RL78/G23

CPU Clock Changing and Standby Settings

Introduction

This application note describes how to change the RL78/G23's CPU clock and set it to standby (changing operation modes).

This application uses switch input to change the CPU clock and the operation mode, while controlling 5 LEDs to indicate the CPU clock status and the operation mode.

Target Device

RL78/G23

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

Contents

1. Specifications

1.1 Outline of Operation

This application describes how to switch the CPU clock and operation mode using switch input, as shown in Figure 1-1.

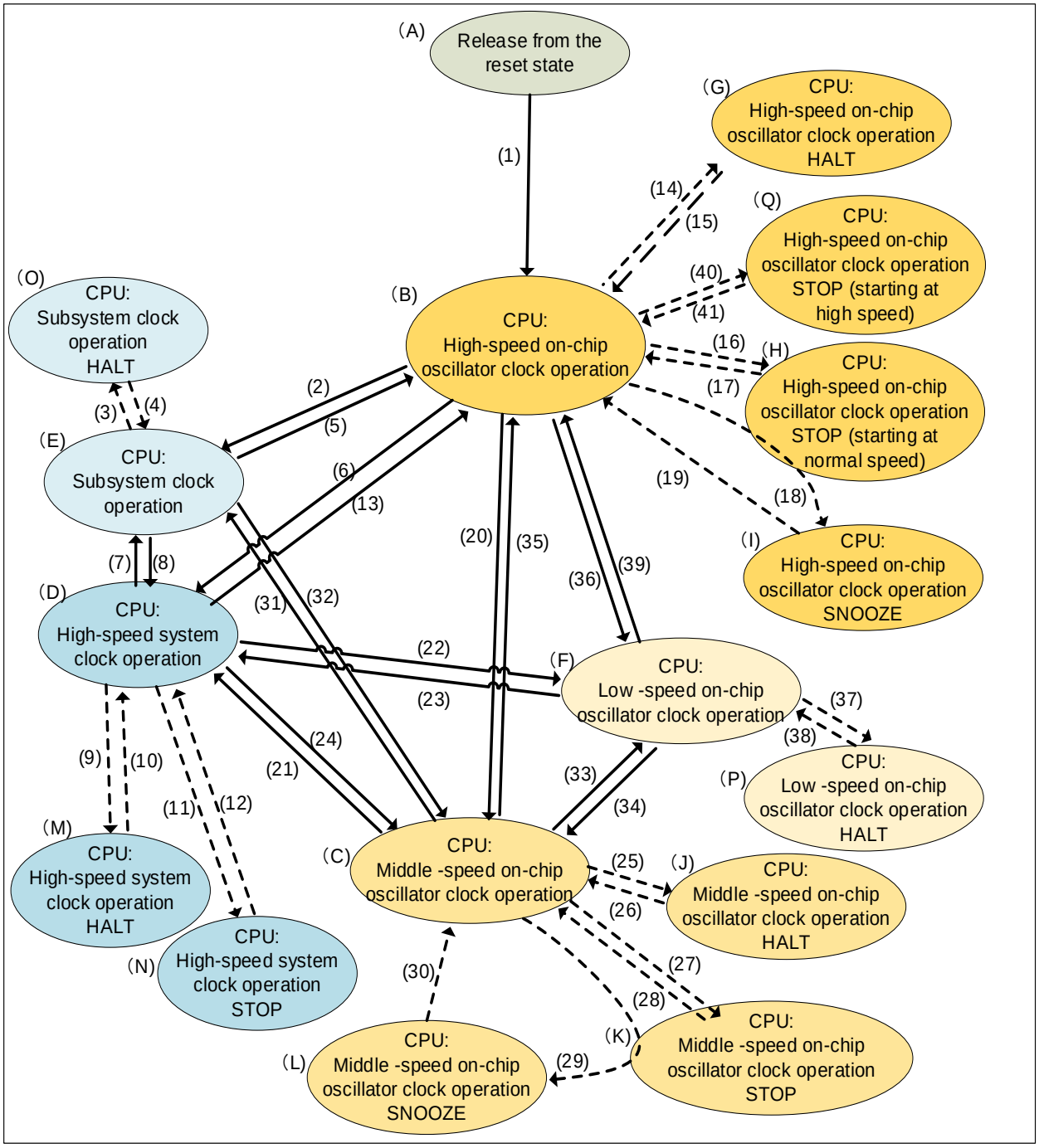
In addition, the application controls five LEDs to indicate the status of the CPU clock and the operation mode.

The Peripheral Functions Used and Their Uses in this application note, Operating Mode Status Transition Diagram, and Operation Modes and Corresponding LED Status are show in Table 1-1, Figure 1-1 and Table 1-2, correspondingly.

Table 1-1 Peripheral Functions Used and Their Uses

Peripheral Function	Use
Port output	Controls the LEDs (LED1 to LED5) connected to the P03, P02, P43, P42, and P77 pins.
External interrupt	Used as a pin input edge detection interrupt (INTP0) by switch input (SW1).
32-bit interval timer	Used as an interval signal detection interrupt (INTITL) of the 32-bit interval timer.
A/D converter	Converts the analog signal input level of the P22 / ANI2 pin to check SNOOZE mode.
Realtime Clock	Uses the fixed-cycle interrupt (INTRTC) as the hardware trigger for the A/D converter.

Figure 1-1 Operating Mode Status Transition Diagram



Note Solid lines indicate CPU clock status transitions and dashed lines indicate CPU operating mode status transitions.

Table 1-2 Operation Modes and Corresponding LED Status

CPU/Peripheral Hardware Clock (f_{CLK})	Operation mode	LED Status				
		LED1	LED2	LED3	LED4	LED5
High-speed on-chip oscillator clock (f_{IH})	Normal operation mode	ON	ON	OFF	OFF	ON
	HALT mode	OFF	ON	OFF	OFF	ON
	SNOOZE mode	ON	OFF	OFF	OFF	ON
	STOP mode (starting at normal speed)	OFF	OFF	OFF	OFF	ON
	STOP mode (starting at high speed)	OFF	OFF	OFF	ON	ON
Middle -speed on-chip oscillator clock (f_{IM})	Normal operation mode	ON	ON	OFF	ON	OFF
	HALT mode	OFF	ON	OFF	ON	OFF
	SNOOZE mode	ON	OFF	OFF	ON	OFF
	STOP mode	OFF	OFF	OFF	ON	OFF
Low-speed on-chip oscillator clock (f_{IL})	Normal operation mode	ON	ON	OFF	ON	ON
	HALT mode	OFF	ON	OFF	ON	ON
High-speed system clock (f_{MX})	Normal operation mode	ON	ON	ON	OFF	ON
	HALT mode	OFF	ON	ON	OFF	ON
	STOP mode	OFF	OFF	ON	OFF	ON
Subsystem clock (f_{SUB})	Normal operation mode	ON	ON	ON	ON	OFF
	HALT mode	OFF	ON	ON	ON	OFF

Note Make sure the current through the pins is 8 mA or less. For limitations on pin current, see the electrical characteristics in the RL78/G23 User's Manual.

1.2 Details of Operation

This application enables the user to change the CPU clock and operating mode using switch input. Steps 1 through 41 in Figure 1-1 below describe how to change the clock and mode.

(1) Perform initial settings for input/output ports.

<Setting conditions>

- P03, P02, P43, P42, and P77 pins: Set as output ports (used for LED control).

(2) Perform initial settings for the clock generation circuit.

<Setting conditions>

- Set the flash operating mode to HS (high-speed main) mode (with the user option byte (000C2H / 040C2H)).
- Set the frequency of the high-speed on-chip oscillator clock to 32 MHz.
- Set the frequency of the middle-speed on-chip oscillator clock to 4 MHz.
- Set operating mode of the subsystem clock pin to XT1 oscillation mode.
- Set oscillation mode of the XT1 oscillation circuit to low power consumption oscillation 1 (default) (by selecting the best oscillation mode for the resonator to be connected).
- Set operating mode of the high-speed system clock pin to X1 oscillation mode.
- Set the frequency of the X1 clock to 20 MHz.
- Set X1 clock oscillation stabilization time to 2^18/fx.
- Select the main system clock (f_{MAIN}) for the CPU/peripheral hardware clock (f_{CLK}).

(3) Perform initial settings for external interrupt processing.

< Setting conditions >

- Set the effective edge of the INTP0 pin to rising edge, and then enable switch input.
- Set interrupt priority level 3.

(4) Perform the interval timer.

< Setting conditions >

- Set the timer in 32-bit counter mode.
- Set the count source to the low-speed peripheral clock (f_{SXP}).
- Set the counter clock to f_{ITL0}.
- Set the interval time to 10ms.
- Enable INTITL interrupts (interrupt priority level 3).

(5) Perform initial settings for the realtime clock (RTC)

< Setting conditions >

- Select the subsystem clock XR (f_{SXR}) at the RTC operation clock.
- Present the time in 12-hour system.
- Disable the RTC1HZ pin output.
- Enable INTRTC interrupts (interrupt priority level 3).
- Enable fixed-cycle interrupt and set their cycle time to 0.5 second.

(6) Perform initial settings for the A/D converter.

< Setting conditions >

- Stop comparator operation.
- As the A/D conversion resolution, use 10 bits.
- As the A/D conversion reference voltages, use V_{DD} for the positive side and V_{SS} for the negative side.
- As the A/D conversion trigger mode, use hardware trigger wait mode.
- As the hardware trigger signal, use realtime clock interrupt signal (INTRTC).
- As the A/D conversion mode, use one-shot conversion mode.
- As the A/D conversion channel selection mode, use select mode.
- Use the P22/ANI2 pin for analog input.
- As the A/D conversion time, use 2816/f_{CLK} in low voltage 1 mode.
- As the A/D conversion result comparison upper limit (ADUL), use 255. As the lower limit (ADLL), use 240.
- The interrupt signal (INTAD) is output the ADLL resister ≦ the ADCRn resister ≦ the ADUL resister.
- Enable A/D conversion end interrupts (INTAD).
- Set interrupt priority level 3.

(7) Each time a rising edge of the P137 / INTP0 pin is detected by pressing the switch, the CPU clock and operating mode change. However, such a change can only be initiated by an A/D conversion completion interrupt in the case of a return from SNOOZE mode (transitions (19) and (30)).

Table 1-3 and Table 1-4 shows the CPU clock and operation mode after the switch is pressed.

Table 1-3 LED status (after the switch is pressed) (1/2)

	CPU clock	Operation mode	LED1	LED2	LED3	LED4	LED5
(1)	High-speed on-chip oscillator clock (f_{IH})	CPU operation mode	ON	ON	OFF	OFF	ON
(2)	Subsystem clock (f_{SUB})	CPU operation mode	ON	ON	ON	ON	OFF
(3)	Subsystem clock (f_{SUB})	HALT mode	OFF	ON	ON	ON	OFF
(4)	Subsystem clock (f_{SUB})	CPU operation mode	ON	ON	ON	ON	OFF
(5)	High-speed on-chip oscillator clock (f_{IH})	CPU operation mode	ON	ON	OFF	OFF	ON
(6)	High-speed system clock (f_{MX})	CPU operation mode	ON	ON	ON	OFF	ON
(7)	Subsystem clock (f_{SUB})	CPU operation mode	ON	ON	ON	ON	OFF
(8)	High-speed system clock (f_{MX})	CPU operation mode	ON	ON	ON	OFF	ON
(9)	High-speed system clock (f_{MX})	HALT mode	OFF	ON	ON	OFF	ON
(10)	High-speed system clock (f_{MX})	CPU operation mode	ON	ON	ON	OFF	ON
(11)	High-speed system clock (f_{MX})	STOP mode	OFF	OFF	ON	OFF	ON
(12)	High-speed system clock (f_{MX})	CPU operation mode	ON	ON	ON	OFF	ON
(13)	High-speed on-chip oscillator clock (f_{IH})	CPU operation mode	ON	ON	OFF	OFF	ON
(14)	High-speed on-chip oscillator clock (f_{IH})	HALT mode	OFF	ON	OFF	OFF	ON
(15)	High-speed on-chip oscillator clock (f_{IH})	CPU operation mode	ON	ON	OFF	OFF	ON
(16)	High-speed on-chip oscillator clock (f_{IH})	STOP mode (starting at normal speed)	OFF	OFF	OFF	OFF	ON
(17)	High-speed on-chip oscillator clock (f_{IH})	CPU operation mode	ON	ON	OFF	OFF	ON
(18)	High-speed on-chip oscillator clock (f_{IH})	SNOOZE mode	ON	OFF	OFF	OFF	ON
(19)	High-speed on-chip oscillator clock (f_{IH})	CPU operation mode	ON	ON	OFF	OFF	ON
(20)	Middle-speed on-chip oscillator clock (f_{IM})	CPU operation mode	ON	ON	OFF	ON	OFF
(21)	High-speed system clock (f_{MX})	CPU operation mode	ON	ON	ON	OFF	ON
(22)	Low-speed on-chip oscillator clock (f_{IL})	CPU operation mode	ON	ON	OFF	ON	ON
(23)	High-speed system clock (f_{MX})	CPU operation mode	ON	ON	ON	OFF	ON
(24)	Middle-speed on-chip oscillator clock (f_{IM})	CPU operation mode	ON	ON	OFF	ON	OFF

Table 1-4 LED status (after the switch is pressed) (2/2)

	CPU clock	Operation mode	LED1	LED2	LED3	LED4	LED5
(25)	Middle-speed on-chip oscillator clock (f_{IM})	HALT mode	OFF	ON	OFF	ON	OFF
(26)	Middle-speed on-chip oscillator clock (f_{IM})	CPU operation mode	ON	ON	OFF	ON	OFF
(27)	Middle-speed on-chip oscillator clock (f_{IM})	STOP mode	OFF	OFF	OFF	ON	OFF
(28)	Middle-speed on-chip oscillator clock (f_{IM})	CPU operation mode	ON	ON	OFF	ON	OFF
(29)	Middle-speed on-chip oscillator clock (f_{IM})	SNOOZE mode	ON	OFF	OFF	ON	OFF
(30)	Middle-speed on-chip oscillator clock (f_{IM})	CPU operation mode	ON	ON	OFF	ON	OFF
(31)	Subsystem clock (f_{SUB})	CPU operation mode	ON	ON	ON	ON	OFF
(32)	Middle-speed on-chip oscillator clock (f_{IM})	CPU operation mode	ON	ON	OFF	ON	OFF
(33)	Low-speed on-chip oscillator clock (f_{IL})	CPU operation mode	ON	ON	OFF	ON	ON
(34)	Middle-speed on-chip oscillator clock (f_{IM})	CPU operation mode	ON	ON	OFF	ON	OFF
(35)	High-speed on-chip oscillator clock (f_{IH})	CPU operation mode	ON	ON	OFF	OFF	ON
(36)	Low-speed on-chip oscillator clock (f_{IL})	CPU operation mode	ON	ON	OFF	ON	ON
(37)	Low-speed on-chip oscillator clock (f_{IL})	HALT mode	OFF	ON	OFF	ON	ON
(38)	Low-speed on-chip oscillator clock (f_{IL})	CPU operation mode	ON	ON	OFF	ON	ON
(39)	High-speed on-chip oscillator clock (f_{IH})	CPU operation mode	ON	ON	OFF	OFF	ON
(40)	High-speed on-chip oscillator clock (f_{IH})	STOP mode (starting at high speed)	OFF	OFF	OFF	ON	ON
(41)	High-speed on-chip oscillator clock (f_{IH})	CPU operation mode	ON	ON	OFF	OFF	ON

After the CPU clock and operation mode have been changed according to steps 1 to 41 above, the falling edge of a signal (switch) input to the P137 / INTP0 pin is detected, all LEDs are turned OFF, and the CPU goes to HALT mode (only RESET input in standby recovery).

Note Refer to the RL78/G23 User's Manual for usage notes concerning this device.

1.3 CPU Clock Changes

This section describes the special function register (SFR) settings required for changing the CPU clock.

- Changing from high-speed on-chip oscillator clock to middle-speed on-chip oscillator clock
- Changing from high-speed on-chip oscillator clock to low-speed on-chip oscillator clock
- Changing from high-speed on-chip oscillator clock to high-speed system clock
- Changing from high-speed on-chip oscillator clock to subsystem clock
- Changing from middle-speed on-chip oscillator clock to high-speed on-chip oscillator clock
- Changing from middle-speed on-chip oscillator clock to low-speed on-chip oscillator clock
- Changing from middle-speed on-chip oscillator clock to high-speed system clock
- Changing from middle-speed on-chip oscillator clock to subsystem clock
- Changing from low-speed on-chip oscillator clock to high-speed on-chip oscillator clock
- Changing from low-speed on-chip oscillator clock to middle-speed on-chip oscillator clock
- Changing from low-speed on-chip oscillator clock to high-speed system clock
- Changing from high-speed system clock to high-speed on-chip oscillator clock
- Changing from high-speed system clock to middle-speed on-chip oscillator clock
- Changing from high-speed system clock to low-speed on-chip oscillator clock
- Changing from high-speed system clock to subsystem clock
- Changing from subsystem clock to high-speed on-chip oscillator clock
- Changing from subsystem clock to middle-speed on-chip oscillator clock
- Changing from subsystem clock to high-speed system clock

1.3.1 Changing from high-speed on-chip oscillator clock to middle-speed on-chip oscillator clock

When changing the CPU clock from the high-speed on-chip oscillator clock to the middle-speed on-chip oscillator clock, start oscillation using the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the timer or other means. After the oscillation stabilization time has passed, set the middle-speed on-chip oscillator clock to f_{CLK} using the system clock control register (CKC).

Check that the main on-chip oscillator clock status has changed to the middle-speed on-chip oscillator clock, and then stop the high-speed on-chip oscillator.

① Set the MIOEN bit in the CSC register to 1 to activate the middle-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	1	×	0	0	0	0	1	0

② Wait until the oscillation of the middle-speed on-chip oscillator becomes stable by software. Count the wait time using the timer function.

③ Clear the MCM0 bit to 0 and set the MCM1 bit to 1 in the CKC register to set the main on-chip oscillator clock for the middle-speed on-chip oscillator clock. When the CSS bit is 1, clear it to 0 because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	0	0	0	0	0	0	0	1

④ Check that the MCS bit is 0 and the MCS1 bit is 1 in the CKC register, and then set the HIOSTOP bit in the CSC register to 1 to stop the high-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	1	×	0	0	0	0	1	1

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.2 Changing from high-speed on-chip oscillator clock to low-speed on-chip oscillator clock

When changing the CPU clock from the high-speed on-chip oscillator clock to the low-speed on-chip oscillator clock, start oscillation using the subsystem clock select register (CKSEL). Next, wait until the oscillation stabilizes using the timer or other means. After the oscillation stabilization time has passed, set the low-speed on-chip oscillator clock to f_{CLK} using the system clock control register (CKC). Check that the CPU/peripheral hardware clock status has changed to the subsystem clock, and then stop the high-speed on-chip oscillator.

① Check that the CLS bit in the CKC register is 0, and then set the SELLOSC bit in the CKSEL register to 1 to activate the low-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CKSEL	0	0	0	0	0	0	0	SELLOSC
	0	0	0	0	0	0	0	1

② Wait until the oscillation of the low-speed on-chip oscillator becomes stable by software. Count the wait time using the timer function.

③ Set the CSS bit in the CKC register to 1 to set the CPU/peripheral hardware clock for the subsystem clock. When the CSS bit is 1, retain that value because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	0	1	0	0	0	0	0	0

④ Check that the CLS bit in the CKC register is 1, and then set the HIOSTOP bit in the CSC register to 1 to stop the high-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	1	1	0	0	0	0	0	1

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.3 Changing from high-speed on-chip oscillator clock to high-speed system clock

When changing the CPU clock from the high-speed on-chip oscillator clock to the high-speed system clock, set the oscillation circuit and start oscillation using the clock operation mode control register (CMC), the oscillation stabilization time select register (OSTS), and the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the oscillation stabilization time counter status register (OSTC).

After the oscillation stabilization time has passed, set the high-speed system clock to f_{CLK} using the system clock control register (CKC).

Check that the main system clock status has changed to the high-speed system clock, and then stop the high-speed on-chip oscillator.

Set the OSCSEL bit in the CMC register to 1, and then set the AMPH bit to 1 (when fx > 10 MHz) to activate the X1 oscillation circuit. For 30-pin to 36-pin products, clear the XTSEL bit to 0. When using an external clock, set the EXCLK and OSCSEL bits to 1. After this register is reset, it can be written only once by the 8-bit memory operation instruction.

[X1 oscillation mode]

	7	6	5	4	3	2	1	0
CMC	EXCLK	OSCSEL	EXCLKS	OSCSELS	XTSEL	AMPHS1	AMPHS0	AMPH
	0	1	×	×	×	×	×	0/1

AMPH bit: clear to 0 when the X1 oscillation clock is 10 MHz or lower.

[External clock input mode]

	7	6	5	4	3	2	1	0
CMC	EXCLK	OSCSEL	EXCLKS	OSCSELS	XTSEL	AMPHS1	AMPHS0	AMPH
	1	1	×	×	×	×	×	×

① Select the oscillation stabilization time of the X1 oscillation circuit using the OSTS register. This setting is not required for external clocks.

Example: Set the following values for a wait of at least 102 µs based on a 10 MHz resonator.

	7	6	5	4	3	2	1	0
OSTS	0	0	0	0	0	OSTS2	OSTS1	OSTS0
	0	0	0	0	0	0	1	0

② Clear the MSTOP bit in the CSC register to 0 to start oscillation of the X1 oscillation circuit. When using an external clock, input the external clock signal and then clear the MSTOP bit to 0.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	0	×	0	0	0	0	0	0

③ Wait for the oscillation of the X1 oscillation circuit to stabilize at the OSTC register. This is not required for external clocks.

Example: Wait until the bits reach the following values for a wait of at least 102 µs based on a 10 MHz resonator

	7	6	5	4	3	2	1	0
OSTC	MOST8	MOST9	MOST10	MOST11	MOST13	MOST15	MOST17	MOST18
	1	1	1	0	0	0	0	0

④ Set the MCM0 bit in the CKC register to 1 to set the high-speed system clock for the main system clock. When the CSS bit is 1, clear it to 0 because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	0	0	0	1	0	0	0	0

⑤ Check that the MCS bit in the CKC register is 1, and then set the HIOSTOP bit in the CSC register to 1 to stop the high-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	0	×	0	0	0	0	0	1

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.4 Changing from high-speed on-chip oscillator clock to subsystem clock

When changing the CPU clock from the high-speed on-chip oscillator clock to the subsystem clock, set the oscillation circuit and start oscillation using the subsystem clock supply mode control register (OSMC), the clock operation mode control register (CMC), and the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the timer or other means. After the oscillation stabilization time has passed, set the subsystem to f_{CLK} using the system clock control register (CKC). Check that the CPU/peripheral hardware clock status has changed to the subsystem clock, and then stop the high-speed on-chip oscillator.

① In this application note, the oscillation stabilization time of the subsystem clock resonator is counted by the 32-bit interval timer. Set the WUTMMCK0 bit to 1 to use the low-speed on-chip oscillator clock as the 32-bit interval timer count clock. Clear the RTCLPC bit to 0 to enable peripheral functions to operate with the subsystem clock in STOP mode or HALT mode (while the CPU is operating with the subsystem clock). To execute the STOP instruction, check that the HIPREC bit is set to 1.

	7	6	5	4	3	2	1	0
OSMC	RTCLPC	0	0	WUTMMCK0	-	-	0	HIPREC
	0	0	0	1	×	×	0	×

② Set the OSCSELS bit in the CMC register to 1 to activate the XT oscillation circuit. For 30-pin to 36-pin products, set the XTSEL bit to 1. To activate the XT oscillation circuit with low power consumption oscillation 1 (default), clear the AMPHS1 and AMPHS0 bits to 0. When using an external clock, set the EXCLKS and OSCSELS bits to 1. After this register is reset, it can be written only once by the 8-bit memory operation instruction.

[XT1 oscillation mode]

	7	6	5	4	3	2	1	0
CMC	EXCLK	OSCSEL	EXCLKS	OSCSELS	XTSEL	AMPHS1	AMPHS0	AMPH
	×	×	0	1	×	0/1	0/1	×

[External clock input mode]

	7	6	5	4	3	2	1	0
CMC	EXCLK	OSCSEL	EXCLKS	OSCSELS	XTSEL	AMPHS1	AMPHS0	AMPH
	×	×	1	1	×	×	×	×

③ Clear the XTSTOP bit in the CSC register to 0 to start oscillation of the XT1 oscillation circuit. When using an external clock, input the external clock signal and then clear the XTSTOP bit to 0.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	1	0	0	0	0	0	0	0

④ Wait until the oscillation of the subsystem clock resonator becomes stable by software. Count the wait time using the timer function. Waiting for oscillation stabilization is not required for external clocks.

⑤ Set the CSS bit in the CKC register to 1 to set the subsystem clock for the CPU/peripheral hardware clock. When the CSS bit is 1, retain that value because the MCM0 and MCM1 bits cannot be changed.

	7	6		5	4	3	2	1	0
CKC	CLS	CSS		MCS	MCM0	0	0	MCS1	MCM1
	0	1		0	0	0	0	0	0

⑥ Check that the CLS bit in the CKC register is 1, and then set the HIOSTOP bit in the CSC register to 1 to stop the high-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	×	0	0	0	0	0	0	1

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.5 Changing from middle-speed on-chip oscillator clock to high-speed on-chip oscillator clock

When changing the CPU clock from the middle-speed on-chip oscillator clock to the high-speed on-chip oscillator clock, start oscillation using the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the timer or other means. After the oscillation stabilization time has passed, set the high-speed on-chip oscillator clock to f_{CLK} using the system clock control register (CKC).

Check that the main on-chip oscillator clock status has changed to the high-speed on-chip oscillator clock, and then stop the middle-speed on-chip oscillator.

① Clear the HIOSTOP bit in the CSC register to 0 to activate the high-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	×	×	0	0	0	0	1	0

② Wait until the oscillation of the high-speed on-chip oscillator becomes stable by software. Count the wait time using the timer function.

③ Clear the MCM0 and MCM1 bits in the CKC register to 0 to set the high-speed on-chip oscillator clock for the main on-chip oscillator clock. When the CSS bit is 1, clear it to 0 because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	0	0	0	0	0	0	1	0

④ Check that the MCS and MCS1 bits in the CKC register are 0, and then clear the MIOEN bit in the CSC register to 0 to stop the middle-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	×	×	0	0	0	0	0	0

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.6 Changing from middle-speed on-chip oscillator clock to low-speed on-chip oscillator clock

When changing the CPU clock from the middle-speed on-chip oscillator clock to the low-speed on-chip oscillator clock, start oscillation using the subsystem clock select register (CKSEL). Next, wait until the oscillation stabilizes using the timer or other means. After the oscillation stabilization time has passed, set the low-speed on-chip oscillator clock to f_{CLK} using the system clock control register (CKC). Check that the CPU/peripheral hardware clock status has changed to the subsystem clock, and then stop the middle-speed on-chip oscillator.

① Check that the CLS bit in the CKC register is 0, and then set the SELLOSC bit in the CKSEL register to 1 to activate the low-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CKSEL	0	0	0	0	0	0	0	SELLOSC
	0	0	0	0	0	0	0	1

② Wait until the oscillation of the low-speed on-chip oscillator becomes stable by software. Count the wait time using the timer function.

③ Set the CSS bit in the CKC register to 1 to set the subsystem clock for the CPU/peripheral hardware clock. When the CSS bit is 1, retain that value because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	0	1	0	0	0	0	1	1

④ Check that the CLS bit in the CKC register is 1, and then clear the MIOEN bit in the CSC register to 0 to stop the middle-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	×	×	0	0	0	0	0	1

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.7 Changing from middle-speed on-chip oscillator clock to high-speed system clock

When changing the CPU clock from the middle-speed on-chip oscillator clock to the high-speed system clock, set the oscillation circuit and start oscillation using the clock operation mode control register (CMC), the oscillation stabilization time select register (OSTS), and the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the oscillation stabilization time counter status register (OSTC).

After the oscillation stabilization time has passed, set the high-speed system clock to f_{CLK} using the system clock control register (CKC).

Check that the main system clock status has changed to the high-speed system clock, and then stop the middle-speed on-chip oscillator.

① Set the OSCSEL bit in the CMC register to 1, and then set the AMPH bit to 1 (when fx > 10 MHz) to activate the X1 oscillation circuit. For 30-pin to 36-pin products, clear the XTSEL bit to 0. When using an external clock, set the EXCLK and OSCSEL bits to 1. After this register is reset, it can be written only once by the 8-bit memory operation instruction.

[X1 oscillation mode]

	7	6	5	4	3	2	1	0
CMC	EXCLK	OSCSEL	EXCLKS	OSCSELS	XTSEL	AMPHS1	AMPHS0	AMPH
	0	1	×	×	×	×	×	0/1

AMPH bit: clear to 0 when the X1 oscillation clock is 10 MHz or lower.

[External clock input mode]

	7	6	5	4	3	2	1	0
CMC	EXCLK	OSCSEL	EXCLKS	OSCSELS	XTSEL	AMPHS1	AMPHS0	AMPH
	1	1	×	×	×	×	×	×

② Select the oscillation stabilization time of the X1 oscillation circuit using the OSTS register. This is not required for external clocks.

Example: Set the following values for a wait of at least 102 μs based on a 10 MHz resonator.

	7	6	5	4	3	2	1	0
OSTS	0	0	0	0	0	OSTS2	OSTS1	OSTS0
	0	0	0	0	0	0	1	0

③ Clear the MSTOP bit in the CSC register to 0 to start oscillation of the X1 oscillation circuit. When using an external clock, input the external clock signal and then clear the MSTOP bit to 0.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	0	×	0	0	0	0	1	1

④ Wait for the oscillation of the X1 oscillation circuit to stabilize at the OSTC register. This is not required for external clocks.

Example: Wait until the bits reach the following values for a wait of at least 102 μs based on a 10 MHz resonator.

	7	6	5	4	3	2	1	0
OSTC	MOST8	MOST9	MOST10	MOST11	MOST13	MOST15	MOST17	MOST18
	1	1	1	0	0	0	0	0

⑤ Set the MCM0 bit in the CKC register to 1 to set the high-speed system clock for the main system clock. When the CSS bit is 1, clear it to 0 because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	0	0	0	1	0	0	1	1

⑥ Check that the MCS bit in the CKC register is 1, and then clear the MIOEN bit in the CSC register to 0 to stop the middle-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	0	×	0	0	0	0	0	1

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.8 Changing from middle-speed on-chip oscillator clock to subsystem clock

When changing the CPU clock from the middle-speed on-chip oscillator clock to the subsystem clock, set the oscillation circuit and start oscillation using the subsystem clock supply mode control register (OSMC), the clock operation mode control register (CMC), and the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the timer or other means. After the oscillation stabilization time has passed, set the subsystem clock to f_{CLK} using the system clock control register (CKC). Check that the CPU/peripheral hardware clock status has changed to the subsystem clock, and then stop the middle-speed on-chip oscillator.

① In this application note, the oscillation stabilization time of the subsystem clock resonator is counted by the 32-bit interval timer. Set the WUTMMCK0 bit to 1 because the low-speed on-chip oscillator clock is used as the 32-bit interval timer count clock. Clear the RTCLPC bit to 0 to enable peripheral functions to operate with the subsystem clock in STOP mode or HALT mode (while the CPU is operating with the subsystem clock). To execute the STOP instruction, check that the HIPREC bit is set to 1.

	7	6	5	4	3	2	1	0
OSMC	RTCLPC	0	0	WUTMMCK0	-	-	0	HIPREC
	0	0	0	1	×	×	0	×

② Set the OSCSELS bit in the CMC register to 1 to activate the XT oscillation circuit. For 30-pin to 36-pin products, set the XTSEL bit to 1. To activate the XT oscillation circuit with low power consumption oscillation 1 (default), clear the AMPHS1 and AMPHS0 bits to 0. When using an external clock, set the EXCLKS and OSCSELS bits to 1. After this register is reset, it can be written only once by the 8-bit memory operation instruction.

[XT1 oscillation mode]

	7	6	5	4	3	2	1	0
CMC	EXCLK	OSCSEL	EXCLKS	OSCSELS	XTSEL	AMPHS1	AMPHS0	AMPH
	×	×	0	1	×	0/1	0/1	×

[External clock input mode]

	7	6	5	4	3	2	1	0
CMC	EXCLK	OSCSEL	EXCLKS	OSCSELS	XTSEL	AMPHS1	AMPHS0	AMPH
	0	0	1	1	×	×	×	×

③ Clear the XTSTOP bit in the CSC register to 0 to start oscillation of the XT1 oscillation circuit. When using an external clock, input the external clock signal and then clear the XTSTOP bit to 0.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	×	0	0	0	0	0	1	0

④ Wait until the oscillation of the subsystem clock resonator becomes stable by software. Count the wait time (oscillation stabilization time) using the timer function. Waiting for oscillation stabilization is not required for external clocks.

⑤ Set the CSS bit in the CKC register to 1 to set the subsystem clock for the CPU/peripheral hardware clock. When the CSS bit is 1, retain that value because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	0	1	0	0	0	0	1	1

⑥ Check that the CLS bit in the CKC register is 1, and then clear the MIOEN bit in the CSC register to 0 to stop the middle-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	×	0	0	0	0	0	0	0

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.9 Changing from low-speed on-chip oscillator clock to high-speed on-chip oscillator clock

When changing the CPU clock from the low-speed on-chip oscillator clock to the high-speed on-chip oscillator clock, start oscillation using the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the timer or other means. After the oscillation stabilization time has passed, set the high-speed on-chip oscillator clock to f_{CLK} using the system clock control register (CKC).

Check that the CPU/peripheral hardware clock status has changed to the main system clock, and then stop the low-speed on-chip oscillator.

① Clear the HIOSTOP bit in the CSC register to 0 to activate the high-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	×	×	0	0	0	0	0	0

② Wait until the oscillation of the high-speed on-chip oscillator becomes stable by software. Count the wait time using the timer function. Clear the MCM0 and MCM1 bits in the CKC register to 0 to set the high-speed on-chip oscillator clock for the main on-chip oscillator clock. When the CSS bit is 1, clear it to 0 because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	1	0	×	0	0	0	×	0

③ Check that the CLS bit in the CKC register is 0, and then clear the SELLOSC bit in the subsystem clock select register (CKSEL) to 0 to stop the low-speed on-chip oscillator. Because the WUTMMCK0 bit in the subsystem clock supply mode control register (OSMC) is 1 in this application note, the low-speed on-chip oscillator does not stop.

	7	6	5	4	3	2	1	0
CKSEL	0	0	0	0	0	0	0	SELLOSC
	0	0	0	0	0	0	0	0

Note Changing the value of the MCM0 bit and MCM1 bit is prohibited while the CPU/peripheral hardware clock is operating with the subsystem clock.

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.10 Changing from low-speed on-chip oscillator clock to middle-speed on-chip oscillator clock

When changing the CPU clock from the low-speed on-chip oscillator clock to the middle-speed on-chip oscillator clock, start oscillation using the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the timer or other means. After the oscillation stabilization time has passed, set the middle-speed on-chip oscillator clock to f_{CLK} using the system clock control register (CKC).

Check that the CPU/peripheral hardware clock status has changed to the main system clock, and then stop the low-speed on-chip oscillator.

① Set the MIOEN bit in the CSC register to 1 to activate the middle-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	1	1	0	0	0	0	1	1

② Wait until the oscillation of the middle-speed on-chip oscillator becomes stable by software. Count the wait time using the timer function.

③ Clear the MCM0 bit to 0 and set the MCM1 bit to 1 in the CKC register to set the middle-speed on-chip oscillator clock for the main on-chip oscillator clock. When the CSS bit is 1, clear it to 0 because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	1	0	×	0	0	0	×	1

④ Check that the CLS bit in the CKC register is 0, and then clear the SELLOSC bit in the subsystem clock select register (CKSEL) to 0 to stop the low-speed on-chip oscillator. Because the WUTMMCK0 bit in the subsystem clock supply mode control register (OSMC) is 1 in this application note, the low-speed on-chip oscillator does not stop.

	7	6	5	4	3	2	1	0
CKSEL	0	0	0	0	0	0	0	SELLOSC
	0	0	0	0	0	0	0	0

Note Changing the value of the MCM0 bit and MCM1 bit is prohibited while the CPU/peripheral hardware clock is operating with the subsystem clock.

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.11 Changing from low-speed on-chip oscillator clock to high-speed system clock

When changing the CPU clock from the low-speed on-chip oscillator clock to the high-speed on-chip oscillator clock, set the oscillation circuit and start oscillation using the clock operation mode control register (CMC), the oscillation stabilization time select register (OSTS), and the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the oscillation stabilization time counter status register (OSTC).

After the oscillation stabilization time has passed, set the high-speed system clock to f_{CLK} using the system clock control register (CKC).

Check that the CPU/peripheral hardware clock status has changed to the main system clock, and then stop the low-speed on-chip oscillator.

① Set the OSCSEL bit in the CMC register to 1, and then set the AMPH bit to 1 (when fx > 10 MHz) to activate the X1 oscillation circuit. For 30-pin to 36-pin products, clear the XTSEL bit to 0. When using an external clock, set the EXCLK and OSCSEL bits to 1. After this register is reset, it can be written only once by the 8-bit memory operation instruction.

[X1 oscillation mode]

	7	6	5	4	3	2	1	0
CMC	EXCLK	OSCSEL	EXCLKS	OSCSELS	XTSEL	AMPHS1	AMPHS0	AMPH
	0	1	×	×	×	×	×	0/1

AMPH bit: clear to 0 when the X1 oscillation clock is 10 MHz or lower.

[External clock input mode]

	7	6	5	4	3	2	1	0
CMC	EXCLK	OSCSEL	EXCLKS	OSCSELS	XTSEL	AMPHS1	AMPHS0	AMPH
	1	1	×	×	×	×	×	×

② Select the oscillation stabilization time of the X1 oscillation circuit in the OSTS register. This is not required for external clocks.

Example: Set the following values for a wait of at least 102 µs based on a 10 MHz resonator.

	7	6	5	4	3	2	1	0
OSTS	0	0	0	0	0	OSTS2	OSTS1	OSTS0
	0	0	0	0	0	0	1	0

③ Clear the MSTOP bit in the CSC register to 0 to start oscillation of the X1 oscillation circuit. When using an external clock, input the external clock signal and then clear the MSTOP bit to 0.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	0	×	0	0	0	0	0	1

④ Wait for the oscillation of the X1 oscillation circuit to stabilize at the OSTC register. This is not required for external clocks.

Example: Wait until the bits reach the following values for a wait of at least 102 µs based on a 10 MHz resonator.

	7	6	5	4	3	2	1	0
OSTC	MOST8	MOST9	MOST10	MOST11	MOST13	MOST15	MOST17	MOST18
	1	1	1	0	0	0	0	0

⑤ Set the MCM0 bit in the CKC register to 1 to set the high-speed system clock for the main system clock. When the CSS bit is 1, clear it to 0 because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	1	0	×	1	0	0	×	×

⑥ Check that the CLS bit in the CKC register is 0, and then clear the SELLOSC bit in the subsystem clock select register (CKSEL) to 0 to stop the low-speed on-chip oscillator. Because the WUTMMCK0 bit in the subsystem clock supply mode control register (OSMC) is 1 in this application note, the low-speed on-chip oscillator does not stop.

	7	6	5	4	3	2	1	0
CKSEL	0	0	0	0	0	0	0	SELLOSC
	0	0	0	0	0	0	0	0

Note Changing the value of the MCM0 bit and MCM1 bit is prohibited while the CPU/peripheral hardware clock is operating with the subsystem clock.

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.12 Changing from high-speed system clock to high-speed on-chip oscillator clock

When changing the CPU clock from the high-speed on-chip oscillator clock to the high-speed system clock, start oscillation using the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the timer or other means. After the oscillation stabilization time has passed, set the high-speed on-chip oscillator clock to f_{CLK} using the system clock control register (CKC).

Check that the main system clock status has changed to the main on-chip oscillator clock, and then deactivate the X1 oscillation circuit.

① Clear the HIOSTOP bit in the CSC register to 0 to activate the high-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	0	×	0	0	0	0	0	0

② Wait until the oscillation of the high-speed on-chip oscillator becomes stable by software. Count the wait time using the timer function.

③ Clear the MCM0 and MCM1 bits in the CKC register to 0 to set the high-speed on-chip oscillator clock for the main on-chip oscillator clock. When the CSS bit is 1, clear it to 0 because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	0	0	1	0	0	0	×	0

④ Check that the MCS and MCS1 bits in the CKC register are 0, and then set the MSTOP bit in the CSC register to 1 to deactivate the X1 oscillation circuit.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	1	×	0	0	0	0	0	0

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.13 Changing from high-speed system clock to middle-speed on-chip oscillator clock

When changing the CPU clock from high-speed system clock to middle-speed on-chip oscillator clock, start oscillation using the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the timer or other means. After the oscillation stabilization time has passed, set the middle-speed on-chip oscillator clock to f_{CLK} using the system clock control register (CKC).

Check that the main system clock status has changed to the main on-chip oscillator clock, and then deactivate the X1 oscillation circuit.

① Set the MIOEN bit in the CSC register to 1 to activate the middle-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	0	×	0	0	0	0	1	1

② Wait until the oscillation of the middle-speed on-chip oscillator becomes stable by software. Count the wait time using the timer function.

③ Clear the MCM0 and MCM1 bits in the CKC register to 0 to set the high-speed on-chip oscillator clock for the main on-chip oscillator clock. When the CSS bit is 1, clear it to 0 because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	0	0	1	0	0	0	×	0

④ Check that the MCS and MCS1 bits in the CKC register are 0, and then set the MSTOP bit in the CSC register to 1 to deactivate the X1 oscillation circuit.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	1	×	0	0	0	0	1	1

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.14 Changing from high-speed system clock to low-speed on-chip oscillator clock

When changing the CPU clock from the high-speed system clock to the low-speed on-chip oscillator clock, start oscillation using the subsystem clock select register (CKSEL). Next, wait until the oscillation stabilizes using the timer or other means. After the oscillation stabilization time has passed, set the low-speed on-chip oscillator clock to f_{CLK} using the system clock control register (CKC).

Check that the CPU/peripheral hardware clock status has changed to the subsystem clock, and then deactivate the X1 oscillation circuit.

① Check that the CLS bit in the CKC register is 0, and then set the SELLOSC bit in the CKSEL register to 1 to activate the low-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CKSEL	0	0	0	0	0	0	0	SELLOSC
	0	0	0	0	0	0	0	1

② Wait until the oscillation of the low-speed on-chip oscillator becomes stable by software. Count the wait time using the timer function.

③ Set the CSS bit in the CKC register to 1 to set the subsystem clock for the CPU/peripheral hardware clock. When the CSS bit is 1, retain that value because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	0	1	1	1	0	0	×	×

④ Check that the CLS bit in the CKC register is 1, and then set the MSTOP bit in the CSC register to 1 to deactivate the X1 oscillation circuit.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	1	1	0	0	0	0	0	1

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.15 Changing from high-speed system clock to subsystem clock

When changing the CPU clock from the high-speed system clock to the subsystem clock, set the oscillation circuit and start oscillation using the subsystem clock supply mode control register (OSMC), the clock operation mode control register (CMC), and the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the timer or other means. After the oscillation stabilization time has passed, set the subsystem clock to f_{CLK} using the system clock control register (CKC). Check that the CPU/peripheral hardware clock status has changed to the subsystem clock, and then deactivate the X1 oscillation circuit.

For 30-pin to 36-pin products, the X1 and XT1 pins and the X2 and XT2 pins are used together respectively. Therefore, changing from the high-speed system clock to the subsystem clock is enabled only for 40-pin to 128-pin products.

① In this application note, the oscillation stabilization time of the subsystem clock resonator is counted by the 32-bit interval timer. Set the WUTMMCK0 bit to 1 because the low-speed on-chip oscillator clock is used as the 32-bit interval timer count clock. Clear the RTCLPC bit to 0 to enable peripheral functions to operate with the subsystem clock in STOP mode or HALT mode (while the CPU is operating with the subsystem clock). To execute the STOP instruction, check that the HIPREC bit is set to 1.

	7	6	5	4	3	2	1	0
OSMC	RTCLPC	0	0	WUTMMCK0	-	-	0	HIPREC
	0	0	0	1	×	×	0	×

② Clear the XTSTOP bit in the CSC register to 0 to start oscillation of the XT1 oscillation circuit. When using an external clock, input the external clock signal and then clear the XTSTOP bit to 0.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	0	0	0	0	0	0	0	1

③ Wait until the oscillation of the subsystem clock resonator becomes stable by software. Count the wait time (oscillation stabilization time) using the timer function. Waiting for oscillation stabilization is not required for external clocks.

④ Set the CSS bit in the CKC register to 1 to set the subsystem clock for the CPU/peripheral hardware clock. When the CSS bit is 1, retain it because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	0	1	1	1	0	0	×	×

⑤ Check that the CLS bit in the CKC register is 1, and then set the MSTOP bit in the CSC register to 1 to deactivate the X1 oscillation circuit.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	1	1	0	0	0	0	0	1

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.16 Changing from subsystem clock to high-speed on-chip oscillator clock

When changing the CPU clock from the subsystem clock to the high-speed on-chip oscillator clock, start oscillation using the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the timer or other means. After the oscillation stabilization time has passed, set the high-speed on-chip oscillator clock to f_{CLK} using the system clock control register (CKC).

Check that the CPU/peripheral hardware clock status has changed to the main system clock, and then deactivate the XT1 oscillation circuit.

① Clear the HIOSTOP bit in the CSC register to 0 to activate the high-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	×	0	0	0	0	0	1	0

② Wait until the oscillation of the high-speed on-chip oscillator becomes stable by software. Count the wait time using the timer function. Clear the MCM0 and MCM1 bits in the CKC register to 0 to set the high-speed on-chip oscillator clock for the main on-chip oscillator clock. When the CSS bit is 1, clear it to 0 because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	1	0	×	0	0	0	×	0

③ Check that the CLS bit in the CKC register is 0, and then set the XTSTOP bit in the CSC register to 1 to deactivate the XT1 oscillation circuit.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	×	1	0	0	0	0	1	0

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.17 Changing from subsystem clock to middle-speed on-chip oscillator clock

When changing the CPU clock from the subsystem clock to the middle-speed on-chip oscillator clock, start oscillation using the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the timer or other means. After the oscillation stabilization time has passed, set the middle-speed on-chip oscillator clock to f_{CLK} using the system clock control register (CKC).

Check that the CPU/peripheral hardware clock status has changed to the main system clock, and then deactivate the XT1 oscillation circuit.

① Set the MIOEN bit in the CSC register to 1 to activate the middle-speed on-chip oscillator.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	×	0	0	0	0	0	1	1

② Wait until the oscillation of the middle-speed on-chip oscillator becomes stable by software. Count the wait time using the timer function.

③ Clear the MCM0 bit to 0 and set the MCM1 bit to 1 in the CKC register to set the middle-speed on-chip oscillator clock for the main on-chip oscillator clock. When the CSS bit is 1, clear it to 0 because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	1	0	×	0	0	0	×	1

④ Check that the CLS bit in the CKC register is 0, and then set the XTSTOP bit in the CSC register to 1 to deactivate the XT1 oscillation circuit.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	×	1	0	0	0	0	1	1

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

1.3.18 Changing from subsystem clock to high-speed system clock

When changing the CPU clock from the subsystem clock to the high-speed system clock, set the oscillation circuit and start oscillation using the clock operation mode control register (CMC), the oscillation stabilization time select register (OSTS), and the clock operation status control register (CSC). Next, wait until the oscillation stabilizes using the oscillation stabilization time counter status register (OSTC).

After the oscillation stabilization time has passed, set the high-speed system clock to f_{CLK} using the system clock control register (CKC).

Check that the CPU/peripheral hardware clock status has changed to the main system clock, and then deactivate the XT1 oscillation circuit.

For 30-pin to 36-pin products, the X1 and XT1 pins and the X2 and XT2 pins are used together respectively. Therefore, changing from subsystem clock to high-speed system clock is enabled only for 40-pin to 128-pin products.

① Select the oscillation stabilization time of the X1 oscillation circuit in the OSTS register. This is not required for external clocks.

Example: Set the following values for a wait of at least 102 µs based on a 10 MHz resonator.

	7	6	5	4	3	2	1	0
OSTS	0	0	0	0	0	OSTS2	OSTS1	OSTS0
	0	0	0	0	0	0	1	0

② Clear the MSTOP bit in the CSC register to 0 to start oscillation of the X1 oscillation circuit. When using an external clock, input the external clock signal and then clear the MSTOP bit to 0.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	1	0	0	0	0	0	×	×

③ Wait for the oscillation of the X1 oscillation circuit to stabilize at the OSTC register. This is not required for external clocks.

Example: Wait until the bits reach the following values for a wait of at least 102 µs based on a 10 MHz resonator.

	7	6	5	4	3	2	1	0
OSTC	MOST8	MOST9	MOST10	MOST11	MOST13	MOST15	MOST17	MOST18
	1	1	1	0	0	0	0	0

④ Set the MCM0 bit in the CKC register to 1 to set the high-speed system clock for the main system clock. When the CSS bit is 1, clear it to 0 because the MCM0 and MCM1 bits cannot be changed.

	7	6	5	4	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	0	0	MCS1	MCM1
	0	0	1	1	0	0	×	×

⑤ Check that the CLS bit in the CKC register is 0, and then set the XTSTOP bit in the CSC register to 1 to deactivate the XT1 oscillation circuit.

	7	6	5	4	3	2	1	0
CSC	MSTOP	XTSTOP	0	0	0	0	MIOEN	HIOSTOP
	0	1	0	0	0	0	×	×

Note Changing the value of the MCM0 bit and MCM1 bit is prohibited while the CPU/peripheral hardware clock is operating with the subsystem clock.

Register setting values:
x: unused bit; blank space; unchanged bit; -: reserved bits or unassigned bit

2. Operation Confirmation Conditions

The operation of the sample code provided with this application note has been tested under the following conditions.

Table 2-1 Operation Confirmation Conditions

Item	Description
MCU used	RL78/G23 (R7F100GLG)
Board used	RL78/G23-64p Fast Prototyping Board (RTK7RLG230CLG000BJ)
Operating frequency	・ High-speed on-chip oscillator clock (f_{IH}): 32 MHz ・ Middle-speed on-chip oscillator clock (f_{IM}): 4 MHz ・ Low-speed on-chip oscillator clock (f_{IL}): 32.768 kHz ・ High-speed system clock (X1clock (f_X)): 20 MHz ・ Subsystem clock (XT1clock (f_{XT})): 32.768 kHz ・ CPU/peripheral hardware clock: 32 MHz, 20 MHz, 4 MHz, 32.768 kHz
Operating voltage	5.0 V (can be operated at 2.0 V to 5.5 V) LVD0 detection voltage: Reset mode At rising edge TYP. 1.90 V (1.84 V to 1.95 V) At falling edge TYP. 1.86 V (1.80 V to 1.91 V)
Integrated development environment (CS+)	CS+ for CC E8.09.00 from Renesas Electronics Corp.
C compiler (CS+)	CC-RL V1.12.00 from Renesas Electronics Corp.
Integrated development environment (e2studio)	e2studio V2023-04 (23.4.0) from Renesas Electronics Corp.
C compiler (e2studio)	CC-RL V1.12.00 from Renesas Electronics Corp.
Integrated development environment (IAR)	IAR Embedded Workbench for Renesas RL78 V4.21.2 from IAR Systems Corp.
C compiler (IAR)	IAR C/C++ Compiler for Renesas RL78 V4.21.2.2420 from IAR Systems Corp.
Smart configurator (SC)	V1.6.0 from Renesas Electronics Corp.
Board support package (BSP)	V1.60 from Renesas Electronics Corp.

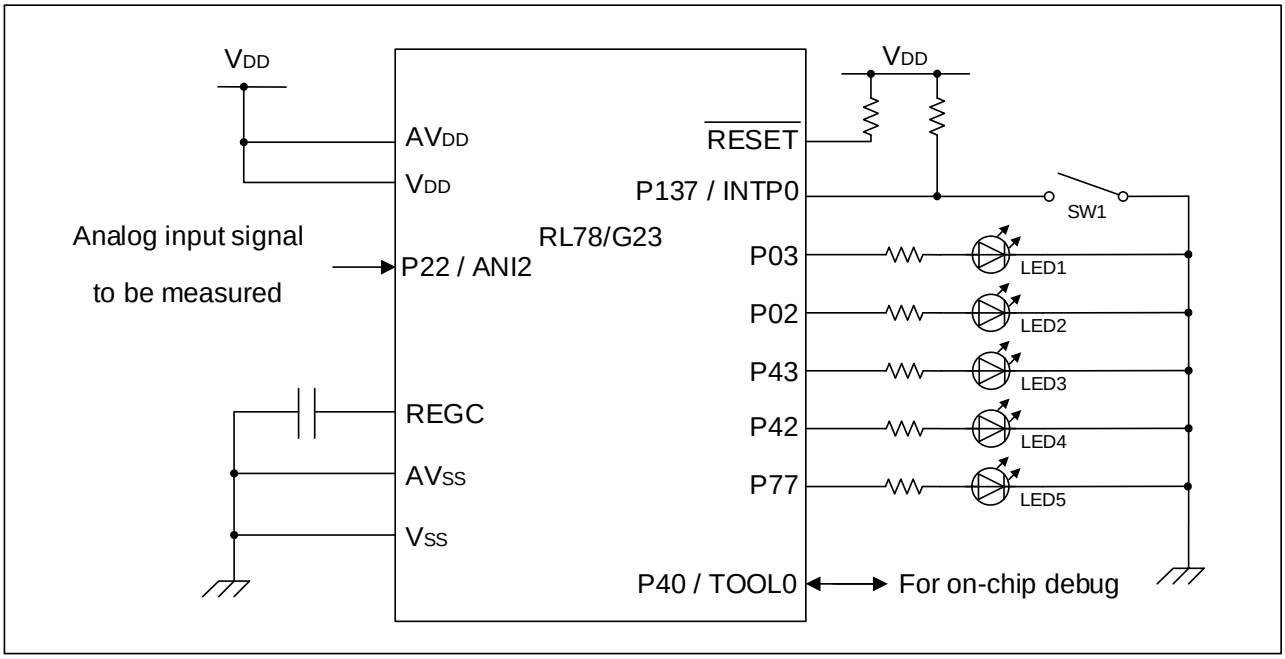
Note CPU/peripheral hardware clock settings are changed in the application.

3. Hardware Descriptions

3.1 Example of Hardware Configuration

Figure 3-1 shows an example of the hardware configuration used in the application note.

Figure 3-1 Hardware Configuration



Note 1. This simplified circuit diagram was created to show an overview of connections only. When actually designing your circuit, make sure the design includes appropriate pin handling and meets electrical characteristic requirements (connect each input-only port to V_{DD} or V_{SS} through a resistor).

Note 2. Connect any pins whose name begins with EV_{SS} to V_{SS}, and any pins whose name begins with EV_{DD} to V_{DD}, respectively.

Note 3. V_{DD} must not be lower than the reset release voltage (V_{LVD0}) that is specified for the LVD0.

.

3.2 List of Pins to be Used

Table 3-1 lists the pins to be used and their functions.

Table 3-1 Pins to be Used and Their Functions

Pin name	I/O	Function
P137 / INTP0	Input	Switch (SW1) input port
P22 / ANI2	Input	A/D converter analog input port
P03	Output	LED (LED0) control port
P02	Output	LED (LED1) control port
P43	Output	LED (LED2) control port
P42	Output	LED (LED3) control port
P77	Output	LED (LED4) control port

Caution In this application note, only the used pins are processed. When actually designing your circuit, make sure the design includes sufficient pin processing and meets electrical characteristic requirements.

4. Software Explanation

4.1 Setting of Option Byte

Table 4-1 shows the option byte settings.

Table 4-1 Option Byte Settings

Address	Setting Value	Contents
000C0H / 040C0H	11101111B	Disables the watchdog timer. (Counting stopped after reset)
000C1H / 040C1H	11111110B	LVD0 detection voltage: reset mode At rising edge TYP. 1.90V (1.84 V to 1.95 V) At falling edge TYP. 1.86V (1.80 V to 1.91 V)
000C2H / 040C2H	11101000B	HS mode, High-speed on-chip oscillator clock (f_{IH}): 32 MHz
000C3H / 040C3H	10000100B	Enables on-chip debugging

4.2 List of Constants

Table 4-2 and Table 4-3 lists the constants that are used in the sample code.

Table 4-2 Constants Used in Sample Code (1/2)

Constant Name	Setting Value	Description
R_LED_STATE_A	0b00000U	LED lighting pattern in state (A)
R_LED_STATE_B	0b10011U	LED lighting pattern in state (B)
R_LED_STATE_C	0b01011U	LED lighting pattern in state (C)
R_LED_STATE_D	0b10111U	LED lighting pattern in state (D)
R_LED_STATE_E	0b01111U	LED lighting pattern in state (E)
R_LED_STATE_F	0b11011U	LED lighting pattern in state (F)
R_LED_STATE_G	0b10010U	LED lighting pattern in state (G)
R_LED_STATE_H	0b10000U	LED lighting pattern in state (H)
R_LED_STATE_I	0b10001U	LED lighting pattern in state (I)
R_LED_STATE_J	0b01010U	LED lighting pattern in state (J)
R_LED_STATE_K	0b01000U	LED lighting pattern in state (K)
R_LED_STATE_L	0b01001U	LED lighting pattern in state (L)
R_LED_STATE_M	0b10110U	LED lighting pattern in state (M)
R_LED_STATE_N	0b10100U	LED lighting pattern in state (N)
R_LED_STATE_O	0b01110U	LED lighting pattern in state (O)
R_LED_STATE_P	0b11010U	LED lighting pattern in state (P)
R_LED_STATE_Q	0b11000U	LED lighting pattern in state (Q)
R_LED_STATE_END	0b11111U	LED lighting pattern at the end of state transitions

Table 4-3 Constants Used in Sample Code (1/2)

Constant Name	Setting Value	Description
R_fCLK_fMAIN_fOCO_HIGH	0x01U	Clock mode: High-speed on-chip oscillator clock is used for f_{CLK}.
R_fCLK_fMAIN_fOCO_MID	0x02U	Clock mode: Middle-speed on-chip oscillator clock is used for f_{CLK}.
R_fCLK_fMAIN_fOCO_LOW	0x03U	Clock mode: Low-speed on-chip oscillator clock is used for f_{CLK}.
R_fCLK_fMAIN_fMX	0x08U	Clock mode: High-speed system clock is used for f_{CLK}.
R_fCLK_fSUB	0x10U	Clock mode: Subsystem clock is used for f_{CLK}.
R_NORMAL_AWAKEN	0U	Normal activation of high-speed on-chip oscillator clock.
R_NORMAL_AWAKEN	1U	High-speed activation of high-speed on-chip oscillator clock
R_PORT_NEGATIVE_MASK	0U	Negative bit mask for port output control.
R_PORT_POSITIVE_MASK	1U	Positive bit mask for port output control.
R_MSEC_PER_COUNT	33UL	32-bit interval timer count value per millisecond
R_WAIT_OSTC_COUNT	0xD0U	Value for waiting for OSTC until X1 oscillation stabilizes
R_WAIT_MSEC_HIGH_OCO	4UL	Time until oscillation of high-speed on-chip oscillator clock stabilizes (millisecond)
R_WAIT_MSEC_MID_OCO	4UL	Time until oscillation of middle-speed on-chip oscillator clock stabilizes (millisecond)
R_WAIT_MSEC_LOW_OCO	4UL	Time until oscillation of low-speed on-chip oscillator clock stabilizes (millisecond)
R_WAIT_MSEC_XT1	4UL	Time until oscillation of subsystem clock is stabilized (millisecond)
R_WAIT_MSEC_CHATTERING	5UL	Time until switch chattering stabilizes (millisecond)
_0_INTERRUPT_FLAG_OFF	0U	Interrupt flag is cleared.
_1_INTERRUPT_FLAG_ON	1U	Interrupt flag is on.

4.3 List of Variables

Table 4-4 lists global variables.

Table 4-4 Global Variables

Type	Variable Name	Description	Function Used
uint8_t	g_intp0	INTP0 interrupt flag When an interrupt is generated, this variable becomes “_1_INTERRUPT_FLAG_ ON”. In other cases, this variable becomes “_0_INTERRUPT_FLAG_OFF”.	R_Config_INTC_Create_UserInit, r_Config_INTC_intp0_interrupt, r_intp0_clear_flag, r_intp0_is_flag_on

4.4 List of Functions (Subroutines)

Table 4-5 and Table 4-6 shows a list of functions (Subroutines).

Table 4-5 Functions (Subroutines) (1/2)

Function name	Outline
r_AtoB	Status transition processing from (A) to (B). State transition (1).
r_BtoE	Status transition processing from (B) to (E). State transition (2).
r_EtoO	Status transition processing from (E) to (O). State transition (3).
r_OtoE	Status transition processing from (O) to (E). State transition (4).
r_EtoB	Status transition processing from (E) to (B). State transition (5).
r_BtoD	Status transition processing from (B) to (D). State transition (6).
r_DtoE	Status transition processing from (D) to (E). State transition (7).
r_EtoD	Status transition processing from (E) to (D). State transition (8).
r_DtoM	Status transition processing from (D) to (M). State transition (9).
r_MtoD	Status transition processing from (M) to (D). State transition (10).
r_DtoN	Status transition processing from (D) to (N). State transition (11).
r_NtoD	Status transition processing from (N) to (D). State transition (12).
r_DtoB	Status transition processing from (D) to (B). State transition (13).
r_BtoG	Status transition processing from (B) to (G). State transition (14).
r_GtoB	Status transition processing from (G) to (B). State transition (15).
r_BtoH	Status transition processing from (B) to (H). State transition (16).
r_HtoB	Status transition processing from (H) to (B). State transition (17).
r_BtoI	Status transition processing from (B) to (I). State transition (18).
r_ItoB	Status transition processing from (I) to (B). State transition (19).
r_BtoC	Status transition processing from (B) to (C). State transition (20).
r_CtoD	Status transition processing from (C) to (D). State transition (21).
r_DtoF	Status transition processing from (D) to (F). State transition (22).
r_FtoD	Status transition processing from (F) to (D). State transition (23).
r_DtoC	Status transition processing from (D) to (C). State transition (24).
r_CtoJ	Status transition processing from (C) to (J). State transition (25).
r_JtoC	Status transition processing from (J) to (C). State transition (26).
r_CtoK	Status transition processing from (C) to (K). State transition (27).
r_KtoC	Status transition processing from (K) to (C). State transition (28).
r_CtoL	Status transition processing from (C) to (L). State transition (29).
r_LtoC	Status transition processing from (L) to (C). State transition (30).
r_CtoE	Status transition processing from (C) to (E). State transition (31).
r_EtoC	Status transition processing from (E) to (C). State transition (32).
r_CtoF	Status transition processing from (C) to (F). State transition (33).
r_FtoC	Status transition processing from (F) to (C). State transition (34).
r_CtoB	Status transition processing from (C) to (B). State transition (35).
r_BtoF	Status transition processing from (B) to (F). State transition (36).
r_FtoP	Status transition processing from (F) to (P). State transition (37).
r_PtoF	Status transition processing from (P) to (F). State transition (38).
r_FtoB	Status transition processing from (F) to (B). State transition (39).
r_BtoQ	Status transition processing from (B) to (Q). State transition (40).
r_QtoB	Status transition processing from (Q) to (B). State transition (41).

Table 4-6 Functions (2/2)

Function name	Outline
r_end	End processing of status transition
r_nop_loop	Repeats the NOP instruction until an external interrupt is generated.
r_halt	Repeats the NOP instruction until an external interrupt is generated after the HALT instruction is executed.
r_stop	Repeats the NOP instruction until an external interrupt is generated after the STOP instruction is executed.
r_snooze	Activates the A/D converter in SNOOZE mode and execute the STOP instruction. After return due to an A/D conversion completion interrupt, repeats the NOP instruction until an external interrupt is generated.
r_start_high_oco	Starts high-speed on-chip oscillator clock.
r_start_mid_oco	Starts middle-speed on-chip oscillator clock.
r_start_low_oco	Starts low-speed on-chip oscillator clock.
r_start_x1	Starts high-speed system clock.
r_start_xt1	Starts subsystem clock.
r_stop_high_oco	Stops high-speed on-chip oscillator clock.
r_stop_mid_oco	Stops middle-speed on-chip oscillator clock.
r_stop_low_oco	Stops low-speed on-chip oscillator clock.
r_stop_x1	Stops high-speed system clock.
r_stop_xt1	Stops subsystem clock.
r_set_fclk	Performs register settings for switching clock.
r_set_high_speed_awakening	Sets startup mode of the high-speed on-chip oscillator clock.
r_wait_OSTC	Waits until oscillation of the X1 clock stabilizes.
r_user_init	Performs the entire user defined initialization procedure.
R_Config_INTC_Create_UserInit	Performs user defined initialization for the interrupt controller.
r_Config_INTC_intp0_interrupt	Sets the INTP0 interrupt flag (external interrupt processing).
r_intp0_clear_flag	Clears the INTP0 interrupt flag.
r_intp0_Is_flag_on	Determines whether the INTP0 interrupt flag is set.
r_ITL000_ITL001_ITL012_ITL013_ set_compare_value	Sets the compare value for the 32-bit interval timer.
r_ITL000_ITL001_ITL012_ITL013_ wait_interval_timer	Waits until the specified time has elapsed using the 32-bit interval timer.
r_port_set_LED	Sets the five LEDs ON/OFF.
r_port_bitmask_to_port	Toggles the port output value (0/1) based on the bit string.

4.5 Specification of Functions (subroutine)

The function specifications of the sample code are shown below.

[Function Name] r_AtoB

Outline	Status transition processing from (A) to (B)
Declaration	void r_AtoB(void)
Description	Controls LEDs and wait for an external interrupt while in high-speed on-chip oscillator clock operating status.
Argument	None
Return Value	None
Notes	None

[Function Name] r_BtoE

Outline	Status transition processing from (B) to (E)
Declaration	void r_BtoE(void)
Description	Changes the CPU clock from the high-speed on-chip oscillator clock to the subsystem clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_EtoO

Outline	Status transition processing from (E) to (O)
Declaration	void r_EtoO(void)
Description	Changes the CPU clock from subsystem clock state to subsystem clock HALT state. After controlling LEDs, executes the HALT instruction and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_OtoE

Outline	Status transition processing from (O) to (E)
Declaration	void r_OtoE(void)
Description	Controls LEDs and waits for an external interrupt in subsystem clock operating status.
Argument	None
Return Value	None
Notes	None

[Function Name] r_EtoB

Outline	Status transition processing from (E) to (B)
Declaration	void r_EtoB(void)
Description	Changes the CPU clock from subsystem to high-speed on-chip oscillator clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_BtoD

Outline	Status transition processing from (B) to (D)
Declaration	void r_BtoD(void)
Description	Changes the CPU clock from high-speed on-chip oscillator clock to high-speed system clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_DtoE

Outline	Status transition processing from (D) to (E)
Declaration	void r_DtoE(void)
Description	Changes the CPU clock from high-speed system clock to subsystem clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_EtoD

Outline	Status transition processing from (E) to (D)
Declaration	void r_EtoD(void)
Description	Changes the CPU clock from subsystem clock to high-speed system clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_DtoM

Outline	Status transition processing from (D) to (M)
Declaration	void r_DtoM(void)
Description	Changes the CPU clock from high-speed system clock state to high-speed system clock HALT state. After controlling LEDs, executes the HALT instruction and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_MtoD

Outline	Status transition processing from (M) to (D)
Declaration	void r_MtoD(void)
Description	Controls LEDs and waits for an external interrupt in high-speed system clock operating status.
Argument	None
Return Value	None
Notes	None

[Function Name] r_DtoN

Outline	Status transition processing from (D) to (N)
Declaration	void r_DtoN(void)
Description	Changes the CPU clock from high-speed system clock state to high-speed system clock STOP state. After controlling LEDs, executes the STOP instruction and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_NtoD

Outline	Status transition processing from (N) to (D)
Declaration	void r_NtoD(void)
Description	Controls LEDs and waits for an external interrupt in high-speed system clock operating status.
Argument	None
Return Value	None
Notes	None

[Function Name] r_DtoB

Outline	Status transition processing from (D) to (B)
Declaration	void r_DtoB(void)
Description	Changes the CPU clock from high-speed system clock to high-speed on-chip oscillator clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_BtoG

Outline	Status transition processing from (B) to (G)
Declaration	void r_BtoG(void)
Description	Changes the CPU clock from high-speed on-chip oscillator clock state to high-speed on-chip oscillator clock HALT state. After controlling LEDs, executes the HALT instruction and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_GtoB

Outline	Status transition processing from (G) to (B)
Declaration	void r_GtoB(void)
Description	Controls LEDs and waits for an external interrupt in high-speed on-chip oscillator clock operating status.
Argument	None
Return Value	None
Notes	None

[Function Name] r_BtoH

Outline	Status transition processing from (B) to (H)
Declaration	void r_BtoH(void)
Description	Changes the CPU clock from high-speed on-chip oscillator clock state to high-speed on-chip oscillator clock STOP state. After controlling LEDs, executes the STOP instruction and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_HtoB

Outline	Status transition processing from (H) to (B)
Declaration	void r_HtoB(void)
Description	Controls LEDs and wait for an external interrupt in high-speed on-chip oscillator clock operating status.
Argument	None
Return Value	None
Notes	None

[Function Name] r_BtoI

Outline	Status transition processing from (B) to (I)
Declaration	void r_BtoI(void)
Description	Changes the CPU clock from high-speed on-chip oscillator clock state to high-speed on-chip oscillator clock SNOOZE state. After controlling LEDs, activates the A/D converter with SNOOZE mode, executes the STOP instruction, and then waits for an A/D conversion completion interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_ItoB

Outline	Status transition processing from (I) to (B)
Declaration	void r_ItoB(void)
Description	Controls LEDs and waits for an external interrupt in high-speed on-chip oscillator clock operating status.
Argument	None
Return Value	None
Notes	None

[Function Name] r_BtoC

Outline	Status transition processing from (B) to (C)
Declaration	void r_BtoC(void)
Description	Changes the CPU clock from high-speed on-chip oscillator clock to middle-speed on-chip oscillator clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_CtoD

Outline	Status transition processing from (C) to (D)
Declaration	void r_CtoD(void)
Description	Changes the CPU clock from middle-speed on-chip oscillator clock to high-speed system clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_DtoF

Outline	Status transition processing from (D) to (F)
Declaration	void r_DtoF(void)
Description	Changes the CPU clock from high-speed system clock to low-speed on-chip oscillator clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_FtoD

Outline	Status transition processing from (F) to (D)
Declaration	void r_FtoD(void)
Description	Changes the CPU clock from low-speed on-chip oscillator clock to high-speed system clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_DtoC

Outline	Status transition processing from (D) to (C)
Declaration	void r_DtoC(void)
Description	Changes the CPU clock from high-speed system clock to middle-speed on-chip oscillator clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_CtoJ

Outline	Status transition processing from (C) to (J)
Declaration	void r_CtoJ(void)
Description	Changes the CPU clock from middle-speed on-chip oscillator clock state to middle-speed on-chip oscillator clock HALT state. After controlling LEDs, executes the HALT instruction and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_JtoC

Outline	Status transition processing from (J) to (C)
Declaration	void r_JtoC(void)
Description	Controls LEDs and waits for an external interrupt in middle-speed on-chip oscillator clock operating status.
Argument	None
Return Value	None
Notes	None

[Function Name] r_CtoK

Outline	Status transition processing from (C) to (K)
Declaration	void r_CtoK(void)
Description	Changes the CPU clock from middle-speed on-chip oscillator clock state to middle-speed on-chip oscillator clock STOP state. After controlling LEDs, executes the STOP instruction and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_KtoC

Outline	Status transition processing from (K) to (C)
Declaration	void r_KtoC(void)
Description	Controls LEDs and waits for an external interrupt in middle-speed on-chip oscillator clock operating status.
Argument	None
Return Value	None
Notes	None

[Function Name] r_CtoL

Outline	Status transition processing from (C) to (L)
Declaration	void r_CtoL(void)
Description	Changes the CPU clock from middle-speed on-chip oscillator clock state to middle-speed on-chip oscillator clock SNOOZE state. After controlling LEDs, activates the A/D converter with SNOOZE mode, executes the STOP instruction, and then waits for an A/D conversion completion interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_LtoC

Outline	Status transition processing from (L) to (C)
Declaration	void r_LtoC(void)
Description	Controls LEDs and waits for an external interrupt in middle-speed on-chip oscillator clock operating status.
Argument	None
Return Value	None
Notes	None

[Function Name] r_CtoE

Outline	Status transition processing from (C) to (E)
Declaration	void r_CtoE(void)
Description	Changes the CPU clock from middle-speed on-chip oscillator clock to subsystem clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_EtoC

Outline	Status transition processing from (E) to (C)
Declaration	void r_EtoC(void)
Description	Changes the CPU clock from subsystem clock to middle-speed on-chip oscillator clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_CtoF

Outline	Status transition processing from (C) to (F)
Declaration	void r_CtoF(void)
Description	Changes the CPU clock from middle-speed on-chip oscillator clock to low-speed on-chip oscillator clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_FtoC

Outline	Status transition processing from (F) to (C)
Declaration	void r_FtoC(void)
Description	Changes the CPU clock from low-speed on-chip oscillator clock to middle-speed on-chip oscillator clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_CtoB

Outline	Status transition processing from (C) to (B)
Declaration	void r_CtoB(void)
Description	Changes the CPU clock from middle-speed on-chip oscillator clock to high-speed on-chip oscillator clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_BtoF

Outline	Status transition processing from (B) to (F)
Declaration	void r_BtoF(void)
Description	Changes the CPU clock from high-speed on-chip oscillator clock to low-speed on-chip oscillator clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_FtoP

Outline	Status transition processing from (F) to (P)
Declaration	void r_FtoP(void)
Description	Changes the CPU clock from low-speed on-chip oscillator clock state to low-speed on-chip oscillator clock HALT state. After controlling LEDs, executes the HALT instruction and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_PtoF

Outline	Status transition processing from (P) to (F)
Declaration	void r_PtoF(void)
Description	Controls LEDs and waits for an external interrupt in low-speed on-chip oscillator clock operating status.
Argument	None
Return Value	None
Notes	None

[Function Name] r_FtoB

Outline	Status transition processing from (F) to (B)
Declaration	void r_FtoB(void)
Description	Changes the CPU clock from low-speed on-chip oscillator clock to high-speed on-chip oscillator clock. After changing the clock, controls LEDs and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_BtoQ

Outline	Status transition processing from (B) to (Q)
Declaration	void r_BtoQ(void)
Description	Changes the CPU clock from high-speed on-chip oscillator clock state to high-speed on-chip oscillator clock STOP state (high-speed startup). Sets high-speed startup for the high-speed on-chip oscillator. After controlling LEDs, executes the STOP instruction and waits for an external interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_QtoB

Outline	Status transition processing from (Q) to (B)
Declaration	void r_QtoB(void)
Description	After resetting high-speed startup for the high-speed on-chip oscillator, controls LEDs and waits for an external interrupt in high-speed on-chip oscillator clock operating status
Argument	None
Return Value	None
Notes	None

[Function Name] r_end

Outline	End processing of status transition.
Declaration	void r_end (void)
Description	Controls LEDs in transition end status.
Argument	None
Return Value	None
Notes	None

[Function Name] r_nop_loop

Outline	Waiting for an external interrupt by executing the NOP instruction repeatedly
Declaration	void r_nop_loop (void)
Description	Executes the NOP instruction repeatedly until an external interrupt is generated.
Argument	None
Return Value	None
Notes	None

[Function Name] r_halt

Outline	Transition to HALT state and waiting for return
Declaration	void r_halt (void)
Description	Executes the HALT instruction repeatedly until an external interrupt is generated.
Argument	None
Return Value	None
Notes	None

[Function Name] r_stop

Outline	Transition to STOP state and waiting for return
Declaration	void r_stop (void)
Description	Executes the STOP instruction repeatedly until an external interrupt is generated.
Argument	None
Return Value	None
Notes	None

[Function Name] r_snooze

Outline	Transition to SNOOZE state and wait for return
Declaration	void r_snooze (void)
Description	Activates the A/D converter in SNOOZE mode, executes the STOP instruction, and then waits for an A/D conversion completion interrupt.
Argument	None
Return Value	None
Notes	None

[Function Name] r_start_high_oco

Outline	Start high-speed on-chip oscillator clock.
Declaration	void r_start_high_oco (void)
Description	Set high-speed on-chip oscillator clock start and wait until the oscillation stabilizes.
Argument	None
Return Value	None
Notes	None

[Function Name] r_start_mid_oco

Outline	Start middle-speed on-chip oscillator clock.
Declaration	void r_start_mid_oco (void)
Description	Set middle-speed on-chip oscillator clock start and wait until the oscillation stabilizes.
Argument	None
Return Value	None
Notes	None

[Function Name] r_start_low_oco

Outline	Start low-speed on-chip oscillator clock.
Declaration	void r_start_ow_oco (void)
Description	Set low-speed on-chip oscillator clock start and wait until the oscillation stabilizes.
Argument	None
Return Value	None
Notes	None

[Function Name] r_start_x1

Outline	Start high-speed system clock.
Declaration	void r_start_x1 (void)
Description	Set high-speed system clock start and wait until the oscillation stabilizes.
Argument	None
Return Value	None
Notes	None

[Function Name] r_start_xt1

Outline	Start subsystem clock.
Declaration	void r_start_xt1 (void)
Description	Set subsystem clock start and wait until the oscillation stabilizes.
Argument	None
Return Value	None
Notes	None

[Function Name] r_stop_high_oco

Outline	Stop high-speed on-chip oscillator clock.
Declaration	void r_ stop_high_oco (void)
Description	Check high-speed on-chip oscillator clock stopping conditions, and then make stop setting.
Argument	None
Return Value	None
Notes	None

[Function Name] r_stop_mid_oco

Outline	Stop middle-speed on-chip oscillator clock.
Declaration	void r_ stop_mid_oco (void)
Description	Check middle-speed on-chip oscillator clock stopping conditions, and then make stop setting.
Argument	None
Return Value	None
Notes	None

[Function Name] r_stop_low_oco

Outline	Stop low-speed on-chip oscillator clock.
Declaration	void r_ stop_low_oco (void)
Description	Check low-speed on-chip oscillator clock stopping conditions, and then make stop setting.
Argument	None
Return Value	None
Notes	None

[Function Name] r_stop_low_x1

Outline	Stop high-speed system clock.
Declaration	void r_ stop_low_x1 (void)
Description	Check high-speed system clock stopping conditions, and then make stop setting.
Argument	None
Return Value	None
Notes	None

[Function Name] r_stop_low_xt1

Outline	Stop subsystem clock.
Declaration	void r_ stop_low_xt1 (void)
Description	Check subsystem clock stopping conditions, and then make stop setting.
Argument	None
Return Value	None
Notes	None

[Function Name] r_set_fclk

Outline	Clock change setting	
Declaration	void r_ stop_set_fclk (r_fclk_mode_t mode)	
Description	Set the CKC register for changing to the specified clock operating mode.	
Argument	r_fclk_mode_t mode	[Clock operating mode]
Return Value	None	
Notes	None	

[Function Name] r_set_high_speed_awakening

Outline	Setting whether to activate high-speed on-chip oscillator	
Declaration	void r_set_high_speed_awakening (r_awaken_mode_t flag)	
Description	Set whether to enable high-speed startup for the high-speed on-chip oscillator.	
Argument	r_awaken_mode_t flag	[1: High-speed startup enabled 0: High-speed startup disabled]
Return Value	None	
Notes	None	

[Function Name] r_wait_OSTC

Outline	Waiting for X1 oscillation stabilization
Declaration	void r_wait_OSTC (void)
Description	Check the OSTC register value and wait for X1 oscillation stabilization.
Argument	None
Return Value	None
Notes	None

[Function Name] R_Config_INTC_Create_UserInit

Outline	User definition initialization processing for interrupt controller
Declaration	void R_Config_INTC_Create_UserInit (void)
Description	User definitions are provided in the interrupt controller initialization processing. In this application note, the INTP0 interrupt flag is initialized.
Argument	None
Return Value	None
Notes	None

[Function Name] r_Config_INTC_intp0_interrupt

Outline	External interrupt (INTP0) processing.
Declaration	void r_Config_INTC_intp0_interrupt (void)
Description	Processing for INTP0 interrupt is provided. In this application note, the INTP0 interrupt flag is set.
Argument	None
Return Value	None
Notes	None

[Function Name] r_intp0_clear_flag

Outline	INTP0 interrupt flag clear
Declaration	void r_intp0_clear_flag (void)
Description	Clear the INTP0 interrupt flag.
Argument	None
Return Value	None
Notes	None

[Function Name] r_intp0_clear_flag_on

Outline	Checking whether the INTP0 interrupt flag is set
Declaration	void r_intp0_clear_flag_on (void)
Description	Return information as to whether the INTP0 interrupt flag is set.
Argument	None
Return Value	1: INTP0 interrupt flag is set. 0: INTP0 interrupt flag is not set.
Notes	None

[Function Name] r_ITL000_ITL001_ITL012_ITL013_set_compare_value

Outline	Compare value setting for 32-bit interval timer	
Declaration	void r_ITL000_ITL001_ITL012_ITL013_set_compare_value (uint32_t count)	
Description	Set the compare value of the 32-bit interval timer.	
Argument	uint32_t count	[Compare value to be set (The lower 16 bits are set to ITLCMP00 and the upper 16 bits are set to ITLCMP01.)]
Return Value	None	
Notes	None	

[Function Name] r_ITL000_ITL001_ITL012_ITL013_wait_interval_timer

Outline	Wait using 32-bit interval timer	
Declaration	void r_ITL000_ITL001_ITL012_ITL013_wait_interval_timer (uint32_t msec)	
Description	Wait for a predetermined time using the 32-bit interval timer.	
Argument	uint32_t msec	[Specify wait time (millisecond)]
Return Value	None	
Notes	None	

[Function Name] r_port_set_LED

Outline	ON/OFF control for five LEDs	
Declaration	void r_port_set_LED (uint8_t state)	
Description	Control ON/OFF condition of the five LEDs connected to port outputs.	
Argument	uint8_t state	[LED ON/OFF state corresponding to LED1 to LED5 from the lower bits (0: OFF 1: ON)]
Return Value	None	
Notes	None	

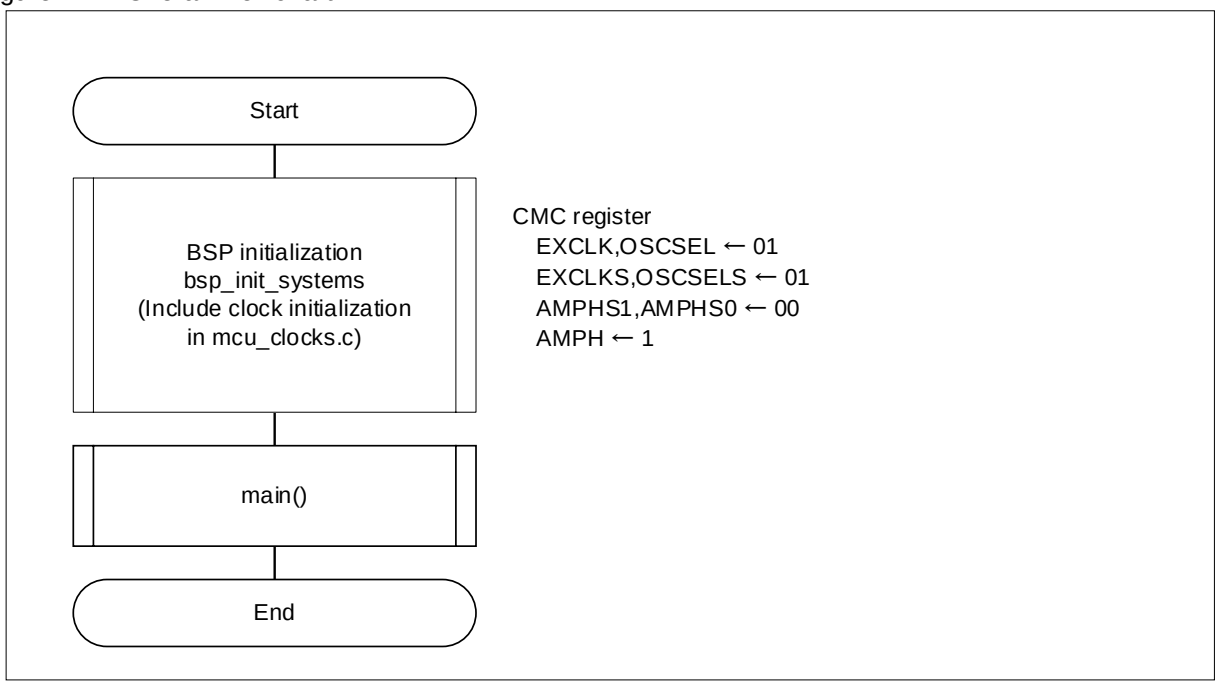
[Function Name] r_port_bitmask_to_port

Outline	Port output value (0 or 1) setting	
Declaration	void r_port_bitmask_to_port (uint8_t mask, volatile unsigned char __near * port, r_config_port_mask_t flag)	
Description	Set whether to specify 0 or 1 as a port output value.	
Argument	uint8_t mask	[Port bit mask]
	volatile unsigned char __near * port	[Port address]
	r_config_port_mask_t flag	[Determine whether to apply real bit mask or inverted bit mask to output value.]
Return Value	None	
Notes	None	

4.6 Flowcharts

Figure 4-1 shows the entire flow for this application note.

Figure 4-1 Overall Flowchart



4.6.1 Main Processing

Figure 4-2 and Figure 4-3 show flowcharts of the Main Processing.

Figure 4-2 Main Processing (1/2)

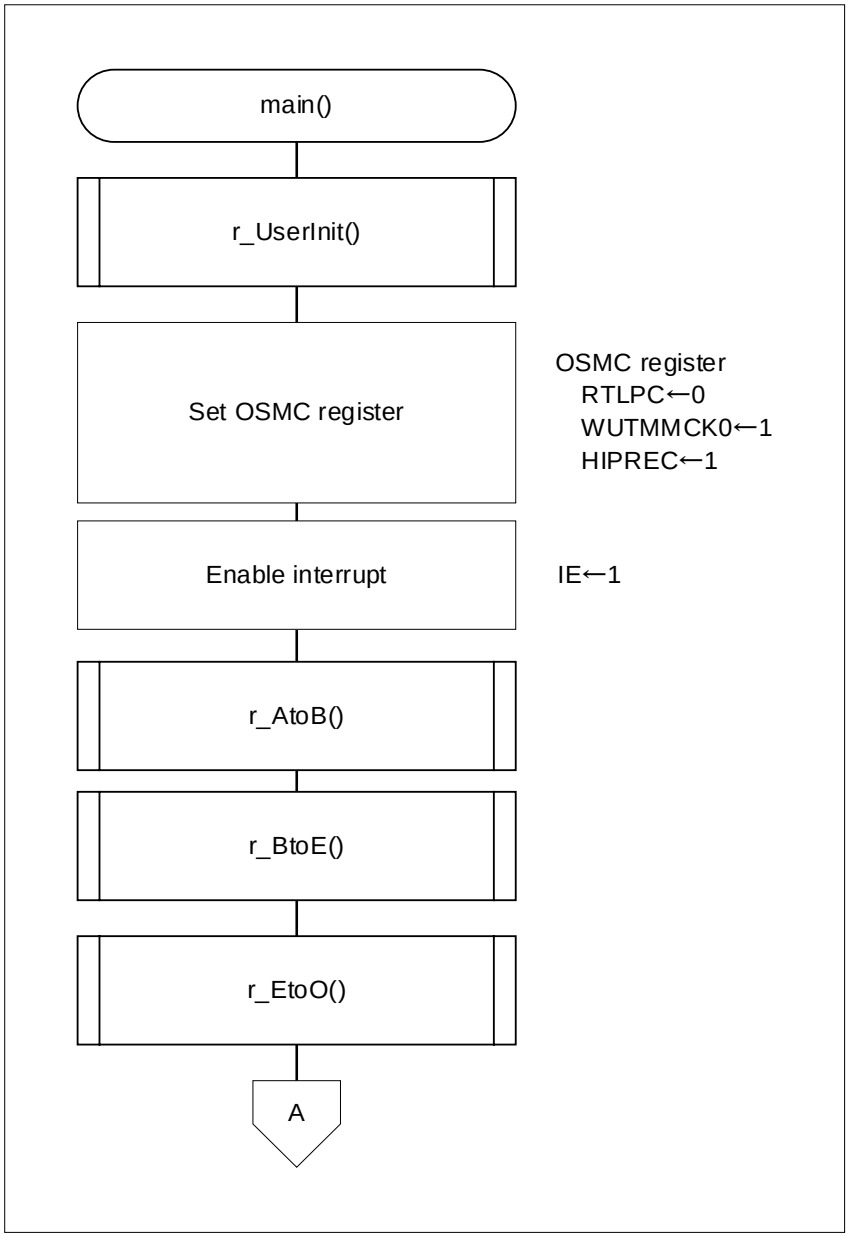
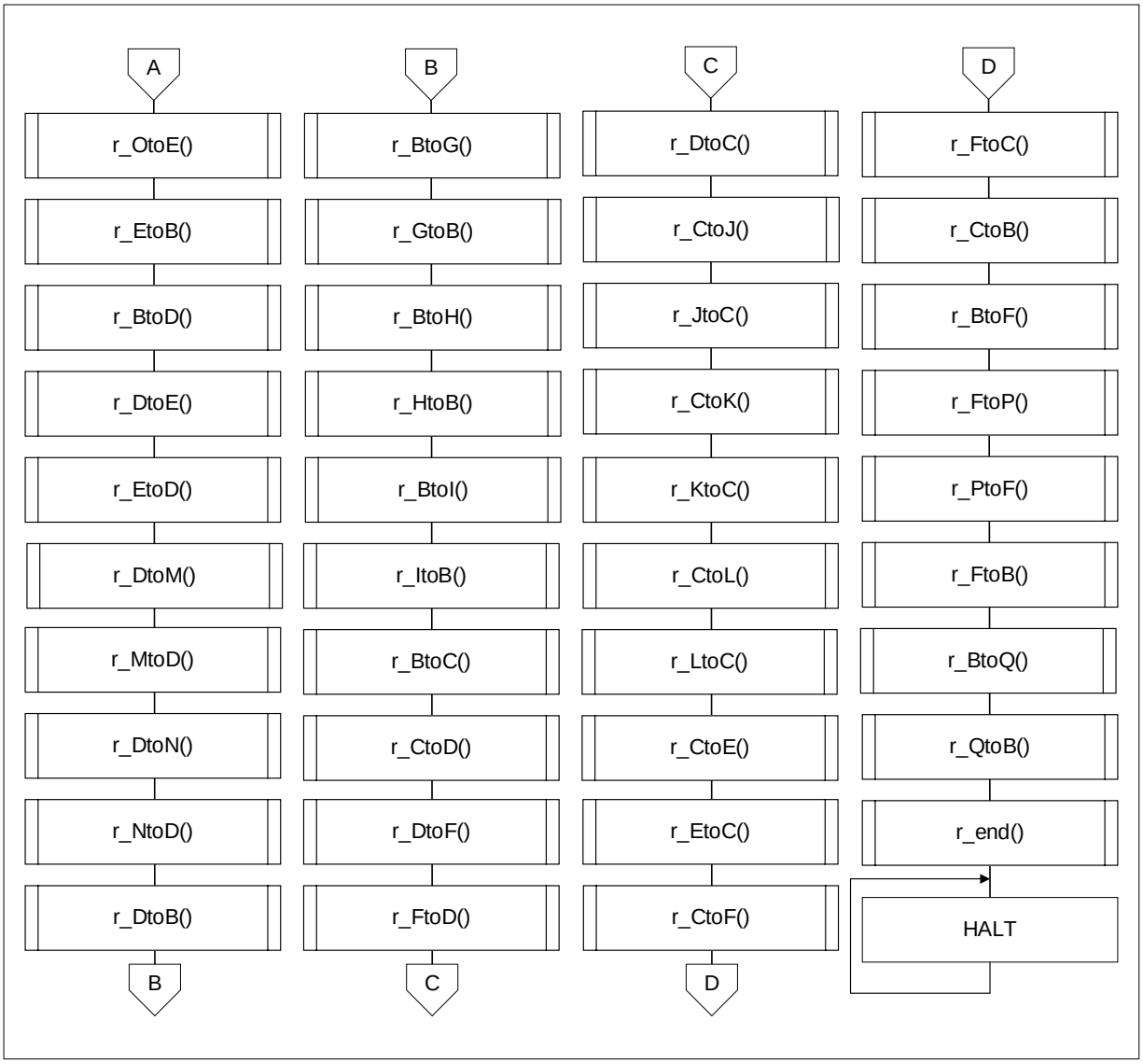


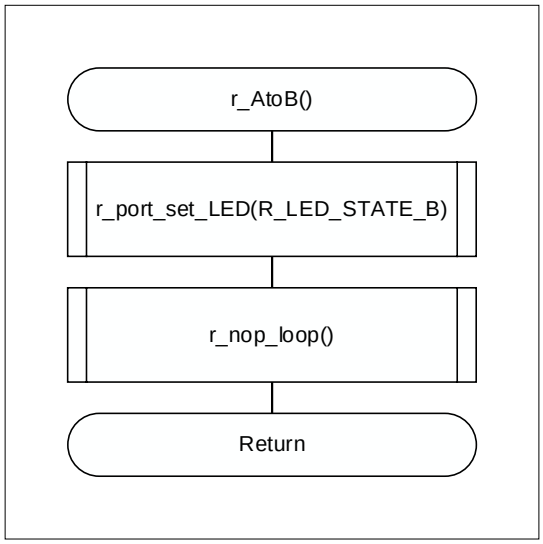
Figure 4-3 Main Processing (2/2)



4.6.2 Status Transition AtoB

Figure 4-4 shows the flowchart of the status transition AtoB.

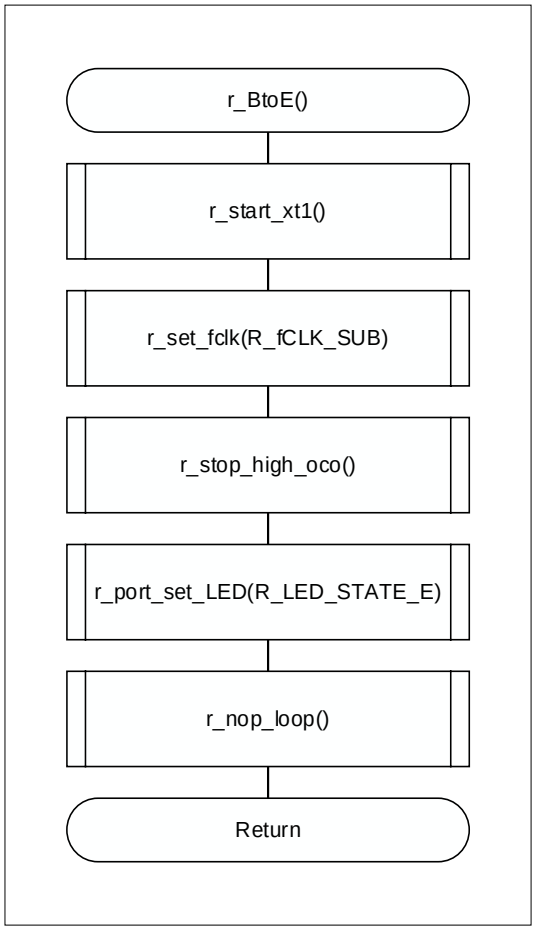
Figure 4-4 Status Transition AtoB



4.6.3 Status Transition BtoE

Figure 4-5 shows the flowchart of the status transition BtoE.

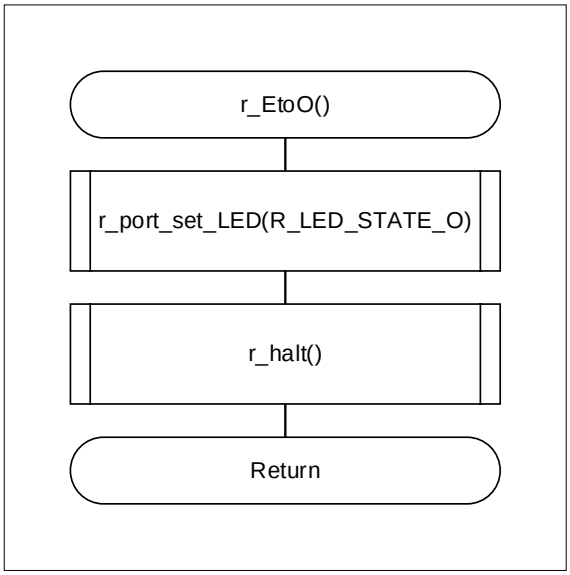
Figure 4-5 Status Transition BtoE



4.6.4 Status Transition EtoO

Figure 4-6 shows the flowchart of the status transition EtoO.

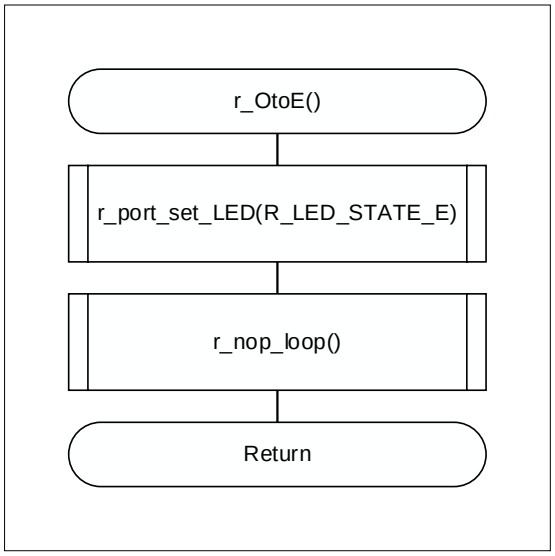
Figure 4-6 Status Transition EtoO



4.6.5 Status Transition OtoE

Figure 4-7 shows the flowchart of the status transition OtoE.

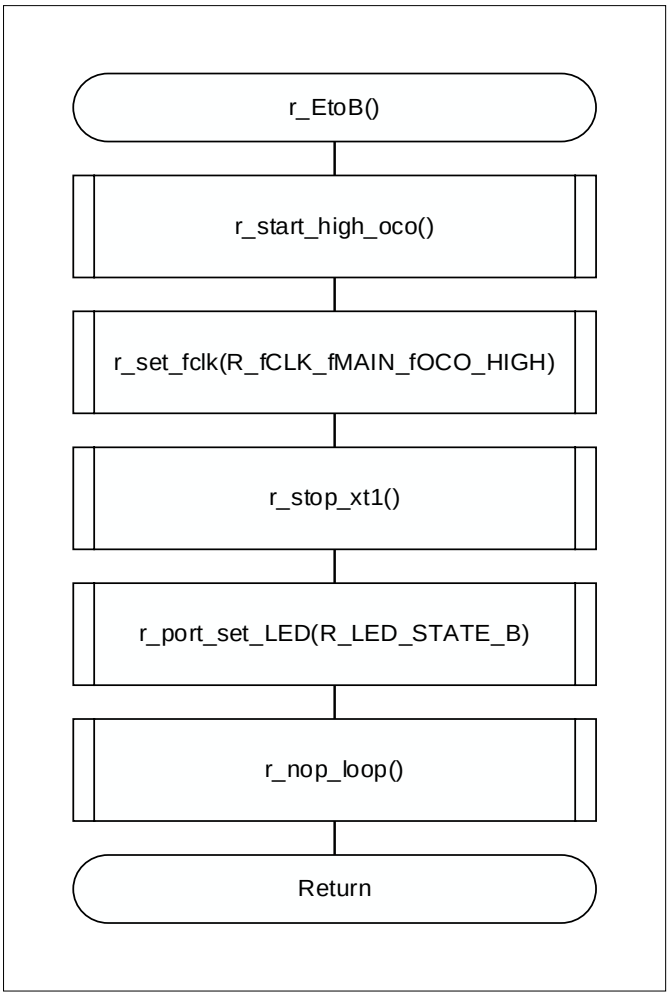
Figure 4-7 Status Transition OtoE



4.6.6 Status Transition EtoB

Figure 4-8 shows the flowchart of the status transition EtoB.

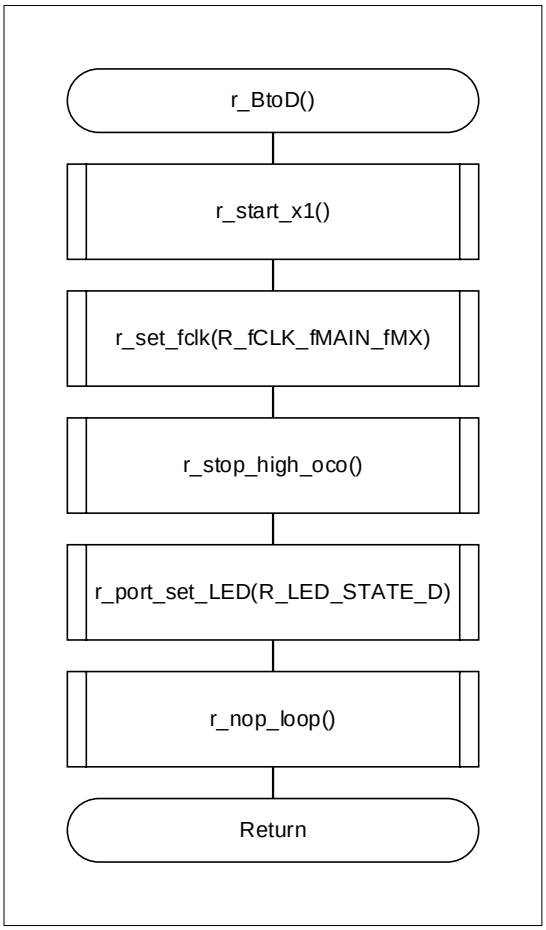
Figure 4-8 Status Transition EtoB



4.6.7 Status Transition BtoD

Figure 4-9 shows the flowchart of the status transition BtoD.

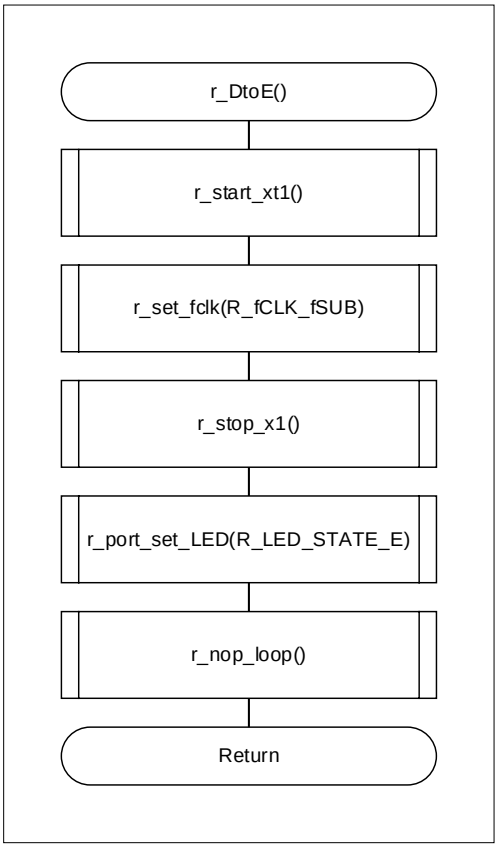
Figure 4-9 Status Transition BtoD



4.6.8 Status Transition DtoE

Figure 4-10 shows the flowchart of the status transition DtoE.

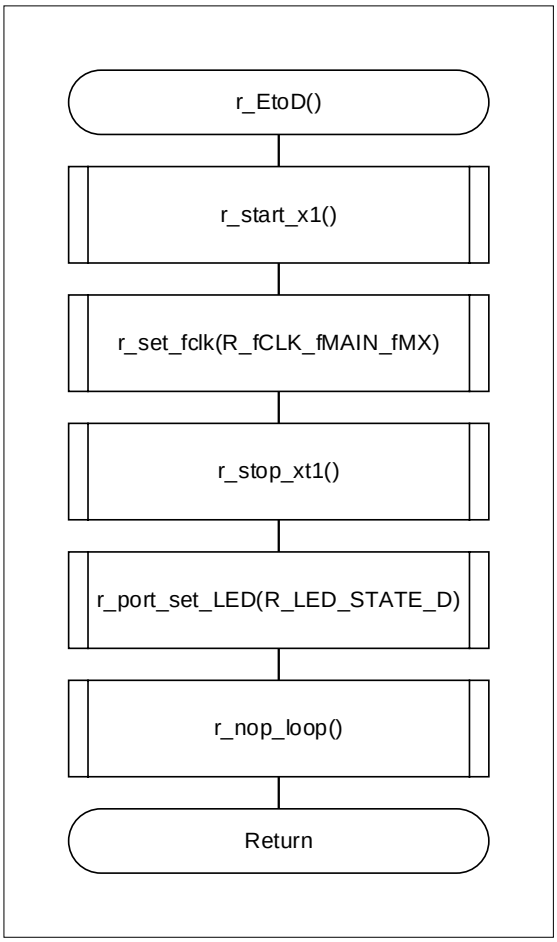
Figure 4-10 Status Transition DtoE



4.6.9 Status Transition EtoD

Figure 4-11 shows the flowchart of the status transition EtoD.

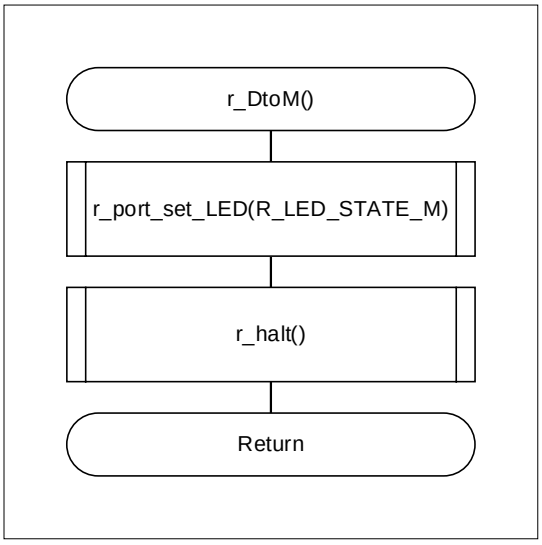
Figure 4-11 Status Transition EtoD



4.6.10 Status Transition DtoM

Figure 4-12 shows the flowchart of the status transition DtoM.

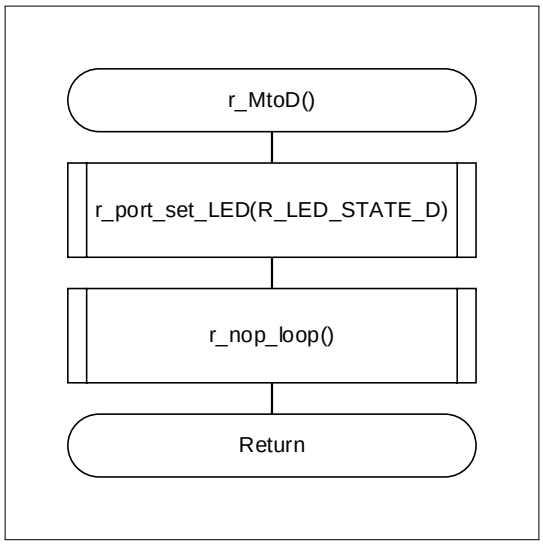
Figure 4-12 Status Transition DtoM



4.6.11 Status Transition MtoD

Figure 4-13 shows the flowchart of the status transition MtoD.

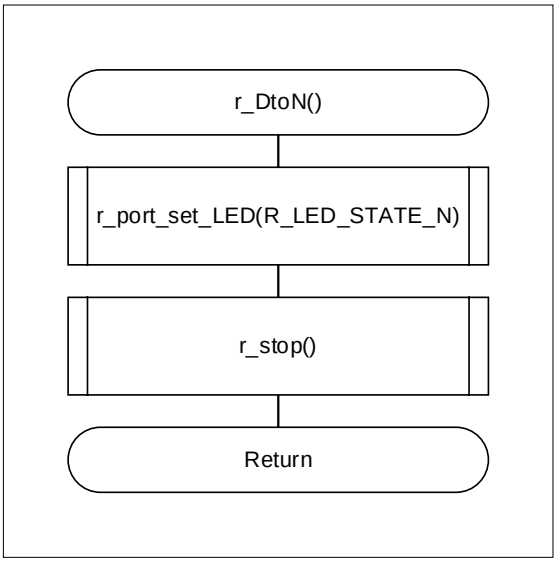
Figure 4-13 Status Transition MtoD



4.6.12 Status Transition DtoN

Figure 4-14 shows the flowchart of the status transition DtoN.

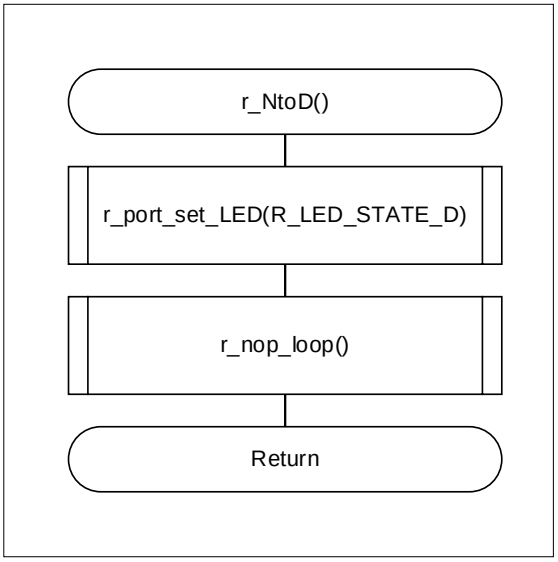
Figure 4-14 Status Transition DtoN



4.6.13 Status Transition NtoD

Figure 4-15 shows the flowchart of the status transition NtoD.

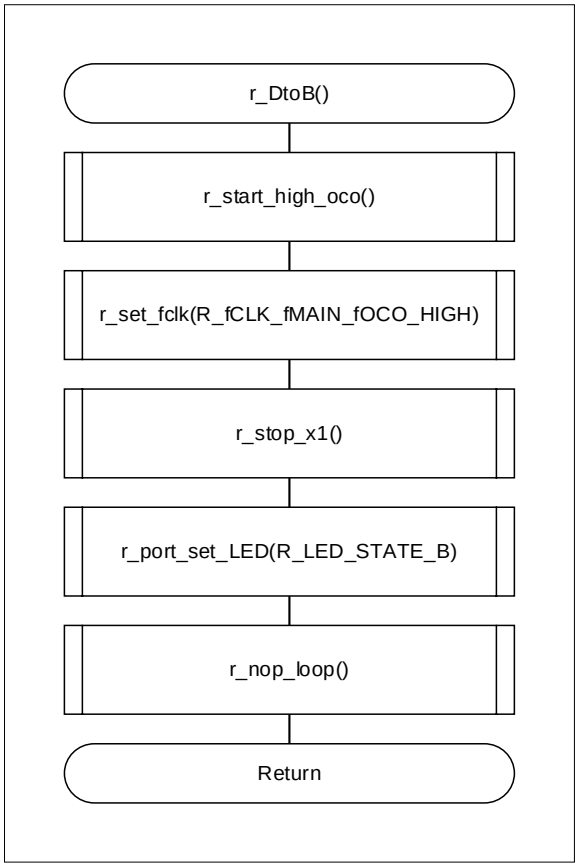
Figure 4-15 Status Transition NtoD



4.6.14 Status Transition DtoB

Figure 4-16 shows the flowchart of the status transition.

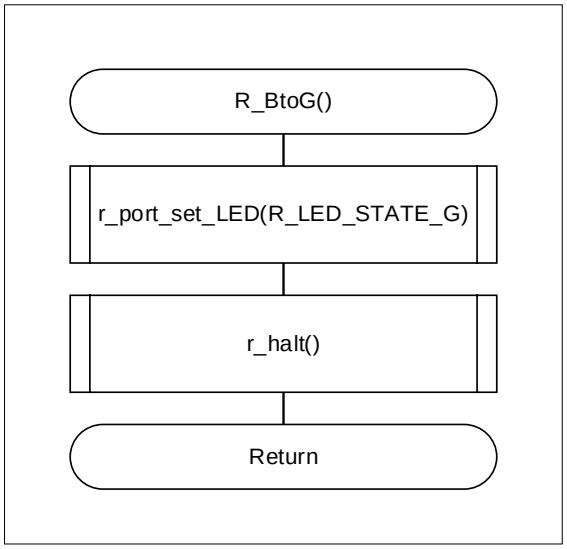
Figure 4-16 Status Transition DtoB



4.6.15 Status Transition BtoG

Figure 4-17 shows the flowchart of the status transition BtoG.

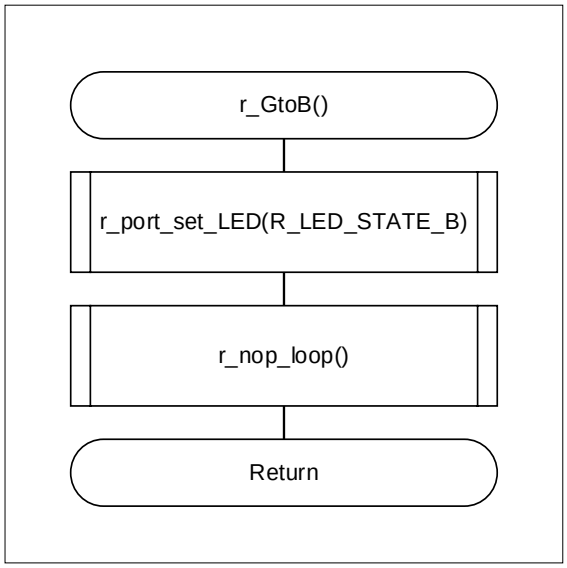
Figure 4-17 Status Transition BtoG



4.6.16 Status Transition GtoB

Figure 4-18 shows the flowchart of the status transition GtoB.

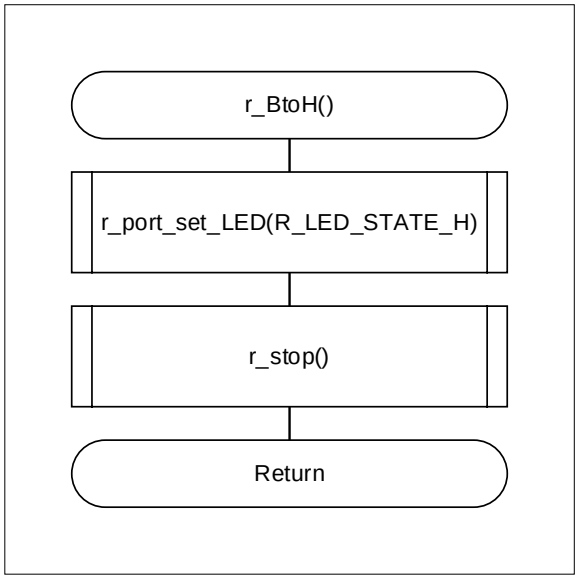
Figure 4-18 Status Transition GtoB



4.6.17 Status Transition BtoH

Figure 4-19 shows the flowchart of the status transition Both.

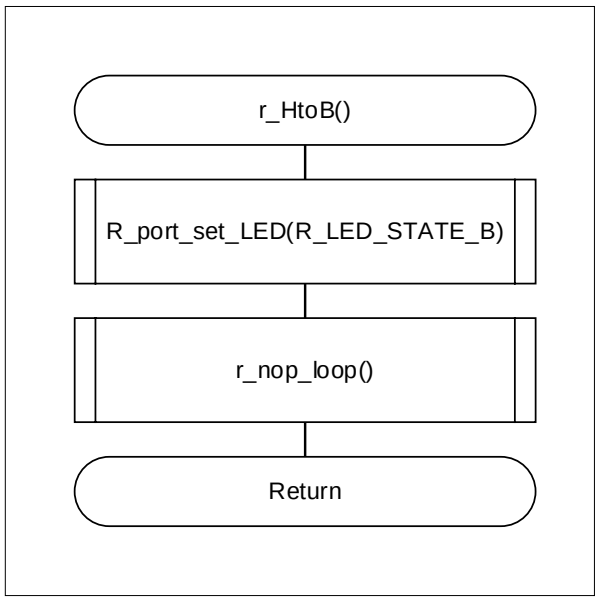
Figure 4-19 Status Transition BtoH



4.6.18 Status Transition HtoB

Figure 4-20 shows the flowchart of the status transition HtoB.

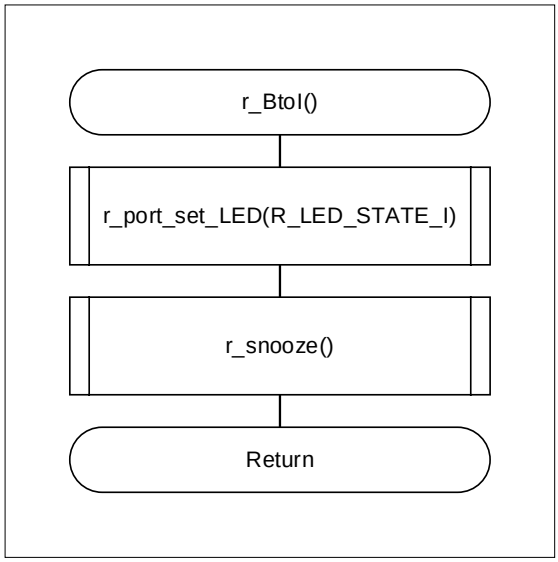
Figure 4-20 Status Transition HtoB



4.6.19 Status Transition BtoI

Figure 4-21 shows the flowchart of the status transition BtoI.

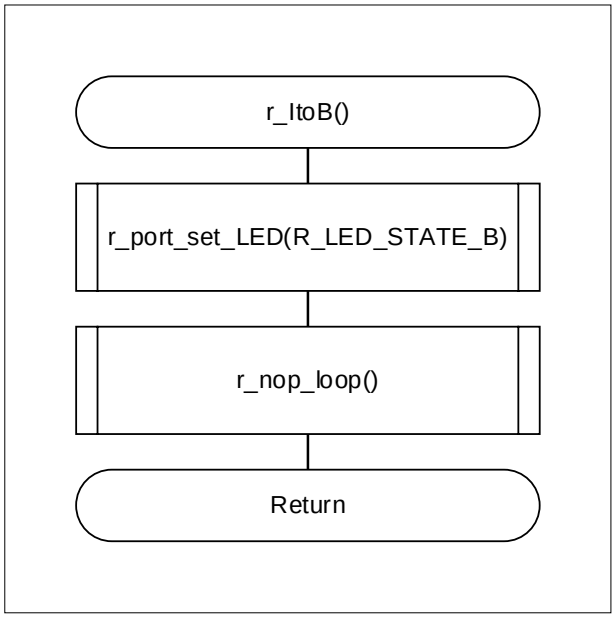
Figure 4-21 Status Transition BtoI



4.6.20 Status Transition ItoB

Figure 4-22 shows the flowchart of the status transition n ItoB.

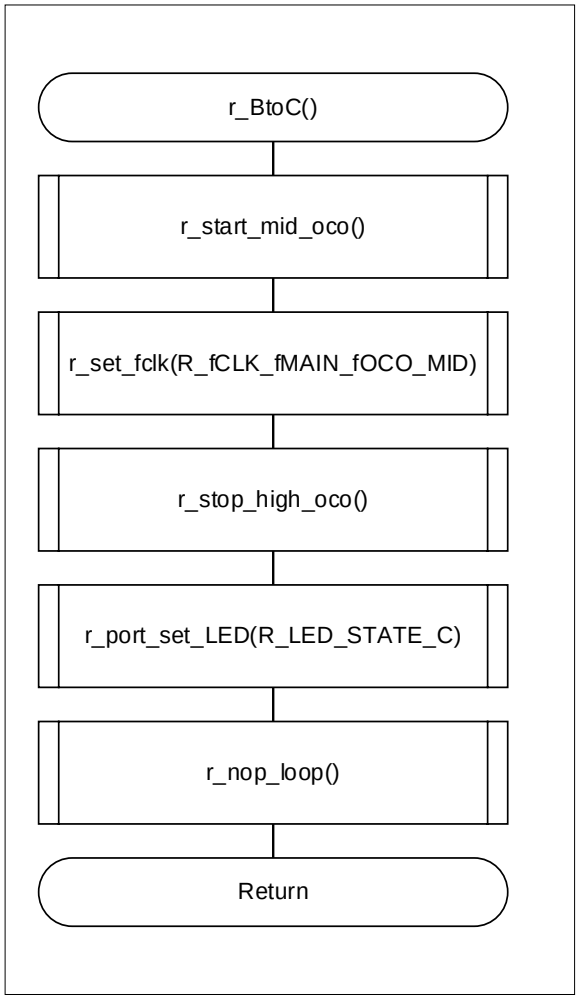
Figure 4-22 Status Transition ItoB



4.6.21 Status Transition BtoC

Figure 4-23 shows the flowchart of the status transition BtoC.

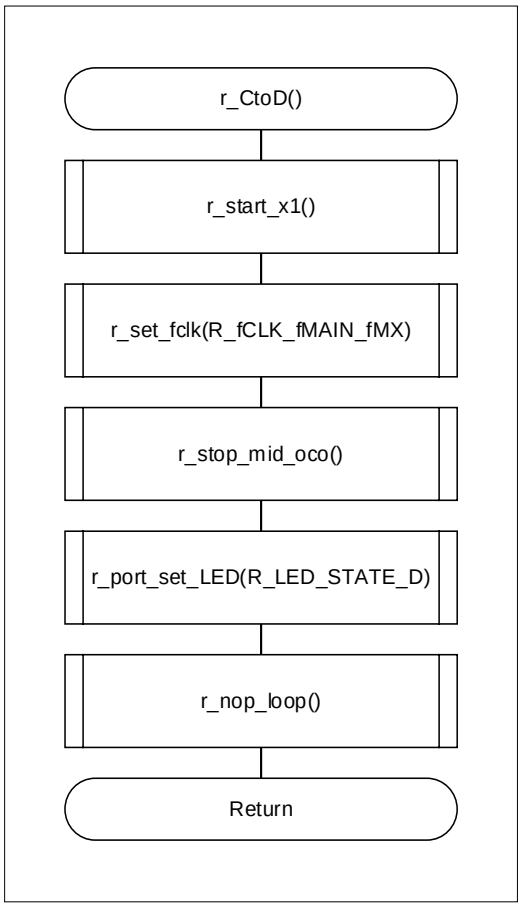
Figure 4-23 Status Transition BtoC



4.6.22 Status Transition CtoD

Figure 4-24 shows the flowchart of the status transition CtoD.

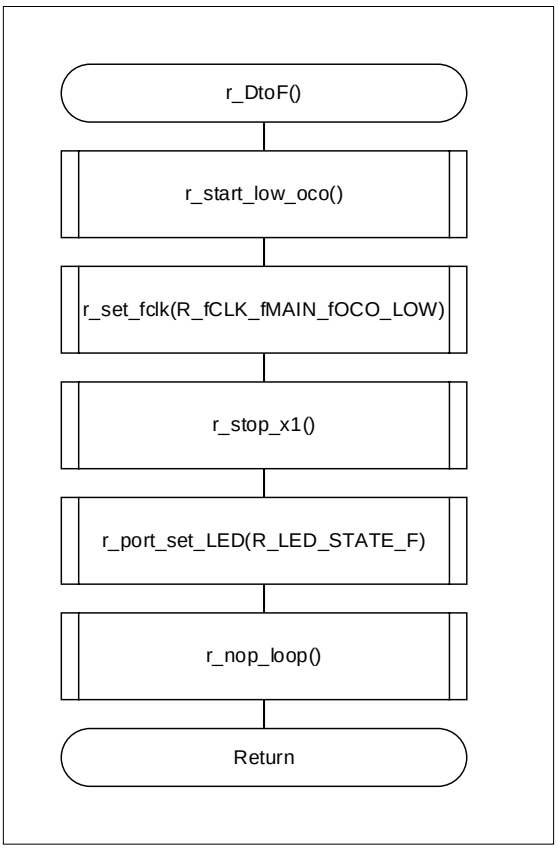
Figure 4-24 Status Transition CtoD



4.6.23 Status Transition DtoF

Figure 4-25 shows the flowchart of the status transition DtoF.

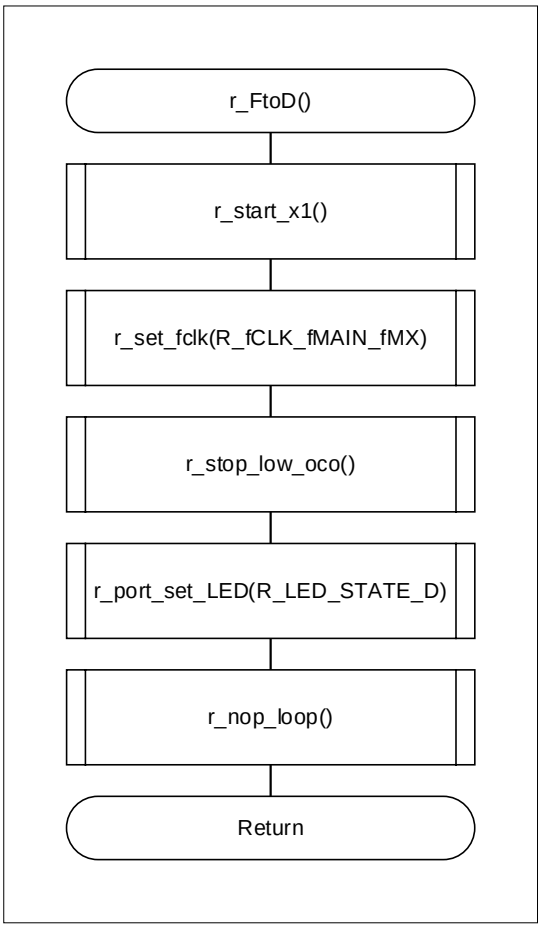
Figure 4-25 Status Transition DtoF



4.6.24 Status Transition FtoD

Figure 4-26 shows the flowchart of the status transition FtoD.

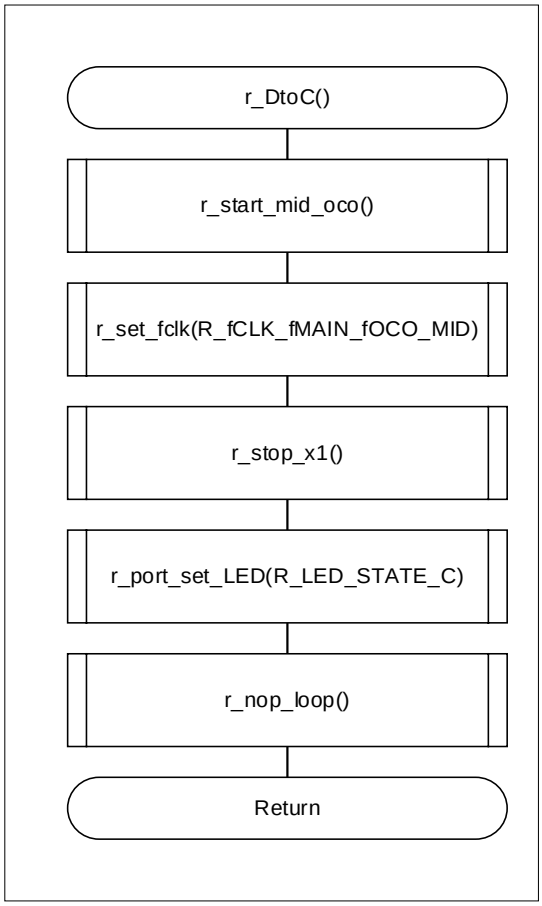
Figure 4-26 Status Transition FtoD



4.6.25 Status Transition DtoC

Figure 4-27 shows the flowchart of the status transition DtoC.

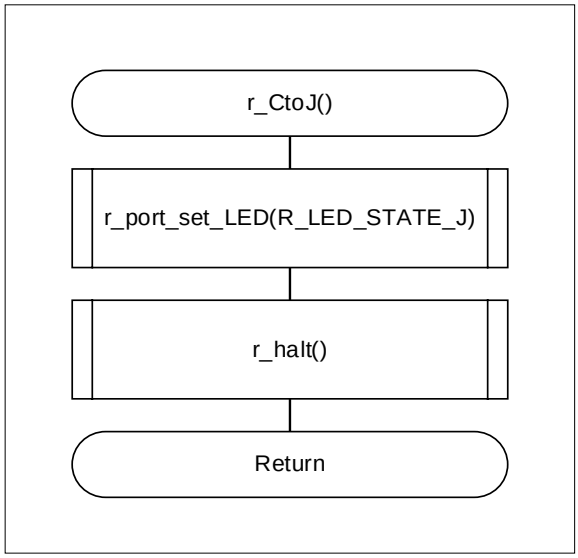
Figure 4-27 Status Transition DtoC



4.6.26 Status Transition CtoJ

Figure 4-28 shows the flowchart of the status transition CtoJ.

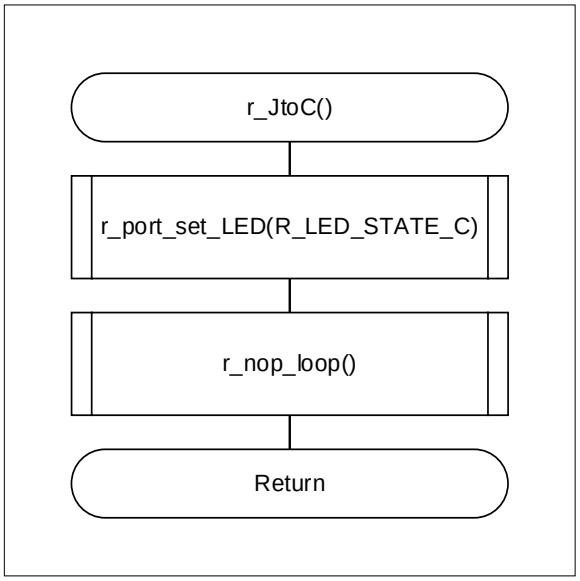
Figure 4-28 Status Transition CtoJ



4.6.27 Status Transition JtoC

Figure 4-29 shows the flowchart of the status transition JtoC.

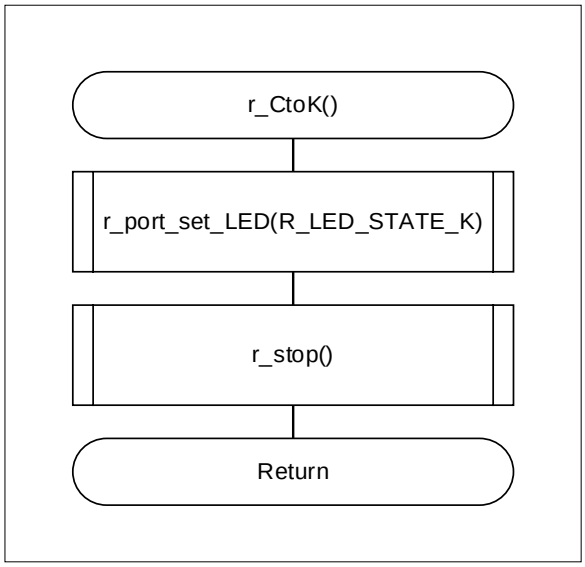
Figure 4-29 Status Transition JtoC



4.6.28 Status Transition CtoK

Figure 4-30 shows the flowchart of the status transition CtoK.

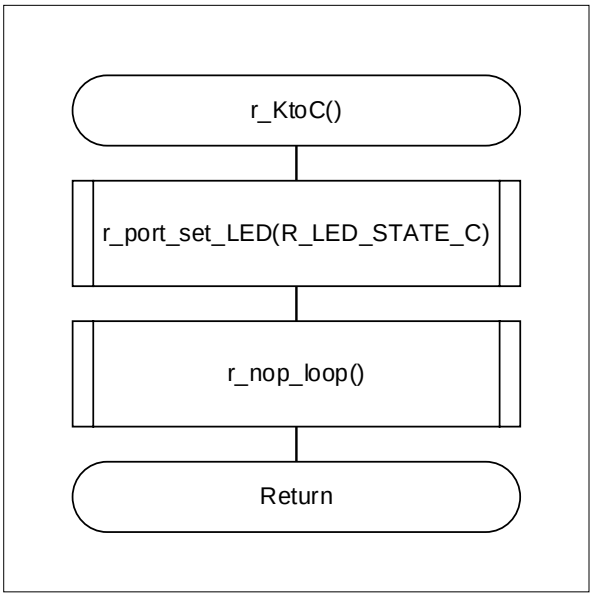
Figure 4-30 Status Transition CtoK



4.6.29 Status Transition KtoC

Figure 4-31 shows the flowchart of the status transition KtoC.

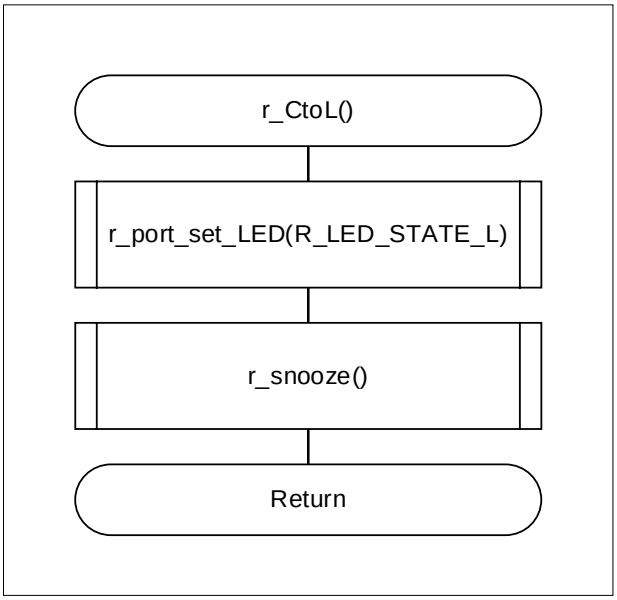
Figure 4-31 Status Transition KtoC



4.6.30 Status Transition CtoL

Figure 4-32 shows the flowchart of the status transition CtoL.

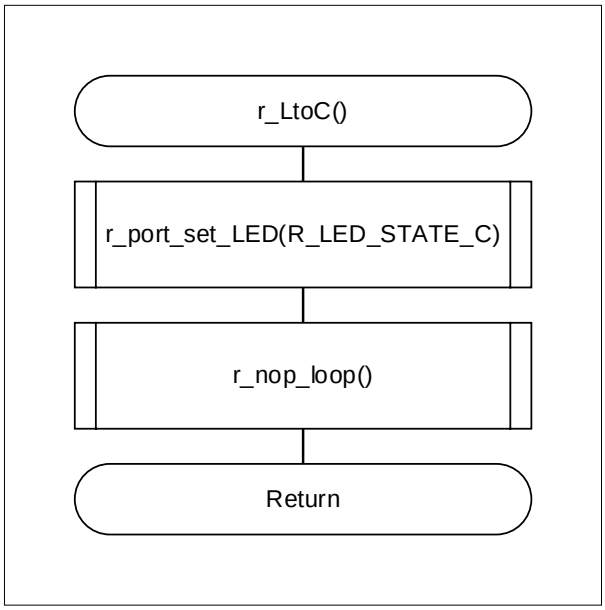
Figure 4-32 Status Transition CtoL



4.6.31 Status Transition LtoC

Figure 4-33 shows the flowchart of the status transition LtoC.

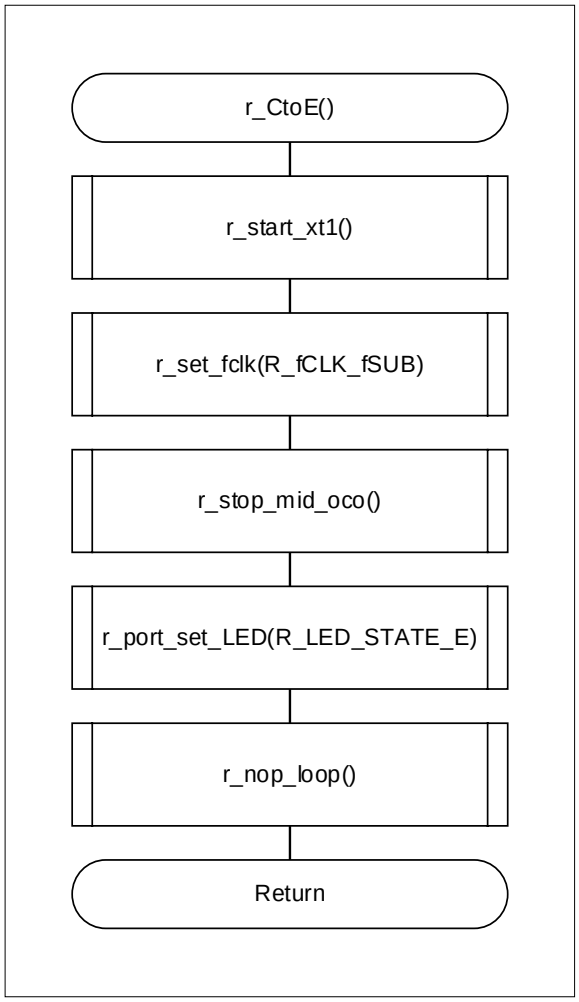
Figure 4-33 Status Transition LtoC



4.6.32 Status Transition CtoE

Figure 4-34 shows the flowchart of the status transition CtoE.

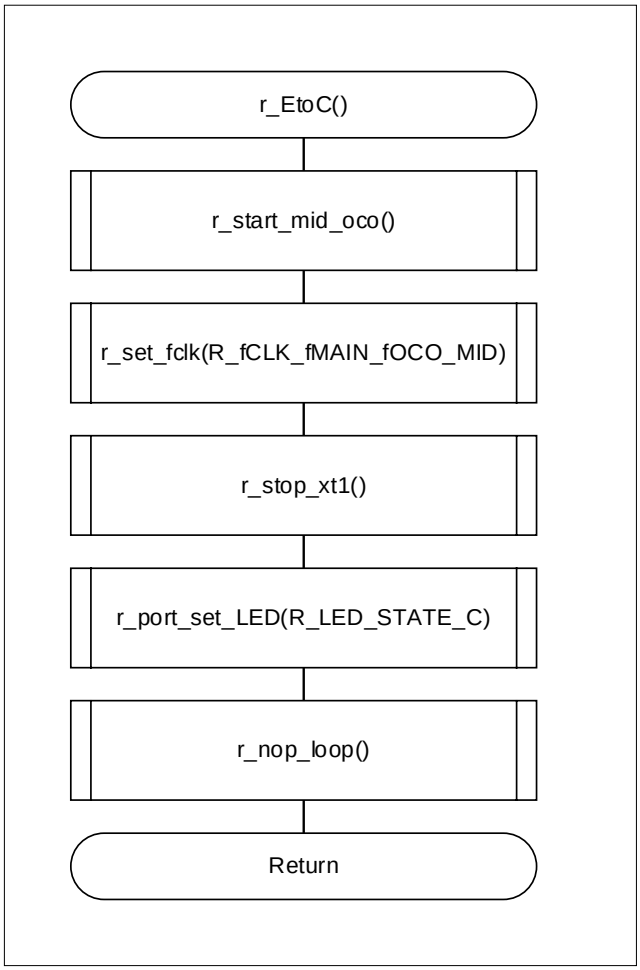
Figure 4-34 Status Transition CtoE



4.6.33 Status Transition EtoC

Figure 4-35 shows the flowchart of the status transition EtoC.

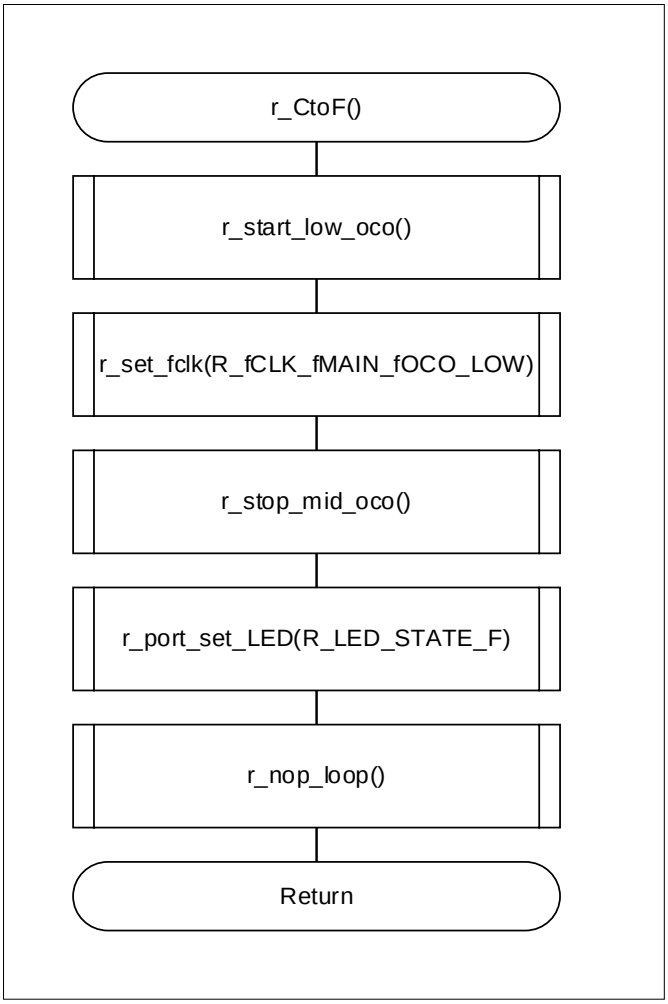
Figure 4-35 Status Transition EtoC



4.6.34 Status Transition CtoF

Figure 4-36 shows the flowchart of the status transition CtoF.

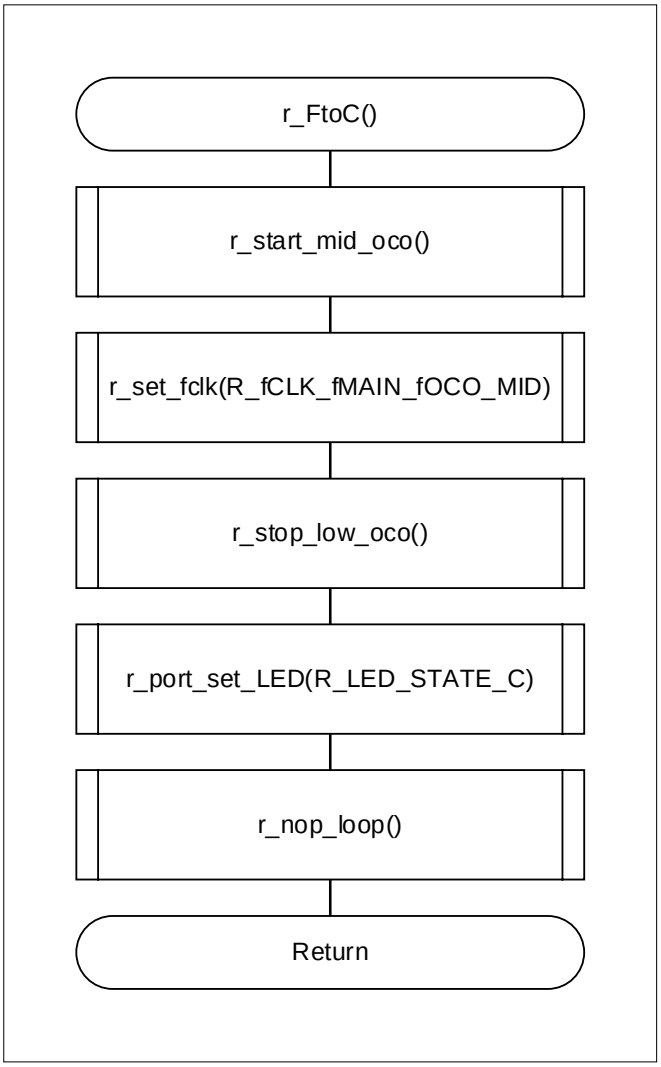
Figure 4-36 Status Transition CtoF



4.6.35 Status Transition FtoC

Figure 4-37 shows the flowchart of the status transition FtoC.

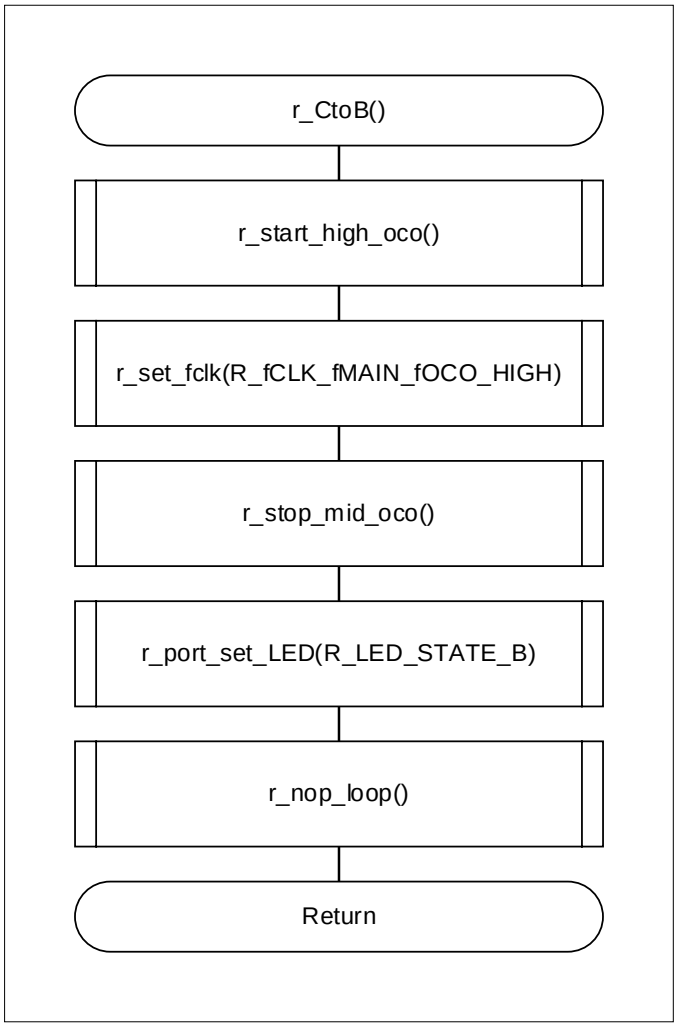
Figure 4-37 Status Transition FtoC



4.6.36 Status Transition CtoB

Figure 4-38 shows the flowchart of the status transition CtoB.

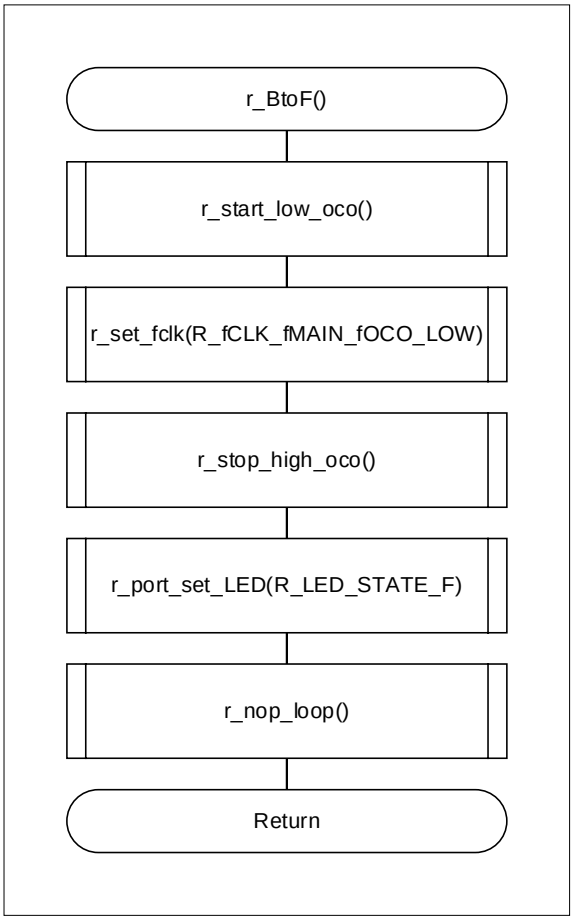
Figure 4-38 Status Transition CtoB



4.6.37 Status Transition BtoF

Figure 4-39 shows the flowchart of the status transition BtoF.

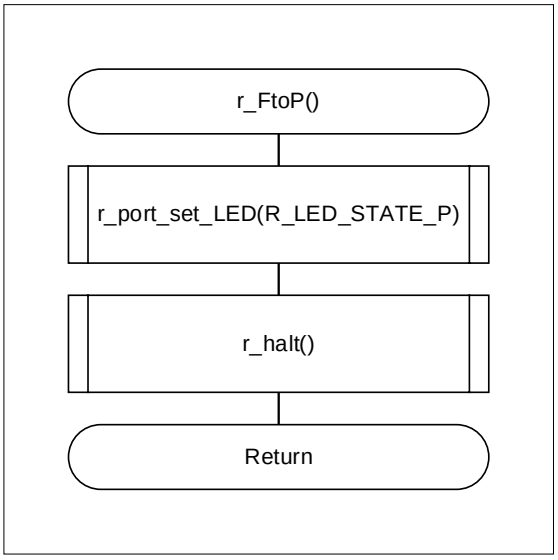
Figure 4-39 Status Transition BtoF



4.6.38 Status Transition FtoP

Figure 4-40 shows the flowchart of the status transition FtoP.

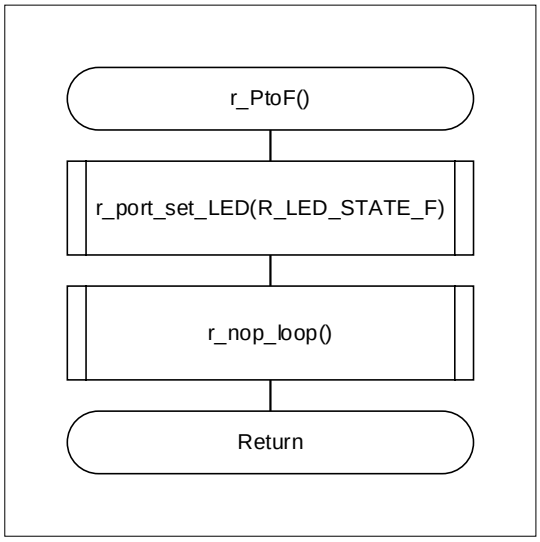
Figure 4-40 Status Transition FtoP



4.6.39 Status Transition PtoF

Figure 4-41 shows the flowchart of the status transition PtoF.

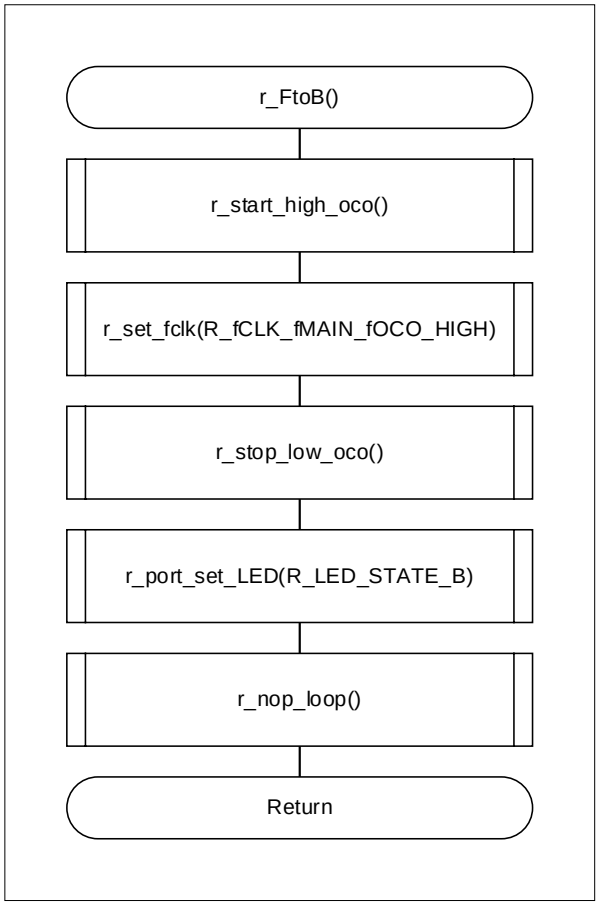
Figure 4-41 Status Transition PtoF



4.6.40 Status Transition FtoB

Figure 4-42 shows the flowchart of the status transition FtoB.

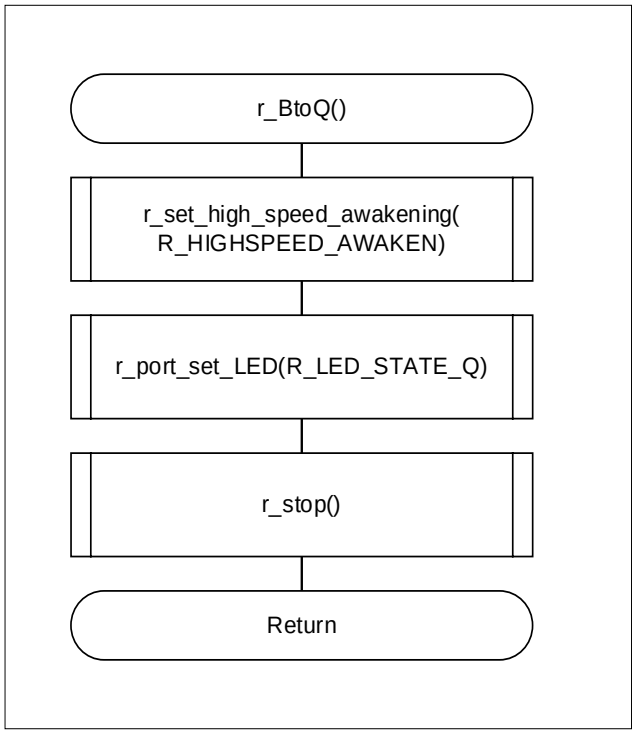
Figure 4-42 Status Transition FtoB



4.6.41 Status Transition BtoQ

Figure 4-43 shows the flowchart of the status transition BtoQ.

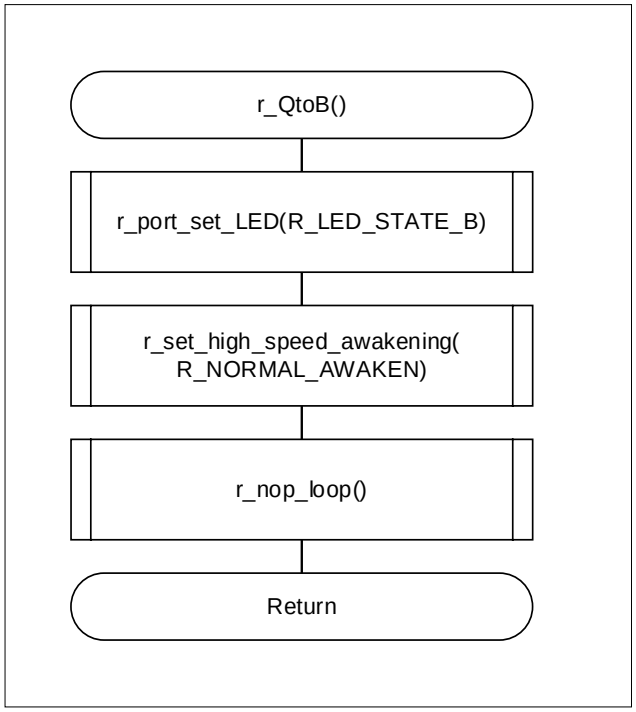
Figure 4-43 Status Transition BtoQ



4.6.42 Status Transition QtoB

Figure 4-44 shows the flowchart of the status transition QtoB.

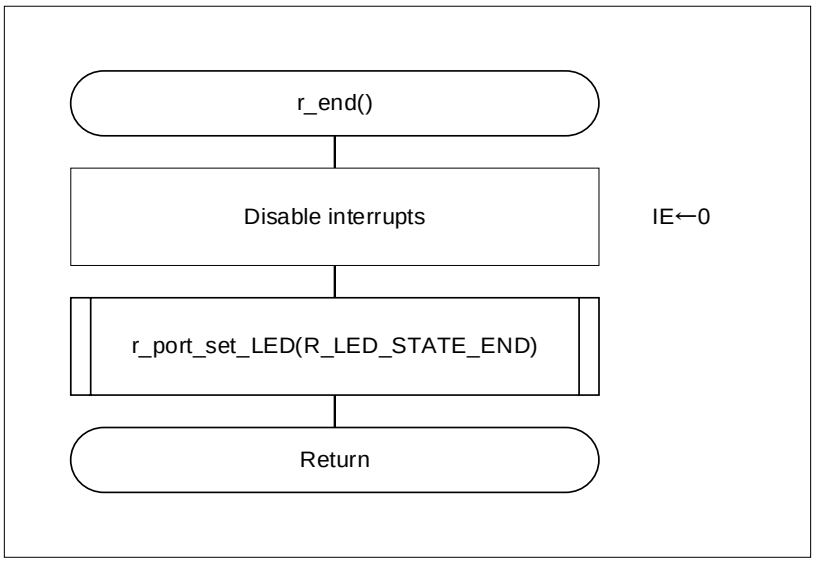
Figure 4-44 Status Transition QtoB



4.6.43 End Processing of Status Transition

Figure 4-45 shows the flowchart of the End processing of status transition.

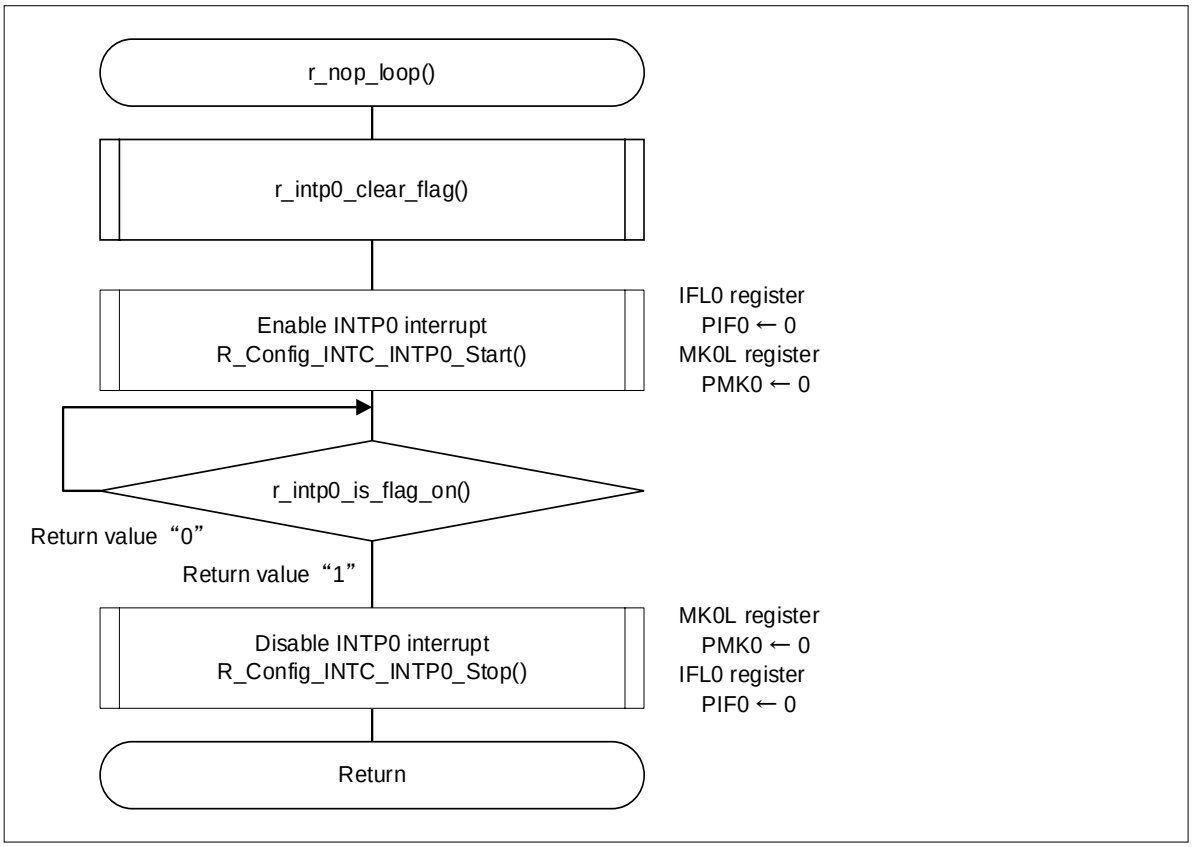
Figure 4-45 End Processing of Status Transition



4.6.44 Waiting for an Interrupt (Repetition of NOP)

Figure 4-46 shows the flowchart for waiting for an interrupt (repetition of NOP).

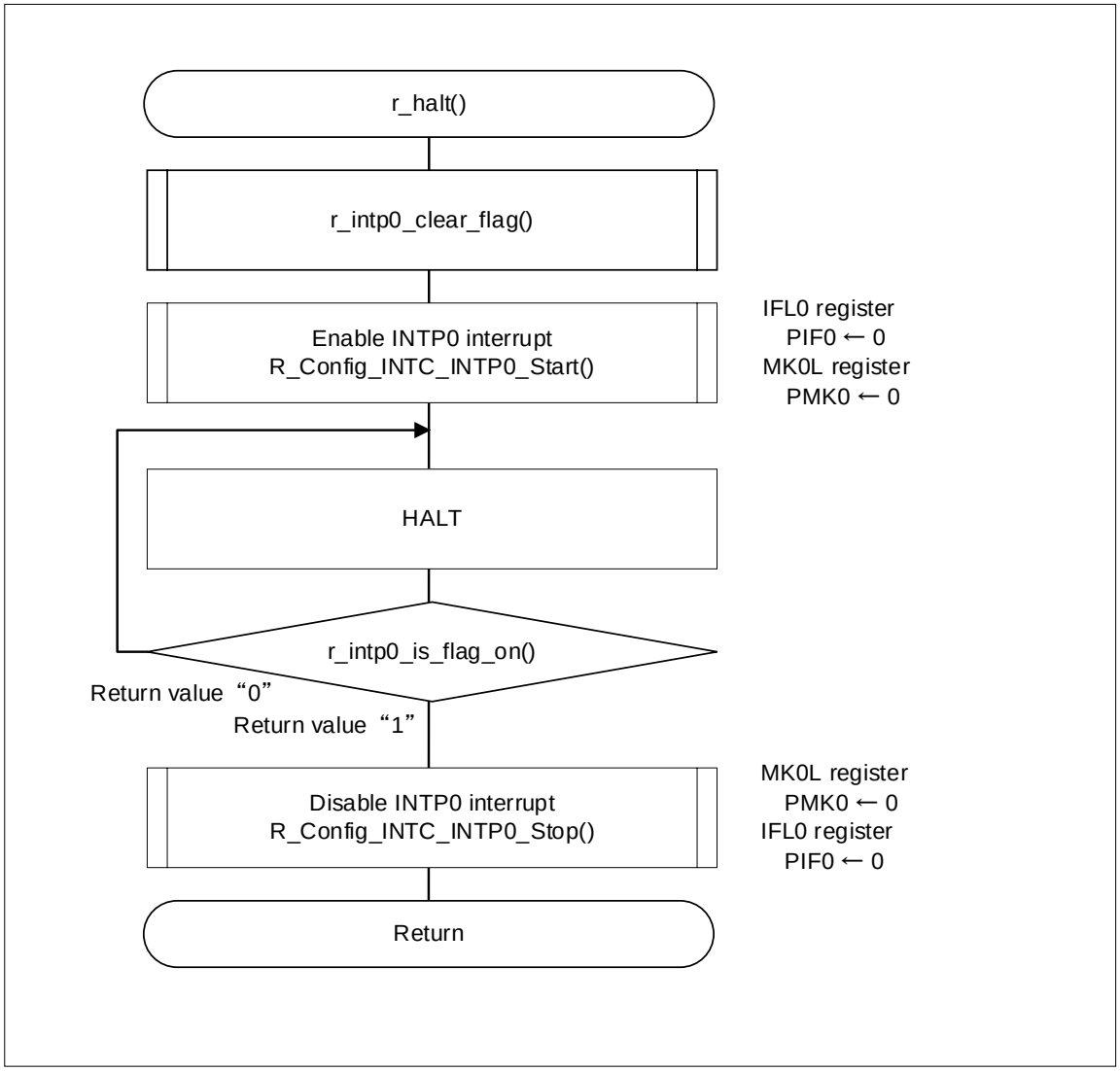
Figure 4-46 Waiting for an Interrupt (Repetition of NOP)



4.6.45 Waiting for an Interrupt (HALT)

Figure 4-47 shows the flowchart for waiting for an interrupt (HALT).

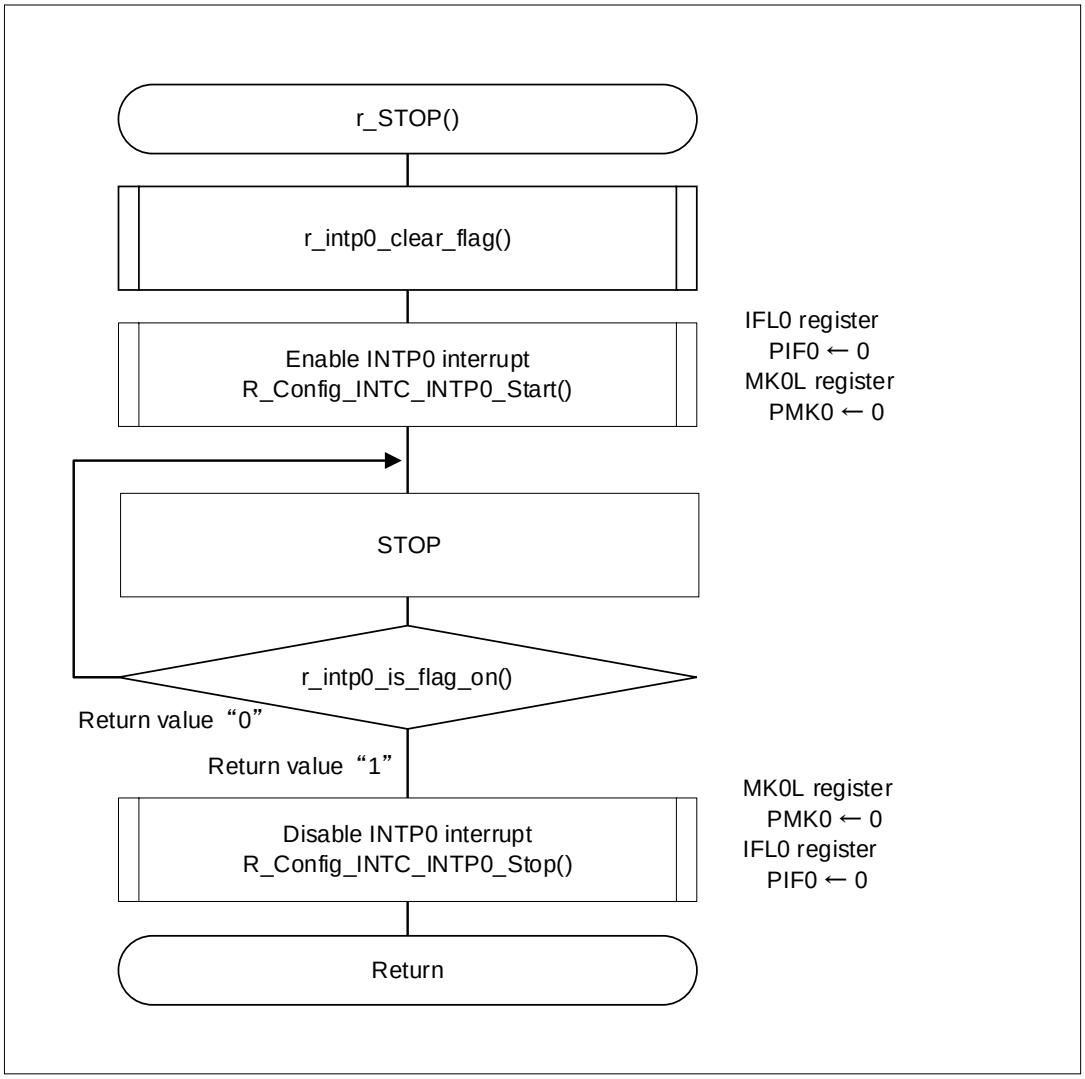
Figure 4-47 Waiting for an Interrupt (HALT)



4.6.46 Waiting for an Interrupt (STOP)

Figure 4-48 shows the flowchart for waiting for an interrupt (STOP).

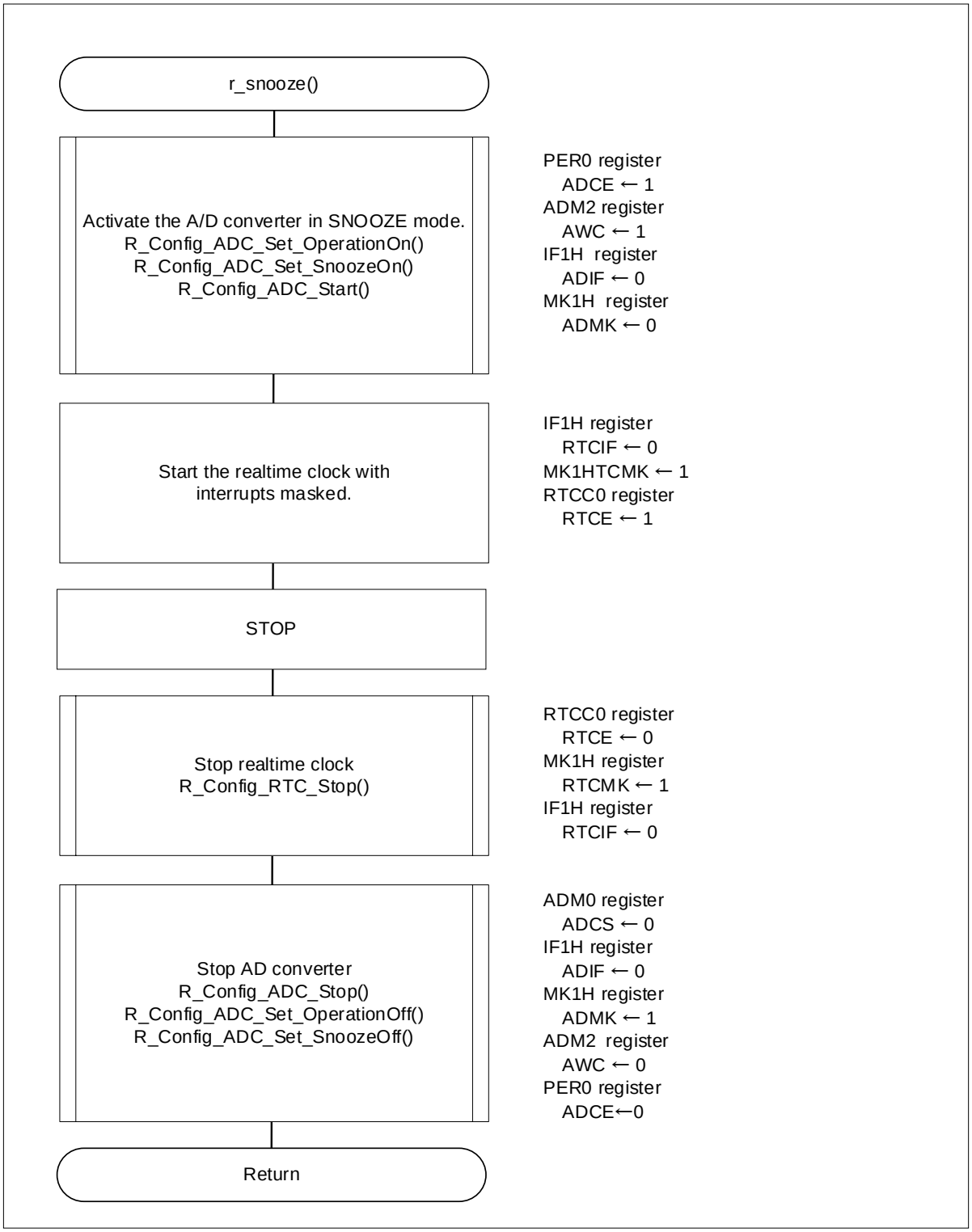
Figure 4-48 Waiting for an Interrupt (STOP)



4.6.47 Waiting for an Interrupt (SNOOZE)

Figure 4-49 shows the flowchart for waiting for an interrupt (SNOOZE).

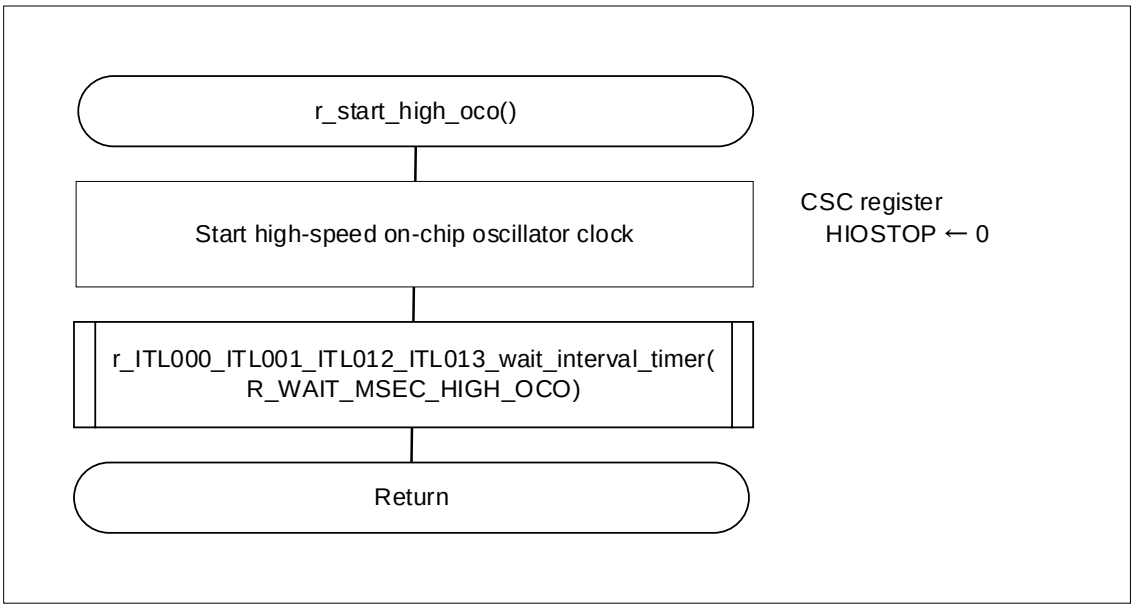
Figure 4-49 Waiting for an Interrupt (SNOOZE)



4.6.48 Starting High-Speed On-Chip Oscillator Clock

Figure 4-50 shows the flowchart for starting the high-speed on-chip oscillator clock.

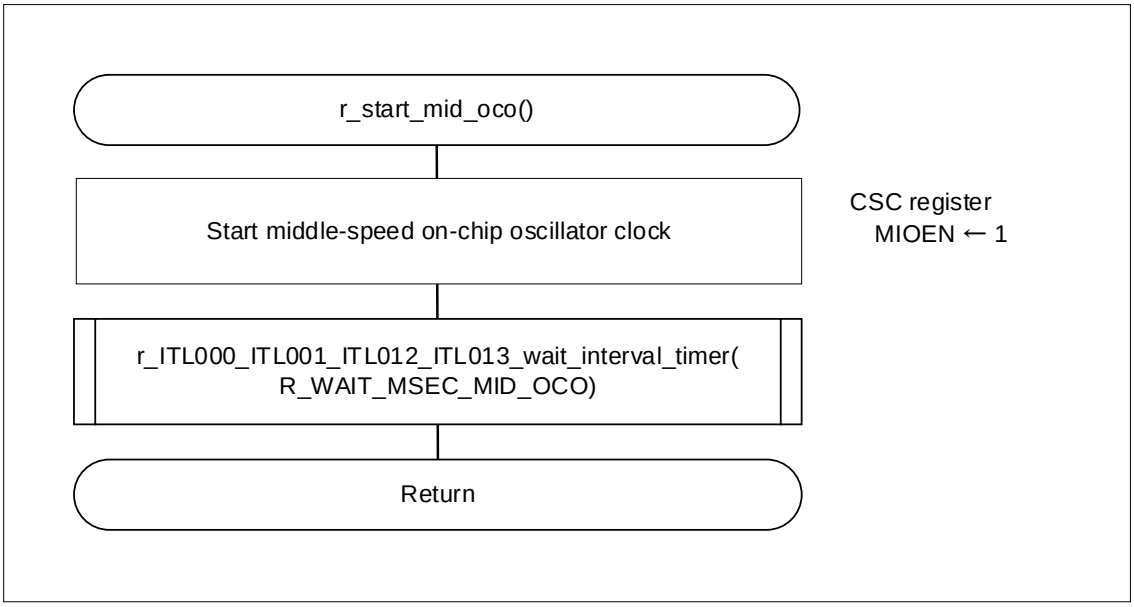
Figure 4-50 Starting High-Speed On-Chip Oscillator Clock



4.6.49 Starting Middle-Speed On-Chip Oscillator Clock

Figure 4-51 shows the flowchart for starting the middle-speed on-chip oscillator clock.

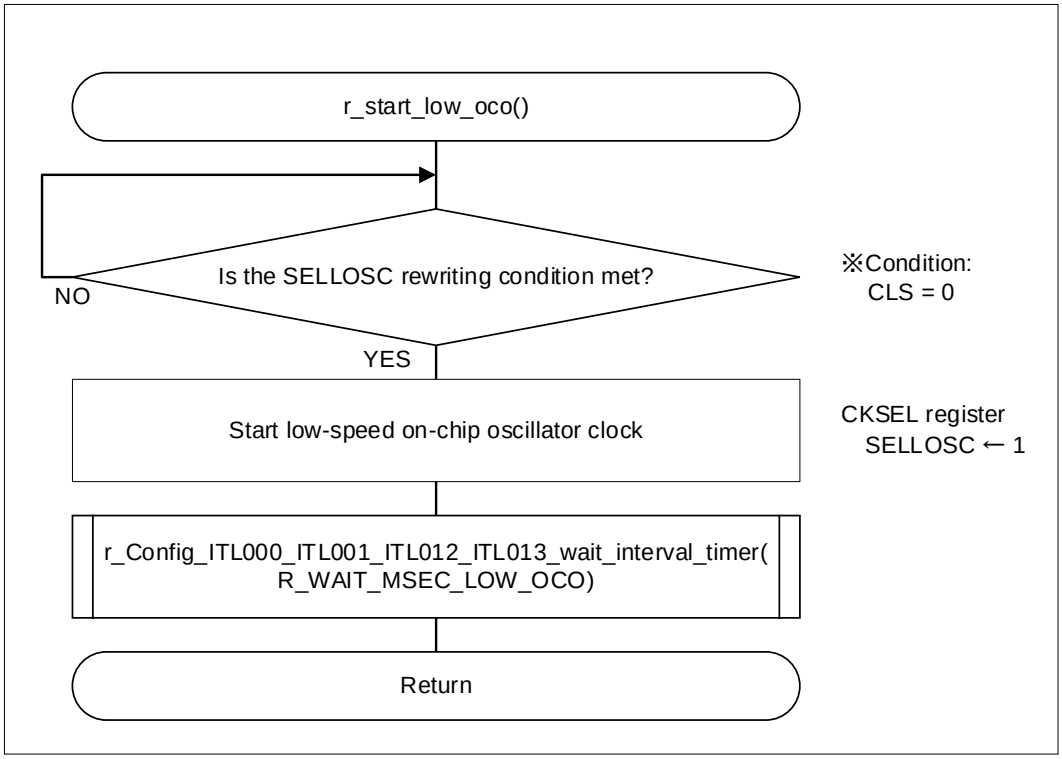
Figure 4-51 Starting Middle-Speed On-Chip Oscillator Clock



4.6.50 Starting Low-Speed On-Chip Oscillator Clock

Figure 4-52 shows the flowchart for starting the low-speed on-chip oscillator clock.

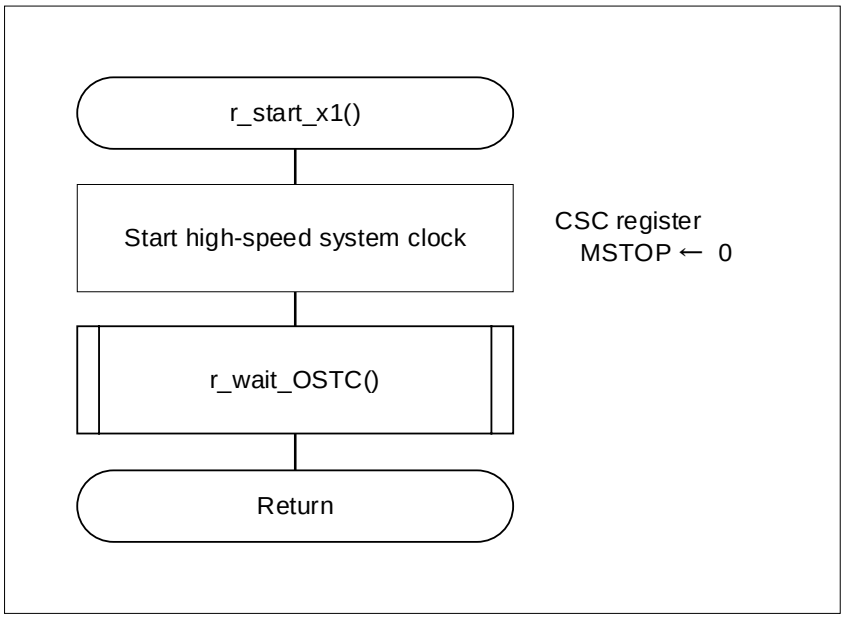
Figure 4-52 Starting Low-Speed On-Chip Oscillator Clock



4.6.51 Starting High-Speed System Clock

Figure 4-53 shows the flowchart for starting the high-speed system clock.

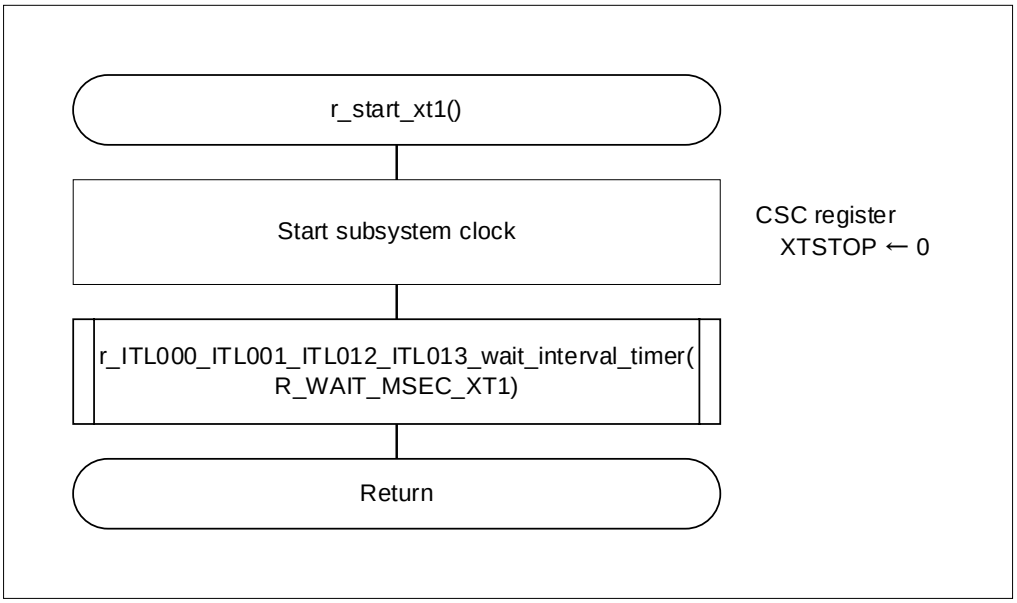
Figure 4-53 Starting High-Speed System Clock



4.6.52 Starting Subsystem Clock

Figure 4-54 shows the flowchart for starting the subsystem clock.

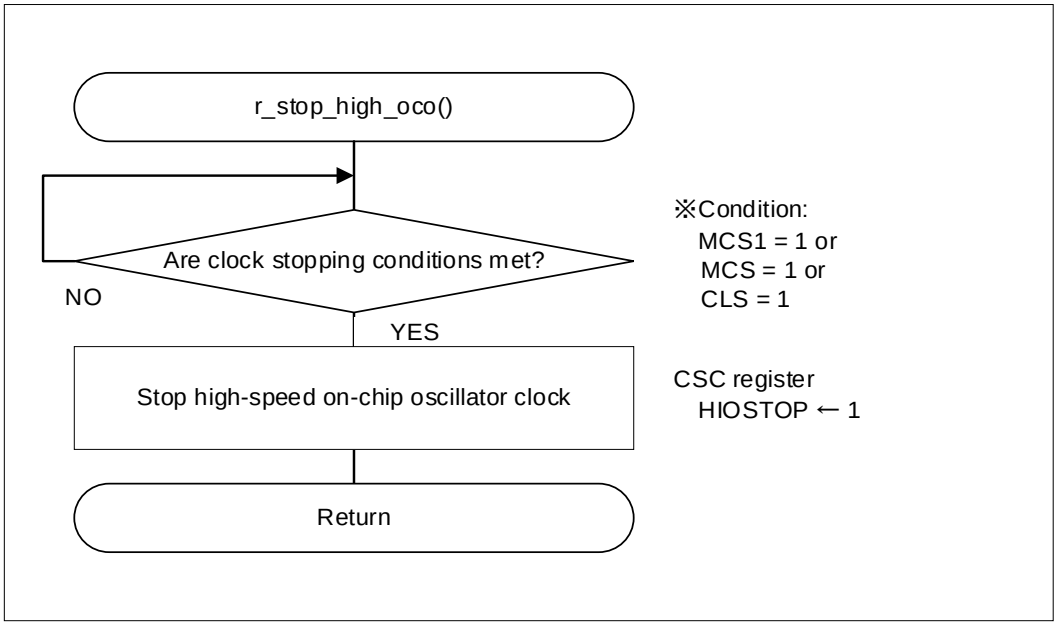
Figure 4-54 Starting Subsystem Clock



4.6.53 Stopping High-Speed On-Chip Oscillator Clock

Figure 4-55 shows the flowchart for stopping the high-speed on-chip oscillator clock.

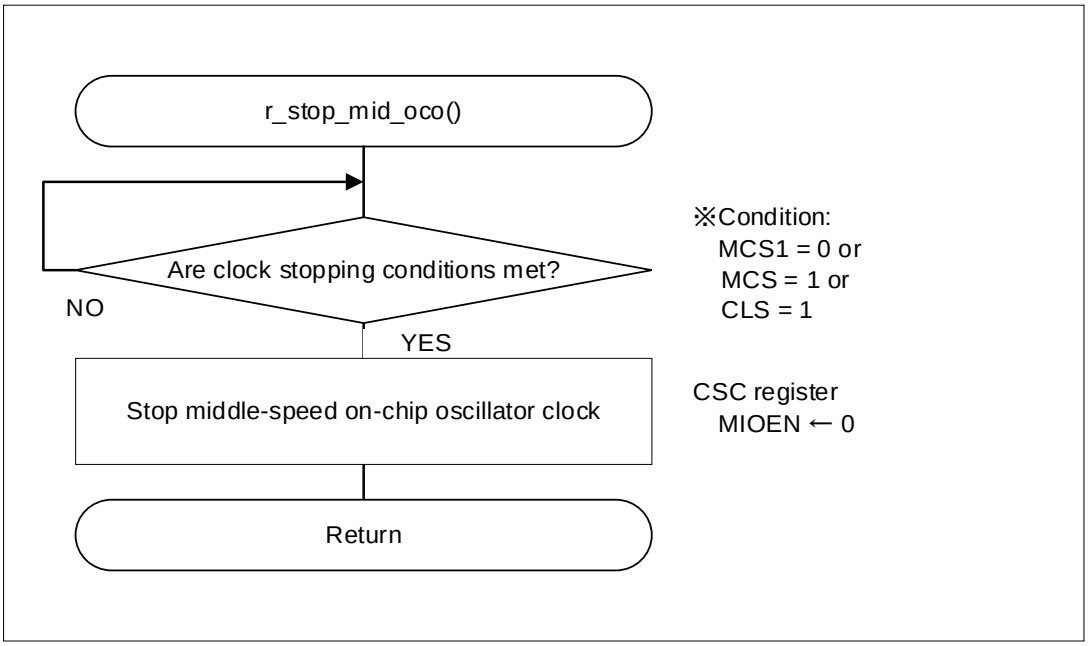
Figure 4-55 Stopping High-Speed On-Chip Oscillator Clock



4.6.54 Stopping Middle-Speed On-Chip Oscillator Clock

Figure 4-56 shows the flowchart for stopping the middle-speed on-chip oscillator clock.

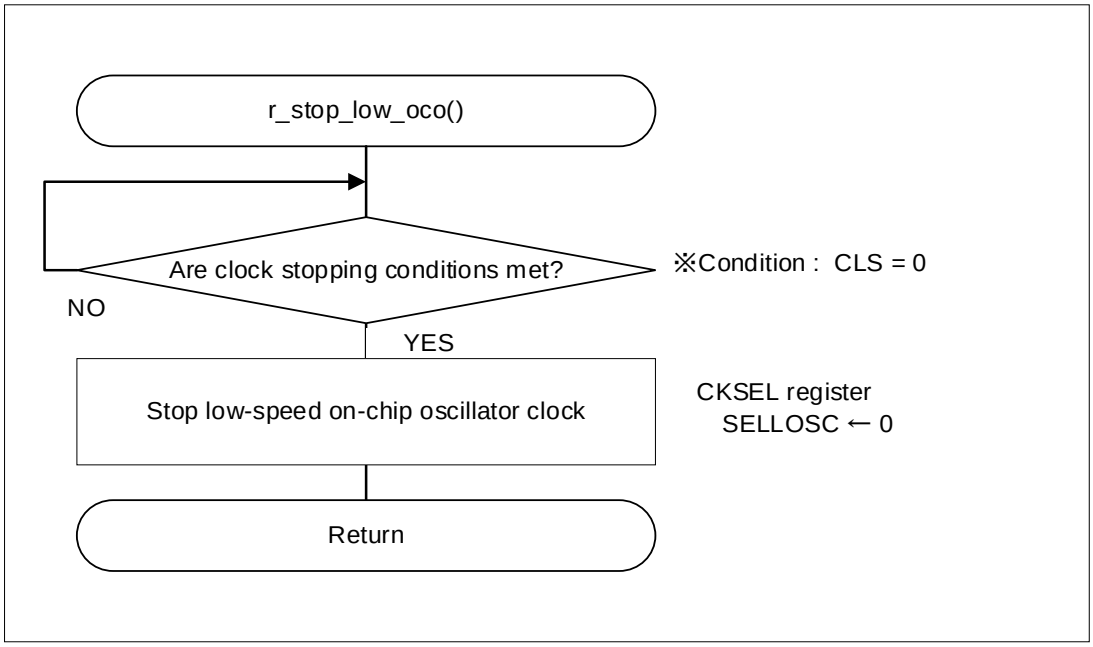
Figure 4-56 Stopping Middle-Speed On-Chip Oscillator Clock



4.6.55 Stopping Low-Speed On-Chip Oscillator Clock

Figure 4-57 shows the flowchart for stopping the low-speed on-chip oscillator clock.

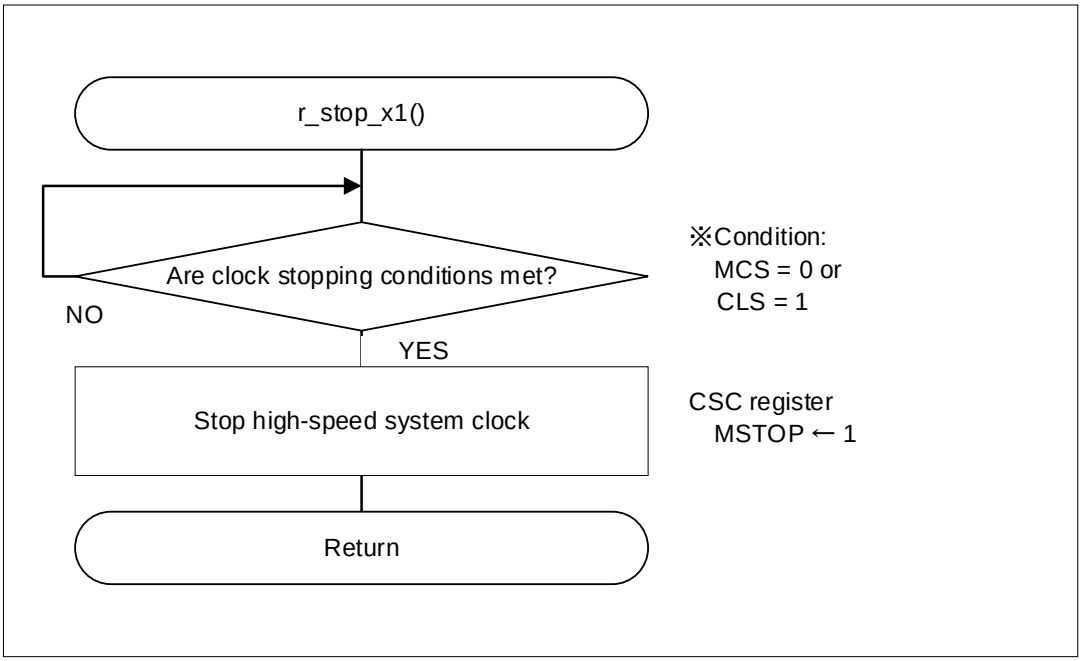
Figure 4-57 Stopping Low-Speed On-Chip Oscillator Clock



4.6.56 Stopping High-Speed System Clock

Figure 4-58 shows the flowchart for stopping the high-speed system clock.

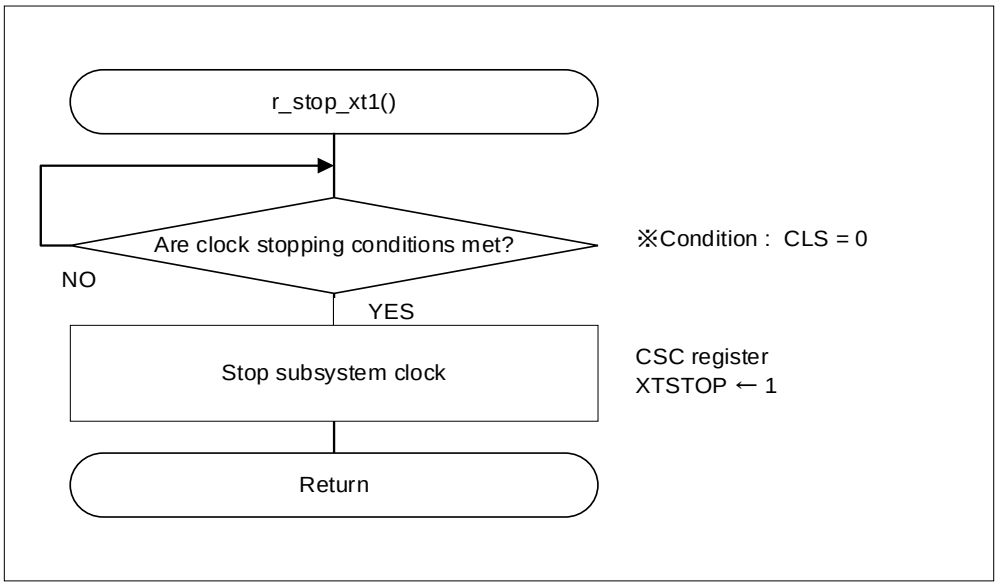
Figure 4-58 Stopping High-Speed System Clock



4.6.57 Stopping Subsystem Clock

Figure 4-59 shows the flowchart for stopping the subsystem clock.

Figure 4-59 Stopping Subsystem Clock



4.6.58 Clock Change Setting

Figure 4-60, Figure 4-61, Figure 4-62, and Figure 4-63 show flowcharts for the clock change setting.

Figure 4-60 Clock Change Setting (1/4)

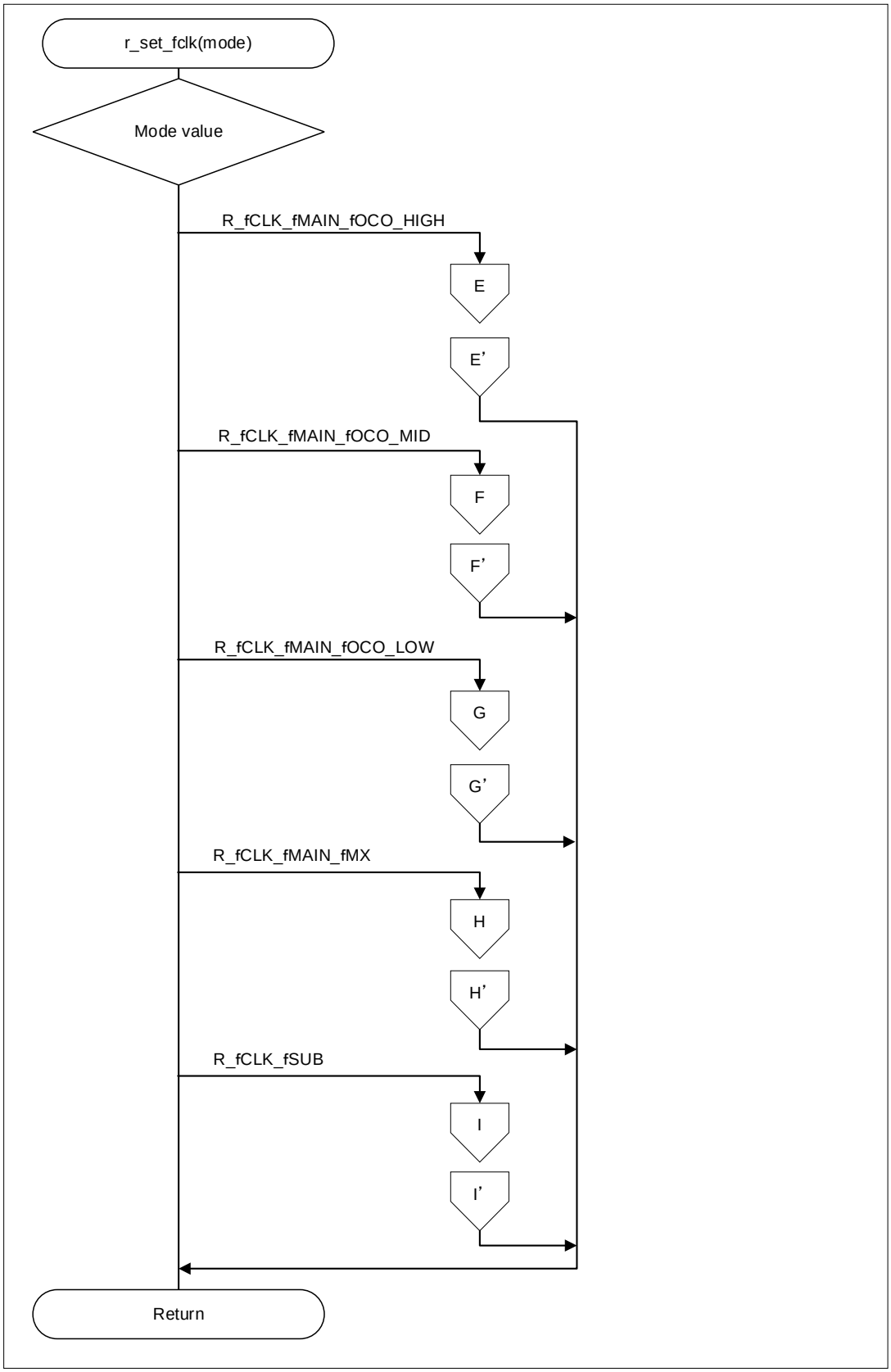


Figure 4-61 Clock Change Setting (2/4)

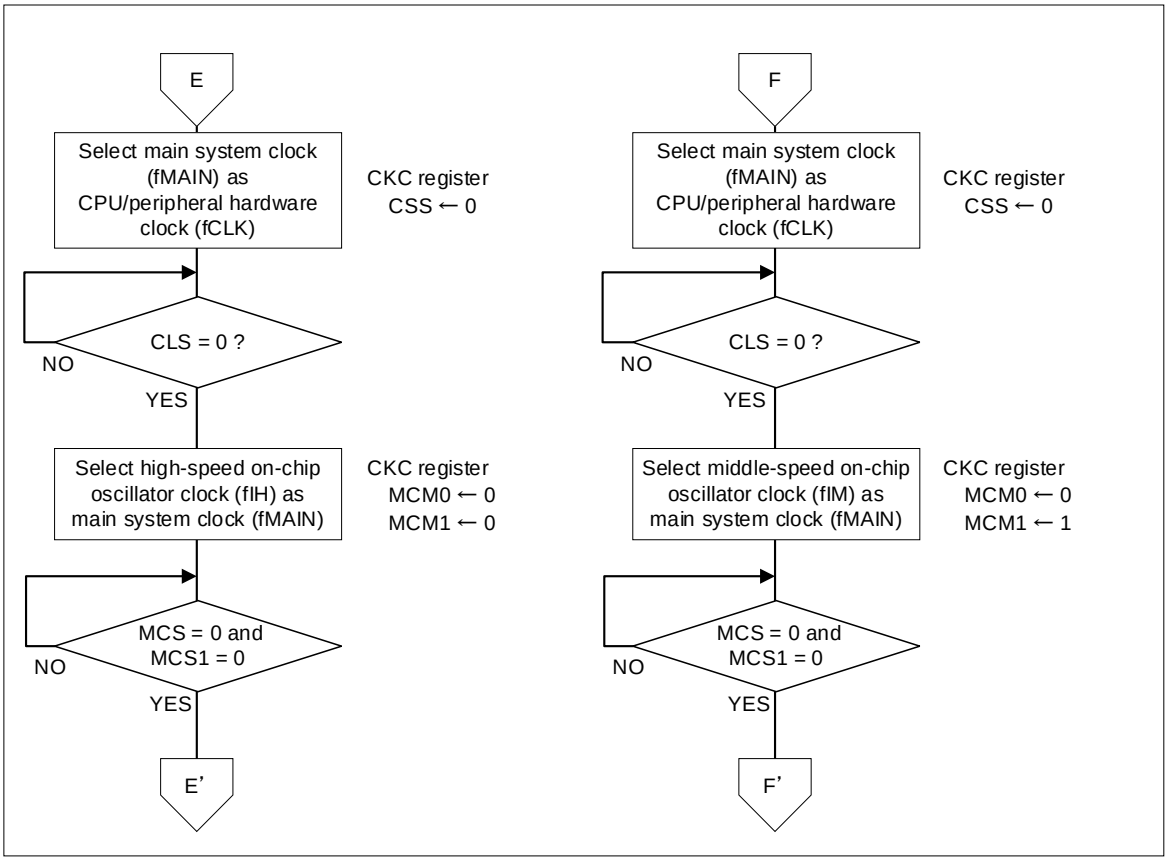


Figure 4-62 Clock Change Setting (3/4)

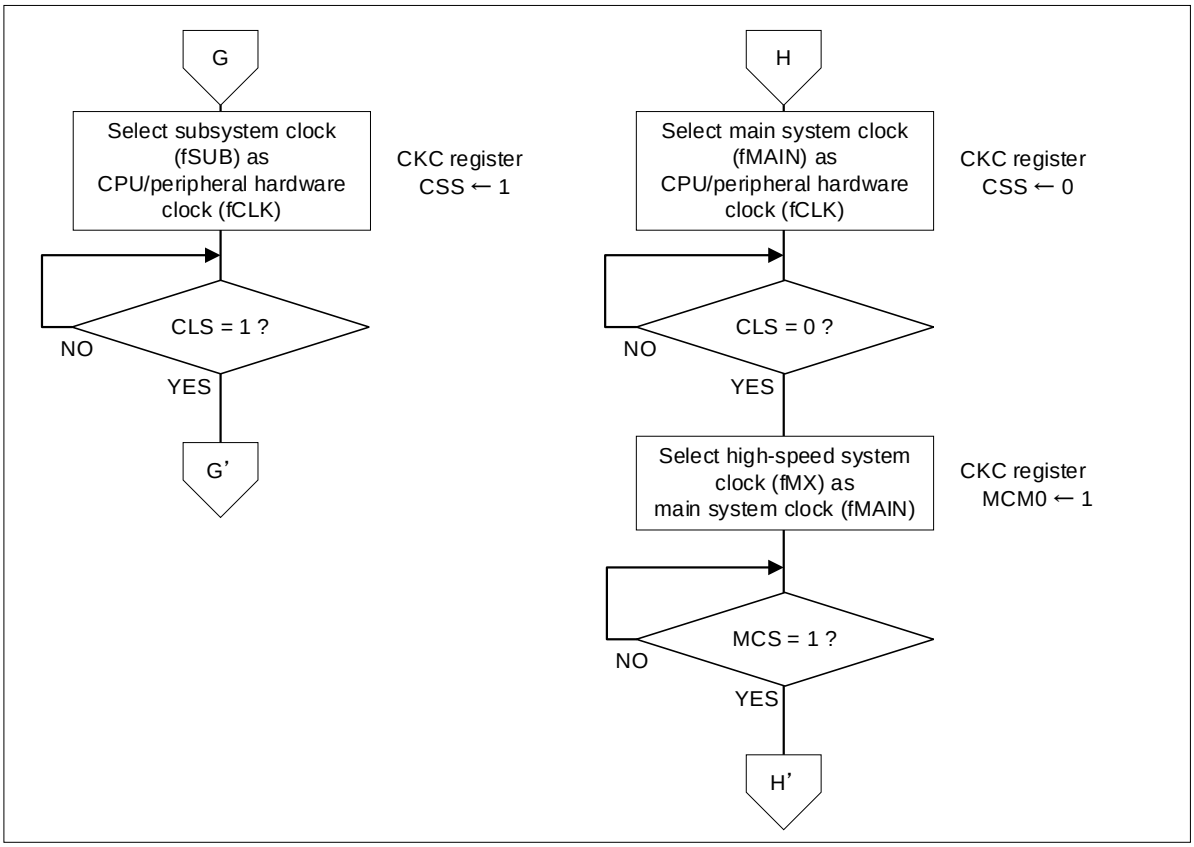
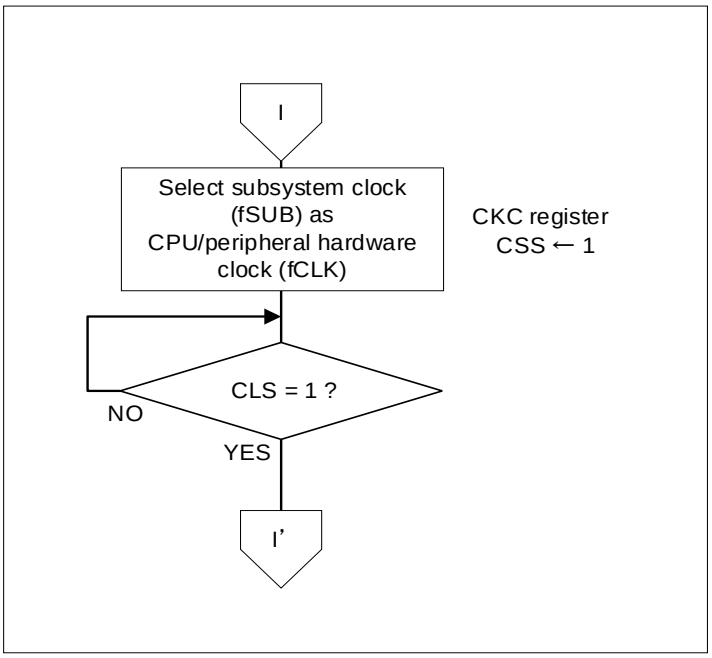


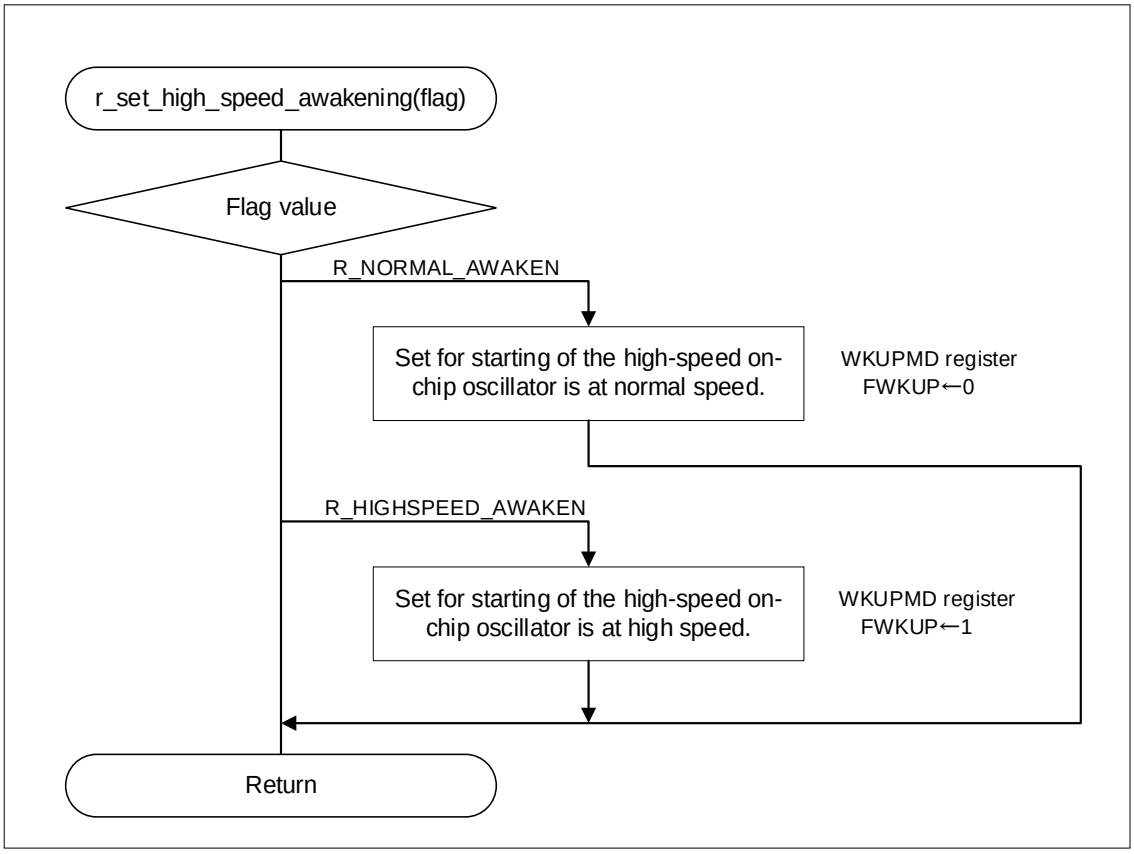
Figure 4-63 Clock Change Setting (4/4)



4.6.59 High-Speed On-Chip Oscillator Startup Setting

Figure 4-64 shows the flowchart of the High-speed on-chip oscillator startup setting.

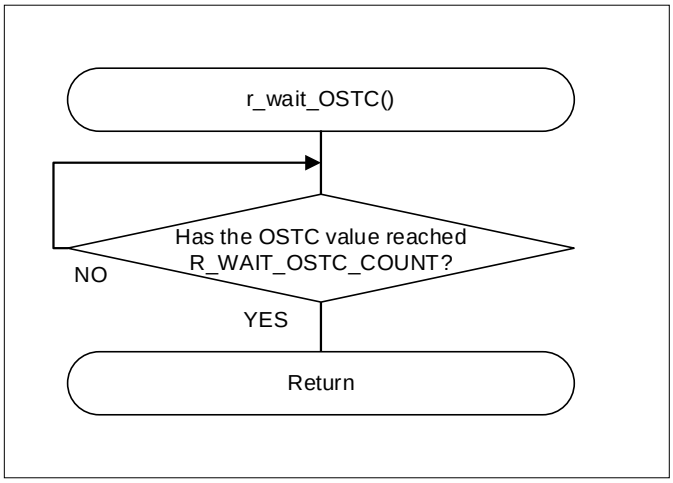
Figure 4-64 High-Speed On-Chip Oscillator Startup Setting



4.6.60 Wait Processing (OSTC register)

Figure 4-65 show flowcharts of the wait processing (OSTC register).

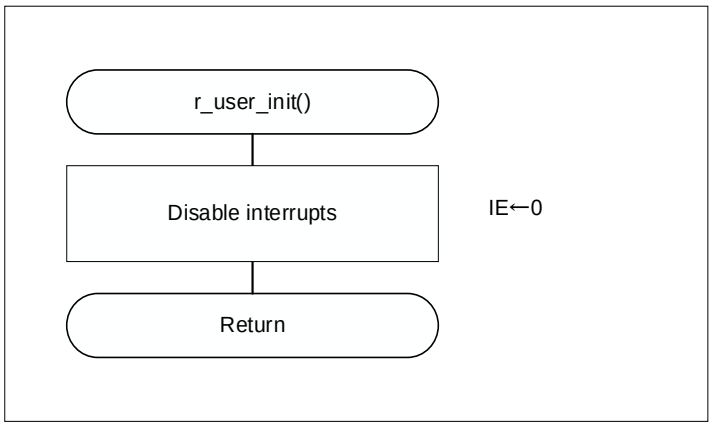
Figure 4-65 Wait Processing (OSTC register)



4.6.61 Main Initialization Processing (User Definitions)

Figure 4-66 shows the flowchart for main initialization processing (user definitions).

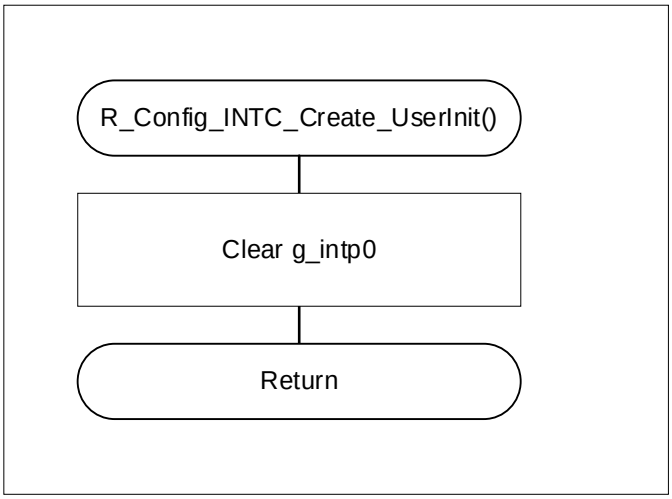
Figure 4-66 Main Initialization Processing (User Definitions)



4.6.62 Interrupt Controller Initialization Processing (User Definitions)

Figure 4-67 shows the flowchart for interrupt controller initialization processing (user definitions).

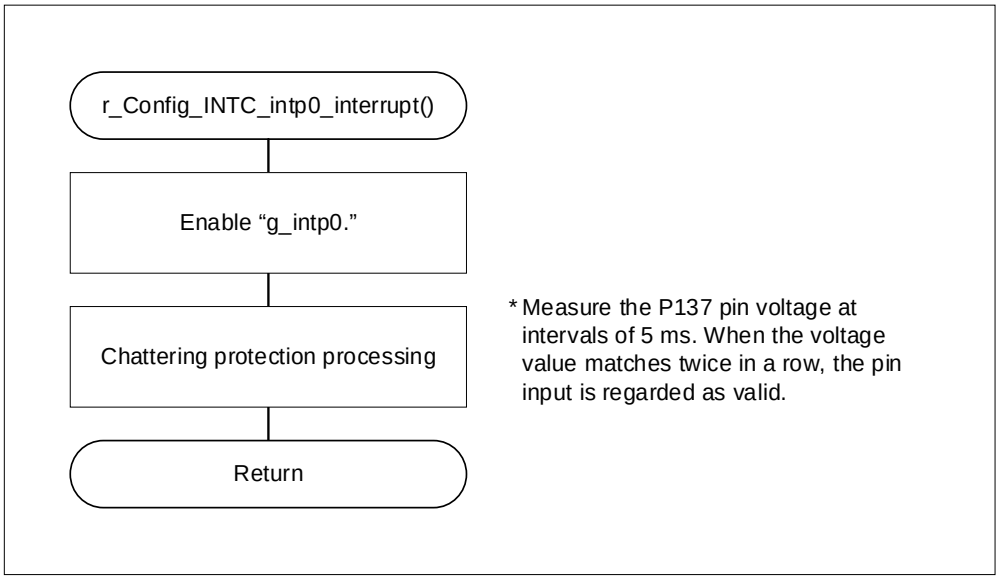
Figure 4-67 Interrupt Controller Initialization Processing (User Definitions)



4.6.63 External Interrupt (INTP0) Processing

Figure 4-68 show flowcharts of the external interrupt (INTP0) processing.

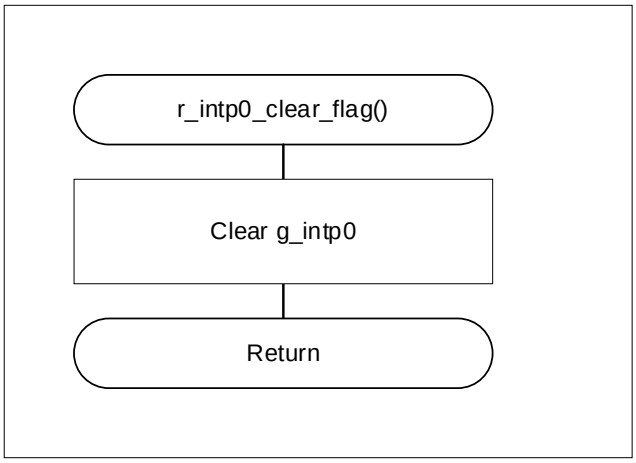
Figure 4-68 External Interrupt (INTP0) Processing



4.6.64 Clearing INTP0 Interrupt Flag

Figure 4-69 shows the flowchart for clearing the INTP0 interrupt flag.

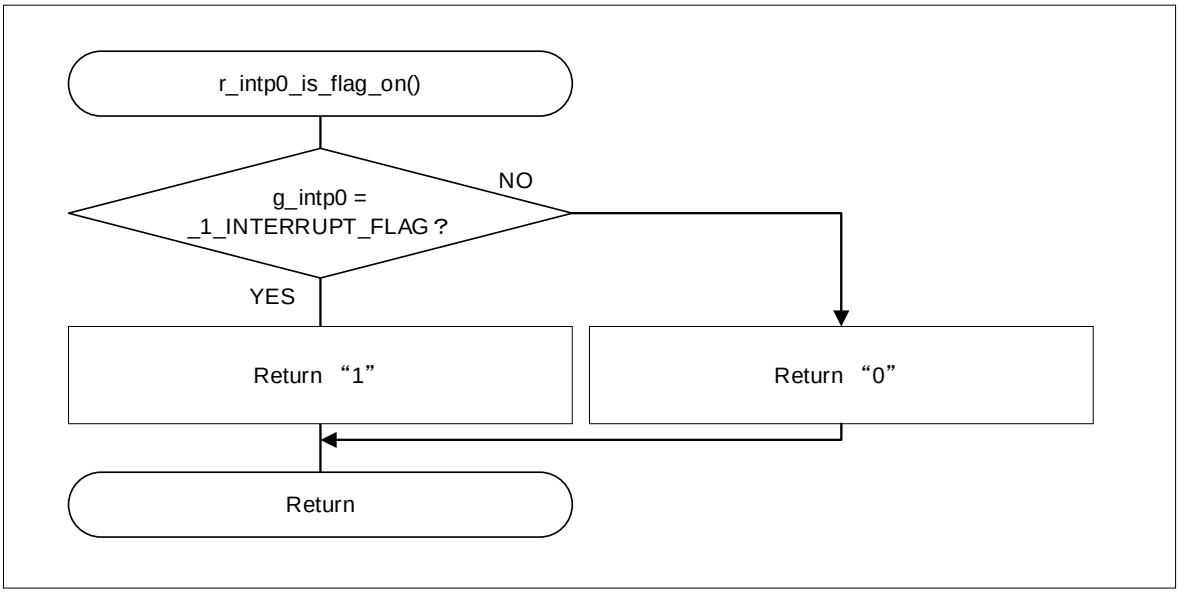
Figure 4-69 Clearing INTP0 Interrupt Flag



4.6.65 Checking INTP0 Interrupt Flag

Figure 4-70 shows the flowchart for checking the INTP0 interrupt flag.

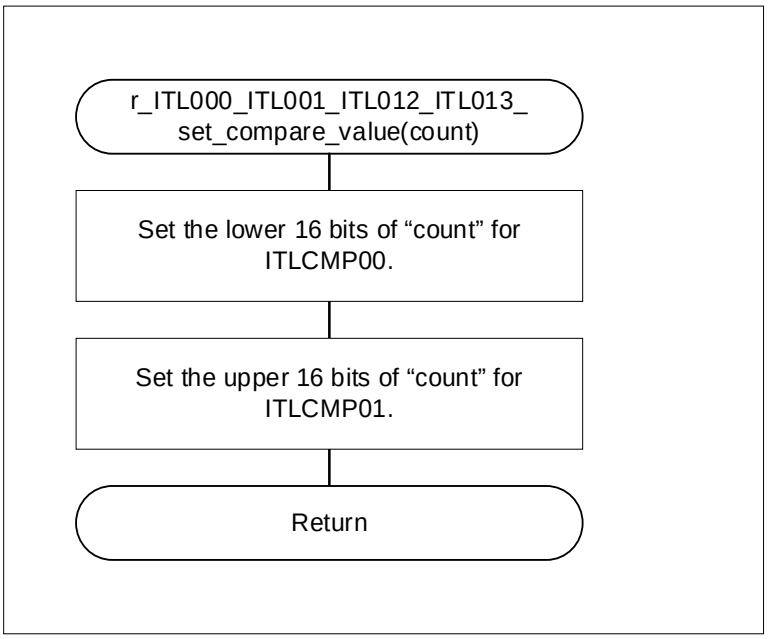
Figure 4-70 Checking INTP0 Interrupt Flag



4.6.66 Setting 32-Bit Interval Timer Compare Value

Figure 4-71 shows the flowchart for setting the 32-bit interval timer compare value.

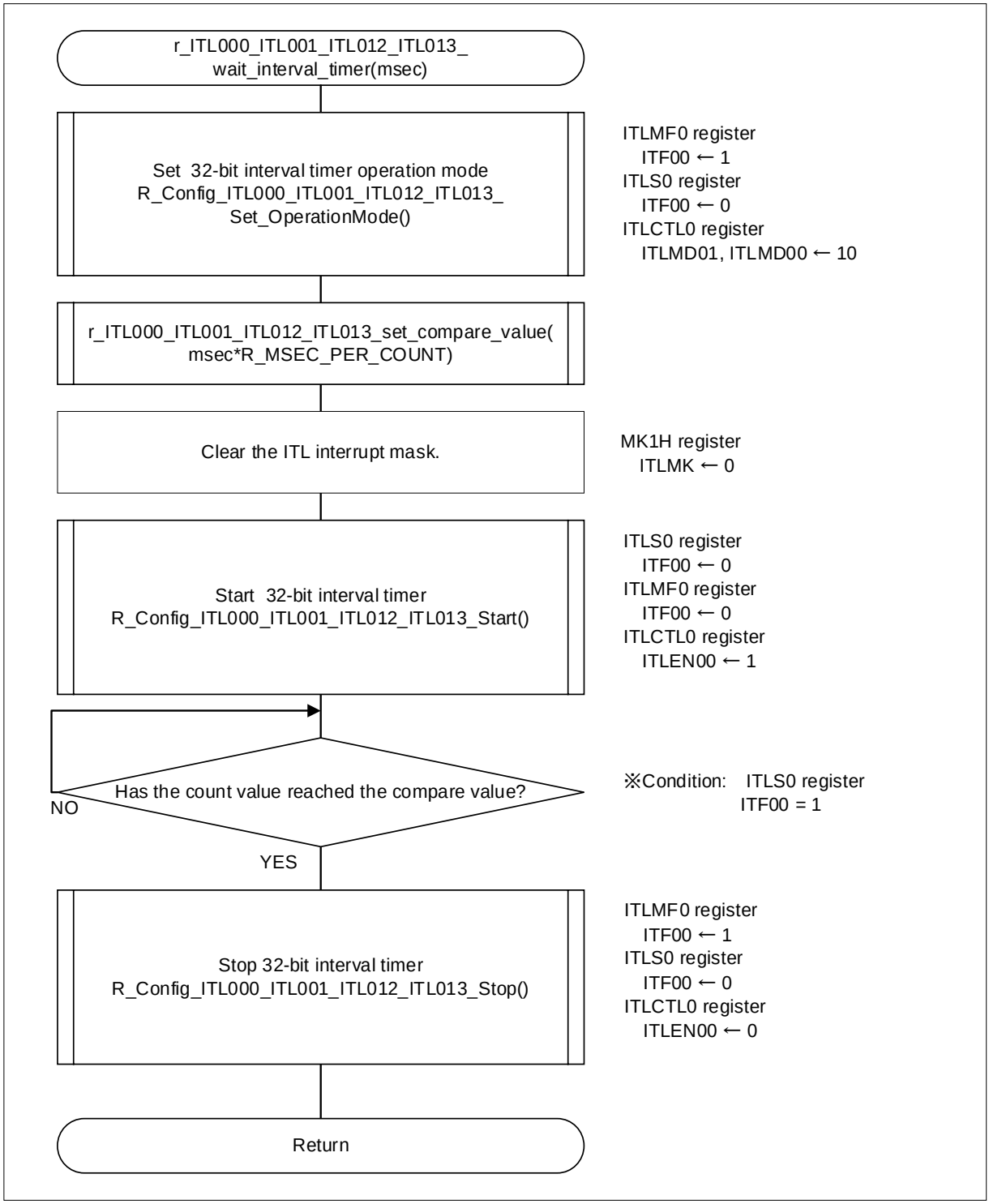
Figure 4-71 Setting 32-Bit Interval Timer Compare Value



4.6.67 Wait Processing (32-bit interval timer)

Figure 4-72 show flowcharts of the wait processing (32-bit interval timer).

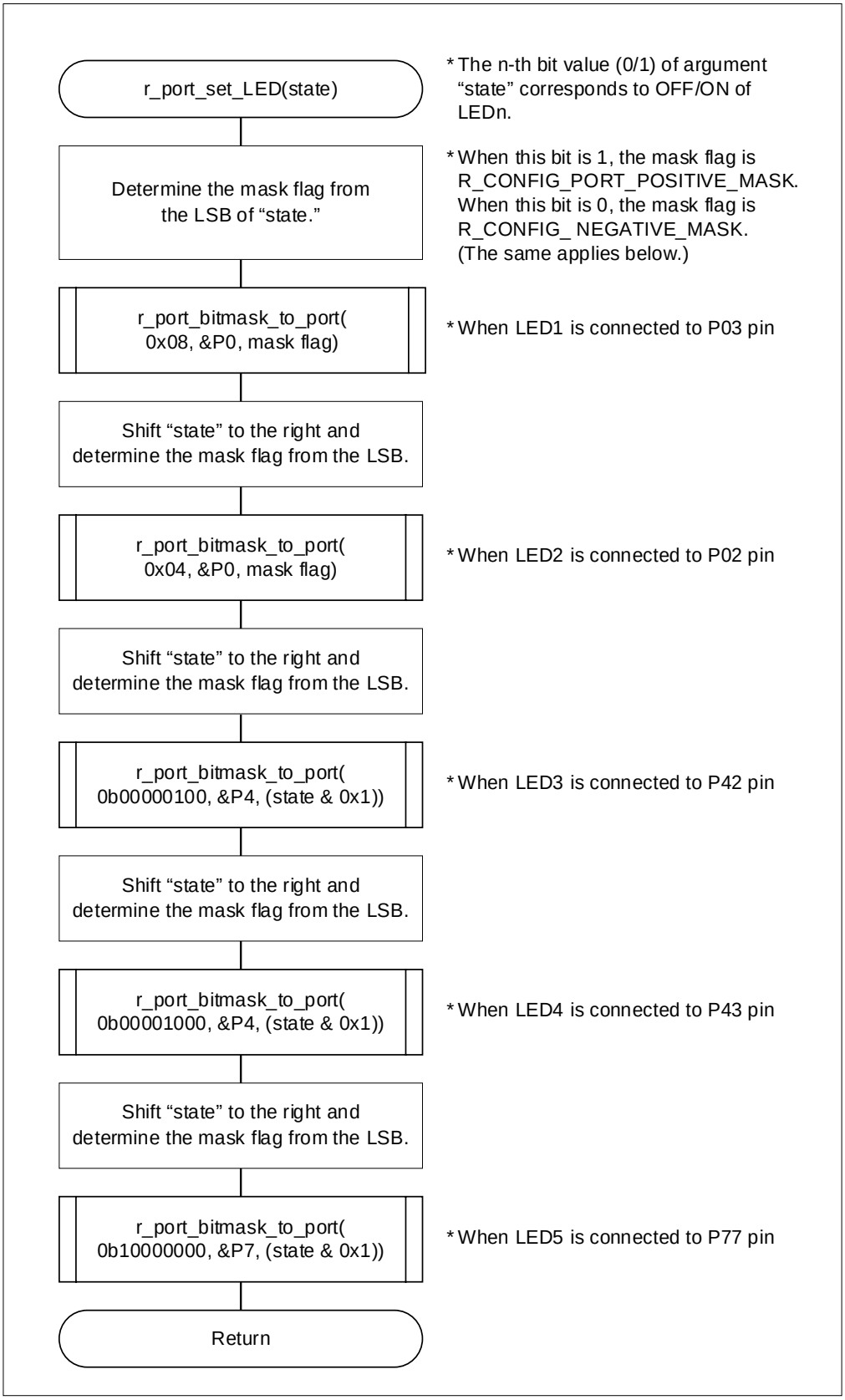
Figure 4-72 Wait Processing (32-bit interval timer)



4.6.68 LED ON/OFF Control

Figure 4-73 shows the flowchart for LED ON/OFF control.

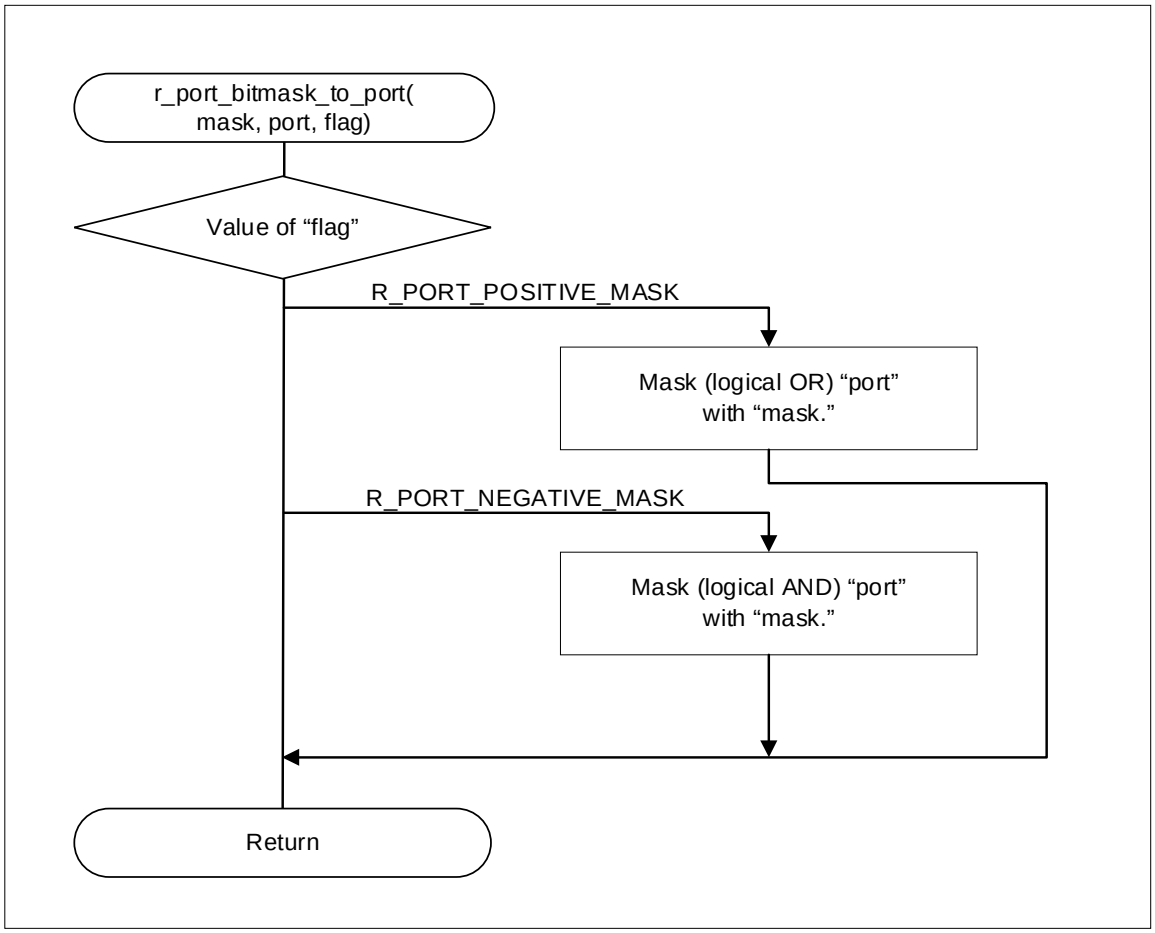
Figure 4-73 LED ON/OFF Control



4.6.69 Port Output Control

Figure 4-74 shows the flowchart for port output control.

Figure 4-74 Port Output Control



5. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

6. Reference Documents

RL78/G23 User’s Manual: Hardware (R01UH0896J)
RL78 family user's manual software (R01US0015J)
The latest versions can be downloaded from the Renesas Electronics website.

Technical update
The latest versions can be downloaded from the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	2021.04.13	—	First Edition
1.01	2021.07.12	37	Updated the Operation Confirmation Conditions
1.02	2023.10.6	37	Updated the Operation Confirmation Conditions

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

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

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

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

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.