RX Family, M16C Family

Migrating From the M16C Family to the RX Family: DMAC and DTC

R01AN2099EJ0100
Rev. 1.00
Sep. 22, 2014

Abstract

This document describes migrating the DMAC in the M16C Family to the DMAC and DTC in the RX Family.

Products

- RX Family
- M16C Family

As an example of migrating from the M16C to the RX, the explanation in this document uses the RX210 Group in the RX Family and the M16C/65C Group in the M16C Family. When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Differences in Terminology

Item	RX Family		M16C Family
	DMAC	DTC	DMAC
Mode for transferring specific data once	Normal transfer mode	Normal transfer mode	Single transfer mode
Mode for transferring specific data multiple times	Repeat transfer mode	Repeat transfer mode	Repeat transfer
Peripheral function registers	I/O registers		Special function registers (SFRs)

Contents

1. Comparison of the DMAC and DTC in the RX Family to the DMAC in the M16C Family

Table 1.1 lists a Comparison of the DMAC in the RX Family and M16C Family, and Table 1.2 lists a Comparison of the DTC in the RX Family and the DMAC in the M16C Family.
When transfer speed is given priority to transfer data, select the DMAC; when multiple request sources are selected to transfer data, select the DTC.

- Each channel of DMAC has I/O registers for setting specifics like the transfer address and transfer mode. After a request source is generated, data transfer starts according to the I/O register setting values. Therefore, the processing time until transfer starts is shorter than that for the DTC.
- Information such as the transfer addresses and transfer mode for the DTC are set on the memory (ROM and RAM). After a request source is generated, the information is read from the memory, and then data transfer starts. Therefore, the processing time until transfer starts is longer than that for the DMAC. However, compared to the DMAC, more request sources and transfer methods can be selected, and there is no limit on the number of transfer channels.

Table 1.1 Comparison of the DMAC in the RX Family and M16C Family

Item	RX (RX210 Group)	M16C (M16C/65C Group)
Number of channels	4 channels	4 channels
Transfer space	0000 0000h to 0FFF FFFFh and F000 0000h to FFFF FFFFh	00000h to FFFFFh
Request sources	56 sources • INT pin (4 sources): Choose low, falling edge, rising edge, or both edges • Timers (15 sources) • Communications (30 sources) • A/D conversion, comparators (4 sources) • ELC (2 sources) • Software trigger	43 sources • INT pin (falling edge) (8 sources) • INT pin (both edges) (8 sources) • Timers (11 sources) • Communications (14 sources) • A/D conversion (1 source) • Software trigger
Transfer units	8, 16, or 32 bits	8 or 16 bits
Transfer direction	4 conditions selectable for transfer source and transfer destination • Address is fixed • Offset addition (can only be specified for DMAC0) • Address is incremented • Address is decremented	Fixed → fixed, fixed → forward, or forward → fixed
Transfer modes	• Normal transfer mode • Repeat transfer mode • Block transfer mode	• Single transfer mode • Repeat transfer mode
Interrupts	• Generated on completion of transferring data volume specified by the transfer counter (transfer end interrupt) • Generated when the repeat size of data transfer is completed or the extended repeat area overflows (transfer escape end interrupt)	When the DMAi transfer counter underflows

Table 1.2 Comparison of the DTC in the RX Family and the DMAC in the M16C Family

Item	DTC in the RX (RX210 Group)	DMAC in the M16C (M16C/65C Group)
Number of channels	There is no concept of channels in the DTC. Transfer corresponding to the interrupt source is possible.	4 channels
Transfer space	• Short-address mode: 0000 0000h to 007F FFFFh and FF80 0000h to FFFF FFFFh • Full-address mode: 0000 0000h to FFFF FFFFh	00000h to FFFFFh
Request sources	97 sources • INT pin (8 sources): Choose low, falling edge, rising edge, or both edges • Timers (48 sources) • Communications (30 sources) • A/D conversion, comparators (4 sources) • ELC (2 sources) • Software trigger	43 sources • INT pin (falling edge) (8 sources) • INT pin (both edges) (8 sources) • Timers (11 sources) • Communications (14 sources) • A/D conversion (1 source) • Software trigger
Transfer units	8, 16, or 32 bits	8 or 16 bits
Transfer direction	3 conditions selectable for transfer source and transfer destination • Address is fixed • Address is incremented • Address is decremented	Fixed → fixed, fixed → forward, or forward → fixed
Transfer modes	• Normal transfer mode • Repeat transfer mode • Block transfer mode • Chain transfer	• Single transfer mode • Repeat transfer mode
Interrupts	• DTC activation • After a single data transfer • After data transfer of a specified volume	When the DMAi transfer counter underflows

Table 1.3 Comparison of the Number of Cycles for the DMAC and DTC in the RX Family and the DMAC in the M16C Family (1)

Item	RX (RX210 Group)		M16C (M16C/65C Group)
	DMAC	DTC	DMAC
Number of cycles	6 cycles	18 cycles	7 cycles (includes 1 dummy cycle)

Note 1: This table assumes the following: 8-bit data transfer, transfer source is an I/O register (fixed), the transfer destination is the RAM (increment), the DTC is in full-address mode, and DTC vector table is the ROM, and ICLK = PCLK multiplied by 2.

2. Peripheral Functions Used

Table 2.1 lists the Peripheral Functions and Modes Used in the Example for Operating the DMAC and DTC.

Table 2.1 Peripheral Functions and Modes Used in the Example for Operating the DMAC and DTC

No.	Operation	RX Family		M16C Family		Ref.
		Peripheral function	Mode	Peripheral function	Mode	
1	Data transmission	DMACA	Normal transfer mode	DMAC	Single transfer mode	4.1
2	Data repeatedly transmitted		Repeat transfer mode		Repeat transfer mode	4.2
3	Data transmission	DTCa	Normal transfer mode		Single transfer mode	5.2
4	Data repeatedly transmitted		Normal transfer mode		Repeat transfer mode	5.3

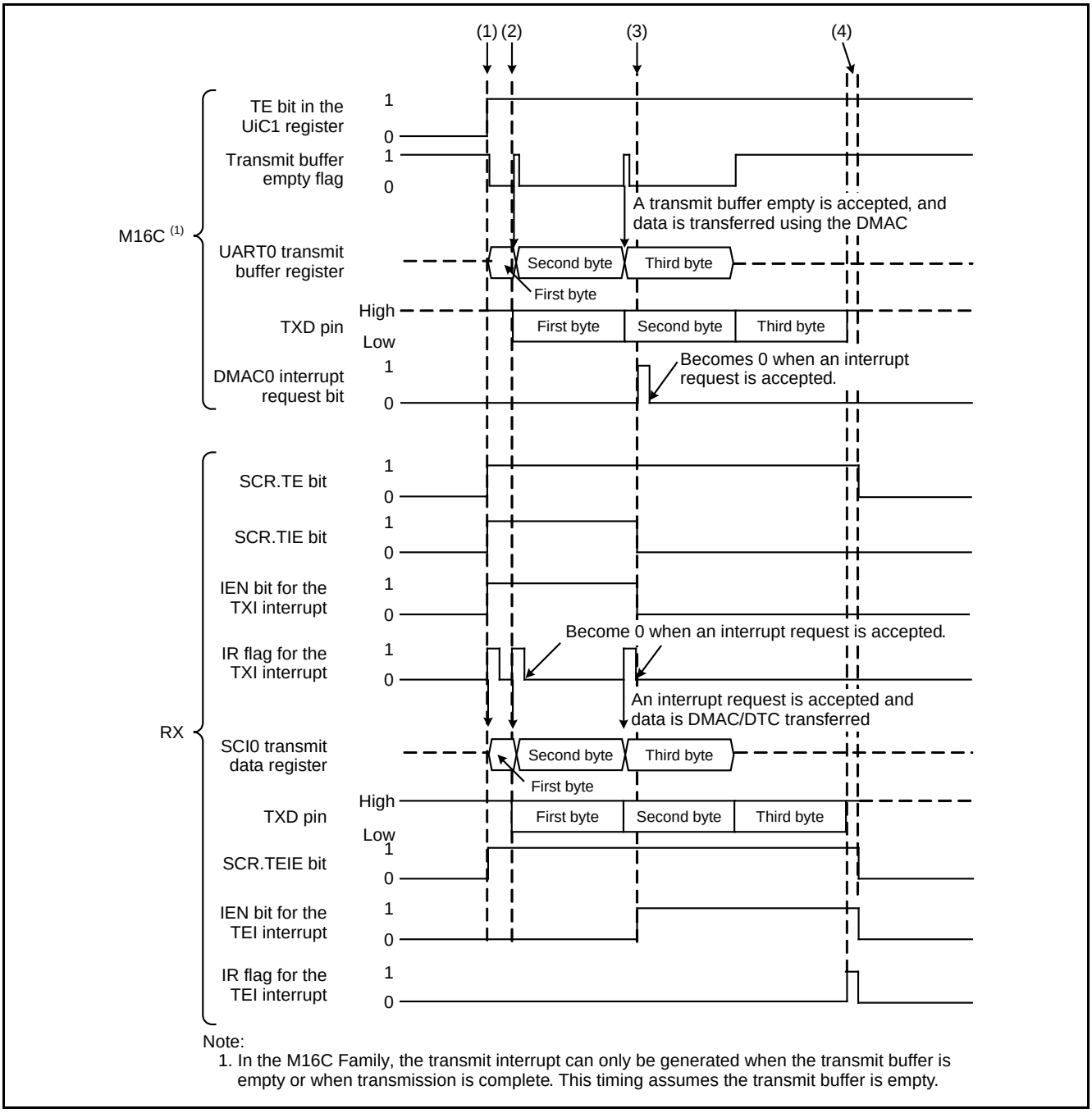
Also, asynchronous serial communications in the serial communications interface is used for the DMAC/DTC request source. The conditions listed in Table 2.2 are used to explain the differences in functions and setting procedures between the RX Family's DMAC and DTC to the M16C Family's DMAC.

Table 2.2 Conditions for DMAC and DTC Transfer Using Asynchronous Serial Communications

Item	Transfer Condition
Request source	Serial communications interface (SCI) Transmit data empty of the asynchronous serial communications
Channel used for asynchronous serial communications	RX Family: SCI0 M16C Family: UART0

3. Transfer Timing

Figure 3.1 shows a Comparison of Data Transfer Timing, and Table 3.1 lists a Comparison of Operation and Processing at Various Timings.
Both Figure 3.1 and Table 3.1 assume 3 bytes of serial data are transferred using the DMAC or DTC.



Note:
1. In the M16C Family, the transmit interrupt can only be generated when the transmit buffer is empty or when transmission is complete. This timing assumes the transmit buffer is empty.

Figure 3.1 Comparison of Data Transfer Timing

Table 3.1 Comparison of Operation and Processing at Various Timings

Timing	RX (RX210 Group)	M16C (M16C/65C Group)
(1) Transmission starts	When transmission is enabled, the transmit interrupt (TXI interrupt) is generated. The interrupt request triggers the DMAC/DTC to transfer the first byte of data to the transmit buffer.	Transmission is enabled. However, even if transmission is enabled, a transmit interrupt is not generated, so an instruction like the MOV instruction must be used in the main processing in order to write the first byte of data to the transmit buffer.
(2) Transmit data is transferred to the transmit shift register	When the transmit interrupt request is generated, the DMAC/DTC transfers the second byte of data to the transmit buffer.	When a transmit interrupt request is generated, the DMAC transfers the second byte of data to the transmit buffer.
(3) DMA transfer interrupt (DMAC) or TXI interrupt (DTC) is generated when the last data is written	When the last data is written to the transmit buffer, the DMA transfer interrupt (DMAC) or TXI interrupt (DTC) is generated. This interrupt enables the transmit end interrupt (TEI interrupt) and disables the transmit interrupt.	DMAC interrupt request is generated.
(4) After the last data is output	The transmit end interrupt is generated. In the interrupt handling, the transmit end interrupt and transmission are disabled. When transmission is disabled, the TXD pin becomes high-impedance.	N/A (no processing)

4. Comparison of the Functions and Setting Procedure for the DMAC in the RX Family and M16C Family

This chapter compares the RX210 Group and M16C/65C Group when using the DMAC to transfer data.

4.1 Comparison of the Setting Procedure When Transmitting 8 Bytes of Data

Table 4.1 lists a Comparison of the Initial Settings For Data Transfer When Transmitting 8 Bytes of Data, Table 4.2 lists the Comparison of Interrupt Handling When DMAC Transfer is Completed (When Transmitting 8 Bytes of Data), and Table 4.3 lists the Comparison of Transmit End Interrupt Handling (When Transmitting 8 Bytes of Data).

Table 4.1 Comparison of the Initial Settings For Data Transfer When Transmitting 8 Bytes of Data

Step		RX (RX210 Group)	M16C (M16C/65C Group)
1	Exit the module stop state (1)	SYSTEM.PRCR.WORD = 0xA502; MSTP(DMAC) = 0; MSTP(SCI0) = 0; SYSTEM.PRCR.WORD = 0xA500;	N/A (no module stop function)
2	Set the serial communications (2)	Set the serial communications interface (SCI0)	Set serial interface UART0
3	Disable DMAC transfer	IEN(DMAC,DMAC0I) = 0; DMAC0.DMCNT.BYTE = 0x00;	N/A (no processing)
4	Select the request source	ICU.DMRSR0 = VECT_SCI0_TXI0;	dm0sl = 0x0A;
5	Select the transfer mode, transfer size, and transfer address direction	DMAC0.DMAMD.WORD = 0x8000; DMAC0.DMTMD.WORD = 0x0001; DMAC0.DMCSL.BYTE = 0x00;	dm0con = 0x11;
6	Set the start address of the transfer source data	DMAC0.DMSAR =(void *)&snd_data;	sar0 = (unsigned short)(&snd_data[1]); sar0h = (unsigned char)(((unsigned long)(&snd_data[1])) >> 16);
7	Set the address of the transfer destination data	DMAC0.DMDAR = (void *)&SCI0.TDR;	dar0 = (unsigned short)(&u0tb); dar0h = (unsigned char)(((unsigned long)(&u0tb)) >> 16);
8	Set the transfer count	DMAC0.DMCRA = C_SEND_CNT;	tcr0 = C_SEND_CNT - 2;
9	Set the transfer end interrupt	IPR(DMAC,DMAC0I) = 1; DMAC0.DMINT.BYTE = 0x10; IEN(DMAC,DMAC0I) = 1;	dm0ic = 0x01;
10	Enable data transfer	DMAC0.DMCNT.BYTE = 0x01;	N/A (no processing)
11	Enable interrupts	setpsw_i();	asm("FSET I");
12	Enable data transfer	N/A (no processing)	dm0con |= 0x08;
13	Start the request source (serial communications) operation	SCI0.SCR.BYTE |= 0xF4; PORT2.PMR.BIT.B0 = 1; IEN(SCI0,TXI0) = 1;	te_u0c1 = 1; u0tb = snd_data[0];

Note 1. Refer to section 6.1.3 for details on the module stop function.
Note 2. The method for enabling the interrupt request differs. Refer to section 6.1.1 for details.

Table 4.2 Comparison of Interrupt Handling When DMAC Transfer is Completed (When Transmitting 8 Bytes of Data)

Step		RX (RX210 Group)	M16C (M16C/65C Group)
1	Stop the module of the request source	Disable the TXI interrupt Enable the TEI interrupt	N/A (no processing) Note that the transmit interrupt does not need to be disabled in the M16C Family.

Table 4.3 Comparison of Transmit End Interrupt Handling (When Transmitting 8 Bytes of Data)

Step		RX (RX210 Group)	M16C (M16C/65C Group)
1	Disable serial transmission	Disable serial transmission	N/A (no processing) Note that when selecting transmit buffer empty as the transmit interrupt source, an interrupt is not generated when transmission is completed.

4.2 Comparison of the Setting Procedure When Repeatedly Transmitting 8 Bytes of Data

Table 4.4 lists a Comparison of the Initial Settings For Data Transfer When Repeatedly Transmitting 8 Bytes of Data, Table 4.5 lists the Comparison of Interrupt Handling When DMAC Transfer is Completed (When Repeatedly Transmitting 8 Bytes of Data), and Table 4.6 lists the Comparison of Transmit End Interrupt Handling (When Repeatedly Transmitting 8 Bytes of Data).

Table 4.4 Comparison of the Initial Settings For Data Transfer When Repeatedly Transmitting 8 Bytes of Data

Step		RX (RX210 Group)	M16C (M16C/65C Group)
1	Exit the module stop state (1)	SYSTEM.PRCR.WORD = 0xA502; MSTP(DMAC) = 0; MSTP(SCI0) = 0; SYSTEM.PRCR.WORD = 0xA500;	N/A (no module stop function)
2	Set the serial communications (2)	Set the serial communications interface (SCI0)	Set serial interface UART0
3	Disable DMAC transfer	IEN(DMAC,DMAC0I) = 0; DMAC0.DMCNT.BYTE = 0x00;	N/A (no processing)
4	Select the request source	ICU.DMRSR0 = VECT_SCI0_TXI0;	dm0sl = 0x0A;
5	Select the transfer mode, transfer size, and transfer address direction	DMAC0.DMAMD.WORD = 0x8000; DMAC0.DMTMD.WORD = 0x5001; DMAC0.DMCSL.BYTE = 0x00;	dm0con = 0x13;
6	Set the start address of the transfer source data	DMAC0.DMSAR =(void *)&snd_data;	sar0 = (unsigned short)(&snd_data[1]); sar0h = (unsigned char)(((unsigned long)(&snd_data[1])) >> 16);
7	Set the address of the transfer destination data	DMAC0.DMDAR = (void *)&SCI0.TDR;	dar0 = (unsigned short)(&u0tb); dar0h = (unsigned char)(((unsigned long)(&u0tb)) >> 16);
8	Set the transfer count	DMAC0.DMCRA = (C_SEND_CNT << 16) | C_SEND_CNT;	tcr0 = C_SEND_CNT - 2;
9	Set the repeat transfer count	DMAC0.DMCRB = 2;	N/A (no processing)
10	Set the transfer end interrupt	IPR(DMAC,DMAC0I) = 1; DMAC0.DMINT.BYTE = 0x10; IEN(DMAC,DMAC0I) = 1;	dm0ic = 0x01;
11	Enable data transfer	DMAC0.DMCNT.BYTE = 0x01;	N/A (no processing)
12	Enable interrupts	setpsw_i();	asm("FSET I");
13	Enable data transfer	N/A (no processing)	dm0con |= 0x08;
14	Start the request source (serial communications) operation	SCI0.SCR.BYTE |= 0xF4; PORT2.PMR.BIT.B0 = 1; IEN(SCI0,TXI0) = 1;	te_u0c1 = 1; u0tb = snd_data[0];

Note 1. Refer to section 6.1.3 for details on the module stop function.
Note 2. The method for enabling the interrupt request differs. Refer to section 6.1.1 for details.

Table 4.5 Comparison of Interrupt Handling When DMAC Transfer is Completed (When Repeatedly Transmitting 8 Bytes of Data)

Step		RX (RX210 Group)	M16C (M16C/65C Group)
1	Set the start address of the transfer source data (data for the second and subsequent transfers)	N/A (no processing)	sar0 = (unsigned int)(&snd_data[0]); sar0h = (unsigned char)(((unsigned long)(&snd_data[0])) >> 16); (1)
2	Set the transfer count (for the second and subsequent transfers)		tcr0 = C_SEND_CNT - 1; (1)
3	Stop the module of the request source	Disable the TXI interrupt Enable the TEI interrupt	dm0con &= 0xF7; (When the number of repeat transfers is completed, the program stops the transfer of data.) (2)

Note 1. The first data transferred in the second and subsequent transfers is not written by the program, but is transferred by the DMAC. Therefore, before performing the second transfer, set the start address of the transfer source data and the transfer counter to the start of the transfer data table.

Note 2. The transmit interrupt does not need to be disabled in the M16C Family.

Table 4.6 Comparison of Transmit End Interrupt Handling (When Repeatedly Transmitting 8 Bytes of Data)

Step		RX (RX210 Group)	M16C (M16C/65C Group)
1	Disable serial transmission	Disable serial transmission (if DMCRB is 00h, then the transfer of data is stopped)	N/A (no processing) Note that when selecting transmit buffer empty as the transmit interrupt source, an interrupt is not generated when transmission is completed.

5. Comparison of the Functions and Setting Procedure for the DTC in the RX Family and the DMAC in the M16C Family

This chapter compares the transfer of data between the DTC in the RX210 Family and the DMAC in the M16C Family.

5.1 Settings to Use the DTC

When using the DTC, the DTC vector table and DTC transfer information must be prepared. Figure 5.1 shows a memory map when the DTC vector table is allocated to the ROM area.

The start address of the DTC transfer information is stored in the DTC vector table, and the DTC vector table is allocated to the ROM or RAM. The base address of the vector table is allocated so the lower 12 bits are 0. The start address of the DTC transfer information for interrupt vector number *n* that is set as a trigger source is stored in address +4*n* from the base address.

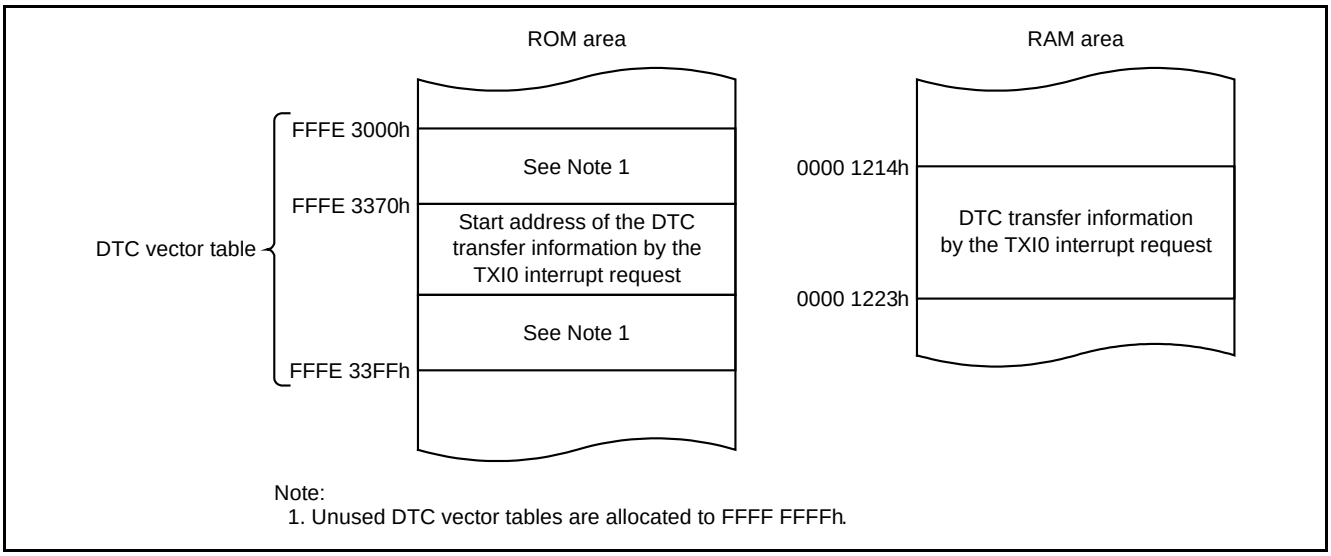
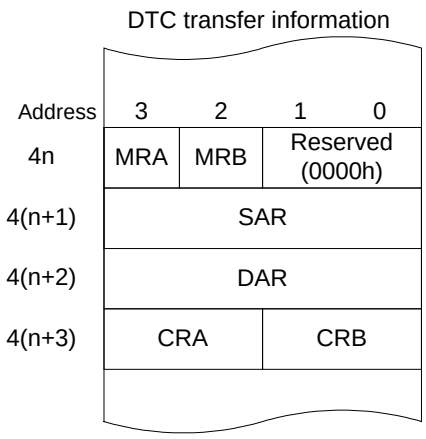


Figure 5.1 Memory Map When the DTC Vector Table is Allocated to the ROM Area

The DTC transfer information is allocated to the RAM, and Figure 5.2 shows a memory map and structure of the DTC transfer information.




```
/* **** DTC Transfer information data **** */
#pragma bit_order left
#pragma unpack
struct st_dtc_full{
        union{
                unsigned long LONG;
                struct{
                        unsigned long MRA_MD :2;
                        unsigned long MRA_SZ :2;
                        unsigned long MRA_SM :2;
                        unsigned long :2;
                        unsigned long MRB_CHNE :1;
                        unsigned long MRB_CHNS :1;
                        unsigned long MRB_DISEL :1;
                        unsigned long MRB_DTS :1;
                        unsigned long MRB_DM :2;
                        unsigned long :2;
                        unsigned long :16;
                }BIT;
        }MR;
        void * SAR;
        void * DAR;
        struct{
                unsigned long CRA:16;
                unsigned long CRB:16;
        }CR;
};
#pragma bit_order
#pragma packoption
```

Figure 5.2 Memory Map and Structure of the DTC Transfer Information

5.2 Comparison of the Setting Procedure When Transmitting 8 Bytes of Data

Table 5.1 lists a Comparison of the Initial Settings For Data Transfer When Transmitting 8 Bytes of Data.

Table 5.1 Comparison of the Initial Settings For Data Transfer When Transmitting 8 Bytes of Data

Step		RX (RX210 Group)	M16C (M16C/65C Group)
1	Exit the module stop state (1)	SYSTEM.PRCR.WORD = 0xA502; MSTP(DTC) = 0; MSTP(SCI0) = 0; SYSTEM.PRCR.WORD = 0xA500;	N/A (no module stop function)
2	Set the serial communications (2)	Set the serial communications interface (SCI0)	Set serial interface UART0
3	Disable transfer	DTC.DTCST.BYTE = 0x00; DTC.DTCCR.BIT.RRS = 0;	N/A (no processing)

4	Select the transfer mode, transfer size, and transfer address direction	DTC.DTCADMOD.BYTE = 0x00;
		dtc_info_txi0.MR.LONG = 0x00000000;
		dtc_info_txi0.MR.BIT.MRA_MD = 0;
		dtc_info_txi0.MR.BIT.MRA_SZ = 0;
		dtc_info_txi0.MR.BIT.MRA_SM = 2;
		dtc_info_txi0.MR.BIT.MRB_CHNE = 0;
		dtc_info_txi0.MR.BIT.MRB_DISEL = 0;
		dtc_info_txi0.MR.BIT.MRB_DM = 0;	

5	Select the request source	N/A (no processing)	dm0sl = 0x0A;
6	Select the transfer mode, transfer size, and transfer address direction		dm0con = 0x11;
7	Set the start address of the transfer source data	dtc_info_txi0.SAR = (void *)snd_data;	sar0 = (unsigned short)(&snd_data[1]); sar0h = (unsigned char)(((unsigned long)(&snd_data[1])) >> 16);
8	Set the address of the transfer destination data	dtc_info_txi0.DAR = (void *)&SCI0.TDR;	dar0 = (unsigned short)(&u0tb); dar0h = (unsigned char)(((unsigned long)(&u0tb)) >> 16);
9	Set the transfer count	dtc_info_txi0.CR.CRA = C_SEND_CNT; dtc_info_txi0.CR.CRB = 0x0000;	tcr0 = C_SEND_CNT - 2;
10	Select the request source	dtc_vect_table[216] = (void *)&dtc_info_txi0; DTC.DTCVBR = (void *)dtc_vect_table; DTC.DTCCR.BIT.RRS = 1;	N/A (no processing)
11	Set the transfer end interrupt	DTCE(SCI0,TXI0) = 1;	dm0ic = 0x01;
12	Enable interrupts	setpsw_i();	asm("FSET I");
13	Enable data transfer	DTC.DTCST.BYTE = 0x01;	dm0con |= 0x08;
14	Start the request source (serial communications) operation	SCI0.SCR.BYTE |= 0xF4; PORT2.PMR.BIT.B0 = 1; IEN(SCI0,TXI0) = 1;	te_u0c1 = 1; u0tb = snd_data[0];

Note 1. Refer to section 6.1.3 for details on the module stop function.

Note 2. The method for enabling the interrupt request differs. Refer to section 6.1.1 for details.

Table 5.2 lists a Comparison of DTC Interrupt Handling (Transmit Data Empty Interrupt) and DMAC Transfer Interrupt Handling (When Transmitting 8 Bytes of Data), and Table 5.3 lists a Comparison of Transmit End Interrupt Handling (When Transmitting 8 Bytes of Data).

Table 5.2 Comparison of DTC Interrupt Handling (Transmit Data Empty Interrupt) and DMAC Transfer Interrupt Handling (When Transmitting 8 Bytes of Data)

Step		RX (RX210 Group)	M16C (M16C/65C Group)
1	Stop the peripheral module of the request source	Disable the TXI interrupt Enable the TEI interrupt	N/A (no processing) Note that the transmit interrupt does not need to be disabled in the M16C Family.

Table 5.3 Comparison of Transmit End Interrupt Handling (When Transmitting 8 Bytes of Data)

Step		RX (RX210 Group)	M16C (M16C/65C Group)
1	Disable serial transmission	Disable serial transmission	N/A (no processing) Note that when selecting transmit buffer empty as the transmit interrupt source, an interrupt is not generated when transmission is completed.

5.3 Comparison of Setting Procedures When Repeatedly Transmitting 8 Bytes of Data

Table 5.4 lists a Comparison of the Initial Settings For Data Transfer When Repeatedly Transmitting 8 Bytes of Data.

Table 5.4 Comparison of the Initial Settings For Data Transfer When Repeatedly Transmitting 8 Bytes of Data

Step		RX (RX210 Group)	M16C (M16C/65C Group)
1	Exit the module stop state (1)	SYSTEM.PRCR.WORD = 0xA502; MSTP(DTC) = 0; MSTP(SCI0) = 0; SYSTEM.PRCR.WORD = 0xA500;	N/A (no module stop function)
2	Set the serial communications (2)	Set the serial communications interface (SCI0)	Set serial interface UART0
3	Disable transfer	DTC.DTCST.BYTE = 0x00; DTC.DTCCR.BIT.RRS = 0;	N/A (no processing)

4	Select the transfer mode, transfer size, and transfer address direction	DTC.DTCADMOD.BYTE = 0x00;
		dtc_info_txi0.MR.LONG = 0x00000000;
		dtc_info_txi0.MR.BIT.MRA_MD = 0;
		dtc_info_txi0.MR.BIT.MRA_SZ = 0;
		dtc_info_txi0.MR.BIT.MRA_SM = 2;
		dtc_info_txi0.MR.BIT.MRB_CHNE = 0;
		dtc_info_txi0.MR.BIT.MRB_DISEL = 0;
		dtc_info_txi0.MR.BIT.MRB_DM = 0;	

5	Select the request source	N/A (no processing)	dm0sl = 0x0A;
6	Select the transfer mode, transfer size, and transfer address direction		dm0con = 0x11;
7	Set the start address of the transfer source data	dtc_info_txi0.SAR = (void *)snd_data;	sar0 = (unsigned short)(&snd_data[1]); sar0h = (unsigned char)(((unsigned long)(&snd_data[1])) >> 16);
8	Set the address of the transfer destination data	dtc_info_txi0.DAR = (void *)&SCI0.TDR;	dar0 = (unsigned short)(&u0tb); dar0h = (unsigned char)(((unsigned long)(&u0tb)) >> 16);
9	Set the transfer count	dtc_info_txi0.CR.CRA = C_SEND_CNT; dtc_info_txi0.CR.CRB = 0x0000;	tcr0 = C_SEND_CNT - 2;
10	Select the request source	dtc_vect_table[216] = (void *)&dtc_info_txi0; DTC.DTCVBR = (void *)dtc_vect_table; DTC.DTCCR.BIT.RRS = 1;	N/A (no processing)
11	Set the transfer end interrupt	DTCE(SCI0,TXI0) = 1;	dm0ic = 0x01;
12	Enable interrupts	setpsw_i();	asm("FSET I");
13	Enable data transfer	N/A (no processing)	dm0con |= 0x08;
14	Start the request source (serial communications) operation	DTC.DTCST.BYTE = 0x01; SCI0.SCR.BYTE |= 0xF4; PORT2.PMR.BIT.B0 = 1; IEN(SCI0,TXI0) = 1;	te_u0c1 = 1; u0tb = snd_data[0];

Note 1. Refer to section 6.1.3 for details on the module stop function.
Note 2. The method for enabling the interrupt request differs. Refer to section 6.1.1 for details.

Table 5.5 lists a Comparison of DTC Interrupt Handling (Transmit Data Empty Interrupt) and DMAC Transfer Interrupt Handling (When Repeatedly Transmitting 8 Bytes of Data), and Table 5.6 lists a Comparison of Transmit End Interrupt Handling (When Repeatedly Transmitting 8 Bytes of Data).

Table 5.5 Comparison of DTC Interrupt Handling (Transmit Data Empty Interrupt) and DMAC Transfer Interrupt Handling (When Repeatedly Transmitting 8 Bytes of Data)

Step		RX (RX210 Group)	M16C (M16C/65C Group)
1	Disable transfer	DTC.DTCST.BYTE = 0x00; DTC.DTCCR.BIT.RRS = 0;	N/A (no processing)
2	Set the start address of the transfer source data (data for the second and subsequent transfers)	dtc_info_txi0.SAR = (void *)snd_data;	sar0 = (unsigned int)(&snd_data[0]); sar0h = (unsigned char)(((unsigned long)(&snd_data[0])) >> 16);
3	Set the transfer count (for the second and subsequent transfers)	dtc_info_txi0.CR.CRA = C_SEND_CNT; dtc_info_txi0.CR.CRB = 0x0000;	tcr0 = C_SEND_CNT - 1;
4	Set the request source	DTC.DTCCR.BIT.RRS = 1 DTCE(SCI0,TXI0) = 1;	N/A (no processing)
5	Start transfer	DTC.DTCST.BYTE = 0x01;	
6	Stop the module of the request source	Disable the TXI interrupt Enable the TEI interrupt (When the number of repeat transfers is completed, the transfer of data is stopped.)	dm0con &= 0xF7; (When the number of repeat transfers is completed, the program stops the transfer of data.) (1)

Note 1. The transmit interrupt does not need to be disabled in the M16C Family.

Table 5.6 Comparison of Transmit End Interrupt Handling (When Repeatedly Transmitting 8 Bytes of Data)

Step		RX (RX210 Group)	M16C (M16C/65C Group)
1	Disable serial transmission	Disable serial transmission (if DMCRB is 00h, then the transfer of data is stopped)	N/A (no processing) Note that when selecting transmit buffer empty as the transmit interrupt source, an interrupt is not generated when transmission is completed.

6. Appendix

6.1 Points on Migrating From the M16C Family to the RX Family

This chapter explains points on migrating from the M16C Family to the RX Family.

6.1.1 Interrupts

For the RX Family, interrupt requests can be accepted while all of the following conditions are met.

- The I flag (PSW.I bit) is 1.
- Registers IER and IPR in the ICU are set to interrupt enabled.
- The interrupt request is enabled in the interrupt request enable bit for peripheral functions.

Table 6.1 lists a Comparison of Conditions for Interrupt Generation.

Table 6.1 Comparison of Conditions for Interrupt Generation

Item	RX Family	M16C Family
I flag	When the I flag is set to 1 (enabled), the maskable interrupt request can be accepted.	
Interrupt request flag	When an interrupt request is generated by a peripheral function, the interrupt request flag becomes 1 (interrupt requested).	
Interrupt priority level	Selected by setting the IPR[3:0] bits.	Selected by setting bits ILVL2 to ILVL0.
Interrupt request enable	Specified by setting the IER register.	N/A
Interrupt enable for peripheral functions	Interrupts can be enabled or disabled in each peripheral function.	N/A

For more information, refer to sections Interrupt Controller (ICU), CPU, and sections for other peripheral functions used in the User's Manual: Hardware.

6.1.2 I/O Ports

In the RX Family, the MPC must be configured in order to assign I/O signals from peripheral functions to pins. Before controlling the I/O pins in the RX Family, the following two items must be set.

- In the MPC.PFS register, select the peripheral functions that are assigned to the pins.
- In the PORTn.PMR register, select the pin function from a general I/O port or peripheral function.

Table 6.2 lists a Comparison of I/O Settings for Peripheral Function Pins.

Table 6.2 Comparison of I/O Settings for Peripheral Function Pins

Function	RX (RX210 Group)	M16C (M16C/65C Group)
Select the pin function	With the PFS register, I/O ports for peripheral functions can be assigned by selecting from multiple pins.	These are not available in the M16C Family. *1 When a mode is set for a peripheral function, appropriate pins are assigned as I/O pins for the peripheral function.
Switch between general I/O port and peripheral function	With the PMR register, the corresponding pin function can be selected as a general I/O port or a peripheral function.	

Note 1. Register for similar functions are available in the M32C Group and R32C Group.

For more information, refer to the Multi-Function Pin Controller (MPC) and I/O port sections in the User's Manual: Hardware.

6.1.3 Module Stop Function

The RX Family has the ability to stop each peripheral module individually. By transitioning unused peripheral modules to the module stop state, power consumption can be reduced. After a reset is released, all modules (with a few exceptions) are in the module stop state. Registers for modules in the module stop state cannot be written to or read. For more information, refer to the Low Power Consumption section in the User's Manual: Hardware.

6.2 I/O Register Macros

Macro definitions listed in Table 6.3 can be found in the RX I/O register definitions (iodefine.h).
The readability of programs can be achieved with these macro definitions.

Table 6.3 lists examples of macros.

Table 6.3 Using Macros

Macro	Usage Example
IR("module name", "bit name")	**IR(MTU0,TGIA0)** = 0 ; The IR bit corresponding to MTU0.TGIA0 is cleared to 0 (no interrupt request is generated).
DTCE("module name", "bit name")	**DTCE (MTU0, TGIA0)** = 1 ; The DTCE bit corresponding to MTU0.TGIA0 is set to 1 (DTC activation is enabled).
IEN("module name", "bit name")	**IEN(MTU0, TGIA0)** = 1 ; The IEN bit corresponding to MTU0.TGIA0 is set to 1 (interrupt request enabled).
IPR("module name", "bit name")	**IPR(MTU0, TGIA0)** = 0x02 ; The IPR[3:0] bits corresponding to MTU0.TGIA0 are set to 0010b (interrupt priority level 2).
MSTP("module name")	**MSTP(MTU)** = 0 ; The MTU0 Module Stop bit is set to 0 (module stop state is canceled).
VECT("module name", "bit name")	#pragma interrupt (Excep_MTU0_TGIA0 (vect=**VECT(MTU0, TGIA0)**) The interrupt function is declared for the corresponding MTU0.TGIA0.

6.3 Intrinsic Functions

The RX Family has intrinsic functions for setting control registers and special instructions. When using intrinsic functions, include machine.h.

Table 6.4 lists a Comparison of the Descriptions of Special Instructions and Control Register Settings as an example.

Table 6.4 Comparison of the Descriptions of Special Instructions and Control Register Settings

Item	Description	
	RX Family	M16C Family
Set the I flag to 1	setpsw_i (); *1	asm("fset i");
Set the I flag to 0	clrpsw_i (); *1	asm("fclr i");
Expanded into the WAIT instruction	wait(); *1	asm("wait");
Expanded into the NOP instruction	nop(); *1	asm("nop");

Note 1. "machine.h" must be included.

7. Reference Documents

User’s Manual: Hardware

RX210 Group User’s Manual: Hardware Rev.1.50 (R01UH0037EJ)
M16C/65C Group User’s Manual: Hardware Rev.1.10 (R01UH0093)
Refer to the corresponding User's Manual: Hardware when using products other than the RX210 Group and M16C/65C Group.
The latest versions can be downloaded from the Renesas Electronics website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

User’s Manual: Development Tools

RX Family C/C++ Compiler Package V.1.01 User's Manual Rev.1.00 (R20UT0570EJ)
M16C Series, R8C Family C Compiler Package V5.45
C Compiler User's Manual Rev.3.00
The latest versions 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	RX Family, M16C Family Application Note Migrating From the M16C Family to the RX Family: DMAC and DTC

Rev.	Date	Description	
		Page	Summary
1.00	Sep. 22, 2014	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

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

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

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

Notice

1. Descriptions of circuits, software and 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.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product 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; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of 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 systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. 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.

RENESAS

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

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

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, 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: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

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

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

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.0