RX64M, RX71M Group

Using the DTC and SCIFA to Perform Asynchronous Serial Transmission and Reception

R01AN2213EJ0110
Rev. 1.10
Dec. 21, 2020

Abstract

This document describes using the data transfer controller (DTC) and the FIFO embedded serial communications interface (SCIF) in the RX64M, RX71M Group to perform asynchronous serial transmission and reception.

Products

RX64M Group
- RX64M Group 177- and 176-pin versions, ROM capacity: 2 MB to 4 MB
- RX64M Group 145- and 144-pin versions, ROM capacity: 2 MB to 4 MB
- RX64M Group 100-pin version, ROM capacity: 2 MB to 4 MB

RX71M Group
- RX71M Group 177- and 176-pin versions, ROM capacity: 2 MB to 4 MB
- RX71M Group 145- and 144-pin versions, ROM capacity: 2 MB to 4 MB
- RX71M Group 100-pin version, ROM capacity: 2 MB to 4 MB

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

This chapter explains using the SCIF to perform asynchronous serial transmission and reception.

Data to be transmitted is set to the RAM transmit data storage area in advance, and the DTC is used to transmit the data. The received data is stored in the RAM receive data storage area using the DTC.

Serial transmission and reception start when a falling edge on the interrupt request pin (IRQ5) is detected.

Bit 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 and Their Applications, and Figure 1.1 shows a Block Diagram.

Table 1.1 Peripheral Functions and Their Applications

Peripheral Function	Application
SCIF	Performs serial transmission and reception in asynchronous mode
DTC	Transfers data received in SCIFA9 to the RAM Transfers data to be transmitted from the RAM to SCIFA9
IRQ5	Start trigger for serial transmission and reception (normal transmission and reception)

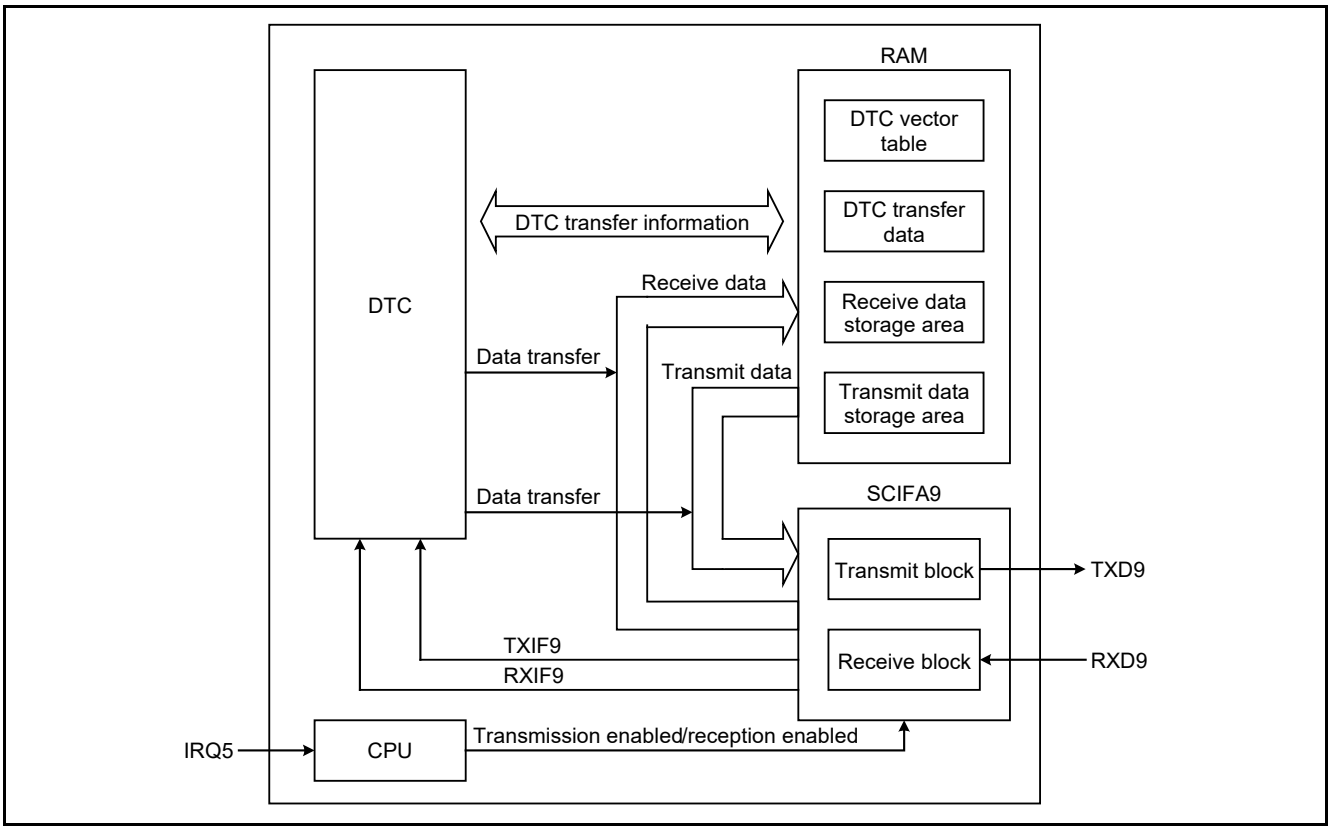


Figure 1.1 Block Diagram

2. Confirmed Operating Conditions

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

Table 2.1 Confirmed Operating Conditions

Item	Contents
MCU used	R5F564MLCDFC (RX64M Group)
Operating frequencies	• Main clock: 24 MHz • PLL clock: 240 MHz (main clock divided by 1 and multiplied by 10) • System clock (ICLK): 120 MHz[Note1] (PLL clock divided by 2) • Peripheral module clock B (PCLKB): 60 MHz (PLL clock divided by 4)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation e² studio Version: 2020-10
C compiler	Renesas Electronics Corporation C/C++ Compiler Package for RX Family V3.02.00[Note2]
	Compile options The integrated development environment default settings are used.
iodefine.h version	V0.9a
Endian	Little endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	Version 1.10
Board used	Renesas Start Kit+ for RX64M (product part no.: R0K50564MSxxxBE)

Note1: When setting the frequency of ICLK to faster than 120 MHz in RX71M, the value of the MEMWAIT register needs to be changed.

Note2: If the same version of the toolchain (C compiler) specified in the original project is not in the import destination, the toolchain will not be selected and an error will occur.
Check the selected status of the toolchain on the project configuration dialog.

For the setting method, refer to FAQ 3000404.

FAQ 3000404 :Program ""make"" not found in PATH' error when attempting to build an imported project (e² studio)"

3. Reference Application Note

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

- RX64M Group Initial Setting Rev. 1.00 (R01AN1918EJ)

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 current at the time this document was created. However, the latest version is always recommended. The latest version can be downloaded from the Renesas Electronics Corporation website.

4. Hardware

4.1 Hardware Configuration

Figure 4.1 shows a Connection Example for RX64M.

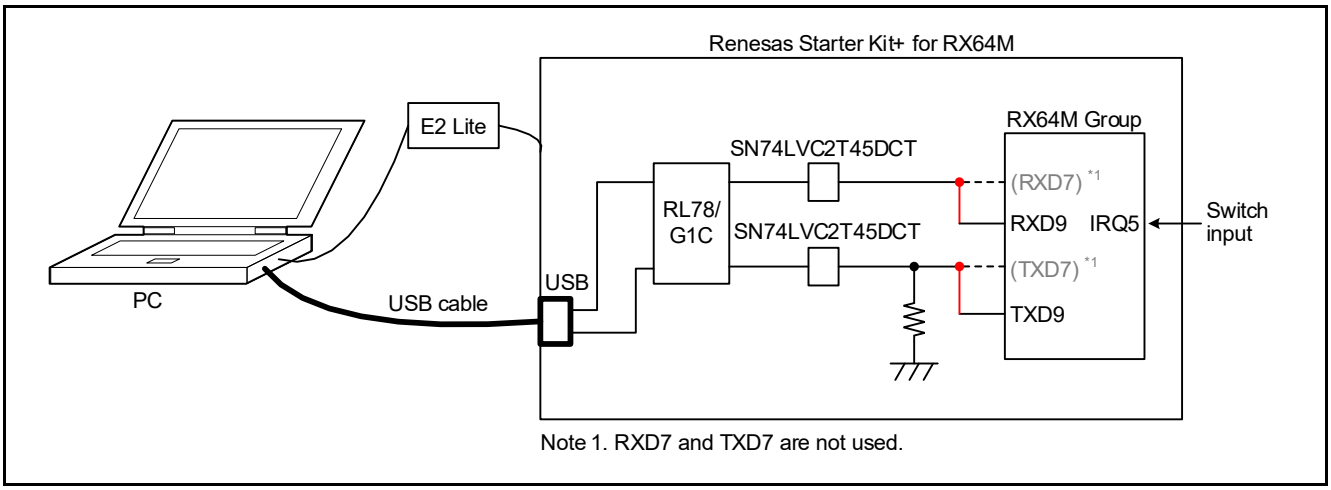


Figure 4.1 Connection Example

4.2 Notes on the Board Used

The Renesas Starter Kit+ for RX64M uses the USB as a virtual COM port to perform serial communication with the PC via the RL78/G1C.

In this application note, SCIFA9 is connected to RL78/G1C for communication with the PC instead of the default SCI7 being connected to RL78/G1C.When confirming operation, connect the RL78/G1C serial port to SCIFA9 (TXD9 and RXD9) that is the RX64M serial port.

4.3 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
P15/IRQ5	Input	Switch input to start transmission and reception (normal transmission)
PB6/RXD9	Input	Input pin for receive data in SCIFA9
PB7/TXD9	Output	Output pin for transmit data in SCIFA9

5. Software

The sample code uses the DTC to automatically perform SCIFA9 transmission and reception. When the switch to start transmission and reception is pushed, 32 blocks (256 bytes) of data are transmitted and received.

Transmission

If transmission is enabled while the amount of unsent data stored in the FTDR register is less than or equal to the transmit FIFO threshold value, the TXIF9 interrupt request is generated, triggering the DTC.

The DTC block transfers 1 block (8 bytes) of data to the FTDR register. Each time the amount of unsent data becomes less than or equal to the transmit FIFO threshold value, the TXIF9 interrupt request is generated, and the DTC is triggered. When the DTC transfers a total of 32 blocks (256 bytes), the TXIF9 interrupt request is output to the CPU. In this interrupt handling, the TXIF9 interrupt is disabled, and the TEIF9 interrupt is enabled.

When SCIFA9 transmission is complete, the TEIF9 interrupt is generated. In this interrupt handling, SCIFA9 transmission is disabled, the TEIF9 interrupt is disabled, and the transmission complete flag is set to 1.

Reception

When the amount of received data stored in the FRDR register becomes equal to or more than the receive FIFO threshold value, the RXIF9 interrupt request is generated, triggering the DTC.

The DTC block transfers 1 block (8 bytes) of data to the RAM's receive data storage area. Each time the amount of received data becomes equal to or greater than the receive FIFO threshold value, the RXIF9 interrupt request is generated, and the DTC is triggered. When the DTC transfers a total of 32 blocks (256 bytes), the RXIF9 interrupt request is output to the CPU. In this interrupt handling, SCIFA9 reception is disabled, the RXIF9 interrupt is disabled, and the reception complete flag is set to 1.

Settings for the peripheral functions that are used are as shown below.

SCIFA9

- Serial communication mode: Asynchronous
- Communication speed: 38,400 bps
- Clock source: PCLKB
- Data length: 8 bits
- Stop bits: 1 bit
- Parity: None
- Data transfer direction: LSB first
- Transmit FIFO threshold value: 8
- Receive FIFO threshold value: 8
- Interrupts: Receive error interrupt (ERIF9) enabled
  Receive FIFO data full interrupt (RXIF9) enabled
  Receive data ready interrupt (DRIF9) enabled
  Break detection or overrun interrupt (BRIF9) enabled
  Transmit FIFO data empty interrupt (TXIF9) enabled
  Transmit end interrupt (TEIF9) enabled

IRQ5 input pin

- Detection method: Falling edge
- Digital filter: Disabled
- Interrupts: Not used

DTC

- Trigger source: TXIF9 or RXIF9 interrupt request.
- DTC address mode: Full-address mode

  DTC transfer setting by the TXIF9 interrupt request
  - Transfer mode: Block transfer mode
  - Transfer source addressing mode: The SAR register is incremented after transfer.
  - Transfer source address: RAM (start address in the transmit data storage area)
  - Transfer destination addressing mode: Address in the DAR register is fixed.
  - Transfer destination address: SCIFA9.FTDR register
  - Data transfer size: 8 bits
  - DTC transfer mode: Transfer destination is a block area
  - Single block size: 8 times
  - Number of transfers: 32
  - Chain transfer: Disabled
  - Interrupts: An interrupt is sent to the CPU after the specified data has been transferred.

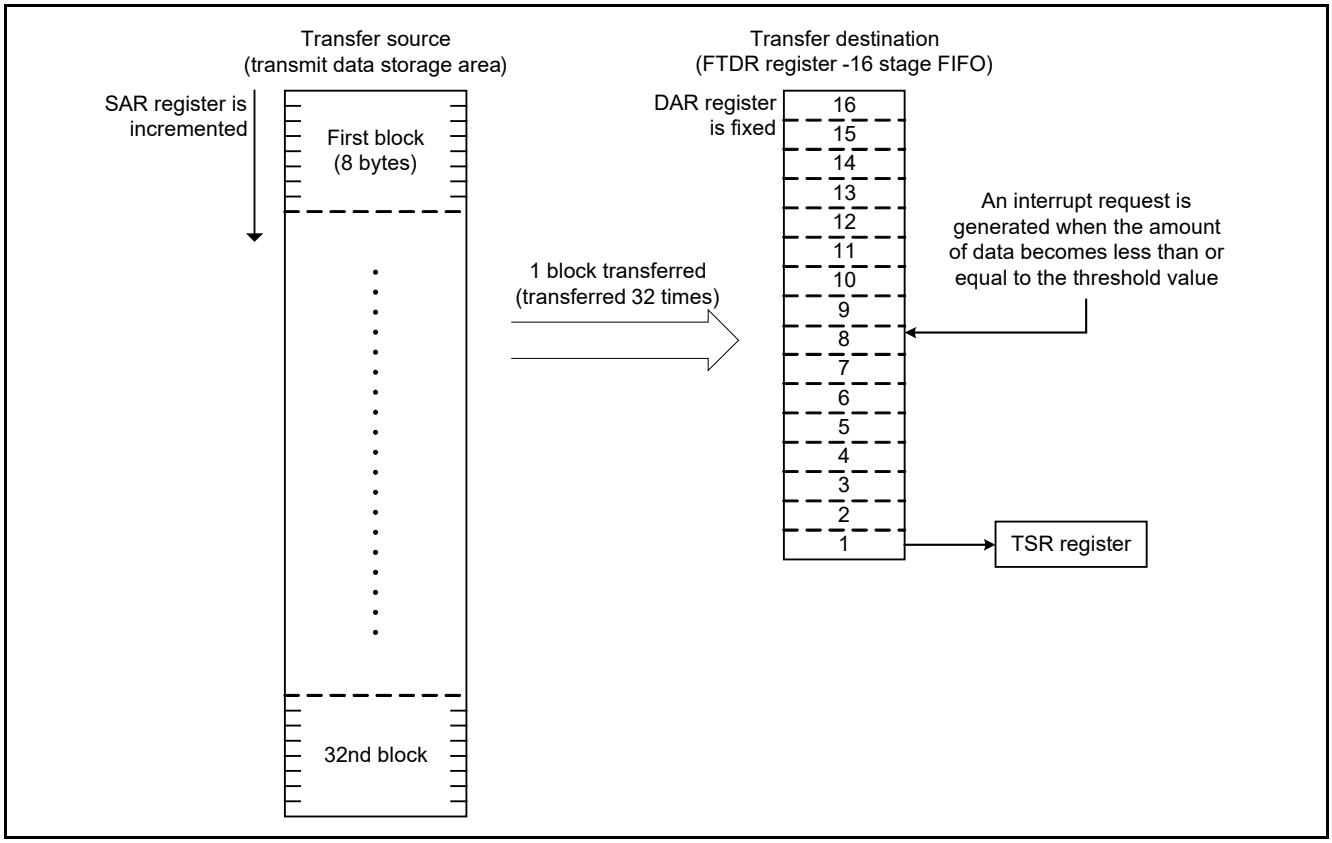


Figure 5.1 Overview of Data Transmission

DTC transfer setting by the RXIF9 interrupt request

- Transfer mode: Block transfer mode
- Transfer source addressing mode: The address in the SAR register is fixed.
- Transfer source address: SCIF9.FRDR register
- Transfer destination addressing mode: The DAR register is incremented after transfer.
- Transfer destination address: RAM (start address in the receive data storage area)
- Data transfer size: 8 bits
- DTC transfer mode: Transfer source is a block area
- Single block size: 8 times
- Number of transfers: 32
- Chain transfer: Disabled
- Interrupts: An interrupt is sent to the CPU after the specified data has been transferred.

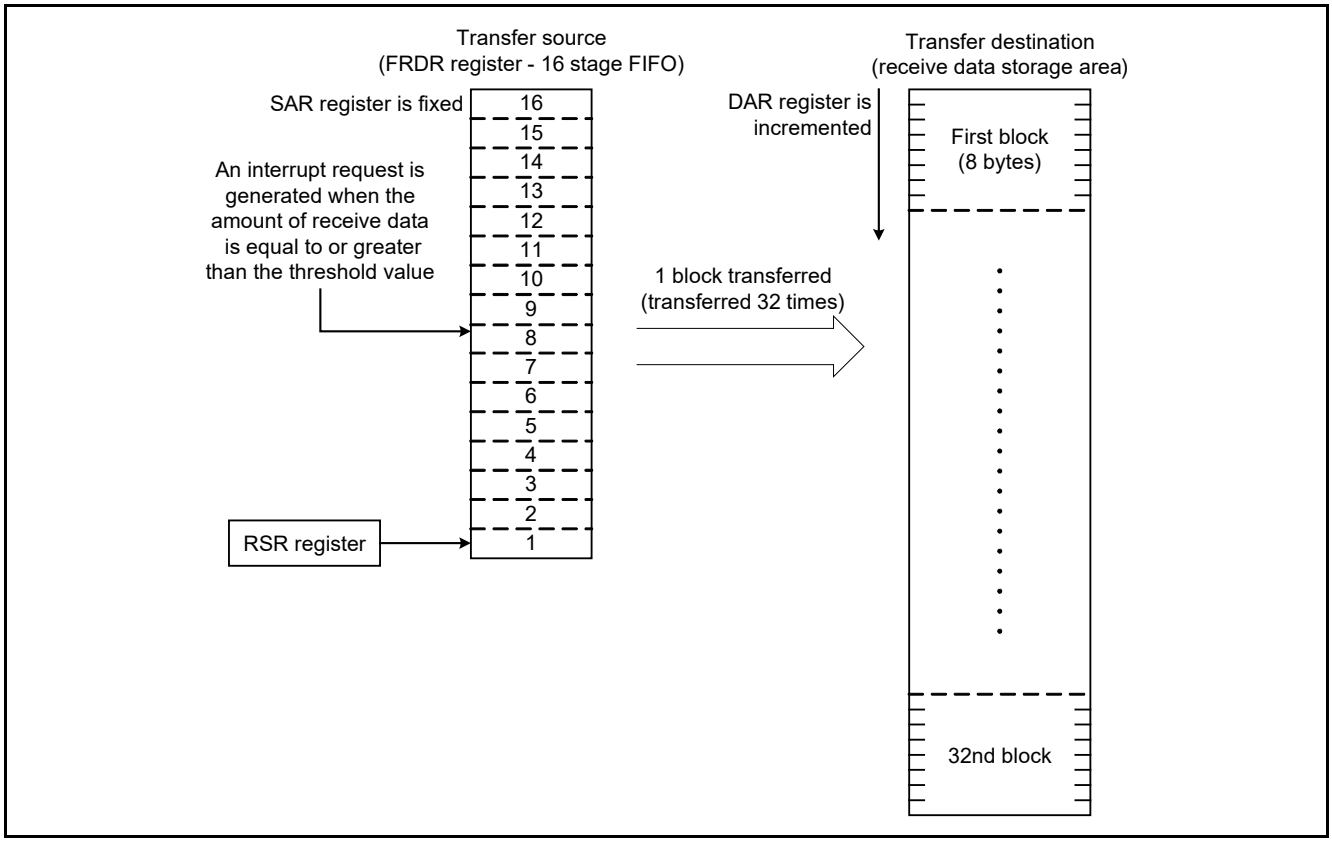


Figure 5.2 Overview of Data Reception

5.1 Operation Overview

5.1.1 Data Transmission

(1) Initialization
After initialization, the MCU waits for switch input to start transmission and reception.

(2) Detecting switch input to start transmission and reception
When switch input to start transmission and reception is detected, the IR flag for the IRQ5 interrupt is set to 0. Read the transmission complete flag and reception complete flag. When transmission and reception are confirmed to be complete, the transmission complete flag is set to 0 (transmitting data). The DTC transfer source address is set, the number of transfers is set, and the DTC start is enabled. The SCIFA9.SCR.REIE, TIE, RIE, TE, and RE bits are set to 1 simultaneously to enable transmission and reception. At this time, the amount of unsent data stored in the SCIFA9.FTDR register is equal to or less than the transmit FIFO threshold value, then the SCIFA9.SCR.TIE and TE bits are set to 1 simultaneously to set the IR flag for the TXIF9 interrupt to 1.

(3) Start data transfer
When the TXIF9 interrupt is enabled, one block of transmit data is block transferred from the RAM's transmit data storage area to the SCIFA9.FTDR register. The IR flag for the TXIF9 interrupt automatically becomes 0.

(4) Start data transmission
Data is transferred from the SCIFA9.FTDR register to the SCIFA9.TSR register, and the transmit data is output from the TXD9 pin. When the amount of unsent data stored in the SCIFA9.FTDR register becomes equal to or less than the transmit FIFO threshold value, the IR flag for the TXIF9 interrupt becomes 1. The TXIF9 interrupt request triggers the DTC, and the next transmit data is transferred.

(5) TXIF9 interrupt
Step (4) is repeated until 32 blocks of data have been transferred, at which time the TXIF9 interrupt request is generated and sent to the CPU. In this interrupt handling, the TXIF9 interrupt is disabled, and the TEIF9 interrupt is enabled.

(6) TEIF9 interrupt
When SCIFA9 transmission is complete, the TEIF9 interrupt is generated. In this interrupt handling, SCIFA9 transmission is disabled, the TEIF9 interrupt is disabled, and the transmission complete flag becomes 1.

This process repeats starting from step (2).

Figure 5.3 shows the Timing Diagram for Data Transmission.

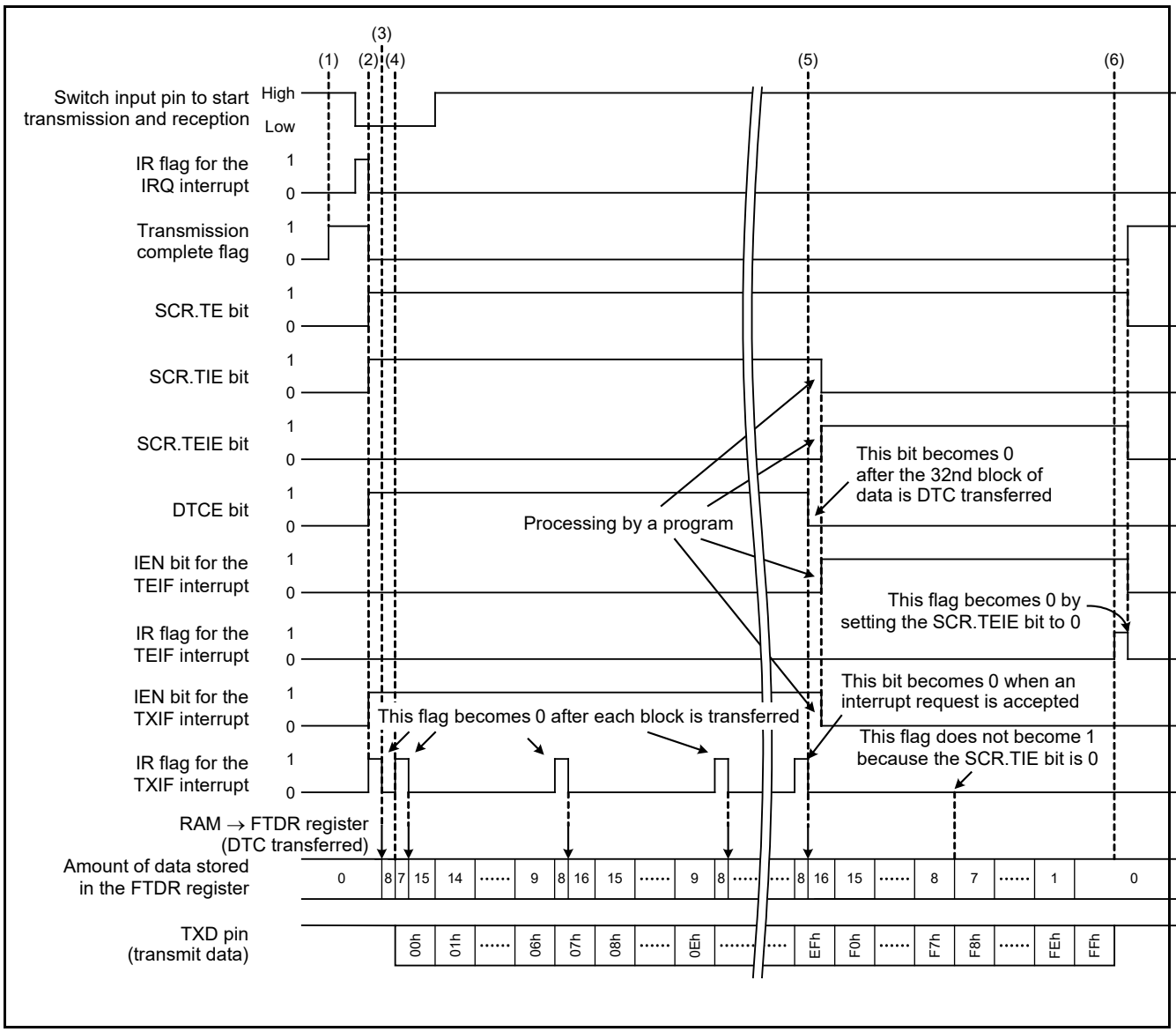


Figure 5.3 Timing Diagram for Data Transmission

5.1.2 Data Reception

(1) Initialization
After initialization, the MCU waits for switch input to start transmission and reception.

(2) Detecting switch input to start transmission and reception
When switch input to start transmission and reception is detected, the IR flag for the IRQ5 interrupt is set to 0. Read the transmission complete flag and reception complete flag. When transmission and reception are confirmed to be complete, the reception complete flag is set to 0 (receiving data). The DTC transfer source address is set, the number of transfers is set, and the DTC is enabled. The SCI7.SCR.REIE, TIE, RIE, TE, and RE bits are set to 1 simultaneously to enable transmission and reception, and the RXIF9 interrupt is enabled.

(3) Data reception complete
After 1 byte of data has been received, the data is transferred from the SCIFA9.RSR register to the SCIFA9.FRDR register. If the amount of data received in the SCIFA9.FRDR register is equal to or more than the receive FIFO threshold value, the IR flag for the RXIF9 interrupt becomes 1.

(4) Start data transfer
The RXIF9 interrupt request triggers the DTC. The IR flag for the RXIF9 interrupt automatically becomes 0. When the DTC is triggered, one block of data is transferred from the SCIFA9.FRDR register to the RAM's receive data storage area.

(5) RXIF9 interrupt
Step (4) is repeated until 32 blocks of data have been transferred, at which time the RXIF9 interrupt request is generated and sent to the CPU. In this interrupt handling, SCIFA9 reception is disabled, the RXIF9 interrupt is disabled, and the reception complete flag is set to 1.

This process repeats starting from step (2).

Figure 5.4 shows the Timing Diagram for Data Reception.

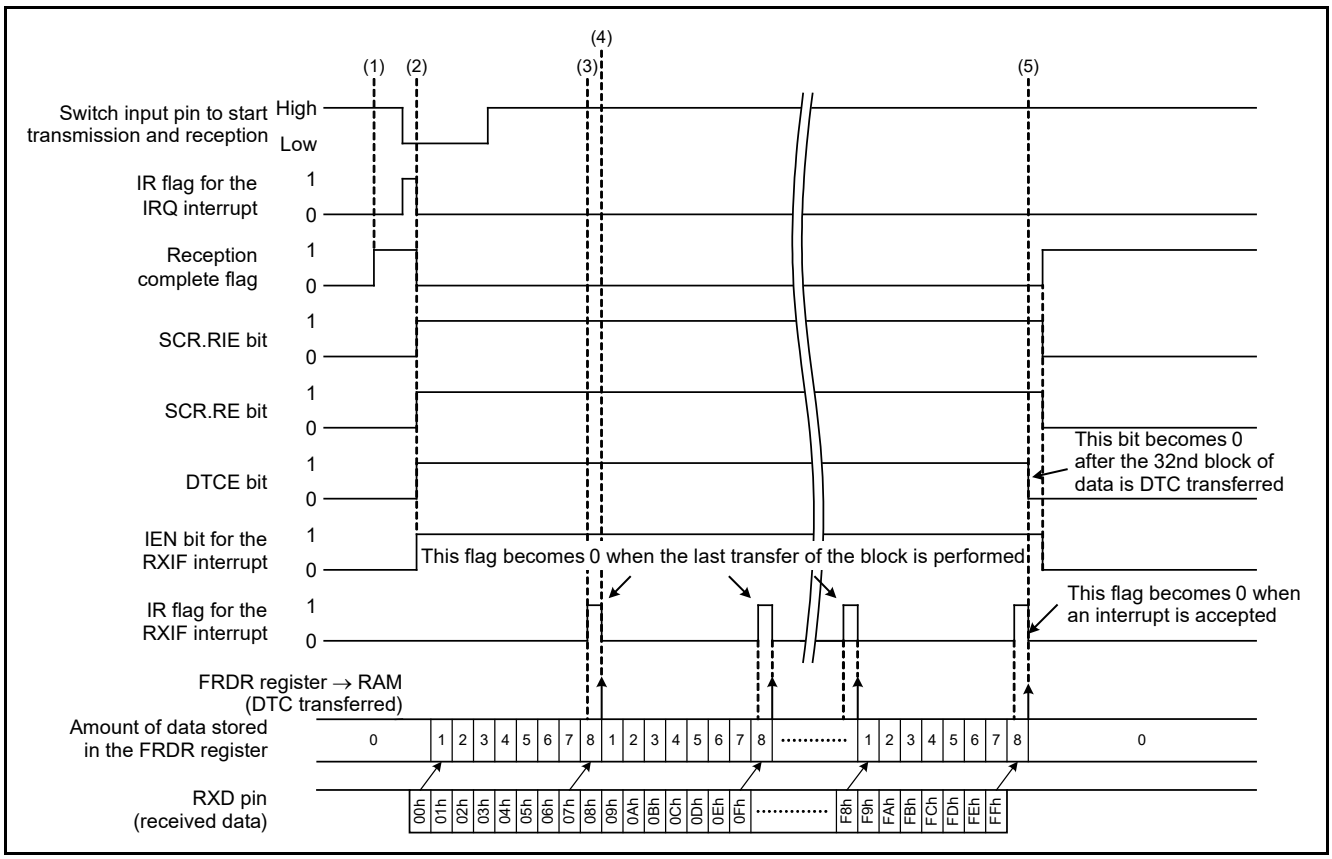


Figure 5.4 Timing Diagram for Data Reception

5.2 Section Composition

Table 5.1 lists the Section Data Changed in the Sample Code. For details on adding, changing, and deleting sections, refer to the RX Family C/C++ Compiler Package User's Manual.

Table 5.1 Section Data Changed in the Sample Code

Section Name	Change	Address	Function
BDTC_SECTION	Addition	0000 3000h	DTC vector table

5.3 File Composition

Table 5.2 lists the Files Used in the Sample Code. Files generated by the integrated development environment are not included in this table.

Table 5.2 Files Used in the Sample Code

File Name	Outline	Remarks
main.c	Main processing	
main.h	Header file for main.c	
r_init_stop_module.c	Stop processing for active peripheral functions after a reset	
r_init_stop_module.h	Header file for r_init_stop_module.c	
r_init_non_existent_port.c	Nonexistent port initialization	
r_init_non_existent_port.h	Header file for r_init_non_existent_port.c	
r_init_clock.c	Clock initialization	
r_init_clock.h	Header file for r_init_clock.c	

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

Symbol	Addresses	Setting Value	Contents
OFS0	0012 0068h to 0012 006Bh	FFFF FFFFh	The IWDT is stopped after a reset. The WDT is stopped after a reset.
OFS1	0012 006Ch to 0012 006Fh	FFFF FFFFh	The voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.
MDE	0012 0064h to 0012 0067h	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 Name	Setting Value	Contents
BUF_SIZE	256	Size of the storage area for transmit and receive data
BLOCK_SIZE	8	
DTC_CNT	BUF_SIZE/ BLOCK_SIZE	Number of DTC transfers

5.6 Structure/Union List

Figure 5.5 shows the Structure/Union Used in the Sample Code.

```
/* **** DTC transfer data **** */
#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 */
typedef struct
{
   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;
}st_dtc_full_t;
#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 5.5 Structure/Union Used in the Sample Code

5.7 Variables

Table 5.5 lists the static Variables.

Table 5.5 static Variables

Type	Variable Name	Contents	Function
static volatile uint8_t	trn_end_flag	Transmission complete flag 0: Transmitting data 1: Transmission complete	main excep_scifa9_txif9
static volatile uint8_t	rcv_end_flag	Reception complete flag 0: Receiving data 1: Reception complete	Main excep_scifa9_rxif9
static uint8_t	trnbuf[BUF_SIZE]	Transmit data storage area	main dtc_init scifa9_start
static uint8_t	rcvbuf[BUF_SIZE]	Receive data storage area	dtc_init scifa9_start
static struct st_dtc_full_t	dtc_info_rxif9	DTC transfer data with the RXIF9 interrupt	dtc_init scifa9_start
static struct st_dtc_full_t	dtc_info_txif9	DTC transfer data with the TXIF9 interrupt	dtc_init scifa9_start
static void *	pdtc_vect_table[256]	DTC vector table	dtc_init

5.8 Functions

Table 5.6 lists the Functions.

Table 5.6 Functions

Function Name	Outline
main	Main processing
port_init	Port initialization
R_INIT_StopModule	Stop processing for active peripheral functions after a reset
R_INIT_NonExistentPort	Nonexistent port initialization
R_INIT_Clock	Clock initialization
peripheral_init	Peripheral function initialization
scifa9_init	SCIFA9 initialization
dtc_init	DTC initialization
irq_init	IRQ initialization
scifa9_start	Start SCIFA9 transmission and reception
excep_scifa9_rxif9	SCIFA9 receive data full interrupt handling
excep_scifa9_txif9	SCIFA9 transmit data empty interrupt handling
excep_icu_groupal0	GROUPAL0 interrupt handling
excep_scifa9_teif9	SCIFA9 transmit end interrupt handling
excep_scifa9_erif9	SCIFA9 receive error interrupt handing
excep_scifa9_brif9	SCIFA9 break detection or overrun interrupt handling
excep_scifa9_drif9	SCIFA9 receive data ready interrupt handling

5.9 Function Specifications

The following tables list the sample code function specifications.

main	
Outline	Main processing
Header	None
Declaration	void main(void)
Description	After initialization, if switch input to start transmission and reception is detected, this function starts SCIFA9 transmission and reception.
Arguments	None
Return values	None

port_init	
Outline	Port initialization
Header	None
Declaration	static void port_init(void)
Description	This function initializes the ports.
Arguments	None
Return values	None

R_INIT_StopModule	
Outline	Stop processing for active peripheral functions after a reset
Header	r_init_stop_module.h
Declaration	void R_INIT_StopModule(void)
Description	This function configures settings to enter the module-stop state.
Arguments	None
Return values	None
Remarks	Transition to the module-stop state is not performed in the sample code. For more information on this function, refer to the RX64M Group Initial Setting Rev. 1.00 application note.

R_INIT_NonExistentPort	
Outline	Nonexistent port initialization
Header	r_init_non_existent_port.h
Declaration	void R_INIT_NonExistentPort(void)
Description	This function initializes port direction registers for ports that do not exist.
Arguments	None
Return values	None
Remarks	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 RX64M Group Initial Setting Rev. 1.00 application note.

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 values	None
Remarks	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 RX64M Group Initial Setting Rev. 1.00 application note.

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

scifa9_init	
Outline	SCIFA9 initialization
Header	None
Declaration	static void scifa9_init(void)
Description	This function initializes SCIFA9.
Arguments	None
Return values	None

dtc_init	
Outline	DTC initialization
Header	None
Declaration	static void dtc_init(void)
Description	This function initializes the DTC.
Arguments	None
Return values	None

irq_init	
Outline	IRQ initialization
Header	None
Declaration	static void irq_init(void)
Description	This function initializes IRQ5.
Arguments	None
Return values	None

scifa9_start	
Outline	Start SCIFA9 transmission and reception
Header	None
Declaration	static void scifa9_start(void)
Description	This function starts SCIFA9 transmission and reception.
Arguments	None
Return values	None

excep_scifa9_rxif9	
Outline	SCIFA9 receive data full interrupt handling
Header	None
Declaration	void excep_scifa9_rxif9 (void)
Description	This function disables reception, and disables the RXIF9, ERIF9, BRIF9, and DRIF9 interrupts. This function sets the reception complete flag.
Arguments	None
Return values	None

excep_scifa9_txif9	
Outline	SCIFA9 transmit data empty interrupt handling
Header	None
Declaration	void excep_scifa9_txif9 (void)
Description	This function disables the TXIF9 interrupt and enables the TEIF9 interrupt.
Arguments	None
Return values	None

excep_icu_groupal0	
Outline	GROUPAL0 interrupt handling
Header	None
Declaration	void excep_icu_groupal0 (void)
Description	This function performs GROUPAL0 interrupt handling.
Arguments	None
Return values	None

excep_scifa9_teif9	
Outline	SCIFA9 transmit end interrupt handling
Header	None
Declaration	void excep_scifa9_teif9 (void)
Description	This function disables transmission and disables the TEIF9 interrupt. This function sets the transmission complete flag.
Arguments	None
Return values	None

excep_scifa9_erif9	
Outline	SCIFA9 receive error interrupt handling
Header	None
Declaration	void excep_scifa9_erif9 (void)
Description	This function performs receive error processing.
Arguments	None
Return values	None
Remarks	Receive error processing is not performed in the sample code (infinite loop). Add processing to the program when necessary.

excep_scifa9_brif9	
Outline	SCIFA9 break detect or overrun error interrupt handling
Header	None
Declaration	void excep_scifa9_brif9 (void)
Description	This function performs the break detect and overrun error interrupt handling.
Arguments	None
Return values	None
Remarks	Break detection and overrun error interrupt handling are not performed in the sample code (infinite loop). Add processing to the program when necessary.

excep_scifa9_drif9	
Outline	SCIFA9 receive data ready interrupt handling
Header	None
Declaration	void excep_scifa9_drif9 (void)
Description	This function performs receive data ready interrupt handling.
Arguments	None
Return values	None
Remarks	Receive data ready interrupt handling is not performed in the sample code (infinite loop). Add processing to the program when necessary.

5.10 Flowcharts

5.10.1 Main Processing

Figure 5.6 shows the Main Processing.

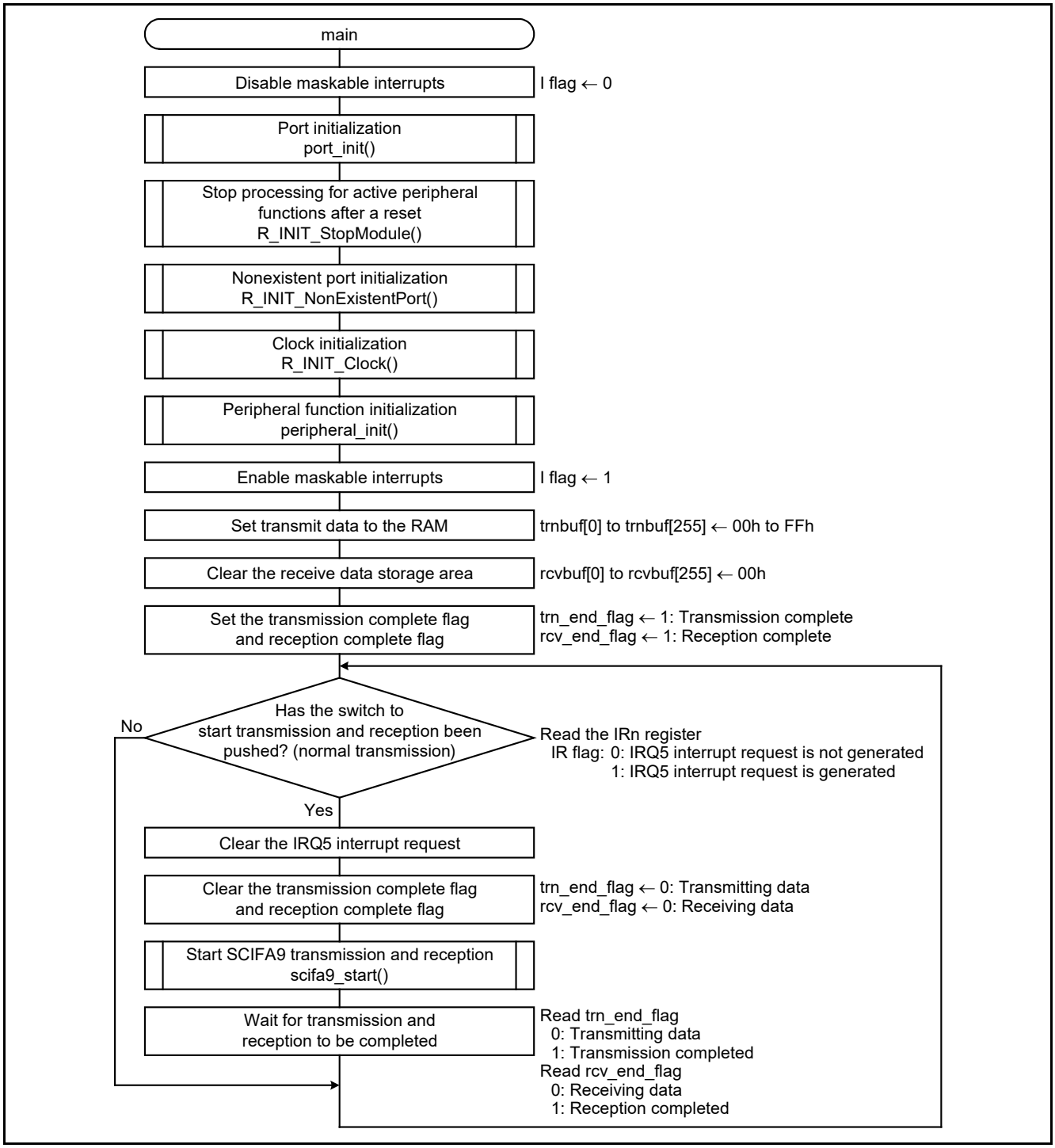


Figure 5.6 Main Processing

5.10.2 Port Initialization

Figure 5.7 shows Port Initialization.

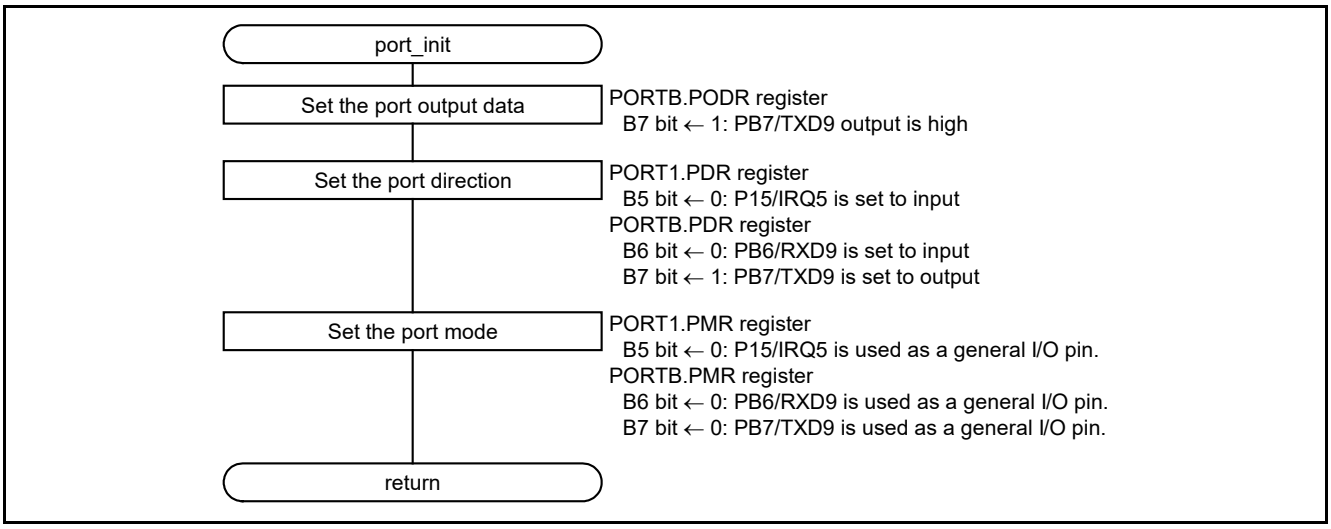


Figure 5.7 Port Initialization

5.10.3 Peripheral Function Initialization

Figure 5.8 shows Peripheral Function Initialization.

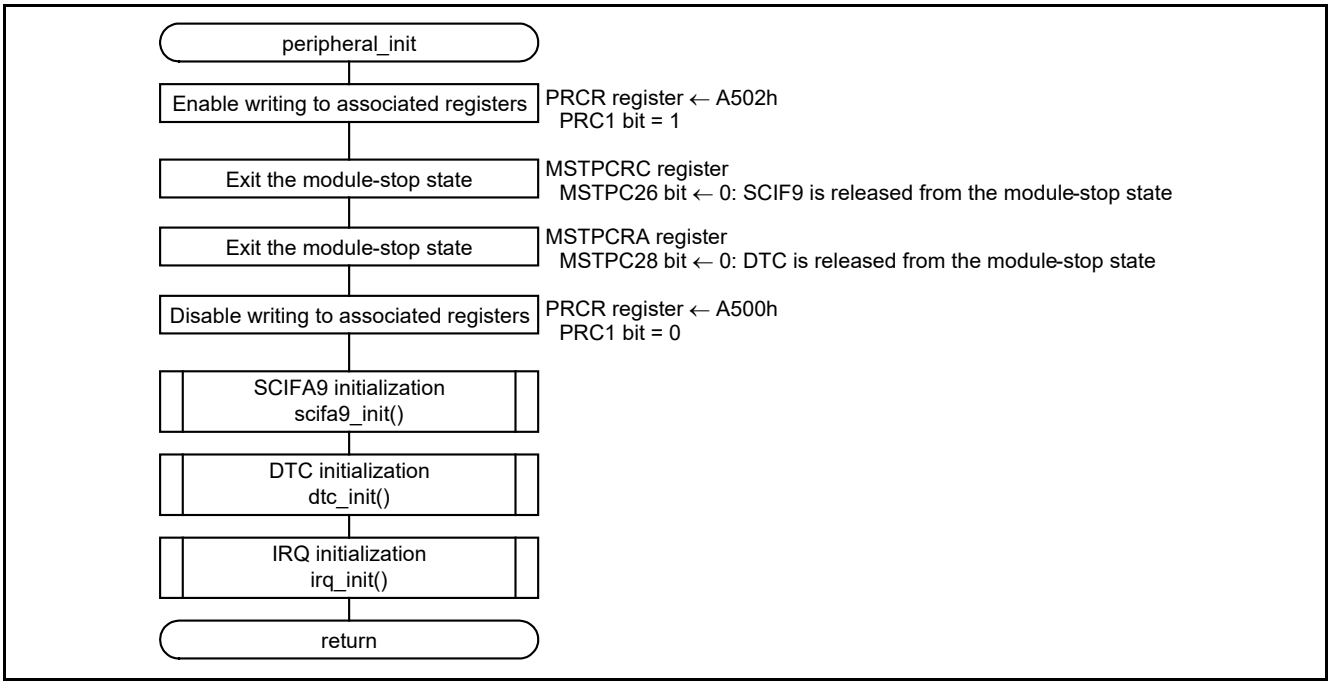


Figure 5.8 Peripheral Function Initialization

5.10.4 SCIFA9 Initialization

Figure 5.9 and Figure 5.10 show the SCIFA9 initialization.

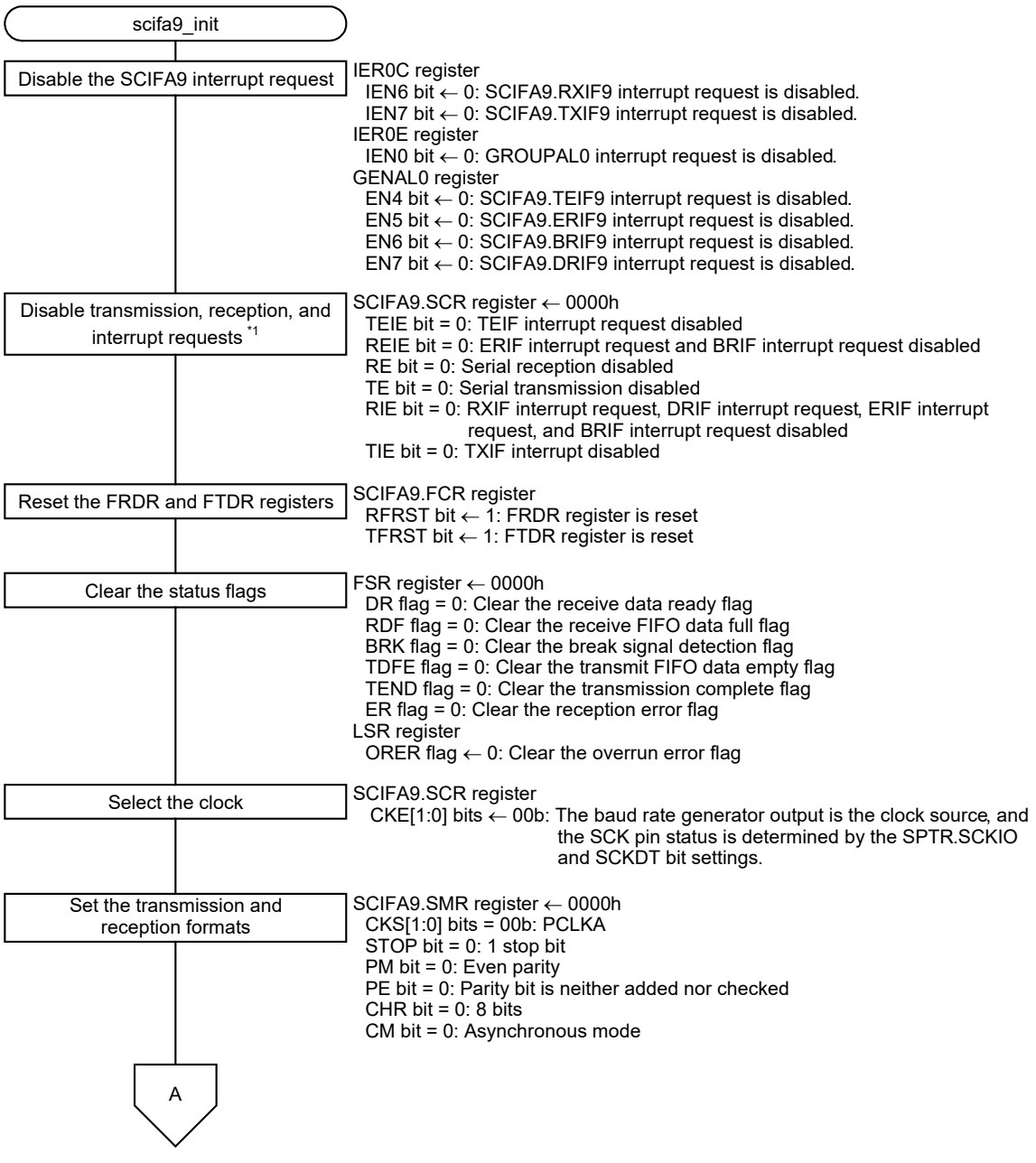
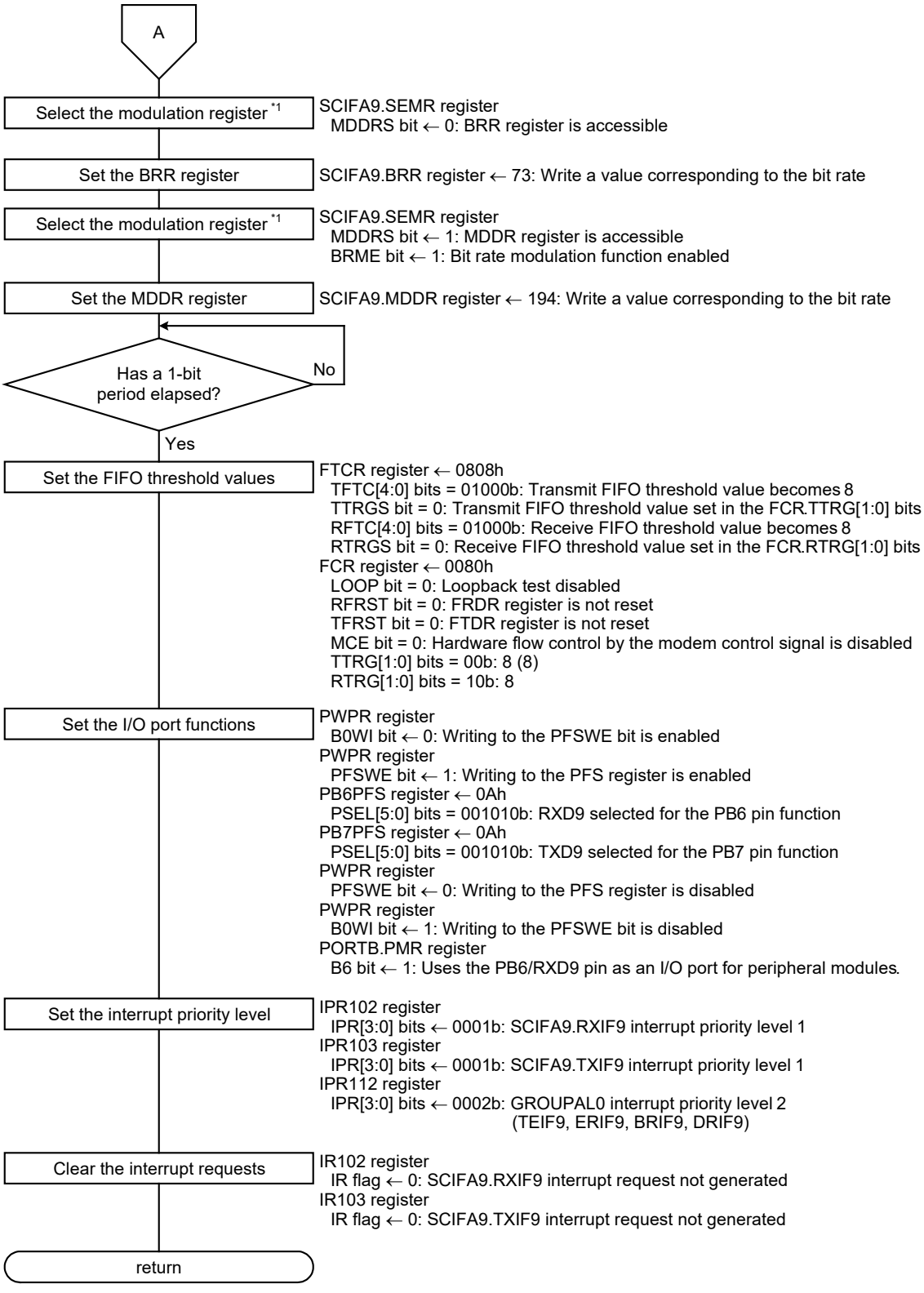


Figure 5.9 SCIFA9 Initialization (1/2)



Note 1. After writing to the MDDRS bit, confirm that the value written can be read.

Figure 5.10 SCIFA9 Initialization (2/2)

5.10.5 DTC Initialization

Figure 5.11 shows DTC Initialization.

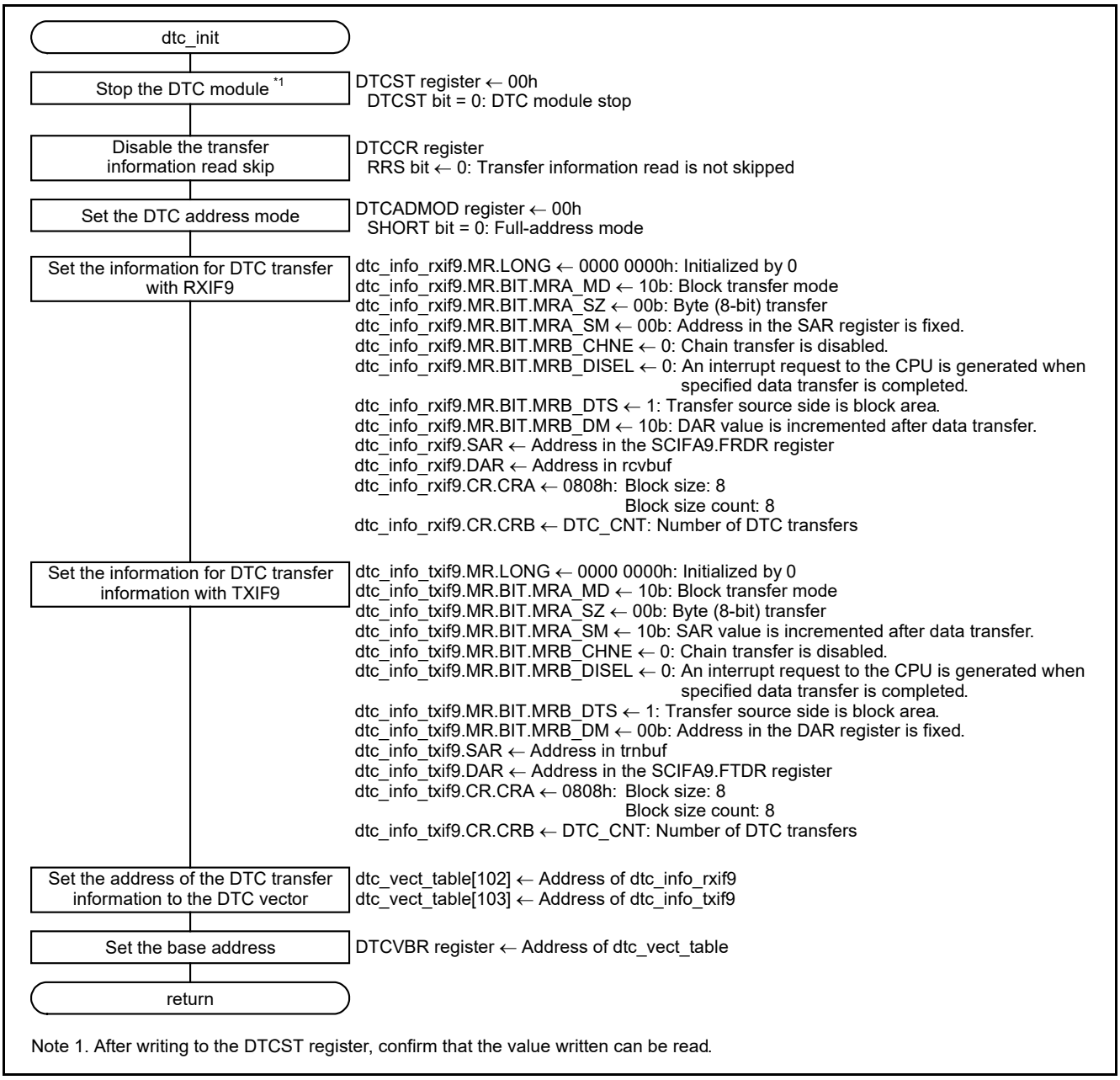


Figure 5.11 DTC Initialization

5.10.6 IRQ Initialization

Figure 5.12 shows IRQ Initialization.

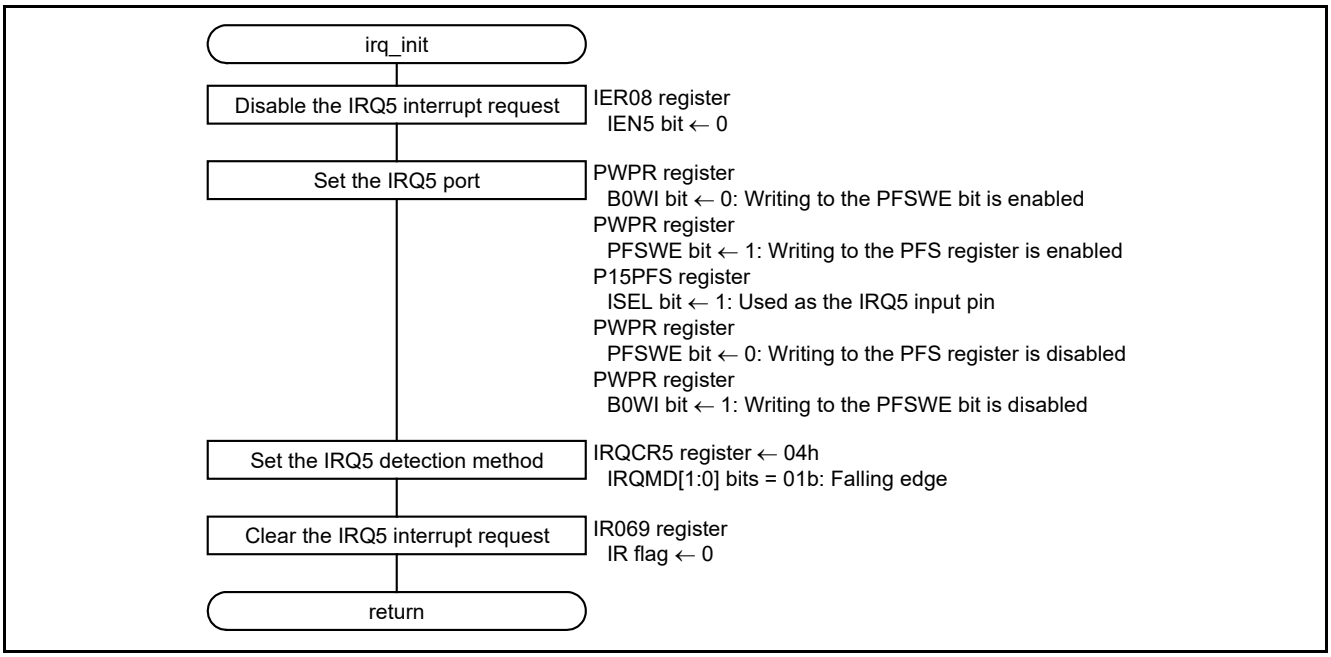
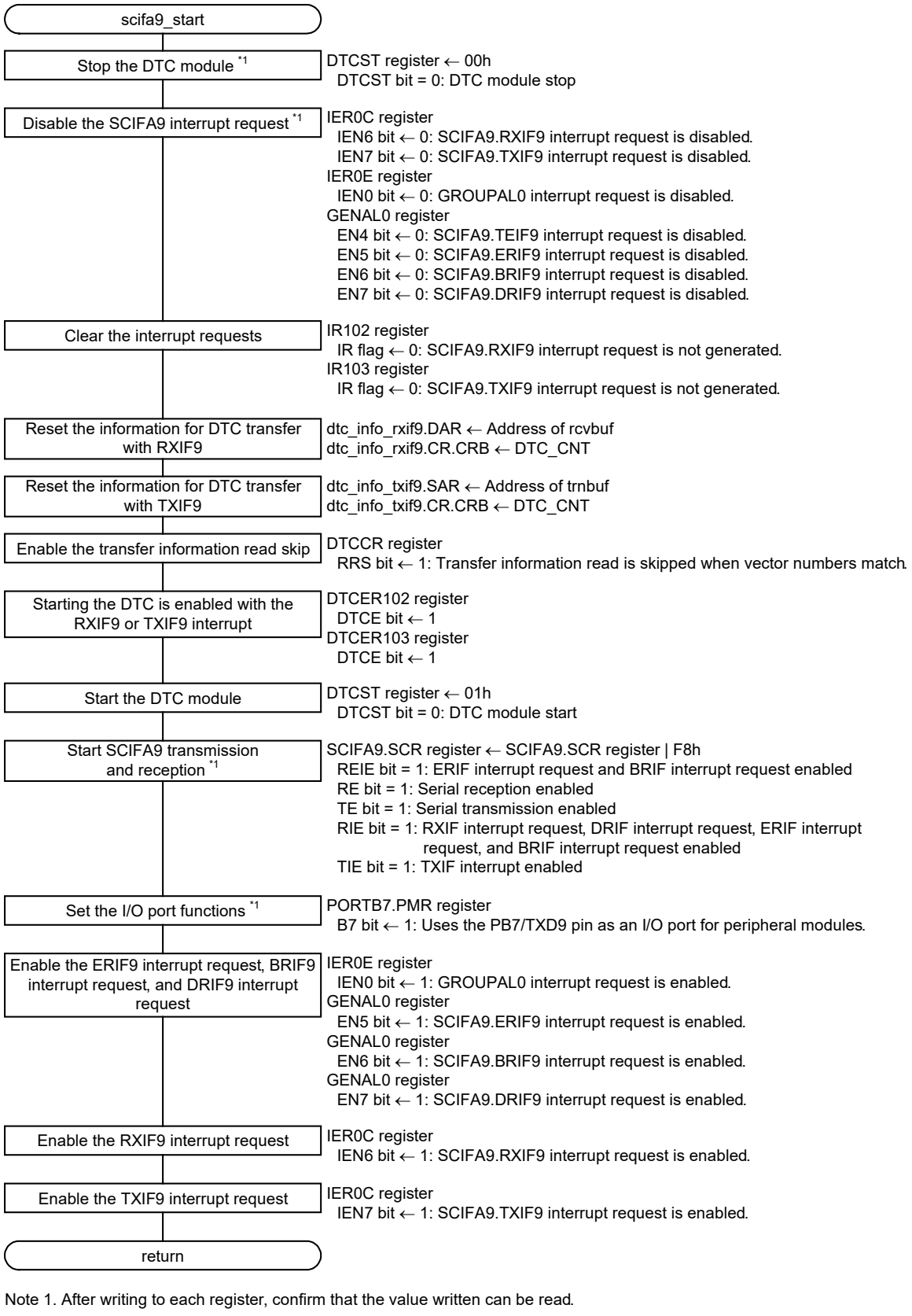


Figure 5.12 IRQ Initialization

5.10.7 Start SCIFA9 Transmission and Reception

Figure 5.13 shows Start SCIFA9 Transmission and Reception.



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

Figure 5.13 Start SCIFA9 Transmission and Reception

5.10.8 SCIFA9 Receive Data Full Interrupt Handling

Figure 5.14 shows SCIFA9 Receive Data Full Interrupt Handling.

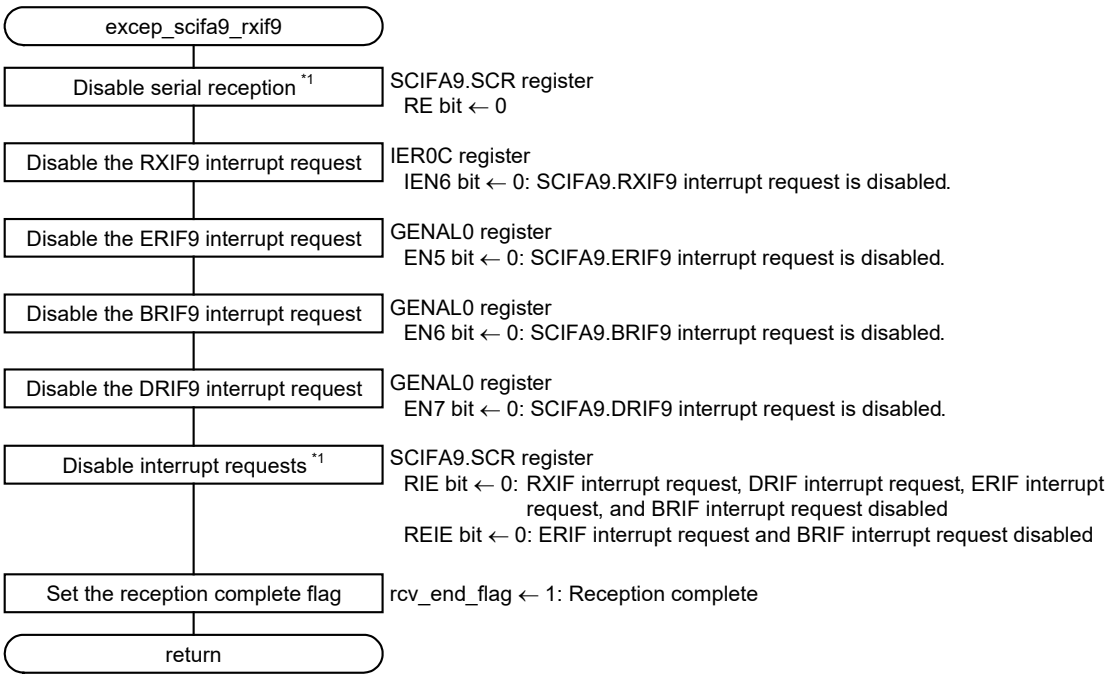


Figure 5.14 SCIFA9 Receive Data Full Interrupt Handling

5.10.9 SCIFA9 Transmit Data Empty Interrupt Handling

Figure 5.15 shows SCIFA9 Transmit Data Empty Interrupt Handling.

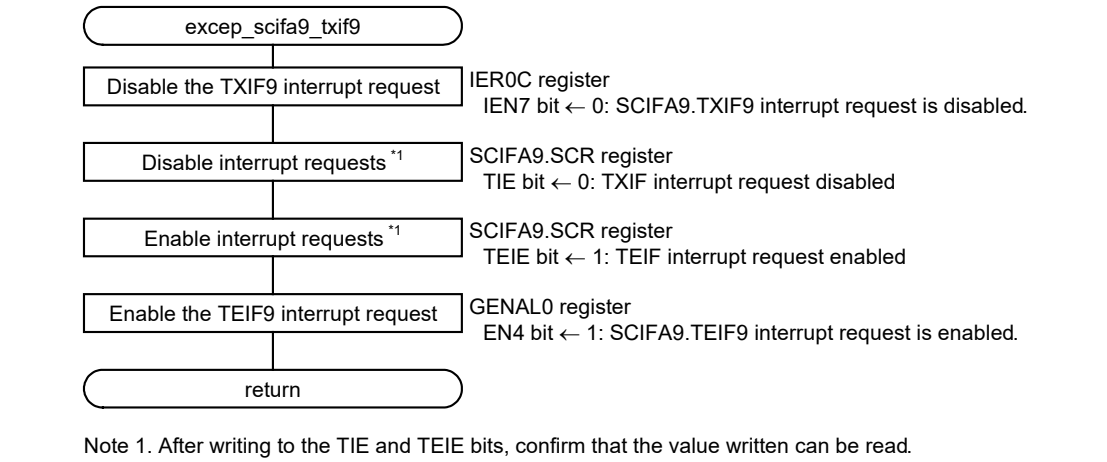


Figure 5.15 SCIFA9 Transmit Data Empty Interrupt Handling

5.10.10 GROUPAL0 Interrupt Handling

Figure 5.16 shows GROUPAL0 Interrupt Handling.

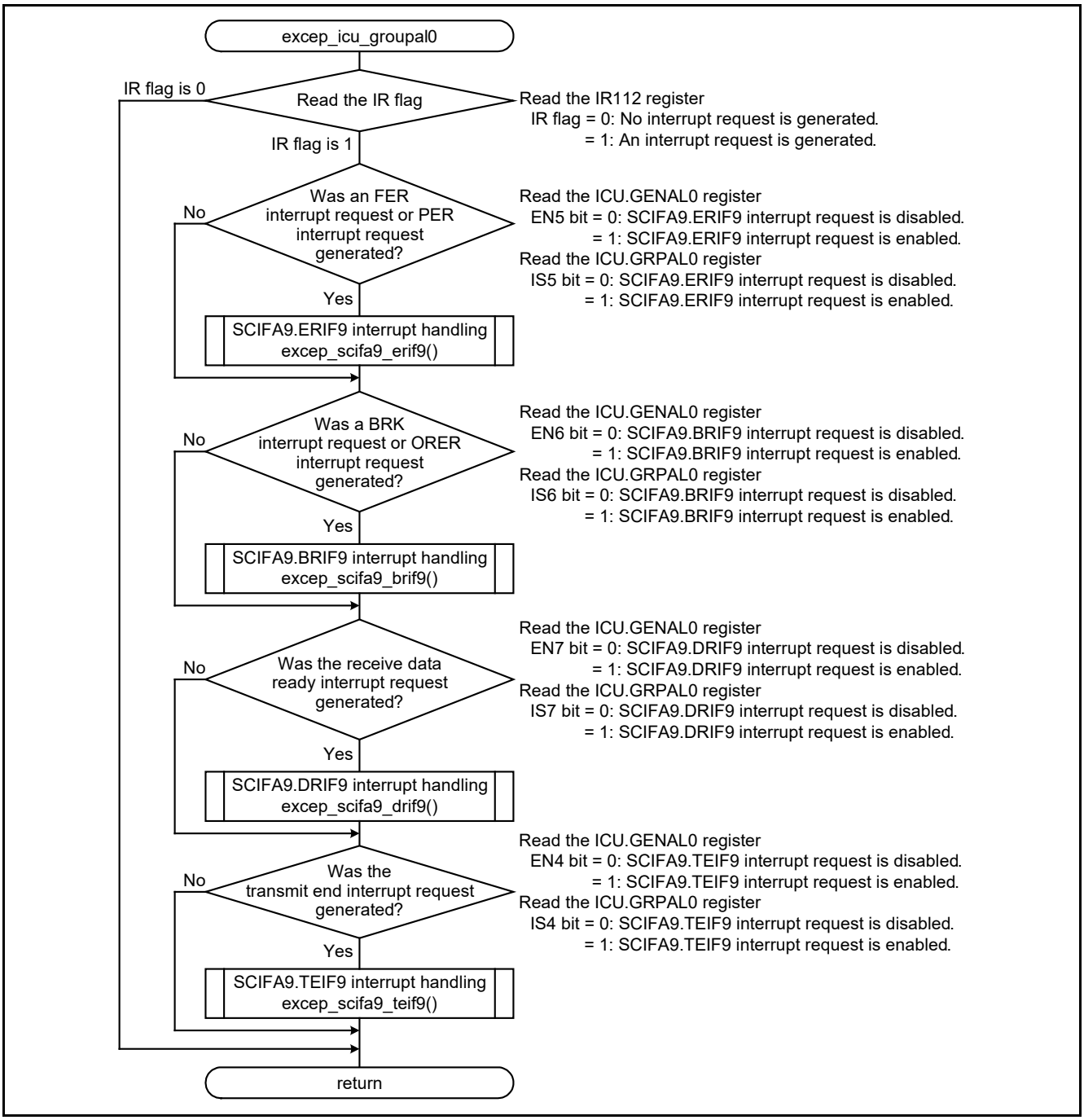


Figure 5.16 GROUPAL0 Interrupt Handling

5.10.11 SCIFA9 Transmit End Interrupt Handling

Figure 5.17 shows SCIFA9 Transmit End Interrupt Handling.

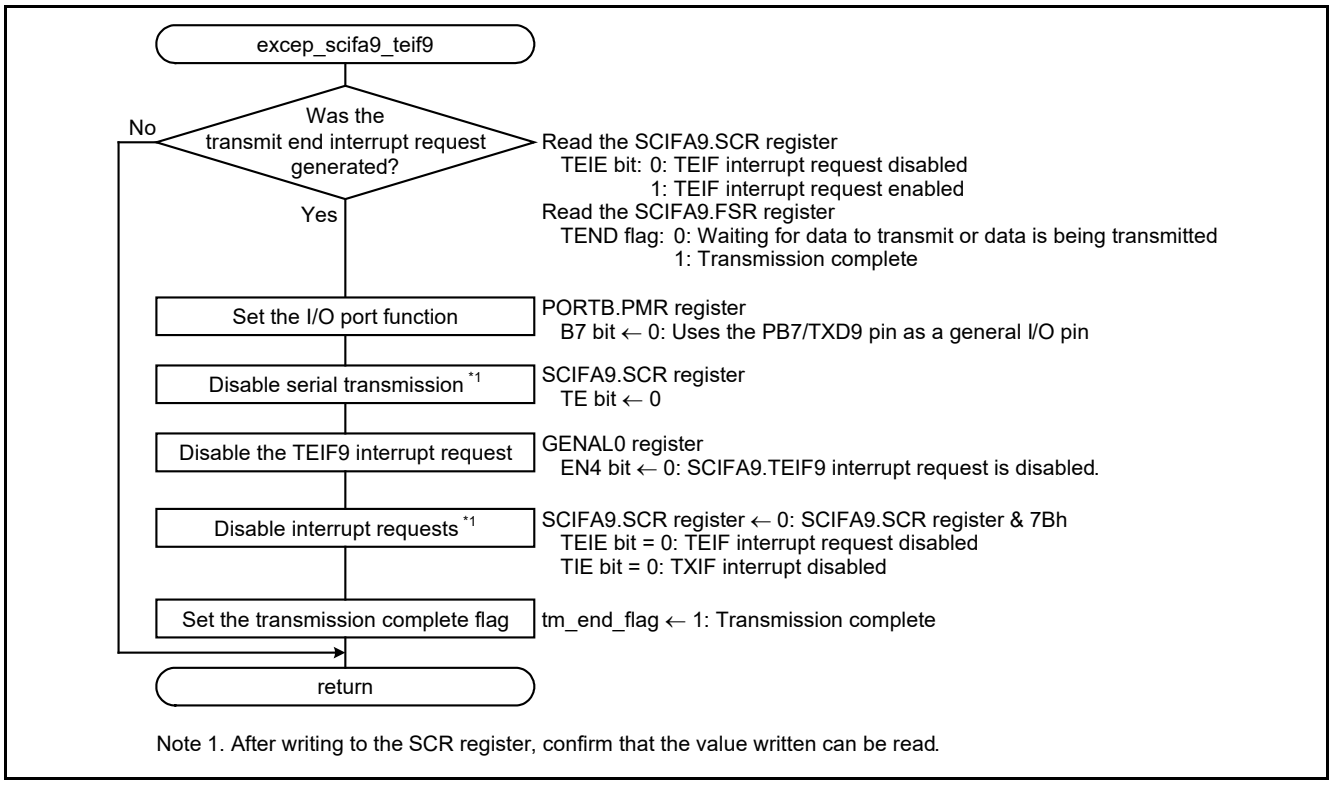


Figure 5.17 SCIFA9 Transmit End Interrupt Handling

5.10.12 SCIFA9 Receive Error Interrupt Handling

Figure 5.18 shows SCIFA9 Receive Error Interrupt Handling.

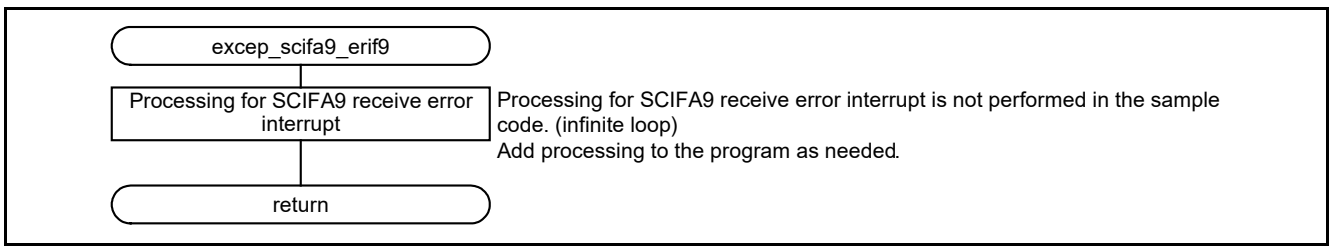


Figure 5.18 SCIFA9 Receive Error Interrupt Handling

5.10.13 SCIFA9 Break Detect and Overrun Error Interrupt Handling

Figure 5.19 shows SCIFA9 Break Detect and Overrun Error Interrupt Handling.

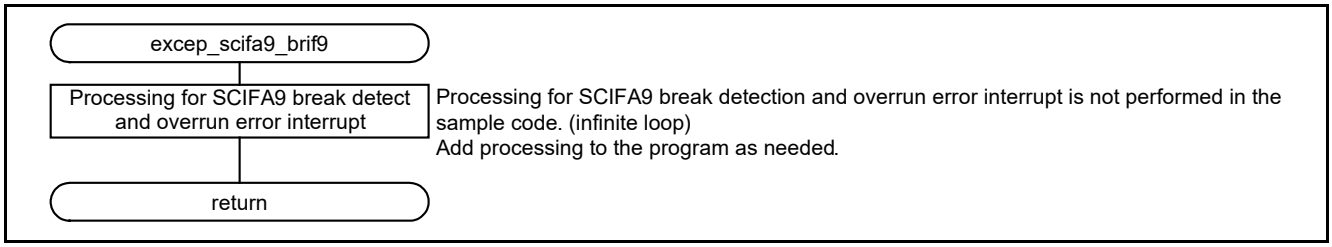


Figure 5.19 SCIFA9 Break Detect and Overrun Error Interrupt Handling

5.10.14 SCIFA9 Receive Data Ready Interrupt Handling

Figure 5.20 shows SCIFA9 Receive Data Ready Interrupt Handling.

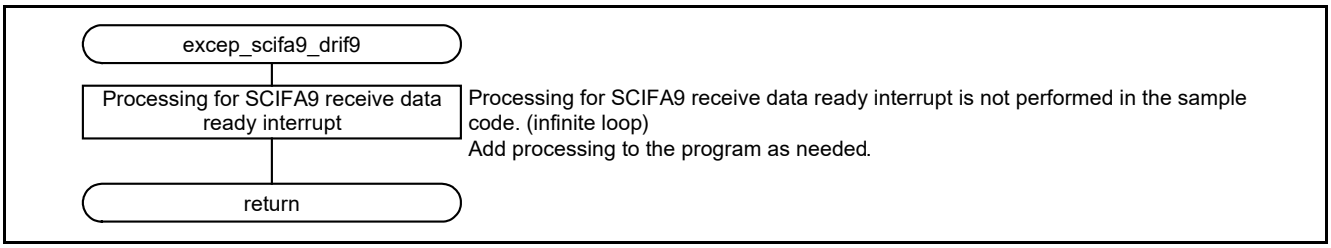


Figure 5.20 SCIFA9 Receive Data Ready Interrupt Handling

6. Note on the Amount of Data to be Transmitted/Received Data and the FIFO Threshold Value

In this application note, when changing the amount of data to be transmitted/received, set the amount of data so it is a whole-number multiple of the transmit/receive FIFO threshold value (in this application note, it is a multiple of 8).

7. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

8. Reference Documents

User's Manual: Hardware
RX64M Group User's Manual: Hardware (R01UH0377EJ)
RX71M Group User's Manual: Hardware (R01UH0493EJ)
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 Compiler CC-RX V.2.01.00 User's Manual: RX Coding Rev.1.00 (R20UT2748EJ)
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	RX64M, RX71M Group Application Note Using the DTC and SCIFA to Perform Asynchronous Serial Transmission and Reception

Rev.	Date	Description	
		Page	Summary
1.00	Oct. 1, 2014	—	First edition issued
1.01	Nov. 2, 2015	—	RX71M Group is added to the target device
		13	“MDE” in table 5.3 is corrected
1.10	Dec. 21, 2020	—	Update the toolchain version.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

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

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. 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 the 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.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. 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 shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating 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. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall 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 shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, 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.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.