# RX220 Group and RX21A Group

## Asynchronous SCI Communication Using the DTCa

R01AN1884EJ0100
Rev.1.00
Feb 14, 2014

### Introduction

This application note describes methods for performing asynchronous serial communication using the data transfer controller (DTC) and serial communications interface (SCI) peripheral modules provided by the RX220 Group and RX21A Group microcontrollers.

### Target Device

- RX220 Group: 100-pin versions with ROM capacities of 32 to 256 KB
- RX21A Group: 100-pin versions with ROM capacities of 256 to 512 KB

When using the code presented in this application note with a different microcontroller, modify the code according to the specifications of that microcontroller and test the code thoroughly.

## Contents

# 1. Specifications

This system performs serial communication using the SCI.

The transmit data is set up in advance in the RAM transmit data storage area and transmitted using the DTC. The receive data is stored in the RAM receive data storage area using the DTC.

Serial communication is started when a falling edge is detected on the interrupt request pin (IRQ1).

- Transfer rate : 38,400 bps
- Data length : 8 bits, LSB first
- Stop bits : 1 bit
- Parity : None
- Hardware flow control : None

Table 1.1 lists the peripheral functions used and their applications and figure 1.1 presents a block diagram.

**Table 1.1 Peripheral Functions Used and Their Applications**

| Peripheral Function | Application |
|---|---|
| SCI | Asynchronous serial communication |
| DTC | Transfers the SCI1 receive data to RAM<br>Transfers the transmit data in RAM to the SCI1 |
| IRQ1 | Serial communication start trigger |

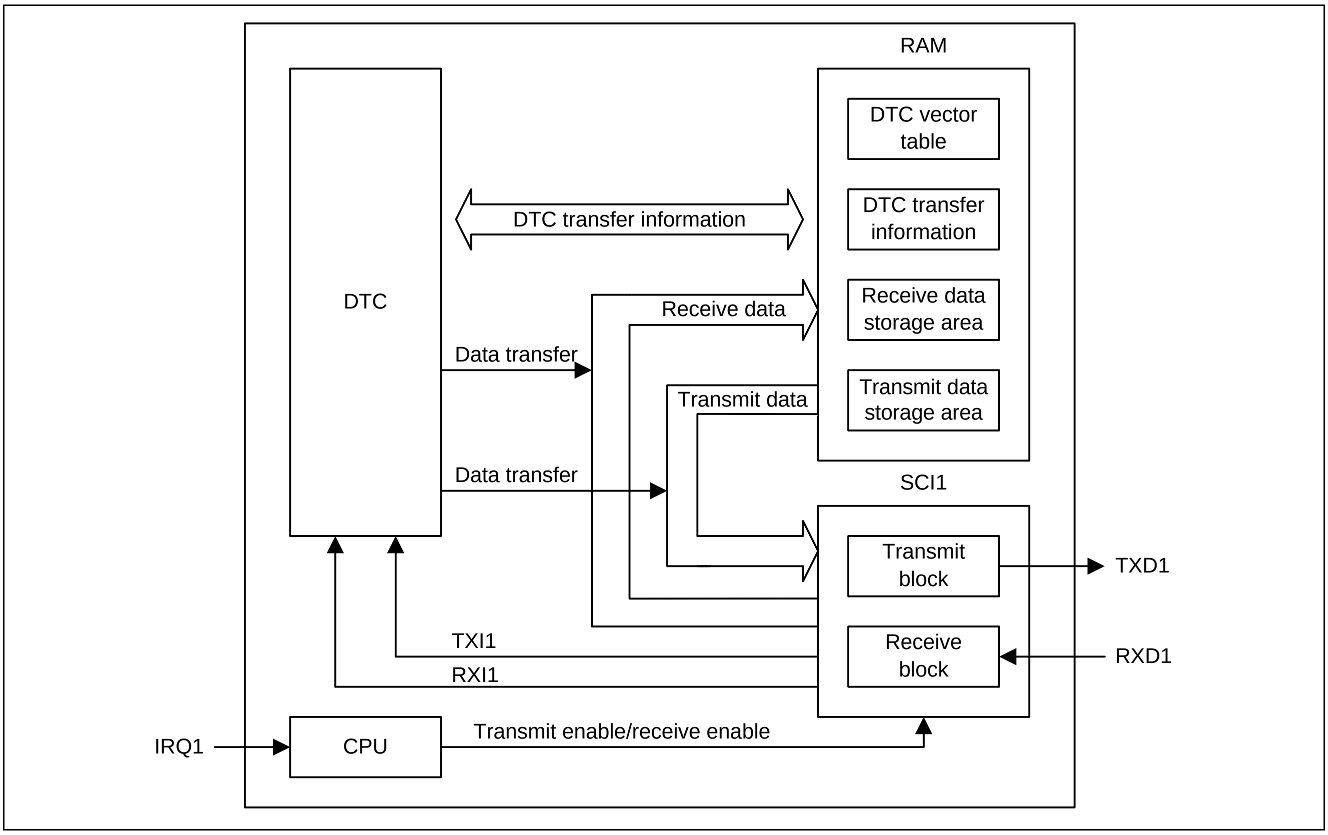



**Figure 1.1 Block Diagram**

## 2. Confirmed Operating Condition

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

### (1) For the RX220

**Table 2.1 Confirmed Operating Condition**

| Item | Description |
|---|---|
| MCU used | R5F52206BDFP (RX220 Group) |
| Operating frequency | Main clock: 20.0 MHz<br>System clock (ICLK): 20 MHz (1 period of the main clock)<br>Peripheral module clock B (PCLKB): 20 MHz (1 period of the main clock) |
| Operating voltage | 5.0 V: Supplied from the E1 emulator |
| Integrated development environment | Renesas Electronics Corporation<br>High-performance Embedded Workshop Version 4.09.01.007 |
| C compiler | Renesas Electronics Corporation<br>C/C++ Compiler Package for RX Family V.1.02 Release 01 |
| | Compiler option<br>-cpu=rx200 -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" -debug –nologo<br>(The integrated development environment default settings are used.) |
| iodefine.h version | Version 1.0A |
| Endian order | Little endian |
| Operating mode | Single-chip mode |
| Processor mode | Supervisor mode |
| Sample code version | Version 1.00 |
| Board used | Renesas Starter Kit for RX220 (Product number: R0K505220S000BE) |

**(2) For the RX21A**

**Table 2.2 Confirmed Operating Condition**

| Item | Description |
|---|---|
| MCU used | R5F521A8BDFP (RX21A Group) |
| Operating frequency | Main clock: 20.0 MHz<br>PLL: 100 MHz (2 period of the main clock and multiplied by 10)<br>System clock (ICLK): 50 MHz (2 period of the PLL)<br>Peripheral module clock B (PCLKB): 25 MHz (4 period of the PLL) |
| Operating voltage | 3.3 V: Supplied from the E1 emulator |
| Integrated development environment | Renesas Electronics Corporation<br>High-performance Embedded Workshop Version 4.09.01.007 |
| C compiler | Renesas Electronics Corporation<br>C/C++ Compiler Package for RX Family V.1.02 Release 01 |
| | Compiler option<br>-cpu=rx200 -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" -debug –nologo<br>(The integrated development environment default settings are used.) |
| iodefine.h version | Version 1.00 |
| Endian order | Little endian |
| Operating mode | Single-chip mode |
| Processor mode | Supervisor mode |
| Sample code version | Version 1.00 |
| Board used | HSB Series microcontroller board (Hokuto Denshi Co., Ltd.)<br>(Catalog number: HSBRZ21AP-B) |
| | Serial connection board<br>• RS-232C serial connector (D-sub 9-pin connector)<br>• RS-232C transceiver (MAX3232CPE) |

## 3. Reference Application Notes

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

- RX220 Group Initial Setting Rev.1.00 (R01AN1494EJ0100_RX220)
- RX21A Group Initial Setting Rev.1.00 (R01AN1486EJ0100_RX21A)

The sample code with this application note uses the initialization from the application note noted above. The revision number is the one that was current when this application note was written.

If there is a more recent version, you should replace this code with the most recent version. The most recent version can be downloaded from the Renesas Electronics Corporation web site.

# 4. Hardware

## 4.1 Hardware Structure

Figure 4.1 shows a sample RX220 connection and figure 4.2 shows a sample RX21A connection.

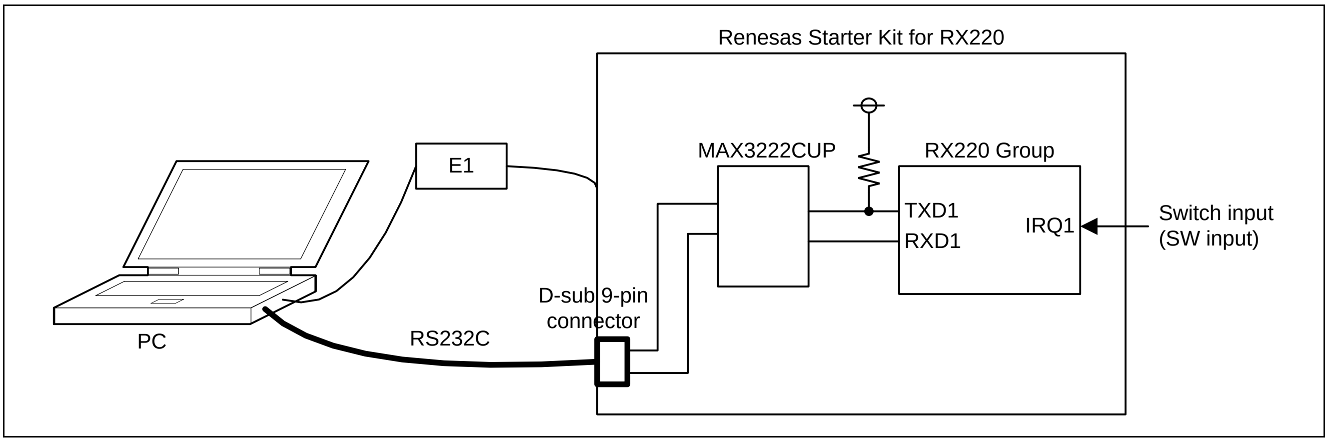



**Figure 4.1 Sample RX220 Connection**

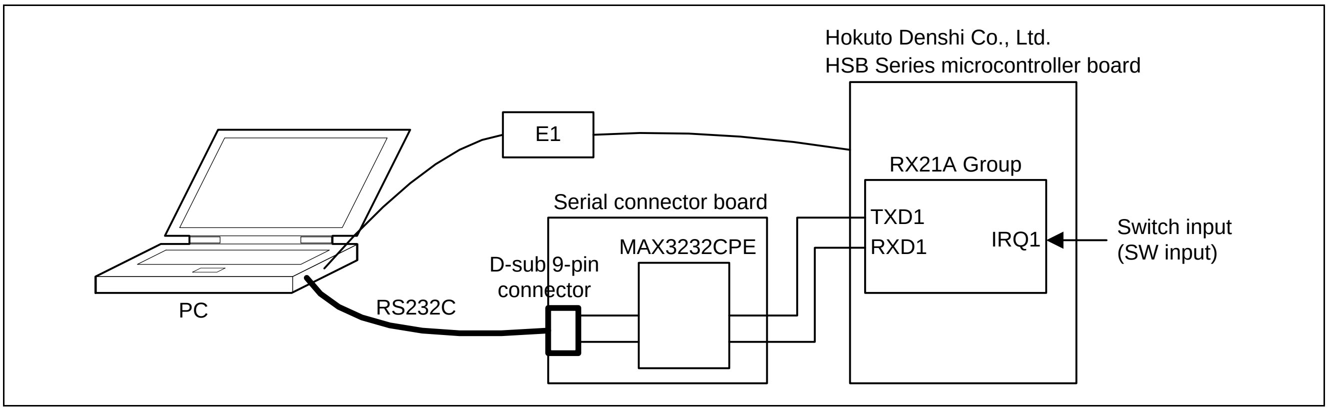



**Figure 4.2 Sample RX21A Connection**

## 4.2 Pins Used

Table 4.1 lists the pins used and their functions.

**Table 4.1 Pins Used and Their Functions**

| Pin Name | I/O | Function |
|---|---|---|
| P31/IRQ1 | Input | Communication start switch input |
| P15/RXD1 | Input | SCI1 receive data input |
| P16/TXD1 | Output | SCI1 transmit data output |

## 5. Software

This sample code automatically processes SCI1 transmit and receive operations by using the DTC module. The communication start switch is pressed SCI1 transmit and receive operations are started.

When transmission is enabled, a TXI1 interrupt request occurs and this acts as the DTC start factor. The data in the transmit data storage area is transferred to the TDR register by the DTC module and transmitted.

When a reception completes, an RXI1 interrupt request occurs and this acts as the DTC start factor. The receive data is transmitted to the receive data storage area by the DTC module.

When 256 transfers of transmit data complete, a TXI1 interrupt request occurs. At this point the TXI1 interrupt is disabled and the TEI1 interrupt is enabled.

When 256 transfers of receive data complete, an RXI1 interrupt occurs. At this point SCI1 reception and the RXI1 interrupt are disabled and the reception complete flag is set to 1.

When the 256-byte transmission has completed, a TEI1 interrupt occurs. At this point SCI1 transmission and the TEI1 interrupt are disabled and the transmission complete flag is set to 1.

The settings for the peripheral functions used are shown below.

SCI1

- Serial communication method : Asynchronous
- Transfer rate : 38400 bps
- Clock source : PCLKB
- Data length : 8 bits
- Stop bits : 1 bit
- Parity function : No parity
- Data transfer method : LSB first
- Interrupts : Reception error interrupt (ERI1) enabled
  : Receive data full interrupt (RXI1) enabled
  : Transmit data empty interrupt (TXI1) enabled
  : Transmit complete interrupt (TEI1) enabled

DTC

- Start factor: TXI1 and RXI1 interrupt requests
- DTC addressing mode: Full addressing mode

  [DTC Transfer Settings for Activation by a TXI1 Interrupt Request]
  - Transfer mode : Normal transfer mode
  - Transfer source addressing mode : SAR register incremented after transfer
  - Transfer source address : RAM (transmit data storage area start address)
  - Transfer target addressing mode : DAR register holds fixed address
  - Transfer target address : SCI1.TDR register
  - Data transfer unit : 8 bits
  - Transfer count : 256 transfers
  - Chained transfers : Disabled
  - Interrupts : The CPU is interrupted when the specified data transfer completes.

[DTC Transfer Settings for Activation by a RXI1 Interrupt Request]

— Transfer mode : Normal transfer mode
— Transfer source addressing mode : SAR register holds fixed address
— Transfer source address : SCI1.RDR register
— Transfer target addressing mode : DAR register incremented after transfer
— Transfer target address : RAM (receive data storage area start address)
— Data transfer unit : 8 bits
— Transfer count : 256 transfers
— Chained transfers : Disabled
— Interrupts : The CPU is interrupted when the specified data transfer completes.

IRQ1 input pin

- Detection method : Falling edge detection
- Digital filter : Disabled
- Interrupt : Not used

## 5.1 Operation Overview

### 5.1.1 Transmit Operation

(1) Initialization

After initialization, the sample code waits for a communication start switch input.

(2) Communication start switch input detection

When a communication start switch input is detected, The IRQ1 IR flag is set to 0. When the sample code has verified that transmission and reception have competed by checking the transmission complete and reception complete flags, the transmission complete flag is set to 0 (transmission in progress). The DTC transfer source address and transfer count are set and DTC activation is enabled. The SCI1.SCR.TEIE, TIE, RIE, TE, and RE bits are set to 1 at the same time to enable transmission and reception. The TXI1 interrupt IR flag goes to 1 when the SCI1.SCR.TIE and TE bits are set to 1 at the same time.

(3) Data transfer start

When the TXI1 interrupt is enabled, the TXI1 interrupt IR flag goes to 0 when the DTC is activated. The first byte of the transmit data is transferred from the RAM transmit data storage area to the SCI1.TDR register.

(4) Data transmission start

The TXI1 interrupt IR flag is set to 1 by the transfer of data from the SCI1.TDR register to the SCI1.TSR register and the first byte of transmit data is output from the TXD1 pin. The DTC module is activated by the TXI1 interrupt and the second byte of data is transferred.

(5) TXI1 interrupt

When the 256th data transfer completes, the CPU accepts a TXI1 interrupt request. In TXI1 interrupt handling, the TXI1 interrupt is disabled and the TEI1 interrupt is enabled.

(6) TEI1 interrupt

When the last bit of the 256th byte is transmitted, the SCI1.TDR register is not updated and therefore a TEI1 interrupt request is generated. In TEI1 interrupt handling, transmission is disabled and the TEI1 interrupt is disabled. The transmission complete flag is set to 1 (transmission complete).

After this, this sequence is repeated starting at item (2) above.

Figure 5.1 shows the timing chart for the transmission operation.

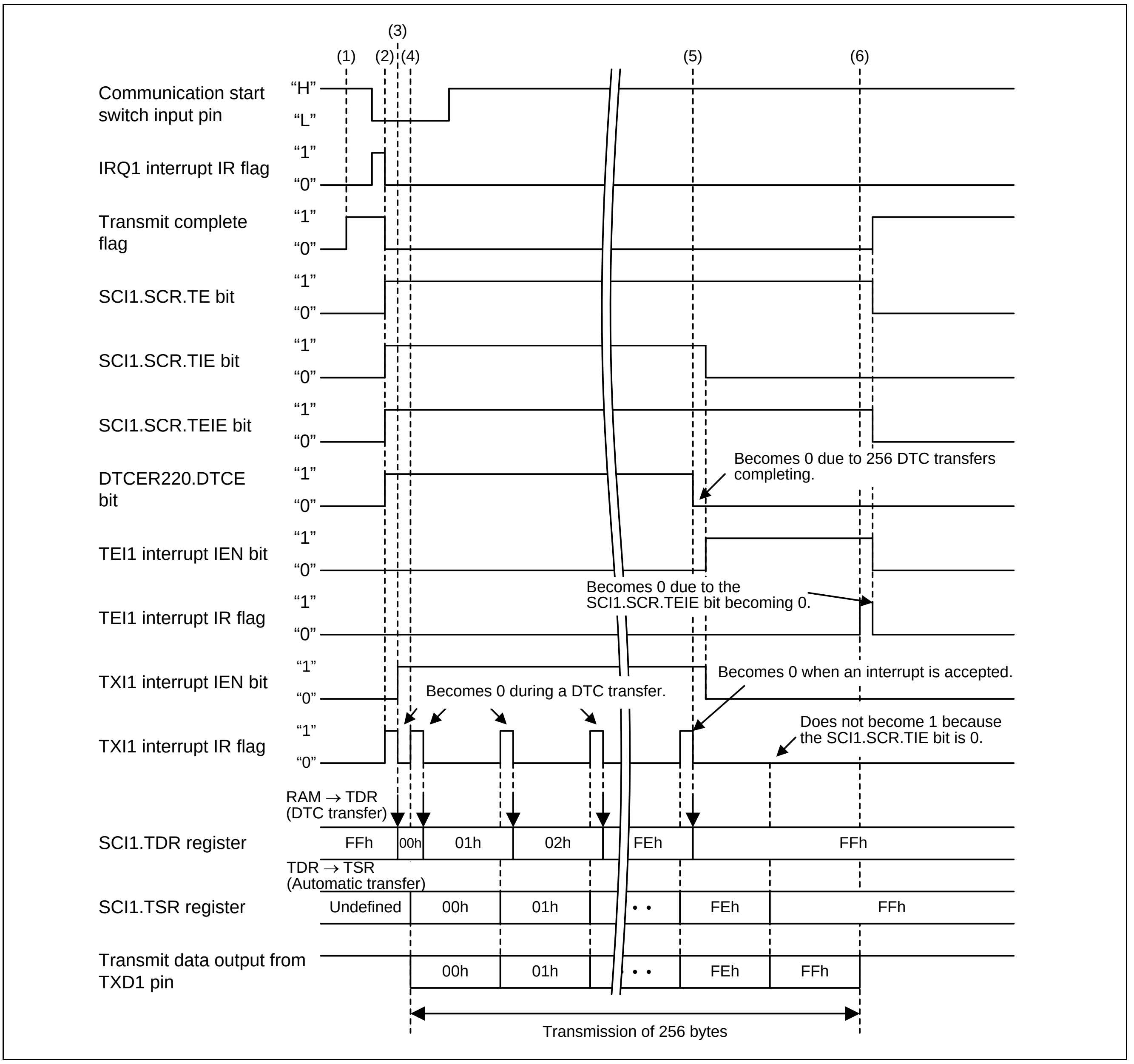



**Figure 5.1 Transmission Operation Timing Chart**

### 5.1.2 Receive Operation

(1) Initialization

After initialization, the sample code waits for a communication start switch input.

(2) Communication start switch input detection

When a communication start switch input is detected, The IRQ1 IR flag is set to 0. When the sample code has verified that transmission and reception have competed by checking the transmission complete and reception complete flags, the reception complete flag is set to 0 (reception in progress). The DTC transfer target address and transfer count are set and DTC activation is enabled. The SCI1.SCR.TEIE, TIE, RIE, TE, and RE bits are set to 1 at the same time to enable transmission and reception and the RXI1 interrupt is enabled.

(3) Data reception termination

When the reception of one byte of data completes, the data is transferred from the SCI1.RSR register to the SCI1.RDR register and the RXI1 interrupt IR flag is set to 1.

(4) Data transfer start

The DTC module is started by an RXI1 interrupt request and the RXI1 interrupt IR flag becomes 0. One byte of receive data is transferred from the SCI1.RDR register to the RAM receive data storage area.

(5) RXI1 interrupt

When 256th data transfer completes, the CPU accepts the RXI1 interrupt request. Reception is disabled and the RXI1 interrupt is disabled in RXI1 interrupt handling. The reception complete flag is set to 1 (reception complete). After this, this sequence is repeated starting at item (2) above.

Figure 5.2 shows the timing chart for the reception operation.

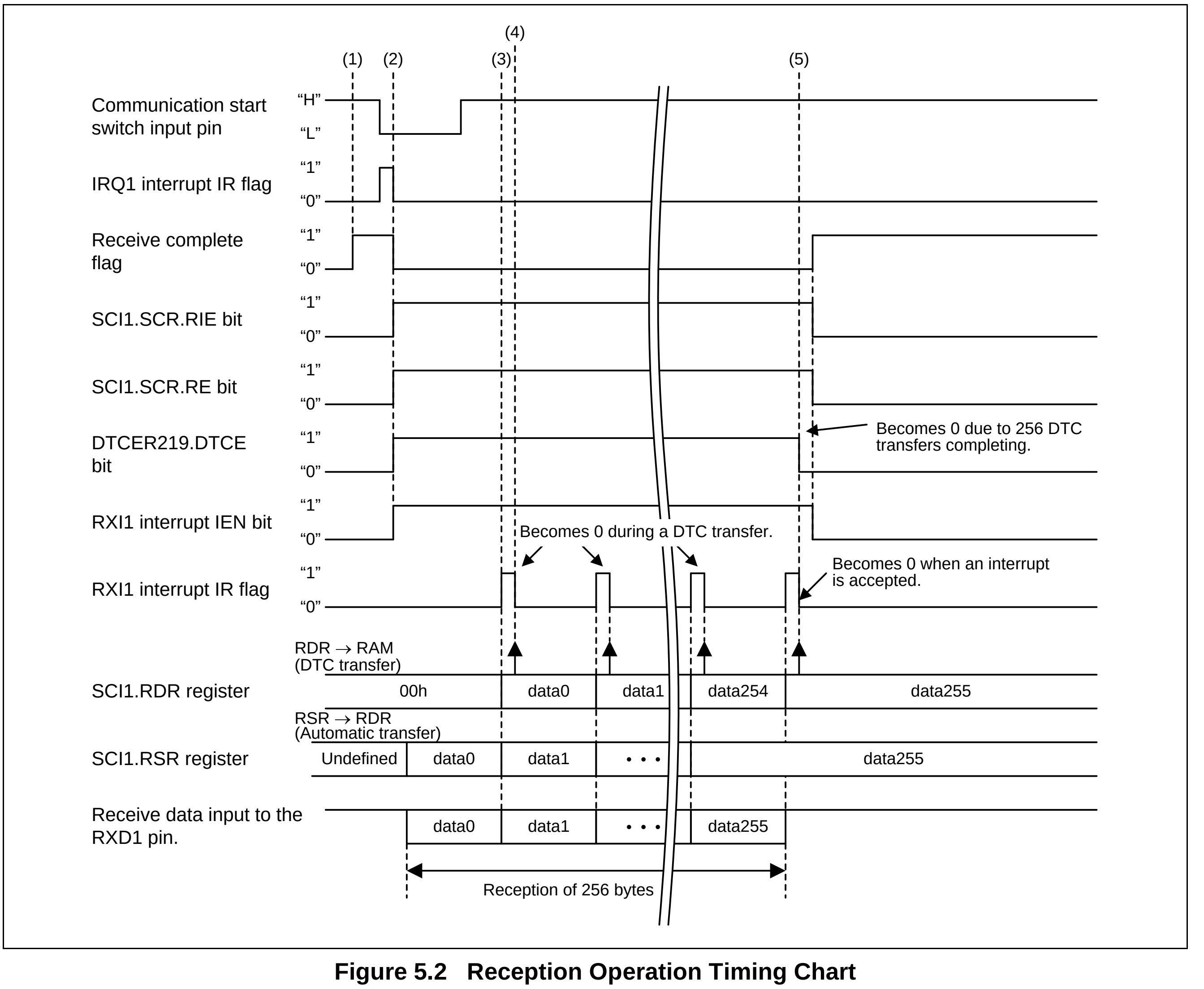



**Figure 5.2 Reception Operation Timing Chart**

Notes on Embedding in an Actual System

The following phenomenon requires care when using the sample code provided with this application note in an actual system.

- This sample code may not operate correctly if an interrupt used by this application note's code is delayed for an extended period due to, for example, the handing of other interrupts.

## 5.2 Section Structure

Table 5.1 lists the section information modified in this sample code.

Refer to the latest version of the RX Family C/C++ Compiler Package User's Manual for the methods for adding, modifying, and deleting sections.

**Table 5.1 Section Information Modified by this Sample Code**

| Section Name | Modification | Address | Contents |
|---|---|---|---|
| DTC_SECTION | Added | 0000 3000h | DTC vector table |

## 5.3 File Composition

Table 5.2 lists the files used for the sample code. Note that the files generated automatically by the integrated development environment are not shown.

**Table 5.2 File Composition**

| Target Device | File Name | Overview | Remarks |
|---|---|---|---|
| Common | main.c | Main processing | |
| | device_cfg.h | Device configuration header file | |
| RX220 | r_init_stop_module.c | Stops peripheral functions operating after a reset | |
| | r_init_stop_module.h | r_init_stop_module_.c header file | |
| | r_init_non_existent_port.c | Nonexistent port initialization | |
| | r_init_non_existent_port.h | r_init_non_existent_port.c header file | |
| | r_init_clock.c | Clock initialization | |
| | r_init_clock.h | r_init_clock.c header file | |
| RX21A | r_init_stop_module.c | Stops peripheral functions operating after a reset | |
| | r_init_stop_module.h | r_init_stop_module_.c header file | |
| | r_init_non_existent_port.c | Nonexistent port initialization | |
| | r_init_non_existent_port.h | r_init_non_existent_port.c header file | |
| | r_init_clock.c | Clock initialization | |
| | r_init_clock.h | r_init_clock.c header file | |

## 5.4 Option-Setting Memory

Table 5.3 lists the states of the option settings memory used by the sample code. Set these locations to appropriate values for your user system as required.

**Table 5.1 Option Settings Memory Set by the Sample Code**

| Symbol | Address | Setting Value | Contents |
|---|---|---|---|
| OFS0 | FFFF FF8Fh to FFFF FF8Ch | FFFF FFFFh | Stops IWDT after a reset<br>Stops WDT after a reset |
| OFS1 | FFFF FF8Bh to FFFF FF88h | FFFF FFFFh | Disables voltage monitoring resets after a reset<br>Disables HOCOC oscillation after a reset |
| MDES | FFFF FF83h to FFFF FF80h | FFFF FFFFh | Little endian |

## 5.5 Constants

Table 5.4 lists the constants used in the sample code.

**Table 5.4 Constants Used in the Sample Code**

| Constant | Set value | Description |
|---|---|---|
| BUF_SIZE | 256 | Transmit and receive data storage area size |
| DTC_CNT | BUF_SIZE | DTC transfer count |
| SCI_BIT_RATE | 15 (When RX220 is selected) | SCI1.BRR register setting value (when PCLKB is 20 MHz) |
| | 19 (When RX21A is selected) | SCI1.BRR register setting value (when PCLKB is 25 MHz) |

## 5.6 Structures and Unions

Figure 5.3 shows the structures and unions used in the sample code.

```
/* **** DTC transfer Information **** */
#pragma bit_order    left        /* Bit field order specification: members are allocated starting with the high order bits. */
#pragma unpack                   /* Structure member boundary alignment: alignment is according to the member type. */
struct st_dtc_full_t
{
    union
    {
        uint32_t    LONG;
        struct
        {
            uint32_t    MRA_MD        :2;
            uint32_t    MRA_SZ        :2;
            uint32_t    MRA_SM        :2;
            uint32_t                  :2;
            uint32_t    MRB_CHNE      :1;
            uint32_t    MRB_CHNS      :1;
            uint32_t    MRB_DISEL     :1;
            uint32_t    MRB_DTS       :1;
            uint32_t    MRB_DM        :2;
            uint32_t                  :2;
            uint32_t                  :16;
        } BIT;
    } MR;
    void *    SAR;
    void *    DAR;
    struct
    {
        uint32_t    CRA:16;
        uint32_t    CRB:16;
    } CR;
};
#pragma packoption               /* End structure member boundary alignment. */
#pragma bit_order                /* End bit field order specification. */
```

**Figure 5.3 Structures and Unions Used in the Sample Code**

## 5.7 Variables

Table 5.5 lists the static variables.

**Table 5.5 Static Variables**

| Type | Name | Description | Function |
|---|---|---|---|
| static uint8_t | trn_end_flag | Transmission complete flag<br>0: Transmission in progress<br>1: Transmission complete | main<br>Excep_SCI1_TEI1 |
| static uint8_t | rcv_end_flag | Reception complete flag<br>0: Reception in progress<br>1: Reception complete | main<br>Excep_SCI1_RXI1 |
| static uint8_t | trnbuf[BUF_SIZE] | Transmit data storage area | main<br>dtc_init<br>sci1_start |
| static uint8_t | rcvbuf[BUF_SIZE] | Receive data storage area | dtc_init<br>sci1_start |
| static struct st_dtc_full_t | dtc_info_rxi1 | RXI1 DTC transfer information | dtc_init<br>sci1_start |
| static struct st_dtc_full_t | dtc_info_txi1 | TXI1 DTC transfer information | dtc_init<br>sci1_start |
| static void * | dtc_vect_table [256] | DTC vector table | dtc_init |

## 5.8 Functions

Table 5.6 shows the functions used in the sample code.

**Table 5.6 Functions**

| Function | Description |
|---|---|
| main | Main processing |
| port_init | Port initialization |
| R_INIT_StopModule | Stops peripheral modules operating after a reset |
| R_INIT_NonExistentPort | Nonexistent port initialization |
| R_INIT_Clock | Clock initialization |
| peripheral_init | Peripheral function initialization |
| sci1_init | SCI1 initialization |
| dtc_init | DTC initialization |
| irq_init | IRQ initialization |
| sci1_start | Start SCI1 transmission |
| Excep_SCI1_RXI1 | SCI1 receive data full interrupt handler |
| Excep_SCI1_TXI1 | SCI1 transmit data empty interrupt handler |
| Excep_SCI1_TEI1 | SCI1 transmission complete interrupt handler |
| Excep_SCI1_ERI1 | SCI1 receive error interrupt handler |

## 5.9 Function Specifications

This section lists the specifications of the functions in the sample code.

| main | |
|---|---|
| **Overview** | Main processing |
| **Header** | None |
| **Declaration** | void main(void) |
| **Description** | After initialization, starts an SCI1 communication operation when a communication start switch input is detected. |
| **Arguments** | None |
| **Return values** | None |

| port_init | |
|---|---|
| **Overview** | Port initialization |
| **Header** | None |
| **Declaration** | static void port_init(void) |
| **Description** | Initializes the ports. |
| **Arguments** | None |
| **Return values** | None |

| R_INIT_StopModule | |
|---|---|
| **Overview** | Stops peripheral modules operating after a reset |
| **Header** | r_init_stop_module.h |
| **Declaration** | void R_INIT_StopModule(void) |
| **Description** | Performs the settings required to transition to the module stop state. |
| **Arguments** | None |
| **Return values** | None |
| **Remarks** | In this sample code, the transition to the module stop state is not performed.<br>See the "RX220 Group Initialization Example, Revision 1.00" and "RX21A Group Initialization Example, Revision 1.00" application notes for details on this function. |

| R_INIT_NonExistentPort | |
|---|---|
| **Overview** | Nonexistent port initialization |
| **Header** | r_init_non_existent_port.h |
| **Declaration** | void R_INIT_NonExistentPort(void) |
| **Description** | Performs port direction register initialization for non-existent port pins for microcontroller versions that have fewer than 100 pins. |
| **Arguments** | None |
| **Return values** | None |
| **Remarks** | In this sample code, settings for the 100-pin versions (PIN_SIZE = 100) are performed.<br>After this function is called, if an application writes in byte units to a PDR or PODR register that includes ports that do not exist, the application should set the direction control bit to 1 and the port output data storage bit to 0 for those nonexistent ports.<br>See the "RX220 Group Initialization Example, Revision 1.00" and "RX21A Group Initialization Example, Revision 1.00" application notes for details on this function. |

| R_INIT_Clock | |
|---|---|
| **Overview** | Clock initialization |
| **Header** | r_init_clock.h |
| **Declaration** | void R_INIT_Clock(void) |
| **Description** | Initializes the clocks. |
| **Arguments** | None |
| **Return values** | None |
| **Remarks** | In the RX220 sample code, processing in which the main clock is used as the system clock and the subclock is not used is selected.<br>In the RX21A sample code, processing in which the PLL circuit is used as the system clock and the subclock is not used is selected.<br>See the "RX220 Group Initialization Example, Revision 1.00" and "RX21A Group Initialization Example, Revision 1.00" application notes for details on this function. |

| peripheral_init | |
|---|---|
| **Overview** | Peripheral function initialization |
| **Header** | None |
| **Declaration** | static void peripheral_init(void) |
| **Description** | Initializes the used peripheral modules. |
| **Arguments** | None |
| **Return values** | None |

| sci1_init | |
|---|---|
| **Overview** | SCI1 initialization |
| **Header** | None |
| **Declaration** | static void sci1_init(void) |
| **Description** | Initializes the SCI1 module. |
| **Arguments** | None |
| **Return values** | None |

| dtc_init | |
|---|---|
| **Overview** | DTC initialization |
| **Header** | None |
| **Declaration** | static void dtc_init(void) |
| **Description** | Initializes the DTC module. |
| **Arguments** | None |
| **Return values** | None |

| irq_init | |
|---|---|
| **Overview** | IRQ initialization |
| **Header** | None |
| **Declaration** | static void irq_init(void) |
| **Description** | Initializes the IRQ1 module. |
| **Arguments** | None |
| **Return values** | None |

| sci1_start | |
|---|---|
| **Overview** | Start SCI1 transmission |
| **Header** | None |
| **Declaration** | static void sci1_start(void) |
| **Description** | Starts an SCI1 communication operation |
| **Arguments** | None |
| **Return values** | None |

| Excep_SCI1_RXI1 | |
|---|---|
| **Overview** | SCI1 receive data full interrupt handler |
| **Header** | None |
| **Declaration** | static void Excep_SCI1_RXI1(void) |
| **Description** | Stops reception and disables the RXI1 interrupt. Sets the reception complete flag. |
| **Arguments** | None |
| **Return values** | None |

| Excep_SCI1_TXI1 | |
|---|---|
| **Overview** | SCI1 transmit data empty interrupt handler |
| **Header** | None |
| **Declaration** | static 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 transmission complete interrupt handler |
| **Header** | None |
| **Declaration** | static void Excep_SCI1_TEI1(void) |
| **Description** | Disables transmission and disables the TEI1 interrupt. Sets the transmission complete flag. |
| **Arguments** | None |
| **Return values** | None |

| Excep_SCI1_ERI1 | |
|---|---|
| **Overview** | SCI1 receive error interrupt handler |
| **Header** | None |
| **Declaration** | static void Excep_SCI1_ERI1(void) |
| **Description** | Performs SCI1 reception error handling. |
| **Arguments** | None |
| **Return values** | None |
| **Remarks** | In this sample code, SCI1 reception errors are not handled. (The code goes into an infinite loop.)<br>If required, add the appropriate processing to this function. |

## 5.10 Flowcharts

### 5.10.1 Main Processing

Figure 5.4 shows the flowchart for the main processing.

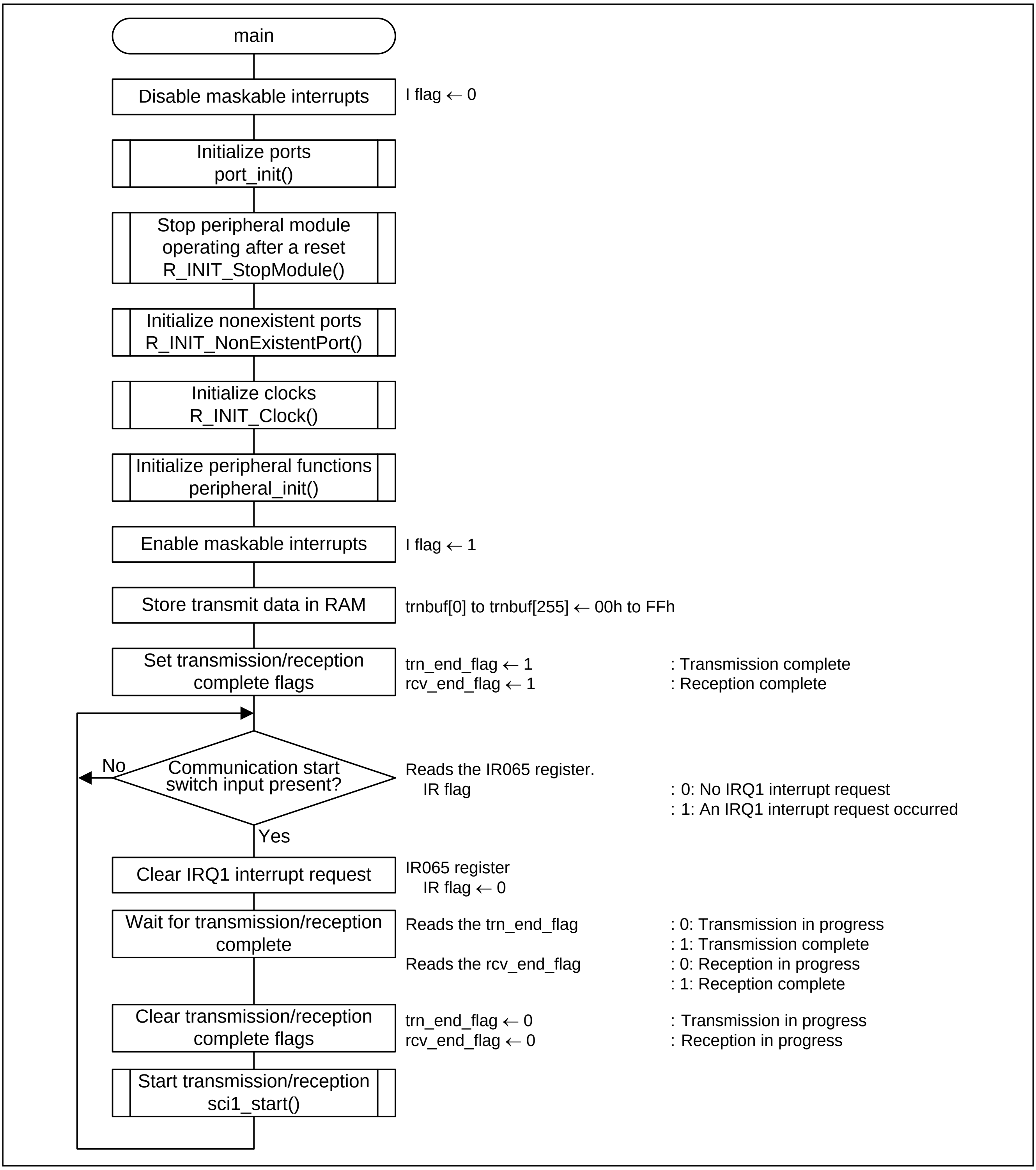



**Figure 5.4 Main Processing**

### 5.10.2 Port Initialization

Figure 5.5 shows the flowchart for port initialization.

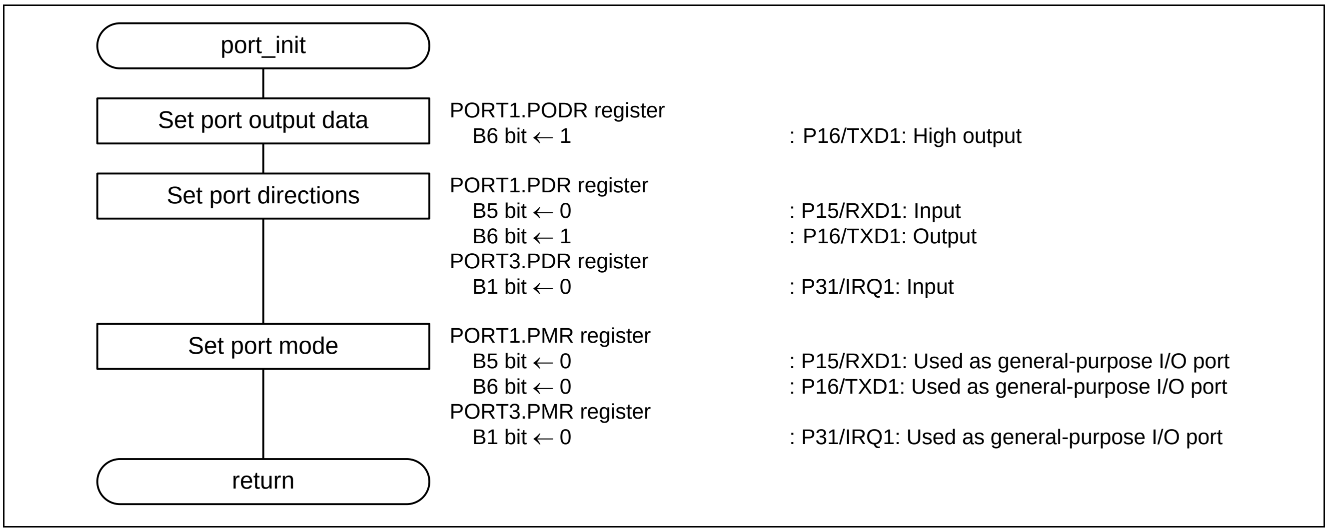



**Figure 5.5 Port Initialization**

### 5.10.3 Peripheral Function Initialization

Figure 5.6 shows the flowchart for peripheral function initialization.

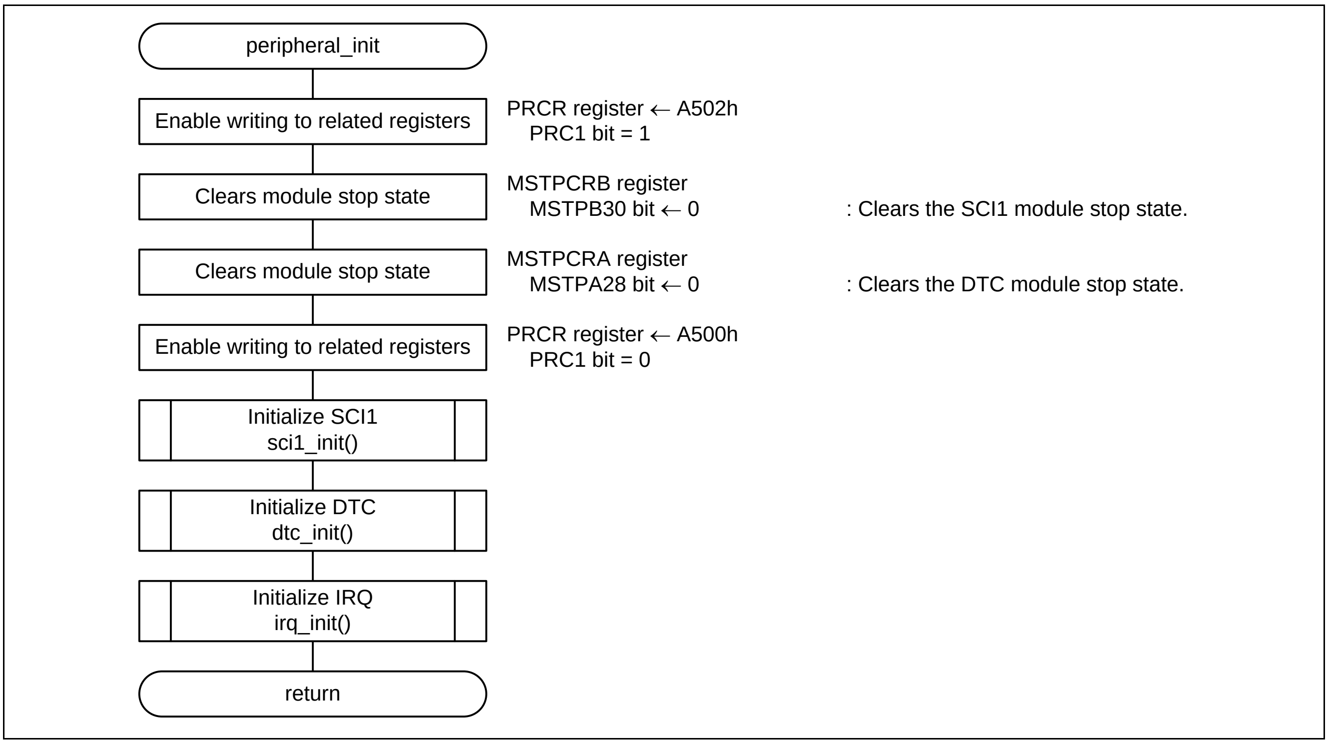



**Figure 5.6 Peripheral Function Initialization**

### 5.10.4 SCI1 Initialization

Figure 5.7 shows the flowchart for SCI1 initialization.

| Process | Setting | Description |
|---|---|---|
| sci1_init | | |
| Disable SCI1 interrupt requests | IER1B register | |
| | IEN2 bit ← 0 | : Disables SCI1.ERI1 interrupt request |
| | IEN3 bit ← 0 | : Disables SCI1.RXI1 interrupt request |
| | IEN4 bit ← 0 | : Disables SCI1.TXI1 interrupt request |
| | IEN5 bit ← 0 | : Disables SCI1.TEI1 interrupt request |
| Disable transmission/reception interrupt requests*1 | SCI1.SCR register ← 00h | |
| | TEIE bit = 0 | : Disables TEI1 interrupt request |
| | RE bit = 0 | : Disables serial reception operation |
| | TE bit = 0 | : Disables serial transmission operation |
| | RIE bit = 0 | : Disables RXI and ERI interrupt requests |
| | TIE bit = 0 | : Disables TXI interrupt request |
| Set up I/O port functions | PWPR register | |
| | B0WI bit ← 0 | : Enables writing the PFSWE bit. |
| | PWPR register | |
| | PFSWE bit ← 1 | : Enables writing the PFS register |
| | P15PFS register ← 0Ah | |
| | PSEL[3:0] bits = 1010b | : Selects P15 pin function: RXD1 |
| | P16PFS register ← 0Ah | |
| | PSEL[3:0] bits = 1010b | : Selects P16 pin function: TXD1 |
| | PWPR register | |
| | PFSWE bit ← 0 | : Disables writing the PFS register |
| | PWPR register | |
| | B0WI bit ← 1 | : Disables writing the PFSWE bit |
| | PORT1.PMR register | |
| | B5 bit ← 1 | : P15/RXD1: Used as peripheral functions |
| Select clock | SCI1.SCR register | |
| | CKE[1:0] bits ← 00b | : Internal baud rate generator |
| Set transmission/reception formats | SCI1.SMR register ← 00h | |
| | CKS[1:0] bits = 00b | : PCLK clock |
| | MP bit = 0 | : Disables multiplexer communication functions |
| | STOP bit = 0 | : One stop bit |
| | PE bit = 0 | : No parity |
| | CHR bit = 0 | : Transmission and reception with 8-bit data length |
| | CM bit = 0 | : Operates in asynchronous mode |
| | SCI1.SCMR register ← F2h | |
| | SMIF bit = 0 | : Serial communications interface mode |
| | SINV bit = 0 | : Transmits the contents of the TDR register without modification, and stores the receive data in the RDR register without modification. |
| | SDIR bit = 0 | : LSB first transmission and reception |
| | SCI1.SEMR register ← 00h | |
| | ABCS bit = 0 | : The one-bit period for the transfer rate is 16 cycles of the base clock. |
| | NFEN bit = 0 | : Disables the RXDn input signal noise exclusion function. |
| Set bit rate | [When RX220 is selected] | |
| | SCI1.BRR register ← 15 | :15.27 = (20 MHz / (64 × 2^(–1) × 38400 bps) ) – 1 |
| | [When RX21A is selected] | |
| | SCI1.BRR register ← 19 | :19.34 = (25 MHz / (64 × 2^(–1) × 38400 bps) ) – 1 |
| Set interrupt priority level | IPR218 register | |
| | IPR[3:0] bits ← 0001b | : Sets the SCI1.RXI1, TXI1, and TEI1 interrupt priority levels to 1. |
| Clear interrupt requests | IR219 register | |
| | IR flag ← 0 | : No SCI1.RXI1 interrupts |
| | IR220 register | |
| | IR flag ← 0 | : No SCI1.TXI1 interrupts |
| return | | |

Note: 1. After writing a value to the SCR register, applications should verify that the written value can be read back.

**Figure 5.7 SCI1 Initialization**

### 5.10.5 DTC Initialization

Figure 5.8 shows the flowchart for DTC initialization.

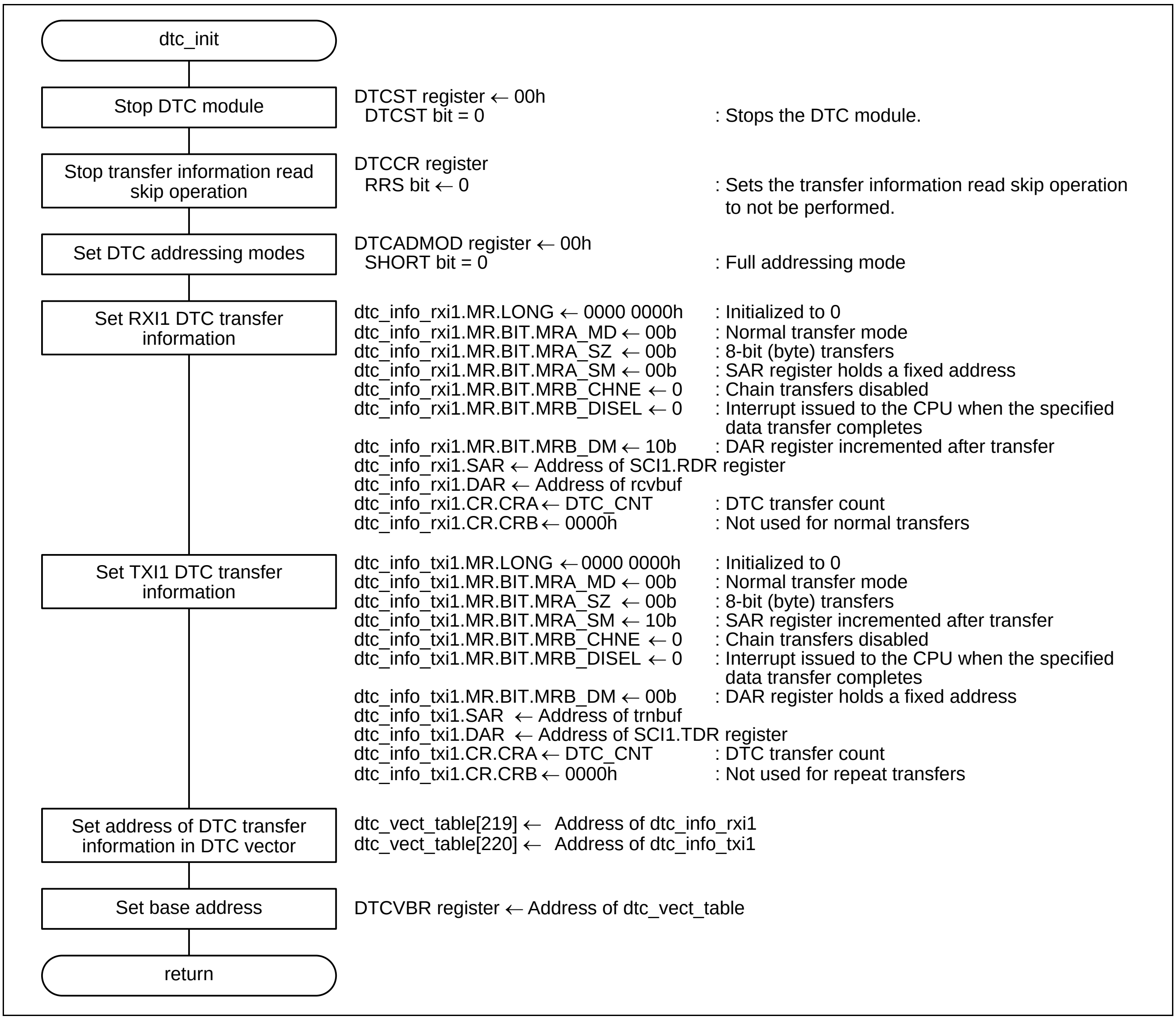



**Figure 5.8 DTC Initialization**

### 5.10.6 IRQ Initialization

Figure 5.9 shows the flowchart for IRQ initialization.

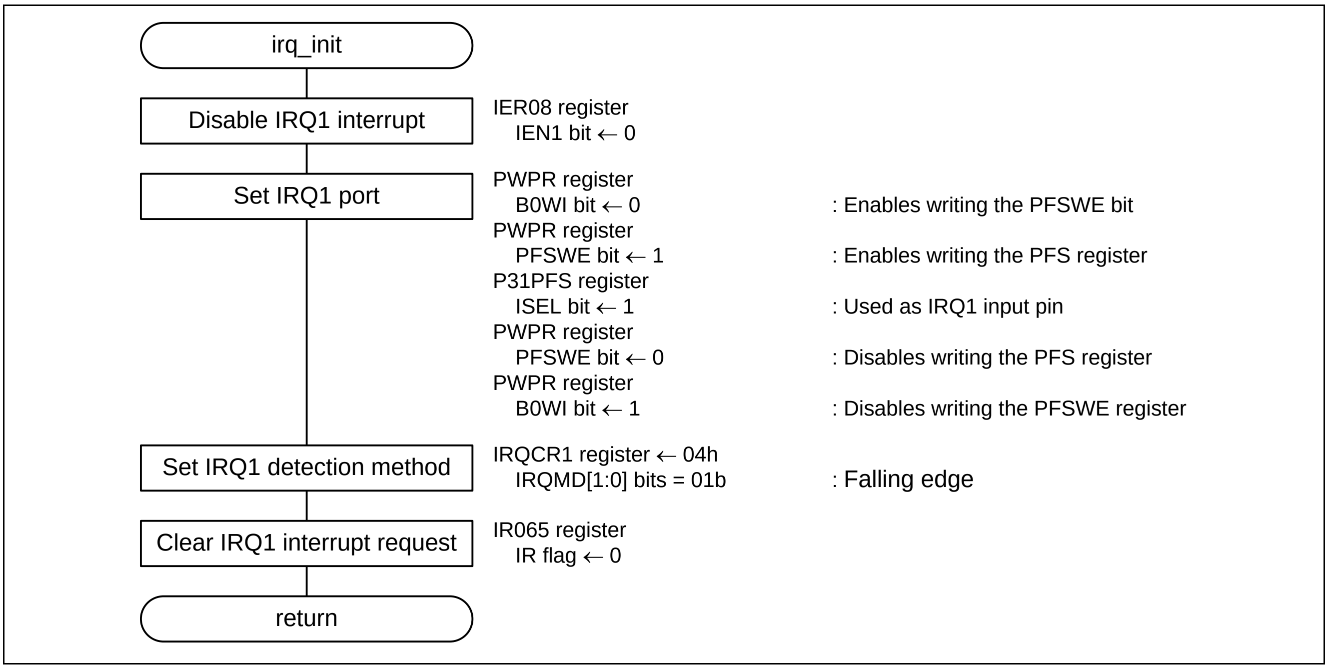



**Figure 5.9 IRQ Initialization**

### 5.10.7 Start SCI1 Communication

Figure 5.10 shows the flowchart for starting SCI1 communication.

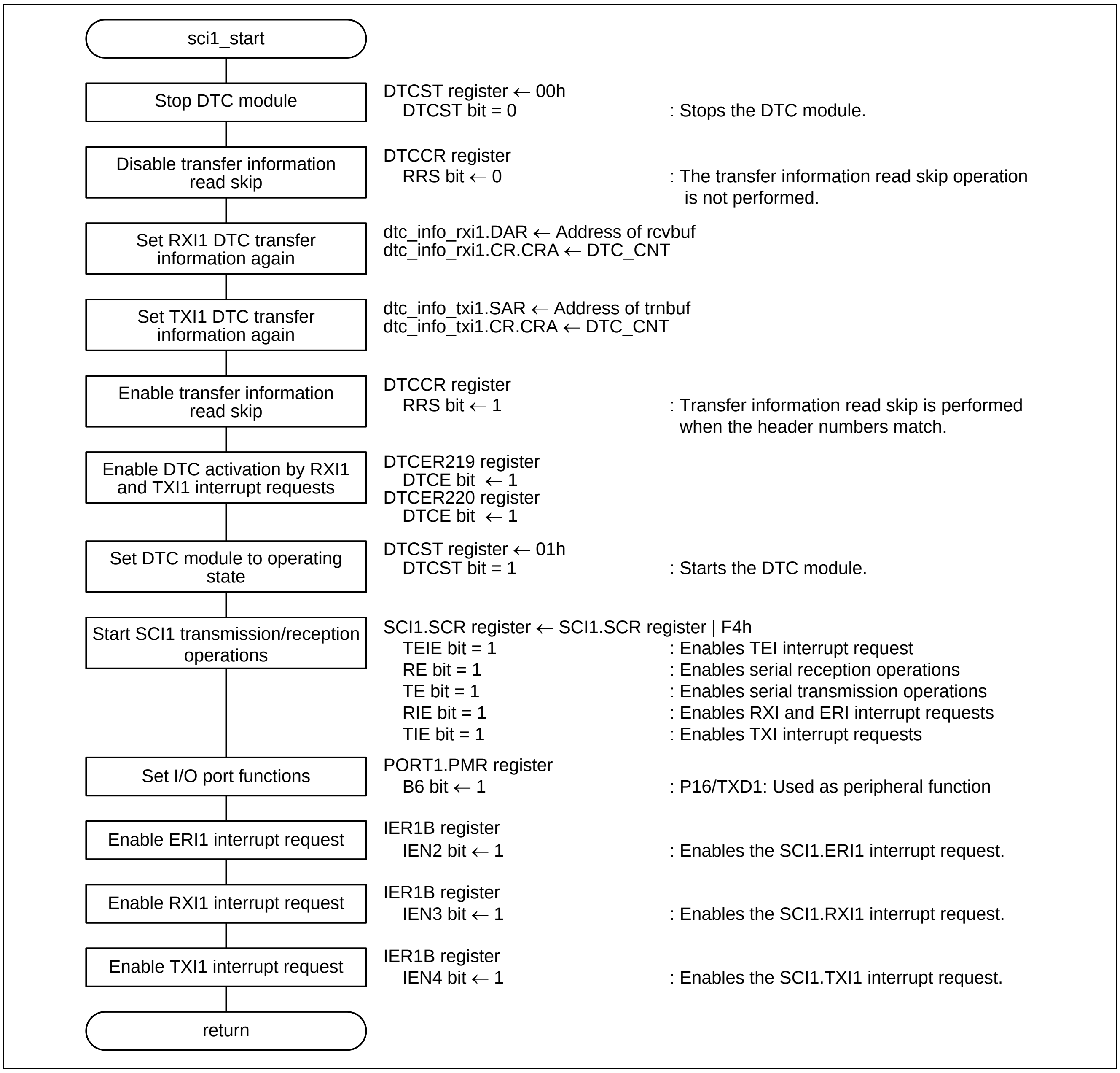
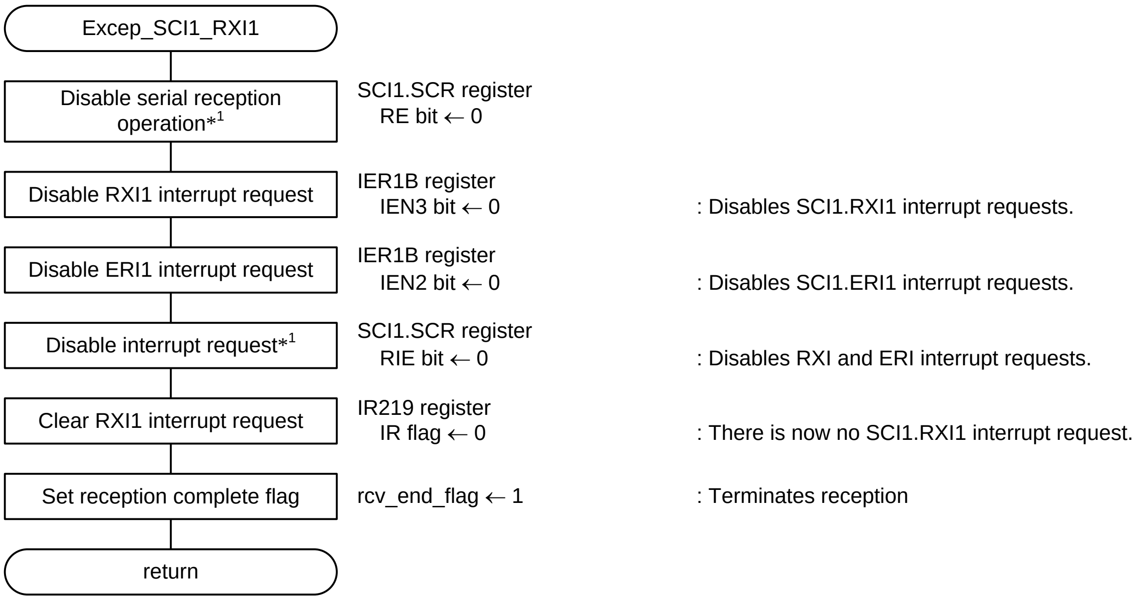



**Figure 5.10 Start SCI1 Communication**

### 5.10.8 SCI1 Receive Data Full Interrupt Handler

Figure 5.11 shows the flowchart for SCI1 receive data full interrupt handling.

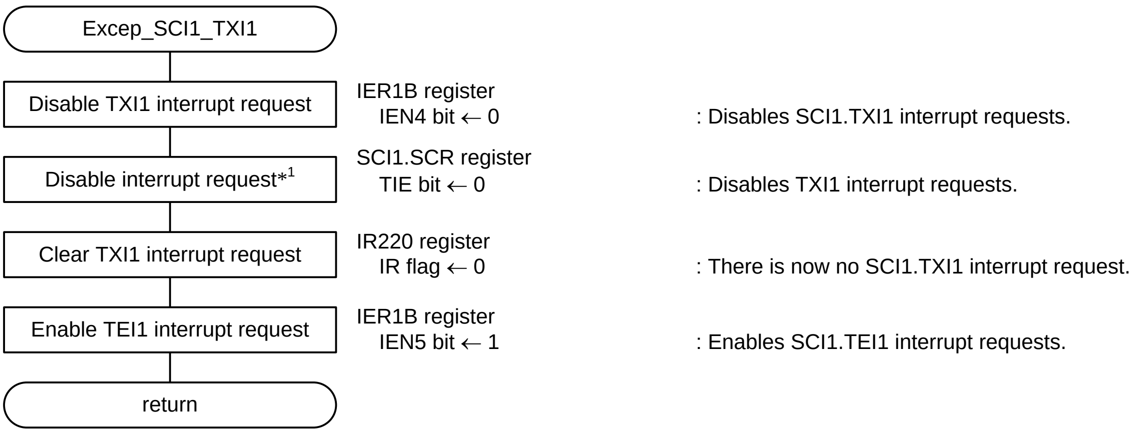



Note: 1. After writing the RE and RIE bits, applications should verify that the written value can be read back.

**Figure 5.11 SCI1 Receive Data Full Interrupt Handler**

### 5.10.9 SCI1 Transmit Data Empty Interrupt Handler

Figure 5.12 shows the flowchart for SCI1 transmit data empty interrupt handling.

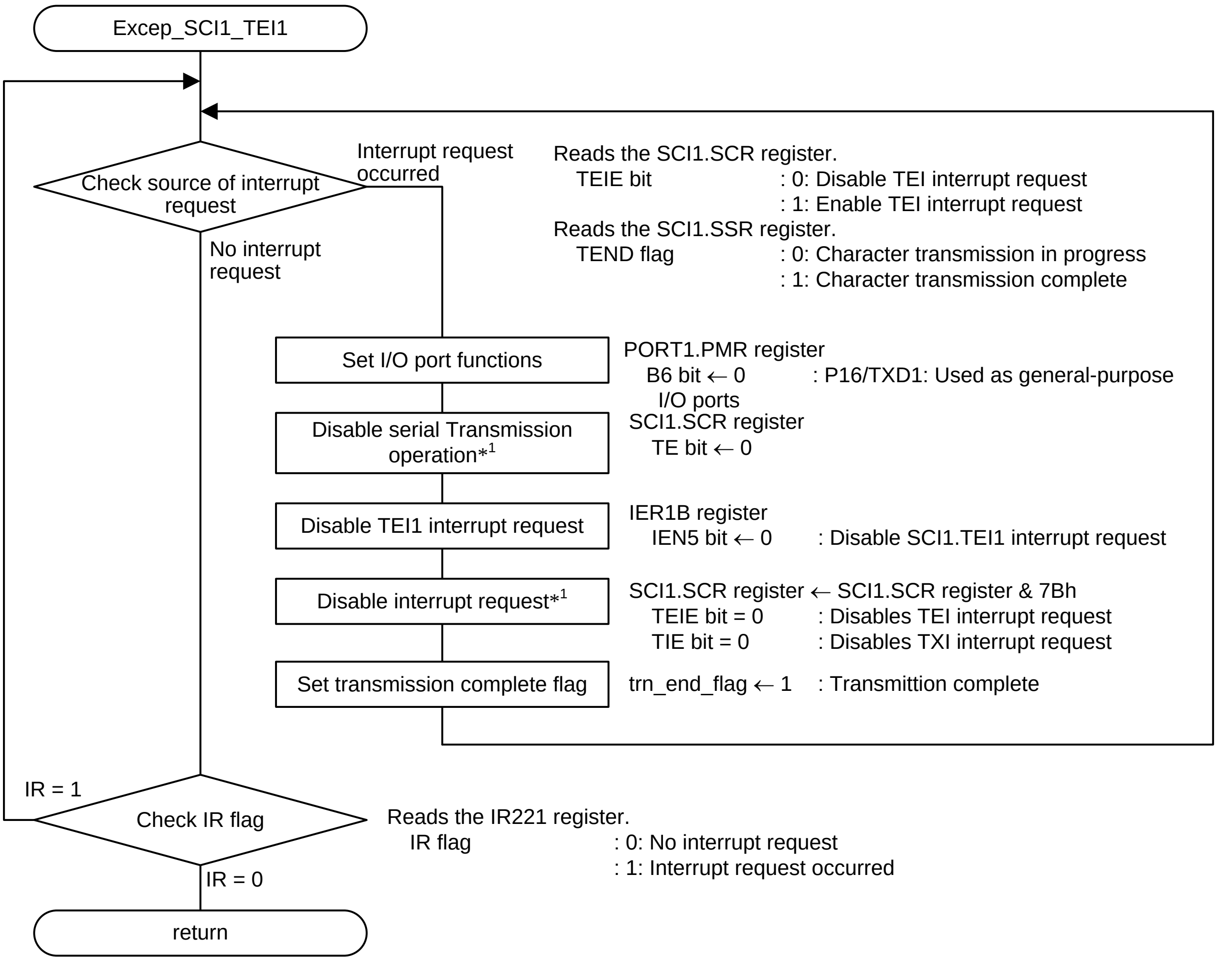



Note: 1. After writing the TIE bits, applications should verify that the written value can be read back.

**Figure 5.12 SCI1 Transmit Data Empty Interrupt Handler**

### 5.10.10 SCI1 Transmission Complete Interrupt Handler

Figure 5.13 shows the flowchart for SCI1 transmission complete interrupt handling.




Note: 1. After writing the SCR register, applications should verify that the written value can be read back.

**Figure 5.13 SCI1 Transmission Complete Interrupt Handler**

### 5.10.11 SCI1 Reception Error Interrupt Handler

Figure 5.14 shows the flowchart for SCI1 reception error interrupt handling.

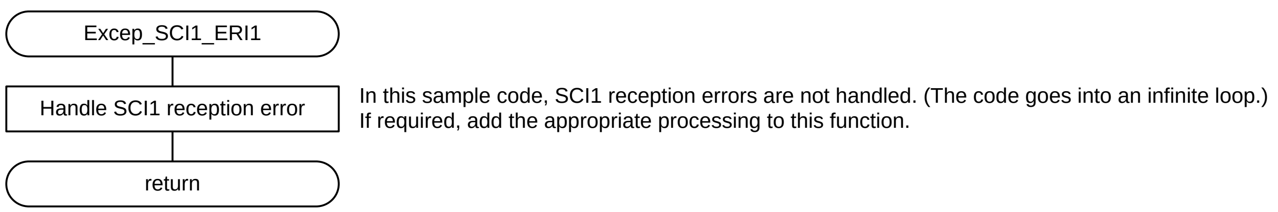



**Figure 5.14 SCI1 Reception Error Interrupt Handler**

# 6. Sample Code

The sample code can be downloaded from the Renesas Electronics Corporation web site.

# 7. Usage Notes

## 7.1 Sample Code Usage Notes

Either an RX220 Group or an RX21A Group device can be selected in the sample code. Use the following settings when selecting the device.

1. In the project tab in the workspace window in the High-performance Embedded Workshop, set the project for the device to be used to be the active project.
   Refer to the latest version of the High-performance Embedded Workshop user's manual for the procedure for setting the active project.

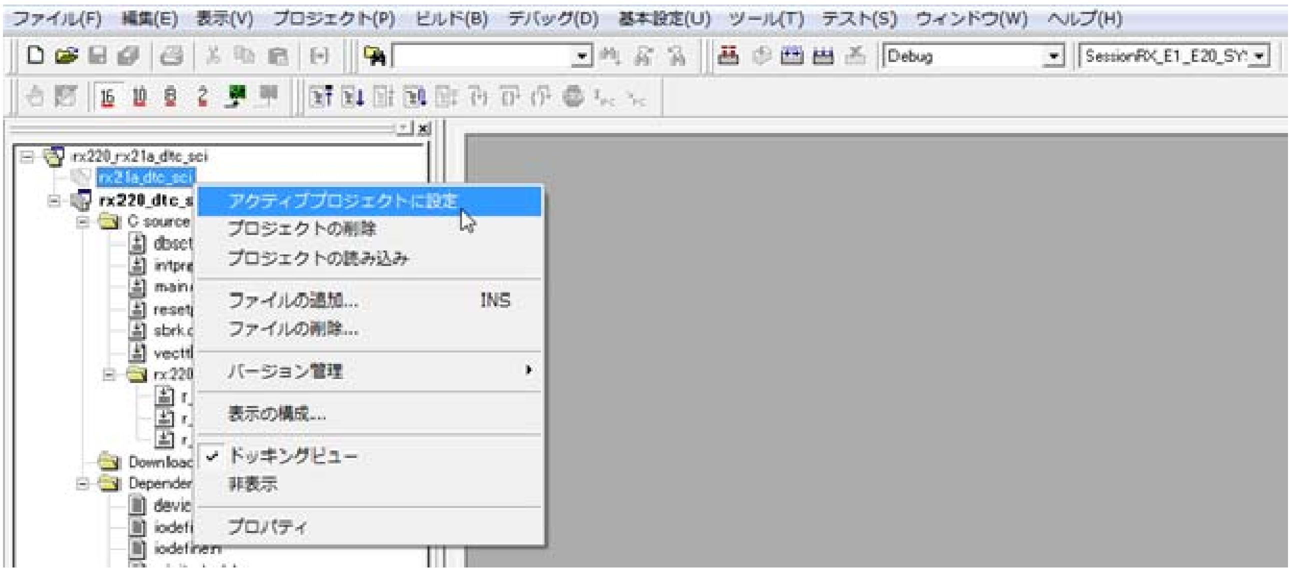



**Figure 7.1 Setting the Active Project**

2. Select the device used in the device configuration file (device_cfg.h). Uncomment the code for the device used and comment out the codes for unused devices.

## 7.2 Board Usage Notes

Keep the following points in mind when verifying the operation of the sample code for the board used as stipulated in this application note.

- Board used: Renesas Starter Kit for RX220 (R0K505220S000BE)
  Although the microcontroller ports P20 and P21 are connected to the RS-232C serial connector through the RS-232C transceiver when the Renesas Starter Kit for RX220 is shipped from the factory, since ports P20 and P21 do not have an SCI function, these connections must be changed to P15/RXD1 and P16/TXD1.
- Board used: Hokuto Denshi Co., Ltd. HSB Series Microcontroller Board (Catalog number: HSBRX21AP-B)
  No RS-232C transceiver or RS-232C serial connector are connected in the Hokuto Denshi Co., Ltd. HSB Series Microcontroller Board. To verify operation, the user must provide an RS-232C transceiver and an RS-232C serial connector.

## 8. Reference Documents

User's Manual: Hardware

RX220 Group User's Manual: Hardware Rev.1.00 (R01UH0292EJ)
RX21A Group User's Manual: Hardware Rev.1.00 (R01UH0251EJ)
The latest version 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.

High-performance Embedded Workshop V.4.09 User's Manual Rev.1.00 (R20UT0372EJ)
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.

## Website and Support

Renesas Electronics Website

http://www.renesas.com/

Inquiries

http://www.renesas.com/contact/

## Revision History

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Feb 14, 2014 | — | First edition issued |

## General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to a product with a different type 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.

RENESAS

**SALES OFFICES** Renesas Electronics Corporation http://www.renesas.com

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

**Renesas Electronics America Inc.**
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

**Renesas Electronics Canada Limited**
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

**Renesas Electronics Europe Limited**
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

**Renesas Electronics Europe GmbH**
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

**Renesas Electronics (China) Co., Ltd.**
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

**Renesas Electronics (Shanghai) Co., Ltd.**
Unit 301, Tower A, Central Towers, 555 LanGao Rd., Putuo District, Shanghai, China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

**Renesas Electronics Hong Kong Limited**
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

**Renesas Electronics Taiwan Co., Ltd.**
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

**Renesas Electronics Singapore Pte. Ltd.**
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

**Renesas Electronics Malaysia Sdn.Bhd.**
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

**Renesas Electronics Korea Co., Ltd.**
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141





Colophon 3.0