

## RX62T Group

CRC Code Calculation Using DTC Block Transfer and Asynchronous  
Serial Communication with CRC Code Using DTC Data Transfer

R01AN0635EJ0100  
Rev.1.00  
Sep 27, 2011

### Introduction

This application note presents a sample program that performs serial data transfer to the CRC calculation unit (CRC) using the data transfer controller (DTC module) block transfer function and then uses the DTC module for a serial communications interface (SCI) data transfer to perform asynchronous serial communication including CRC code.

### Target Devices

RX62T Group

Other members of the RX Family that have the same I/O registers (peripheral unit control registers) as the RX62T Group products can also use the code from this application note. Note, however, that since certain aspects of the functions used may be changed in other devices due to function additions or other differences, the documentation for the device used must be checked carefully before using this code. When using this code in an end product or other application, its operation must be tested and evaluated thoroughly.

### Contents

1. Specifications .....	2
2. Operation Confirmation Environment.....	3
3. Functions Used .....	4
4. Operation.....	4
5. Software .....	10
6. Reference Documents.....	24

## 1. Specifications

The code presented in this application note performs transmission and reception of 130 bytes of data using the DTC module both for creation of the transmit data with CRC code and for CRC calculation for the receive data with CRC code, and also using DTC transfer for serial communication with CRC code.

- $X^{16} + X^{15} + X^2 + 1$  is used as the CRC generator polynomial to generate the CRC code for LSB first communication.
- SCI channel 0 is used with an 8-bit length, one stop bit, no parity bit communication format.
- The communication bit rate is 38,400 bps.
- The DTC module uses the ICU software interrupt (SWINT), the SCI0 transmit data empty interrupt (TXI0), and the SCI0 receive data full interrupt (RXI0).
- CRC calculation unit initialization data (H'82) is stored in internal RAM in advance.
- The CRC generator polynomial is set up, LSB first communication CRC code generation is specified, and the CRC data output register (CRCDOR) is cleared by transferring the CRC initialization data (H'82) to the CRC calculation unit.
- The transfer data (128 bytes) is stored in advance in the transfer data storage area in internal RAM.
- Consecutive data bytes with the values H'00, H'01, H'02, through H'7D, H'7E, and H'7F is used as the 128 bytes of transfer data.

### (1) Transmit Operation

1. Generate a SWINT interrupt with the CPU to start the DTC module.
2. Use a DTC transfer initialize the CRC calculation unit, generate the CRC code, and add the CRC code to the transmit data (for a total of 130 bytes).
3. After the DTC transfer initiated by the SWINT interrupt completes, enable the SCI0 transmit data empty interrupt (TXI0) with a CPU SWINT interrupt and start the DTC module again.
4. The transmit data with the added CRC code (130 bytes, total) is transferred with a DTC transfer.
5. After the transfer completes, reinitialize the DTC module for transmission.
6. Return to step 1. above to repeat the transmit operation.

### (2) Receive Operation

1. Enable the SCI0 receive data full interrupt (RXI0).
2. Start the DTC module with an RXI0 interrupt.
3. Transfer the receive data with the added CRC code (130 bytes, total) to internal RAM with a DTC transfer.
4. After transfer of 130 bytes of data completes, use a chain transfer to initialize the CRC calculation unit, calculate the CRC result, and save the result of the CRC calculation.
5. When the DTC transfer completes with the RXI0 interrupt, check for CRC errors on the CPU RXI0 interrupt.
6. If a CRC error is detected, stop the reception operation and report that an error occurred with the LEDs connected to the port 71. If no CRC error is detected, reinitialize the DTC module for reception.
7. Return to step 1. above to repeat the receive operation.

Figure 1 shows the specifications for the system implemented in this application note.

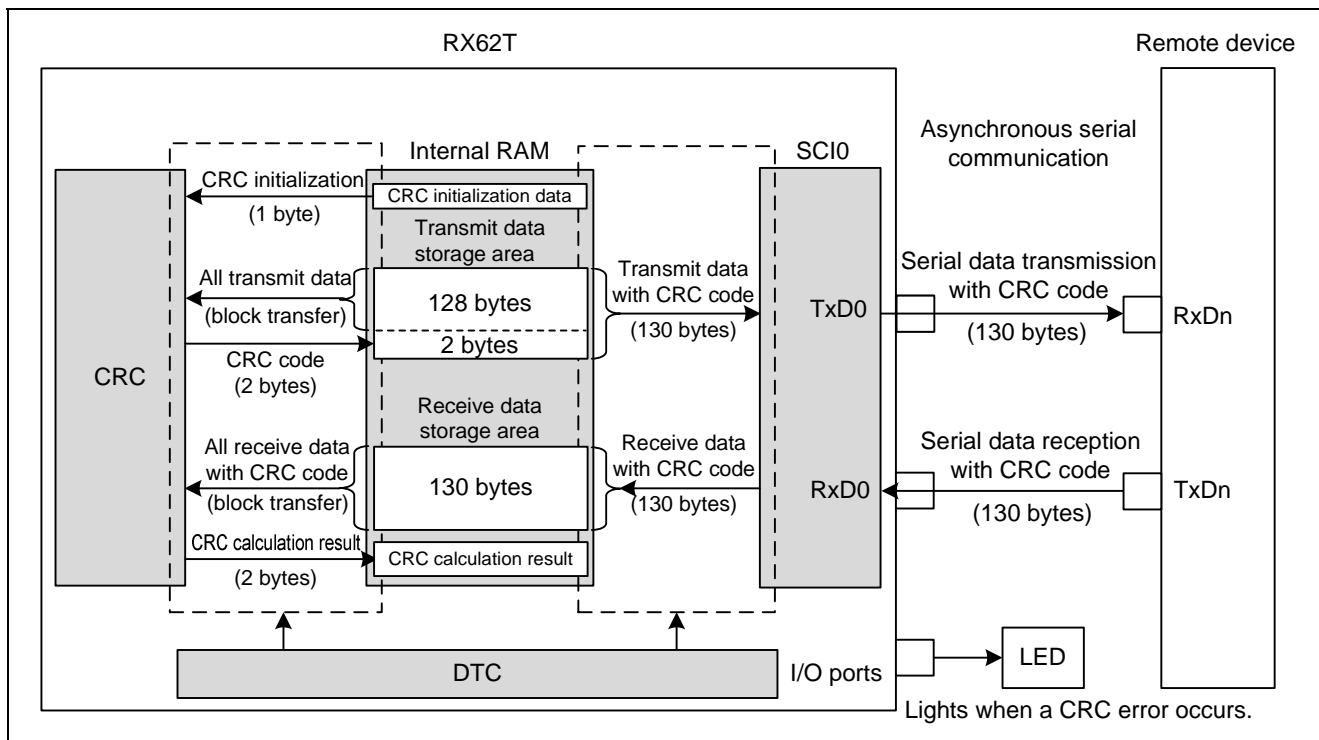


Figure 1 Specifications

## 2. Operation Confirmation Environment

Table 1 lists the environment required for confirming master operation.

Table 1 Operation Confirmation Environment

Item	Description
Device	RX62T (R5F562TAADFP)
Board	Renesas Starter Kit (R0K5562T0S000BE)
Power supply voltage	5.0 V
Input clock	12.5 MHz (ICLK = 100 MHz, PCLK = 50 MHz)
Operating temperature	Room temperature
HEW	Version 4.08.00.011
Toolchain	RX Standard Toolchain (V.1.0.1.0)
Debugger/emulator	E20 emulator
Debugger component	RX E1/E20 SYSTEM V.1.01.00

### 3. Functions Used

- Clock generation circuit
- Low power consumption functions
- I/O ports
- Interrupt control unit (ICU)
- Serial communication interface (SCI)
- CRC calculation unit (CRC)

Note: If other data is written to the CRCDIR register, either during a CRC calculation or during the period up to the point the CRC result has been saved (for example, if the CRC calculation unit is being used by two or more different bus masters), the result calculated by the CRC calculation unit may be affected. In the examples in this application note, if the CRC calculation unit is accessed from the DMAC during access to the CRC calculation unit by the DTC, the calculation result will be lost. Therefore, the CRC calculation unit should not be accessed by the DMAC. Note, however, that the CRC calculation unit may be accessed by multiple DTCs without problem.

- Data transfer controller (DTC)

See the RX62T Group User's Manual: Hardware for detailed information.

## 4. Operation

### 4.1 Operation Mode Settings

In the sample program, mode pins are set to MD1 = 1, MD0 = 1 to select single-chip mode as the operating mode and the ROME bit in system control register 0 (SYSCR0) is set to 1 to enable the on-chip ROM.

Table 2 lists the operating mode settings used in the sample program.

**Table 2 Operating Mode Settings**

Mode Pin		SYSCR0 Register		
MD1	MD0	ROME	Operating Mode	On-chip ROM
1	1	1	Single-chip mode	Enabled

Note: The initial setting of the ROME bit in the SYSCR0 register is SYSCR0.ROME = 1, so it is not necessary for the sample program to make settings to the SYSCR0 register.

### 4.2 Clock Settings

The evaluation board used for this application note includes a 12.5 MHz crystal oscillator.

Therefore this application note uses the following settings for the system clock (ICLK) and the peripheral module clock (PCLK): 8× (100 MHz) and 4× (50 MHz).

### 4.3 Endian Mode Setting

The sample program presented in this application note supports both big- and little-endian mode. Table 3 lists the hardware endian mode settings of the master device.

**Table 3 Endian Mode Settings (Hardware)**

MDE pin	Endian
0	Little endian
1	Big endian

Table 4 lists the endian settings used in the compiler options.

**Table 4 Endian Mode Settings (Compiler Options)**

MCU Option	Endian
endian = little	Little endian
endian = big	Big endian

Note: Set the MDE pin to match the endian mode selected as a compiler option.

#### 4.4 Bit Order Settings

The program in this application note supports both right and left as the bit order. Table 5 lists the bit order settings in the microcontroller option in the compiler options.

**Table 5 Bit Order Settings (Compiler Options)**

MCU Option	Bit Order
bit_order = right	Bit field members are allocated in order starting with the low-order bit. (Default)
bit_order = left	Bit field members are allocated in order starting with the high-order bit.

Notes: 1. In this application note, bit fields are used in the I/O register definitions file (iodefine.h). In the I/O register definitions file, "left" is specified with the #pragma bit\_order extension, and the bit field members are allocated in order starting with the high-order bit.

2. If both the bit\_order compiler option and the #pragma bit\_order extension are specified, the #pragma bit\_order extension specification takes precedence. Thus the bit fields defined in the I/O register definitions file will be allocated in order starting with the high-order bit, regardless of the compiler options bit\_order specification.

#### 4.5 SCI Settings

The program in this application note uses the SCI channel 0 for asynchronous serial data transmission and reception. Table 6 lists the SCI settings.

**Table 6 SCI Settings and Conditions**

Channel Used	SCI 0
Communication mode	Asynchronous serial communication mode
Interrupts	<ul style="list-style-type: none"> <li>• Receive error interrupt (ERI0)</li> <li>• Receive data full interrupt (RXI0)</li> <li>• Transmit data empty interrupt (TXI0)</li> <li>• Transmit complete interrupt (TEI0)</li> </ul>
Communication speed	38,400 bps (PCLK = 50 MHz)
Data length	8-bit data
Stop bits	1 stop bit
Parity	None

## 4.6 CRC Settings

The program in this application note generates a 2-byte CRC code using the CRC calculation unit. Table 7 lists the CRC settings

**Table 7 CRC Settings and Conditions**

Generating polynomial	$X^{16} + X^{15} + X^2 + 1$
CRC calculation	Generates a CRC code for LSB first communication

## 4.7 DTC Settings

The program in this application note uses the DTC module for the SCI data transfer, creation of the transmit data with CRC code, and the CRC calculation for the receive data with CRC code. Table 8 lists the DTC settings for reception and table 9 lists the DTC settings for transmission.

**Table 8 DTC Settings and Conditions (Reception)**

Condition	SCI Transfer of Receive Data with CRC Code	CRC Initialization	CRC Transfer of Receive Data with CRC Code	Save of the Calculation Result
Transfer mode	Normal transfer mode	Normal transfer mode	Block transfer mode	Normal transfer mode
Transfer operations	130 transfer	One transfer	One transfer	One transfer
Block size	—	—	130 bytes	—
Transfer data size	Byte	Byte	Byte	Word
Transferred data	Receive data with CRC code (130 bytes, total)	CRC initialization data (1 byte)	Receive data with CRC code (130 bytes, total)	CRC calculation result (2 bytes)
Transfer source	SCI0 receive data register (SCI0.RDR)	Internal RAM	Internal RAM	CRC data output register (CRCDOR)
Transfer destination	Internal RAM	CRC control register (CRCCR)	CRC data input register (CRCDIR)	Internal RAM
Transfer source address	The transfer source is a fixed location.	The transfer source is a fixed location.	The transfer source address is incremented after the transfer.	The transfer source is a fixed location.
Transfer destination address	After the transfer, increment the transfer destination address.	The transfer destination is a fixed location.	The transfer destination is a fixed location.	The transfer destination is a fixed location.
Start event	Activated by a receive data full interrupt (RXI0)	Executed after the receive data SCI transfer	Executed on the completion of the CRC initialization	Executed after the receive data CRC transfer
Chaining	Enabled	Enabled	Enabled	Disabled
Interrupt	None	None	None	CPU interrupts are enabled after completion of the specified data transfer.

Table 9 DTC Settings and Conditions (Transmission)

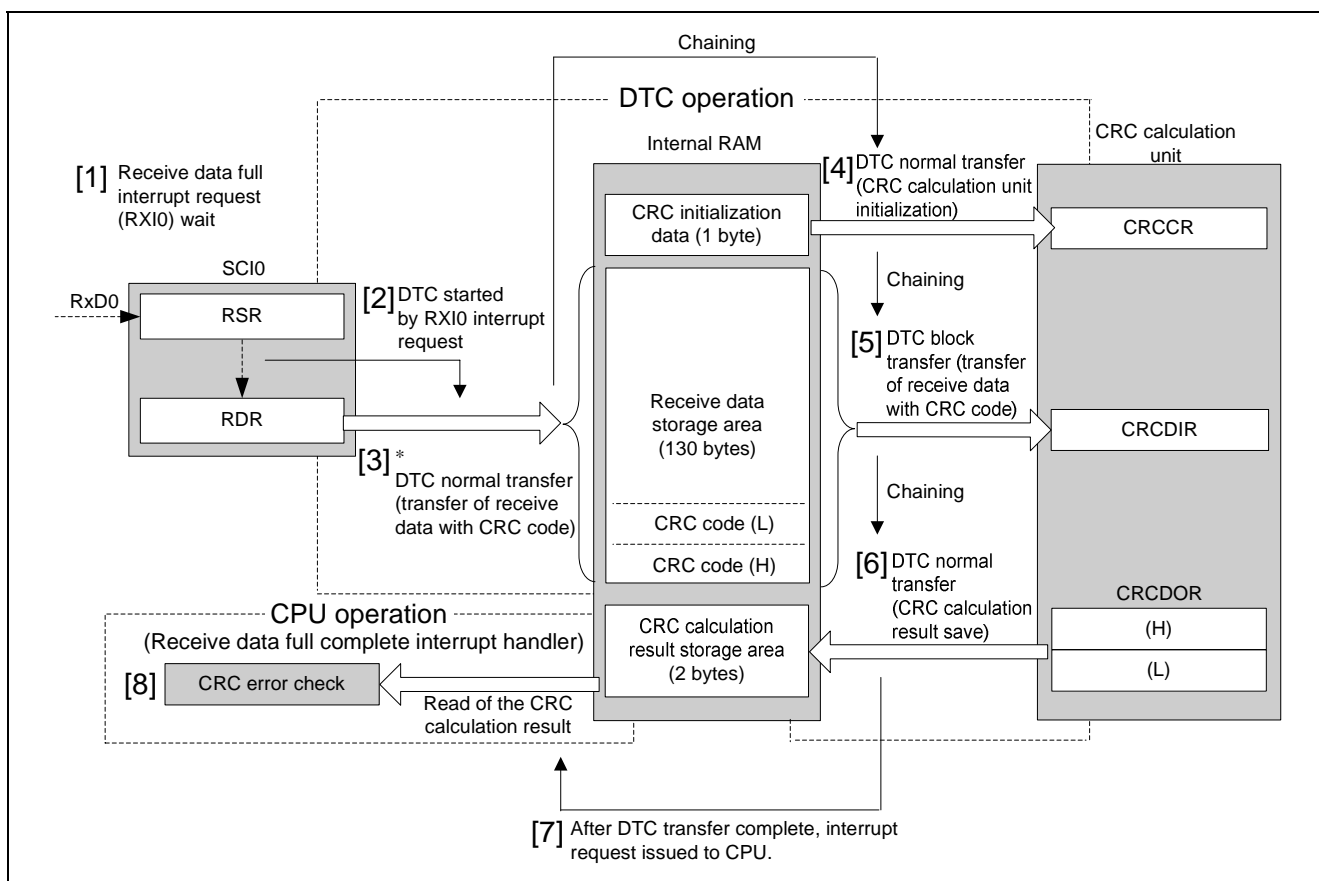
Condition	CRC Initialization	CRC Transfer of the Transmit Data	Adding the CRC Calculation Result	SCI Transfer of Transmit Data with CRC Code
Transfer mode	Normal transfer mode	Block transfer mode	Normal transfer mode	Normal transfer mode
Transfer operations	One transfer	One transfer	One transfer	130 transfer
Block size	—	128 bytes	—	—
Transfer data size	Byte	Byte	Word	Byte
Transferred data	CRC initialization data (1 byte)	Transmit data (128 bytes)	CRC calculation result (2 bytes)	Transmit data with CRC code (130 bytes)
Transfer source	Internal RAM	Internal RAM	CRC data output register (CRCDOR)	Internal RAM
Transfer destination	CRC control register (CRCCR)	CRC data input register (CRCDIR)	Internal RAM	SCI0 transmit data register (SCI0.TDR)
Transfer source address	The transfer source is a fixed location.	After the transfer, increment the transfer source address.	The transfer source is a fixed location.	After the transfer, increment the transfer destination address.
Transfer destination address	The transfer destination is a fixed location.	The transfer destination is a fixed location.	The transfer destination is a fixed location.	The transfer destination is a fixed location.
Start event	Activated by the ICU software interrupt (SWINT)	Executed on the completion of the CRC initialization	Executed after the CRC transfer of the transmit data	Activated by the transmit data empty interrupt (TXI0)
Chaining	Enabled	Enabled	Disabled	Disabled
Interrupt	None	None	CPU interrupts are enabled after completion of the specified data transfer.	CPU interrupts are enabled after completion of the specified data transfer.

## 4.8 Operation in Detail

### 4.8.1 Reception Operation

- [1] Wait for an SCI0 receive data full interrupt request (RXI0)
- [2] Start the DTC module on the RXI0 interrupt request.
- [3] Use DTC to transfer the receive data with CRC code (130 bytes, total) from the SCI0 receive data register (RDR) to internal RAM.
- [4] After the 130-byte data transfer has completed, use a chain transfer to transfer the CRC initialization data (H'82) in internal RAM to the CRC control register (CRCCR) to initialize the CRC calculation unit.
- [5] After initializing the CRC calculation unit, perform a block transfer of the 130 bytes of receive data with CRC code in internal RAM to the CRC data input register (CRCDIR) to calculate the CRC result.
- [6] Save the CRC calculation result in the CRC data output register (CRCDOR) in internal RAM using a chain transfer operation.
- [7] After the DTC transfer completes, generate a CPU interrupt request (the RXI0 interrupt request).
- [8] In the RXI0 interrupt handler, read the CRC calculation result in internal RAM and perform a CRC error check. If a CRC error is detected, stop the reception operation and report that an error occurred with the LEDs connected to port 71. If no CRC error is detected, reinitialize the DTC module for reception and return to step 1.

Figure 2 shows the block diagram of the reception operation.



**Figure 2 Receive Operation Block Diagram**

Note: Note that when the DTC module is used in combination with communication functions in the RX62T Group microcontroller, if the next transfer request occurs before the IR flag is cleared automatically, the transfer request will be lost.

See section 11.7, Usage Notes, in the RX62T Group User's Manual - Hardware for details.

4.8.2 Transmission Operation

- [1] Issue a software interrupt (SWINT) by writing 1 to the SWINT bit in the ICU software interrupt request register (SWINTR).
- [2] The DTC module is started by the SWINT interrupt request.
- [3] The DTC module transfers the CRC initialization data (H'82) in internal RAM to the CRCCR to initialize the CRC calculation unit.
- [4] After CRC calculation unit initialization, a chain transfer is used to perform a block transfer of the transfer data (128 bytes) in internal RAM to the CRCDIR. This generates the 2-byte CRC code.
- [5] The CRC code generated in the CRCDOR register is added to the transmit data with a chain transfer thus creating the 130 bytes of transmit data with CRC code.
- [6] After the DTC transfer completes, an interrupt request will be issued to the CPU (an ICU SWINT interrupt request).
- [7] In the ICU SWINT interrupt handler, read the MDE bit in the mode monitor register (MDMONR) and if the MDE bit is 1 (big endian), swap the CRC code (L) and CRC code (H) bytes so that the CRC code (H) byte is the last byte in the transmit data storage area.
- [8] Enable the SCI0 transmit data empty interrupt (TXIO) and start DTC.
- [9] DTC (transmit) transfers the transmit data with CRC code (130 bytes, total) to the SCI0 transmit data register (TDR). When the transfer of the 130 bytes of data completes, reinitialize the DTC module and return to step [1].

Figure 3 shows the block diagram of the transmission operation.

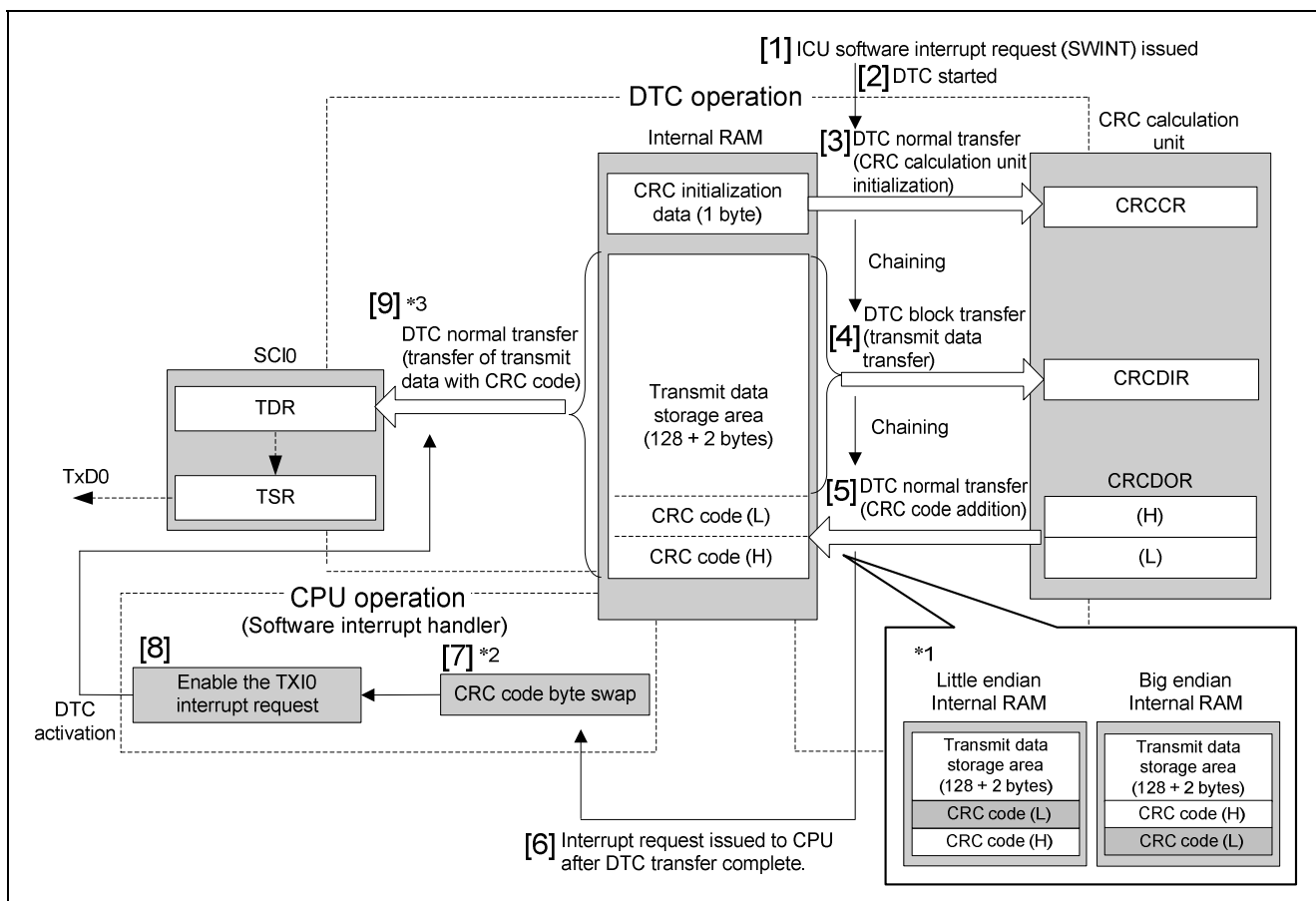


Figure 3 Transmit Operation Block Diagram

- Notes:
1. Since CRCDOR is a 16-bit register, it is affected by the endian setting when transferring data to internal RAM with the DTC module. For the little endian byte order, the CRC code (H) byte will be located in the last byte of the transmit data storage area, and for the big endian byte order, the CRC code (L) byte will be located in the last byte of the storage area.
  2. When the endian setting is set to big endian, the CRC code will be stored in reversed order. Thus it is necessary to swap the bytes in the CRC code. The program in this application note swaps the CRC code (L)

and CRC code (H) bytes in internal RAM so that the CRC code (H) byte will be the last byte in the transmit data storage area. Note that if the program in this application note is used with the little endian byte order, there will be no need for this processing.

- Note that when the DTC module is used in combination with communication functions in the RX62T Group microcontroller, if the next transfer request occurs before the IR flag is cleared automatically, the transfer request will be lost. See section 11.7, Usage Notes, in the RX62T Group User's Manual - Hardware for details.

## 5. Software

### 5.1 Constants

Table 10 lists the constants used in the sample code.

**Table 10 Constants**

Constant Name	Set Value	Usage
DTC_CNT	130	DTC transfer count
BLOCK_SIZE_TX	128	DTC transfer (transmit) block size
BLOCK_SIZE_RX	130	DTC transfer (receive) block size
BIG_ENDIAN	1	Big endian
NON_CRC_ERR	0000h	No CRC error occurred
SET	1	Value used to set flags
CLEAR	0	Value used to clear flags

### 5.2 Structures and Unions

Figures 4 and 5 show the structures and unions used in the sample program.

```

struct st_ram_data{
    union {
        unsigned char ALL[130];          /* Serial data with CRC code */
        struct {
            unsigned char DATA[128];   /* Serial data */
            unsigned short CRC_CODE;    /* CRC code */
        }MEMBER;
    }BUFF;
};

```

**Figure 4 Structures and Unions Used in the Sample Code (Serial Data with CRC Code)**

```

#pragma bit_order left
#pragma unpack
struct st_dtc_full{
  union{
    unsigned long LONG;
    struct{
      unsigned long MRA_MD      :2; /* MRA.MD bits */
      unsigned long MRA_SZ      :2; /* MRA.SZ bits */
      unsigned long MRA_SM      :2; /* MRA.SM bits */
      unsigned long :2;
      unsigned long MRB_CHNE    :1; /* MRB.CHNE bit */
      unsigned long MRB_CHNS    :1; /* MRB.CHNS bit */
      unsigned long MRB_DISEL   :1; /* MRB.DISEL bit */
      unsigned long MRB_DTS     :1; /* MRB.DTS bit */
      unsigned long MRB_DM      :2; /* MRB.DM bits */
      unsigned long :2;
      unsigned long :16;
    }BIT;
  }MR;
  void * SAR; /* SAR register */
  void * DAR; /* DAR register */
  struct{
    unsigned long CRA:16; /* CRA register */
    unsigned long CRB:16; /* CRB register */
  }CR;
};
#pragma bit_order
#pragma packoption

```

**Figure 5 Structures and Unions Used in the Sample Code (DTC Transfer Information)**

Note: The DTC transfer information alignment count is stipulated to be 4 by the "#pragma unpack" statement.

### 5.3 Variables

Table 11 lists the variables used in the sample program.

**Table 11 Variables**

Type	Variable	Description	Functions
unsigned char	send_end_flag	Transmit complete flag 0: Transmit in progress 1: Transmit complete	main, int_sci_tei0
unsigned char	receive_end_flag	Receive complete flag 0: Receive in progress or receive wait 1: Receive complete	main, int_sci_rxi0
unsigned char	crc_error_flag	CRC error flag 0: No error 1: Error occurred	main, int_sci_rxi0
unsigned char	CRC_RESET_DATA	CRC initialization data	main, dtc_send, dtc_receive
st_dtc_full	dtc_tx_crc[3]	DTC transfer information (transmit: CRC transfer)	dtc_send
st_dtc_full	dtc_tx_sci	DTC transfer information (transmit: SCI transfer)	dtc_send
st_dtc_full	dtc_rx[4]	DTC transfer information (receive)	dtc_receive
st_ram_data	ram_tx	Transmit data with CRC code	main, dtc_send
st_ram_data	ram_rx	Receive data with CRC code	main, dtc_receive
unsigned short	CRC_RESULT	CRC calculation unit	dtc_receive
void*	dtc_table[256]	DTC vector table*	dtc_init

Note: Allocated at address 0x00000000.

### 5.4 Functions

Table 12 lists the functions used in this application note's sample program.

**Table 12 Functions**

Function Name	Operation
HardwareSetup	Initialization, clock settings, and clearing the module stop state
main	Main processing ICU initialization and setting the interrupt levels
sci0_init	SCI initialization and setting the transfer clock
dtc_init	DTC initialization
dtc_receive	DTC transfer information placement (receive)
dtc_send	DTC transfer information placement (transmit)
crc_error	CRC error display
byte_reverse_16	Swaps the high and low order bytes in word data
int_icu_sw	ICU software interrupt
int_sci_txi0	Transmit data empty interrupt
int_sci_tei0	Transmit complete interrupt
int_sci_rxi0	Receive data full interrupt
int_sci_eri0	Receive error interrupt

## 5.5 Function Specifications

This section presents the specifications of the individual functions used in the sample code.

sci0_init	
Overview	Performs SCI0 initialization.
Header	iodefine.h
Declaration	void sci0_init(void)
Description	See table 6 in section 4.5 for details on the content set.
Arguments	None
Return value	None
Notes	

dtc_init	
Overview	Performs DTC initialization.
Header	iodefine.h
Declaration	void dtc_init(void)
Description	See tables 8 and 9 in section 4.7 for details on the content set.
Arguments	None
Return value	None
Notes	

dtc_receive	
Overview	Sets the DTC transfer information (receive).
Header	iodefine.h, dtc_def.h
Declaration	void dtc_receive(void)
Description	See table 8 in section 4.7 for details on the content set.
Arguments	None
Return value	None
Notes	This is used both for DTC initialization and reinitialization (receive).

dtc_send	
Overview	Sets the DTC transfer information (transmit).
Header	iodefine.h, dtc_def.h
Declaration	void dtc_send(void)
Description	See table 9 in section 4.7 for details on the content set.
Arguments	None
Return value	None
Notes	This is used both for DTC initialization and reinitialization (transmit).

crc_error	
Overview	Performs CRC error handling.
Header	iodefine.h
Declaration	void crc_error(void)
Description	Turns on the LED attached to port 71.
Arguments	None
Return value	None
Notes	

byte_reverse_16	
Overview	Swaps the upper and lower bytes in a word datum.
Header	iodefine.h
Declaration	unsigned short byte_reverse_16(unsigned short rev_data)
Description	Swaps the upper and lower bytes in the argument rev_data (type: unsigned short) and returns the result.
Arguments	rev_data (type: unsigned short)
Return value	(rev_data << 8)   (rev_data >> 8)
Notes	Used to swap the CRC code byte order.

int_icu_sw	
Overview	SWINT Interrupt function
Header	iodefine.h, vect.h
Declaration	void int_icu_sw(void)
Description	<ul style="list-style-type: none"> <li>• Reads the MDMONR MDE bit and if it is one (big endian) calls the byte_reverse_16() function.</li> <li>• Enables the TXI0 interrupt (IEN = 1).</li> </ul>
Arguments	None
Return value	None
Notes	

int_sci_txi0	
Overview	TXI0 Interrupt function
Header	iodefine.h, vect.h
Declaration	void int_sci_txi0(void)
Description	<ul style="list-style-type: none"> <li>• Disables the TXI0 interrupt (IEN = 0).</li> <li>• Enables the TEI0 interrupt (IEN = 1).</li> </ul>
Arguments	None
Return value	None
Notes	

int_sci_tei0	
Overview	TEI0 Interrupt function
Header	iodefine.h, vect.h
Declaration	void int_sci_tei0(void)
Description	<ul style="list-style-type: none"> <li>• Disables the TEI0 interrupt (IEN = 0).</li> <li>• Sets the transmit complete flag (send_end_flag).</li> </ul>
Arguments	None
Return value	None
Notes	

int_sci_rxi0	
Overview	RXI0 Interrupt function
Header	iodefine.h, vect.h
Declaration	void int_sci_rxi0(void)
Description	<ul style="list-style-type: none"> <li>Disables the RXI0 interrupt (IEN = 0).</li> <li>Checks for CRC errors and sets the CRC error flag (crc_error_flag) if a CRC error is detected.</li> <li>Sets the receive complete flag (receive_end_flag).</li> </ul>
Arguments	None
Return value	None
Notes	

int_sci_eri0	
Overview	ERI0 Interrupt function
Header	iodefine.h, vect.h
Declaration	void int_sci_eri0(void)
Description	Performs reception error handling (omitted in this sample code).
Arguments	None
Return value	None
Notes	

### 5.6 Processing Flow

Figures 6 to 19 show the processing flow of the sample program.

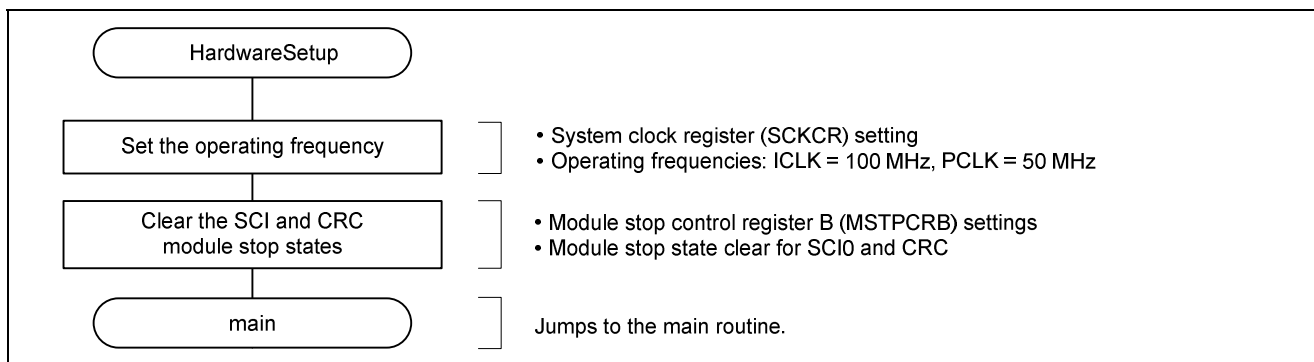


Figure 6 Initialization Processing

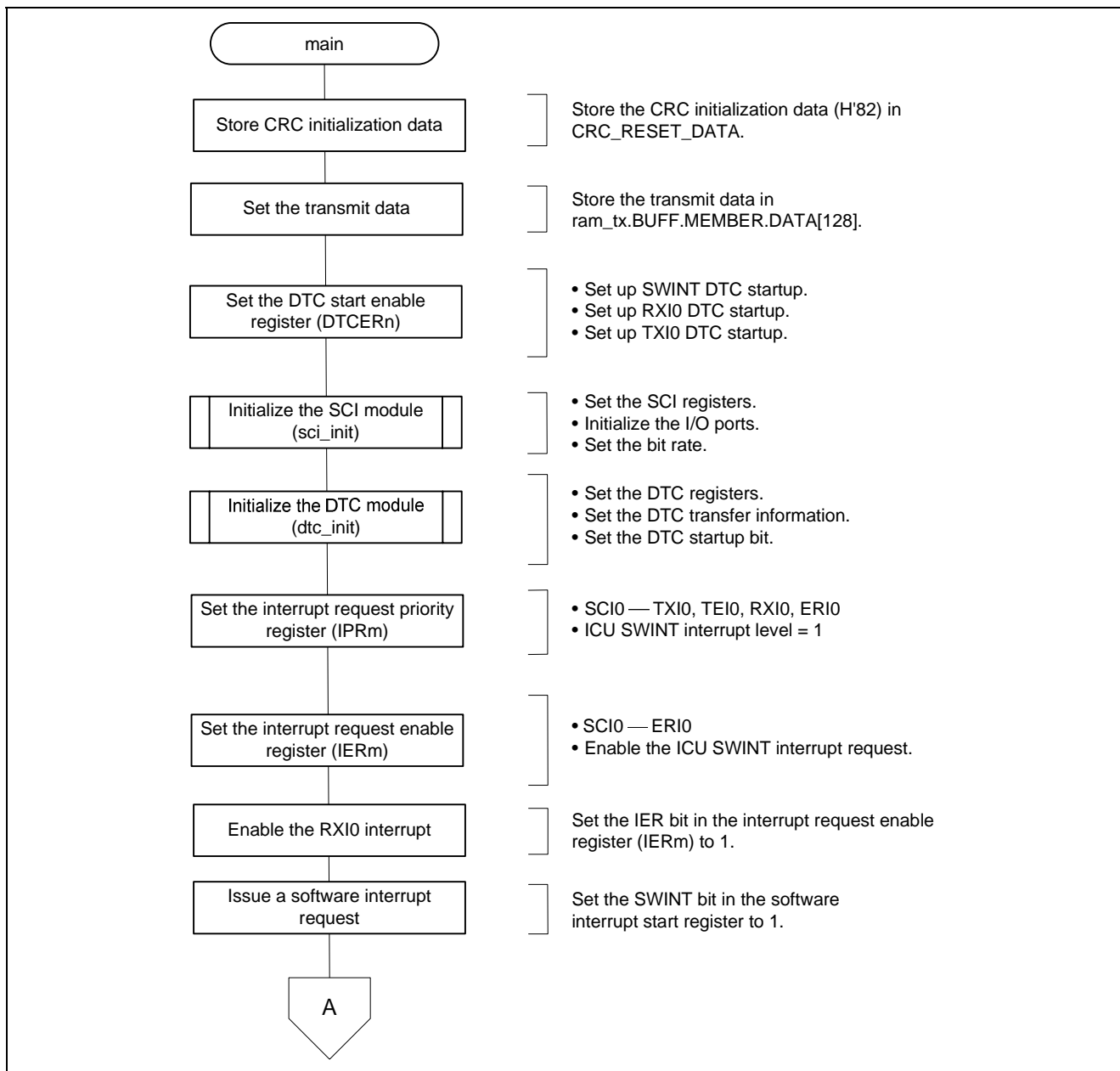


Figure 7 Main Processing (1)

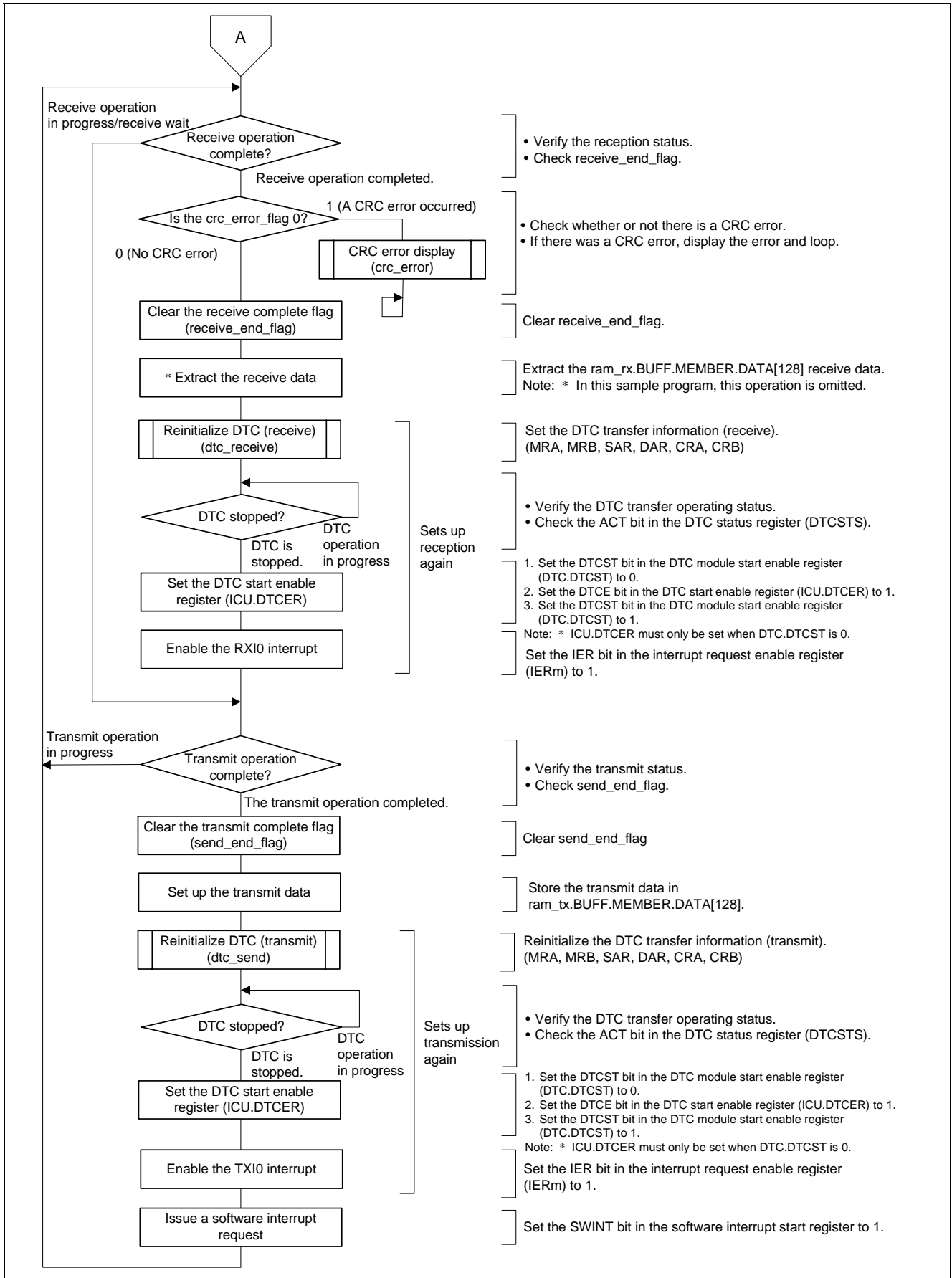


Figure 8 Main Processing (2)

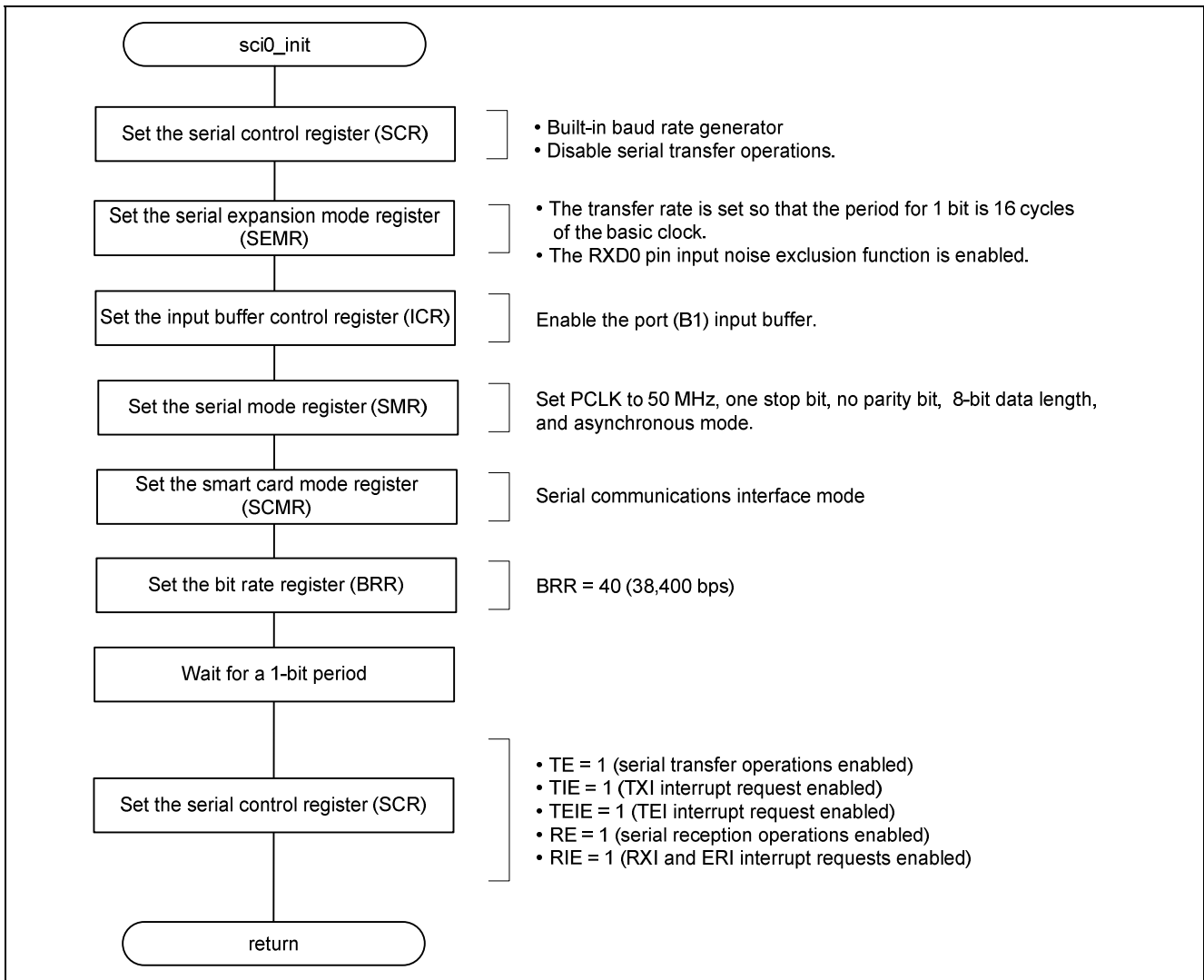


Figure 9 SCI Initialization

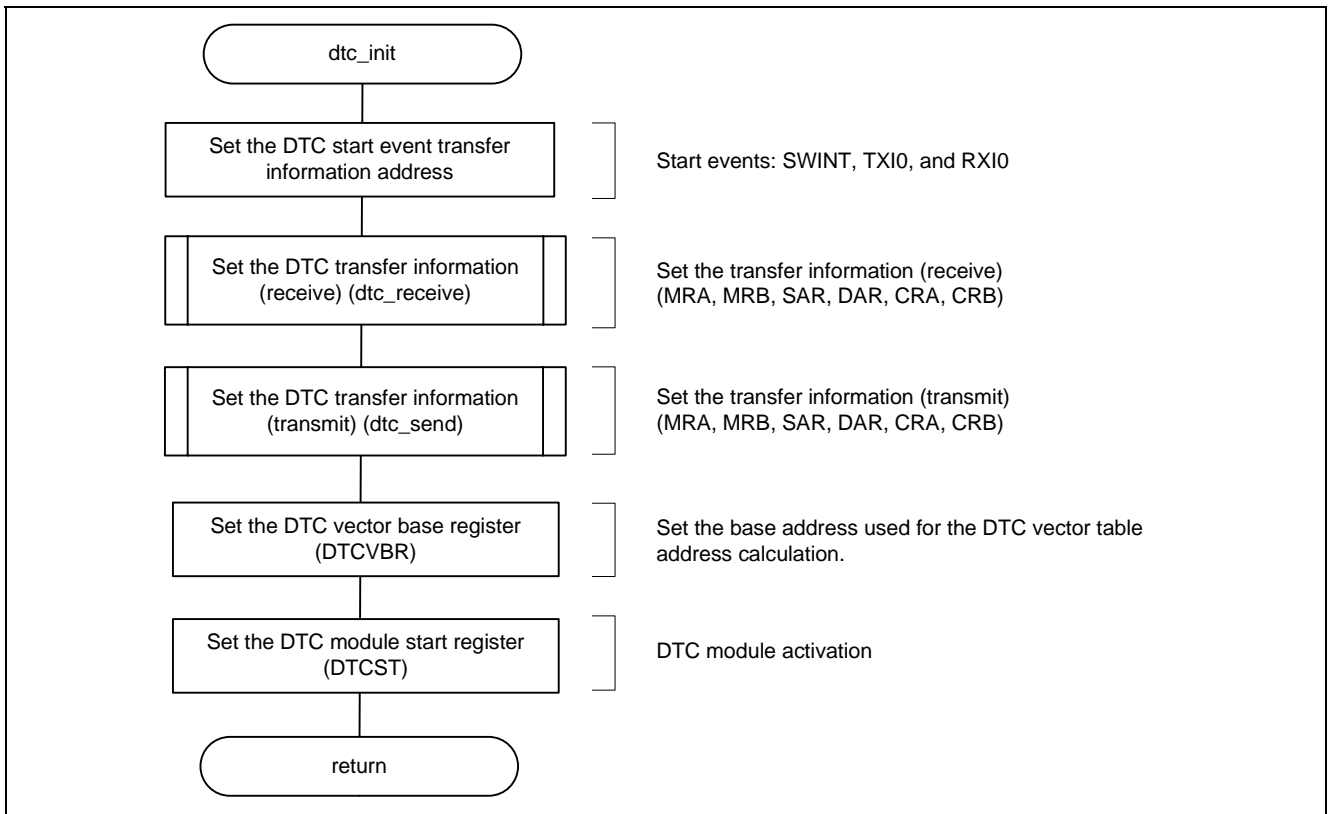


Figure 10 DTC Initialization

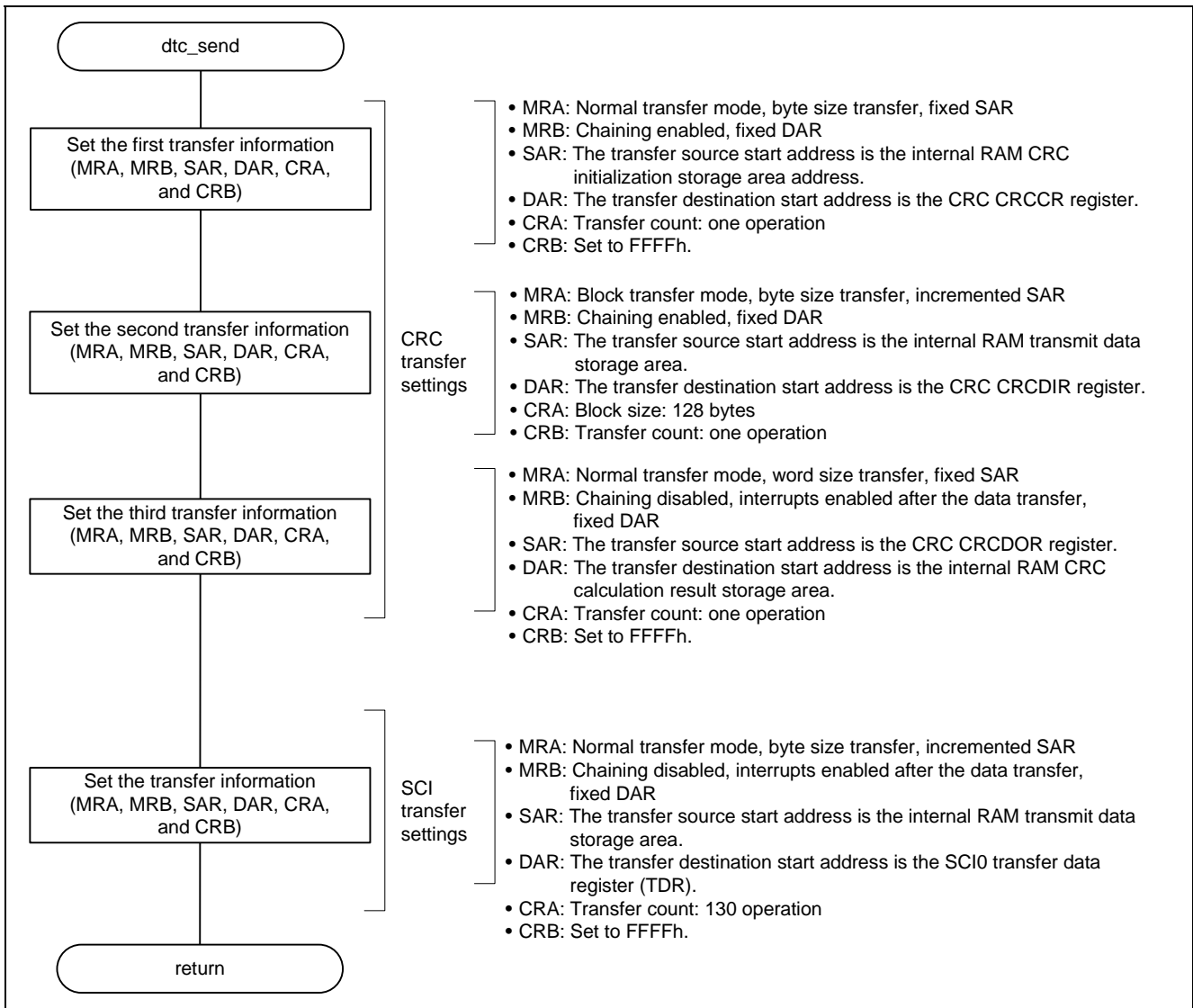


Figure 11 Transmit Transfer Information Allocation Processing

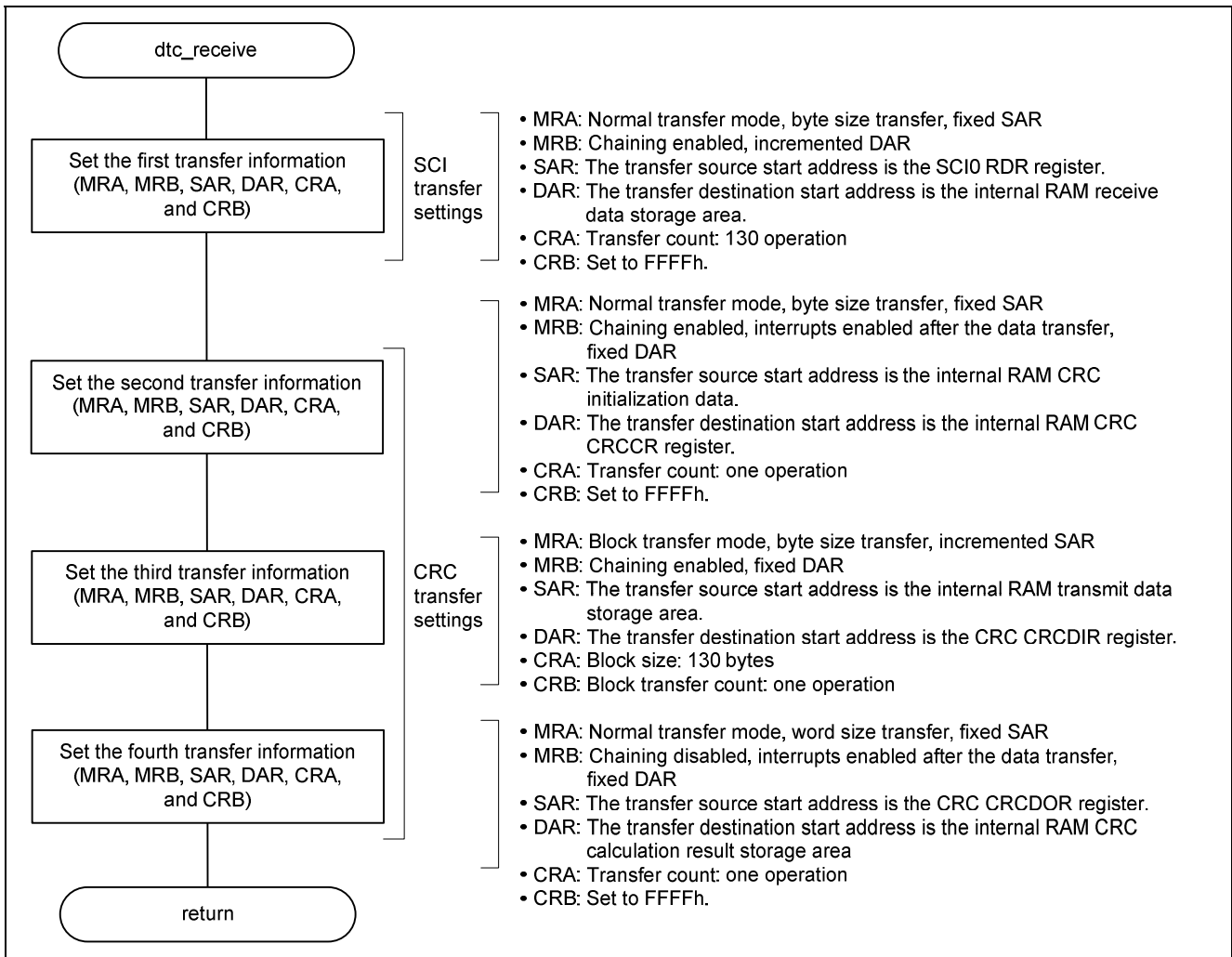


Figure 12 Receive Transfer Information Allocation Processing

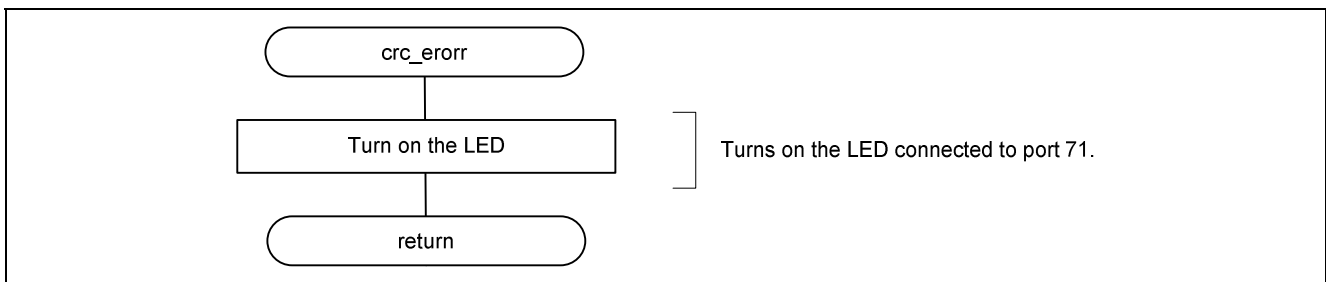


Figure 13 CRC Error Display

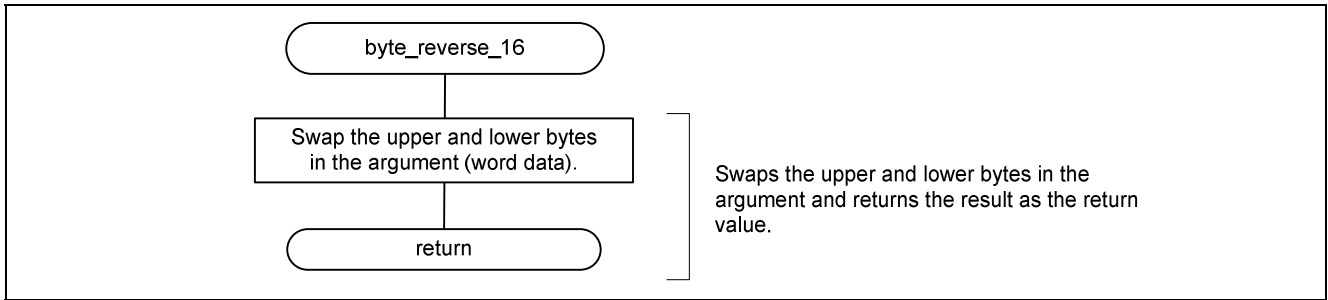


Figure 14 Word Data Byte Reordering

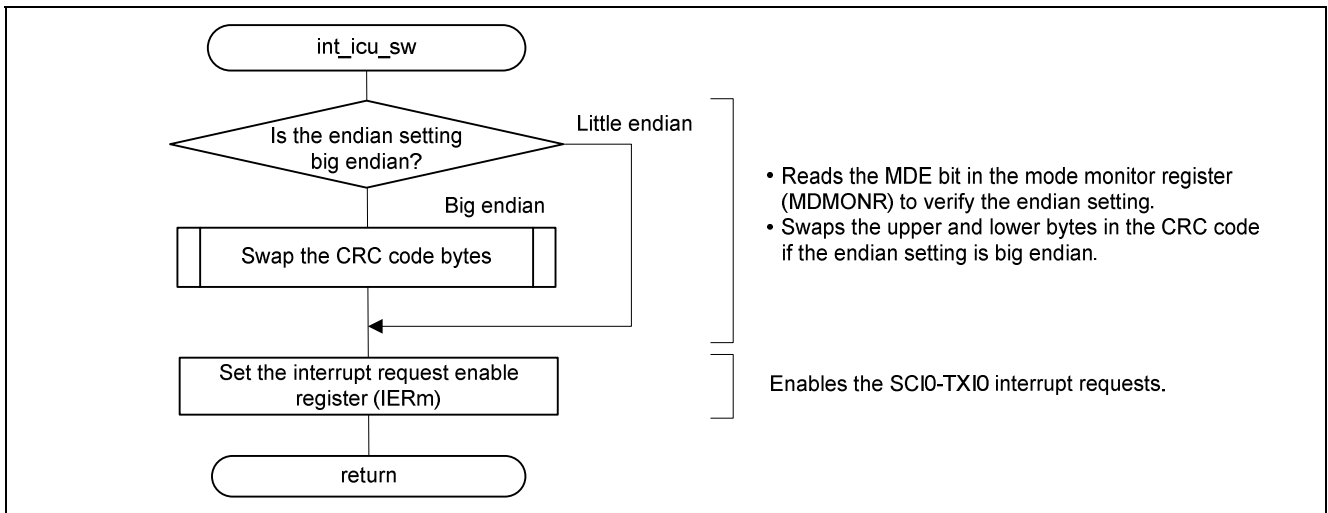


Figure 15 Software Interrupt

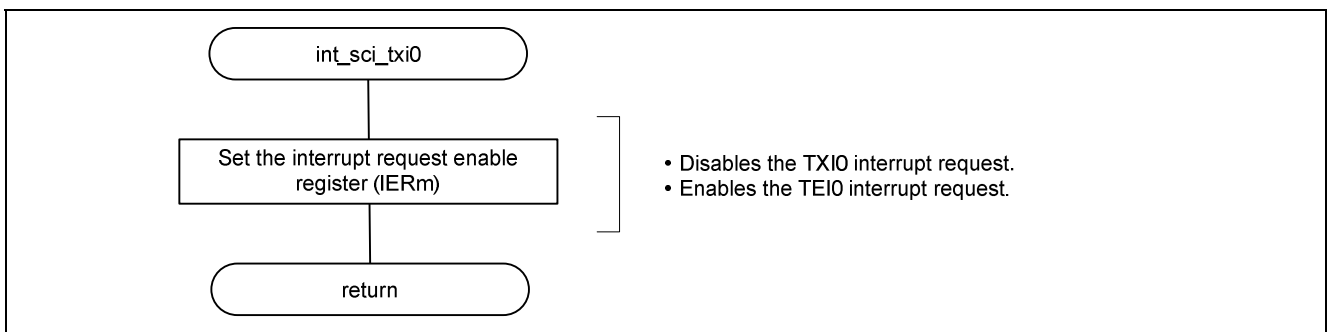


Figure 16 Transmit Data Empty Interrupt

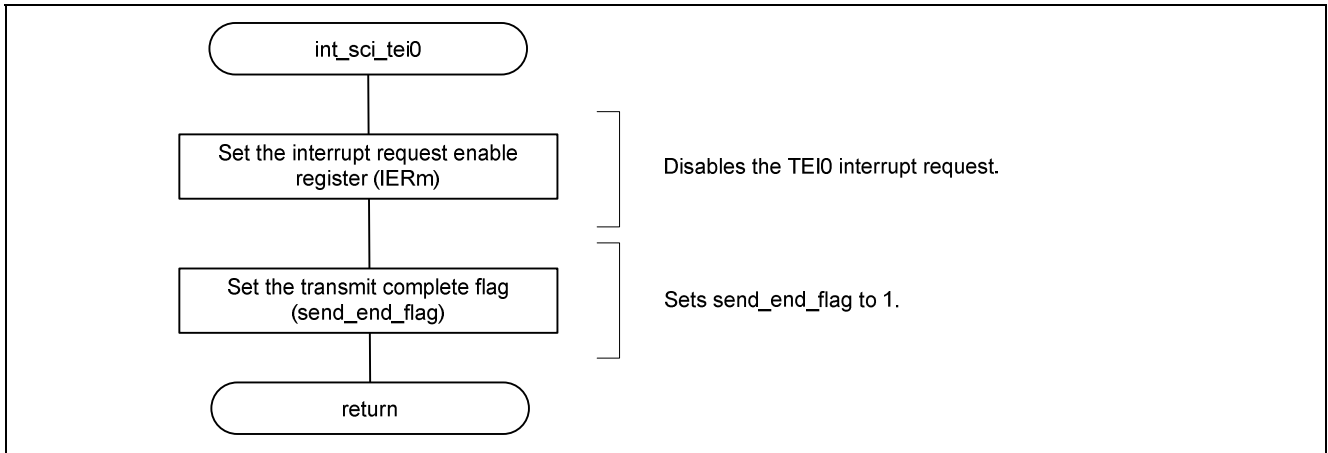


Figure 17 Transmit Complete Interrupt

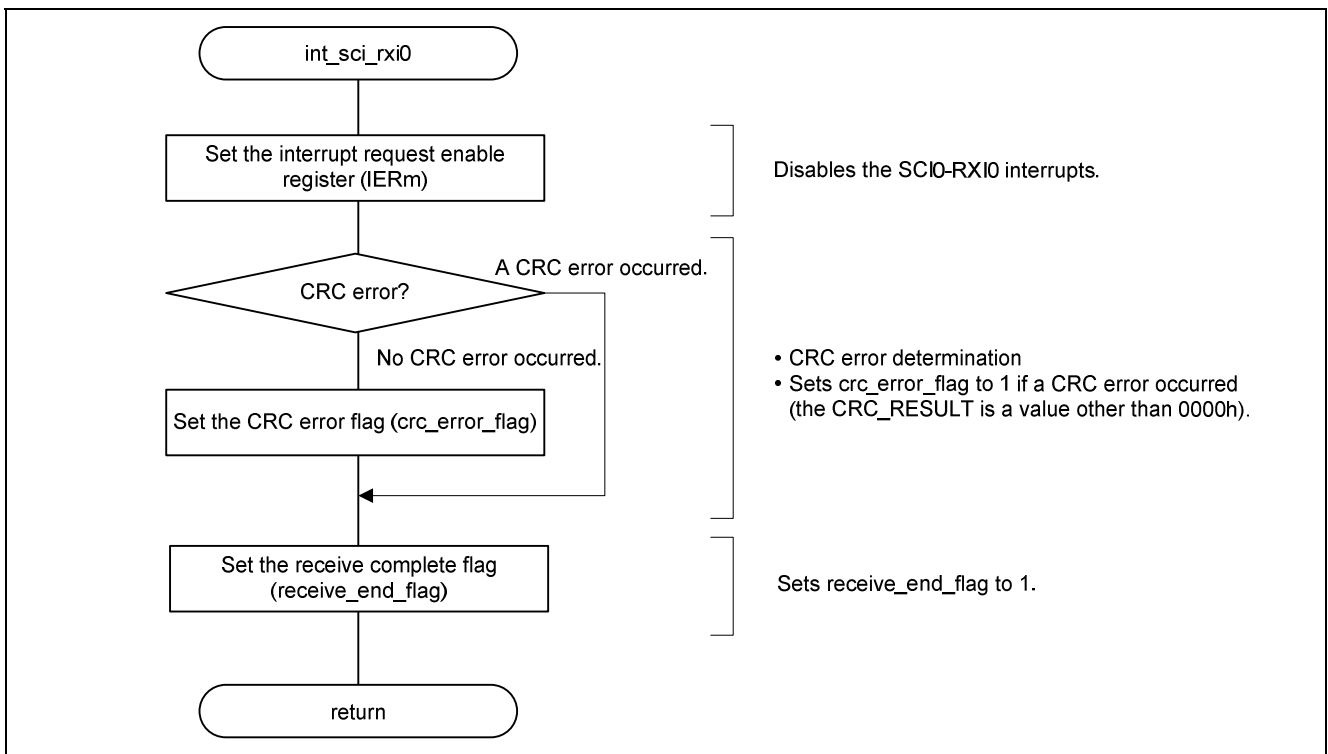


Figure 18 Receive Data Empty Interrupt

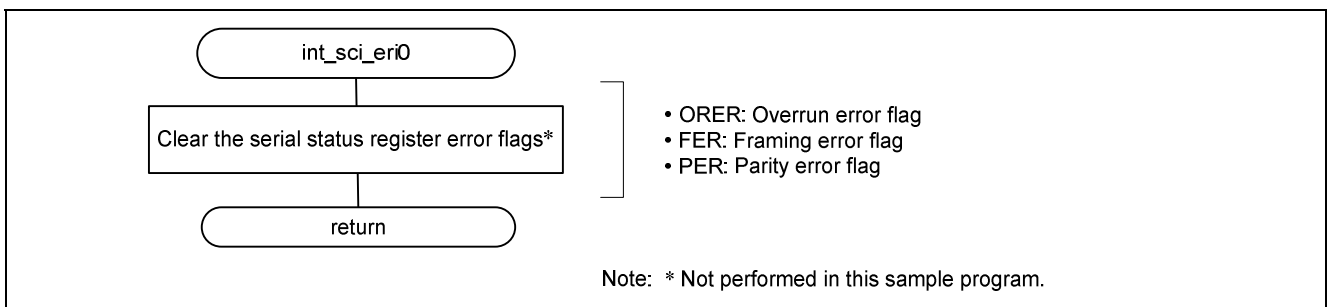


Figure 19 Receive Error Interrupt

## 6. Reference Documents

- Hardware Manual  
RX62T Group User's Manual: Hardware Rev.1.10  
(The latest version can be downloaded from the Renesas Electronics Web site.)
- Software Manual  
RX Family User's Manual: Software Rev.1.00  
(The latest version can be downloaded from the Renesas Electronics Web site.)
- Development Environment Manual  
RX Family C/C++ Compiler Package User's Manual Rev.1.01  
(The latest version can be downloaded from the Renesas Electronics Web site.)
- Technical Updates  
(The latest information can be downloaded from the Renesas Electronics Web site.)

## **Website and Support**

Renesas Electronics Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/inquiry>

All trademarks and registered trademarks are the property of their respective owners.



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

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

### 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 one with a different type number, confirm that the change will not lead to problems.

- The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

## Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
  2. 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.
  3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
  4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
  5. When exporting the 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. You should not use Renesas Electronics products or the 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. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
  6. 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.
  7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. 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 categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.  
"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.  
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.  
"Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
  8. 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.
  9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them 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 system manufactured by you.
  10. 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.
  11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
  12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.
- (Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.  
(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.



### 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: +44-1628-585-100, Fax: +44-1628-585-900

#### 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 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China  
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

#### 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.

1 HarbourFront Avenue, #06-10, Keppel Bay Tower, Singapore 098632  
Tel: +65-6213-0200, Fax: +65-6278-8001

#### 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.

11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea  
Tel: +82-2-558-3737, Fax: +82-2-558-5141