RX63N Group, RX631 Group
Asynchronous SCIc Transmission/Reception Using DMACA

Abstract
This application note describes how to perform asynchronous transmission/reception using the serial communications interface (SCI) with the DMA controller (DMAC) in the RX63N Group, RX631 Group.

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

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

1. Specifications ..................................................................................................................................... 3
2. Operation Confirmation Conditions .................................................................................................... 4
3. Reference Application Note ................................................................................................................ 4
4. Hardware ............................................................................................................................................ 5
   4.1 Hardware Configuration ............................................................................................................... 5
   4.2 Pins Used ..................................................................................................................................... 5
5. Software ............................................................................................................................................. 6
   5.1 Operation Overview ..................................................................................................................... 7
   5.1.1 Transmitting .......................................................................................................................... 7
   5.1.2 Receiving ................................................................................................................................ 9
   5.2 File Composition ........................................................................................................................ 11
   5.3 Option-Setting Memory .............................................................................................................. 11
   5.4 Constants ................................................................................................................................... 12
   5.5 Variables .................................................................................................................................... 12
   5.6 Functions .................................................................................................................................... 12
   5.7 Function Specifications .............................................................................................................. 13
   5.8 Flowcharts .................................................................................................................................. 16
   5.8.1 Main Processing ................................................................................................................. 16
   5.8.2 Port Initialization ................................................................................................................. 17
   5.8.3 Peripheral Function Initialization ........................................................................................ 18
   5.8.4 SCI9 Initialization ............................................................................................................... 19
   5.8.5 DMAC0 Initialization ........................................................................................................... 20
   5.8.6 DMAC1 Initialization ........................................................................................................... 21
   5.8.7 SCI9 Transmission/Reception Start ................................................................................... 22
   5.8.8 DMAC0 Transfer End Interrupt Handling ............................................................................ 23
   5.8.9 DMAC1 Transfer End Interrupt Handling ............................................................................ 23
   5.8.10 SCI9 Transmit End Interrupt Handling .............................................................................. 24
   5.8.11 Group 12 Interrupt Handling (SCI9 Receive Error Interrupt) ............................................. 24
6. Sample Code .................................................................................................................................... 25
7. Reference Documents ...................................................................................................................... 25
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 DMAC. Receive data is stored in the RAM's receive data storage area using the DMAC.

Serial transmission/reception starts when a falling edge is detected on the IRQ15 interrupt request pin.

- Bit rate: 38,400 bps
- Communication format: 8-bit length, 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 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>SCIc channel 9 (SCI9)</td>
<td>Asynchronous serial transmission/reception</td>
</tr>
<tr>
<td>DMACA channel 0 (DMAC0)</td>
<td>Transfer data received by SCI9 to the RAM</td>
</tr>
<tr>
<td>DMACA channel 1 (DMAC1)</td>
<td>Transfer transmit data in the RAM to SCI9</td>
</tr>
<tr>
<td>IRQ15</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 2.1  Operation Confirmation Conditions

<table>
<thead>
<tr>
<th>Item</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU used</td>
<td>R5F563NBDDFC (RX63N Group)</td>
</tr>
<tr>
<td>Operating frequencies</td>
<td>• Main clock: 12 MHz</td>
</tr>
<tr>
<td></td>
<td>• PLL clock: 192 MHz (main clock divided by 1 and multiplied by 16)</td>
</tr>
<tr>
<td></td>
<td>• System clock (ICLK): 96 MHz (PLL divided by 2)</td>
</tr>
<tr>
<td></td>
<td>• 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</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></td>
<td>Compile options</td>
</tr>
<tr>
<td></td>
<td>-cpu=rx600 -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>Iodfine.h version</td>
<td>Version 1.6A</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 RX63N (product part 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. 1.10 (R01AN1245EJ0110)

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

4.1 Hardware Configuration

Figure 4.1 shows a Connection Example.

![Connection Example Diagram](image)

Figure 4.1 Connection Example

4.2 Pins Used

Table 4.1 lists the Pins Used and Their Functions.

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

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P07/IRQ15</td>
<td>Input</td>
<td>Switch input to start transmission and reception</td>
</tr>
<tr>
<td>PB6/RXD9</td>
<td>Input</td>
<td>Receive data input to SCI9</td>
</tr>
<tr>
<td>PB7/TXD9</td>
<td>Output</td>
<td>Transmit data output from SCI9</td>
</tr>
</tbody>
</table>
5. Software

In the sample code, SCI9 transmission and reception are processed automatically using the DMAC. SCI9 data transmission and reception are started by pressing a switch.

When data transmission is enabled, the TXI9 interrupt request is generated which becomes the DMAC1 transfer request. Using DMAC1, data in the transmit data storage area is transferred to the TDR register and then transmitted.

When reception is complete, the RXI9 interrupt request is generated which becomes the DMAC0 transfer request. Using DMAC0, receive data is transferred to the receive data storage area.

After the transmit data is transferred 256 times, the DMAC1 interrupt is generated. At this point, the TXI9 interrupt is disabled and the TEI9 interrupt is enabled.

After the receive data is transferred 256 times, the DMAC0 interrupt is generated. At this point, SCI9 reception is disabled, the RXI9 interrupt is disabled, and the receive end flag becomes 1.

After 256 bytes of data are transmitted, the TEI9 interrupt is generated. At this point, SCI9 transmission and TEI9 interrupt are disabled, and the transmit end flag becomes 1.

Settings for the peripheral functions are listed below.

SCI9
- Communication mode: Asynchronous mode
- Clock source: PCLKB/4
- Communication speed: 38,400 bps \( \text{BRR register setting value} = \frac{\text{PCLKB}}{(64 \times 2 \times 38,400 \text{ bps})} - 1 \)
- Data length: 8 bits
- Stop bits: 1
- Parity: None
- Data transfer direction: LSB first
- Interrupts: Transmit end interrupt (TEI9), transmit data empty interrupt (TXI9), receive data full interrupt (RXI9), and receive error interrupt (ERI9) are used

DMAC0
- Activation source: RXI9 interrupt request. The IR flag for the RXI9 interrupt is cleared to 0 when transfer starts.
- Transfer source address: SCI9.RDR register
- Transfer source address update mode: Address fixed
- Transfer destination address: RAM (start address in the receive data storage area)
- Transfer destination address update mode: Increment
- Transfer mode: Normal transfer
- Data transfer size: 8 bits
- Number of transfers: 256
- Interrupts: Transfer end interrupt (DMAC0I) is used

DMAC1
- Activation source: TXI9 interrupt request. The IR flag for the TXI9 interrupt is cleared to 0 when transfer starts.
- Transfer source address: RAM (start address in the transmit data storage area)
- Transfer source address update mode: Increment
- Transfer destination address: SCI9.TDR register
- Transfer destination address update mode: Address fixed
- Transfer mode: Normal transfer
- Data transfer size: 8 bits
- Number of transfers: 256
- Interrupts: Transfer end interrupt (DMAC1I) is used

IRQ15 input pin
- Detection method: Falling edge
- Digital filter: Enabled (sampling clock: PCLKB/8)
- Interrupts: Not used
5.1 Operation Overview

5.1.1 Transmitting

(1) Initial setting
After the initial setting, the program waits for the switch input to start transmission and reception.

(2) Detecting transmit/receive start switch input
When the switch input to start transmission and reception is detected, the IR flag for the IRQ15 interrupt is set to 0. Determine the value of the transmit end flag and receive end flag to confirm that transmission and reception are complete. If completion is confirmed, the transmit end flag is set to 0 (transmitting). Set the DMAC1 transfer source address, set the number of transfers, and enable DMA transfer. Set bits SCI9.SCR.TEIE, TIE, RIE, TE, and RE to 1 simultaneously to enable transmission and reception. By setting bits SCI9.SCR.TIE and TE to 1 simultaneously, the IR flag for the TXI9 interrupt becomes 1.

(3) Start data transfer
When the TXI9 interrupt is enabled, DMAC1 is activated, and the IR flag for the TXI9 interrupt becomes 0. The first byte of transmit data is transferred from the transmit data storage area in the RAM to the SCI9.TDR register.

(4) Start data transmission
Data is transferred from the SCI9.TDR register to the SCI9.TSR register, the IR flag for the TXI9 interrupt becomes 1, and the first byte of transmit data is output from the TXD9 pin. The TXI9 interrupt request triggers DMAC1 activation, and the second byte of transmit data is transferred.

(5) DMAC1I interrupt
After the 256th byte of data is transferred, the DMAC1I interrupt request is generated. The TXI9 interrupt is disabled and the TEI9 interrupt is enabled in the DMAC1I interrupt handling.

(6) TEI9 interrupt
When the last bit in the 256th byte of data is transmitted, the SCI9.TDR register is not updated, so the TEI9 interrupt request is generated. Transmission is disabled and the TEI9 interrupt is disabled in the TEI9 interrupt handling. Then the transmit end flag is set to 1 (transmission ended).

This procedure is repeated starting from step (2) above.
Figure 5.1 shows the Timing Diagram When Transmitting Data.
5.1.2 Receiving

(1) Initial setting
After the initial setting, the program waits for the switch input to start transmission and reception.

(2) Detecting transmit/receive start switch input
When the switch input to start transmission and reception is detected, the IR flag for the IRQ15 interrupt becomes 0. Determine the value of the transmit end flag and receive end flag to confirm that transmission and reception are complete. If completion is confirmed, the receive end flag is set to 0 (receiving). Set the DMAC0 transfer destination address, set the number of transfers, and enable DMA transfer. Set bits SCI9.SCR.TEIE, TIE, RIE, TE, and RE to 1 simultaneously to enable transmission and reception, and enable the RXI9 interrupt.

(3) Complete data reception
After the first byte of data is received, data is transferred from the SCI9.RSR register to the SCI9.RDR register, and the IR flag for the RXI9 interrupt becomes 1.

(4) Start data transfer
The RXI9 interrupt request activates DMAC0, and the IR flag for the RXI9 interrupt becomes 0. The first byte of receive data is transferred from the SCI9.RDR register to the receive data storage area in the RAM.

(5) DMAC0I interrupt
After the 256th byte of data is transferred, the DMAC0I interrupt request is generated. Reception is disabled and the RXI9 interrupt is disabled in the DMAC0I interrupt handling. The receive end flag is set to 1 (reception ended). This procedure is repeated starting from step (2) above.
Figure 5.2 shows the Timing Diagram When Receiving Data.

![Timing Diagram When Receiving Data](image)

**Notes when incorporating the sample code into the user system**

When incorporating the sample codes accompanying this application note into the user system, note the following:

When an interrupt used in this application note is delayed for a prolonged time due to other interrupt handlers, the sample code may not be executed properly.
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>Stop processing for active peripheral functions after a reset</td>
<td></td>
</tr>
<tr>
<td>r_init_stop_module.h</td>
<td>Header file for r_init_stop_module.c</td>
<td></td>
</tr>
<tr>
<td>r_init_non_existent_port.c</td>
<td>Nonexistent port initialization</td>
<td></td>
</tr>
<tr>
<td>r_init_non_existent_port.h</td>
<td>Header file for r_init_non_existent_port.c</td>
<td></td>
</tr>
<tr>
<td>r_init_clock.c</td>
<td>Clock initialization</td>
<td></td>
</tr>
<tr>
<td>r_init_clock.h</td>
<td>Header file for r_init_clock.c</td>
<td></td>
</tr>
</tbody>
</table>

5.3 Option-Setting Memory

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

Table 5.2 Option-Setting Memory Configured in the Sample Code

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Addresses</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFS0</td>
<td>FFFF FF8Fh to FFFF FF8Ch</td>
<td>FFFF FFFFh</td>
<td>The IWDT is stopped after a reset. The WDT is stopped after a reset.</td>
</tr>
<tr>
<td>OFS1</td>
<td>FFFF FF8Bh to FFFF FF88h</td>
<td>FFFF FFFFh</td>
<td>The voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.</td>
</tr>
<tr>
<td>MDES</td>
<td>FFFF FF83h to FFFF FF80h</td>
<td>FFFF FFFFh</td>
<td>Little endian</td>
</tr>
</tbody>
</table>
5.4 Constants

Table 5.3 lists the Constants Used in the Sample Code.

Table 5.3 Constants Used in the Sample Code

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUF_SIZE</td>
<td>256</td>
<td>Transmit or receive data storage area size</td>
</tr>
<tr>
<td>DMAC_CNT</td>
<td>BUF_SIZE</td>
<td>Number of DMAC transfers</td>
</tr>
<tr>
<td>SW_ON</td>
<td>1</td>
<td>Switch input on</td>
</tr>
<tr>
<td>SW_OFF</td>
<td>0</td>
<td>Switch input off</td>
</tr>
</tbody>
</table>

5.5 Variables

Table 5.4 lists the Global Variables.

Table 5.4 Global Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>rcv_end_flag</td>
<td>Receive end flag</td>
<td>main</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Receiving</td>
<td>Excep_DMAC_DMAC0I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Reception ended</td>
<td>Excep_SCI9_TEI9</td>
</tr>
<tr>
<td></td>
<td>trn_end_flag</td>
<td>Transmit end flag</td>
<td>main</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Transmitting</td>
<td>Excep_DMAC_DMAC0I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Transmission ended</td>
<td>Excep_SCI9_TEI9</td>
</tr>
<tr>
<td></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></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.

Table 5.5 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>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>SCI9 transmission/reception start</td>
</tr>
<tr>
<td>Excep_DMAC_DMAC0I</td>
<td>DMAC0 transfer end interrupt handling</td>
</tr>
<tr>
<td>Excep_DMAC_DMAC1I</td>
<td>DMAC1 tranfer end interrupt handling</td>
</tr>
<tr>
<td>Excep_SCI9_TEI9</td>
<td>SCI9 transmit end interrupt handling</td>
</tr>
<tr>
<td>Excep_ICU_GROUP12</td>
<td>Group 12 interrupt handling (SCI9 receive error interrupt)</td>
</tr>
</tbody>
</table>
## 5.7 Function Specifications

The following tables list the specifications for the functions in the sample code.

### main

<table>
<thead>
<tr>
<th>Outline</th>
<th>Main processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void main(void)</td>
</tr>
<tr>
<td>Description</td>
<td>After the initial settings, when the switch input to start transmission and reception is detected, SCI9 transmission and reception are started.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### port_init

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

### R_INIT_StopModule

<table>
<thead>
<tr>
<th>Outline</th>
<th>Stop processing for active peripheral functions after a reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_init_stop_module.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_INIT_StopModule(void)</td>
</tr>
<tr>
<td>Description</td>
<td>This function configures settings to enter module-stop state.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td>Remark</td>
<td>Transition to the module-stop state is not performed in the sample code. For more information on this function, refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.10 application note.</td>
</tr>
</tbody>
</table>

### R_INIT_NonExistentPort

<table>
<thead>
<tr>
<th>Outline</th>
<th>Nonexistent port initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_init_non_existent_port.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_INIT_NonExistentPort(void)</td>
</tr>
<tr>
<td>Description</td>
<td>This function initializes port direction registers for ports that do not exist in products with less than 176 pins.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>The number of pins in the sample code is set for the 176-pin package (PIN_SIZE=176). After this function is called, when writing in byte units to the PDR and PODR registers which have nonexistent ports, set the corresponding bits for nonexistent ports as follows: set the I/O select bits in the PDR registers to 1 and set the output data store bits in the PODR registers to 0. For more information on this function, refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.10 application note.</td>
</tr>
</tbody>
</table>
### R_INIT_Clock

**Outline:** Clock initialization  
**Header:** r_init_clock.h  
**Declaration:** void R_INIT_Clock(void)  
**Description:** This function initializes the clocks.  
**Arguments:** None  
**Return Value:** None  
**Remark:** In the sample code, the PLL clock is selected as the system clock, and the sub-clock is not used. For more information on this function, refer to the RX63N Group, RX631 Group Initial Setting Rev. 1.10 application note.

### peripheral_init

**Outline:** Peripheral function initialization  
**Header:** None  
**Declaration:** static void peripheral_init(void)  
**Description:** This function initializes the peripheral functions being used.  
**Arguments:** None  
**Return Value:** None

### sci9_init

**Outline:** SCI9 initialization  
**Header:** None  
**Declaration:** void sci9_init(void)  
**Description:** This function initializes channel SCI9.  
**Arguments:** None  
**Return Value:** None

### dmac0_init

**Outline:** DMAC0 initialization  
**Header:** None  
**Declaration:** void dmac0_init(void)  
**Description:** This function initializes DMAC0.  
**Arguments:** None  
**Return Value:** None

### dmac1_init

**Outline:** DMAC1 initialization  
**Header:** None  
**Declaration:** void dmac1_init(void)  
**Description:** This function initializes DMAC1.  
**Arguments:** None  
**Return Value:** None
SCI9 transmission/reception start

**Declaration**

```c
void sci9_start(void)
```

**Description**

This function starts transmission and reception on channel SCI9.

**Arguments**

None

**Return Value**

None

---

DMAC0 transfer end interrupt handling

**Declaration**

```c
static void Excep_DMAC_DMAC0I(void)
```

**Description**

This function disables reception, disables the RXI9 interrupt, and sets the receive end flag.

**Arguments**

None

**Return Value**

None

---

DMAC1 transfer end interrupt handling

**Declaration**

```c
static void Excep_DMAC_DMAC1I(void)
```

**Description**

This function disables the TXI9 interrupt and enables the TEI9 interrupt.

**Arguments**

None

**Return Value**

None

---

SCI9 transmit end interrupt handling

**Declaration**

```c
static void Excep_SCI9_TEI9(void)
```

**Description**

This function disables transmission, disables the TEI9 interrupt, and sets the transmit end flag.

**Arguments**

None

**Return Value**

None

---

Group 12 interrupt handling (SCI9 receive error interrupt)

**Declaration**

```c
static void Excep_ICU_GROUP12(void)
```

**Description**

This function performs group 12 interrupt handling (SCI9 receive error processing).

**Arguments**

None

**Return Value**

None

**Remarks**

SCI9 receive error processing is not performed in the sample code (infinite loop). Add a program as necessary.
5.8 Flowcharts

5.8.1 Main Processing

Figure 5.3 shows the Main Processing.

Figure 5.3 Main Processing
5.8.2  Port Initialization

Figure 5.4 shows the Port Initialization.

```
port_init

Set port output data
PORTB.PODR register
B7 bit ← 1: PB7/TXD9: High output

Set the port direction
PORT0.PDR register
B7 bit ← 0: P07/IRQ15: Input
PORTB.PDR register
B6 bit ← 0: PB6/RXD9: Input
B7 bit ← 1: PB7/TXD9: Output

Set the port mode
PORT0.PMR register
B7 bit ← 0: P07/IRQ15: Uses the pin as a general I/O pin.
PORTB.PMR register
B6 bit ← 0: PB6/RXD9: Uses the pin as a general I/O pin.
B7 bit ← 1: PB7/TXD9: Uses the pin as a general I/O pin.

return
```

Figure 5.4  Port Initialization
5.8.3 Peripheral Function Initialization

Figure 5.5 shows the Peripheral Function Initialization.

```plaintext
peripheral_init

Enable writing to related registers
PRCR register ← 0502h
PRC1 bit = 1

Cancel the module-stop state
MSTPCRC register
MSTPC26 bit ← 0: SCI9 module-stop state is canceled

Cancel the module-stop state
MSTPCRA register
MSTPA28 bit ← 0: DMAC module-stop state is canceled

Disable writing to related registers
PRCR register ← 0500h
PRC1 bit = 0

SCI9 initialization
sci9_init()

DMAC0 initialization
dmac0_init()

DMAC1 initialization
dmac1_init()

Enable DMAC activation
DMAST register ← 01h
DMST bit = 1

Disable the IRQ15 interrupt request
IER09 register
IEN7 bit ← 0

Disable the IRQ15 digital filter
IRQFLTE1 register
FLTEN15 bit ← 0

Set the IRQ15 sampling clock
IRQFLTC1 register
Bits FCLKSEL15[1:0] ← 01b: PCLKB/8

Set the IRQ15 port
PWPR register
B0WI bit ← 0; Writing to the PFSWE bit is enabled

PWPR register
PFWE bit ← 1; Writing to the PFS register is enabled
P07FFS register
ISEL bit ← 1: Used as IRQ15 input pin
PWPR register
PFSWE bit ← 0; Writing to the PFS register is disabled
PWPR register
B0WI bit ← 1; Writing to the PFSWE bit is disabled

Set the IRQ15 detection method
IRQCR15 register ← 04h
Bits IRQMD[1:0] = 01b: Falling edge

Clear the IRQ15 interrupt request
IR079 register
IR flag ← 0

Enable the 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 SCI9 Initialization.

```
sci9_init

Disable the SCI9 interrupt request
IER0E register
  IEN2 bit ← 0: ICU.GROUP12 (SCI9.ERI9) interrupt request is disabled
GEN12 register
  EN9 bit ← 0: SCI9.ERI9 (GROUP12) interrupt request is disabled
IER1E register
  IEN1 bit ← 0: SCI9.RXI9 interrupt request is disabled
IEN2 bit ← 0: SCI9.TXI9 interrupt request is disabled
IEN3 bit ← 0: SCI9.TEI9 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
PB6PFS register ← 0Ah
Bits PSEL[4:0] = 01010b: PB6 pin function select: RXD9
PB7PFS register ← 0Ah
Bits PSEL[4:0] = 01010b: PB7 pin function select: TXD9
PWPR register
  PFSEWE bit ← 0: Writing to the PFS register is disabled
PWPR register
  B0WI bit ← 1: Writing to the PFSWE bit is disabled
PORTB.PMR register
  B6 bit ← 1: Uses the PB6/RXD9 pin as an I/O port for peripheral functions

Select the clock
SCI9.SCR register
  Bits CKE[1:0] ← 00b: On-chip baud rate generator

Select the operating mode
SCI9.SIMR1 register
  IICM bit ← 0: Serial interface mode

Set the clock phase and polarity
SCI9.SPMR register
  CKPH bit ← 0: Clock is not delayed
CKPOL bit ← 0: Clock polarity is not inverted

Set the transmit and receive formats
SCI9.SMR register ← 01h
Bits CKS[1:0] = 01b: PCLKB/4
MP bit = 0: Multi-processor communications function is disabled
STOP bit = 0: 1 stop bit
PE bit = 0: Parity bit addition is not performed
CHR bit = 0: Selects 8 bits as the data length
CM bit = 0: Asynchronous mode
SCI9.SCMR register ← F2h
SMIF bit = 0: Serial communications interface mode
SINV bit = 0: TDR contents are transmitted as they are. Receive data is stored as it is in RDR.
SDIR bit = 0: Transfer with LSB-first
SCI9.SEMR register ← 00h
ABCs bit = 0: Selects 16 base clock cycles for 1-bit period
NFEN bit = 0: Noise cancellation function for the RXDn input signal is disabled.

Set the bit rate
SCI9.BRR register ← 9: 8.766 = (48 MHz ÷ (64 × 2 × 38,400 bps)) - 1

Set the interrupt priority levels
IPR241 register
  Bits IPR[3:0] ← 0001b: Interrupt priority level for SCI9.RXI9, TXI9, or TEI9 is 1
IPR114 register
  Bits IPR[3:0] ← 0001b: Interrupt priority level for ICU.GROUP12 (SCI9.ERI9) is 1

Clear the interrupt requests
IR flag ← 0: No SCI9.RXI9 interrupt request is generated
IR242 register
  IR flag ← 0: No SCI9.TXI9 interrupt request is generated

return

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

Figure 5.6 SCI9 Initialization
5.8.5 DMAC0 Initialization

Figure 5.7 shows the DMAC0 Initialization.

- **dmac0_init**

  **IE**
  - Disable the DMAC0I interrupt request
  - IER18 register
  - IEN6 bit ← 0

  **DM**
  - Disable DMA transfer
  - DMAC0.DMCNT register ← 00h
  - DTE bit = 0

  **RS**
  - Set the activation source
  - DMRSR0 register ← 241
  - Bits DMRS[7:0] = 1111 0001b: The vector number for DMAC0 activation request (RXI9) is specified.

  **MD**
  - Set the address mode
  - DMAC0.DMAMD register ← 0080h
  - Bits DARA[4:0] = 00000b: Extended repeat area not specified
  - Bits DM[1:0] = 10b: Destination address is incremented.
  - Bits SARA[4:0] = 00000b: Extended repeat area not specified
  - Bits SM[1:0] = 00b: Destination address is fixed.

  **MT**
  - Set the transfer mode
  - DMAC0.DMTMD register ← 0001h
  - Bits DCTG[1:0] = 01b: Interrupts from peripheral modules
  - Bits SZ[1:0] = 00b: 8-bit transfer
  - Bits MD[1:0] = 00b: Normal transfer

  **CS**
  - Clear the activation source
  - DMAC0.DMCSL register ← 00h
  - DISEL bit = 0: At the beginning of transfer, clear the interrupt flag of the activation source to 0.

  **MS**
  - Set the transfer source address
  - DMAC0.DMSAR register ← SCI9.RDR register address

  **MD**
  - Set the transfer destination address
  - DMAC0.DMDAR register ← rcvbuf[0] address

  **MC**
  - Set the number of transfers
  - DMAC0.DMCRA register ← DMAC_CNT

  **IP**
  - Set the interrupt priority level
  - IPR198 register
  - Bits IPR[3:0] ← 0001b: DMAC0I interrupt priority level 1

  **ME**
  - Enable the transfer end interrupt
  - DMAC0.DMINT register ← 10h
  - DTIE bit = 1

  **MI**
  - Enable the DMAC0I interrupt request
  - IER18 register
  - IEN6 bit ← 1

  **MA**
  - Enable DMA transfer
  - DMAC0.DMCNT register ← 01h
  - DTE bit = 1

  **return**
5.8.6 DMAC1 Initialization

Figure 5.8 shows the DMAC1 Initialization.

```
dmac1_init

- IER18 register
  IEN7 bit ← 0

- Disable DMA transfer
  DMCNT register ← 00h
  DTE bit = 0

- Set the activation source
  DMRS register ← 242
  Bits DMRS[7:0] = 1111 0010b: The vector number for DMAC1 activation request (TXI9) is specified.

- Set the address mode
  DMTMD register ← 8000h
  Bits DARA[4:0] = 00000b: Extended repeat area not specified
  Bits DM[1:0] = 00b: Destination address is fixed.
  Bits SARA[4:0] = 00000b: Extended repeat area not specified
  Bits SM[1:0] = 10b: Destination address is incremented.

- Set the transfer mode
  DMTMD register ← 0001h
  Bits DCTG[1:0] = 01b: Interrupts from peripheral modules
  Bits SZ[1:0] = 00b: 8-bit transfer
  Bits MD[1:0] = 00b: Normal transfer

- Set the transfer source address
  DMSAR register ← trnbuf[0] address

- Set the transfer destination address
  DMDAR register ← SCI9.TDR register address

- Set the number of transfers
  DMCRA register ← DMAC_CNT

- Set the interrupt priority level
  IPR register
  Bits IPR[3:0] ← 0001b: DMAC1 interrupt priority level 1

- Enable the transfer end interrupt
  DMI register
  DTIE bit = 1

- Enable the DMAC1 interrupt request
  IER18 register
  IEN7 bit ← 1

- Enable DMA transfer
  DMCNT register ← 01h
  DTE bit = 1

return
```

Figure 5.8 DMAC1 Initialization
5.8.7 SCI9 Transmission/Reception Start

Figure 5.9 shows SCI9 Transmission/Reception Start.

```
sci9_start

1. Disable DMA transfer
   DMAC0.DMCNT register ← 00h
   DTE bit = 0

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

3. Set the number of DMAC0 transfers
   DMAC0.DMCRA register ← DMAC_CNT

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

5. Disable DMA transfer
   DMAC0.DMCNT register ← 00h
   DTE bit = 0

6. Set the DMAC1 transfer source address
   DMAC1.DMSAR register ← trnbuf[0] address

7. Set the number of DMAC1 transfers
   DMAC1.DMCRA register ← DMAC_CNT

8. Enable DMA transfer
   DMAC1.DMCNT register ← 01h
   DTE bit = 1

9. Start SCI9 transmission and reception
   SCI9.SCR register ← SCI9.SCR register | F4h
   TEIE bit = 1: A TEI interrupt request is enabled
   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: A TXI interrupt request is enabled

10. Set the I/O port functions
    PORTB.PMR register
    B7 bit ← 1: Uses PB7/TXD9 as an I/O port for peripheral functions.

11. Set the ERI9 interrupt request
    GEN12 register
    EN9 bit ← 1: SCI9.ERI9 (GROUP12) interrupt request is enabled.

12. Enable the GROUP12 interrupt request
    IER0E register
    IEN2 bit ← 1: ICU.GROUP12 (SCI9.ERI9) interrupt request is enabled.

13. Enable the RXI9 interrupt request
    IER1E register
    IEN1 bit ← 1: SCI9.RXI9 interrupt request is enabled.

14. Enable the TXI9 interrupt request
    IER1E register
    IEN2 bit ← 1: SCI9.TXI9 interrupt request is enabled.

return
```

Figure 5.9 SCI9 Transmission/Reception Start
5.8.8 DMAC0 Transfer End Interrupt Handling

Figure 5.10 shows DMAC0 Transfer End Interrupt Handling.

```
Excep_DMAC_DMAC0I

Enable the TXI9 interrupt request
IER1E register
IEN2 bit ← 0: SCI9.TXI9 interrupt request is disabled

Disabling ERI interrupts
SCI9.SCR register
TIE bit ← 0: A TXI interrupt request is disabled

Clear the TXI9 interrupt request
IR242 register
IR flag ← 0: SCI9.TXI9 interrupt request not generated

Set the receive end flag
rcv_end_flag ← 1: Reception ended

return

Note 1: After writing a value to the TIE bit, confirm that the written value can be read.
```

Figure 5.10 DMAC0 Transfer End Interrupt Handling

5.8.9 DMAC1 Transfer End Interrupt Handling

Figure 5.11 shows DMAC1 Transfer End Interrupt Handling.

```
Excep_DMAC_DMAC1I

Enable the TEI9 interrupt request
IER1E register
IEN3 bit ← 0: SCI9.TEI9 interrupt request is enabled

Clear the TXI9 interrupt request
IR242 register
IR flag ← 0: SCI9.TXI9 interrupt request not generated

Disabling ERI interrupts
SCI9.SCR register
TIE bit ← 0: A TXI interrupt request is disabled

Disable the TXI9 interrupt request
IER1E register
IEN2 bit ← 0: SCI9.TXI9 interrupt request is disabled

Disable interrupt requests
SCI9.SCR register
RIE bit ← 0: RXI and ERI interrupt requests are disabled

Clear the TXI9 interrupt request
IR241 register
IR flag ← 0: SCI9.RXI9 interrupt request not generated

Set the receive end flag
rcv_end_flag ← 1: Reception ended

return

Note 1: After writing values to bits RE and RIE, confirm that written values can be read.
```

Figure 5.11 DMAC1 Transfer End Interrupt Handling
5.8.10 SCI9 Transmit End Interrupt Handling

Figure 5.12 shows SCI9 Transmit End Interrupt Handling.

![Diagram](image)

5.8.11 Group 12 Interrupt Handling (SCI9 Receive Error Interrupt)

Figure 5.13 shows Group 12 Interrupt Handling (SCI9 Receive Error Interrupt).

![Diagram](image)

Note 1: After writing a value to the SCR register, confirm that the written value can be read.
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.1.70 (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>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Apr. 1, 2014</td>
<td>—</td>
<td>—</td>
<td>First edition issued</td>
</tr>
</tbody>
</table>
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 accordance with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

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

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

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

5. Differences between Products
   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.
   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.
Notice

1. Descriptions of circuits, software and 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 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 depend on the product's quality grade, as indicated below.

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

   "High-Quality": Transportation equipment, automobiles, ships, etc.; traffic control systems; anti-disaster systems; anti-crisis 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 any 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 failures at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, 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 where 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, disperses 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. The 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.

Renesas Electronics Corporation

Refer to “http://www.renesas.com” for the latest and detailed information.