

RX Family, M16C Family

R01AN1927EJ0100

Rev. 1.00

July 1, 2014

Migrating from the M16C Family to the RX Family: Clock Synchronous Serial Data Communications

Abstract

This document describes migrating from the clock synchronous serial I/O mode for the serial I/O in the M16C Family to the clock synchronous mode for the SCI 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
Serial communication interface (SCI)	SCI	Serial I/O
Clock synchronous serial data communications mode	Clock synchronous mode	Clock synchronous serial I/O mode
Clock I/O pin for clock synchronous communications (SCK pin)	SCK pin	CLK pin
SCI operating clock (clock source)	Clock source	Count source
Peripheral function operating clock	Peripheral module clocks: PCLKA, PCLKB, PCLKC, PCLKD	Peripheral function clocks: f1, fOCO40M, fOCO-F, fOCO-S, fC32
Transmit buffer	TDR register	UiTB register
Transmit shift register	TSR register	UART transmit shift register
Receive buffer	RDR register	UiRB register
Transmit interrupt	TXI interrupt	UARTi transmit interrupt (transmit buffer empty)
Transmission complete interrupt (M16C) Transmit end interrupt (RX)	TEI interrupt	UARTi transmit interrupt (transmission completed)
Reception complete interrupt (M16C) Receive end interrupt (RX)	RXI interrupt	UARTi receive interrupt
Function to select I/O of peripheral functions for pins	MPC ^{*1}	Function select register, input function select register ^{*2}

Note 1. The MPC is not available in some groups.

Note 2. Only available in the M32C Group and R32C Group.

Contents

1. Differences in Clock Synchronous Serial Data Communications.....	3
2. Peripheral Functions Used	4
3. Differences in Clock Synchronous Serial Data Communications.....	4
3.1 Differences in Master Transmission and Reception	5
3.1.1 Differences in Timing During Master Transmission and Reception	5
3.1.2 Differences in Setting Procedures During Master Transmission and Reception	7
3.2 Differences in Master Transmission.....	10
3.2.1 Differences in Timing During Master Transmission.....	10
3.2.2 Differences in Setting Procedures During Master Transmission.....	12
3.3 Differences in Slave Reception.....	14
3.3.1 Differences in Timing During Slave Reception.....	15
3.3.2 Differences in Setting Procedures During Slave Reception	17
3.4 Calculating the Bit Rate	19
4. Appendix.....	20
4.1 Points on Migrating from the M16C to the RX	20
4.1.1 Interrupts.....	20
4.1.2 I/O ports	20
4.1.3 Module Stop Function.....	20
4.2 I/O Register Macros	21
4.3 Intrinsic Functions	21
5. Reference Documents.....	22

1. Differences in Clock Synchronous Serial Data Communications

Table 1.1 lists the Differences in Clock Synchronous Serial Data Communications.

Table 1.1 Differences in Clock Synchronous Serial Data Communications

Item	RX (RX210)	M16C (M16C/65C)
Operation clock source	PCLKB	Selectable from f1, fOCO40M, fOCO-F, fOCO-S, or fC32
Data length	8 bits	8 bits
Data format	Selectable from LSB first or MSB first	Selectable from LSB first or MSB first
Hardware flow control	Available (selectable)	Available (selectable)
Separate CTS/RTS pins	Not available	Available (UART0)
Interrupt sources	Transmit data empty (TXI) interrupt Transmit end (TEI) interrupt Receive data full (RXI) interrupt Receive error (ERI) interrupt	Transmit interrupt Receive interrupt
Error detection	Overrun error	Overrun error
Selectable clock polarity	Available (selectable)	Available (selectable)
Transmit/receive clock multiple-pin output	Not available	Available (UART1)
Noise cancellation	The signal paths from input on the RXD pins incorporate digital noise filters	Not available
Data logic switch	Available	Available
TXD, RXD I/O polarity switch	Not available	Available

2. Peripheral Functions Used

Table 2.1 lists Peripheral Functions and Modes Used When Performing Clock Synchronous Serial Data Communications.

Table 2.1 Peripheral Functions and Modes Used When Performing Clock Synchronous Serial Data Communications

No.	Operating Example	RX		M16C		Reference
		Peripheral Function	Mode	Peripheral Function	Mode	
1	Clock synchronous serial data communications (master transmit and receive operations)	SCI	Clock synchronous mode	Serial I/O	Clock synchronous serial I/O mode	Section 3.1
2	Clock synchronous serial data communications (master transmit operation)					Section 3.2
3	Clock synchronous serial data communications (slave receive operation)					Section 3.3

3. Differences in Clock Synchronous Serial Data Communications

Differences in clock synchronous serial data communications listed in Table 3.1 Conditions for Clock Synchronous Serial Data Communications describe examples of conditions.

Table 3.1 Conditions for Clock Synchronous Serial Data Communications

Item	Conditions for Transmission and Reception
Peripheral function operating clock	16 MHz
Transfer rate	100 kbps
Data format	LSB first
Hardware flow control	None
Channels used	RX Family: SCI0 M16C Family: UART0

3.1 Differences in Master Transmission and Reception

This section describes the differences in master transmission and reception for clock synchronous operation.

3.1.1 Differences in Timing During Master Transmission and Reception

Figure 3.1 shows the Differences in Timing When Transmitting and Receiving 3 Bytes at a Time.

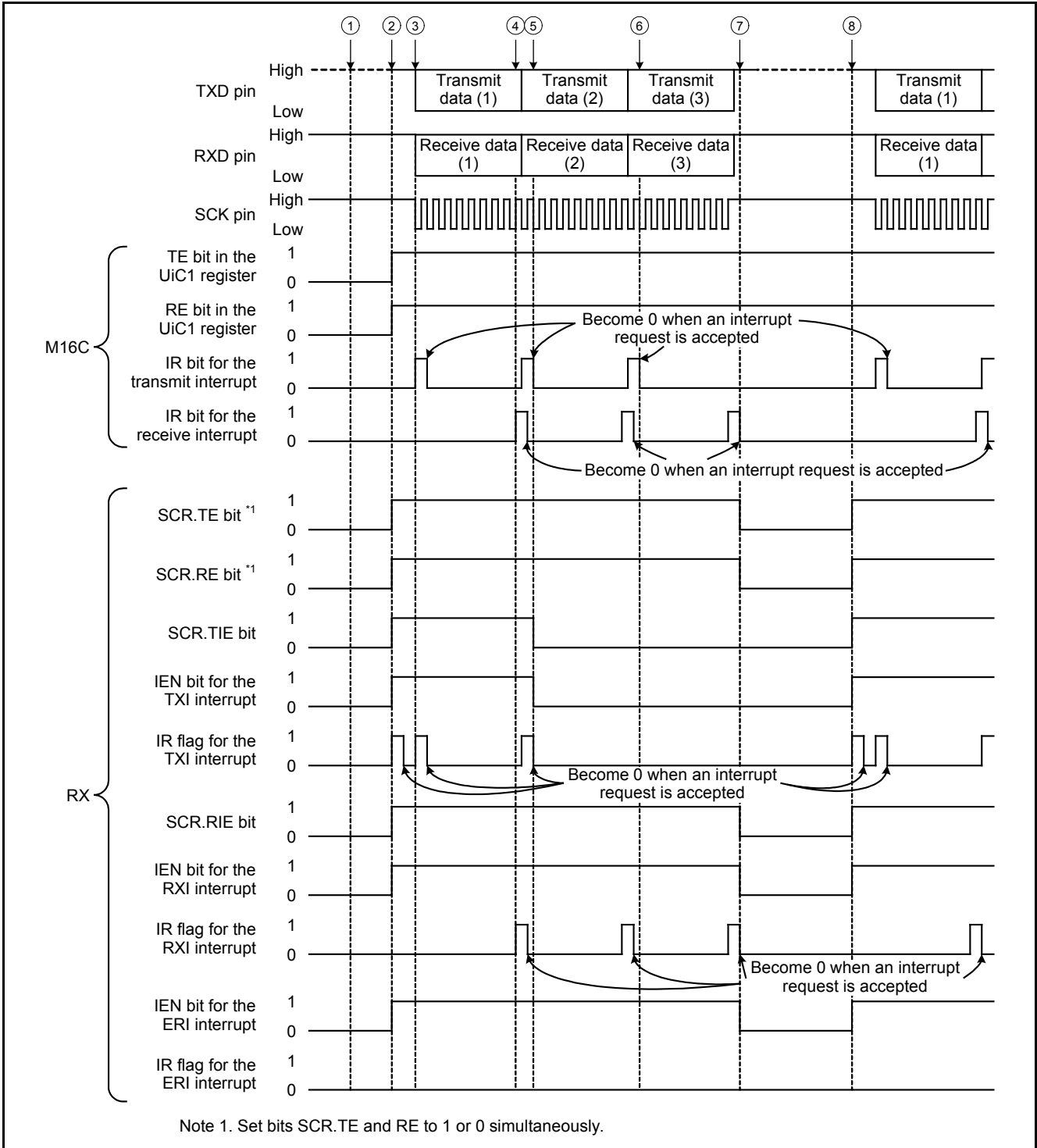


Figure 3.1 Differences in Timing When Transmitting and Receiving 3 Bytes at a Time

Refer to section 3.3.1 for differences in timing when a reception error occurs.

Table 3.2 lists Differences in Operation at Various Timing and Processing When Transmitting and Receiving 3 Bytes at a Time.

Table 3.2 Differences in Operation at Various Timing and Processing When Transmitting and Receiving 3 Bytes at a Time

Timing	RX (RX210)	M16C (M16C/65C)
① Before transmission starts	The TXD pin is in high-impedance until the SCR.TE bit is set to 1 (serial transmission is enabled).	The pin status is fixed when serial I/O mode is selected.
② When transmission starts	To enable transmission and reception, set the SCR.TE bit to 1, and set the SCR.RE bit to 1. To enable interrupts, set the SCR.TIE bit to 1, set the SCR.RIE bit to 1, set the IEN bit for the TXI interrupt to 1, set the IEN bit for the RXI interrupt to 1, and set the IEN bit for the ERI interrupt to 1. When the SCR.TE bit is set to 1, the IR flag for the transmit interrupt (TXI interrupt) becomes 1. Write the first byte of transmit data in the transmit interrupt handling.	Set the TE bit to 1 (transmission enabled). The transmit interrupt does not occur even if the TE bit is set to 1. The first byte of data is written in the main processing, etc.
③ When transmit data is transferred to the transmit shift register	The IR flag (IR bit) for the transmit interrupt becomes 1, and the transmit interrupt is generated. The second byte of data is written in the transmit interrupt handling.	
④ When transmission is complete	After receiving 1 byte of data, the data received is stored in the receive buffer, and the IR bit for the receive interrupt (RXI interrupt) becomes 1. The value stored in the receive buffer is read in the receive interrupt handling.	
⑤ Transmit interrupt when writing the last data	After writing the third byte of transmit data, set the SCR.TIE bit to 0 (a TXI interrupt request is disabled), and set the IEN bit for the TXI interrupt to 0 (TXI interrupt request is disabled).	N/A
⑥ Transmit interrupt after writing the last data	N/A (transmit interrupt is not generated)	Transmit data is not written, and the interrupt handling is ended.
⑦ Receive end interrupt of the last data	In the receive interrupt handling, after reading the receive data, set the SCR.TE bit to 0 (serial transmission is disabled), and the RE bit to 0 (serial reception is disabled) simultaneously. In addition, set the RIE bit to 0 (RXI interrupt request is disabled), set the IEN bit for the RXI interrupt to 0 (RXI interrupt request is disabled), and set the IEN bit for the ERI interrupt to 0 (ERI interrupt request is disabled). When transmission is disabled, the TXD pin becomes high-impedance.	The same processing as in ④ occurs.
⑧ When transmission and reception restart	The same processing as in ② occurs.	The next data is written in the main processing, etc.

3.1.2 Differences in Setting Procedures During Master Transmission and Reception

The tables in this section list the differences during transmission and reception.

Table 3.3 Differences in the Initial Setting Procedures When Transmitting and Receiving Data

Step		RX (RX210)	M16C (M16C/65C)
1	Exiting the module-stop state ^{*1}	SYSTEM.PRCR.WORD = 0xA502; MSTP(SCI0) = 0; SYSTEM.PRCR.WORD = 0xA500;	N/A (no module-stop function)
2	Setting the I/O port functions ^{*2}	PORT2.PMR.BIT.B0 = 0; PORT2.PMR.BIT.B1 = 0; PORT2.PMR.BIT.B2 = 0; MPC.PWPR.BIT.B0WI = 0; MPC.PWPR.BIT.PFSWE = 1; MPC.P20PFS.BYTE = 0x0A; MPC.P21PFS.BYTE = 0x0A; MPC.P22PFS.BYTE = 0x0A; MPC.PWPR.BYTE = 0x80; PORT2.PMR.BIT.B0 = 1; PORT2.PMR.BIT.B1 = 1; PORT2.PMR.BIT.B2 = 1;	N/A (no processing) ^{*3}
3	Setting the transmit/receive mode, etc.	SCI0.SCR.BIT.CKE = 0; SCI0.SIMR1.BYTE = 0x00; SCI0.SPMR.BYTE = 0x00; SCI0.SMR.BYTE = 0x80; SCI0.SCMR.BYTE = 0xF2;	uclksel0 = 0x00; pclk1 = 1; u0mr = 0x01; u0c0 = 0x10; ucon = 0x00 u0c1 = 0x00;
4	Setting the bit rate ^{*4}	SCI0.BRR = 39;	u0brg = 79;
5	Setting the interrupt priority level	IPR(SCI0,) = 0x01	
6	Clear the interrupt requests	IR(SCI0,RX10) = 0; IR(SCI0,TX10) = 0;	s0tic = 0x01; s0ric = 0x01;
7	Enable peripheral function interrupt requests ^{*5}	SCI0.SCR.BYTE = 0xF0; /* Note 6 */	
8	Enable transmission/reception		u0c1 = 0x05;
9	Enable interrupt requests ^{*5}	IEN(SCI0,ER10) = 1; /* Note 7 */ IEN(SCI0,RX10) = 1; IEN(SCI0,TX10) = 1;	N/A (no processing)
10	Enable maskable interrupts	setpsw_i();	asm("fset i");
11	Write the first byte of transmit data	N/A (no processing)	/* Write the first byte of data to the U0TB register */

Note 1. For details on the module stop function, refer to section 4.1.3 Module Stop Function.

Note 2. In the RX Family, pin settings for peripheral functions are configured in the MPC. Refer to section 4.1.2 I/O ports for details.

Note 3. In the M32C/80 Series and R32C/100 Series, select the pin function in the function select register.

Note 4. The bit rate calculation method differs between the RX Family and M16C Family. Refer to section 3.2 Calculating the Bit Rate for details.

Note 5. The method of enabling interrupt requests differs. Refer to 4.1.1 Interrupts for details.

Note 6. Set the SCR.TE bit to 1 (serial transmission is enabled) and the RE bit to 1 (serial reception is enabled) simultaneously.

Note 7. Specifications for the receive error interrupt differ according to the MCU used. Refer to the User's Manual: Hardware (UMH) for details.

Table 3.4 Differences in Transmit Interrupt Handling During Transmission and Reception

Step		RX (RX210)	M16C (M16C/65C)
1	Write transmit data	<code>/* Write transmit data to the SCI0.TDR register */</code>	<code>if (Transmit data present?)</code> <code>{</code>
2	Confirm last data was written	<code>if (Last data written?)</code> <code>{</code>	<code>/* Write transmit data to U0TB register */</code> <code>}</code>
3	Disable the TXI interrupt (only after last data is written)	<code>IEN(SCI0, TXI0) = 0;</code> <code>SCI0.SCR.BIT.TIE = 0;</code> <code>while (0 != SCI0.SCR.BIT.TIE)</code> <code>{</code> <code>}</code>	N/A (no processing) ^{*1}
4	Clear the interrupt request (only after the last data is written)	<code>IR(SCI0, TXI0) = 0;</code> <code>}</code>	

Table 3.5 Differences in Receive Interrupt Handling During Transmission and Reception

Step		RX (RX210)	M16C (M16C/65C)
1	Read receive data	<code>/* Read receive data from the SCI0.RDR register */</code>	<code>/* Read receive data from the U0RB register */</code>
2	Determine what error was received	N/A (no processing) ^{*1}	<code>if (Receive error occurred?)</code> <code>{</code>
3	Processing when a receive error occurs		<code>/* Describe processing when an error occurs */</code>
4	Clear the receive error (disable reception)		<code>re_u0c1 = 0;</code> <code>u0mr = u0mr & 0xF8</code> <code>}</code>
5	Confirm that the last data were received	<code>if (Last data received?)</code> <code>{</code>	N/A (no processing) ^{*2}
6	Disable transmission and reception, and disable the receive interrupt (only after receiving the last data)	<code>SCI0.SCR.BYTE =</code> <code>(SCI0.SCR.BYTE&0x0F);</code> <code>/* Note 3 */</code> <code>while (0x00 != (SCI0.SCR.BYTE &</code> <code>0xF0))</code> <code>{</code> <code>}</code>	
7	Clear the interrupt request (only after the last data is written)	<code>IR(SCI0, RXI0) = 0;</code> <code>}</code>	

Note 1. When a receive error occurs in the RX Family, the ERI interrupt is generated making error processing unnecessary for the RXI interrupt.

Note 2. In the M16C Family, the receive interrupt does not need to be disabled.

Note 3. Set the SCR.TE bit to 0 (serial transmission is disabled) and the RE bit to 0 (serial reception is disabled) simultaneously.

Table 3.6 Differences in ERI Interrupt Handling During Transmission and Reception

Step		RX (RX210)	M16C (M16C/65C)
1	Check the source of the generated interrupt request	do { while ((0 != SCIO.SCR.BIT.RIE) && (0x00 != (SCIO.SSR.BYTE & 0x38))) {	N/A *1
2	Processing when a receive error occurs	/* Describe processing when an error occurs */	
3	Dummy read the receive buffer	dummy = SCIO.RDR;	
4	Disable reception	SCIO.SCR.BYTE = (SCIO.SCR.BYTE & 0x0B); /* Note 2 */ while (0x00 != (SCIO.SCR.BYTE & 0xF4)) { }	
5	Clear the interrupt requests	IR(SCIO, TXIO) = 0; IR(SCIO, RXIO) = 0; { }	
6	Clear the reception error	dummy SCIO.SSR.BYTE; /* Note 3 */ SCIO.SSR.BYTE = 0xC0; while (0x00 != (SCIO.SSR.BYTE & 0x38)) { }	
7	Read the IR flag	} }while(0 != IR(SCIO, ERI0));	

Note 1. In the M16C Family, a separate interrupt is not generated when a receive error occurs. Error determination or processing is performed in the receive interrupt.

Note 2. Set the SCR.TE bit to 0 (serial transmission is disabled) and the RE bit to 0 (serial reception is disabled) simultaneously.

Note 3. When clearing the PER flag (parity error flag), FER flag (framing error flag), and ORER flag (overrun error flag), the flags must be read as 1 in advance.

3.2 Differences in Master Transmission

This section describes the differences in master transmission for clock synchronous operation.

3.2.1 Differences in Timing During Master Transmission

Figure 3.2 shows Differences in Timing When Transmitting 3 Bytes at a Time.

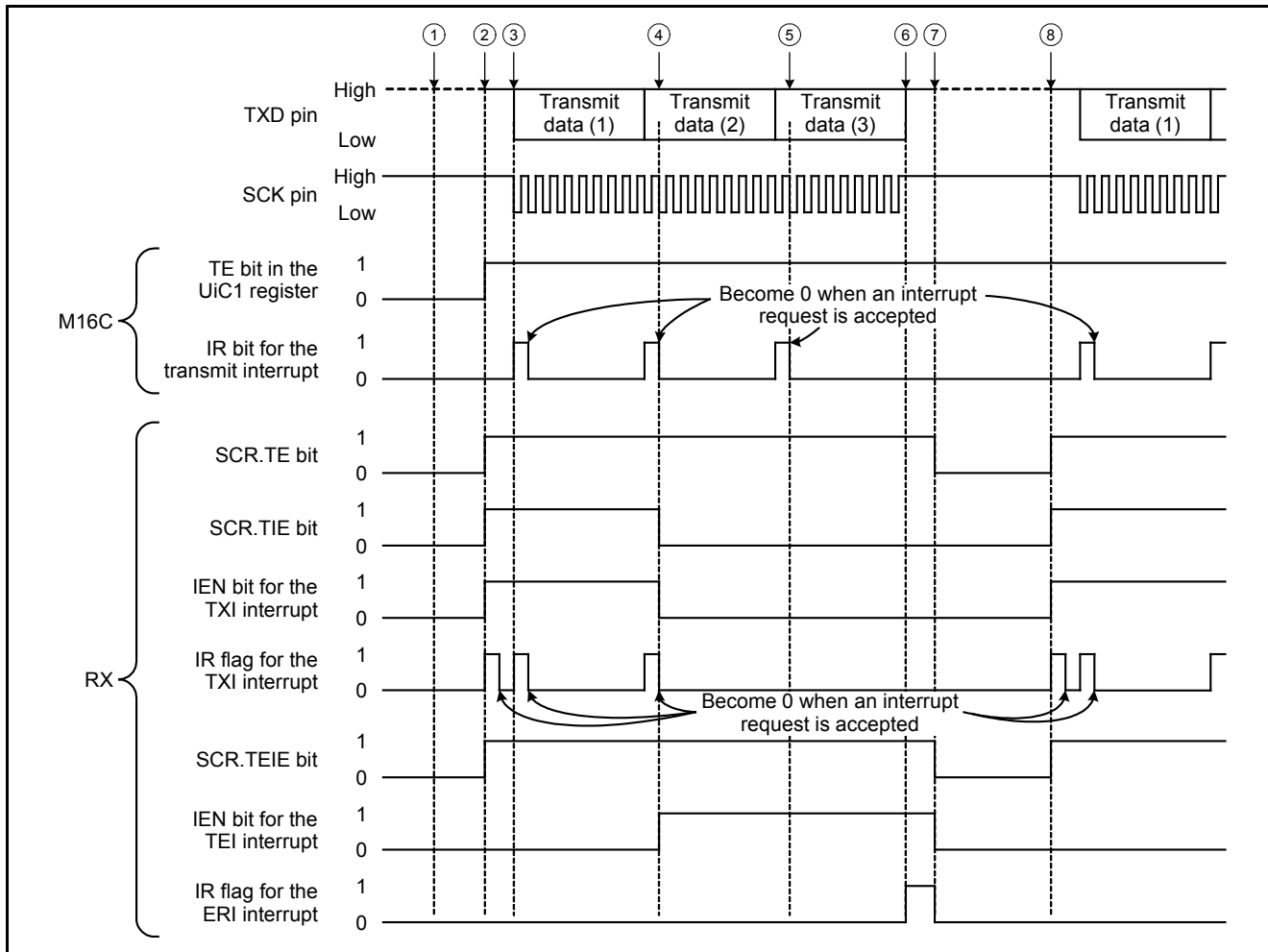


Figure 3.2 Differences in Timing When Transmitting 3 Bytes at a Time

Table 3.7 Differences in Operation at Various Timing and Processing When Transmitting 3 Bytes at a Time

Timing	RX (RX210)	M16C (M16C/65C)
① Before transmission starts	The TXD pin is in high-impedance until the SCR.TE bit is set to 1 (serial transmission is enabled).	The pin status is fixed when serial I/O mode is selected.
② When transmission starts	Set the SCR.TE bit to 1, set the TIE bit to 1 (a TXI interrupt request is enabled), set the TEIE bit to 1 (a TEI interrupt request is enabled), and set the IEN bit for the TXI interrupt to 1 (TXI interrupt request enabled). When the SCR.TE bit is set to 1, the IR flag for the transmit interrupt (TXI interrupt) becomes 1. Write the first byte of transmit data in the transmit interrupt handling.	Set the TE bit to 1 (transmission enabled). The transmit interrupt does not occur even if the TE bit is 1. The first byte of data is written in the main processing, etc.
③ When transmit data is transferred to the transmit shift register	The IR flag (IR bit) for the transmit interrupt becomes 1, and the transmit interrupt is generated. The second byte of data is written in the transmit interrupt handling.	
④ Transmit interrupt when writing the last data	Set the IEN bit for the TEI interrupt to 1 (TEI interrupt enabled), set the SCR.TIE bit to 0 (a TXI interrupt request is disabled), and set the IEN bit for the TXI interrupt to 0 (TXI interrupt disabled).	N/A
⑤ Transmit interrupt after writing the last data	N/A (The transmit interrupt is not generated.)	Interrupt handling is completed without transmit data being written.
⑥ When transmission is complete	The transmit end interrupt is generated.	
⑦ Transmission end interrupt handling	In the transmit end interrupt handling, set the SCR.TE bit to 0 (serial transmission is disabled), set the TEIE bit to 0 (a TEI interrupt request is disabled), and set the IEN bit for the transmit end interrupt to 0 (TEI interrupt disabled) to disable transmission. When transmission is disabled, the IR flag for the transmit end interrupt becomes 0, and the TXD pin becomes high-impedance.	N/A
⑧ When transmission restarts	The same processing as in ② occurs.	The next data is written in the main processing, etc.

3.2.2 Differences in Setting Procedures During Master Transmission

Table 3.8 lists the Differences in the Initial Setting Procedures When Transmitting Data, Table 3.9 lists the Differences in Transmit Interrupt Handling When Transmitting Data, and Table 3.10 lists the Differences in Transmit End Interrupt Handling When Transmitting Data.

Table 3.8 Differences in the Initial Setting Procedures When Transmitting Data

Step		RX (RX210)	M16C (M16C/65C)
1	Exiting the module-stop state ^{*1}	SYSTEM.PRCR.WORD = 0xA502; MSTP(SCI0) = 0; SYSTEM.PRCR.WORD = 0xA500;	N/A (no module-stop function)
2	Setting the I/O port functions ^{*2}	PORT2.PMR.BIT.B0 = 0; PORT2.PMR.BIT.B2 = 0; MPC.PWPR.BIT.B0WI = 0; MPC.PWPR.BIT.PFSWE = 1; MPC.P20PFS.BYTE = 0x0A; MPC.P22PFS.BYTE = 0x0A; MPC.PWPR.BYTE = 0x80; PORT2.PMR.BIT.B0 = 1; PORT2.PMR.BIT.B2 = 1;	N/A (no processing) ^{*3}
3	Setting the transmit mode, etc.	SCI0.SCR.BIT.CKE = 0; SCI0.SIMR1.BYTE = 0x00; SCI0.SPMR.BYTE = 0x00; SCI0.SMR.BYTE = 0x80; SCI0.SCMR.BYTE = 0xF2;	uclksel0 = 0x00; pclk1 = 1; u0mr = 0x01; u0c0 = 0x10; ucon = 0x00 u0c1 = 0x00;
4	Setting the bit rate ^{*4}	SCI0.BRR = 39;	u0brg = 79;
5	Setting the interrupt priority level	IPR(SCI0,) = 0x01	
6	Clear the interrupt requests	IR(SCI0, TXI0) = 0;	s0tic = 0x01;
7	Enable peripheral function interrupt requests ^{*5}	SCI0.SCR.BYTE = 0xA4;	
8	Enable transmission		u0c1 = 0x01;
9	Enable interrupt requests ^{*5}	IEN(SCI0, TXI0) = 1;	N/A (no processing)
10	Enable maskable interrupts	setpsw_i();	asm("fset i");
11	Write the first byte of transmit data	N/A (no processing)	/* Write the first byte of data to the U0TB register */

Note 1. For details on the module stop function, refer to section 4.1.3 Module Stop Function.

Note 2. In the RX Family, pin settings for peripheral functions are configured in the MPC. Refer to section 4.1.2 I/O ports for details.

Note 3. In the M32C/80 Series and R32C/100 Series, select the pin function in the function select register.

Note 4. The bit rate calculation method differs between the RX Family and M16C Family. Refer to section 3.2 Calculating the Bit Rate for details.

Note 5. The method of enabling interrupt requests differs. Refer to 4.1.1 Interrupts for details.

Table 3.9 Differences in Transmit Interrupt Handling When Transmitting Data

Step		RX (RX210)	M16C (M16C/65C)
1	Write the transmit data	<code>/* Write transmit data to the SCI0.TDR register */</code>	<code>if (Is transmit data present?)</code> <code>{</code>
2	Determine if the last data have been written	<code>if (Has the last data already been written?)</code> <code>{</code>	<code>/* Write transmit data to the U0TB register */</code> <code>}</code>
3	Disable the TXI interrupt (only after the last data have been written)	<code>IEN(SCI0, TXI0) = 0;</code> <code>SCI0.SCR.BIT.TIE = 0;</code> <code>while (0 != SCI0.SCR.BIT.TIE)</code> <code>{</code> <code>}</code>	N/A (no processing) ^{*1}
4	Clear the interrupt request (only after the last data have been written)	<code>IR(SCI0, TXI0) = 0;</code>	
5	Enable the transmit end interrupt (only after the last data have been written)	<code>IEN(SCI0, TEI0) = 1;</code> <code>}</code>	

Note 1. In the M16C Family, the transmit interrupt does not need to be disabled.

Table 3.10 Differences in Transmit End Interrupt Handling When Transmitting Data

Step		RX (RX210)	M16C (M16C/65C)
1	Check the source of the generated interrupt request	<code>do</code> <code>{</code> <code> while ((0 != SCI0.SCR.BIT.TEIE) &&</code> <code> (0 != SCI0.SSR.BIT.TEND))</code> <code>}</code>	N/A ^{*1}
2	Disable transmission	<code>SCI0.SCR.BIT.TE = 0;</code> <code>while (0 != SCI0.SCR.BIT.TE)</code> <code>{</code> <code>}</code>	
3	Disable the TEI interrupt	<code>SCI0.SCR.BIT.TEIE = 0;</code> <code>while (0 != SCI0.SCR.BIT.TEIE)</code> <code>{</code> <code>}</code>	
4	Check the IR flag	<code>}</code> <code>}while(0 != IR(SCI0, TEI0));</code>	

Note 1. When transmit buffer empty is selected for the transmit interrupt source, an interrupt does not occur when transmission is completed.

3.3 Differences in Slave Reception

This section describes the differences in conditions for enabling reception. Table 3.11 lists Differences in Conditions for Enabling Reception.

Table 3.11 Differences in Conditions for Enabling Reception

Setting	RX	M16C
Master operation	When the SCR.TE bit and RE bit are set to 1, if dummy data is written to the transmit buffer, reception is enabled, and 1 byte of the synchronization clock is output from the SCK pin. When the SCR.TE bit is set to 0 and the RE bit is set to 1, reception becomes enabled when these settings are made, and either the CTS function is disabled, or the synchronization clock continues to be output from the SCK pin when input to the CTS# pin is low.	The TE bit and RE bit are set to 1. ^{*1} When continuous receive mode is disabled, if dummy data is written to the transmit buffer, 1 byte of the synchronous clock is output. When continuous receive mode is enabled, if the receive buffer is read, 1 byte of the synchronous clock is output.
Slave operation	When the SCR.RE bit is set to 1, reception is enabled. Reception starts when the synchronization clock is input to the SCK pin. ^{*1}	The TE bit and RE bit are set to 1. ^{*2} When continuous receive mode is disabled, if dummy data is written to the transmit buffer, reception is enabled. When continuous receive mode is enabled, if the receive buffer is read, reception is enabled. When reception is enabled, reception starts if the synchronous clock is input to the CLK pin.

Note 1. When the SCR.TE bit is set to 1, write dummy data to the transmit buffer before the synchronization clock is input to the SCK pin.

Note 2. In the M16C Family, the TE bit must be set to 1 even when only receiving data.

Section 3.3 describes the differences for slave reception when the conditions in the RX Family assume the TE bit is 0 and the RE bit is 1, and when the conditions in the M16C Family assume the TE bit and RE bit are both 1 and continuous receive mode is disabled.

3.3.1 Differences in Timing During Slave Reception

This section describes an example of the receive interrupt being made to wait because of another interrupt during data reception.

Figure 3.3 shows the Differences in Timing When Receiving Data, and Table 3.12 lists the Differences in Operation at Various Timings and Processing When Receiving Data.

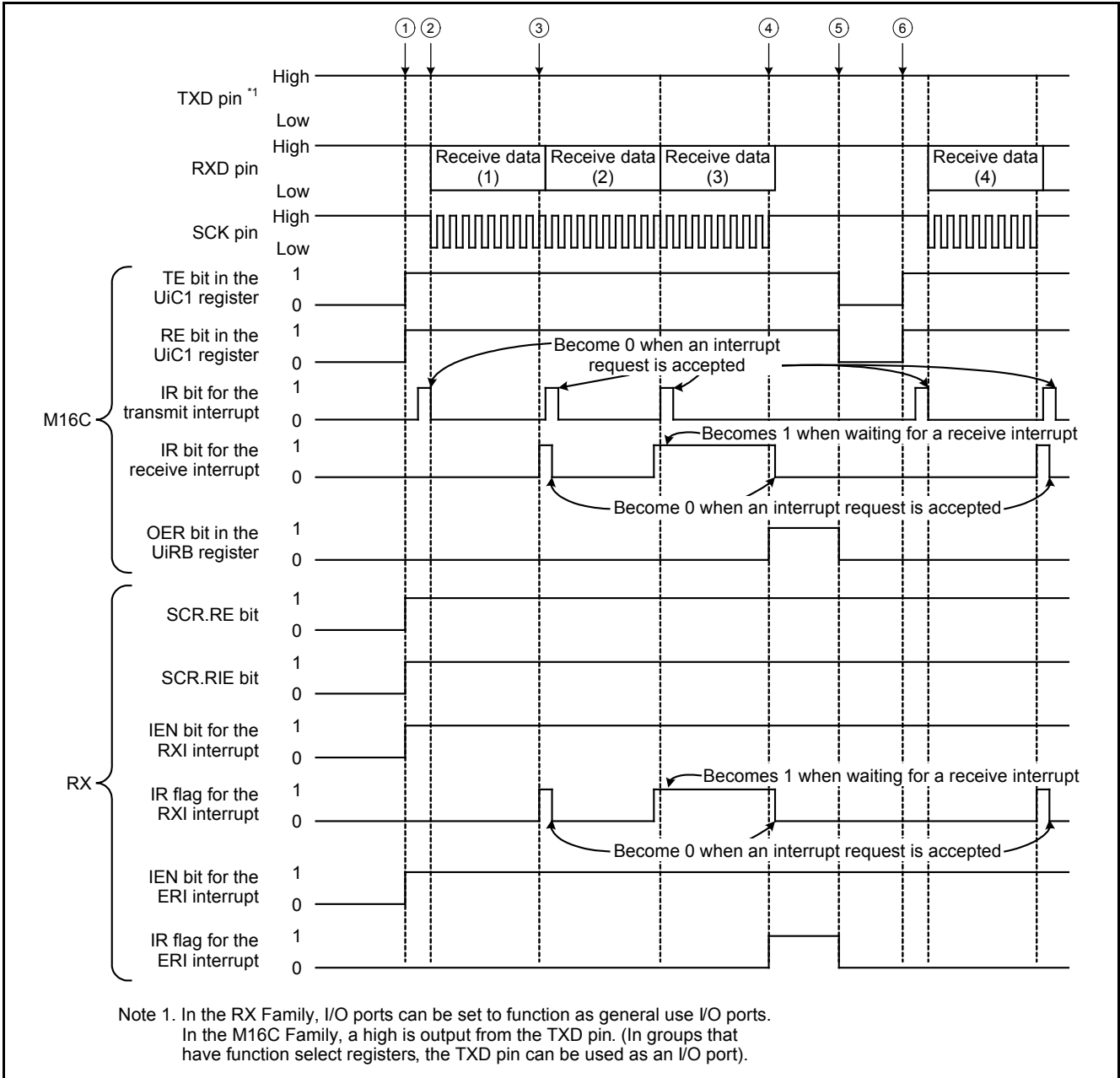


Figure 3.3 Differences in Timing When Receiving Data

Table 3.12 Differences in Operation at Various Timings and Processing When Receiving Data

Timing	RX (RX210)	M16C (M16C/65C)
① Setting to enable reception	Set the SCR.RE bit to 1 (serial reception is enabled), set the RIE bit to 1 (RXI interrupt request is enabled), set the IEN bit for the RXI interrupt to 1 (RXI interrupt request is enabled), and set the IEN bit for the ERI interrupt to 1 (ERI interrupt request is enabled) to enable reception.	After setting the TE bit to 1 (transmission enabled) and setting the RE bit to 1 (reception enabled), write dummy data to the transmit buffer. When transmit data is transferred to the transfer shift register, the IR bit for the transmit interrupt becomes 1, and the transmit interrupt occurs. Dummy data is rewritten to the transmit buffer in the transmit interrupt handling.
② Reception starts	Reception starts when a clock is input to the SCK pin.	
③ When reception is complete	When 1 byte of data is received, receive data is transferred to the receive buffer, the IR flag (IR bit) for the receive interrupt (RXI interrupt) becomes 1, and a receive interrupt is generated. The value is read from the receive buffer in the receive interrupt handling.	
④ When a receive error occurs	The ERI interrupt is generated when an overrun error occurs. Receive error processing is performed in the ERI interrupt handling.	When an overrun error occurs, the OER bit in the receive buffer (UiRB register) becomes 1.
⑤ Clear the receive error flags	After reading the error flags in the SSR register, write 0 to clear the error flags. After clearing all the error flags, the IR flag for the ERI interrupt becomes 0, and reception is enabled.	Set the TE bit to 0 (transmission disabled), set the RE bit to 0 (reception disabled), and set bits SMD2 to SMD0 in the UiMR register to 000b (serial interface disabled).
⑥ When re-enabling reception		After setting bits SMD2 to SMD0 in the UiMR register to 001b (clock synchronous serial I/O mode), and setting both the TE bit and RE bit to 1, if dummy data is written to the transmit buffer, reception is enabled.

3.3.2 Differences in Setting Procedures During Slave Reception

Table 3.13 lists the Differences in the Initial Setting Procedures When Receiving Data, Table 3.14 lists the Differences in Transmit Interrupt Handling When Receiving Data, Table 3.15 lists the Differences in Receive Interrupt Handling When Receiving Data, and Table 3.16 lists the Differences in ERI Interrupt Handling When Receiving Data.

Table 3.13 Differences in the Initial Setting Procedures When Receiving Data

Step		RX (RX210)	M16C (M16C/65C)
1	Exiting the module-stop state ^{*1}	SYSTEM.PRCR.WORD = 0xA502; MSTP(SCI0) = 0; SYSTEM.PRCR.WORD = 0xA500;	N/A (no module-stop function)
2	Setting the I/O port functions ^{*2}	PORT2.PMR.BIT.B1 = 0; PORT2.PMR.BIT.B2 = 0; MPC.PWPR.BIT.B0WI = 0; MPC.PWPR.BIT.PFSWE = 1; MPC.P21PFS.BYTE = 0x01; MPC.P22PFS.BYTE = 0x01; MPC.PWPR.BYTE = 0x80; PORT2.PMR.BIT.B1 = 1; PORT2.PMR.BIT.B2 = 1;	N/A (no processing) ^{*3}
3	Setting the receive mode, etc.	SCI0.SCR.BIT.CKE = 2; SCI0.SMR.BYTE = 0x80; SCI0.SCMR.BYTE = 0xF2; SCI0.SEMR.BYTE = 0x00;	u0mr = 0x09; u0c0 = 0x10; ucon = 0x00 u0c1 = 0x00;
4	Setting the interrupt priority level	IPR(SCI0,) = 0x01	
5	Clear the interrupt requests	IR(SCI0,RX10) = 0;	s0tic = 0x01; s0ric = 0x01;
6	Enable peripheral function interrupt requests ^{*4}	SCI0.SCR.BYTE = 0x50;	
7	Enable receive transmission		u0c1 = 0x05;
8	Enable interrupt requests ^{*4}	IEN(SCI0,ERI0) = 1; /* Note 5 */ IEN(SCI0,RX10) = 1;	N/A (no processing)
9	Enable maskable interrupts	setpsw_i();	asm("fset i");
10	Write dummy data to the transmit buffer	N/A (no processing)	/* Write dummy data to the U0TB register */

Note 1. For details on the module stop function, refer to section 4.1.3 Module Stop Function.

Note 2. In the RX Family, pin settings for peripheral functions are configured in the MPC. Refer to section 4.1.2 I/O ports for details.

Note 3. In the M32C/80 Series and R32C/100 Series, select the pin function in the function select register.

Note 4. The method of enabling interrupt requests differs. Refer to 4.1.1 Interrupts for details.

Note 5. Specifications for the receive error interrupt differ according to the MCU used. Refer to the UMH for details.

Table 3.14 Differences in Transmit Interrupt Handling When Receiving Data

Step		RX (RX210)	M16C (M16C/65C)
1	Write transmit data	N/A (no processing)	/* Write dummy data to the U0TB register */

Table 3.15 Differences in Receive Interrupt Handling When Receiving Data

Step		RX (RX210)	M16C (M16C/65C)
1	Read the receive data	/* Read the receive data from the SCI0.RDR register */	/* Read receive data from the U0RB register */
2	Determine the receive error	N/A (no processing) *1	if(Did a receive error occur? */ {
3	Processing when an error occurs		/* Description of processing when an error occurs */
4	Clear the receive error (disable reception)		re_u0c1= 0; u0mr = u0mr & 0xF8 }

Note 1. In the RX Family, when a receive error occurs, error processing for the RXI interrupt is not necessary because the ERI interrupt is generated.

Table 3.16 Differences in ERI Interrupt Handling When Receiving Data

Step		RX (RX210)	M16C (M16C/65C)
1	Check the source of the generated interrupt request	do { while ((0 != SCI0.SCR.BIT.RIE) && (0x00 != SCI0.SSR.BYTE & 0x38)) {	N/A *1
2	Processing when a receive error occurs	/* Description of processing when an error occurs */	
3	Dummy read the receive buffer	dummy = SCI0.RDR;	
4	Disable reception	SCI0.SCR.BIT.RE = 0; while (0 != SCI0.SCR.BIT.RE) { }	
5	Clear the receive error	dummy = SCI0.SSR.BYTE; /* Note 2 */ SCI0.SSR.BYTE = 0xC0; while (0x00 != SCI0.SSR.BYTE & 0x38) { }	
6	Check the IR flag	} }while(0 != IR(SCI0, ERI0));	

Note 1. In the M16C Family, a separate interrupt is not generated when a receive error occurs. Error determination or processing is performed in the receive interrupt. Refer to Table 3.15 Differences in Receive Interrupt Handling When Receiving Data for details.

Note 2. When clearing the PER flag (parity error flag), FER flag (framing error flag), and ORER flag (overrun error flag), the flags must be read as 1 in advance.

3.4 Calculating the Bit Rate

There are differences in calculating the bit rate between the RX Family and M16C Family.

Table 3.17 Differences in Calculating the Bit Rate

Item	RX (RX210)	M16C (M16C/65C)
Calculating the bit rate using the internal clock	$\text{Clock source} \div 4(N + 1)^{**1}$ where the clock source is PCLK, PCLK/4, PCLK/16, or PCLK/64, and N is the BRR register setting value	$\text{Clock source} \div 2(n + 1)$ where the clock source is f1SIO, f2SIO, f8SIO, or f32SIO, and n is the UiBRG setting value
Calculating the bit rate using the external clock	$f\text{EXT} \div 16$ where fEXT is input from the SCK pin	fEXT where fEXT is input from the CLK pin

Note 1. Based on the "Relationships between N Setting in BRR and Bit Rate B" in the UMH:

$$\begin{aligned}
 B &= \text{PCLK} \div (8 \times 2^{2n-1} \times (N + 1)) \\
 &= \text{PCLK} \div (4 \times 2^{2n} \times (N + 1)) \\
 &= (\text{PCLK} \div 2^{2n}) \div (4 \times (N + 1)) \\
 &= \text{Clock source} \div (4 \times (N + 1))
 \end{aligned}$$

4. Appendix

4.1 Points on Migrating from the M16C to the RX

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

4.1.1 Interrupts

In the RX Family, if an interrupt request is received when all of the following conditions are met, the interrupt can be accepted.

- 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 by the interrupt request enable bits for the peripheral function.

Table 4.1 lists the Comparison of Conditions for Generating an Interrupt.

Table 4.1 Comparison of Conditions for Generating an Interrupt

Item	RX	M16C
I flag	When the I flag is set to 1 (enabled), the maskable interrupts are accepted.	
Interrupt request flag	When there is an interrupt request from 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	Interrupt enable or disable can be specified 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 UMH.

4.1.2 I/O ports

In the RX Family, the MPC must be configured in order to assign I/O signals of peripheral functions to pins. Before controlling the input and output 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 appropriate pins.
- In the PORT.PMR register, select the function for the pin to be used as a general I/O port or I/O port for a peripheral function.

Table 4.2 lists Comparison of I/O Settings for Peripheral Function Pins.

Table 4.2 Comparison of I/O Settings for Peripheral Function Pins

Function	RX (RX210)	M16C (M16C/65C)
Select the pin function	A pin to which a peripheral I/O is assigned can be selected from multiple pins. The PFS register is used to select a peripheral function I/O that is assigned to a pin used.	These are not available in the M16C Group. *1 When a mode of a peripheral function is selected, an appropriate pin is assigned as an I/O pin for the peripheral function.
Switch between general I/O port and peripheral function	With the PMR register, the pin function can be selected as a general I/O port or a peripheral function.	

Note 1. The M32C Group and R32C Group have registers to assign a peripheral function to a pin.

For more information, refer to the Multi-Function Pin Controller (MPC) and I/O port sections in the UMH.

4.1.3 Module Stop Function

The RX Family has the ability to stop each peripheral module individually. By placing unused peripheral modules in 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 read or written.

For more information, refer to the Low Power Consumption section in the UMH.

4.2 I/O Register Macros

Macro definitions listed in Table 4.3 can be found in the RX I/O register definitions (iodefine.h).

The readability of programs can be achieved with these macro definitions.

Table 4.3 lists examples of Macros.

Table 4.3 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).
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 register.

4.3 Intrinsic Functions

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

Table 4.4 lists examples of Descriptions of Special Instructions and Control Register Settings.

Table 4.4 Descriptions of Special Instructions and Control Register Settings

Item	Description	
	RX	M16C
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.

5. 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 UMH 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: Clock Synchronous Serial Data Communications
-------------------------	--

Rev.	Date	Description	
		Page	Summary
1.00	July 1, 2014	—	First edition issued

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 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.



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: +1-408-586-6000, 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