APPLICATION NOTE

RX651 Group

SH7083/SH7084/SH7085/SH7086 to RX651 Microcontroller Migration Guide

R01AN3995EJ0110
Rev.1.10
Dec 21, 2020

Summary

This application note describes points requiring special attention, points of difference, etc., that need to be borne in mind when replacing the SH7083, SH7084, SH7085, or SH7086 with the RX651 in a user system. For detailed information on each function, refer to the latest version of the User's Manual: Hardware.

In this application note the SH7083, SH7084, SH7085, and SH7086 are referred to collectively as the SH7080 Group, and the specifications of the SH7086 are treated as representative. Although there are minor differences in functions and pins among the products composing the SH7080 Group, functionally they are all basically equivalent to the SH7086. This application note therefore applies to the entire SH7080 Group.

Target Devices

RX651/RX65N

Contents

1. CPU Architecture

1.1 System Registers

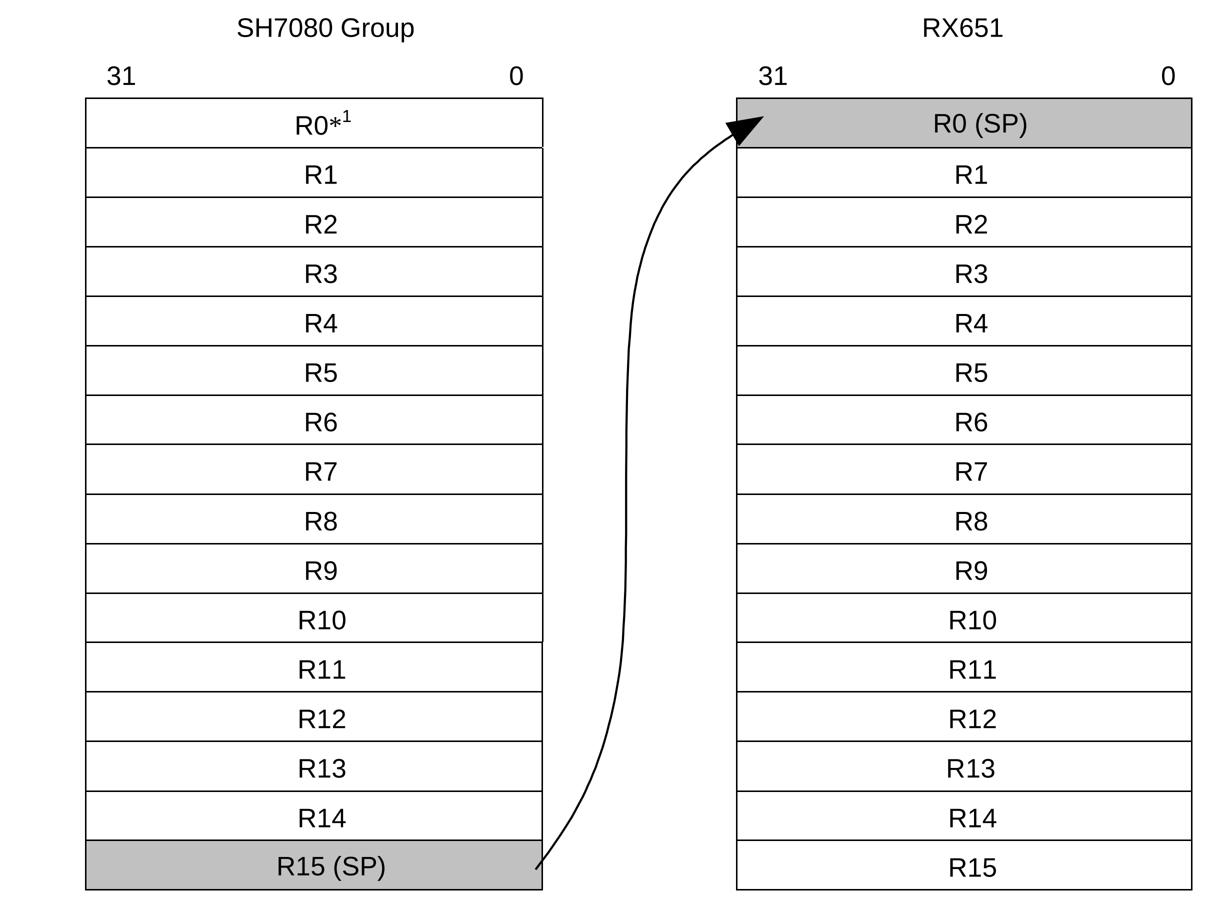
The points of difference between the registers of the SH7080 Group and the RX651 are described below.

1.1.1 General-Purpose Registers

The SH7080 Group and RX651 each have 16 32-bit general-purpose registers. They differ in that the register used as the stack pointer (SP) is different.

- SH7080 Group: R15
- RX651: R0

On the SH7080 Group, R0 is also used as an index register.



Note 1. Used as the index register in the indexed register indirect and indexed GBR indirect addressing modes. R0 may be fixed as the source or destination register, depending on the instruction.

Figure 1.1 Differences Between General-Purpose Registers

1.1.2 Control Registers

Figure 1.2 shows the points of difference between the control registers of the SH7080 Group and the RX651.

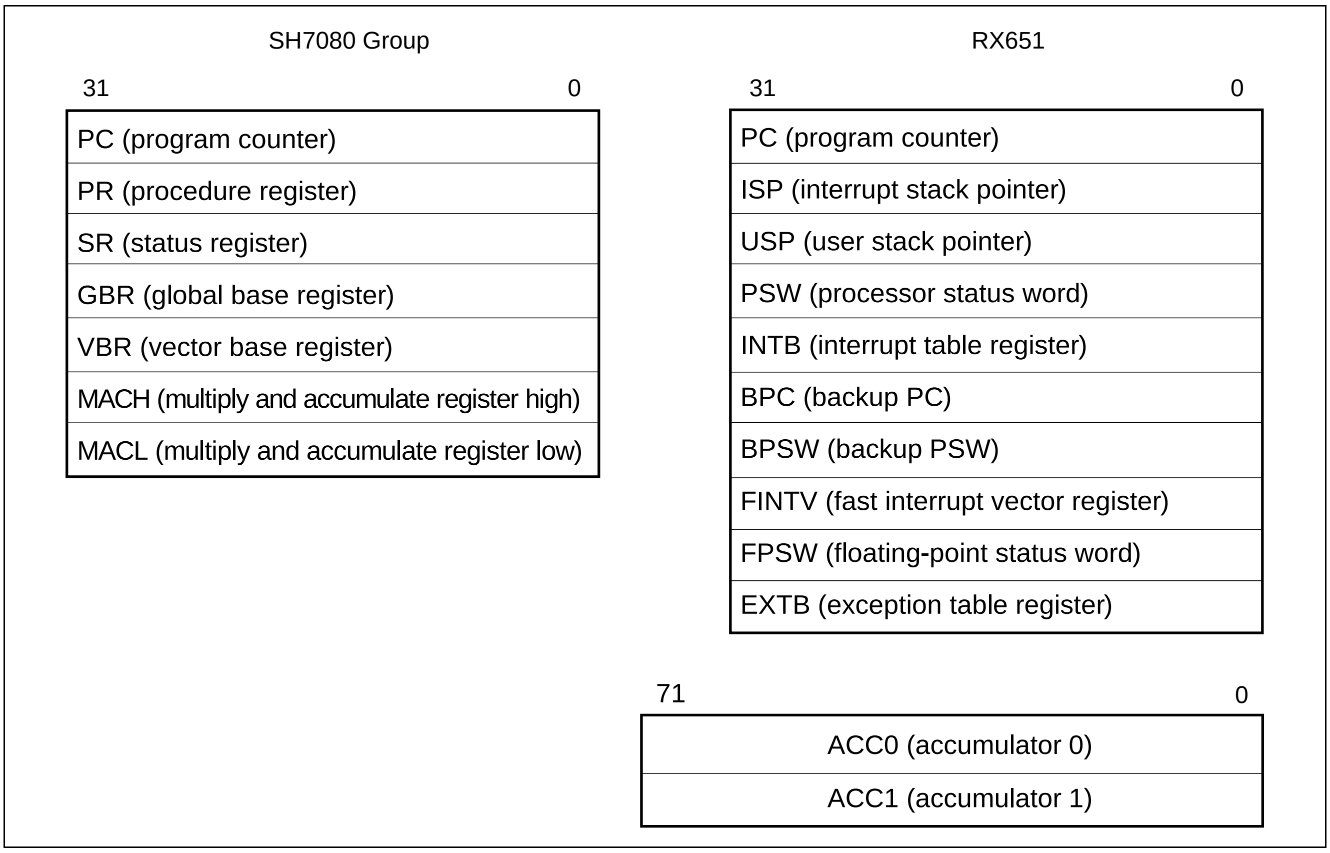

SH7080 Group
31
0
PC (program counter)
PR (procedure register)
SR (status register)
GBR (global base register)
VBR (vector base register)
MACH (multiply and accumulate register high)
MACL (multiply and accumulate register low)
RX651
31
0
PC (program counter)
ISP (interrupt stack pointer)
USP (user stack pointer)
PSW (processor status word)
INTB (interrupt table register)
BPC (backup PC)
BPSW (backup PSW)
FINTV (fast interrupt vector register)
FPSW (floating-point status word)
EXTB (exception table register)
71
0
ACC0 (accumulator 0)
ACC1 (accumulator 1)

Figure 1.2 Differences Between Control Registers

The RX651 has no registers corresponding to PR and GBR on the SH7080 Group. The ACC0 and ACC1 registers on the RX651 corresponds to MACH and MACL on the SH7080 Group. An outline of the control registers that are implemented on the RX651 but not on the SH7080 Group is presented below.

- Interrupt stack pointer/user stack pointer (ISP/USP)
  There are two types of stack pointer (SP): the interrupt stack pointer (ISP) and the user stack pointer (USP). Switching the stack pointer in use (ISP or USP) is accomplished by means of the stack pointer select bit (U) in the processor status word (PSW) register.
- Interrupt table register (INTB)*1
  Specifies the start address of the interrupt vector table.
- Exception table register (EXTB)*1
  Specifies the start address of the exception vector table.
- Backup PC/backup PSW (BPC/BPSW)
  The RX651 supports fast interrupts in addition to ordinary interrupts. For fast interrupts, the contents of PC and PSW are saved to dedicated registers (BPC and BPSW), thereby reducing the processing time needed to save the register data.
- Fast interrupt vector register (FINTV)
  This register specifies the jump destination when a fast interrupt occurs.
- Floating-point status word (FPSW)
  This register indicates the status of the calculation result (floating-point calculation result) generated by the RX651's on-chip FPU.

Note 1. The functionality of this register is equivalent to that of VBR on the SH7080 Group.

Figure 1.3 and Table 1.1 show the points of difference between the status registers.

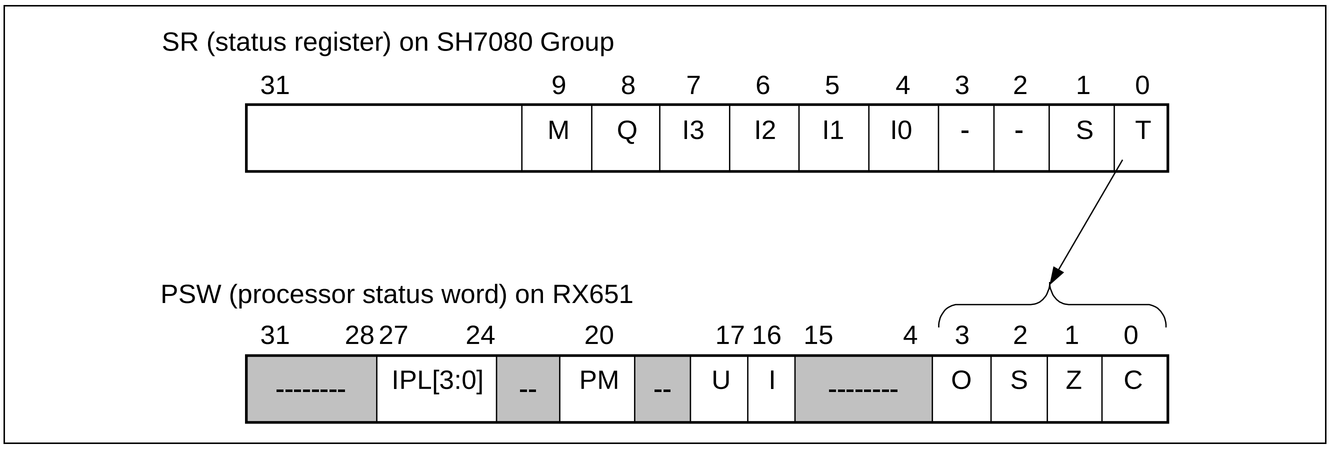


Figure 1.3 Differences Between SR (SH7080 Group) and PSW (RX651)

Table 1.1 Differences Between SR (SH7080 Group) and PSW (RX651)

SH Bit Name	RX Bit Name	Description
T	C Z S O	The calculation result (true/false, carry/borrow, etc.) indicated by the T bit on the SH7080 Group is shown by four flags (C, Z, S, and O) on the RX651. C: Carry flag (0/1 = No carry has occurred./A carry has occurred.) Z: Zero flag S: Sign flag O: Overflow flag
S	—	Controls the functionality that prevents overflows during ALU arithmetic operations performed by the DSP unit of the SH7080 Group. On the RX651 there is no bit corresponding to the S bit, and the occurrence of an overflow during a floating-point operation is reported by the FPSW flag. It is also possible to perform exception handling when an overflow occurs.
I0, I1, I2, I3	IPL[3:0]	These are the interrupt mask bits. Both the SH7080 Group and the RX651 support level settings from 0 (lowest) to 15 (highest). Only interrupts with a priority level higher than this setting are accepted.
Q	—	The Q bit is used by the DIV0U, DIV0S, and DIV1 instructions on the SH7080 Group. There is no corresponding bit on the RX651.
M	—	The M bit is used by the DIV0U, DIV0S, and DIV1 instructions on the SH7080 Group. There is no corresponding bit on the RX651.
—	I	Interrupt enable bit 0: Interrupts are disabled. 1: Interrupts are enabled. This bit is used to enable interrupt requests on the RX651. The initial state is 0, so it is necessary to set this bit to 1 in order to accept interrupts. Also, this bit is cleared to 0 when an exception is accepted, and no interrupts are accepted while its value remains 0. Note that the interrupt status flag of the interrupt controller is reset when an interrupt request occurs, regardless of the setting of this bit.
—	U	This bit specifies the stack pointer used by the RX651. 0: Interrupt stack pointer (ISP) 1: User stack pointer (USP) This bit is cleared to 0 when an exception is accepted.
—	PM	This bit specifies the processor mode of the RX651. 0: Supervisor mode 1: User mode This bit is cleared to 0 when an exception is accepted.

1.2 Option-Setting Memory

The RX651 is provided with an option-setting memory area containing registers for selecting the microcontroller state after a reset of the endian mode, watchdog timer operation, etc. The option-setting memory is allocated in the ROM, and it cannot be overwritten by a software program. When programming the ROM, it is necessary to program appropriate values in the option-setting memory as well.

1.2.1 Outline of Option-Setting Memory

Figure 1.4 shows an outline of the option-setting memory area.

b31 b0

Address	...	Register Description
FE7F 5D00h to FE7F 5D03h	Endian select register (MDE)	Register for selecting the endian setting of the CPU.
FE7F 5D04h to FE7F 5D07h	Option function select register 0 (OFS0)	The OFS0 register is used to make settings for the independent watchdog timer (IWDT) and watchdog timer (WDT).
FE7F 5D08h to FE7F 5D0Bh	Option function select register 1 (OFS1)	The OFS1 register is used to make the following settings: • Voltage monitor 0 reset is enabled/ disabled after a reset. • Voltage monitor 0 level is selected after a reset. • HOCO oscillation is enabled/disabled after a reset.
	—	—
FE7F 5D10h to FE7F 5D13h	TM identification data register (TMINF)	Provides an area for storing 32 bits of user-defined data, such as codes for identifying programs stored in TM-enabled areas.
	—	—
FE7F 5D40h to FE7F 5D43h	Serial programmer command control register (SPCC)	Register for enabling or disabling connection to a serial programmer.
	—	—
FE7F 5D48h to FE7F 5D4Bh	TM enable flag register (TMEF)	Register for enabling or disabling the TM function.
	—	—
FE7F 5D50h to FE7F 5D5Fh	OCD/serial programmer ID setting register (OSIS)	Provides an area for storing the ID for the ID code protection function of the OCD/serial programmer.
	—	—
FE7F 5D64h to FE7F 5D67h	Flash access window setting register (FAW)	Register for making flash access window settings.
	—	—
FE7F 5D70h to FE7F 5D73h	ROM code protection register (ROMCODE)	Register for enabling or disabling ROM code protection.
	...	—

Figure 1.4 RX 651 Option-Setting Memory Area

Figure 1.5 to Figure 1.8 show sample settings for the option-setting memory.

```
/* Settings for big-endian */
#define __BIG
#pragma address MDE_REG = 0xFE7F5D00 // MDE register
#ifdef __BIG
            const unsigned long MDE_REG = 0xFFFFFFF8; // big
#else
            const unsigned long MDE_REG = 0xFFFFFFFF; // little
#endif
```

Figure 1.5 RX 651 Endian Setting Example

```
/* Settings for enabling serial programmer connection after a reset */
#pragma address SPCC_REG = 0xFE7F5D40 // SPCC register
const unsigned long SPCC_REG = 0xFFFFFFFF;

/* ID code settings for OCD/serial programmer */
/* ID1 = 0xFF, ID2 = 0x02, ID3 = 0x03, ID4 = 0x04 */
/* ID5 = 0x05, ID6 = 0x06, ID7 = 0x07, ID8 = 0x08 */
/* ID9 = 0x09, ID10 = 0x0A, ID11= 0x0B, ID12 = 0x0C */
/* ID13 = 0x0D, ID14 = 0x0E, ID15 = 0x0F, ID16 = 0x10 */
#pragma address OSIS1_REG = 0xFE7F5D50 // OSIS register
const unsigned long OSIS1_REG = 0x040302FF; // ID1, ID2, ID3, ID4

#pragma address OSIS5_REG = 0xFE7F5D54 // OSIS register
const unsigned long OSIS5_REG = 0x08070605; // ID5, ID6, ID7, ID8

#pragma address OSIS9_REG = 0xFE7F5D58 // OSIS register
const unsigned long OSIS9_REG = 0x0C0B0A09; // ID9, ID10, ID11, ID12

#pragma address OSIS13_REG = 0xFE7F5D5C // OSIS register
const unsigned long OSIS13_REG = 0x100F0E0D; // ID13, ID14, ID15, ID16
```

Figure 1.6 RX651 OCD/Serial Programmer Setting Example

```
/* Flash access window settings */
#pragma address FAW_REG = 0xFE7F5D64 // FAW register
const unsigned long FAW_REG = 0xF5A572A;
```

Figure 1.7 RX651 Flash Access Window Setting Example

```
#pragma address OFS1_REG = 0xFE7F5D08 // OFS1 register
const unsigned long OFS1_REG = 0xFFFFFFFF;

#pragma address OFS0_REG = 0xFE7F5D04 // OFS0 register
const unsigned long OFS0_REG = 0xFFFFFFFF;
```

Figure 1.8 RX651 OFS0 and OFS1 Setting Example

1.2.2 Endian Setting

The SH7080 Group is fixed in big-endian mode. On the RX651, instructions are fixed in little-endian, and the data order is selectable between little-endian and big-endian. The endian setting is specified by means of the endian select bits (MDE[2:0]) in the MDE register in the option-setting memory.

When switching from the SH7080 Group to the RX651, it is possible to use big-endian order by specifying big-endian in the option settings of the genuine Renesas compiler. This allows migration without the need to be conscious of endianness in the user program.

The endian setting can be switched for each CS area in the external address space. However, instruction code cannot be allocated to an external space with an endian setting that differs from that of the chip. When allocating instruction code to an external space, ensure that an area with the same endian setting as the chip is used. (For details, see the User's Manual: Hardware.)

Endian settings based on the compiler option setting are illustrated in Figure 1.9. The files generated automatically based on the compiler option setting have been confirmed to run in the environment described in 3.1, Operating Environment.

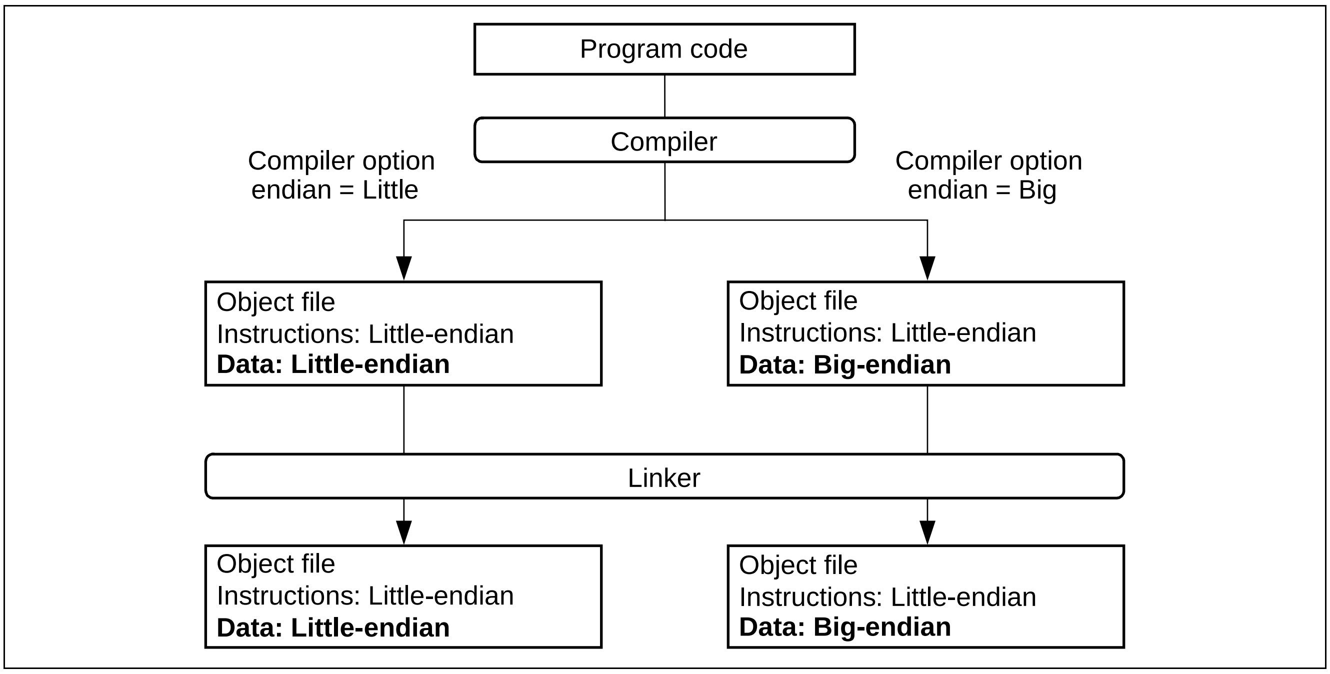

Program code
Compiler
Compiler option endian = Little
Compiler option endian = Big
Object file Instructions: Little-endian Data: Little-endian
Object file Instructions: Little-endian Data: Big-endian
Linker
Object file Instructions: Little-endian Data: Little-endian
Object file Instructions: Little-endian Data: Big-endian

Figure 1.9 RX651 Specifying Endianness by Compiler Option

1.2.3 Specifying TM Identification Data and Setting TM Enable Flags

The RX651 is provided with a trusted memory (TM) function to prevent third parties from reading software stored in blocks 8 and 9 of the code flash memory. The TM function prevents reading of some designated areas even internally by the microcontroller, such as the on-chip flash memory, and allows instruction execution only.

This function is useful when storing software for processing encryption algorithms, device control processing software incorporating valuable intellectual property, commercial middleware, or the like in the code flash memory.

1.2.4 OCD/Serial Programmer Settings

The RX651 supports selection of serial programming functions by means of serial programmer commands. The SPCC register is used to enable serial programming.

When an OCD/serial programmer is connected, the data written in the option-setting memory is used to determine whether or not to accept the connection. A check is performed to determine if the code sent by the OCD/serial programmer matches the ID code in the option-setting memory. The connection to the OCD/serial programmer is enabled if the codes match, but no connection is possible if the codes do not match. The ID code of the OCD/serial programmer is stored in the OSIS register.

1.3 Reset Function

1.3.1 Reset Sources

Table 1.2 lists the reset sources of the SH7080 Group and RX651.

Table 1.2 Reset Sources

Item	SH7080 Group	RX651
Reset type	• Power-on reset (RES# pin reset/WDT overflow) • Manual reset	• RES# pin reset • Power-on reset (internal reset) • Voltage monitor 0 reset • Voltage monitor 1 reset • Voltage monitor 2 reset • Deep software standby reset • Independent watchdog timer reset • Watchdog timer reset • Software reset

(1) Reset Vector Configuration

The SH7080 Group has separate vectors for power-on resets and for manual resets (PC and SP).*[1]

The RX651 has a single reset vector for multiple reset sources. The reset source is identified in reset status registers 0 to 2 during reset processing, and processing for the corresponding source is performed.

(2) Stack Pointer

On the SH7080 Group, it is necessary to specify the end address (+1) of the stack area in the reset vector. There is no stack pointer setting area in the vector table on the RX651, so the stack pointer is set in ISP and USP.

Note 1. See 1.7.4, Vector Configuration, for details of the vector tables.

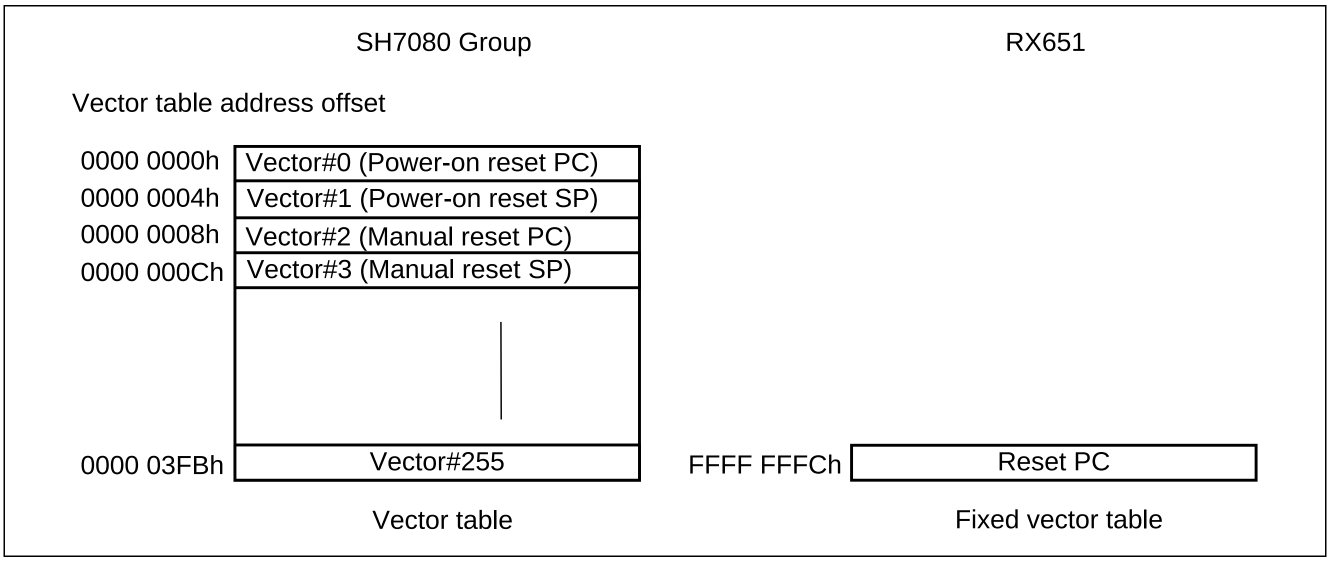

SH7080 Group
RX651
Vector table address offset
0000 0000h
Vector#0 (Power-on reset PC)
0000 0004h
Vector#1 (Power-on reset SP)
0000 0008h
Vector#2 (Manual reset PC)
0000 000Ch
Vector#3 (Manual reset SP)
0000 03FBh
Vector#255
FFFF FFFCh
Reset PC
Vector table
Fixed vector table

Figure 1.10 Reset Vector Comparison

1.3.2 Reset Sources and Initialization Scope

The initialization scope of the reset sources differs between the SH7080 Group and the RX651. Table 1.3 lists the reset sources and their initialization scope on the SH7080 Group, and Table 1.4 lists the reset sources and their initialization scope on the RX651. For details, see the User’s Manual: Hardware.

Table 1.3 SH7080 Group Reset Sources and Initialization Scope

Item	Power-On Reset	Manual Reset
CPU	○	○
On-chip peripheral modules	○	—

○: Reset —: No reset

Table 1.4 RX651 Reset Sources and Initialization Scope

	Reset Sources								
Reset Target	Res# Pin Reset	Power-On Reset	Voltage Monitor 0 Reset	Independent Watchdog Timer Reset	Watchdog Timer Reset	Voltage Monitor 1 Reset	Voltage Monitor 2 Reset	Deep Software Standby Reset	Software Reset
Power-on reset detection flag	○	—	—	—	—	—	—	—	—
Cold start/warm start determination flag	—	○	—	—	—	—	—	—	—
Voltage monitor 0 reset detection flag	○	○	—	—	—	—	—	—	—
Independent watchdog timer reset detection flag	○	○	○	—	—	—	—	○	—
Independent watchdog timer registers	○	○	○	—	—	—	—	○	—
Watchdog timer reset detection flag	○	○	○	○	—	—	—	○	—
Watchdog timer registers	○	○	○	○	—	—	—	○	—
Voltage monitor 1 reset detection flag	○	○	○	○	○	—	—	—	—
Voltage monitor function 1 registers	○	○	○	○	○	—	—	*1	—
Voltage monitor 2 reset detection flag	○	○	○	○	○	○	—	—	—
Voltage monitor function 2 registers	○	○	○	○	○	○	—	*2	—
Deep software standby reset detection flag	○	○	○	○	○	○	○	—	—
Software reset detection flag	○	○	○	○	○	○	○	○	—
Realtime clock registers	—	—	—	—	—	—	—	—	—
High-speed on-chip oscillator–related registers	○	○	○	○	○	○	○	—	○
Main clock oscillator–related registers	○	○	○	○	○	○	○	—	○
Pin states	○	○	○	○	○	○	○	—	○
Low power consumption–related registers	○	○	○	○	○	○	○	—	○
Registers other than the above, CPU, and internal state	○	○	○	○	○	○	○	○	○

○: Reset —: No change

Note 1. Only LVD1CR1 and LVD1SR are initialized.

Note 2. Only LVD2CR1 and LVD2SR are initialized.

1.3.3 Cold/Warm Start Determination Function

On the RX651 it is possible to determine whether the most recent reset processing was caused by a power-on reset (cold start) or by a reset signal during operation (warm start).

When a power-on reset occurs because the external voltage (VCC) has exceeded the threshold, the cold/warm start determination flag (RSTSR1.CWSF) is cleared to 0, indicating a cold start. Since the flag is not cleared to zero by any other type of reset, 1 can be written to it by a program, indicating a warm start.

1.3.4 Write Protection

The RX651 has a register write protection function to protect important registers from being overwritten if program runaway occurs. The software reset register is protected by this function.

If necessary, set protect bit 1 (PRCR.PRC1) to 1 to enable writes before writing to the software reset register.

1.4 Clock Settings

1.4.1 Clock Sources

Table 1.5 lists the clock sources of the SH7080 Group and RX651.

Table 1.5 Clock Sources

SH7080 Group	RX651
Oscillator (EXTAL and XTAL) + PLL circuit	• Main clock oscillator (EXTAL and XTAL) + PLL circuit • Subclock oscillator (XCIN and XCOUT) • High-speed on-chip oscillator (HOCO) + PLL circuit • Low-speed on-chip oscillator (LOCO) • IWDT-dedicated on-chip oscillator

In the description below, the high-speed on-chip oscillator is referred to as the HOCO and the low-speed on-chip oscillator as the LOCO.

1.4.2 Clock Generation Circuit

On the SH7080 Group application of divider settings and oscillation stop detection control are performed in software. On the RX651 a variety of clock control operations are performed in software.

On the RX651 the LOCO operates as the clock source after a reset. The operation of necessary clock sources and PLL circuits other than the LOCO is started during system initialization, and various clocks are selected, such as the system clock and bus clocks. When making changes to clock-related settings, it is necessary to consider the register setting sequence and the oscillation and clock oscillation stabilization time.

See the following application note for details of the clock setting procedure.

RX65N Group, RX651 Group Initial Setting (R01AN3034EJ)

1.4.3 Write Protection

The RX651 has a register write protection function to protect important registers from being overwritten if program runaway occurs, and the registers related to the clock generation circuit are protected by this function.

If necessary, set protect bit 0 (PRCR.PRC0) or protect bit 1 (PRCR.PRC1) to 1 to enable writes before writing to these registers.

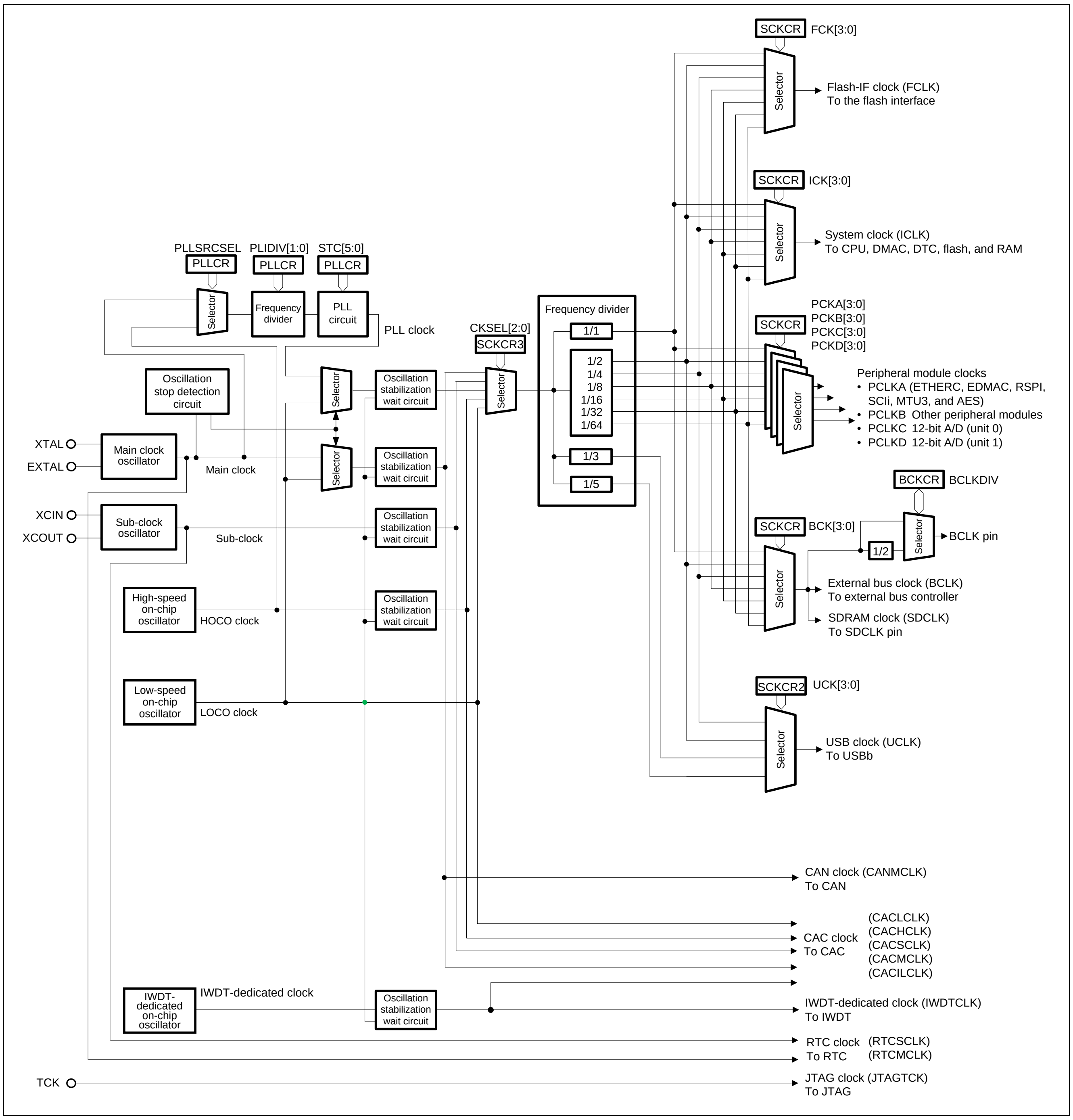

SCKCR FCK[3:0]
Selector
Flash-IF clock (FCLK) To the flash interface
SCKCR ICK[3:0]
System clock (ICLK) To CPU, DMAC, DTC, flash, and RAM
PLLSRCSEL PLLCR
PLIDIV[1:0] PLLCR
STC[5:0] PLLCR
Frequency divider
PLL circuit
PLL clock
CKSEL[2:0] SCKCR3
Frequency divider
1/1
1/2 1/4 1/8 1/16 1/32 1/64
1/3
1/5
SCKCR PCKA[3:0] PCKB[3:0] PCKC[3:0] PCKD[3:0]
Peripheral module clocks
• PCLKA (ETHERC, EDMAC, RSPI, SCIi, MTU3, and AES)
• PCLKB Other peripheral modules
• PCLKC 12-bit A/D (unit 0)
• PCLKD 12-bit A/D (unit 1)
Oscillation stop detection circuit
Oscillation stabilization wait circuit
XTAL
EXTAL
Main clock oscillator
Main clock
XCIN
XCOUT
Sub-clock oscillator
Sub-clock
High-speed on-chip oscillator
HOCO clock
Low-speed on-chip oscillator
LOCO clock
BCKCR BCLKDIV
SCKCR BCK[3:0]
1/2
BCLK pin
External bus clock (BCLK) To external bus controller
SDRAM clock (SDCLK) To SDCLK pin
SCKCR2 UCK[3:0]
USB clock (UCLK) To USBb
CAN clock (CANMCLK) To CAN
CAC clock To CAC
(CACLCLK) (CACHCLK) (CACSCLK) (CACMCLK) (CACILCLK)
IWDT-dedicated on-chip oscillator
IWDT-dedicated clock
IWDT-dedicated clock (IWDTCLK) To IWDT
RTC clock To RTC
(RTCSCLK) (RTCMCLK)
TCK
JTAG clock (JTAGTCK) To JTAG

Figure 1.11 RX651 Block Diagram of Clock Generation Circuit

1.5 Operation Modes

1.5.1 Comparison of Operation Modes

Table 1.6 shows a comparison of the operation modes of the SH7080 Group and RX651.

For details of each operation mode, see the User's Manual: Hardware.

Table 1.6 Comparison of Operation Modes

SH7080 Group	RX651	Description
MCU extension mode 0	On-chip ROM disabled extended mode	An operation mode in which the on-chip ROM is disabled and the external address space is enabled. The external bus width differs from that of mode 0 and mode 1 on the SH7080 Group.
MCU extension mode 1		
MCU extension mode 2	On-chip ROM enabled extended mode	An operation mode in which the on-chip ROM is enabled and the external address space is enabled
Single-chip mode	Single-chip mode	An operation mode in which the external address space is disabled
Boot mode	Boot mode (SCI interface)	An operation mode in which the on-chip flash memory modifying program (boot program), which is stored in a dedicated area internal to the microcontroller, is run. The on-chip ROM can be programmed by a device external to the microcontroller by using the asynchronous serial interface. On the SH7080 Group the user MAT and user boot MAT are programmed. On the RX651 the code flash memory is programmed.
—	Boot mode (USB interface)	An operation mode in which the on-chip flash memory modifying program (boot program), which is stored in a dedicated area internal to the microcontroller, is run. The on-chip ROM (code flash memory) can be programmed by a device external to the microcontroller by using the USB interface.
—	Boot mode (FINE interface)	An operation mode in which the on-chip flash memory modifying program (boot program), which is stored in a dedicated area internal to the microcontroller, is run. The on-chip ROM (code flash memory) can be programmed by a device external to the microcontroller by using the FINE interface.
User boot mode	—	An operation mode in which the on-chip flash memory modifying program (boot program), which is stored in a dedicated area internal to the microcontroller, is run. The user MAT can be programmed by using a user-defined interface.
User program mode	—	An operation mode in which the on-chip flash memory modifying program (boot program), which is stored in a dedicated area internal to the microcontroller, is run. The user MAT can be programmed by creating a user boot program that provides a user-defined interface.

1.5.2 Comparison of Memory

Figure 1.12 shows a comparison of memory maps in on-chip ROM enabled mode.

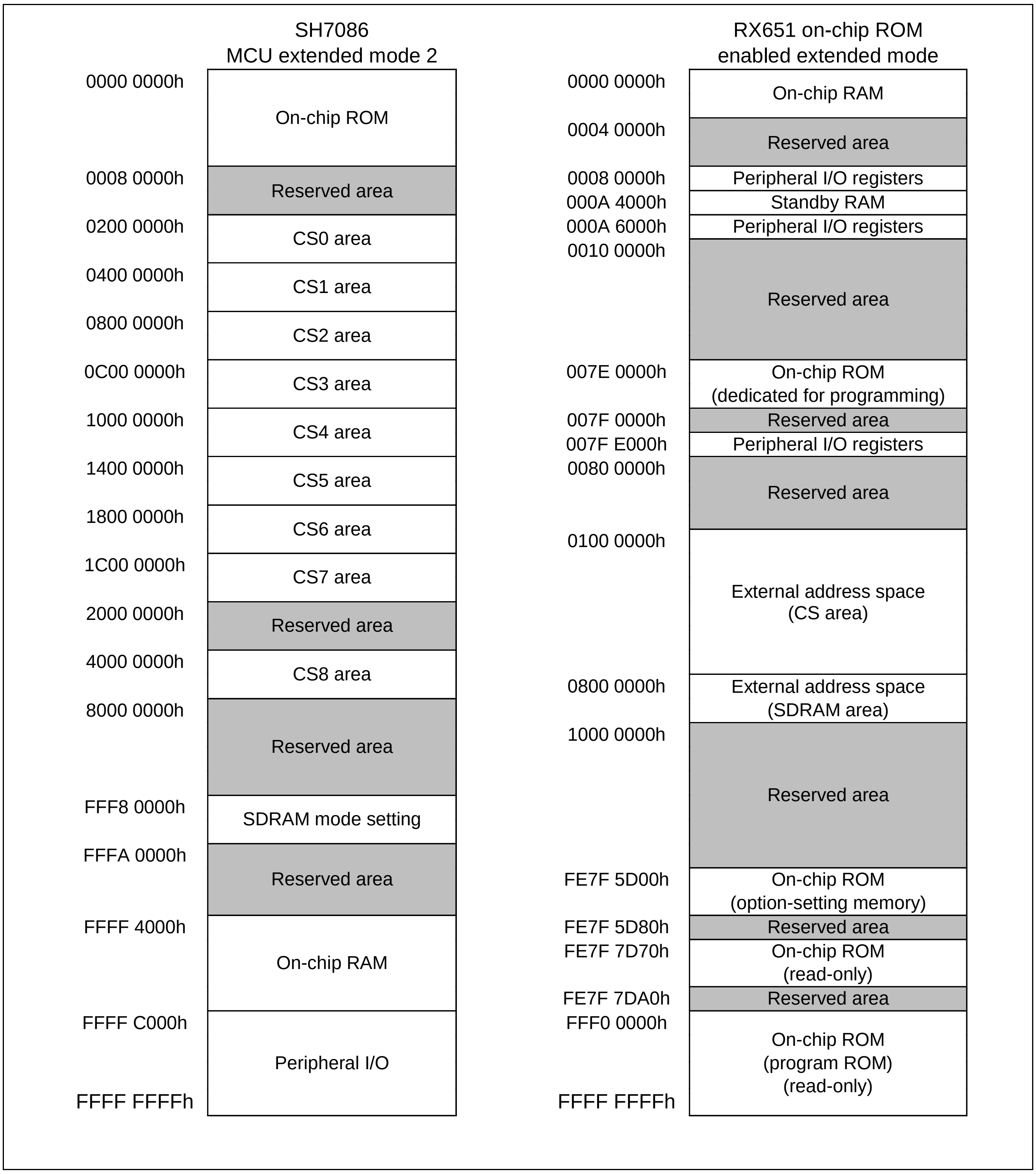


Figure 1.12 Memory Map Comparison (On-Chip ROM Enabled Mode)

Figure 1.13 shows a comparison of memory maps in single-chip mode.

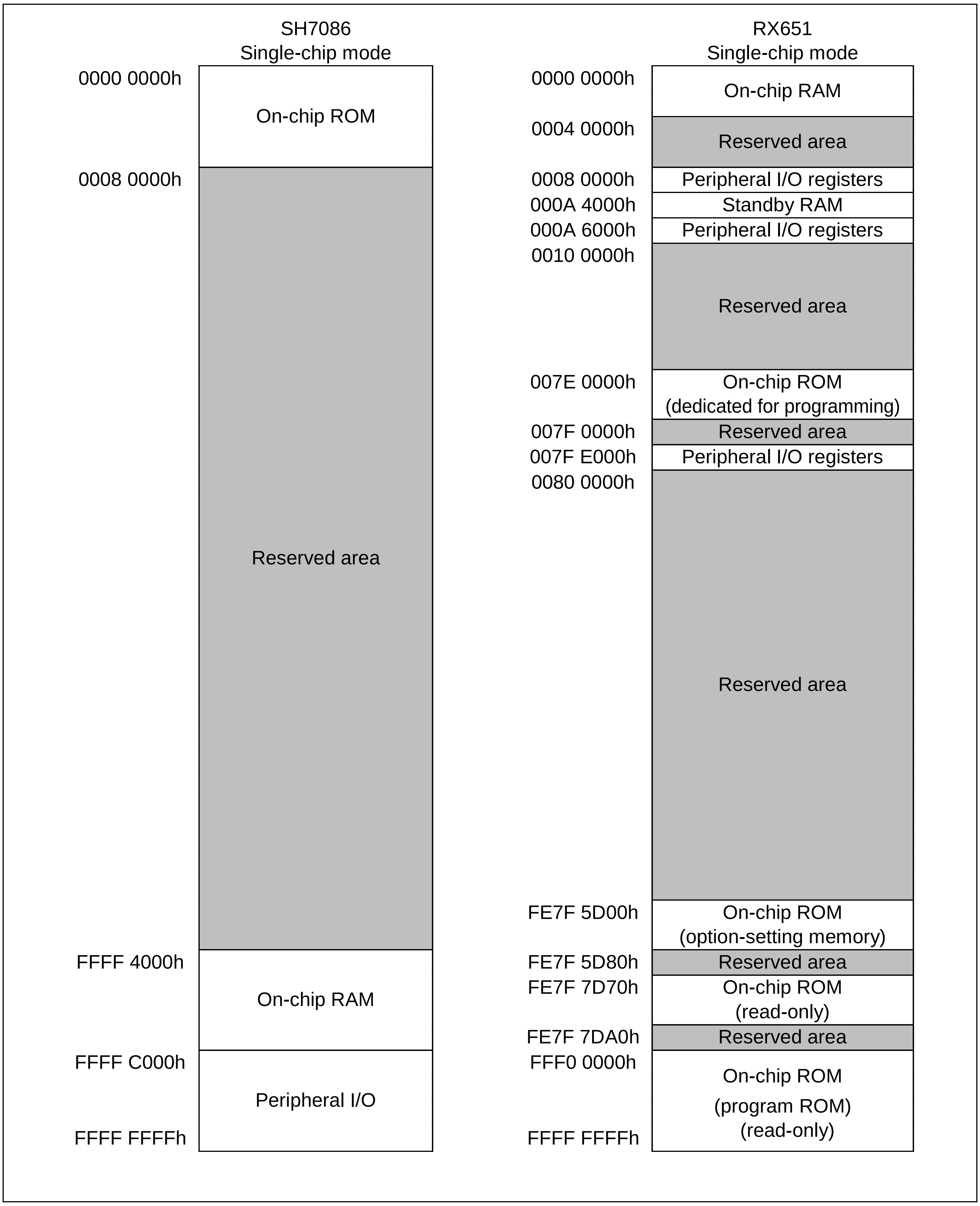


Figure 1.13 Memory Map Comparison (Single-Chip Mode)

Figure 1.14 shows a comparison of memory maps in on-chip ROM disabled mode.

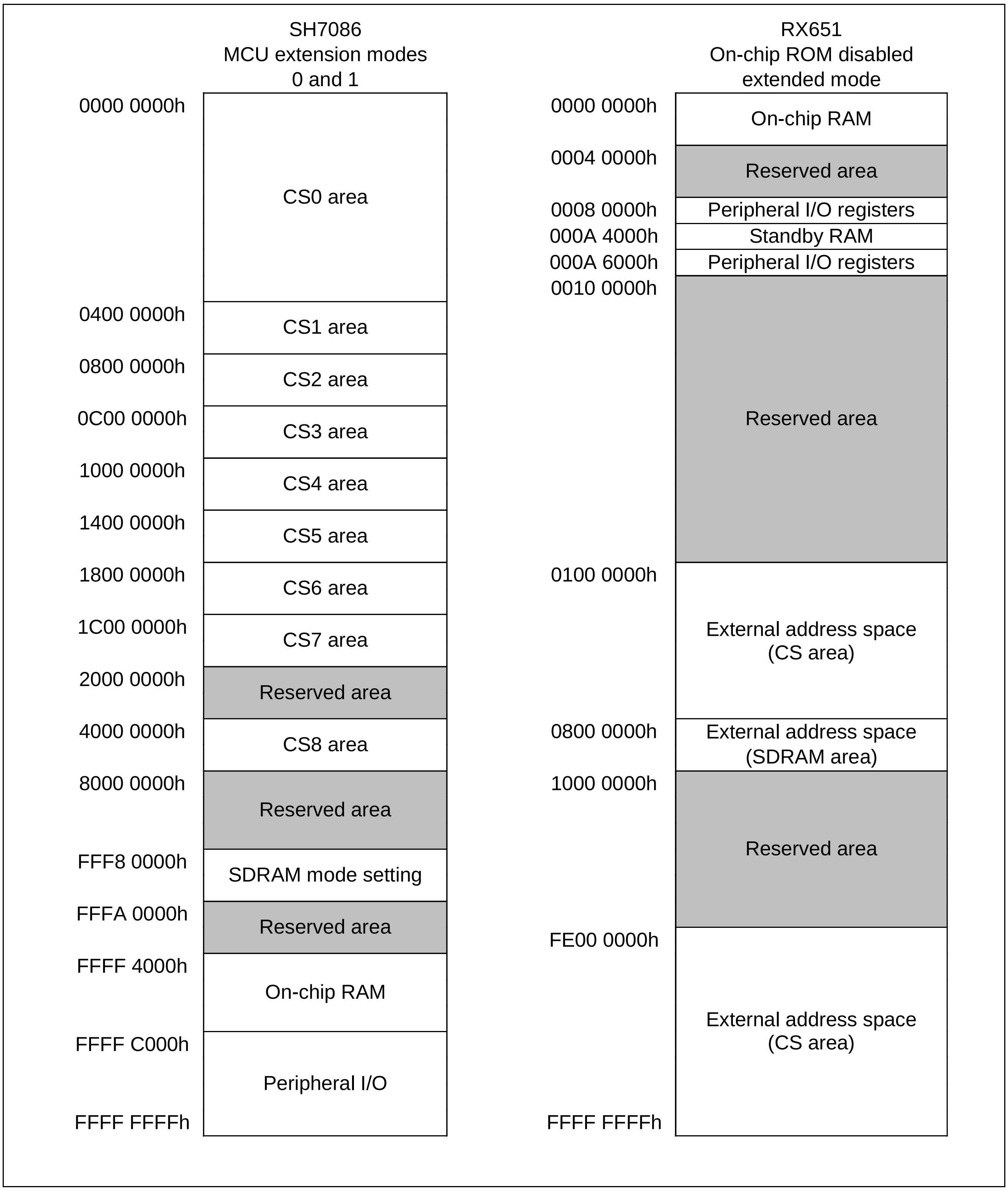


Figure 1.14 Memory Map Comparison (On-Chip ROM Disabled Mode)

- On the RX651 the RAM is allocated to addresses adjacent to 0000 0000h and ROM (for reading data) to addresses adjacent to FFFF FFFFh.
- On the RX651 the peripheral I/O registers are allocated within the address range from 0008 0000h to 000F FFFFh, and only the flash-related registers are allocated within the address range from 007F C000h to 007F FFFFh.
- On the RX651 the external address space is allocated within the address ranges from 0100 0000h to 0FFF FFFFh and FF00 0000h to FFFF FFFFh, and configured as eight CS spaces of 16 MB each and a 128 MB SDRAM space.

1.5.3 Operation Mode Settings

Whereas on the SH7080 Group operation mode settings are made only with the MD1, MD0, and FWE pins, on the RX651 operation mode settings can be made by means of the MD and UB pins when a reset is canceled, or by software after a reset is canceled.

Table 1.7 lists the operation modes that are determined by pin settings, and Table 1.8 lists the operation modes that are set in software after a reset is canceled.

Table 1.7 Pin Settings and Operation Modes on RX651

Pin		
MD	**UB**	**Mode Name**
High	—	Single-chip mode
Low	Low	Boot mode (SCI interface)
	High	Boot mode (USB interface)
Low → High*1	Low	Boot mode (FINE interface)

Note 1. After canceling the reset by driving the MD pin low, switch it to high for a period of 20 to 100 msec.

Table 1.8 SYSCR0 Register Settings and Operation Modes on RX651

SYSCR0 Register		
ROME Bit*1	**EXBE Bit**	**Mode Name**
0 (on-chip ROM disabled)	0 (external bus disabled)	Single-chip mode
1 (on-chip ROM enabled)*2	0 (external bus disabled)*2	
0 (on-chip ROM disabled)	1 (external bus enabled)	On-chip ROM disabled extended mode
1 (on-chip ROM enabled)	1 (external bus enabled)	On-chip ROM enabled extended mode

Note 1. Once the ROME bit is cleared to 0 it cannot be set to 1 again.
Note 2. After the STSCR0 register is reset, ROME = 1 and EXBE = 0.

1.5.4 Write Protection

The RX651 has a register write protection function to protect important registers from being overwritten if program runaway occurs, and the operation mode–related registers are protected by this function.

If necessary, set protect bit 1 (PRCR.PRC1) to 1 to enable writes before writing to these registers.

1.6 Processor Modes

The RX CPU supports two processing modes: supervisor mode and user mode. These processor modes enable hierarchical CPU resource protection.

This makes it possible, when replacing the SH7080 Group with the RX651, to replace the software by operating in supervisor mode only, without using user mode. In other words, software can be replaced without the need to be conscious of the processor mode.

Table 1.9 RX651 Processor Modes

Processor Modes	Transition Conditions	Outline
Supervisor mode	• Reset cancellation • Exception occurrence (PSW.PM bit cleared to 0)	All CPU resources are accessible, and all instructions can be executed (no limitations). This is the mode in which the OS and other system programs ordinarily operate.
User mode	• PSW.PM bit set to 1 In this case, first set to 1 the PSW.PM bit saved to the stack, then execute the RTE instruction. Alternately, first set to 1 the PSW.PM bit saved to BPSW, then execute the RTFI instruction.	Write access to some CPU resources, such as some bits in PSW and to BPC and BPSW, is restricted, and privileged instructions cannot be used. This is the mode in which user programs such as application programs ordinarily operate.

Transitioning from supervisor mode to user mode

```
MVFC    PSW,R1        : The RTE instruction is used to simulate return from an exception.
OR      #00110000h,R1 :
PUSH.L  R1            :
MVFC    PC,R1         :
ADD     #10,R1        :
RTE
NOP
NOP
```

Figure 1.15 Processor Mode Setting Example (User Mode)

Transitioning from user mode to supervisor mode

Operation transitions to supervisor mode when exception handling occurs. Operation then transitions again to user mode after the return from exception handling.

Another way to cause a transition to supervisor mode is to use an instruction that generates an unconditional trap, such as the INT instruction or BRK instruction.

Figure 1.16 Processor Mode Setting Example (Supervisor Mode)

1.7 Exception Handling

The points of difference regarding exception handling in general on the SH7080 Group and RX651, including interrupts, are described below.

1.7.1 Types of Exception Handling

Table 1.10 shows a comparative listing of exception sources on the SH7080 Group and RX651.

Table 1.10 Exception Source Comparison

SH7080 Group	RX651	Main Points of Difference
Power-on reset	Reset	On the SH7080 Group there are separate vectors for power-on resets and manual resets. On the RX651 there is a single reset vector. The reset source is identified in reset status registers 0 to 2 during reset interrupt handling, and appropriate processing is performed.
Manual reset		
Address error	Access exception	On the SH7080 Group this exception occurs when an attempt is made to access an access-prohibited area or an address to which access is prohibited. On the RX651 this exception occurs when a memory protection error occurs.
Interrupt (NMI)	Non-maskable interrupt	None
Interrupt (external/internal)	Interrupt (external/internal)	The RX651 also supports fast interrupts (level 15)
TRAP instruction (TRAPA instruction)	Unconditional trap (INT, BRK instruction)	The SH7080 Group has 32 sources, but the RX651 has 16 sources with dedicated vectors and up to 256 sources when sources also used for interrupts are included.
General illegal instruction	Undefined instruction	The SH7080 Group has no exceptions corresponding to the privileged instruction exception or floating-point exception.
Illegal slot instruction		
—	Privileged instruction	
—	Floating-point exception	

1.7.2 Exception Handling Priority

Table 1.11 shows the comparative priority of exception sources on the SH7080 Group and the RX651.

Table 1.11 Exception Event Priority

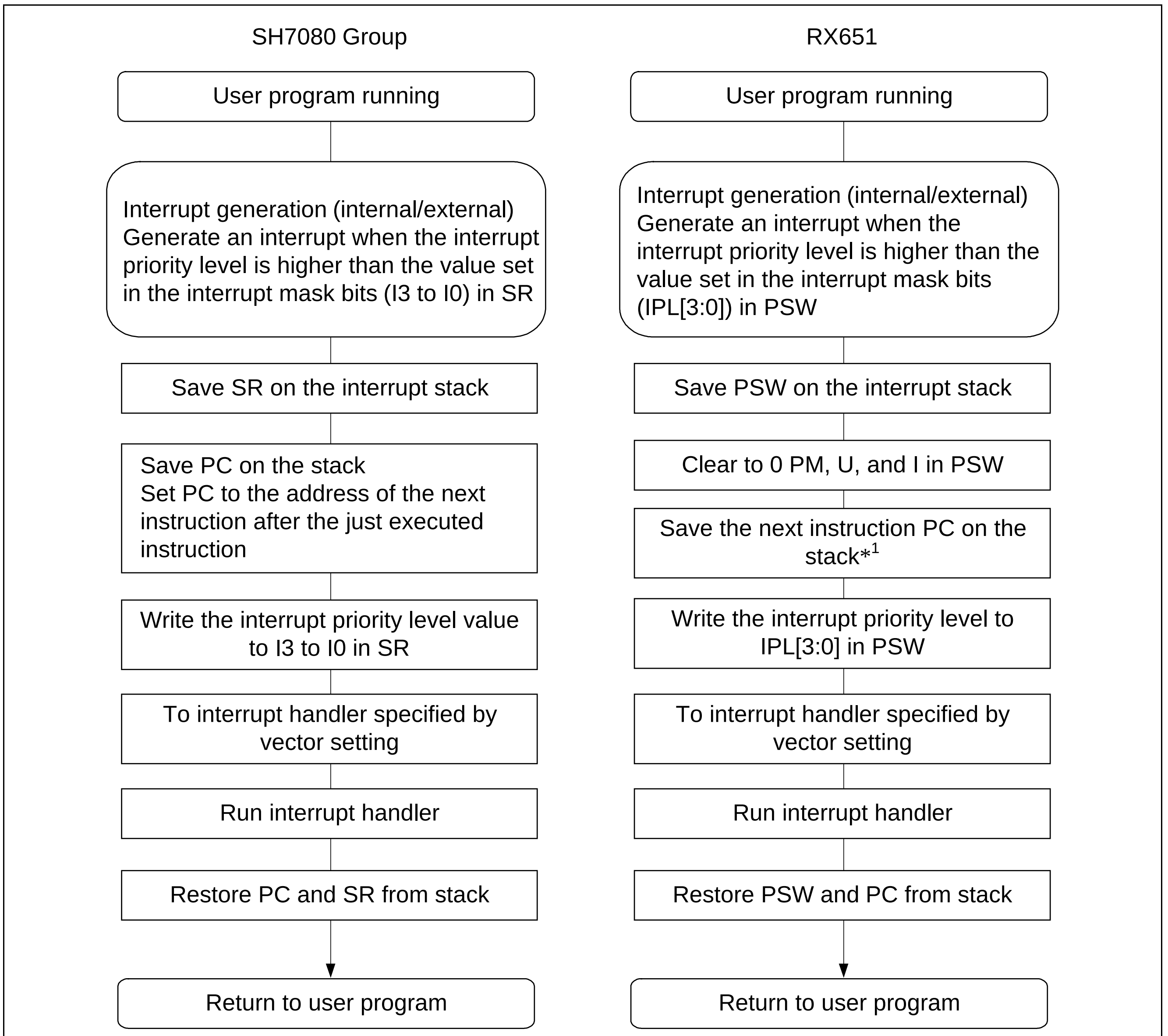
Priority*1	SH7080 Group	RX651
High	Reset	Reset
↑	Interrupt (break before instruction execution)	Non-maskable interrupt
	Address error (instruction fetch)	Interrupt
	Instruction	Instruction access exception
	Address error (CPU data access)	Undefined instruction exception, privileged instruction exception
	Interrupt (break after instruction execution, or operand break)	Unconditional trap
	Address error (DMAC/DTC data access)	Operand access exception
Low	Interrupt (NMI, IRQ, or on-chip peripheral module)	Floating-point exception

Note 1. Among interrupts, the priority is determined by the interrupt controller.

On the SH7080 Group address errors have higher priority than interrupts (internal or external), but on the RX651 both instruction access exceptions and operand access exceptions have lower priority than interrupts.

1.7.3 Basic Processing Sequence of Exception Handling

Figure 1.17 is a flowchart of interrupt exception handling on the SH7080 Group and the RX651.


SH7080 Group
User program running
Interrupt generation (internal/external) Generate an interrupt when the interrupt priority level is higher than the value set in the interrupt mask bits (I3 to I0) in SR
Save SR on the interrupt stack
Save PC on the stack Set PC to the address of the next instruction after the just executed instruction
Write the interrupt priority level value to I3 to I0 in SR
To interrupt handler specified by vector setting
Run interrupt handler
Restore PC and SR from stack
Return to user program
RX651
User program running
Interrupt generation (internal/external) Generate an interrupt when the interrupt priority level is higher than the value set in the interrupt mask bits (IPL[3:0]) in PSW
Save PSW on the interrupt stack
Clear to 0 PM, U, and I in PSW
Save the next instruction PC on the stack*1
Write the interrupt priority level to IPL[3:0] in PSW
To interrupt handler specified by vector setting
Run interrupt handler
Restore PSW and PC from stack
Return to user program

Note 1. When the instruction being executed is RMPA, SCMPU, SMOVB, SMOVF, SMOVU, SSTR, SUNTIL, or SWHILE, the PC of the instruction being executed is saved on the stack.

Figure 1.17 Interrupt (Internal/External) Processing Sequence

1.7.4 Vector Configuration

Both the SH7080 Group and the RX651 have a relocatable vector configuration, which allows vector tables to be reallocated.

On the SH7080 Group VBR (the vector base register) specifies the start of the vector table. (Note that VBR is initialized to 0 after a reset, so it is not possible to change the reset vector.)

On the RX651 INTB (the interrupt table register) specifies the start of the interrupt vector table, and EXTB (the exception table register) specifies the start of the exception vector table. Relocatable interrupt and unconditional trap vectors are assigned in the interrupt vector table. System exceptions are assigned in the exception vector table. The RX651 has a fixed reset vector. Also, the fast interrupt vector address is set in the FINTV register.

Figure 1.18 shows the differences between the vector tables.

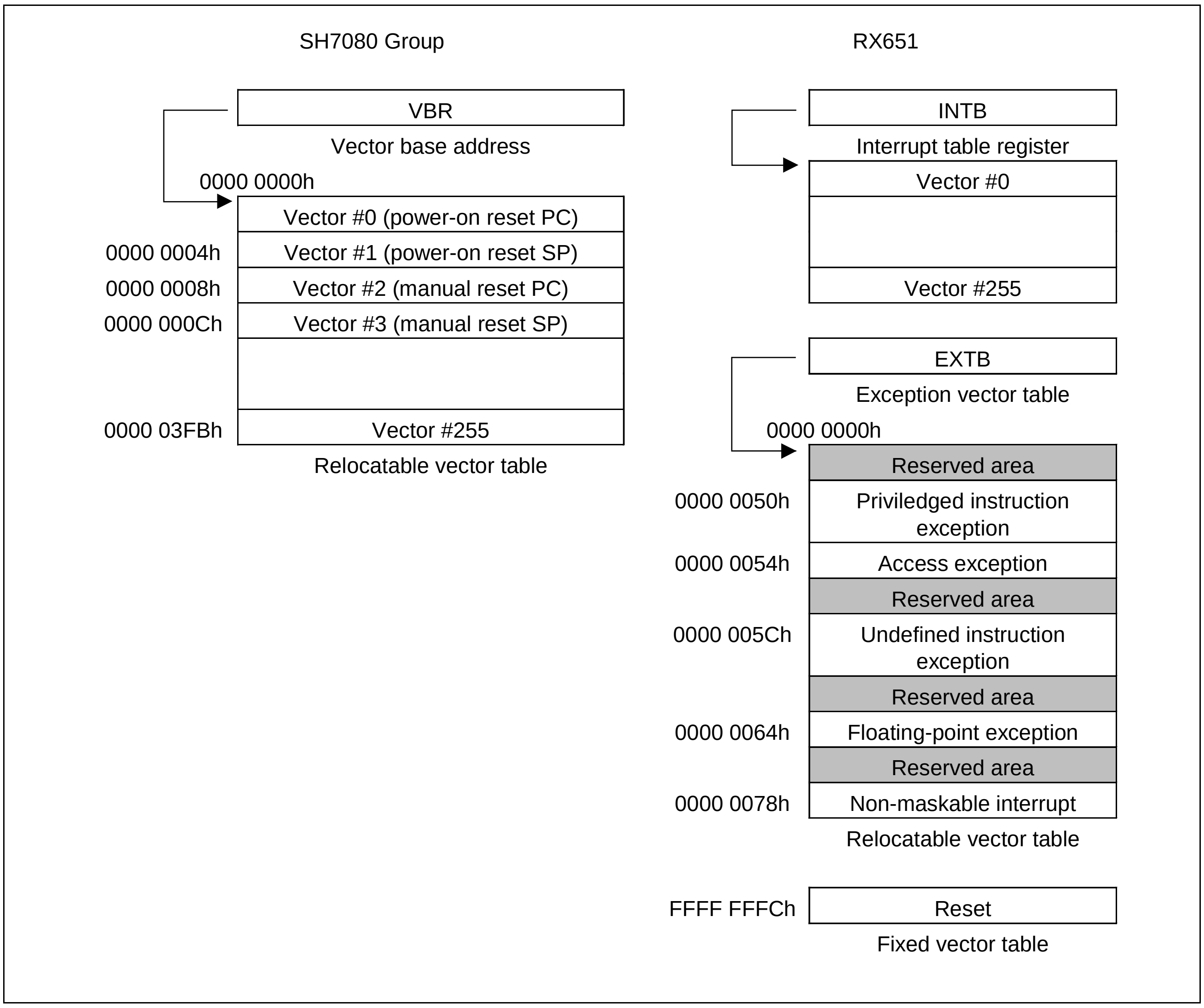


Figure 1.18 Vector Table Settings

1.7.5 Interrupt Masking by SR (SH7080 Group) and PSW (RX651)

On the RX651 the I bits in control register PSW are used to set the interrupt mask level. The I bits indicate which interrupts are enabled and which are disabled.

Table 1.12 Interrupt-Related Bits in SR and PSW

SH7080 Group	**RX651**	
SR Register	**PSW Register**	**Description**
I0, I1, I2, I3	IPL[3:0]	CPU interrupt mask level (priority level) Setting value: 0 to Fh (levels 0 to 15) When an interrupt request occurs, this level setting is compared with the priority level set for the individual interrupt source, and the interrupt is enabled if its level setting is higher than the mask level.
—	I	Interrupt enable bit 0: Interrupts are disabled. 1: Interrupts are enabled. When an interrupt occurs, the interrupt status flag in the interrupt controller is set to 1. After a system reset, this bit is set to 1, enabling acceptance of interrupts. When an exception is accepted, this bit is cleared to 0 and no interrupts are accepted while its value remains 0.

1.8 Interrupt Handling

This section describes the differences in interrupt handling between the SH7080 Group and RX651, with the focus on the interrupt controller.

1.8.1 Interrupt Controller

Table 1.13 lists the differences in the interrupt controller specifications.

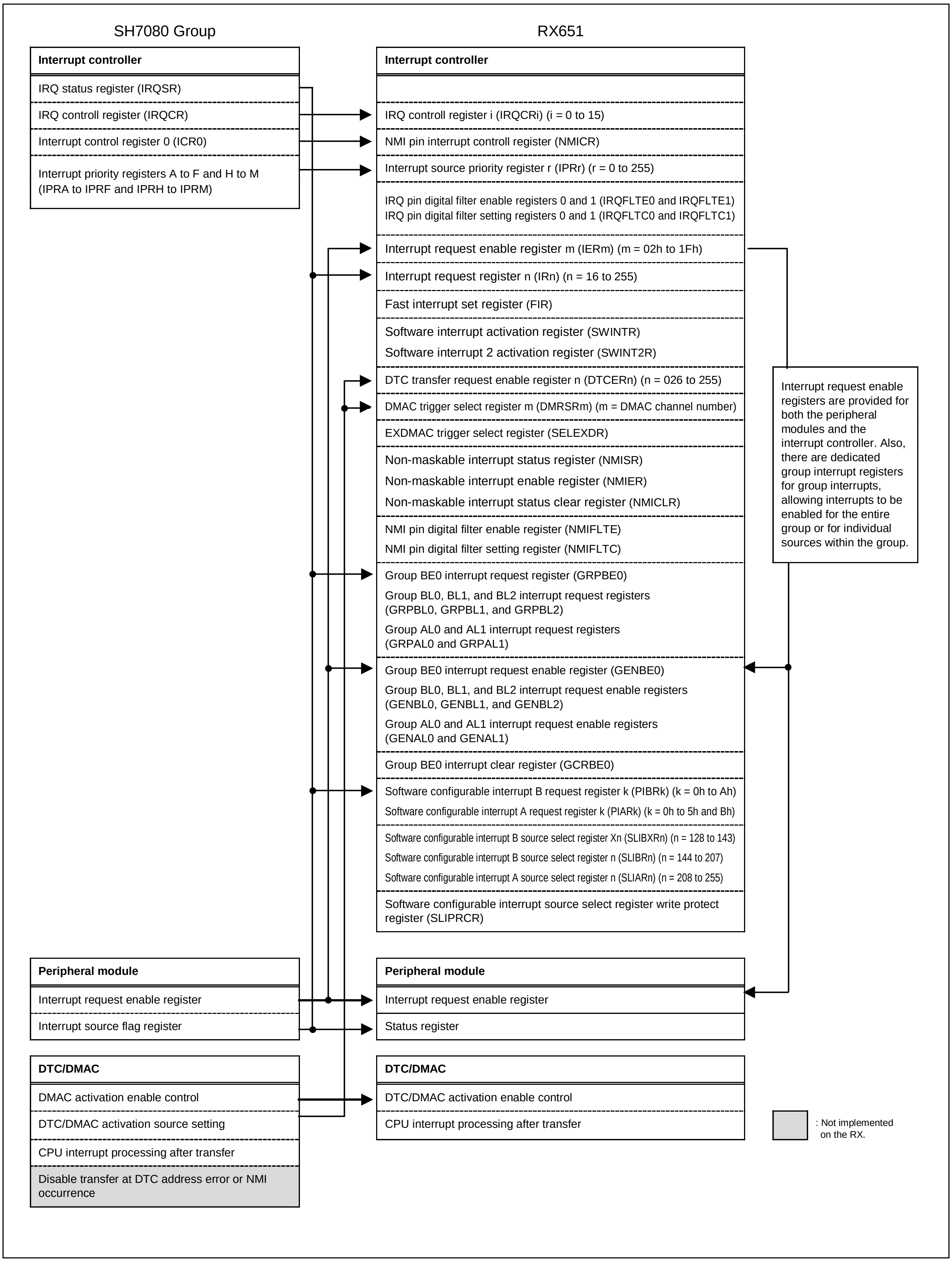
Table 1.13 Comparison of SH7080 Group and RX651 Specifications (Interrupt Controller)

Item		SH7080 Group	RX651
Interrupts	Peripheral function interrupts	• Interrupts from peripheral modules • Interrupt detection: Edge	• Interrupts from peripheral modules • Interrupt detection: Edge/level*[1] • Group interrupt function support
	External pin interrupts	• IRQ0 to IRQ7 pins • Sources: 8 • Interrupt detection: Low level, falling edge, rising edge, or both edges can be specified for each source.	• IRQ0 to IRQ15 pins • Sources: 16 • Interrupt detection: Low level, falling edge, rising edge, or both edges can be specified for each source. • Noise canceler function
	Noise cancellation	None	Digital filter settings are supported for the IRQi pins.
	Software interrupts	None	Supported
	Interrupt priority	A level from 0 to Fh can be specified for each source by a register setting.	A level from 0 to Fh can be specified for each source by a register setting.
	Fast interrupt function	None	Supported
	DTC/DMAC activation	DTC/DMAC activation supported*[2]	DTC/DMAC activation supported
	EXCMAC control	None	A software configurable interrupt can be used to start the EXDMAC.
Non-maskable interrupts	NMI pin interrupts	• Interrupt detection method (selection of falling or rising edge) • NMI input level read bit provided	• Interrupt detection method (selection of falling or rising edge) • Noise canceler function
	Other sources	• CPU address error • DMAC or DTC address error • TRAP instruction (TRAPA instruction) • General illegal instruction (undefined code) • Illegal slot instruction	• Interrupt at oscillation stop detection • WDT underflow or refresh error • IWDT underflow or refresh error • Voltage monitor 1 interrupt • Voltage monitor 2 interrupt • RAM error interrupt • Undefined instruction exception • Privileged instruction exception • Access exception • Floating-point exception • Unconditional trap
	Noise cancellation	Checking of the NMI input level can be used as a noise canceler function.	Digital filter settings are supported for the NMI pin.

Note 1. The detection method is fixed for fixed-connection peripheral modules.

Note 2. On the SH7080 Group activation source setting is performed on the DTC or DMAC.

Figure 1.19 shows the points of difference between the interrupt controller of the SH7080 Group and the RX651.



SH7080 Group
Interrupt controller
IRQ status register (IRQSR)
IRQ controll register (IRQCR)
Interrupt control register 0 (ICR0)
Interrupt priority registers A to F and H to M (IPRA to IPRF and IPRH to IPRM)
Peripheral module
Interrupt request enable register
Interrupt source flag register
DTC/DMAC
DMAC activation enable control
DTC/DMAC activation source setting
CPU interrupt processing after transfer
Disable transfer at DTC address error or NMI occurrence
RX651
Interrupt controller
IRQ controll register i (IRQCRi) (i = 0 to 15)
NMI pin interrupt controll register (NMICR)
Interrupt source priority register r (IPRr) (r = 0 to 255)
IRQ pin digital filter enable registers 0 and 1 (IRQFLTE0 and IRQFLTE1)
IRQ pin digital filter setting registers 0 and 1 (IRQFLTC0 and IRQFLTC1)
Interrupt request enable register m (IERm) (m = 02h to 1Fh)
Interrupt request register n (IRn) (n = 16 to 255)
Fast interrupt set register (FIR)
Software interrupt activation register (SWINTR)
Software interrupt 2 activation register (SWINT2R)
DTC transfer request enable register n (DTCERn) (n = 026 to 255)
DMAC trigger select register m (DMRSRm) (m = DMAC channel number)
EXDMAC trigger select register (SELEXDR)
Non-maskable interrupt status register (NMISR)
Non-maskable interrupt enable register (NMIER)
Non-maskable interrupt status clear register (NMICLR)
NMI pin digital filter enable register (NMIFLTE)
NMI pin digital filter setting register (NMIFLTC)
Group BE0 interrupt request register (GRPBE0)
Group BL0, BL1, and BL2 interrupt request registers (GRPBL0, GRPBL1, and GRPBL2)
Group AL0 and AL1 interrupt request registers (GRPAL0 and GRPAL1)
Group BE0 interrupt request enable register (GENBE0)
Group BL0, BL1, and BL2 interrupt request enable registers (GENBL0, GENBL1, and GENBL2)
Group AL0 and AL1 interrupt request enable registers (GENAL0 and GENAL1)
Group BE0 interrupt clear register (GCRBE0)
Software configurable interrupt B request register k (PIBRk) (k = 0h to Ah)
Software configurable interrupt A request register k (PIARk) (k = 0h to 5h and Bh)
Software configurable interrupt B source select register Xn (SLIBXRn) (n = 128 to 143)
Software configurable interrupt B source select register n (SLIBRn) (n = 144 to 207)
Software configurable interrupt A source select register n (SLIARn) (n = 208 to 255)
Software configurable interrupt source select register write protect register (SLIPRCR)
Peripheral module
Interrupt request enable register
Status register
DTC/DMAC
DTC/DMAC activation enable control
CPU interrupt processing after transfer
Interrupt request enable registers are provided for both the peripheral modules and the interrupt controller. Also, there are dedicated group interrupt registers for group interrupts, allowing interrupts to be enabled for the entire group or for individual sources within the group.
▨ : Not implemented on the RX.

Figure 1.19 Differences Between Interrupt Controller Registers

The interrupt controller of the SH7080 Group controls IRQ interrupt flags, while peripheral module interrupt flags are controlled by the peripheral modules.

On the RX651 the interrupt controller controls all interrupt status flags, for both IRQs and peripheral modules.*1 In addition, the interrupt controller controls the activation source settings for the DTC and DMAC. The disable transfer at NMI occurrence function of the DTC and DMAC on the SH7080 Group is not implemented on the RX651.

Note 1. The interrupt controller contains an interrupt request register for each interrupt source, but there are also interrupt enable bits implemented in the peripheral modules. (For details, see the User's Manual: Hardware.)

1.8.2 Interrupt Flag Management

When a peripheral module of the SH7080 Group generates an interrupt by edge detection, the corresponding interrupt source flag is cleared (the flag is cleared and a dummy read is performed) by the interrupt handler. This is done because the interrupt will be generated once again if the flag is not cleared by the handler.

On the RX651 the interrupt status flags are managed internally by the interrupt controller, and interrupt requests are sent to the CPU or DTC/DMAC. The interrupt controller has a function whereby, when edge detection is used, the corresponding interrupt status flag is cleared automatically when a response is received indicating acceptance of an interrupt. When level detection is used, both the request flag within the peripheral module and the corresponding interrupt status flag are cleared automatically. For details, see the User’s Manual: Hardware.

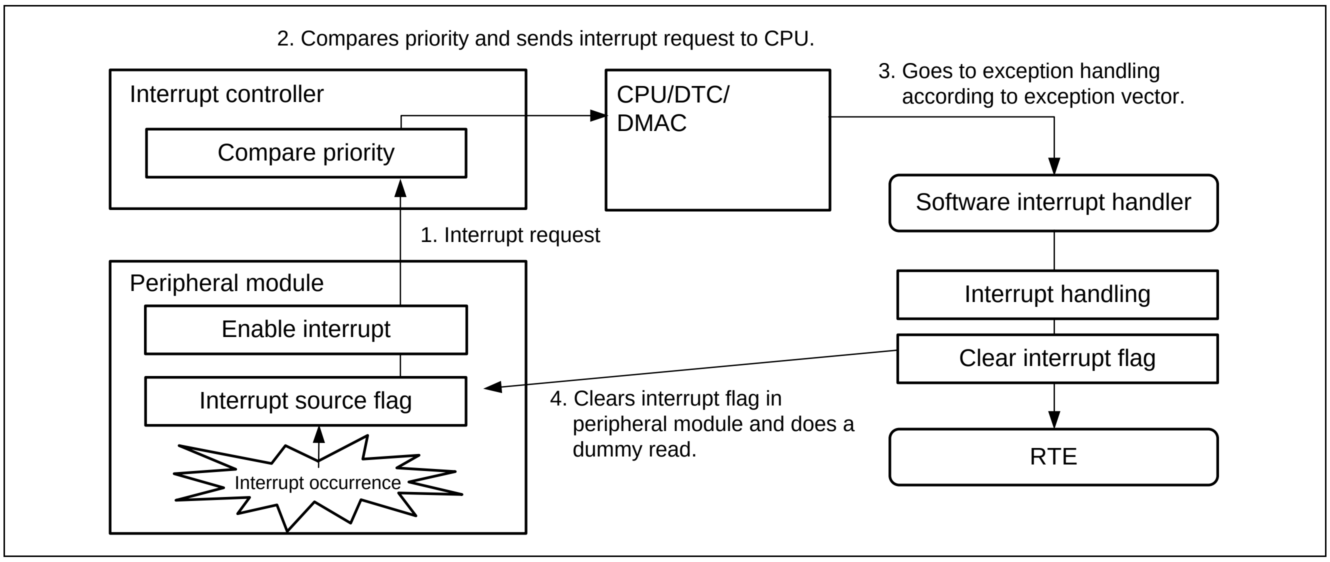

2. Compares priority and sends interrupt request to CPU.
Interrupt controller
Compare priority
CPU/DTC/ DMAC
3. Goes to exception handling according to exception vector.
Software interrupt handler
1. Interrupt request
Peripheral module
Enable interrupt
Interrupt source flag
Interrupt occurrence
Interrupt handling
Clear interrupt flag
4. Clears interrupt flag in peripheral module and does a dummy read.
RTE

Figure 1.20 SH7080 Group Peripheral Module Interrupt (Edge Detection)

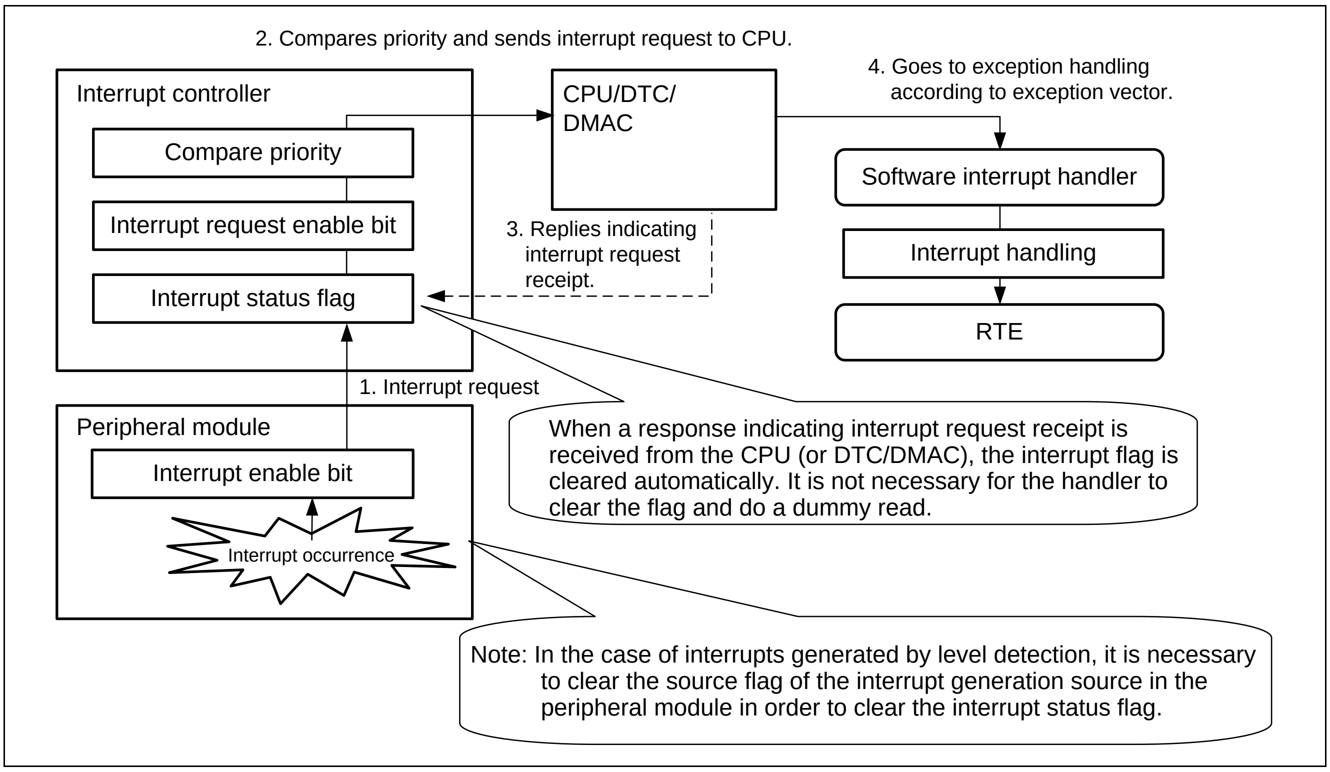

2. Compares priority and sends interrupt request to CPU.
Interrupt controller
Compare priority
Interrupt request enable bit
Interrupt status flag
CPU/DTC/ DMAC
4. Goes to exception handling according to exception vector.
Software interrupt handler
Interrupt handling
RTE
3. Replies indicating interrupt request receipt.
1. Interrupt request
Peripheral module
Interrupt enable bit
Interrupt occurrence
When a response indicating interrupt request receipt is received from the CPU (or DTC/DMAC), the interrupt flag is cleared automatically. It is not necessary for the handler to clear the flag and do a dummy read.
Note: In the case of interrupts generated by level detection, it is necessary to clear the source flag of the interrupt generation source in the peripheral module in order to clear the interrupt status flag.

Figure 1.21 RX651 Peripheral Module Interrupt (Edge Detection)

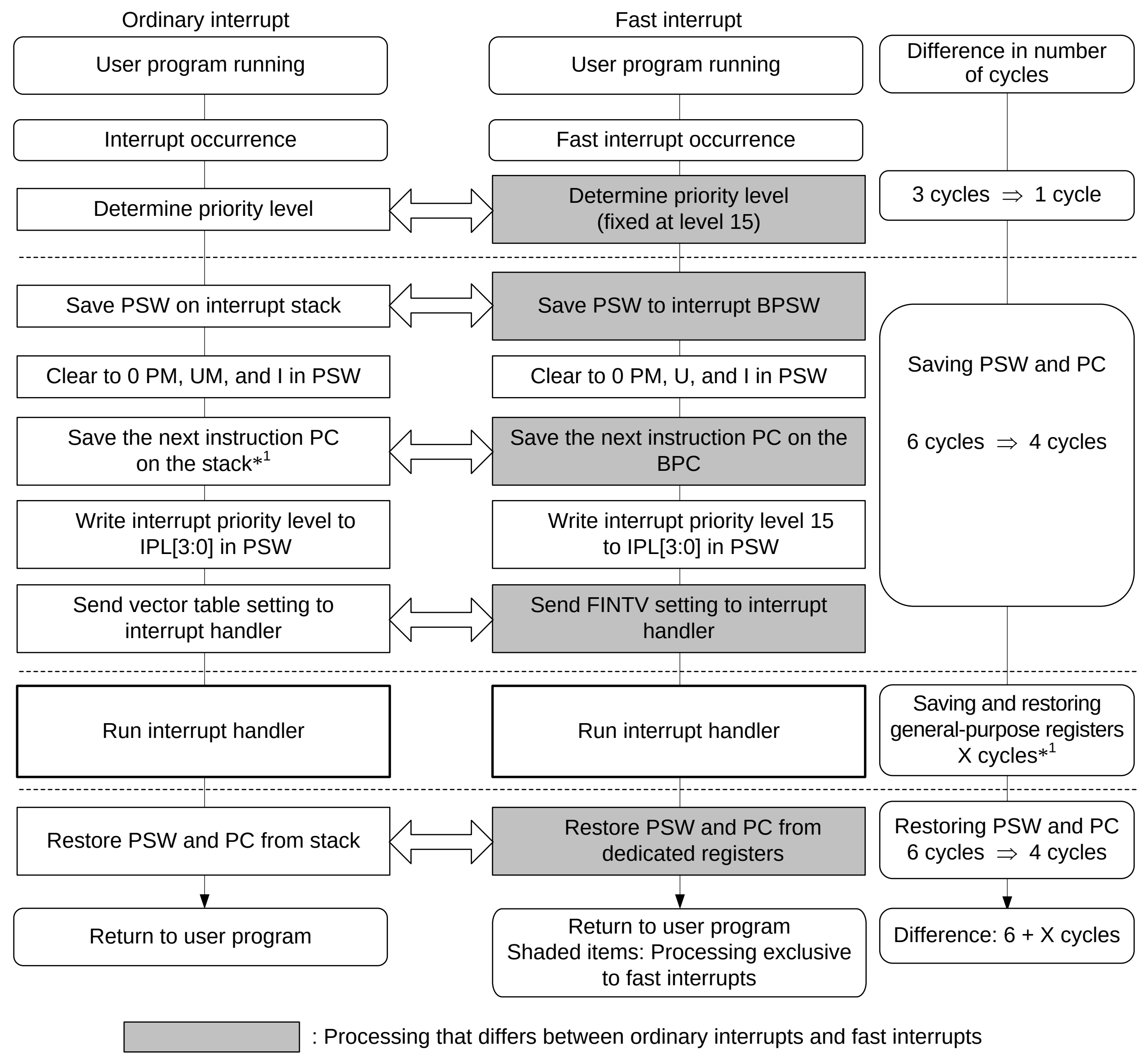
1.8.3 Fast Interrupt Control

In addition to ordinary interrupts, the RX651 supports fast interrupts.

Ordinary interrupt: After determining the interrupt priority it is necessary to save the contents of the control registers and general-purpose registers to the internal RAM or the external RAM by software.

Fast interrupt: Operation gives the interrupt the highest priority. When the interrupt occurs, the contents of the control registers are saved to dedicated registers, allowing interrupt activation to be realized faster than an ordinary interrupt.

It is possible to assign a portion of the general-purpose registers to exclusive use for interrupts by setting a compiler option. This eliminates the need to save and restore the contents of the general-purpose registers, further speeding up the interrupt.


Ordinary interrupt
User program running
Interrupt occurrence
Determine priority level
Save PSW on interrupt stack
Clear to 0 PM, UM, and I in PSW
Save the next instruction PC on the stack*1
Write interrupt priority level to IPL[3:0] in PSW
Send vector table setting to interrupt handler
Run interrupt handler
Restore PSW and PC from stack
Return to user program
Fast interrupt
User program running
Fast interrupt occurrence
Determine priority level (fixed at level 15)
Save PSW to interrupt BPSW
Clear to 0 PM, U, and I in PSW
Save the next instruction PC on the BPC
Write interrupt priority level 15 to IPL[3:0] in PSW
Send FINTV setting to interrupt handler
Run interrupt handler
Restore PSW and PC from dedicated registers
Return to user program Shaded items: Processing exclusive to fast interrupts
Difference in number of cycles
3 cycles ⇒ 1 cycle
Saving PSW and PC
6 cycles ⇒ 4 cycles
Saving and restoring general-purpose registers X cycles*1
Restoring PSW and PC 6 cycles ⇒ 4 cycles
Difference: 6 + X cycles
: Processing that differs between ordinary interrupts and fast interrupts

Note 1. Fast interrupts have a single save register stage. Therefore, multistage fast interrupts are not possible. It is necessary to decide on a single function that will use fast interrupts.

Figure 1.22 Differences Between Ordinary Interrupts and Fast Interrupts on the RX651

1.8.4 Noise Cancellation

The SH7080 Group has an NMI level bit (ICR0.NMIL) that indicates the state of the NMI pin. An interrupt handler service routine can check the pin state by reading this bit, and this capability can be used as a noise canceler function.

The RX651 is provided with a digital filter function for signals input on the IRQi pins and NMI pin. The sampling clock for the digital filter can be specified, and interrupt signals that do not last for at least three cycles of the sampling clock base are not accepted as interrupts. This improves the system's noise tolerance.

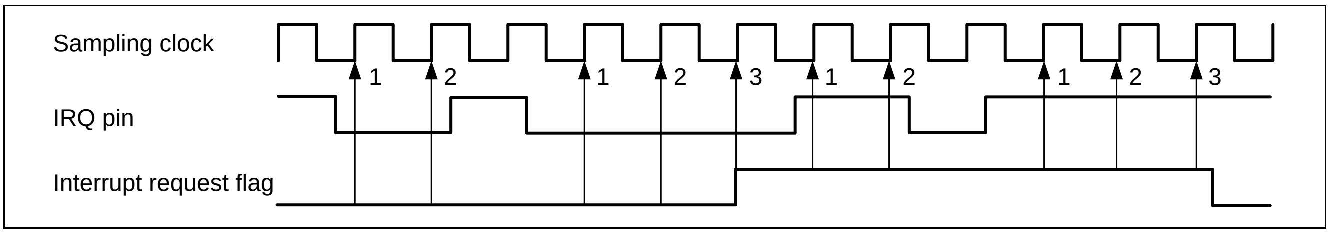


Figure 1.23 RX651 Digital Filter Operation Example

1.8.5 Multiple Interrupts

On the SH7080 Group if a high-priority interrupt occurs while a low-priority interrupt handler is running, the low-priority interrupt handler is suspended and the high-priority interrupt handler is executed. Once the high-priority interrupt handler finishes, the suspended low-priority interrupt handler is restarted.

On the RX651 if a high-priority interrupt occurs while a low-priority interrupt handler is running, the high-priority interrupt is not accepted until the low-priority interrupt handler finishes. This is because the PSW.I bit is cleared to 0 (interrupts are disabled) in a normal interrupt handler. In order to realize handling of multiple interrupts equivalent to that of the SH7080 Group, it is necessary to set the PSW.I bit to 1 (interrupts are enabled) in the interrupt handler.

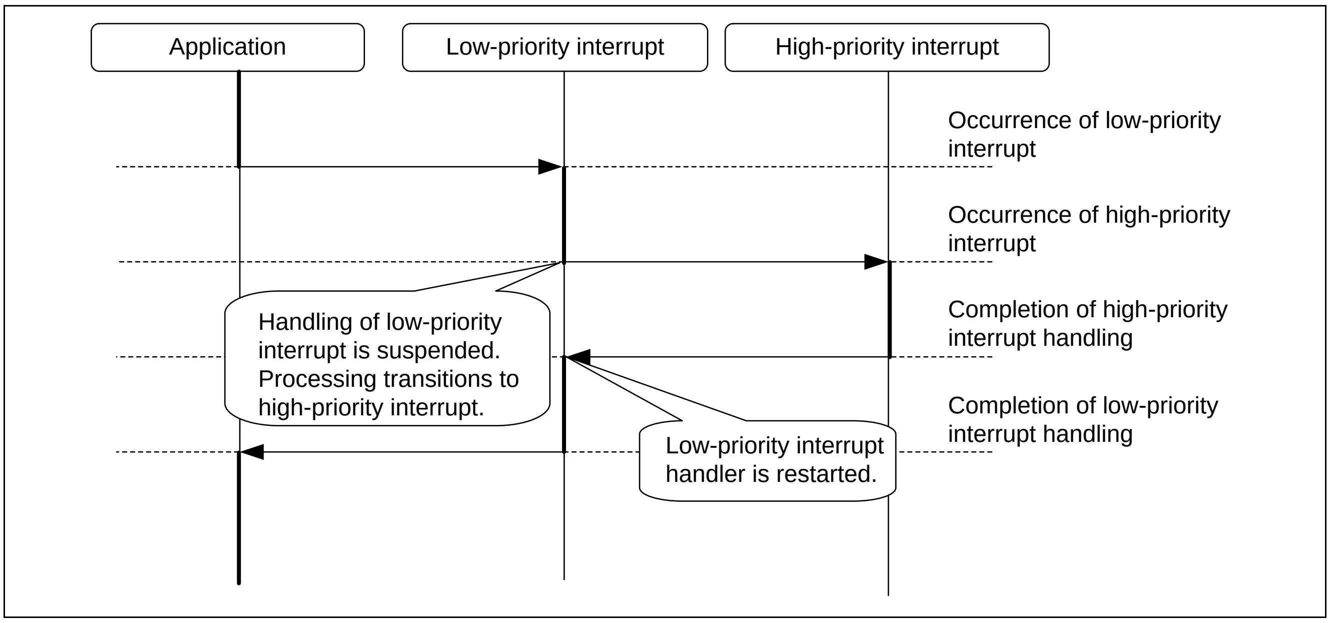

Application
Low-priority interrupt
High-priority interrupt
Occurrence of low-priority interrupt
Occurrence of high-priority interrupt
Completion of high-priority interrupt handling
Completion of low-priority interrupt handling
Handling of low-priority interrupt is suspended. Processing transitions to high-priority interrupt.
Low-priority interrupt handler is restarted.

Figure 1.24 SH7080 Group Multiple Interrupt Sequence

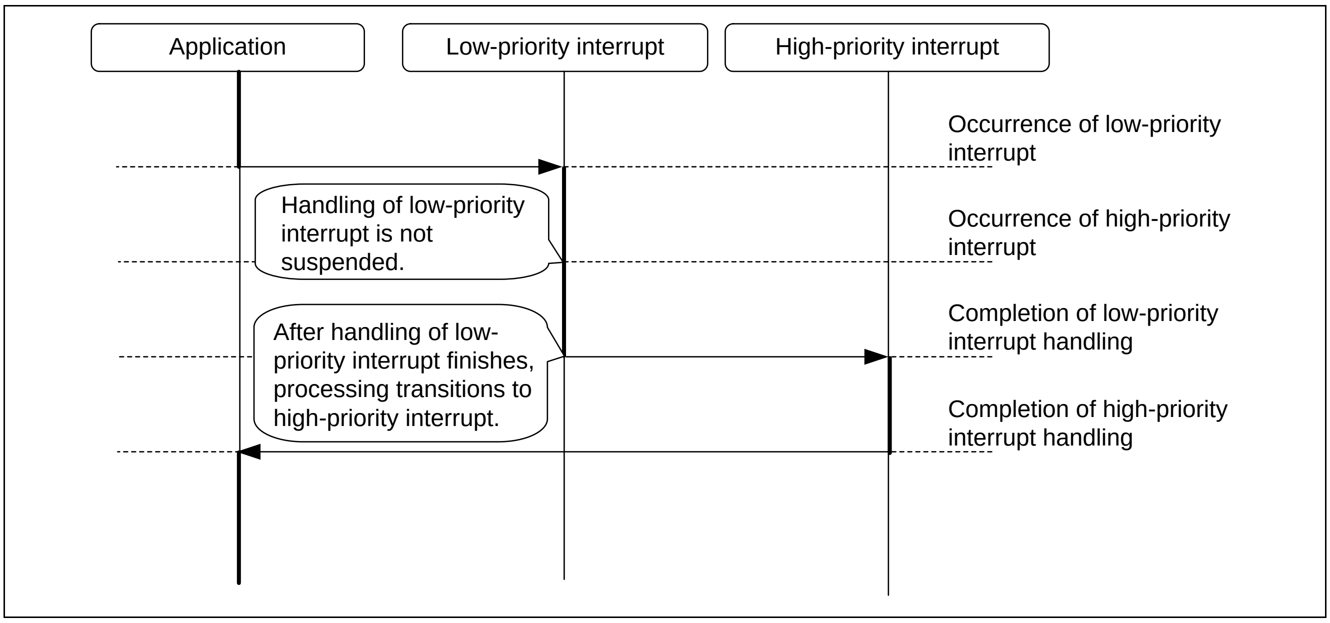

Application
Low-priority interrupt
High-priority interrupt
Occurrence of low-priority interrupt
Handling of low-priority interrupt is not suspended.
Occurrence of high-priority interrupt
After handling of low-priority interrupt finishes, processing transitions to high-priority interrupt.
Completion of low-priority interrupt handling
Completion of high-priority interrupt handling

Figure 1.25 RX651 Interrupt Sequence (Not Controlled by PSW.I Bit)

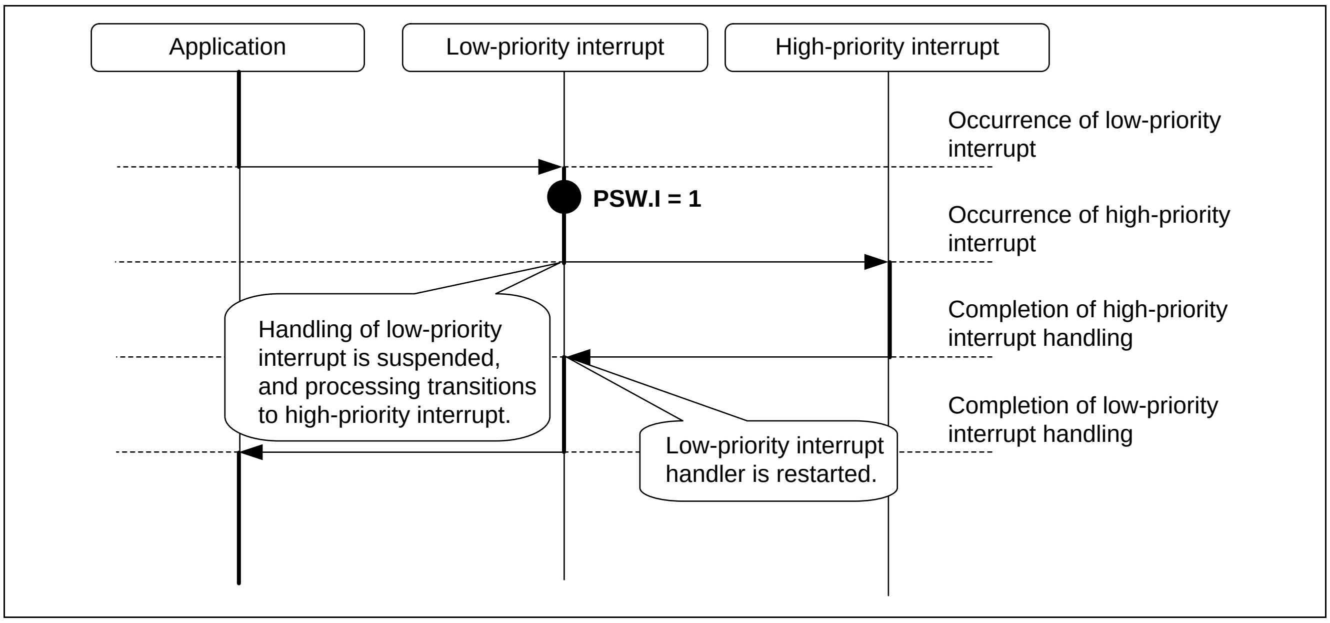

Application
Low-priority interrupt
High-priority interrupt
Occurrence of low-priority interrupt
PSW.I = 1
Occurrence of high-priority interrupt
Handling of low-priority interrupt is suspended, and processing transitions to high-priority interrupt.
Completion of high-priority interrupt handling
Low-priority interrupt handler is restarted.
Completion of low-priority interrupt handling

Figure 1.26 RX651 Interrupt Sequence (Controlled by PSW.I Bit)

1.8.6 Group Interrupts

Group interrupts allow multiple interrupt sources to be assigned to a single vector. Group interrupt detection is by means of a logical OR operation on all the interrupt requests assigned to the group. This means that when an interrupt request is detected, it is necessary to identify the interrupt request from among those in the group by means of software.

Interrupt sources are assigned to different groups according to the operating clock of the peripheral module and the interrupt detection method.

The clearing condition for each group interrupt status flag differs according to the interrupt detection method. Table 1.14 lists the types of group interrupts and the clearing conditions of their status flags.

Table 1.14 RX651 Group Interrupt Types

Group	Peripheral Module Operating Clock	Interrupt Detection Method	Group Interrupt Status Flag
Group BE0	PCLKB	Edge detection	Cleared automatically when 1 is written to the corresponding interrupt source clear bit (GCRBE0.CLRn) of the interrupt controller.
Group BL0		Level detection	Cleared automatically when the peripheral module's interrupt status flag is cleared.
Group BL1			
Group BL2			
Group AL0	PCLKA		
Group AL1			

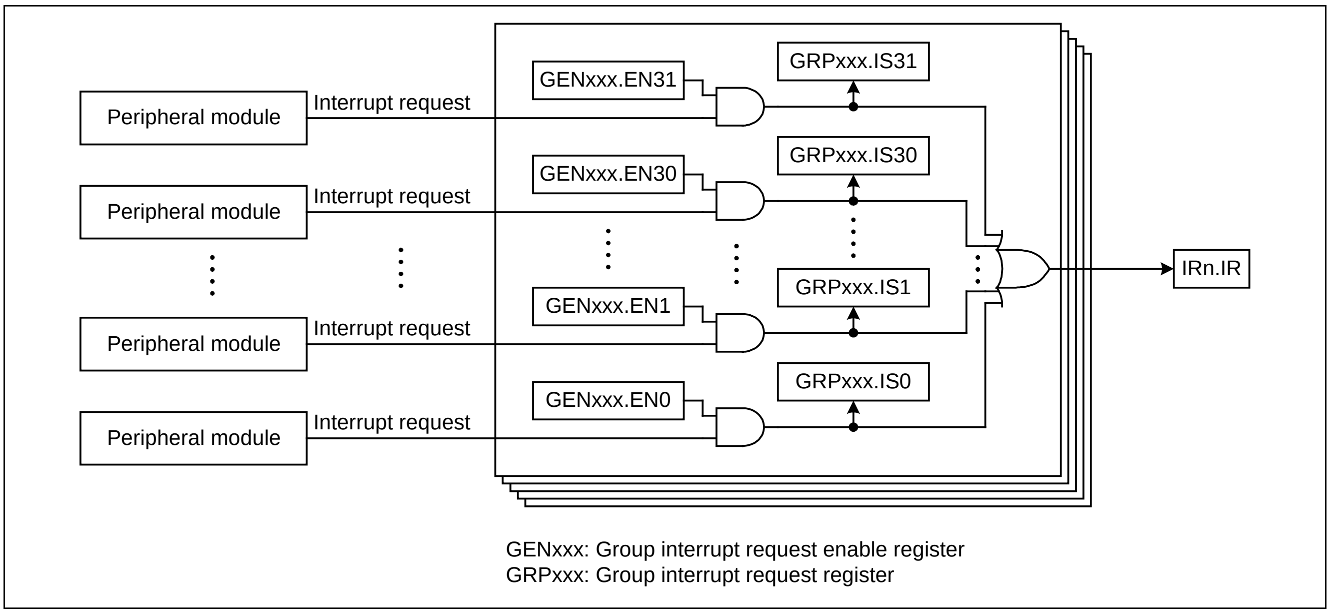


Figure 1.27 Group Interrupt Configuration on the RX651

1.8.7 Software Configurable Interrupts

A single interrupt source among multiple peripheral modules can be selected for each software configurable interrupt, which is then assigned an interrupt vector number from 128 to 255.

Software configurable interrupts are classified into two types, A and B, according to the peripheral module operating clock. Table 1.15 lists the types of software configurable interrupts.

The software configurable interrupt status flags are not cleared automatically, but there is no effect on the generation of interrupt requests even if the corresponding flags are not cleared.

Table 1.15 Types of Software Configurable Interrupts on the RX651

Software Configurable Interrupt Name	Peripheral Module Operating Clock	Interrupt Detection Method	Software Configurable Interrupt Status Flag
Software Configurable Interrupt A	PCLKA	Edge detection	Not cleared automatically, but there is no effect on interrupt request generation even if the flag is not cleared.
Software Configurable Interrupt B	PCLKB		

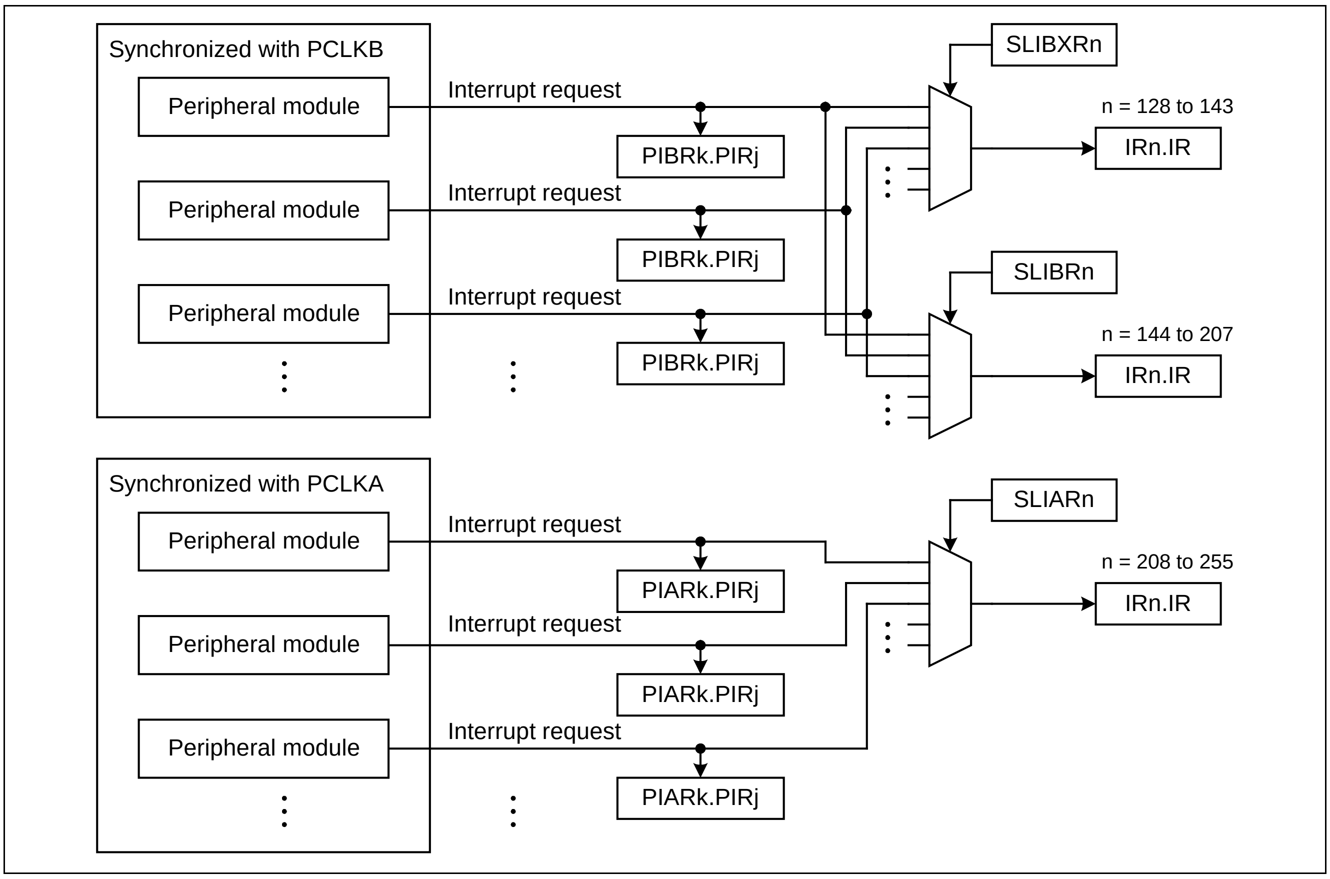


Figure 1.28 Software Configurable Interrupt Configuration on the RX651

2. On-Chip Functions

2.1 List of On-Chip Functions

Table 2.1 lists the on-chip functions of the SH7080 Group and RX651.

Table 2.1 On-Chip Functions

SH7080 Group	RX651
Clock oscillator (CPG)	Clock generation circuit
Interrupt controller (INTC)	Interrupt controller (ICUB)
User break controller (UBC)	—
Data transfer controller (DTC)	Data transfer controller (DTCb)
Bus state controller (BSC)	Bus
Direct memory access controller (DMAC)	DMA controller (DMACAa) EXDMA controller (EXDMACa)
Multi-function timer pulse unit 2 (MTU2) Multi-function timer pulse unit 2S (MTU2S)	Multi-function timer pulse unit 3 (MTU3a)
Port output enable (POE)	Port output enable 3 (POE3a)
Watchdog timer (WDT)	Watchdog timer (WDTA) Independent watchdog timer (IWDTa)
Serial communication interface (SCI) Serial communication interface with FIFO (SCIF)	Serial communication interface (SCIg, SCIi, SCIh)
Synchronous serial communication unit (SSU)	Serial peripheral interface (RSPIc)
I^2C bus interface 2 (IIC2)	I^2C bus interface (RIICa)
A/D converter (ADC)	12-bit A/D converter (S12ADFa)
Compare match timer (CMT)	Compare match timer (CMT) Compare match timer W (CMTW)
Pin function controller (PFC)	Multi-function pin function controller (MPC)
I/O port	I/O port
Flash memory*[1]	Flash memory*[2]
RAM (max. 32 KB)	RAM (max. 256 KB)
Low power consumption mode	Low power consumption function

SH7080 Group	RX651
—	Voltage detection circuit (LVDA) Clock frequency accuracy measurement circuit (CAC) Low power consumption function Battery backup function Register write protection function Memory protection unit (MPU) Event link controller (ELC) 16-bit timer pulse unit (TPUa) Programmable pulse generator (PPG) 8-bit timer (TMR) Realtime clock (RTCd) USB 2.0 FS Host/Function module (USBb) CAN module (CAN) Quad serial peripheral interface (QSPI) CRC calculator (CRCA) SD host interface (SDHI) SD slave interface (SDSI) MultiMediaCard interface (MMCIF) Parallel data capture unit (PDC) Boundary scan AESa RNGa 12-bit D/A converter (R12DA) Temperature sensor (TEMPS) Data operation circuit (DOC)

Note 1. Some versions of the SH7080 Group have on-chip mask ROM.

Note 2. The RX651 Group has up to 1 MB of on-chip flash memory (ROM) for storing code.
For details, see the User's Manual: Hardware.

2.2 I/O Ports/Pin Function Controller (PFC)

2.2.1 Number of I/O Ports

Table 2.2 lists the number of I/O ports on the SH7080 Group and RX651.

Table 2.2 Number of I/O Ports

Item	Package	Port Function
Number of I/O ports on SH7080 Group	TQFP1414-100 (SH7083)	I/O: 65 Input: 8 Total: 73
	LQFP2020-112 (SH7084)	I/O: 76 Input: 8 Total: 84
	LQFP2020-144 (SH7085)	I/O: 100 Input: 8 Total: 108
	LQFP2424-176 (SH7086)	I/O: 118 Input: 16 Total: 134
Number of I/O ports on RX651	TFLGA-145 LFQFP-144	I/O: 111 Input: 1 Pull-up resistor: 111 Open-drain output: 111 5 V tolerant: 18
	TFLGA-100 LFQFP-100	I/O: 78 Input: 1 Pull-up resistor: 78 Open-drain output: 78 5 V tolerant: 17

2.2.2 I/O Settings

Both the SH7080 Group and RX651 have multiplexed pins. Therefore, it is necessary to make pin settings to assign each pin to either general I/O or an on-chip module function.

On the SH7080 Group port functions are determined by settings made to the pin function controller (PFC). The I/O ports are configured as ports A to F.

The SH7080 Group's I/O port register settings are shown in Figure 2.1, the I/O port register configuration in Table 2.3, and the pin function controller (PFC) register configuration in Table 2.4.

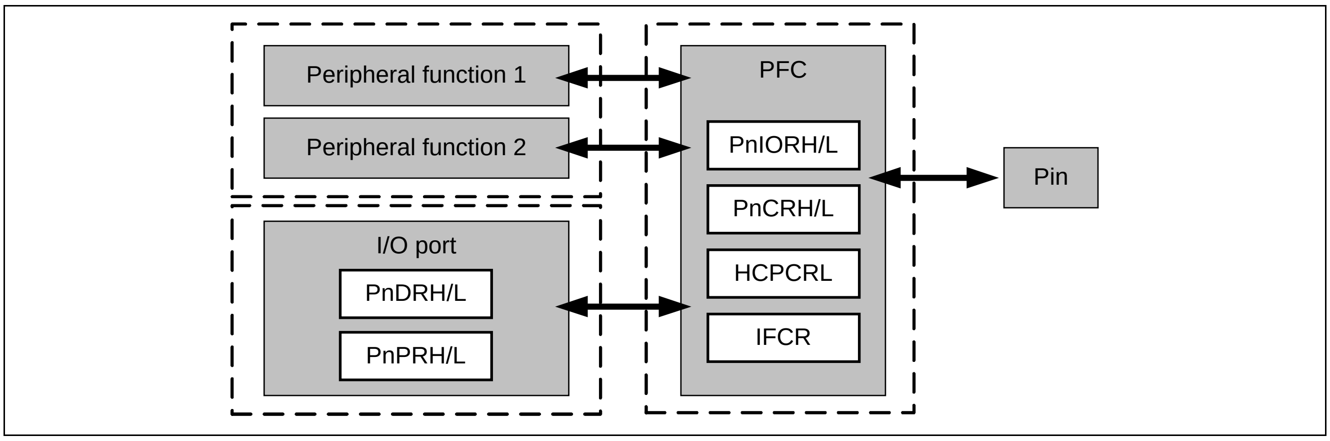


Figure 2.1 SH7080 Group I/O Settings

Table 2.3 SH7080 Group Register Configuration (I/O Ports)

Register	Function Name	Function
PnDRH PnDRL	Port n data register H Port n data register L	Port n data registers They store pin output data.
PnPRH PnPRL	Port n Port register H Port n Port register L	Port n data read-only registers They reflect pin states.

n: Port name (n = A to E)

Table 2.4 SH7080 Group Register Configuration (PFC)

Register	Function Name	Function
PnIORH PnIORL	Port n IO register H Port n IO register L	Pin input/output direction selection They specify whether ports operate as input or output.
PnCRHm PnCRLm	Port n control register Hm Port n control register Lm	Multiplexed pin function selection
HCPCR	High-current port control register	Sets the state of high-current ports.
IFCR	IRQOUT function control register	Sets the state of IRQ output pin.

n: Port name (n = A to E)
m: Setting number (m = 1 to 4)

Note that the functions that can be assigned to pins and the functions that can be specified by the PFC differ according to the SH7080 Group's operation mode (microcontroller mode 0, 1, or 2, or single-chip mode).

On the RX651 port functions are specified by making settings to the multi-function pin controller (MPC). The I/O ports are configured as ports 0 to 9, A to F, and J.

Unlike the SH7080 Group, where registers are provided for each port, on the RX651 registers are provided for each pin for selection of pin functions.

The following types of I/O port settings are supported on the RX651.

- Open drain control register: Port output format selection
  CMOS output, N-channel open-drain output, or P-channel open-drain output
- Pull-up control register: Input pull-up resistor on/off selection
- Drive capacity control register: Selection between normal drive output and high drive output
- 5 V tolerant input ports are provided.

The RX651's I/O port register settings are shown in Figure 2.2, the I/O port register configuration in Table 2.5, and the multi-function pin controller (MPC) register configuration in Table 2.6.

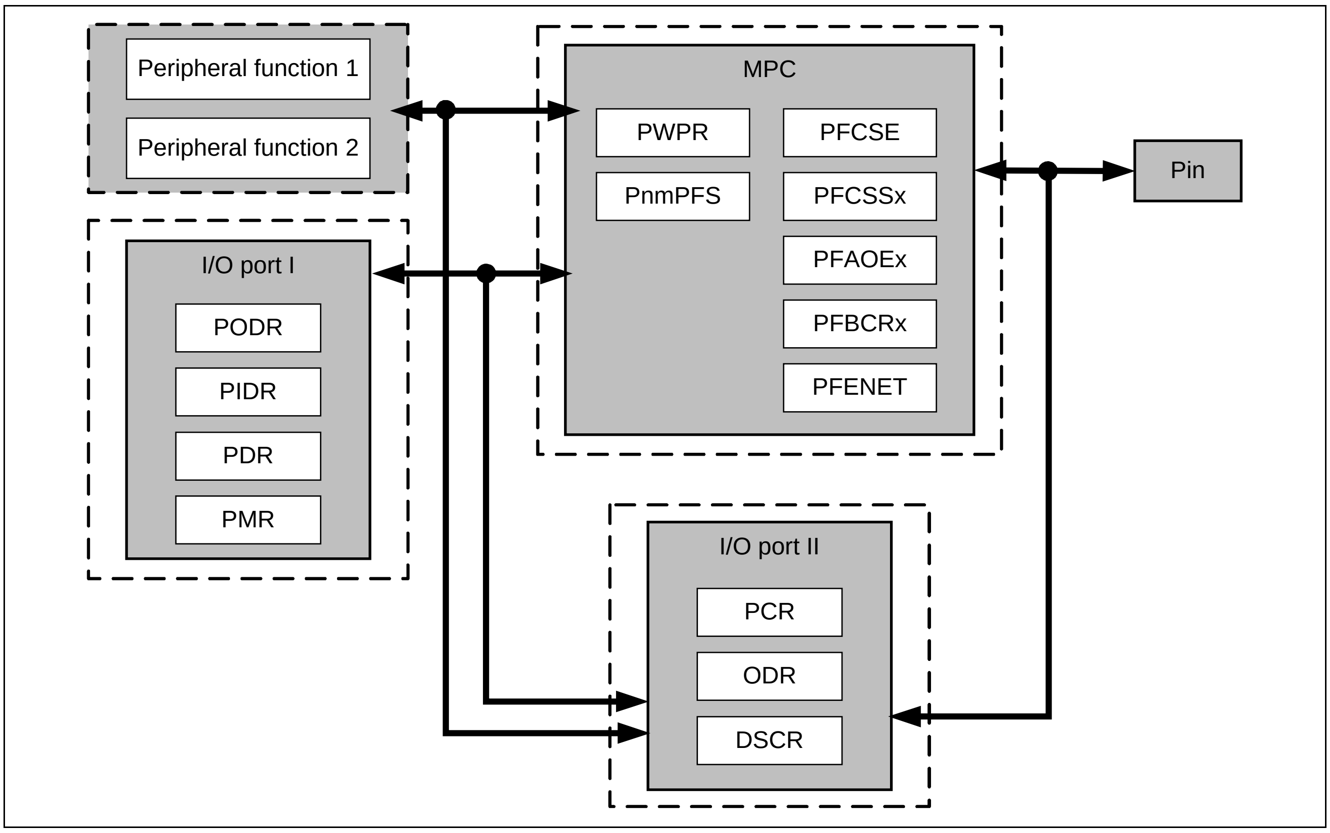


Figure 2.2 I/O Settings on the RX651

To use a pin as a general I/O pin it is sufficient to make a setting in the appropriate I/O port register. Figure 2.3 shows the initialization sequence for using pins as general I/O pins on the RX651.

The pin function control registers (PnmPFS) of the MPC are used to assign peripheral functions to pins. For setting examples when using peripheral functions that include general I/O, refer to the individual chapters for each of the peripheral functions. Figure 2.4 shows the initialization sequence for assigning pins to peripheral functions on the RX651.

Table 2.5 RX651 Register Configuration (I/O Ports)

Register	Function Name	Function
PDR	Port direction register	Specifies input or output for pins selected as general I/O ports.
PODR	Port output register	Stores pin output data for general output ports.
PIDR	Port input register	Reflects pin states for general input ports.
PMR	Port mode register	Used for port pin function settings. Specifies whether each pin is used as a general I/O port or for a peripheral function.
ODR0	Open drain control register 0	Selects the port output format from among the following: • CMOS output • N-channel open drain • P-channel open drain
ODR1	Open drain control register 1	Selects the port output format from among the following: • CMOS output • N-channel open drain
PCR	Pull-up control register	Turns the port input pull-up resistor on or off.
DSCR	Drive capacity control register	Specifies the drive capacity. • Normal drive output • High drive output
DSCR2	Drive capacity control register 2	Specifies the drive capacity. • Normal drive/high-drive output • High-speed interface high-drive output

Table 2.6 RX651 Register Configuration (MPC)

Register	Function Name	Function
PWPR	Write-protect register	Write-protect function for PnmPFS register
PnmPFS	Pnm pin function control register	Selects functions of multiplexed pins.
PFCSE	CS output enable register	Disables or enables output on CSn# (n: 0 to 7).
PFCSS0	CS output pin select register 0	Selects output pins for CS0 to CS3.
PFCSS1	CS output pin select register 1	Selects output pins for CS4 to CS7.
PFAOE0	Address output enable register 0	Settings when using pins for address bus
PFAOE1	Address output enable register 1	Settings when using pins for address bus
PFBCR0	External bus control register 0	Settings when using pins for external bus
PFBCR1	External bus control register 1	Settings when using pins for external bus

n: Port name (n = 0 to 9, A to F, J)
m: Pin number (m = 0 to 7)

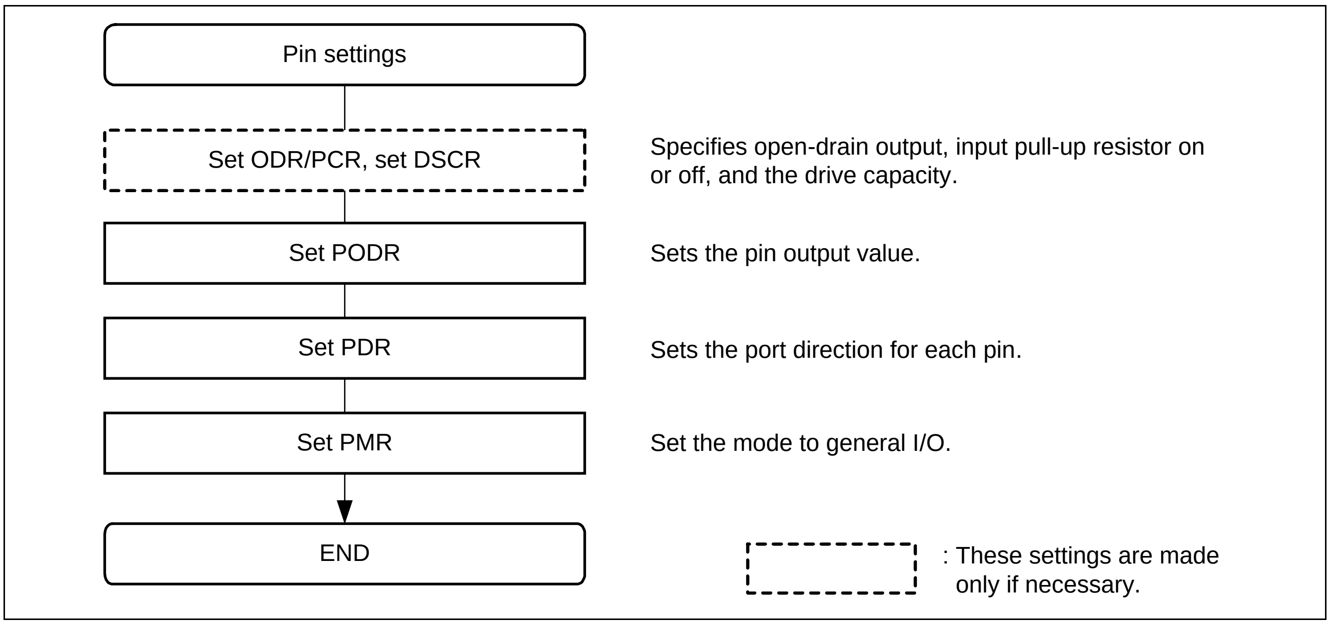
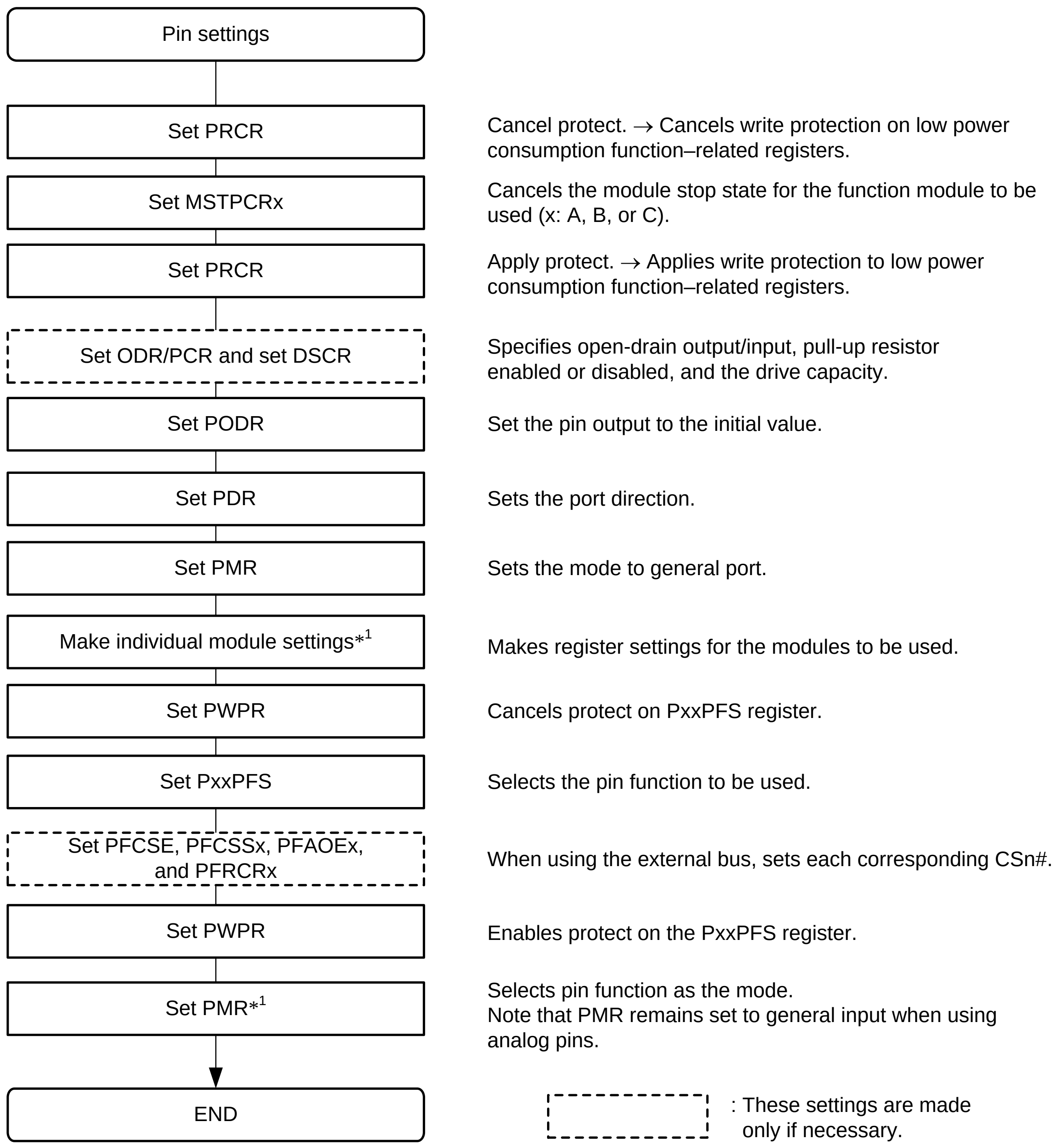

Pin settings
Set ODR/PCR, set DSCR
Specifies open-drain output, input pull-up resistor on or off, and the drive capacity.
Set PODR
Sets the pin output value.
Set PDR
Sets the port direction for each pin.
Set PMR
Set the mode to general I/O.
END
: These settings are made only if necessary.

Figure 2.3 RX651 Pin General I/O Setting Flowchart



Cancel protect. → Cancels write protection on low power consumption function–related registers.

Cancels the module stop state for the function module to be used (x: A, B, or C).

Apply protect. → Applies write protection to low power consumption function–related registers.

Specifies open-drain output/input, pull-up resistor enabled or disabled, and the drive capacity.

Set the pin output to the initial value.

Sets the port direction.

Sets the mode to general port.

Makes register settings for the modules to be used.

Cancels protect on PxxPFS register.

Selects the pin function to be used.

When using the external bus, sets each corresponding CSn#.

Enables protect on the PxxPFS register.

Selects pin function as the mode.
Note that PMR remains set to general input when using analog pins.

: These settings are made only if necessary.

Note: For details on the MPC settings used to assign functions to pins, see the section describing the specific peripheral function.

Note 1. The order of PMR settings and module settings differs according to the module.

Figure 2.4 RX651 Pin Peripheral Function Setting Flowchart

2.2.3 General I/O Setting Example

A general I/O port setting example for the SH7080 Group and RX651 is shown below.

The register names in the setting example are those when using iodefine.h.

Table 2.7 shows an example where PB2 on the SH7080 Group and P34 on the RX651, respectively, are used as general input pins.

Table 2.7 General Input Setting Example

Procedure		SH7080 Group	RX651
1	Set the pin I/O direction to input.	PFC.PBIORL.B2 = 0b	PORT3.PDR.B4 = 0b
2	Set pins as general ports.	PFC.PBCRL1.PB2MD = 000b	PORT3.PMR.B4 = 0b

Table 2.8 shows an example of using PB2 on the SH7080 Group, and P34 on the RX651, for general output. The output value is high.

Table 2.8 General Output Setting Example

Procedure		SH7080 Group	RX651
1	Set the pin output to high.	PB.DRL.B2 = 1b	PORT3.PODR.B4 = 1b
2	Set the pin I/O direction to output.	PFC.PBIORL.B2 = 1b	PORT3.PDR.B4 = 1b
3	Set pins as general ports.	PFC.PBCRL1.PB2MD = 000b	PORT3.PMR.B4 = 0b

2.3 Buses

2.3.1 Comparison of Specifications

The SH7080 Group incorporates a BSC that provides bus state controller functionality.

Table 2.9 is a comparative listing of the specifications of the SH7080 Group and RX651.

Table 2.9 Comparison of SH7080 Group and RX651 Specifications (Bus)

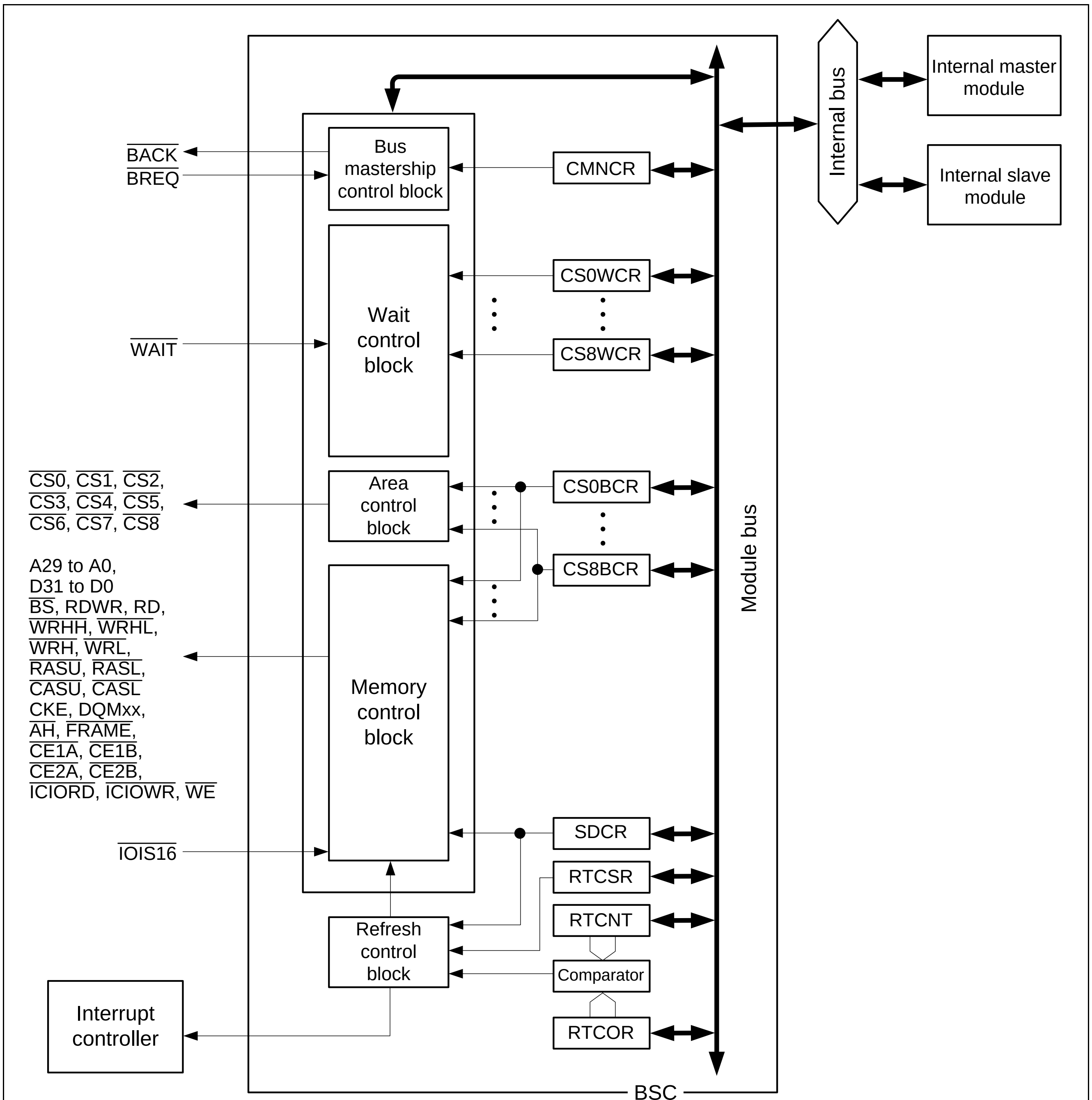
Item	SH7080 Group (BSC)	RX651
External bus address space	• External address space designated as areas CS0 to CS7 (max. 64 MB each) • External address space designated as area CS8 (max. 1 GB) Ability to select SDRAM for up to two CS areas (max. 64 MB)	• External address space designated as areas CS0 to CS7 (16 MB each) • Independent SDRAM space (max. 128 MB)
Bus width	Ability to select the data bus width (8, 16, or 32 bits) for each area	Ability to select the data bus width (8 or 16 bits) for each area
Endianness	Big-endian (fixed)	Endian setting by area*[1]
Bus arbitration	CPU bus and external bus have fixed priority.	• Fixed or toggled priority — Memory bus — Internal peripheral bus — External bus • Fixed priority — CPU bus — Internal main bus
Other	• CS area — Access wait control — CSn assert duration extension — MPX I/O interface (address data multiplexed) — Support for SRAM with byte selection — PCMCIA interface support — Burst ROM (synchronous/asynchronous) support — Burst MPX I/O support • SDRAM area — Auto refresh and self-refresh — CAS latency setting	• CS area — Ability to insert recovery cycles — Cycle wait function — CSn# signal timing control — RD# and WR# signal control timing — Write access mode — Ability to access address data multiplexed I/O devices • SDRAM area — Multiplexed output of row and column addresses — Auto refresh and self-refresh — CAS latency setting • Write buffer — Write buffer function

Note 1. Refer to 1.2.2 for information on endian settings.

2.3.2 Bus Block Diagrams

Comparative bus block diagrams of the SH7080 Group and RX651 are presented below.

Figure 2.5 is a block diagram of the BSC of the SH7080 Group, and Figure 2.6 is a bus block diagram of the RX651.



[Legend]

CMNCR:	Common control register
CSnWCR:	CSn space wait control register (n = 0 to 8)
CSnBCR:	CSn space bus control register (n = 0 to 8)
SDCR:	SDRAM control register
RTCSR:	Refresh timer control/status register
RTCNT:	Refresh timer counter
RTCOR:	Refresh time constant register

Figure 2.5 SH7080 Group Bus Block Diagram

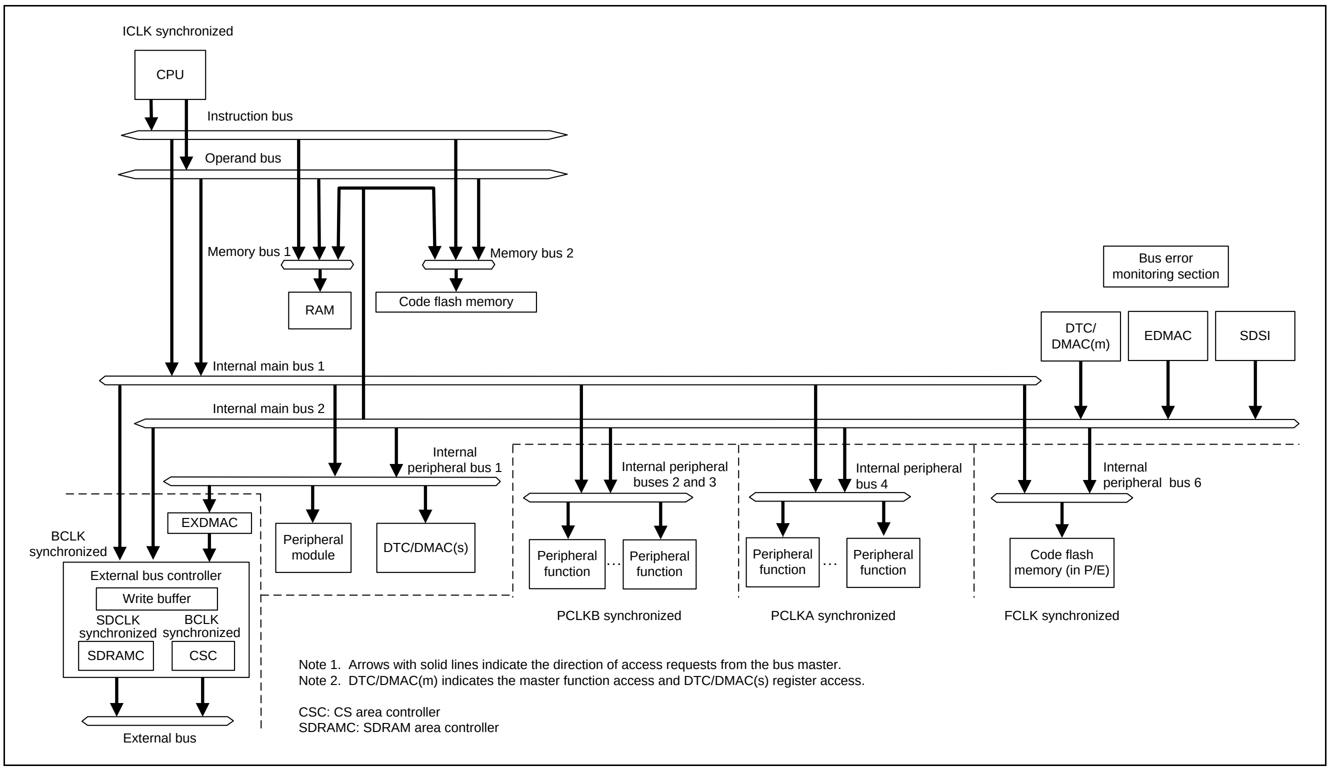


Figure 2.6 RX651 Bus Block Diagram

Table 2.10 shows the bus types on the RX651. The RX651 has a different bus architecture than the SH7080 Group, and the memory buses, internal buses, and peripheral buses each have multiple stages. This enables parallel operation by the CPU and DMAC or DTC, and between the modules on the peripheral buses, thereby speeding up operation overall.

Table 2.10 RX651 Buses

Bus	Connected Modules, etc.	Clock
CPU buses (instruction bus and operand bus)	Instruction bus: CPU, on-chip memory Operand bus: CPU, on-chip memory	ICLK
Memory bus 1	On-chip RAM	ICLK
Memory bus 2	Code flash memory	ICLK
Internal main bus 1	CPU	ICLK
Internal main bus 2	DTC, DMAC, SDSI, on-chip memory	ICLK
Internal peripheral bus 1	Peripheral functions (DTC, DMAC, EXDMAC, interrupt controller, bus error monitoring block)	ICLK (EXDMAC: BCLK)
Internal peripheral bus 2	Peripheral functions (peripheral functions other than those connected to peripheral buses 1, 3, 4, and 5)	PCLKB
Internal peripheral bus 3	Peripheral functions (USBb, PDC, standby RAM)	PCLKB
Internal peripheral bus 4	Peripheral functions (MTU3, SCIi, RSPI, AES)	PCLKA
Internal peripheral bus 5	Reserved area	—
Internal peripheral bus 6	Code flash memory (P/E)	FCLK
External buses (CS areas)	External devices	BCLK
External buses (SDRAM)	SDRAM	SDCLK

ICLK: System clock PCLKA, PCLKB: Peripheral module clock
FCLK: FlashIF clock BCLK: External bus clock SDCLK: SDRAM clock

2.3.3 SDRAM Read/Write Setting Example

As an example of bus settings on the SH7080 Group and RX651, the SDRAMC is used to make read/write settings for a 128 Mbit SDRAM area (MT48LC8M16A2P-6A from Micron Technology: 2 megawords × 16 bits × 4 banks).

< Specifications >

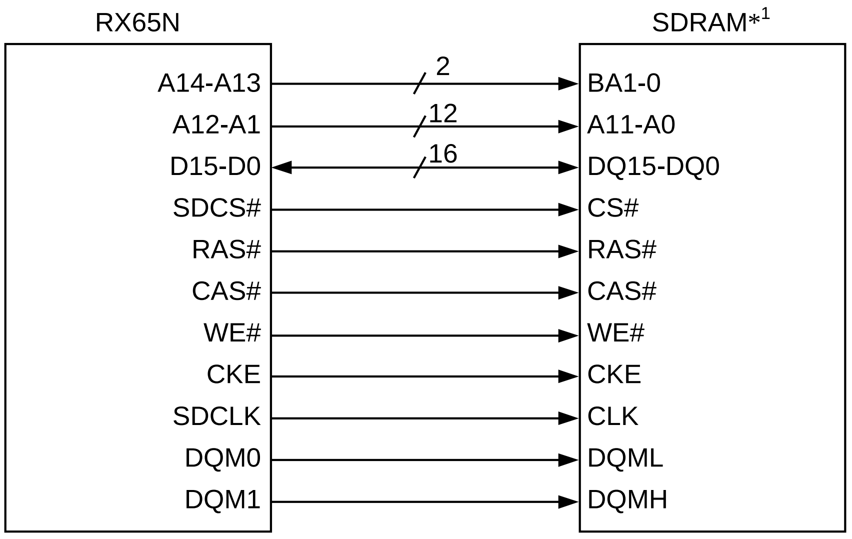
1. The Renesas Starter Kit+ for RX65N (RSK+RX65N) is used.
2. The SDRAM is initialized.
3. Data is written incrementally to the 128 Mbit SDRAM area in word units.
4. After data has been written to the entire area, the written value is read.
5. If the read value matches the anticipated value, LED0 is illuminated.
   If the anticipated value is not matched, LED1 is illuminated.

Table 2.11 SDRAM (MT48LC8M16A2P-6A) Specifications

Item	Description
Configuration	2 megawords × 16 bits × 4 banks (Micron Technology)
Capacity	128 MB
Row addresses	A11 to A0
Column addresses	A8 to A0
Auto-refresh interval	4,096 refresh cycles (max.) every 64 msec.
CAS latency	3 cycles
Initialization auto-refresh count	2 or more
Auto-refresh time (tRFC)	60 ns (min.)
Write recovery time (tWR)	Auto-precharge mode: 1 CLK + 6 ns (min.) Precharge mode: 12 ns (min.)
Precharge command time (tRP)	18 ns (min.)
Time from active command to precharge command (tRAS)	42 ns (min.) to 120,000 ns (max.)
Delay time from active command to read/write command (tRCD)	18 ns (min.)

Table 2.12 Pins Used

Pin	Description
P03	LED0 output (verification complete)
P05	LED1 output (verification error)
PA7-PA0	Address output pins (A7 to A0)
PB6-PB0	Address output pins (A14 to A8)
PD7-PD0	Data I/O pins (D7 to D0)
PE7-PE0	Data I/O pins (D15 to D8)
P70	SDCLK pin output
P61	SDCS# pin output
P62	RAS# pin output
P63	CAS# pin output
P64	WE# pin output
P65	CKE pin output
P66	DQM0 pin output
P67	DQM1 pin output



Note 1. Using MT48LC8M16A2P-6A (2 megawords × 16 bits × 4 banks).

Figure 2.7 SDRAM Connection

Settings for the initialization sequence, SDRAM mode register, auto-refresh interval, and SDRAM read/write timing are made to match the SDRAM module used.

Table 2.13 Initialization Sequence Settings when Connecting SDRAM (MT48LC8M16A2P-6A)

SDRAM Timing	Symbol	Description	SH7080 Group Setting Bϕ (Bus clock): 40 MHz	RX651 Setting SDCLK (SDRAM clock): 60 MHz
Wait time after clock input to precharge command input	—	100 µs	—	After stable output of SDCLK starts, the software waits 100 µsec. and then starts the initialization sequence.
Initialization precharge cycle	tRP	18 ns (min.)	CS3WCR.WTRP[1:0] = 00b: 0 cycles (Bϕ = 40 MHz, so 25 ns)	SDIR.PRC[2:0] = 000b: 3 cycles (SDCLK = 60 MHz, so approx. 50 ns)
Initialization auto-refresh interval	tRFC	60 ns (min.)	CS3WCR.WTRC[1:0] = 00b: 2 cycles (Bϕ = 40 MHz, so 75 ns)	SDIR.ARFI[2:0] = 001b: 4 cycles (SDCLK = 60 MHz, so approx. 66 ns)
Initialization refresh count	—	2 or more	Fixed at 8	SDIR.ARFC[3:0] = 0010b: 2

Table 2.14 SDRAM Mode Register of SDRAM (MT48LC8M16A2P-6A)

Bit	Symbol	Description	SH7080 Group Setting Bϕ (Bus clock): 40 MHz	RX651 Setting SDCLK (SDRAM clock): 60 MHz
b2-b0	BurstLength	Burst length selection 000:1 001:2 010:4 011:8 111:FullPage (only when b3 = 1) Do not use values other than the above.	000:1	000:1
b3	BurstType	Burst type selection 0:Sequential 1:Interleaved	0:Sequential	0:Sequential
b6-b4	CASLatency	CAS latency selection 001:1 010:2 011:3 Do not use values other than the above.	011:3	011:3
b8-b7	OperatingMode	00:StandardOperation Do not use values other than the above.	00:StandardOperation	00:StandardOperation
b9	WriteBurstMode	Write burst mode selection 0:ProgrammedBurstLength 1:SingleLocationAccess	1:SingleLocationAccess	1:SingleLocationAccess
b11-b10	Reserved	Write 00b to these bits.	00	00

Table 2.15 Auto-Refresh Timing of SDRAM (MT48LC8M16A2P-6A)

SDRAM Timing	Symbol	Description	SH7080 Group Setting Bϕ (Bus clock): 40 MHz	RX651 Setting SDCLK (SDRAM clock): 60 MHz
Refresh cycle	tREF	64 ms (max.)	(Used to calculate the auto-refresh interval)	(Used to calculate the auto-refresh interval)
Row address count	—	4,096	(Used to calculate the auto-refresh interval)	(Used to calculate the auto-refresh interval)
Auto-refresh interval	—	15.625 µs (max.) (tREF/row address count)	RTCOR = 0xA55A009C: 156 cycles RTCSR.CKS[2:0] = 001b: RTCNT incrementation clock = Bϕ/4 (Bϕ = 40 MHz, so 15.6 µs)	SDRFCR.RFC[11:0] = 0x03A9: 937 cycles (SDCLK = 60 MHz, so 15.617 µs)
Auto-refresh cancel cycle	tRFC	60 ns (min.)	CS3WCR.WTRC[1:0] = 00b: 2 cycles (Bϕ = 40 MHz, so 75 ns)	SDRFCR.REFW[3:0] = 0011b: 4 cycles (SDCLK = 60 MHz, so approx. 66 ns)

Table 2.16 Read/Write Timing when Connecting SDRAM (MT48LC8M16A2P-6A)

SDRAM Timing	Symbol	Description	SH7080 Group Setting Bϕ (Bus clock): 40 MHz	RX651 Setting SDCLK (SDRAM clock): 60 MHz
Column latency*[2]	—	3 cycle*[1]	CS3WCR.A3CL[1:0] = 10b: 3 cycles	SDTR.CL[2:0] = 011b: 3 cycles
Write recovery time	tWR	1CLK + 6 ns (min.) SH7080: 31 ns (min.) RX651: 22.66 ns (min.)	CS3WCR.TRWL[1:0] = 01b: 1 cycle (Bϕ = 40 MHz, so 50 ns)	SDTR.WR = 1: 2 cycles (SDCLK = 60 MHz, so approx. 33 ns)
Row precharge time	tRP	18 ns (min.)	CS3WCR.WTRP[1:0] = 00b: 0 cycles (Bϕ = 40 MHz, so 25 ns)	SDTR.RP[2:0] = 001b: 2 cycles (SDCLK = 60 MHz, so approx. 33 ns)
Row active time*[2]	tRAS	42 ns (min.)	—	SDTR.RAS[2:0] = 010b: 3 cycles (SDCLK = 60 MHz, so approx. 50 ns)
Row/column latency*[2]	tRCD	18 ns (min.)	CS3WCR.WTRCD[1:0] = 00b: 0 cycles (Bϕ = 40 MHz, so 25 ns)	SDTR.RCD[1:0] = 01b: 2 cycles (SDCLK = 60 MHz, so approx. 33 ns)

Note 1. Select 3 in SDRAM mode register.
Note 2. Set the row active time to a value less than or equal to the row/column latency + column latency.

A setting example when connecting SDRAM is presented below.

Table 2.17 SDRAM Data Transfer Initial Setting Example

Procedure		SH7080 Group Setting Example Bϕ (Bus clock): 40 MHz	RX651 Setting Example SDCLK (SDRAM clock): 60 MHz
1	Cancel register protect.	—	SYSTEM.PRCR = A503h (register writes enabled)

2	Disable SDCLK and BCLK output.	—	SYSTEM.SCKCR.PSTOP0 = 1
			(SDCLK pin output halted)
			SYSTEM.SCKCR.PSTOP1 = 1
			(BCLK pin output halted)

3	Make bus error monitoring function setting.	—	BSC.BEREN.IGAEN = 0
			(unauthorized address access detection disabled)
			BSC.BEREN.TOEN = 0
			(bus timeout detection disabled)

4	Make bus priority setting.	—	BSC.BUSPRI.BPEB[1:0] = 00b (priority fixed)

5	Make SDRAM pin function settings.	PFC.PCCRL4 = 1111h (A12 to A15 output)
		PFC.PCCRL3 = 1111h (A8 to A11 output)
		PFC.PCCRL2 = 1111h (A4 to A7 output)
		PFC.PCCRL1 = 1111h (A0 to A3 output)
		PFC.PDCRL4 = 1111h (D12 to D15 output)
		PFC.PDCRL3 = 1111h (D8 to D11 output)
		PFC.PDCRL2 = 1111h (D4 to D7 output)
		PFC.PDCRL1 = 1111h (D0 to D3 output)
		PFC.PACRL4 = 1011h
		(CK output, WRH#/DQMLU output, WRL#/DQMLL output)
		PFC.PACRL3 = 0055h
		(CKE output, RDWR output)
		PFC.PACRL2 = 2000h (CS3# output)
		PFC.PBCRL2 = 0044h
		(CASL# output, RASL# output)
			MPC.PFBCR0 = 11h
			(setting for external address bus A0 to A7 and external data bus D8 to D15)
			MPC.PFBCR1 = D0h
			(SDCLK output enabled, DQM1 output enabled, output enabled on CKE, SDCS#, RAS#, CAS#, WE#, and DQM0 pins)
			MPC.PFAOE0 = 7Fh
			(A15 output disabled, A14 to A8 output enabled)
			MPC.PFAOE1 = 00h
			(A23 to A16 output disabled)
			PORTA.PMR &= 00h (A0 to A7)
			PORTB.PMR &= 80h (A8 to A14)
			PORTD.PMR &= 00h (D0 to D7)
			PORTE.PMR &= 00h (D8 to D15)
			PORT6.PMR &= 01h
			(SDCS#, RAS#, CAS#, WE#, CKE, DQM0, DQM1)
			PORT7.PMR &= FEh (SDCLK)

6	Make external bus setting.	—	SYSTEM.SYSCR0 = 5A03h (external bus enabled/on-chip ROM enabled) SYSTEM.SYSCR0.EXBE reflection confirmation
7	Enable SDCLK output.	—	SYSTEM.SCKCR.PSTOP0 = 0 (SDCLK pin output operation)
8	Make register protect setting.	—	SYSTEM.PRCR = A500h (register writes disabled)
9	Wait 100 µs after SDCLK output.	←	←

Procedure		SH7080 Group Setting Example Bϕ (Bus clock): 40 MHz	RX651 Setting Example SDCLK (SDRAM clock): 60 MHz
10	Make initialization sequence settings.	BSC.CS3WCR.WTRP[1:0] = 00b (precharge completion wait cycle count: no wait cycles) Note: Auto-refresh count fixed at 8 BSC.CS3WCR.WTRC[1:0] = 00b (REF command/self-refresh canceled → idle cycle count during ACTV/REF/MRS command: 2 cycles)	BSC.SDIR.PRC[2:0] = 000b (initialization precharge cycle count: 3 cycles) BSC.SDIR.ARFC[3:0] = 0010b (initialization auto-refresh count: 2) BSC.SDIR.ARFI[3:0] = 0001b (initialization auto-refresh interval: 4 cycles)
11	Start initialization sequence.	When the SH7080 Group accesses the SDRAM mode register, the initialization sequence starts, access to the SDRAM address space is enabled, and the final step is executed.	BSC.SDICR.INIRQ = 1 (initialization sequence start)
12	Wait for initialization sequence end.	—	Confirm that BSC.SDSR = 0x00.
13	Make SDRAM bus width setting.	BSC.CS3BCR.BSZ[1:0] = 10 (16-bit space) BSC.CS3BCR.TYPE[2:0] = 100b (SDRAM)	BSC.SDCCR.EXENB = 0 (SDRAM address space operation disabled) BSC.SDCCR.BSIZE[1:0] = 00b (16-bit bus space)
14	Make SDRAM mode register setting.	SDMR3 address = FFF85460h (16-bit bus width/CAS latency: 3)	BSC.SDMOD = 0230h (mode register value)
15	Make SDRAM timing settings.	BSC.CS3WCR.WTRCD[1:0] = 00b (wait cycle count between ACTV command → READ(A)/WRIT(A) command: 0 cycles) BSC.CS3WCR.WTRP[1:0] = 00b (precharge completion wait cycle count: 0 cycles) BSC.CS3WCR.A3CL[1:0] = 10b (area 3 CAS latency: 3 cycles) BSC.CS3WCR.TRWL[1:0] = 01b (precharge start wait cycle count: 1 cycle) BSC.CS3WCR.WTRC[1:0] = 00b (idle cycle count between REF command/self-refresh cancelation → ACTV/REF/MRS command: 2 cycles)	BSC.SDTR.RCD[1:0] = 01b (row/column latency: 2 cycles) BSC.SDTR.RP[2:0] = 001b (row precharge time: 2 cycles) BSC.SDTR.CL[2:0] = 011b (SDRAMC column latency: 3 cycles) BSC.SDTR.WR = 1 (write recovery time: 2 cycles) BSC.SDTR.RAS[2:0] = 010b (row active time: 3 cycles)
16	Make address multiplexing settings.	BSC.SDCR.A3ROW[1:0] = 01b (12-bit row addresses) BSC.SDCR.A3COL[1:0] = 00b (8-bit column addresses)	BSC.SDADR.MXC[1:0] = 01b (address multiplexing: 9 bits)
17	Make endian setting.	—	BSC.SDCMOD.EMODE = 0 (same endian setting for SDRAM address space and operating mode)
18	Make access mode setting.	—	BSC.SDAMOD.BE = 0 (continuous access disabled)
19	Make auto-refresh timing settings.	BSC.RTCOR = A55A009Ch (auto-refresh interval: 156 cycles (15.6 μs)) BSC.RTCSR setting performed in step 21. This is because BSC.RTCNT is integrated after BSC.RTCSR.CKS[2:0] is set.	BSC.SDRFCR.RFC[11:0] = 3A9h (auto-refresh request interval: 937 cycles (15.617 μs)) BSC.SDRFCR.REFW[3:0] = 0011b (auto-refresh cycle/self-refresh cancelation cycle count: 4 cycles)

Procedure		SH7080 Group Setting Example Bϕ (Bus clock): 40 MHz	RX651 Setting Example SDCLK (SDRAM clock): 60 MHz

20	Enable auto-refresh.	BSC.SDCR.RFSH = 1 (refresh)
		BSC.SDCR.RMODE = 0 (auto-refresh)
			BSC.SDRFEN.RFEN = 1 (auto-refresh enabled)

21	Make refresh setting.	BSC.RTCSR = A55A0008h (refresh count: 1/RTCNT incrementation clock: Bϕ/4)	—
22	Enable SDRAM address space operation.	SDMR3 = 0 (write to SDRAM mode register)	BSC.SDCCR.EXENB = 1 (SDRAM address space operation enabled)

2.4 Interrupt Controller

2.4.1 IRQ Usage Example

A setting example using IRQ3 on the SH7080 Group and RX651 is shown below.

The register names in the setting example are those when using iodefine.h.

PB5 on the SH7080 Group and P13 on the RX651 is used as the IRQ3 input pin in this example.

Table 2.18 Interrupt Initial Setting Example (IRQ3 Settings)

Procedure	SH7080 Group	RX651

1 Make I/O port settings (RX651 only).	—	PORT1.PDR.B3 = 0b (P13 set as input)
		PORT1.PMR.B3 = 0b (P13 set as general-purpose)
		MPC.PWPR.B0WI = 0b (PFSWE write enabled)
		MPC.PWPR.PFSWE = 1b (PFS write enabled)
		MPC.P13PFS.ISEL = 1b (P13 set to IRQ3)
		MPC.PWPR.PFSWE = 0b (PFS write disabled)
		MPC.PWPR.B0WI = 1b (PFSWE write disabled)

2 Make interrupt controller settings.	INTC.IRQCR.IRQ3S = 01b
	(IRQ3 falling-edge detection)
	INTC.IPRA._IRQ3 = 15
	(IRQ3 priority level: 15)
		IRQFLTC0.FCLKSEL3[1:0] = 11b
		(IRQ3 digital filter sampling: PCLKB/64)
		IRQCR3.IRQMD[1:0] = 01b
		(IRQ3 falling-edge detection)
		ICU.IR[67].IR = 0b
		(IRQ3 interrupt request flag cleared)
		IRQFLTE0.FLTEN3 = 1b
		(IRQ3 digital filter enabled)
		ICU.IPR[67].IPR[3:0] = 15
		(IRQ3 priority level: 15)
		ICU.IER[8].IEN3 = 1b
		(IRQ3 interrupt enabled)

3 Make I/O port settings (SH7080 Group only).	PFC.PBIORL.B5 = 0b
	(PB5 set as input)
	PFC.PBCRL2.PB5MD = 001b
	(PB5 set to IRQ3)
		—

2.5 Data Transfer Controller (DTC)

2.5.1 Comparison of Specifications

Data transfer controller functionality is provided on the SH7080 Group by the DTC and on the RX651 by the DTCb.

On both the SH7080 Group and RX651 transfer information is located in the RAM and specified by means of DTC vectors. The basic operation of the three transfer modes (normal transfer mode, repeat transfer mode, and block transfer mode) is identical on the two platforms. Table 2.19 is a comparative listing of the specifications of the SH7080 Group and RX651.

Table 2.19 Comparison of SH7080 Group and RX651 Specifications (DTC)

Item	SH7080 Group (DTC)	RX651 (DTCb)
Transfer modes	• Normal transfer mode • Repeat transfer mode • Block transfer mode	
Activation sources	• External interrupt • Peripheral function interrupt	• External interrupt • Peripheral function interrupt • Software interrupt
Activation enable/ disable control	Activated by DTC enable register of DTC module.	Activated by DTC activation enable register of interrupt controller.
Transfer spaces	Transfer between the following spaces is possible: • On-chip memory space • On-chip peripheral module space (excluding DMAC, DTC, BSC, UBC, and FLASH) • External memory space • Memory-mapped external device space One of the specified areas must be in the on-chip memory space or on-chip peripheral module space.	Transfer between the following spaces is possible: • On-chip memory space • On-chip peripheral module space • External memory space
Transfer units	• 1 data unit: Selectable among 8, 16, and 32 bits • Repeat count: Selectable from 1 to 256 • 1 block: Selectable 1 to 256 data units • Block count: Selectable from 1 to 65,536	
CPU interrupt requests	• An interrupt generated by a CPU interrupt request may be used as the DTC activation source. • A CPU interrupt at single data unit transfer-end may be used. • A CPU interrupt after transfer of a specified number of data units may be used.	
Method	Control information is allocated for each interrupt source by using DTC vectors.	
Other	• Chain transfer • Transition to module-stop state • The following functions can be used to shorten the transfer duration and reduce memory usage: — Transfer information read skipping — Write-back skipping — Short-address mode — Bus mastership release timing setting	• Chain transfer • Sequence transfer • Event link • Transition to module-stop state • The following functions can be used to shorten the transfer duration and reduce memory usage: — Transfer information read skipping — Write-back skipping — Write-back disable — Short-address mode

2.5.2 Register Comparison

On the SH7080 Group operation of the DTC is enabled by canceling the module-stop state for the DTC. On the RX651, in addition to canceling the module-stop state for the DTC, it is necessary to make a setting in the DTC module start register (DTCST) to enable DTC operation.

Table 2.20 provides a comparative listing of the registers of the SH7080 Group and the RX651.

Guide to Symbols in "Changes" Column of Table

◎: Register with same bit assignments on SH7080 Group and RX651

△: Register with different bit assignments on SH7080 Group and RX651

—: Register not present on SH7080 Group or RX651

Table 2.20 SH7080 Group and RX651 Register Comparison (DTC)

SH7080 Group (DTC)	RX651 (DTCb)	Changes
DTC mode register A (MRA)	DTC mode register A (MRA)	△
DTC mode register B (MRB)	DTC mode register B (MRB)	△
—	DTC mode register C (MRC)	—
DTC source address register (SAR)	DTC transfer source register (SAR)	◎
DTC destination address register (DAR)	DTC transfer destination register (DAR)	◎
DTC transfer count register A (CRA)	DTC transfer count register A (CRA)	◎
DTC transfer count register B (CRB)	DTC transfer count register B (CRB)	◎
DTC control register (DTCCR)	DTC control register (DTCCR)	△
DTC vector base register (DTCVBR)	DTC vector base register (DTCVBR)	◎
DTC enable registers A to E (DTCERA to DTCERE)*1	—	—
Bus function extending register (BSCEHR) DTC short address mode (DTSA bit)	DTC address mode register (DTCADMOD)	△
—	DTC module start register (DTCST)	—
	DTC status register (DTCSTS)	
	DTC index table base register (DTCIBR)	
	DTC operation register (DTCOR)	
	DTC sequence transfer enable register (DTCSQE)	
	DTC address displacement register (DTCDISP)	

Note 1. On the RX651 transfer request settings from peripheral modules are made by means of the interrupt controller.

2.5.3 Activation Source Settings

On the SH7080 Group peripheral modules can activate the DTC by making settings in activation source DTC enable registers A to E (DTCERA to DTCERE) of the DTC module. On the RX651 DTC activation sources are specified by means of settings to DTC transfer request enable register n (DTCERn) of the interrupt controller. This allows specific interrupts to be enabled as activation sources for enabling the DTC.

2.5.4 DTC Vector Configuration

The DTC vector configuration of the SH7080 Group and RX651 is shown below.

On the SH7080 Group the upper 20 bits of the start address of the DTC vector table are the DTC vector base address (DTCVBR) and the lower 12 bits are calculated as “400h + vector number × 4”. The base address of the DTC vector table is aligned with a 4 KB boundary such that the lower 12 bits are 0.

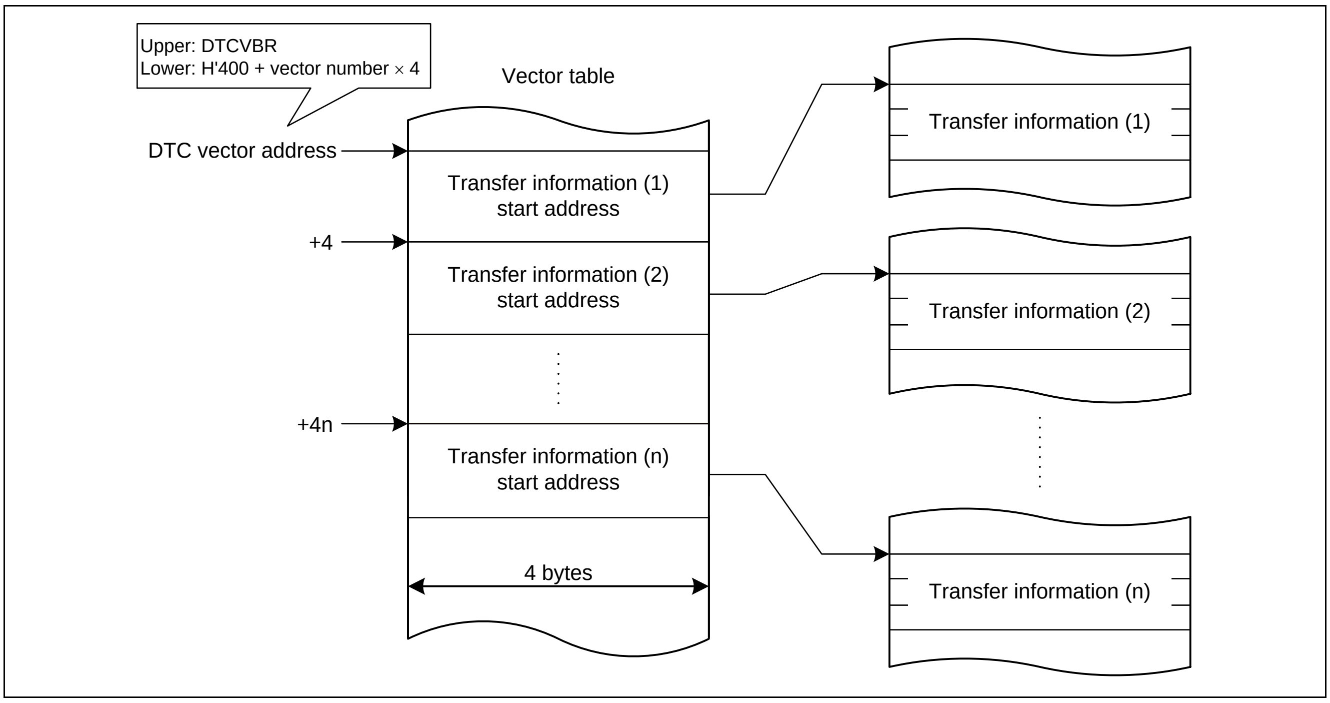

Upper: DTCVBR
Lower: H'400 + vector number × 4
Vector table
DTC vector address
Transfer information (1) start address
+4
Transfer information (2) start address
+4n
Transfer information (n) start address
4 bytes
Transfer information (1)
Transfer information (2)
Transfer information (n)

Figure 2.8 DTC Vector Configuration on SH7080 Group

On the RX651 the start address of the DTC vector table is calculated as “DTC vector base address (DTCVBR) + (vector number × 4)”. The base address of the DTC vector table is aligned with a 1 KB boundary such that the lower 10 bits are 0.

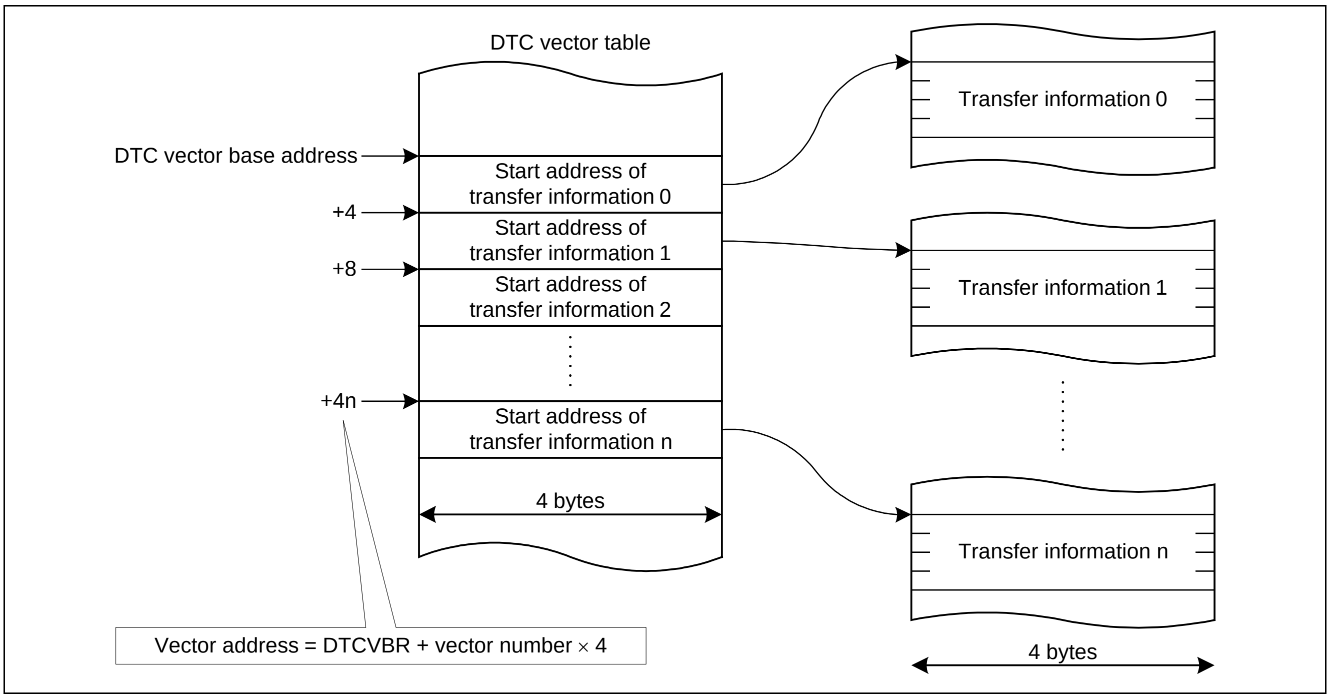

DTC vector table
DTC vector base address
Start address of transfer information 0
+4
Start address of transfer information 1
+8
Start address of transfer information 2
+4n
Start address of transfer information n
4 bytes
Vector address = DTCVBR + vector number × 4
Transfer information 0
Transfer information 1
Transfer information n
4 bytes

Figure 2.9 DTC Vector Configuration on RX651

2.5.5 Allocation of Transfer Information

On the SH7080 Group the format of DTC transfer information is fixed at big-endian. On the RX651 the endian setting for DTC transfer information depends on the allocation area.

Short address mode is selected on the SH7080 Group by making a setting in the bus function extending register (BSCEHR) of the BSC and on the RX651 by making a setting in the DTC address mode register (STCADMOD). Figure 2.10 illustrates the DTC transfer source and transfer destination addresses in short address mode.

On the RX651 the transfer information includes the contents of DTC mode register C (MRC), which specifies the displacement value added to the ordinary transfer source address.

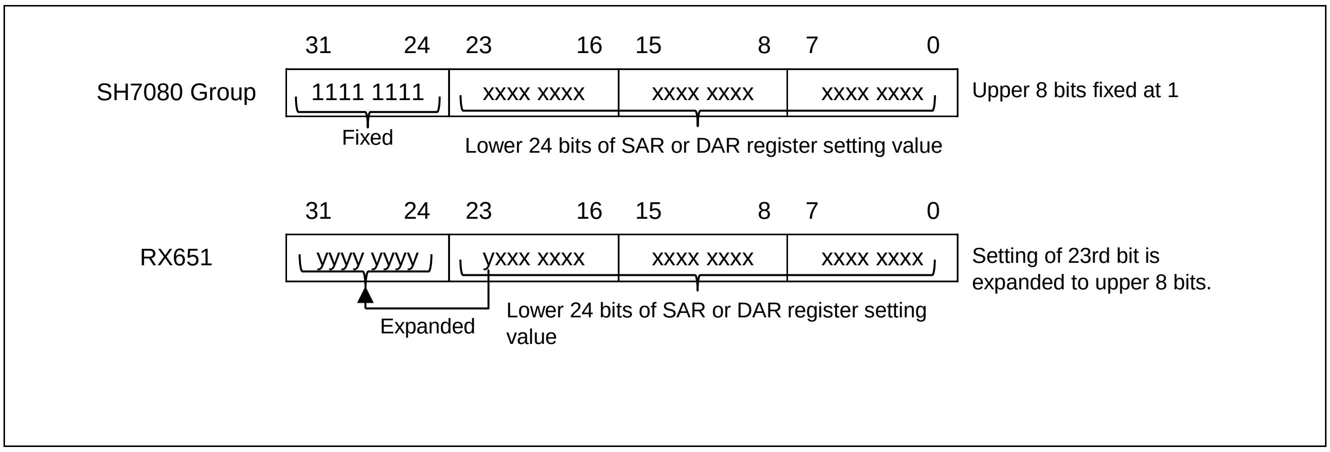


Figure 2.10 Transfer Source and Transfer Destination Addresses in Short Address Mode

Figure 2.11 shows the usual allocation of DTC transfer information on the SH7080 Group and RX651.

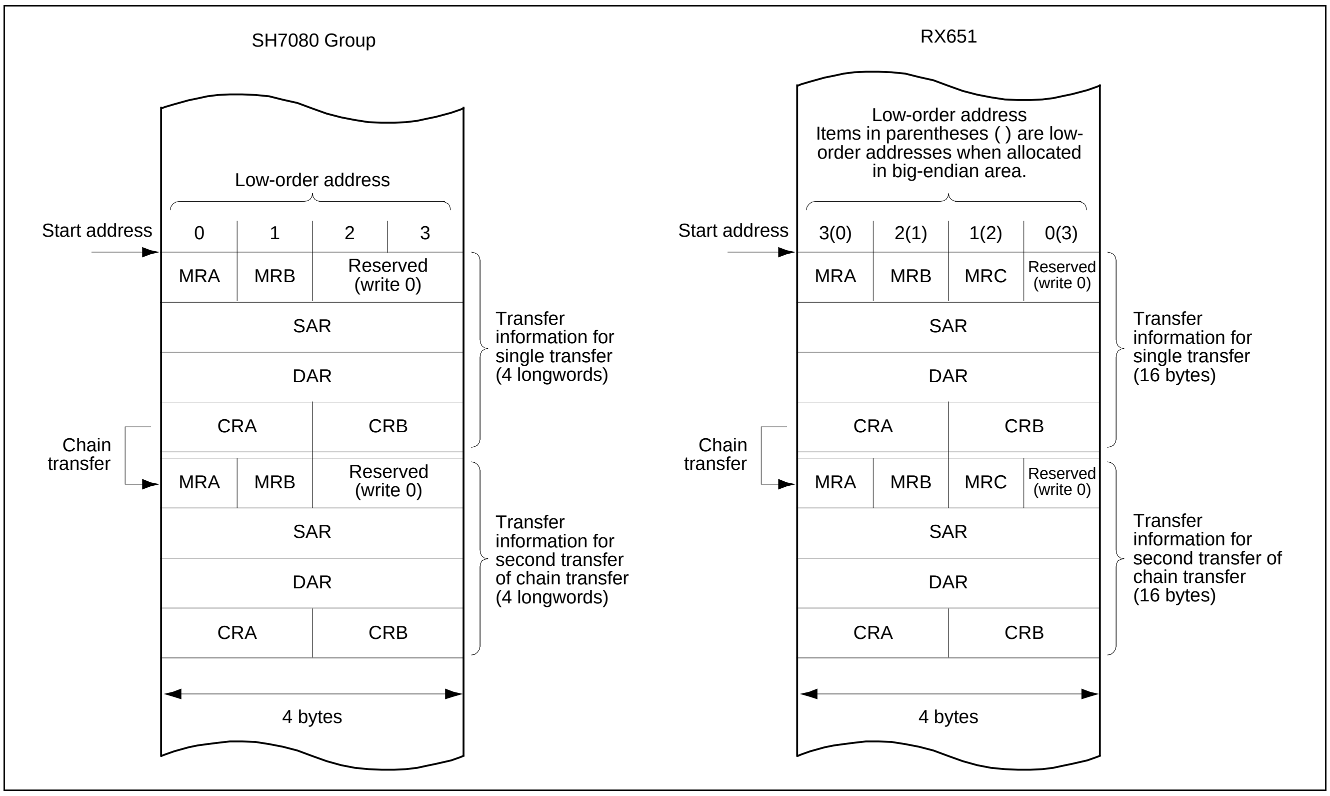

SH7080 Group
Low-order address
Start address
0
1
2
3
MRA
MRB
Reserved (write 0)
SAR
DAR
CRA
CRB
Transfer information for single transfer (4 longwords)
Chain transfer
MRA
MRB
Reserved (write 0)
SAR
DAR
CRA
CRB
Transfer information for second transfer of chain transfer (4 longwords)
4 bytes
RX651
Low-order address Items in parentheses ( ) are low-order addresses when allocated in big-endian area.
Start address
3(0)
2(1)
1(2)
0(3)
MRA
MRB
MRC
Reserved (write 0)
SAR
DAR
CRA
CRB
Transfer information for single transfer (16 bytes)
Chain transfer
MRA
MRB
MRC
Reserved (write 0)
SAR
DAR
CRA
CRB
Transfer information for second transfer of chain transfer (16 bytes)
4 bytes

Figure 2.11 Allocation of Transfer Information (Ordinary)

2.5.6 Module Stop

On the RX651 the DTCb module-stop state is canceled after a reset.

The module-stop setting bit (MSTPCRA.MSTPA28) is common to both the DTCb and DMACAa on the RX651, so module-stop control for these two modules is simultaneous.

Refer to 2.16 for information on the module-stop state.

2.5.7 Normal Transfer Setting Example

In the data transfer controller (DTC) setting example shown below for the SH7080 Group and RX651, the DTC is used to implement data transfer between the serial communication interface (SCI) and the on-chip RAM. Refer to Table 2.56 for an initial setting example for the SCI. In the example shown here, the use of SCI interrupts for DTC activation is the only portion of the settings that differs between the microcontrollers.

< Specifications >

1. The RSK+RX65N board is used, and the SCI transfer mode is asynchronous serial transfer.
2. When an SCI2 transmit data-empty interrupt request occurs, the DTC transfers one byte of transmit data from the transmit buffer in the on-chip RAM to the transmit data register of the SCI2.
3. When an SCI2 receive data-full interrupt request occurs, the DTC transfers one byte of receive data to the receive buffer in the on-chip RAM.
4. When transmission of 32 bytes finishes (DTC transfer-end), a transmit interrupt (TXI) is generated.
5. When reception of 32 bytes finishes (DTC transfer-end), a receive interrupt (RXI) is generated.
6. At successful completion, LED1 turns on. LED2 turns on if an error interrupt occurs.

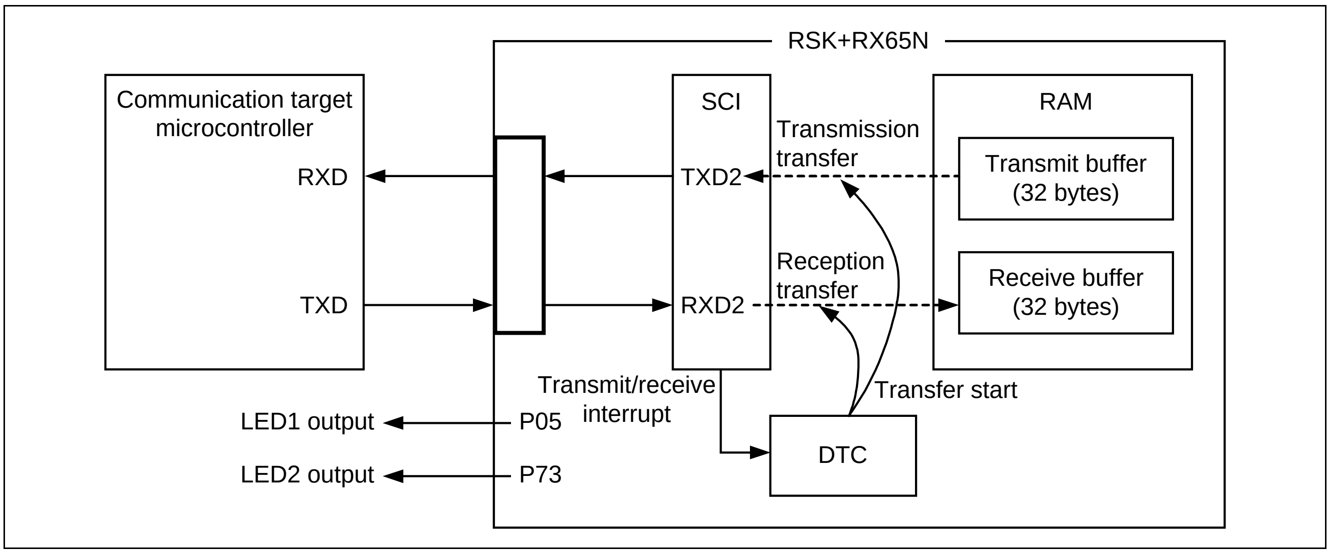

RSK+RX65N
Communication target microcontroller
RXD
TXD
SCI
TXD2
RXD2
Transmission transfer
Reception transfer
RAM
Transmit buffer (32 bytes)
Receive buffer (32 bytes)
Transmit/receive interrupt
Transfer start
DTC
LED1 output
LED2 output
P05
P73

Figure 2.12 Example of Data Transfer between RAM and SCI Using DTC

Table 2.21 DTC Transfer Specifications

Item	Transmit Transfer	Receive Transfer
Transfer mode	Normal transfer mode	Normal transfer mode
Transfer count	32	32
Transfer size	Byte	Byte
Transfer source	On-chip RAM (transmit buffer)	Receive data register (SCI2)
Transfer destination	Transmit data register (SCI2)	On-chip RAM (receive buffer)
Transfer source address	Transfer source address incremented following transfer	Fixed
Transfer destination address	Fixed	Transfer destination address incremented following transfer
Activation sources	SCI2 transmit data-empty interrupt	SCI2 receive data-full interrupt
Interrupt handling	Interrupt to CPU when transfer of specified data finishes	Interrupt to CPU when transfer of specified data finishes
Address mode	Full address mode	Full address mode
Pins used	P50/TXD2	P52/RXD2
	P05/GPIO (LED1 output)	
	P73/GPIO (LED2 output)	

A DTC initial setting example is shown below.

1. DTC_TX is the transmit transfer information structure.
   DTC_RX is the receive transfer information structure.
2. The DTC vector tables are allocated as follows.

SH7080 Group	#pragma address DTC_VECT_TABLE = 0x400 (address defined by user) volatile unsigned long DTC_VECT_TABLE[240];
RX651	#pragma address DTC_VECT_TABLE = 0x00010000 (address defined by user) volatile unsigned long DTC_VECT_TABLE[256];

Table 2.22 DTC Normal Transfer Initial Setting Example

Procedure		SH7080 Group Setting Example	RX651 Setting Example

1	Cancel module-stop state.	STB.STBCR2.MSTP4 = 0	SYSTEM.PRCR = 0xA502
			SYSTEM.MSTPCRA.MSTPA28 = 0
			SYSTEM.PRCR = 0xA500

2	Disable transfer information read skip.	DTC.DTCCR.RRS = 0 (no transfer information read skip)	DTC.DTCCR.RRS = 0 (no transfer information read skip)
3	Make transfer information settings (transmission).	DTC_TX.MRA.MD[1:0] = 00b (normal transfer mode) DTC_TX.MRA.SZ[1:0] = 00b (byte transfer) DTC_TX.MRA.SM[1:0] = 10b (SAR incremented) DTC_TX.MRB.CHNE = 0 (chain transfer disabled) DTC_TX.MRB.DISEL = 0 (interrupt issued at completion of specified number of data transfers) DTC_TX.MRB.DM[1:0] = 00b (DAR fixed) DTC_TX.SAR = transmit buffer start address DTC_TX.DAR = SCI2.SCTDR address DTC_TX.CRA = 32 (transfer count)	DTC_TX.MRA.MD[1:0] = 00b (normal transfer mode) DTC_TX.MRA.SZ[1:0] = 00b (byte transfer) DTC_TX.MRA.SM[1:0] = 10b (SAR incremented) DTC_TX.MRA.WBDIS = 0 (write back) DTC_TX.MRB.CHNE = 0 (chain transfer disabled) DTC_TX.MRB.DISEL = 0 (interrupt issued at completion of specified number of data transfers) DTC_TX.MRB.DM[1:0] = 00b (DAR fixed) DTC_TX.MRC.DISPE = 0 (no displacement value added) DTC_TX.SAR = transmit buffer start address DTC_TX.DAR = SCI2.TDR address DTC_TX.CRA = 32 (transfer count)
4	Make transfer information settings (reception).	DTC_RX.MRA.MD[1:0] = 00b (normal transfer mode)) DTC_RX.MRA.SZ[1:0] = 00b (byte transfer) DTC_RX.MRA.SM[1:0] = 00b (SAR fixed) DTC_RX.MRB.CHNE = 0 (chain transfer disabled) DTC_RX.MRB.DISEL = 0 (interrupt issued at completion of specified number of data transfers) DTC_RX.MRB.DM[1:0] = 10b (DAR incremented) DTC_RX.SAR = SCI2.SCTDR address DTC_RX.DAR = receive buffer start address DTC_RX.CRA = 32 (transfer count)	DTC_RX.MRA.MD[1:0] = 00b (normal transfer mode) DTC_RX.MRA.SZ[1:0] = 00b (byte transfer) DTC_RX.MRA.SM[1:0] = 00b (SAR fixed) DTC_TX.MRA.WBDIS = 0 (write back) DTC_RX.MRB.CHNE = 0 (chain transfer disabled) DTC_RX.MRB.DISEL = 0 (interrupt issued at completion of specified number of data transfers) DTC_RX.MRB.DM[1:0] = 10b (DAR incremented) DTC_TX.MRC.DISPE = 0 (no displacement value added) DTC_RX.SAR = SCI2.RDR address DTC_RX.DAR = receive buffer start address DTC_RX.CRA = 32 (transfer count)

Procedure		SH7080 Group Setting Example	RX651 Setting Example

5	Make DTC vector table settings.	DTC_VECT_TABLE[225] = DTC_RX address
		DTC_VECT_TABLE[226] = DTC_TX address
		DTC.DTCVBR = 0x00000000
			DTC_VECT_TABLE[62] = DTC_RX address
			DTC_VECT_TABLE[63] = DTC_TX address
			DTC.DTCVBR = 0x00010000

6	Make address mode setting.	BSC.BSCEHR.DTSA = 0 (full address mode)	DTC.DTCADMOD.SHORT = 0 (full address mode)

7	Make activation source settings.	DTC.DTCERE.DTCE11 = 1
		(DTC activation by SCI.RXI2)
		DTC.DTCERE.DTCE10 = 1
		(DTC activation by SCI.TXI2)
			ICU.DTCER[62].DTCE = 1
			(DTC activation by SCI.RXI2)
			ICU.DTCER[63].DTCE = 1
			(DTC activation by SCI.TXI2)

8	Make SCI settings.	Specify SCI asynchronous transfer. Make settings listed in Table 2.56 for SCI function and ICU function. Enable TXI interrupts, RXI interrupts, and error interrupts. The DTC will not operate if interrupts are not enabled.	
9	Activate DTC module.	—	DTC.DTCST.DTCST = 1 (DTC module operating)

When transmission of 32 bytes of data finishes, the SCI2's transmit interrupt (TXI) is generated.

When reception of 32 bytes of data finishes, the SCI2's receive interrupt (RXI) is generated.

The processing associated with these interrupts is not specified. In the sample code end processing is implemented for the transmit and receive interrupts.

2.6 Direct Memory Access Controller (DMAC)

2.6.1 Comparison of Specifications

Direct memory access control functionality is implemented on the SH7080 Group by an on-chip DMAC and on the RX651 by an on-chip DMACAa and by a dedicated on-chip EXDMACa for transfers between external areas.

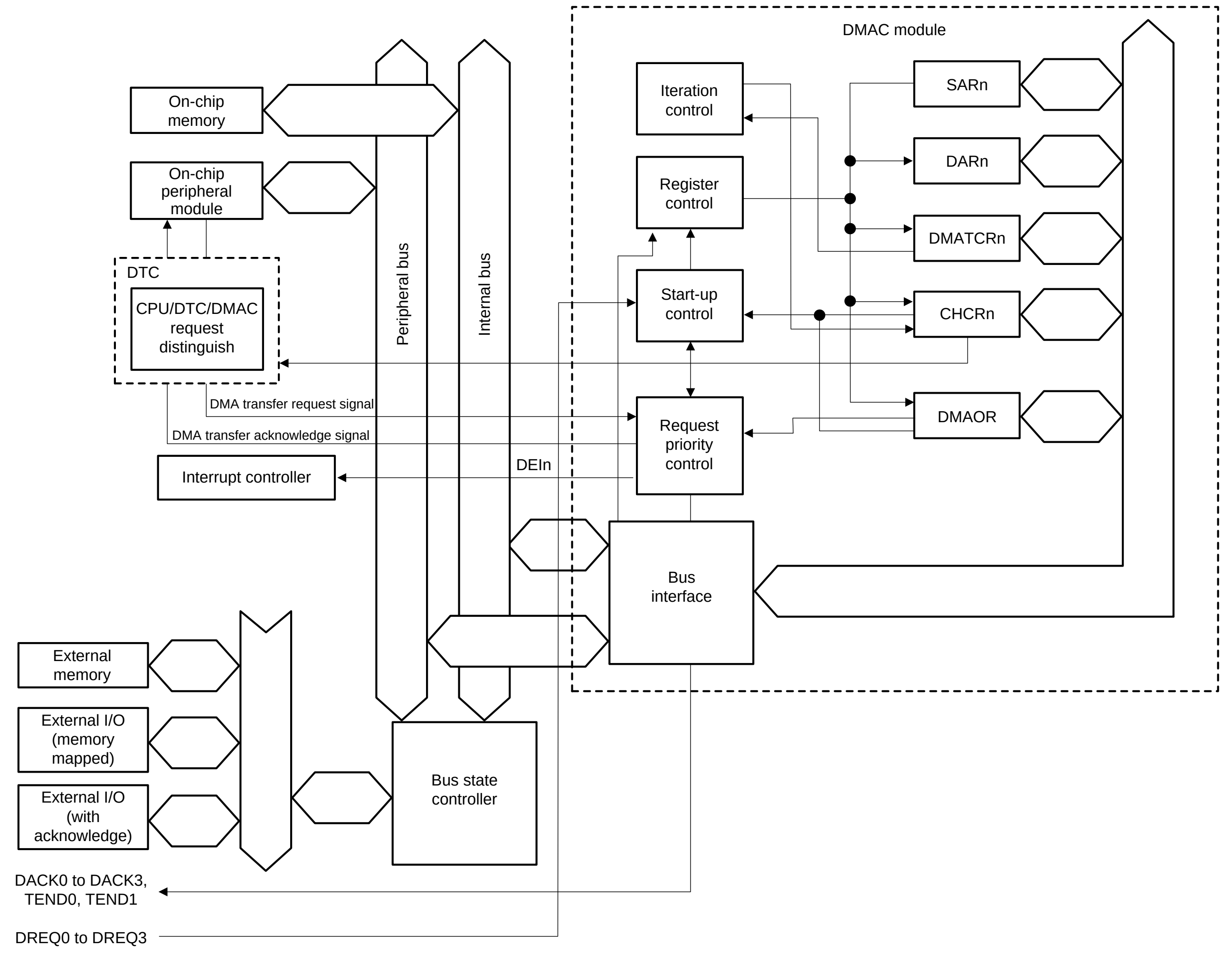
The internal bus configuration of the RX651 differs from that of the SH7080 Group. It supports independent data transfers by CPU instruction execution and by the DMAC or DTC for improved transfer performance. Table 2.23 is a comparative listing of the specifications of the SH7080 Group and RX651.

Table 2.23 Comparison of SH7080 Group and RX651 Specifications (DMAC)

		SH7080 Group	RX651	
Item		**DMAC**	**DMACAa**	**EXDMACa**
Number of channels		4 channels	8 channels	2 channels
Maximum transfer count (maximum transfer data unit count on RX)		16 M (16,777,216)	64 M data units (block transfer mode max. total transfer count: 1,024 data units × 65,536 blocks) Free running is also supported.	1 M data units (block transfer mode max. total transfer count: 1,024 data units × 1,024 blocks)
Activation sources		• External request • On-chip module request • Auto request (software trigger equivalent)	(External requests not supported.) • On-chip module request • Software trigger • External interrupt	• External request • On-chip module request • Software trigger
Channel priority		Selectable between the following: • channel 0 > channel 1 > channel 2 > channel 3 • channel 0 > channel 2 > channel 3 > channel 1 • Round robin	Fixed (channel 0 > channel 1 > ... > channel 3)	Fixed (channel 0 > channel 1)
Transfer data	1 data unit	8 bits, 16 bits, 32 bits, 128 bits	8 bits, 16 bits, 32 bits	8 bits, 16 bits, 32 bits
	Block size	—	Data units: 1 to 1,024	Data units: 1 to 1,024
	Cluster size	—	—	Data units: 1 to 8
Transfer modes		None (The transfer mode on the SH is equivalent to normal transfer mode on the RX.)	• Normal transfer mode • Repeat transfer mode • Block transfer mode	• Normal transfer mode • Repeat transfer mode • Block transfer mode • Cluster transfer mode
Bus modes		• Cycle-steal mode • Burst mode	—	—
Address modes		• Single address mode • Dual address mode	—	• Single address mode • Dual address mode
Interrupt request	Transfer-end interrupt	Generated when number of transfers specified by transfer counter finishes		
	Transfer escape-end interrupt	—	Generated after completion of data transfer equivalent to the repeat size or when the extended repeat area overflows.	
Other		—	• Extended repeat area • Event link • Offset address updating	• Extended repeat area

2.6.2 DMAC Block Diagram

Figure 2.13 is a block diagram of the SH7080 Group's DMAC.



[Legend]

SARn:	DMA source address register
DARn:	DMA destination address register
DMATCRn:	DMA transfer count register
CHCRn:	DMA channel control register
DMAOR:	DMA operation register
DEIn:	DMA transfer-end interrupt request to the CPU
n:	0, 1, 2, 3

Figure 2.13 SH7080 Group DMAC Block Diagram

Figure 2.14 is a block diagram of the RX651’s DMACAa.

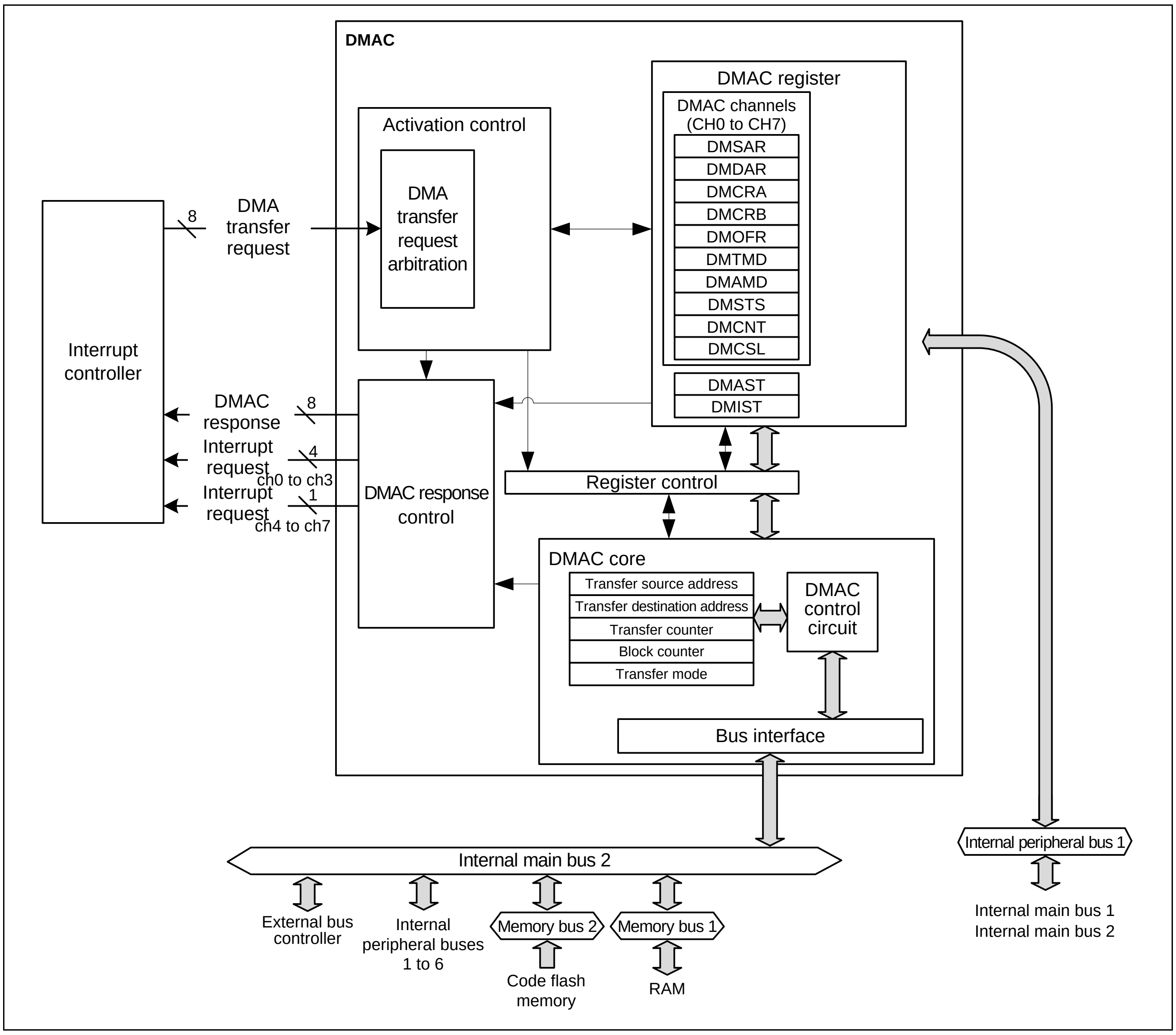
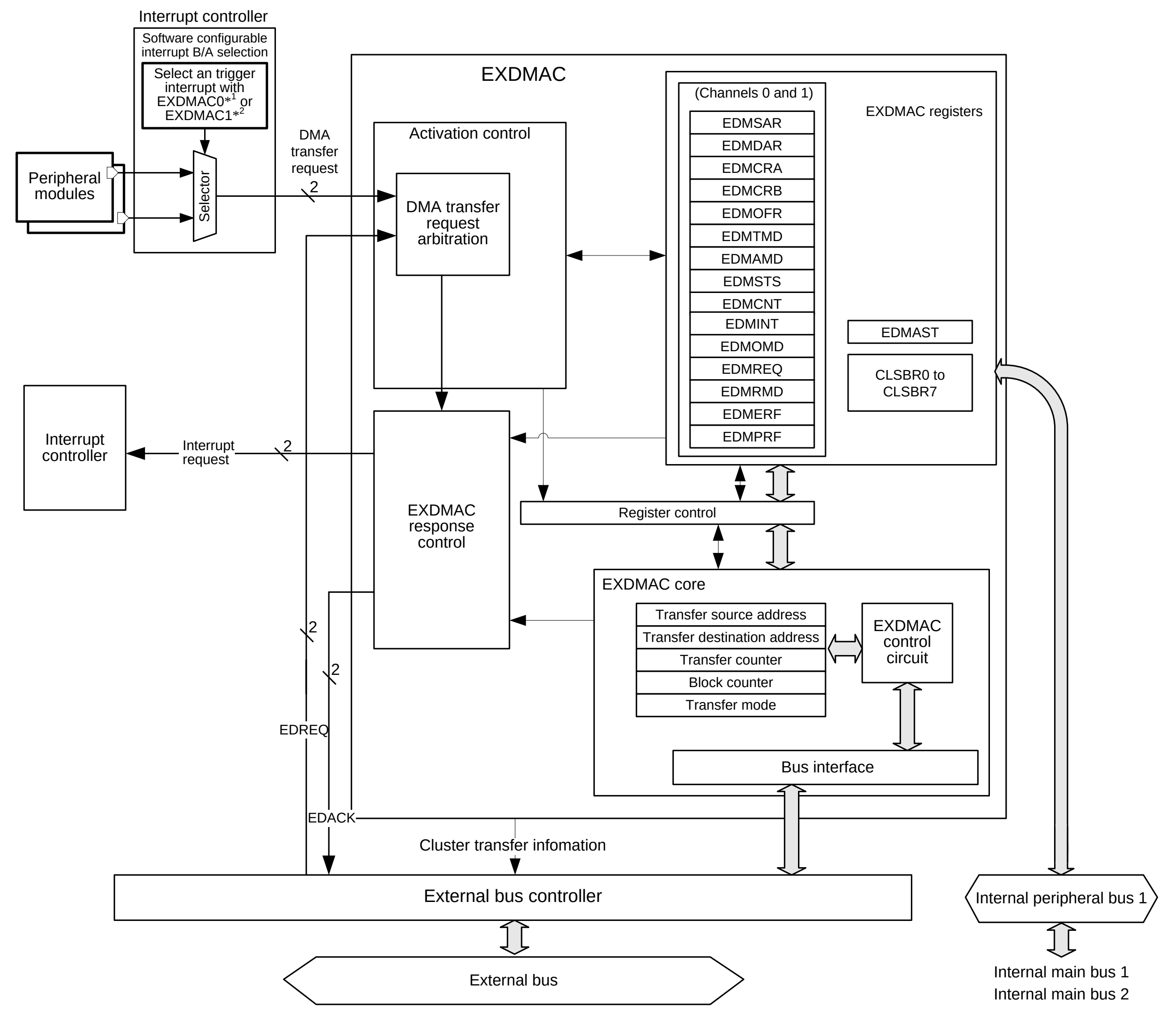


Figure 2.14 RX651 DMACAa Block Diagram

Figure 2.15 is a block diagram of the RX651's EXDMACa.



Note 1. A software configurable interrupt B request from TPU1.TRGA selected in ICU.SLIBR144 or a software configurable interrupt A request from MTU1.TRGA selected in ICU.SLIAR208

Note 2. A software configurable interrupt B request from TPU1.TRGA selected in ICU.SLIBR145 or a software configurable interrupt A request from MTU1.TRGA selected in ICU.SLIAR209

Figure 2.15 RX651 EXDMACa Block Diagram

2.6.3 Register Comparison

Table 2.24 and Table 2.25 provide a comparative listing of the registers of the SH7080 Group and the RX651.

Guide to Symbols in "Changes" Column of Table

◎: Register with same bit assignments on SH7080 Group and RX651

△: Register with different bit assignments on SH7080 Group and RX651

—: Register not present on SH7080 Group or RX651

Table 2.24 SH7080 Group and RX651 Register Comparison (DMAC/DMACAa)

SH7080 Group (DMAC)	RX651 (DMACAa)	Changes
DMAC n: 0 to 3	DMACAa m: 0 to 7	
DMA operation register (DMAOR)	DMAC module start register (DMAST)	△
DMA source address register n (SAR_n)	DMA transfer source address register (DMACm.DMSAR)	◎
DMA destination address register n (DAR_n)	DMA transfer destination address register (DMACm.DMDAR)	◎
DMA transfer count register n (DMATCR_n)	DMA transfer count register (DMACm.DMCRA)	◎
DMA channel control register n (CHCR_n)*1	DMA transfer mode register (DMACm.DMTMD) DMA address mode register (DMACm.DMAMD) DMA interrupt setting register (DMACm.DMINT) DMA transfer enable register (DMACm.DMCNT) DMA status register (DMACm.DMSTS) DMA software start register (DMACm.DMREQ)	△
—	DMA block transfer count register (DMACm.DMCRB)	—
	DMAC activation source flag control register (DMACm.DMCSL)	
	DMA offset register (DMAC0.DMOFR)	
	DMAC74 interrupt status monitor register (DMIST)	

Note 1. On the RX651 transfer request settings from peripheral modules are made by means of the interrupt controller.

Table 2.25 SH7080 Group and RX651 Register Comparison (DMAC/EXDMACa)

SH7080 Group (DMAC)	RX651 (EXDMACa)	Changes
DMAC n: 0 to 3	EXDMACa m: 0 or 1	
DMA operation register (DMAOR)	EXDMAC module start register (EDMAST)	△
DMA source address register n (SAR_n)	EXDMA transfer source address register (EXDMACm.EDMSAR)	◎
DMA destination address register n (DAR_n)	EXDMA transfer destination address register (EXDMACm.EDMDAR)	◎
DMA transfer count register n (DMATCR_n)	EXDMA transfer count register (EXDMACm.EDMCRA)	◎
DMA channel control register n (CHCR_n)*[1]	EXDMA transfer mode register (EXDMACm.EDMTMD) EXDMA address mode register (EXDMACm.EDMAMD) EXDMA interrupt setting register (EXDMACm.EDMINT) EXDMA transfer enable register (EXDMACm.EDMCNT) EXDMA external request sense mode register (EXDMACm.EDMRMD) EXDMA output setting register (EXDMACm.EDMOMD) EXDMA status register (EXDMACm.EDMSTS) EXDMA software start register (EXDMACm.EDMREQ)	△
—	EXDMA block transfer count register (EXDMACm.EDMCRB)	—
	EXDMA offset register (EXDMAC0.EDMOFR)	
	EXDMA external request flag register (EXDMACm.EDMERF)	
	EXDMA peripheral request flag register (EXDMACm.EDMPRF)	
	Cluster buffer register y (CLSBRy) (y = 0 to 7)	

Note 1. On the RX651 transfer request settings from peripheral modules are made by means of the interrupt controller.

2.6.4 Activation Source Settings

On the SH7080 Group activation sources that enable peripheral modules to activate the DMA are specified by setting the resource select bits in the DMA channel control registers (RS[3:0] in CHCR_0 to CHCR_3). On the RX651 DMA activation sources are specified by setting activation source vector numbers in the DMAC trigger select registers (DMRSRm) of the interrupt controller, thereby enabling DMA activation by the corresponding interrupts.

Table 2.26 lists the types of DMA activation sources.

Table 2.26 DMA Activation Source Comparison

DMA Activation Sources	SH7080 Group	RX651	
	DMAC	**DMACAa**	**EXDMACa**
Activation by software	Supported	Supported	Supported
Activation by external device via request pin	Supported (DREQn pin) Rising edge Falling edge Low level High level	Not supported	Supported (DREQm pin) Rising edge Falling edge Low level
Activation by interrupt from external interrupt input pin	Not supported	supported (IRQ pin)	Not supported
Activation by peripheral module	Supported (MTU, ADC, SCI)	Supported (CMT, USB, RSPI, QSPI, SDHI, MMCIF, RIIC, SCI, ICU, PDC, CMT, TPU, S12AD, RNG, ELC)	Supported (TPU, MTU)

n, m: Number of DMA channels (n = 0 to 3, m = 0 or 1)

2.6.5 Transfer Count

The RX651 supports free running operation, in which transfer count is not specified. Table 2.27 lists transfer count settings in normal transfer mode on the SH7080 Group and RX651.

Table 2.27 Transfer Count Setting Values

Transfer count	SH7080 Group	RX651	
	DMAC	**DMACAa**	**EXDMACa**
1	00000001h	00000001h	0001h
65,535	FFFFh	FFFFh (max. transfer count)	FFFFh (max. transfer count)
16,777,215	00FFFFFFh	—	—
16,777,216	00000000h (max. transfer count)	—	—
Free running (no transfer count specified)	—	00000000h	0000h

2.6.6 Transfer Sources and Destinations

Table 2.28 to Table 2.30 list the transfer sources and destinations supported by each DMA controller.

Table 2.28 SH7080 Group DMAC Transfer Sources and Destinations

	Transfer Destination				
Transfer Source	External Device with DACK	External Memory	Memory-Mapped External Device	On-Chip Memory	On-Chip Peripheral Module
External Device with DACK	—	●	●	—	—
External Memory	●	○	○	○	○
Memory-Mapped External Device	●	○	○	○	○
On-Chip Peripheral Module	—	○	○	○	○
On-Chip Memory	—	○	○	○	○

●: Single address mode transfers supported. ○: Dual address mode transfers supported.
—: Transfer not supported

Table 2.29 RX651 DMACAa Transfer Sources and Destinations

	Transfer Destination				
Transfer Source	External Device with DACK	External Memory	Memory-Mapped External Device	On-Chip Memory	On-Chip Peripheral Module
External Device with DACK	—	—	—	—	—
External Memory	—	○	○	○	○
Memory-Mapped External Device	—	○	○	○	○
On-Chip Peripheral Module	—	○	○	○	○
On-Chip Memory	—	○	○	○	○

○: Transfers supported. —: Transfer not supported

Table 2.30 RX651 EXDMACa Transfer Sources and Destinations

	Transfer Destination				
Transfer Source	External Device with EDACK	External Memory	Memory-Mapped External Device	On-Chip Memory	On-Chip Peripheral Module
External Device with EDACK	—	●	●	—	—
External Memory	●	○	○	—	—
Memory-Mapped External Device	●	○	○	—	—
On-Chip Peripheral Module	—	—	—	—	—
On-Chip Memory	—	—	—	—	—

●: Single address mode transfers supported. ○: Dual address mode transfers supported.
—: Transfer not supported

2.6.7 Address Modes

The SH7080 Group has two address modes: single address mode and dual address mode.

The EXDMACa of the RX651 has a single address mode and a dual address mode like the SH7080 Group. In single address mode a DMA transfer can be completed in a single bus cycle. Two bus cycles are required to complete a DMA transfer in dual address mode. On the DMACAa the address mode concept does not apply, but the method of specifying addresses and the operation are equivalent to dual address mode on the SH7080 Group.

2.6.8 Bus Modes

On the SH7080 Group the bus mode can be specified as either cycle-steal mode or burst mode. In cycle-steal mode the bus is released to another bus master when a single transfer finishes. In burst mode the bus is not released after the start of a DMA transfer until the transfer finishes.

On the RX651 it is not possible to specify the bus mode of the DMACAa or EXDMACa. This is because the bus architecture differs from that of the SH7080 Group. The RX651 supports parallel operation when the bus master accesses a different slave. On the RX651 it is possible for the DMAC to perform transfers between the peripheral bus and the external bus while the CPU is accessing the ROM to fetch CPU instructions or the RAM to manipulate operands.

Figure 2.16 shows an example in which the DMAC accesses the peripheral bus and the external bus using internal main bus 2 while the CPU is accessing the code flash memory and RAM.

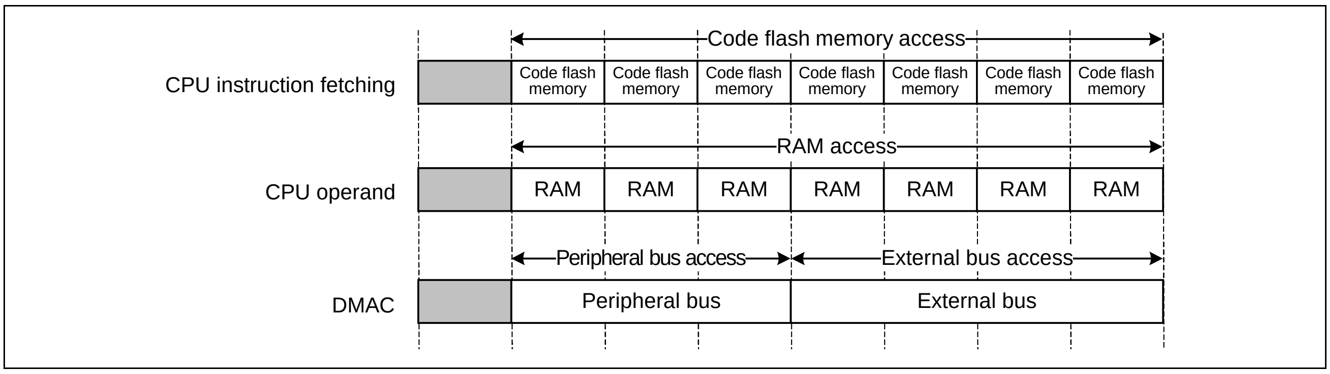


Figure 2.16 RX651 Parallel Bus Operation

2.6.9 Module Stop

On the RX651 the DMACAa and EXDMACa module-stop state is canceled after a reset.

The module-stop setting bit (MSTPCRA.MSTPA28) is common to both the DTCb and DMACAa on the RX651, so module-stop control for these two modules is simultaneous. The EXDMACa has an independent module-stop setting bit (MSTPCRA.MSTPA29), allowing it to be controlled individually.

Refer to 2.16 for information on the module-stop state.

2.6.10 Setting Example for Data Transfer between SCI and On-Chip RAM

In the direct memory access controller (DMAC) setting example shown below for the SH7080 Group and RX651, the DMAC is used to implement data transfer between the serial communication interface (SCI) and the on-chip RAM. Refer to 2.10.8 for an initial setting example for the SCI. In the example shown here, the use of SCI interrupts for DMAC activation is the only portion of the settings that differs between the microcontrollers.

< Specifications >

1. The RSK+RX65N board is used, and the SCI transfer mode is clock-synchronous slave reception.
2. When an SCI2 receive data-full interrupt request occurs, the DMAC transfers one byte of receive data to the receive buffer in the on-chip RAM.
3. When reception of 32 bytes finishes (DMA transfer-end), a DMA transfer-end interrupt is generated.
4. At successful completion, LED1 turns on. LED2 turns on if an error interrupt occurs.

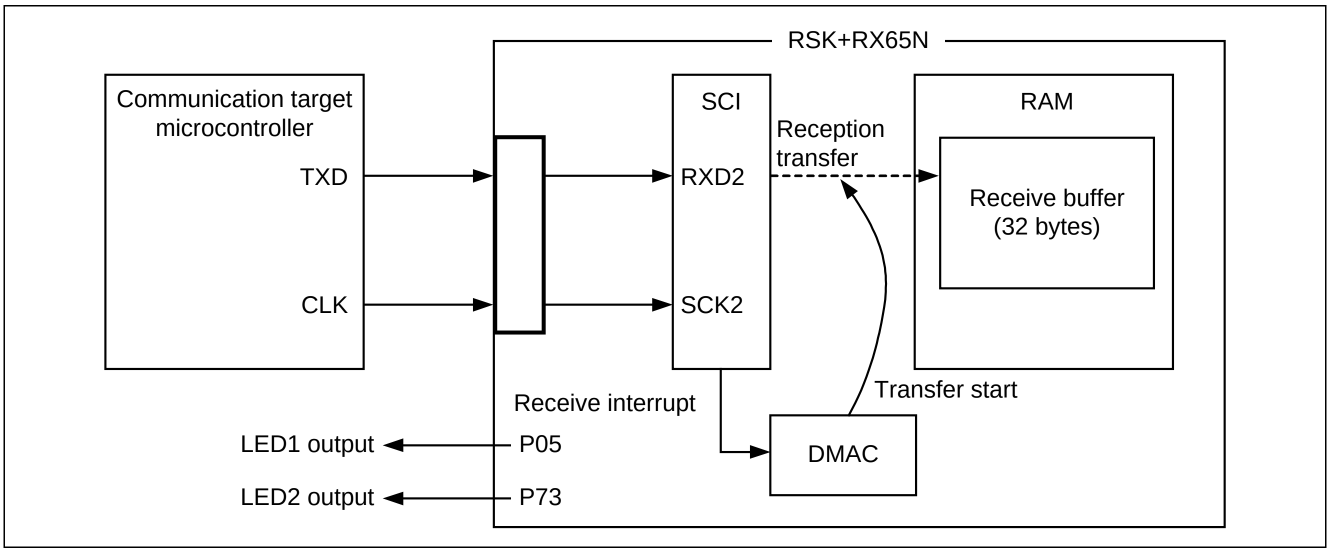


Figure 2.17 Example of Data Transfer between RAM and SCI Using DMAC

Table 2.31 DMAC Transfer Specifications

Item	Receive Transfer	Remarks
Channel used	DMAC0	
Transfer mode	Normal transfer mode	
Transfer count	32	
Transfer size	Byte	
Transfer source	Receive data register (SCI2)	
Transfer destination	On-chip RAM (receive buffer)	
Transfer source address	Fixed	
Transfer destination address	Transfer destination address incremented following transfer	
Activation sources	SCI2 receive data-full interrupt	RXI2 interrupt
Interrupt handling	DMAC transfer-end interrupt	DMAC0I
Pins used	P52/RXD2	
	P51/SCK2	
	P05/GPIO	LED1 output
	P73/GPIO	LED2 output

An initial setting example in which the DMAC is used to transfer data between the SCI and a receive buffer (in on-chip RAM) is shown below.

Table 2.32 DMAC Normal Transfer Initial Setting Example

Procedure		SH7080 Group Setting Example	RX651 Setting Example

1	Cancel module-stop state.	STB.STBCR2.MSTP3 = 0	SYSTEM.PRCR = 0xA502
			SYSTEM.MSTPCRA.MSTPA28 = 0
			SYSTEM.PRCR = 0xA500

2	Make peripheral function settings (SCI initial settings).	Make setting for SCI clock synchronous slave receive operation. Make settings for items 1 to 11 in Table 2.68 for SCI function and ICU function. Enable RXI interrupts and error interrupts. (Make settings for items 10 and after in Table 2.68 after making DMA settings.)	
3	Disable DMAC interrupt requests.	—	ICU.IER0F.IEN0 = 0 (DMAC0I interrupt disabled)
4	Stop DMA transfer.	DMAC0.CHCR0.DE = 0 (DMAC0 operation disabled)	DMAC0.DMCNT.DTE = 0 (DMA transfer disabled)
5	Set DMAC activation source.	—	ICU.DMRSR0 = 62 (vector number set to 62/RXI2)
6	Make DMA address mode settings.	DMAC0.CHCR0.SM[1:0] = 00b (fixed transfer source address mode) DMAC0.CHCR0.DM[1:0] = 01b (increment transfer destination address mode)	DMAC0.DMAMD.SM[1:0] = 00b (fixed transfer source address mode) DMAC0.DMAMD.DM[1:0] = 10b (increment transfer destination address mode)
7	Make DMA transfer mode settings.	DMAC0.CHCR0.RS[3:0] = 1101b (transfer request source specified as SCI_0 (RXI_0)) DMAC0.CHCR0.TB = 0 (cycle-steal mode) DMAC0.CHCR0.TS[1:0] = 00b (transfer data size: 8 bits)	DMAC0.DMTMD.DCTG[1:0] = 01b (transfer requests: peripheral module) DMAC0.DMTMD.MD[1:0] = 00b (transfer mode: normal transfer) DMAC0.DMTMD.SZ[1:0] = 00b (transfer data size: 8 bits)
8	Set transfer source address.	DMAC0.SAR = SCI.SCRDR_0 address	DMAC0.DMSAR = SCI2.RDR address
9	Set transfer destination address.	DMAC0.DAR = receive buffer address	DMAC0.DMDAR = receive buffer address
10	Set transfer size.	DMAC0.DMATCR0 = 32	DMAC0.DMCRA = 32
11	Make interrupt selection setting.	—	DMAC0.DMCSL.DISEL = 0 (activation source interrupt flag 0 cleared at transfer start)
12	Set DMA priority.	DMAC0.DMAOR.PR[1:0] = 00b (priority mode: CH0 > CH1 > CH2 > CH3)	—
13	Set DMA interrupt level.	INTC.IPRC = 0x5000 (DMAC0 interrupt priority set to 5)	ICU.IPR120 = 5 (DMAC0I interrupt level set to 5)
14	Start peripheral function (enable SCI interrupt) (SH7080 Group only).	Make SCI clock-synchronous slave receive start setting. Make settings for items 10 and after in Table 2.68 for SCI function and ICU function to start SCI function operation.	—

15	Enable DMA interrupts.	DMAC0.CHCR0.IE = 1
		(interrupt requests enabled)
			DMAC0.DMINT.DTIE = 1
			(transfer-end interrupt enabled)
			ICU.IER0F.IEN0 = 1
			(DMAC0I interrupt enabled)

16	Enable DMA transfer.	DMAC0.CHCR0.DE = 1 (DMAC0 operation enabled)	DMAC0.DMCNT.DTE = 1 (DMA transfer enabled)

Procedure		SH7080 Group Setting Example	RX651 Setting Example
17	Start peripheral function (enable SCI interrupt) (RX651 only).	—	Make SCI clock-synchronous slave receive start setting. Make settings for items 10 and after in Table 2.68 for SCI function and ICU function to start SCI function operation.
18	Activate DMA module.	DMAC.DMAOR.DME = 1 (DMA master enable)	DMAC.DMAST.DMST = 1 (DMAC activation enabled)

A DMA transfer-end interrupt (DMA0I) is generated when reception of 32 bytes of data finishes. The details of DMA transfer-end interrupt handling are not stipulated. The sample code implements SCI end processing.

2.7 Multi-function Timer Pulse Unit (MTU)

2.7.1 Comparison of Specifications

Multi-function timer pulse unit functionality is provided on the SH7080 Group by the MTU2 and MTU2S and on the RX651 by the MTU3a.

The RX651 includes the MTU functionality of the SH7080 Group (backward compatibility). Table 2.33 lists comparative specifications of the SH7080 Group and RX651.

Table 2.33 Comparison of SH7080 Group and RX651 Specifications (MTU)

		SH7080 Group		**RX651**
Item		**MTU2**	**MTU2S**	**MTU3a**
Functional compatibility by channel	16-bit timer	MTU0	—	MTU0
		MTU1	—	MTU1
		MTU2	—	MTU2
		MTU3	MTU3S	MTU3, MTU6
		MTU4	MTU4S	MTU4, MTU7
		MTU5	MTU5S	MTU5
	32-bit timer	—	—	MTU8
Pulse I/O		Max. 16	Max. 8	Max. 28
Pulse input		3	3	3
Count clock		Selectable for each channel among eight clocks employing the MTU2 clock (MPϕ) and external clocks (TCLKA, TCLKB, TCLKC, and TCLKD)	Selectable for each channel among six clocks employing the MTU2S clock (MIϕ).	Selectable for each channel among 14 clocks employing the peripheral module clock (PCLKA) and external clocks (MTCLKA, MTCLKB, MTCLKC, MTCLKD, and MTIOC1A).
DTC/DMAC activation		DTC/DMAC activation supported	DTC activation supported	DTC/DMAC activation supported
A/D conversion start triggers		Trigger generation supported	Trigger generation supported	Trigger generation supported
Interrupt sources		28	13	43
Noise cancellation		None	None	Ability to enable noise filtering for external clock pins
Other		• Cascade connection	—	• Event link • Cascade connection

2.7.2 Interrupts

The RX651 does not have timer status register (TSR) interrupt flags, but equivalent processing can be accomplished by using the corresponding MTU interrupt request registers of the interrupt controller.

The MTU2S of the SH7080 Group can activate the DTC only, but the MTU2 of the SH7080 Group and the RX651 can activate the DTC and DMAC on all channels.

The RX651 is provided with software configurable interrupt A. The interrupt controller's software configurable interrupt A status flags (PIARk.PIRn) are not cleared automatically, but even if left uncleared they do not affect the generation of interrupt requests.

Refer to 1.8 for information about interrupts.

Table 2.34 List of MTU Interrupt Sources on SH7080 Group and RX651

	SH7080 Group							
	MTU0	**MTU1**	**MTU2**	—	**MTU3 MTU3S**	**MTU4 MTU4S**	**MTU5 MTU5S**	—
	RX651							
Item	**MTU0**	**MTU1**	**MTU2**	**MTU1 MTU2*2**	**MTU3 MTU6**	**MTU4 MTU7**	**MTU5**	**MTU8**
Compare match nA*3	○	○	○	—	○	○	—	○
Input capture nA*3	○	○	○	○	○	○	—	○
Compare match nB*3	○	○	○	—	○	○	—	○
Input capture nB*3	○	○	○	○	○	○	—	○
Compare match nC*3	○	—	—	—	○	○	—	○
Input capture nC*3	○	—	—	—	○	○	—	○
Compare match nD*3	○	—	—	—	○	○	—	○
Input capture nD*3	○	—	—	—	○	○	—	○
Overflow	○	○	○	○	○	○	—	○
Underflow	—	○	○	○	—	○*1	—	—
Compare match nE	○	—	—	—	—	—	—	—
Compare match nF	○	—	—	—	—	—	—	—
Compare match nU*3	—	—	—	—	—	—	○	—
Input capture nU*3	—	—	—	—	—	—	○	—
Compare match nV*3	—	—	—	—	—	—	○	—
Input capture nV*3	—	—	—	—	—	—	○	—
Compare match nW*3	—	—	—	—	—	—	○	—
Input capture nW*3	—	—	—	—	—	—	○	—

n: Channel number ○: Supported —: Not supported

Note 1. Complementary PWM mode only

Note 2. 32-bit access

Note 3. "S" is appended at the end to indicate MTU2S.

2.7.3 Register Comparison

Table 2.35 is a comparative listing of the registers on the SH7080 Group and RX651.

Guide to Symbols in "Changes" Column of Table

◎: Register with same bit assignments on SH7080 Group and RX651

△: Register with different bit assignments on SH7080 Group and RX651

—: Register not present on SH7080 Group or RX651

Table 2.35 SH7080 Group and RX651 Register Comparison (MTU)

Register Name	SH7080 Group (MTU2)	RX651 (MTU3a)	Changes
Timer control register	TCR_0 to TCR_4 TCRU/V/W_5 TCR_3/4S	MTU0.TCR to MTU4.TCR MTU5.TCRU/V/W MTU6/7.TCR	◎
	TCRU/V/W_5S	—	—
	—	MTU8.TCR	—
Timer control register 2	—	MTU0.TCR2 to MTU4.TCR2 MTU6.TCR2 to MTU8.TCR2 MTU5.TCR2U/V/W	—
Timer mode register (SH7080 Group) Timer mode register 1 (RX651)	TMDR_0/3/4 TMDR_3/4S	MTU0/3/4.TMDR1 MTU6/7.TMDR1	◎
	TMDR_1/2	MTU1/2.TMDR1	△
	—	MTU8.TMDR1	—
Timer mode register 2	—	MTU.TMDR2A/B	—
Timer mode register 3	—	MTU1.TMDR3	—
Timer I/O control register	TIORH_0 TIORU/V/W_5	MTU0.TIORH MTU5.TIORU/V/W	△
	TIORL_0 TIOR_1/2 TIORH/L_3/4 TIORH/L_3/4S	MTU0.TIORL MTU1/2.TIOR MTU3/4.TIORH/L MTU6/7.TIORH/L	◎
	TIORU/V/W_5S	—	—
	—	MTU8.TIORH/L	—
Timer compare match clear register	TCNTCMPCLR	MTU5.TCNTCMPCLR	◎
	TCNTCMPCLRS	—	—
Timer interrupt enable register	TIER_0 to TIER_5 TIER_3/4S TIER2_0	MTU0.TIER to MTU5.TIER MTU6/7.TIER MTU0.TIER2	◎
	TIER_5S	—	—
	—	MTU8.TIER	—
Timer status register	TSR_1 to TSR_4 TSR_3/4S	MTU1.TSR to MTU4.TSR MTU6/7.TSR	△
	TSR_0 TSR2_0 TSR_5 TSR_5S	—	—

Register Name	SH7080 Group (MTU2)	RX651 (MTU3a)	Changes
Timer buffer operation transfer mode register	TBTM_0/3/4 TBTM_3/4S	MTU0/3/4.TBTM MTU6/7.TBTM	◎
Timer input capture control register	TICCR	MTU1.TICCR	◎
Timer A/D conversion start request control register	TADCR, TADCRS	MTU4.TADCR, MTU7.TADCR	◎
Timer A/D conversion start request cycle set register	TADCORA/B_4 TADCORA/B_4S	MTU4.TADCORA/B MTU7.TADCORA/B	◎
Timer A/D conversion start request cycle set buffer register	TADCOBRA/B_4 TADCOBRA/B_4S	MTU4.TADCOBRA/B MTU7.TADCOBRA/B	◎
Timer counter	TCNT_0 to TCNT_4 TCNTU/V/W_5 TCNT_3/4S	MTU0.TCNT to MTU4.TCNT MTU5.TCNTU/V/W MTU6/7.TCNT	◎
	TCNTU/V/W_5S	—	—
	—	MTU8.TCNT	—
Timer longword counter	—	MTU1.TCNTLW	—
Timer general register	TGR_0 (A to F) TGR_1/2 (A, B) TGR_3/4 (A to D) TGR_5 (U, V, W) TGR_3/4S (A to D)	MTU0.TGR (A to F) MTU1/2.TGR (A, B) MTU3/4.TGR (A to D) MTU5.TGR (U, V, W) MTU6/7.TGR (A to D)	◎
	TGR_5S (U, V, W)	—	—
	—	MTU3/6.TGR (E) MTU4/7.TGR (E, F) MTU8.TGR (A to D)	—
Timer longword general register	—	MTU1.TGRA/BLW	—
Timer start register	TSTR	MTU.TSTRA	△
	TSTRS, TSTR_5	MTU.TSTRB, MTU5.TSR	◎
	TSTR_5S	—	—
Timer synchronous register	TSYR, TSYRS	MTU.TSYRA, MTU.TSYRB	◎
Timer synchronous clear register	TSYCRS	MTU6.TSYCR	◎
Timer counter synchronous start register	TCSYSTR	MTU.TCSYSTR	◎
Timer read/write enable register	TRWER, TRWERS	MTU.TRWERA, MTU.TRWERB	◎
Timer output master enable register	TOER, TOERS	MTU.TOERA, MTU.TOERB	◎
Timer output control register 1	TOCR1, TOCR1S	MTU.TOCR1A, MTU.TOCR1B	◎
Timer output control register 2	TOCR2, TOCR2S	MTU.TOCR2A, MTU.TOCR2B	◎
Timer output level buffer register	TOLBR, TOLBRS	MTU.TOLBRA, MTU.TOLBRB	◎
Timer gate control register (SH7080 Group) Timer gate control register A (RX651)	TGCR	MTU.TGCRA	◎
	TGCRS	—	—
Timer sub counter	TCNTS, TCNTSS	MTU.TCNTSA, MTU.TCNTSB	◎
Timer period data register	TCDR, TCDRS	MTU.TCDRA, MTU.TCDRB	◎
Timer period buffer register	TCBR, TCBRS	MTU.TCBRA, MTU.TCBRB	◎

Register Name	SH7080 Group (MTU2)	RX651 (MTU3a)	Changes
Timer dead time data register	TDDR, TDDRS	MTU.TDDRA, MTU.TDDRB	◎
Timer dead time enable register	TDER, TDERS	MTU.TDERA, MTU.TDERB	◎
Timer buffer transfer set register	TBTER, TBTERS	MTU.TBTERA, MTU.TBTERB	◎
Timer waveform control register	TWCR, TWCRS	MTU.TWCRA, MTU.TWCRB	◎
Timer interrupt skipping set register (SH7080 Group) Timer interrupt skipping set register 1 (RX651)	TITCR, TITCRS	MTU.TITCR1A, MTU.TITCR1B	◎
Timer interrupt skipping set register 2	—	MTU.TITCR2A, MTU.TITCR2B	—
Timer interrupt skipping counter (SH7080 Group) Timer interrupt skipping counter 1 (RX651)	TITCNT, TITCNTS	MTU.TITCNT1A, MTU.TITCNT1B	◎
Timer interrupt skipping counter 2	—	MTU.TITCNT2A, MTU.TITCNT2B	—
Timer interrupt skipping mode register	—	MTU.TITMRA, MTU.TITMRB	—
Noise filter control register n	—	NFCR0 to NFCR4 in MTU0 to MTU4 NFCR6 to NFCR8 in MTU6 to MTU8 MTU0.NFCRC	—
Noise filter control register 5	—	MTU5.NFCR5	—

2.7.4 Module Stop

As on the SH7080 Group, the MTU3a of the RX651 is set to the module-stop state after a reset, and no clock is supplied.

Refer to 2.16 for information on the module-stop state.

2.7.5 Output Compare Match Setting Example

In the setting example shown below for the SH7080 Group and RX651, the multi-function timer pulse unit (MTU) is used to implement output compare match functionality.

< Specifications >

1. The RSK+RX65N board is used.
2. The MTU4 is used to output pulses with 50% duty of the specified cycle. The specified cycle is fixed at 1 ms.

Table 2.36 MTU Output Compare Match Specifications

Item	Description	Remarks
Count clock	PCLKA/1	PCLKA = 120 MHz
Operating mode	Normal mode	
Synchronous operation	Not used.	
Counter clear source	TGRA output compare	
Timer general register	Used as output compare register.	
Pin used	PA0/MTIOC4A	For pulse output

Figure 2.18 illustrates the operation. In this setting example no software processing is involved after the initial settings to the MTU. The pulse output is generated automatically in hardware.

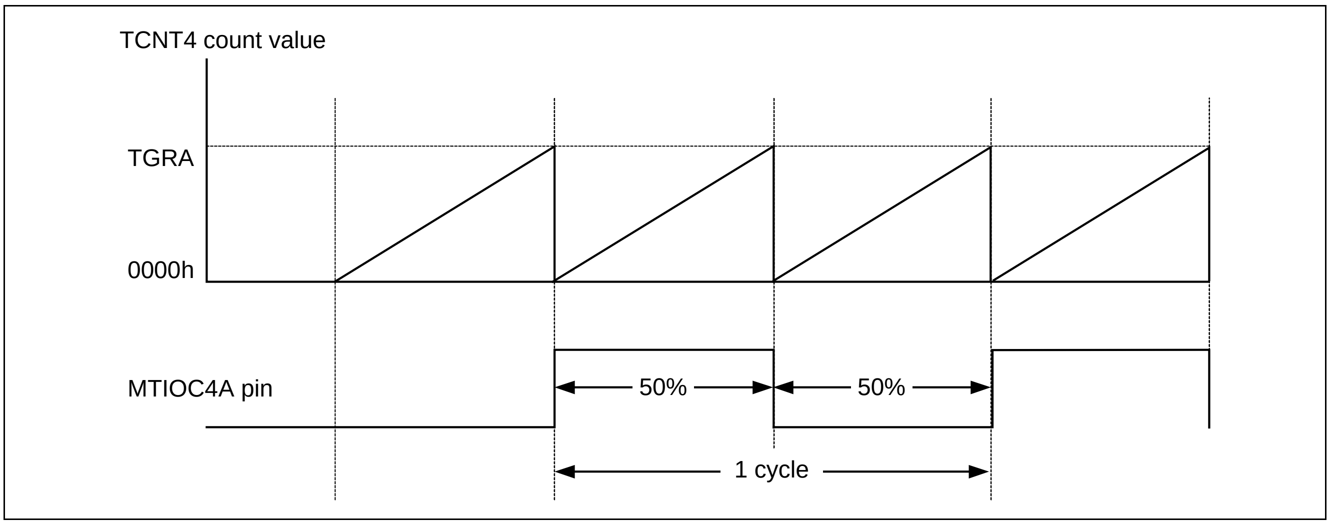


Figure 2.18 MTU Output Compare Match Operation

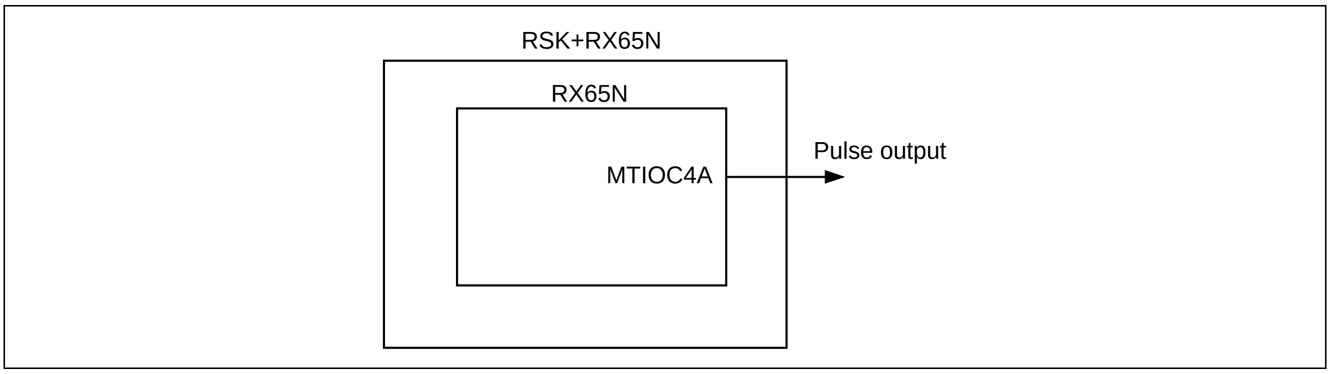


Figure 2.19 MTU Output Compare Match Connection Diagram

Table 2.37 MTU Output Compare Match Initial Setting Example

Procedure		SH7080 Group Setting Example MPϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock A): 120 MHz

1	Cancel module-stop state.	STBCR4.MSTP22 = 0	SYSTEM.PRCR = A502h
			SYSTEM.MSTPCRA.MSTPA9 = 0
			SYSTEM.PRCR = A500h

2	Enable register read/write.	TRWER.RWE = 1 (read/write enabled)	MTU.TRWERA.RWE = 1 (access enabled to registers/counters with erroneous write protection)

3	Stop MTU.	TSTR.CST4 = 0 (TCNT stopped)
		TSYR.SYNC4 = 0
		(independent operation enabled)
		TCNT = 0000h (TCNT_4 cleared)
			MTU.TSTRA.CST4 = 0 (TCNT stopped)
			MTU.TSYRA.SYNC4 = 0
			(TCNT operates independently)
			MTU4.TCNT = 0000h (timer counter cleared)

4	Select counter clock; select edge.	TCR.TPSC[2:0] = 000b (internal clock: set as MPϕ/1) TCR.CKEG[1:0] = 00b (counting at rising edge)	MTU4.TCR.TPSC[2:0] = 000b (internal clock: set as PCLKA/1) MTU4.TCR.CKEG[1:0] = 00b (counting at rising edge) MTU4.TCR2.TPSC2[2:0] = 000b (internal clock: set as PCLKA/1)
5	Make counter operation and TCNT clear source settings.	TGRA compare match, input capture, and TCNT clear source TGRA TCR.CCLR[2:0] = 001b	MTU4.TGRA compare match/input capture/TCNT clear source TGRA MTU4.TCR.CCLR[2:0] = 001b (TCNT cleared at TGRA compare match)
6	Enable TOIC4A output (MTU3 and MTU4 only).	TOER.OE4A = 1	MTU.TOERA.OE4A = 1
7	Make timer I/O control settings.	Output compare register: TGRA Initial output: 0, output toggled at compare match TIORH_4.IOA[3:0] = 0011b	Output compare register: MTU4.TGRA Initial output: 0, output toggled at compare match MTU4.TIORH.IOA[3:0] = 0011b
8	Set TGRA (setting value: 1/2 cycle duration).	TGRA = 4E1Fh	MTU4.TGRA = EA5Fh
9	Make timer mode register settings.	TMDR.BFA = 0 (normal operation) TMDR.MD[3:0] = 0 (normal operation)	MTU4.TMDR1.BFA = 0 (normal operation) MTU4.TMDR1.MD[3:0] = 0000b (normal operation)
10	Disable register read/write.	TRWER.RWE = 0 (read/write disabled)	MTU.TRWERA.RWE = 0 (access disabled to registers/counters with erroneous write protection)

11	Make I/O port settings (pin I/O and pin function settings).	PFC settings
		PEIOR.PE12IOR = 1 (output)
		PECRL4.PE12MD = 001b (TIOC4A selected)
			MTIOC4A pin settings in MPC
			PORTA.PDR.B0 = 1 (set to output)
			PORTA.PMR.B0 = 0 (GPIO)
			MPC.PWPR.B0WI = 0 (PFSWE write enabled)
			MPC.PWPR.PFSWE = 1 (PFS write enabled)
			MPC.PA0PFS = 01h (pin function setting)
			MPC.PWPR.PFSWE = 0 (PFS write disabled)
			MPC.PWPR.B0WI = 1
			(PFSWE write disabled)
			PORTA.PMR.B0 = 1 (peripheral function)

12	Enable timer operation.	TSTR.CST4 = 1 (TCNT_4 starts counter operation)	MTU.TSTRA.CST4 = 1 (TCNT4 starts counter operation)

2.7.6 Input Capture Setting Example

In the setting example shown below for the SH7080 Group and RX651, the input capture function of the multi-function timer pulse unit (MTU) is used to measure the input pulse width.

< Specifications >

1. The RSK+RX65N board is used.
2. The high duration of the pulse input on the pin is measured, and the result is stored in the RAM.
3. If the pulse width measurement range*[1] is exceeded, LED1 turns on and processing ends.

Note 1. Measurement is not possible when the TCNT overflow count exceeds 0xFFFF.

Table 2.38 MTU Input Capture Specifications

Item	Description	Remarks
Count clock	Rising edge of PCLKA/1	PCLKA = 120 MHz
Operating mode	Normal mode	
Synchronous operation	Not used.	
Counter clear source	TGRA Input capture	
Timer general register	Input capture register	
Pins used	PA0/MTIOC4A (input capture at both edges)	Pulse input
	P05 (GPIO)	LED1 output
Interrupt sources	MTU4 input capture A interrupt Overflow interrupt	

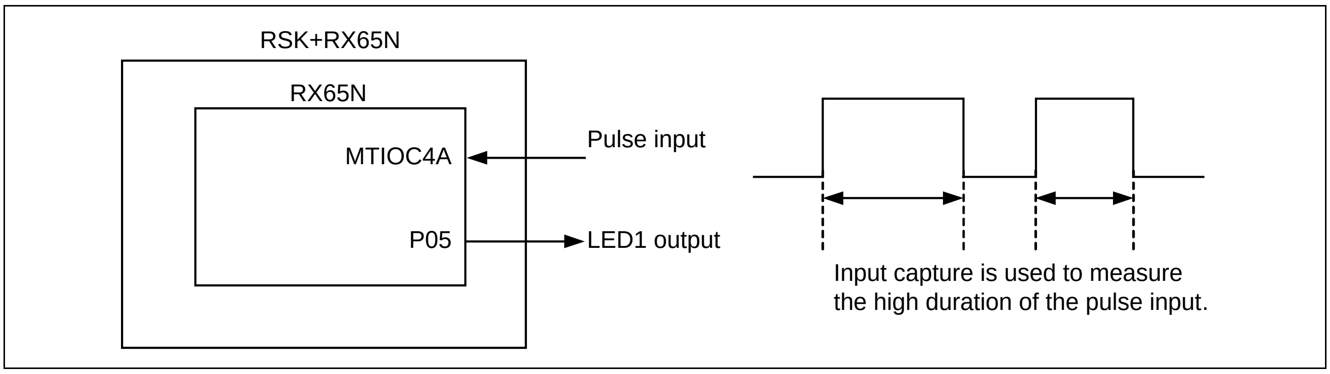
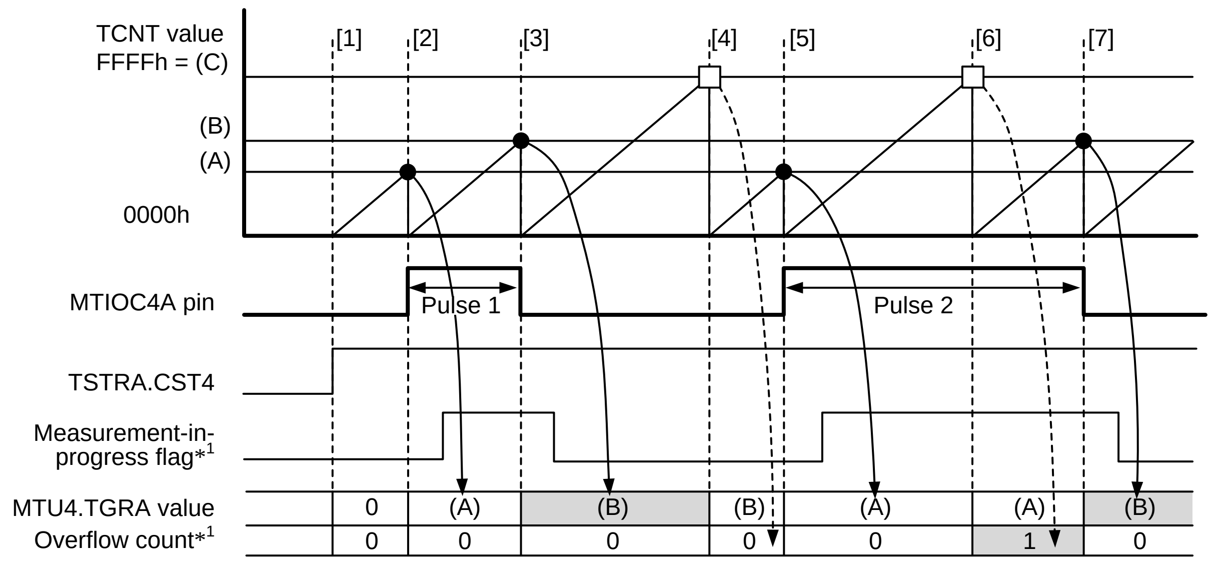

RSK+RX65N
RX65N
MTIOC4A
P05
Pulse input
LED1 output
Input capture is used to measure the high duration of the pulse input.

Figure 2.20 MTU Pin Connections



●: Input capture generation timing ([2], [3], [5], and [7] in figure)
The value of TCNT is transferred to TGRA. Also, an input capture interrupt is generated at the same time.

□: TCNT overflow generation timing ([4] and [6] in figure)
An overflow interrupt is generated when a TCNT overflow occurs. When the value of TCNT is FFFFh, indicating an edge input, the value of TCNT is transferred to TGRA and no overflow interrupt is generated, but an input capture interrupt is generated.

Shaded portion: Counter values used as measurement values
Overflow count (set in variable area)

Note 1. Processed by software.

Figure 2.21 MTU Input Capture Operation

< Description of Pulse Width Measurement Operation >

The operating principle of pulse width measurement of pulses 1 and 2 is described below, assuming the conditions shown in Figure 2.21 above.

[1] MTU4 starts counting when the TSTR.CST4 bit is set to 1 (start count).

[2] An input capture interrupt is generated when a rising edge is input on the MTIOC4A pin. The handler of this interrupt first confirms that the pin is in the high state, then it sets the measurement-in-progress flag to 1, clears the overflow count to 0, and starts measuring pulse 1.

[3] An input capture interrupt is generated when a falling edge is input on the MTIOC4A pin. The handler of this interrupt first confirms that the pin is in the low state, then it determines that measurement of the width of pulse 1 has finished, clears the measurement-in-progress flag to 0, and calculates the width of pulse 1 based on the MTU4.TCNT overflow count (0) and the value of MTU4.TGRA (B).

[4] An overflow interrupt is generated when MTU4.TCNT overflows, and the handler of this interrupt checks the measurement-in-progress flag. The value of the measurement-in-progress flag is 0, so the overflow count is not incremented.

[5] An input capture interrupt is generated when a rising edge is input on the MTIOC4A pin. The handler of this interrupt first confirms that the pin is in the high state, then it sets the measurement-in-progress flag to 1, clears the overflow count to 0, and starts measuring pulse 2.

[6] An overflow interrupt is generated when MTU4.TCNT overflows, and the handler of this interrupt checks the measurement-in-progress flag. The value of the measurement-in-progress flag is 1, so the overflow count is incremented, changing the overflow count from (0) to (1).

[7] An input capture interrupt is generated when a falling edge is input on the MTIOC4A pin. The handler of this interrupt first confirms that the pin is in the low state, then it determines that measurement of the width of pulse 2 has finished, clears the measurement-in-progress flag to 0, and calculates the width of pulse 2 based on the MTU4.TCNT overflow count (1) and the value of MTU4.TGRA (B).

Table 2.39 MTU Input Capture Initial Setting Example

Procedure		SH7080 Group Setting Example MPϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock A): 120 MHz

1	Cancel module-stop state.	STBCR4.MSTP22 = 0	SYSTEM.PRCR = A502h
			SYSTEM.MSTPCRA.MSTPA9 = 0
			SYSTEM.PRCR = A500h

2	Enable register read/write.	TRWER.RWE = 1 (read/write enabled)	MTU.TRWERA.RWE = 1 (access enabled to registers/counters with erroneous write protection)

3	Disable interrupts.	TIER.TGIEA = 0 (TGIA disabled)
		TIER.TCIEV = 0 (TCIV disabled)
			ICU.IER1A.IEN2 = 0
			(vector 210: software configurable interrupt A)
			ICU.IER1A.IEN3 = 0
			(vector 211: software configurable interrupt A)
			MTU4.TIER.TGIEA = 0 (TGIA4 disabled)
			MTU4.TIER.TCIEV = 0 (TCIV4 disabled)

4	Stop MTU.	TSTR.CST4 = 0 (TCNT stopped)
		TSYR.SYNC4 = 0
		(independent operation enabled)
		TCNT = 0000h (TCNT_4 cleared)
		TGRA = 0000h (TGRA_4 cleared)
			MTU.TSTRA.CST4 = 0 (TCNT stopped)
			MTU.TSYR.SYNC4 = 0
			(independent operation enabled)
			MTU4.TCNT = 0000h (TCNT cleared)
			MTU4.TGRA = 0000h (TGRA cleared)

5	Select counter clock; select edge.	TCR.TPSC[2:0] = 000b (internal clock: set as MPϕ/1) TCR.CKEG[1:0] = 00b (counting at rising edge)	MTU4.TCR.TPSC[2:0] = 000b (internal clock: set as PCLKA/1) MTU4.TCR.CKEG[1:0] = 00b (counting at rising edge) MTU4.TCR2.TPSC2[2:0] = 000b (internal clock: set as PCLKA/1)
6	Make counter operation and TCNT clear source settings.	TGRA compare match, input capture, and TCNT clear source TGRA TCR.CCLR[2:0] = 001b	MTU4.TGRA compare match, input capture, and TCNT clear source TGRA MTU4.TCR.CCLR[2:0] = 001b
7	Make timer I/O control settings.	TGRA: input capture register Input capture at both edges on input pin TIOC4A TIORH.IOA[3:0] = 1010b	MTU4.TGRA: input capture register Input capture at both edges on input pin MTIOC0A MTU4.TIORH.IOA[3:0] = 1010b
8	Make timer mode register settings.	TMDR.BFA = 0 (normal operation) TMDR.MD[3:0] = 0 (normal operation)	MTU4.TMDR1.BFA = 0 (normal operation) MTU4.TMDR1.MD[3:0] = 0 (normal operation)

9	Make software configurable interrupt source settings.	—	ICU.SLIAR210 = 21 (TGIA4)
			ICU.SLIAR211 = 25 (TCIV4)
			ICU.SLIPRCR.WPRC = 1*1
			(software configurable interrupt source select register write protection)
			Confirm that ICU.SLIPRCR.WPRC = 1.

10	Make interrupt priority register settings.	INTC.IPRF.WORD = 5600h
		(TGIA: level 5, TCIV: level 6)
			ICU.IPR210 = 5 (interrupt priority level: 5)
			ICU.IPR211 = 6 (interrupt priority level: 6)

11	Clear the interrupt source.	—	ICU.IR210 = 0 (TGIA4)
			ICU.IR211 = 0 (TCIV4)

Procedure		SH7080 Group Setting Example MPϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock A): 120 MHz

12	Make I/O port settings (pin I/O and pin function settings).	PFC settings
		PEIOR.PE12IOR = 0 (input)
		PECRL4.PE12MD = 001b (TIOC4A selected)
			MTIOC4A pin settings in MPC
			PORTA.PDR.B0 = 0 (set to input)
			PORTA.PMR.B0 = 0 (GPIO)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1 (PFS write enabled)
			MPC.PA0PFS = 01h (pin function setting)
			MPC.PWPR.PFSWE = 0 (PFS write disabled)
			MPC.PWPR.B0WI = 1
			PORTA.PMR.B0 = 1 (peripheral function)

13	Enable interrupts.	TIER.TGIEA = 1 (TGIA enabled)
		TIER.TCIEV = 1 (TCIV enabled)
			MTU4.TIER.TGIEA = 1 (TGIA4 enabled)
			MTU4.TIER.TCIEV = 1 (TCIV4 enabled)
			ICU.IER1A.IEN2 = 1
			(vector 210 and TGIA4 enabled)
			ICU.IER1A.IEN3 = 1
			(vector 211 and TCIV4 enabled)

14	Enable timer operation.	TSTR.CST4 = 1 (TCNT_4 count operation)	MTU.TSTRA.CST4 = 1 (TCNT4 count operation)

Note 1. Once ICU.SLIPRCR.WPRC is set to 1 it cannot be cleared to 0 by software.

2.8 Port Output Enable (POE)

2.8.1 Comparison of Specifications

Port output enable functionality is provided on the SH7080 Group by the POE and on the RX651 by the POE3a.

The RX651 includes the POE functionality of the SH7080 Group (backward compatibility). Table 2.40 lists comparative specifications of the SH7080 Group and RX651.

Table 2.40 Comparison of SH7080 Group and RX651 Specifications (POE)

Item	SH7080 Group (POE)	RX651 (POE3a)
Clock source	Peripheral clock (Pϕ)	Peripheral module clock (PCLKB)
Pins subject to high-impedance control	• MTU0 pins • MTU2 and MTU2S high-current pins — MTU3 pins — MTU4 pins — MTU3S pins — MTU4S pins	• MTU0 pins • MTU complementary PWM output pins — MTU3 pins — MTU4 pins — MTU6 pins — MTU7 pins
High-impedance request generation conditions	• Change in input pin state — Falling edge — Low level for Pϕ/8 × 16 cycles — Low level for Pϕ/16 × 16 cycles — Low level for Pϕ/128 × 16 cycles • Combined output signal level match for 1 cycle or more (short) • Register setting	• Change in input pin state — Falling edge — Low level for PCLKB/8 × 16 cycles — Low level for PCLKB/16 × 16 cycles — Low level for PCLKB/128 × 16 cycles • Combined output signal level match for 1 cycle or more (short) • Register setting • Detection of clock generation circuit oscillation stop
Interrupt sources	• High-impedance request by change in input pin state • High-impedance request by output signal level comparison	• High-impedance request by change in input pin state • High-impedance request by output signal level comparison
Other	—	Ability to add high-impedance control conditions for MTU complementary PWM output pins and MTU0 pins

2.8.2 Input/Output Pins

On the SH7080 Group the only supported input pins are POE0# to POE8# for the MTU, but on the RX651, in addition to the MTU input pins, POE10# and POE11# input signals are supported.

On the SH7080 Group the MTU0 pins are high-impedance only when assigned as general I/O pins or to the MTU2 or MTU2S function. On the RX651 the MTU complementary PWM output pins and multiplexed MTU0 pins are high-impedance when assigned to other than the MTU function.

Table 2.41 lists the input pins on the SH7080 Group and RX651, and Table 2.42 provides a comparative listing of output pin combinations.

Table 2.41 POE Input Pins

SH7080 Group	RX651	Subject to High-Impedance Control
POE0# to POE3#	POE0#	MTU3 and MTU4 pins*[1]
POE4# to POE7#	POE4#	SH7080 Group: MTU3S and MTU4S pins RX651: MTU6 and MTU7 pins*[1]
POE8#	POE8#	MTU0 pins*[1]
—	POE10#	*[1]
—	POE11#	*[1]

Note 1. On the RX651 the addition of high-impedance control conditions enables control of other pins as well.

Table 2.42 POE Output Pin Combinations

SH7080 Group	RX651	Subject to High-Impedance Control
TIOC3B and TIOC3D	MTIOC3B and MTIOC3D	MTU3 and MTU4 pins
TIOC4A and TIOC4C	MTIOC4A and MTIOC4C	
TIOC4B and TIOC4D	MTIOC4B and MTIOC4D	
TIOC3BS and TIOC3DS	MTIOC6B and MTIOC6D	SH7080 Group: MTU3S and MTU4S pins RX651: MTU6 and MTU7 pins
TIOC4AS and TIOC4CS	MTIOC7A and MTIOC7C	
TIOC4BS and TIOC4DS	MTIOC7B and MTIOC7D	

2.8.3 Register Comparison

On the SH7080 Group the port impedance state is specified by making settings to the port output enable control registers (POECR1 and POECR2). On the RX651 the port impedance state is specified by making settings to the port output enable control registers (POECR1 and POECR2), and the ports assigned to the various pins are specified by making settings to the pin select registers (M0SELR1 and M0SELR2, M3SELR, and M4SELR1 and M4SELR2) of the MTU channels.

Table 2.43 is a comparative listing of the registers on the SH7080 Group and RX651.

Guide to Symbols in "Changes" Column of Table

◎: Register with same bit assignments on SH7080 Group and RX651

△: Register with different bit assignments on SH7080 Group and RX651

—: Register not present on SH7080 Group or RX651

Table 2.43 SH7080 Group and RX651 Register Comparison (POE)

SH7080 Group (POE)	RX651 (POE3a)	Changes
Input level control/status register 1 (ICSR1)	Input level control/ status register 1 (ICSR1)	△
Input level control/status register 2 (ICSR2)	Input level control/ status register 2 (ICSR2)	△
Input level control/status register 3 (ICSR3)	Input level control/ status register 3 (ICSR3)	◎
—	Input level control/ status register 4 (ICSR4) Input level control/ status register 5 (ICSR5) Input level control/ status register 6 (ICSR6)	—
Output level control/status register 1 (OCSR1)	Output level control/ status register 1 (OCSR1)	◎
Output level control/status register 2 (OCSR2)	Output level control/ status register 2 (OCSR2)	◎
Software port output enable register (SPOER)	Software port output enable register (SPOER)	△
Port output enable control register 1 (POECR1)	Port output enable control register 1 (POECR1) MTU0 pin select register 1 (M0SELR1) MTU0 pin select register 2 (M0SELR2)	△
Port output enable control register 2 (POECR2)	Port output enable control register 2 (POECR2) MTU3 pin select register (M3SELR) MTU4 pin select register 1 (M4SELR1) MTU4 pin select register 2 (M4SELR2)	△
—	Port output enable control register 4 (POECR4) Port output enable control register 5 (POECR5)	—
	Active level register 1 (ALR1)	—

2.8.4 High-Impedance Control by Oscillation Stop Detection

The RX651 provides the ability to transition user-specified MTU complementary PWM output pins or MTU0 pins to the high-impedance state when oscillation stop is detected by the oscillation stop detection function of the clock generation circuit.

Pins that transition to the high-impedance state when oscillation stop is detected revert to the default state after a reset, but the high-impedance state is canceled by means of register settings.

2.8.5 Addition of High-Impedance Control Conditions

The RX651 supports the addition of high-impedance control conditions for the MTU complementary PWM output pins and MTU0 pins. Table 2.44 lists the high-impedance control conditions that can be added.

Table 2.44 Additional High-Impedance Control Conditions on RX651

Subject to High-Impedance Control	Additional High-Impedance Control Conditions
MTU3 and MTU4 pins	Input level detection on POE4#, POE8#, POE10#, and POE11#
MTU6 and MTU7 pins	Input level detection on POE0#, POE8#, POE10#, and POE11#.
MTU0 pins	Input level detection on POE0#, POE4#, POE10#, and POE11#.

2.8.6 Interrupts

On the RX651 POE3a is assigned to group interrupt BL1. The interrupt controller’s group BL1 interrupt status flag (GRPBL1.ISn) is automatically cleared when the corresponding bit in the module’s status register is cleared.

Refer to 1.8 for information about interrupts.

2.9 Watchdog Timers (WDT)

2.9.1 Comparison of Specifications

The SH7080 Group incorporates the WDT as its watchdog timer module. The RX651 incorporates, in addition to the WDTA, the IWDTa, which operates on a dedicated independent clock and is able to operate when the microcontroller is in the low-power-consumption state.

Table 2.45 lists comparative specifications for the SH7080 Group and RX651.

Table 2.45 Comparison of SH7080 Group and RX651 Specifications (WDT)

Item	SH7080 Group (WDT)	RX651 (WDTA, IWDTa)
Clock sources	Peripheral clock (Pϕ)	WDTA: Peripheral module clock (PCLKB) IWDTa: IWDT dedicated clock (IWDTCLK) Note that PCLKB $\geq 4 \times$ IWDTCLK
Clock frequency division ratio	Pϕ /1, 4, 16, 32, 64, 256, 1024, 4096	WDTA: PCLKB/4, 64, 128, 512, 2048, 8192 IWDTa: IWDTCLK/1, 16, 32, 64, 128, 256
Count operation	8-bit up-counter	14-bit down-counter
Operating modes	• Watchdog timer mode • Interval timer mode	None • Reset output enabled (equivalent to watchdog timer mode) • Interrupt requests enabled (equivalent to interval timer mode)
Count start condition	• Timer enable bit setting • After internal reset caused by overflow	In auto-start mode • After a reset • After an underflow • After a refresh error In register start mode • Refresh operation
Count stop condition	• Timer enable bit setting • After internal reset caused by overflow • Power-on reset caused by RES pin (counter and setting initialization)	• Reset (counter and setting initialization) • Underflow • Refresh error
Operation at overflow/ underflow	Watchdog timer mode • Internal reset (power-on reset and manual reset) • WDTOVF output Interval timer mode • Interrupt	When reset output enabled • Internal reset When interrupt request output enabled • Interrupt
Interrupt sources	• Overflow of up-counter	• Underflow of down-counter • Refresh error
Other	—	• Event link (IWDTa only) • Window function • Also operates in low-power-consumption state (IWDTa only) • Settings made in option function select register 0 in auto-start mode — Clock division ratio — Refresh window start/end — Timeout period — Underflow operation with interrupt requests/resets enabled

2.9.2 Count Start Conditions

On the SH7080 Group count operation starts when 1 is written to the timer enable bit. The RX651 supports a register start mode, in which count operation is started with a write to a register (writing a setting to the option function select register), as on the SH7080 Group, and an auto-start mode, in which count operation starts automatically after a reset.

When auto-start mode is selected on the RX651, count operation starts automatically after a reset, in accordance with the setting of option function select register (OFS0). When register start mode is selected, count operation is started by a refresh, after the appropriate register settings are made following reset cancelation.

2.9.3 Refresh Operation

On the RX651 the count is refreshed after 00h and then FFh is written to the WDT refresh register (WDTRR). Writes to the WDT refresh register must take place within the refresh-enabled interval. To refresh the count of IWDTa, perform the same write operation to the IWDT refresh register (IWDTRR) within the refresh-enabled interval.

Table 2.46 Comparison of Refresh Operation

Item	SH7080 Group	RX651 (WDTA)
Refresh condition	Write to watchdog timer counter (WTCNT)	00h and then FFh written to refresh register (WDTRR) within refresh-enabled interval
Counter initial value after refresh	Value written to watchdog timer counter (WTCNT)	Value selected by timeout period selection bits (WDTCR.TOPS)

2.9.4 Register Write Limitations

On the SH7080 Group writing to the WDT registers must be in the form of word-size writes in which the top byte has a value of 5Ah and the bottom byte contains the write data. Reads are performed in byte units.

On the RX651 writes to the WDT control register (WDTCR) and WDT reset control register (WDTRCR) can only take place in the interval from reset cancelation to reset operation. For the IWDTa this applies to the IWDT control register, IWDT reset control register, and IWDT count stop control register.

2.9.5 Interrupts

On the RX651 WDTA and IWDTa interrupts may be non-maskable or maskable. The interrupt controller interrupt status flag (IRn.IR) is cleared automatically when the corresponding interrupt is accepted.

Refer to 1.8 for information about interrupts.

2.9.6 All-Module Stop

The WDTA and IWDTa do not support a module-stop function.

The WDTA and IWDTa behave differently when the RX651 is in the all-module stop state. Table 2.47 lists the states of these modules when the microcontroller is in the all-module stop state.

Table 2.47 Module States in All-Module Stop State on RX651

Module Name	Module State
Watchdog timer (WDTA)	Count stopped (state retained)
Independent watchdog Timer (IWDTa)	Selectable in option setting memory

2.9.7 Option Settings

On the RX651 it is possible to specify the microprocessor’s state after a reset by setting the start mode select bits (OFS0.IWDTSTRT and OFS0.WDTSTRT).

2.10 Serial Communication Interface (SCI)/Serial Communication Interface with FIFO (SCIF)

2.10.1 Comparison of Specifications

The SH7080 Group incorporates the SCI and SCIF, which provide serial communication interface functionality, and the RX651 incorporates the SCIg, SCIi, and SCIh.

The SCIg provides, in addition to conventional asynchronous and clock-synchronous transfer capabilities, extended asynchronous functionality that supports a smartcard (IC card) interface. In addition, it supports simple I^2C bus interface single-master operation and simple SPI bus interface operation. The SCIh adds to the functions of the SCIg support for an extended serial interface. For details of the transfer methods not supported on the SH7080 Group, refer to RX651 Group User's Manual: Hardware.

Table 2.48 and Table 2.49 provide a comparative listing of the specifications of the SH7080 Group and RX651.

Table 2.48 Comparison of SH7080 Group and RX651 Specifications (SCI)

Item		SH7080 Group (SCI)	RX651 (SCIg, SCIh)
Number of channels		3 channels (SCI0 to SCI2)	SCIg: 10 channels (SCI0 to SCI9) SCIh: 1 channel (SCI12)
Clock source		Peripheral clock (Pϕ)	Peripheral module clock (PCLKB)
Serial communication modes		• Asynchronous • Clock-synchronous	• Asynchronous • Clock-synchronous • Smartcard interface • Simple I^2C bus • Simple SPI bus
Transfer speed		Any bit rate may be selected using the on-chip baud rate generator.	
Full-duplex communication		Double-buffer configurations for transmission and reception to enable continuous transmission and continuous reception	
Data transfer		Selectable between LSB-first and MSB-first (except for asynchronous 7-bit data)	Selectable between LSB-first and MSB-first (MSB-first only on simple I^2C bus)
DTC/DMAC activation		DTC/DMAC activation supported	DTC/DMAC activation supported
Interrupt sources		• Transmit data empty • Transmit end • Receive data full • Receive error	• Transmit data empty • Transmit end • Receive data full • Receive error Used in simple I^2C mode. • Start condition • Restart condition • Stop condition generation-end
Asynchronous mode	Data length	7 bits, 8 bits	7 bits, 8 bits, 9 bits
	Stop bits	1 bit, 2 bits	
	Parity function	Even parity, odd parity, or no parity	
	Receive error detection	Parity error, overrun error, or framing error	
	Modem control	No	Support for hardware flow control
	Break detection	Detection of when a framing error occurs is possible by directly reading the level of the RXDn pin.	
	Clock source	Selectable between internal and external clock	Selectable between internal and external clock Ability to input transfer rate clock from TMR (SCI5, SCI6, and SCI12)
	Multi-processor communication	Yes	
	Noise cancellation	No	On-chip digital noise filter for input on RXDn pins
	Other	—	• Double-speed mode • Start bit detection
Clock-synchronous mode	Data length	8 bits	
	Receive error detection	Overrun error	
	Modem control	No	Support for hardware flow control
Other		—	• Event link • Expanded serial mode (SCIh only) • Bit rate modulation

Table 2.49 Comparison of SH7080 Group and RX651 Specifications (SCIF)

Item		SH7080 Group (SCIF)	RX651 (SCIi)
Number of channels		1 channel (SCI3)	2 channels (SCI10 to SCI11)
Clock source		Peripheral clock (Pϕ)	Peripheral module clock (PCLKA)
Serial communication modes		• Asynchronous • Clock-synchronous	• Asynchronous • Clock-synchronous • Smartcard interface • Simple I^{2}C bus • Simple SPI bus
Transfer speed		Any bit rate may be selected using the on-chip baud rate generator.	
Full-duplex communication		16-stage FIFO buffer configurations for transmission and reception to enable continuous transmission and reception	
Data transfer		LSB-first	Selectable between LSB-first and MSB-first (MSB-first only on simple I^{2}C bus)
DTC/DMAC control		DTC control supported	DTC/DMAC control supported
Interrupt sources		• Transmit FIFO data empty • Break • Receive FIFO data full • Receive error	• Transmit data empty • Transmit end • Receive data full • Receive error • Receive data ready • Data match Used in simple I^{2}C mode. • Start condition • Restart condition • Stop condition generation-end
Asynchronous mode	Data length	7 bits, 8 bits	7 bits, 8 bits, 9 bits
	Stop bits	1 bit, 2 bits	
	Parity function	Even parity, odd parity, or no parity	
	Receive error detection	Parity error, overrun error, or framing error	
	Modem control	Yes	Support for hardware flow control
	Break detection	Break detection is possible. Also, detection of when a framing error occurs is possible by directly reading the level of the RXDn pin.	Detection of when a framing error occurs is possible by directly reading the level of the RXDn pin.
	Clock source	Selectable between internal and external clock	
	Multi-processor communication	Yes	
	Noise cancellation	No	On-chip digital noise filter for input on RXDn pins
	Other	—	• Double-speed mode • Start bit detection
Clock-synchronous mode	Data length	8 bits	
	Receive error detection	Overrun error	
	Modem control	No	Support for hardware flow control
Other		—	Bit rate modulation

2.10.2 Register Comparison

Table 2.50 and Table 2.51 are a comparative listing of the registers on the SH7080 Group and RX651.

Guide to Symbols in "Changes" Column of Table

◎: Register with same bit assignments on SH7080 Group and RX651

△: Register with different bit assignments on SH7080 Group and RX651

—: Register not present on SH7080 Group or RX651

Table 2.50 SH7080 Group and RX651 Register Comparison (SCI)

SH7080 Group (SCI)	RX651 (SCIg, SCIh)	Changes
SCI n: 0 to 2	SCI m: 0 to 9, 12	
Transmit data register n (SCTDR_n)	Transmit data register (SCIm.TDR)	◎
Transmit shift register (SCTSR)	Transmit shift register (TSR)	◎
Receive data register n (SCRDR_n)	Receive data register (SCIm.RDR)	◎
Receive shift register (SCRSR)	Receive shift register (RSR)	◎
Serial mode register n (SCSMR_n)	Serial mode register (SICm.SMR)	◎
Serial control register n (SCSCR_n)	Serial control register (SCIm.SCR)	◎
Serial status register n (SCSSR_n)	Serial status register (SCIm.SSR)	◎
Bit rate register n (SCBRR_n)	Bit rate register (SCIm.BRR)	◎
Serial direction control register n (SCSDCR_n)	Smartcard mode register (SCIm.SCMR)	△
Serial port register n (SCSPTR_n)	—	—
—	Modulation duty register (SCIm.MDDR)	—
	Serial extended mode register (SCIm.SEMR)	—
	Noise filter setting register (SCIm.SNFR)	—
	I^{2}C mode registers 1 to 3 (SCIm.SIMR1 to SCIm.SIMR3)	—
	I^{2}C status register (SCIm.SISR)	—
	SPI mode register (SCIm.SPMR)	—
	Extended serial mode enable register (SCI12.ESMER)	—
	Control registers 0 to 3 (SCI12.CR0 to SCI12.CR3)	—
	Port control register (SCI12.PCR)	—
	Interrupt control register (SCI12.ICR)	—
	Status register (SCI12.STR)	—
	Status clear register (SCI12.STCR)	—
	Control field 0 data register (SCI12.CF0DR)	—
	Control field 0 compare enable register (SCI12.CF0CR)	—
	Control field 0 receive data register (SCI12.CF0RR)	—
	Primary control field 1 data register (SCI12.PCF1DR)	—
	Secondary control field 1 data register (SCI12.SCF1DR)	—
	Control field 1 compare enable register (SCI12.CF1CR)	—
	Control field 1 receive data register (SCI12.CF1RR)	—
	Timer control register (SCI12.TCR)	—
	Timer mode register (SCI12.TMR)	—
	Timer prescaler register (SCI12.TPRE)	—
	Timer count register (SCI12.TCNT)	—

Table 2.51 SH7080 Group and RX651 Register Comparison (SCIF)

SH7080 Group (SCIF)	RX651 (SCIi)	Changes
SCI n: 3	SCI m: 10 or 11	
—	Transmit data register (SCIm.TDR)	—
Transmit FIFO data register n (SCFTDR_n)	Transmit FIFO data register (SCIm.FTDR)	△
Transmit shift register (SCTSR)	Transmit shift register (TSR)	◎
—	Receive data register (SCIm.RDR)	—
Receive FIFO data register n (SCFRDR_n)	Receive FIFO data register (SCIm.FRDR)	△
Receive shift register (SCRSR)	Receive shift register (RSR)	◎
Serial mode register n (SCSMR_n)	Serial mode register (SCIm.SMR)	△
Serial control register n (SCSCR_n)	Serial control register (SCIm.SCR)	△
Serial status register n (SCFSR_n)	Serial status register (SCIm.SSR/ SCIm.SSRFIFO)	△
Bit rate register n (SCBRR_n)	Bit rate register (SCIm.BRR)	◎
Serial port register n (SCSPTR_n)	Serial port register (SCIm.SPTR)	△
FIFO control register n (SCFCR_n)	FIFO control register (SCIm.FCR)	△
FIFO data count register n (SCFDR_n)	FIFO data count register (SCIm.FDR)	◎
Line status register n (SCLSR_n)	Line status register (SCIm.LSR)	△
—	Smartcard mode register (SCIm.SCMR)	—
	Modulation duty register (SCIm.MDDR)	
	Serial extended mode register (SCIm.SEMR)	
	Noise filter setting register (SCIm.SNFR)	
	I^{2}C mode registers 1 to 3 (SCIm.SIMR1 to SCIm.SIMR3)	
	I^{2}C status register (SCIm.SISR)	
	SPI mode register (SCIm.SPMR)	
	Comparison data register (SCIm.CDR)	
	Data comparison control register (SCIm.DCCR)	

2.10.3 Clock Source Selection

For asynchronous mode communication on the RX651 a setting in the serial extended mode register (SEMR) is used to select external clock input or TMR clock input (SCI5, SCI6, or SIC12 only) as the clock source that determines one bit period. In addition, an 8-bit or 16-bit clock can be selected as the base clock for one bit period.

2.10.4 Interrupts

The serial communication interface with FIFO on the SH7080 Group can activate the DTC, and on the RX651 the serial communication interface with FIFO can activate both the DTC and the DMAC. When FIFO functionality is not used, the interrupt source specifications for the SH7080 Group and RX651 are identical.

On the RX651 when a receive data full or transmit data empty interrupt occurs while the corresponding interrupt status flag (IRn.IR) is set to 1, the interrupt request is also stored internally by the module, and after the interrupt status flag (IRn.IR) is cleared to 0 it is reset to 1 by the stored request.

On the RX651 some interrupts are assigned to multiple group interrupts. The interrupt controller's interrupt status flag (IRn.IR) is cleared automatically when the corresponding interrupt is accepted. Each group interrupt status flag (ISn) is cleared automatically when the corresponding bit in the module's status register is cleared.

Table 2.52 and Table 2.53 list interrupt sources for the SH7080 Group and RX651.

Refer to 1.8 for information about interrupts.

Table 2.52 SH7080 Group SCIF Interrupt Sources

Interrupt Source	DTC Activation	DMAC Activation	Priority
Receive error	Not possible	Not possible	High
Receive FIFO data full or data ready	Possible		↑
Break or overrun error	Not possible		
Transmit FIFO data empty	Possible		Low

Table 2.53 RX651 SCIi Interrupt Sources

Interrupt Source	DTC Activation	DMAC Activation	Priority
Receive error	Not possible	Not possible	High
Receive FIFO full	Possible	Possible	↑
Receive data ready*[1]			
Data match			
Transmit FIFO empty			
Transmit end	Not possible	Not possible	Low

Note 1. Only when FIFO is enabled

2.10.5 Module Stop

As on the SH7080 Group, the SCI of the RX651 is set to the module-stop state after a reset and no clock is supplied.

Refer to 2.16 for information on the module-stop state.

2.10.6 Asynchronous Communication Setting Example (Interrupt/Polling)

A setting example for asynchronous serial communication using the serial communication interface (SCI) of the SH7080 Group and RX651 is presented below.

< Specifications >

1. SCI2 on the RSK+RX65N board is used to make a loopback connection to TXD2 and RXD2.
2. A total of 32 bytes of data are transmitted from the transmit buffer, and then the same data is received.
3. For the interrupt method, transmit and receive interrupts are used; transmission starts when the transmit data-empty interrupt occurs, and reception starts when the receive data register-full interrupt occurs.
4. For the polling method, no interrupts are used; the timing of data transmission and reception are based on polling of the serial status register.
5. After the microcontroller is initialized, LED0 turns on when the SCI2 is ready for transmit and receive operation. LED1 turns on when transmission and reception end. LED2 turns on if a receive error occurs.

Notes

On the RX651 when serial transmit operation is prohibited for a pin set as TXD2, the output of that pin becomes high-impedance, so a pull-up resistor should be connected.

On the RSK+RX65N the TXD2 line is connected to a pull-down resistor and the RXD2 line to U8 (SN74LVC2T45DCT). Therefore, the following modifications must be made to the RSK+RX65N in order to connect a pull-up resistor to TXD2.

(Modifications)

- Remove R116 (0 Ω resistor) and R153 (0 Ω resistor) from the board.
- Connect a pull-up resistor to the TXD2 line.

Note that the sample code enables serial transmit operation and then changes the pin function assignment to TXD2, so processing is required to prevent the pin output from changing to high-impedance.

Table 2.54 SCI Asynchronous Communication Specifications

Item	Description	Remarks
Communication mode	Asynchronous serial communication	
Transfer speed	38,400 bps	
Data length	8 bits	
Stop bits	1 bit	
Parity	None	
Hardware flow control	None	
Bit order	LSB-first	
SCI channel used	SCI2	
Pins used	P50/TXD2	
	P52/RXD2	
	P03/GPIO	LED0 output
	P05/GPIO	LED1 output
	P73/GPIO	LED2 output

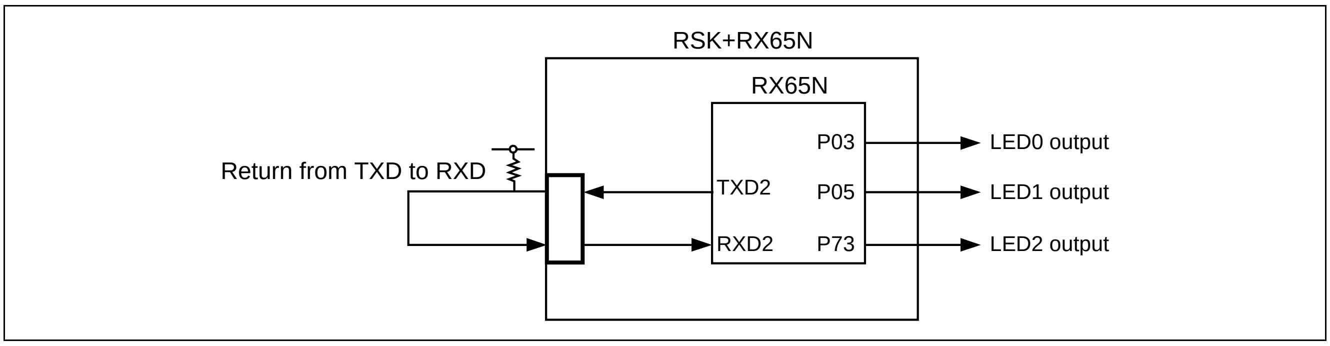


Figure 2.22 SCI Connection Specifications

< List of related registers >

The SCI2 interrupt-related registers and the interrupt sources on the SH7080 Group and RX651 are listed below. In order to reproduce the receive, transmit, transmit-end, and receive error interrupts of the SH7080 Group on the RX651, it is necessary to be aware of the resource settings for each and flags listed in Table 2.55.

Table 2.55 SCI Interrupt-Related Resources (Asynchronous Communication)

Item	SH7080 Group				RX651			
Interrupt sources	RXI	TXI	TEI	ERI	RXI	TXI	TEI	ERI
Interrupt priority registers*[1]	IPRL(7-4)				IPR062	IPR063	IPR110	
Interrupt enable registers*[1]	—	—	—	—	IER07 .IEN6	IER07 .IEN7	IER0D.IEN6	
							GENBL0 .EN4	GENBL0 .EN5
Interrupt request registers*[1]	—	—	—	—	IR062	IR063	IR110	
							GRPBL0 .IS4	GRPBL0 .IS5
Interrupt enable bits*[1]	SCSCR .RIE	SCSCR .TIE	SCSCR .TEIE	SCSCR .RIE	SCR .RIE	SCR .TIE	SCR .TEIE	SCR .RIE
Status registers*[2]	SCSSR .RDRF	SCSSR .TDRE	SCSSR .TEND	SCSSR .ORER SCSSR .FER SCSSR .PER	SSR .RDRF	SSR .TDRE	SSR .TEND	SSR .ORER SSR .FER SSR .PER

Note 1. Used for interrupt handling. Not used when the polling method is employed.
Note 2. When the polling method is employed, source detection is implemented by polling these registers.

The register symbols and full names are as follows:

- SH7080 Group
  IPRL: Interrupt priority level setting register L
  SCSCR: Serial control register
  SCSSR: Serial status register

- RX651
  IPRr: Interrupt source priority register (r: vector number)
  IER07 and IER0D: Interrupt request enable registers 07 and 0D
  IRn: Interrupt request register (n: vector number)
  GENBL0: Group BL0 interrupt request enable register
  GRPBL0: Group BL0 interrupt request register
  SCR: Serial control register
  SSR: Serial status register

The initial setting procedure for asynchronous communication using the SCI is shown below.

Note that due to the I/O port setting the TXD2 output is high-level during the period when the value of the transmit enable bit (TE bit) is 0.

Table 2.56 SCI Asynchronous Communication Initial Setting Example (Common to Interrupt Method and Polling Method)

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKB (Peripheral Clock B): 60 MHz

1	Cancel module-stop state.	STB.STBCR3.MSTP13 = 0	SYSTEM.PRCR = A502h
			SYSTEM.MSTPCRB.MSTPB29 = 0
			SYSTEM.PRCR = A500h

2	Disable SCI interrupts.	On the SH7080 Group the interrupt controller does not have enable registers.	ICU.IER07.IEN6 = 0 (RXI2)
			ICU.IER07.IEN7 = 0 (TXI2)
			ICU.IER0D.IEN6 = 0
			(TEI2, ERI2: group interrupt)
			ICU.GENBL0.EN4 = 0 (TEI2: SCI2)
			ICU.GENBL0.EN5 = 0 (ERI2: SCI2)

3	Make I/O port settings.	PEDRL.PE10DR = 1 (set to output 1)
		PEIORL.PE10IOR = 1 (set to output)
		PEIORL.PE7IOR = 0 (set to input)
		PECRL3.PE10MD = 000b
		(PE10 set to general I/O)
		PECRL2.PE7MD = 000b
		(PE7 set to general I/O)
			PORT5.PODR.B0 = 1 (set to output 1)
			PORT5.PDR.B0 = 1 (set to output)
			PORT5.PDR.B2 = 0 (set to input)
			PORT5.PMR.B0 = 0
			(P50 set to general I/O)
			PORT5.PMR.B2 = 0
			(P52 set to general I/O)

4	Initialize SCR.	SCSCR.TIE, RIE, TE, RE, TEIE = 0	SCR.TIE, RIE, TE, RE, TEIE = 0
5	Enable clock.	Internal clock/SCK pin: input (input signal ignored) SCSCR.CKE[1:0] = 00b	On-chip baud rate generator SCKn pin: I/O port SCR.CKE[1:0] = 00b
6	Initialize SIMR and SPMR.	—	SIMR1.IICM = 0 SPMR1.CKPH, CKPOL = 0 (may be omitted if same as initial value)

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKB (Peripheral Clock B): 60 MHz

7	Make transmit and receive format settings.	SCSMR.C/_A = 0 (asynchronous)
		SCSMR.CHR = 0 (8 bits)
		SCSMR.PE = 0 (no parity)
		SCSMR.STOP = 0 (1 stop bit)
		SCSMR.MP = 0
		(multi-processor disabled)
		SCSMR.CKS[1:0] = 00b (Pϕ)
			SMR.CM = 0 (asynchronous)
			SMR.CHR = 0 (8 bits)
			SMR.PE = 0 (no parity)
			SMR.STOP = 0 (1 stop bit)
			SMR.MP = 0 (multi-processor disabled)
			SMR.CKS[1:0] = 00b (PCLKB)
			SCMR.SMIF = 0
			(serial communication interface mode)
			SCMR.SINV = 0
			(no inversion of transmit and receive data)
			SCMR.SDIR = 0 (LSB-first)
			SCMR.CHR1 = 1 (8 bits)
			SEMR.ABCS = 1
			(1 bit period transfer rate = 8 cycles of base clock)
			SEMR.NFEN = 0
			(digital noise filter disabled)
			SEMR.BGDM = 1
			(clock output frequency twice that of baud rate generator)

8	Set bit rate (BRR).	38,400 bps SCBRR = 32	38,400 bps BRR = 194
9	Wait for 1 bit period.	←	←

10	• Enable interrupts on interrupt controller.
	• Set priority.
	• Clear interrupt request.
	Note: Skip in case of polling.
		INTC.IPRL.WORD = 0050h (level 5)	ICU.IPR062 = 5 (level 5)
			ICU.IPR063 = 5 (level 5)
			ICU.IPR110 = 5 (level 5)
			ICU.IR062 = 0 (RXI2)
			ICU.IR063 = 0 (TXI2)
			ICU.GENBL0.EN5 = 1 (ERI2: SCI2)
			ICU.IER07.IEN6 = 1 (RXI2)
			ICU.IER07.IEN7 = 1 (TXI2)
			ICU.IER0D.IEN6 = 1
			(TEI2, ERI2: group interrupt)

11	Enable transmit/receive.	SCSCR.TIE, RIE, TE, RE = 1 Note: SCSCR.RIE = 0 and SCSCR.TIE = 0 in case of polling.	SCR.RIE, RE = 1 SCR.TIE, TE = 1 Note: SCR.RIE = 0 and SCR.TIE = 0 in case of polling.

12	Make I/O port settings.	PECRL3.PE10MD = 010b
		(TXD2 pin setting)
		PECRL2.PE7MD = 010b
		(RXD2 pin setting)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enabled)
			MPC.P50PFS = 0Ah (TXD2 pin setting)
			MPC.P52PFS = 0Ah (RXD2 pin setting)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1
			PORT5.PMR.B0 = 1 (peripheral function)
			PORT5.PMR.B2 = 1 (peripheral function)

Note 1. Shaded portions indicate places where polling settings differ.

SCI transmission and reception during asynchronous communication (interrupt method) is described below.

The details of interrupt handling are not stipulated in the sample code. However, on the RX651 the receive error interrupt is assigned to a group interrupt. It is therefore necessary to detect the interrupt flag from the group.

Table 2.57 Example of Receive Data-Full Interrupt Handling during SCI Asynchronous Communication

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Read receive data.	Read out contents of SCRDR to receive buffer.	Read out contents of RDR to receive buffer.
2	Clear receive data register-full flag.	After reading SCSSR.RDRF, clear to 0.	SSR.RDRF and ICU.IR062 are cleared automatically.
3	Confirm read of all receive data.	End interrupt handler if receive byte count has not reached 32 bytes.	End interrupt handler if receive byte count has not reached 32 bytes.

4	End receive operation.	SCSCR.RIE = 0
		SCSCR.RE = 0
			ICU.IER07.IEN6 = 0 (RXI2)
			ICU.GENBL0.EN5 = 0 (ERI2: group BL0)
			SCR.RIE = 0
			SCR.RE = 0
			ICU.IR062 = 0

Table 2.58 Example of Transmit Data-Empty Interrupt Handling during SCI Asynchronous Communication

Procedure	SH7080 Group Setting Example	RX651 Setting Example
Write transmit data to TDR.	Write data to SCTDR.	Write data to TDR.
Clear the transmit data-empty flag to 0.	After reading SCSSR.TDRE, clear to 0.	SSR.TDRE and ICU.IR063 are cleared automatically.
Confirm write of all transmit data.	End interrupt handler if transmit byte count has not reached 32 bytes.	End interrupt handler if transmit byte count has not reached 32 bytes.

Make transmit end interrupt settings.	SCSCR.TIE = 0
	
	<Transmit end interrupt settings>
	SCSCR.TEIE = 1
		ICU.IER07.IEN7 = 0 (TXI2)
		SCR.TIE = 0
		ICU.IR063 = 0
		
		<Transmit end interrupt settings>
		SCR.TEIE = 1
		ICU.GENBL0.EN4 = 1 (TEI2: SCI2)

Table 2.59 Example of Error Interrupt Handling during SCI Asynchronous Communication

Procedure	SH7080 Group Setting Example	(RX651 Setting Example
Determine group interrupt.	—	If ICU.GRPBL0.IS5 (SCI2 receive error) is 1, continue processing from step 2.
Determine overrun error.	Perform error processing if SCSSR.ORER is set to 1.	Perform error processing if SSR.ORER is set to 1.
Determine framing error.	Perform error processing if SCSSR.FER is set to 1.	Perform error processing if SSR.FER is set to 1.
Determine parity error.	Perform error processing if SCSSR.PER is set to 1.	Perform error processing if SSR.PER is set to 1.

Table 2.60 Example Transmit End Interrupt Handler for SCI Asynchronous Communication

Procedure	SH7080 Group Setting Example	RX651 Setting Example

Disable transmit end interrupts.	SCSCR.TEIE = 0	ICU.GENBL0.EN4 = 0 (TEI2: SCI2)
		SCR.TEIE = 0

Make I/O port settings.	PECRL3.PE10MD = 000b (PE10 set to general I/O)	PORT5.PMR.B0 = 0 (P50 set to general I/O)
End transmit operation.	SCSCR.TE = 0	SCR.TE = 0

Transmit and receive operation processing during SCI asynchronous communication (polling method) is described below.

No interrupts are used during polling operation. The processing shown below is an extension of that shown in Table 2.56, SCI Asynchronous Communication Initial Setting Example (Common to Interrupt Method and Polling Method).

Table 2.61 Example of Transmit and Receive Processing during SCI Asynchronous Communication (Polling Method)

Procedure		SH7080 Group Setting Example	RX651 Setting Example
Receive processing			
1	Read receive error and determine error type.	If ORER, FER, or PER in SCSSR ≠ 0, go to receive error handling. ⇒ If not receive error, go to step 2.	If ORER, FER, or PER in SSR ≠ 0, go to receive error handling. ⇒ If not receive error, go to step 2.
2	Monitor receive data register-full flag.	If SCSSR.RDRF = 1, perform receive processing. ⇒ Go to step 3. If SCSSR.RDRF = 0, go to transmit processing.	If SSR.RDRF = 1, perform receive processing. ⇒ Go to step 3. If SSR.RDRF = 0, go to transmit processing.
3	Read receive data.	Read SCRDR and store the data in the receive buffer.	Read RDR and store the data in the receive buffer.
4	Clear receive data register-full flag.	Clear SCSSR.RDRF to 0.	SSR.RDRF is cleared automatically.
5	Confirm read of all receive data.	Go to transmit processing if receive byte count has not reached 32 bytes.	Go to transmit processing if receive byte count has not reached 32 bytes.
6	End receive operation.	SCSCR.RE = 0	SCR.RE = 0
Transmit processing			
7	Monitor transmit data-empty flag.	If SCSSR.TDRE = 1, perform transmit processing. ⇒ Go to step 8. SCSSR.TDRE = 0, go to receive processing.	If SSR.TDRE = 1, perform transmit processing. ⇒ Go to step 8. SSR.TDRE = 0, go to receive processing.
8	Write transmit data to TDR.	Write transmit data to SCTDR.	Write transmit data to TDR.
9	Clear transmit data-empty flag.	Clear SCSSR.TDRE to 0.	SSR.TDRE is cleared automatically.
10	Confirm write of all transmit data.	Go to step 14 if receive byte count has not reached 32 bytes.	Go to step 14 if receive byte count has not reached 32 bytes.
11	Wait for transmit completion.	Wait until SCSSR.TEND = 1.	Wait until SSR.TEND = 1.
12	Make I/O port settings.	PECRL3.PE10MD = 000b (PE10 set to general I/O)	PORT5.PMR.B0 = 0 (P50 set to general I/O)
13	End transmit operation.	SCSCR.TE = 0	SCR.TE = 0
14	If transmit and receive are both finished, end processing. Otherwise, go to step 1.		
Error handling			
15	Receive error handling	The details of error handling are not stipulated.	The details of error handling are not stipulated.

2.10.7 Clock-Synchronous Master Transmit Setting Example (Interrupt/Polling)

A setting example for clock-synchronous master transmit processing using the serial communication interface (SCI) of the SH7080 Group and RX651 is described below.

< Specifications >

1. SCI2 on the RSK+RX65N board is used.
2. For the interrupt method, the transmit data-empty interrupt is used to start transmission.
3. No interrupts are used with the polling method. Data transfer starts after setting of the transmit data empty flag in the serial status register is confirmed.
4. Master transmit processing ends after 32 bytes of data have been transmitted.
5. LED0 turns on when transmission starts, and LED1 turns on when transmission ends.

Notes

On the RX651 when serial transmit operation is prohibited for a pin set as TXD2, the output of that pin becomes high-impedance.

On the RSK+RX65N the TXD2 line is connected to a pull-down resistor. Therefore, the following modifications must be made to the RSK+RX65N in order to connect a pull-up resistor to TXD2.

(Modifications)

- Remove R116 (0 Ω resistor) from the board.
- Connect a pull-up resistor to the TXD2 line.

Note that the sample code enables serial transmit operation and then changes the pin function assignment to TXD2, so processing is required to prevent the pin output from changing to high-impedance.

Table 2.62 SCI Clock-Synchronous Communication Specifications (Master Transmit)

Item	Description	Remarks
Communication mode	Clock-synchronous serial communication	
Transfer speed	100 kbps	
Data length	8 bits	
Hardware flow control	None	
SCI channel used	SCI2	
Bit order	LSB-first	
Synchronous clock	Internal clock	
Pins used	P50/TXD2	
	P51/SCK2	
	P03/GPIO	LED0 output
	P05/GPIO	LED1 output
	P73/GPIO	LED2 output

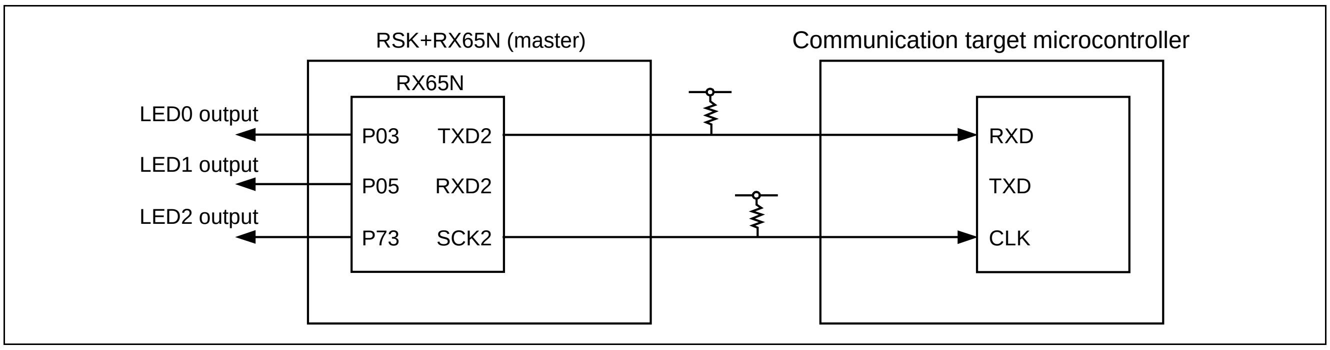


Figure 2.23 Clock-Synchronous Serial Communication Connection Specifications (Master Transmit)

< List of related registers >

On the SH7080 Group and RX651 the settings of the interrupt-related registers of the SCI2 are the same as those for asynchronous operation, as listed in Table 2.55. Unlike asynchronous communication, overrun error is the only error interrupt source.

The initial setting procedure for SCI clock-synchronous master transmit operation is shown below. Note that the initial setting processing is common to the interrupt method and the polling method.

Note that due to the I/O port setting, output on the TxD2 and SCK2 pins is high-level during the period when the value of the transmit enable bit (TE bit) is 0.

Table 2.63 SCI Clock-Synchronous Master Transmit Initial Setting Example

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKB (Peripheral Clock B): 60 MHz

1	Cancel module-stop state.	STB.STBCR3.MSTP13 = 0	SYSTEM.PRCR = A502h
			SYSTEM.MSTPCRB.MSTPB29 = 0
			SYSTEM.PRCR = A500h

2	Disable SCI interrupts.	On the SH7080 Group the interrupt controller does not have enable registers.	ICU.IER07.IEN6 = 0 (RXI2)
			ICU.IER07.IEN7 = 0 (TXI2)
			ICU.IER0D.IEN6 = 0
			(TEI2, ERI2: group interrupt)
			ICU.GENBL0.EN4 = 0 (TEI2: SCI2)
			ICU.GENBL0.EN5 = 0 (ERI2: SCI2)

3	Make I/O port settings.	PEDRL.PE10DR = 1 (set to output 1)
		PEDRL.PE8DR = 1 (set to output 1)
		PEIORL.PE10IOR = 1 (set to output)
		PEIORL.PE8IOR = 1 (set to output)
		PECRL3.PE10MD = 000b
		(PE10 set to general I/O)
		PECRL3.PE8MD = 000b
		(PE8 set to general I/O)
			PORT5.PODR.B0 = 1 (set to output 1)
			PORT5.PODR.B1 = 1 (set to output 1)
			PORT5.PDR.B0 = 1 (set to output)
			PORT5.PDR.B1 = 1 (set to output)
			PORT5.PMR.B0 = 0
			(P50 set to general I/O)
			PORT5.PMR.B1 = 0
			(P51 set to general I/O)

4	Initialize SCR.	SCSCR.TIE, RIE, TE, RE, TEIE = 0	SCR.TIE, RIE, TE, RE, TEIE = 0
5	Enable clock.	Internal clock/SCK pin: output SCSCR.CKE[1:0] = 00b	Internal clock SCKn pin: output port SCR.CKE[1:0] = 00b
6	Initialize SIMR and SPMR.	—	SIMR1.IICM = 0 SPMR.CKPH, CKPOL = 0 (may be omitted if no change from initial value)

7	Make transmit and receive format settings.	SCSMR.C/_A = 1 (clock-synchronous)
		SCSMR.CKS[1:0] = 00b (Pϕ)
			SMR.CM = 1 (clock-synchronous)
			SMR.CKS[1:0] = 00b (PCLKB)
			SCMR.SMIF = 0
			(serial communication interface mode)
			SCMR.SINV = 0
			(no inversion of transmit and receive data)
			SCMR.SDIR = 0 (LSB-first)

8	Set bit rate (BRR).	100 kbps SCBRR = 99	100 kbps BRR = 149
9	Wait for 1 bit period.	←	—

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKB (Peripheral Clock B): 60 MHz

10	• Enable interrupts on interrupt controller.
	• Set priority.
	• Clear the interrupt source.
	Note: Skip in case of polling.
		INTC.IPRL.WORD = 0050h (level 5)	ICU.IPR063 = 05h (level 5)
			ICU.IPR110 = 05h (level 5)
			ICU.IR063 = 0
			ICU.IER07.IEN7 = 1 (TXI2)

11	Enable transmission.	SCSCR.TIE = 1
		SCSCR.TE = 1
		Note: SCR.TIE = 0 in case of polling.
			SCR.TIE = 1
			SCR.TE = 1
			Note: SCR.TIE = 0 in case of polling.

12	Make I/O port settings.	PECRL3.PE10MD = 010b (TXD2 pin setting)
		PECRL3.PE8MD = 010b (SCK2 pin setting)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1 (PFS write enabled)
			MPC.P50PFS = 0Ah (TXD2 pin setting)
			MPC.P51PFS = 0Ah (SCK2 pin setting)
			MPC.PWPR.PFSWE = 0 (PFS write disabled)
			MPC.PWPR.B0WI = 1
			PORT5.PMR.B0 = 1 (peripheral function)
			PORT5.PMR.B1 = 1 (peripheral function)

Table 2.64 Example of Transmit Data Empty Interrupt Handling during SCI Clock-Synchronous Master Transmission (Interrupt Handling Method)

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Write transmit data to TDR.	Write data to SCTDR.	Write data to TDR.
2	Clear the transmit data-empty flag to 0.	After reading SCSSR.TDRE, clear to 0.	SSR.TDRE and ICU.IR063 are cleared automatically.
3	Write all transmit data.	End interrupt handler if transmit byte count has not reached 32 bytes.	End interrupt handler if transmit byte count has not reached 32 bytes.

4	Make transmit end interrupt settings.	SCSCR.TIE = 0
		
		<TEI interrupt settings>
		SCSCR.TEIE = 1
			ICU.IER07.IEN7 = 0 (TXI2)
			SCR.TIE = 0
			ICU.IR063 = 0
			
			<TEI interrupt settings>
			SCR.TEIE = 1
			ICU.GENBL0.EN4 = 1 (TEI2)
			ICU.IER0D.IEN6 = 1 (TEI2: group interrupt)

Table 2.65 Example of Transmit End Interrupt Handling during SCI Clock-Synchronous Master Transmission (Interrupt Handling Method)

Procedure		SH7080 Group Setting Example	RX651 Setting Example

1	Disable transmit end interrupts.	(No particular details specified)
		SCSCR.TEIE = 0
			(No particular details specified)
			ICU.IER0D.IEN6 = 0 (TEI2)
			ICU.GENBL0.EN4 = 0 (TEI2: group interrupt)
			SCR.TEIE = 0

2	Make I/O port settings.	PECRL3.PE10MD = 000b
		(PE10 set to general I/O)
		PECRL3.PE8MD = 000b
		(PE8 set to general I/O)
			PORT5.PMR.B0 = 0
			(P50 set to general I/O)
			PORT5.PMR.B1 = 0
			(P51 set to general I/O)

3	End transmit operation.	SCSCR.TE = 0	SCR.TE = 0

The processing for SCI clock-synchronous master transmission (polling method) is shown below. The processing shown below is an extension of that shown in Table 2.63, SCI Clock-Synchronous Master Transmit Initial Setting Example.

Table 2.66 Example of SCI Clock-Synchronous Master Transmit Processing (Polling Method)

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Poll transmit data-empty flag. Run transmit processing when transmit-empty occurs.	Polling target: SCSSR.TDRE = 1 Perform transmission processing when SCSSR.TDRE = 1. ⇒ Go to step 2.	Polling target: SSR.TDRE = 1 Perform transmission processing when SSR.TDRE = 1. ⇒ Go to step 2.
2	Write transmit data to TDR.	Write transmit data to SCTDR.	Write transmit data to TDR.
3	Clear transmit data-empty flag.	Clear SCSSR.TDRE.	SSR.TDRE is cleared automatically.
4	Confirm write of all transmit data.	Go to step 1 if transmit byte count has not reached 32 bytes.	Go to step 1 if transmit byte count has not reached 32 bytes.
5	Wait for transmit completion.	(No particular details specified) Wait until SCSSR.TEND = 1.	(No particular details specified) Wait until SSR.TEND = 1.

6	Make I/O port settings.	PECRL3.PE10MD = 000b
		(PE10 set to general I/O)
		PECRL3.PE8MD = 000b
		(PE8 set to general I/O)
			PORT5.PMR.B0 = 0
			(P50 set to general I/O)
			PORT5.PMR.B1 = 0
			(P51 set to general I/O)

7	End transmit operation.	SCSCR.TE = 0	SCR.TE = 0

2.10.8 Clock-Synchronous Slave Receive Setting Example (Interrupt/Polling)

A setting example for clock-synchronous slave receive processing using the serial communication interface (SCI) of the SH7080 Group and RX651 is presented below.

< Specifications >

1. SCI2 on the RSK+RX65N board is used.
2. For the interrupt method, the receive data register-full interrupt is used to start receive processing.
3. No interrupts are used with the polling method. Data reception starts after setting of the receive data full flag in the serial status register is confirmed.
4. Slave receive processing ends after 32 bytes of data have been received.
5. LED0 turns on when reception starts, and LED1 turns on when reception ends. LED2 turns on if a receive error occurs.

Notes

On the RSK+RX65N the RXD2 line is connected to U8 (SN74LVC2T45DCT). Therefore, the following modifications must be made to the RSK+RX65N.

(Modifications)

Remove R153 (0 Ω resistor) from the board.

Table 2.67 SCI Clock-Synchronous Communication Specifications (Slave Receive)

Item	Description	Remarks
Communication mode	Clock-synchronous serial communication	
Data length	8 bits	
Hardware flow control	None	
SCI channel used	SCI2	
Bit order	LSB-first	
Synchronous clock	External clock	
Pins used	P52/RXD2	
	P51/SCK2	
	P03/GPIO	LED0 output
	P05/GPIO	LED1 output
	P73/GPIO	LED2 output

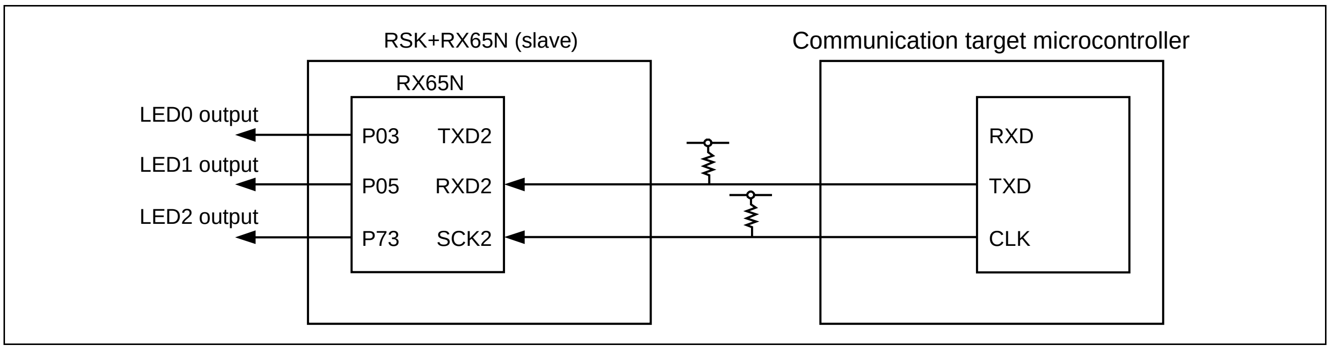


Figure 2.24 Clock-Synchronous Serial Communication Connection Specifications (Slave Receive)

< List of related registers >

On the SH7080 Group and RX651 the settings of the interrupt-related registers of the SCI2 are the same as those for asynchronous operation, as listed in Table 2.55, SCI Interrupt-Related Resources (Asynchronous Communication). The point of difference from asynchronous operation is that overrun error is the only error interrupt source.

The initial setting procedure for SCI clock-synchronous slave receive operation is shown below. Note that the initial setting processing is the same for the interrupt and polling methods. For interrupt resources, refer to Table 2.55.

Table 2.68 SCI Clock-Synchronous Slave Receive Initial Setting Example

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKB (Peripheral Clock B): 60 MHz

1	Cancel module-stop state.	STB.STBCR3.MSTP13 = 0	SYSTEM.PRCR = A502h
			SYSTEM.MSTPCRB.MSTPB29 = 0
			SYSTEM.PRCR = A500h

2	Disable SCI interrupts.	On the SH7080 Group the interrupt controller does not have enable registers.	ICU.IER07.IEN6 = 0 (RXI2)
			ICU.IER07.IEN7 = 0 (TXI2)
			ICU.IER0D.IEN6 = 0
			(TEI2, ERI2: group interrupt)
			ICU.GENBL0.EN4 = 0 (TEI2: SCI2)
			ICU.GENBL0.EN5 = 0 (ERI2: SCI2)

3	Initialize SCR.	SCSCR.TIE, RIE, TE, RE, TEIE = 0	SCR.TIE, RIE, TE, RE, TEIE = 0

4	Make I/O port settings (RX651 only).	PFC setting is performed in step 9.	PORT5.PDR.B1 = 0 (set to input)
			PORT5.PDR.B2 = 0 (set to input)
			PORT5.PMR.B1 = 0
			(P51 set to general I/O)
			PORT5.PMR.B2 = 0
			(P52 set to general I/O)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enabled)
			MPC.P52PFS = 0Ah (RXD pin setting)
			MPC.P51PFS = 0Ah (SCK pin setting)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1
			PORT5.PMR.B1 = 1 (peripheral function)
			PORT5.PMR.B2 = 1 (peripheral function)

5	Enable clock.	External clock/SCK pin clock input SCSCR.CKE[1:0] = 1xb	External clock/SCKn pin as input port SCR.CKE[1:0] = 1xb
6	Initialize SIMR and SPMR.	—	SIMR1.IICM = 0 SPMR.CKPH,CKPOL = 0 (may be omitted if no change from initial value)

7	Make transmit and receive format settings.	SCSMR.C/_A = 1 (clock-synchronous)
		SCSMR.CKS[1:0] = 00b (Pϕ)
			SMR.CM = 1 (clock-synchronous)
			SMR.CKS[1:0] = 00b (PCLKB)
			SCMR.SMIF = 0
			(serial communication interface mode)
			SCMR.SINV = 0
			(no inversion of transmit and receive data)
			SCMR.SDIR = 0 (LSB-first)

8	Wait for 1 bit period.	←	—

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKB (Peripheral Clock B): 60 MHz

9	Make I/O port settings (SH7080 Group only).	PFC setting is performed.
		PEIORL.PE8IOR = 0 (input)
		PEIORL.PE7IOR = 0 (input)
		PECRL3.PE8MD = 010b (SCK2)
		PECRL2.PE7MD = 010b (RXD2)
			Implemented in step 4.

10	• Enable interrupts on interrupt controller.
	• Set priority.
	• Clear the interrupt source.
	Note: Skip in case of polling.
		INTC.IPRL.WORD = 0050h (level 5)	ICU.IPR062 = 05h (level 5)
			ICU.IPR110 = 05h (level 5)
			ICU.IR062 = 0
			ICU.IER07.IEN6 = 1 (RXI2)
			ICU.IER0D.IEN6 = 1
			(TEI2, ERI2: group interrupt)
			ICU.GENBL0.EN5 = 1 (ERI2: SCI2)

11	Enable reception.	SCSCR.RIE = 1
		SCSCR.RE = 1
		Note: SCR.RIE = 0 in case of polling.
			SCR.RIE = 1
			SCR, RE = 1
			Note: SCR.RIE = 0 in case of polling.

Interrupt handling during SCI clock-synchronous slave receive operation (interrupt handling method) is described below.

Table 2.69 Example of Receive Data Full Interrupt Handling during SCI Clock-Synchronous Slave Receive Operation (Interrupt Handling Method)

Procedure	SH7080 Group Setting Example	RX651 Setting Example
Read receive data.	Read out contents of SCRDR to receive buffer.	Read out contents of RDR to receive buffer.
Clear receive data register-full flag.	After reading SCSSR.RDRF, clear to 0.	SSR.RDRF and ICU.IR062 are cleared automatically.
Read all receive data.	Go to step 1 if receive byte count has not reached 32 bytes.	Go to step 1 if receive byte count has not reached 32 bytes.

End receive operation.	(No particular details specified)
	SCSCR.RIE = 0
	SCSCR.RE = 0
		(No particular details specified)
		ICU.IER07.IEN6 = 0 (RXI2)
		ICU.IER0D.IEN6 = 0 (ERI2: group interrupt)
		ICU.GENBL0.EN5 = 0 (ERI2: SCI2)
		ICU.IR062 = 0
		SCR.RIE = 0
		SCSCR.RE = 0

During clock-synchronous communication overrun errors are the only receive errors detected. Implement error handler code to accommodate overrun errors. On the RX651 the receive error interrupt is assigned to a group interrupt. Therefore, it is necessary to detect the interrupt flag from the group.

SCI clock-synchronous slave receive processing (polling method) is described below. In the polling method no interrupts are used. As the procedure, step 11 in Table 2.68, SCI Clock-Synchronous Slave Receive Initial Setting Example, is extended as shown below.

Table 2.70 Example of SCI Clock-Synchronous Slave Receive (Polling Method)

Procedure		SH7080 Group Setting Example	RX651 Setting Example
Receive processing			
1	Read receive error and determine error type.	If ORER in SCSSR ≠ 0, go to receive error handling. ⇒ If not receive error, go to step 2.	If ORER in SSR ≠ 0, go to receive error handling. ⇒ If not receive error, go to step 2.
2	Poll the receive data register-full flag, and if the register is full perform receive processing in step 3 and after.	Polling target: SCSSR.RDRF If SCSSR.RDRF = 1, perform receive processing. ⇒ Go to step 3. If SCSSR.RDRF = 0, go to step 1.	Polling target: SSR.RDRF If SSR.RDRF = 1, perform receive processing. ⇒ Go to step 3. If SSR.RDRF = 0, go to step 1.
3	Read receive data from RDR.	Read SCRDR and store the data in the receive buffer.	Read RDR and store the data in the receive buffer.
4	Clear receive data register-full flag.	Clear SCSSR.RDRF to 0.	SSR.RDRF is cleared to 0 automatically.
5	Confirm read of all receive data.	Go to step 1 if receive byte count has not reached 32 bytes.	Go to step 1 if receive byte count has not reached 32 bytes.
6	End receive operation.	SCSCR.RE = 0	SCR.RE = 0
Receive error handling			
7	Receive error handling	The details of error handling are not stipulated.	The details of error handling are not stipulated.

2.10.9 Asynchronous Communication Setting Example (SCI with FIFO)

The setting example shown below illustrates the replacement of asynchronous communication using the serial communication interface with FIFO (SCIF) on the SH7080 Group with communication in the asynchronous mode of the serial communication interface (SCIi) on the RX651.

< Specifications >

1. SCI10 on the RSK+RX65N board is used, with TXD and RXD connected in a loopback configuration.
2. The 128 bytes of data in the transmit buffer are transmitted, and then the same data is received back again.
3. Transmit and receive interrupts are used. Transmission is started by the transmit data empty interrupt, and reception is started by the receive data register full interrupt.
4. After initialization finishes, LED0 illuminates to show that the SCI is ready for transmit and receive operation. When transmission and reception finish, LED1 illuminates. LED2 illuminates if a receive error occurs.

Notes

On the RX651 when serial transmit operation is prohibited for a pin set as TXD10, the output of that pin becomes high-impedance, so a pull-up resistor should be connected.

On the RSK+RX65N the JA6, TXD10, and RXD10 pins are not connected. Therefore, the following modifications must be made to the RSK+RX65N in order to perform debugging using the sample code.

(Modifications)

- Mount R85 (0 Ω resistor) and R83 (0 Ω resistor) on the board (direct connection to R85 and R83).
- Remove R282 (0 Ω resistor) and R285 (0 Ω resistor) from the board.
- Connect a pull-up resistor to the TXD10 line.

Note that the sample code enables serial transmit operation and then changes the pin function assignment to TXD10, so processing is required to prevent the pin output from changing to high-impedance.

It is possible to prevent the pin output from changing to high-impedance by setting the TXD10 pin state to high in the serial port register (SPTR) before enabling serial transmission.

Table 2.71 Specifications of Asynchronous Communication Using SCI (with FIFO)

Item	Description	Remarks
Communication mode	Asynchronous serial communication	
Transfer speed	38,400 bps	
Data length	8 bits	
Stop bits	1 bit	
Parity	None	
Hardware flow control	None	
Bit order	LSB-first	
SCI channel used	SCI10	
Pins used	P87/TXD10	
	P86/RXD10	
	P03/GPIO	LED0 output
	P05/GPIO	LED1 output
	P73/GPIO	LED2 output

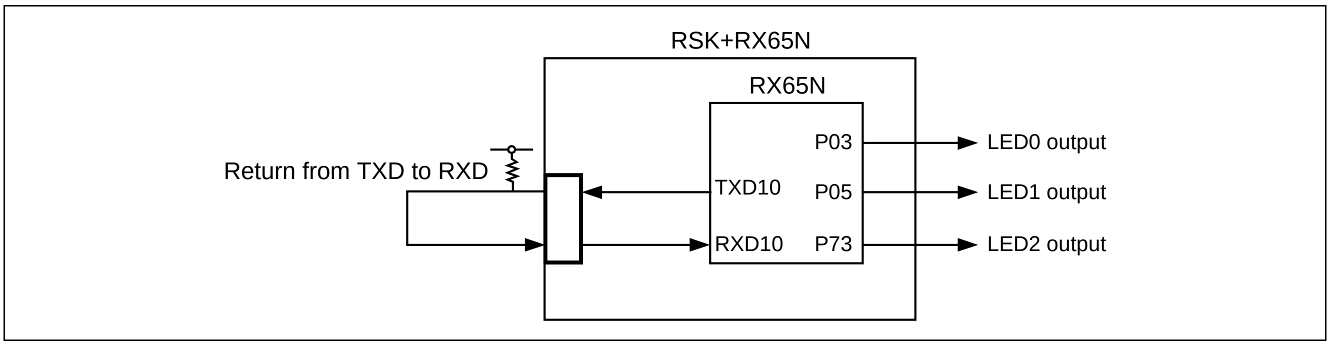


Figure 2.25 SCI (with FIFO) Connection Specifications

< List of related registers >

The interrupt-related registers of the SCI10 on the SH7080 Group and RX651 are listed below by interrupt source. Table 2.72 lists the resource settings and flag checking necessary to implement on the RX651 operation equivalent to the receive data full, transmit data empty, and receive error interrupts on the SH7080 Group.

Note that the RX651 does not support the break interrupt functionality of the SH7080 Group. However, it is possible to implement functionality similar to the break interrupt by reading the RXD line monitor flag (RXDMON) in the serial port register (SPTR) whenever a framing error occurs.

Finally, the SH7080 Group has no transmit end interrupt source, but a transmit end interrupt source is provided on the RX651.

Table 2.72 SCI Interrupt-Related Resources (with FIFO) (Asynchronous Communication)

Item	SH7080 Group				RX651			
Interrupt sources	RXIF	TXIF	ERIF	BRIF	RXI	TXI	ERI	TEI
Interrupt priority registers	IPRL(3-0)				IPR104	IPR105	IPR112	
Interrupt enable registers	—	—	—	—	IER0D. IEN0	IER0D. IEN1	IER0E.IEN0	
							GENAL0 .EN9	GENAL0 .EN8
Interrupt request registers	—	—	—	—	IR104	IR105	IR112	
							GRPAL0 .IS9	GRPAL0 .IS8
Interrupt enable bits	SCSCR .RIE	SCSCR .TIE	SCSCR .RIE SCSCR .REIE	SCSCR .RIE SCSCR .REIE	SCR .RIE	SCR .TIE	SCR .RIE	SCR .TEIE
Status flag	SCFSR .RDF SCFSR .DR	SCFSR .TDFE	SCFSR .ER SCFSR .FER SCFSR .PER	SCFSR .BRK SCLSR .ORER	SSRFIFO .RDF SSRFIFO .DR	SSRFIFO .TDFE	SSRFIFO .ORER SSRFIFO .FER SSRFIFO .PER SSRFIFO .DR	SSRFIFO .TEND

The register symbols and full names are as follows:

- SH7080 Group
  IPRL: Interrupt priority level setting register L
  SCSCR: Serial control register
  SCFSR: Serial status register
  SCLSR: Line status register

- RX651
  IPRr: Interrupt source priority register (r: vector number)
  IER0D and IER0E: Interrupt request enable registers 0D and 0E
  IRn: Interrupt request register (n: vector number)
  GENAL0: Group AL0 interrupt request enable register
  GRPAL0: Group AL0 interrupt request register
  SCR: Serial control register
  SSRFIFO: Serial status register
  DCCR: SDC control register

The initial setting procedure for asynchronous serial communication using the SCI with FIFO is presented below.

Note that due to the I/O port setting the TxD10 pin output is high-level during the period when the value of the transmit enable bit (TE bit) is 0.

Table 2.73 Initial Setting Example for Asynchronous Communication Using SCI (with FIFO)

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock A): 120 MHz

1	Cancel module-stop state.	STB.STBCR3.MSTP14 = 0	SYSTEM.PRCR = A502h
			SYSTEM.MSTPCRC.MSTPC25 = 0
			SYSTEM.PRCR = A500h

2	Disable SCI interrupts.	On the SH7080 Group the interrupt controller does not have enable registers.	ICU.IER0D.IEN0 = 0 (RXI10)
			ICU.IER0D.IEN1 = 0 (TXI10)
			ICU.IER0E.IEN0 = 0
			(TEI10, ERI10: group interrupt)
			ICU.GENAL0.EN8 = 0 (TEI10: SCI10)
			ICU.GENAL0.EN9 = 0 (ERI10: SCI10)

3	Make I/O port settings.	PEDRL.PE12DR = 1 (set to output 1)
		PEIORL.PE12IOR = 1 (set to output)
		PEIORL.PE11IOR = 0 (set to input)
		PECRL4.PE12MD = 000b
		(PE10 set to general I/O)
		PECRL4.PE11MD = 000b
		(PE11 set to general I/O)
			PORT8.PODR.B7 = 1 (set to output 1)
			PORT8.PDR.B7 = 1 (set to output)
			PORT8.PDR.B6 = 0 (set to input)
			PORT8.PMR.B7 = 0 (set to general I/O)
			PORT8.PMR.B6 = 0 (set to general I/O)

4	Initialize control registers.	SCSCR.TIE, RIE, TE, RE = 0	SCR.TIE, RIE, TE, RE, TEIE = 0

5	Reset FIFO.	SCFCR.TFRST = 1
		(transmit FIFO reset operation enabled)
		SCFCR.RFRST = 1
		(receive FIFO reset operation enabled)
			FCR.FM = 1 (FIFO mode)
			FCR.TFRST = 1 (transmit FIFO reset)
			FCR.RFRST = 1 (receive FIFO reset)

6	Make FIFO mode settings (RX651 only).	FIFO mode settings are performed in step 13.	FCR.TTRG[3:0] = 8 (threshold value for setting SSRFIFO.TDFE flag to 1) FCR.RTRG[3:0] = 8 (threshold value for setting SSRFIFO.RDF flag to 1)

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock A): 120 MHz

7	Initialize status registers (SH7080 Group only).	SCFSR.ER = 0 (receive error cleared)
		SCFSR.DR = 0 (receive data ready cleared)
		SCFSR.BRK = 0 (break detection cleared)
		SCLSR.ORER = 0 (overrun error cleared)
		Note: Cleared to 0 after being read.
			—

8	Enable clock.	Internal clock/SCK pin: input SCSCR.CKE[1:0] = 00b	On-chip baud rate generator SCKn pin: I/O port SCR.CKE[1:0] = 00b
9	Initialize SIMR1 and SPMR.	—	SIMR1.IICM = 0 SPMR.CKPH, CKPOL = 0 (may be omitted if no change from initial value)

10	Make transmit and receive format settings.	SCSMR.C/_A = 0 (asynchronous)
		SCSMR.CHR = 0 (8 bits)
		SCSMR.PE = 0 (no parity)
		SCSMR.STOP = 0 (1 stop bit)
		
		SCSMR.CKS[1:0] = 00b (Pϕ)
			SMR.CM = 0 (asynchronous)
			SMR.CHR = 0 (8 bits)
			SMR.PE = 0 (no parity)
			SMR.STOP = 0 (1 stop bit)
			SMR.MP = 0 (multi-processor disabled)
			SMR.CKS[1:0] = 00b (PCLKA)
			SCMR.SMIF = 0
			(serial communication interface mode)
			SCMR.SINV = 0
			(no inversion of transmit and receive data)
			SCMR.SDIR = 0 (LSB-first)
			SCMR.CHR1 = 1 (8 bits)
			SEMR.ABCS = 1
			(1 bit period transfer rate = 8 cycles of base clock)
			SEMR.NFEN = 0
			(digital noise filter disabled)
			SEMR.BGDM = 0
			(clock output frequency = normal frequency of baud rate generator)

11	Set bit rate (BRR).	38,400 bps SCBRR = 32	38,400 bps BRR = 194
12	Wait for 1 bit period.	←	—
13	Make FIFO mode settings (SH7080 Group only).	SCFCR.RTRG[1:0] = 10b (receive FIFO data count trigger: 8) SCFCR.TTRG[1:0] = 00b (transmit FIFO data count trigger: 8) SCFCR.MCE = 0 (CTS and RTS disabled) SCFCR.TFRST = 0 (transmit FIFO reset operation disabled) SCFCR.RFRST = 0 (receive FIFO reset operation disabled)	FIFO mode settings are performed in step 6.

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock A): 120 MHz

14	• Make interrupt priority setting.
	• Clear the interrupt source.
	• Enable interrupts on interrupt controller.
		INTC.IPRL.WORD = 0005h (level 5)	ICU.IPR104 = 05h (level 5)
			ICU.IPR105 = 05h (level 5)
			ICU.IPR112 = 05h (level 5)
			ICU.IR104 = 0 (RXI10)
			ICU.IR105 = 0 (TXI10)
			ICU.IER0D.IEN0 = 1 (RXI10)
			ICU.IER0D.IEN1 = 1 (TXI10)
			ICU.GENAL0.EN9 = 1 (ERI10: SCI10)
			ICU.IER0E.IEN0 = 1
			(TEI10, ERI10: group interrupt)

15	Enable interrupt requests.	SCSCR.TIE, RIE, REIE = 1	SCR.TIE, RIE = 1
16	Enable transmit/receive.	SCSCR.RE, TE = 1	SCR.TE, RE = 1

17	Make I/O port settings.	PECRL4.PE12MD = 011b (TXD3)
		PECRL3.PE11MD = 011b (RXD3)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enabled)
			MPC.P87PFS = 0Ah (TXD10 pin setting)
			MPC.P86PFS = 0Ah (RXD10 pin setting)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1
			PORT8.PMR.B7 = 1 (peripheral function)
			PORT8.PMR.B6 = 1 (peripheral function)

Table 2.74 Example of Receive Data Full Interrupt Handling during Asynchronous Communication Using SCI (with FIFO)

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Read receive data.	Read receive data count specified by SCFDR.R[4:0] from SCFRDR to receive buffer.	Read receive data count specified by FDR.R[4:0] from FRDR to receive buffer.
2	Clear receive data register-full flag.	After reading SCFSR.RDF, clear to 0.	Read SSRFIFO.RDF, then clear to 0. ICU.IR104 is cleared automatically.
3	Confirm read of all receive data.	End interrupt handler if receive byte count has not reached 128 bytes.	End interrupt handler if receive byte count has not reached 128 bytes.

4	End receive operation.	SCSCR.RIE = 0
		SCSCR.RE = 0
			ICU.IER0D.IEN0 = 0 (RXI10)
			ICU. GENAL0.EN9 = 0 (ERI10)
			SCR.RIE = 0
			SCR.RE = 0
			ICU.IR104 = 0

Table 2.75 Example of Transmit Data Empty Interrupt Handling during Asynchronous Communication Using SCI (with FIFO)

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Write transmit data.	Write data to SCFTDR.	Write data to FTDR.
2	Clear the transmit data-empty flag to 0.	Read SCFSR.TDFE and SCFSR.TEND, then clear to 0.	After reading SSRFIFO.TDFE, clear to 0. ICU.IR105 is cleared automatically.
3	Confirm write of all transmit data.	End interrupt handler if transmit byte count has not reached 128 bytes.	End interrupt handler if transmit byte count has not reached 128 bytes.

4	Determine transmit end.	Wait until SCFSR.TEND = 1.
		SCSCR.TIE = 0
			ICU.IER0D.IEN1 = 0 (TXI10)
			SCR.TIE = 0
			ICU.IR105 = 0
			
			<Transmit end interrupt settings>
			SCR.TEIE = 1
			ICU. GENAL0.EN8 = 1 (TEI10)

5	Make I/O port settings.	PECRL4.PE12MD = 000b	—
6	End transmit operation.	SCSCR.TE = 0	—

The details of break interrupt and error handling are not stipulated in the sample code. However, on the RX651 the receive error interrupt is assigned to a group interrupt. It is therefore necessary to detect the interrupt flag from the group.

Table 2.76 Example of Error Interrupt Handling during Asynchronous Communication Using SCI (with FIFO)

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Determine group interrupt.	The SH7080 Group does not support group interrupts.	If ICU.GRPAL0.IS9 (SCI10 receive error) = 1, continue processing from step 2.
2	Determine overrun error.	Perform error processing if SCLSR.ORER is set to 1.	Perform error processing if SSRFIFO.ORER is set to 1.
3	Determine framing error.	Perform error processing if SCFSR.FER is set to 1.	Perform error processing if SSRFIFO.FER is set to 1.
4	Determine parity error.	Perform error processing if SCFSR.PER is set to 1.	Perform error processing if SSRFIFO.PER is set to 1.

Table 2.77 Example of Transmit End Interrupt Handling during Asynchronous Communication Using SCI (with FIFO)

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Disable transmit end interrupts.	— (No transmit end interrupt)	ICU. GENAL0.EN8 = 0 (TEI10) SCR.TEIE = 0
2	Make I/O port settings.	—	PORT8.PMR.B7 = 0
3	End transmit operation.	—	SCR.TE = 0

2.10.10 Setting Example for (SCI with FIFO) Clock-Synchronous Master Transmission

An example is presented below of settings for replacing clock-synchronous communication using the serial communication interface with FIFO (SCIF) on the SH7080 Group with the clock-synchronous mode of the serial communication interface (SCIi) on the RX651.

< Specifications >

1. The SCI10 on the RSK+RX65N board is used.
2. Transmission is activated by the transmit data empty interrupt.
3. Processing ends once 128 bytes of data has been transmitted.
4. LED0 illuminates after initialization finishes and the SCI is ready to transmit, and LED1 illuminates when transmission is finished.

Notes

On the RSK+RX65N the JA6, TXD10, and SCK10 pins are not connected. Therefore, the following modifications must be made to the RSK+RX65N in order to perform debugging using the sample code.

(Modifications)

- Mount R85 (0 Ω resistor) and R80 (0 Ω resistor) on the board (direct connection to R85 and R80).
- Remove R282 (0 Ω resistor) and R82 (0 Ω resistor) from the board.

Note that on the RX651 when serial transmit operation is prohibited for a pin set as TXD10, the output of that pin becomes high-impedance. The sample code enables serial transmit operation and then changes the pin function assignment to TXD10, so processing is required to prevent the pin output from changing to high-impedance.

Table 2.78 Specifications of Clock-Synchronous Master Transmission Using SCI (with FIFO)

Item	Description	Remarks
Communication mode	Clock-synchronous serial communication	
Transfer speed	100 kbps	
Data length	8 bits	
Hardware flow control	None	
Bit order	LSB-first	
SCI channel used	SCI10	
Synchronous clock	Internal clock	
Pins used	P87/TXD10	
	P83/SCK10	
	P03/GPIO	LED0 output
	P05/GPIO	LED1 output
	P73/GPIO	LED2 output

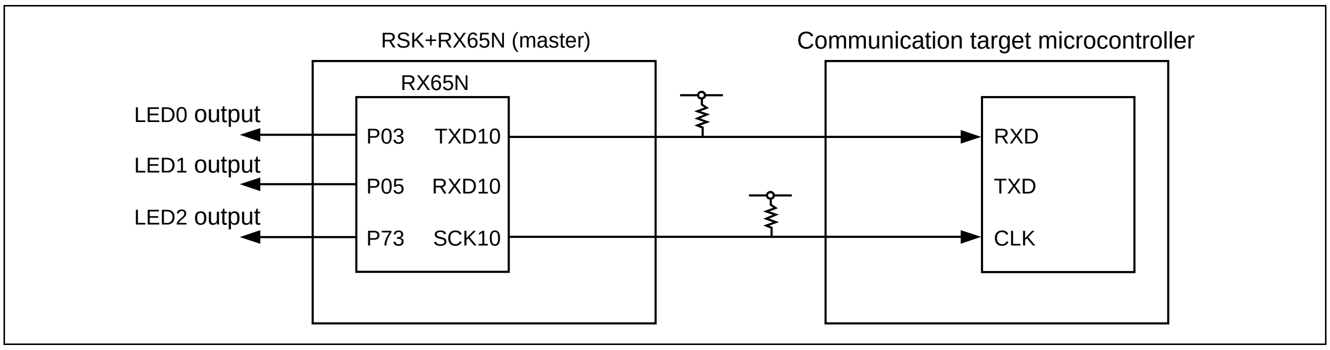


Figure 2.26 Clock-Synchronous Serial Communication Connection Specifications (Master Transmit)

< List of related registers >

On the SH7080 Group and RX651 the settings of the interrupt-related registers of the SCI10 are the same as those for asynchronous operation, as listed in Table 2.72. The points of difference from asynchronous operation are that parity error, framing error, and receive data ready are not available as sources for the receive data full and error interrupts.

The initial settings for clock-synchronous serial communication using the SCI with FIFO are presented below.

Note that due to the I/O port setting the TXD10 and SCK10 pin output is high-level during the period when the value of the transmit enable bit (TE bit) is 0.

Table 2.79 Initial Setting Example for Clock-Synchronous Master Transmission Using SCI (with FIFO)

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock A): 120 MHz

1	Cancel module-stop state.	STB.STBCR3.MSTP14 = 0	SYSTEM.PRCR = A502h
			SYSTEM.MSTPCRC.MSTPC25 = 0
			SYSTEM.PRCR = A500h

2	Disable SCI interrupts.	On the SH7080 Group the interrupt controller does not have enable registers.	ICU.IER0D.IEN0 = 0 (RXI10)
			ICU.IER0D.IEN1 = 0 (TXI10)
			ICU.IER0E.IEN0 = 0
			(TEI10, ERI10: group interrupt)
			ICU.GENAL0.EN8 = 0 (TEI10: SCI10)
			ICU.GENAL0.EN9 = 0 (ERI10: SCI10)

3	Make I/O port settings (RX651 only).	PEDRL.PE12DR = 1 (set to output 1)
		PEDRL.PE9DR = 1 (set to output 1)
		PEIORL.PE12IOR = 1 (set to output)
		PEIORL.PE9IOR = 1 (set to output)
		PECRL4.PE12MD = 000b
		(PE12 set to general I/O)
		PECRL3.PE9MD = 000b
		(PE9 set to general I/O)
			PORT8.PODR.B7 = 1 (set to output 1)
			PORT8.PODR.B3 = 1 (set to output 1)
			PORT8.PDR.B7 = 1 (set to output)
			PORT8.PDR.B3 = 1 (set to output)
			PORT8.PMR.B7 = 0 (set to general I/O)
			PORT8.PMR.B3 = 0 (set to general I/O)

4	Initialize control registers.	SCSCR.TIE, RIE, TE, RE = 0	SCR.TIE, RIE, TE, RE, TEIE = 0

5	Reset FIFO.	SCFCR.TFRST = 1
		(transmit FIFO reset operation enabled)
			FCR.FM = 1 (FIFO mode)
			FCR.TFRST = 1 (transmit FIFO reset)

6	Make FIFO mode settings (RX651 only).	FIFO mode settings are performed in step 13.	FCR.TTRG[3:0] = 8 (threshold value for setting SSRFIFO.TDFE flag to 1)

7	Initialize status registers (SH7080 Group only).	SCFSR.ER = 0 (receive error cleared)
		SCFSR.DR = 0
		(receive data ready cleared)
		SCFSR.BRK = 0 (break detection cleared)
		SCLSR.ORER = 0 (overrun error cleared)
		Note: Cleared to 0 after being read.
			—

8	Enable clock.	Internal clock/SCK pin: output SCSCR.CKE[1:0] = 00b	SCKn pin as clock output pin SCR.CKE[1:0] = 00b
9	Initialize SIMR1 and SPMR.	—	SIMR1.IICM = 0 SPMR.CKPH, CKPOL = 0 (may be omitted if no change from initial value)

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock A): 120 MHz

10	Make transmit and receive format settings.	SCSMR.C/_A = 1 (clock-synchronous)
		SCSMR.CKS[1:0] = 00b (Pϕ)
			SMR.CM = 1 (clock-synchronous)
			SMR.CKS[1:0] = 01b (PCLKA/4)
			SCMR.SMIF = 0
			(serial communication interface mode)
			SCMR.SINV = 0
			(no inversion of transmit and receive data)
			SCMR.SDIR = 0 (LSB-first)

11	Set bit rate (BRR).	100 kbps BRR = 99	100 kbps BRR = 74
12	Wait for 1 bit period.	←	—
13	Make FIFO mode settings (SH7080 Group only).	SCFCR.TTRG[1:0] = 00b (transmit FIFO data count trigger: 8) SCFCR.MCE = 0 (CTS and RTS disabled) SCFCR.TFRST = 0 (transmit FIFO reset operation disabled)	FIFO mode settings are performed in step 6.

14	• Make interrupt priority setting.
	• Clear the interrupt source.
	• Enable interrupts on interrupt controller.
		INTC.IPRL.WORD = 0005h (level 5)	ICU.IPR105 = 05h (level 5)
			ICU.IPR112 = 05h (level 5)
			ICU.IR105 = 0 (TXI10)
			ICU.IER0D.IEN1 = 1 (TXI10)

15	Enable transmission.	SCSCR.TIE = 1
		SCSCR.TE = 1
			SCR.TIE = 1
			SCR.TE = 1

16	Make I/O port settings.	PECRL4.PE12MD = 011b (TXD3)
		PECRL3.PE9MD = 011b (SCK3)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enabled)
			MPC.P87PFS = 0Ah (TXD10 pin setting)
			MPC.P83PFS = 0Ah (SCK10 pin setting)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1
			PORT8.PMR.B7 = 1 (peripheral function)
			PORT8.PMR.B3 = 1 (peripheral function)

Table 2.80 Example of Transmit Data Empty Interrupt Handling during Clock-Synchronous Master Transmission Using SCI (with FIFO)

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Write transmit data.	Write data to SCFTDR.	Write data to FTDR.
2	Clear the transmit data-empty flag to 0.	After reading SCFSR.TDFE, clear to 0.	After reading SSRFIFO.TDFE, clear to 0. ICU.IR105 is cleared automatically.
3	Confirm write of all transmit data.	End interrupt handler if transmit byte count has not reached 128 bytes.	End interrupt handler if transmit byte count has not reached 128 bytes.

4	Determine transmit end.	Wait until SCFSR.TEND = 1.
		SCSCR.TIE = 0
			ICU.IER0D.IEN1 = 0 (TXI10)
			SCR.TIE = 0
			ICU.IR105 = 0
			
			<Transmit end interrupt settings>
			SCR.TEIE = 1
			ICU. GENAL0.EN8 = 1 (TEI10)
			ICU.IER0E.IEN0 = 1
			(TEI10: group interrupt)

5	Make I/O port settings.	PECRL4.PE12MD = 000b
		(PE12 set to general I/O)
		PECRL3.PE9MD = 000b
		(PE9 set to general I/O)
			—

6	End transmit operation.	SCSCR.TE = 0	—

Table 2.81 Example of Transmit End Interrupt Handling during Clock-Synchronous Master Transmission Using SCI (with FIFO)

Procedure		SH7080 Group Setting Example	RX651 Setting Example

1	Disable transmit end interrupts.	— (No transmit end interrupt)	ICU.IER0E.IEN0 = 0
			(TEI10: group interrupt)
			ICU. GENAL0.EN8 = 0 (TEI10)
			SCR.TEIE = 0

2	Make I/O port settings.	—	PORT8.PMR.B7 = 0 (set to general I/O)
			PORT8.PMR.B3 = 0 (set to general I/O)

3	End transmit operation.	—	SCR.TE = 0

2.10.11 Setting Example for Clock-Synchronous Slave Reception Using SCI (with FIFO)

An example is presented below of settings for replacing clock-synchronous communication using the serial communication interface with FIFO (SCIF) on the SH7080 Group with the clock-synchronous mode of the serial communication interface (SCIi) on the RX651.

< Specifications >

1. The SCI10 on the RSK+RX65N board is used.
2. Reception is activated by the receive data register full interrupt.
3. Processing ends once 128 bytes of data has been received.
4. After initialization finishes, LED0 illuminates to show that the SCI is ready for receive operation. When reception finish, LED1 illuminates. LED2 illuminates if a receive error occurs.

Notes

On the RSK+RX65N the JA6, RXD10, and SCK10 pins are not connected. Therefore, the following modifications must be made to the RSK+RX65N in order to perform debugging using the sample code.

(Modifications)

- Mount R83 (0 Ω resistor) and R80 (0 Ω resistor) on the board (direct connection to R83 and R80).
- Remove R285 (0 Ω resistor) and R82 (0 Ω resistor) from the board.

Table 2.82 SCI Clock-Synchronous Communication Specifications (With FIFO)

Item	Description	Remarks
Communication mode	Clock-synchronous serial communication	
Data length	8 bits	
Hardware flow control	None	
Bit order	LSB-first	
SCI channel used	SCI10	
Synchronous clock	External clock	
Pins used	P86/RXD10	
	P83/SCK10	
	P03/GPIO	LED0 output
	P05/GPIO	LED1 output
	P73/GPIO	LED2 output

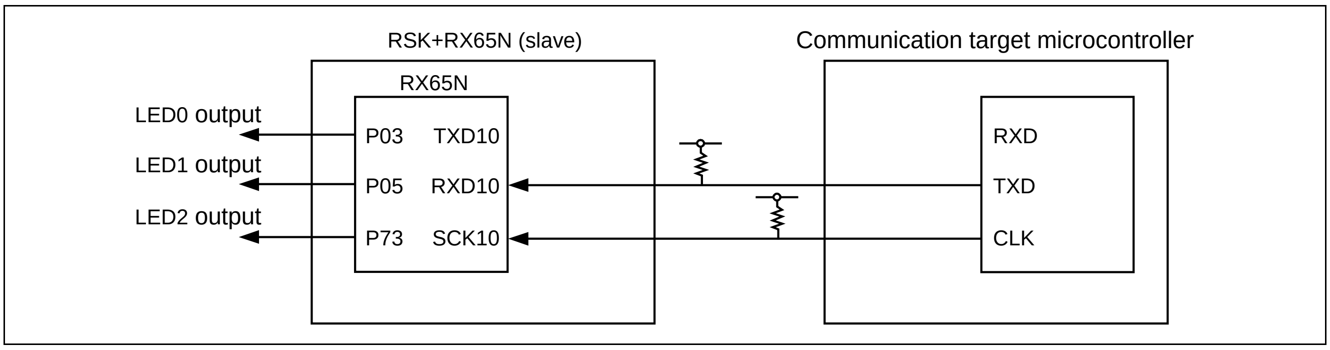


Figure 2.27 Clock-Synchronous Serial Communication Connection Specifications (Slave Receive)

< List of related registers >

On the SH7080 Group and RX651 the settings of the interrupt-related registers of the SCI10 are the same as those for asynchronous operation, as listed in Table 2.72. The points of difference from asynchronous operation are that parity error, framing error, and receive data ready are not available as sources for the receive data full and error interrupts.

The initial settings for clock-synchronous serial communication using the SCI with FIFO are presented below.

Table 2.83 Initial Setting Example for Clock-Synchronous Slave Reception Using SCI (with FIFO)

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock): 120 MHz

1	Cancel module-stop state.	STB.STBCR3.MSTP14 = 0	SYSTEM.PRCR = A502h
			SYSTEM.MSTPCRC.MSTPC25 = 0
			SYSTEM.PRCR = A500h

2	Disable SCI interrupts.	On the SH7080 Group the interrupt controller does not have enable registers.	ICU.IER0D.IEN0 = 0 (RXI10)
			ICU.IER0D.IEN1 = 0 (TXI10)
			ICU.IER0E.IEN0 = 0
			(TEI10, ERI10: group interrupt)
			ICU.GENAL0.EN8 = 0 (TEI10: SCI10)
			ICU.GENAL0.EN9 = 0 (ERI10: SCI10)

3	Initialize control registers.	SCSCR.TIE, RIE, TE, RE0	SCR.TIE, RIE, TE, RE, TEIE = 0

4	Make I/O port settings (RX651 only).	PFC setting is performed in step 13.	PORT8.PDR.B6 = 0 (set to input)
			PORT8.PDR.B3 = 0 (set to input)
			PORT8.PMR.B6 = 0 (set to general I/O)
			PORT8.PMR.B3 = 0 (set to general I/O)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enabled)
			MPC.P86PFS = 0Ah (RXD10 pin setting)
			MPC.P83PFS = 0Ah (SCK10 pin setting)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1
			PORT8.PMR.B6 = 1 (peripheral function)
			PORT8.PMR.B3 = 1 (peripheral function)

5	Reset FIFO.	SCFCR.RFRST = 1
		(receive FIFO reset operation enabled)
			FCR.FM = 1 (FIFO mode)
			FCR.RFRST = 1 (receive FIFO reset)

6	Make FIFO mode settings (RX651 only).	FIFO mode settings are performed in step 12.	FCR.RTRG[3:0] = 8 (threshold value for setting SSRFIFO.RDF flag to 1)

7	Initialize status registers (SH7080 Group only).	SCFSR.ER = 0 (receive error cleared)
		SCFSR.DR = 0
		(receive data ready cleared)
		SCFSR.BRK = 0 (break detection cleared)
		SCLSR.ORER = 0
		(overrun error cleared)
		Note: Cleared to 0 after being read.
			—

8	Enable clock.	External clock/SCK pin clock input SCSCR.CKE[1:0] = 10b	SCKn pin clock input pin SCR.CKE[1:0] = 10b

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock): 120 MHz
9	Initialize SIMR1 and SPMR.	—	SIMR1.IICM = 0 SPMR.CKPH, CKPOL = 0 (may be omitted if no change from initial value)

10	Make transmit and receive format settings.	SCSMR.C/_A = 1 (clock-synchronous)
		SCSMR.CKS[1:0] = 00b (Pϕ)
			SMR.CM = 1 (clock-synchronous)
			SMR.CKS[1:0] = 01b (PCLKA/4)
			SCMR.SMIF = 0
			(serial communication interface mode)
			SCMR.SINV = 0
			(no inversion of transmit and receive data)
			SCMR.SDIR = 0 (LSB-first)

11	Wait for 1 bit period.	←	—
12	Make FIFO mode settings (SH7080 Group only).	SCFCR.TTRG[1:0] = 00b (transmit FIFO data count trigger: 8) SCFCR.MCE = 0 (CTS and RTS disabled) SCFCR.TFRST = 0 (transmit FIFO reset operation disabled)	FIFO mode setting is performed in step 6.

13	Make I/O port settings (SH7080 Group only).	PFC setting is performed.
		PEIORL.PE11IOR = 0 (input)
		PEIORL.PE9IOR = 0 (input)
		PECRL4.PE11MD = 011b (RXD3)
		PECRL3.PE9MD = 011b (SCK3)
			Performed in step 4.

14	• Make interrupt priority setting.
	• Clear the interrupt source.
	• Enable interrupts on interrupt controller.
		INTC.IPRL.WORD = 0050h (level 5)	ICU.IPR104 = 05h (level 5)
			ICU.IPR112 = 05h (level 5)
			ICU.IR104 = 0 (RXI10)
			ICU.IER0D.IEN0 = 1 (RXI10)
			ICU.GENAL0.EN9 = 1 (ERI10: SCI10)
			ICU.IER0E.IEN0 = 1
			(ERI10: group interrupt)

15	Enable SCR.RIE and SCR.RE (reception enabled).	SCSCR.RIE = 1
		SCSCR.RE = 1
			SCR.RIE = 1
			SCR.RE = 1

Table 2.84 Example of Data Full Interrupt Handling during Clock-Synchronous Slave Reception Using SCI (with FIFO)

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Read receive data.	Read receive data count specified by SCFDR.R[4:0] from SCFRDR to receive buffer.	Read receive data count specified by FDR.R[4:0] from FRDR to receive buffer.
2	Clear receive data register-full flag.	Read SCFSR.RDF then clear to 0.	Read SSRFIFO.RDF then clear to 0. ICU.IR104 is cleared automatically.
3	Confirm read of all receive data.	End interrupt handler if receive byte count has not reached 128 bytes.	End interrupt handler if receive byte count has not reached 128 bytes.

4	End receive operation.	SCSCR.RIE = 0
		SCSCR.RE = 0
			ICU.IER0D.IEN0 = 0 (RXI10)
			ICU.IER0E.IEN0 = 0
			(ERI10: group interrupt)
			ICU.GENAL0.EN9 = 0 (ERI10: SCI10)
			SCR.RIE = 0
			SCR.RE = 0
			ICU.IR104 = 0

The details of error handling are not stipulated in the sample code. However, on the RX651 the receive error interrupt is assigned to a group interrupt. It is therefore necessary to detect the interrupt flag from the group.

2.11 Synchronous Serial Communication Unit (SSU)

2.11.1 Comparison of Specifications

On the SH7080 Group the integrated SSU and on the RX651 the integrated RSPIc with multi-master mode support provide synchronous serial communication unit functionality.

Table 2.85 presents a comparison of the specifications of the SH7080 Group and RX651.

Table 2.85 Comparison of SH7080 Group and RX651 Specifications (SSU)

Item		SH7080 Group (SSU)	RX651 (RSPIc)
Number of channels		1 channel	3 channels
Clock sources		Peripheral clock (Pϕ) External clock (SSCK)	Peripheral module clock (PCLKA) External clock (RSPCK)
Transmit/receive data length		8, 16, or 32 bits	8 to 16, 20, 24, or 32 bits
Transfer operation		SSU (4-wire) Clock-synchronous communication (3-wire)	SPI (4-wire) Clock-synchronous (3-wire)
Data format		Selectable between MSB-first and LSB-first	
SSU (SPI)	Clock phase/ polarity	Variable	
	Operating modes	• Master transmit mode • Master receive mode • Slave transmit mode • Slave receive mode	
	Communication operating mode	Full-duplex communication	Selectable between full duplex and transmit only
	Multi-master support	None	Multi-master and multi-slave support
	Bidirectional mode	Data transmission/reception on SSO pin	None
	Other	—	Sequence control
DTC/DMAC activation		DTC activation supported	DTC/DMAC activation supported
Interrupt sources		• Transmit data empty • Receive data full • Transmit end • Overrun error • Conflict error	• Transmit data empty • Receive buffer full • RSPI idle • Overrun error • Underrun error • Parity error • Mode fault error
Other		—	• Event link • Parity bit addition • Loopback mode • Byte swapping • SSLn pin active polarity setting

2.11.2 Register Comparison

Table 2.86 is a comparative listing of the registers on the SH7080 Group and RX651.

Guide to Symbols in "Changes" Column of Table

◎: Register with same bit assignments on SH7080 Group and RX651

△: Register with different bit assignments on SH7080 Group and RX651

—: Register not present on SH7080 Group or RX651

Table 2.86 SH7080 Group and RX651 Register Comparison (SSU)

SH7080 Group (SSU)	RX651 (RSPIc)	Changes
—	RSPIn n: 0 to 2	
SS control register H (SSCRH)	RSPI control register (RSPIn.SPCR)*1 SPI pin control register (RSPIn.SPPCR)	△
SS control register L (SSCRL)	RSPI control register (RSPIn.SPCR)*1 RSPI data control register (RSPIn.SPDCR)*1 RSPI command registers 0 to 7 (SPCMD0 to 7)*1	△
SS mode register (SSMR)	RSPI command registers 0 to 7 (SPCMD0 to 7)*1 RSPI bit rate register (RSPIn.SPBR)	△
SS enable register (SSER)	RSPI control register (RSPIn.SPCR)*1	△
SS status register (SSSR)	RSPI status register (RSPIn.SPSR)	△
SS control register 2 (SSCR2)	—	—
SS transmit data registers 0 to 3 (SSTDR0 to 3) SS receive data registers 0 to 3 (SSRDR0 to 3)	RSPI data register (RSPIn.SPDR) RSPI data control register (RSPIn.SPDCR)*1	△
—	RSPI data control register 2 (RSPIn.SPDCR2)	—
	RSPI control register 2 (RSPIn.SPCR2)	
	RSPI slave select polarity register (RSPIn.SSLP)	
	RSPI sequence control register (RSPIn.SPSCR)	
	RSPI sequence status register (RSPIn.SPSSR)	
	RSPI slave select negation delay register (RSPIn.SSLND)	
	RSPI clock delay register (RSPIn.SPCKD)	
	RSPI next access delay register (RSPIn.SPND)	

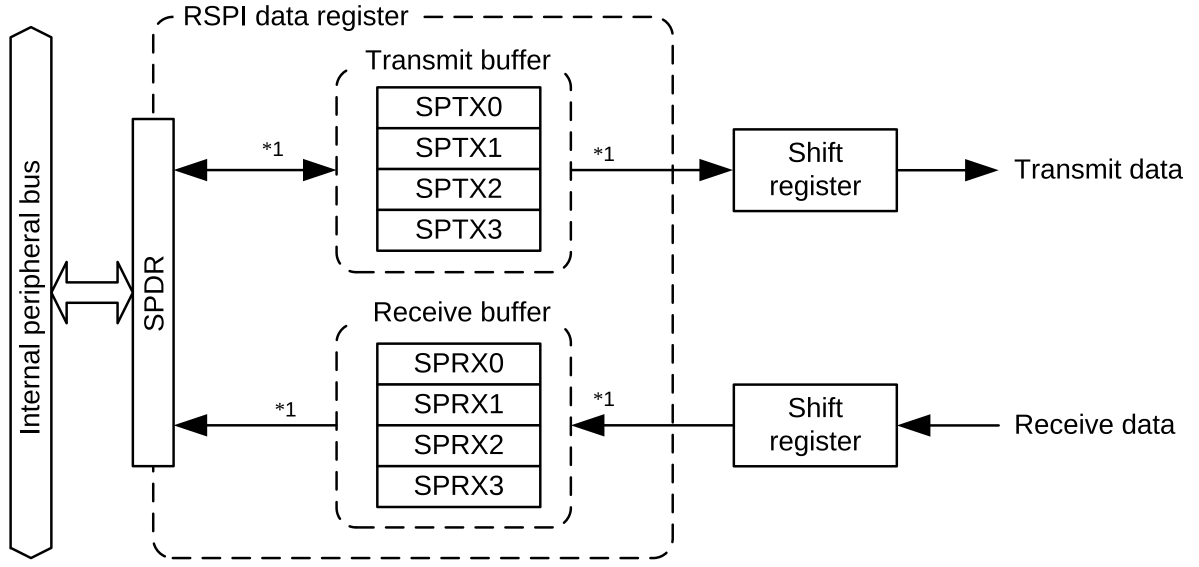
Note 1. The functions of some registers on the SH7080 Group are spit between multiple registers on the RX651.

2.11.3 Data Register Configuration

The SH7080 Group has separate data registers for transmission and reception, and data is automatically exchanged with a shift register in order to accomplish transmission and reception of serial data.

On the RX651 the same data register is used for both transmission and reception, and data is automatically exchanged among the receive buffer, transmit buffer, and shift register in order to accomplish transmission and reception of serial data. Values can be written to the transmit buffer by writing to the RSPI data register (SPDR), and once the specified number of frames of transmit data have been written, the transmit data is transferred to the shift register. When data reception finishes, the receive data is stored in the receive buffer. When reading data from the RSPI data register (SPDR), the receive buffer or transmit buffer can be selected by making settings to the RSPI data control register (SPDCR).

Figure 2.28 shows the configuration of the data register on the RX651.



Note 1. The access target buffers are switched automatically in hardware.

Figure 2.28 RX651 Data Register Configuration

2.11.4 Input/Output Pins

The Names of the input and output pins differ on the SSU of the SH7080 Group and the RSPIc of the RX651.

The RX651 has one slave select I/O pins and three slave select output pins, enabling single-master/multi-slave or multi-master /multi-slave communication among multiple devices.

Table 2.87 lists the names of the input and output pins.

Table 2.87 SSU Input/Output Pins

SH7080 Group	RX651	I/O	Function
SSCK	RSPCKn	Iunput/output	Clock I/O
SSI	MOSIn	Iunput/output	Data I/O
SSO	MISOn	Iunput/output	Data I/O
SCS#	SSLn0	Iunput/output	SH7080 Group: Chip select I/O RX651: Slave select I/O
—	SSLn1, SSLn2, SSLn3	Output	Slave select output

n: A, B, and C for each channel

2.11.5 RSPI Initialization

On the RX651 clearing the SPE bit in the RSPI control register (SPCR) to 0 disables the RSPI function, making it possible to initialize a portion of the module's functionality. In addition to a register write, the SPE bit can be cleared to 0 when a mode fault error or underrun error is detected. The RSPI control bits are not initialized when the RSPI function is disabled. This allows the RSPI to start in the same transfer mode as that before initialization when the SPE bit is once again set to 1.

When the RSPI function is disabled the RSPI transmit buffer is initialized into the empty state. Therefore, enabling transmit buffer empty interrupt requests following RSPI initialization will cause an interrupt to be generated. To disable transmit buffer empty interrupts when initializing the RSPI with the CPU, write 0 to the SPE bit simultaneously. This will prevent a transmit buffer empty interrupt request from being generated.

The following initialization takes place when the RSPI function is disabled.

- Any serial transfer that are in progress are halted.
- When in slave mode, output signal drive is halted (Hi-Z).
- The RSPI internal state is initialized.
- The RSPI transmit buffer is emptied

After a system reset the RSPI is completely initialized, with the following items being initialized in addition to the items that are initialized when the RSPI function is disabled.

- All RSPI control bits are initialized.
- The status bits are initialized.
- The data register is initialized.

2.11.6 Interrupts

On the SH7080 Group the receive buffer full and transmit buffer empty interrupts can be used to activate the DTC, but on the RX651 they can activate the DTC and the DMAC.

On the RX651 an interrupt request generated while the receive buffer full or transmit buffer empty interrupt status flag (IRn.IR) is set to 1 is stored internally by the module, and after the interrupt status flag (IRn.IR) is cleared to 0 it is once again set to 1 by the stored request.

On the RX651 some interrupts are assigned to group interrupt AL0. The interrupt controller's interrupt status flag (IRn.IR) is cleared automatically when the corresponding interrupt is accepted. The group AL0 interrupt status flag (GRPAL0.ISn) is cleared automatically when the corresponding bit in the module's status register is cleared.

Refer to 1.8 for information about interrupts.

2.11.7 Module Stop

The RSPIc of the RX651, like the SSU of the SH7080 Group, is set to the module-stop state after a reset, and no clock is supplied.

Refer to 2.16 for information on the module-stop state.

2.11.8 Setting Example for Master Transmission/Reception Using SSU Mode/SPI Operation

An example is presented below of settings for master transmission/reception processing using the SSU mode on the synchronous serial communication unit (SSU) of the SH7080 Group and using SPI operation on the serial peripheral interface (RSPIc) of the RX651.

< Specifications >

1. The RSPI0 on the RSK+RX65N board is used.
2. Transmission is activated by the transmit data empty interrupt, and reception is activated by the receive data register full interrupt.
3. Master transmission processing ends once 128 bytes of data has been transmitted to the specified address on the slave device.
4. Master reception processing ends once 128 bytes of data has been received from the specified address on the slave device.
5. After initialization finishes, LED0 illuminates to show that the RSPI is ready for transmit and receive operation. When transmission and reception finish, LED1 illuminates. LED2 illuminates if a receive error occurs.

Table 2.88 SPI Communication Specifications

Item	Description	Remarks
Communication mode	SPI operation (4-wire)	
Transfer speed	2.5 Mbps	
Bit length	8 bits	
Bit order	MSB-first	
RSPCK phase	Data change at odd edges Data sampling at even edges	
RSPCK polarity	RSPCK high in idle state	
Bit rate	Base bit rate/4	
SSL assert signal	SSLA0	
SSL negate operation	All SSL signals negated when transfer finishes	
RSPCK delay	1 RSPCK	
SSL negation delay	1 RSPCK	
Next access delay	1 RSPCK + 2 CLK	
Channel used	RSPI0	
Pins used	PA4/SSLA0	J14 on RSK+RX65N is open.
	PA5/RSPCKA	J12 on RSK+RX65N is open.
	PA6/MOSIA	J13 on RSK+RX65N is open.
	PA7/MISOA	J11 on RSK+RX65N is open.
	P03/GPIO	LED0 output
	P05/GPIO	LED1 output
	P73/GPIO	LED2 output

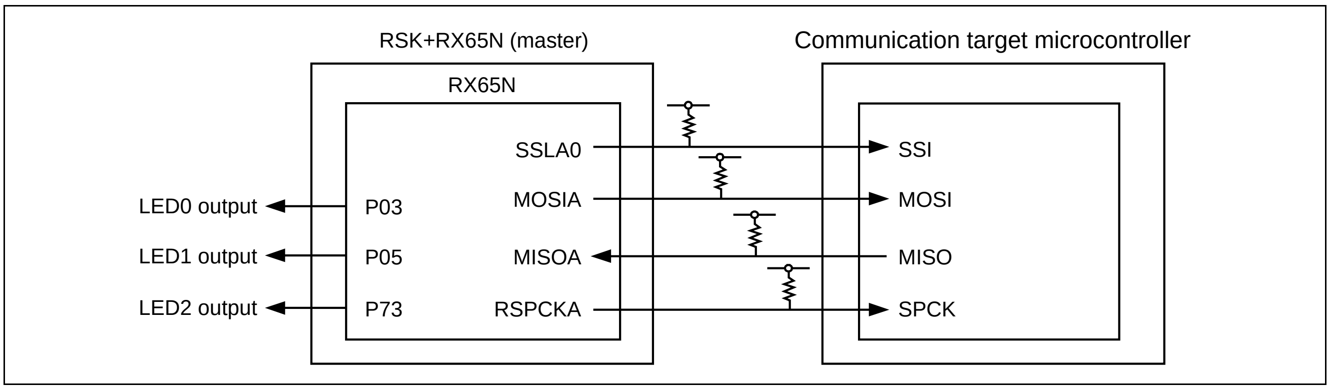


Figure 2.29 SPI Connection Specifications

< List of related registers >

The interrupt-related registers of the SSU on the SH7080 Group and RSPI0 on the RX651 are listed below by interrupt source. Table 2.89 lists the resource settings and flag checking necessary to implement on the RX651 operation equivalent to the receive buffer full, transmit buffer empty, transmit end, and error interrupts on the SH7080 Group.

Note that the RX651 does not support transmit end interrupt requests, but it is possible to implement equivalent functionality by using RSPI idle interrupt requests.

Table 2.89 SSU/RSPI Interrupt-Related Resources

Item	SH7080 Group			RX651			
Interrupt sources	SSRXI	SSTXI	SSERI	SPRI	SPTI	SPII	SPEI
Interrupt priority registers	IPRM (15-12)			IPR038	IPR039	IPR112	
Interrupt enable registers	—	—	—	IER04 .IEN6	IER04 .IEN7	IER0E.IEN0	
						GENAL0 .EN16	GENAL0 .EN17
Interrupt request registers	—	—	—	IR038	IR039	IR112	
						GRPAL0 .IS16	GRPAL0 .IS17
Interrupt enable bits	SSER .RIE	SSER .TIE SSER .TEIE	SSER .RIE SSER .CEIE	SPCR .SPRIE	SPCR .SPTIE	SPCR2 .SPIIE	SPCR .SPEIE
Status register	SSSR .RDRF	SSSR .TDRE SSSR .TEND	SSSR .ORER SSSR .CE	SPSR .SPRF	SPSR .SPTEF	SPSR .IDLNF	SPSR .MODF SPSR .OVRF SPSR .PERF SPSR .UDRF*[1]

Note 1. Not generated in master mode.

The register symbols and full names are as follows:

- SH7080 Group
  IPRM: Interrupt priority level setting register M
  SSER: SS enable register
  SSSR: SS status register
- RX651
  IPRr: Interrupt source priority register (r: vector number)
  IER04 and IER0E: Interrupt request enable registers 04 and 0E
  IRn: Interrupt request register (n: vector number)
  GENAL0: Group AL0 interrupt request enable register
  GRPAL0: Group AL0 interrupt request register
  SPCR: RSPI control register
  SPSR: RSPI status register

Table 2.90 SSU/RSPI Initial Setting Example

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock): 120 MHz

1	Cancel module-stop state.	STB.STBCR3.MSTP10 = 0	SYSTEM.PRCR = A502h
			SYSTEM.MSTPCRB.MSTPB17 = 0
			SYSTEM.PRCR = A500h

2	Disable interrupts.	—	ICU.IER04.IEN6 = 0 (SPRI0)
			ICU.IER04.IEN7 = 0 (SPTI0)
			ICU.IER0E.IEN0 = 0
			(SPII0, SPEI0: group interrupt)
			ICU.GENAL0.EN16 = 0 (SPII0: RSPI0)
			ICU.GENAL0.EN17 = 0 (SPEI0: RSPI0)

3	Stop transmit/receive operation.	SSER.TE, RE = 0	SPCR.SPE = 0

4	Make I/O port settings (SH7080 Group only).	PFC setting is performed.
		PECRL4.PE12MD = 101b (SCS)
		PECRL3.PE10MD = 101b (SSO)
		PECRL3.PE8MD = 101b (SSCK)
		PECRL2.PE7MD = 101b (SSI)
			—

5	Make transmit and receive format settings.	SSCRH.MSS = 1 (master mode)
		SSCRH.BIDE = 0 (standard mode)
		SSCRH.CSS[1:0] = 11b
		(SCS auto-output function)
		SSCRL.SSUMS = 0 (SSU mode)
		SSCRL.FCLRM = 1
		(interrupt flag cleared when register accessed)
			—

		—	SSLP.SSL0P = 0 (active low)
			SPPCR.MOIFE = 0
			(MOSI output value during SSL negation period is final data of previous transfer)

		—	SPDCR.SPFC[1:0] = 00b (frame count: 1) SPDCR.SPBYT = 1 (byte access to SPDR register) SPCR2.SPPE = 0 (no parity) SPCR2.SPIIE = 0 (idle interrupts disabled) SPSCR.SPSLN[2:0] = 000b (sequence length: 1)

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock): 120 MHz
5	Make transmit and receive format settings.	SSCRL.DATS[1:0] = 00b (8-bit data length)	SPCMD0.SPB[3:0] = 0111b (data Length: 8 bits)
		SSMR.MLS = 1 (MSB-first)	SPCMD0.LSBF = 0 (MSB-first)

		—	SPCMD0.SPNDEN = 0
			(next access delay = 1 RSPCK + 2 PCLK)
			SPCMD0.SLNDEN = 0
			(SSL negation delay: 1 RSPCK)
			SPCMD0.SCKDEN = 0
			(RSPCK delay: 1 RSPCK)

		SSMR.CPHS = 0 (SSCK clock phase: data change at initial edge)	SPCMD0.CPHA = 1 (data change at odd edges, data sampling at even edges)
		SSMR.CPOS = 0 (SSCK clock polarity: high output when idle, low output when active)	SPCMD0.CPOL = 1 (RSPCK high when idle)
		—	SPCMD0.SSLA[2:0] = 000b (SSL signal assert control: SSL0) SPCMD0.SSLKP = 0 (all SSL signals negated when transfer finishes)
6	Set bit rate (BRR).	SSMR.CKS[2:0] = 011b (Pϕ/16)	SPCMD0.BRDV[1:0] = 10b (base bit rate/4) SPBR = 5

7	Make timing settings (SH7080 Group only).	SSCR2.TENDSTS = 1
		(TEND bit set after transmission of last bit)
		SSCR2.SCSATS = 1
		(min. tLEAD and tLAG output period set to 3/2 × tSUcyc)
		SSCR2.SSODTS = 0
		(data output on SSO pin when BIDE = 0, MSS = 1, and TE = 1 or BIDE = 1, TE = 1, and RE = 0)
			—

8	Make interrupt priority setting.	IPRM.IPR[15:12] = 5h (level 5)	ICU.IPR038 = 05h (level 5)
			ICU.IPR039 = 05h (level 5)
			ICU.IPR112 = 05h (level 5)

9	Clear interrupt request.	—	ICU.IR038 = 0 (SPRI0)
			ICU.IR039 = 0 (SPTI0)

10	Enable interrupts on interrupt controller.	—	ICU.GENAL0.EN16 = 1 (SPII0: RSPI0)
			ICU.GENAL0.EN17 = 1 (SPEI0: RSPI0)
			ICU.IER04.IEN6 = 1 (SPRI0)
			ICU.IER04.IEN7 = 1 (SPTI0)
			ICU.IER0E.IEN0 = 1
			(SPEI0: group interrupt)

11	Make I/O port settings (RX651 only).	—	PORTA.PODR.B4 = 1 (set to output 1)
			PORTA.PODR.B5 = 1 (set to output 1)
			PORTA.PDR.B4 = 1 (set to output)
			PORTA.PDR.B5 = 1 (set to output)
			PORTA.PDR.B6 = 1 (set to output)
			PORTA.PDR.B7 = 0 (set to input)
			PORTA.PMR.B4 = 0 (set to general I/O)
			PORTA.PMR.B5 = 0 (set to general I/O)
			PORTA.PMR.B6 = 0 (set to general I/O)
			PORTA.PMR.B7 = 0 (set to general I/O)

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock): 120 MHz

12	Make mode settings.	—	SPCR.SPMS = 0 (SPI operation (4-wire))
			SPCR.TXMD = 0
			(full duplex synchronous serial communication)
			SPCR.MSTR = 1 (master mode)

13	Processing before transfer start	—	SPSR.PERF = MODF = OVRF = 0
			(error source clears)
			SPCR2.SPIIE = 0
			(SPII interrupts disabled)

14	Enable transmit/receive operation.	Simultaneous ON SSER.TE, RE, TEIE, TIE, RIE, CEIE = 1	Enable necessary interrupts at same time SPE bit is enabled. SPCR.SPE, SPTIE, SPRIE, SPEIE = 1
15	Start transmission/reception (SH7080 Group only).	<Transmission/reception> Confirm that TDRE = 1, then write transmit data to SSTDR. <Reception> Perform dummy read of SSRDR.	—

16	Make I/O port settings (RX651 only).	—	MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enabled)
			MPC.PA4PFS = 0Dh (SSL pin setting)
			MPC.PA5PFS = 0Dh (RSPCK pin setting)
			MPC.PA6PFS = 0Dh (MOSI pin setting)
			MPC.PA7PFS = 0Dh (MISO pin setting)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1
			PORTA.PMR.B4 = 1 (peripheral function)
			PORTA.PMR.B5 = 1 (peripheral function)
			PORTA.PMR.B6 = 1 (peripheral function)
			PORTA.PMR.B7 = 1 (peripheral function)

The details of interrupt handling are not stipulated in the sample code. However, on the RX651 the idle interrupt and error interrupt are assigned to group interrupts. It is therefore necessary to detect the interrupt flag from the group.

Table 2.91 Example of Receive Data Full Interrupt Handling during SPI Communication

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Read receive data.	Read receive data from SSRDR.	Read receive data from SPDR.
2	Read all receive data.	End interrupt handler if receive byte count has not reached 128 bytes. Go to receive end processing when receive byte count reaches 128 bytes.	End interrupt handler if receive byte count has not reached 128 bytes.

3	End receive operation.	SSER.RE = 0	ICU.IER04.IEN6 = 0 (SPRI0)
			SPCR.SPRIE = 0
			ICU.IR038 = 0 (SPRI0)
			Operation is disabled by transmit processing.

Table 2.92 Example of Transmit Data Empty Interrupt Handling during SPI Communication

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Write all transmit data.	Go to transmit end interrupt settings (step 3) when transmit byte count reaches 128 bytes. Continue transmitting data if transmit byte count has not reached 128 bytes.	Go to transmit end interrupt settings (step 3) when transmit byte count reaches 128 bytes. Continue transmitting data if transmit byte count has not reached 128 bytes.
2	Write transmit data (continuation of transmission).	Write transmit data to SSTDR and end interrupt handling.	Write transmit data to SPDR and end interrupt handling.

3	Make transmit end interrupt settings.	Confirm that TEND = 1, clear TEND to 0, and confirm that TEND = 0.
		Wait for 1 bit period, then clear SSER.TE to 0.
			ICU.IER04.IEN7 = 0 (SPTI0)
			Clear SPCR.SPTIE to 0 clear, then confirm that SPCR.SPTIE = 0.
			ICU.IR039 = 0 (SPTI0)
			<SPII interrupt settings>
			SPCR2.SPIIE = 1
			ICU.GENAL0.EN16 = 1 (SPII0: RSPI0)

Table 2.93 Example of Error Interrupt Handling during SPI Communication

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Halt operation.	SSER.TE, RE = 0	SPCR.SPE = 0

2	Disable interrupts.	SSER.TEIE, TIE, RIE, CEIE = 0	ICU.IER04.IEN7 = 0 (SPTI0)
			ICU.IER04.IEN6 = 0 (SPRI0)
			ICU.IER0E.IEN0 = 0
			(SPII0, SPEI0: group interrupt)
			ICU.GENAL0.EN17 = 0 (SPEI0: RSPI0)
			ICU.GENAL0.EN16 = 0 (SPII0: RSPI0)
			SPCR.SPTIE, SPRIE, SPEIE, SPCR2.SPIIE = 0
			ICU.IR038 = 0 (SPRI0)
			ICU.IR039 = 0 (SPTI0)

3	Determine overrun error.	Perform error processing if SSSR.ORER is set to 1.	Perform error processing if SPSR.OVRF is set to 1.
4	Determine parity error.	—	Perform error processing if SPSR.PERF is set to 1.
5	Determine underrun error (slave only).	—	Perform error processing if SPSR.UDRF is set to 1.

Table 2.94 Example of Transmit End Interrupt Handling during SPI Communication

Procedure		SH7080 Group Setting Example	RX651 Setting Example

1	End transmit operation.	—	SPCR.SPE = 0
			ICU.GENAL0.EN16 = 0 (SPII0: RSPI0)
			SPCR2.SPIIE = 0

2.11.9 Clock-Synchronous Master Transmission Setting Example

An example is presented below of settings for master transmission processing using the synchronous serial communication unit (SSU) on the SH7080 Group and the clock-synchronous communication mode (three-wire) of the serial peripheral interface (RSPIc) on the RX651.

< Specifications >

1. The RSPI0 on the RSK+RX65N board is used.
2. Transmission is activated by the transmit data empty interrupt.
3. Processing ends once 128 bytes of data has been transmitted.
4. LED0 illuminates after initialization finishes and the SCI is ready to transmit, and LED1 illuminates when transmission is finished. LED2 illuminates if a receive error occurs.

Table 2.95 RSPI Clock-Synchronous Communication Specifications

Item	Description	Remarks
Communication mode	Clock-synchronous operation (3-wire)	
Transfer speed	2.5 Mbps	
Data length	8 bits	
Bit order	MSB-first	
RSPCK phase	Data change at odd edges Data sampling at even edges	
RSPCK polarity	RSPCK high in idle state	
Bit rate	Base bit rate/4	
SSL assert signal	SSL0	
SSL negate operation	All SSL signals negated when transfer finishes	
RSPCK delay	1 RSPCK	
SSL negation delay	1 RSPCK	
Next access delay	1 RSPCK + 2 CLK	
Channel used	RSPI0	
Pins used	PA6/MOSIA	J13 on RSK+RX65N is open.
	PA5/RSPCKA	J12 on RSK+RX65N is open.
	P03/GPIO	LED0 output
	P05/GPIO	LED1 output
	P73/GPIO	LED2 output

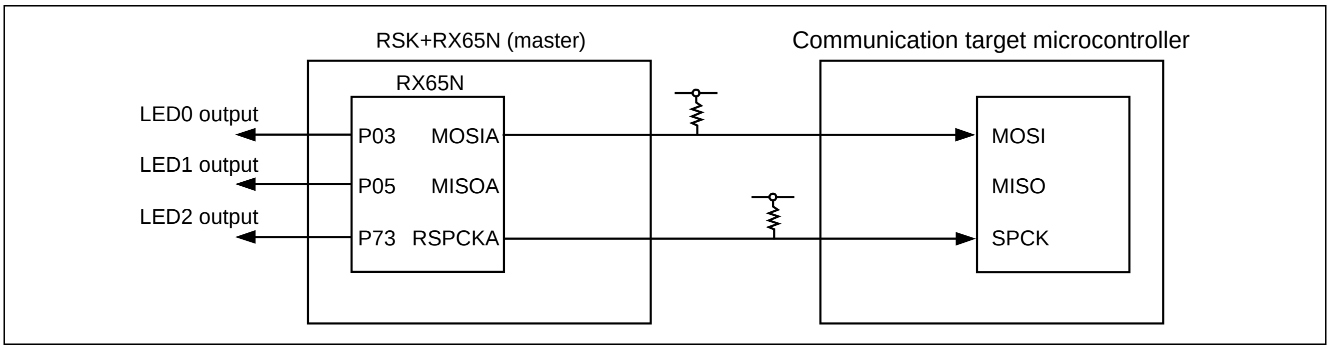


Figure 2.30 Clock-Synchronous Serial Communication Connection Specifications (Master Transmit)

< List of related registers >

Table 2.89 lists the interrupt-related registers of the SSU on the SH7080 Group and the RSPI0 on the RX651. The point of difference from SSU mode is that no conflict errors are generated.

The initial setting procedure for clock-synchronous mode on the SSU and RSPI is presented below.

Table 2.96 Example of Initial Settings for SSU/RSPI Clock-Synchronous Master Mode

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock): 120 MHz

1	Cancel module-stop state.	STB.STBCR3.MSTP10 = 0	SYSTEM.PRCR = A502h
			SYSTEM.MSTPCRB.MSTPB17 = 0
			SYSTEM.PRCR = A500h

2	Disable interrupts.	—	ICU.IER04.IEN6 = 0 (SPRI0)
			ICU.IER04.IEN7 = 0 (SPTI0)
			ICU.IER0E.IEN0 = 0
			(SPII0, SPEI0: group interrupt)
			ICU.GENAL0.EN16 = 0 (SPII0: RSPI0)
			ICU.GENAL0.EN17 = 0 (SPEI0: RSPI0)

3	Stop transmit/receive operation.	SSER.TE, RE = 0	SPCR.SPE = 0

4	Make I/O port settings (SH7080 Group only).	PFC setting is performed.
		PECRL3.PE10MD = 101b (SSO)
		PECRL3.PE8MD = 101b (SSCK)
			—

5	Make transmit and receive format settings.	SSCRH.MSS = 1 (master mode)
		SSCRH.BIDE = 0 (standard mode)
		SSCRL.SSUMS = 1
		(clock-synchronous communication mode)
		SSCRL.FCLRM = 1
		(interrupt flag cleared when register accessed)	

		—	SPDCR.SPFC[1:0] = 00b (frame count: 1) SPDCR.SPBYT = 1 (byte access to SPDR register) SPCR2.SPPE = 0 (no parity) SPCR2.SPIIE = 0 (idle interrupts disabled) SPSCR.SPSLN[2:0] = 000b (sequence length: 1)
		SSCRL.DATS[1:0] = 00b (8-bit data length) SSMR.MLS = 1 (MSB-first)	SPCMD0.SPB[3:0] = 0111b (data length: 8 bits) SPCMD0.LSBF = 0 (MSB-first)

		—	SPCMD0.SPNDEN = 0
			(next access delay = 1 RSPCK + 2 PCLK)
			SPCMD0.SCKDEN = 0
			(RSPCK delay: 1 RSPCK)

		SSMR.CPHS = 0
		(SSCK clock phase: data change at initial edge)
		SSMR.CPOS = 0
		(SSCK clock polarity: high output when idle, low output when active)
			SPCMD0.CPHA = 1
			(data change at odd edges, data sampling at even edges)
			SPCMD0.CPOL = 1
			(RSPCK high when idle)

6	Set bit rate (BRR).	SSMR.CKS[2:0] = 011b (Pϕ/16)	SPCMD0.BRDV[1:0] = 10b (base bit rate/4) SPBR = 5

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock): 120 MHz

7	Make timing settings (SH7080 Group only).	SSCR2.TENDSTS = 1
		(TEND bit set after transmission of last bit)
		SSCR2.SCSATS = 1
		(min. tLEAD and tLAG output period set to 3/2 × tSUcyc)
		SSCR2.SSODTS = 0
		(data output on SSO pin when BIDE = 0, MSS = 1, and TE = 1 or BIDE = 1, TE = 1, and RE = 0)
			—

8	Make interrupt priority setting.	IPRM.IPR[15:12] = 5h (level 5)	ICU.IPR038 = 05h (level 5)
			ICU.IPR039 = 05h (level 5)
			ICU.IPR112 = 05h (level 5)

9	Clear interrupt request.	—	ICU.IR038 = 0 (SPRI0)
			ICU.IR039 = 0 (SPTI0)

10	Enable interrupts on interrupt controller.	—	ICU.GENAL0.EN16 = 1 (SPII0: RSPI0)
			ICU.GENAL0.EN17 = 1 (SPEI0: RSPI0)
			ICU.IER04.IEN6 = 1 (SPRI0)
			ICU.IER04.IEN7 = 1 (SPTI0)
			ICU.IER0E.IEN0 = 1
			(SPEI0: group interrupt)

11	Make I/O port settings (RX651 only).	—	PORTA.PODR.B5 = 1 (set to output 1)
			PORTA.PDR.B5 = 1 (set to output)
			PORTA.PDR.B6 = 1 (set to output)
			PORTA.PMR.B5 = 0 (set to general I/O)
			PORTA.PMR.B6 = 0 (set to general I/O)

12	Make mode settings.	—	SPCR.SPMS = 1
			(clock-synchronous operation (3-wire))
			SPCR.TXMD = 0
			(full duplex synchronous serial communication)*[1]
			SPCR.MSTR = 1 (master mode)

13	Processing before transfer start	—	SPSR.PERF = OVRF = 0
			(error source clears)
			SPCR2.SPIIE = 0 (SPII interrupts disabled)

14	Enable transmit operation.	Simultaneous ON SSER.TE, TEIE, TIE = 1	Enable necessary interrupts at same time SPE bit is enabled. SPCR.SPE, SPRIE*[1], SPTIE, SPEIE = 1
15	Start transmission.	Confirm that TDRE = 1, then write transmit data to SSTDR.	Confirm that SPTEF flag = 1, then write transmit data to SPDR.

16	Make I/O port settings (RX651 only).	—	MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enabled)
			MPC.PA5PFS = 0Dh (RSPCK pin setting)
			MPC.PA6PFS = 0Dh (MOSI pin setting)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1
			PORTA.PMR.B5 = 1 (peripheral function)
			PORTA.PMR.B6 = 1 (peripheral function)

Note 1. Since the sample code uses full duplex synchronous serial communication, receive processing is also required.

Interrupt handling during clock-synchronous operation is equivalent to that during SPI operation.

Refer to 2.11.8, Setting Example for Master Transmission/Reception Using SSU Mode/SPI Operation. Note that no mode fault errors are generated.

2.11.10 Clock-Synchronous Slave Reception Setting Example

An example is presented below of settings for slave reception processing using the synchronous serial communication unit (SSU) on the SH7080 Group and the clock-synchronous communication mode (three-wire) of the serial peripheral interface (RSPIc) on the RX651.

< Specifications >

1. The RSPI0 on the RSK+RX65N board is used.
2. Reception is activated by the receive data register full interrupt.
3. Processing ends once 128 bytes of data has been received.
4. LED0 illuminates after initialization finishes and the SCI is ready to receive, and LED1 illuminates when reception is finished. LED2 illuminates if a receive error occurs.

Table 2.97 RSPI Clock-Synchronous Communication Specifications

Item	Description	Remarks
Communication mode	Clock-synchronous operation (3-wire)	
Data length	8 bits	
Bit order	MSB-first	
RSPCK phase	Data change at odd edges Data sampling at even edges	
RSPCK polarity	RSPCK high in idle state	
Bit rate	Base bit rate/2	
SSL assert signal	SSL0	
SSL negate operation	All SSL signals negated when transfer finishes	
RSPCK delay	1 RSPCK	
SSL negation delay	1 RSPCK	
Next access delay	1 RSPCK + 2 CLK	
Channel used	RSPI0	
Pins used	PA6/MOSIA	J13 on RSK+RX65N is open.
	PA5/RSPCKA	J12 on RSK+RX65N is open.
	P03/GPIO	LED0 output
	P05/GPIO	LED1 output
	P73/GPIO	LED2 output

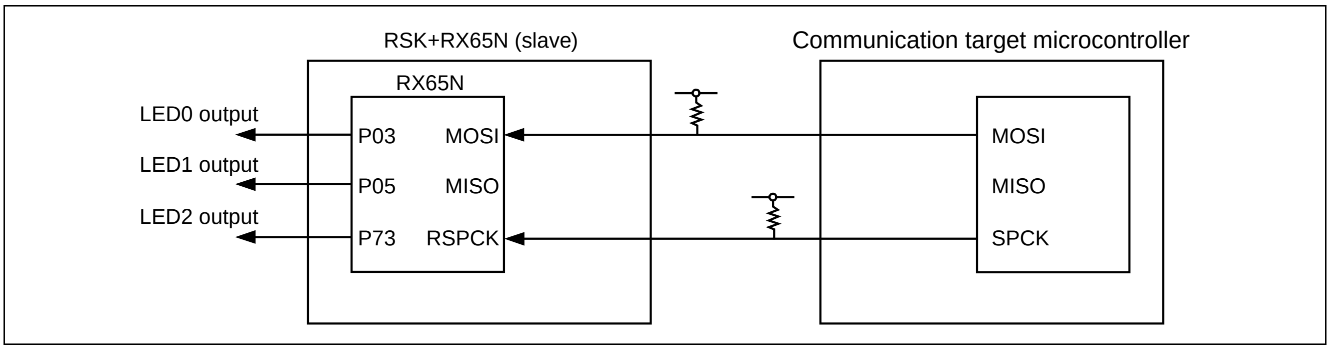


Figure 2.31 Clock-Synchronous Serial Communication Connection Specifications (Slave Receive)

< List of related registers >

Table 2.89 lists the interrupt-related registers of the SSU on the SH7080 Group and the RSPI0 on the RX651. The point of difference from SSU mode is that no conflict errors are generated.

The initial setting procedure for clock-synchronous mode on the SSU and RSPI is presented below.

Table 2.98 Example of Initial Settings for SSU/RSPI Clock-Synchronous Slave Mode

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock): 120 MHz

1	Cancel module-stop state.	STB.STBCR3.MSTP10 = 0	SYSTEM.PRCR = A502h
			SYSTEM.MSTPCRB.MSTPB17 = 0
			SYSTEM.PRCR = A500h

2	Disable interrupts.	—	ICU.IER04.IEN6 = 0 (SPRI0)
			ICU.IER04.IEN7 = 0 (SPTI0)
			ICU.IER0E.IEN0 = 0
			(SPII0, SPEI0: group interrupt)
			ICU.GENAL0.EN16 = 0 (SPII0: RSPI0)
			ICU.GENAL0.EN17 = 0 (SPEI0: RSPI0)

3	Stop transmit/receive operation.	SSER.TE, RE = 0	SPCR.SPE = 0
4	Make I/O port settings (SH7080 Group only).	PFC setting is performed. PECRL3.PE8MD2, 1, 0 = 101b (SSCK) PECRL2.PE7MD2, 1, 0 = 101b (SSI)	—

5	Make transmit and receive format settings.	SSCRH.MSS = 0 (slave mode)
		SSCRH.BIDE = 0 (standard mode)
		SSCRH.CSS[1:0] = 11b
		(SCS auto-output function)
		SSCRL.SSUMS = 1
		(clock-synchronous communication mode)
		SSCRL.FCLRM = 1
		(interrupt flag cleared when register accessed)	

		—	SPDCR.SPFC[1:0] = 00b (frame count: 1) SPDCR.SPBYT = 1 (byte access to SPDR register) SPCR2.SPPE = 0 (no parity) SPCR2.SPIIE = 0 (idle interrupts disabled) SPSCR.SPSLN[2:0] = 000b (sequence length: 1)
		SSCRL.DATS[1:0] = 00b (8-bit data length)	SPCMD0.SPB[3:0] = 0111b (data length: 8 bits)
		SSMR.MLS = 1 (MSB-first)	SPCMD0.LSBF = 0 (MSB-first)

		—	SPCMD0.SPNDEN = 0
			(next access delay = 1 RSPCK + 2 PCLK)
			SPCMD0.SCKDEN = 0
			(RSPCK delay: 1 RSPCK)

		SSMR.CPHS = 0 (SSCK clock phase: data change at initial edge)	SPCMD0.CPHA = 1 (data change at odd edges, data sampling at even edges)
		SSMR.CPOS = 0 (SSCK clock polarity: high output when idle, low output when active)	SPCMD0.CPOL = 1 (RSPCK high when idle)

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKA (Peripheral Clock): 120 MHz

6	Make interrupt priority setting.	IPRM.IPR[15:12] = 5h (level 5)	ICU.IPR039 = 05h (level 5)
			ICU.IPR038 = 05h (level 5)
			ICU.IPR112 = 05h (level 5)

7	Clear interrupt request.	—	ICU.IR038 = 0 (SPRI0)
			ICU.IR039 = 0 (SPTI0)

8	Enable interrupts on interrupt controller.	—	ICU.GENAL0.EN16 = 1 (SPII0: RSPI0)
			ICU.GENAL0.EN17 = 1 (SPEI0: RSPI0)
			ICU.IER04.IEN7 = 1 (SPTI0)
			ICU.IER04.IEN6 = 1 (SPRI0)
			ICU.IER0E.IEN0 = 1
			(SPEI0: group interrupt)

9	Make I/O port settings (RX651 only).	—	PORTA.PDR.B5 = 0 (set to input)
			PORTA.PDR.B6 = 0 (set to input)
			PORTA.PMR.B5 = 0 (set to general I/O)
			PORTA.PMR.B6 = 0
			(set to general I/O)MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enabled)
			MPC.PA5PFS = 0Dh (RSPCK pin setting)
			MPC.PA6PFS = 0Dh (MOSI pin setting)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1
			PORTA.PMR.B5 = 1 (peripheral function)
			PORTA.PMR.B6 = 1 (peripheral function)

10	Make mode settings.	—	SPCR.SPMS = 1
			(clock-synchronous operation (3-wire))
			SPCR.TXMD = 0
			(full duplex synchronous serial communication)*1
			SPCR.MSTR = 0 (slave mode)

11	Processing before transfer start	—	SPSR.PERF, OVRF = 0 (error source clears) SPCR2.SPIIE = 0 (SPII interrupts disabled)
12	Enable receive operation.	Simultaneous ON SSER. RE, RIE, CEIE = 1	Enable necessary interrupts at same time SPE bit is enabled. SPCR.SPE, SPRIE, SPTIE*1, SPEIE = 1
13	Start transmission/reception.	<Transmission/reception> Confirm that TDRE = 1, then write transmit data to SSTDR. <Reception> Perform dummy read of SSRDR.	—

Note 1. Since the sample code uses full duplex synchronous serial communication, receive processing is also required.

Interrupt handling during clock-synchronous operation is equivalent to that during SPI operation.

Refer to 2.11.8, Setting Example for Master Transmission/Reception Using SSU Mode/SPI Operation. Note that no mode fault errors are generated.

2.12 I^{2}C Bus Interface (IIC)

2.12.1 Comparison of Specifications

I^{2}C bus interface functionality is provided on the SH7080 Group by the IIC2 and on the RX651 by the RIICa, which supports communication operation compliant with SMBus (ver. 2.0).

Table 2.99 is a comparative listing of the specifications of the SH7080 Group and RX651.

Table 2.99 Comparison of SH7080 Group and RX651 Specifications (IIC)

Item		SH7080 Group (IIC2)	RX651 (RIICa)
Number of channels		1 channel	2 channels
Clock source		Peripheral clock (Pϕ)	Peripheral module clock (PCLKB)
Communication format		• I^{2}C bus format • Clock-synchronous serial format*[1]	• I^{2}C bus format • SMBus format
Data transfer		Selectable between MSB-first and LSB-first	Fixed at MSB-first
I^{2}C bus format (SMBus)	Operating modes	• Master transmit mode • Master receive mode • Slave transmit mode • Slave receive mode	
	Start condition/ stop condition	Automatically generated	
	Address detection	• 7-bit slave addresses	• 3 types of 7- or 10-bit slave addresses • General call address • Device ID address • Host address
	DTC/DMAC activation	DTC activation supported	DTC/DMAC activation supported
	Interrupt sources	• Arbitration lost/overrun error • NACK detection • Stop condition detection • Receive data full • Transmit data empty • Transmit end	• Arbitration detection • NACK detection • Timeout detection • Start condition detection • Stop condition detection • Receive data full • Transmit data empty • Transmit end
	Multi-master support	Bit synchronization circuit Ability to specify a transfer rate at least 1/1.8 times the fastest transfer rate of another master	SCL synchronization circuit
Noise cancellation		Ability to specify the noise cancellation width for the SCL and SDA pins Up to 3-stage latch circuit	Ability to enable digital noise filter and specify the noise cancellation width for the SCL and SDA pins Up to 4-stage noise filter
Other		—	• Event link • SCL clock duty ratio setting • SDA output delay function • SCL auto low-hold function • Bus hang-up support

Note 1. The RIICa on the RX651 does not support clock-synchronous serial format, but the clock-synchronous communication format of the SCIg and SCIh can be used as a substitute.

2.12.2 Register Comparison

Table 2.100 is a comparative listing of the registers on the SH7080 Group and RX651.

Guide to Symbols in "Changes" Column of Table

◎: Register with same bit assignments on SH7080 Group and RX651

△: Register with different bit assignments on SH7080 Group and RX651

—: Register not present on SH7080 Group or RX651

Table 2.100 SH7080 Group and RX651 Register Comparison (IIC)

SH7080 Group (IIC2)	RX651 (RIICa)	Changes
—	RIICn n: 0 or 2	
I^2C bus control register 1 (ICCR1) I^2C bus control register 2 (ICCR2)	I^2C bus control register 1 (RIICn.ICCR1) I^2C bus control register 2 (RIICn.ICCR2)	△
I^2C bus mode register (ICMR)	I^2C bus mode register 1 (RIICn.ICMR1)	△
—	I^2C bus mode register 2 (RIICn.ICMR2)	—
NF2CYC register (NF2CYC) I^2C bus interrupt enable register (ICIER)	I^2C bus mode register 3 (RIICn.ICMR3) I^2C bus interrupt enable register (RIICn.ICIER)	△
I^2C bus status register (ICSR)	I^2C bus status register 1 (RIICn.ICSR1) I^2C bus status register 2 (RIICn.ICSR2)	△
Slave address register (SAR)	Slave address register Ly (RIICn.SARLy) (y = 0 to 2)	△
—	Slave address register Uy (RIICn.SARUy) (y = 0 to 2)	—
I^2C bus transmit data register (ICDRT)	I^2C bus transmit data register (RIICn.ICDRT)	◎
I^2C bus receive data register (ICDRR)	I^2C bus receive data register (RIICn.ICDRR)	◎
I^2C bus shift register (ICDRS)	I^2C bus shift register (ICDRS)	◎
—	I^2C bus bit rate high-level register (RIICn.ICBRL)	—
	I^2C bus bit rate low-level register (RIICn.ICBRH) I^2C bus function enable register (RIICn.ICFER)	
	I^2C bus status enable register (RIICn.ICSER)	

2.12.3 Address Detection

The SH7080 Group can detect 7-bit slave addresses of a single type.

The RX651 can detect three types of slave addresses, as well as general call addresses, device ID addresses, and host addresses. In addition, the slave address bit count can be specified as either 7-bit or 10-bit.

Figure 2.32 shows the RX651 I^{2}C bus format.

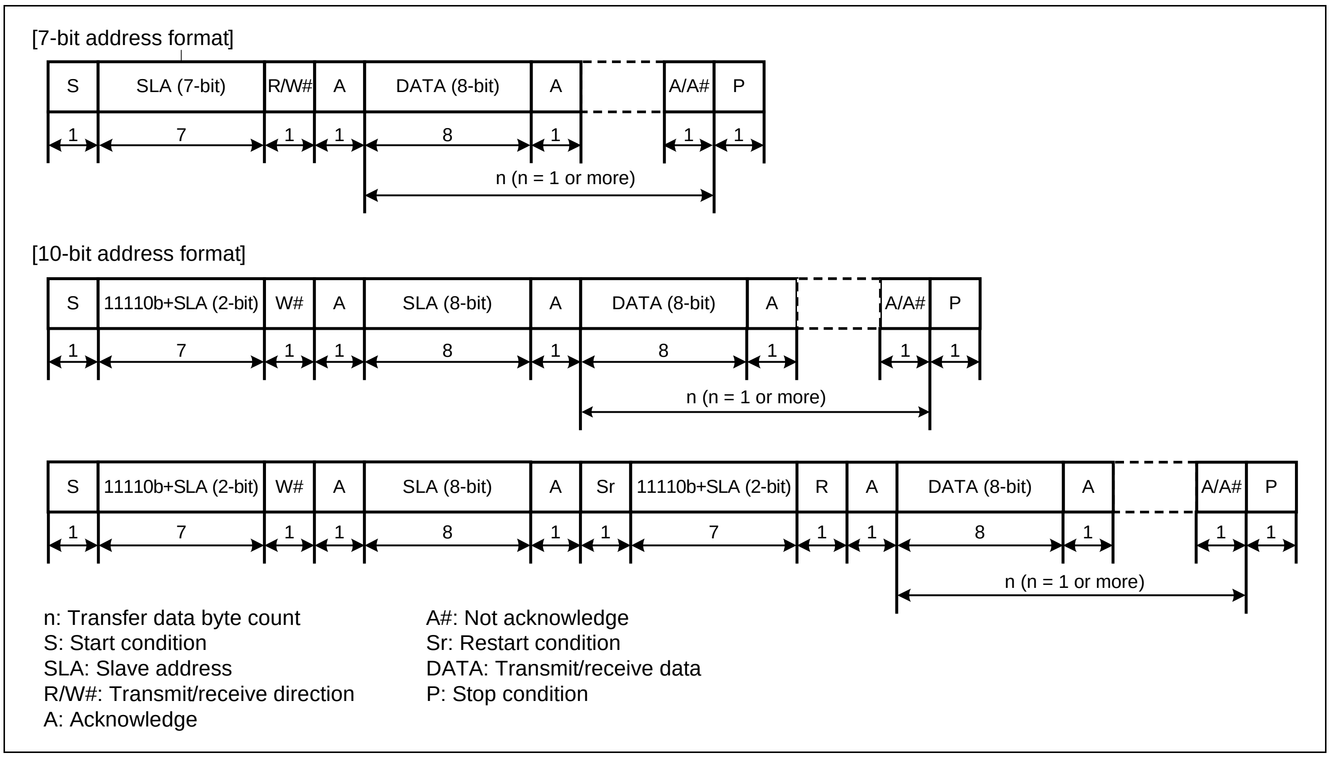


Figure 2.32 RX651 I^{2}C Bus Format

2.12.4 Arbitration Detection

In addition to the ordinary arbitration lost detection function stipulated in the I^{2}C bus specification, the RX651 provides functions for prevention of issuance of overlapping start requests, arbitration lost detection during NACK transmission, and arbitration lost detection during slave receive operation.

2.12.5 Bus Hang-up

If synchronization of the master device and slave device on the I^{2}C bus is disrupted due to noise or the like, a bus hang-up may occur where the SCL line or SDA line becomes fixed at a single level.

To deal with bus hang-ups, the RX651 provides a timeout detection function that monitors the SCL line to detect bus hang-up states and, to recover from bus hang-up states caused by disrupted synchronization, an SCL clock additional output function, an RIIC reset function, and an internal reset function.

2.12.6 SCL Clock

Under the I²C bus format transmission and reception of data are synchronized with the SCL clock output by the master device.

When operating in master mode, the SCL clock transfer rate on the SH7080 Group is determined by the peripheral clock division ratio setting in I²C bus control register 1 (ICCR1). On the RX651 the SCL transfer rate and duty ratio are determined by the SCL clock high-level period setting in the I²C bus bit rate high-level register (ICBRH) and the SCL clock low-level setting in the I²C bus bit rate low-level register (ICBRL).

The RX651 provides a transmit data accidental transmission prevention function, a NACK receive transfer cutoff function, and a receive data loss prevention function. The SCL line is automatically held low when certain conditions are met.

When the I²C bus format is used in a multi-master configuration, conflicts can arise between the SCL clock that that of the other master device. This is why the SH7080 Group is provided with a bit synchronization circuit, and the RX651 with an SCL synchronization circuit, that monitors the SCLn line in master mode and generates the SCL clock with bit-by-bit synchronization.

Figure 2.33 illustrates SCL clock generation and SCL synchronization on the RX651.

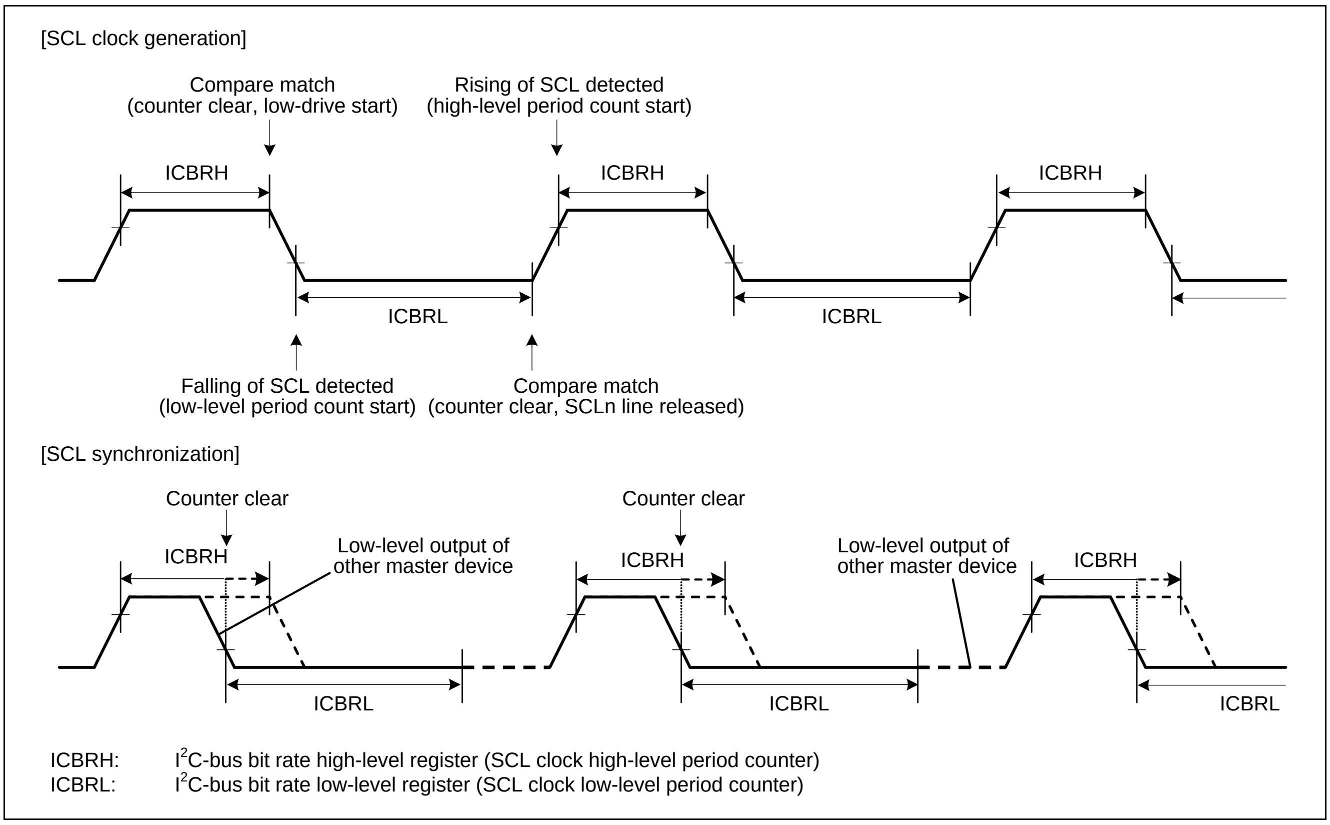

[SCL clock generation]
Compare match (counter clear, low-drive start)
Rising of SCL detected (high-level period count start)
ICBRH
ICBRH
ICBRH
ICBRL
ICBRL
Falling of SCL detected (low-level period count start)
Compare match (counter clear, SCLn line released)
[SCL synchronization]
Counter clear
Counter clear
ICBRH
Low-level output of other master device
ICBRH
Low-level output of other master device
ICBRH
ICBRL
ICBRL
ICBRL
ICBRH: I²C-bus bit rate high-level register (SCL clock high-level period counter)
ICBRL: I²C-bus bit rate low-level register (SCL clock low-level period counter)

Figure 2.33 SCL Clock Generation and SCL Synchronization

2.12.7 Noise Cancellation

On the RX651 it is possible to select the noise cancellation width as on the SH7080 Group, and the noise canceler function is enabled in the default state. Use the I²C bus function enable register (ICFER) to enable or disable the digital noise filter circuit.

2.12.8 Interrupts

On the SH7080 Group the receive data full and transmit data empty interrupts can be used to activate the DTC, but on the RX651 they can activate the DTC and the DMAC.

On the RX651 when a receive data full or transmit data empty interrupt occurs while the corresponding interrupt status flag (IRn.IR) is set to 1, the interrupt request is also stored internally by the module, and after the interrupt status flag (IRn.IR) is cleared to 0 it is reset to 1 by the stored request.

On the RX651 some interrupts are assigned to group interrupt BL1. The interrupt controller's interrupt status flag (IRn.IR) is cleared automatically when the corresponding interrupt is accepted. The group BL1 interrupt status flag (GRPBL1.ISn) is cleared automatically when the corresponding bit in the module's status register is cleared.

Table 2.101 and Table 2.102 list interrupt sources for the SH7080 Group and RX651.

Refer to 1.8 for information about interrupts.

Table 2.101 SH7080 Group IIC2 Interrupt Sources (I^2C Bus Format)

Interrupt Source	DTC Activation	DMAC Activation	Priority
NACK detection	Not possible	Not possible	High
Arbitration lost/overrun error			↑
Transmit end			
Stop condition detection			
Transmit data empty	Possible		
Receive data full			Low

Table 2.102 RX651 RIICa Interrupt Sources

Interrupt Source		DTC Activation	DMAC Activation	Priority
Communication error/event occurrence	Arbitration lost	Not possible	Not possible	High
	NACK detection			↑
	Timeout			
	Start condition detection			
	Stop condition detection			
Receive data full		Possible	Possible	
Transmit data empty				
Transmit end		Not possible	Not possible	Low

2.12.9 Module Stop

As on the SH7080 Group, the RIICa of the RX651 is set to the module-stop state after a reset, and no clock is supplied.

Refer to 2.16 for information on the module-stop state.

2.12.10 Setting Example for Master Transmission/Reception

An example is presented below of settings for master transmit/receive processing using the IIC bus interface.

< Specifications >

1. The RIIC0 on the RSK+RX65N board is used.
2. Transmission is activated by the transmit data empty interrupt, and reception is activated by the receive data full interrupt.
3. Master transmit processing is used to transmit 32 bytes of data, then master receive processing is used to receive 32 bytes of data. 50h is used as the address of the slave device.
4. LED0 illuminates to show that the IIC bus interface is ready for transmit and receive operation. LED1 illuminates when transmission finishes, and LED2 illuminates when receive finishes. LED3 illuminates if an error occurs.

Table 2.103 IIC Communication Specifications

Item	Description	Remarks
Communication mode	I^{2}C bus	
Transfer speed	400 kHz	
Data bit count	9 bits (including ACK)	
Wait between data and ACK	None	
ACK determination	Transfer halted when ACK = 1 received	
Pins used	P12/SCL0	LED0 output
	P13/SDA0	LED1 output
	P03/GPIO	LED2 output
	P05/GPIO	LED3 output
	P73/GPIO	LED0 output

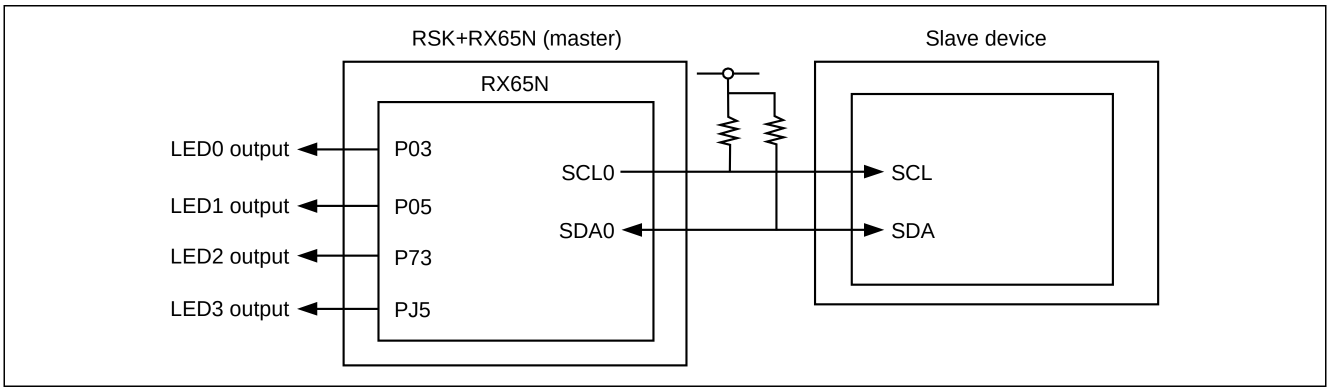


Figure 2.34 IIC Connection Specifications

< List of related registers >

The interrupt-related registers of RIIC0 on the SH7080 Group and RX651 are listed below by interrupt source. Table 2.104 lists the resource settings and flag checking necessary to implement on the RX651 operation equivalent to the receive, transmit, transmit end, and receive error interrupts on the SH7080 Group.

Table 2.104 IIC Interrupt-Related Resources

Item	SH7080 Group					RX651			
Interrupt sources	IIRXI	IITXI	IITEI	IISTPI	IINAKI	RXI	TXI	TEI	EEI
Interrupt priority registers	IPRM (11-8)				IPRH (11-8)	IPR052	IPR053	IPR111	
Interrupt enable registers	—	—	—	—	—	IER06 .IEN4	IER06 .IEN5	IER0D .IEN7	
								GENBL1 .EN13	GENBL1 .EN13
Interrupt request registers	—	—	—	—	—	IR052	IR053	IR111	
								GRPBL1 .IS13	GRPBL1 .IS13
Interrupt request flag	ICIER. RIE	ICIER. TIE	ICIER .TEIE	ICIER .STIE	ICIER .NAKIE	ICIER .RIE	ICIER .TIE	ICIER .TEIE	ICIER .ALIE ICIER .NAKIE ICIER .TMOIE ICIER .STIE ICIER .SPIE
Status register	ICSR .RDRF	ICSR. TDRE	ICSR. TEND	ICSR. STOP	ICSR. AL/OVE ICSR. NACKF	ICSR2 .RDRF	ICSR2 .TDRE	ICSR2 .TEND	ICSR2 .AL ICSR2 .NACKF ICSR2 .TMOF ICSR2 .START ICSR2 .STOP

The register symbols and full names are as follows:

- SH7080 Group
  IPRH and IPRM: Interrupt priority level registers H and M
  ICIER: IIC bus interrupt enable register
  ICSR: Input level control/status register 1

- RX651
  IPRr: Interrupt source priority register (r: vector number)
  IER06 and 0D: Interrupt request enable registers 06 and 0D
  IRn: Interrupt request register (n: vector number)
  GENBL1: Group BL1 interrupt request enable register
  GRPBL1: Group BL1 interrupt request register
  ICIER: IIC bus interrupt enable register
  ICSR2: Input level control/status register 2

Table 2.105 IIC Initial Setting Example

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKB (Peripheral Clock B): 60 MHz

1	Cancel module-stop state.	STBCR3.MSTP15 = 0	SYSTEM.PRCR = A502h
			SYSTEM.MSTPCRB.MSTPB21 = 0
			SYSTEM.PRCR = A500h

2	Disable interrupts.	On the SH7080 Group the interrupt controller does not have enable registers.	ICU.IER06.IEN5 = 0 (TXI0)
			ICU.IER06.IEN4 = 0 (RXI0)
			ICU.IER0D.IEN7 = 0
			(TEI0, EEI0: group interrupt)
			ICU.GENBL1.EN13 = 0 (TEI0)
			ICU.GENBL1.EN14 = 0 (EEI0)

3	Halt IIC function.	ICCR1.ICE = 0 (IIC2 function halted)	ICCR1.ICE = 0 (SCL0 and SDA0 pins in non-drive state)
4	Reset IIC.	ICCR2.IICRST = 1 (IIC2 reset)	ICCR1.IICRST = 1 (IIC reset)

5	Switch to slave receive mode (SH7080 Group only).	ICCR1.MST = 0 (slave mode)
		ICCR1.TRS = 0 (receive mode)
			—

6	Clear busy flag (SH7080 Group only).	SAR.FS = 1 (BBSY always 0 when clock synchronization selected)	—
7	Select I^{2}C bus format (SH7080 Group only).	SAR.FS = 0 (I^{2}C bus format selected)	—
8	Clear status flag (SH7080 Group only).	ICSR = 00h (writing 0 only allowed after reading bit as 1)	—
9	Cancel IIC2 reset (SH7080 Group only).	ICCR2.IICRST = 0 (IIC2 reset canceled)	—
10	Start IIC operation (RX651 only).	—	ICCR1.ICE = 1 (SCL0 and SDA0 pins in drive state)

11	Select function.	ICIER.ACKE = 1
		(transfer halted when receive ACK = 1)
			ICFER.NACKE = 1
			(transfer halted when NACK received)
			ICFER.SCLE = 1
			(SCL synchronization circuit enabled)

12	Make communication bit rate settings.	ICCR1.CKS[3:0] = 0101b (transfer rate: 400 kHz)	ICMR1.CKS[2:0] = 010b (internal reference clock: PCLKB/4) ICBRH.BRH[4:0] = 01000b (high-level period of SCL clock) ICBRL.BRL[4:0] = 10011b (low-level period of SCL clock)
13	Make initial settings (SH7080 Group only).	ICCR1, ICCR2, ICMR, ICIER, and NF2CYC in IIC2 initialized ICCR1.ICE = MST = TRS = 0	—
14	Set slave address (master mode only).	—	ICSER = 00h (all enabled addresses disabled)

15	Set slave address (slave mode only).	SAR.SVA[6:0] = slave address	SARU0.FS = 0 (7-bit address format)
			SARL0.SVA[6:0] = slave address
			ICSER.SAR0E = 1
			(SARL0 and SARU0 setting values = enabled)

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKB (Peripheral Clock B): 60 MHz

16	Make transmit ACK settings (slave mode only).	ICIER.ACKBT = 0
		(0 transmitted at transmit ACK timing)
			ICMR3.ACKWP = 1
			(writes to acknowledge bit enabled)
			ICMR3.ACKBT = 0
			(send 0 to acknowledge bit)
			ICMR3.ACKWP = 0
			(writes to acknowledge bit disabled)

17	Disable interrupts.	—	ICIER = 00h (interrupts disabled)

18	Make I/O port settings.	PFC.PBCRL1.PB2MD = 100b (SCL I/O)
		PFC.PBCRL1.PB3MD = 100b (SDA I/O)
			PORT1.PMR.B3 = 0 (general I/O setting)
			PORT1.PMR.B2 = 0 (general I/O setting)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enabled)
			MPC.P13PFS.PSEL[5:0] = 001111b (SDA0)
			MPC.P12PFS.PSEL[5:0] = 001111b (SCL0)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1
			PORT1.PMR.B3 = 1 (peripheral function)
			PORT1.PMR.B2 = 1 (peripheral function)

19	Make interrupt priority level settings	INTC.IPRM.WORD = 0500h (level 5)	ICU.IPR053 = 05h (level 5)
			ICU.IPR052 = 05h (level 5)
			ICU.IPR111 = 05h (level 5)

20	Clear interrupt request.	—	ICU.IR053 = 0
			ICU.IR052 = 0

21	Enable interrupts.	ICIER.TIE, RIE, NAKIE, STIE = 1	ICIER.TIE, RIE, NAKIE, SPIE, ALIE, TMOIE = 1

22	Enable interrupts on interrupt controller.	—	ICU.GENBL1.EN14 = 1 (EEI0)
			ICU.IER06.IEN5 = 1 (TXI0)
			ICU.IER06.IEN4 = 1 (RXI0)
			ICU.IER0D.IEN7 = 1
			(TEI0, EEI0: group interrupt)

23	Start IIC operation (SH7080 Group only).	ICCR1.ICE = 1 (IIC2 operation enabled)	—
24	Cancel reset (RX651 only).	—	ICCR1.IICRST = 0 (IIC reset canceled)

Table 2.106 Example of Processing at Start of Transmission

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Confirm bus released state.	Wait until ICCR2.BBSY = 0.	Wait until ICCR2.BBSY = 0.

2	Make master transmission mode settings (SH7080 Group only).	ICCR1.MST = 1 (master mode)
		ICCR1.TRS = 1 (transmit mode)
			—

3	Issue start condition.	ICCR2.BBSY = 1
		ICCR2.SCP = 0
			ICCR2.ST = 1
			(start condition issuance request)

Table 2.107 Example of Processing at End of Transmission

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Confirm bus released state.	Wait until ICCR2.BBSY = 0.	Wait until ICCR2.BBSY = 0.
2	Clear status flag.	ICSR = 00h (writing 0 only allowed after reading bit as 1)	ICSR2 = 00h (writing 0 only allowed after reading bit as 1)
3	Enable interrupts.	ICIER.TIE, RIE, NAKIE, STIE = 1	ICIER.TIE, RIE, NAKIE, SPIE, ALIE, TMOIE = 1 ICU.IER06.IEN5 = 1 (TXI0)

4	Make master receive mode settings (SH7080 Group only).	ICCR1.MST = 1 (master mode)
		ICCR1.TRS = 1 (transmit mode)
			—

5	Issue start condition.	ICCR2.BBSY = 1
		ICCR2.SCP = 0
			ICCR2.ST = 1

Table 2.108 Example of Interrupt Handling at Transfer End

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Wait until SCL output level is low (SH7080 Group only).	Wait until ICCR2.SCLO = 0 is read.	—
2	Issue stop condition.	ICCR2.BBSY = 0 and ICCR2.SCP = 0 (stop condition issued) Clear TEND and NACKF in ICSR to 0.	ICCR2.SP = 1 (stop condition issuance request)
3	Issue stop condition and wait for completion.	Wait until ICSR.STOP = 1.	Wait until ICSR2.STOP = 1.

4	End transfer.	Slave receive mode settings
		ICCR1.MST = 0 (slave mode)
		ICCR1.TRS = 0 (receive mode)
		Clear ICSR.TDRE to 0 after reading it as 1.
		(transmit data register empty flag cleared)
			ICSR2.NACKF = 0
			(NACK detection flag cleared)
			ICSR2.STOP = 0
			(stop condition detection flag cleared)

Table 2.109 Example of Transmit Data Empty Interrupt Handling

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Check operating mode.	For master transmission of the 1st frame (slave address + W), perform step 2 and end interrupt handling. For master transmission of data for the 2nd and subsequent frames, go to step 3. For master reception, go to step 5.	For master transmission of the 1st frame (slave address + W), perform step 2 and end interrupt handling. For master transmission of data for the 2nd and subsequent frames, go to step 3. For master reception, go to step 5.
2	Transmit 1st frame (slave address + W).	ICDRT = transmit data (slave address + W)	ICDRT = transmit data (slave address + W)
3	Transmit data of 2nd and subsequent frames.	ICDRT = transmit data	ICDRT = transmit data
4	If data is not last transmit data, end interrupt handling.	←	←
5	Transmit 1st frame (slave address + R).	ICDRT = transmit data (slave address + R)	ICDRT = transmit data (slave address + R)

6	Make transmit end interrupt settings.	Clear ICIER.TIE to 0 and confirm that ICIER.TIE = 0.
		<Transmit end interrupt settings>
		ICIER.TEIE = 1
			ICU.IER06.IEN5 = 0 (TXI0)
			Clear ICIER.TIE to 0 and confirm that ICIER.TIE = 0.
			ICU.IR053 = 0
			<Transmit end interrupt settings>
			ICIER.TEIE = 1
			ICU.GENBL1.EN13 = 1 (TEI0)

Table 2.110 Example of Receive Data Full Interrupt Handling

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Confirm receive state.	If reception has not started, perform steps 2 to 4 and end interrupt handling. If reception has already started, go to step 5.	If reception has not started, perform steps 2 to 4 and end interrupt handling. If reception has already started, go to step 5.
2	Confirm acknowledge from slave device.	Confirm that ICSR.NACKF = 0. If ICSR.NACKF = 1, issue the stop condition and end.	Confirm that ICSR2.NACKF = 0. If ICSR2.NACKF = 1, issue the stop condition and end.

3	Make master receive mode settings.	ICSR.TEND = 0 (transmit end flag cleared)
		ICCR1.TRS = 0 (receive mode)
		ICSR.TDRE = 0
		(transmit data register empty flag cleared)
			—

4	Start reception.	ICIER.ACKBT = 0 (transmit 0 at acknowledge timing) Dummy data = ICDRR	ICMR3.ACKWP = 1 (writes to acknowledge bit enabled) ICMR3.ACKBT = 0 (send 0 to acknowledge bit) ICMR3.ACKWP = 0 (writes to acknowledge bit disabled) Dummy data = ICDRR
5	Check receive data count.	If remaining receive data count is 1, perform processing of steps 10 and after. If remaining receive data count is 2 bytes, skip processing of steps 6 and 7 and perform processing of steps 8 and 9, then end interrupt handling.	If remaining receive data count is 1, perform processing of steps 10 and after. If remaining receive data count is 2 bytes, skip processing of steps 6 and 7 and perform processing of steps 8 and 9, then end interrupt handling. If remaining receive data count is 3 bytes, make WAIT setting. ICMR3.WAIT = 1 (held low during period between 9th and 1st clock cycles)
6	Receive data.	Receive data = ICDRR	Receive data = ICDRR
7	End interrupt handling.	←	←
8	Make NACK transmit settings. Disable continuous receive operation.	If the next receive operation is the final frame, make ACKBT and RCVD settings. ICIER.ACKBT = 1 (transmit 1 as acknowledge) ICCR1.RCVD = 1 (continuous receive operation disabled)	To notify that communication will finish with the next receive operation, make ACKBT setting. ICMR3.ACKWP = 1 (writes to acknowledge bit enabled) ICMR3.ACKBT = 1 (transmit 1 as acknowledge) ICMR3.ACKWP = 0 (writes to acknowledge bit disabled)
9	Receive last data unit – 1.	Receive data = ICDRR	Receive data = ICDRR
10	Wait until SCL goes low-level.	Wait until ICCR2.SCLO = 0.	—
11	Issue stop condition.	ICCR2.BBSY = 0 and ICCR2.SCP = 0 (stop condition issued)	ICCR2.SP = 1 (stop condition issuance request)
12	Receive last data unit (before stop condition completion on RX651).	—	Receive data = ICDRR ICMR3.WAIT = 0

Procedure		SH7080 Group Setting Example	RX651 Setting Example
13	Issue stop condition and wait for completion.	Wait until ICSR.STOP = 1.	Wait until ICSR2.STOP = 1.
14	Receive last data unit (after stop condition completion on RX651)	Receive data = ICDRR	—

15	End receive operation.	ICCR1.RCVD = 0
		(continuous receive operation enabled)
		ICCR1.MST = 0 (slave mode)
			ICSR2.NACKF = 0
			(NACK detection flag cleared)
			ICSR2.STOP = 0
			(stop condition detection flag cleared)

Table 2.111 Example of Transmit End Interrupt Handling

Procedure		SH7080 Group Setting Example	RX651 Setting Example

1	Disable transmit end interrupts.	ICIER.TEIE = 1	ICU.GENBL1.EN13 = 0 (TEI0)
			ICSR2.TEND = 0
			ICIER.TEIE = 0

2	Wait until SCL output level is low (SH7080 Group only).	Wait until ICCR2.SCLO = 0 is read.	—
3	Issue stop condition (only during master transmission).	ICCR2.BBSY = 0 and ICCR2.SCP = 0 (stop condition issued) Clear TEND and NACKF in ICSR to 0.	ICCR2.SP = 1 (stop condition issuance request)
4	Issue stop condition and wait for completion (only during master transmission).	Wait until ICSR.STOP = 1.	Wait until ICSR2.STOP = 1.

5	End transmission (only during master transmission).	Slave receive mode settings
		ICCR1.MST = 0 (slave mode)
		ICCR1.TRS = 0 (receive mode)
		Clear ICSR.TDRE to 0 after reading it as 1.
		(transmit data register empty flag cleared)
			ICSR2.NACKF = 0
			(NACK detection flag cleared)
			ICSR2.STOP = 0
			(stop condition detection flag cleared)

The details of error handling are not stipulated in the sample code. However, on the RX651 the error interrupt is assigned to a group interrupt. It is therefore necessary to detect the interrupt flag from the group.

Table 2.112 Example of IIC Error Handling

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Determine group interrupt.	—	If ICU.GRPBL1.IS14 (EEI0) = 1, perform processing for step 2 and after.
2	Determine arbitration lost.	Perform error handling if ICSR.AL/OVE = 1.	If ICSR2.AL = 1, perform error handling.*[1]
3	Determine NACK detection.	If ICSR.NACKF = 1, perform error handling or transfer end processing.	If ICSR2.NACKF = 1, perform error handling or transfer end processing.
4	Determine timeout detection.	—	If ICSR2.TMOF = 1, perform error handling.*[1]
5	Stop condition detection.	If ICSR.STOP = 1, enter finished state.	If ICSR2.STOP = 1, enter finished state.

Note 1. The arbitration lost detection function and timeout detection function are disabled in the ICFER register.

2.12.11 Setting Example for Slave Transmission/Reception

An example is presented below of settings for slave transmit/receive processing using the IIC bus interface.

< Specifications >

1. The RIIC0 on the RSK+RX65N board is used.
2. Transmission is activated by the transmit data empty interrupt, and reception is activated by the receive data full interrupt.
3. Slave transmit processing is used to transmit 32 bytes of data, then slave receive processing is used to receive 32 bytes of data. 50h is used as the address of the slave device.
4. LED0 illuminates to show that the IIC bus interface is ready for transmit and receive operation. LED1 illuminates when transmission finishes, and LED2 illuminates when receive finishes. LED3 illuminates if an error occurs.

Table 2.113 IIC Communication Specifications

Item	Description	Remarks
Communication mode	I^2C bus	
Data bit count	9 bits (including ACK)	
Wait between data and ACK	None	
ACK determination	Transfer halted when ACK = 1 received	
Pins used	P12/SCL0	
	P13/SDA0	
	P03/GPIO	LED0 output
	P05/GPIO	LED1 output
	P73/GPIO	LED2 output
	PJ5/GPIO	LED3 output

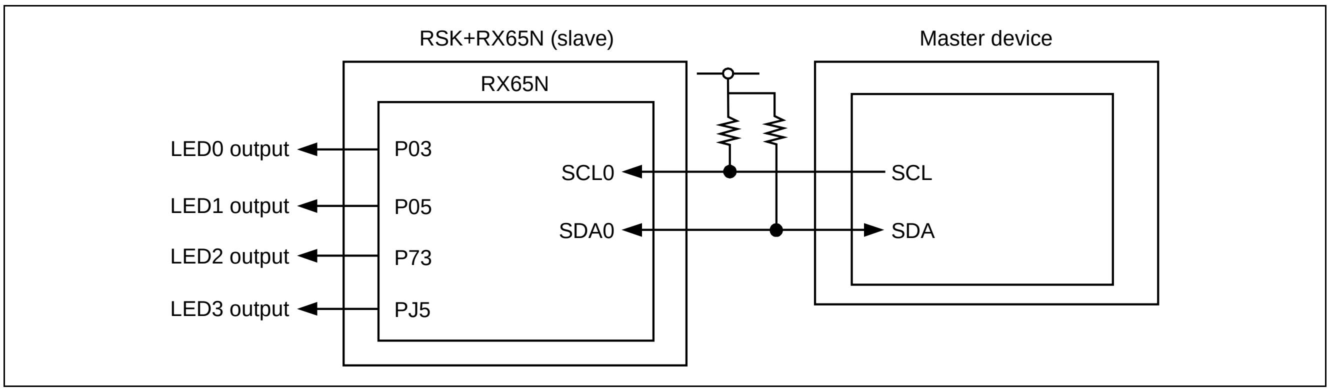


Figure 2.35 IIC Connection Specifications

< List of related registers >

Table 2.104 lists the interrupt-related registers of the RIIC0 on the SH7080 Group and RX651.

For an example of initial settings, refer to the initial settings example for master transmission/reception (Table 2.105). A setting example for slave transmit/receive is presented below.

Table 2.114 Example of Processing at Start of Transmission/Reception

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Match slave address.	Wait until ISCR.AAS = 1. Clear ISCR.AAS to 0 after reading it as 1.	—
2	Check transmission/reception mode.	If ICCR1.TRS = 1, transition to slave transmit mode. If ICCR1.TRS = 0, transition to slave receive mode.	—

Table 2.115 Example of Transfer End Processing

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Perform dummy read of receive data register to release SCL.	ICCR1.TRS = 0 Dummy data = ICDRR	Dummy data = ICDRR
2	Issue stop condition and wait for completion.	Wait until ICSR.STOP = 1.	Wait until ICSR2.STOP = 1.

3	End transfer.	Slave receive mode settings
		ICCR1.MST = 0 (slave mode)
		Clear ICSR.TDRE to 0 after reading it as 1.
		(transmit data register empty flag cleared)
			ICSR2.NACKF = 0
			(NACK detection flag cleared)
			ICSR2.STOP = 0
			(stop condition detection flag cleared)

Table 2.116 Example of Transmit Data Empty Interrupt Handling

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Transmit data.	ICDRT = transmit data	ICDRT = transmit data
2	If data is not last transmit data, end interrupt handling.	←	←

3	Make transmit end interrupt settings.	Clear ICIER.TIE to 0 and confirm that ICIER.TIE = 0.
		
		<Transmit end interrupt settings>
		ICIER.TEIE = 1
			ICU.IER06.IEN5 = 0 (TXI0)
			Clear ICIER.TIE to 0 and confirm that ICIER.TIE = 0.
			ICU.IR053 = 0
			<Transmit end interrupt settings>
			ICIER.TEIE = 1
			ICU.GENBL1.EN13 = 1 (TEI0)

Table 2.117 Example of Receive Data Full Interrupt Handling

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Confirm receive state.	If reception has not started, perform step 2 and end interrupt handling. If reception has already started, go to step 3.	If reception has not started, perform step 2 and end interrupt handling. If reception has already started, go to step 3.
2	Start reception.	ICIER.ACKBT = 0 (transmit 0 at acknowledge timing) Dummy data = ICDRR	ICMR3.ACKWP = 1 (writes to acknowledge bit enabled) ICMR3.ACKBT = 0 (send 0 to acknowledge bit) ICMR3.ACKWP = 0 (writes to acknowledge bit disabled) Dummy data = ICDRR
3	Check receive data count.	If remaining receive data count is 3 or more, perform processing of step 4, then end interrupt handling. If remaining receive data count is 2, skip processing of step 4 and perform processing of steps 5 and 6, then end interrupt handling. If remaining receive data count is 1, skip processing of steps 4 to 6 and perform processing of steps 7 and after.	If remaining receive data count is 3 or more, perform processing of step 4, then end interrupt handling. If remaining receive data count is 2, skip processing of step 4 and perform processing of steps 5 and 6, then end interrupt handling. If remaining receive data count is 1, skip processing of steps 4 to 6 and perform processing of steps 7 and after.
4	Receive data.	Receive data = ICDRR	Receive data = ICDRR

5	Make acknowledge bit settings.	ICIER.ACKBT = 1	ICMR3.ACKWP = 1
			(writes to acknowledge bit enabled)
			ICMR3.ACKBT = 1
			(transmit 1 as acknowledge)
			ICMR3.ACKWP = 0
			(writes to acknowledge bit disabled)

6	Receive last data unit – 1.	Receive data = ICDRR	Receive data = ICDRR
7	Receive last data.	Receive data = ICDRR	Receive data = ICDRR

8	End receive operation.	ICIER.ACKBT = 0	ICMR3.ACKWP = 1
			(writes to acknowledge bit enabled)
			ICMR3.ACKBT = 0
			(send 0 to acknowledge bit)
			ICMR3.ACKWP = 0
			(writes to acknowledge bit disabled)

The details of error handling are not stipulated in the sample code. However, on the RX651 the error interrupt is assigned to a group interrupt. It is therefore necessary to detect the interrupt flag from the group.

Table 2.118 Example of IIC Error Handling

Procedure		SH7080 Group Setting Example	RX651 Setting Example
1	Determine group interrupt.	—	If ICU.GRPBL1.IS16 (EEI2) = 1, perform processing for step 2 and after.
2	Determine arbitration lost.	—	If ICSR2.AL = 1, perform error handling.*1
3	Determine NACK detection.	If ICSR.NACKF = 1, perform error handling or transfer end processing.	If ICSR2.NACKF = 1, perform error handling or transfer end processing.
4	Determine timeout detection.	—	If ICSR2.TMOF = 1, perform error handling.*1
5	Stop condition detection.	If ICSR.STOP = 1, enter finished state.	If ICSR2.STOP = 1, enter finished state.

Note 1. The arbitration lost detection function and timeout detection function are disabled in the ICFER register.

2.13 A/D Converter (ADC)

2.13.1 Comparison of Specifications

A/D converter functionality is provided on the SH7080 Group by the ADC and on the RX651 by the 12-bit A/D converter (S12ADFa).

Table 2.119 is a comparative listing of the registers on the SH7080 Group and RX651.

Table 2.119 Comparison of SH7080 Group and RX651 Specifications (ADC)

Item	SH7080 Group (ADC)	RX651 (S12ADFa)

Number of input channels	16 channels
	(4 channels × 2, 8 channels × 1)
		Unit 0 for high-speed conversion (S12AD):
		8 channels
		Unit 1 for medium-speed conversion (S12AD1):
		21 channels + 1 extension

Clock source	Peripheral clock (Pϕ)	Peripheral module clock (PCLKD)
Resolution	10 bits	Max. 12 bits (selectable among 8, 10, and 12 bits)
A/D conversion method	Successive approximation	Successive approximation
Conversion speed	2.0 µs per channel (operating frequency: 25 MHz)	0.48 µs per channel (12-bit conversion mode, A/D converter clock: 60 MHz)
Conversion modes	• Single mode • Scan mode — Continuous scan mode — Single-cycle scan mode	• Single scan mode • Continuous scan mode • Group scan mode
A/D conversion start conditions	• Software trigger • Synchronous trigger (MTU2, MTU2S) • Asynchronous trigger (ADTRG pin)	• Software trigger • Synchronous trigger (MTU, TMR, TPU, ELC) • Asynchronous trigger (ADTRG0# pin, ADTRG1# pin)
Operations linked to A/D conversion-end interrupt	• CPU interrupt generation • DMAC or DTC activation	• CPU interrupt generation • DMAC or DTC activation
Conversion targets	• AN pin	• AN pin • Internal reference voltage (S12AD1) • Temperature sensor (S12AD1)
DTC/DMAC activation	DTC/DMAC activation supported	DTC/DMAC activation supported
Interrupt sources	• A/D conversion end	• A/D conversion end • Digital compare
Other	• Sample and hold function	• Event link • Sample and hold function (S12AD) • Variable sampling state count function • A/D converter self-diagnostic function • Selectable between A/D-converted value addition mode or average mode • Analog input disconnection detection assist function • Double trigger mode • 12-/10-/8-bit conversion switching • A/D data register auto-clear function • Extended analog input function • Comparison function (windows A and B)

2.13.2 Input Channels

On the SH7080 Group the ADC comprises three modules, each of which has either four or eight analog input channels. On the RX651 the S12ADFa comprises two units, S12AD and S12AD1, one with eight channels and the other with 21 channels. As on the SH7080 Group, on the RX651 each unit incorporates an A/D converter. Simultaneous operation is possible, but continuous scan operation spanning the two units is not supported.

Figure 2.36 compares the A/D converter configurations of the SH7080 Group and RX651.

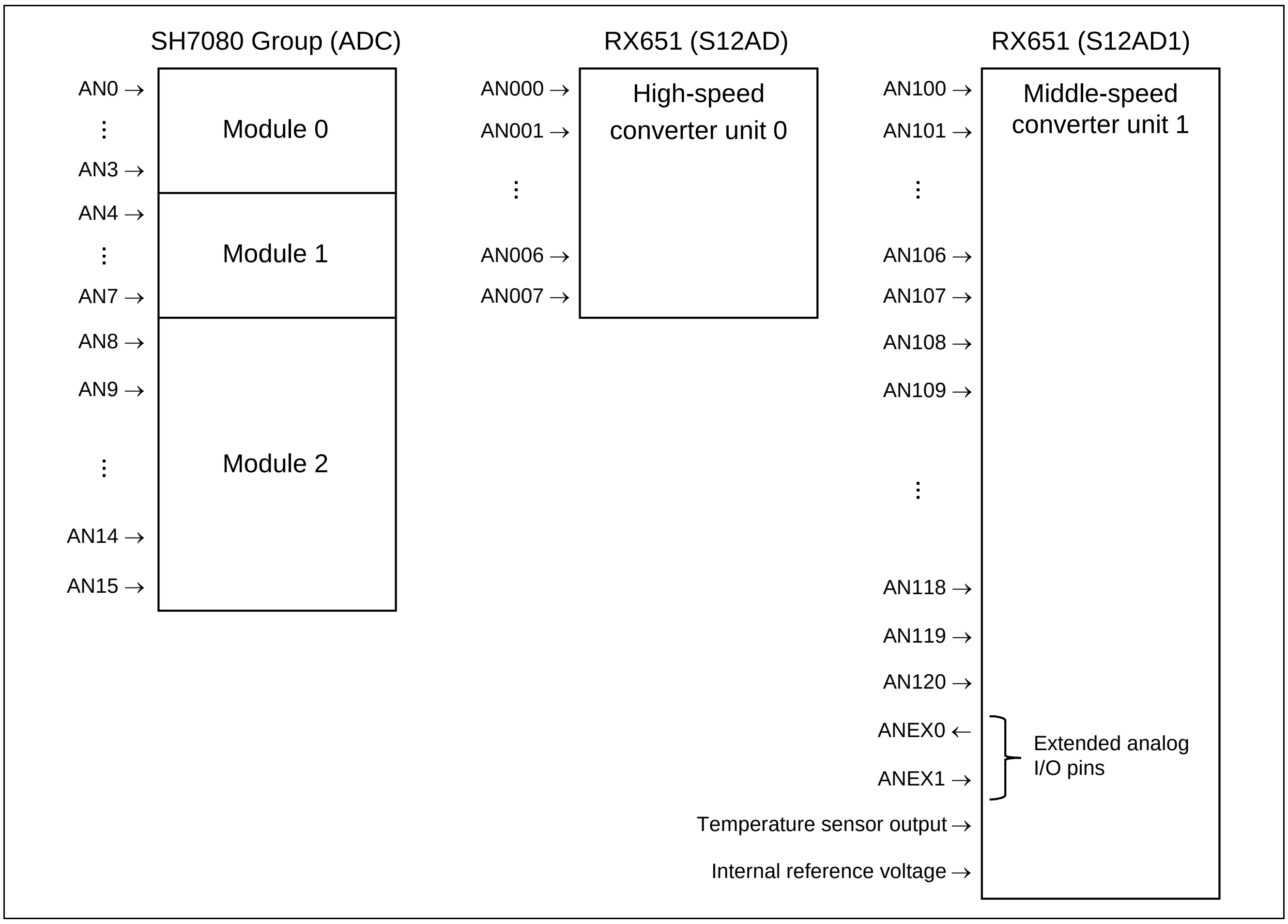


Figure 2.36 Comparison of A/D Converter Configurations

2.13.3 Scanning Sequence

Table 2.120 lists the scanning sequence when all channels are specified.

Table 2.120 A/D Converter Scanning Sequence

Microcontroller	A/D Converter	Conversion Sequence
SH7080 Group	ADC (module 0)	AN0 ⇒ AN1 ⇒ AN2 ⇒ AN3
	ADC (module 1)	AN4 ⇒ AN5 ⇒ AN6 ⇒ AN7
	ADC (module 2)	AN8 ⇒ AN9 ⇒ omitted ⇒ AN14 ⇒ AN15
RX651	S12AD	AN0 ⇒ AN1 ⇒ omitted ⇒ AN6 ⇒ AN7 ⇒ Temperature sensor output ⇒ Internal reference voltage Group scan: A ⇒ B ⇒ C
	S12AD1	AN100 ⇒ AN101 ⇒ omitted ⇒ AN119 ⇒ AN120 ⇒Temperature sensor output ⇒ Internal reference voltage Group scan: A ⇒ B ⇒ C

2.13.4 Operating Modes

Table 2.121 lists correspondences between the operating modes of the SH7080 Group and RX651, and Table 2.122 provides an overview of each operating mode.

Table 2.121 Correspondences between A/D Converter Operating Modes

SH7080 Group	RX651
Single mode	Single scan mode (only 1 channel specified)
Scan mode (Single-cycle scan)	Single scan mode (multiple channels specified)
Scan mode (Continuous scan)	Continuous scan mode
—	Group scan mode

Table 2.122 Overview of A/D Converter Operating Modes

Microcontroller	Operating Mode	Operational Overview
SH7080 Group	Single mode	A/D conversion occurs once on the specified channel only. After A/D conversion finishes, an interrupt is generated (if interrupts are enabled).
	Scan mode	A/D conversion of the analog input on multiple specified channels occurs. After A/D conversion on all the specified channels finishes, an interrupt is generated (if interrupts are enabled). • Single-cycle scan mode: A/D conversion occurs once on multiple channels. • Continuous scan mode: A/D conversion occurs continuously in sequence on multiple channels. A/D conversion restarts after an interrupt is generated.
RX651	Single scan mode	A/D conversion occurs once on the specified channel or channels. After A/D conversion on all the specified channels finishes, an interrupt is generated (if interrupts are enabled).
	Continuous scan mode	A/D conversion occurs repeatedly on the specified channel or channels until stopped by software. After A/D conversion on all the specified channels finishes, an interrupt is generated (if interrupts are enabled), and A/D conversion restarts.
	Group scan mode (groups A, B, and C)	When the specified synchronous trigger occurs, A/D conversion occurs once on the specified channels in each group. After A/D conversion of each group finishes, an interrupt is generated (if interrupts are enabled). The following channels can be used in group scan mode: • Unit 0: AN000 to AN007 • Unit 1: AN100 to AN120

2.13.5 Interrupts

On the SH7080 Group the A/D_0 and A/D_2 conversion end interrupts can be used to activate the DTC, but on the RX651 all conversion end interrupts can activate the DTC and the DMAC.

On the RX651 the S12ADFa interrupts are assigned to group interrupt BL1 and to software configurable interrupt B. The group BL1 interrupt status flag (GRPBL1.ISn) is cleared automatically when the corresponding bit in the module's status register is cleared. The software configurable interrupt B status flag (PIBRk.PIRn) is not cleared automatically, but there is no effect on the generation of interrupt requests.

Refer to 1.8 for information about interrupts.

2.13.6 Module Stop

As on the SH7080 Group, the S12ADFa of the RX651 is set to the module-stop state after a reset, and no clock is supplied.

Refer to 2.16 for information on the module-stop state.

2.13.7 Continuous Scan Mode Setting Example

An example is presented below of settings for continuous scan mode on the A/D converter of the SH7080 Group and the 12-bit A/D converter of the RX651. In addition, Table 2.125 lists the differences with the settings for the other operating modes of the SH7080 Group.

< Specifications >

1. Unit 0 of the 12-bit A/D converter on the RSK+RX65N board is used.
2. The A/D conversion start timing is determined by a software trigger.
3. Analog input is accepted on three channels (AN001, AN002, and AN003), and the operating mode is continuous scan mode. The conversion result is stored in the RAM by the handler of the S12ADI interrupt, which is generated when conversion finishes.

Table 2.123 12-Bit A/D Converter Setting Specifications

Item	Description	Remarks
Channel used	AN001, AN002, AN003	
Interrupt handling	A/D conversion-end interrupt (S12ADI interrupt)	
Operating mode	Continuous scan mode	
Conversion start trigger and cycle	Software trigger (repeating conversion after start)	
Extended analog input	Not used	
Data alignment	Flush-left	AN001: ADDR1
		AN002: ADDR2
		AN003: ADDR3
Pins used	P41/AN001	Analog input 1
	P42/AN002	Analog input 2
	P43/AN003	Analog input 3

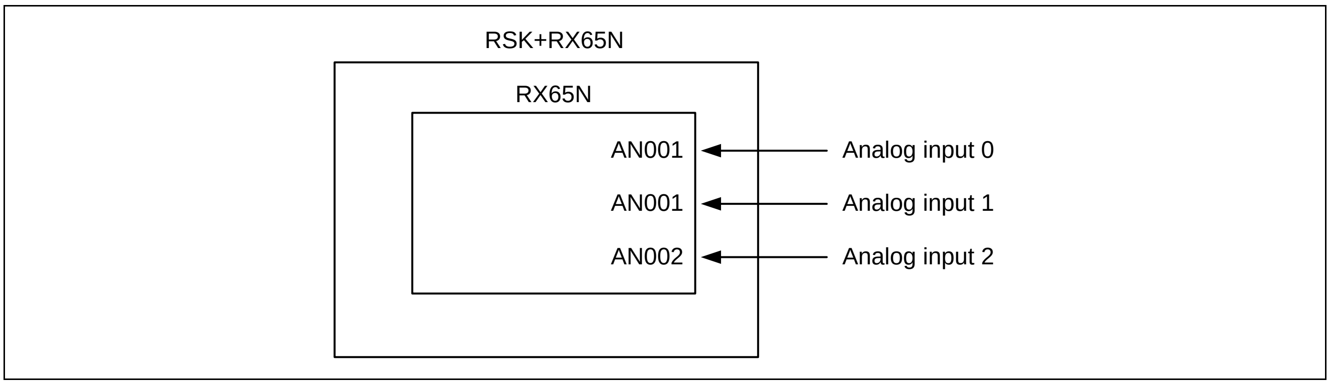


Figure 2.37 12-Bit A/D Converter Setting Connection Specifications

An initial setting example of A/D converter is shown below.

Table 2.124 A/D Converter Initial Setting Example

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKC (Peripheral Clock C): 60 MHz

1	Cancel module-stop state.	STB.STBCR4.MSTP16 = 0	SYSTEM.PRCR = A502h
			SYSTEM.MSTPCRA.MSTPA17 = 0
			SYSTEM.PRCR = A500h

2	Disable interrupts.	ADCSR.ADIE = 0 (interrupts disabled)	ICU.IER10.IEN0 = 0
			(vector 128: software configurable interrupt B)
			S12AD.ADCSR.ADIE = 0
			(interrupts disabled)

3	Stop A/D conversion.	ADCR.ADST = 0 (A/D0 stopped)	S12AD.ADCSR.ADST = 0 (A/D0 stopped)

4	Make I/O port settings (pin I/O and pin function settings).	No I/O settings needed for AN0, AN1, and AN2 because they are assigned as input-only ports (PFDR).	Make settings in MPC for AN001, AN002, and AN003.
			PORT4.PDR.B1 = 0 (set to input)
			PORT4.PDR.B2 = 0 (set to input)
			PORT4.PDR.B3 = 0 (set to input)
			PORT4.PMR.B1 = 0 (GPIO)
			PORT4.PMR.B2 = 0 (GPIO)
			PORT4.PMR.B3 = 0 (GPIO)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enabled)
			MPC.P41PFS = 80h (analog function)
			MPC.P42PFS = 80h (analog function)
			MPC.P43PFS = 80h (analog function)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1

5	Make operating mode, input channel, and conversion time settings.	ADCSR.ADCS = 1
		(continuous scan mode)
		ADCSR.ADM[1:0] = 01b
		(4-channel scan mode)
		ADCSR.CH[2:0] = 010b (AN0 to AN2)
		ADCSR.STC = 0
		ADCSR.CKSL[1:0] = 10b
		(A/D conversion time: 100 states)
			S12AD.ADCSR.ADCS[1:0] = 10b
			(continuous scan mode)
			S12AD.ADANSA0 = 0Eh
			(AN001, AN002, and AN003 subject to change)
			S12AD.ADSSTR1 = 150
			S12AD.ADSSTR2 = 150
			S12AD.ADSSTR3 = 150
			(A/D conversion processing sampling time: 150 cycles (2.5 μs))

6	ADDR format	Only flush-left supported, so no setting needed.	S12AD.ADCER.ADRFMT = 1 (data flush-left) S12AD.ADCER.ADPRC[1:0] = 01b (10-bit precision)
7	Make software configurable interrupt source settings.	—	ICU.SLIBXR128 = 64 (interrupt source number 64: S12ADI) ICU.SLIPRCR.WPRC = 1*[1] (software configurable interrupt source select register write protection) Confirm that ICU.SLIPRCR.WPRC = 1.
8	Make interrupt priority register setting.	INTC.IPRK.WORD = 5000h (A/D0 and A/D1: level 5)	ICU.IPR128 = 5 (S12ADI: level 5)

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKC (Peripheral Clock C): 60 MHz
9	Clear interrupt request.	—	ICU.IR128 = 0 (S12ADI)

10	Enable interrupts.	ADCSR.ADIE = 1 (interrupts enabled)	S12AD.ADCSR.ADIE = 1
			(interrupts enabled)
			ICU.IER10.IEN0 = 1
			(vector 128: software configurable interrupt B)

11	Start A/D conversion.	ADCR.ADST = 1 (A/D start)	S12AD.ADCSR.ADST = 1 (A/D start)
12	Perform handling of A/D conversion-end interrupt.	ADCSR.ADF = 0 • Read interrupt flag and clear to 0.	The interrupt flag is cleared automatically.

Note 1. Once ICU.SLIPRCR.WPRC is set to 1 it cannot be cleared to 0 by software.

I/O port settings and the operating mode can be selected or changed by changing the values in the shaded portions.

Table 2.125 lists the corresponding settings on the RX651 to match the operating modes on the SH7080 Group.

The various modes can be selected by changing the values in the shaded portions of Table 2.124 to those listed below.

Table 2.125 Example of Settings Corresponding to A/D Converter Operating Modes (SH7080 Group to RX651)

No.	SH7080 Group Setting Example	RX651 Setting Example
1	Single mode	Single scan mode (individual channel)

	ADCSR_0.ADCS = 0 (single-cycle scan mode)
	ADCSR_0.ADM[1:0] = 00b (single mode)
	ADCSR_0.CH[2:0] = 000b (AN0)
		PORT4.PDR.B1 = 0 (set to input)
		PORT4.PMR.B1 = 0 (GPIO)
		MPC.P41PFS = 80h (analog function)
		S12AD.ADCSR.ADCS[1:0] = 00b (single scan mode)
		S12AD.ADANSA0 = 02h (AN001 subject to change)

2	Single-cycle scan mode	Single scan mode (multiple channels)

	ADCSR_0.ADCS = 0 (single-cycle scan mode)
	ADCSR_0.ADM[1:0] = 01b (4-channel scan mode)
	ADCSR_0.CH[2:0] = 010b (AN0 to AN2)
		PORT4.PDR.B1 = 0 (set to input)
		PORT4.PDR.B2 = 0 (set to input)
		PORT4.PDR.B3 = 0 (set to input)
		PORT4.PMR.B1 = 0 (GPIO)
		PORT4.PMR.B2 = 0 (GPIO)
		PORT4.PMR.B3 = 0 (GPIO)
		MPC.P41PFS = 80h (analog function)
		MPC.P42PFS = 80h (analog function)
		MPC.P43PFS = 80h (analog function)
		S12AD.ADCSR.ADCS[1:0] = 00b (single scan mode)
		S12AD.ADANSA0 = 0Eh
		(AN001, AN002, and AN003 subject to change)

3	Group scan mode	Group scan mode

	ADCSR_0.ADCS = 0 (single-cycle scan mode)
	ADCSR_0.ADM[1:0] = 11b (2-channel scan mode)
	ADCSR_0.CH[2:0] = 000b
	(group 0: AN0, group 1: AN2)
	ADTSR_0.TRG0S[3:0] = 0011b
	(MTU2 A/D conversion start request delayed (TRG4AN))
	ADTSR_0.TRG01S[3:0] = 0100b
	(MTU2 A/D conversion start request delayed (TRG4BN))
		PORT4.PDR.B1 = 0 (set to input)
		PORT4.PDR.B2 = 0 (set to input)
		PORT4.PMR.B1 = 0 (GPIO)
		PORT4.PMR.B2 = 0 (GPIO)
		MPC.P41PFS = 80h (analog function)
		MPC.P42PFS = 80h (analog function)
		S12AD.ADCSR.ADCS[1:0] = 01b (group scan mode)
		S12AD.ADANSA0 = 02h
		(group A: AN001 subject to change)
		S12AD.ADANSB0 = 04h
		(group B: AN002 subject to change)
		S12AD.ADSTRGR.TRSA[5:0] = 001001b
		(compare match of MTU4.TADCORA and MTU4.TCNT)
		S12AD.ADSTRGR.TRSB[5:0] = 001010b
		(compare match of MTU4.TADCORB and MTU4.TCNT)

Note that software triggers are not used in group scan mode.

In order to use the A/D conversion start request delayed function, MTU settings are required in addition to A/D conversion start trigger settings.

2.14 Compare Match Timer (CMT)

2.14.1 Comparison of Specifications

Compare match timer functionality is provided on the SH7080 Group by the CMT and on the RX651 by the CMT, which has a 16-bit timer, and the CMTW, which has a 32-bit timer.

The RX651 includes all the CMT functionality of the SH7080 Group (backward compatibility). Table 2.126 provides a comparative listing of the specifications of the SH7080 Group and RX651.

Table 2.126 Comparison of SH7080 Group and RX651 Specifications (CMT)

	SH7080 Group	RX651	
Item	**CMT**	**CMT**	**CMTW**
Number of units (channels)	1 unit (total 2 channels)	2 units (total 4 channels)	2 units (total 2 channels)
Clock source	Internal clock (Pϕ)	Peripheral module clock (PCLKB)	Peripheral module clock (PCLKB)
Clock frequency division ratio	Pϕ/8, 32, 128, 512	PCLKB/8, 32, 128, 512	PCLKB/8, 32, 128, 512
Count operation	16-bit up-counter	16-bit up-counter	Max. 32-bit up-counter (selectable between 16 and 32 bits)
DTC/DMAC activation	DTC activation supported	DTC/DMAC activation supported	DTC/DMAC activation supported
Interrupt sources	• Compare match	• Compare match	• Compare match • Input compare • Output compare
Other	—	Event link	Event link

2.14.2 Register Comparison

The CMT of the RX651 does not have interrupt flags, but equivalent processing can be accomplished by using the interrupt controller.

Table 2.127 and Table 2.128 are a comparative listing of the registers on the SH7080 Group and RX651.

Guide to Symbols in "Changes" Column of Table

◎: Register with same bit assignments on SH7080 Group and RX651

△: Register with different bit assignments on SH7080 Group and RX651

—: Register not present on SH7080 Group or RX651

Table 2.127 SH7080 Group and RX651 Register Comparison (CMT)

SH7080 Group (CMT)	RX651 (CMT)	Changes
CMT n: 0 or 1	CMT m: 0 to 3	
Compare match timer start register (CMSTR)	Compare match timer start register 0 (CMSTR0) Compare match timer start register 1 (CMSTR1)	◎
Compare match timer control/ status register n (CMCSR_n)	Compare match timer control register (CMTm.CMCR)	△
Compare match counter n (CMCNT_n)	Compare match timer counter (CMTm.CMCNT)	◎
Compare match constant register n (CMCOR_n)	Compare match constant register (CMTm.CMCOR)	◎

Table 2.128 SH7080 Group and RX651 Register Comparison (CMTW)

SH7080 Group (CMT)	RX651 (CMTW)	Changes
CMT n: 0 or 1	CMTW m: 0 or 1	
Compare match timer start register (CMSTR)	Timer start register (CMTWm.CMWSTR)	◎
Compare match timer control/ status register n (CMCSR_n)	Timer control register (CMTWm.CMWCR)	□
Compare match counter n (CMCNT_n)	Timer counter (CMTWm.CMWCNT)	□
Compare match constant register n (CMCOR_n)	Compare match constant register (CMTWm.CMWCOR)	□
—	Timer I/O control register (CMTWm.CMWIOR)	—
	Input capture registers 0 and 1 (CMTWm.CMWICR0 and CMTWm.CMWICR1)	
	Output compare registers 0 and 1 (CMTWm.CMWOCR0 and CMTWm.CMWOCR1)	

2.14.3 Interrupts

On the SH7080 Group the CMT can be used to activate the DTC, but on the RX651 it can activate the DTC and the DMAC.

On the RX651 some of the CMT and CMTW interrupts are assigned to software configurable interrupt B. The interrupt controller’s interrupt status flag (IRn.IR) is cleared automatically when the corresponding interrupt is accepted. The software configurable interrupt B status flag (PIBRk.PIRn) is not cleared automatically, but there is no effect on the generation of interrupt requests.

Refer to 1.8 for information about interrupts.

2.14.4 Module Stop

As on the SH7080 Group, the CMT of the RX651 is set to the module-stop state after a reset, and no clock is supplied.

Refer to 2.16 for information on the module-stop state.

2.14.5 Compare Match Timer Setting Example

A compare match timer setting example comparing the SH7080 Group and RX651 is shown below.

< Specifications >

1. CMT unit 0, channel 1 on the RSK+RX65N board is used.
2. The compare match interrupt (CMI1) is used to turn LED1 one and off in 0.5-second cycles.

Table 2.129 Compare Match Timer Setting Specifications

Item	Description	Remarks
Count clock	PCLKB/512	PCLKB = 60 MHz
Counter value (CMCOR)	0xE4E1	Set to the value calculated using the period (0.5 seconds) and count clock (60 MHz/512), minus 1
Other	P05/GPIO	LED1 output

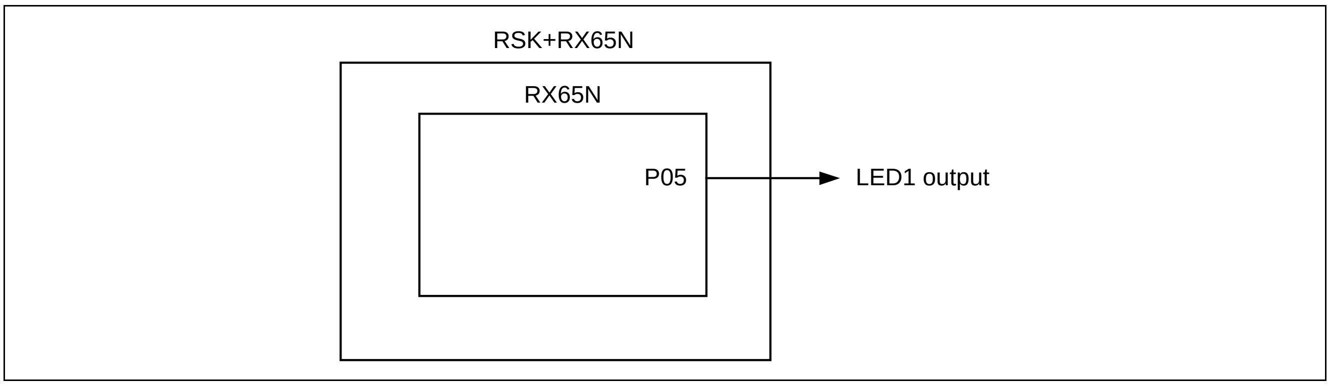
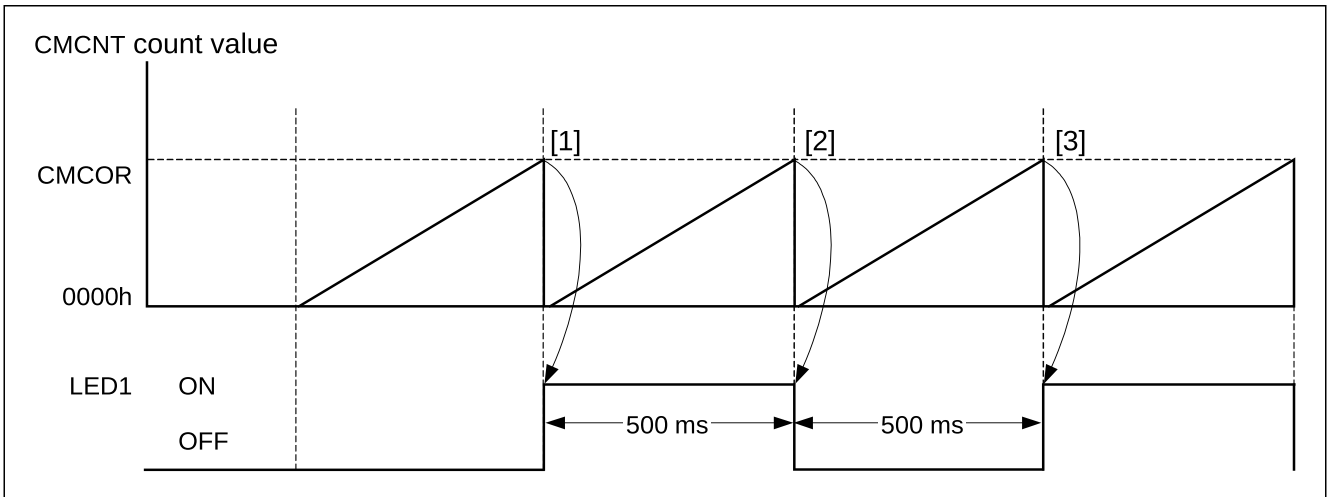


Figure 2.38 Compare Match Timer Connection Specifications



[1] LED1 turns on at a compare match interrupt.
[2] LED1 turns off at a compare match interrupt.
[3] Afterward, [1] and [2] are repeated multiple times.

Figure 2.39 Compare Match Timer Operation Example

Table 2.130 Compare Match Timer Initial Setting Example

Procedure		SH7080 Group Setting Example Pϕ (Peripheral Clock): 40 MHz	RX651 Setting Example PCLKB (Peripheral Clock B): 60 MHz

1	Cancel module-stop state.	STB.STBCR4.MSTP21 = 0	SYSTEM.PRCR = A502h
			SYSTEM.MSTPCRA.MSTPA15 = 0
			SYSTEM.PRCR = A500h

2	Disable interrupts.	CMCSR.CMIE = 1
		(compare match interrupt disabled)
			ICU.IER03.IEN5 = 0
			(vector 29, CMI1 disabled)
			CMT1.CMCR.CMIE = 0
			(compare match interrupt disabled)

3	Disable timer.	CMSTR.STR1 = 0	CMSTR0.STR1 = 0
4	Select counter clock.	CMCSR.CKS[1:0] = 11b (Pϕ/512)	CMT1.CMCR.CKS[1:0] = 11b (PCLKB/512)
5	Clear timer counter.	CMCNT = 0000h (counter cleared)	CMT1.CMCNT = 0000h (counter cleared)
6	Set compare match cycle.	CMCOR = 9896h	CMT1.CMCOR = E4E1h
7	Enable interrupts.	INTC.IPRJ.WORD = 0x0500 (interrupt priority: 5)	ICU.IPR005 = 05h (CMI1 interrupt priority: 5)
8	Clear interrupt request.	—	ICU.IR029 = 0 (CMI1 interrupt flag cleared)

9	Enable interrupts.	CMCSR.CMIE = 1
		(compare match enabled)
			CMT1.CMCR.CMIE = 1
			(compare match enabled)
			ICU.IER03.IEN5 = 1
			(vector 29, CMI1 enabled)

10	Enable timer operation.	CMSTR.STR1 = 1 (start timer)	CMSTR0.STR1 = 1 (start timer)
11	Interrupt handler (clear flag.)	CMCSR.CMF = 0 (after reading CMCSR, CMF = 0)	The interrupt flag is cleared automatically.

2.15 Flash Memory

2.15.1 Comparison of Specifications

Table 2.131 is a comparative listing of the specifications of the SH7080 Group and RX651.

Table 2.131 Comparison of SH7080 Group and RX651 Specifications (Flash Memory)

Item	SH7080 Group	RX651
Size	User MAT: 512 KB or 256 KB User boot MAT: 12 KB	User area: Max. 1 MB
Block size × block count	512 KB product — 64 KB × 7 blocks (448 KB) — 32 KB × 1 block (32 KB) — 4K B × 8 blocks (32 KB) 256 KB product — 64 KB × 3 blocks (192 KB) — 32 KB × 1 block (32 KB) — 4 KB × 8 blocks (32 KB)	• 8 KB × 8 blocks (64 KB) • 32 KB × 2 blocks (64 KB): TM target area • 32 KB × 28 blocks (896 KB)
Programming command	Dedicated write/erase program integrated on-chip	Self-programming using FACI commands
Write unit	128 bytes	128 bytes
Erase unit	Block	Block
Write count	500 times	10,000 times
Programming modes	On-board programming • Boot mode • User programming mode • User boot mode Off-board programming • Writer mode	On-board programming • Boot mode (SCI interface) • Boot mode (USB interface) • Boot mode (FINE interface) • Programming by a routine for code flash memory programming within a user program Off-board programming • Parallel writer mode
Other	• Automatic bit rate adjustment • RAM-based flash memory emulation function • Protect mode	• Security function (prevention of unauthorized modification/ unauthorized reads) • Protection function (prevention of accidental overwriting of data) • TM function (prevention of unauthorized reads) • Ability to select startup area • Suspend/resume function • Automatic bit rate adjustment

On the RX651 FACI commands can be used to program the code flash memory. Refer to the following application note for details:

RX65N Group, RX651 Group Flash Memory User's Manual: Hardware Interface (R01UH0602EJ)

2.16 Low Power Consumption Function

2.16.1 Comparison of Mode Specifications

Table 2.132 and Table 2.133 summarize the methods for transitioning to and canceling the various low-power states on the SH7080 Group and RX651, and list the operating states of the clock, CPU, and on-chip modules.

Table 2.132 SH7080 Group Low-Power States

Transition and Cancelation Methods, and Operating States	Sleep Mode	Module Standby Function	Software Standby Mode	Deep Software Standby Mode
Transition method	Control register + instruction	Control register	Control register + instruction	Control register + instruction
Cancelation method other than reset	—	Control register	Interrupt	—
Clock	Operating	Operating	Stopped	Stopped
CPU	Stopped	Operating	Stopped	Stopped
On-chip peripheral modules	Operating	Specified modules stopped	Stopped	Stopped

Table 2.133 RX651 Low-Power States

Transition and Cancelation Methods, and Operating States	Sleep Mode	All-Module Clock Stop Mode	Software Standby Mode	Deep Software Standby Mode
Transition method	Control register + instruction	Control register + instruction	Control register + instruction	Control register + instruction
Cancelation method other than reset	Interrupt	Interrupt	Interrupt	Interrupt
Main clock oscillator, sub-clock oscillator	Operation possible	Operation possible	Operation possible	Operation possible
High-speed on-chip oscillator, low-speed on-chip oscillator	Operation possible	Operation possible	Stopped	Stopped
IWDT dedicated on-chip oscillator	Operation possible	Operation possible	Operation possible	Stopped (settings undetermined)
PLL	Operation possible	Operation possible	Stopped	Stopped
CPU	Stopped (settings retained)	Stopped (settings retained)	Stopped (settings retained)	Stopped (settings undetermined)
RAM	Operation possible (settings retained)	Stopped (settings retained)	Stopped (settings retained)	Stopped (settings undetermined)
Standby RAM	Operation possible (settings retained)	Stopped (settings retained)	Stopped (settings retained)	Stopped (settings retained/ undetermined)*^{1}
Flash memory	Operation possible	Stopped (settings retained)	Stopped (settings retained)	Stopped (settings retained)
USBFS Host/Function module (USBb)	Operation possible	Stopped	Stopped	Stopped (settings retained/ undetermined)*^{1}
Watchdog timer (WDT)	Stopped (settings retained)	Stopped (settings retained)	Stopped (settings retained)	Stopped (settings undetermined)

Transition and Cancelation Methods, and Operating States	Sleep Mode	All-Module Clock Stop Mode	Software Standby Mode	Deep Software Standby Mode
Independent watchdog timer (IWDT)	Operation possible	Operation possible	Operation possible	Stopped (settings undetermined)
Realtime clock (RTC)	Operation possible	Operation possible	Operation possible	Operation possible
8-bit timer (TMR)	Operation possible	Operation possible	Stopped (settings retained)	Stopped (settings undetermined)
Voltage detection circuit (LVD)	Operation possible	Operation possible	Operation possible	Operation possible
Power-on reset circuit	Operation possible	Operation possible	Operation possible	Operation possible
Peripheral modules	Operation possible	Stopped (settings retained)	Stopped (settings retained)	Stopped (settings undetermined)
I/O ports	Operation possible	Settings retained	Settings retained	Settings retained

Stopped (settings retained):

State in which the values of the internal registers are retained and the internal state is operation suspended.

Stopped (settings undetermined):

State in which the values of the internal registers are undetermined and the internal state is power-off.

Note 1. Either “settings retained” or “settings undetermined” may be selected by means of a control register setting

2.16.2 Mode Transitions

Figure 2.40 diagrams the transitions between the modes of the RX651, and Table 2.134 lists transition conditions.

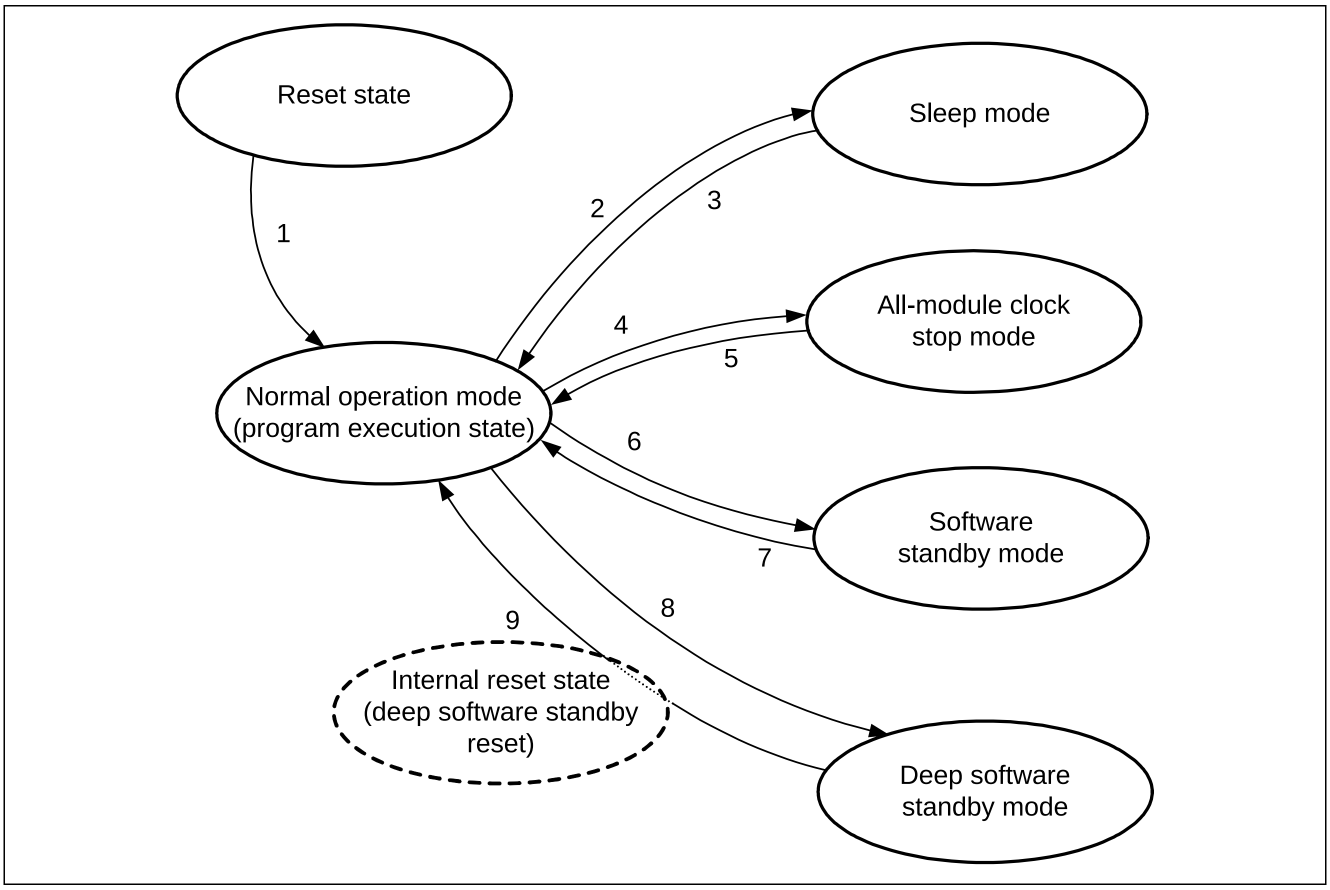

Reset state
Sleep mode
1
2
3
All-module clock stop mode
4
5
Normal operation mode (program execution state)
6
Software standby mode
7
9
8
Internal reset state (deep software standby reset)
Deep software standby mode

Figure 2.40 RX651 Mode Transitions

Table 2.134 List of RX651 Mode Transitions and Events

No.	Event	Transition Condition (The following conditions are specified before the event.)
1	RES# pin = high	—
2	WAIT instruction executed	SBYCR.SSBY = 0
3	All interrupts	—

4	WAIT instruction executed	SBYCR.SSBY = “0”, MSTPCRA.ACSE = “1”, MSTPCRA = “FFFF FF[C-F]Fh”
		MSTPCRB = “FFFF FFFFh”, MSTPCRC[31:16] = “FFFFh”, MSTPCRD = “FFFF FFFFh”

5	External and peripheral interrupts	External pin interrupts (NMI, IRQ0 to IRQ15) Peripheral function interrupts (8-bit timer, RTC alarm, RTC cycle, IWDT, USB suspend/resume, voltage monitor 1, voltage monitor 2, oscillation stop detection)*1
6	WAIT instruction executed	SBYCR.SSBY = 1, DPSBYCR.DPSBY = 0
7	External and peripheral interrupts	External pin interrupts (NMI, IRQ0 to IRQ15) Peripheral function interrupts (RTC alarm, RTC cycle, IWDT, USB suspend/resume, voltage monitor 1, voltage monitor 2)*1
8	WAIT instruction executed	SBYCR.SSBY = 1, DPSBYCR.DPSBY = 1
9	External and peripheral interrupts	Some pins used as external pin interrupt sources (NMI, IRQ0-DS to IRQ15-DS, SCL2-DS, SDA2-DS, CRX1-DS), peripheral function interrupts (RTC alarm, RTC cycle, USB suspend/resume, voltage monitor 1, voltage monitor 2)*1 After one of the above interrupts occurs the internal reset state lasts for a specified duration, after which the internal reset and deep software standby mode are canceled at the same time, and the CPU operates in normal operation mode using the LOCO (recovery after a reset).

Note 1. Each interrupt has detailed conditions.
For descriptions, see the User’s Manual: Hardware.

2.16.3 Module-Stop State

On the SH7080 Group the modules other than the RAM and ROM are set to the module-stop state after a reset, and no clock is supplied.

On the RX651 the modules other than the DMACAa, EXDMACa, DTCb, and RAM are set to the module-stop state after a reset, and no clock is supplied. The DTCb and DMACAa share a common module-stop setting bit (MSTPCRA.MSTPA28), so module-stop control is applied to them both simultaneously. The EXDMACa has an independent module-stop setting bit (MSTPCRA.MSTPA29), so it can be controlled individually.

As on the SH7080 Group, on the RX651 it is necessary to cancel the module-stop state before using any module that enters the module-stop state after a reset.

When changing module-stop state settings on the RX651 it is necessary to turn off register write protection in the protect register (PRCR) before accessing the module-stop control register (MSTPCRn).

Table 2.135 lists the clock supply state after a reset of each module.

Table 2.135 Clock Supply State after Reset

Function Name*1	**SH7080 Group**	**RX651***2
RAM	Clock supplied (operating)	Clock supplied (operating)
User break controller (UBC)	No clock supplied	—
Data transfer controller (DTC)		Clock supplied (operating)
Direct memory access controller (DMAC)		
Multi-function timer pulse unit (MTU)		No clock supplied
Serial communication interface (SCI, SCIF)		
Synchronous serial communication (SSU)		
I^{2}C bus interface (IIC)		
A/D converter (ADC)		
Compare match timer (CMT)		

Note 1. The function names listed are those for the SH7080 Group.

Note 2. On the RX651 there are other modules affected by the module-stop function in addition to those listed in the table.

2.16.4 Write Protection

The RX651 has a register write protection function to protect important registers from being overwritten if program runaway occurs. The low power consumption function-related registers are protected by this function.

If necessary, set protect bit 1 (PRCR.PRC1) to 1 to enable writes before writing to these registers.

2.16.5 Mode Transition Setting Example

A mode transition setting example using the RX651 is shown below.

< Specifications >

1. The RSK+RX65N board is used.
2. After a reset, settings are made to enable input from SW3 (IRQ11-DS), to wait for SW3 to be pressed, and to implement all of the mode transitions listed below by pressing SW3.
3. The MTU4 (compare match A) and TMR compare match pin output is monitored to confirm the mode transitions. (TMR stands for TMR0 and TMR1, and is used as a 16-bit timer.) Note that TMR is set to operate even after the transition to all-module clock stop mode.

Notes

SW3 and the IRQ11-DS pin are not connected on the RSK+RX65N. In addition, it is necessary to make a connection to pin 13 of JA2 in order to check TMO0 pin output. Therefore, the following modifications must be made to the RSK+RX65N in order to perform debugging using the sample code.

(Modifications)

- Connect pin 8 of JA1 (which is connected to SW3) to pin 12 of JA1 (which is connected to the IRQ11-DS pin).
- Remove R278 (0 Ω resistor) from the board, and mount R186 (0 Ω resistor) on the board.

Table 2.136 lists the mode transitions and the operation of the modules.

Table 2.136 RX651 Mode Transition Setting and Operation Specifications

No.	SW3 Pressed	State Transition	LED2 (GPIO)	LED3 (GPIO)	MTIOC4A Pin	TMO0 Pin
1	—	RES pin ⇒ normal operation mode	Flashing	Off	Toggled output	Toggled output
2	1st	SLEEP mode	Sustained		Toggled output	Toggled output
3	2nd	Normal operation mode	Flashing		Toggled output	Toggled output
4	3rd	All-module clock stop mode	Sustained		Stop sustained	Toggled output
5	4th	Normal operation mode	Flashing		Toggled output	Toggled output
6	5th	Software standby mode	Sustained		Stop sustained	Stop sustained
7	6th	Normal operation mode	Flashing		Toggled output	Toggled output
8	7th	Deep software standby mode	Undefined		Stop undefined	Stop undefined
9	8th	Deep software standby mode ⇒ normal operation mode	Off	On	Stop	Stop

Note: When returning to normal mode, MTU and TMR are both initialized.

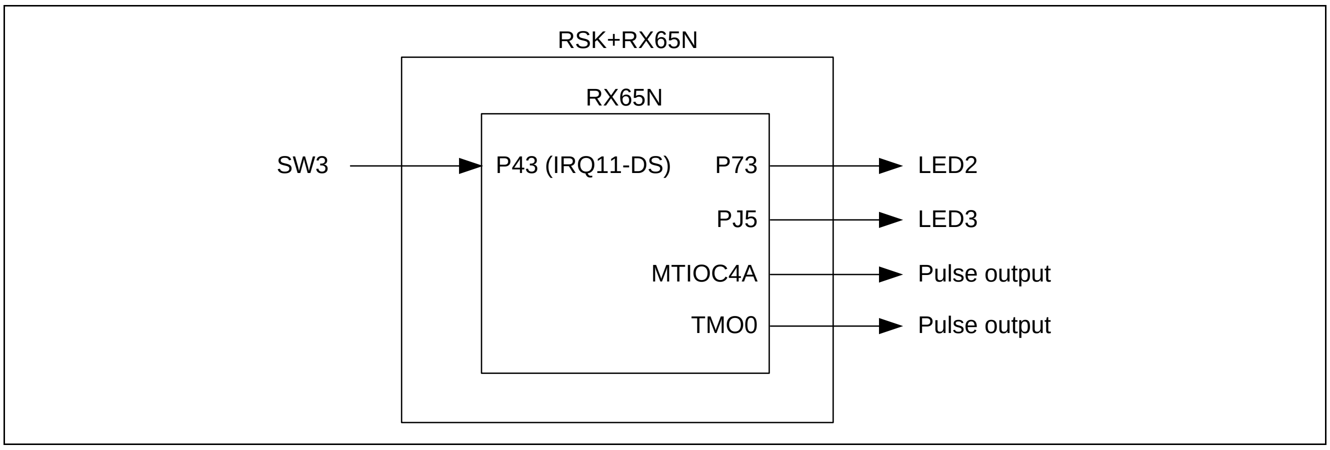


Figure 2.41 Mode Transition Setting Connection Specifications

Table 2.137 Setting Specifications

Item	Description	Remarks
CPU		
Processor mode	Supervisor mode	
TMR0, TMR1		
Count clock	PCLKB/1	PCLKB = 60 MHz
Operating mode	16-bit counter mode (TMR0 and TMR1 used in cascade connection)	
Counter clear setting	Cleared by compare match A	
Interrupts	Compare match A and B disabled Overflow interrupt disabled (Also disabled by interrupt controller)	
TCORA setting value	EA5Fh (1 ms period)	TMR0 + TMR1
Output selection	Inverted output	
Pins used	P22/TMO0	Pulse output
SW3 (IRQ11-DS)		
SW3 (IRQ11-DS)	Used as mode transition trigger switch P43/IRQ11-DS	
Interrupt priority	Level 15	
Digital noise filter	Used*[1]	
Return from deep software standby	The SW3 signal is used as the deep software standby cancel signal, so connect it to P43.*[2]	
LED		
LED2	Flashing during SW3 press wait duration (normal state).	P73
LED3	Turns on at return from deep software standby.	PJ5
Pins used by MTU4		
Compare match A output pin	PA0/MTIOC4A	Pulse output

Note 1. The digital noise filter is used when transitioning from normal operation mode to any of the low-power modes. The digital noise filter is not used when returning.

Note 2. SW3 is not connected to P43 (IRQ11-DS) by default.

Figure 2.42 is a flowchart of mode transition processing.

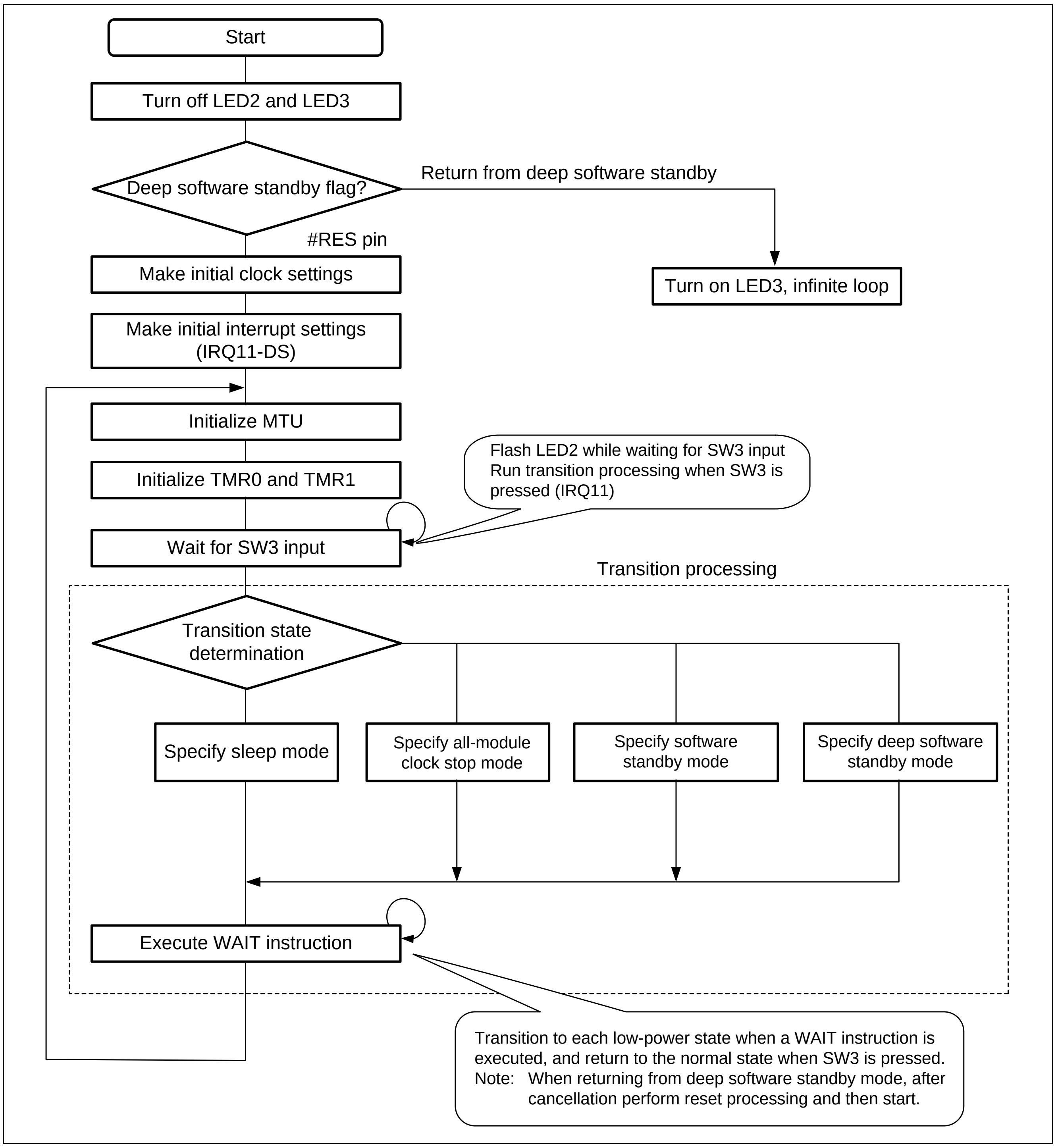

Start
Turn off LED2 and LED3
Deep software standby flag?
Return from deep software standby
#RES pin
Turn on LED3, infinite loop
Make initial clock settings
Make initial interrupt settings (IRQ11-DS)
Initialize MTU
Initialize TMR0 and TMR1
Flash LED2 while waiting for SW3 input Run transition processing when SW3 is pressed (IRQ11)
Wait for SW3 input
Transition processing
Transition state determination
Specify sleep mode
Specify all-module clock stop mode
Specify software standby mode
Specify deep software standby mode
Execute WAIT instruction
Transition to each low-power state when a WAIT instruction is executed, and return to the normal state when SW3 is pressed.
Note: When returning from deep software standby mode, after cancellation perform reset processing and then start.

Figure 2.42 Mode Transition Processing Flowchart

Settings associated with mode transitions are listed below.

Refer to Table 2.37, MTU Output Compare Match Initial Setting Example for details of MTU4 settings.

Table 2.138 LED2 and LED3 Settings (Initially Off)

Procedure		Setting Example

1	Make GPIO settings (LED2 and LED3 off).	PORT7.PODR.B3 = 1 (LED2 off)
		PORT7.PDR.B3 = 1 (output)
		PORT7.PMR.B3 = 0 (GPIO)
		PORTJ.PODR.B5 = 1 (LED3 off)
		PORTJ.PDR.B5 = 1 (output)
		PORTJ.PMR.B5 = 0 (GPIO)

Table 2.139 Interrupt Initial Setting Example (IRQ11-DS Settings)

Procedure		Setting Example

1	Make interrupt settings and pin settings.	PORT4.PDR.B3 = 0 (P43 input)
		PORT4.PMR.B3 = 0 (P43GPIO)
		MPC.PWPR.B0WI = 0
		MPC.PWPR.PFSWE = 1 (PFS write enabled)
		MPC.P43PFS.ISEL = 1 (interrupt function setting IRQ11-DS)
		MPC.PWPR.PFSWE = 0 (PFS write disabled)
		MPC.PWPR.B0WI = 1

2	Enable interrupts, etc.	IRQCR11.IRQMD[1:0] = 01b (IRQ detection: falling edge)
		IRQFLTE1.FLTEN11 = 1 (IRQ11 digital noise filter enabled)
		IRQFLTC1.FCLKSEL11[1:0] = 11b (digital noise filter sampling: PCLKB/64)
		ICU.IPR075 = 0Fh (interrupt level: 15)
		ICU.IR075 = 0 (interrupt flag cleared)
		ICU.IER09.IEN3 = 1 (IRQ11 enabled)

Table 2.140 TMR0 and TMR1 Example Settings
(Cascade Connection, 16-Bit Timer, Compare Match A Toggled Output)

Procedure		Setting Example

1	Cancel module stop on TMR0 and TMR1.	SYSTEM.PRCR.WORD = A502h;
		SYSTEM.MSTPCRA.MSTPA5 = 0
		SYSTEM.PRCR.WORD = A500h;

2	Clear and stop TMR timers.	TMR0.TCNT = 00h (TMR0 TCNT cleared)
		TMR1.TCNT = 00h (TMR1 TCNT cleared)
		TMR0.TCCR = 00h (TMR0 clock stopped)
		TMR1.TCCR = 00h (TMR1 clock stopped)

3	Make TMO0 I/O settings.	PORT2.PDR.B2 = 1 (P22 output)
		PORT2.PMR.B2 = 0 (P22GPIO)
		MPC.PWPR.B0WI = 0
		MPC.PWPR.PFSWE = 1 (PFS write enabled)
		MPC.P22PFS = 05h (pin P22 set to TMO0)
		MPC.PWPR.PFSWE = 0 (PFS write disabled)
		MPC.PWPR.B0WI = 1
		PORT2.PMR.B2 = 1 (use as peripheral module)

4	Make TOCRA settings.	TMR0.TCORA = EAh
		TMR1.TCORA = 5Fh

5	Make TCR settings.	TMR0.TCR.CCLR[1:0] = 01b (cleared by compare match A) TMR0.TCR.OVIE = 0 (overflow interrupt requests disabled) TMR0.TCR.CMIEA = 0 (compare match A interrupt requests disabled) TMR0.TCR.CMIEB = 0 (compare match B interrupt requests disabled) TMR1.TCR: Left at default
6	Make TCSR settings.	TMR0.TCSR.OSA[1:0] = 11b (pin TMO0 inverted output) TMR1.TCSR: Default setting
7	Make TCCR settings. (TCNT start)	TMR0.TCCR.CSS[1:0] = 11b (TMR1.TCNT counts at overflow signal) TMR1.TCCR.CKS[2:0] = 000b (counting at PCLKB ⇒ CKS and CSS combined) TMR1.TCCR.CSS[1:0] = 01b

Table 2.141 Sleep Mode Setting Example

Procedure		Setting Example
1	Cancel protect.	SYSTEM.PRCR = A503h (cancel protect)
2	Set standby control register.	SYSTEM.SBYCR.SSBY = 0 (transition to sleep mode)
3	Enable protect.	SYSTEM.PRCR = A500h (protect enabled)

Table 2.142 All-Module Clock Stop Mode Setting Example

Procedure		Setting Example
1	Cancel protect.	SYSTEM.PRCR = A503h (cancel protect)
2	Set standby control register.	SYSTEM.SBYCR.SSBY = 0 (transition to sleep mode) SYSTEM.SBYCR.OPE = 0 (high-impedance bus output)

3	Set module stop registers A, B, C, and D.	SYSTEM.MSTPCRA.ACSE = 1
		(all-module clock stop enabled)
		SYSTEM.MSTPCRA = FFFF FFDFh
		(transition to module stop state, excluding TMR0 and TMR1)
		SYSTEM.MSTPCRB = FFFF FFFFh
		(transition to module stop state)
		SYSTEM.MSTPCRC = FFFF 0000h
		(transition to module stop state, excluding RAM)
		SYSTEM.MSTPCRD = FFFF FFFFh
		(transition to module stop state)

4	Enable protect.	SYSTEM.PRCR = A500h (protect enabled)

Table 2.143 Software Standby Setting Example

Procedure		Setting Example
1	Cancel protect.	SYSTEM.PRCR = A503h (cancel protect)

2	Set standby control register.	SYSTEM.SBYCR.SSBY = 1 (software standby enabled)
		SYSTEM.SBYCR.OPE = 0 (high-impedance bus output)

3	Make deep software standby mode setting.	SYSTEM.DPSBYCR.DPSBY = 0 (deep software standby disabled)
4	Enable protect.	SYSTEM.PRCR = A500h (protect enabled)

Table 2.144 Deep Software Standby Setting Example

Procedure		Setting Example
1	Cancel protect.	SYSTEM.PRCR = A503h (cancel protect)
2	Set standby control register.	SYSTEM.SBYCR.SSBY = 1 (software standby enabled) SYSTEM. DPSBYCR.DEEPCUT[1:0] = 01b (power not supplied to standby RAM and USB resume detection block in deep software standby mode) SYSTEM.SBYCR.OPE = 0 (high-impedance bus output) SYSTEM. DPSBYCR.IOKEEP = 0 (I/O port retention canceled simultaneously with cancelation of deep software standby mode)
3	Make deep software standby mode setting.	SYSTEM.DPSBYCR.DPSBY = 1 (deep software standby enabled) SYSTEM.DPSIER1.DIRQ11E = 1 (deep software standby enabled by IRQ11-DS)
4	Clear deep software standby interrupt flag.	SYSTEM.DPSIFR1.DIRQ11F = 0 (cancel request flag cleared by IRQ11-DS pin)
5	Enable protect.	SYSTEM.PRCR = A500h (protect enabled)

3. Sample Code

3.1 Operating Environment

The sample code associated with this application note has been confirmed to run in the following conditions.

Table 3.1 Operating Environment

Item	Description
Microcontroller used	R5F565N9 (RX65N Group)
Operating frequency	• Main clock: 24 MHz • Sub clock: 32.768 kHz • PLL: 240 MHz (main clock divided by 1 and multiplied by 10) • HOCO: Stopped • System clock (ICLK): 120 MHz (PLL divided by 2) • Peripheral module clock A(PCLKA): 120 MHz (PLL divided by 2) • Peripheral module clock B(PCLKB): 60 MHz (PLL divided by 4) • Peripheral module clock C(PCLKC): 60 MHz (PLL divided by 4) • Peripheral module clock D(PCLKD): 60 MHz (PLL divided by 4) • External bus clock (BCLK): 60 MHz (PLL divided by 4) • USB clock (UCLK): 48 MHz (PLL divided by 5)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation e^2 studio 2020-10
C compiler	Renesas Electronics Corporation C/C++ Compiler Package for RX Family (V.3.02.00)
CPU series (type)	RX600 (RX65N)
Optimization	2 (optimization of entire module) Optimization type: Priority on size
iodefine.h version	1.0C
Endian order	Little endian
Operating mode	Single-chip mode (on-chip ROM enabled extended mode only when using SDRAM)
Processor mode	Supervisor mode
Board used	Renesas Starter Kit for RX65N (Product type: RTK500565NS10000BE)

Notes:

If the same version of the toolchain (C compiler) specified in the original project is not in the import destination, the toolchain will not be selected and an error will occur.

Check the selected status of the toolchain on the project configuration dialog.

For the setting method, refer to FAQ 3000404.

FAQ 3000404 :Program ""make"" not found in PATH' error when attempting to build an imported project (e^2 studio)"

3.2 Sample Code Configuration

The configuration of the sample code is shown below.

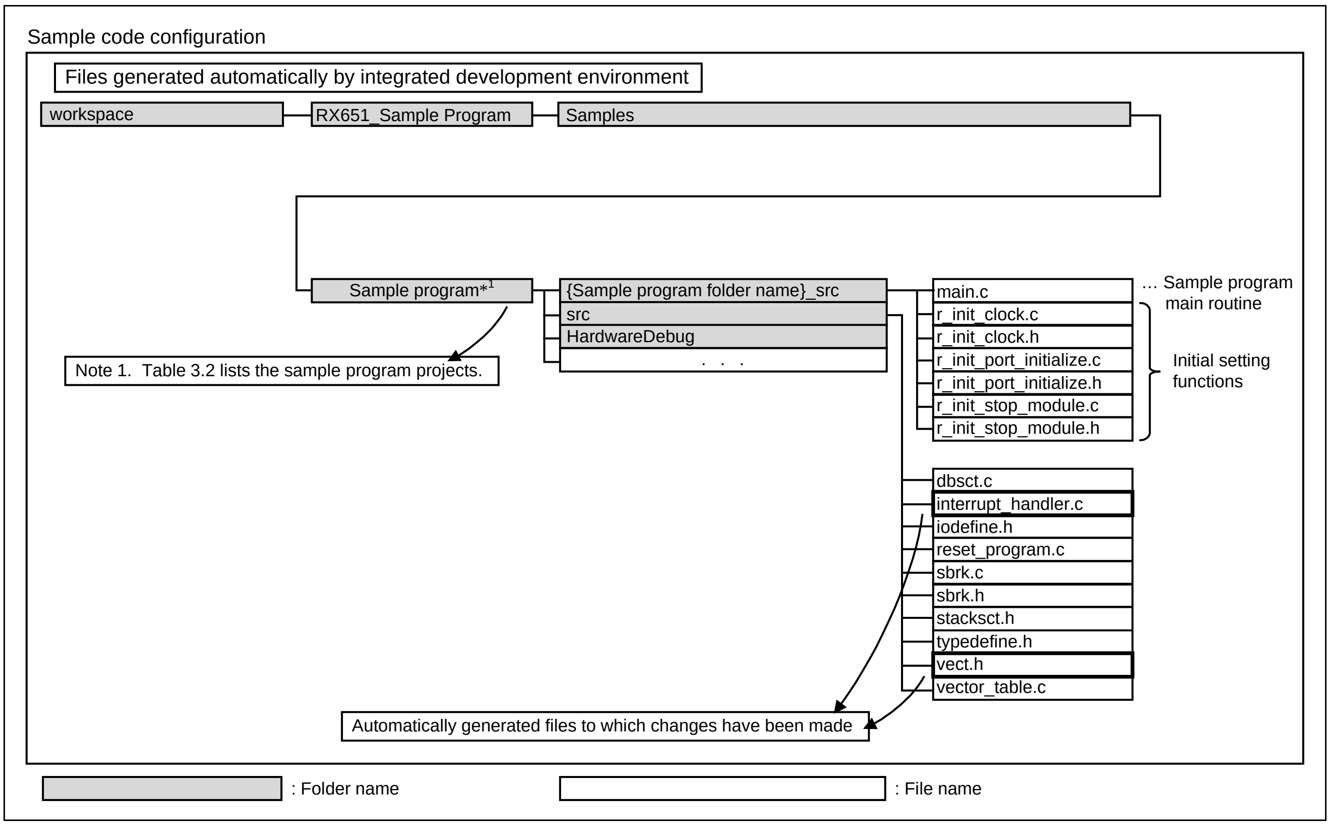


Figure 3.1 Sample Code Configuration

Initial Settings

The initial setting function of this application note uses the sample code from RX65N Group, RX651 Group: Initial Setting Example, Rev. 1.00. This revision was current when this application note was produced.

Note that the RX65N Group and RX651 Group support drivers and middleware (Firmware Integration Technology) and a sample code generation tool (Code Generator) that can be used to shorten the amount of time required for development.

Items Requiring Changes in Automatically Generated Files

The file main.c specifies interrupt declarations, vector registrations, and interrupt handlers. Portions of the automatically generated files interrupt_handlers.c and vect.h duplicate settings and code in main.c, so they have been modified as follows:

- interrupt_handlers.c: Interrupt handlers that are specified in main.c have been commented out.
- vect.h: The interrupt function declarations and vector registrations in vect.h have been commented out.

Table 3.2 List of Sample Code Projects

Sample Project Name	Related Items
BSC_sdram_read_write	2.3.3
DTC_normal_transfer_mode	2.5.7
DMA_normal_transfer_mode	2.6.10
MTU_compare_match	2.7.5
MTU_input_capture	2.7.6
SCI_asynchronous_interrupt	2.10.6
SCI_asynchronous_polling	
SCI_sync_master_transmit_int	2.10.7
SCI_sync_master_transmit_pol	
SCI_sync_slave_receive_int	2.10.8
SCI_sync_slave_receive_pol	
SCIF_asynchronous_interrupt	2.10.9
SCIF_sync_master_transmit_int	2.10.10
SCIF_sync_slave_receive_int	2.10.11
SPI_4wire_master_transceiver	2.11.8
SPI_3wire_master_transmit	2.11.9
SPI_3wire_slave_receive	2.11.10
IIC_master_transceiver	2.12.10
IIC_slave_transceiver	2.12.11
AD_continuous_scan_multi_ch	2.13.7
CMT_compare_match	2.14.5
Low_power_consumption_mode	2.16.5

4. Reference Documents

4.1 Reference Documents

Section 4.1 lists the documents referenced in the preparation of this application note. When referring to the documents listed below, substitute the latest version if a newer version is available. The latest versions of these documents can be confirmed and downloaded from the Renesas Electronics Website.

Table 4.1 Reference Documents

Reference Documents
SH7080 Group User's Manual: Hardware (R01UH0198EJ0600)
SH-1/SH-2/SH-DSP Software Manual (REJ09B0171-0500)
RX65N Group, RX651 Group User's Manual: Hardware (R01UH0590EJ0100)
RX Family RXv2 Instruction Set Architecture User's Manual: Software (R01US0071EJ0100)
SH7086 CPU Board M3A-HS86 User's Manual (REJ10J0916)
RX65N Group Renesas Starter Kit+ User's Manual (R20UT3558EG)
Renesas Starter Kit+ for RX65N CPU Board Schematics (R20UT3557EG)
RX65N Group, RX651 Group Initial Settings Example (R01AN3034EJ)

Revision History

Rev.	Date	Description	
		Page	**Summary**
1.00	Oct. 19, 2017	—	First edition issued
1.10	Dec. 21, 2020	—	Update the toolchain version.

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

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

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

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

Contact information

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