RX220 Group

Initial Setting

R01AN1494EJ0110
Rev. 1.10
July 1, 2014

Abstract

This document describes settings required after a reset such as clock settings or stop processing for active peripheral functions after a reset.

Products

- RX220 Group 100-pin package with a ROM size between 64 KB and 256 KB
- RX220 Group 64-pin package with a ROM size between 32 KB and 256 KB
- RX220 Group 48-pin package with a ROM size between 32 KB and 256 KB

Using this application note with the other application note

Some RX200 Series application notes support only the RX210 Group and some support multiple groups. However the sample code accompanying these application notes which support multiple groups includes the start-up program for the product listed on the Operation Confirmation Conditions section in the application note. Therefore the sample code cannot operate on a product other than the product listed on the Operation Confirmation Conditions section as it is.

To use an RX200 Series application note which supports multiple groups with the RX220 Group, replace the start-up program accompanying the RX200 Series application note with the start-up program in this application note.

Refer to “4. Applying the RX200 Series Application Note to the RX220 Group” for details.

Contents

1. Specifications

In the sample code, peripheral functions operating after a reset are stopped, and nonexistent port and clock settings are configured. The application note assumes processing at power-on (cold start).

1.1 Stopping Peripheral Functions Operating after a Reset

Some peripheral functions operate at power-on, and the module-stop function is disabled for some. These include the DMAC, DTC, and RAM0. Although the sample code includes processing for stopping these peripheral functions, it is not executed in the sample code. Change the oscillation parameters as required to execute processing.

1.2 Configuring Nonexistent Ports

Ports which are not connected to pins must be set as output for products with less than 100 pins. In the sample code, initial values are set for 100-pin products. Change the value according to the product used.

1.3 Setting Clocks

1.3.1 Overview

The clock setting procedure is as follows:

1. Sub-clock setting
2. Main clock setting
3. HOCO clock setting
4. System clock switching

In this application note, the clock settings are switched by changing the constants defined in r_init_clock.h.

In the sample code, the main clock is used as the system clock without using the sub-clock. Change the constant to select the required clock setting.

1.3.2 Clock Specifications Used in the Sample Code

Table 1.1 lists the Clock Specifications Used in the Sample Code. Values such as the oscillation stabilization wait time are calculated using values listed in Table 1.1.

Table 1.2 lists the Peripheral Function and Its Application.

Table 1.1 Clock Specifications Used in the Sample Code

Clock	Oscillation Frequency	Oscillation Stabilization Time	Remarks
Crystal/ceramic resonator for the main clock	20 MHz	4.2 ms (2)	Crystal used
Crystal for the sub-clock	32.768 kHz (1)	1.3 sec. (2)	For low clock loads
HOCO clock	32 MHz (1)	Maximum of 10 μs (3)	

Notes:

1. Sub-clock oscillation is disabled in the sample code.
2. The oscillation stabilization time of a crystal/ceramic resonator differs depending on the wiring pattern, conditions of oscillation parameters, and other settings in the user system. Ask the crystal/ceramic resonator manufacturer to evaluate the user system and provide an appropriate oscillation stabilization time.
3. Refer to the Electrical Characteristics chapter in the User's Manual: Hardware.

Table 1.2 Peripheral Function and Its Application

Peripheral Function	Application
Compare match timer, channel 0 (CMT0)	Measuring the clock oscillation stabilization wait time (1)

Note:

1. When using OS, select a channel for a timer that is not being used by OS.

1.3.3 Selecting Clocks

In the sample code, users can select the system clock source, whether clocks to be oscillating or stopped, and other settings by changing constants defined in r_init_clock.h. Refer to Table 3.9 and Table 3.10 for constants that can be changed.

Table 1.3 lists Examples of Clock Selections. In the sample code, processing No. 1, which uses the main clock as the system clock without using the sub-clock, is selected.

Table 1.3 Examples of Clock Selections

No.		1	2	3	4
System clock		Main clock	Main clock	HOCO	HOCO
Main clock		Oscillating	Oscillating	Stopped	Stopped
HOCO clock		Stopped	Stopped	Oscillating (32 MHz)	Oscillating (32 MHz)
Sub-clock (2)		Stopped	Oscillating (RTC used)	Stopped	Oscillating (RTC used)
Operating mode		Middle-speed operating mode 1A	Middle-speed operating mode 1A	Middle-speed operating mode 1A	Middle-speed operating mode 1A
Constants	SEL_SYSCLK	CLK_MAIN	CLK_MAIN	CLK_HOCO	CLK_HOCO
	SEL_MAIN	B_USE	B_USE	B_NOT_USE	B_NOT_USE
	SEL_HOCO	B_NOT_USE	B_NOT_USE	B_USE	B_USE
	SEL_SUB (1)	B_NOT_USE	B_NOT_USE	B_NOT_USE	B_NOT_USE
	SEL_RTC (1)	B_NOT_USE	B_USE	B_NOT_USE	B_USE
	REG_OPCCR	OPCM_MID_1A	OPCM_MID_1A	OPCM_MID_1A	OPCM_MID_1A

Notes:

1. When using the sub-clock as the system clock, set the value of the SEL_SUB constant to B_USE (sub-clock used). When using the sub-clock as the RTC count source, set the value of the SEL_RTC constant to B_USE. When either SEL_SUB or SEL_RTC, or both are set to B_USE, the sub-clock operates.
2. The sub-clock oscillator is controlled by bits SOSCCR.SOSTP and RCR3.RTCEN. When the sub-clock is used as the system clock, it is controlled by the SOSCCR.SOSTP bit, and when the sub-clock is used as the RTC count source, it is controlled by the RCR3.RTCEN bit. Therefore the initial setting for the sub-clock differs depending on whether the sub-clock is used as the system clock or not. Also the sub-clock starts oscillating at power-on. Therefore processing to stop the sub-clock is performed even when the sub-clock is not used.

2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

Item		Contents
MCU used		R5F52206BDFP (RX220 Group)
Operating frequencies	When the main clock is selected as the system clock (Nos. 1 and 2 in Table 1.3)	- Main clock: 20 MHz - Sub-clock: 32.768 kHz (stopped when the sub-clock is not used) - HOCO: Stopped - System clock (ICLK): 20 MHz (main clock divided by 1) - Peripheral module clock B (PCLKB): 20 MHz (main clock divided by 1) - Peripheral module clock D (PCLKD): 20 MHz (main clock divided by 1) - External bus clock (BCLK): 20 MHz (main clock divided by 1) - FlashIF clock (FCLK): 20 MHz (main clock divided by 1)
	When HOCO is selected as the system clock (Nos. 3 and 4 in Table 1.3)	- Main clock: Stopped - Sub-clock: 32.768 kHz (stopped when the sub-clock is not used) - HOCO: 32 MHz - System clock (ICLK): 32 MHz (HOCO divided by 1) - Peripheral module clock B (PCLKB): 32 MHz (HOCO divided by 1) - Peripheral module clock D (PCLKD): 32 MHz (HOCO divided by 1) - External bus clock (BCLK): 16 MHz (HOCO divided by 2) - FlashIF clock (FCLK): 32 MHz (HOCO divided by 1)
Operating voltage		5.0 V
Integrated development environment		Renesas Electronics Corporation High-performance Embedded Workshop Version 4.09.01
C compiler		Renesas Electronics Corporation C/C++ Compiler Package for RX Family V.1.02 Release 01
		Compile options -cpu=rx200 -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" -debug -nologo (The default setting is used in the integrated development environment.)
iodefine.h version		Version 1.0A
Endian		Little endian
Operating mode		Single-chip mode
Processor mode		Supervisor mode
Sample code version		Version 1.00
Board used		Renesas Starter Kit for RX220 (R0K505220S000BE)

3. Software

In the sample code, peripheral functions operating after a reset are stopped, nonexistent ports are configured, and then clock settings are configured.

3.1 Stop Processing for Active Peripheral Functions after a Reset

Peripheral functions that are operating after a reset are stopped in this processing.

The module-stop state is canceled after a reset only for modules listed in Table 3.1. To enter the module-stop state, set the module stop bit to 1 (transition to the module-stop state is made). Power consumption can be reduced by entering the module-stop state.

In the sample code, set the MSTP_STATE_"target module" constant to 0 (MODULE_STOP_DISABLE), so the target module does not enter the module-stop state. When the system requires a module to enter the module-stop state, set the constant in r_init_stop_module.h to 1 (MODULE_STOP_ENABLE).

Table 3.1 lists the Peripheral Modules whose Module-Stop States are Canceled after a Reset.

Table 3.1 Peripheral Modules whose Module-Stop States are Canceled after a Reset

<table>
<tr><th>Peripheral Module</th><th>Module Stop Bit</th><th>Value after a Reset</th><th>Value when not Using the Module</th></tr>
<tr><td>DMAC/DTC</td><td>MSTPCRA.MSTPA28 bit</td><td rowspan="2">0
(module-stop state is canceled)</td><td rowspan="2">1
(transition to the module-stop state is made)</td></tr>
<tr><td>RAM0</td><td>MSTPCRC.MSTPC0 bit</td></tr>
</table>

3.2 Nonexistent Port Initialization

3.2.1 Overview

When using a product with less than 100 pins, set the corresponding bits of nonexistent ports in the PDR register to 1 (output). After the nonexistent port initialization function is called, when writing in byte units to the PDR registers or PODR registers which have nonexistent ports, set the corresponding bits for nonexistent ports as follows: set the I/O select bits in the PDR registers to 1 and set the output data store bits in the PODR registers to 0.

Table 3.2 lists Nonexistent Ports.

Table 3.2 Nonexistent Ports

Port Symbol	64-Pin Package	Number of Pins	48-Pin Package	Number of Pins
PORT0	P07	1	P03, P05, P07	3
PORT1	P12, P13	2	P12, P13	2
PORT2	P20 to P25	6	P20 to P25	6
PORT3	P33, P34	2	P32 to P34	3
PORT4	P45, P47	2	P43 to P45, P47	4
PORT5	P50 to P53	4	P50 to P55	6
PORTA	PA2, PA5, PA7	3	PA0, PA2, PA5, PA7	4
PORTB	PB2, PB4	2	PB2, PB4, PB6, PB7	4
PORTC	PC0, PC1	2	PC0 to PC3	4
PORTD	PD0 to PD7	8	PD0 to PD7	8
PORTE	PE6, PE7	2	PE0, PE5 to PE7	4
PORTH	—	—	—	—
PORTJ	PJ1, PJ3	2	PJ1, PJ3	2

3.2.2 Selecting the Number of Pins

The number of pins in the sample code is set for the 100-pin package (PIN_SIZE=100). This application note covers 100-pin, 64-pin, and 48-pin packages. When using products with less than 100 pins, change PIN_SIZE in r_init_non_existent_port.h to the number of pins on the package.

3.3 Clock Settings

3.3.1 Clock Setting Procedure

Table 3.3 lists the Clock Setting with each processing and setting in the sample code. In the sample code, the main clock is operated, and HOCO and the sub-clock are stopped.

Table 3.3 Clock Setting Procedure

Step	Processing	Details		Setting in the Sample Code
1	Sub-clock setting (2)	Not used	Initialize the sub-clock control circuit.	Sub-clock is not used.
		Used	Initialize the sub-clock control circuit, set the driving ability, and set the SOSCWTCR register with a wait time until the sub-clock output is provided to the internal clock, and then enable the sub-clock oscillation. Then wait for the oscillation stabilization wait time (1) by software.	
2	Main clock setting (2)	Not used	No setting is required.	Main clock is used.
		Used	Set the main clock driving ability, set the MOSCWTCR register with a wait time until the main clock output is provided to the internal clock, and then enable the main clock oscillation. Then wait for the oscillation stabilization wait time (1) by software.	
3	HOCO clock setting (2)	Not used	Turn off the HOCO power supply.	HOCO clock is not used.
		Used	Set the HOCO frequency, set the HOCOWTCR2 register with a wait time until the HOCO clock output is provided to the internal clock, and then enable the HOCO clock oscillation. Then wait for the oscillation stabilization wait time (1) by software.	
4	Clock division ratio setting	Change the clock division ratio.		ICLK, PCLKB, PCLKD, BCLK, FCLK: Divided by 1
5	System clock switching	Switch the system clock according to the user system.		Switched to the main clock.
6	Operating power control mode setting	Set the operating power control mode according to the operating frequency and operating voltage in the user system.		Middle-speed operating mode 1A is set

Notes:

1. Refer to 3.3.2 Oscillation Stabilization Wait Time for Each Clock for details on the oscillation stabilization wait time.
2. When selecting each clock usage, change the appropriate constant in r_init_clock.h as required.

3.3.2 Oscillation Stabilization Wait Time for Each Clock

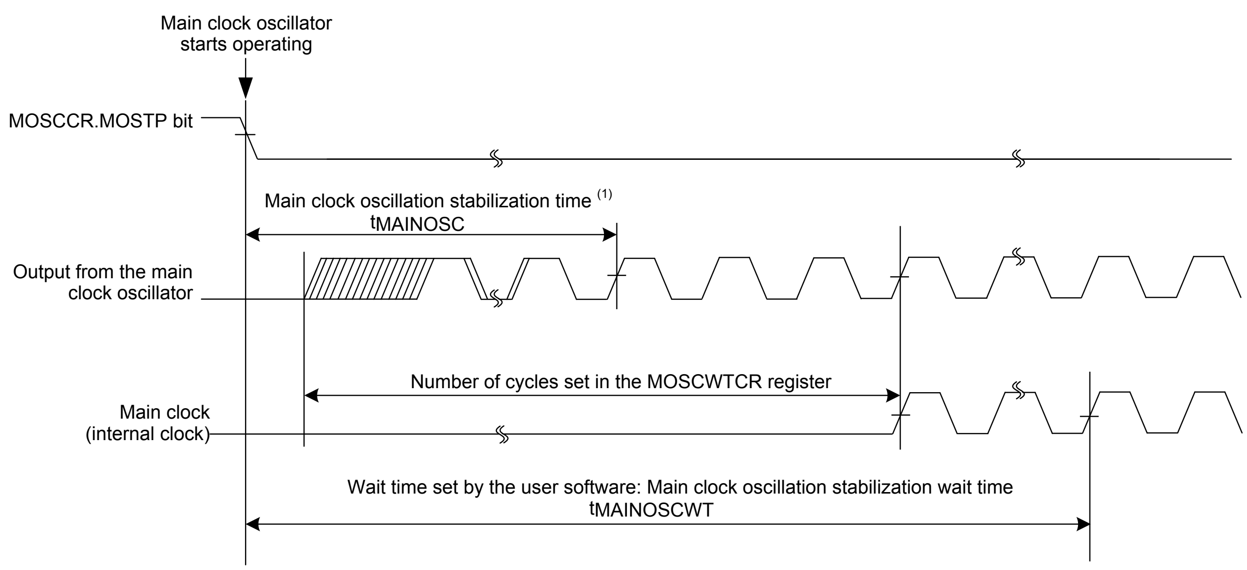
This section describes the wait control registers and oscillation stabilization wait times for the main clock, sub-clock, and HOCO clock.

3.3.2.1 Main Clock Oscillation Stabilization Wait Time

Figure 3.1 shows the Main Clock Oscillation Stabilization Wait Time and Table 3.4 lists the Setting Value for the MOSCWTCR Register and Oscillation Stabilization Wait Time.

Set the main clock oscillator wait control register (MOSCWTCR) to a value greater than or equal to the main clock oscillation stabilization time (tMAINOSC) recommended by the crystal/ceramic resonator manufacturer. Set the main clock oscillation stabilization wait time (tMAINOSCWT) to a value greater than two times the number of cycles set in the MOSCWTCR register.

tMAINOSC used in the sample code is 4.2 ms, thus the setting value in the MOSCWTCR register is 0Dh (approximately 6.55 ms), and the setting value for tMAINOSCWT is approximately 13.1 ms.



Note:
1. The oscillation stabilization time of a crystal/ceramic resonator differs depending on the wiring pattern, conditions of oscillation parameters, and other settings in the user system. Ask the crystal/ceramic resonator manufacturer to evaluate the user system and provide an appropriate oscillation stabilization time.

Figure 3.1 Main Clock Oscillation Stabilization Wait Time

Table 3.4 Setting Value for the MOSCWTCR Register and Oscillation Stabilization Wait Time

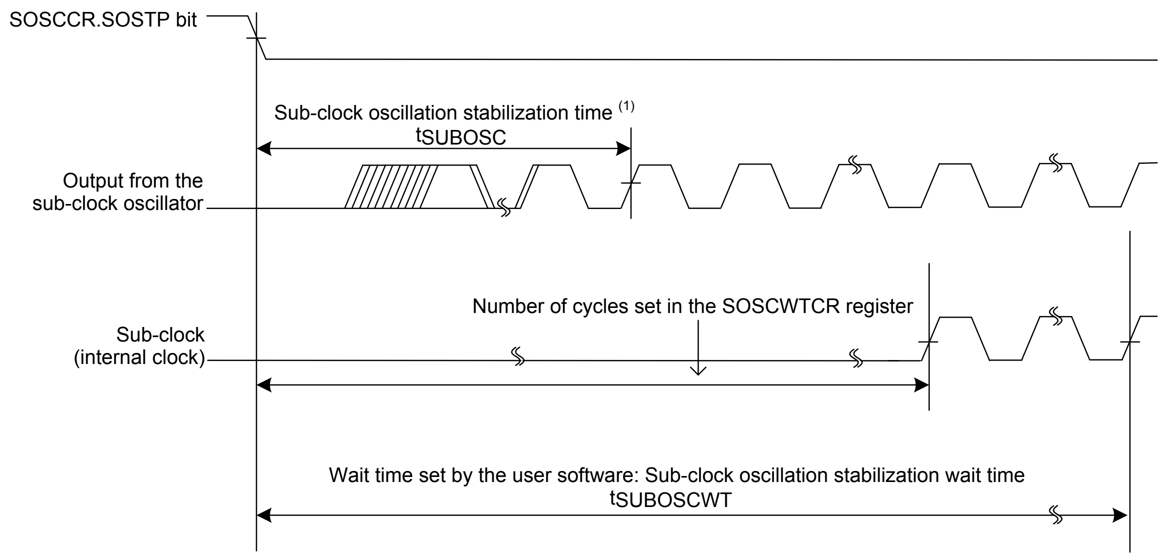
Setting Item	Condition of Setting Value	Setting Value in the Sample Code
MOSCWTCR.MSTS[4:0] bits	Value greater than or equal to tMAINOSC recommended by the crystal/ceramic resonator manufacturer	0Dh (approx. 6.5536 ms)
Oscillation stabilization wait time (tMAINOSCWT)	Value greater than or equal to two times the number of cycles set in the MOSCWTCR register.	Approx. 13.1072 ms

3.3.2.2 Sub-Clock Oscillation Stabilization Wait Time

Figure 3.2 shows the Sub-Clock Oscillation Stabilization Wait Time and Table 3.5 lists the Setting Value of the SOSCWTCR Register and Oscillation Stabilization Wait Time.

Set the sub-clock oscillator wait control register (SOSCWTCR) to a value greater than or equal to the sub-clock oscillation stabilization time (tSUBOSC) recommended by the crystal/ceramic resonator manufacturer. Set the sub-clock oscillation stabilization wait time (tSUBOSCWT) to a value greater than or equal to two times the value set in the SOSCWTCR register.

tSUBOSC used in the sample code is 1.3 seconds, thus the setting value in the SOSCWTCR register is 00h (2 sec. + approx. 61 µs), and the setting value for tSUBOSCWT is approximately 4 seconds.



Notes:
1. Contact the crystal/ceramic resonator manufacturer to determine the oscillation stabilization time of a crystal/ceramic resonator for the user system. The oscillation stabilization time is not a condition for MCU operation but for a crystal/ceramic resonator to start oscillation.
2. Refer to the Electrical Characteristics chapter in the User's Manual: Hardware.

Figure 3.2 Sub-Clock Oscillation Stabilization Wait Time

Table 3.5 Setting Value of the SOSCWTCR Register and Oscillation Stabilization Wait Time

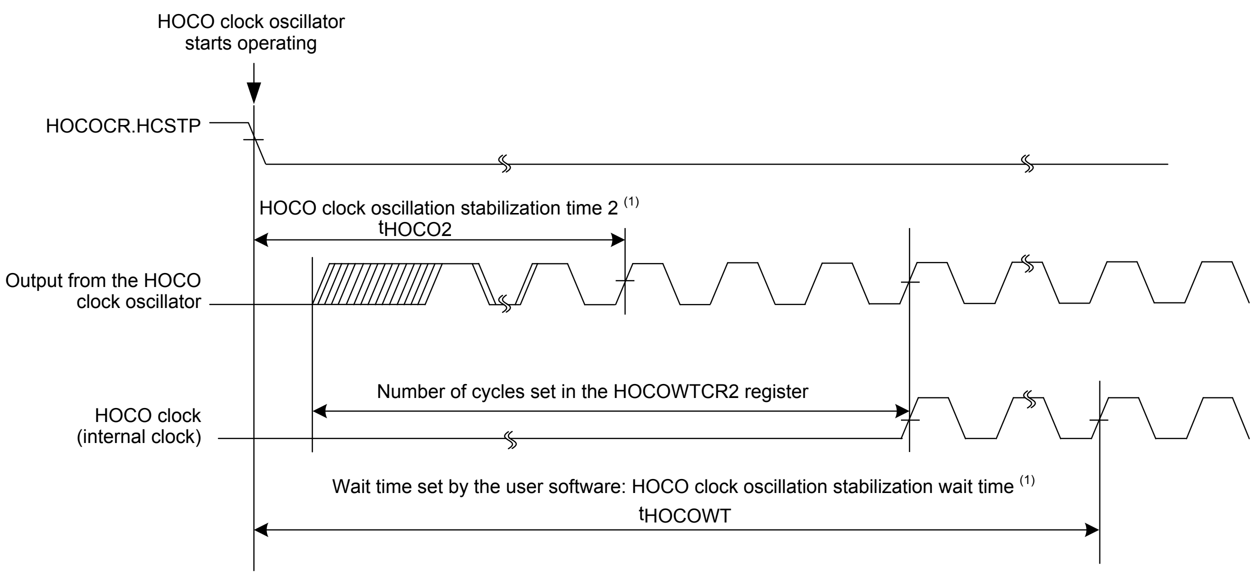
Setting Item	Condition of Setting Value	Setting Value in the Sample Code
SOSCWTCR.SSTS[4:0] bits	Value greater than or equal to tSUBOSC recommended by the crystal/ceramic resonator manufacturer.	00h (2 sec. + approx. 61 µs)
Oscillation stabilization wait time (tSUBOSCWT)	Value greater than or equal to two times the set value (2 sec. + specified number of cycles) in the SOSCWTCR register	Approx. 4 sec.

3.3.2.3 HOCO Clock Oscillation Stabilization Wait Time

Figure 3.3 shows the HOCO Clock Oscillation Stabilization Wait Time and Table 3.6 lists the Setting Value of the HOCOWTCR2 Register and Oscillation Stabilization Wait Time.

Set 14h (180 cycles) to the HOCO wait control register (HOCOWTCR2) when the HOCO clock oscillation frequency (fHOCO) is other than 50 MHz (32/36.864/40 MHz), and 15h (200 cycles) when fHOCO is 50 MHz. Set the HOCO clock oscillation stabilization wait time (tHOCOWT) to a value greater than or equal to 20 µs.

fHOCO used in the sample code is 32 MHz, thus the setting value in the HOCOWTCR2 register is 14h (approximately 5.625 µs), and the setting value for tHOCOWT is approximately 20 µs.



Note:
1. Refer to the Electrical Characteristics chapter in the User's Manual: Hardware.

Figure 3.3 HOCO Clock Oscillation Stabilization Wait Time

Table 3.6 Setting Value of the HOCOWTCR2 Register and Oscillation Stabilization Wait Time

Setting Item	Condition of Setting Value	Setting Value in the Sample Code
HOCOWTCR2. HSTS2[4:0] bits	- When fHOCO is other than 50 MHz: 14h (180 cycles) - When fHOCO is 50 MHz: 15h (200 cycles)	14h (approx. 5.625 µs)
Oscillation stabilization wait time (tHOCOWT)	Value greater than or equal to 20 µs	Approx. 20 µs

3.4 File Composition

Table 3.7 lists the Files Used in the Sample Code. Files generated by the integrated development environment should not be listed in this table.

Table 3.7 Files Used in the Sample Code

File Name	Outline	Remarks
main.c	Main processing	
r_init_stop_module.c	Stop processing for active peripheral functions after a reset	
r_init_stop_module.h	Header file for r_init_stop_module.c	
r_init_non_existent_port.c	Nonexistent port initialization	
r_init_non_existent_port.h	Header file for r_init_non_existent_port.c	
r_init_clock.c	Clock initialization	
r_init_clock.h	Header file for r_init_clock.c	

3.5 Option-Setting Memory

Table 3.8 lists the Option-Setting Memory Configured in the Sample Code. When necessary, set a value suited to the user system.

Table 3.8 Option-Setting Memory Configured in the Sample Code

Symbol	Address	Setting Value	Contents
OFS0	FFFF FF8Fh to FFFF FF8Ch	FFFF FFFFh	The IWDT is stopped after a reset.
OFS1	FFFF FF8Bh to FFFF FF88h	FFFF FFFFh	The voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.
MDES	FFFF FF83h to FFFF FF80h	FFFF FFFFh	Little endian

3.6 Constants

Table 3.9 and Table 3.10 list the constants used in the sample code, which can be changed by users. Table 3.11 lists the constants used in the sample code, which cannot be changed by users. Table 3.12 lists the Constants when a 100-Pin Package is Used (PIN_SIZE=100), Table 3.13 lists the Constants when a 64-Pin Package is Used (PIN_SIZE=64), and Table 3.14 lists the Constants when a 48-Pin Package is Used (PIN_SIZE=48).

Table 3.9 Constants Used in the Sample Code (1/2)
(Users can change the constants listed in this table.)

Constant Name	Setting Value	Contents
SEL_MAIN (1)	B_USE	Selection of the main clock operation: - B_USE: Used (main clock oscillating) - B_NOT_USE: Not used (main clock stopped)
MAIN_CLOCK_Hz (1)	20,000,000 L	Oscillation frequency of a crystal/ceramic resonator for the main clock (Hz)
REG_MOFCR (1)	30h	Setting for the driving ability of the main clock oscillator (set value in the MOFCR register)
REG_MOSCWTCR (1)	0Dh	Set value in the main clock wait control register
WAIT_TIME_FOR_MAIN_OSCILLATION (1)	13,107,200 L	Main clock oscillation stabilization wait time (ns)
SEL_SUB (1, 2)	B_NOT_USE	Selection of the sub-clock usage for the system clock: - B_USE: Used - B_NOT_USE: Not used
SEL_RTC (1, 2)	B_NOT_USE	Selection of the sub-clock usage for the RTC count source: - B_USE: Used - B_NOT_USE: Not used
SUB_CLOCK_Hz (1)	32,768 L	Oscillation frequency of a crystal for the sub-clock (Hz)
REG_SOSCWTCR (1)	00h	Set value in the sub-clock wait control register
WAIT_TIME_FOR_SUB_OSCILLATION (1)	4,000,000,000 L	Sub-clock oscillation stabilization wait time (ns)
REG_RCR3 (1)	CL_LOW	Selection of the sub-clock oscillator driving ability: - CL_STD: Standard clock loads - CL_LOW: Low clock loads
SEL_CNTMD (1)	CNTMD_CAL	Selection of the real-time clock count mode - CNTMD_CAL: Calendar count mode - CNTMD_BIN: Binary count mode

Notes:
1. Change the setting value in r_init_clock.h according to the user system.
2. The sub-clock operation is set to be oscillating by setting B_USE (sub-clock used) to either of the SEL_SUB constant or SEL_RTC constant, or both.

Table 3.10 Constants Used in the Sample Code (2/2)
(Users can change the constants listed in this table.)

Constant Name	Setting Value	Contents
SEL_HOCO (1)	B_NOT_USE	Selection of the HOCO clock operation: - B_USE: Used (HOCO clock oscillating) - B_NOT_USE: Not used (HOCO clock stopped)
REG_HOCOCR2 (1)	FREQ_32MHz	Selection of the HOCO clock frequency - FREQ_32MHz: 32 MHz - FREQ_36MHz: 36.864 MHz - FREQ_40MHz: 40 MHz - FREQ_50MHz: 50 MHz
WAIT_TIME_FOR_HOCO_OSCILLATION (1)	20,000 L	HOCO clock oscillation stabilization wait time (ns)
SEL_SYSCLK (1)	CLK_MAIN	Clock source selection for the system clock - CLK_HOCO: HOCO - CLK_MAIN: Main clock - CLK_SUB: Sub-clock
REG_OPCCR (1)	OPCM_MID_1A	Selection of the operating power control mode (5) - OPCM_MID_1A: Middle-speed operating mode 1A - OPCM_MID_1B: Middle-speed operating mode 1B - OPCM_LOW_1: Low-speed operating mode 1 - OPCM_LOW_2: Low-speed operating mode 2 (4)
MSTP_STATE_DMACDTC (2)	MODULE_STOP_DISABLE	Selection of the module-stop state for DMAC and DTC - MODULE_STOP_DISABLE: Module-stop state canceled - MODULE_STOP_ENABLE: Entering the module-stop state
MSTP_STATE_RAM0 (2)	MODULE_STOP_DISABLE	Selection of the module-stop state for RAM0 - MODULE_STOP_DISABLE: Operating - MODULE_STOP_ENABLE: Stopped
PIN_SIZE (3)	100	Number of pins on the product used

Notes:
1. Change the setting value in r_init_clock.h according to the user system.
2. Change the setting value in r_init_stop_module.h according to the user system.
3. Change the setting value in r_init_non_existent_port.h according to the user system.
4. When HOCO is set to be oscillating, low-speed operating mode 2 is not available.
5. The ranges of the operating frequency and operating voltage differ depending on operating modes. Refer to the User’s Manual: Hardware for details.

Table 3.11 Constants Used in the Sample Code
(Users cannot change the constants listed in this table.)

Constant Name	Setting Value	Contents
B_NOT_USE	0	Not used
B_USE	1	Used
CL_LOW	02h	Sub-clock: Drive ability for low clock loads
CL_STD	0Ch	Sub-clock: Drive ability for standard clock loads
CNTMD_CAL	0	RTC: Calendar count mode
CNTMD_BIN	1	RTC: Binary count mode
FREQ_32MHz	00h	HOCO frequency: 32 MHz
FREQ_36MHz	01h	HOCO frequency: 36.684 MHz
FREQ_40MHz	02h	HOCO frequency: 40 MHz
FREQ_50MHz	03h	HOCO frequency: 50 MHz
CLK_HOCO	0100h	Clock source: HOCO
CLK_SUB	0300h	Clock source: Sub-clock
CLK_MAIN	0200h	Clock source: Main clock
REG_HOCOWTCR2 (1)	- 15h (when 50 MHz is selected) - 14h (when other than 50 MHz is selected)	Set value in the HOCO wait control register
REG_SCKCR (2)	- 0001 1010h (when HOCO is selected) - 0000 1010h (other than above)	Setting for the internal clock division ratio (set value in the SCKCR register)
OPCM_MID_1A	02h	Middle-speed operating mode 1A
OPCM_MID_1B	03h	Middle-speed operating mode 1B
OPCM_LOW_1	06h	Low-speed operating mode 1
OPCM_LOW_2	07h	Low-speed operating mode 2
MAIN_CLOCK_CYCLE	(1,000,000,000L / MAIN_CLOCK_Hz)	Main clock cycles (ns)
SUB_CLOCK_CYCLE	(1,000,000,000L / SUB_CLOCK_Hz)	Sub-clock cycles (ns)
FOR_CMT0_TIME	232727	Count cycles (ns) for the CMT0 timer to wait for the oscillation stabilization wait time: (1/LOCO) × 32, where LOCO = 137.5 kHz (max.), and 32 = PCLKB divided by 32
MODULE_STOP_ENABLE	1	Transition to the module stop-state is made
MODULE_STOP_DISABLE	0	Module stop-state is canceled

Notes:

1. The setting value varies depending on the HOCO frequency selected.
2. The setting value varies depending on the clock source of the system clock selected.

Table 3.12 Constants when a 100-Pin Package is Used (PIN_SIZE=100)

Constant Name	Setting Value	Contents
DEF_P0PDR	0x00	Setting value for the port P0 direction register
DEF_P1PDR	0x00	Setting value for the port P1 direction register
DEF_P2PDR	0x00	Setting value for the port P2 direction register
DEF_P3PDR	0x00	Setting value for the port P3 direction register
DEF_P4PDR	0x00	Setting value for the port P4 direction register
DEF_P5PDR	0x00	Setting value for the port P5 direction register
DEF_PAPDR	0x00	Setting value for the port PA direction register
DEF_PBPDR	0x00	Setting value for the port PB direction register
DEF_PCPDR	0x00	Setting value for the port PC direction register
DEF_PDPDR	0x00	Setting value for the port PD direction register
DEF_PEPDR	0x00	Setting value for the port PE direction register
DEF_PHPDR	0x00	Setting value for the port PH direction register
DEF_PJPDR	0x00	Setting value for the port PJ direction register

Table 3.13 Constants when a 64-Pin Package is Used (PIN_SIZE=64)

Constant Name	Setting Value	Contents
DEF_P0PDR	0x80	Setting value for the port P0 direction register
DEF_P1PDR	0x0C	Setting value for the port P1 direction register
DEF_P2PDR	0x3F	Setting value for the port P2 direction register
DEF_P3PDR	0x18	Setting value for the port P3 direction register
DEF_P4PDR	0xA0	Setting value for the port P4 direction register
DEF_P5PDR	0x0F	Setting value for the port P5 direction register
DEF_PAPDR	0xA4	Setting value for the port PA direction register
DEF_PBPDR	0x14	Setting value for the port PB direction register
DEF_PCPDR	0x03	Setting value for the port PC direction register
DEF_PDPDR	0xFF	Setting value for the port PD direction register
DEF_PEPDR	0xC0	Setting value for the port PE direction register
DEF_PHPDR	0x00	Setting value for the port PH direction register
DEF_PJPDR	0x0A	Setting value for the port PJ direction register

Table 3.14 Constants when a 48-Pin Package is Used (PIN_SIZE=48)

Constant Name	Setting Value	Contents
DEF_P0PDR	0xA8	Setting value for the port P0 direction register
DEF_P1PDR	0x0C	Setting value for the port P1 direction register
DEF_P2PDR	0x3F	Setting value for the port P2 direction register
DEF_P3PDR	0x1C	Setting value for the port P3 direction register
DEF_P4PDR	0xB8	Setting value for the port P4 direction register
DEF_P5PDR	0x3F	Setting value for the port P5 direction register
DEF_PAPDR	0xA5	Setting value for the port PA direction register
DEF_PBPDR	0xD4	Setting value for the port PB direction register
DEF_PCPDR	0x0F	Setting value for the port PC direction register
DEF_PDPDR	0xFF	Setting value for the port PD direction register
DEF_PEPDR	0xE1	Setting value for the port PE direction register
DEF_PHPDR	0x00	Setting value for the port PH direction register
DEF_PJPDR	0x0A	Setting value for the port PJ direction register

3.7 Functions

Table 3.15 lists the Functions Used in the Sample Code.

Table 3.15 Functions Used in the Sample Code

Function Name	Outline
main	Main processing
R_INIT_StopModule	Stop processing for active peripheral functions after a reset
R_INIT_NonExistentPort	Nonexistent port initialization
R_INIT_Clock	Clock initialization
CGC_oscillation_main	Main clock oscillation setting
CGC_oscillation_HOCO	HOCO clock oscillation setting
CGC_oscillation_sub	Sub-clock oscillation setting
CGC_disable_subclk	Sub-clock stop setting
oscillation_subclk	Enabling sub-clock oscillation
no_use_subclk_as_sysclk	Processing when the sub-clock is not used as the system clock
resetting_wtcr_subclk	Resetting the sub-clock wait control register
enable_RTC	Initialization when using the RTC
cmt0_wait	Wait processing

3.8 Function Specifications

The following tables list the sample code function specifications.

main	
Outline	Main processing
Header	None
Declaration	void main(void)
Description	Call the following functions: Stop processing for active peripheral functions after a reset, nonexistent port initialization, and clock initialization.
Arguments	None
Return Value	None

R_INIT_StopModule	
Outline	Stop processing for active peripheral functions after a reset
Header	r_init_stop_module.h
Declaration	void R_INIT_StopModule(void)
Description	Configure the setting to enter the module-stop state.
Arguments	None
Return Value	None
Remarks	Transition to the module-stop state is not performed in the sample code.

R_INIT_NonExistentPort	
Outline	Nonexistent port initialization
Header	r_init_non_existent_port.h
Declaration	void R_INIT_NonExistentPort(void)
Description	Initialize port direction registers for ports that do not exist in products with less than 100 pins.
Arguments	None
Return Value	None
Remarks	The number of pins in the sample code is set for the 100-pin package (PIN_SIZE=100). After this function is called, when writing in byte units to the PDR registers or PODR registers which have nonexistent ports, set the corresponding bits for nonexistent ports as follows: set the I/O select bits in the PDR registers to 1 and set the output data store bits in the PODR registers to 0.

R_INIT_Clock	
Outline	Clock initialization
Header	r_init_clock.h
Declaration	void R_INIT_Clock(void)
Description	Initialize the clock.
Arguments	None
Return Value	None
Remarks	The sample code selects processing which uses the main clock as the system clock without using the sub-clock.

CGC_oscillation_main	
Outline	Main clock oscillation setting
Header	r_init_clock.h
Declaration	void CGC_oscillation_main(void)
Description	Set the main clock driving ability, set the MOSCWTCR register, and enable main clock oscillation. Then wait for the main clock oscillation stabilization wait time by software.
Arguments	None
Return Value	None

CGC_oscillation_HOCO	
Outline	HOCO clock oscillation setting
Header	r_init_clock.h
Declaration	void CGC_oscillation_HOCO(void)
Description	Set the HOCO frequency, set the HOCOWTCR2 register, and enable HOCO oscillation. Then wait for the HOCO clock oscillation stabilization wait time by software.
Arguments	None
Return Value	None

CGC_oscillation_sub	
Outline	Sub-clock oscillation setting
Header	r_init_clock.h
Declaration	void CGC_oscillation_sub(void)
Description	Configure the setting when the sub-clock is used as either the system clock or the RTC count source, or both.
Arguments	None
Return Value	None

CGC_disable_subclk	
Outline	Sub-clock stop setting
Header	r_init_clock.h
Declaration	void CGC_disable_subclk(void)
Description	Configure the setting when the sub-clock is not used as the system clock or the RTC count source.
Arguments	None
Return Value	None

oscillation_subclk	
Outline	Enabling the sub-clock oscillation
Header	None
Declaration	static void oscillation_subclk(void)
Description	Configure the sub-clock oscillation.
Arguments	None
Return Value	None

no_use_subclk_as_sysclk	
Outline	Processing when the sub-clock is not used as the system clock
Header	None
Declaration	static void no_use_subclk_as_sysclk(void)
Description	Stop the sub-clock as the system clock when the sub-clock is used only as the RTC count source.
Arguments	None
Return Value	None

resetting_wtcr_ subclk	
Outline	Resetting the sub-clock wait control register
Header	None
Declaration	static void resetting_wtcr_subclk(void)
Description	Reset the wait control register when exiting from software standby mode. Set the minimum value to the wait control register.
Arguments	None
Return Value	None

enable_RTC	
Outline	Initialization when using the RTC
Header	None
Declaration	static void enable_RTC (void)
Description	Initialize the settings when using the RTC (setting for clock provision and RTC software reset).
Arguments	None
Return Value	None

cmt0_wait		
Outline	Wait processing	
Header	None	
Declaration	static void cmt0_wait(uint32_t cnt)	
Description	This function is used when waiting for the oscillation stabilization wait time.	
Arguments	uint32_t cnt:	Oscillation stabilization wait time cnt = oscillation stabilization wait time (ns) (1) ÷ FOR_CMT0_TIME (2)
Return Value	None	
Remarks	1. The oscillation stabilization wait time varies depending on the crystal/ceramic resonator. Set the value referring to 3.3.2 Oscillation Stabilization Wait Time for Each Clock. 2. The value of FOR_CMT0_TIME is calculated when LOCO is 137.5 kHz (max.). The actual wait time may differ depending on the LOCO frequency.	

3.9 Flowcharts

3.9.1 Main Processing

Figure 3.4 shows the Main Processing.

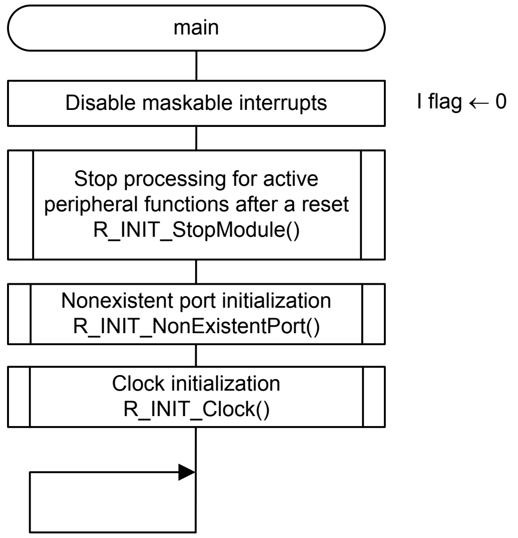
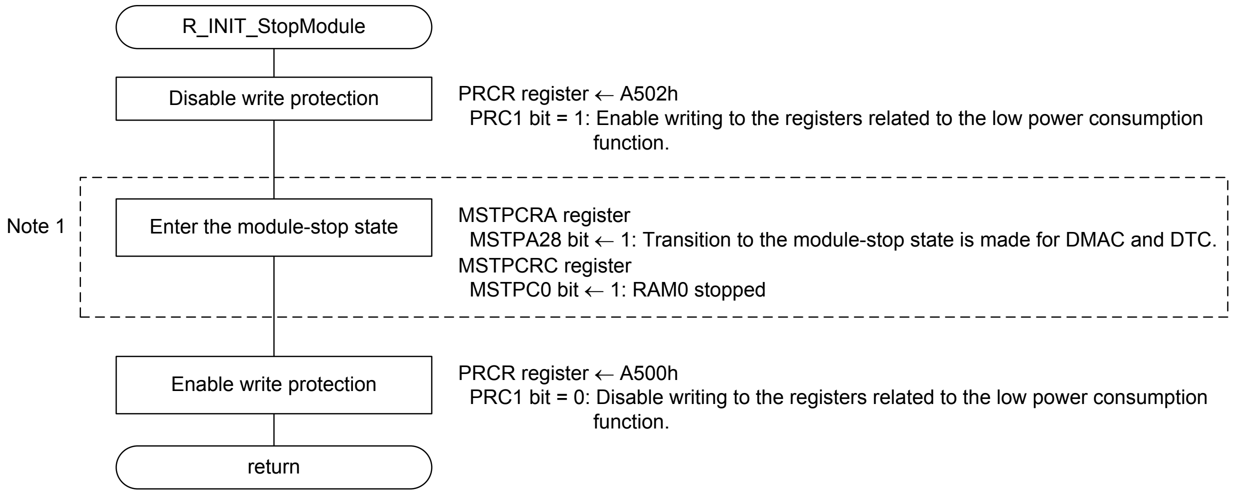


Figure 3.4 Main Processing

3.9.2 Stop Processing for Active Peripheral Functions after a Reset

Figure 3.5 shows the Stop Processing for Active Peripheral Functions after a Reset.



Note:
1. The module-stop state is canceled in the sample code. When entering the module-stop state for any peripheral functions, set the #define MSTP_STATE_"target module" constant to 1.

Figure 3.5 Stop Processing for Active Peripheral Functions after a Reset

3.9.3 Nonexistent Port Initialization

Figure 3.6 shows the Nonexistent Port Initialization.

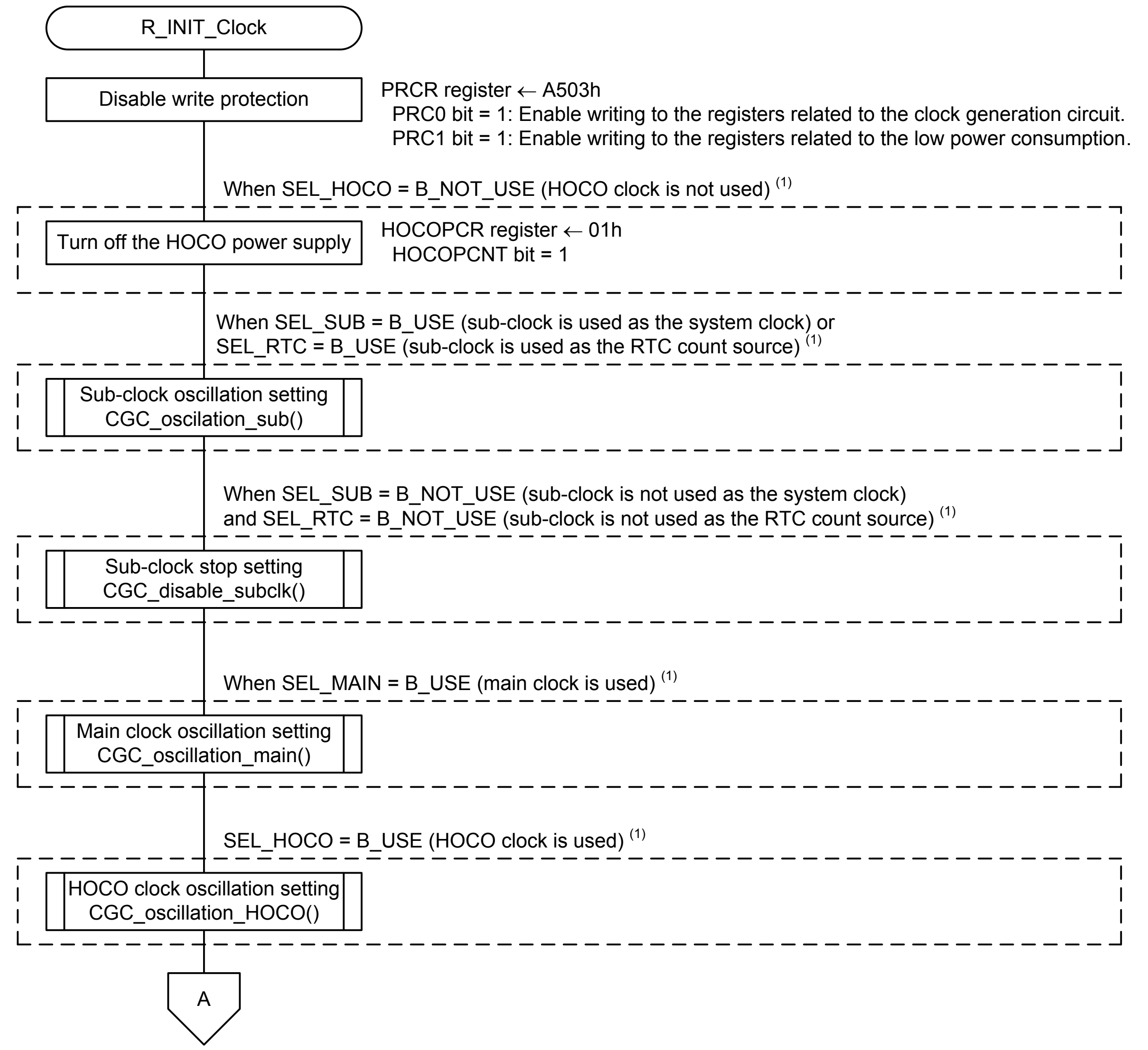
Note:

1. Port initialization processing is not executed for registers which have all pins (processing is skipped when compiling).

Figure 3.6 Nonexistent Port Initialization

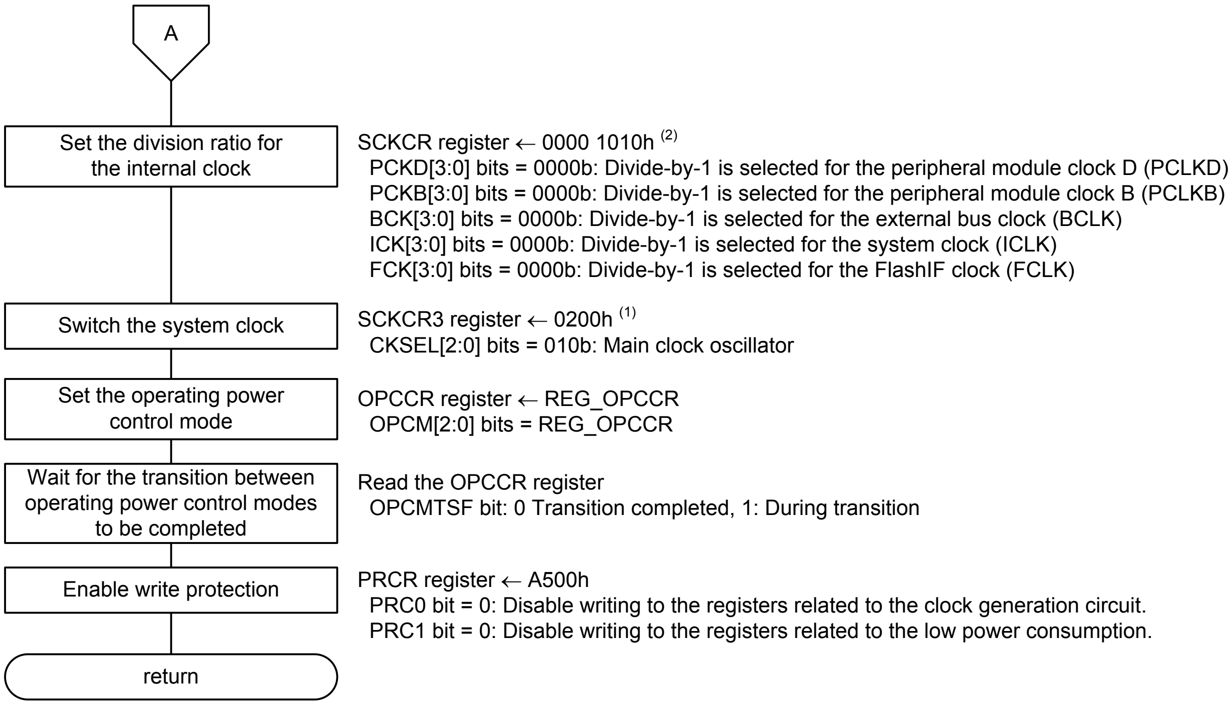
3.9.4 Clock Initialization

Figure 3.7 and Figure 3.8 show the clock initialization.



Note:
1. Change the constant value according to the user system.

Figure 3.7 Clock Initialization (1/2)



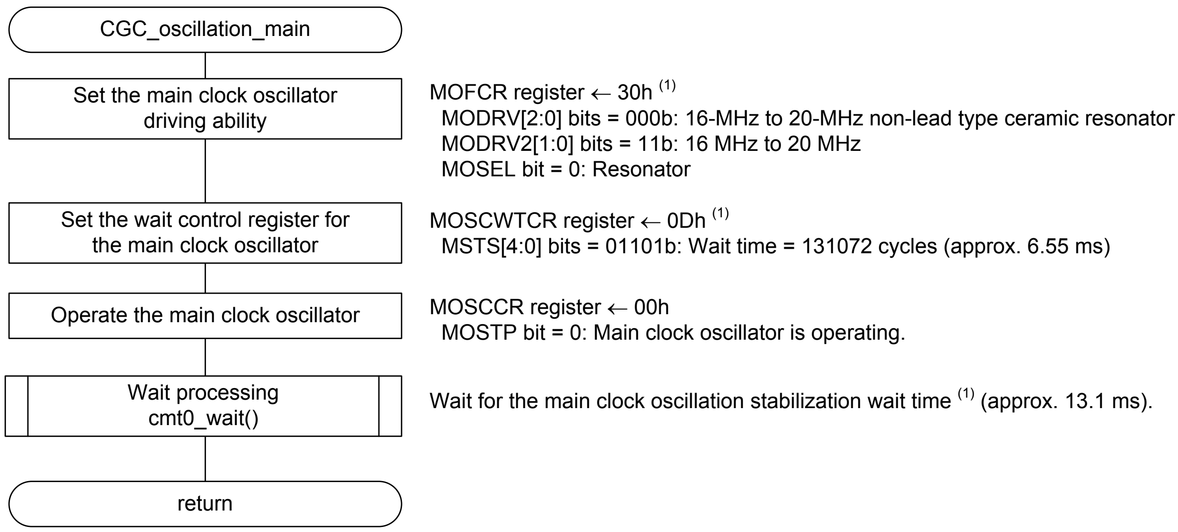
Notes:

1. Change the constant value according to the user system.
2. The setting value varies depending on the system clock selected by the constant.

Figure 3.8 Clock Initialization (2/2)

3.9.5 Main Clock Oscillation Setting

Figure 3.9 shows the Main Clock Oscillation Setting.



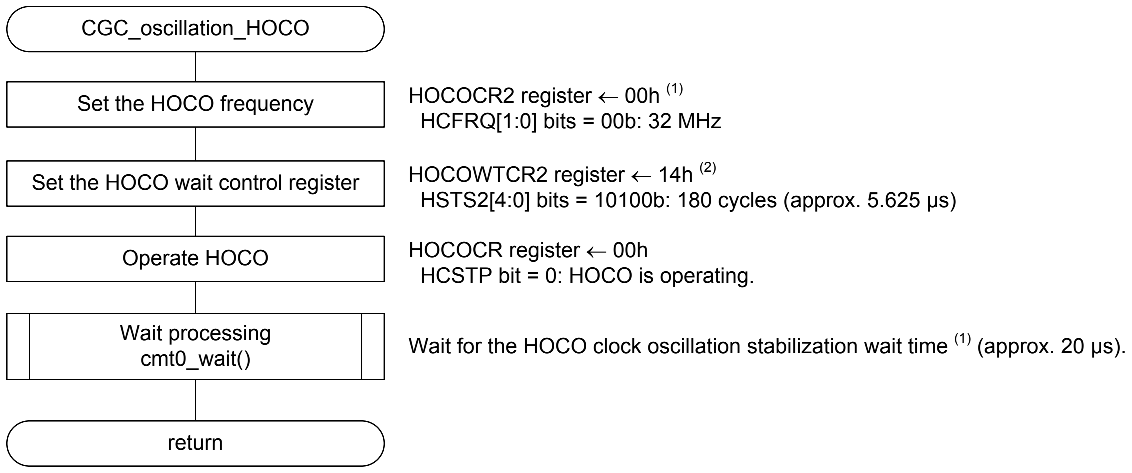
Note:

1. Change the constant value according to the user system.

Figure 3.9 Main Clock Oscillation Setting

3.9.6 HOCO Clock Oscillation Setting

Figure 3.10 shows the HOCO Clock Oscillation Setting.

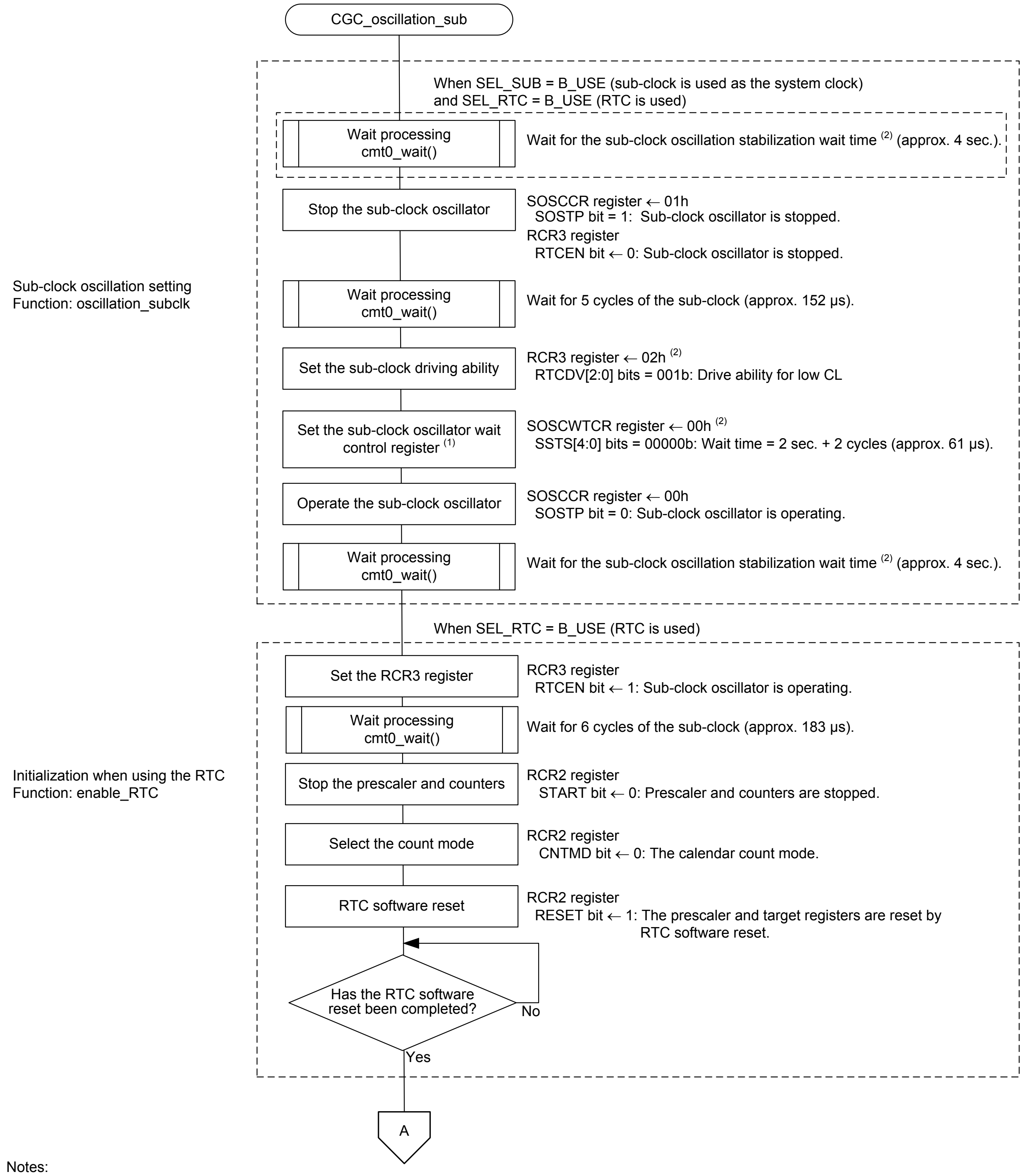


Notes:
1. Change the constant value according to the user system.
2. The setting value varies depending on the HOCO frequency selected by the constant.

Figure 3.10 HOCO Clock Oscillation Setting

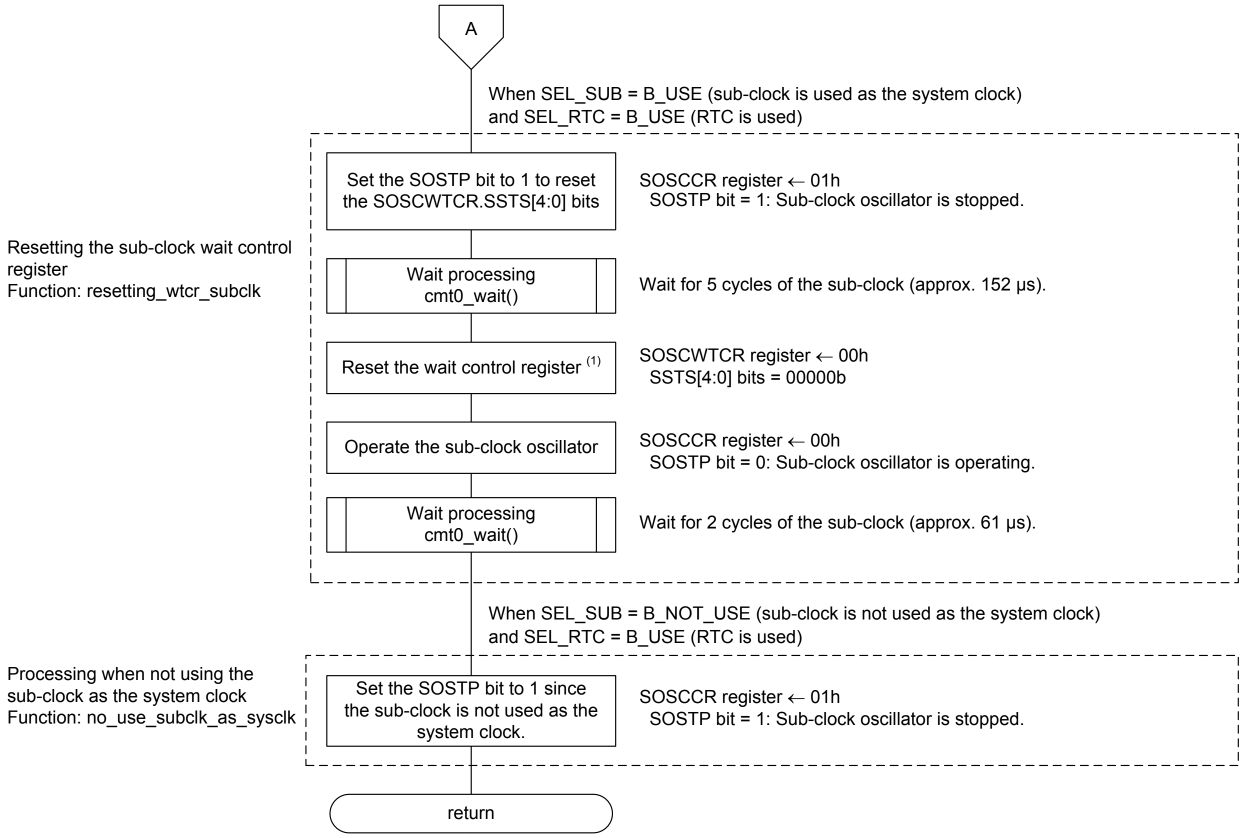
3.9.7 Sub-Clock Oscillation Setting

Figure 3.11 and Figure 3.12 show the sub-clock oscillation setting.



Notes:
1. Set the sub-clock oscillator wait control register to a value greater than or equal to the sub-clock oscillation stabilization time recommended by the crystal/ceramic resonator manufacturer.
2. Change the constant value according to the user system.

Figure 3.11 Sub-Clock Oscillation Setting (1/2)



Note:
1. When exiting from software standby mode while the sub-clock is selected as the system clock, the clock provision into the LSI is started after the specified time set in the SOSCWTCR.SSTS[4:0] bits has elapsed since the condition to exit has occurred. However when the RCR3.RTCEN bit is 1 (operating), the sub-clock does not stop even during software standby mode. Then the sub-clock oscillation stabilization wait time is not required when exiting. Therefore 00000b is set to the SOSCWTCR.SSTS[4:0] bits to make the sub-clock oscillation stabilization wait time become minimum.

Figure 3.12 Sub-Clock Oscillation Setting (2/2)

3.9.8 Sub-Clock Stop Setting

Figure 3.13 shows the Sub-Clock Stop Setting.

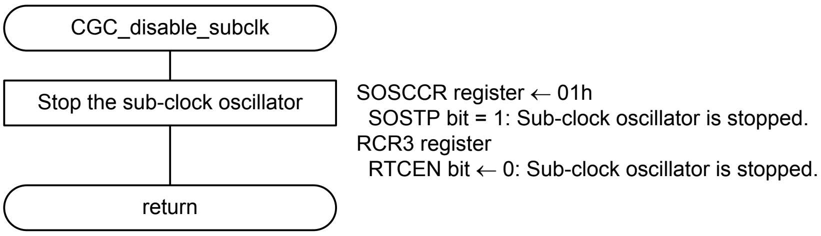
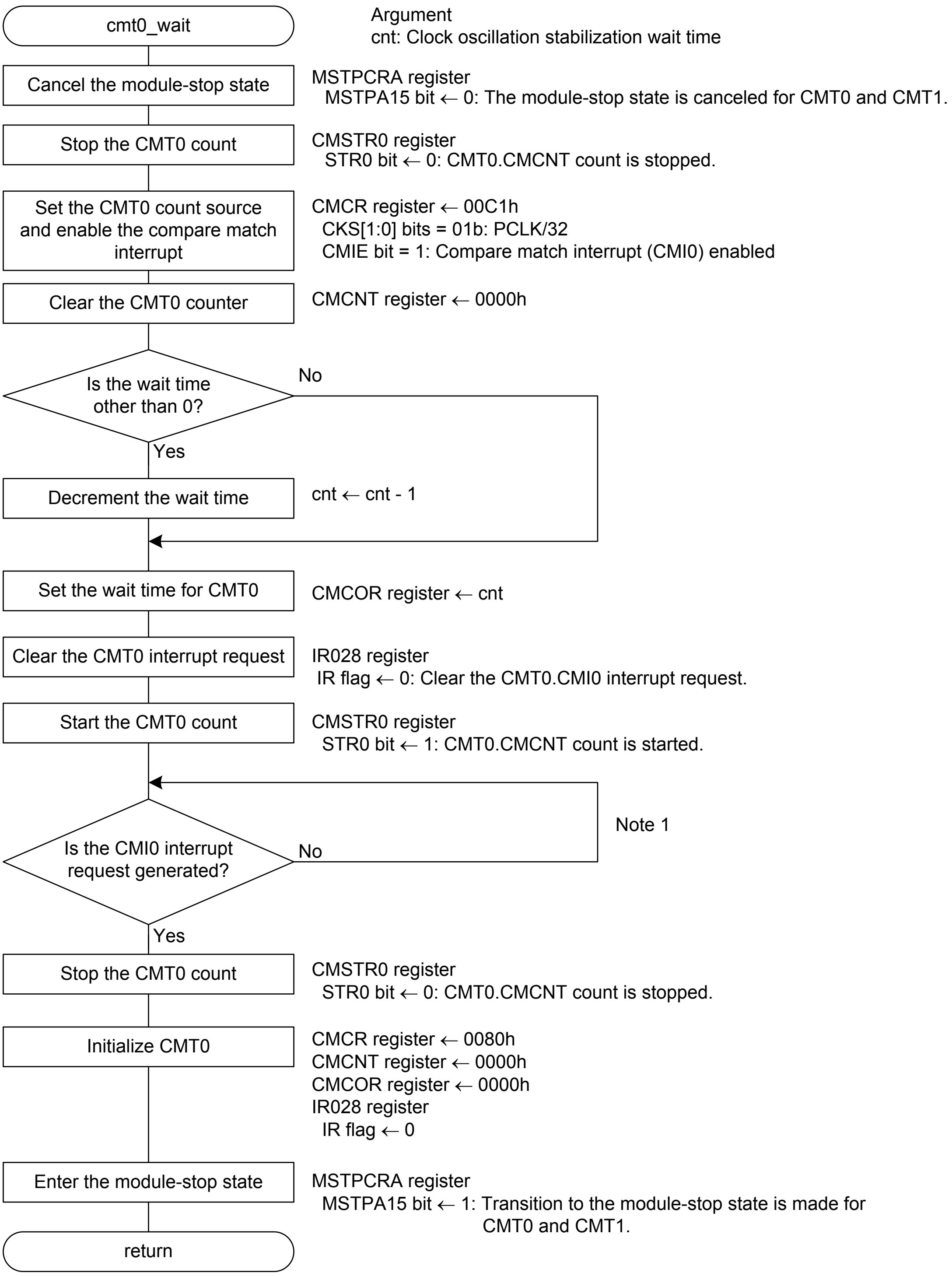


Figure 3.13 Sub-Clock Stop Setting

3.9.9 Wait Processing

Figure 3.14 shows the Wait Processing.



Note:

1. When the counter of the independent watchdog timer (IWDT) is operating, refresh the counter in this loop as required.

Figure 3.14 Wait Processing

4. Applying the RX200 Series Application Note to the RX220 Group

Some of the peripheral functions in the RX220 Group and the RX200 Series are the same. In that case the RX200 Series application notes using these peripheral functions can apply to the RX220 Group by replacing the start-up program with the one accompanying the RX220 Group Initial Setting application note.

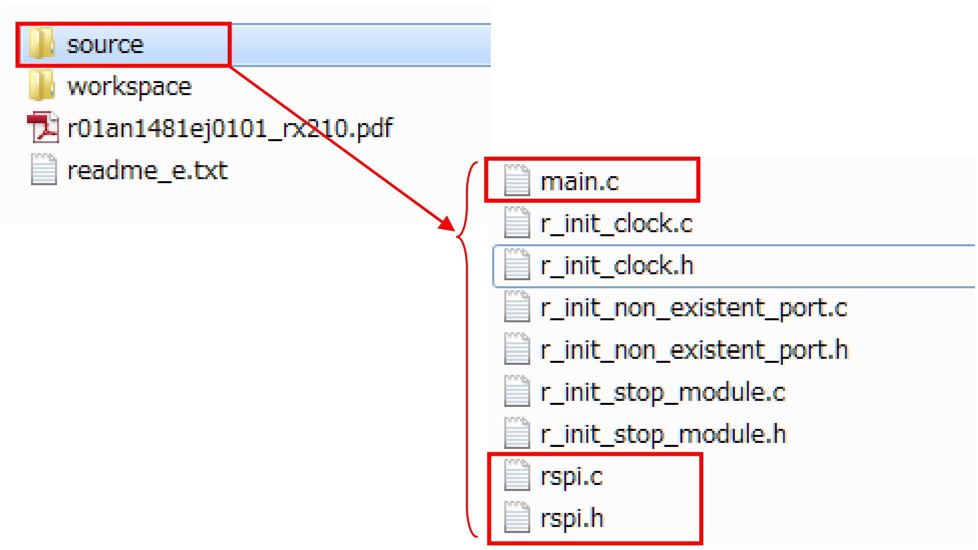
The procedure is explained below using the application note “RX210, RX21A, and RX220 Groups Communication Example Using the RSPI (R01AN1481EJ0101)”.

(1) Open the project of the RX220 Group Initial Setting application note and copy the files shown below from the project of the RX200 Series application note used.

In the “source” folder:

Copy the source and header files *except* r_init_clock, r_init_non_existent_port, and r_init_stop_module.

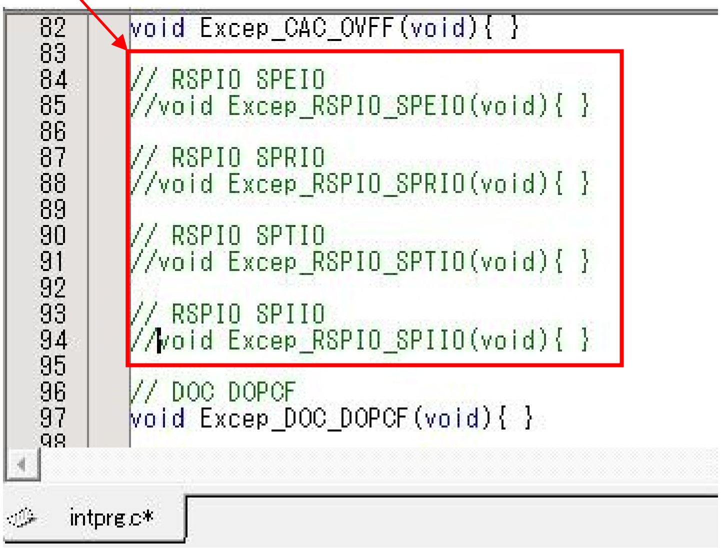
The figure below shows files to be copied. In this example, copy main.c, rspi.c, and rspi.h (source files for the peripheral function used).



(2) Go to the Functions section in the RX200 Series application note and check function names if there is a function name which starts with "Excep_" in the Functions table. If there is, open intprg.c in the rx220_clock_port_r01an1494 folder, and delete (or comment out) the definition of an interrupt function which has the same name as the function with name "Excep_".

Table 5.7 Functions

Function Name	Outline
main	Main processing
port_init	Port initialization
R_INIT_StopModule	Stop processing for active peripheral functions after a reset
R_INIT_NonExistentPort	Nonexistent port initialization
R_INIT_Clock	Clock initialization
peripheral_init	Peripheral function initialization
cb_rspi_slave0_end	Callback function (completion of RSPI transmission to/reception from slave 0)
cb_rspi_slave1_end	Callback function (completion of RSPI transmission to/reception from slave 1)
cb_rspi_rx_error	Callback function (RSPI receive error)
RSPI_Init	User interface function (RSPI initialization)
RSPI_PreTrans	User interface function (RSPI transmit/receive start)
RSPI_GetState	User interface function (obtain RSPI state)
rspi_spti_isr	RSPI transmit interrupt
rspi_spii_isr	RSPI idle interrupt
rspi_spri_isr	RSPI receive interrupt
rspi_spei_isr	RSPI error interrupt
Excep_RSPI0_SPEI0	RSPI0_SPEI0 interrupt handling
Excep_RSPI0_SPRI0	RSPI0_SPRI0 interrupt handling
Excep_RSPI0_SPTI0	RSPI0_SPTI0 interrupt handling
Excep_RSPI0_SPII0	RSPI0_SPII0 interrupt handling



(3) Change the settings in r_init_clock_h according to the clocks used. The settings to be changed are described in this application note.

(4) Review the settings in the application note to be applied to the RX220 Group.

- Check if the same functions are allocated to pins used in the RX220 Group.
- Check if there are any settings of the peripheral function that need to be modified due to the change of the PCLK or ICLK frequency.

  The PCLK frequency is different between the groups. Therefore setting values such as the communication bit rate need to be modified accordingly.

Note: • When applying an application note using the serial communications interface, channel 1 may be connected to the on-chip debugging emulator.

5. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

6. Reference Documents

User's Manual: Hardware
RX220 Group User's Manual: Hardware Rev.1.10 (R01UH0292EJ)
The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools
RX Family C/C++ Compiler Package V.1.01 User's Manual Rev.1.00 (R20UT0570EJ)
The latest version can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics website
http://www.renesas.com

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

REVISION HISTORY	RX220 Group Application Note Initial Setting

Rev.	Date	Description	
		Page	Summary
1.00	Apr. 1, 2013	—	First edition issued
1.10	July 1, 2014	30-32	Added “4. Applying the RX200 Series Application Note to the RX220 Group”.

General Precautions in the Handling of MPU/MCU Products

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

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

Notice

RENESAS

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

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

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

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

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

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

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

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

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

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

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

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

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

Colophon 4.0