RX631 Group

SH7044 to RX631 Microcontroller Migration Guide

R01AN2207EJ0100
Rev.1.00
Sep 30, 2014

Introduction

This application note describes points requiring special attention, points of difference, etc., that need to be borne in mind when replacing the SH7044 with the RX631 in a user system. For detailed information on each function, refer to the latest version of the User’s Manual: Hardware.

Target Device

RX631/RX63N

Contents

1. CPU Architecture

1.1 Registers

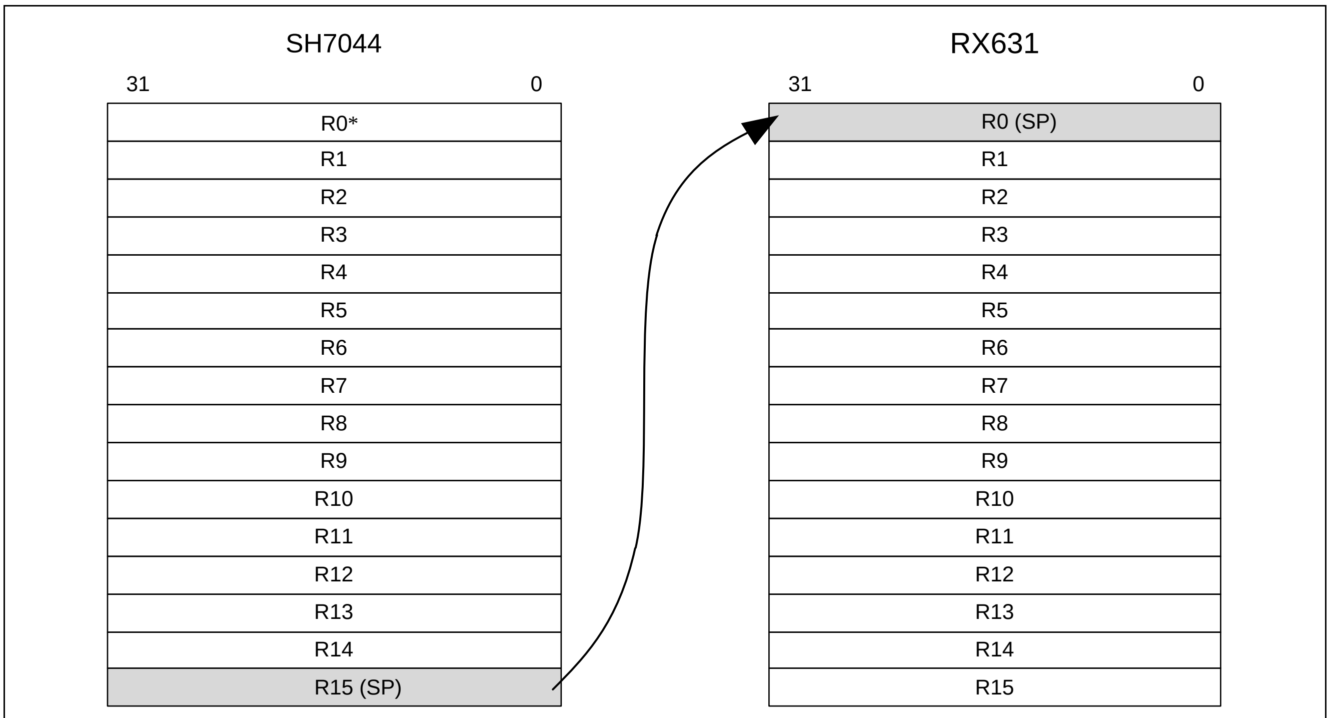
The points of difference between the registers of the SH7044 and the RX631 are described below.

1.1.1 General-Purpose Registers

The SH7044 and RX631 each have 16 32-bit general-purpose registers. They differ in that the register used as the stack pointer (SP) is different.

- SH7044: R15
- RX631: R0

On the SH7044, R0 is also used as an index register.



Note: * 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 SH7044 and the RX631.

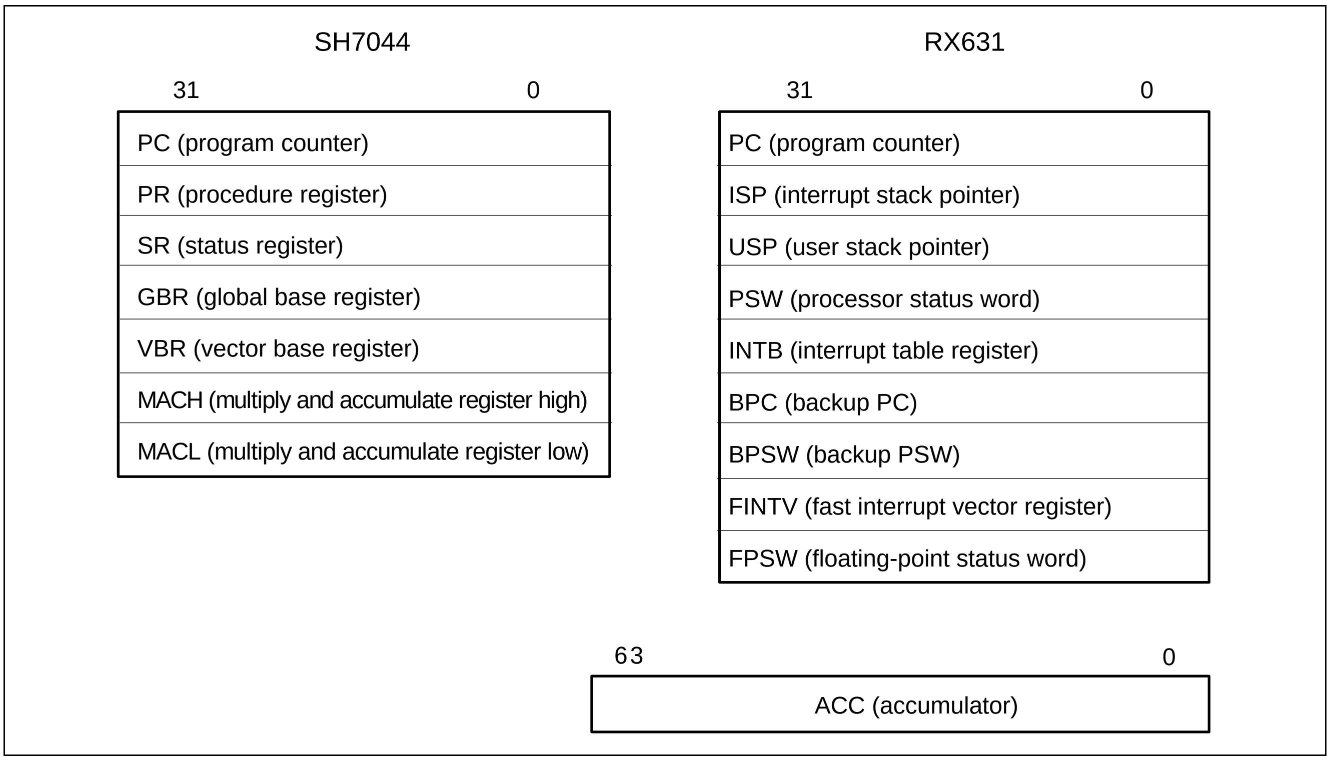

SH7044
31
0
PC (program counter)
PR (procedure register)
SR (status register)
GBR (global base register)
VBR (vector base register)
MACH (multiply and accumulate register high)
MACL (multiply and accumulate register low)
RX631
31
0
PC (program counter)
ISP (interrupt stack pointer)
USP (user stack pointer)
PSW (processor status word)
INTB (interrupt table register)
BPC (backup PC)
BPSW (backup PSW)
FINTV (fast interrupt vector register)
FPSW (floating-point status word)
63
0
ACC (accumulator)

Figure 1.2 Differences Between Control Registers

The RX631 has no registers corresponding to PR and GBR on the SH7044. The ACC register on the RX631 corresponds to MACH and MACL on the SH7044. An outline of the control registers that are implemented on the RX631 but not on the SH7044 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)*
  Specifies the start address of the relocatable vector table.
- Backup PC/backup PSW (BPC/BPSW)
  The RX631 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. Note that BPC and BPSW do not support multiple interrupts at the same time.
- 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 RX631's on-chip FPU.

Note: * The functionality of this register is equivalent to that of VBR on the SH7044.

- Differences between status registers

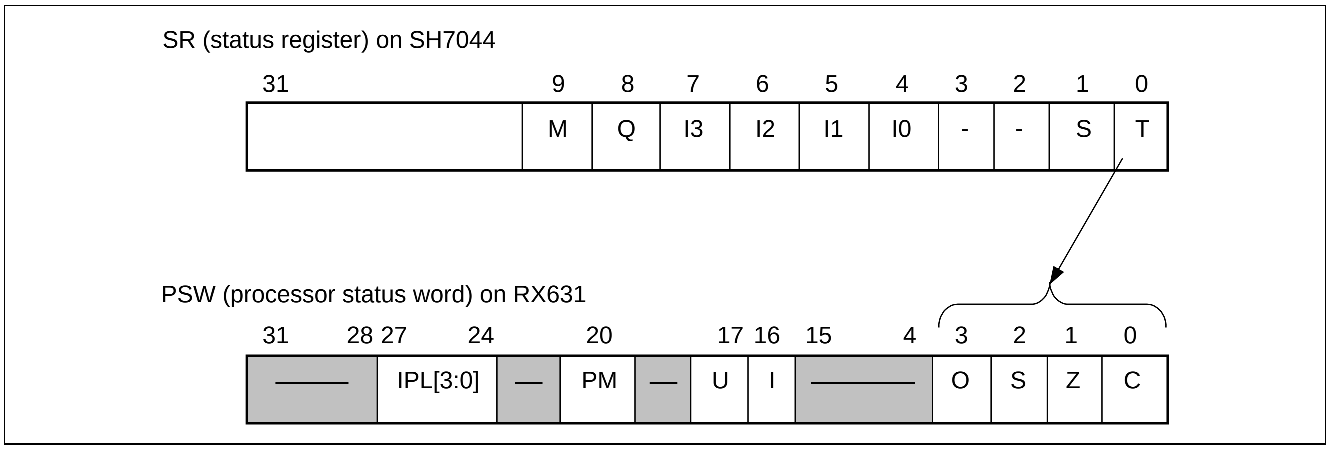


Figure 1.3 Differences Between SR (SH7044) and PSW (RX631)

Table 1.1 Differences Between SR (SH7044) and PSW (RX631)

SR Bit Name	PSW Bit Name	Description
T	C Z S O	The calculation result (true/false, carry/borrow, etc.) indicated by the T bit on the SH7044 is shown by four flags (C, Z, S, and O) on the RX631. 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 SH7044. On the RX631 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 SH7044 and the RX631 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 SH7044. There is no corresponding bit on the RX631.
M	—	The M bit is used by the DIV0U, DIV0S, and DIV1 instructions on the SH7044. There is no corresponding bit on the RX631.
—	I	Interrupt enable bit 0: Interrupts are disabled. 1: Interrupts are enabled. This bit is used to enable interrupt requests on the RX631. 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 RX631. 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 RX631. 0: Supervisor mode 1: User mode This bit is cleared to 0 when an exception is accepted.

1.2 Option-Setting Memory

The RX631 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

An outline of the option-setting memory area is shown below.

b31 b0

Address	...	Register Description
FF7F FFE8h to FF7F FFEFh	UB code A	Codes necessary when using user boot mode. (Do not overwrite these codes when using USB boot mode.)
FF7F FFF0h to FF7F FFF7h	UB code B	
FF7F FFF8h to FF7F FFFBh	Endian select register B (MDEB) (user boot mode)	Register for selecting the endian setting of the CPU.
	—	—
FFFF FF80h to FFFF FF83h	Endian select register S (MDES) (single-chip mode)	Register for selecting the endian setting of the CPU.
	—	—
FFFF FF88h to FFFF FF8Bh	Option function select register 1 (OFS1)	The OFS1 register is used to make the following two settings: • Voltage monitor 0 reset is enabled/ disabled after a reset. • HOCO oscillation is enabled/ disabled after a reset.
FFFF FF8Ch to FFFF FF8Fh	Option function select register 0 (OFS0)	The OFS0 register is used to make settings for the independent watchdog timer (IWDT) and watchdog timer (WDT).
	...	—

Figure 1.4 Option-Setting Memory Area

Sample settings for the option-setting memory are shown below.

```
/* Settings for single-chip mode and big-endian */
#define __BIG
#pragma address MDEreg=0xffffff80 // MDE register (Single Chip Mode)
#ifdef __BIG
          const unsigned long MDEreg = 0xfffffff8; // big
#else
          const unsigned long MDEreg = 0xffffffff; // little
#endif
```

Figure 1.5 Endian Setting Example

Sample settings for OFS0 and OFS1 are shown below. (The code below is included in the automatically generated files.)

```
#pragma address OFS1_REG = 0xFFFFFF88   /* OFS1 register */
const unsigned long OFS1_REG = 0xFFFFFFFF;

#pragma address OFS0_REG = 0xFFFFFF8C   /* OFS0 register */
const unsigned long OFS0_REG = 0xFFFFFFFF;
```

Figure 1.6 OFS0 and OFS1 Setting Example

1.2.2 Endian Setting

The SH7044 is fixed in big-endian mode. On the RX631, 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 MDES and MDEB registers in the option-setting memory.

When switching from the SH7044 to the RX631, 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.)

In actuality, code such as that shown in figure 1.5, Endian Setting Example, is generated automatically according to the compiler option setting.*

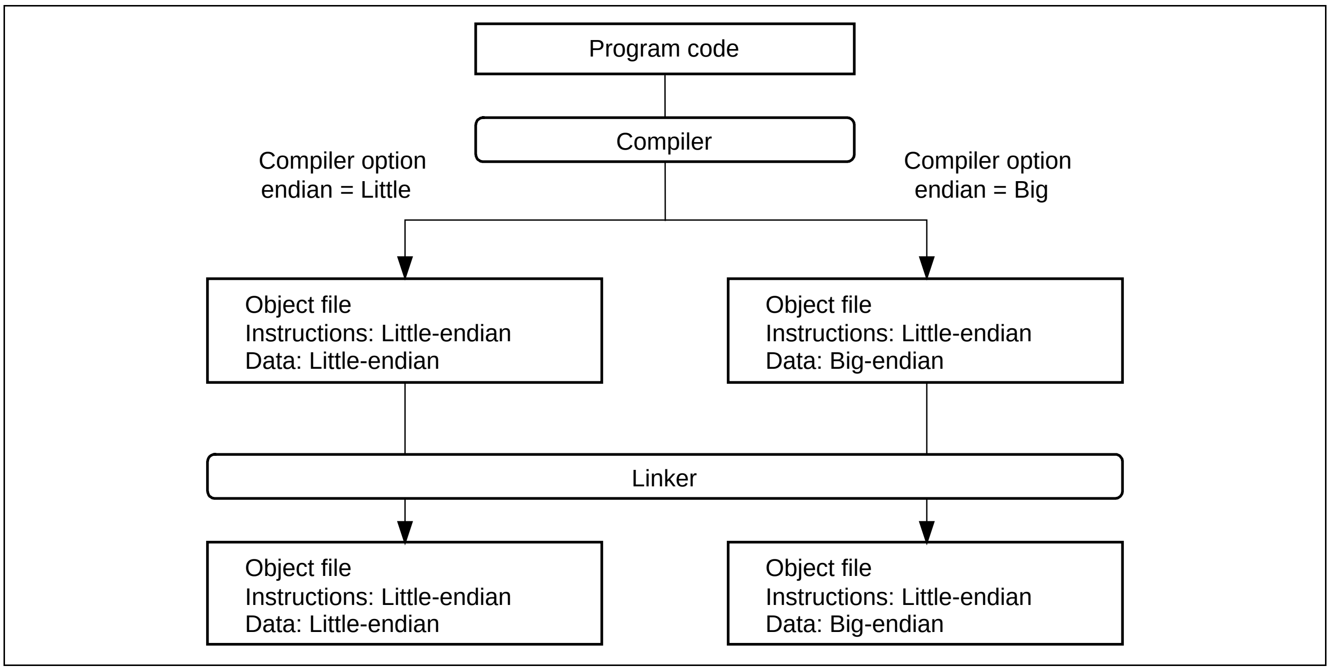

Program code
Compiler
Compiler option endian = Little
Compiler option endian = Big
Object file Instructions: Little-endian Data: Little-endian
Object file Instructions: Little-endian Data: Big-endian
Linker
Object file Instructions: Little-endian Data: Little-endian
Object file Instructions: Little-endian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

Figure 1.7 Specifying Endianness by Compiler Option

Note: * The automatically generated files work in the sample code operating environment described in section 3.1.

1.3 Reset Function

1.3.1 Reset Sources

Table 1.2 lists the reset sources of the SH7044 and RX631.

Table 1.2 Reset Sources

	SH7044	RX631
Reset type	• Power-on reset (pin reset) • Manual reset (pin 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

- Reset vector configuration
  The SH7044 has separate vectors for power-on resets and for manual resets (PC and SP).*
  The RX631 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.
- Stack pointer
  On the SH7044, 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 RX631, so the stack pointer is set in ISP and USP.

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

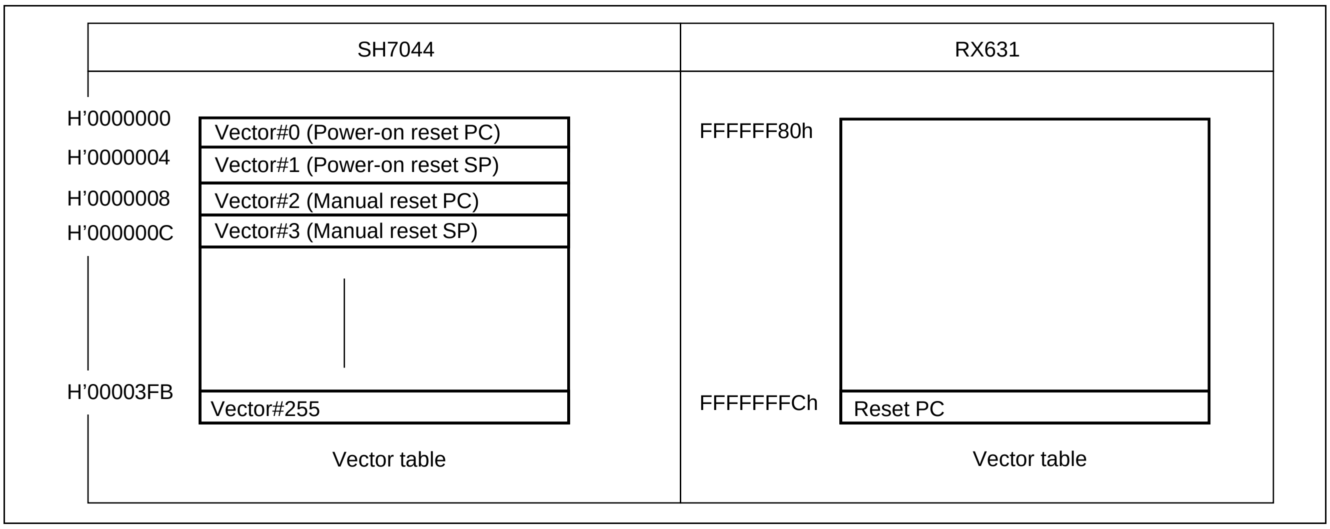


Figure 1.8 Reset Vectors on SH7044 and RX631

1.3.2 Reset Sources and Initialization Scope

The initialization scope of the reset sources differs between the SH7044 and the RX631. Table 1.3 lists the reset types and their initialization scope on the SH7044, and table 1.4 lists the reset types and their initialization scope on the RX631. (For details, see the User's Manual: Hardware.)

Table 1.3 SH7044 Reset Sources and Initialization Scope

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

○: Reset —: No reset

Table 1.4 RX631 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

Notes: 1. Only LVD1CR1 and LVD1SR are initialized.
2. Only LVD1CR2 and LVD2SR are initialized.

1.4 Clock Settings

1.4.1 Clock Sources

The clock sources and clock generation circuits of the SH7044 and RX631 are listed below.

Table 1.5 List of SH7044 and RX631 Clock Sources

SH7044	RX631
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) • Low-speed on-chip oscillator (LOCO) • IWDT-dedicated on-chip oscillator

Note: 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 SH7044 clock control is not performed in software. Each peripheral device operations in synchronization with the system clock (ϕ) or a clock generated by the prescaler. On the RX631 a large variety of clocks operate under software control.

On the RX631 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.

RX63N Group, RX631 Group Initial Setting (R01AN1245EJ)

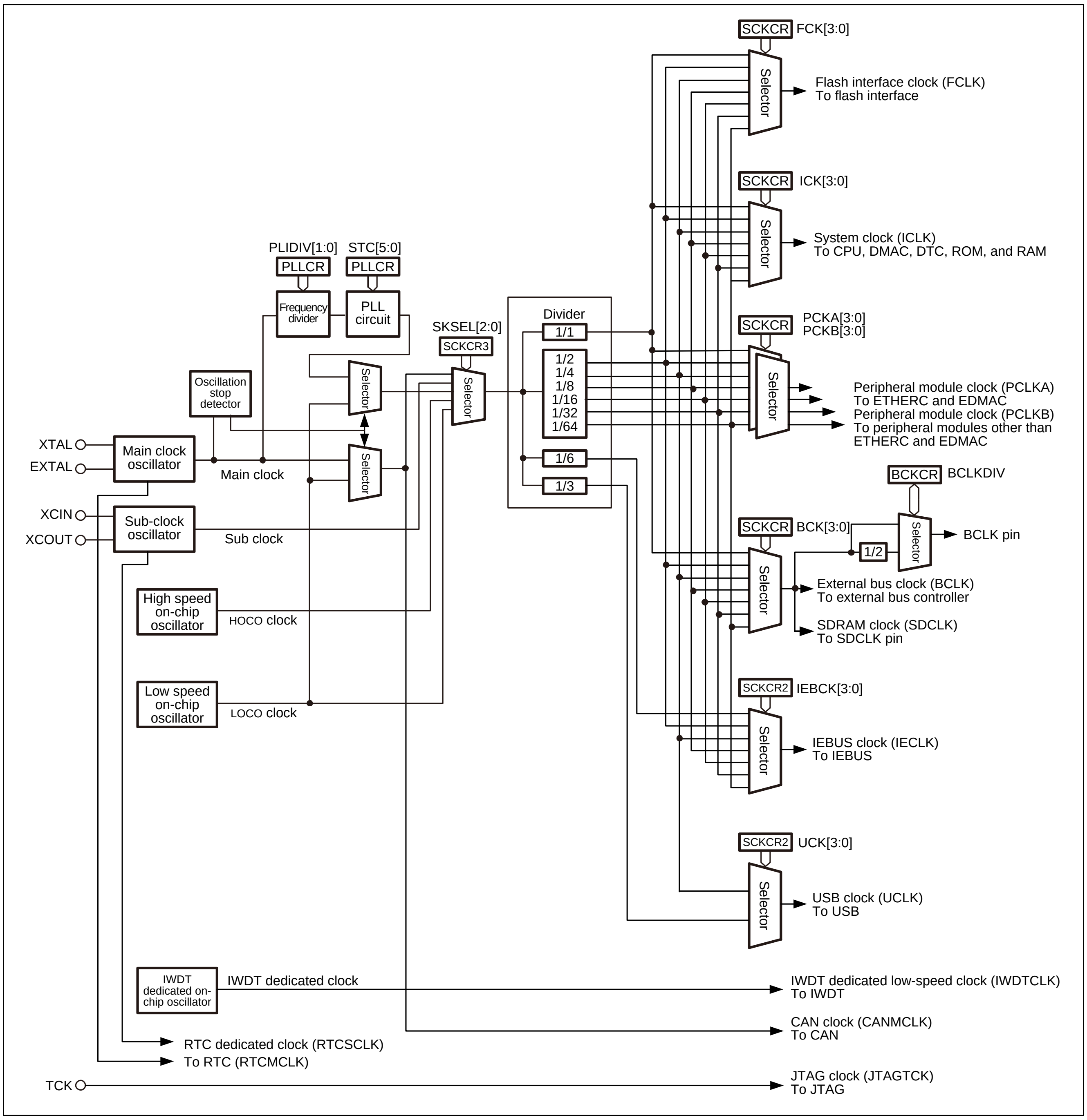


Figure 1.9 RX631 Clock Generation Circuit

1.5 Operation Modes

1.5.1 Comparison of Operation Modes

The table below shows a comparison of the operation modes of the SH7044 and RX631. For details of each operation mode, see the User's Manual: Hardware.

Table 1.6 Comparison of Operation Modes

SH7044 Operation Mode	RX631 Operation Mode	Description
MCU 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 SH7044.
MCU mode 1		
MCU 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 on-chip ROM is enabled and the external address space is disabled
Boot mode	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 on-chip flash memory can be programmed by a device external to the microcontroller by using the asynchronous serial interface.
User program mode	Functionality equivalent to the SH7044 can be implemented in ordinary operation mode.	An operation mode that is only transitioned to when the setting value of the FWP pin changes and in which the on-chip flash memory is programmed by a programming/erase control program that has been prepared ahead of time by the user. It is possible to implement equivalent functionality in ordinary operation mode on the RX631, so it is not necessary to change the pin states.
—	USB boot mode / user boot mode	An operation mode in which the on-chip flash memory modifying program stored in the user boot area is run. When the microcontroller is in the default factory state the mode is USB boot mode, and when a flash memory modifying program created by the user has been stored in the user boot area the mode is user boot mode. It is possible to use a user-defined interface to select between the USB and user modes and program the on-chip flash memory with a device external to the microcontroller. Programming of the user boot area is only possible in boot mode.

1.5.2 Comparison of Memory

The figure below shows a comparison of memory maps in on-chip ROM enabled mode (on-chip ROM enabled extended mode on the RX631).

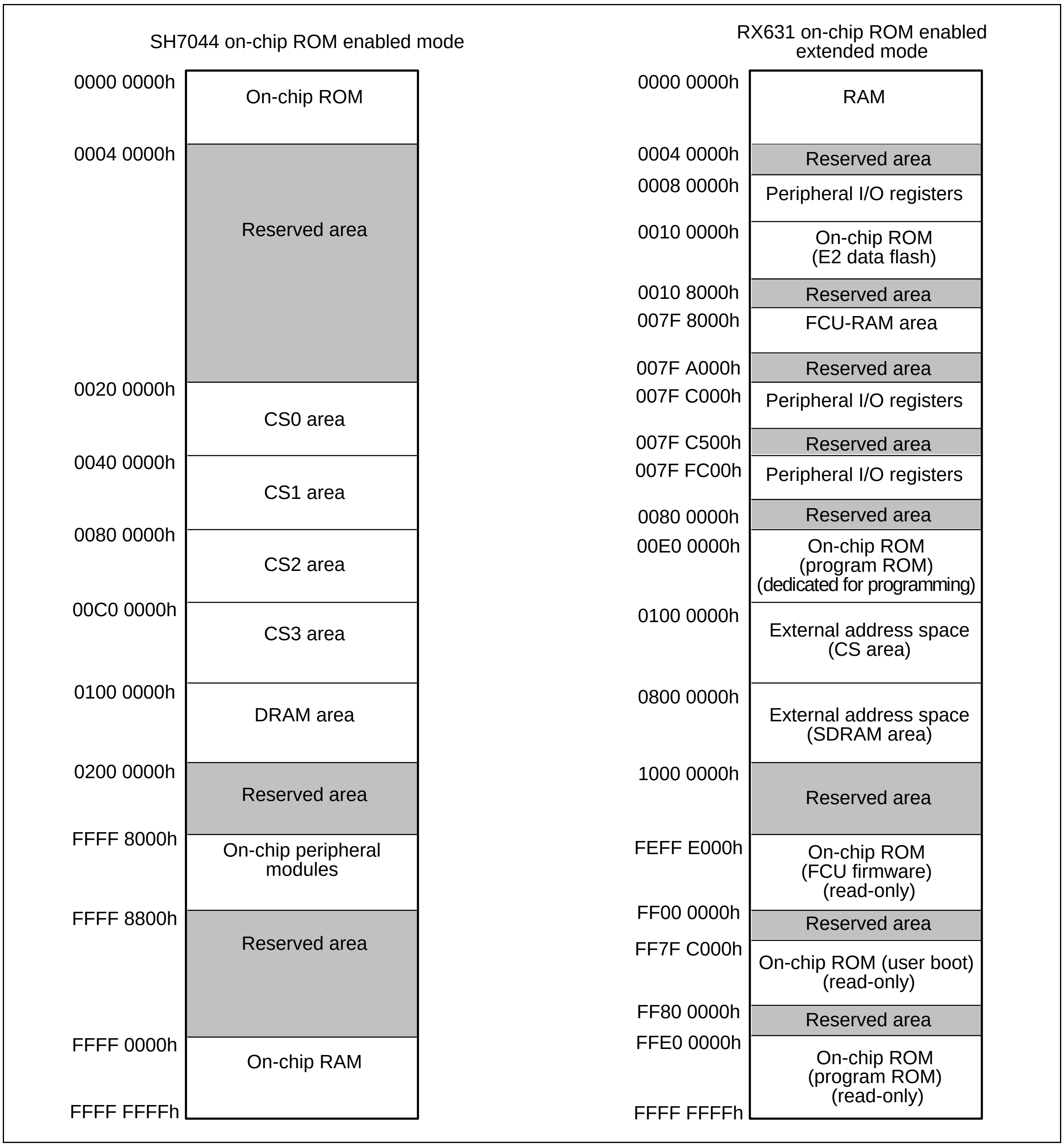


Figure 1.10 SH7044 and RX631 Memory Map Comparison (On-Chip ROM Enabled Mode)

The figure below shows a comparison of memory maps in single-chip mode.

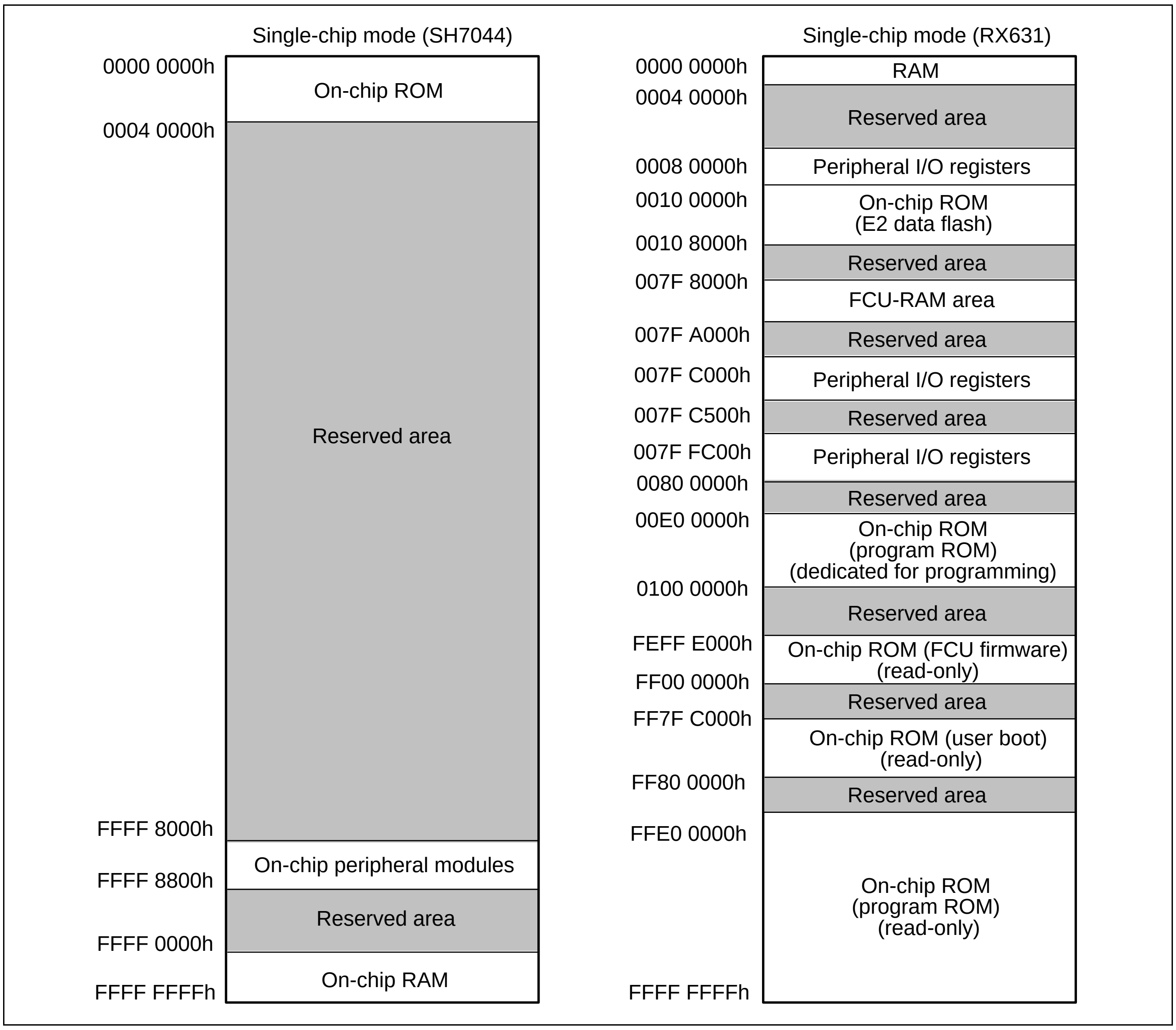


Figure 1.11 SH7044 and RX631 Memory Map Comparison (Single-Chip Mode)

The figure below shows a comparison of memory maps in on-chip ROM disabled mode.

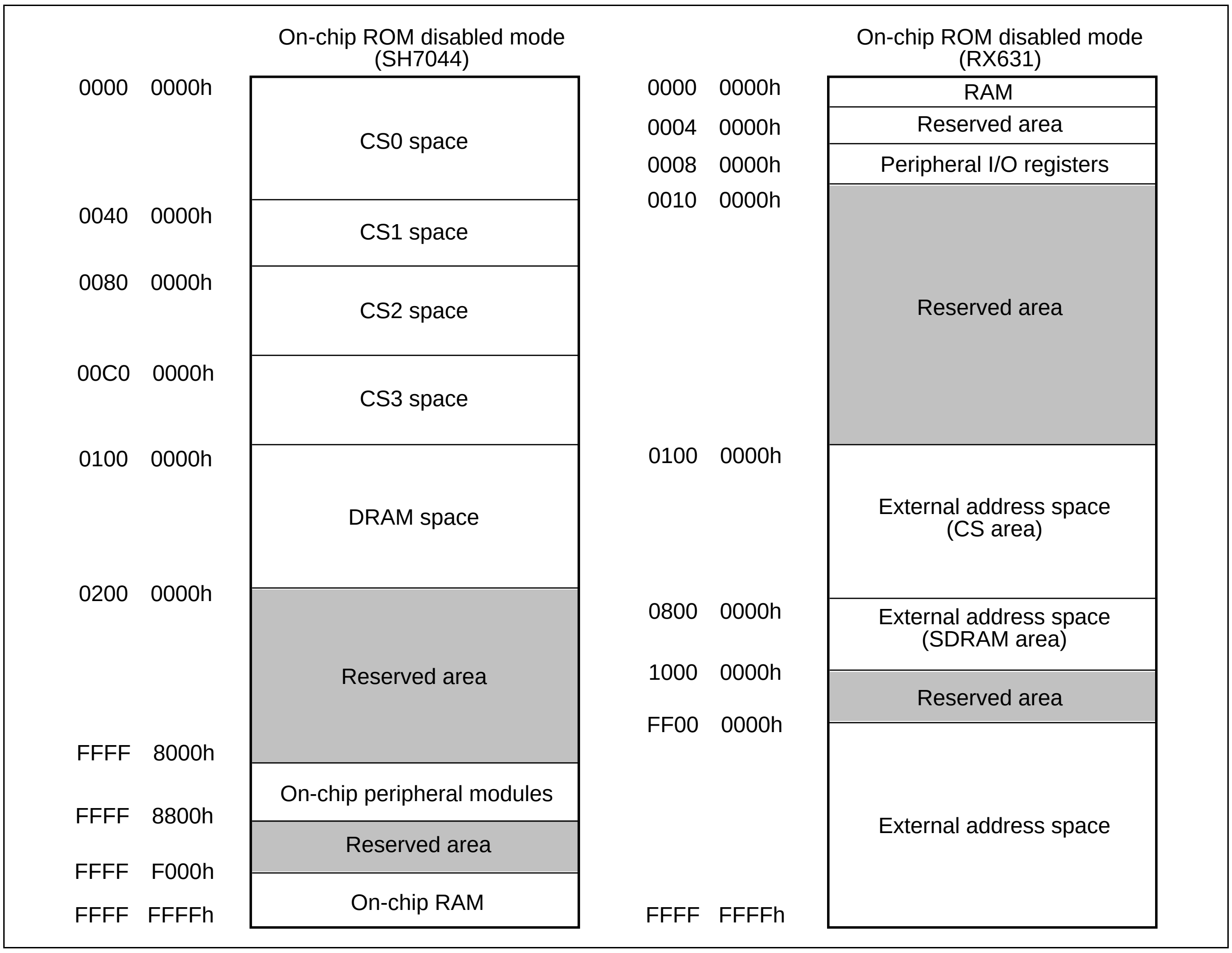


Figure 1.12 SH7044 and RX631 Memory Map Comparison (On-Chip ROM Disabled Mode)

- On the RX631 the RAM is allocated to addresses adjacent to 0000 0000h and ROM (for reading data) to addresses adjacent to FFFF FFFFh. Also, the RX631 has on-chip E2 data flash for storing data.
- On the RX631 the peripheral IO registers are allocated within the address range from 0008 0000h to 000F FFFFh, and only the flash-related registers and peripheral clock notification register are allocated within the address range from 007F C000h to 007F FFFFh.
- On the RX631 the external address space is allocated within the address range from 0100 0000h to 0FFF FFFFh and configured as seven CS spaces of 16 MB each and a 128 MB SDRAM space.

1.5.3 Operation Mode Settings

Whereas on the SH7044 operation mode settings are made only with the MD1, MD0, and FWP pins, on the RX631 operation mode settings can be made by means of the MD pin and PC7 or PA6 pin 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 RX631

Pin		
MD	**PC7*[2], PA6*[2]**	**Mode Name**
1	—	Single-chip mode
0	0	Boot mode
	1	USB boot mode / user boot mode*[1]

Notes: 1. When the microcontroller is in the default factory state the USB boot program is stored in the user boot area and the microcontroller starts in USB boot mode.
2. The pin differs according to the package type. For details, see the User's Manual: Hardware.

Table 1.8 SYSCR0 Register Settings and Operation Modes on RX631

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

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

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 SH7044 with the RX631, 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 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        #00100000h,R1   ;
PUSH.L    R1              ;
MVFC      PC,R1           ;
ADD       #10,R1          ;
PUSH.L    R1              ;
RTE
NOP
NOP
```

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.

1.7 Exception Handling

The points of difference regarding exception handling in general on the SH7044 and RX631, including interrupts, are described below.

1.7.1 Types of Exception Handling

A comparative listing of exception sources on the SH7044 and RX631 is shown below.

Table 1.10 Exception Sources on SH7044 and RX631

SH7044	RX631	Main Points of Difference
Power-on reset Manual reset	Reset	On the SH7044 there are separate vectors for power-on resets and manual resets. On the RX631 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.
Address error	Access exception	On the SH7044 this exception occurs when an attempt is made to access an access-prohibited area or an address to which access is prohibited. On the RX631 this exception occurs when a memory protection error occurs. On the SH7044 the next instruction is saved to PC when this exception occurs. On the RX631 the instruction that generated this exception is saved to PC.
Interrupt (NMI)	Non-maskable interrupt	None
Interrupt (external/internal)	Interrupt (external/internal)	The RX631 also supports fast interrupts (level 15)
TRAP instruction (TRAPA instruction)	Unconditional trap (INT, BRK instruction)	The SH7044 has 32 sources, but the RX631 has 16 sources with dedicated vectors and up to 256 sources when sources also used for interrupts are included.
General illegal instruction Illegal slot instruction	Undefined instruction	The SH7044 has no exceptions corresponding to the privileged instruction exception or floating-point exception. On the SH7044 the next instruction is saved to PC when one of these exceptions occurs, but on the RX631 the instruction that generated this exception is saved to PC.
—	Privileged instruction	
—	Floating-point exception	

1.7.2 Exception Handling Priority

The comparative priority of exception sources on the SH7044 and RX631 is shown below.

Table 1.11 Exception Event Priority

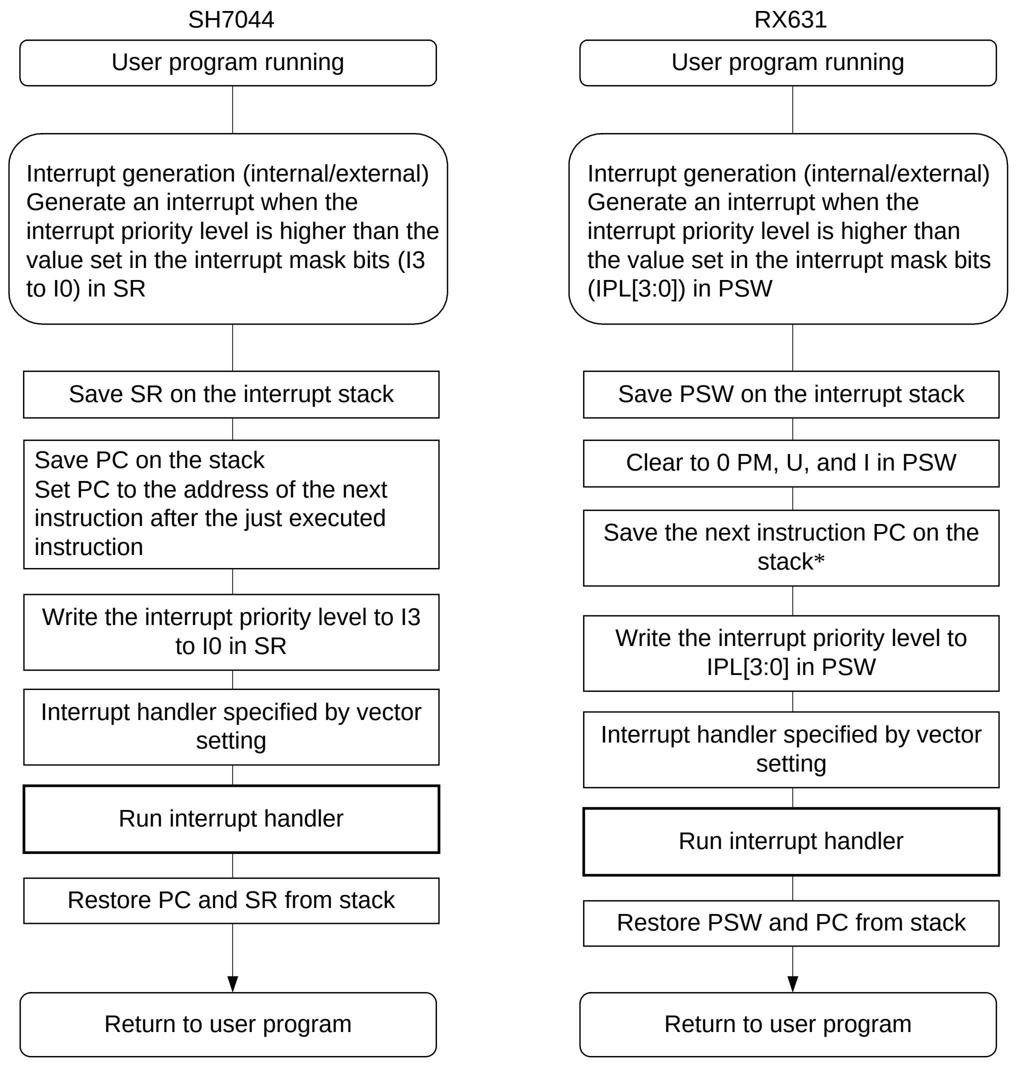
Priority	SH7044	RX631	Remarks
High ↑	Power-on reset	Reset	
	Manual reset	Non-maskable interrupt	
	Address error exception	Interrupt (external/internal)	
	Interrupt (NMI)	Instruction access exception	
	Interrupt (external/internal)	Undefined instruction exception, privileged instruction exception	
	TRAP instruction	Unconditional trap	
	General illegal instruction exception	Operand access exception	
Low	Illegal slot instruction exception	Floating-point exception	

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

On the SH7044 address errors have higher priority than interrupts (internal or external), but on the RX631 both instruction access exceptions and operand access exceptions have lower priority than interrupts.

1.7.3 Basic Processing Sequence of Exception Handling

The basic processing sequence interrupt exception handling on the SH7044 and RX631 is shown below.


SH7044
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 to I3 to I0 in SR
Interrupt handler specified by vector setting
Run interrupt handler
Restore PC and SR from stack
Return to user program
RX631
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*
Write the interrupt priority level to IPL[3:0] in PSW
Interrupt handler specified by vector setting
Run interrupt handler
Restore PSW and PC from stack
Return to user program

Note: * 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 1.13 Interrupt (Internal/External) Processing Sequence

1.7.4 Vector Configuration

Both the SH7044 and RX631 have a relocatable vector configuration, which allows the vector table to be reallocated. On the SH7044 the vector base register (VBR) 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 RX631 there are clearly separated fixed relocatable vector tables. System exceptions such as resets are assigned a fixed vector that cannot be reallocated. Reallocatable interrupt and unconditional trap vectors are assigned in a relocatable vector table, and the start address of the relocatable vector table is set in the interrupt table register (INTB). Also, the fast interrupt vector is set in the FINTV register.

Figure 1.14 shows the differences between the vector tables.

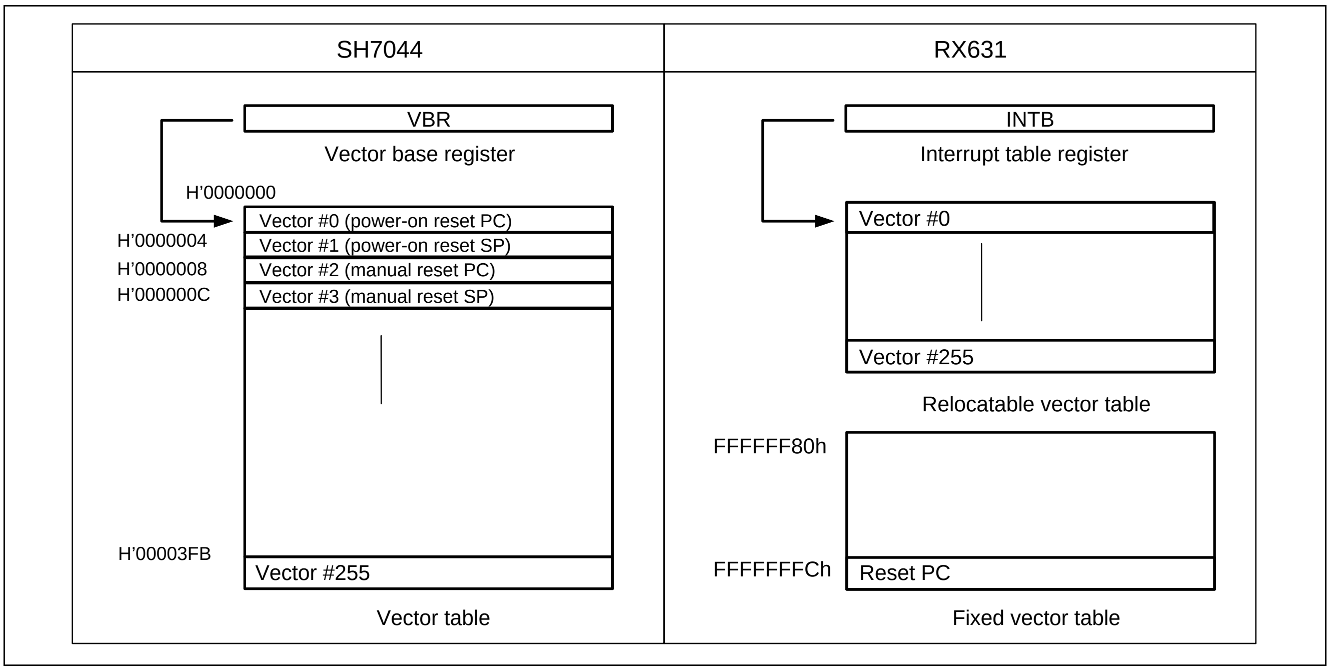


Figure 1.14 Vector Table Settings

1.7.5 Interrupt Masking by SR (SH7044) and PSW (RX631)

On the RX631 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

SH7044	**RX631**	
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 SH7044 and RX631, 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 Interrupt Controller Specifications

Item		SH7044	RX631
Interrupts	Peripheral function interrupts	• Interrupts from peripheral modules • Interrupt detection: Edge/level*1	• Interrupts from peripheral modules • Interrupt detection: Edge/level*1 • Group interrupt function support • Unit selection function support
	External pin interrupts	• IRQ0 to IRQ7 pins • Sources: 8 • Interrupt detection: Low level or falling edge 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. • Digital filter function support
	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 and DMAC control	Activation supported*2	Activation supported
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) • Digital filter function support
	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 • Undefined instruction exception • Privileged instruction exception • Access exception • Floating-point exception • Unconditional trap

Notes: 1. The detection method is fixed for fixed-connection peripheral modules.
2. On the SH7044 activation source setting is performed on the DTC or DMAC.

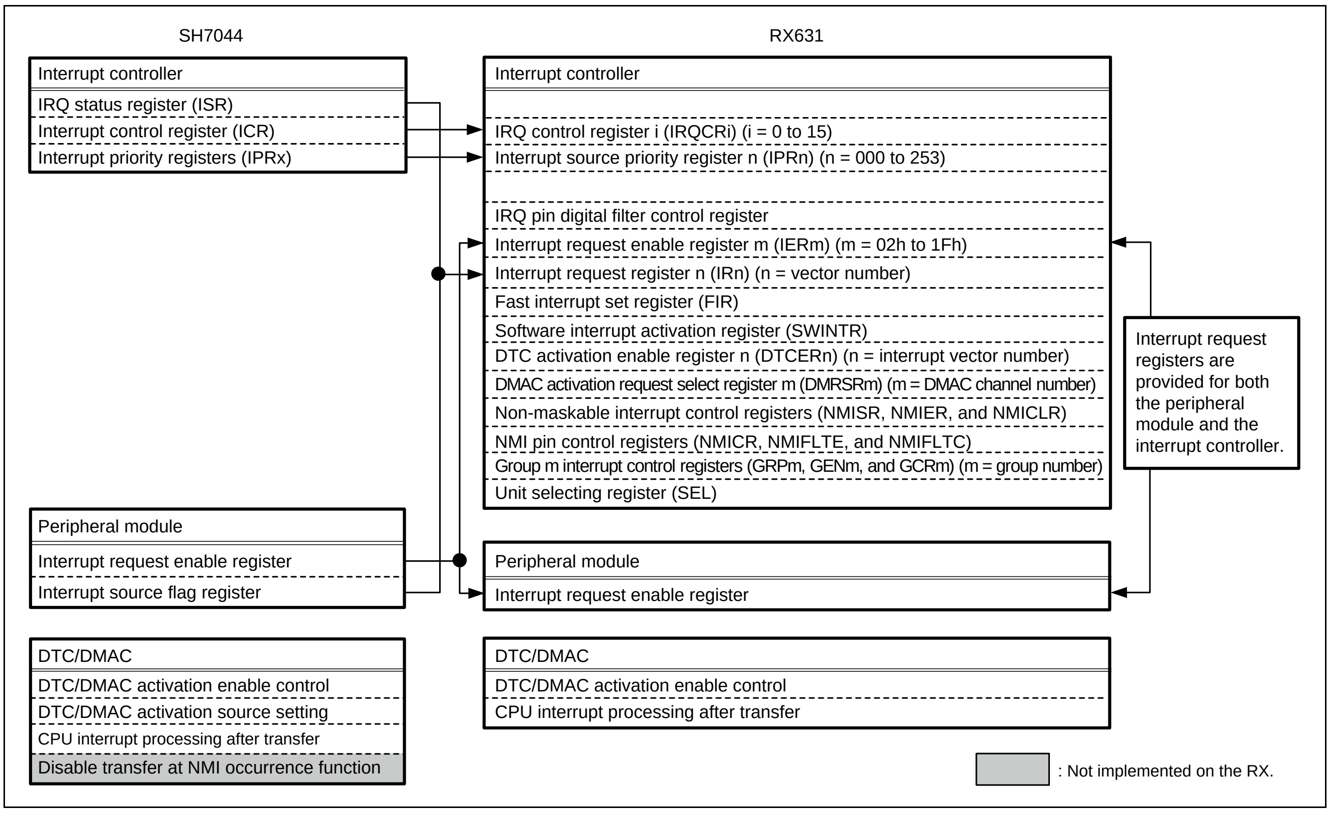

SH7044
RX631
Interrupt controller
IRQ status register (ISR)
Interrupt control register (ICR)
Interrupt priority registers (IPRx)
Interrupt controller
IRQ control register i (IRQCRi) (i = 0 to 15)
Interrupt source priority register n (IPRn) (n = 000 to 253)
IRQ pin digital filter control register
Interrupt request enable register m (IERm) (m = 02h to 1Fh)
Interrupt request register n (IRn) (n = vector number)
Fast interrupt set register (FIR)
Software interrupt activation register (SWINTR)
DTC activation enable register n (DTCERn) (n = interrupt vector number)
DMAC activation request select register m (DMRSRm) (m = DMAC channel number)
Non-maskable interrupt control registers (NMISR, NMIER, and NMICLR)
NMI pin control registers (NMICR, NMIFLTE, and NMIFLTC)
Group m interrupt control registers (GRPm, GENm, and GCRm) (m = group number)
Unit selecting register (SEL)
Interrupt request registers are provided for both the peripheral module and the interrupt controller.
Peripheral module
Interrupt request enable register
Interrupt source flag register
Peripheral module
Interrupt request enable register
DTC/DMAC
DTC/DMAC activation enable control
DTC/DMAC activation source setting
CPU interrupt processing after transfer
Disable transfer at NMI occurrence function
DTC/DMAC
DTC/DMAC activation enable control
CPU interrupt processing after transfer
: Not implemented on the RX.

Figure 1.15 Differences Between Interrupt Controller Registers

Figure 1.15 shows the differences between the interrupt controllers of the SH7044 and RX631.

The interrupt controller of the SH7044 controls IRQ interrupt flags, while peripheral module interrupt flags are controlled by the peripheral modules.

On the RX631 the interrupt controller controls all interrupt status flags, for both IRQs and peripheral modules.* 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 SH7044 is not implemented on the RX631.

Note: * 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 SH7044 generates an interrupt by edge detection, the corresponding interrupt flag (interrupt source flag) in the interrupt handler is cleared (the flag is cleared and a dummy read is performed). This is done because the interrupt will be generated once again if the flag is not cleared by the handler. On the RX631 the interrupt flags (interrupt status flags) are managed internally by the interrupt controller. The interrupt controller has a function whereby when it sends an interrupt request to the CPU or DTC/DMAC and receives a response indicating that it was accepted, it automatically clears the corresponding interrupt status flag. It is therefore not necessary to clear the flag and do a dummy read as on the SH7044. Note that in the case of interrupts generated by level detection the source flags reside in the peripheral modules, so they do need to be cleared. For details, see the User's Manual: Hardware.

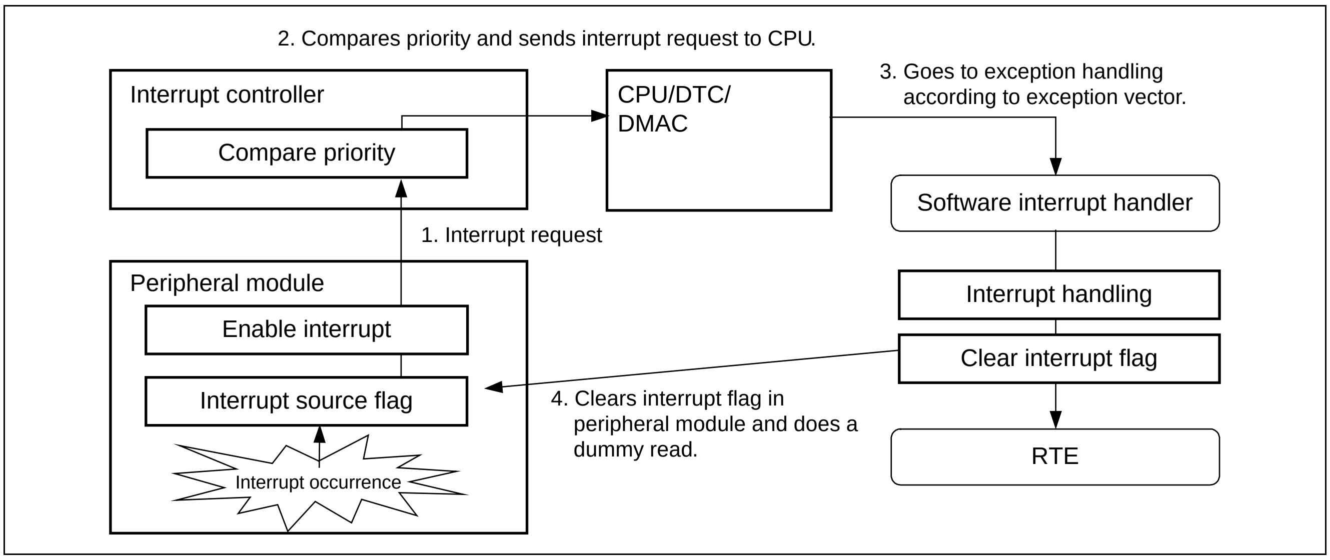

2. Compares priority and sends interrupt request to CPU.
Interrupt controller
Compare priority
CPU/DTC/ DMAC
3. Goes to exception handling according to exception vector.
Software interrupt handler
1. Interrupt request
Peripheral module
Enable interrupt
Interrupt source flag
Interrupt occurrence
Interrupt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

Figure 1.16 SH7044 Peripheral Module Interrupt (Edge Detection)

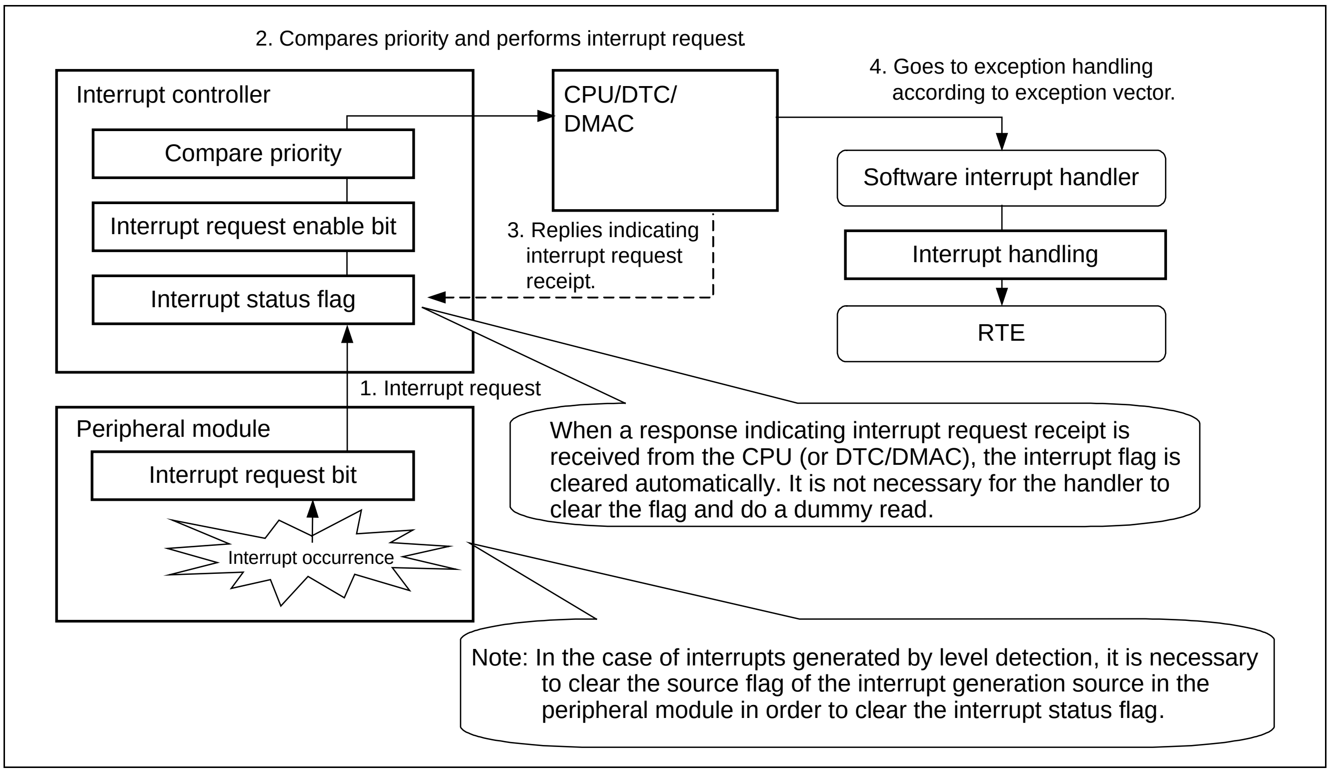

2. Compares priority and performs interrupt request.
Interrupt controller
Compare priority
Interrupt request enable bit
Interrupt status flag
CPU/DTC/ DMAC
4. Goes to exception handling according to exception vector.
Software interrupt handler
Interrupt handling
RTE
3. Replies indicating interrupt request receipt.
1. Interrupt request
Peripheral module
Interrupt request 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 1.17 RX631 Peripheral Module Interrupt (Edge Detection)

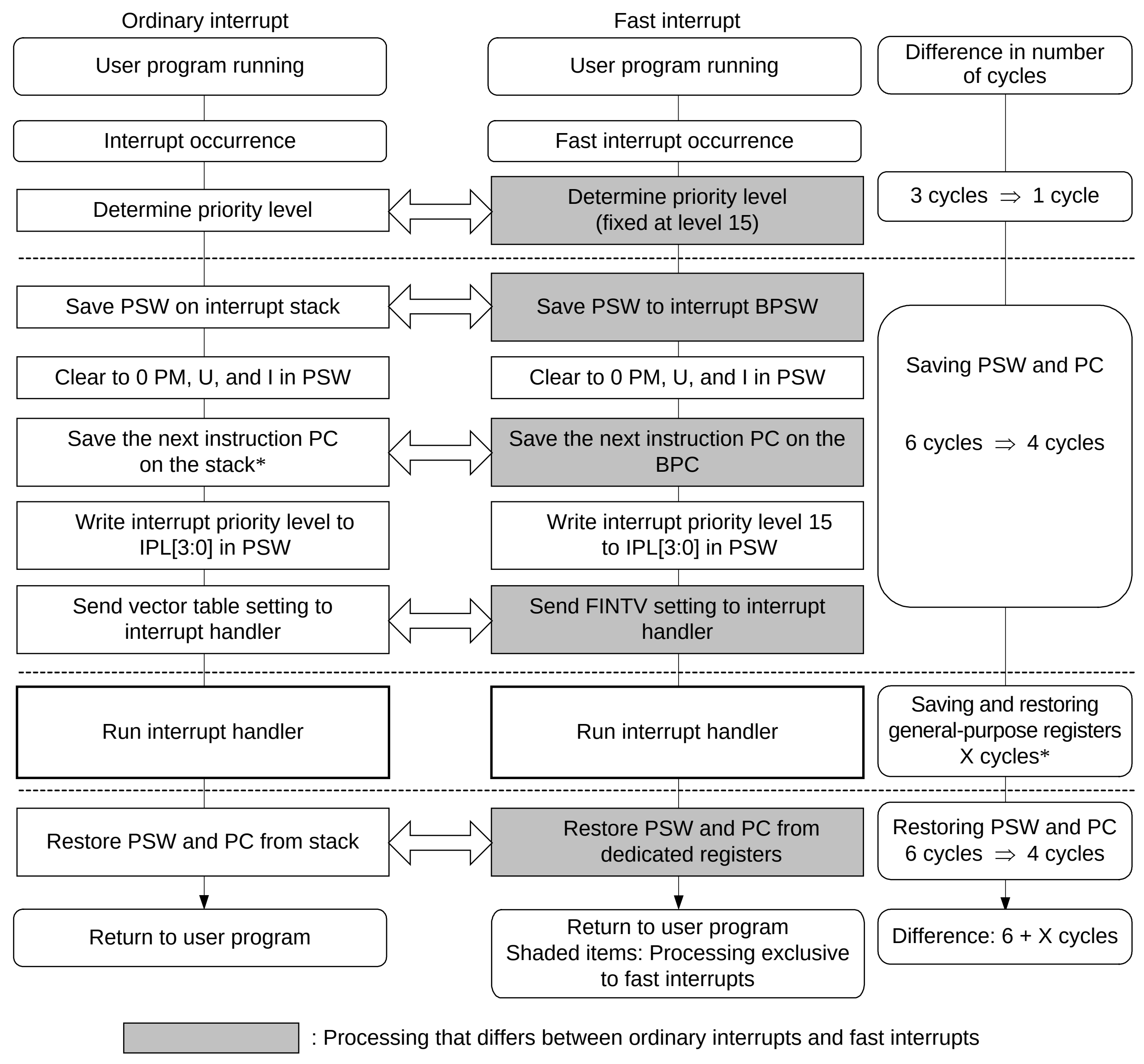
1.8.3 Fast Interrupt Control

In addition to ordinary interrupts, the RX631 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
Fast interrupt
User program running
User program running
Difference in number of cycles
Interrupt occurrence
Fast interrupt occurrence
Determine priority level
Determine priority level (fixed at level 15)
3 cycles ⇒ 1 cycle
Save PSW on interrupt stack
Save PSW to interrupt BPSW
Clear to 0 PM, U, and I in PSW
Clear to 0 PM, U, and I in PSW
Saving PSW and PC
6 cycles ⇒ 4 cycles
Save the next instruction PC on the stack*
Save the next instruction PC on the BPC
Write interrupt priority level to IPL[3:0] in PSW
Write interrupt priority level 15 to IPL[3:0] in PSW
Send vector table setting to interrupt handler
Send FINTV setting to interrupt handler
Run interrupt handler
Run interrupt handler
Saving and restoring general-purpose registers X cycles*
Restore PSW and PC from stack
Restore PSW and PC from dedicated registers
Restoring PSW and PC 6 cycles ⇒ 4 cycles
Return to user program
Return to user program Shaded items: Processing exclusive to fast interrupts
Difference: 6 + X cycles
▨ : Processing that differs between ordinary interrupts and fast interrupts

Note: * 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 1.18 Differences Between Ordinary Interrupts and Fast Interrupts

1.8.4 Digital Filter

The RX631 is provided with a digital filter function for the IRQ and NMI level signals. 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. This improves the system's noise tolerance.

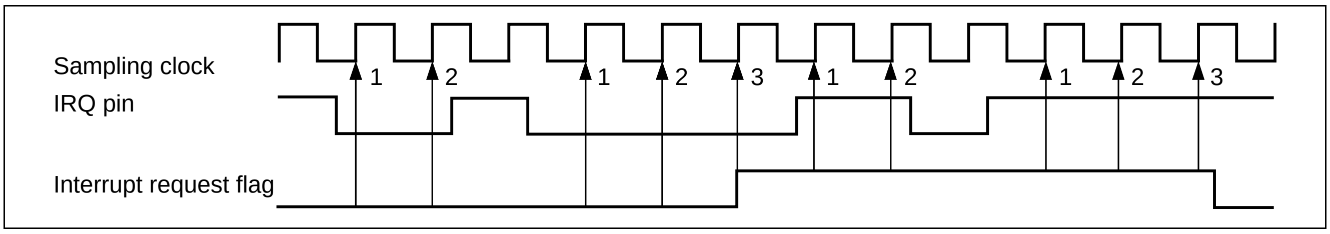


Figure 1.19 Digital Filter Operation Example

1.8.5 Multiple Interrupts

On the SH7044 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 RX631 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 SH7044, it is necessary to set the PSW.I bit to 1 (interrupts are enabled) in the interrupt handler.

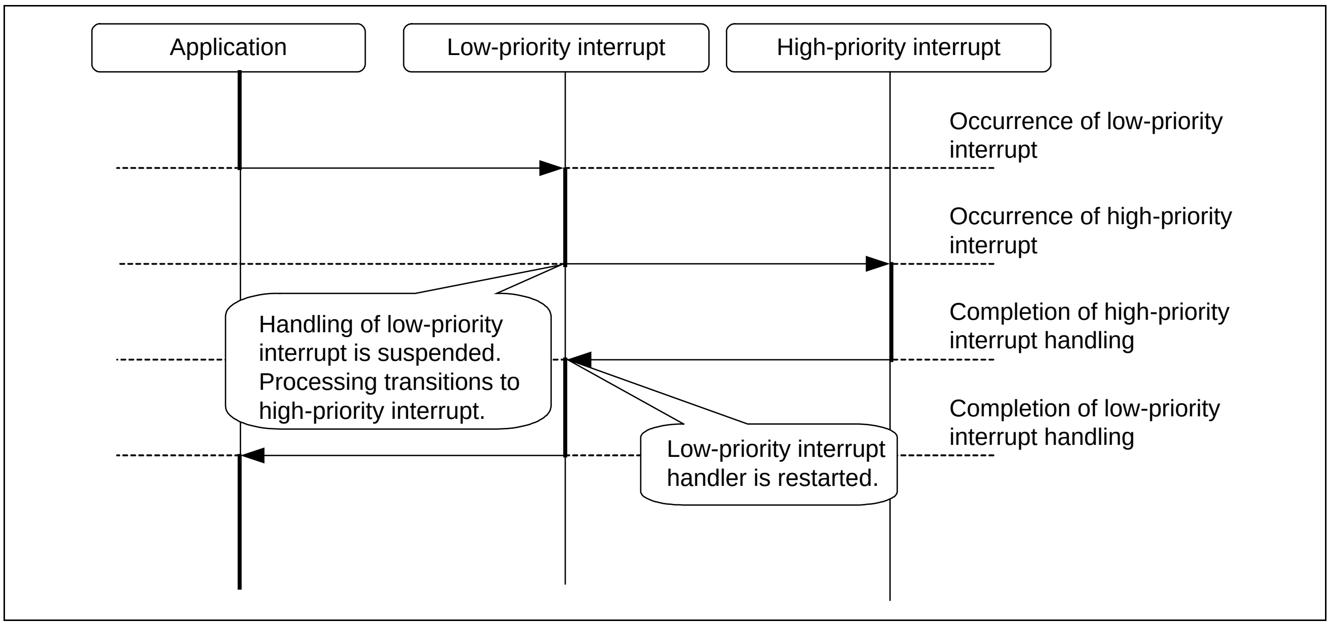

Application
Low-priority interrupt
High-priority interrupt
Occurrence of low-priority interrupt
Occurrence of high-priority interrupt
Completion of high-priority interrupt handling
Completion of low-priority interrupt handling
Handling of low-priority interrupt is suspended. Processing transitions to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

Figure 1.20 SH7044 Multiple Interrupt Sequence

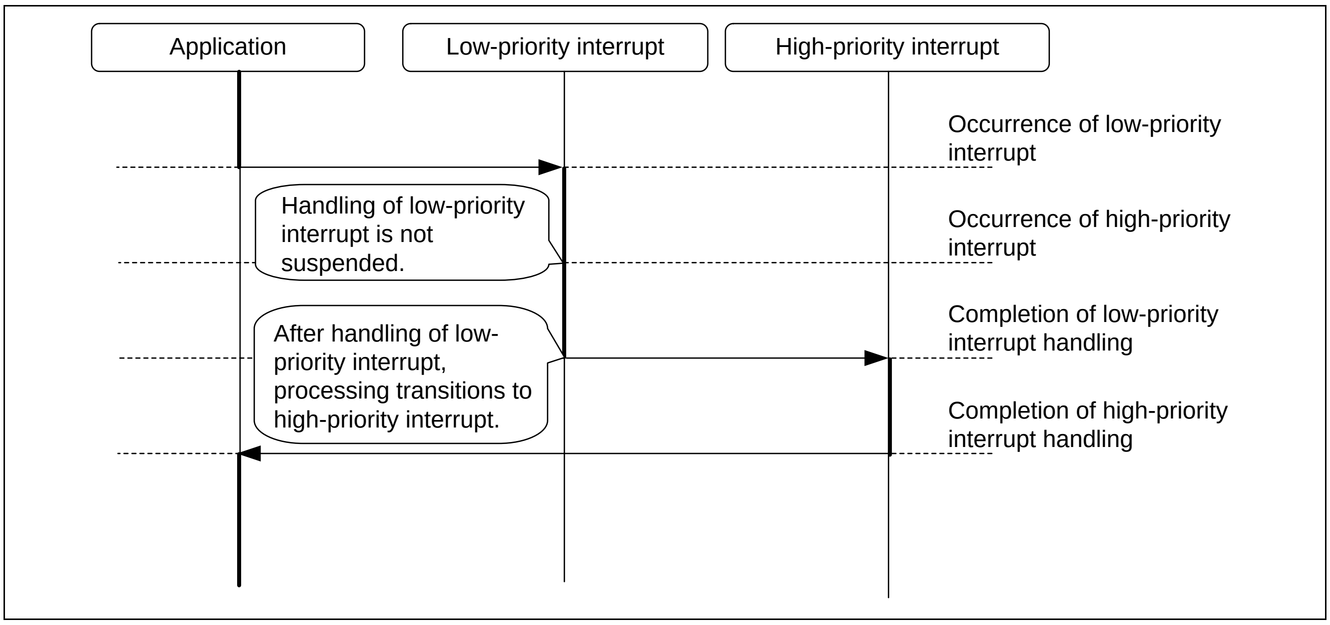

Application
Low-priority interrupt
High-priority interrupt
Occurrence of low-priority interrupt
Handling of low-priority interrupt is not suspended.
Occurrence of high-priority interrupt
After handling of low-priority interrupt, processing transitions to high-priority interrupt.
Completion of low-priority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

Figure 1.21 RX631 Interrupt Sequence (Not Controlled by PSW.I Bit)

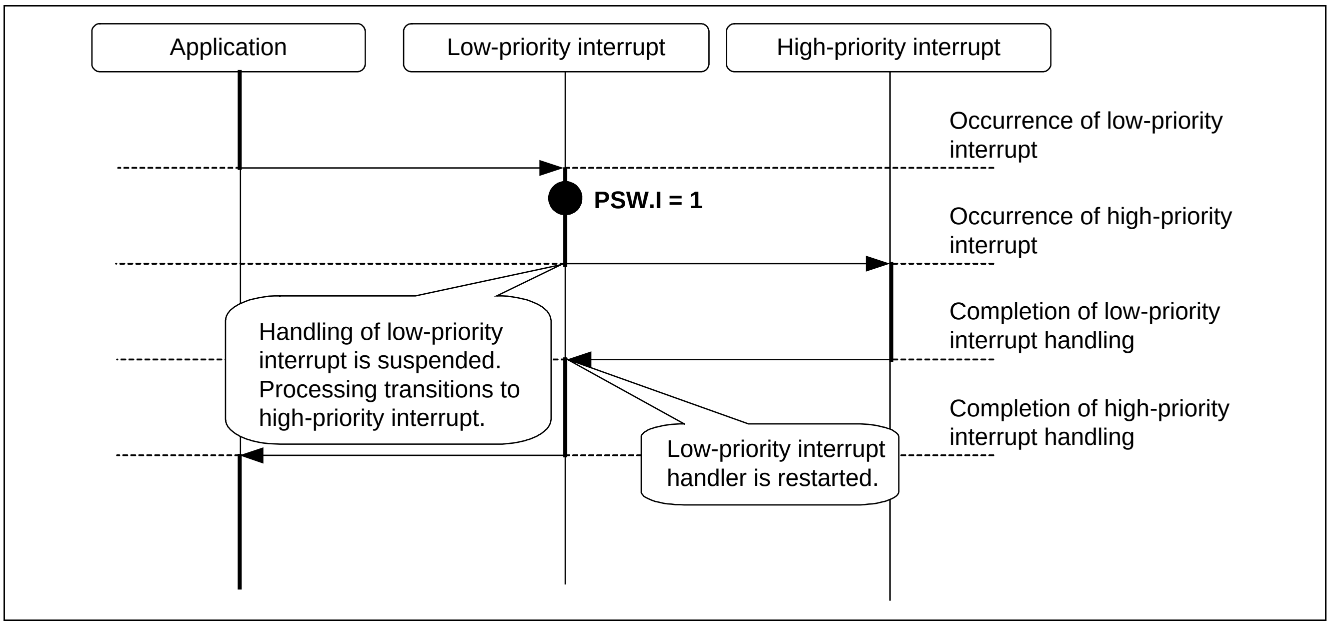

Application
Low-priority interrupt
High-priority interrupt
Occurrence of low-priority interrupt
PSW.I = 1
Occurrence of high-priority interrupt
Handling of low-priority interrupt is suspended. Processing transitions to high-priority interrupt.
Completion of low-priority interrupt handling
Low-priority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

Figure 1.22 RX631 Interrupt Sequence (Controlled by PSW.I Bit)

1.8.6 Unit Selection Function

As shown in figure 1.23, among the interrupts on the RX631, some of the interrupt sources of the MTU and TPU are assigned to common vectors. When using the unit selection function, it is necessary to select the interrupt source by means of a selector (register).

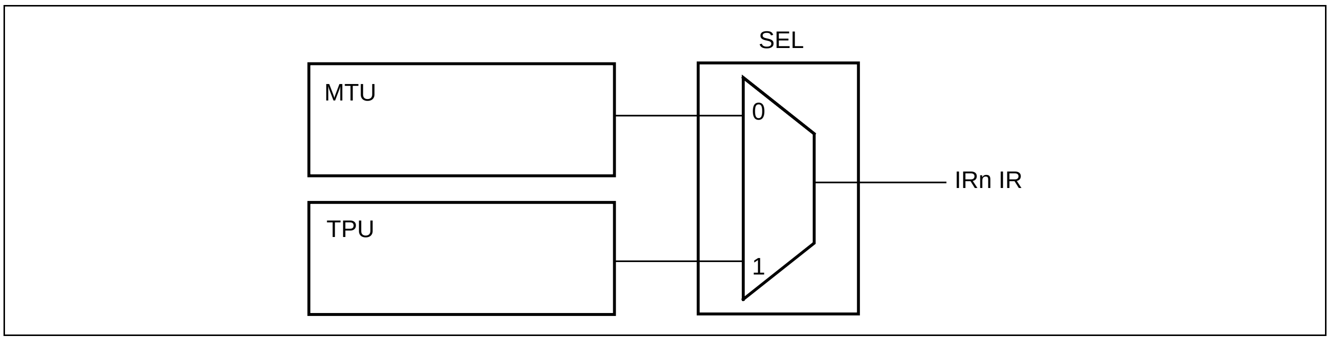


Figure 1.23 Unit Selection Function

1.8.7 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.

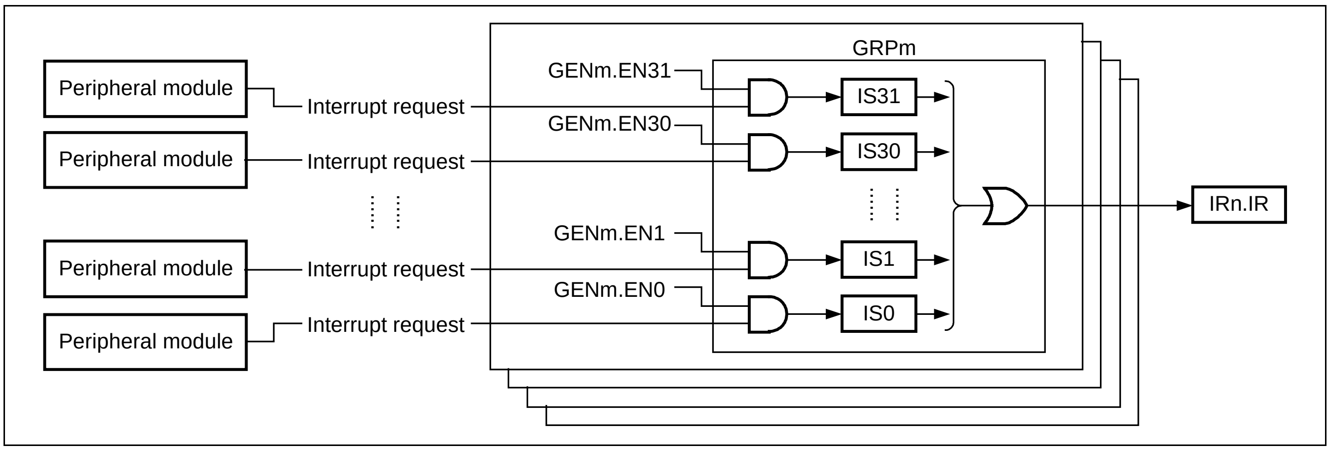


Figure 1.24 Group Interrupts

2. On-Chip Functions

2.1 List of On-Chip Functions

Table 2.1 List of Peripheral Functions

SH7044	**RX631**
Clock oscillator (CPG)	Clock generation circuit
User break controller (UBC)	—
Data transfer controller (DTC)	Data transfer controller (DTCa)
Bus state controller (BSC)	Bus controller (BSC)
Direct memory access controller (DMAC)	DMA controller (DMACA) EXDMA controller (EXDMACa)
Multifunction timer pulse unit (MTU)	Multifunction timer pulse unit 2 (MTU2a)
Watchdog timer (WDT)	Watchdog timer (WDTA) Independent watchdog timer (IWDTa)
Serial communication interface (SCI)	Serial communication interfaces (SCIc and SCId)
High-speed A/D converter (other than A mask) Mid-speed A/D converter (A mask)	12-bit A/D converter (S12ADa) 10-bit A/D converter (ADb)
Compare match timer (CMT)	Compare match timer (CMT)
Pin function controller (PFC)	Multi-function pin controller (MPC)
I/O ports (I/O)	I/O port
Flash memory (256 KB)*[1]	Flash memory*[2]
RAM (4 KB)	RAM (maximum 256 KB)
Low power consumption function	Low power consumption function
—	Voltage detection circuit (LVDA)
	Frequency measurement circuit
	Battery backup function
	Register write protection function
	Memory protection unit (MPU)
	Port output enable 2 (POE2a)
	16-bit timer pulse unit (TPUa)
	Programmable pulse generator (PPG)
	8-bit timer (TMR)
	Realtime clock (RTCa)
	Ethernet controller (ETHERC)
	Ethernet controller direct memory access controller (EDMAC)
	USB 2.0 Host/Function module (USBa)
	I^{2}C bus interface (RIIC)
	CAN module (CAN)
	Serial peripheral interface (RSPI)
	IEBus™ controller (IEB)
	CRC calculator (CRC)
	D/A converter (DAa)
	Parallel data capture unit (PDC)
	Temperature sensor
	Boundary scan

Notes: 1. Some versions of the SH7044 have on-chip mask ROM.
2. The RX631 group has up to 2 KB of on-chip flash memory (ROM) for storing code and up to 32 KB of on-chip flash memory for storing data (E2 data flash). There are also ROM-less versions of the RX631. For details, see the User's Manual: Hardware.

2.2 I/O Ports

2.2.1 Number of I/O Ports

Table 2.2 Number of I/O Ports on SH7044 and RX631

Item	Package	Port Function
Number of I/O ports on SH7044	QFP-112	I/O: 74 Input: 8 Total: 82
Number of I/O ports on RX631	TFLGA-177 LFBGA -176 LQFP -176	I/O: 133 Input: 1 Pull-up resistor: 133 Open-drain output: 133 5 V tolerant: 18
	TFLGA-145 LQFP-144	I/O: 111 Input: 1 Pull-up resistor: 111 Open-drain output: 111 5 V tolerant: 18
	TFLGA-100 LQFP-100	I/O: 78 Input: 1 Pull-up resistor: 78 Open-drain output: 78 5 V tolerant: 17
	TFLGA-64 LQFP-64	I/O: 42 Input: 1 Pull-up resistor: 42 Open-drain output: 42 5 V tolerant: 23 8-bit port switching function
	LQFP-48	I/O: 30 Input: 1 Pull-up resistor: 30 Open-drain output: 30 5 V tolerant: 18 8-bit port switching function

2.2.2 I/O Settings

Both the SH7044 and RX631 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 SH7044 port functions are determined by settings made to the pin function controller (PFC). The I/O ports range from A to F, and with the exception of port F, which is input-only, each port can be assigned to either general I/O or an on-chip module function. Ports A to E are assigned to either general I/O or an on-chip module function the making settings in registers PnIOR and PnCR (n: port A to E). The general concept of I/O settings on the SH7044 and the functions of the various registers are described below.

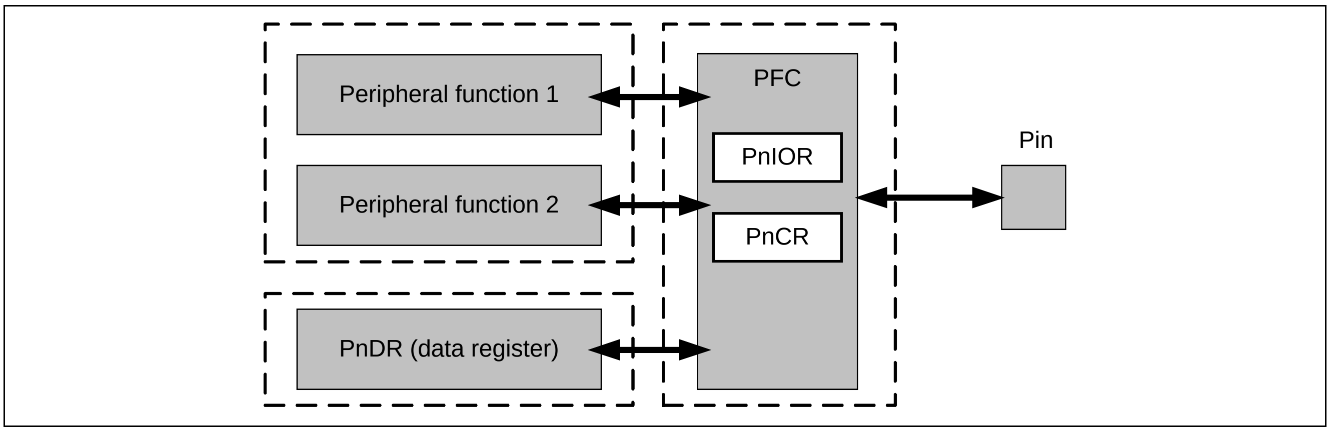


Figure 2.1 SH7044 I/O Settings

Table 2.3 Register Configuration on SH7044 for I/O Ports and Pin Function Controller

Module	Name	Function Name	Function
I/O port	PnDR	Port n data register	Port n data register
PFC	PnIOR	Port n IO register	Selects the port n I/O direction.
	PnCR	Port n control register	Selects the pin function.
	IFCR	IRQ function control register	Specifies the IRQ output pin state.

Note that the functions that can be assigned to pins and the functions that can be specified by the PFC differ according to the SH7044's operation mode (microcontroller mode 0, 1, or 2, or single-chip mode).

The RX631 is provided with I/O ports 0 to 9, A to G, and J, and the configuration of the registers corresponding to these I/O ports is shown below. The port I/O registers include dedicated input and dedicated output registers.

The following types of I/O port settings are supported on the RX631.

- 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.

As on the SH7044, the pins are multiplexed, so it is necessary to make pin function settings in the I/O port module and the multi-function pin controller (MPC).

I/O settings on the RX631 are described below.

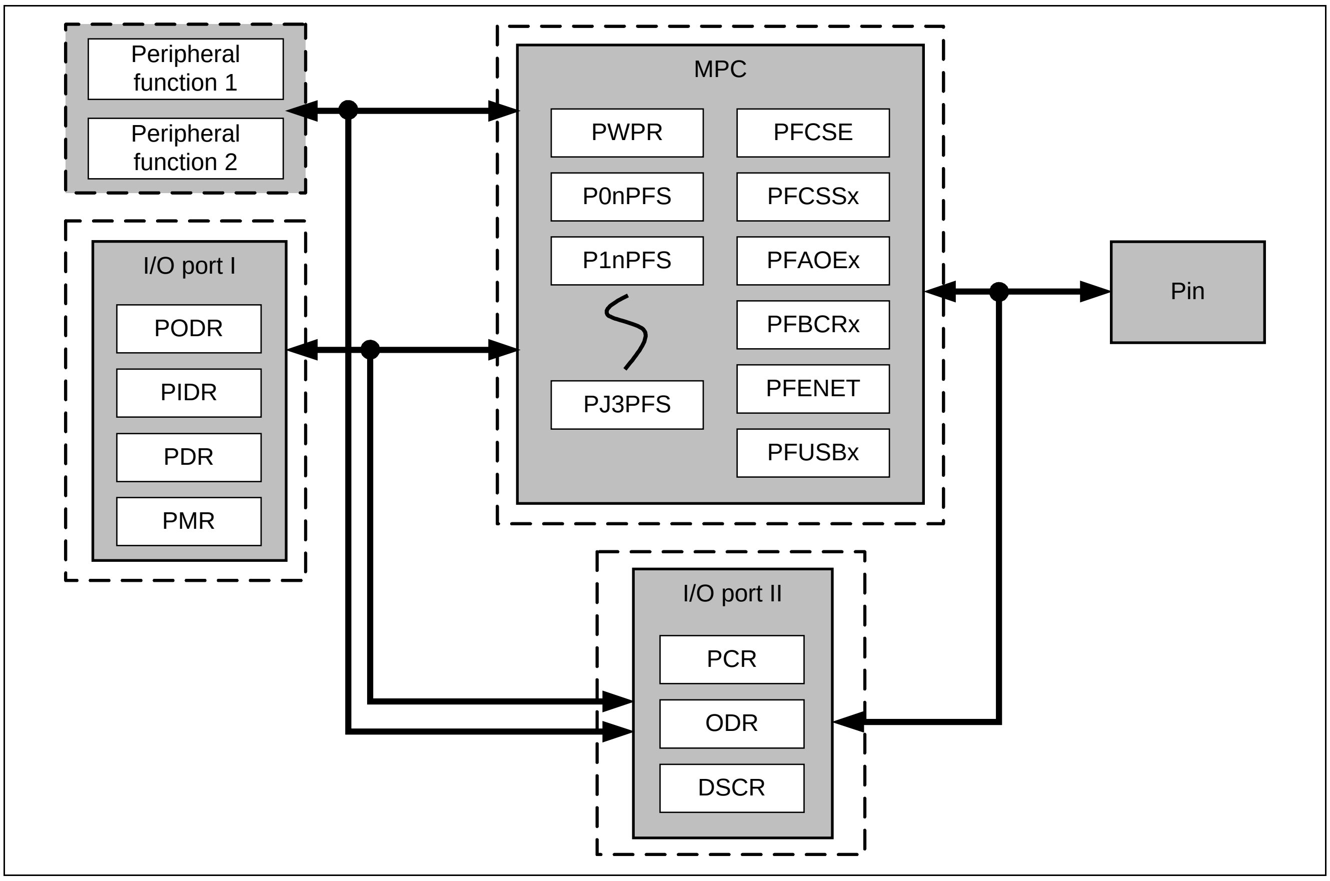


Figure 2.2 I/O Settings on the RX631

To use a pin for general I/O, it is sufficient to make settings in the I/O port registers (settings in PMR, PDR, ODR, PCR, and DSCR). Table 2.4 lists the registers in which the settings are made. Figure 2.3 is a flowchart of the setting procedure.

To use a pin for a peripheral function, the pin must be assigned to the peripheral function in the pin function control register (PxnPFS) in the MPC. Tables 2.4 and 2.5 list the registers in which the settings are made. Figure 2.4 is a flowchart of the setting procedure.

For example settings for use with peripheral functions that include general I/O, see the section describing the specific peripheral function.

Table 2.4 RX631 I/O Port Register Configuration

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
PSRA	Port switching register A	Dedicated register for 64-pin packages Selects between PB6, PB7 and PC0, PC1 general I/O function.
PSRB	Port switching register B	Dedicated register for 48-pin packages Selects between PB0, PB1, PB3, and PB5 and PC0, PC1, PC2, and PC3 general I/O function.

Table 2.5 RX631 Multi-Function Pin Controller Registers

Register	Function Name	Function
PWPR	Write-protect register	Write-protect function for PxxPFS register xx: 0n to 9n, An to Gn, J3
P0nPFS	P0n pin function control register	Register for selecting the pin function (port 0 pin function selection)
P1nPFS	P1n pin function control register	Register for selecting the pin function (port 1 pin function selection)
P2nPFS	P2n pin function control register	Register for selecting the pin function (port 2 pin function selection)
⁓	⁓	⁓
PFnPFS	PFn pin function control register	Register for selecting the pin function (port F pin function selection)
PJ3PFS	PJ3 pin function control register	Register for selecting the pin function (port J pin function selection)
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
PFENET	Ethernet control register	Ethernet mode setting (PMII or MII)
PFUSB0	USB0 control register	USB0 pin function settings
PFUSB1	USB1 control register	USB1 pin function settings

The initialization sequence when using RX631 I/O ports for general I/O is shown below.

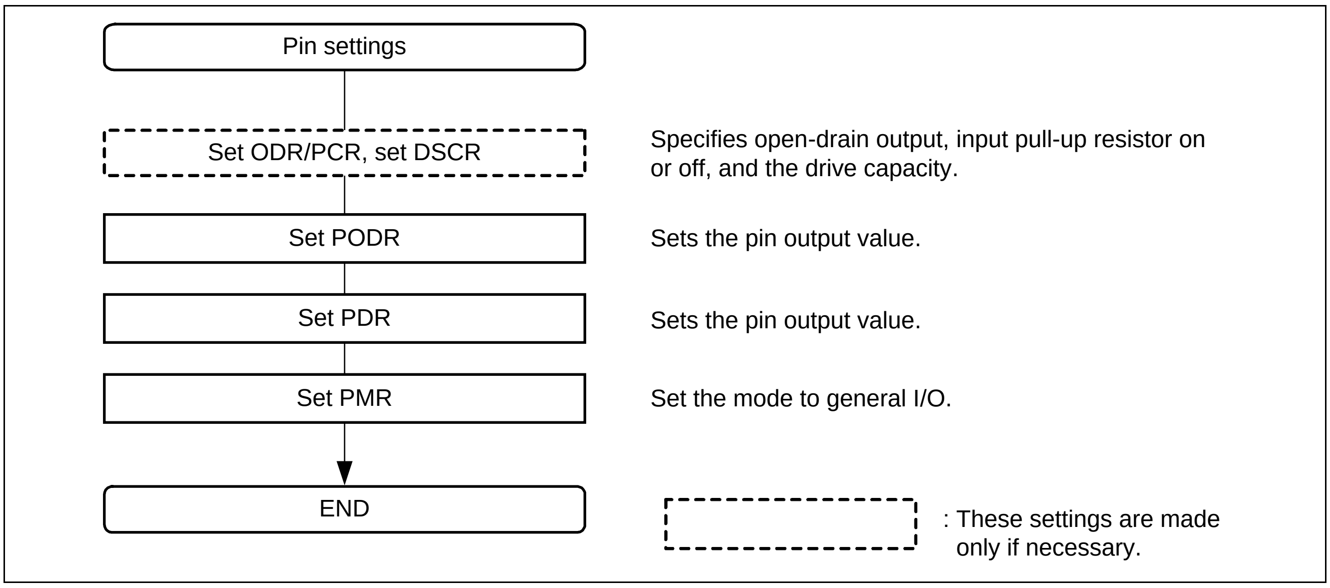

Pin settings
Set ODR/PCR, set DSCR
Specifies open-drain output, input pull-up resistor on or off, and the drive capacity.
Set PODR
Sets the pin output value.
Set PDR
Sets the pin output value.
Set PMR
Set the mode to general I/O.
END
: These settings are made only if necessary.

Figure 2.3 Using RX631 I/O Ports for General I/O

The initialization sequence when assigning pin functions to RX631 I/O ports is shown below.

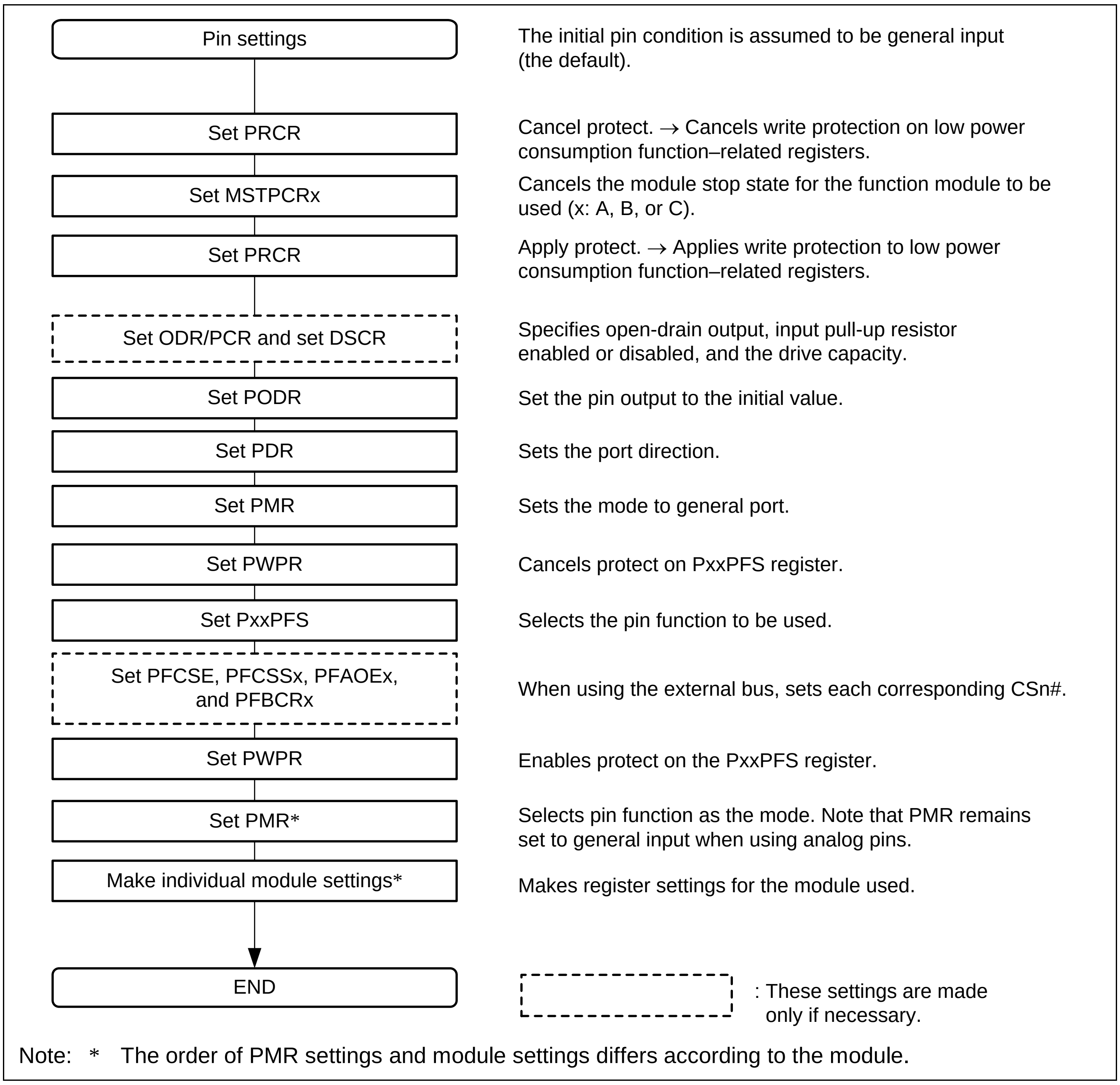

Pin settings
The initial pin condition is assumed to be general input (the default).
Set PRCR
Cancel protect. → Cancels write protection on low power consumption function–related registers.
Set MSTPCRx
Cancels the module stop state for the function module to be used (x: A, B, or C).
Set PRCR
Apply protect. → Applies write protection to low power consumption function–related registers.
Set ODR/PCR and set DSCR
Specifies open-drain output, input pull-up resistor enabled or disabled, and the drive capacity.
Set PODR
Set the pin output to the initial value.
Set PDR
Sets the port direction.
Set PMR
Sets the mode to general port.
Set PWPR
Cancels protect on PxxPFS register.
Set PxxPFS
Selects the pin function to be used.
Set PFCSE, PFCSSx, PFAOEx, and PFBCRx
When using the external bus, sets each corresponding CSn#.
Set PWPR
Enables protect on the PxxPFS register.
Set PMR*
Selects pin function as the mode. Note that PMR remains set to general input when using analog pins.
Make individual module settings*
Makes register settings for the module used.
END
: These settings are made only if necessary.
Note: * The order of PMR settings and module settings differs according to the module.

Figure 2.4 Assigning Pin Functions to RX631 I/O Ports

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

On the RX631 the individual modules are in the stopped state*[1] by default. Therefore, it is necessary to cancel module stop with the module stop control register (MSTPCRx) of the low power consumption function before making peripheral function settings. In addition, write protection has been applied to MSTPCRx by the register write protection function. Thus, to overwrite MSTPCRx it is necessary to first make it writeable by using the protect register (PRCR).

Note: 1. The DMAC, DTC, and RAM are in the operable state by default.

2.2.3 General I/O

General I/O port setting examples for the SH7044 and RX631 are shown below.

Table 2.6 shows an example of using PB2 on the SH7044, and P34 on the RX631, for general input.

Table 2.6 Pin Settings for General Input

Procedure		SH7044 Setting Example	RX631 Setting Example
1	Set the pin I/O direction to input.	PBIOR.PB2IOR = 0	PORT3.PDR.B4 = 0

2	Set general pins as general ports.	PBCR2.PB2MD1 = 0
		PBCR2.PB2MD0 = 0
			PORT3.PMR.B4 = 0

Table 2.7 shows an example of using PB2 on the SH7044, and P34 on the RX631, for general output. The output value is 1.

Table 2.7 Pin Settings for General Output

Procedure		SH7044 Setting Example	RX631 Setting Example
1	Set the pin to output.	PBDR.PB2DR = 1	PORT3.PODR.B4 = 1
2	Set the pin I/O direction to output.	PBIOR.PB2IOR = 1	PORT3.PDR.B4 = 1

3	Set pins as general ports.	PBCR2.PB2MD1 = 0
		PBCR2.PB2MD0 = 0
			PORT3.PMR.B4 = 0

2.3 Buses

This section describes the points of difference between the bus specifications of the two microcontrollers.

2.3.1 Comparison of Specifications

The main differences between the buses of the SH7044 and RX631 are shown below.

Table 2.8 SH7044 and RX631 Bus Comparison

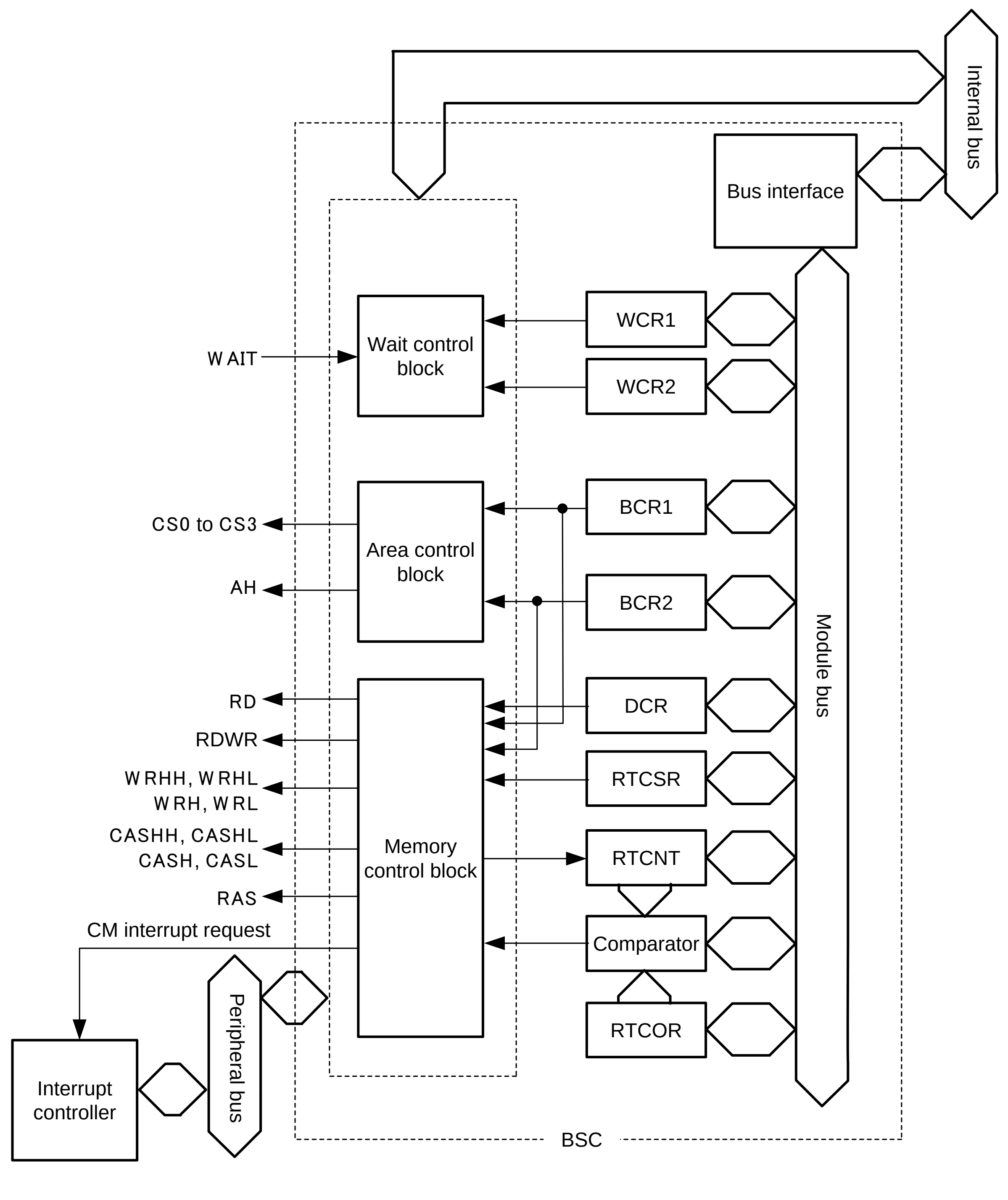
Item	SH7044	RX631
External bus address space	• External address spaces CS0 to CS3 (4 MB each) Notes: 1. CS0 is 2 MB when on-chip ROM is enabled. 2. 4 MB in on-chip ROM disabled mode.	• External address spaces CS0 to CS7 (16 MB × 8)
DRAM/SDRAM dedicated space	DRAM space (maximum 16 MB)	SDRAM space (maximum 128 MB)
Bus width	Settable to 8 or 16 bits by area.	Settable to 8, 16, or 32 bits by area.
Endianness	Big-endian (fixed)	The endianness can be set independently for each area.*
Bus arbitration	• CPU bus and external bus have fixed priority.	• External bus: Priority selectable from the following: 1) fixed priority, 2) toggle priority • Internal bus: fixed
Other access control	• Output of _RAS and _CAS signals for DRAM • Ability to generate a RAS precharge time assurance Tp cycle • DRAM burst access function • Ability to specify the DRAM refresh interval • Ability to insert wait cycles using an external _WAIT signal • Ability to access address data multiplexed I/O devices	CS area • Ability to insert recovery cycles • Cycle wait function • CSn# signal timing setting • RD# and WR# signal timing control • 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: * See 1.2.2.

2.3.2 Bus Configuration

The bus configurations of the SH7044 and RX631 are compared below.

The configuration of the SH7044's bus state controller is shown below.

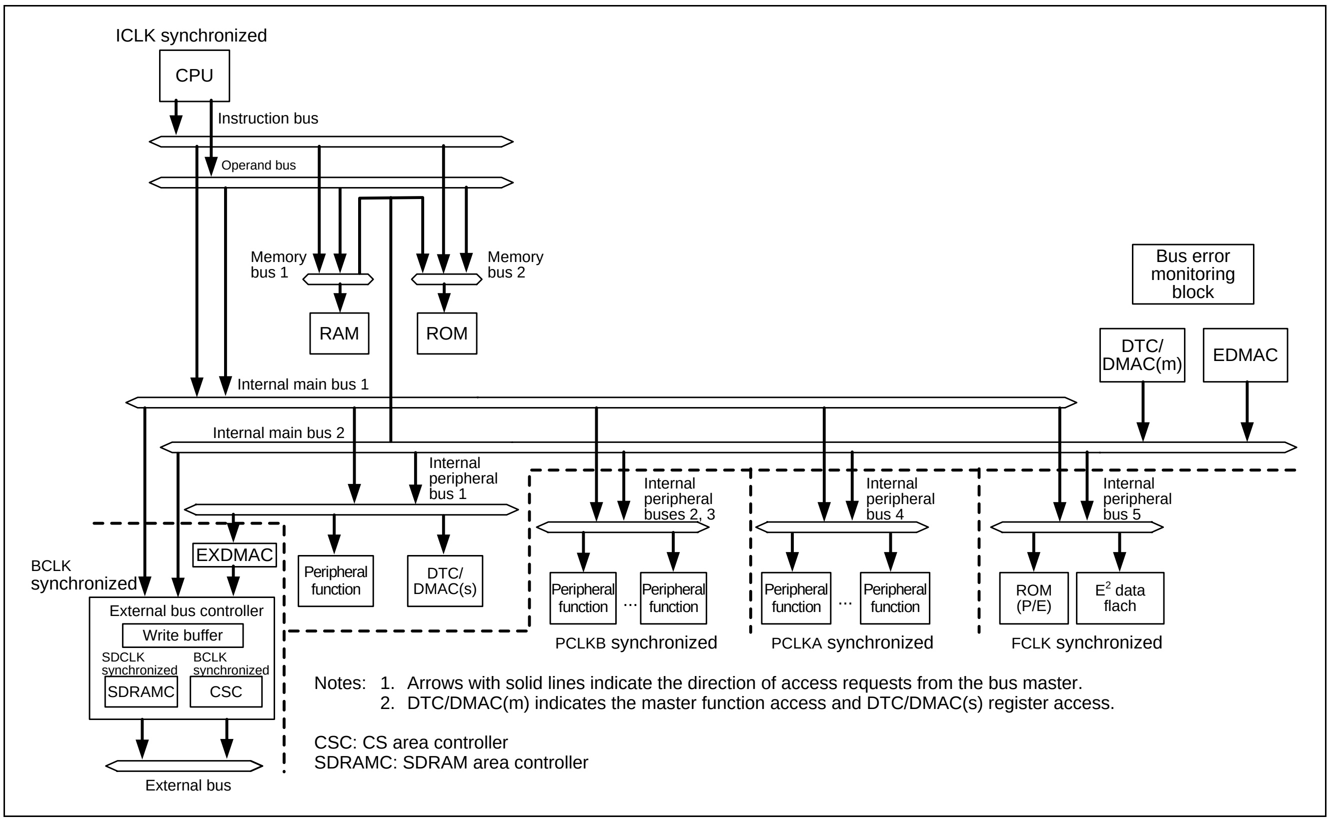


WCR1: Wait control register 1
WCR2: Wait control register 2
BCR1: Bus control register 1
BCR2: Bus control register 2

DCR: DRAM area control register
RTCSR: Refresh timer control/status register
RTCNT: Refresh timer counter
RTCOR: Refresh time constant register

Figure 2.5 SH7044 Bus State Controller Configuration

The bus configurations of the RX631 is shown below.



Notes: 1. Arrows with solid lines indicate the direction of access requests from the bus master.
2. DTC/DMAC(m) indicates the master function access and DTC/DMAC(s) register access.

CSC: CS area controller
SDRAMC: SDRAM area controller

Figure 2.6 RX631 Bus Configuration

The bus types on the RX631 are listed below. The RX631 has a different bus architecture than the SH7044, 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 2.9 RX631 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	On-chip ROM	ICLK
Internal main bus 1	CPU	ICLK
Internal main bus 2	DTC, DMAC, EDMAC	ICLK
Internal peripheral bus 1	DTC, DMAC, EXDMAC, interrupt controller, bus error monitoring block	ICLK (EXDMA: PCLKB)
Internal peripheral bus 2	Peripheral functions (peripheral functions other than those connected to peripheral buses 1, 3, 4, 5, and 6)	PCLKB
Internal peripheral bus 3	USB	PLCKB
Internal peripheral bus 4	EDMAC, ETHERC	PLCKA
Internal peripheral bus 5	Normally reserved area	—
Internal peripheral bus 6	ROM (P/E), E2 data flash	FCLK
External buses (CS areas)	External devices	BCLK
External buses (SDRAM)	SDRAM	SDCLK

ICLK: System clock PCLKA: Peripheral clock A PCLKB: Peripheral clock B
FCLK: FlashIF clock BCLK: External bus clock SDCLK: SDRAM clock

2.3.3 External Bus Interface Setting Examples

Refer to the following application note for external bus interface setting examples.

RX63N Group, RX631 Group Read/Write Operations in 16-Bit SDRAM Using the SDRAMC (R01AN1705EJ)

2.4 Interrupt Controller

2.4.1 IRQ Usage Example

A setting example using IRQ3 is shown below. PB5 is used as the IRQ3 input pin on the SH7044. P33 is used as the IRQ3 input pin on the RX631.

Table 2.10 Interrupt Initial Setting Example (IRQ3 Settings)

Procedure		SH7044	RX631

1	Make I/O port settings.	PBIOR.PB5IOR = 0
		(general input pin setting)
		PBCR2.PB5MD1, PBCR2.PB5MD0 = 01b
		(IRQ3 interrupt input pin)
			PORT3.PDR.B3 = 0
			(P33 input setting)
			PORT3.PMR.B3 = 0
			(P33 GPIO setting)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enabled)
			MPC.P33PFS.ISEL = 1
			(interrupt function setting IRQ3-DS)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1

2	Make interrupt controller settings.	ICR.IRQ3S = 1
		(IRQ detection: Falling edge)
		IPRA = 0x000F
		(bits 3 to 0: interrupt level 15)
			IRQCR3.IRQMD = 1
			(IRQ detection: Falling edge)
			IRQFLTE0.FLTEN3 = 1
			(IRQ3 digital noise filter enabled)
			IRQFLTC0.FCLKSEL3 = 3;
			(sampling PCLK/64)
			IR067 = 0 (interrupt flag cleared)
			IER08.IEN3 = 1 (IRQ3 enabled)
			IPR067 = 15 (interrupt level 15)

2.5 Data Transfer Controller (DTC)

2.5.1 Comparison of Specifications

On both the SH7044 and RX631 the transfer information is located in RAM, and DTC vectors are used to specify transfer information. The basic operation of the three transfer modes (normal transfer mode, repeat transfer mode, and block transfer mode) is the same on both microcontrollers. The DTC specifications of the SH7044 and RX631 are listed below.

Table 2.11 Comparison of DLC Specifications on SH7044 and RX631

Item	SH7044	RX631
Transfer modes	• Normal transfer mode • Repeat transfer mode • Block transfer mode	• Normal transfer mode • Repeat transfer mode • Block transfer mode
Activation sources	• External interrupt • Peripheral function interrupt • Software trigger	• External interrupt • Peripheral function interrupt • Software trigger
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 and DTC) • External memory space • Memory-mapped external device space Note: 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 (excluding DMAC and DTC) • External memory space • Memory-mapped external device space Note: One of the specified areas must be in the on-chip memory space or on-chip peripheral module space.
Transfer units	May be specified as 8, 16, or 32 bits. Block size: May be specified within range of 0 to 65,535.	1 data unit: May be specified as 8, 16, or 32 bits. 1 block: May be specified within range of 1 to 256 data units.
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	• Chain transfer • 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

2.5.2 Register Configuration

The register configuration of the DTC is shown below.

Table 2.12 List of DTC Registers on SH7044 and RX631

Item		SH7044	RX631
Transfer mode selection		DTC mode register (DTMR) DTC mode 1, 0 (MD1 or MD0)	DTC mode register A (MRA) DTC transfer mode select bits
Selection of repeat area or block area as transfer destination or transfer source		DTC mode register (DTMR) DTC transfer mode select (DTS)	DTC mode register B (MRB) DTC transfer mode select bits
Data transfer size selection		DTC mode register (DTMR) DTC data transfer size 1 or 0 (SZ1 or SZ0)	DTC mode register A (MRA) DTC data transfer size bits
Transfer source: Address state after transfer		DTC mode register (DTMR) Source address mode 1 or 0 (SM1 or SM0)	DTC mode register A (MRA) Transfer source address addressing mode bits
Transfer destination: Address state after transfer		DTC mode register (DTMR) Destination address mode 1 or 0 (DM1 or DM0)	DTC mode register B (MRB) Transfer destination address addressing mode bits
Chain transfer selection	Transfer-end/ continue, enable/ disable	DTC mode register (DTMR) DTC chain enable (CHNE)	DTC mode register B (MRB) DTC chain transfer enable bit (CHNE)
	Continuous transfer/ transfer at change of transfer counter	—	DTC mode register B (MRB) DTC chain transfer select bit (CHNE)
Interrupt request enable/ disable		DTC mode register (DTMR) DTC interrupt select (DISEL)	DTC mode register B (MRB) DTC interrupt select bit (DISEL)
DTC transfer suspend/ resume by NMI		DTC mode register (DTMR) DTCNMI mode (NMIM)*[1]	—
Transfer source address		DTC source address register (DTSAR)	DTC transfer source register (SAR)
Transfer destination address		DTC destination address register (DTDAR)	DTC transfer destination register (DAR)
Initial address		DTC initial address register (DTIAR)*[2]	—
Transfer count specification		DTC transfer count register A (DTCRA) Specifies the transfer count.	DTC transfer count register A (CRA) Specifies the transfer count.
Block transfer mode	Data unit transfer count	DTC transfer count register A (DTCRA) Specifies the block transfer count.	DTC transfer count register B (CRB) Specifies the block transfer count.
	Block length specification	DTC transfer count register B (DTCRB) Specifies the block length.	DTC transfer count register A (CRA) Specifies the block length.
DTC activation disable/enable		DTC enable register (DTER) DTC enable bit	DTC activation enable register (ICU.DTCERn)
DTC module operate/stop		—	DTC module start register (DTCST) DTC module start bit
Base address		DTC information base register (DTBR)*[3]	DTC vector base register (DTCVBR)
Full address mode/ Short address mode		—	DTC address mode register (DTCADMOD)
NMI interrupt generation enable/disable		DTC control/status register (DTCSR) NMI flag bit (NMIF)	Non-maskable interrupt status register (ICU.NMISR) NMI status flag

Item	**SH7044**	**RX631**
DTC activation by software enable/disable	DTC control/status register (DTCSR) DTC software activation enable bit (SWDTE)	Software interrupt activation register (ICU.SWINTR) Software interrupt activation bit (SWINT)
DTC vector address setting for DTC activation by software	DTC control/status register (DTCSR) Software activation vectors 7 to 0 (DTVEC7 to DTVEC0)	DTC status register (DTCSTS) VECN[7:0] bits (DTC-activating vector number monitoring bits)
Showing of DTC transfer operation state	—	DTC status register (DTCSTS) DTC active flag
Read skipping enable	—	DTC Control Register (DTCCR) DTC transfer information read skipping enable bit

Notes: 1. Not implemented on the RX631.
2. Initial address setting is necessary on the SH7044 but not on the RX631.
3. The information base register content on the SH7044 is included in the DTC vector base register address content on the RX631.

2.5.3 Transfer Modes

The differences in the operation of the transfer modes is described below.

Table 2.13 Normal Transfer Mode

Item	SH7044	RX631
Transfer size	1 byte, 1 word, or 1 longword	1 byte, 1 word, or 1 longword
Transfer count	1 to 65,536	1 to 65,536

Table 2.14 Repeat Transfer Mode (The method of specifying the repeat area differs.)

Item	SH7044	RX631
Transfer size	1 byte, 1 word, or 1 longword	1 byte, 1 word, or 1 longword
Transfer count	1 to 256	1 to 256
Repeat area specification method	The repeat mode and whether either the source or destination is the repeat area is specified in the mode register. The repeat address is specified in the repeat initial address register.	The concept of the repeat initial address does not apply, and the initial value of SAR or DAR is repeated.

Table 2.15 Block Transfer Mode (The way of conceptualizing the single block size differs.)

Item	SH7044	RX631
Transfer size	Transferring a single block	Transferring a single block
Single block size	1 to 65,536 bytes	1 to 256 data units The data unit can be byte, word, or longword.
Transfer count	1 to 65,536	1 to 65,536

2.5.4 Activation Source Setting

On the SH7044 activation sources of the DTC are set in the DTC enable registers (DTEA to DTEE). On the RX631 DTC activation sources are set in the DTC activation enable registers (DTCERn, n: vector number) of the interrupt controller, and this enables DTC activation by interrupts.

2.5.5 DTC Vector Configuration

The DTC vector configuration on the SH7044 and RX631 is shown below.

On the SH7044 the DTC vector table starts from the fixed address 400h. The upper 16 bits of the transfer information addresses are stored in the DTC information base register (DTBR), and the 16-bit address for each activation source is stored in the DTC vector table.

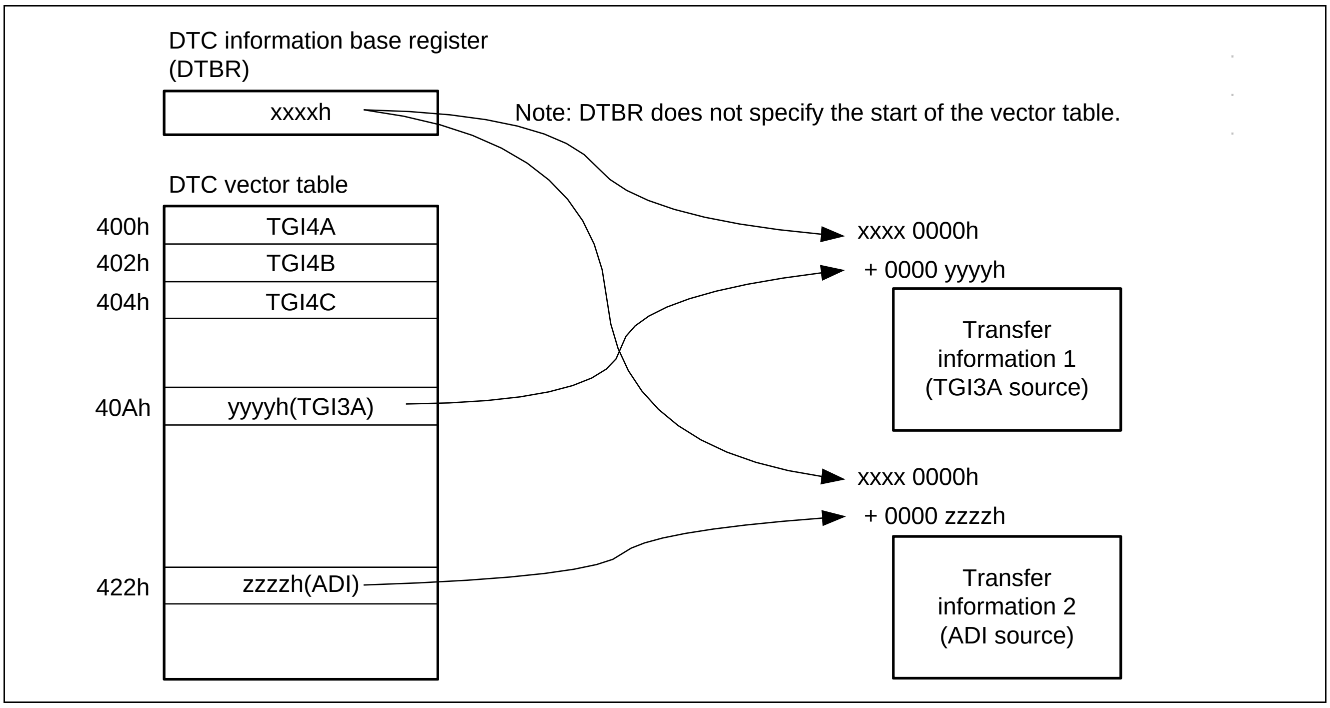

DTC information base register
(DTBR)
xxxxh
Note: DTBR does not specify the start of the vector table.
DTC vector table
400h
TGI4A
402h
TGI4B
404h
TGI4C
40Ah
yyyyh(TGI3A)
422h
zzzzh(ADI)
xxxx 0000h
+ 0000 yyyyh
Transfer information 1 (TGI3A source)
xxxx 0000h
+ 0000 zzzzh
Transfer information 2 (ADI source)

Figure 2.7 DTC Vector Configuration on SH7044

On the RX631 the vector table start address is specified by the DTC vector base register (DTVBR). Vectors can be set in 4 KB units within the range from 0000 0000h to 07FF F000h and from F800 0000h to FFFF F000h. As with interrupt vectors, the vectors are numbered 0 to 255, and a 32-bit transfer information address can be specified for each vector. In contrast to the SH7044's DTC vector table, which starts from the fixed address 400h, on the RX631 the start address can be set in the DTC vector base register, so there is more flexibility in specifying the DTC vector table area.

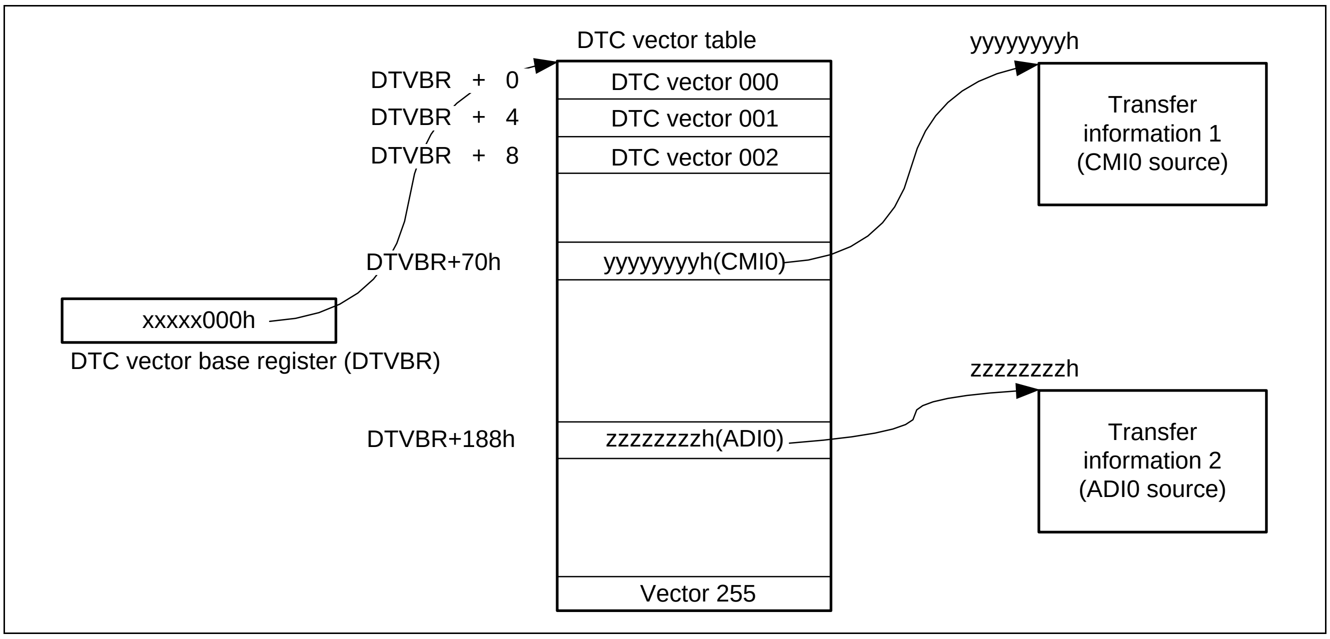
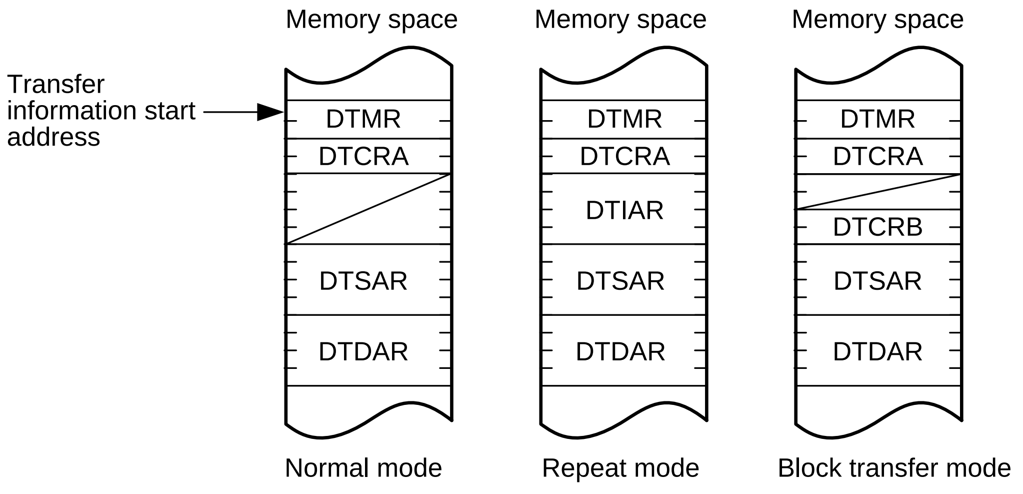


Figure 2.8 DTC Vector Configuration on RX631

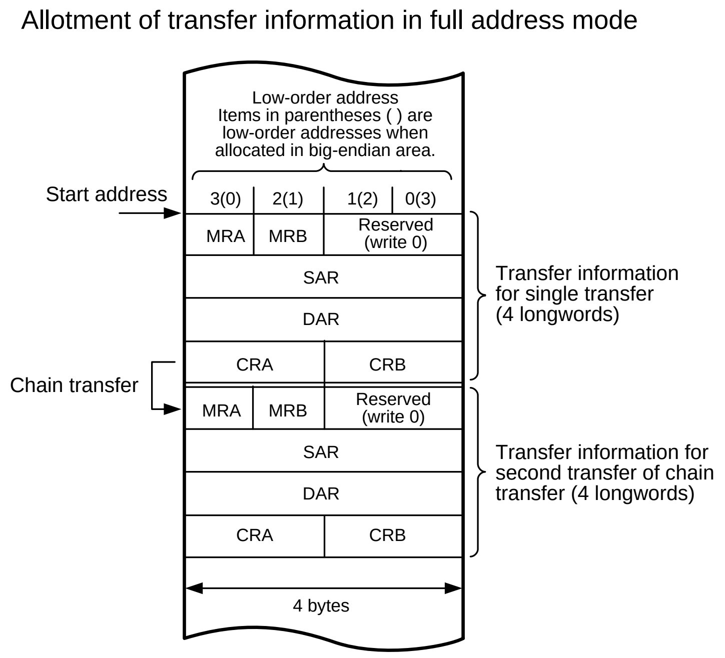
2.5.6 Allocation of Transfer Information

The format of transfer information differs between the SH7044 and the RX631.

On the SH7044 a different transfer information format is used for each transfer mode. On the RX631 all transfer modes use the same transfer information format. Note, however, that on the RX631 the DTC transfer information is affected by the endianness setting. The transfer information format of each mode on the SH7044 (a) and the full-address mode transfer information format on the RX631 (b) are shown below.



(a) SH7044 transfer information format


Allotment of transfer information in full address mode
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
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

(b) RX631 transfer information format in full address mode

Figure 2.9 DTC Transfer Information Formats on SH7044 and RX631

The RX631 supports a short-address mode in which addresses can be specified in 24 bits. The size of the transfer information is four longwords in full-address mode but only three longwords in short-address mode. This reduces the time it takes the DTC to read transfer information and enables it to start up faster. In addition, less RAM is needed to store the transfer information. The transfer information format in short-address mode is shown below.

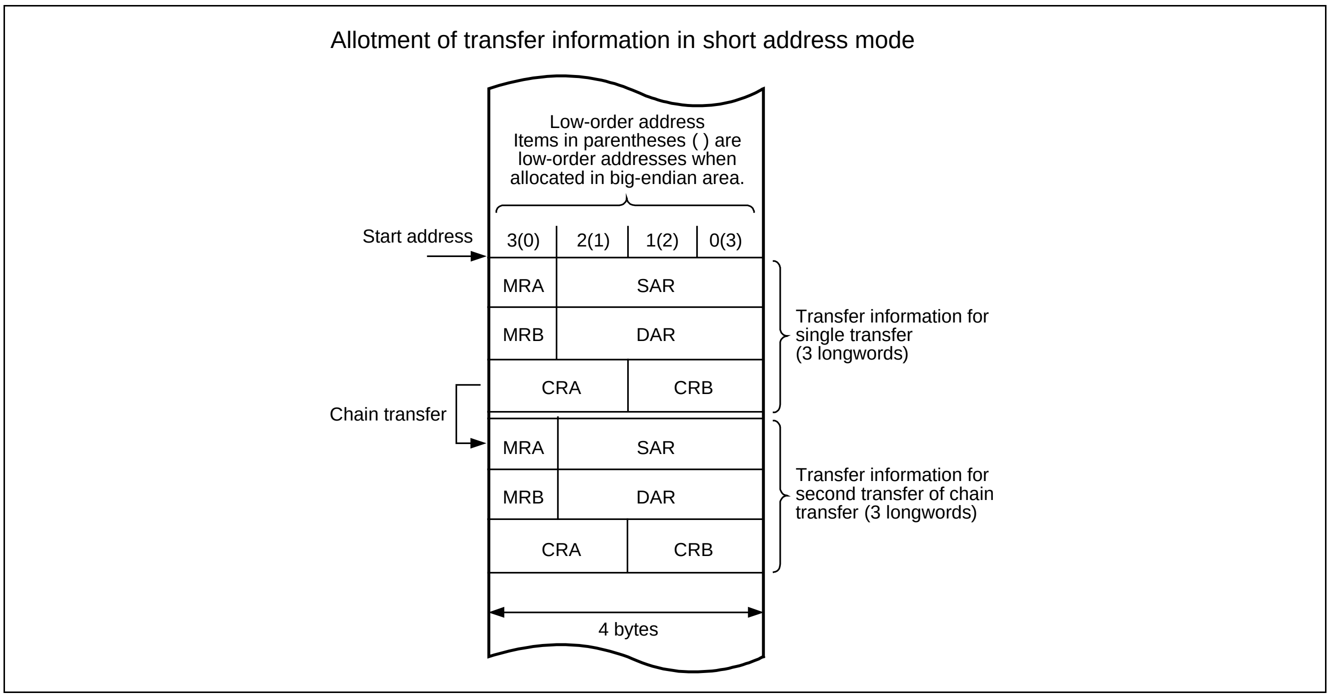

Allotment of transfer information in short address mode
Low-order address
Items in parentheses ( ) are low-order addresses when allocated in big-endian area.
Start address
3(0)
2(1)
1(2)
0(3)
MRA
SAR
MRB
DAR
CRA
CRB
Chain transfer
MRA
SAR
MRB
DAR
CRA
CRB
Transfer information for single transfer (3 longwords)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

Figure 2.10 RX631 DTC Transfer Information Format in Short-Address Mode

Short-address mode supports 16 megabytes of transfer space in address ranges 00000000h to 007FFFFFh and FF800000h to FFFFFFFFh (excluding reserved areas).

2.5.7 Module Stop

The initial state of the peripheral modules of the RX631 is stopped, due to the low power consumption function. However, the initial state of the DTC is operational, so there is no need to cancel the module stop state. Module stop can be applied to the DTC, but doing so also stops the DMAC because the same control bit in the module stop control register is used for both the DTC and the DMAC. (The EXDAMC and EDMAC are controlled separately.)

2.5.8 Data Transfer Controller (DTC) Setting Example (Repeat Transfer)

In the data transfer controller (DTC) setting example shown below for the SH7044 and RX631, the DTC is used to implement data transfer between the serial communication interface (SCI) and the on-chip RAM. Refer to 2.9.4 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+RX63N board is used, and the SCI transfer mode is asynchronous serial transfer.
2. When an SCI 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 SCI.
3. When an SCI 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.

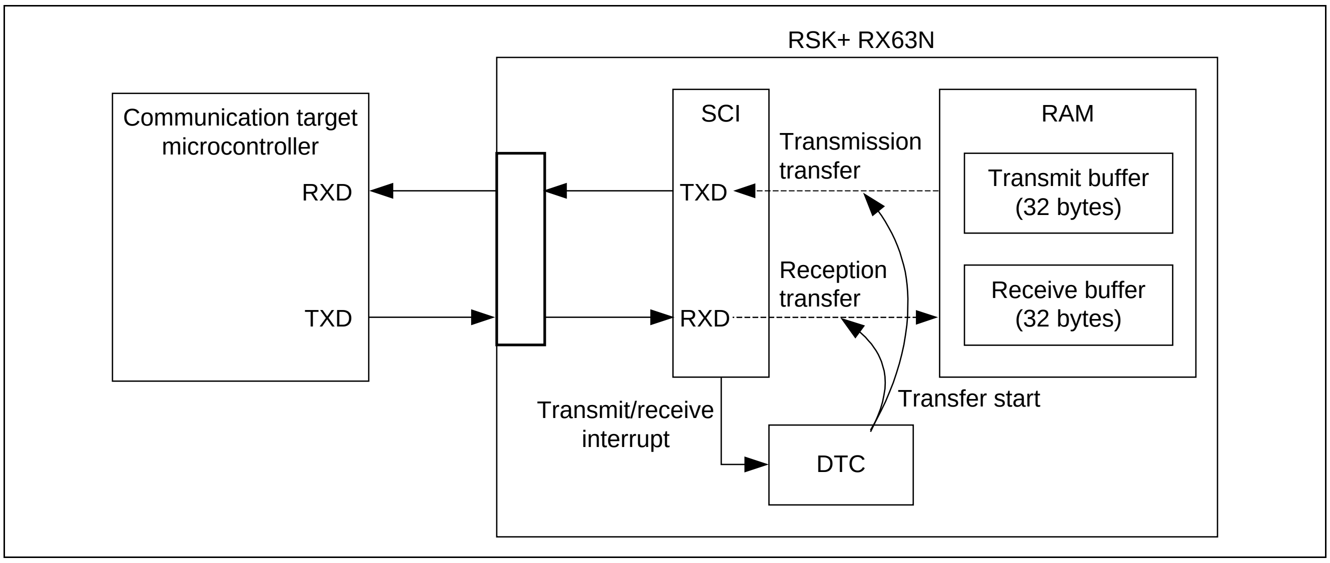

RSK+ RX63N
Communication target microcontroller
RXD
TXD
SCI
TXD
RXD
Transmission transfer
Reception transfer
RAM
Transmit buffer (32 bytes)
Receive buffer (32 bytes)
Transfer start
Transmit/receive interrupt
DTC

Figure 2.11 Example of Data Transfer between RAM and SCI Using DTC

Table 2.16 DTC Transfer Specifications

Item	**Transmit Transfer**	**Receive Transfer**
Transfer mode	Repeat mode	Repeat mode
Transfer count	32	32
Transfer size	Byte	Byte
Transfer source	On-chip RAM (transmit buffer)	Receive data register (SCI)
Transfer destination	Transmit data register (SCI)	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	SCI transmit data-empty interrupt	SCI 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

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.
   SH7044 #pragma address DTC_VECT_TABLE = 0x400 (address defined by user)
   volatile unsigned short DTC_VECT_TABLE[34];
   RX631 #pragma address DTC_VECT_TABLE = 0x00010000 (address defined by user)
   volatile unsigned long DTC_VECT_TABLE[256];

Table 2.17 DTC Normal Transfer Initial Setting Example

Procedure		SH7044 Setting Example	RX631 Setting Example

1	Make transfer information settings (transmitting side).	DTC_TX.DTMR.SM1, DTC_TX.DTMR.SM0 = 2 (increment transfer source)
		DTC_TX.DTMR.DM1, DTC_TX.DTMR.DM0 = 0 (fixed transfer destination address)
		DTC_TX.DTMR.MD1, DTC_TX.DTMR.MD0 = 1 (repeat transfer mode)
		DTC_TX.DTMR.SZ1, DTC_TX.DTMR.SZ0 = 0 (data size: byte)
		DTC_TX.DTMR.DTS = D.C
		DTC_TX.DTMR.CHNE = 0 (DTC data transfer-end)
		DTC_TX.DTMR.DISEL = 1 (interrupt enabled at data transfer-end)
		DTC_TX.DTSAR = transmit buffer start address
		DTC_TX.DTDAR = SCI.TDR address
		DTC_TX.DTCRAH = 32 (transfer count)
			DTC_TX.MRA.SM = 2 (increment transfer source)
			DTC_TX.MRA.SZ = 0 (data size: byte)
			DTC_TX.MRA.MD = 1 (repeat transfer mode)
			DTC_TX.MRB.DM = 0 (fixed transfer destination address)
			DTC_TX.MRB.DISEL = 0 (interrupt generation at data transfer-end)
			DTC_TX.MRB.CHNE = 0 (chain transfer disabled)
			DTC_TX.SAR = transmit buffer start address
			DTC_TX.DAR = SCI.TDR address
			DTC_TX.CRAH = 32 (transfer count)

2	Make transfer information settings (receiving side).	DTC_RX.DTMR.SM1, DTC_RX.DTMR.SM0 = 0 (fixed transfer source)
		DTC_RX.DTMR.DM1, DTC_RX.DTMR.DM0 = 2 (increment transfer destination)
		DTC_RX.DTMR.MD1, DTC_RX.DTMR.MD0 = 1 (repeat transfer mode)
		DTC_RX.DTMR.SZ1, DTC_RX.DTMR.SZ0 = 0 (data size: byte)
		DTC_RX.DTMR.DTS = D.C
		DTC_RX.DTMR.CHNE = 0 (DTC data transfer-end)
		DTC_RX.DTMR.DISEL = 1 (interrupt enabled at data transfer-end)
		DTC_RX.DTSAR = SCI.RDR address
		DTC_RX.DTDAR = receive buffer address
		DTC_RX.DTCRAH = 32 (transfer count)
			DTC_RX.MRA.SM = 0 (fixed transfer source address)
			DTC_RX.MRA.SZ = 0 (data size: byte)
			DTC_RX.MRA.MD = 1 (repeat transfer mode)
			DTC_RX.MRB.DM = 2 (increment transfer destination)
			DTC_RX.MRB.DISEL = 0 (interrupt generation at data transfer-end)
			DTC_RX.MRB.CHNE = 0 (chain transfer disabled)
			DTC_RX.SAR = SCI.RDR address
			DTC_RX.DAR = receive buffer address
			DTC_RX.CRAH = 32 (transfer count)

3	Make DTC vector table settings.	DTC_VECT_TABLE[29] = lower bits of DTC_RX address
		DTC_VECT_TABLE[30] = lower bits of DTC_TX address
		DTBR = upper bits of DTC_RX address
			DTC_VECT_TABLE[215] = DTC_TX address
			DTC_VECT_TABLE[214] = DTC_RX address
			DTC.DTCVBR = DTC vector address

4	Set address mode.	No address mode setting on SH7044	DTC.DTCADMOD = 0 (full address mode)
5	Initial address	DTC_RX. DTIAR = Initial address	—

6	Set activation sources.	DTED3 = 1 (DTC activated by SCI.RXI0)
		DTED2 = 1 (DTC activated by SCI.TXI0)
			DTCER214 = 1 (DTC activation by SCI receive interrupt)
			DTCER215 = 1 (DTC activation by SCI transmit interrupt)

7	Make SCI settings.	Make SCI asynchronous transfer settings. Make settings in table 2.41 for SCI function or ICU function. Enable TXI interrupts, RXI interrupts, and error interrupts. The DTC will not operate if interrupts are not enabled.	
8	Activate DTC module.	No module activation setting on SH7044	DTC.DTCST.DTCST = 1 (DTC module operating)

Note: When transmission of 32 bytes of data finishes, a transmit interrupt (TXI) is generated.
When reception of 32 bytes of data finishes, a receive interrupt (RXI) is generated.
The details of the handling of the above interrupts are not stipulated. The sample code implements end processing for the transmit and receive interrupts.

2.6 Direct Memory Access Controller (DMAC)

Direct memory access control functionality is implemented on the SH7044 by an on-chip DMAC and on the RX631 by an on-chip DMACA and by a dedicated on-chip EXDMACa for transfers between external areas. The internal bus configuration of the RX631 differs from that of SH microcontrollers. It supports independent data transfers by CPU instruction execution and by the DMAC or DTC for improved transfer performance.

2.6.1 Comparison of Specifications

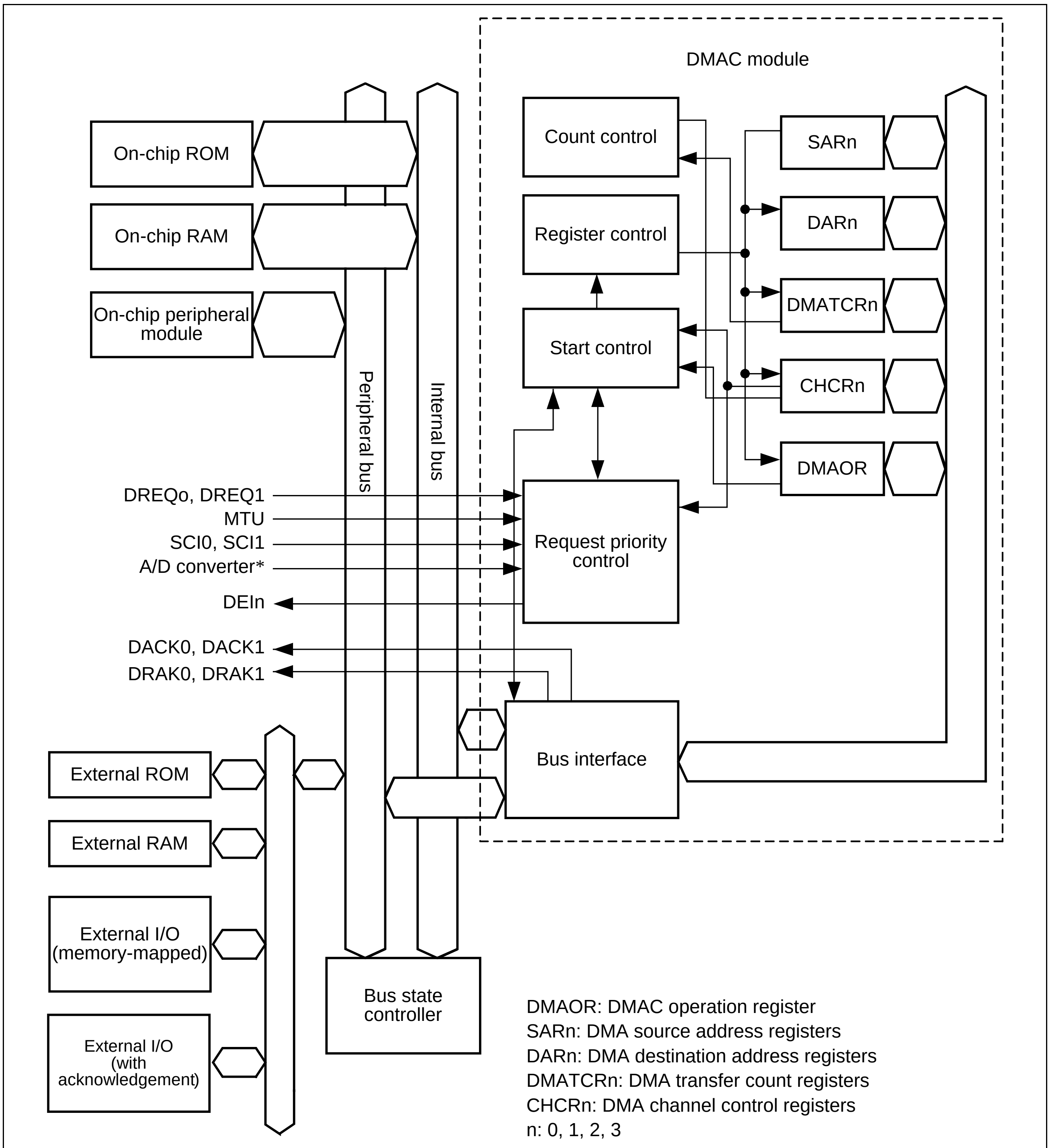
The functions and features of the SH7044 and RX631 are shown below.

Table 2.18 Comparison of SH7044 (DMAC) and RX631 (DMACA and EXDMACa) Functions

		SH7044	RX631	
Item		**DMAC**	**DMACA**	**EXDMACa**
Number of channels		4 channels	4 channels	2 channels
Maximum transfer count (maximum transfer data unit count on RX)		16 M (16,777,216)	1 M data units (block transfer mode max. total transfer count: 1,024 data units × 1,024 blocks)	1 M data units (block transfer mode max. total transfer count: 1,024 data units × 1,024 blocks)
DMA activation sources		• External request • On-chip module request • Auto request	• (External requests not supported.) • On-chip module request • Software trigger • Trigger input to external interrupt input pin	• External request • On-chip module request • Software trigger
Channel priority		Selectable between the following: • Fixed • Round robin	Fixed (channel 0 > channel 1 > channel 2 > channel 3)	Fixed (channel 0 > channel 1)
Transfer data	1 data unit	8 bits, 16 bits, 32 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 after completion of the number of transfers specified by the transfer counter.	Generated after completion of the number of transfers specified by the transfer counter.	
	Transfer escape-end interrupt	—	Generated after completion of data transfer equivalent to the repeat size or when the extended repeat area overflows.	
Other		• Source address reload function	• Extended repeat area function	• Extended repeat area function Direct data transfer to the TFTLCD panel is possible.

2.6.2 DMAC Block Diagram

A block diagram of the SH7044's DMAC is shown below.



DMAOR: DMAC operation register
SARn: DMA source address registers
DARn: DMA destination address registers
DMATCRn: DMA transfer count registers
CHCRn: DMA channel control registers
n: 0, 1, 2, 3

Note: * A/D on other than A mask products, A/D1 on A mask products.

Figure 2.12 SH7044 DMAC Block Diagram

A block diagram of the RX631's DMACA is shown below.

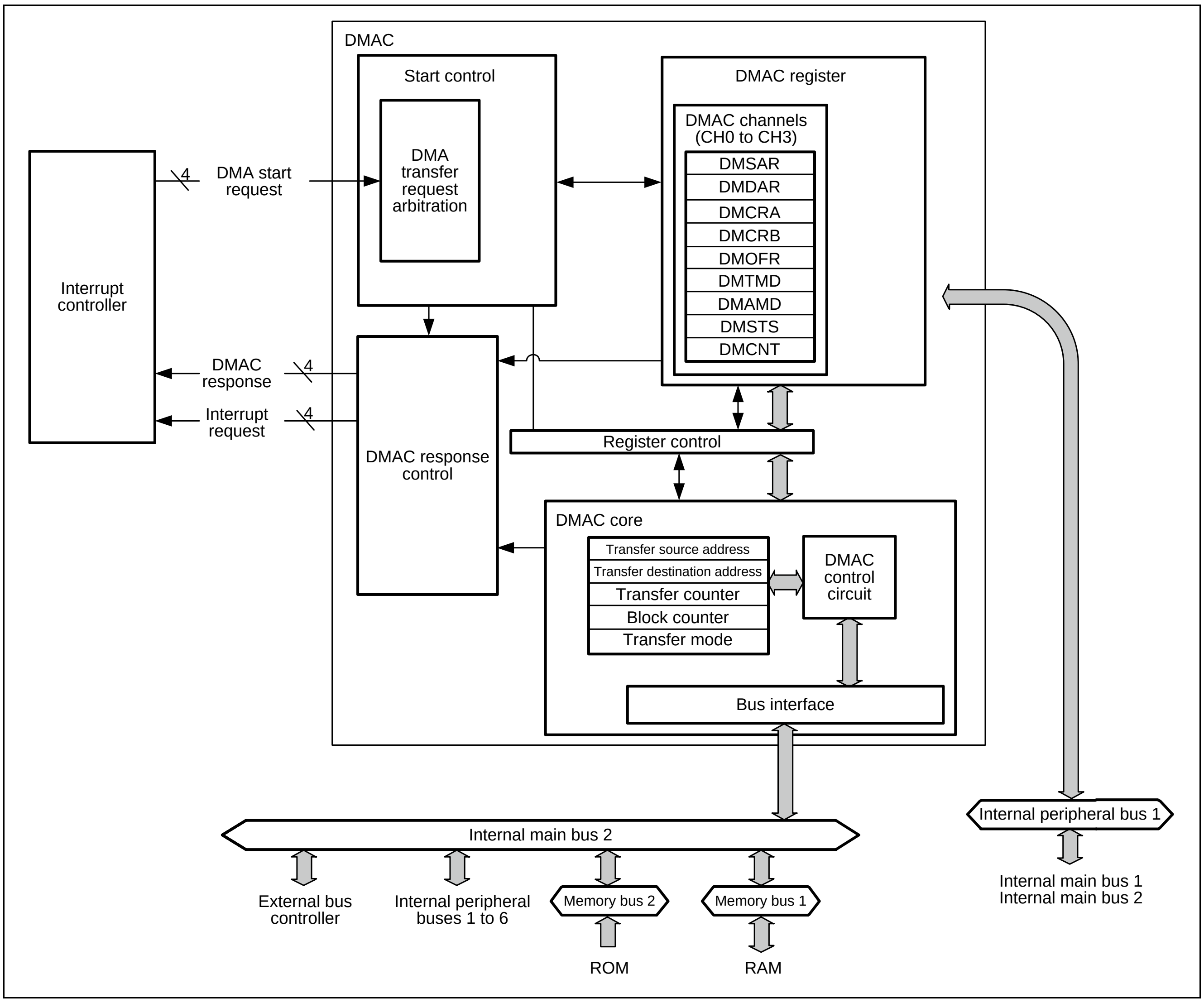


Figure 2.13 RX631 DMACA Block Diagram

A block diagram of the RX631's EXDMACa is shown below.

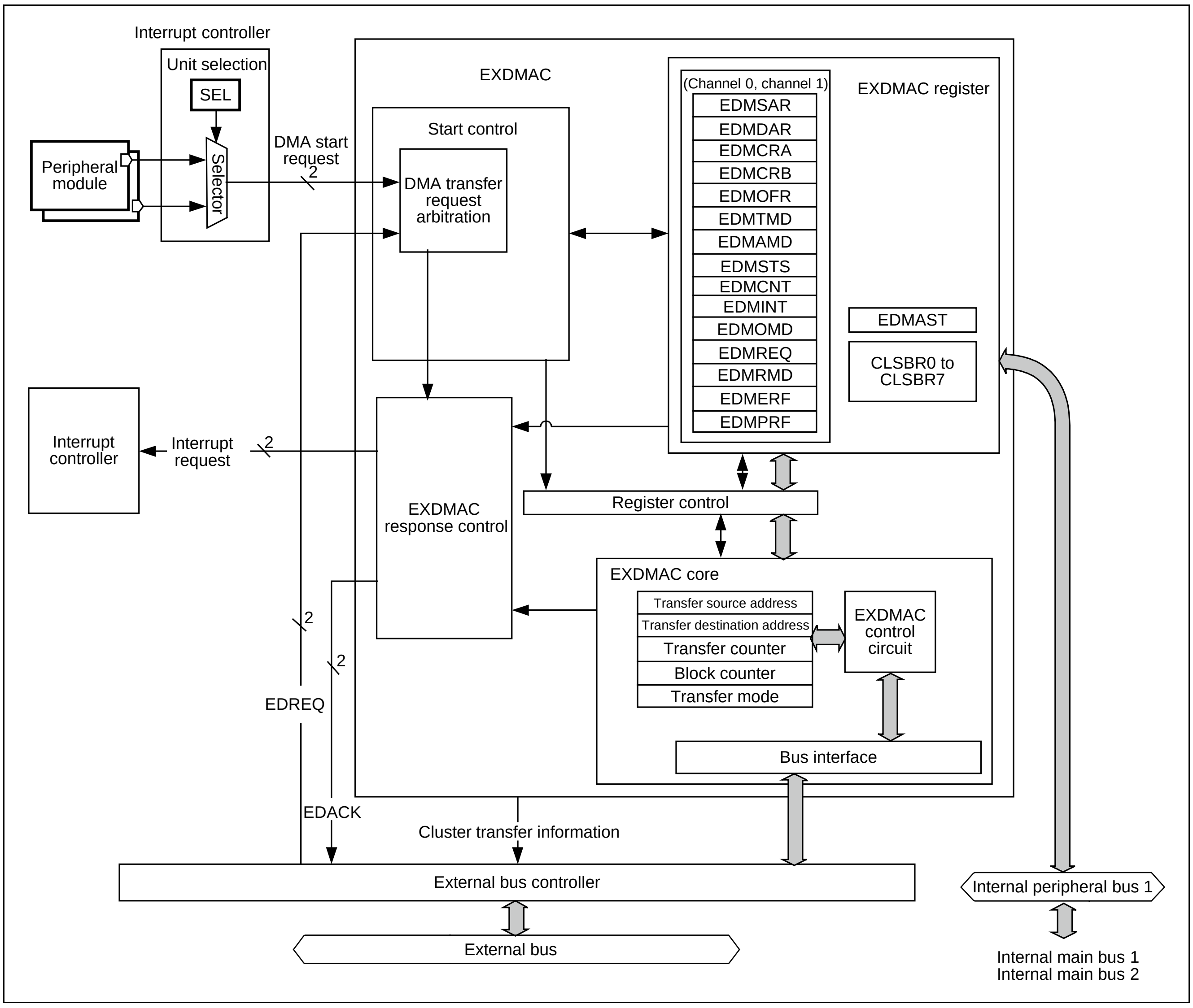


Figure 2.14 RX631 EXDMACa Block Diagram

2.6.3 Comparison of Registers

Table 2.19 compares the DMAC registers of the SH7044 and the DMACA registers of the RX631.

Table 2.20 compares the DMAC registers of the SH7044 and the EXDMACa registers of the RX631.

Table 2.19 SH7044/RX631 DMAC/DMACA Register Comparison

SH7044	RX631
DMAC n: 0 to 3	DMACA m: 0 to 3
DMA operation register (DMAOR)	DMA module start register (DMAST)
DMA source address register n (SARn)	DMA transfer source register m (DMACm.DMSAR)
DMA destination register n (DARn)	DMA transfer destination register m (DMACm.DMDAR)
DMA transfer count register n (DMATCRn)	DMA transfer counter register m (DMACm.DMCRA)
DMA channel control register (CHCRn)	DMA block transfer count register m (DMACm.DMCRB)
—	DMA transfer mode register m (DMACm.DMTMD)
	DMA interrupt setting register m (DMACm.DMINT)
	DMA address mode register m (DMACm.DMAMD)
	DMA transfer enable register m (DMACm.DMCNT)
	DMA software start register m (DMACm.DMREQ)
	DMA status register m (DMACm.DMSTS)
	DMA activation source flag control register m (DMACm.DMCSL)
	DMA offset register (DMAC0.DMOFR)

Note: In the register symbols above, n and m represent the respective DMA channel numbers.

Table 2.20 SH7044/RX631 DMAC/EXDMACa Register Comparison

SH7044	RX631
DMAC n: 0 to 3	EXDMACa p: 0 to 1
DMA operation register (DMAOR)	EXDMA module start register (EDMAST)
DMA source address register n (SARn)	EXDMA transfer source register p (EXDMACp.EDMSAR)
DMA destination register n (DARn)	EXDMA transfer destination register p (EXDMACp.EDMDAR)
DMA transfer count register n (DMATCRn)	EXDMA transfer counter register p (EXDMACp.EDMCRA)
DMA channel control register (CHCRn)	EXDMA block transfer count register p (EXDMACp.EDMCRB)
—	EXDMA output setting register p (EXDMACp.EDMOMD)
	EXDMA transfer mode register p (EXDMACp.EDMTMD)
	EXDMA interrupt setting register p (EXDMACp.EDMINT)
	EXDMA address mode register p (EXDMACp.EDMAMD)
	EXDMA transfer enable register p (EXDMACp.EDMCNT)
	EXDMA software start register p (EXDMACp.DEMREQ)
	EXDMA status register p (EXDMACp.EDMSTS)
	EXDMA external request sense mode register p (EXDMACp.EDMRMD)
	EXDMA external request flag register p (EXDMACp.EDMERF)
	EXDMA peripheral request flag register p (EXDMACp.EDMPRF)
	EXDMA offset register (EXDMAC0.EDMOFR)
	Cluster buffer register y (CLDBR0 to CLDBR7)

Note: In the register symbols above, n and p represent the respective DMA channel numbers.

2.6.4 Channel Priority

Table 2.21 shows the channel priority for DMA transfers. On the RX631 the channel priority is fixed.

Table 2.21 DMA Transfer Channel Priority

	SH7044	RX631	
Type	**DMAC**	**DMACA**	**EXDMACa**
Fixed	One of the following three patterns: 1. CH0 > CH1> CH2 > CH3 2. CH0 > CH2 > CH3 > CH1 3. CH2 > CH0 > CH1 > CH3	CH0 > CH1 > CH2 > CH3	CH0 > CH1
Round robin	When transfer of one transfer unit finishes, the priority of the channel on which transfer has finished is reduced to the lowest level.	—	—

2.6.5 DMA Activation Sources and Settings

Table 2.22 lists the types of transfer activation sources of the respective DMAC modules.

Table 2.22 DMA Activation Source Comparison

	SH7044	RX631	
DMA Activation Sources	**DMAC**	**DMACA**	**EXDMACa**
Activation by software	Supported	Supported	Supported
Activation by external device via request pin	Supported (activation by _DREQ signal)	Not supported	Supported (activation by EDREQn signal)
Activation by peripheral module	Supported	Supported (activation by interrupt via external interrupt input also supported)	Supported (MTU1 or TPU7 compare match A)

On the SH7044, DMA activation by peripheral module requires that the activation source be specified by a resource selector setting in the DMA channel control register (RS3 to RS0 in CHCRx). On the RX631 (DMACA) DMA activation by peripheral module requires specification of the activation source's vector number in the DMAC activation request select register (DMRSRm, m: channel 0 to 3) of the interrupt controller.

2.6.6 Transfer Sources and Destinations

The transfer sources and destinations supported by each DMA are listed below.

Table 2.23 SH7044 DMAC Transfer Sources and Destinations

	Transfer Destination				
Transfer Sources	**External Device with DACK**	**External Memory**	**Memory-Mapped External Device**	**On-Chip Memory**	**On-Chip Peripheral Module**
External Device with DACK	Not supported	●	●	Not supported	Not supported
External Memory	●	○	○	○	○
Memory-Mapped External Device	●	○	○	○	○
On-Chip Memory	Not supported	○	○	○	○
On-Chip Peripheral Module	Not supported	○	○	○	○

●: Single address mode transfers supported. ○: Dual address mode transfers supported.

Table 2.24 RX631 DMACA Transfer Sources and Destinations

	Transfer Destination				
Transfer Sources	**External Device with EDACK**	**External Memory**	**Memory-Mapped External Device**	**On-Chip Memory**	**On-Chip Peripheral Module**
External Device with DACK	Not supported	Not supported	Not supported	Not supported	Not supported
External Memory	Not supported	○	○	○	○
Memory-Mapped External Device	Not supported	○	○	○	○
On-Chip Memory	Not supported	○	○	○	○
On-Chip Peripheral Module	Not supported	○	○	○	○

○: Transfers supported.

Table 2.25 RX631 EXDMACa Transfer Sources and Destinations

	Transfer Destination				
Transfer Sources	**External Device with EDACK**	**External Memory**	**Memory-Mapped External Device**	**On-Chip Memory**	**On-Chip Peripheral Module**
External Device with EDACK	Not supported	●	●	Not supported	Not supported
External Memory	●	○	○	Not supported	Not supported
Memory-Mapped External Device	●	○	○	Not supported	Not supported
On-Chip Memory	Not supported	Not supported	Not supported	Not supported	Not supported
On-Chip Peripheral Module	Not supported	Not supported	Not supported	Not supported	Not supported

●: Single address mode transfers supported. ○: Dual address mode transfers supported.

2.6.7 Transfer Modes

The transfer modes of the SH7044 and RX631 are described below.

The concept of transfer mode does not apply on the SH7044. When switching to the RX631, the equivalent transfer mode is normal transfer mode. However, if the source address reload function was used on the SH7044, it is possible to achieve equivalent results on the RX631 by using repeat mode to repeat the source address for four transfer units. This makes it possible to reproduce the transfer method of the SH7044 by using the transfer modes of the RX631.

Table 2.26 RX631 Transfer Modes

Transfer Mode	DMACA	EXDMACa	Remarks
Normal transfer	○	○	Equivalent to the transfer method of the SH7044
Repeat transfer	○	○	Usable as a substitute for source address reload on the SH7044
Block transfer	○	○	
Cluster transfer	Not supported	○	

2.6.8 Address Modes

The SH7044 has two address modes: single address mode and dual address mode.

The EXDMACa of the RX631 has a single address mode and a dual address mode like the SH7044. 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 DMACA the address mode concept does not apply, but the method of specifying addresses and the operation are equivalent to dual address mode on the SH7044.

2.6.9 Bus Modes

On the SH7044 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 RX631 it is not possible to specify the bus mode of the DMACA or EXDMACa. This is because the bus architecture differs from that of the SH7044. The RX631 supports parallel operation when the bus master accesses a different slave. On the RX631 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 2.15 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 ROM and RAM.

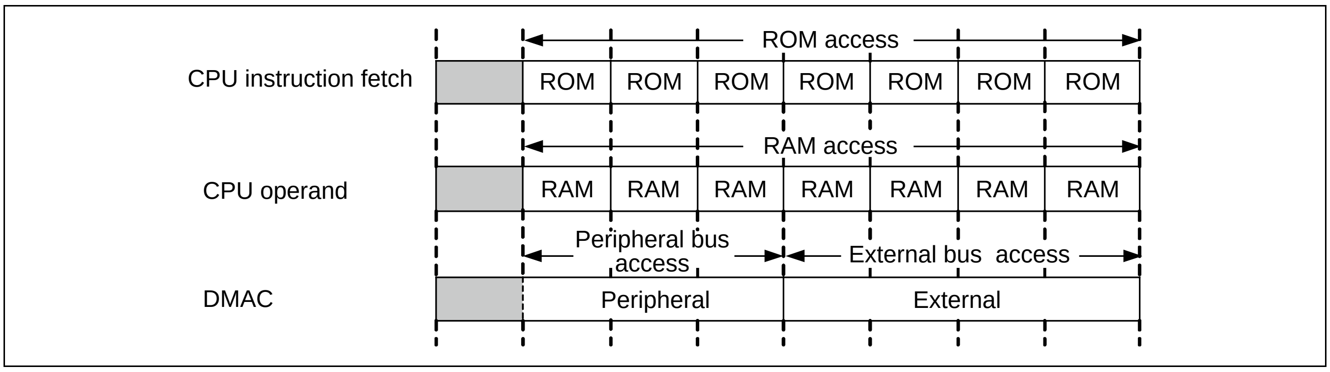


Figure 2.15 Parallel Bus Operation

2.6.10 Module Stop

The initial state of the peripheral modules of the RX631 is stopped, due to the low power consumption function. However, the initial state of the DMACA is operational, so there is no need to cancel the module stop state. Module stop can be applied to the DMACA, but doing so also stops the DTC because the same control bit in the module stop control register is used for both the DTC and the DMAC.

The initial state of the EXDMAC is stopped. Do not fail to cancel the module stop state when making settings to the module. Before accessing the module stop control register to cancel the module stop state, first cancel register write protection.

2.6.11 Direct Memory Access Controller (DMAC) Setting Example

In the direct memory access controller (DMAC) setting example shown below for the SH7044 and RX631, the DMAC is used to implement data transfer between the serial communication interface (SCI) and the on-chip RAM. Refer to 2.9.6 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+RX63N board is used, and the SCI transfer mode is clock-synchronous serial transfer.
2. When an SCI 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.

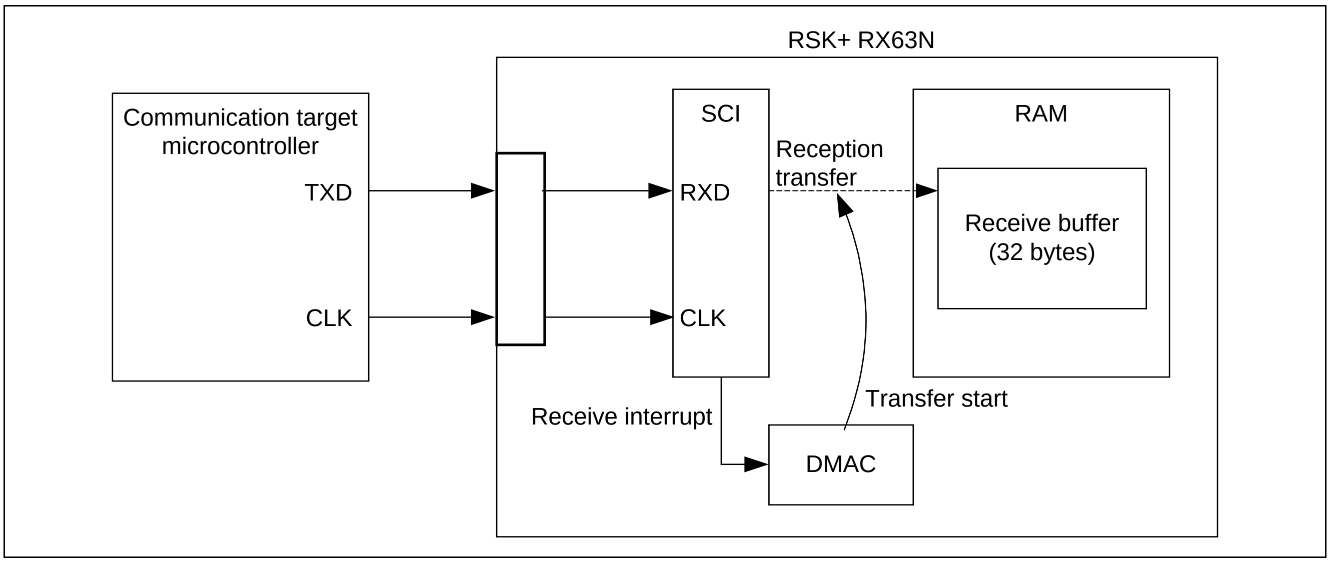


Figure 2.16 Example of Data Transfer between RAM and SCI Using DMAC

Table 2.27 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 (SCI)	
Transfer destination	On-chip RAM (receive buffer)	
Transfer source address	Fixed	
Transfer destination address	Transfer destination address incremented following transfer	
Activation sources	SCI receive data-full interrupt	RXI0 interrupt
Interrupt handling	DMAC transfer-end interrupt	DMAC0I
Pins used	P21/RXD0	
	P22/SCK0	

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 2.28 DMAC Normal Transfer Initial Setting Example

Procedure		SH7044 Setting Example	RX631 Setting Example
1	Make peripheral function settings (SCI initial settings).	Make SCI clock-synchronous slave receive settings. Make settings in table 2.56 for SCI function or ICU function. Enable RXI interrupts and error interrupts. The DTC will not operate if interrupts are not enabled. (Set item 12 in table 2.56, interrupt enable in the interrupt controller, after making DMA settings.)	

2	Stop DMA transfer.	CHCR0.DE = 0
		(DMAC0 operation disabled)
			IER18.IEN16 = 0
			(DMAC0I interrupt disabled)
			DMAC0.DMCNT.DTE = 0
			(DMA transfer disabled)

3	Set DMAC activation source.	None	DMRSR0 = 214 (vector number set to 214/RXI0)

4	Make DMA address mode settings.	CHCR0.SM1, CHCR0.SM0 = 0
		(fixed transfer source address mode)
		CHCR0.DM1, CHCR0.DM0 = 1
		(increment transfer destination address mode)
			DMAC0.DMAMD.SM = 0
			(fixed transfer source address mode)
			DMAC0.DMAMD.DM = 2
			(increment transfer destination address mode)

5	Make DMA transfer mode settings.	CHCR0.RS3 to CHCR0.RS0 = 1101b
		(transfer request sources set to SCI0 and RXI0)
		CHCR0.TM = 0 (cycle-steal bus mode)
		CHCR0.TS1, CHCR0.TS0 = 0
		(transfer data size: 8 bits)
			DMAC0.DMTMD.DCTG = 1
			(transfer requests: peripheral module)
			DMAC0.DMTMD.SZ = 0
			(transfer data size: 8 bits)
			DMAC0.DMTMD.MD = 0
			(transfer mode: normal transfer)

6	Set transfer source address.	SAR0 = RDR address	DMAC0.DMSAR = SCI.RDR address
7	Set transfer destination address.	DAR0 = receive buffer address	DMAC0.DMDAR = receive buffer address
8	Set transfer size.	DMATCR0 = 32	DMAC0.DMCRA = 32
9	Make interrupt selection setting.	None	DMAC0.DMCSL.DISEL = 0 (activation source interrupt flag 0 cleared at transfer start)
10	Set DMA priority.	DMAOR.PR1, DMAOR.PR0 = 0 (priority mode: CH0 > 1 > 2 > 3 fixed)	None
11	Set DMA interrupt level.	IPRC = 0x5000 (DMAC0 interrupt priority set to 5)	IPR198 = 5 (DMAC0I interrupt level set to 5)
12	Set DMA interrupt.	CHCR0.IE = 1 (transfer-end interrupt enabled)	DMAC0.DMINT.DTIE = 1 (transfer-end interrupt enabled)

13	Enable DMA transfer.	CHCR0.DE = 1
		(DMAC0 operation enabled)
			IER18.IEN16 = 1
			(DMAC0I interrupt enabled)
			DMAC0.DMCNT.DTE = 1
			(DMA transfer enabled)

14	Start peripheral functions.	Make settings in item 12 in table 2.56, SCI Clock-Synchronous Slave Receive Initial Setting Example, for SCI function or ICU function to start SCI function operation.	
15	Activate DMA module.	DMACOR.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 Multifunction Timer Pulse Unit (MTU)

2.7.1 Comparison of Specifications

Table 2.29 Comparison of MTU Specifications on SH7044 and RX631

Item		SH7044	RX631
Pulse I/O		Maximum 16	
Pulse input		—	3
Count clock		Selectable for each channel among six clocks based on the internal clock (ϕ) and eight clocks employing external clocks (TCLKA, TCLKB, TCLKC, and TCLKD).	Selectable for each channel among seven or eight clocks using PCLK, MTCLKA, MTCLKB, MTCLKC, and MTCLKD (four for MTU5).
Function description		The MTU2a of the RX631 includes the functionality of the MTU of the SH7044 (and they are software compatible).	
Function settings	MTU0 to MTU4	• Compare match waveform output (selectable among 0, 1, and toggled) • Input capture function (selectable among rising, falling, and both edges) • Synchronous operation — Synchronized writing to multiple timers (TCNT) — Clearing synchronized with compare match or input capture — I/O with various registers in synchronization with counter • PWM mode — PWM output with user-specified duty — Up to 12-phase PWM output combined with synchronous operation	
	MTU0, MTU3, MTU4	• Support for buffer operation settings • Input capture register with double-buffer configuration • Auto-overwriting of output compare register • AC synchronous motor drive mode on RX631	
	MTU1, MTU2	Up- or down-counting of two-phase encoder pulses in phase counting mode	
	MTU3, MTU4	A total of six-phase waveform output, including three phases each for positive and negative complementary PWM, by interlocking operation	
	MTU5	—	Counter function for dead time compensation
Complementary PWM mode		Interrupts at counter peaks and troughs	
Interrupt sources (See separate listing for details.)		23	28
Buffer operation		Automatic transfer of register contents	
Trigger generation		A/D converter start trigger	A/D converter start trigger PPG output trigger
DMAC activation		MTU0 to MTU4: TGRA compare match or input capture Note: On the SH7044 the registers are named TGRnA (n: channel number).	
DTC activation	MTU0 to MTU3	TGR compare match or input capture	
	MTU4	TGR compare match or input capture, TCNT overflow or underflow	
	MTU5	—	TGR compare match or input capture
A/D conversion start triggers		MTU0 to MTU4: TGRA compare match or input capture Added to MTU0 on RX631: TGRE and TGRF compare match Added to MTU4 on RX631: TCNT underflow in complementary PWM mode (troughs)	

Item	SH7044	RX631
PPG triggers	—	MTU0 to MTU3: TGRA and TGRB compare match or input capture
A/D conversion start request delay function	—	MTU4: Start request at match of TADCORA or TADCORB and TCNT
Interrupt skipping function	—	MTU3: TGRA compare match interrupt skipping MTU4: TCIV interrupt skipping

Table 2.30 List of MTU Interrupt Sources on SH7044 and RX631

	SH7044/RX631					RX631
Item	**MTU0**	**MTU1**	**MTU2**	**MTU3**	**MTU4**	**MTU5**
Compare match/input capture nA	○	○		○	○	
Compare match/input capture nB	○	○		○	○	
Compare match/input capture nC	○			○	○	
Compare match/input capture nD	○			○	○	
Overflow	○	○	○	○	○	
Underflow		○	○		○	
Compare match nE	Δ					
Compare match nF	Δ					
Compare match/input capture nU						Δ
Compare match/input capture nV						Δ
Compare match/input capture nW						Δ

n: Channel number ○: Compatible between SH7044 and RX631 Δ: Added on RX631

2.7.2 Handling of Interrupt Flags

The RX631's MTU2a and the SH7044's MTU are software compatible. With the exception of changes to the timer status register (TSR) interrupt flags, it is possible to migrate the functions of MTU0 to MTU4 on the SH7044 without changing the registers. (It is necessary to make separate changes to the initial settings, such as the pin settings.) The one significant difference is that on the RX631 the timer status register (TSR) contains no interrupt flags. Nevertheless, it is possible to implement equivalent processing by using the interrupt request register (IR) in the interrupt controller corresponding to the MTU (IR142 and above).

2.7.3 List of Registers

Whether or not changes to the register settings are needed when switching from the SH7044 to the RX631 is indicated below.

Table 2.31 List of MTU Registers

Register Name	SH7044 (MTU)	RX631 (MTU2a)	Change
Timer control register	TCR0 to TCR4	MTU0.TCR to MTU5.TCR	◎
		MTU5.TCRU/V/W	*
Timer mode register	TMDR0 to TMDR4	MTU0.TMDR to MTU4.TMDR	◎
Timer I/O control register	TIOR0H, TIOR3H, TIOR4H	MTU0.TIORH, MTU3.TIORH, MTU4.TIORH	◎
	TIOR1, TIOR2	MTU1.TIOR, MTU2.TIOR	◎
	TIOR0L, TIOR3L, TIOR4L	MTU0.TIORL, MTU3.TIORL, TU4.TIORL	◎
Timer compare match clear register		TCNTCMPCLR	*
Timer interrupt enable register	TIER0	MTU0.TIER	◎
	TIER1, TIER2	MTU1.TIER , MTU2.TIER	◎
	TIER3, TIER4	MTU3.TIER, MTU4.TIER	◎
		MTU0.TIER2 MTU5.TIER	*[1]
Timer status register	TSR0	MTU0.TSR	Δ
	TSR1, TSR2	MTU1.TSR MTU2.TSR	Δ
	TSR3, TSR4	MTU3.TSR MTU4.TSR	Δ
Timer buffer operation transfer mode register		MTU0.TBTM, MTU3.TBTM, MTU4.TBTM	*
Timer input capture control register		MTU1.TICCR	*
Timer A/D conversion start request control register		MTU4.TADCR	*
Timer A/D conversion start request cycle set registers A and B		MTU4.TADCORA, MTU4.TADCORB	*
Timer A/D conversion start request cycle set buffer registers A and B		MTU4.TADCOBRA, MTU4.TADCOBRB	*
Timer counter	TCNT0 to TCNT4	MTU0.TCNT to MTU4.TCNT	◎
		MTU5.TCNTU/V/W	*
Timer general register	TGR0, TGR3, TGR4 (A, B, C, D)	MTU0.TGRA to D MTU3.TGRA to D MTU4.TGRA to D	◎
		MTU0.TGRE,F	*
	TGR1, TGR2 (A, B)	MTU1.TGRA,B MTU2.TGRA,B	◎
Timer start register	TSTR	MTU.TSTR	◎
Timer synchronous register	TSYR	MTU.TSYR	◎
Timer read/write enable register		MTU.TRWER	*
Timer output master enable register	TOER	MTU.TOER	◎
Timer output control register	TOCR	MTU.TOCR1	○
		MTU.TOCR2	*
Timer output level buffer register		MTU.TOLBR	*
Timer gate control register	TGCR	MTU.TGCR	◎
Timer sub counter	TCNTS	MTU.TCNTS	◎
Timer dead time data register	TDDR	MTU.TDDR	◎
Timer period data register	TCDR	MTU.TCDR	◎

Register Name	SH7044 (MTU)	RX631 (MTU2a)	Change
Timer period buffer register	TCBR	MTU.TCBR	◎
Timer interrupt skipping set register		MTU.TITCR	*
Timer interrupt skipping counter		MTU.TITCNT	*
Timer buffer transfer set register		MTU.TBTER	*
Timer dead time enable register		MTU.TDER	*
Timer waveform control register		MTU.TWCR	*
Noise filter control register		MTU0.NFCR to MTU4.NFCR	*

◎: Registers with identical bit assignments on the SH7044 and RX631

○: Registers where the RX631 has new functions (bits) assigned. (Except for the new function bits, the bit assignments are identical.)

∆: On the RX631 these registers contain no interrupt flags.

Note: * Registers with no equivalents on the SH7044. (These registers are for new functions added in the MTU2. When migrating programs that use the SH7044's MTU, the initial values can be used unaltered without any problem.)

2.7.4 Unit Selection Function

Some interrupt sources of the MTU and TPU are assigned to common vectors. It is therefore necessary when using the MTU to specify which interrupt will be using each vector by setting the corresponding selector. (For details, see 1.8.6, Unit Selection Function.)

2.7.5 Module Stop

The initial state of the peripheral modules of the RX631 is stopped, due to the low power consumption function. The initial state of the MTU is stopped as well. Do not fail to cancel the module stop state when making settings to the module. Before accessing the module stop control register to cancel the module stop state, first cancel register write protection.

2.7.6 MTU Output Compare Match Setting Example

In the setting example shown below for the SH7044 and RX631, the multifunction timer pulse unit (MTU) is used to implement output compare match functionality.

< Specifications >

1. The RSK+RX63N 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 2.32 MTU Output Compare Match Specifications

Item	Description	Remarks
Count clock	Rising edge of PCLKB/1	PCLKB = 48 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	P24/MTIOC4A	For pulse output

Figure 2.17 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.

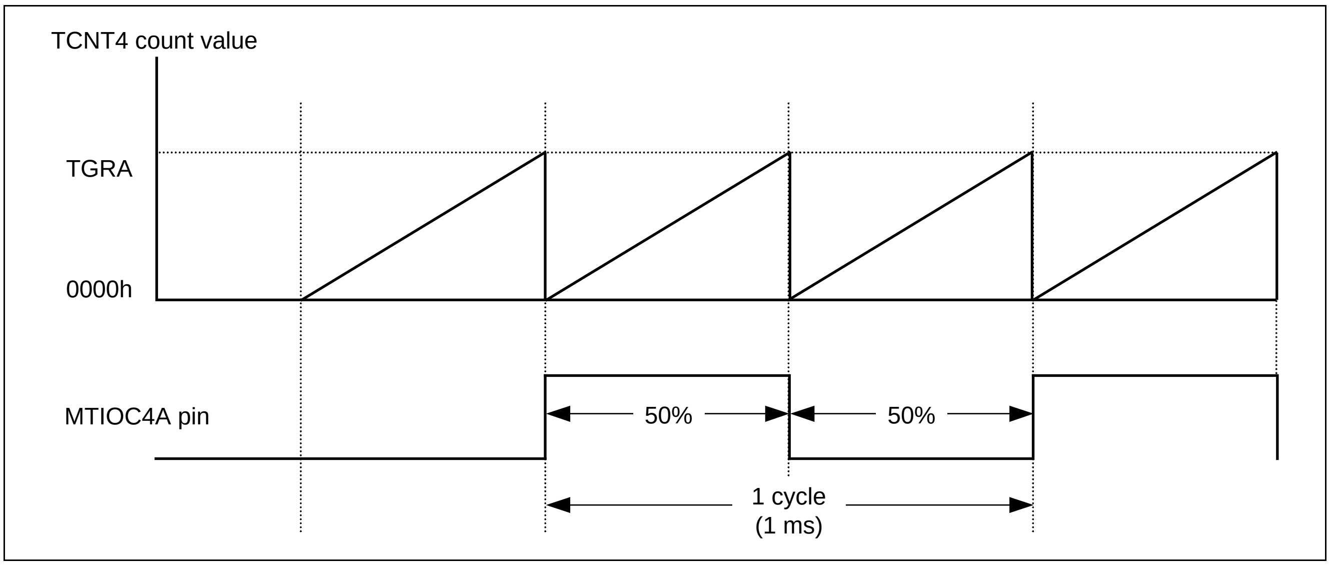


Figure 2.17 MTU Output Compare Match Operation

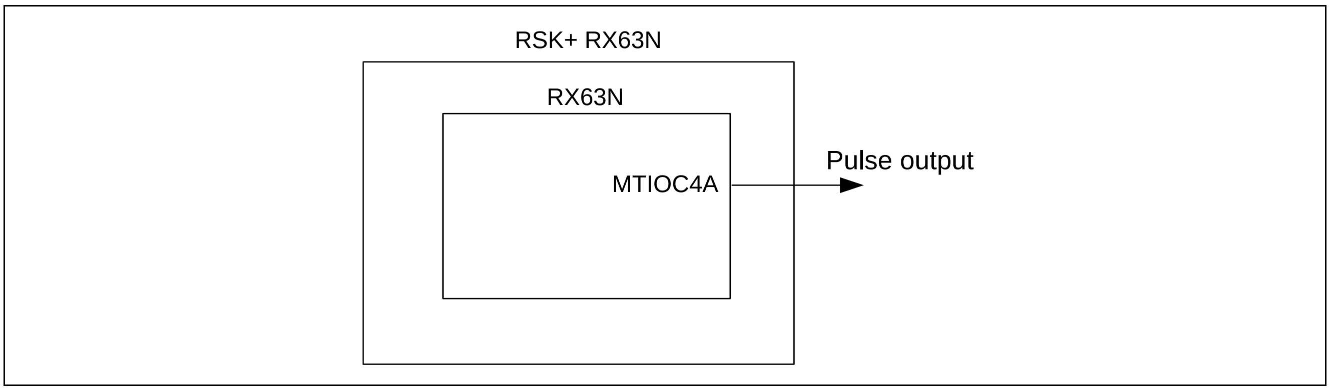


Figure 2.18 MTU Output Compare Match Connection Diagram

Table 2.33 MTU Output Compare Match Initial Setting Example

Procedure		SH7044 Example Settings Pϕ (Peripheral Clock): 20 MHz	RX631 Example Settings PCLK (Peripheral Clock): 48 MHz
1	Cancel module stop state.	(No module stop function)	SYSTEM .PRCR = 0xA502 SYSTEM .MSTPCRA.MSTPA9 = 0 SYSTEM .PRCR = 0xA500

2	Stop MTU.	TSTR.CST4 = 0 (TCNT stopped)
		TSYR.SYNC4 = 0
		(independent operation enabled)
		TCNT4 = 0x0000 (TCNT0 cleared)
		TGR4A = 0x0000 (TGR0A cleared)
			MTU.TSTR.CST4 = 0 (TCNT stopped)
			MTU.TSYR.SYNC4 = 0
			(independent operation enabled)
			MTU4.TCNT = 0x0000
			(TCNT cleared)
			MTU4.TGRA = 0x0000
			(TGRA cleared)

3	Make I/O port settings (pin I/O and pin function settings).	PFC settings
		PEIOR.PE12IOR = 1 (output)
		PECR1.PE12MD = 1
		(TIOC4A selected)
			MTIOC4A pin settings in MPC
			PORT2.PDR.B4 = 1 (output)
			PORT2.PMR.B4 = 0 (GPIO)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enabled)
			MPC.P24PFS = 0x01 (pin function setting)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1

4	Select counter clock; select edge.	Internal clock: ϕ/1 TCR4.TPSC2 to TCR4.TPSC0 = 000b TCR4.CKEG1 to TCR4.CKEG0 = TCR4.CKEG0 (counting at rising edge)	Internal clock: ϕ/1 MTU4.TCR.TPSC2 to MTU4.TCR.TPSC0 = 000b MTU4.TCR.CKEG1 to MTU4.TCR.CKEG0 = 0 (counting at rising edge)
5	Make counter operation and TCNT clear source settings.	TGR4A compare match, input capture, and TCNT clear source TGR4A TCR4.CCLR2 to TCR4.CCLR0 = 001b	TGRA.MTU4 compare match, input capture, and TCNT clear source TGR4A MTU4.TCR.CCLR2 to MTU4.TCR.CCLR0 = 001b
6	Enable TOIC4A output (MTU3 and MTU4 only).	TOER.OE4A = 1	MTU.TOER.OE4A = 1
7	Make timer I/O control settings.	Output compare register: TGR4A Initial output: 0, output toggled at compare match TIOR4H.IOA3 to TIOR4H.IOA0 = 0011b	Output compare register: MTU4.TGRA Initial output: 0, output toggled at compare match MTU4.TIORH.IOA3 to MTU4.TIORH.IOA0 = 0011b
8	Set TGRA (setting value: 1/2 cycle duration).	TGR4A = 2710h	MTU4.TGRA = 5DE6h

9	Make timer mode register settings.	TMDR4.BFB = 0 (normal operation)
		TMDR4.BFA = 0 (normal operation)
		TMDR4.MD3 to TMDR4.MD0 = 0
		(normal operation)
			MTU4.TMDR.BFB = 0 (normal operation)
			MTU4.TMDR.BFA = 0 (normal operation)
			MTU4.TMDR.MD3 to MTU4.TMDR.MD0
			= 0 (normal operation)
			PORT2.PMR.B4 = 1 (peripheral function)

10	Enable timer operation.	TSTR.CST4 = 1 (TCNT0 count operation)	MTU.TSTR.CST4 = 1 (TCNT0 count operation)

2.7.7 MTU Input Capture Setting Example

In the setting example shown below for the SH7044 and RX631, the input capture function of the multifunction timer pulse unit (MTU) is used to measure the input pulse width.

< Specifications >

1. The RSK+RX63N 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* is exceeded, LED1 turns on and processing ends.

Note: * Measurement is not possible when the TCNT overflow count exceeds FFFFh.

Table 2.34 MTU Input Capture Specifications

Item	Description	Remarks
Count clock	Rising edge of PCLKB/1	PCLKB = 48 MHz
Operating mode	Normal mode	
Synchronous operation	Not used.	
Counter clear source	TGRA Input capture	
Timer general register	Input capture register	
Pin used	P34/MTIOC0A (input capture at both edges)	Pulse input
	P05 (GPIO)	LED1 output
Interrupt sources	MTU0 input capture A interrupt Overflow interrupt	

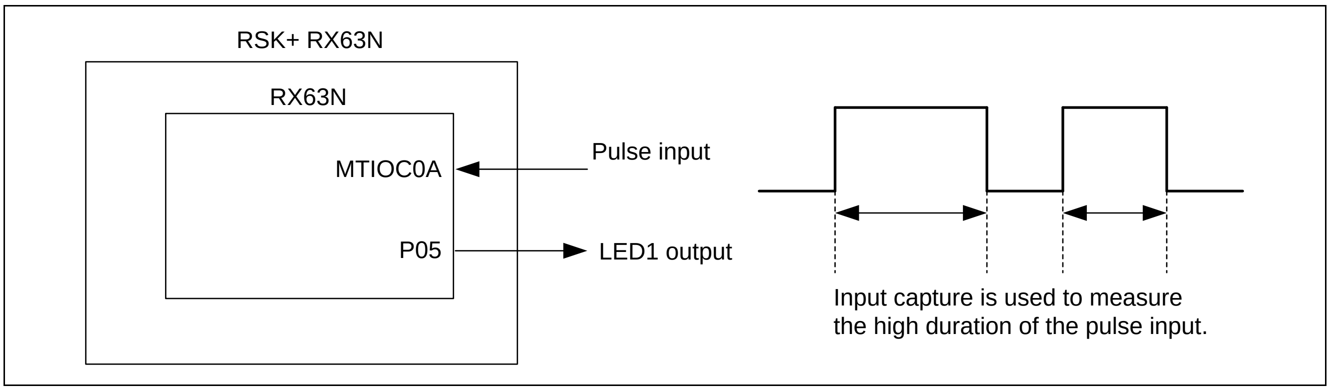
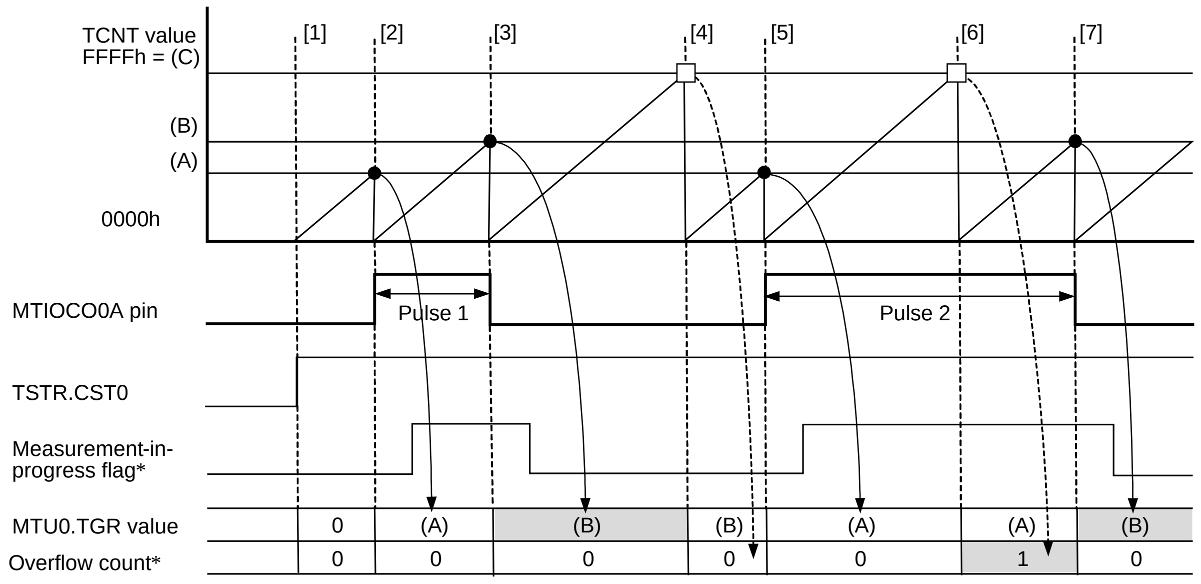

RSK+ RX63N
RX63N
MTIOC0A
P05
Pulse input
LED1 output
Input capture is used to measure the high duration of the pulse input.

Figure 2.19 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: * Processed by software.

Figure 2.20 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 2.20 above.

[1] MTU0 starts counting when the TSTR.CST0 bit is set to 1 (start count).

[2] An input capture interrupt is generated when a rising edge is input on the MTIOC0A 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 MTIOC0A 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 MTU0.TCNT overflow count (0) and the value of MTU0.TGRA (B).

[4] An overflow interrupt is generated when MTU0.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 MTIOC0A 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 MTU0.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 MTIOC0A 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 MTU0.TCNT overflow count (1) and the value of MTU0.TGRA (B).

Table 2.35 MTU Input Capture Initial Setting Example

Procedure		SH7044 Example Settings Pϕ (Peripheral Clock): 20 MHz	RX631 Example Settings PCLK (Peripheral Clock): 48 MHz

1	Cancel module stop state.	(No module stop function)	SYSTEM.PRCR = 0xA502
			SYSTEM.MSTPCRA.MSTPA9 = 0
			SYSTEM.PRCR = 0xA500

2	Disable interrupts.	TIER0.TGIEA = 0 (TGIA disabled)
		TIER0.TCIEV = 0 (TCIV disabled)
			IER11.IEN6 = 0
			(vector 142 and TGIA0 disabled)
			IER0D.IEN3 = 0
			(vector 107 interrupt disabled)
			GEN01.EN0 = 0
			(group 01 interrupts disabled)
			MTU0.TIER.TGIEA = 0
			MTU0.TIER.TCIEV = 0

3	Make noise filter settings. (Use of the noise filter is not essential.)	—	MTU0.NFCR.BIT.NFAEN = 1;
			MTU0.NFCR.BIT.NFCS = 0;
			2-cycle wait

3	Clear the interrupt source.	—	IR142 = 0
4	Set the unit selector.	—	SEL.CN0 = 0 (MTU0 setting)

5	Stop MTU.	TSTR.CST0 = 0 (TCNT stopped)
		TSYR.SYNC0 = 0
		(independent operation enabled)
		TCNT0 = 0x0000 (TCNT0 cleared)
		TGR0A = 0x0000 (TGR0A cleared)
			MTU.TSTR.CST0 = 0 (TCNT stopped)
			MTU.TSYR.SYNC0 = 0
			(independent operation enabled)
			MTU0.TCNT = 0x0000 (TCNT cleared)
			MTU0.TGRA = 0x0000 (TGRA cleared)

6	Make I/O port settings (pin I/O and pin function settings).	PFC settings
		PEIOR.PE0IOR = 1 (input)
		PECR2.PE0MD1 to PECR2.PE0MD0 = 01 (TIOC0A selected)
			MTIOC0A pin settings in MPC
			PORT3.PDR.B4 = 0 (input)
			PORT3.PMR.B4 = 0 (GPIO)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enabled)
			MPC.P34PFS = 0x01 (pin function setting)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1
			PORT3.PMR.B4 = 1 (peripheral function)

7	Select counter clock; select edge.	Internal clock: ϕ/1 TCR0.TPSC2 to TCR0.TPSC0 = 000b TCR0.CKEG1 to TCR0.CKEG0 = 0 (counting at rising edge)	Internal clock: ϕ/1 MTU0.TCR.TPSC2 to MTU0.TCR.TPSC0 = 000b MTU0.TCR.CKEG1 to MTU0.TCR.CKEG0 = 0 (counting at rising edge)
8	Make counter operation and TCNT clear source settings.	TGR0A compare match, input capture, and TCNT clear source TGR0A TCR0.CCLR2 to TCR0.CCLR0 = 001b	MTU0.TGRA compare match, input capture, and TCNT clear source TGRA MTU0.TCR.CCLR2 to MTU0.TCR.CCLR0 = 001b
9	Make timer I/O control settings.	TGR0A: input capture register Input capture at both edges on input pin TIOC0A TIOR0H.IOA3 to TIOR0H.IOA0 = 1010b	MTU0.TGRA: input capture register Input capture at both edges on input pin MTIOC0A MTU0.TIORH.IOA3 to MTU0.TIORH.IOA0 = 1010b

10	Make timer mode register settings.	TMDR0.BFB = 0 (normal operation)
		TMDR0.BFA = 0 (normal operation)
		TMDR0.MD3 to TMDR0.MD0 = 0
		(normal operation)
			MTU0.TMDR.BFB = 0 (normal operation)
			MTU0.TMDR.BFA = 0 (normal operation)
			MTU0.TMDR.MD3 to MTU0.TMDR.MD0 = 0 (normal operation)

11	Make interrupt priority register settings.	IPRD.WORD = 0x5000 (MTU0: level 5)	IPR142 = 5 (TGIA0: level 3)
			IPR107 = 5 (GROUP1: level 4)

Procedure		SH7044 Example Settings Pϕ (Peripheral Clock): 20 MHz	RX631 Example Settings PCLK (Peripheral Clock): 48 MHz

12	Enable interrupts.	TIER0.TGIEA = 1 (TGIA enabled)
		TIER0.TCIEV = 1 (TCIV enabled)
			MTU0.TIER.TGIEA = 1
			MTU0.TIER.TCIEV = 1
			IER11.IEN6 = 1
			(vector 142 and TGIA0 enabled)
			IER0D.IEN3 = 1
			(vector 107 interrupt enabled)
			GEN01.EN0 = 1
			(group 01 interrupts enabled)

13	Enable timer operation.	TSTR.CST0 = 1 (TCNT0 count operation)	MTU.TSTR.CST0 = 1 (TCNT0 count operation)

2.8 Watchdog Timers

2.8.1 Comparison of Specifications

The SH7044 incorporates the WDT as its watchdog timer module. The RX631 incorporates, in addition to the WDTA, the IWDTa, which operates on a dedicated independent clock. The specifications of these modules are compared below.

Table 2.36 Comparison of WDT, WDTA, and IWDTa Specifications on SH7044 and RX631

	SH7044	**RX631**	
Item	**WDT**	**WDTA**	**IWDTa**
Clock source	System clock (ϕ)	Peripheral clock (PCLK)	IWDT dedicated clock (IWDTCLK)
Clock frequency division ratio	ϕ/2, 64, 128, 256, 512, 1024, 4096, 8192	PCLK/4, 64, 128, 512, 4096, 8192	IWDTCLK/1, 16, 32, 64, 128, 256
Count operation	8-bit up-counter	14-bit down-counter	14-bit down-counter
Operating modes	• Watchdog timer mode • Interval timer mode	Watchdog timer mode only	Watchdog timer mode only
Count start condition	Timer enable bit in timer control register set to "enabled"	Selectable between the following: 1. Automatic count start after a reset (auto-start mode) 2. Count start by refresh operation (register start mode)	Selectable between the following: 1. Automatic count start after a reset (auto-start mode) 2. Count start by refresh operation (register start mode)
Count stop condition	Watchdog timer mode • Overflow • Power-on reset Interval timer mode • Timer enable bit in timer control register set to "disabled" • Power-on reset	• Underflow • Reset (down-counter, return to register initial value) • Refresh error	• Underflow • Reset (down-counter, return to register initial value) • Refresh error
Operation at overflow/ underflow	Watchdog timer mode • WDTOVF output • Internal reset Interval timer mode • Interrupt	• Internal reset • Interrupt	• Internal reset • Interrupt
Other	—	The following are specified by settings in option function select register 0: • Clock frequency division ratio • Refresh window start/end • Timeout period • Operation at underflow	The following are specified by settings in option function select register 0: • Clock frequency division ratio • Refresh window start/end • Timeout period • Operation at underflow

—: Function not implemented on SH7044.

2.8.2 Module Stop

The initial state of the peripheral modules of the RX631 is stopped, due to the low power consumption function. However, the WDTA and IWDTa have no module stop function. Their initial operating state is determined by settings in the option-setting memory. Note that when all modules are stopped, the WDTA stops counting and retains its state. The operation of the IWDTa when all modules are stopped is selectable between operational and stopped by a setting in the option-setting memory.

2.9 Serial Communication Interface

2.9.1 Comparison of Specifications

In contrast to the SCI of the SH7044, the RX631 integrates the SCIc and SCId. In addition to the conventional asynchronous and clock-synchronous transfer modes, the SCIc provides smartcard (IC card) interface support as an extended asynchronous mode. In addition, it supports simple I^2C bus interface single master operation and simple SPI bus interface mode. The SCId provides all the functions of the SCIc and adds extended serial interface support. For details of the transfer modes that are not supported on the SH7044, refer to the User’s Manual: Hardware.

Table 2.37 SCI Differences

Item		SH7044	RX631
Number of channels		2 channels (SCI0, SCI1)	13 channels SCIc: SCI0 to SCI11 SCId: SCI12
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		Transmit block: Support for continuous transmission using double-buffer configuration Receive block: Support for continuous reception using double-buffer configuration	
Data transfer		LSB-first only	Selectable between LSB-first and MSB-first (MSB-first only on simple I^2C bus)
Interrupt sources		• Transmit data-empty • Transmit-end • Receive data-full • Receive error	• Transmit data-empty • Transmit-end • Receive data-full • Receive error • Start condition* • Restart condition* • Stop condition generation-end* Note: * Used in simple I^2C mode.
Asynchronous mode	Data length	7 bits, 8 bits	
	Stop bits	1 bit, 2 bits	
	Parity	Even parity, odd parity, or no parity	
	Receive error detection	Parity error, overrun error, or framing error	
	Hardware flow control	No	Yes (controllable using CTS and RTSn pins)
	Break detection	Possible by reading level of RxDn pin directly when a framing error occurs	
	Clock source	Selectable between internal and external clock	Selectable between internal and external clock Ability to input transfer rate clock from TMR (SCI5 and SCI6)
	Multi-processor communication	Yes	
	Noise cancellation	No	On-chip digital noise filter for input on RXDn pins
Clock-synchronous mode	Data length	8 bits	
	Receive error detection	Overrun error	
	Hardware flow control	No	Yes (controllable using CTS and RTSn pins)
Smartcard interface		No	Yes
Simple I^2C mode		No	Yes
Simple SPI mode		No	Yes
Extended serial mode		No	Implemented on SCId (SCI12) only

A comparison of the on-chip SCI registers is shown below.

Table 2.38 SCI Communication Registers

SH7044	RX631	Changed
Transmit data register (TDR)	Transmit data register (TDR)	◎
Transmit shift register (TSR)	Transmit shift register (TSR)	◎
Receive data register (RDR)	Receive data register (RDR)	◎
Receive shift register (RSR)	Receive shift register (RSR)	◎
Serial mode register (SMR)	Serial mode register (SMR)	◎
Serial control register (SCR)	Serial control register (SCR)	◎
Serial status register (SSR)	Serial status register (SSR)	◎*1
Bit rate register (BBR)	Bit rate register (BBR)	◎
	Smartcard mode register (SCMR)	○
	Serial extended mode register (SEMR)	○
	Noise filter setting register (SNFR)	○*2
	I^2C mode registers 1 to 3 (SIMR1 to SIMR3)	*2
	I^2C status register (SISR)	—
	SPI mode register (SPMR)	*2*3
	Extended serial mode enable register (ESMER)	—
	Control registers 0 to 3 (CR0 to CR3)	—
	Port control register (PCR)	—
	Interrupt control register (ICR)	—
	Status register (STR)	—
	Status clear register (STCR)	—
	Control field 0 data register (CF0DR)	—
	Control field 0 compare enable register (CF0CR)	—
	Control field 0 receive data register (CF0RR)	—
	Primary control field 1 data register (PCF1DR)	—
	Secondary control field 1 data register (SCF1DR)	—
	Control field 1 compare enable register (CF1CR)	—
	Control field 1 receive data register (CF1RR)	—
	Timer control register (TCR)	—
	Timer mode register (TMR)	—
	Timer prescaler register (TPRE)	—
	Timer count register (TCNT)	—

◎: Registers with identical bit assignments on the SH7044 and RX631

○: Registers not present on the SH7044 that are required when using functions.

—: Registers with no equivalents on the SH7044. (When migrating programs that use the SH7044's SCI, the initial values can be used unaltered without any problem.)

Notes: 1. Only TDRE and RDRF differ.

2. When migrating programs, the initial values can be used unaltered.

3. For information on register settings required when performing flow control using the CTS and RTS pins, and register bit assignments, see the User's Manual: Hardware.

2.9.2 Switching SCIs

Differences such as the following should be borne in mind when switching from the SH7044's SCI to the SCIc or SCId on the RX631:

1. TDRE and RDRF
   The transmit register-empty (TDRE) and receive data-full (RDRF) flags in the serial status register of the SH7044 are not implemented on the SCIc or SCId modules of the RX631. The TDRE and RDRF flags on the SH7044 correspond to the IR (TXI) and IR (RXI) flags, respectively, of the interrupt controller on the RX631. When using an interrupt handler, IR (TXI) and IR (RXI) are both cleared automatically by the interrupt controller, so no additional processing is needed to clear the flags. Note that when polling is used the interrupt flags must be cleared in the same manner as on the SH7044.
2. Determination of one-bit period and clock source selection
   For communication in asynchronous mode, external clock input or TMR clock input can be selected as the clock source for determining the one-bit period by a setting in the serial extended mode register (SEMR). Also, the number of base clock cycles per one-bit period can be set to 8 or 16.
3. Digital noise filter
   The digital noise filter is activated or disabled by a setting in the serial extended mode register (SEMR). When enabling the noise filter, make sure to make the appropriate noise filter clock select setting in the noise filter setting register (SNFR).
4. Receive error interrupt
   The receive error interrupt is assigned to a group interrupt. The use of a group interrupt means that receive errors for 12 channels, SCI0 to SCI12, are assigned to a single vector. Therefore, when a receive error interrupt is generated it is necessary to detect the channel on which the error occurred by means of the ISn (n: channel number) flags in group interrupt source register 12 (GRP12). Within each channel, error handling for overrun errors, framing errors, and parity errors is the same as on the SH7044.

2.9.3 Module Stop

The initial state of the peripheral modules of the RX631 is stopped, due to the low power consumption function. The initial state of the SCI is also stopped. Do not fail to cancel the module stop state when making settings to the module. Before accessing the module stop control register to cancel the module stop state, first cancel register write protection.

2.9.4 Asynchronous Communication Setting Example (Interrupt Method/Polling Method)

A setting example for asynchronous serial communication using the serial communication interface (SCI) of the SH7044 and RX631 is presented below.

< Specifications >

1. SCI0 on the RSK+RX63N board is used to make a loopback connection to TXD and RXD.
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 transmit and receive interrupt source flags.
5. After the microcontroller is initialized, LED0 turns on when the SCI is ready for transmit and receive operation. LED1 turns on when transmission and reception end. LED2 turns on if a receive error occurs.

Table 2.39 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	SCI0 fixed	
Pins used	P20/TXD0	
	P21/RXD0	
	P03/GPIO	LED0 output
	P05/GPIO	LED1 output
	P10/GPIO	LED2 output

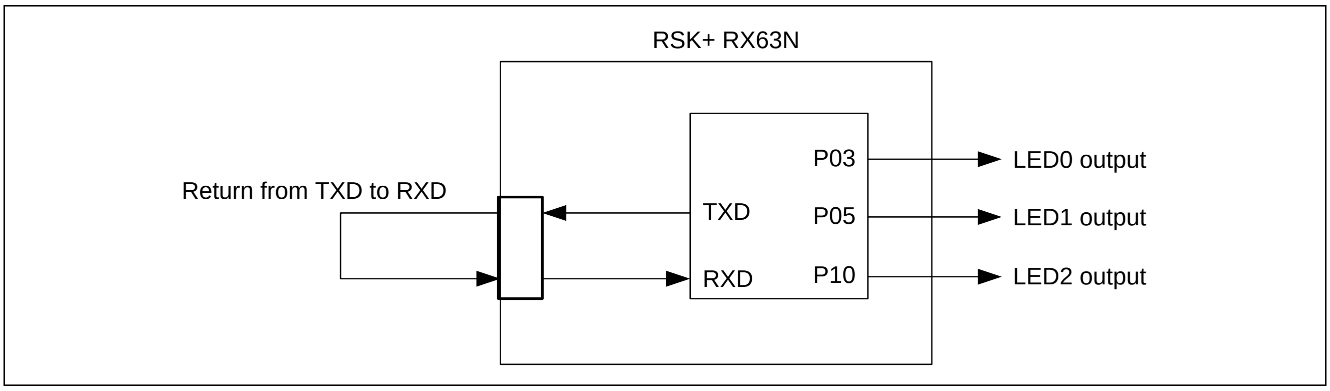


Figure 2.21 SCI Connection Specifications

List of related registers

The SCI0 interrupt-related registers and the interrupt sources on the SH7044 and RX631 are listed below. In order to reproduce the receive, transmit, transmit-end, and receive error interrupts of the SH7044 on the RX631, it is necessary to be aware of the resource settings for each and flags listed in table 2.40.

Table 2.40 SCI Interrupt-Related Resources (Asynchronous Communication)

	SH7044				RX631			
Item	**RXI0**	**TXI0**	**TEI0**	**ERI0**	**RXI0**	**TXI0**	**TEI0**	**ERI0**
Interrupt priority registers	IPRF (7-4)*[1]				IPR214*[1]			IPR114*[1]
Interrupt enable registers	SCR .RIE	SCR .TIE	SCR .TEIE	SCR .RIE	IER1A .IEN6*[1]	IER1A .IEN7*[1]	IER1B .IEN0*[1]	IER0E .IEN2*[1]
					SCR .RIE	SCR .TIE	SCR .TEIE	SCR .RIE
								GEN12 .EN0*[1]
Interrupt request registers (source flags)*[2]	SSR .RDRF	SSR .TDRE	SSR .TEND	SSR .ORER SSR .FER SSR .PER	IR214	IR215	IR216	IR114
								GRP12 .IS0
								SSR .ORER SSR .FER SSR .PER

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

The register symbols and full names are as follows:

- SH7044
  IPRF: Interrupt priority level setting register
  FSCR and SSR are listed in table 2.38.
- RX631
  IPRxxx: Interrupt source priority register (xxx: vector number)
  IER1A, IER1B, and IER0E: Interrupt request enable registers 1A, 1B, and 0E
  IRxxx: Interrupt request register (xxx: vector number)
  GENxx: Group xx interrupt enable register
  GRPxx: Group xx interrupt source register
  SCR and SSR are listed in table 2.38.

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

Table 2.41 SCI Asynchronous Communication Initial Setting Example (Common to Interrupt Method and Polling Method)

Procedure		SH7044 Setting Example Pϕ (Peripheral Clock): 20 MHz	RX631 Setting Example PCLK (Peripheral Clock): 48 MHz

1	Disable SCI interrupts.	The interrupt controller has no enable register.	IER1A.IEN6 = 0 (RXI0)
			IER1A.IEN7 = 0 (TXI0)
			IER1B.IEN0 = 0 (TEI0)
			IER0E.IEN2 = 0 (ERI0: group interrupt)
			GEN12.EN0 = 0 (ERI0: SCI0)

2	Cancel module stop state.	(No module stop function)	SYSTEM.PRCR = 0xA502
			SYSTEM.MSTPCRB.MSTPB31 = 0
			SYSTEM.PRCR = 0xA500

3	Initialize SCR.	SCR.TIE, RIE, TE, RE, TEIE = 0	SCR.TIE, RIE, TE, RE, TEIE = 0 Wait until SCR is cleared to 0.

4	Make I/O port settings (RX631 only)	PFC setting is performed in step 11.	PORT2.PODR.B0 = 1 (set to output 1)
			PORT2.PDR.B0 = 1 (set to output)
			PORT2.PDR.B1 = 0 (set to input)
			PORT2.PMR.B0 = 0 (set to general I/O)
			PORT2.PMR.B1 = 0 (set to general I/O)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1 (PFS write enables)
			MPC.P20PFS = 0x0A (TX pin setting)
			MPC.P21PFS = 0x0A (RX pin setting)
			MPC.PWPR.PFSWE = 0 (PFS write disabled)
			MPC.PWPR.B0WI = 1
			PORT2.PMR.B0 = 1 (set to peripheral function)
			PORT2.PMR.B1 = 1 (set to peripheral function)

5	Enable clock.	Internal clock/SCK pin: input SCR.CKE0, SCR.CKE1 = 00b	On-chip baud rate generator SCKn pin: I/O port SCR.CKE0, SCR.CKE1 = 00b
6	Initialize SIMR and SPMR.	—	SIMR.IICM = 0 SPMR.CKPH, CKPOL = 0 (Items that are the initial value are omitted.)

7	Make transmit and receive format settings.	SMR.C/_A = 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.CKS0, SMR.CKS1 = 00b
			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.CKS0, SMR.CKS1 = 00b

8	Make SCMR and SEMR settings.	(This function not implemented.)	SCMR.SMIF = 0
			(serial communication interface mode)
			SCMR.SINV = 0
			(no inversion of transmit and receive data)
			SCMR.SDIR = 0 (LSB-first)
			SEMR.ABCS = 0
			(transfer rate 1 bit period equal to 16 cycles of base clock)
			SEMR.NFEN = 0 (digital noise filter disabled)

9	Set bit rate (BRR).	38,400 bps BRR = 15	38,400 bps BRR = 38

Procedure		SH7044 Setting Example Pϕ (Peripheral Clock): 20 MHz	RX631 Setting Example PCLK (Peripheral Clock): 48 MHz
10	At initialization, wait one-bit period before enabling transmit and receive.	At initialization, transmit and receive are enabled after one-bit period ends.	←

11	Make I/O port settings (SH7044 only)	PFC setting is performed.
		PAIORL.PA1IOR = 1 (output)
		PAIORL.PA0IOR = 0 (input)
		PACRL2.PA1MD = 1 (TX0)
		PACRL2.PA0MD = 1 (RX0)
			Port setting is performed in step 4.

12	Clear interrupt sources.	—	IR214 = 0 (RXI0)
			IR215 = 0 (TXI0)

13	Set RIE, RE, TIE, and TE in SCR to "enabled" (enable transmit and receive).	SCR.RIE, RE = 1 SCR.TIE, TE = 1 Note: For polling: RIE and TIE in SCR are cleared to 0.	SCR.RIE, RE = 1 SCR.TIE, TE = 1 Note: Polling target is IR, so RIE and TIE in SCR are set to 1.

14	• Enable interrupts on interrupt controller.
	• Set priority.
	Note: For polling, skip step 14.
		INTC.IPRF.WORD = 0x0050 (level 5)	IPR214 = 0x05 (level 5)
			IPR114 = 0x05 (level 5)
			IER1A.IEN6 = 1 (RXI0)
			IER0E.IEN2 = 1 (ERI0)
			GEN12.EN0 = 1 (ERI0: SCI0)
			IER1A.IEN7 = 1 (TXI0)

Note: Shaded portions indicate places where polling settings differ.

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

Table 2.42 Example of Receive Data-Full Interrupt Handling during SCI Asynchronous Communication

Procedure	SH7044 Setting Example	RX631 Setting Example
Read receive data.	Read out contents of RDR to receive buffer.	Read out contents of RDR to receive buffer.
Clear receive data register-full flag.	After reading SSR.RDRF, clear to 0.	IR214 is cleared automatically.

End reception when the receive byte count reaches 32.	SCR.RIE = 0
	SCR.RE = 0
		SCR.RIE = 0
		SCR.RE = 0
		IER1A.IEN6 = 0 (RXI0)
		IER0E.IEN2 = 0 (ERI0: vector 114)
		GEN12.EN0 = 0 (ERI0: group 12)
		IR214 = 0

Table 2.43 Example of Transmit Data-Empty Interrupt Handling during SCI Asynchronous Communication

Procedure		SH7044 Setting Example	RX631 Setting Example
1	Write transmit data.	Write data to TDR.	Write data to TDR.
2	Clear the transmit data register-empty flag to 0.	After reading SSR.TDRE, clear to 0.	IR215 is cleared automatically.
3	Determine state of TEND.	If TEND is ON, go to step 4	If TEND is ON, go to step 4 (because transmission may not succeed if TE is cleared before data transmission completes).

4	End transmission when the transmit byte count reaches 32. (On RX631, perform TEND interrupt handling.)	SCR.TIE = 0
		SCR.TE = 0
		< TEND interrupt setting >
		SCR.TEIE = 1
			SCR.TIE = 0
			SCR.TE = 0
			IER1A.IEN7 = 0 (TXI0)
			IR215 = 0
			< TEND interrupt setting >
			IER1B.IEN0 = 1
			SCR.TEIE = 1

The details of the handling of errors and the TEND interrupt are not stipulated in the sample software. However, on the RX631 the receive error interrupt is assigned to a group interrupt. Therefore, it is necessary to detect the interrupt flag from the group.

Table 2.44 Example of Error Interrupt Handling during SCI Asynchronous Communication

Procedure		SH7044 Setting Example	RX631 Setting Example
1	Group interrupt determination	The SH7044 does not have group interrupts.	Continue with following processing if GRP12.IS0 (SCI0 receive error) is set to 1.
2	Overrun error determination	Perform error processing if SSR.ORER is set to 1.	Perform error processing if SSR.ORER is set to 1.
3	Framing error determination	Perform error processing if SSR.FER is set to 1.	Perform error processing if SSR.FER is set to 1.
4	Parity error determination	Perform error processing if SSR.PER is set to 1.	Perform error processing if SSR.PER is set to 1.

Table 2.45 Example of TEND Interrupt Handling during SCI Asynchronous Communication

Procedure		SH7044 Setting Example	RX631 Setting Example

1	Perform TEND interrupt handling.	Set TX port to GPIO.
		PACRL2.PA1MD = 0 (TX0)
		SCR.TEIE = 0
			Set TX port to GPIO.
			PORT2.PMR.B0 = 0 (GPIO)
			IER1B.IEN0 = 0
			SCR.TEIE = 0

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

In the polling method no interrupts are used, and step 13 in table 2.41, SCI Asynchronous Communication Initial Setting Example (Common to Interrupt Method and Polling Method), is extended as shown below.

Table 2.47 Example of Transmit and Receive Processing during SCI Asynchronous Communication (Polling Method)

Procedure		SH7044 Setting Example	RX631 Setting Example
Receive processing			
1	Read receive error and determine error type. ⇒ If receive error, 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 ORER, FER, or PER in SSR ≠ 0, go to receive error handling.
2	Monitor receive data register-full flag. ⇒ If ON, go to step 3. ⇒ If OFF, go to transmit processing.	If SSR.RDRF = 1, perform receive processing. ⇒ Go to step 3. If SSR.RDRF = 0, go to transmit processing.	If IR214 = 1, perform receive processing. ⇒ Go to step 3. If IR214 = 0, go to transmit processing.
3	Read receive data from RDR.	Read RDR 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 SSR.RDRF to 0.	Clear IR214 to 0.

5	If receive counter value is 32 bytes or more, end receive.	Receive is finished.
		SCR.RE = 0
			Receive is finished.
			SCR.RIE = 0
			SCR.RE = 0
			IR214 = 0

Transmit processing			
6	Monitor transmit data-empty flag. ⇒ If ON, go to step 7. ⇒ If OFF, go to receive processing.	If SSR.TDRE = 1, perform transmit processing. ⇒ If ON, go to step 7. ⇒ If OFF, go to receive processing.	If IR215 = 1, perform transmit processing. ⇒ If ON, go to step 7. ⇒ If OFF, go to receive processing.
7	Write transmit data to TDR.	Write transmit data to TDR.	Write transmit data to TDR.
8	Clear transmit data-empty flag.	Clear SSR.TDRE.	Clear IR215 to 0.

9	If receive counter value is 32 bytes or more, end transmit.	Transmit and receive are finished.
		SCR.TE = 0
			Transmit and receive are finished.
			SCR.TE = 0
			SCR.TIE = 0
			IR215 = 0

10	If transmit and receive are both finished, end processing. Otherwise, go to step 1.		
Error handling			
11	Receive error handling	The details of error handling are not stipulated.	The details of error handling are not stipulated.

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

A setting example for clock-synchronous master transmit processing using the serial communication interface (SCI) of the SH7044 and RX631 is described below.

< Specifications >

1. SCI0 on the RSK+RX63N board is used.
2. For the interrupt method, the transmit data-empty interrupt is used to start transmission.
3. For the polling method, no interrupts are used; the interrupt source flag (IR215) is polled and data transmission starts when the interrupt source is detected.
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.

Note: LED2 turns on if an error occurs.

Table 2.48 SCI Clock-Synchronous Communication Specifications (Master Transmit)

Item	Description	Remarks
Communication mode	Clock-synchronous serial communication	
Transfer speed	100 kbps	B = 119
Data length	8 bits	
Hardware flow control	None	
SCI channel used	SCI0 fixed	
Bit order	LSB-first	
Synchronous clock	Internal clock	The SCK pin is the sync clock output.
Pins used	P20/TXD0	
	P22/SCK0	
	P03/GPIO	LED0 output
	P05/GPIO	LED1 output
	P10/GPIO	LED2 output

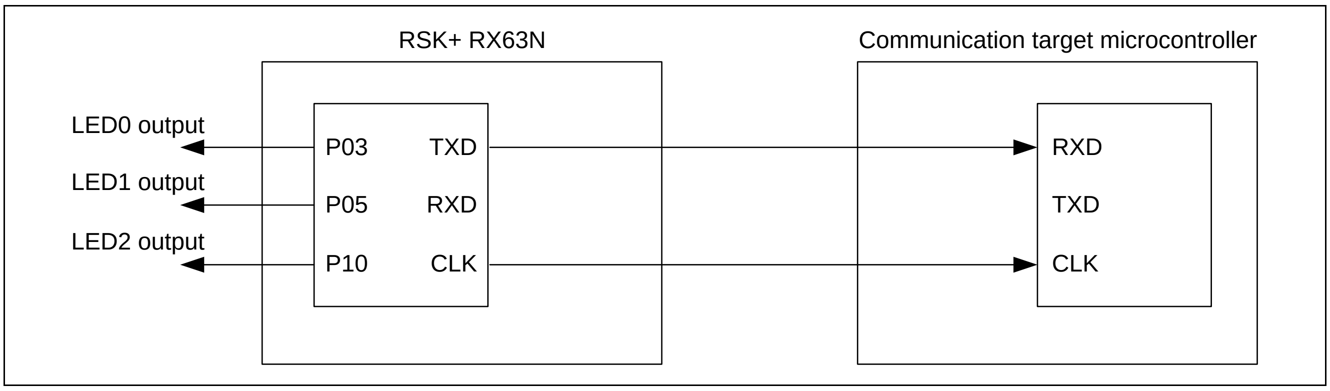


Figure 2.22 Clock-Synchronous Serial Communication Connection Specifications (Master Transmit)

List of related registers

The SCI0 interrupt-related registers and the interrupt sources on the SH7044 and RX631 are listed below. In order to reproduce the receive, transmit, transmit-end, and receive error interrupts of the SH7044 on the RX631, it is necessary to be aware of the resource settings for each and flags listed in table 2.49. Unlike asynchronous communication, overrun error is the only error interrupt source.

Table 2.49 SCI Interrupt-Related Resources (Clock Synchronous Communication)

	SH7044				**RX631**			
Item	**RXI0**	**TXI0**	**TEI0**	**ERI0**	**RXI0**	**TXI0**	**TEI0**	**ERI0**
Interrupt priority registers	IPRF (7-4)*[1]				IPR214*[1]			IPR114*[1]
Interrupt enable registers	SCR .RIE	SCR .TIE	SCR .TEIE	SCR .RIE	IER1A .IEN6*[1]	IER1A .IEN7*[1]	IER1B .IEN0*[1]	IER0E .IEN2*[1]
					SCR .RIE	SCR .TIE	SCR .TEIE	SCR .RIE
								GEN12 .EN0*[1]
Interrupt request registers (source flags)*[2]	SSR .RDRF	SSR .TDRE	SSR .TEND	SSR .ORER	IR214	IR215	IR216	IR114
								GRP12 .IS0
								SSR .ORER

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

The register symbols and full names are as follows:

- SH7044
  IPRF: Interrupt priority level setting register F
  SCR and SSR are listed in table 2.38.
- RX631
  IPRxxx: Interrupt source priority register (xxx: vector number)
  IER1A, IER1B, and IER0E: Interrupt request enable registers 1A, 1B, and 0E
  IRxxx: Interrupt request register (xxx: vector number)
  GENxx: Group xx interrupt enable register
  GRPxx: Group xx interrupt source register
  SCR and SSR are listed in table 2.38.

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.

Table 2.50 SCI Clock-Synchronous Master Transmit Initial Setting Example

Procedure		SH7044 Setting Example Pϕ (Peripheral Clock): 20 MHz	RX631 Setting Example PCLK (Peripheral Clock): 48 MHz

1	Disable SCI interrupts.	(This function not implemented.)	IER1A.IEN6 = 0 (RXI0)
			IER1A.IEN7 = 0 (TXI0)
			IER1B.IEN0 = 0 (TEI0)
			IER0E.IEN2 = 0 (ERI0: group interrupt)
			GEN12.EN0 = 0 (ERI0: SCI0)

2	Cancel module stop state.	(No module stop function)	SYSTEM.PRCR = 0xA502
			SYSTEM.MSTPCRB.MSTPB31 = 0
			SYSTEM.PRCR = 0xA500

3	Initialize SCR.	SCR.TIE, RIE, TE, RE, TEIE = 0	SCR.TIE, RIE, TE, RE, TEIE = 0 Wait until SCR is cleared to 0.

4	Make I/O port settings (RX631 only)	PFC setting is performed in step 11.	PORT2.PODR.B0 = 1 (set to output 1)
			PORT2.PODR.B2 = 1 (set to output 1)
			PORT2.PDR.B0 = 1 (TX output)
			PORT2.PDR.B2 = 1 (SCK output)
			PORT2.PMR.B0 = 0 (GPIO)
			PORT2.PMR.B2 = 0 (GPIO)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1 (PFS write enables)
			MPC.P20PFS = 0x0A (TX pin setting)
			MPC.P22PFS = 0X0A (SCK pin setting)
			MPC.PWPR.PFSWE = 0 (PFS write disabled)
			MPC.PWPR.B0WI = 1

5	Enable clock.	Internal clock/SCK pin: output SCR.CKE0, SCR.CKE1 = 00b	On-chip baud rate generator SCKn pin: output port SCR.CKE0, SCR.CKE1 = 00b
6	Initialize SIMR and SPMR.	(This function not implemented.)	SIMR.IICM = 0 SPMR.CKPH, CKPOL = 0 (Items that are the initial value are omitted.)
7	Make transmit and receive format settings.	SMR.C/_A = 1 (clock-synchronous) SMR.CKS0, SMR.CKS1 = 00b	SMR.CM = 1 (clock-synchronous) SMR.CKS0, SMR.CKS1 = 00b

8	Make SCMR settings.	(This function not implemented.)	SCMR.SMIF = 0
			(serial communication interface mode)
			SCMR.SINV = 0
			(no inversion of transmit and receive data)
			SCMR.SDIR = 0 (LSB-first)

9	Set bit rate (BRR).	100 kbps BRR = 49	100 kbps BRR = 119
10	At initialization, wait one-bit period before enabling transmit.	At initialization, transmit is enabled after one-bit period ends.	←

11	Make I/O port settings (SH7044 only)	PFC setting is performed.
		PAIORL.PA1IOR = 1 (output)
		PAIORL.PA0IOR = 0 (input)
		PAIORL.PA2IOR = 1 (output)
		PACRL2.PA1MD = 1 (TX0)
		PACRL2.PA0MD = 1 (RX0)
		PACRL2.PA2MD0,
		PACRL2.PA2MD1 = 01b (SCK0)
			Implemented in step 4.

Procedure		SH7044 Setting Example Pϕ (Peripheral Clock): 20 MHz	RX631 Setting Example PCLK (Peripheral Clock): 48 MHz

12	• Enable interrupts on interrupt controller.
	• Set priority.
	• Clear interrupt sources.
		INTC.IPRF.WORD = 0x0050 (level 5)
		SCR.TIE = 1
		SCR.TE = 1
		Note: For polling: TIE in SCR are cleared to 0.
			IPR214 = 0x05 (level 5)*
			IPR114 = 0x05 (level 5)*
			IR215 = 0
			SCR.TIE, TE, TEIE = 1
			(both turned ON simultaneously)
			PORT2.PMR.B0 = 1 (peripheral function)
			PORT2.PMR.B2 = 1 (peripheral function)
			IER1A.IEN7 = 1 (TXI0)*
			Note: * Not set when polling is used.

Note: Shaded portions indicate places where polling settings differ.

Transmit interrupt handling during SCI clock-synchronous master transmit operation (interrupt handling method) is described below.

Table 2.51 Example of Transmit Interrupt Handling during SCI Clock-Synchronous Master Transmit Operation (Interrupt Handling Method)

Procedure		SH7044 Setting Example	RX631 Setting Example
1	Write transmit data to TDR.	Write data to TDR.	Write data to TDR.
2	Clear the transmit data register-empty flag to 0.	After reading SSR.TDRE, clear to 0.	IR215 is cleared automatically.

3	End transmission when the transmit byte count reaches 32. (On RX631, perform TEND interrupt handling.)	SCR.TIE = 0	SCR.TIE = 0
			IER1A.IEN7 = 0 (TXI0)
			IR215 = 0
			< TEND interrupt setting >
			Determines that SSR.TEND == 1 and sets IER1B.IEN0 = 1.

Table 2.52 Example of TEND Interrupt Handling during SCI Clock-Synchronous Master Transmit Operation (Interrupt Handling Method)

Procedure		SH7044 Setting Example	RX631 Setting Example

1	Perform TEND interrupt handling.	Perform TEND interrupt handling (details not stipulated).	Set TX port to GPIO.
			PORT2.PMR.B0 = 0 (GPIO)
			IER1B.IEN0 = 0
			SCR.TEIE = 0

SCI clock-synchronous master transmit processing (polling method) is described below. In the polling method no interrupts are used. As the procedure, step 12 in table 2.53, SCI Clock-Synchronous Master Transmit Initial Setting Example, is extended as shown below.

Table 2.54 Example of SCI Clock-Synchronous Master Transmit Processing (Polling Method)

Procedure		SH7044 Setting Example	RX631 Setting Example
1	Poll transmit data-empty flag. Run transmit processing when transmit-empty occurs.	Polling target: SSR.TDRE = 1 If SSR.TDRE = 1, perform transmit processing in step 2 and after.	Polling target: IR215 If IR215 = 1, perform transmit processing in step 2 and after.
2	Write transmit data to TDR.	Write transmit data to TDR.	Write transmit data to TDR.
3	Clear transmit data-empty flag.	Clear SSR.TDRE.	Clear IR215 to 0.

4	End transmission when the transmit byte count reaches 32.	SCR.TIE = 0	SCR.TIE = 0
			IR215 = 0
			Note: Handling of TEND is up to the user.

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

A setting example for clock-synchronous slave receive processing using the serial communication interface (SCI) of the SH7044 and RX631 is presented below.

< Slave Receive Processing >

1. SCI0 on the RSK+RX63N board is used.
2. For the interrupt method, the receive data register-full interrupt is used to start receive processing.
3. For the polling method, no interrupts are used; the interrupt source flag (IR214) is polled. Data reception takes place when an interrupt request is detected.
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.

Table 2.55 SCI Clock-Synchronous Communication Specifications (Slave Receive)

Item	Description	Remarks
Communication mode	Clock-synchronous serial communication	
Transfer speed	100 kbps	B = 119
Data length	8 bits	
Hardware flow control	None	
SCI channel used	SCI0 fixed	
Bit order	LSB-first	
Synchronous clock	External clock	The SCK pin is the sync clock input.
Pins used	P21/RXD0	
	P22/SCK0	
	P03/GPIO	LED0 output
	P05/GPIO	LED1 output
	P10/GPIO	LED2 output

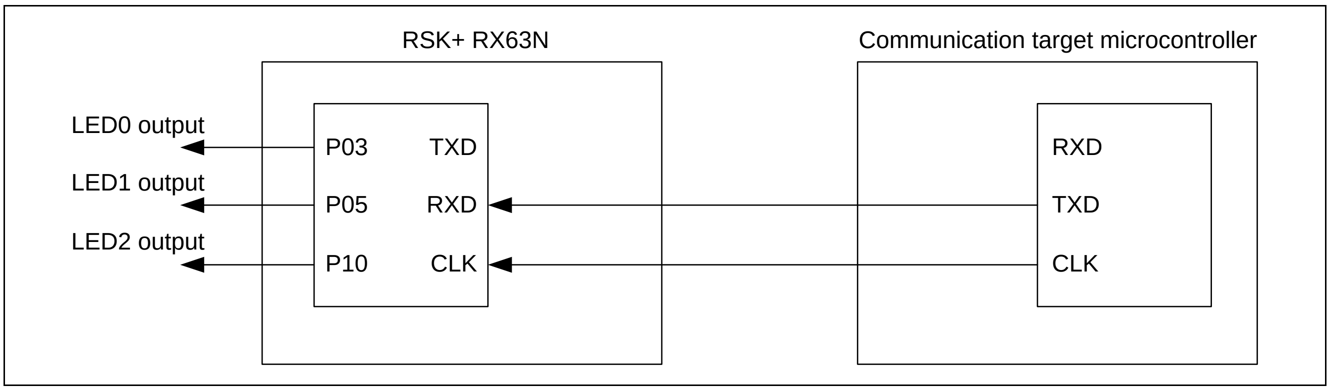


Figure 2.23 Clock-Synchronous Serial Communication Connection Specifications (Slave Receive)

The initial setting procedure for SCI clock-synchronous slave receive operation is shown below. Note that the initial setting processing is common to the interrupt method and the polling method. For information on interrupt-related resources, see table 2.49.

Table 2.56 SCI Clock-Synchronous Slave Receive Initial Setting Example

Procedure		SH7044 Setting Example Pϕ (Peripheral Clock): 20 MHz	RX631 Setting Example PCLK (Peripheral Clock): 48 MHz

1	Disable SCI interrupts.	The interrupt controller has no enable register.	IER1A.IEN6 = 0 (RXI0)
			IER1A.IEN7 = 0 (TXI0)
			IER1B.IEN0 = 0 (TEI0)
			IER0E.IEN2 = 0 (ERI0: group interrupt)
			GEN12.EN0 = 0 (ERI0: SCI0)

2	Cancel module stop state.	(No module stop function)	SYSTEM.PRCR = 0xA502
			SYSTEM.MSTPCRB.MSTPB31 = 0
			SYSTEM.PRCR = 0xA500

3	Initialize SCR.	SCR.TIE, RIE, TE, RE, TEIE = 0	SCR.TIE, RIE, TE, RE, TEIE = 0 Wait until SCR is cleared to 0.

4	Make I/O port settings (RX631 only)	PFC setting is performed in step 11.	PORT2.PDR.B1 = 0 (RX input)
			PORT2.PDR.B2 = 0 (SCK input)
			PORT2.PMR.B1 = 0 (GPIO)
			PORT2.PMR.B2 = 0 (GPIO)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enables)
			MPC.P21PFS = 0x0A (RX pin setting)
			MPC.P22PFS = 0X0A (SCK pin setting)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1
			PORT2.PMR.B1 = 1 (peripheral function)
			PORT2.PMR.B2 = 1 (peripheral function)

5	Enable clock.	External clock/SCK pin clock input SCR.CKE0, SCR.CKE1 = 10b	External clock/SCKn pin as input port SCR.CKE0, SCR.CKE1 = 10b
6	Initialize SIMR and SPMR	(No such setting on the SH7044)	SIMR.IICM = 0 SPMR.CKPH, CKPOL = 0 (Items that are the initial value are omitted.)
7	Make transmit and receive format settings.	SMR.C/_A = 1 (clock-synchronous) SMR.CKS0, SMR.CKS1 = 00b	SMR.CM = 1 (clock-synchronous) SMR.CKS0, SMR.CKS1 = 00b

8	Make SCMR settings.	(No such setting on the SH7044)	SCMR.SMIF = 0
			(serial communication interface mode)
			SCMR.SINV = 0
			(no inversion of transmit and receive data)
			SCMR.SDIR = 0 (LSB-first)

9	Set bit rate (BRR).	100 kbps BRR = 49	100 kbps BRR = 119
10	At initialization, wait one-bit period before enabling receive.	At initialization, transmit/receive is enabled after one-bit period ends.	←

11	Make I/O port settings (SH7044 only)	PFC setting is performed.
		PAIORL.PA1IOR = 1 (output)
		PAIORL.PA0IOR = 0 (input)
		PAIORL.PA2IOR = 0 (input)
		PACRL2.PA1MD = 1 (TX0)
		PACRL2.PA0MD = 1 (RX0)
		PACRL2.PA2MD0, PACRL2.PA2MD1 = 01b (SCK0)
			Implemented in step 4.

Procedure	SH7044 Setting Example Pϕ (Peripheral Clock): 20 MHz	RX631 Setting Example PCLK (Peripheral Clock): 48 MHz

12 • Enable interrupts on interrupt controller.
• Set priority.
• Clear interrupt sources.
	INTC.IPRF.WORD = 0x0050 (level 5)
	SCR.RIE = 1
	SCR.RE = 1
	Note: For polling: RIE in SCR are cleared to 0.
		IPR214 = 0x05 (level 5)*
		IPR114 = 0x05 (level 5)*
		IR214 = 0
		SCR.RIE, RE = 1
		(both turned ON simultaneously)
		IER1A.IEN6 = 1 (RXI0)*
		IER0E.IEN2 = 1 (ERI0)*
		GEN12.EN0 = 1 (ERI0: SCI0)*
		Note: * Not set by processing when polling is used.

Note: Shaded portions indicate places where polling settings differ.

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

Table 2.57 Example of Interrupt Handling during SCI Clock-Synchronous Slave Receive Operation (Interrupt Handling Method)

Procedure		SH7044 Setting Example	RX631 Setting Example
1	Read receive data.	Read out contents of RDR to receive buffer.	Read out contents of RDR to receive buffer.
2	Clear receive data register-full flag.	After reading SSR.RDRF, clear to 0.	IR214 is cleared automatically.

4	End reception when the receive byte count reaches 32.	SCR.RIE = 0	SCR.RIE = 0
			IER1A.IEN6 = 0 (RXI0)
			IER0E.IEN2 = 0
			(ERI0: group interrupt)
			GEN12.EN0 = 0 (ERI0: SCI0)
			IR214 = 0

During clock-synchronous communication overrun errors are the only receive errors detected. Implement error handler code to accommodate overrun errors. On the RX631 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 12 in table 2.56, SCI Clock-Synchronous Slave Receive Initial Setting Example, is extended as shown below.

Table 2.59 Example of SCI Clock-Synchronous Slave Receive (Polling Method)

Procedure		SH7044 Setting Example	RX631 Setting Example
1	Read receive error and determine error type. ⇒ If receive error, go to receive error handling. ⇒ If not receive error, go to step 2.	If ORER in SSR ≠ 0, go to receive error handling.	If ORER in SSR ≠ 0, go to receive error handling.
2	Poll the receive data register-full flag, and if the register is full perform receive processing in step 3 and after.	Polling target: SSR.RDRF If SSR.RDRF = 1, perform receive processing.	If IR214 = 1, perform receive processing. ⇒ Go to step 3. If IR214 = 0, go to receive processing.
3	Read receive data from RDR.	Read RDR 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 SSR.RDRF to 0.	Clear IR214 to 0.

5	If receive counter value is 32 bytes or more, end receive.	Receive is finished.
		SCR.RIE = 0
			Receive is finished.
			SCR.RIE = 0
			IR214 = 0

Error handling			
6	Receive error handling	The details of error handling are not stipulated.	The details of error handling are not stipulated.

2.10 Mid-Speed A/D Converter

2.10.1 Comparison of Specifications

The functions and features of the mid-speed A/D converter on the SH7044, and the 10-bit A/D converter (ADb) and 12-bit A/D converter (S12ADa) on the RX631, are compared below.

Table 2.60 Comparison of Mid-Speed A/D Converter Specifications on SH7044 and RX631

	SH7044	**RX631**	
Item	**Mid-Speed A/D Converter**	**10-Bit A/D Converter (ADb)**	**12-Bit A/D Converter (S12ADa)**
Resolution	10 bits	10 bits	12 bits
Number of input channels	8 channels (4 channels × 2)	8 channels + 1 extended channel	Max. 21 channels
A/D conversion method	Successive approximation	Successive approximation	Successive approximation
Conversion speed	6.7 µs per channel (operating frequency: 20 MHz, CKS = 1)	1.0 µs per channel (PCLK: 50 MHz)	1.0 µs per channel (ADCLK: 50 MHz)
Conversion modes	• Single mode • Scan mode	• Single channel mode • Scan mode — Continuous scan mode — Single scan mode	• (No single channel mode) • Scan mode — Continuous scan mode — Single scan mode
A/D conversion start conditions	• Software trigger • Trigger by timer (MTU) • Asynchronous trigger (ADTRG pin)	• Software trigger • Trigger by timer (MTU, TPU, TMR) • Asynchronous trigger (ADTRG# pin)	• Software trigger • Trigger by timer (MTU, TPU, TMR) • Asynchronous trigger (ADTRG0# pin)
Other functions	• Support for simultaneous conversion of 2 channels	• Adjustable number of sampling states • Self-diagnostic function	• Adjustable number of sampling states • A/D-converted value addition mode
Operations linked to A/D conversion-end interrupt	• CPU interrupt generation • DMAC or DTC activation	• CPU interrupt generation • DMAC or DTC activation	• CPU interrupt generation • DMAC or DTC activation
Low power consumption function	None	Support for module stop state setting	Support for module stop state setting
Conversion targets	AN pin	AN pin Self-diagnostic (fault detection)	AN pin Internal reference voltage Temperature sensor

2.10.2 Input Channels and Operation

The mid-speed A/D converter of the SH7044 and the A/D converters of the RX631 differ as described below.

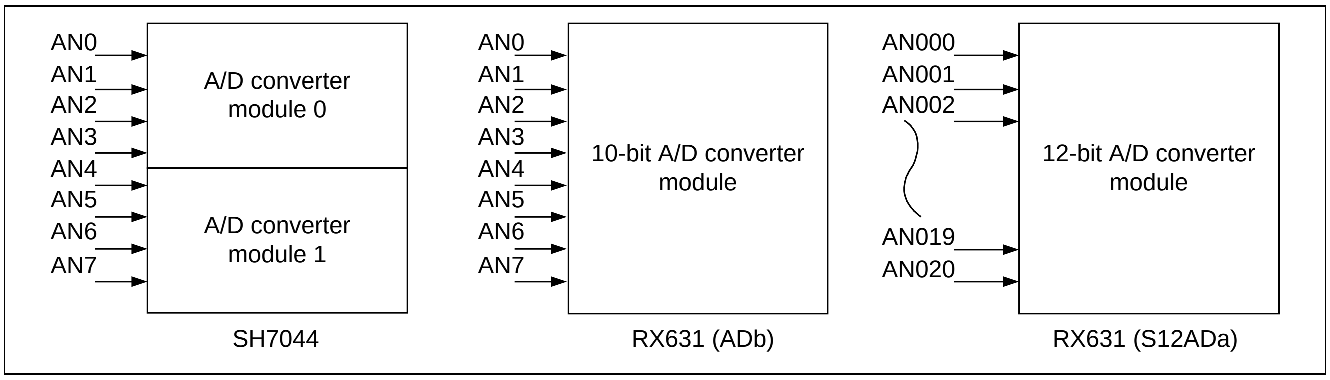


Figure 2.24 Comparison of SH7044 and RX631 A/D Converter Configurations

As shown in figure 2.24, each module of the SH7044's A/D converter supports four analog input channels. The two modules can operate at the same time, but continuous scan bridging both modules is not supported. The ADb and S12ADa A/D converters of the RX631 support eight and 21 channels, respectively, but each is a single converter module. The A/D converters of the RX631 can perform sequential A/D conversion of the input on specified channels, but they cannot convert multiple channels simultaneously. The scanning sequence of each module is listed below.

Table 2.61 A/D Converter Conversion Sequence (All Channels Specified)

Microcontroller	A/D Converter	Conversion Sequence
SH7044	AD0	AN0 ⇒ AN1 ⇒ AN2 ⇒ AN3
	AD1	AN4 ⇒ AN5 ⇒ AN6 ⇒ AN7
RX631	ADb	AN0 ⇒ AN1 ⇒ AN2 ⇒ AN3 ⇒ AN4 ⇒ AN5 ⇒ AN6 ⇒ AN7
	S12ADa	AN000 ⇒ AN001 ⇒ AN002 ⇒⇒⇒ Omitted ⇒⇒ AN019 ⇒ AN020

2.10.3 Operating Modes

The mid-speed A/D converter of the SH7044 has two operating modes: single mode and scan mode.

Table 2.62 lists the conversion modes of the SH7044 and the equivalent conversion modes on the RX631.

Table 2.62 Correspondence of A/D Converter Operating Modes

No.	SH7044 (Mid-Speed A/D Converter)	RX631 (ADb)	RX631 (S12ADa)
1	Single mode	Single channel mode	Single scan mode (1 channel only specified)
2	Scan mode (single-cycle conversion end)	Scan mode (single scan mode)	Single scan mode (multiple channels specified)
3	Scan mode (continuous conversion)	Scan mode (continuous scan mode)	Continuous scan mode

An overview of the various modes is provided below.

Table 2.63 Overview of A/D Converter Operating Modes

Microcontroller	Operating Mode	Operational Overview
SH7044	Single mode	A/D conversion is performed once on the single specified channel only. If interrupts are enabled, an ADI interrupt is generated.
	Scan mode	Conversion is performed successively on analog input from the specified channels (or channel), starting from the lowest-numbered channel. When conversion of all the specified channels finishes (single-cycle conversion end), an ADI interrupt is generated. If conversion has not finished, it continues.
RX631 (ADb)	Single channel mode	A/D conversion is performed once on the single specified channel only. If interrupts are enabled, an ADI0 interrupt is generated.
	Scan mode	Single scan mode: • Conversion is performed successively on analog input from the specified channels (or channel), starting from the lowest-numbered channel. • In single scan mode conversion is performed for one cycle only. • When a single conversion cycle finishes, an ADI0 interrupt is generated.
		Continuous scan mode: The above single scan mode operation is repeated multiple times.
RX631 (S12ADa)	Single scan mode	Conversion is performed successively on analog input from the specified channels (or channel), starting from the lowest-numbered channel. When conversion of all the specified channels finishes (single-cycle conversion end), an S12ADI0 interrupt is generated. In single scan mode conversion is performed for one cycle only.
	Continuous scan mode	Single scan mode is repeated multiple times on the S12ADa.

2.10.4 Module Stop

The initial state of the peripheral modules of the RX631 is stopped, due to the low power consumption function. The initial state of the A/D converter modules (ADb and S12ADa) is also stopped. Do not fail to cancel the module stop state when making settings to these modules. Before accessing the module stop control register to cancel the module stop state, first cancel register write protection.

2.10.5 A/D Converter Single Channel Mode Setting Example

A setting example for switching from the SH7044 (single mode) to the RX631 (single channel mode) is shown below.

< Single Channel Mode Specifications >

1. The 10-bit A/D converter on the RSK+RX63N board is used.
2. The A/D conversion start timing is based on the MTU4 compare match trigger.
3. AN0 is used for analog input, and the operating mode is single channel mode. An ADI0 interrupt is generated when conversion finishes, and the result is stored in the RAM.

The above operations are repeated multiple times.

Table 2.64 10-Bit A/D Converter Setting Specifications

Item	Description	Remarks
Channel used	AN0	
Interrupt handling	A/D conversion-end interrupt (ADI0 interrupt)	
Operating mode	Single channel mode	SH7044 single mode
Clock selection	PCLK/2	PCLK = 48 MHz
Conversion start trigger and cycle	MTU4 compare match A (1 ms cycle)*	
Extended analog input	Not used	
Data alignment	Flush-left	Only used for AN0:ADDRA
Pins used	PE02/AN0	Analog input

Note: * Refer to tables 2.32 and 2.33 for MTU settings.

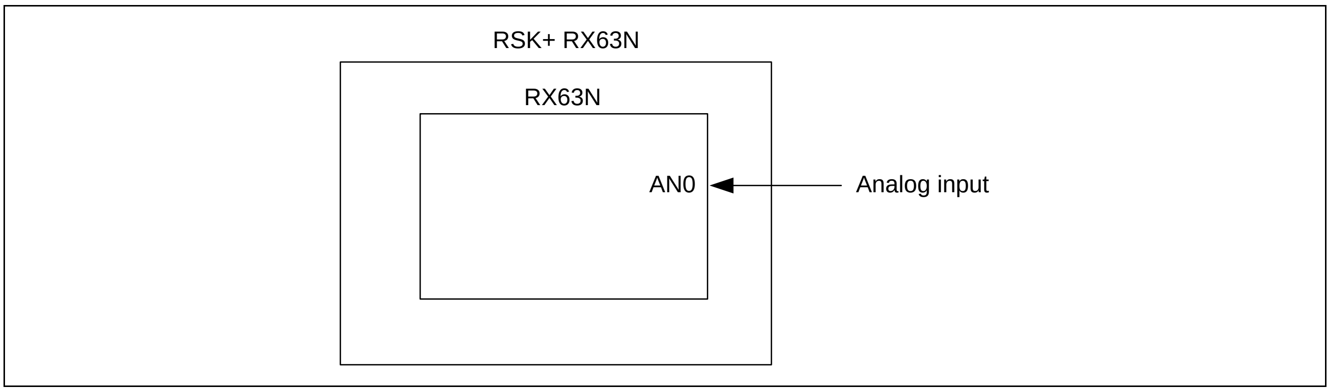


Figure 2.25 10-Bit A/D Converter Setting Connection Specifications

A setting example for switching from the mid-speed A/D converter on the SH7044 to the 10-bit A/D converter on the RX631 is shown below.

Table 2.65 10-Bit A/D Converter Initial Setting Example

Procedure		SH7044 Setting Example Pϕ (Peripheral Clock): 20 MHz	RX631 Setting Example PCLK (Peripheral Clock): 48 MHz

1	Cancel module stop state.	(No module stop function)	SYSTEM.PRCR = 0xA502
			SYSTEM.MSTPCRA.MSTPA23 = 0
			SYSTEM.PRCR = 0xA500

2	Disable interrupts.	ADCSR0.ADIE = 0
		(interrupts disabled)
		ADCSR1.ADIE = 0
		(interrupts disabled)
			IER0C.IEN2 = 0
			(vector 98, ADI0 disabled)
			ADCSR.ADIE = 0
			(interrupts disabled)

3	Disable A/D converter.	ADCSR0.ADST = 0
		(A/D0 disabled)
		ADCSR1.ADST = 0
		(A/D1 disabled)
			ADCSR.ADST = 0 (A/D disabled)

4	Make I/O port settings (pin I/O and pin function settings).	I/O settings are not needed because AN0 is assigned as an input-only port (PFDR).	Pin AN0 set in MPC
			PORTE.PDR.B2 = 0 (input)
			PORTE.PMR.B2 = 0 (GPIO)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enables)
			MPC.PE2PFS = 0x80 (analog function)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1

5	Set operating mode.
	Select clock.
	Specify channel.
	Set start trigger.
		ADCSR0.SCAN = 0
		(single mode)
		ADCSR0.CKS = 0
		(conversion duration: 266 states)
		ADCSR0.CH1, ADCSR0.CH0
		= 0 (AN0)
		ADCR0.TRGE = 1
		(trigger enabled)
			ADCSR.CH = 000b (AN0)
			ADCR.MODE = 0
			(single channel mode)
			ADCR.CKS = 2 (PCLK/2)
			ADCR.TRGS = 001b
			(MTU0 to MTU4 trigger enabled)

6	ADDR format	Only flush-left supported, so no setting needed.	ADCR2.DPSEL = 1 (data flush-left)
7	Make settings to interrupt priority register.	IPRG.WORD = 0x5000 (A/D0 and A/D1: level 5)	IPR098 = 5 (ADI0: level 5)

8	Enable interrupts.	ADCSR0.ADIE = 1
		(interrupts enabled)
			IER0C.IEN2 = 1
			(vector 98, ADI0 enabled)
			ADCSR.ADIE = 1 (interrupts enabled)

9	Start A/D conversion.	ADCSR0.ADST = 1 (A/D0 start) Note: Simultaneous A/D0 and A/D1 conversion start when using external trigger.	ADCSR.ADST = 1 (A/D0 start) Note: The sample program uses MTU4 to start A/D conversion.
10	Perform handling of A/D conversion-end interrupt.	ADCSR.ADF = 0 • Read interrupt flag and clear to 0.	The interrupt flag is cleared automatically.

2.10.6 A/D Converter Continuous Scan Mode Setting Example

Settings for continuous scan mode operation on the SH7044 and RX631 are shown below.

< Continuous Scan Mode Specifications >

1. The 10-bit A/D converter on the RSK+RX63N board is used.
2. The A/D conversion start timing is based on the software trigger.
3. Three channels of analog input are used, AN0, AN1, and AN2, and the operating mode is continuous scan mode.

When conversion finishes the conversion result is stored in the RAM by the handler of the ADI0 interrupt.

Table 2.66 10-Bit A/D Converter Setting Specifications

Item	Description	Remarks
Channel used	AN0, AN1, and AN2	
Interrupt handling	A/D conversion-end interrupt (ADI0 interrupt)	
Operating Mode	Continuous scan mode	SH7044 scan mode
Clock selection	PCLK/2	PCLK = 48 MHz
Conversion start trigger and cycle	Software trigger (Conversion repeats after start.)	
Extended analog input	Not used	
Data alignment	Flush-left	AN0: ADDRA AN1: ADDRB AN2: ADDRC
Pins used	PE02/AN0	Analog input 0
	PE03/AN1	Analog input 1
	PE04/AN2	Analog input 2

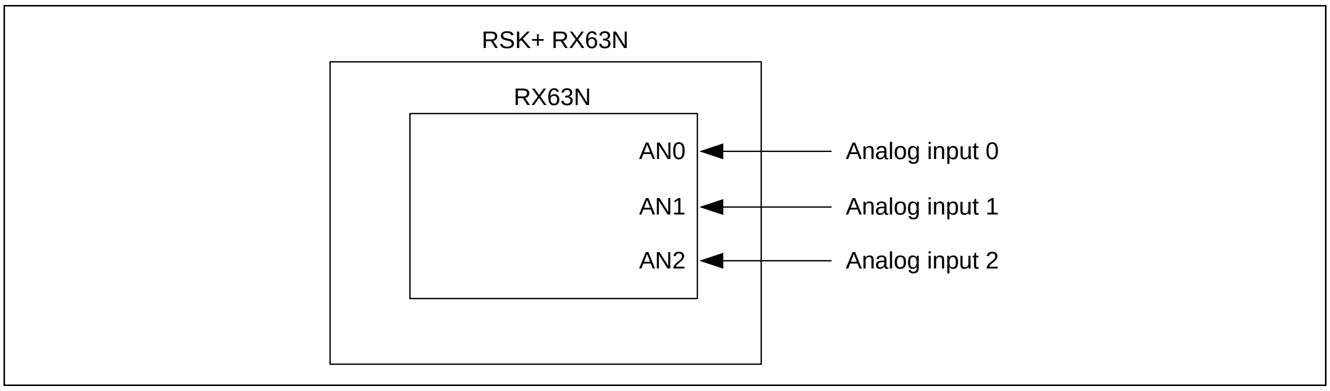


Figure 2.26 10-Bit A/D Converter Setting Connection Specifications

An initial setting example for switching from the mid-speed A/D converter on the SH7044 to the 10-bit A/D converter on the RX631 is shown below.

Table 2.67 10-Bit A/D Converter Initial Setting Example

Procedure		SH7044 Setting Example Pϕ (Peripheral Clock): 20 MHz	RX631 Setting Example PCLK (Peripheral Clock): 48 MHz

1	Cancel module stop state.	(No module stop function)	SYSTEM.PRCR = 0xA502
			SYSTEM.MSTPCRA.MSTPA23 = 0
			SYSTEM.PRCR = 0xA500

2	Disable interrupts.	ADCSR0.ADIE = 0 (interrupts disabled)
		ADCSR1.ADIE = 0 (interrupts disabled)
			IER0C.IEN2 = 0 (vector 98, ADI0 disabled)
			ADCSR.ADIE = 0 (interrupts disabled)

3	Disable A/D converter.	ADCSR0.ADST = 0 (A/D0 disabled)
		ADCSR1.ADST = 0 (A/D1 disabled)
			ADCSR.ADST = 0 (A/D disabled)

4	Make I/O port settings (pin I/O and pin function settings).	I/O settings are not needed because AN0 is assigned as an input-only port (PFDR).	Pin AN0 set in MPC
			PORTE.PDR.B2 = 0 (input)
			PORTE.PDR.B3 = 0 (input)
			PORTE.PDR.B4 = 0 (input)
			PORTE.PMR.B2 = 0 (GPIO)
			PORTE.PMR.B3 = 0 (GPIO)
			PORTE.PMR.B4 = 0 (GPIO)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1 (PFS write enables)
			MPC.PE2PFS = 0x80 (analog function)
			MPC.PE3PFS = 0x80 (analog function)
			MPC.PE4PFS = 0x80 (analog function)
			MPC.PWPR.PFSWE = 0 (PFS write disabled)
			MPC.PWPR.B0WI = 1

5	Set operating mode.
	Select clock.
	Specify channel.
	Set start trigger.
		ADCSR0.SCAN = 1 (scan mode)
		ADCSR0.CKS = 0 (conversion duration: 266 states)
		ADCSR0.CH1, ADCSR0.CH0 = 10b (AN0, AN1, and AN2)
		ADCR0.TRGE = 0 (software trigger)
			ADCSR.CH = 010b (AN0, AN1, and AN2)
			ADCR.MODE = 2 (continuous scan mode)
			ADCR.CKS = 2 (PCLK/2)
			ADCR.TRGS = 0 (software trigger)

6	ADDR format	Only flush-left supported, so no setting needed.	ADCR2.DPSEL = 1 (data flush-left)
7	Make settings to interrupt priority register.	IPRG.WORD = 0x5000 (A/D0 and A/D1: level 5)	IPR098 = 5 (ADI0: level 5)

8	Enable interrupts.	ADCSR0.ADIE = 1 (interrupts enabled)	IER0C.IEN2 = 1 (vector 98, ADI0 enabled)
			ADCSR.ADIE = 1 (interrupts enabled)

9	Start A/D conversion.	ADCSR0.ADST = 1 (A/D0 start)	ADCSR.ADST = 1 (A/D start)
10	Perform handling of A/D conversion-end interrupt.	ADCSR.ADF = 0 • Read interrupt flag and clear to 0.	The interrupt flag is cleared automatically.

2.11 High-Speed A/D Converter

2.11.1 Comparison of Specifications

The functions and features of the high-speed A/D converter on the SH7044, and the 10-bit A/D converter (ADb) and 12-bit A/D converter (S12ADa) on the RX631, are compared below.

Table 2.68 Comparison of High-Speed A/D Converter Specifications on SH7044 and RX631

	SH7044	**RX631**	
Item	**High-Speed A/D Converter**	**10-Bit A/D Converter (ADb)**	**12-Bit A/D Converter (S12ADa)**
Resolution	10 bits	10 bits	12 bits
Number of input channels	8 channels	8 channels + 1 extended channel	Max. 21 channels
A/D conversion method	Successive approximation	Successive approximation	Successive approximation
Conversion speed	2.9 μs per channel (operating frequency: 28 MHz)	1.0 μs per channel (PCLK: 50 MHz)	1.0 μs per channel (ADCLK: 50 MHz)
Operating modes	• Selectable between select mode and group mode • Selectable between single mode and scan mode	• Single channel mode • Scan mode — Continuous scan mode — Single scan mode	• (No single channel mode) • Scan mode — Continuous scan mode — Single scan mode
A/D conversion start conditions	• Software trigger • Trigger by timer (MTU) • Asynchronous trigger (ADTRG pin)	• Software trigger • Trigger by timer (MTU, TPU, TMR) • Asynchronous trigger (ADTRG# pin)	• Software trigger • Trigger by timer (MTU, TPU, TMR) • Asynchronous trigger (ADTRG0# pin)
Other functions	• Buffer operation • 2-channel simultaneous sampling	• Adjustable number of sampling states • Self-diagnostic function	• Adjustable number of sampling states • A/D-converted value addition mode
Operations linked to A/D conversion-end interrupt	• CPU interrupt generation • DMAC or DTC activation	• CPU interrupt generation • DMAC or DTC activation	• CPU interrupt generation • DMAC or DTC activation
Low power consumption function	None	Support for module stop state setting	Support for module stop state setting
Conversion targets	AN pin	AN pin Self-diagnostic (fault detection)	AN pin Internal reference voltage Temperature sensor

2.11.2 Operating Modes

The operation of the SH7044's high-speed A/D converter is determined by the following mode settings in combination.

- Channel designation mode
  Select mode: A single channel is specified.
  Group mode: Multiple channels are specified.
- Converter operation mode
  Single mode: A/D conversion is activated once.
  Scan mode: A/D conversion is activated repeatedly.

Table 2.69 SH7044 High-Speed A/D Converter Operating Modes

Operating Mode	Single Mode	Scan Mode
Select mode	1 conversion of 1 channel	Repeated conversions of 1 channel
Group mode	1 conversion of multiple channels	Repeated conversions of multiple channels

The corresponding operating modes, when switching from the SH7044's high-speed A/D converter, are listed below.

Table 2.70 A/D Converter Operating Mode Correspondences

No.	SH7044 (high-speed A/D converter)	RX631 (ADb)	RX631 (S12ADa)
1	Select single mode	Single channel mode	Single scan mode (only 1 channel specified)
2	Select scan mode	Continuous scan mode in scan mode	Continuous scan mode (only 1 channel specified)
3	Group single mode	Single scan mode in scan mode	Single scan mode (multiple channels specified)
4	Group scan mode	Continuous scan mode in scan mode	Continuous scan mode (multiple channels specified)

For a description of the operating modes on the RX631 (ADb and S12ADa), see table 2.63.

2.11.3 Module Stop

The initial state of the peripheral modules of the RX631 is stopped, due to the low power consumption function. The initial state of the A/D converter modules (ADb and S12ADa) is also stopped. Do not fail to cancel the module stop state when making settings to these modules. Before accessing the module stop control register to cancel the module stop state, first cancel register write protection.

2.11.4 Other Differences

The 10-bit A/D converter on the RX631 has no functions equivalent to simultaneous sampling, low-power conversion mode, and buffer operation, all of which are supported by the high-speed A/D converter on the SH7044.

2.11.5 A/D Converter Setting Example

An A/D converter setting example is shown below.

The following setting example applies to the case where the high-speed A/D converter (group scan mode) of the SH7044 is being replaced by the RX631 (continuous scan mode). In addition, table 2.73 lists differences in the settings corresponding to the other operating modes of the SH7044.

< Continuous Scan Mode Specifications >

1. The 10-bit A/D converter on the RSK+RX63N board is used.
2. The A/D conversion start timing is based on the software trigger.
3. Three channels of analog input are used, AN0, AN1, and AN2, and the operating mode is continuous scan mode.

When conversion finishes the conversion result is stored in the RAM by the handler of the ADI0 interrupt.

Table 2.71 10-Bit A/D Converter Setting Specifications

Item	Description	Remarks
Channel used	AN0, AN1, and AN2	
Interrupt handling	A/D conversion-end interrupt (ADI0 interrupt)	
Operating Mode	Continuous scan mode	SH7044 scan mode
Clock selection	PCLK/2	PCLK = 48 MHz
Conversion start trigger and cycle	Software trigger (Conversion repeats after start.)	
Extended analog input	Not used	
Data alignment	Flush-right	AN0: ADDRA AN1: ADDRB AN2: ADDRC
Pins used	PE02/AN0	Analog input 0
	PE03/AN1	Analog input 1
	PE04/AN2	Analog input 2

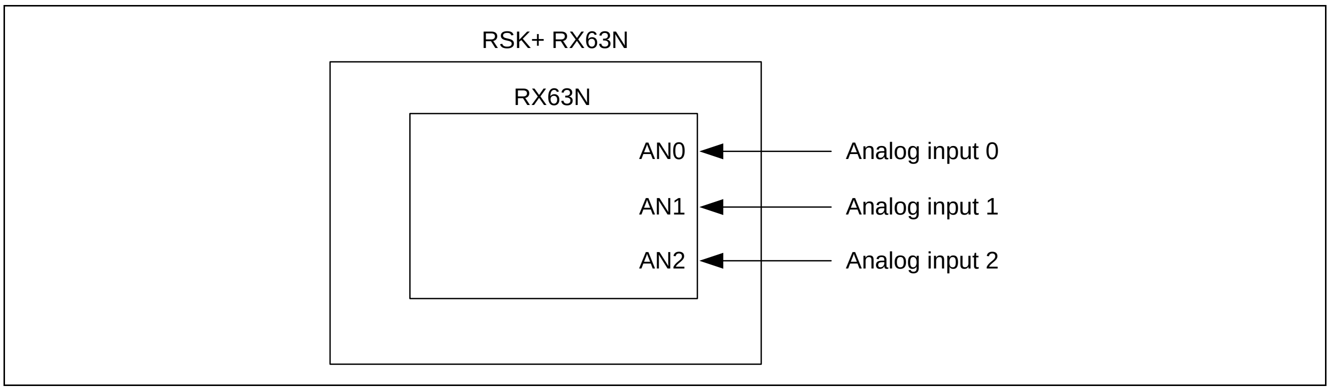


Figure 2.27 10-Bit A/D Converter Setting Connection Specifications

An initial setting example for switching from the high-speed A/D converter on the SH7044 to the 10-bit A/D converter on the RX631 is shown below.

Table 2.72 10-Bit A/D Converter Initial Setting Example

Procedure		SH7044 Setting Example Pϕ (Peripheral Clock): 20 MHz	RX631 Setting Example PCLK (Peripheral Clock): 48 MHz

1	Cancel module stop state.	(No module stop function)	SYSTEM.PRCR = 0xA502
			SYSTEM.MSTPCRA.MSTPA23 = 0
			SYSTEM.PRCR = 0xA500

2	Disable interrupts.	ADCSR.ADIE = 0
		(interrupts disabled)
			IER0C.IEN2 = 0 (vector 98, ADI0 disabled)
			ADCSR.ADIE = 0 (interrupts disabled)

3	Disable A/D converter.	ADCSR.ADST = 0 (A/D0 disabled)	ADCSR.ADST = 0 (A/D disabled)

4	Make I/O port settings (pin I/O and pin function settings).	I/O settings are not needed because AN0 is assigned as an input-only port (PFDR).	Pin AN0 set in MPC
			PORTE.PDR.B2 = 0 (input)
			PORTE.PDR.B3 = 0 (input)
			PORTE.PDR.B4 = 0 (input)
			PORTE.PMR.B2 = 0 (GPIO)
			PORTE.PMR.B3 = 0 (GPIO)
			PORTE.PMR.B4 = 0 (GPIO)
			MPC.PWPR.B0WI = 0
			MPC.PWPR.PFSWE = 1
			(PFS write enables)
			MPC.PE2PFS = 0x80 (analog function)
			MPC.PE3PFS = 0x80 (analog function)
			MPC.PE4PFS = 0x80 (analog function)
			MPC.PWPR.PFSWE = 0
			(PFS write disabled)
			MPC.PWPR.B0WI = 1

5	Make ADCSR settings.	ADCSR.CKS = 0 (conversion duration: 40 states) ADCSR.GRP = 1 (group mode) ADCSR.CH2 to ADCSR.CH0 = 2 (AN0, AN1, and AN2)	ADCSR.CH = 2 (AN0, AN1, and AN2)

6	Make ADCR settings.	ADCR.PWR = 1
		(high-speed start mode)
		ADCR.TRGS1, ADCR.TRGS0 = 0
		(software trigger)
		ADCR.SCAN = 1 (scan mode)
		ADCR.DSMP = 0 (normal sampling)
		ADCR.BUFE1, ADCR.BUFE0 = 0
		(normal operation)
			ADCR.MODE = 2 (continuous scan mode)
			ADCR.CKS = 2 (PCLK/2)
			ADCR.TRGS = 000b (software trigger)

7	ADDR format	Only flush-right supported, so no setting needed.	ADCR2.DPSEL = 0 (data flush-right)
8	Make settings to interrupt priority register.	IPRG.WORD = 0x5000 (A/D0 and A/D1: level 5)	IPR098 = 5 (ADI0: level 5)

9	Enable interrupts.	ADCSR.ADIE = 1
		(interrupts enabled)
			IER0C.IEN2 = 1 (vector 98, ADI0 enabled)
			ADCSR.ADIE = 1 (interrupts enabled)

10	Start A/D conversion.	ADCSR.ADST = 1 (A/D start)	ADCSR.ADST = 1 (A/D start)
11	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: Make changes to the values in the shaded portions to make I/O port settings or select/change the operating mode. Table 2.73 lists the settings on the RX631 that correspond to the various operating modes on the SH7044.

Change the setting values in the shaded portions of table 2.72 to select among the modes listed below:

Table 2.73 Corresponding A/D Converter Operating Mode Settings (SH7044 to RX631)

No.	SH7044 High-Speed A/D Converter	RX631 (ADb)
1	Select single mode	Single channel mode

	ADCSR.GRP = 0 (select mode)
	ADCSR.CH2 to ADCSR.CH0 = 0 (AN0)
	ADCR.SCAN = 0 (single mode)
		PORTE.PDR.B2 = 0 (input)
		PORTE.PMR.B2 = 0 (GPIO)
		MPC.PE2PFS = 0x80 (analog function)
		
		ADCSR.CH = 0 (AN0)
		ADCR.MODE = 0 (single channel mode)

2	Select scan mode	Continuous scan mode (single channel)

	ADCSR.GRP = 0 (select mode)
	ADCSR.CH2 to ADCSR.CH0 = 0 (AN0)
	ADCR.SCAN = 1 (scan mode)
		PORTE.PDR.B2 = 0 (input)
		PORTE.PMR.B2 = 0 (GPIO)
		MPC.PE2PFS = 0x80 (analog function)
		
		ADCSR.CH = 0 (AN0)
		ADCR.MODE = 2 (continuous scan mode)

3	Group single mode	Scan mode (multiple channels)

	ADCSR.GRP = 1 (group mode)
	ADCSR.CH2 to ADCSR.CH0 = 2 (AN0, AN1, and AN2)
	ADCR.SCAN = 0 (single mode)
		PORTE.PDR.B2 = 0 (input)
		PORTE.PDR.B3 = 0 (input)
		PORTE.PDR.B4 = 0 (input)
		PORTE.PMR.B2 = 0 (GPIO)
		PORTE.PMR.B3 = 0 (GPIO)
		PORTE.PMR.B4 = 0 (GPIO)
		MPC.PE2PFS = 0x80 (analog function)
		MPC.PE3PFS = 0x80 (analog function)
		MPC.PE4PFS = 0x80 (analog function)
		
		ADCSR.CH = 2 (AN0, AN1, and AN2)
		ADCR.MODE = 1 (scan mode)

4	Group scan mode	Continuous scan mode (multiple channels)

	ADCSR.GRP = 1 (group mode)
	ADCSR.CH2 to ADCSR.CH0 = 2 (AN0, AN1, and AN2)
	ADCR.SCAN = 1 (scan mode)
		PORTE.PDR.B2 = 0 (input)
		PORTE.PDR.B3 = 0 (input)
		PORTE.PDR.B4 = 0 (input)
		PORTE.PMR.B2 = 0 (GPIO)
		PORTE.PMR.B3 = 0 (GPIO)
		PORTE.PMR.B4 = 0 (GPIO)
		MPC.PE2PFS = 0x80 (analog function)
		MPC.PE3PFS = 0x80 (analog function)
		MPC.PE4PFS = 0x80 (analog function)
		
		ADCSR.CH = 2 (AN0, AN1, and AN2)
		ADCR.MODE = 2 (continuous scan mode)

Note that the changes for each operating mode and the conversion start triggers in the included sample code are as follows:

Table 2.74 Sample Code Description

Operating Mode	Conversion Channel(s)	Conversion Start Trigger	Remarks
Single channel mode	AN0	MTU4 compare match	
Continuous scan mode (single channel)	AN0	Software trigger	
Scan mode (multiple channels)	AN0, AN1, and AN2	MTU4 compare match	
Continuous scan mode (multiple channels)	AN0, AN1, and AN2	Software trigger	

2.12 Compare Match Timer (CMT)

2.12.1 Comparison of Specifications

Table 2.75 Comparison of SH7044 and RX631 CMT Specifications

Item	SH7044	RX631
Clock	Each channel selectable among 4 internal clocks (ϕ/8, ϕ/32, ϕ/128, and ϕ/512)	Each channel selectable among 4 internal clocks (PCLK/8, PCLK/32, PCLK/128, and PCLK/512)
Number of units (channels)	1 unit (total 2 channels)	2 units (total 4 channels)
Interrupt sources	Support for separate compare match interrupt requests for each (CMI0 and CMI1)	Support for separate compare match interrupt requests for each (CMI0, CMI1, CMI2, and CMI3)

2.12.2 CMT Replacement

The CMT of the SH7044 and the CMT of the RX631 are software compatible. However, the compare match timer control and status registers (CMCSR0 and CMCSR1) on the RX631 do not contain interrupt flags, so it is necessary to use the interrupt controller’s interrupt flags instead. In addition, it is not necessary to clear the flags in the compare match interrupt handler. (The interrupt controller automatically clears the associated flag when an interrupt is accepted.) A comparison of the compare match timer registers of the SH7044 and RX631 is shown below.

Table 2.76 List of Compare Match Timer Registers

Register Name	SH7044	RX631	Changed
Unit 0 (channels 0 and 1) corresponds to channels 0 and 1 on the SH7044.			
Compare match timer start register	CMSTR	CMSTR0	◎
Compare match timer control/status registers	CMCSR0, CMCSR1	CMT0.CMCR, CMT1.CMCR	◎*
Compare match timer counters	CMCNT0, CMCNT1	CMT0.CMCNT, CMT1, CMCNT	◎
Compare match timer constant registers	CMCOR0, CMCOR1	CMT0.CMCOR, CMT1, CMCOR	◎
Unit 1 (channels 2 and 3) below has no corresponding channels on the SH7044.			
—		CMSTR1	○
		CMT2.CMCR, CMT3.CMCR	○*
		CMT2.CMCNT, CMT3, CMCNT	○
		CMT2.CMCOR, CMT3, CMCOR	○

◎: Registers with identical bit assignments on the SH7044 and RX631
○: Unit 1 registers. The bit assignments are the same as for unit 0.
Note: * These registers so not contain interrupt flags. Use the IR bits of the interrupt controller instead.

2.12.3 Module Stop

The initial state of the peripheral modules of the RX631 is stopped, due to the low power consumption function. The initial state of the CMT is also stopped. Do not fail to cancel the module stop state when making settings to the module. Before accessing the module stop control register to cancel the module stop state, first cancel register write protection.

2.12.4 Compare Match Timer Setting Example

A compare match timer setting example comparing the SH7044 and RX631 is shown below.

< Specifications >

1. CMT unit 0, channel 0 on the RSK+RX63N board is used.
2. The compare match interrupt (CMI0) is used to turn LED1 one and off in 0.5-second cycles.

Table 2.77 Compare Match Timer Setting Specifications

Item	Description	Remarks
Count clock	PCLK/512	PCLK = 48 MHz
Counter value (CMCOR)	B71Bh	
Other	LED1 P05	GPIO

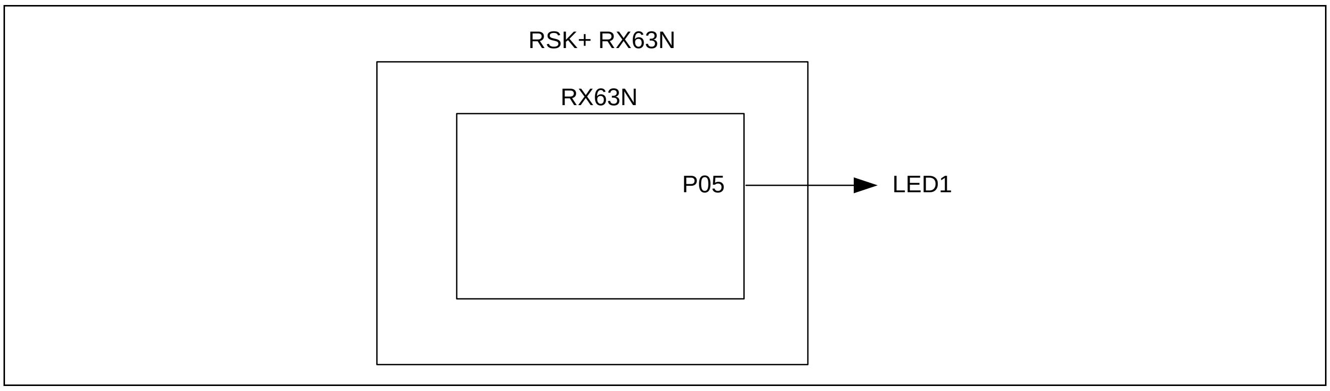
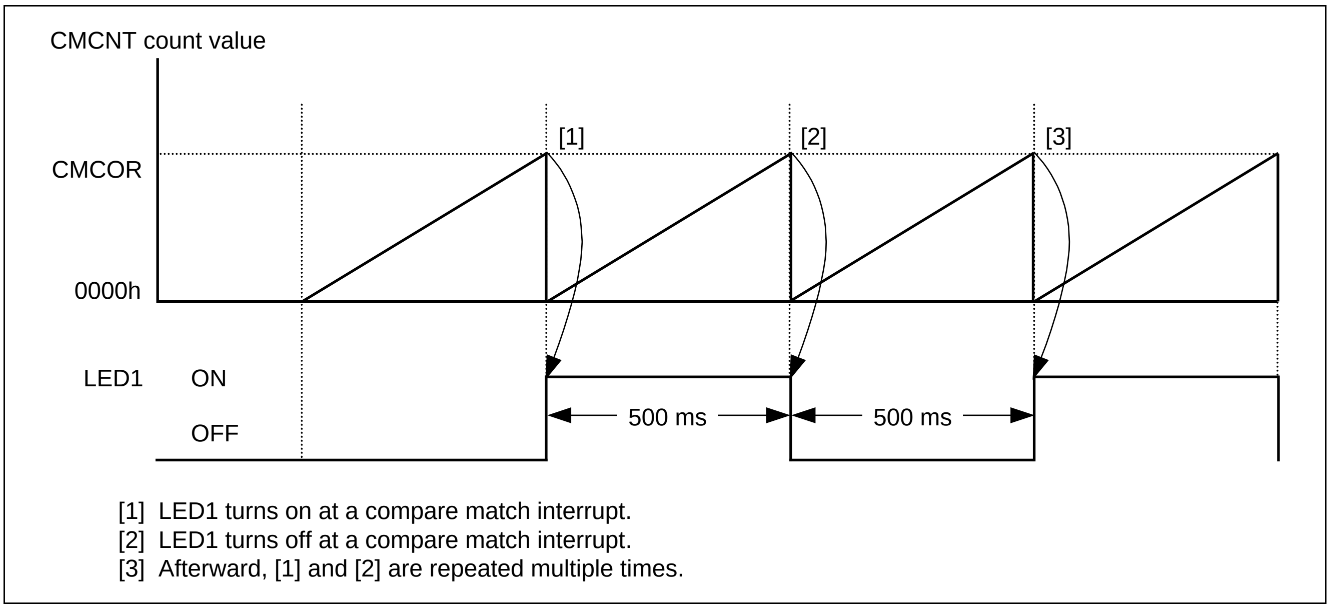


Figure 2.28 Compare Match Timer Connection Specifications



[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 2.29 Compare Match Timer Operation Example

Table 2.78 Compare Match Timer Initial Setting Example

Procedure		SH7044 Setting Example Pϕ (Peripheral Clock): 20 MHz	RX631 Setting Example PCLK (Peripheral Clock): 48 MHz

1	Cancel module stop state.	(No module stop function)	SYSTEM.PRCR = 0xA502
			SYSTEM.MSTPCRA.MSTPA15 = 0
			SYSTEM.PRCR = 0xA500

2	Disable interrupts.	CMCSR.CMIE = 1
		(compare match interrupt disabled)
			IER03.IEN4 = 0
			(vector 28, CMI0 disabled)
			CMT0.CMCR.CMIE = 0

3	Disable timer.	CMSTR.STR0 = 0	
4	Select counter clock.	CMCSR.CKS0 to CMCSR.CKS1 = 11b (ϕ/512)	CMT0.CMCR.CKS0 to CMT0.CMCR.CKS1 = 11b (PCLK/512)
5	Clear timer counter.	CMCNT0 = 0000h (counter cleared)	CMT0.CMCNT = 0000h (counter cleared)
6	Set compare match cycle.	CMCOR0 = 4C4Bh	CMT0.CMCOR = B71Bh

7	Enable interrupts.	CMCSR.CMIE = 1
		(compare match enables)
		INTC.IPRG.WORD = 0x0050
		(interrupt priority: 5)
			CMT0.CMCR.CMIE = 1
			(compare match enables)
			IPR004 = 5 (CMI0 interrupt priority: 5)
			IR028 = 0 (CMI0 interrupt flag cleared)
			IER03.IEN4 = 1
			(vector 28, CMI0 enabled)

8	Enable timer operation.	CMSTR.STR0 = 1 (start timer)	
9	Interrupt handler (Clear flag.)	CMCSR.CMF = 0 (after reading CMCSR, CMF = 0)	The interrupt flag is cleared automatically.

2.13 Flash Memory

2.13.1 Comparison of Specifications

Table 2.79 Comparison of Flash Memory Specifications on SH7044 and RX631

Item	SH7044	RX631
Size	• 256 KB	• ROM area User area: Max. 2 MB User boot area: 16 KB
Block size × block count	• 1 KB × 4 (4 KB) • 28 KB × 1 (28 KB) • 32 KB × 7 (224 KB)	Each area: 512 KB • Area 0 4 KB × 8 (32 KB) 16 KB × 30 (480 KB) • Area 1 32 K × 16 (512 KB) • Area 2 64 K × 8 (512 KB) • Area 3 64 K × 8 (512 KB)
Operating modes	• Program mode • Erase mode • Program verify mode • Erase verify mode	On-chip dedicated programming sequencer (FCU) P/E execution by FCU commands • FCU mode P/E normal mode Status read mode Lock bit read mode
Write and erase units	• Write: 32-byte units • Erase: Block units	• Write — User area: 128-byte units — User boot area: 128-byte units • Erase — User area: Block units — User boot area: 16 KB units
Write count	100 times	1,000 times
Programming modes	• On-board programming — Boot mode — User programming mode • Writer mode	• On-board programming — Boot mode — USB boot mode — User boot mode — Single-chip mode • Off-board programming Ability to program user boot area using a flash writer
Other	• Automatic bit rate adjustment • RAM-based flash memory emulation function • Protect mode	• Automatic bit rate adjustment • Suspend/resume function • Protect function

P/E: Program/erase

The RX Simple Flash API can be used to program the on-chip flash memory on the RX631.

The RX Simple Flash API is provided to customers to allow them to easily program and erase the on-chip flash memory on the RX631. See the following application note for instructions on using the API and embedding it in applications:

RX600 & RX200 Series Simple Flash API for RX (R01AN0544EU)

2.14 Low Power Consumption Function

2.14.1 Comparison of Mode Specifications

The low-power modes on the SH7044 are sleep mode and standby mode. The states of the clock, CPU, and on-chip modules in each mode are listed below:

Table 2.80 SH7044 Low-Power Modes

Item	Clock	CPU	On-Chip Modules
Sleep mode	Operating	Stopped	Operating
Standby mode	Stopped	Stopped	Stopped

The low-power modes on the RX631 are sleep mode, all-module clock stop mode, software standby mode, and deep software standby mode. Table 2.81 lists the states of the on-chip modules in each mode.

Table 2.81 RX631 Low-Power Modes

Function/State	Sleep Mode	All-Module Clock Stop Mode	Software Standby Mode	Deep Software Standby Mode
Main clock oscillator	Operation possible	Operation possible	Operation possible	Operation possible
Sub-clock oscillator	Operation possible	Operation possible	Operation possible	Operation possible
High-speed on-chip oscillator	Operation possible	Operation possible	Stopped	Stopped
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)
RAM1 (0001 0000h to 0003 FFFFh)	Operation possible (settings retained)	Stopped (settings retained)	Stopped (settings retained)	Stopped (settings undetermined)
RAM0 (0000 0000h to 0000 FFFFh)	Operation possible (settings retained)	Stopped (settings retained)	Stopped (settings retained)	Stopped (settings retained/ undetermined)*
Flash memory	Operating	Stopped (settings retained)	Stopped (settings retained)	Stopped (settings retained)
USB 2.0 Host/Function module (USB)	Operation possible	Stopped	Stopped	Stopped (settings retained/ undetermined)
Watchdog timer (WDT)	Stopped (settings retained)	Stopped (settings retained)	Stopped (settings retained)	Stopped (settings undetermined)
Independent watchdog timer (IWDT)	Operation possible	Operation possible	Operation possible	Stopped (settings undetermined)
Realtime clock (RTC)	Operation possible	Operation possible	Operation possible	Operation possible
Port output enable 2 (POE2)	Operation possible	Operation possible	Stopped (settings retained)	Stopped (settings undetermined)
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	Operating	Operating	Operating	Operating
Peripheral modules	Operation possible	Stopped (settings retained)	Stopped (settings retained)	Stopped (settings undetermined)
I/O ports	Operating	Settings retained	Settings retained	Settings retained

"Operation possible" indicates a state in which a control register setting can be used to start and stop the module.
"Stopped (settings retained)" indicates a state in which the values of the internal registers are retained and the internal state is operation suspended.
"Stopped (settings undetermined)" indicates a state in which the values of the internal registers are undetermined and the internal state is power-off.
Note: * Either "settings retained" or "settings undetermined" may be selected by means of a register setting.

2.14.2 Mode Transitions

Figure 2.30 diagrams the transitions between the operating modes of the RX631.

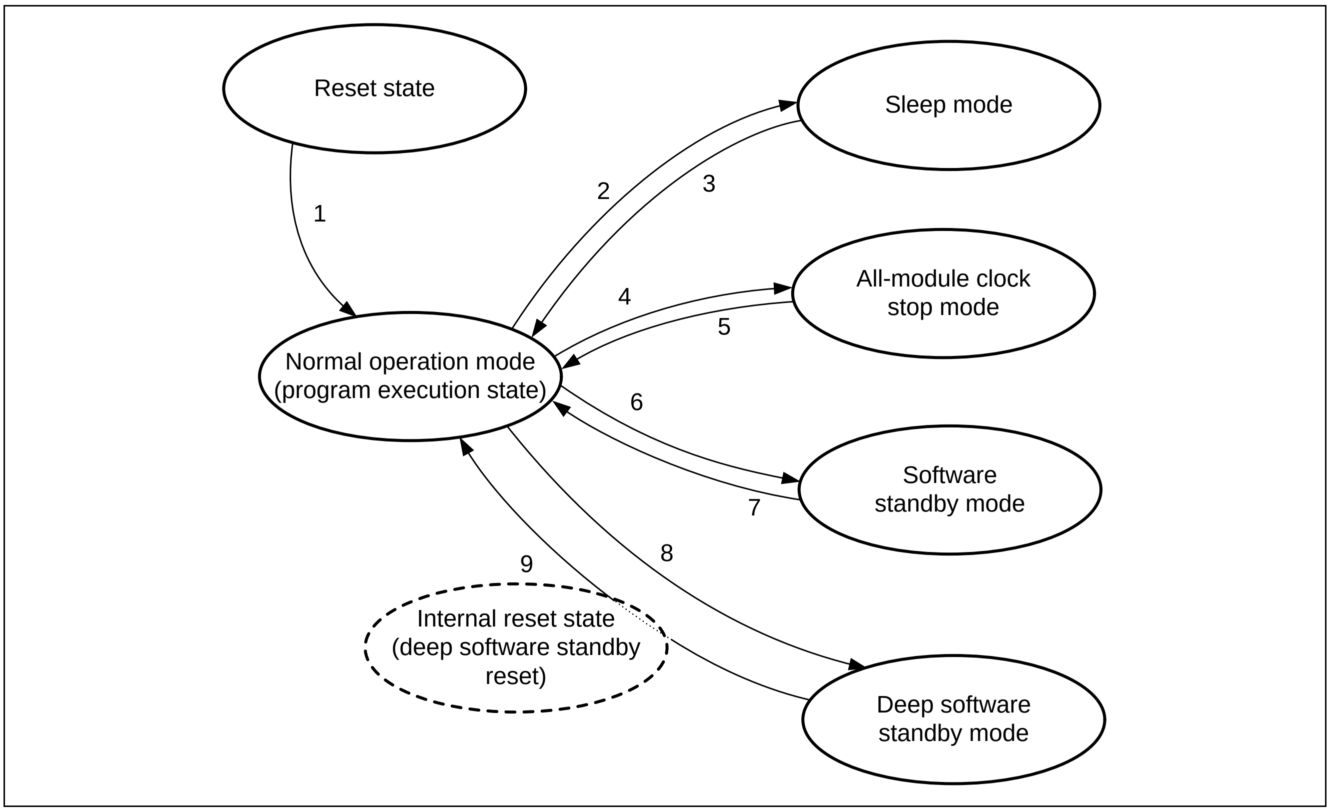

Reset state
Sleep mode
All-module clock stop mode
Normal operation mode (program execution state)
Software standby mode
Internal reset state (deep software standby reset)
Deep software standby mode
1
2
3
4
5
6
7
8
9

Figure 2.30 RX631 Mode Transitions

The events and transition conditions shown in figure 2.30 are listed below:

Table 2.82 List of RX631 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
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)*
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)*
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)* 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: * Each interrupt has detailed conditions. For descriptions, see the User’s Manual: Hardware.

2.14.3 Mode Transition Setting Example

A mode transition setting example using the RX631 is shown below.

< Specifications >

1. The RSK+RX63N board is used.
2. After a reset, settings are made to enable input from SW2 (IRQ8-DS), to wait for SW2 to be pressed, and to implement all of the mode transitions listed below by pressing SW2.
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.

Note

On the RX63N RSK, SW2 is not connected to the IRQ8-DS pin. Therefore, the following changes must be made to the RSK when doing debugging using the sample source code.

(Details of changes)

- Connect the JA1 23-pin connector that is connected to SW2 to the JA1 9-pin connector that is connected to the IRQ8-DS pin.
- Mount the unmounted R83 (0 Ω resistor). (Make a direct connection to the R83).
- Remove the R84 (resistor) mounted on the board.

Table 2.83 lists the mode transitions and the operation of the modules.

Table 2.83 RX631 Mode Transition Setting and Operation Specifications

No.	SW2 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.

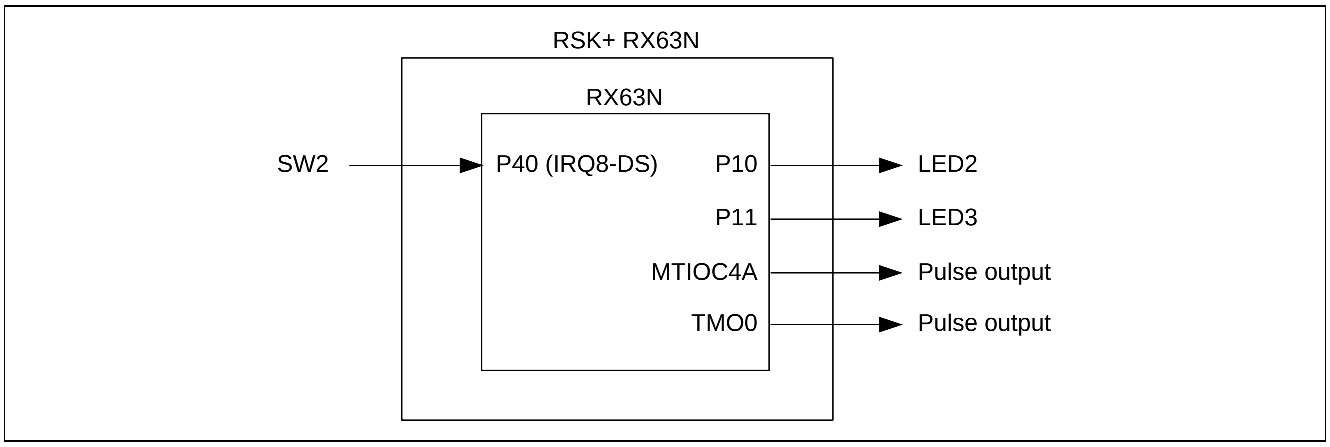


Figure 2.31 Mode Transition Setting Connection Specifications

Table 2.84 Setting Specifications

Item		**Description**	**Remarks**
CPU			
	Processor mode	Supervisor mode	
TMR0, TMR1			
	Count clock	PCLK/1	PCLK = 48 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	5DE6h	TMR0 + TMR1
	Output selection	Inverted output	
	Pins used	P22/TMO0	Pulse output
SW2 (IRQ8-DS)			
	SW2 (IRQ8-DS)	Used as mode transition trigger switch P40/IRQ8-DS	
	Interrupt priority	Level 15	
	Digital noise filter	Used*1	
	Return from deep software standby	The SW2 signal is used as the deep software standby cancel signal, so connect it to P40.*2	
LED			
	LED2	Flashing during SW2 press wait duration (normal state).	P10
	LED3	Turns on at return from deep software standby.	P11
Pins used by MTU4			
	Compare match A output pin	P24/MTIOC4A	Pulse output

Notes: 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.
2. SW2 is not connected to P40 (IRQ8-DS) by default.

Figure 2.32 is a flowchart of mode transition processing.

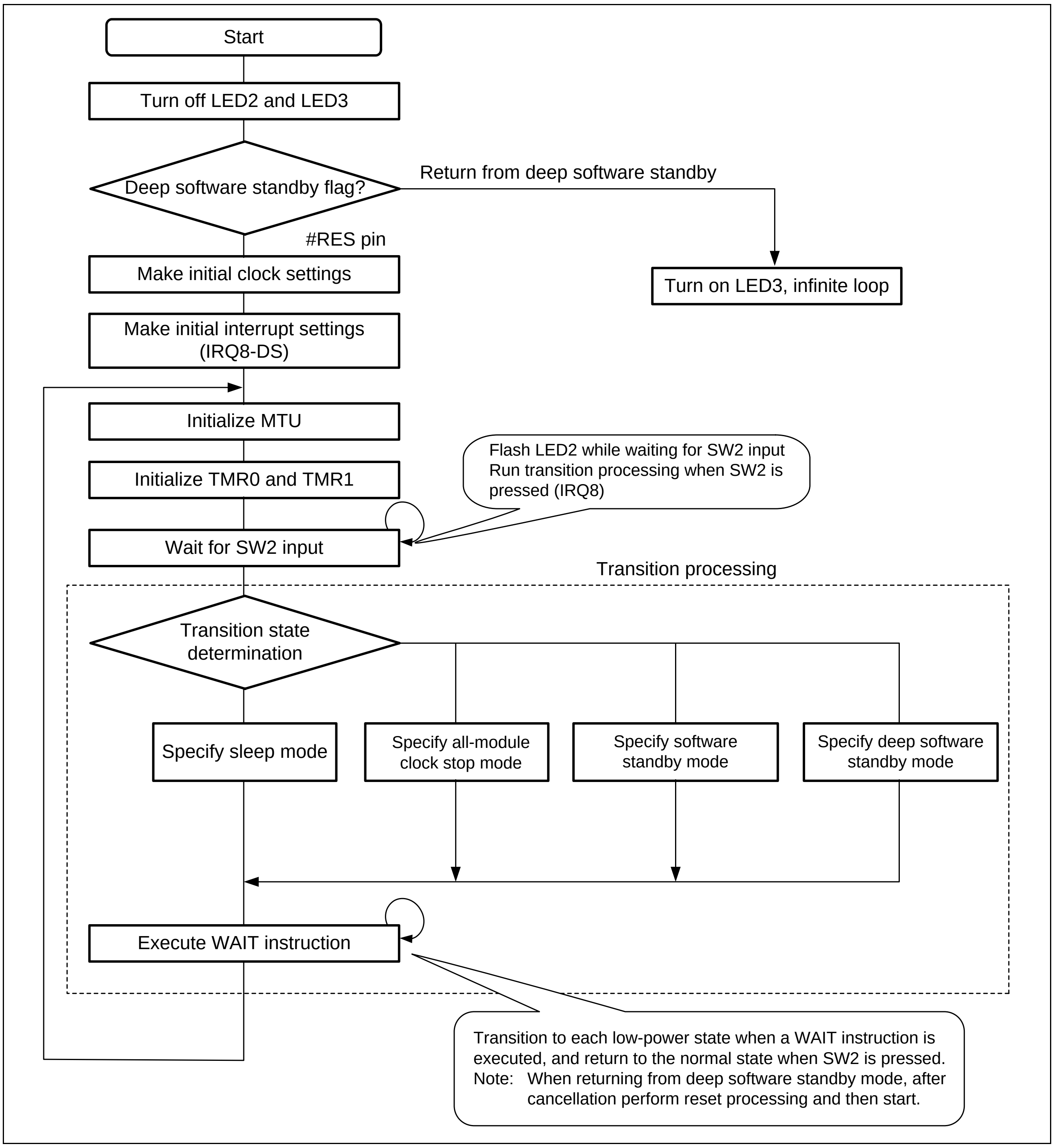

Start
Turn off LED2 and LED3
Deep software standby flag?
Return from deep software standby
#RES pin
Turn on LED3, infinite loop
Make initial clock settings
Make initial interrupt settings (IRQ8-DS)
Initialize MTU
Initialize TMR0 and TMR1
Flash LED2 while waiting for SW2 input Run transition processing when SW2 is pressed (IRQ8)
Wait for SW2 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 SW2 is pressed.
Note: When returning from deep software standby mode, after cancellation perform reset processing and then start.

Figure 2.32 Mode Transition Processing Flowchart

Settings associated with mode transitions are listed below.

Refer to section 2.7.6 for details of MTU4 settings.

Table 2.85 LED2 and LED3 Settings (Initially Off)

Procedure		Setting Example

1	Make GPIO settings (LED2 and LED3 off).	PORT1.PODR.B0 = 1 (LED2 off)
		PORT1.PDR.B0 = 1 (output)
		PORT1.PMR.B0 = 0 (GPIO)
		PORT1.PODR.B1 = 1 (LED3 off)
		PORT1.PDR.B1 = 1 (output)
		PORT1.PMR.B1 = 0 (GPIO)

Table 2.86 Interrupt Initial Setting Example (IRQ8-DS Settings)

Procedure		Setting Example

1	Make interrupt settings and pin settings.	PORT4.PDR.B0 = 0 (P00 input)
		PORT4.PMR.B0 = 0 (P00GPIO)
		MPC.PWPR.B0WI = 0
		MPC.PWPR.PFSWE = 1 (PFS write enables)
		MPC.P40PFS.ISEL = 1 (interrupt function setting IRQ8-DS)
		MPC.PWPR.PFSWE = 0 (PFS write disabled)
		MPC.PWPR.B0WI = 1

2	Enable interrupts, etc.	IRQCR8.IRQMD = 1 (IRQ detection: Falling edge)
		IRQFLTE1.FLTEN8 = 1 (IRQ8 digital noise filter enabled)
		IRQFLTC1.FCLKSEL8 = 3; (digital noise filter sampling: PCLK/32)
		IR072 = 0 (interrupt flag cleared)
		IPR072 = 15 (interrupt level: 15)
		IER09.IEN0 = 1 (IRQ8 enabled)

Table 2.87 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 = 0xA502;
		SYSTEM.MSTPCRA. MSTPA5 = 0
		SYSTEM.PRCR.WORD = 0xA500;

2	Clear and stop TMR timers.	TMR0.TCNT = 0x00 (TMR0 TCNT cleared)
		TMR1.TCNT = 0x00 (TMR1 TCNT cleared)
		TMR0.TCCR = 0x00 (TMR0 clock stopped)
		TMR1.TCCR = 0x00 (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 enables)
		MPC.P22PFS = 05h (pin P22 set to TMO0)
		MPC.PWPR.PFSWE = 0 (PFS write disabled)
		MPC.PWPR.B0WI = 1
		PORT2.PMR.B2 = 1 (pin function setting)

4	Make TOCRA settings.	TMR0.TOCRA = 5Dh
		TMR1.TOCRA = E6h

5	Make TCR settings.	TMR0.TCR.CCLR = 1 (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 = 3 (pin TMO0 inverted output) TMR1.TCSR: Default setting

7	Make TCCR settings. (TCNT start)	TMR0.TCCR.CSS = 3 (TMR1.TCNT counts at overflow signal)
		TMR1.TCCR.CKS = 000b (counting at PCLK/1 ⇒ CKS and CSS combined)
		TMR1.TCCR.CSS = 01b

Table 2.88 Sleep Mode Setting Example

Procedure		Setting Example
1	Cancel protect.	SYSTEM.PRCR = A503h (cancel protect)
2	Set standby control register.	SYSTEM.SBYCR.SSBY = 0 (no software standby)
3	Enable protect.	SYSTEM.PRCR = A500h (enable protect)

Table 2.89 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 (no software standby)
		SYSTEM.SBYCR.OPE = 0 (high-impedance bus output)

3	Set module stop registers A, B, and C.	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 = FFFF0000h (transition to module stop state, excluding RAM)
4	Enable protect.	SYSTEM.PRCR = A500h (enable protect)

Table 2.90 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 (enable protect)

Table 2.91 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.SBYCR.OPE = 0 (high-impedance bus output)

3	Make deep software standby mode setting.	SYSTEM.DPSBYCR.DPSBY = 1 (deep software standby enabled) SYSTEM.DPSIER1.DIRQ8E = 1 (deep software standby enabled by IRQ8-DS)
4	Clear deep software standby interrupt flag.	SYSTEM.DPSIFR1.DIRQ8F = 0 (cancel request flag cleared by IRQ8-DS pin)
5	Enable protect.	SYSTEM.PRCR = A500h (enable protect)

3. Sample Code

3.1 Operating Environment

The sample code associated with this application note has been confirmed to run in the following environment.

Table 3.1 Operating Environment

Item	Description
Microcontroller used	R5F563NB (RX63N Group)
Operating frequency	• Main clock: 12 MHz • Sub clock: 32.768 kHz • PLL: 192 MHz (main clock divided by 1 and multiplied by 16) • HOCO: Stopped • System clock (ICLK): 96 MHz (PLL divided by 2) • Peripheral module clock A (PCLKA): 96 MHz (PLL divided by 2) • Peripheral module clock B (PCLKB): 48 MHz (PLL divided by 4)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance Embedded Workshop (Version 4.09.01.007)
C compiler	Renesas Electronics Corporation C/C++ Compiler Package for RX Family (V.1.02 Release 01)
CPU series (type)	RX600 (RX63N)
Optimization	None
iodefine.h version	1.6A
Endian	Big endian
Operating Mode	Single-chip mode (on-chip ROM enabled extended mode only when using SDRAM)
Processor mode	Supervisor mode
Sample code version	1.00
Board used	Renesas Starter Kit+ for RX63N (Product type: R0K50563NC010BR)

3.2 Sample Code Configuration

The configuration of the sample code is shown below.

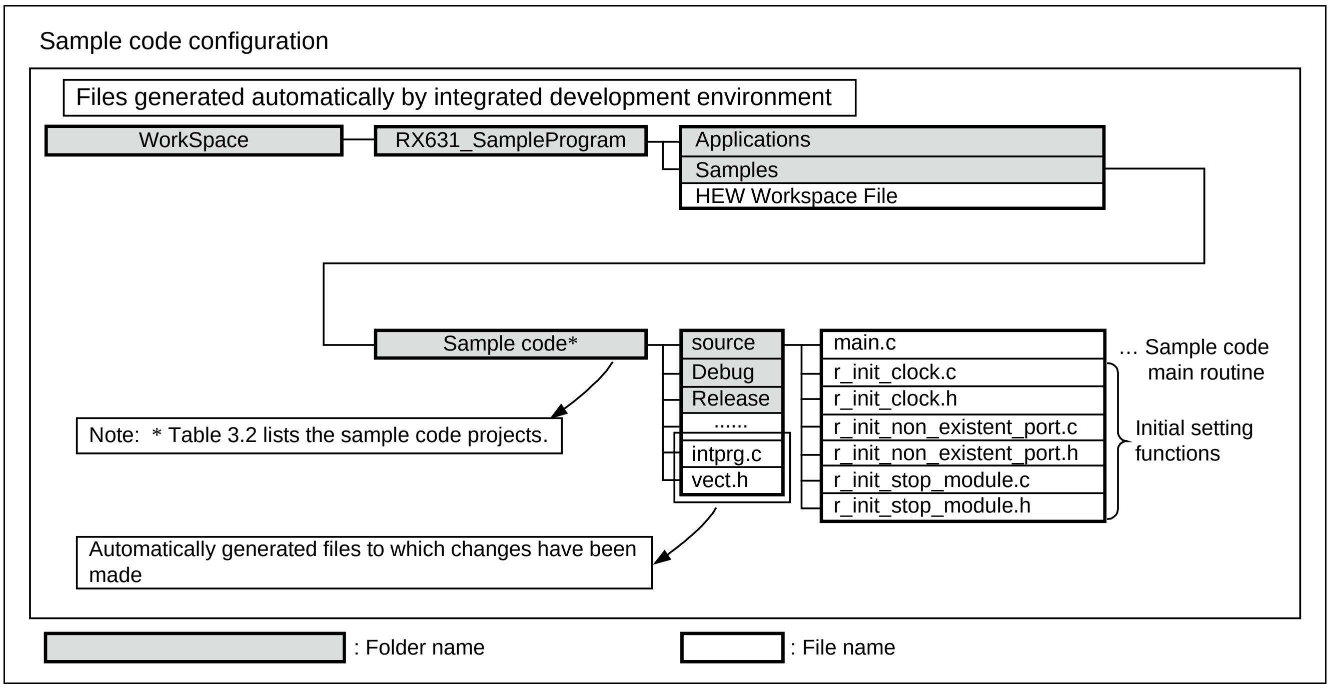

Sample code configuration
Files generated automatically by integrated development environment
WorkSpace
RX631_SampleProgram
Applications
Samples
HEW Workspace File
Sample code*
source
Debug
Release
......
intprg.c
vect.h
main.c
r_init_clock.c
r_init_clock.h
r_init_non_existent_port.c
r_init_non_existent_port.h
r_init_stop_module.c
r_init_stop_module.h
… Sample code main routine
Initial setting functions
Note: * Table 3.2 lists the sample code projects.
Automatically generated files to which changes have been made
: Folder name
: File name

Figure 3.1 Sample Code Configuration

Initial Settings

The initial setting function of this application note uses the sample code from Group, RX631 Group: Initial Setting Example, Rev. 1.00. This revision was current when this application note was produced.

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 intprg.c and vect.h duplicate settings and code in main.c, so they have been modified as follows:

intprg.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
DTC_normal_transfer_mode	2.5.8
DMA_normal_transfer_mode	2.6.11
MTU_compare_match	2.7.6
MTU_input_capture	2.7.7
SCI_asynchronous_interrupt	2.9.4
SCI_asynchronous_polling	
SCI_sync_master_transmit_int	2.9.5
SCI_sync_master_transmit_pol	
SCI_sync_slave_receive_int	2.9.6
SCI_sync_slave_receive_pol	
AD_single_channel_mode	2.10.5
AD_continuous_scan_single_ch	2.11.5
AD_continuous_scan_multi_ch	2.10.6 2.11.5
AD_single_scan_mode_multi_ch	2.11.5
CMT_compare_match	2.12.4
Low_power_consumption_mode	2.14.3

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
SH7040, SH7041, SH7042, SH7043, SH7044, SH7045 Group Hardware Manual (REJ09B0044-0600O)
SH-1/SH-2/SH-DSP Software Manual (REJ09B0171-0500O)
RX63N Group, RX631 Group User’s Manual: Hardware (R01UH0041EJ)
RX Family User’s Manual: Software (R01US0032EJ)
[HEW] Renesas Starter Kit+ for RX63N User’s Manual (R20UT0438EG)
Renesas Starter Kit+ for RX63N CPU Board Schematics (R20UT0437EG)
RX63N Group, RX631 Group Initial Setting (R01AN1245EJ)
RX63N Group, RX631 Group Asynchronous Communication Using the SCI (R01AN1449EJ)
RX63N Group, RX631 Group Synchronous SCIc Communication Using the DMACA (R01AN1064EJ)
RX63N Group, RX631 Group Pulse Width Measurement Using MTU2a (R01AN1237EJ)
RX63N Group, RX631 Group I2S Communication Using RSPI, DTCa, and MTU2a (R01AN1339EJ)
RX63N Group, RX631 Group Exiting Software Standby Mode Using the RTCa (R01AN1067EJ)
RX63N Group, RX631 Group Read/Write Operations in 16-Bit SDRAM Using the SDRAMC (R01AN1705EJ)
RX600 & RX200 Series Simple Flash API for RX (R01AN0544EU)

Website and Support

Renesas Electronics Website

http://www.renesas.com/

Inquiries

http://www.renesas.com/contact/

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Sep 30, 2014	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to a product with a different type number, confirm that the change will not lead to problems.
   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

RENESAS

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

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

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

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

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

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

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

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

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

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