporation. All rights reserved.

Colophon 4.0