RX610 Group

CRC Code Calculation by Using DTC Block Transfer and Asynchronous Serial Communication with Appended CRC Codes by Using DMAC Data Transfer

Introduction

This application note presents an example of serial data transfer to the CRC calculator (CRC) by using block transfer by the data transfer controller (DTC). Note that the DMA controller (DMAC) is employed for data transfer via the serial communications interface (SCI) in order to perform asynchronous serial communication with appended CRC codes.

Target Device

RX610 Group MCUs

The sample program can be used with other RX Family MCUs that have the same I/O registers (peripheral device control registers) as the RX610 Group. Check the latest version of the manual for any additions and modifications to the functions used. Careful evaluation is recommended before using the sample application.

Contents

1. Specifications

In the sample application the DTC is used to generate transmit data with appended CRC codes and for CRC calculation on receive data with appended CRC codes, and the DMAC is used to transmit and receive 130 bytes of data by serial communication with appended CRC codes.

- The polynomial expression used for CRC generation is $X^{16} + X^{15} + X^2 + 1$, and CRC code generation for LSB-first communication is performed.
- SCI channel 1 is used and the communication format is 8 data bits, 1 stop bit, and no parity.
- The communication bit rate is 38,400 bps.
- DMAC channel 0 (receive) and channel 1 (transmit) are used.
- The DTC uses the SCI1 transmit data empty interrupt (TXI1) and the DMAC0 transfer end interrupt (DMTEND0).
- The CRC calculator initialization data (H'82) is prepared in the on-chip RAM beforehand.
- Transferring the CRC initialization data (H'82) to the CRC calculator specifies the polynomial expression for CRC generation, specifies CRC code generation for LSB-first communication, and clears the CRC data output register (CRCDOR).
- In the transmit data storage area of the on-chip RAM, 128 bytes of transmit data is prepared beforehand.
- The 128 bytes of transmit data comprises sequential values as follows: H'00, H'01, H'02, …, H'7D, H'7E, H'7F.

(1)   Transmit operation

1. Enable the SCI1 transmit data empty interrupt (TXI1).
2. Activate the DTC by means of the TXI1 interrupt.
3. By means of a DTC transfer, initialize the CRC calculator, generate the CRC code, and append the CRC code to the transmit data (making a total of 130 bytes).
4. After DTC transfer end at the TXI1 interrupt, set the TXI1 interrupt as the DMAC activation interrupt request destination by a CPU TXI1 interrupt, and enable the TXI1 interrupt.
5. Generate a TXI1 interrupt by writing the first byte of transmit data (H'00) to the transmit data register (TDR) by using the CPU, thereby activating DMAC1.
6. Transmit the transmit data with appended CRC code (total 129 bytes) by DMAC transfer.
7. After transmit end, reset the DTC and DMAC1.
8. Return to step 1 and repeat the transmit operation.

(2)   Receive operation

1. Enable the SCI1 receive data full interrupt (RXI1).
2. Activate the DMAC by means of the RXI1 interrupt.
3. Transfer the receive data with appended CRC code (total 130 bytes) to the on-chip RAM by DMAC transfer.
4. Activate the DTC by means of the DMAC0 transfer end interrupt (DMTEND0).
5. By means of a DTC transfer, initialize the CRC calculator, compute the CRC calculation result, and save the CRC calculation result.
6. After DTC transfer end, determine if a CRC error occurred by using the CPU DMTEND0 interrupt.
7. If a CRC error occurred, stop receive operation and illuminate the LED connected to the I/O port to provide notification of the error. If no CRC error occurred, reset the DTC and DMAC0.
8. Return to step 1 and repeat the receive operation.

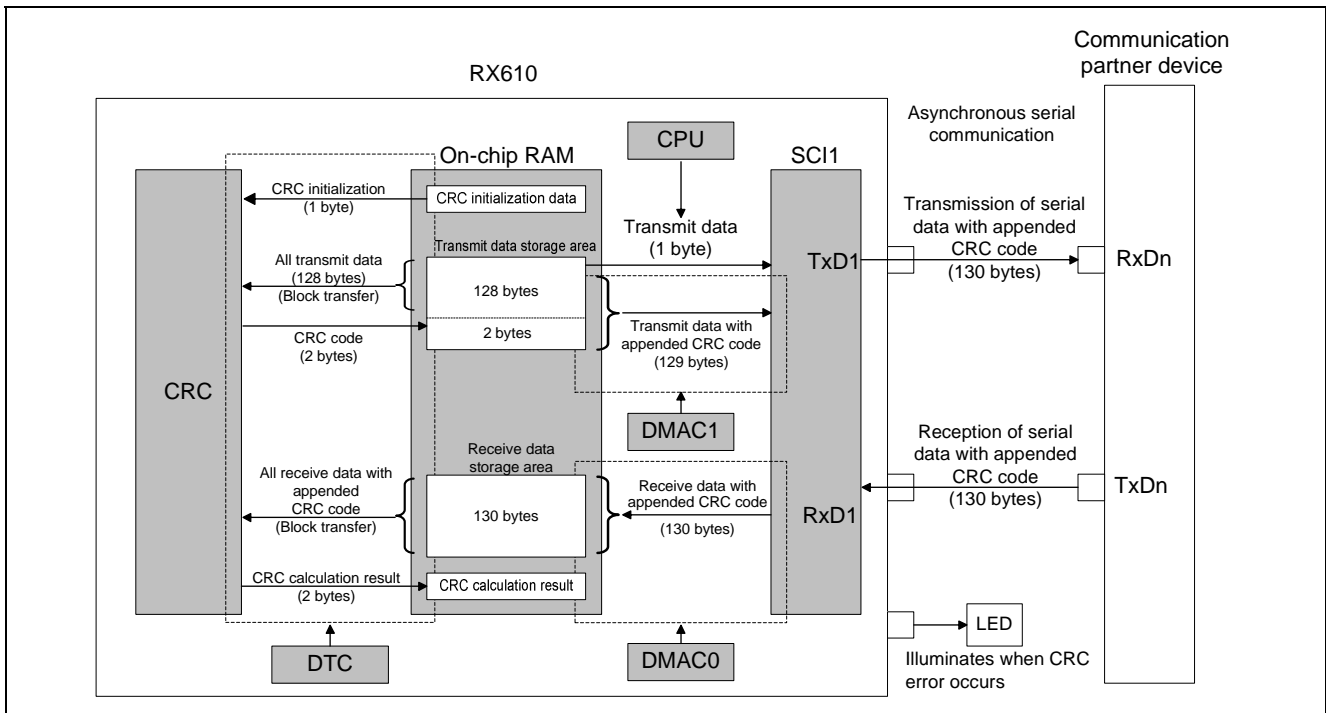Figure 1 shows the specifications used in the sample application.

Figure 1   Specifications

2.   Operation Confirmation Environment

Table 1 lists the components of the environment used to confirm operation of the sample application.

Table 1   Operation Confirmation Environment

Item	Name
Device	RX610 (R5F56108VNFP)
Board	Renesas Starter Kit (R0K556100S000BE)
Power supply voltage	5.0 V (CPU operating voltage of 3.3 V)
Input clock	12 MHz (ICLK = 100 MHz, PCLK = 50 MHz, BCLK = 25 MHz)
Operating temperature	Room temperature
HEW	Version 4.08.00.011
Toolchain	RX Standard Toolchain (V.1.0.0.0)
Debugger/Emulator	E20 emulator
Debugger component	RX E1/E20 SYSTEM V.1.00.00.000

3.　Functions

- Clock generation circuit
- Low power consumption functions
- I/O ports
- Interrupt control unit (ICU)
- Serial communications interface (SCI)
- DMA controller (DMAC)
- CRC calculator (CRC)*
- Data transfer controller (DTC)

Note: 　* 　When using the CRC calculator, writing other data to the CRCDIR register while CRC calculation and
storing of the CRC calculation result are in progress (for example, when two or more different bus masters
are using the CRC calculator) can affect the calculation result. In the case of this application note, the
DMAC must not access the CRC calculator while the DTC is accessing it, as this will cause the calculation
result to be destroyed. Note that multiple accesses to the CRC by the DTC do not pose any problem.

For details of the above functions, see the RX610 Group Hardware Manual.

4.　Operation

4.1　Operating Mode Settings

In the sample application, mode pins MD1 and MD0 are both set to 1 to select single-chip mode as the operating mode,
the ROME bit in system control register 0 (SYSCR0) is set to 1 to enable the on-chip ROM, and the EXBE bit in the
SYSCR0 register is cleared to 0 to disable the external bus.

Table 2 lists the operating mode setting used in the sample application.

Table 2　Operating Mode Settings

Mode Pin		SYSCR0 Register				
MD1	MD0	ROME	EXBE	Operating Mode	On-Chip ROM	External Bus
1	1	1	0	Single-chip mode	Enabled	Disabled

Note: The initial values of the ROME and EXBE bits in the SYSCR0 register are 1 and 0, respectively, so
the sample program does not make settings to the SYSCR0 register.

4.2　Clock Settings

The evaluation board used by the sample application has a 12.5 MHz crystal resonator mounted on it.
Therefore, the sample application sets frequencies of the system clock (ICLK), peripheral module clock (PCLK), and
external bus clock (BCLK) to ×8 (100 MHz), ×4 (50 MHz), and ×2 (25 MHz), respectively.

4.3　Endian Mode Settings

The sample application supports both big endian and little endian modes. Table 3 lists the hardware-based endian mode
settings.

Table 3　Endian Mode Settings (Hardware)

MDE Pin	Endian Mode
0	Little endian
1	Big endian

Table 4 lists the endian mode settings available as MCU options in the compiler options.

Table 4   Endian Mode Settings (Compiler Options)

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

Note: Set the MDE pins to match the endian mode selected in the program's compiler options.

4.4   Bit Order Settings

The sample application supports both left and right bit order settings. Table 5 lists the bit order settings available as MCU options in the compiler options.

Table 5   Bit Order Settings (Compiler Options)

MCU Option	Bit Order
bit_order = right	The order of members in the bit field is allocated starting from the lowest bit (default when no option specified)
bit_order = left	The order of members in the bit field is allocated starting from the highest bit

Notes: 1. The sample application uses the I/O register definition file (iodefine.h) to specify the bit order in bit fields. The bit order is specified as left by the **#pragma bit_order** extension in the I/O register definition file, so the order of members in the bit field is allocated starting from the highest bit.

      2. When the **bit_order** compiler option and **#pragma bit_order** extension are both specified, the **#pragma bit_order** extension takes precedence. Therefore, the order of members in the bit field is allocated starting from the highest bit, as specified in the I/O register definition file, regardless of the setting of the **bit_order** compiler option.

4.5   SCI Settings

The sample application uses SCI channel 1 to transmit and receive asynchronous serial data. Table 6 lists the SCI setting conditions.

Table 6   SCI Setting Conditions

Channel used	SCI 1
Communication mode	Asynchronous mode
Interrupts	• Transmit data empty interrupt (TXI1) • Receive data full interrupt (RXI1) • Receive error interrupt (ERI1) • Transmit end interrupt (TEI1)
Communication speed	38400 bps (PCLK = 50 MHz)
Data length	8 data bits
Stop bits	1 stop bit
Parity	None

4.6   CRC Settings

The sample application uses the CRC calculator to generate 2-byte CRC codes. Table 7 lists the CRC setting conditions.

Table 7   CRC Setting Conditions

Polynomial expression for generating CRC	$X^{16} + X^{15} + X^2 + 1$
CRC calculation	CRC code generation for LSB-first communication

4.7   DMAC Settings

The sample application uses DMC channel 0 (receive) and channel 1 (transmit) for SCI data transfer. Table 8 lists the setting conditions for DMAC0 and table 9 lists the setting conditions for DMAC1.

Table 8   DMAC0 Setting Conditions (Receive)

Transfer mode	Single-operand transfer
Transfer count	130
Transfer data size	Byte
Transfer data contents	Receive data with appended CRC code (130 bytes)
Transfer source	SCI1 receive data register (SCI1.RDR)
Transfer destination	On-chip RAM
Transfer source address	Fixed transfer source
Transfer destination address	Transfer destination address incremented after transfer
Activation source	SCI1 receive data full interrupt (RXI1)
Interrupts	Interrupts to CPU enabled after end of specified data transfer

Table 9   DMAC1 Setting Conditions (Transmit)

Transfer mode	Single-operand transfer
Transfer count	129
Transfer data size	Byte
Transfer data contents	Transmit data with appended CRC code (129 bytes)
Transfer source	On-chip RAM
Transfer destination	SCI1 transmit data register (SCI1.TDR)
Transfer source address	Transfer source address incremented after transfer
Transfer destination address	Fixed transfer destination
Activation source	SCI1 transmit data empty interrupt (TXI1)
Interrupts	Interrupts to CPU requested after end of specified data transfer

4.8   DTC Settings

The sample applications uses the DTC to generate CRC codes and for data transfer of CRC calculation results. Table 10 lists the DTC setting conditions (receive) and table 11 lists the DTC setting conditions (transmit).

Table 10 DTC Setting Conditions (Receive)

	1st Transfer Information (CRC Calculator Initialization)	2nd Transfer Information (CRC Calculation Result Computation)	3rd Transfer Information (Saving of CRC Calculation Result)
Transfer mode	Normal transfer mode	Block transfer mode	Normal transfer mode
Transfer count	1	1	1
Block size	—	130 bytes	—
Transfer data size	Byte	Byte	Word
Transfer data contents	CRC initialization data (H'82) (1 byte)	Receive data with appended CRC code (130 bytes)	CRC calculation result (2 bytes)
Transfer source	On-chip RAM	On-chip RAM	CRC data output register (CRCDOR)
Transfer destination	CRC control register (CRCCR)	CRC data input register (CRCDIR)	On-chip RAM
Transfer source address	Fixed transfer source	Transfer destination address incremented after transfer	Fixed transfer source
Transfer destination address	Fixed transfer destination	Fixed transfer destination	Fixed transfer destination
Activation source	DMAC0 transfer end interrupt (DMTEND0)	End of 1st transfer	End of 2nd transfer
Chain transfer	Enabled	Enabled	Disabled
Interrupts	None	None	Interrupts to CPU enabled after end of specified data transfer

Table 11 DTC Setting Conditions (Transmit)

	1st Transfer Information (CRC Calculator Initialization)	2nd transfer information (CRC Code Generation)	3rd Transfer Information (CRC Code Appending)
Transfer mode	Normal transfer mode	Block transfer mode	Normal transfer mode
Transfer count	1	1	1
Block size	—	128 bytes	—
Transfer data size	Byte	Byte	Word
Transfer data contents	CRC initialization data (H'82) (1 byte)	128 bytes from H'00 to H'7F	CRC calculation result (2 bytes)
Transfer source	On-chip RAM	On-chip RAM	CRC data output register (CRCDOR)
Transfer destination	CRC control register (CRCCR)	CRC data input register (CRCDIR)	On-chip RAM
Transfer source address	Fixed transfer source	Transfer destination address incremented after transfer	Fixed transfer source
Transfer destination address	Fixed transfer destination	Fixed transfer destination	Fixed transfer destination
Activation source	SCI1 transmit data empty interrupt (TXI1)	End of 1st transfer	End of 2nd transfer
Chain transfer	Enabled	Enabled	Disabled
Interrupts	None	None	Interrupts to CPU requested after end of specified data transfer

4.9   Operation

4.9.1   Receive Operation

[1] Wait for SCI1 receive data full interrupt (RXI1) request.

[2] At SCI1 receive data full interrupt request, activate DMAC0.

[3] Transfer receive data with appended CRC code (total 130 bytes) from SCI1 receive data register (RDR) to on-chip RAM by using DMAC0.

[4] After 130 bytes of data is received, activate DTC by means of DMAC0 transfer end interrupt (DMTEND0) request.

[5] Transfer CRC initialization data (H'82) from on-chip RAM to CRC control register (CRCCR) by using DTC, and initialize CRC calculator.

[6] After initialization of CRC calculator, use chain transfer to perform block transfer of receive data with appended CRC (total 130 bytes) from on-chip RAM to CRC data input register (CRCDIR), and compute CRC calculation result.

[7] Use chain transfer to save computed CRC calculation result in CRC data output register (CRCDOR) to on-chip RAM.

[8] After end of DTC transfer, issue interrupt request to CPU (DMTEND0 interrupt request).

[9] As part of DMTEND0 interrupt routine, read CRC calculation result from on-chip RAM and determine if CRC error occurred. If no CRC error, reset DTC and DMAC0 and return to step 1.

Figure 2 is a block diagram illustrating receive operation.
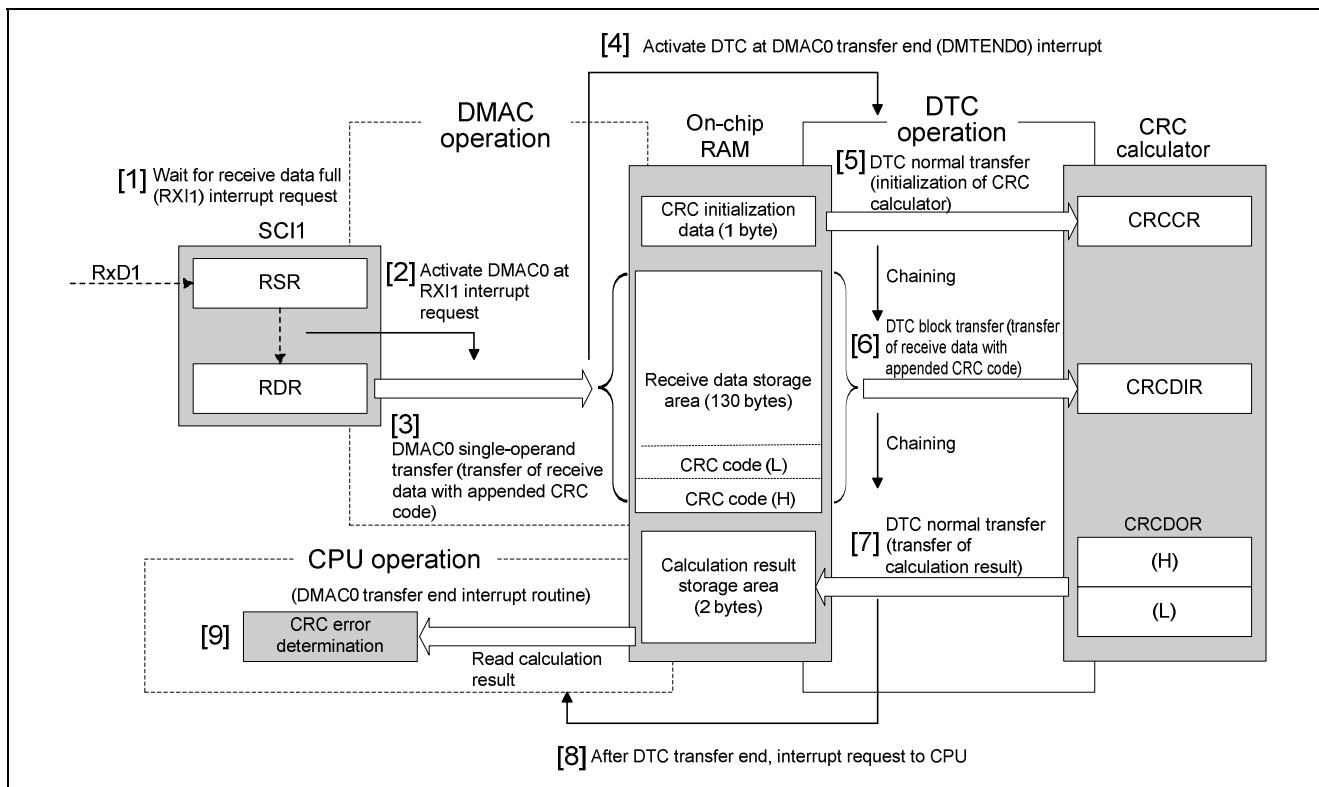
Figure 2   Block Diagram of Receive Operation

4.9.2   Transmit Operation

[1] Wait for SCI1 transmit data empty interrupt (TXI1) request.

[2] At SCI1 transmit data empty interrupt request, activate DTC.

[3] Transfer CRC initialization data (H'82) from on-chip RAM to CRCCR by using DTC, and initialize CRC calculator.

[4] After initialization of CRC calculator, use chain transfer to perform block transfer of transmit data (128 bytes) from on-chip RAM to CRCDIR, and generate 2-byte CRC code.

[5] Use chain transfer to save generated CRC code in CRCDOR to on-chip RAM.

[6] After end of DTC transfer, issue interrupt request to CPU (TXI1 interrupt request).

[7] As part of TXI1 interrupt routine, read MDE bit in mode monitor register (MDMONR). If MDE is set to 1 (big endian), exchange CRC code (L) and CRC code (H).

[8] Set TXI1 interrupt as DMAC1 activation interrupt request destination and enable TXI1 interrupt requests.

[9] Generate TXI interrupt to activate DMAC1 by writing first byte of transmit data (H'00) to SCI1 transmit data register (TDR).

[10] Transfer transmit data with appended CRC code (total 129 bytes) to TDR by using DMAC1. After specified number of transfers end, reset DTC and DMAC1 and return to step 1.

Figure 3 is a block diagram illustrating transmit operation.
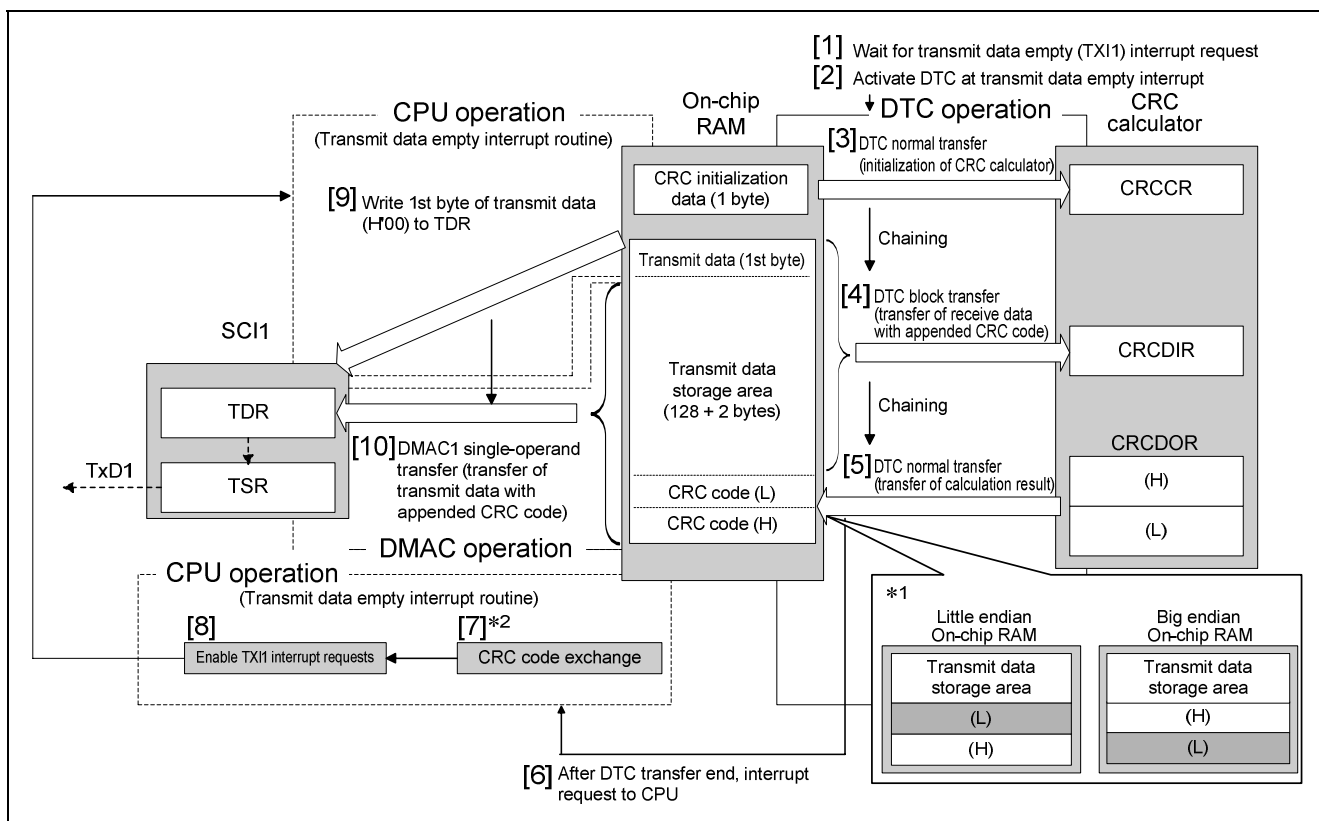
Figure 3   Block Diagram of Transmit Operation

Notes: 1. CRCDOR is a 16-bit register, so the endian mode setting affects data transfer to the on-chip RAM by the DTC. CRC code (H) is assigned to the last byte in the transmit data storage area in little-endian mode, and CRC code (L) is assigned to the last byte in the transmit data storage area in big-endian mode.

2. The CRC code is inverted when the endian mode is set to big endian, so it is necessary to exchange the bytes in the CRC code. The sample application performs processing to exchange CRC code (L) and CRC code (H) so that CRC code (H) is the last byte in the transmit data storage area in the on-chip RAM. Note that this processing is not needed when the sample program in this application note is used in little-endian mode.

5. Software

5.1 Constants

Table 12 lists the constants used in the sample code.

Table 12 Constants

Constant	Setting Value	Description
DMAC_CNT_TX	129	DMAC (transmit) transfer count
DMAC_CNT_RX	130	DMAC (receive) 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
SET	1	Set flag
CLEAR	0	Clear flag

5.2 Structures and Unions

Figures 4 and 5 illustrate the structures and unions used in the sample code.

```
struct st_dtc_full{
   union{
     unsigned long LONG;
     struct{
        unsigned long MRA_MD    :2; /* MD bits in MRA */
        unsigned long MRA_SZ    :2; /* SZ bits in MRA */
        unsigned long MRA_SM    :2; /* SM bits in MRA */
        unsigned long           :2;
        unsigned long MRB_CHNE  :1; /* CHNE bit in MRB */
        unsigned long MRB_CHNS  :1; /* CHNS bit in MRB */
        unsigned long MRB_DISEL :1; /* DISEL bit in MRB */
        unsigned long MRB_DTS   :1; /* DTS bit in MRB */
        unsigned long MRB_DM    :2; /* DM bits in MRB */
        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;
};
```

Figure 4 Structures and Unions Used in Sample Code (DTC Transfer Information)

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

Figure 5   Structures and Unions Used in Sample Code (Serial Data with Appended CRC Code)

5.3   Variables

Table 13 lists the variables used in the sample code.

Table 13 Variables

Type	Variable	Description	Function Used By
unsigned char	send_end_flag	Transmit end flag 0: Transmit in progress 1: Transmit end	main, int_sci_tei1
unsigned char	receive_end_flag	Receive end flag 0: Receive in progress/waiting 1: Receive end	main, DMAC0_dmtend_int
unsigned char	crc_error_flag	CRC error flag 0: No error 1: Error occurred	main, DMAC0_dmtend_int
unsigned char	CRC_RESET_DATA	CRC initialization data	main, dtc_send, dtc_receive
st_dtc_full	dtc_tx[3]	DTC transfer information (transmit)	dtc_init, dtc_send
st_dtc_full	dtc_rx[3]	DTC transfer information (receive)	dtc_init, dtc_receive
st_ram_data	ram_tx	Transmit data with appended CRC code	main, dtc_send dmac1_send, int_sci_txi1
st_ram_data	ram_rx	Receive data with appended CRC code	dtc_receive, dmac0_receive
unsigned short	CRC_RESULT	CRC calculation result	dtc_receive, DMAC0_dmtend_int
void	*dtc_table[256]	DTC vector table	dtc_init

5.4 Functions

Table 14 lists the functions used in the sample application.

Table 14 Functions

Function	Description
HardwareSetup	Initialization, clock setting, canceling of module stop state
main	Main process ICU initial settings, interrupt level settings
sci1_init	SCI initial settings, transfer clock setting
dmac0_init	DMAC0 initial settings
dmac1_init	DMAC1 initial settings
dtc_init	DTC initial settings
dmac0_receive	DMAC0 (receive) transfer source address, transfer destination address, and transfer count settings
dmac1_send	DMAC1 (transmit) transfer source address, transfer destination address, and transfer count settings
dtc_receive	Allocation of DTC transfer information (receive)
dtc_send	Allocation of DTC transfer information (transmit)
crc_error	CRC error indication
byte_reverse_16	Exchanging of top and bottom bytes of data word
DMAC0_dmtend_int	DMAC0 transfer end interrupt handler
DMAC1_dmtend_int	DMAC1 transfer end interrupt handler
int_sci_tei1	Transmit end interrupt handler
int_sci_txi1	Transmit data empty interrupt handler
int_sci_eri1	Receive error interrupt handler

5.5 Function Specifications

The specifications of the functions used in the sample code are listed below.

sci1_init	
Overview	Makes initial settings to SCI1.
Header	iodefine.h
Declaration	void sci1_init(void)
Description	See table 6 in section 4.5 for setting details.
Arguments	None
Return values	None
Notes	

dmac0_init	
Overview	Makes initial settings to DMAC0.
Header	iodefine.h
Declaration	void dmac0_init(void)
Description	See table 8 in section 4.7 for setting details.
Arguments	None
Return values	None
Notes	

dmac1_init	
Overview	Makes initial settings to DMAC1.
Header	iodefine.h
Declaration	void dmac1_init(void)
Description	See table 9 in section 4.7 for setting details.
Arguments	None
Return values	None
Notes	

dtc_init	
Overview	Makes initial settings to DTC.
Header	iodefine.h
Declaration	void dtc_init(void)
Description	See tables 10 and 11 in section 4.8 for setting details.
Arguments	None
Return values	None
Notes	

dmac0_receive	
Overview	Sets the DMAC0 (receive) transfer source address, transfer destination address, and transfer count.
Header	iodefine.h
Declaration	void dmac0_receive(void)
Description	The following settings are made. • Transfer source address: SCI1 receive data register (SCI1.RDR) • Transfer destination address: Receive data storage area in on-chip RAM (ram_rx.BUFF.ALL) • Transfer count: 130
Arguments	None
Return values	None
Notes	Makes initial settings to, and resets, DMAC0.

dmac1_send	
Overview	Sets the DMAC1 (transmit) transfer source address, transfer destination address, and transfer count.
Header	iodefine.h
Declaration	void dmac1_send(void)
Description	The following settings are made. • Transfer source address: Transmit data storage area in on-chip RAM (ram_tx.BUFF.ALL[1]) • Transfer destination address: SCI1 transmit data register (SCI1.TDR) • Transfer count: 129
Arguments	None
Return values	None
Notes	Makes initial settings to, and resets, DMAC1.

dtc_receive	
Overview	Sets DTC transfer information (receive).
Header	iodefine.h, dtc_def.h
Declaration	void dtc_receive (void)
Description	See table 10 in section 4.8 for setting details.
Arguments	None
Return values	None
Notes	Makes initial settings to, and resets, the DTC (receive).

dtc_send	
Overview	Sets DTC transfer information (transmit).
Header	iodefine.h, dtc_def.h
Declaration	void dtc_send (void)
Description	See table 11 in section 4.8 for setting details.
Arguments	None
Return values	None
Notes	Makes initial settings to, and resets, the DTC (transmit).

RX610 Group

CRC Code Calculation by Using DTC Block Transfer and Asynchronous Serial Communication with Appended CRC Codes by Using DMAC Data Transfer

crc_error	
Overview	Displays a CRC error indication.
Header	iodefine.h
Declaration	void crc_error(void)
Description	Illuminates the LED connected to port 03.
Arguments	None
Return values	None
Notes	

byte_reverse_16	
Overview	Exchanges the top and bottom bytes of the data word.
Header	iodefine.h
Declaration	unsigned short byte_reverse_16(unsigned short rev_data)
Description	Returns a value in which the top and bottom bytes of the argument rev_data (unsigned short type) have been reversed.
Arguments	rev_data (unsigned short type)
Return values	(rev_data << 8)	(rev_data >> 8)
Notes	Used to exchanges the bytes of the CRC code.

5.6   Processing Sequence

Figures 6 to 23 show the processing sequences of the sample program.
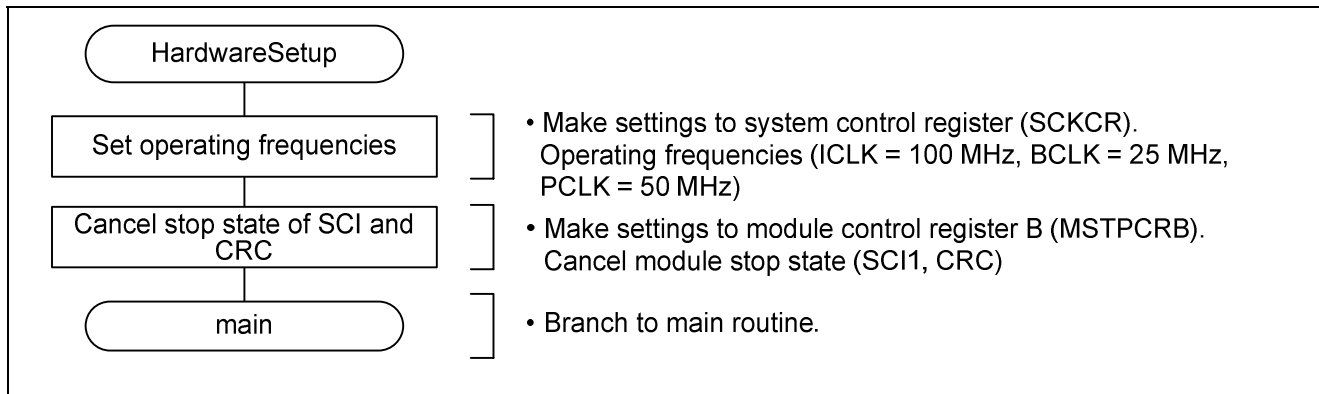
Figure 6   Initialization
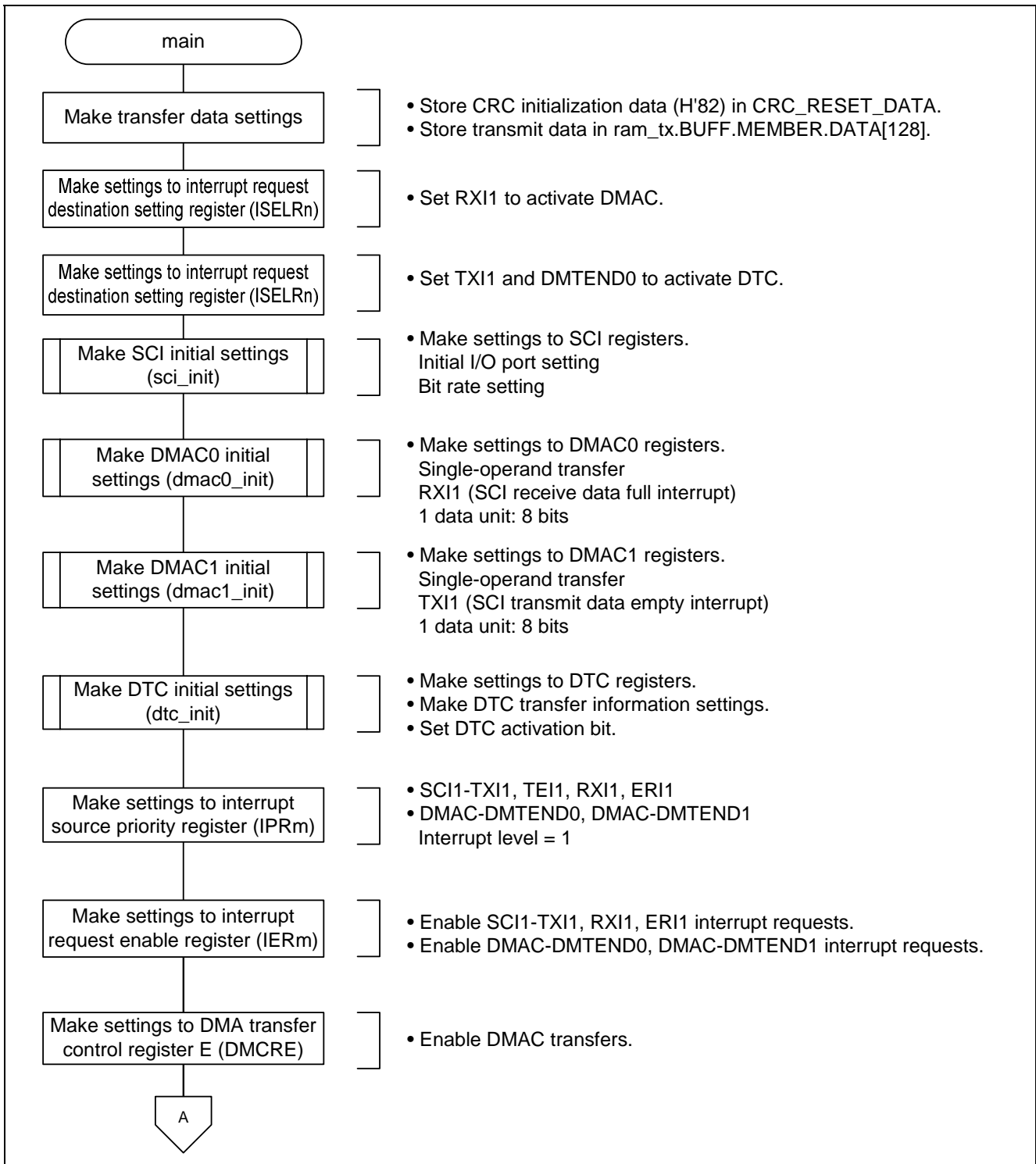
Figure 7   Main Process (1)
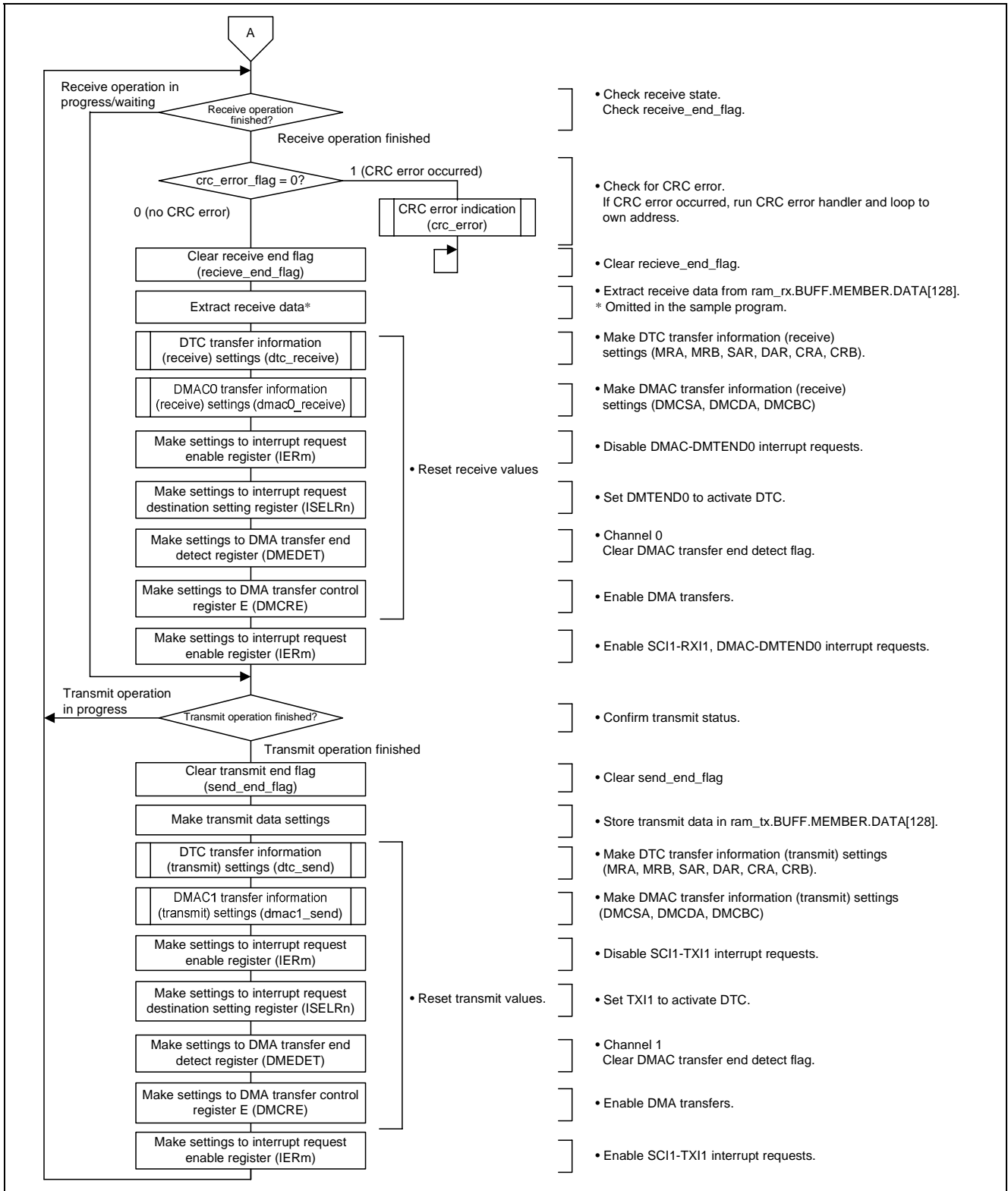
Figure 8   Main Process (2)
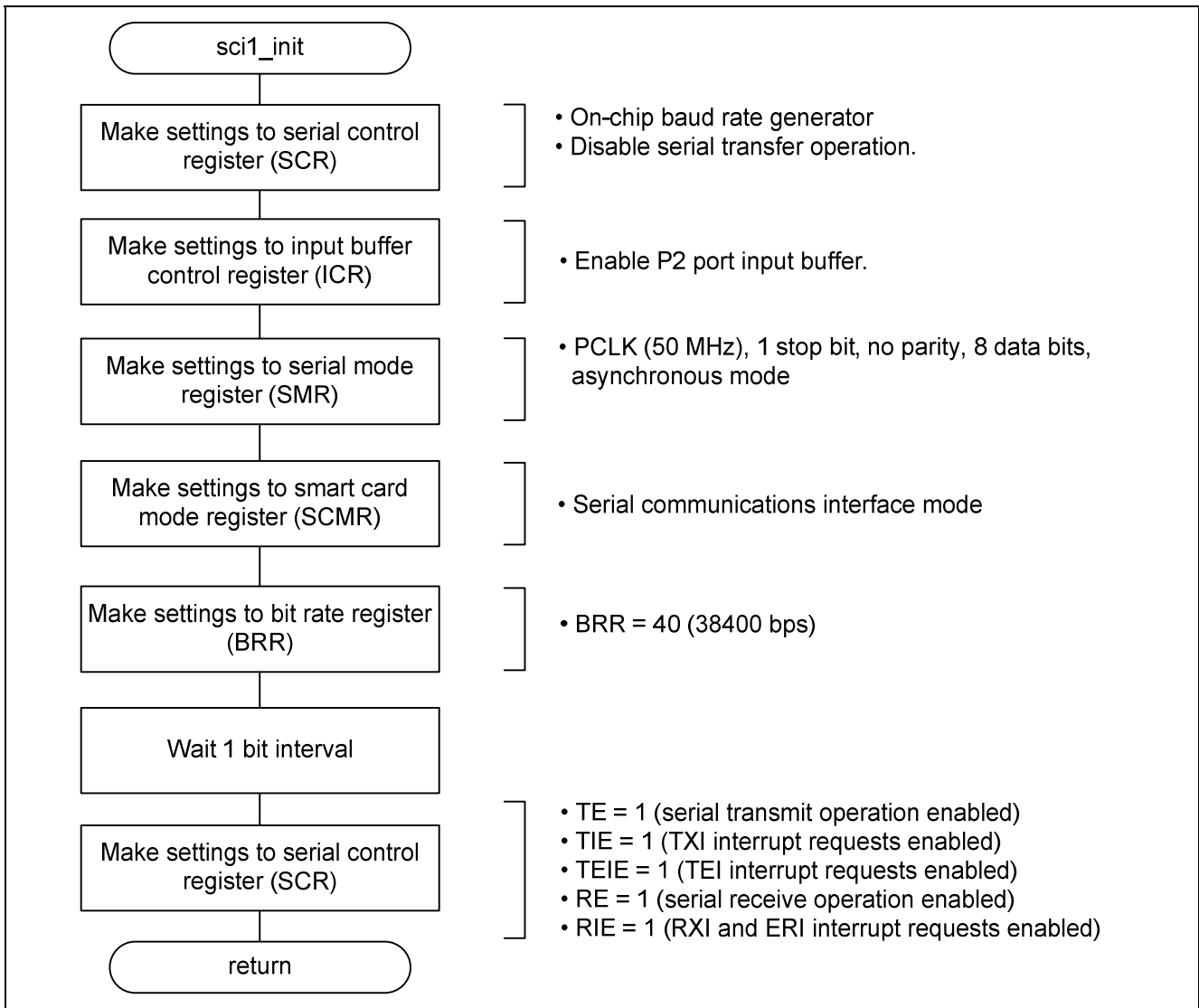
Figure 9 SCI Initial Settings

```
                 ┌────────────────────────┐
                 │      dmac0_init        │
                 └────────────────────────┘
                            │
 ┌──────────────────────────────┐      • Disable DMA transfers.
 │ Make settings to DMA transfer │
 │ control register E (DMCRE)    │
 └──────────────────────────────┘
                            │
 ┌──────────────────────────────┐      • Incrementing of transfer destination address
 │ Make settings to DMA mode     │      • Foxed transfer source address
 │ register (DMMOD)              │      • 1 data unit: 8 bits
 └──────────────────────────────┘
                            │
 ┌──────────────────────────────┐      • Single-operand transfer
 │ Make settings to DMA control  │      • RXI1 (SCI1 receive data full interrupt)
 │ register A (DMCRA)            │
 └──────────────────────────────┘
                            │
 ┌──────────────────────────────┐      • Initialize DMAC internal state.
 │ Make settings to DMA control  │
 │ register B (DMCRB)            │
 └──────────────────────────────┘
                            │
 ┌──────────────────────────────┐      • When DMAC transfer finishes, clear DEN bit to 0.
 │ Make settings to DMA control  │
 │ register C (DMCRC)            │
 └──────────────────────────────┘
                            │
 ┌──────────────────────────────┐      • Clear DMA request bit.
 │ Make settings to DMA control  │
 │ register D (DMCRD)            │
 └──────────────────────────────┘
                            │
 ┌──────────────────────────────┐      • Transfer source address
 │ │  dmac0 settings        │ │          Transfer destination start address
 │ │  (dmac0_send)          │ │          DMA transfer byte count
 └──────────────────────────────┘
                            │
 ┌──────────────────────────────┐      • Enable DMAC0 interrupt.
 │ Make settings to DMA interrupt│
 │ control register (DMICNT)     │
 └──────────────────────────────┘
                            │
 ┌──────────────────────────────┐      • Make DMAC operation settings.
 │ Make settings to DMA start    │
 │ register (DMSCNT)             │
 └──────────────────────────────┘
                            │
                 ┌────────────────────────┐
                 │        return          │
                 └────────────────────────┘
```

Figure 10   DMAC0 Initial Settings
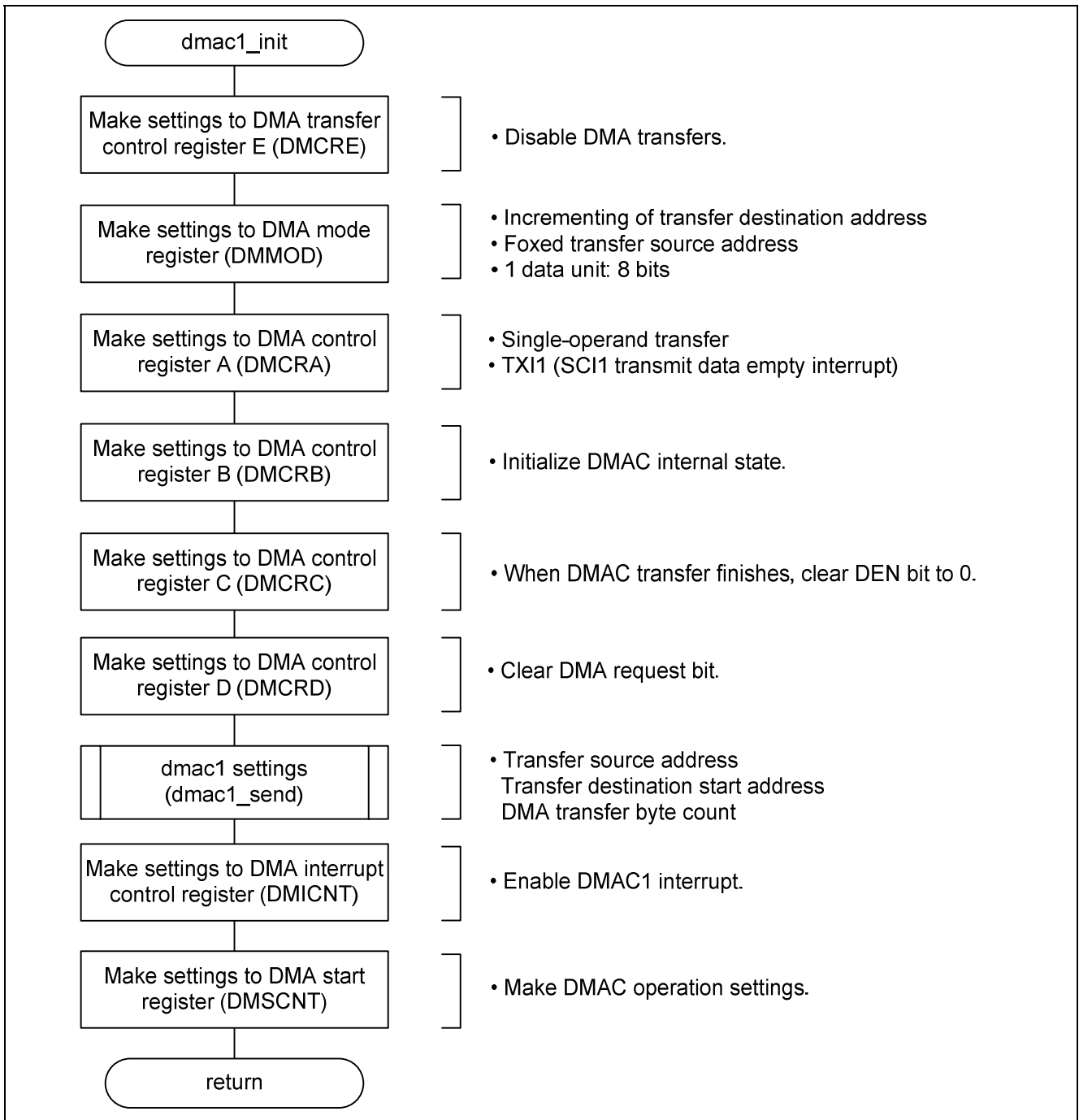
Figure 11   DMAC1 Initial Settings
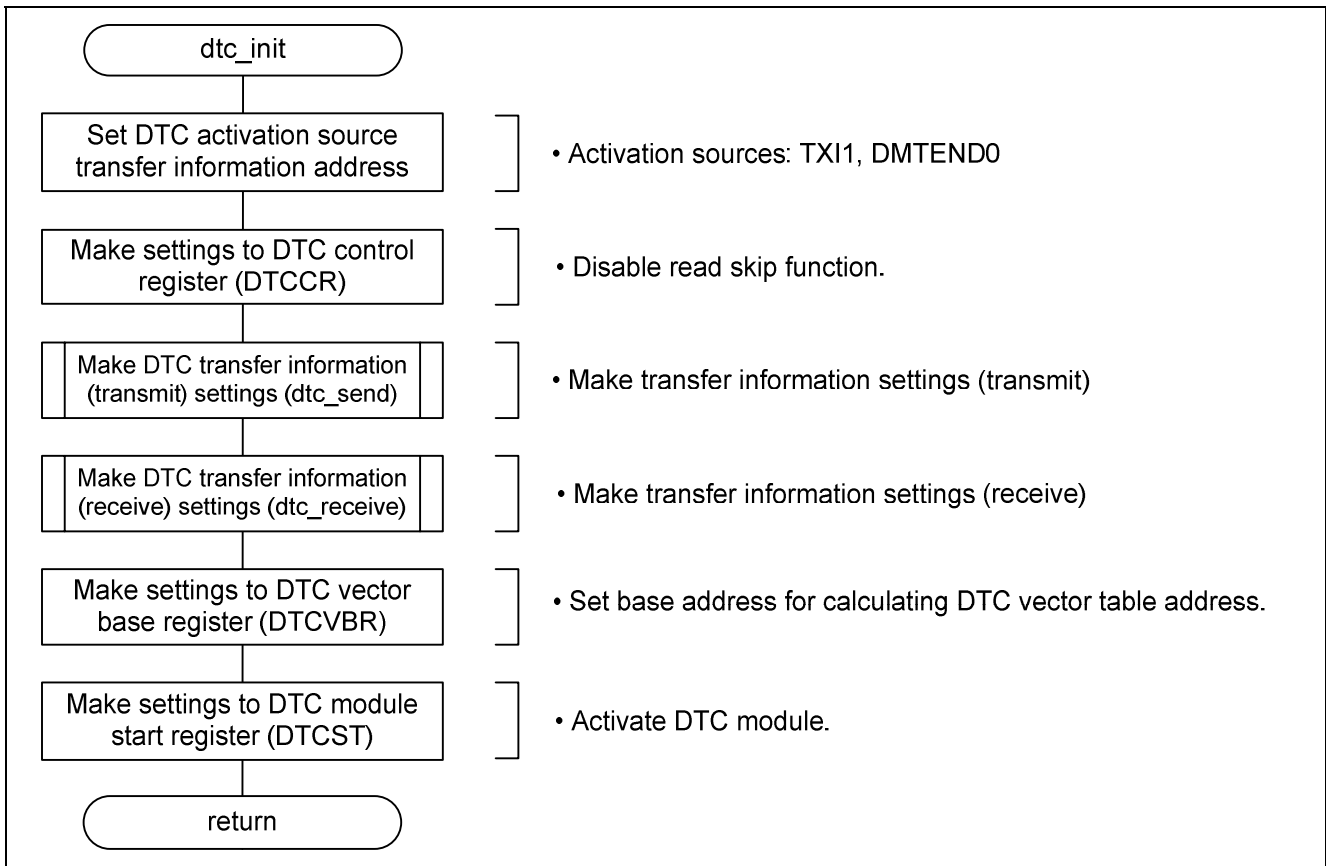
Figure 12   DTC Initial Settings
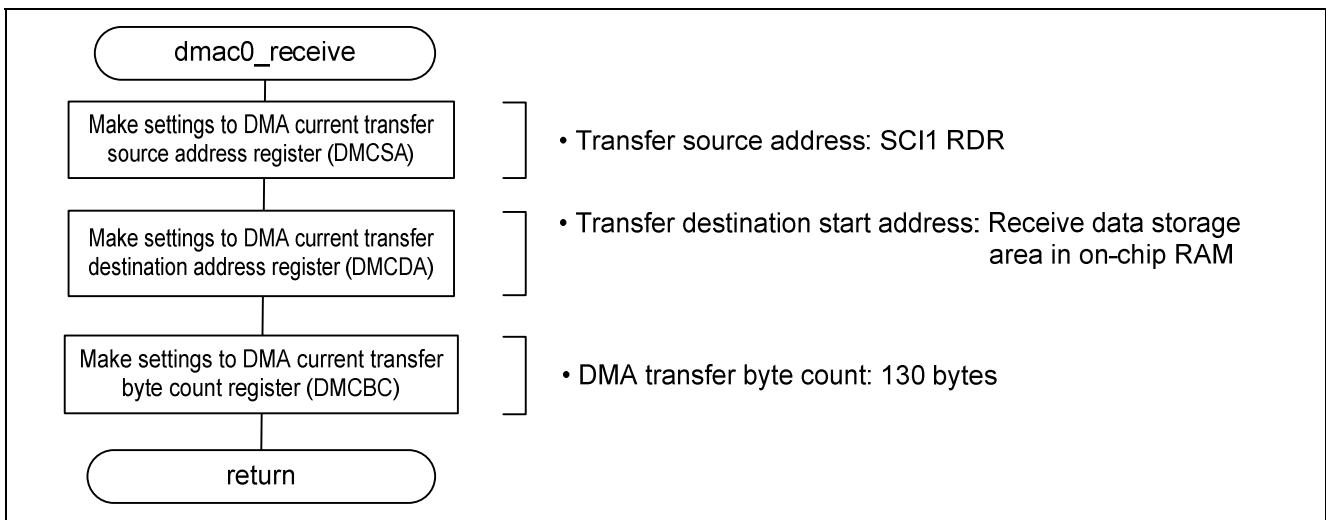
Figure 13   DMAC0 (Receive) Transfer Source Address, Transfer Destination Address, and Transfer Count Settings
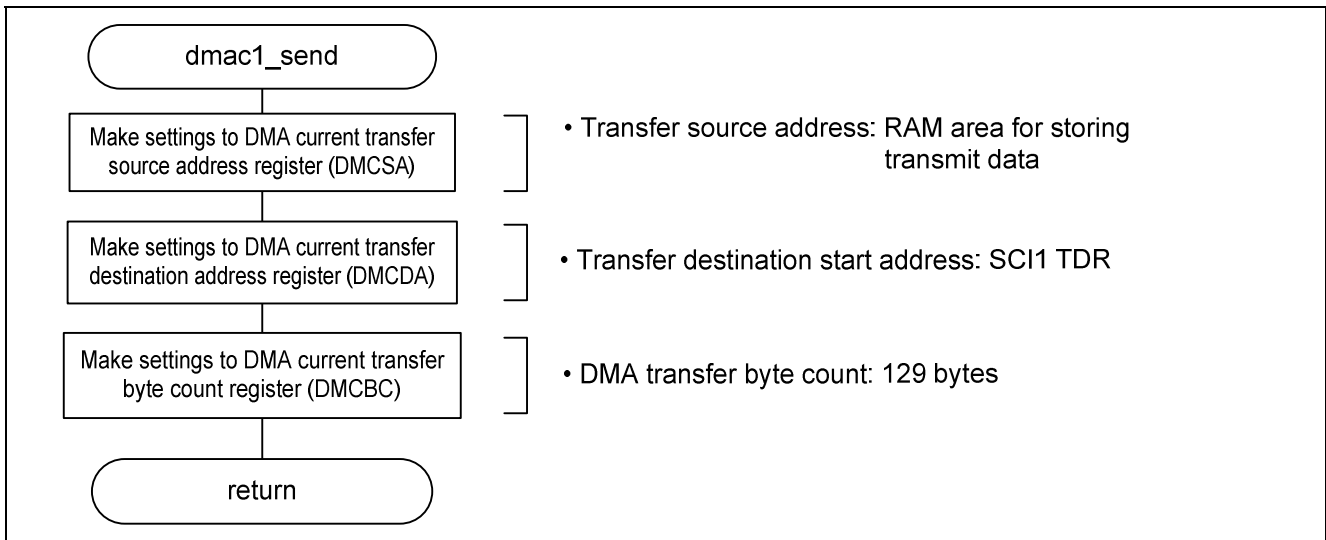
**Figure 14   DMAC1 (Transmit) Transfer Source Address, Transfer Destination Address,
and Transfer Count Settings**
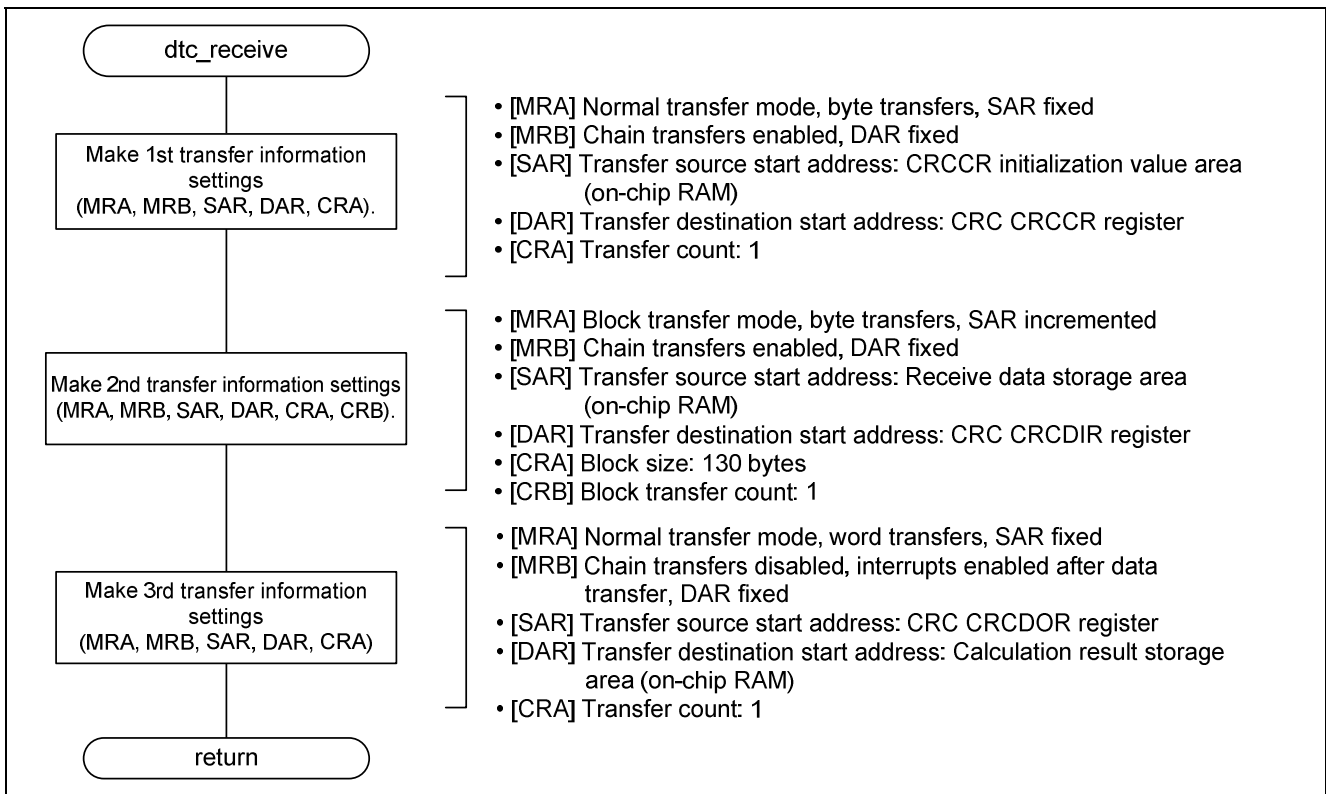
Figure 15   Receive Transfer Information Allocation

dtc_send

Make 1st transfer information settings
(MRA, MRB, SAR, DAR, CRA).

- [MRA] Normal transfer mode, byte transfers, SAR fixed
- [MRB] Chain transfers enabled, DAR fixed
- [SAR] Transfer source start address: CRCCR initialization value area (on-chip RAM)
- [DAR] Transfer destination start address: CRC CRCCR register
- [CRA] Transfer count: 1

Make 2nd transfer information settings
(MRA, MRB, SAR, DAR, CRA, CRB).

- [MRA] Block transfer mode, byte transfers, SAR incremented
- [MRB] Chain transfers enabled, DAR fixed
- [SAR] Transfer source start address: Transmit data storage area (on-chip RAM)
- [DAR] Transfer destination start address: CRC CRCDIR register
- [CRA] Block size: 128 bytes
- [CRB] Block transfer count: 1

Make 3rd transfer information settings
(MRA, MRB, SAR, DAR, CRA)

- [MRA] Normal transfer mode, word transfers, SAR fixed
- [MRB] Chain transfers disabled, interrupts enabled after data transfer, DAR fixed
- [SAR] Transfer source start address: CRC CRCDOR register
- [DAR] Transfer destination start address: Transmit data storage area (on-chip RAM)
- [CRA] Transfer count: 1

return

Figure 16   Transmit Transfer Information Allocation

crc_error

Illuminate LED

- Indicate CRC error.
- Illuminate LED connected to port 03.

return

Figure   17 CRC Error Indication

byte_reverse_16

Exchange top and bottom bytes of argument (data word)

- Exchange top and bottom bytes of argument and send back as return value.

return

Figure 18   Exchanging Bytes of Data Word

Figure 19   DMAC0 Transfer End Interrupt Handler
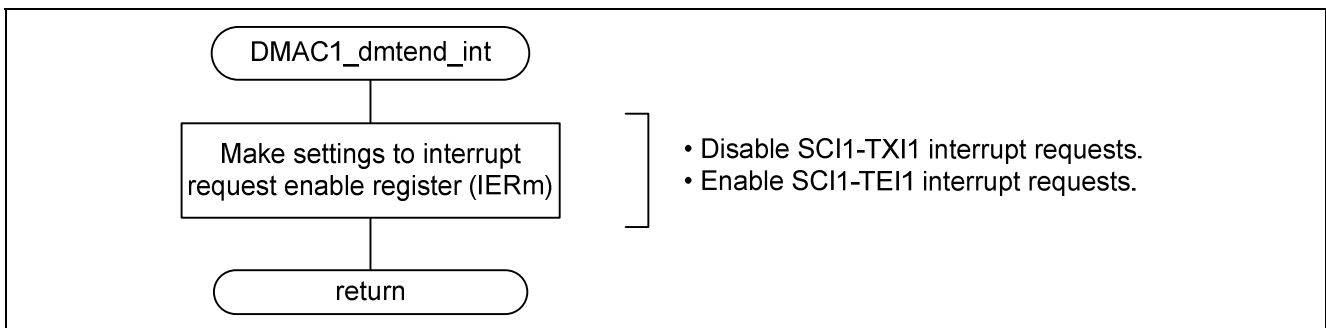
Figure 20   DMAC1 Transfer End Interrupt Handler
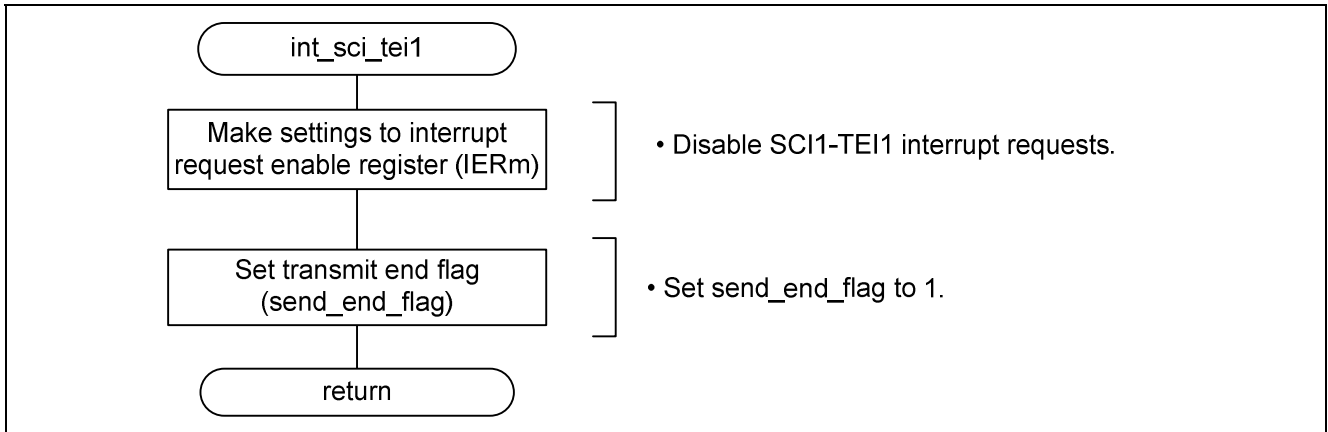
Figure 21   Transmit End Interrupt Handler
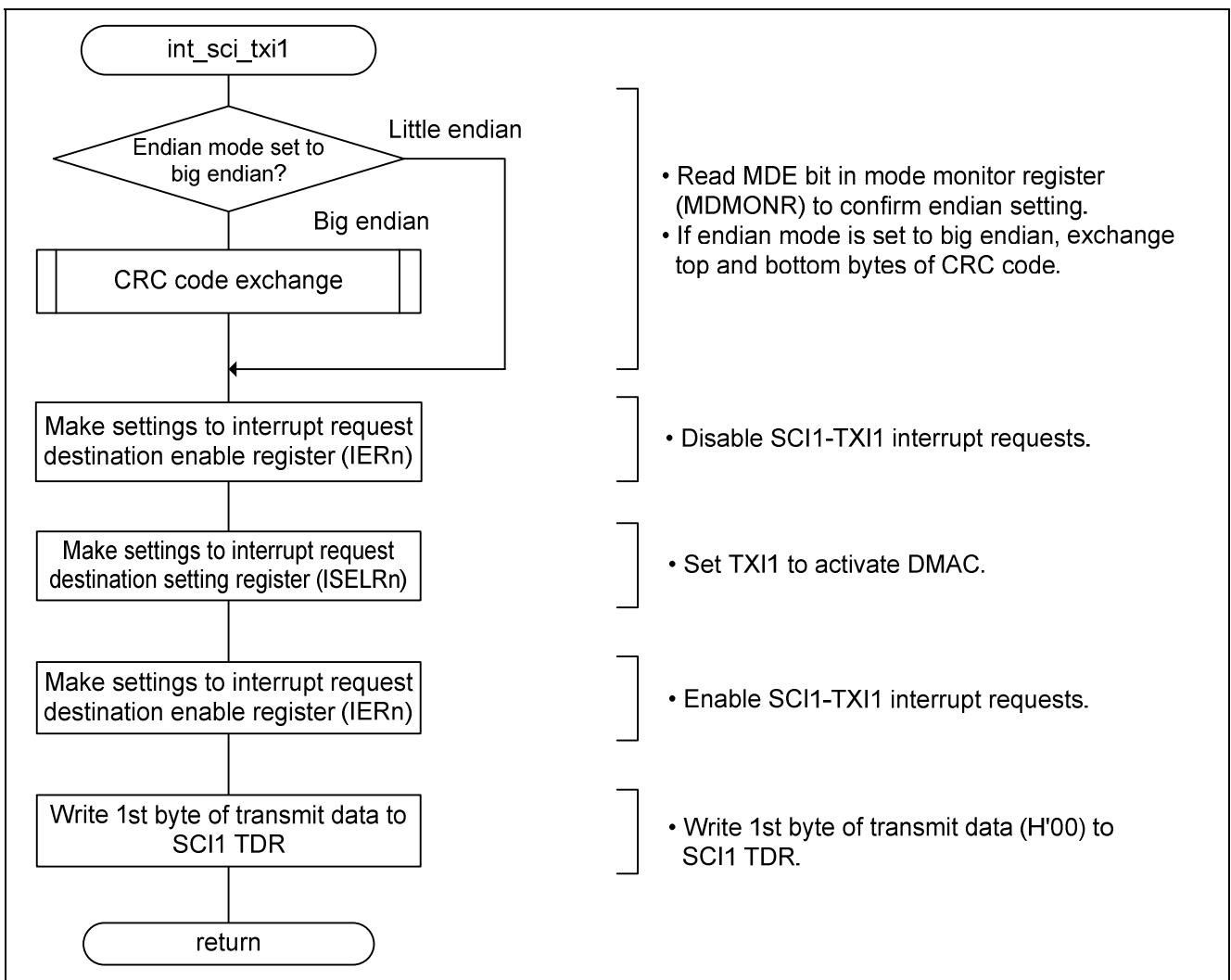
Figure 22   Transmit Data Empty Interrupt Handler
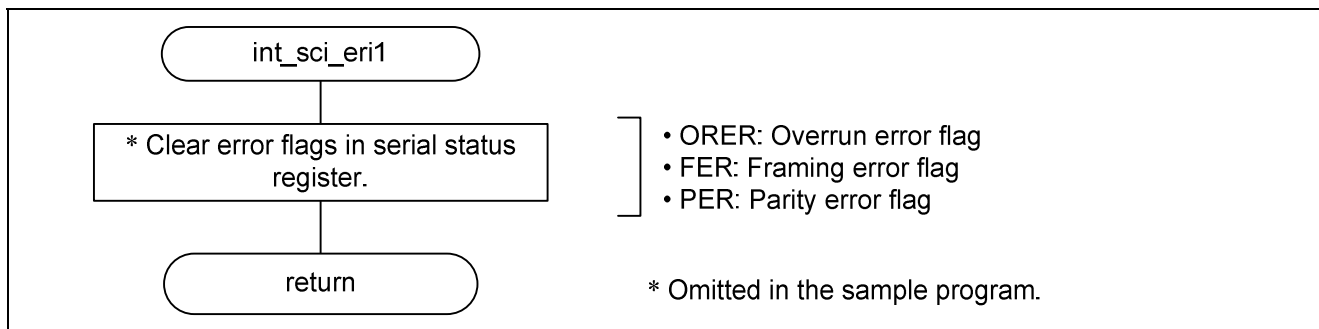
Figure 23   Receive Error Interrupt Handler

6.   Reference Documents

- Hardware Manual
  RX610 Group Hardware Manual
  (The latest version can be downloaded from the Renesas Electronics Web site.)

- Software Manual
  RX Family User's Manual: Software
  (The latest version can be downloaded from the Renesas Electronics Web site.)

- Development Environment Manual
  RX Family C/C++ Compiler Package User's Manual
  (The latest version can be downloaded from the Renesas Electronics Web site.)

- Technical Updates
  (The latest information can be accessed at 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.

Revision Record

Rev.	Date	Description		
		Page	**Summary**	
1.00	Oct.28.11	—	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 part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

RENESAS

Renesas Electronics Corporation

http://www.renesas.com

SALES OFFICES