R8C/M12A Group

Power Control in Stop Mode

R01AN0369EJ0101
Rev. 1.01
Aug. 29, 2011

Abstract

This document describes the setting method and an application example for power control using stop mode in the R8C/M12A Group.

Product

MCU: R8C/M12A Group

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. Specifications

After reset and while in standard operating mode, determine the signal level input from an external source every 10 ms and enter stop mode when the level is low. Input the falling edge to the $\overline{\text{INT0}}$ pin to return from stop mode. This document describes the three sample programs described below.

- Sample program 1
  Reset → Low-speed on-chip oscillator mode (no division) → High-speed clock mode (no division) → Stop mode → High-speed clock mode (no division)
- Sample program 2
  Reset → Low-speed on-chip oscillator mode (no division) → Low-speed on-chip oscillator mode (divided-by-8) → Stop mode → High-speed clock mode (divided-by-8) → High-speed clock mode (no division)
- Sample program 3
  Reset → Low-speed on-chip oscillator mode (no division) → Low-speed on-chip oscillator mode (divided-by-8) → Stop mode → Low-speed on-chip oscillator mode (divided-by-8) → Low-speed on-chip oscillator mode (no division)

Table 1.1 lists the Peripheral Functions and Their Applications (Sample Programs 1 to 3). Figures 1.1 to 1.3 show Usage Examples.

Table 1.1 Peripheral Functions and Their Applications (Sample Programs 1 to 3)

Peripheral Function	Application
$\overline{\text{INT0}}$ interrupt	Return from stop mode
Timer RJ2	10 ms period timer

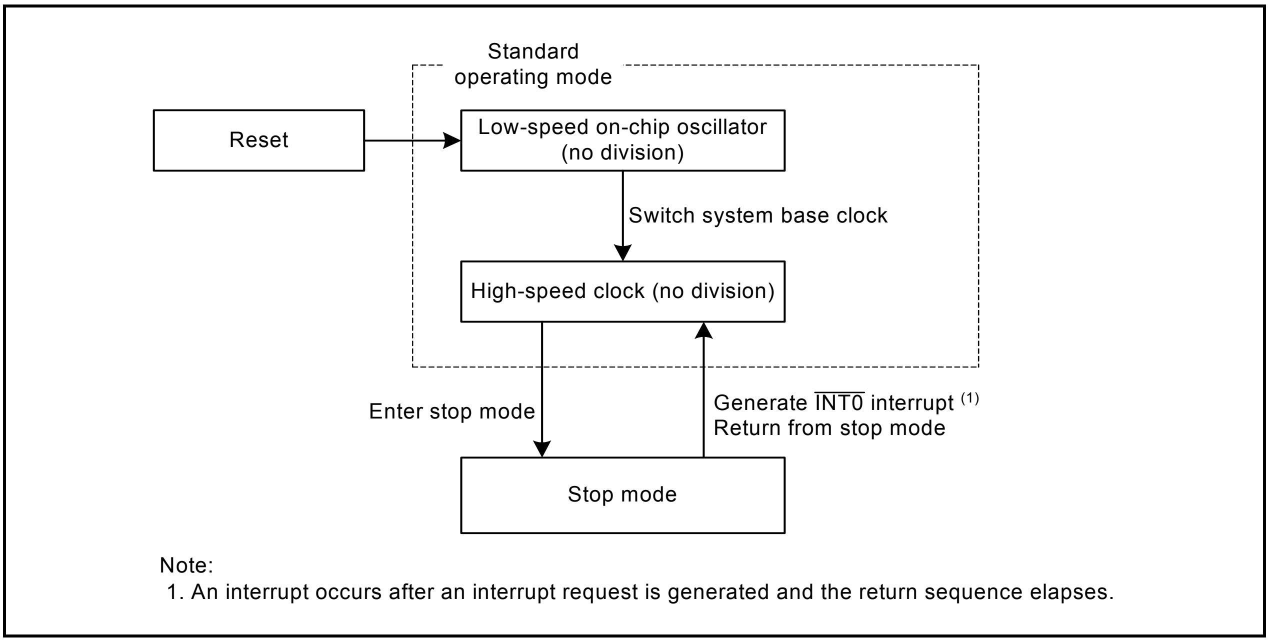


Figure 1.1 Usage Example for Sample Program 1

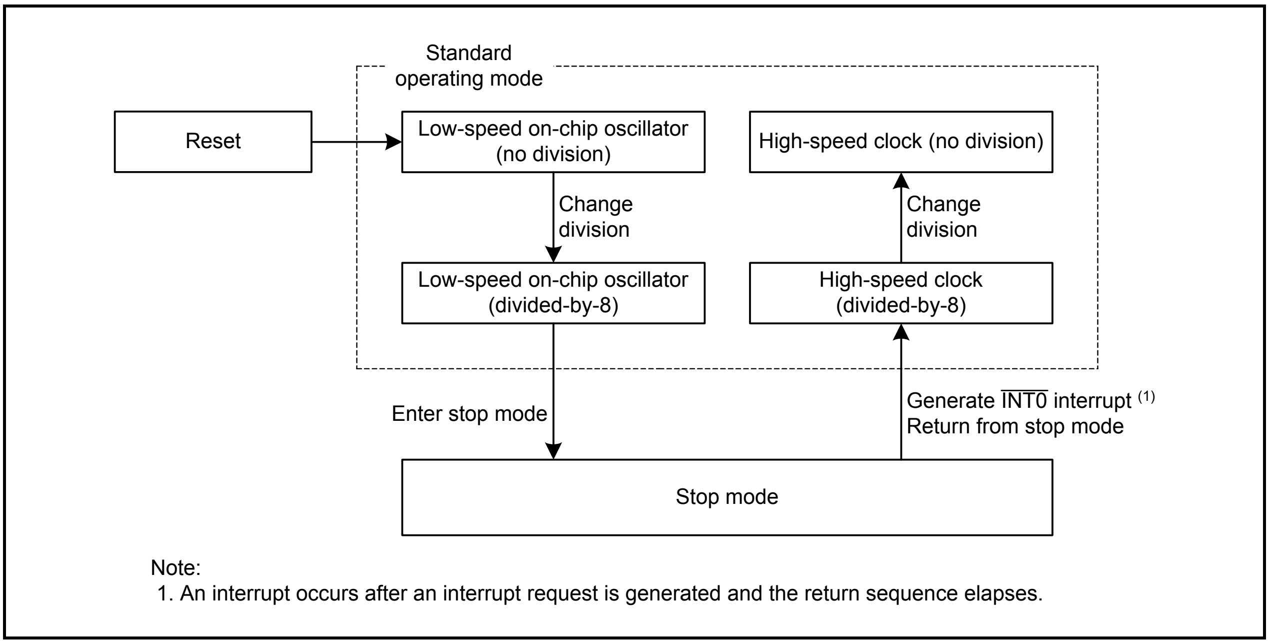
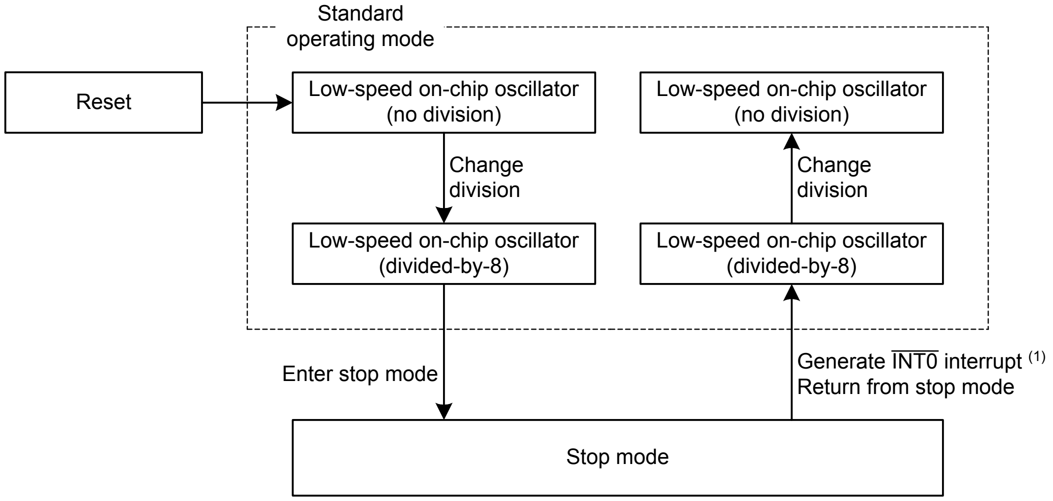


Figure 1.2 Usage Example for Sample Program 2



Note:
1. An interrupt occurs after an interrupt request is generated and the return sequence elapses.

Figure 1.3 Usage Example for Sample Program 3

2. Operation Confirmation Conditions

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

Table 2.1 Operation Confirmation Conditions

Item	Contents
MCU used	R8C/M12A Group
Operating frequencies	Sample program 1 • XIN clock: 20 MHz • System clock (f) (before entering stop mode): 20 MHz • System clock (f) (after returning from stop mode): 20 MHz • CPU clock (fs) (before entering stop mode): 20 MHz • CPU clock (fs) (after returning from stop mode): 20 MHz Sample program 2 • Low-speed on-chip oscillator clock: 125 kHz (typical) • XIN clock: 20 MHz • System clock (f) (before entering stop mode): 125 kHz (typical) • System clock (f) (after returning from stop mode): 20 MHz • CPU clock (fs) (before entering stop mode): 15.625 kHz (typical) • CPU clock (fs) (after returning from stop mode): 2.5 MHz • CPU clock (fs) (after changing division): 20 MHz Sample program 3 • Low-speed on-chip oscillator clock: 125 kHz (typical) • System clock (f) (before entering stop mode): 125 kHz (typical) • System clock (f) (after returning from stop mode): 125 kHz • CPU clock (fs) (before entering stop mode): 15.625 kHz (typical) • CPU clock (fs) (after returning from stop mode): 15.625 kHz (typical) • CPU clock (fs) (after changing division): 125 kHz (typical)
Operating voltage	5.0 V (2.7 to 5.5 V)
Integrated development environment	Renesas Electronics Corporation High-performance Embedded Workshop Version 4.07
C compiler	Renesas Electronics Corporation M16C Series, R8C Family C Compiler V.5.45 Release 01
	Compile options -D__UART0__ -c -finfo -dir "$(CONFIGDIR)" -R8C (Default setting is used in the integrated development environment.)

3. Hardware

3.1 Pins Used

Table 3.1 lists the Pins Used and Their Functions.

Table 3.1 Pins Used and Their Functions

Pin Name	I/O	Function
P1_4/$\overline{\text{INT0}}$	Input	$\overline{\text{INT0}}$ interrupt input
P1_7	Input	CMOS I/O port

4. Software for Sample Program 1

4.1 Operation Overview of Sample Program 1

After reset, the MCU enters high-speed clock mode (no division) from low-speed on-chip oscillator mode (no division) by a program. Then the timer RJ2 count set to a 10 ms period starts after entering high-speed clock mode. Read the P1_7 pin every 10 ms and determine if the MCU enters stop mode. When the state is held low three times consecutively, write 1 (stop mode) to the variable (mode). Disable maskable interrupts, enable the $\overline{\text{INT0}}$ interrupt input used to return from stop mode, disable CPU rewrite mode, disable the oscillation stop detection function, and set a clock after returning from stop mode. Then, enable maskable interrupts, set the STPM bit in the CKSTPR register to 1 (all clocks stop (stop mode)) to enter stop mode.

When applying the falling edge to the $\overline{\text{INT0}}$ pin, the MCU returns from stop mode. High-speed clock mode (no division) is automatically selected for the operating mode when returning from stop mode.

Disable $\overline{\text{INT0}}$ interrupt input, write 0 (standard operating mode) to the variable (mode), and return to the reading process of the P1_7 pin.

(1) Oscillate the XIN clock by a program after reset.
(2) After the XIN clock oscillation stabilizes, switch the system base clock from low-speed on-chip oscillator to the XIN clock to enter high-speed clock mode (no division).
(3) Start counting timer RJ2. After the count starts, read the P1_7 pin every 10 ms.
(4) Read the P1_7 pin as a low level three times consecutively, disable maskable interrupts, and perform the settings described below.

Settings
- Enable the $\overline{\text{INT0}}$ interrupt input.
- Disable CPU rewrite mode.
- Disable oscillation stop detection.
- Set wait states.
- Set the PHISRS bit in the CKRSCR register to 0 (setting values of bits PHISSEL0 to PHISSEL2 in the SCKCR register are enabled).
- Set the STOPRS bit in the CKRSCR register to 0 (return from stop mode using the system base clock immediately before entering stop mode).

(5) Enable maskable interrupts, set the STPM bit in the CKSTPR register to 1 to enter stop mode.
(6) Return from stop mode using the $\overline{\text{INT0}}$ interrupt (falling edge signal). Clocks set in (4) are selected for the CPU clock when returning from stop mode. Initialize the count value of timer RJ2. Set 0 (standard operating mode) to the variable (mode) to return to the main processing.
(7) Repeat steps (3) to (6).

Figure 4.1 shows the Stop Mode Operating Example.

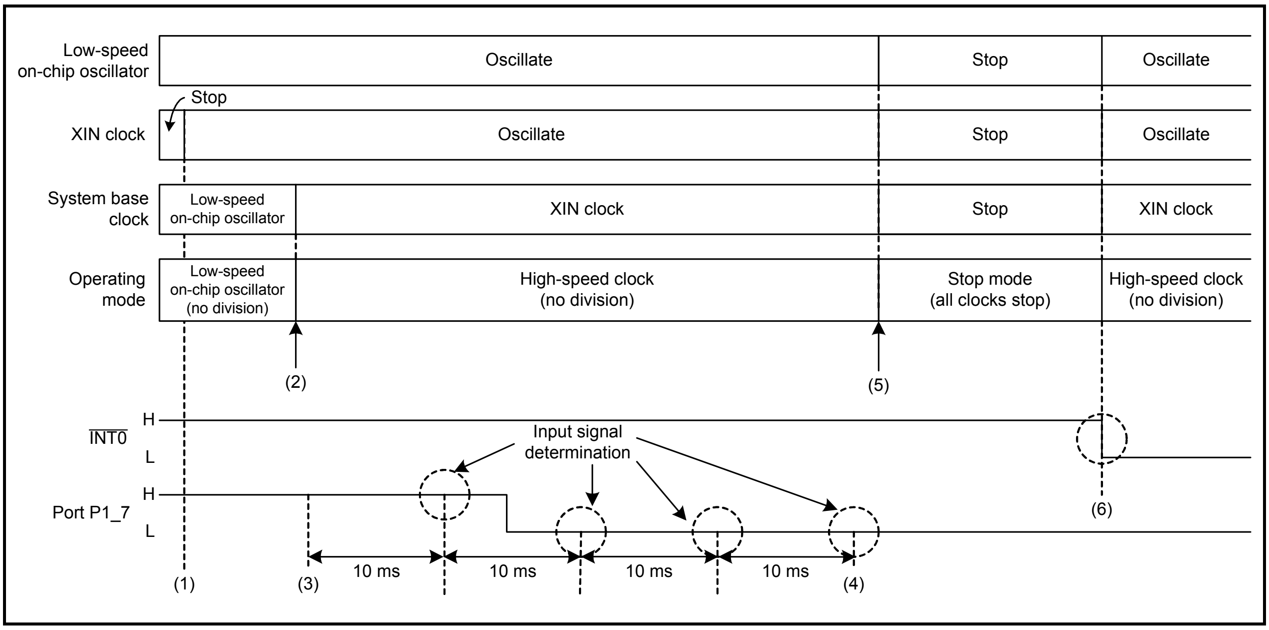


Figure 4.1 Stop Mode Operating Example

4.2 Required Memory Size

Table 4.1 lists the Required Memory Size.

Table 4.1 Required Memory Size for Sample Program 1

Memory Used	Size	Remarks
ROM	403 bytes	In the r01an0369_src_sample1.c module
RAM	2 bytes	In the r01an0369_src_sample1.c module
Maximum user stack usage	10 bytes	
Maximum interrupt stack usage	18 bytes	

The required memory size varies depending on the C compiler version and compile options.

4.3 Constant

Table 4.2 lists the Constant Used in the Sample Code.

Table 4.2 Constant Used in the Sample Code

Constant Name	Setting Value	Contents
LOW	0	Port P1_7 input level low

4.4 Variables

Table 4.3 lists the Global Variable, and Table 4.4 lists the static Variable.

Table 4.3 Global Variable

Type	Variable Name	Contents	Function Used
unsigned char	mode	Select to enter stop mode	stop_signal_in, power_control

Table 4.4 static Variable

Type	Variable Name	Contents	Function Used
static unsigned char	stp_sig_bit	Input information for stop mode enter signal	stop_signal_in

4.5 Functions

Table 4.5 lists the Functions.

Table 4.5 Functions

Function Name	Outline
mcu_init	System clock setting
int0_init	Initial setting of $\overline{\text{INT0}}$ interrupt
timer_rj2_init	Initial setting of timer RJ2
stop_signal_in	Input processing of stop mode enter signal
power_control	Stop mode processing
_int0	$\overline{\text{INT0}}$ interrupt handling

4.6 Function Specifications

The following tables list the sample code function specifications.

mcu_init	
Outline	System clock setting
Header	None
Declaration	void mcu_init(void)
Description	Set the system clock.
Argument	None
Returned value	None
Remark	—

int0_init	
Outline	Initial setting of $\overline{\text{INT0}}$ interrupt
Header	None
Declaration	void int0_init(void)
Description	Perform initial setting to use the $\overline{\text{INT0}}$ interrupt.
Argument	None
Returned value	None
Remark	—

timer_rj2_init	
Outline	Initial setting of timer RJ2
Header	None
Declaration	void timer_rj2_init(void)
Description	Perform initial setting to use timer RJ2 in timer mode.
Argument	None
Returned value	None
Remark	—

stop_signal_in	
Outline	Input processing of stop mode enter signal
Header	None
Declaration	void stop_signal_in(void)
Description	Perform stop mode enter determination.
Argument	None
Returned value	None
Remark	—

power_control	
Outline	Stop mode processing
Header	None
Declaration	void power_control(void)
Description	Enter stop mode.
Argument	None
Returned value	None
Remark	—

_int0	
Outline	$\overline{\text{INT0}}$ interrupt handling
Header	None
Declaration	void _int0(void)
Description	Perform $\overline{\text{INT0}}$ interrupt handling
Argument	None
Returned value	None
Remark	—

4.7 Flowcharts

4.7.1 Main Processing

Figure 4.2 shows the Main Processing.

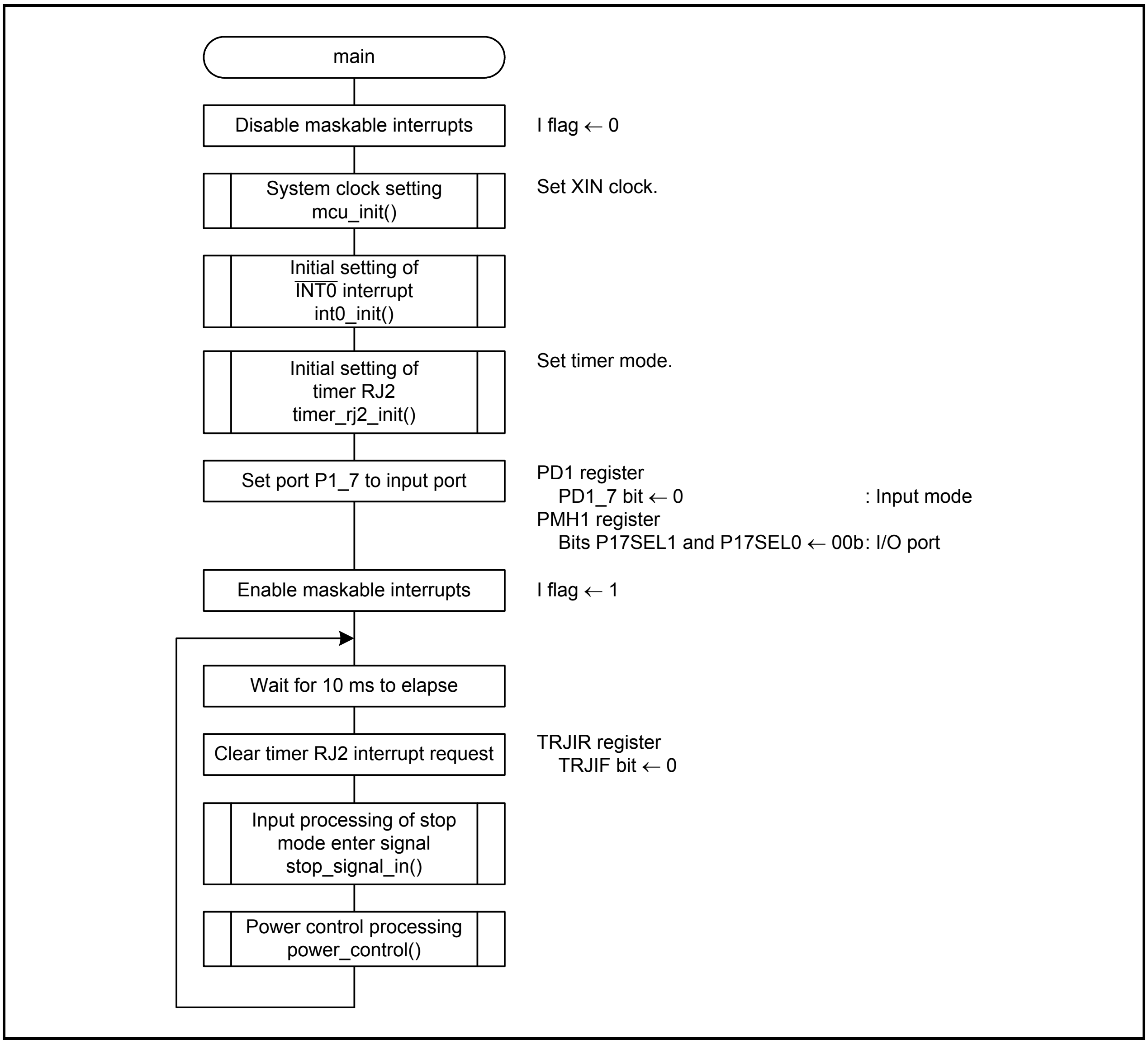


Figure 4.2 Main Processing

4.7.2 System Clock Setting

Figure 4.3 shows the System Clock Setting.

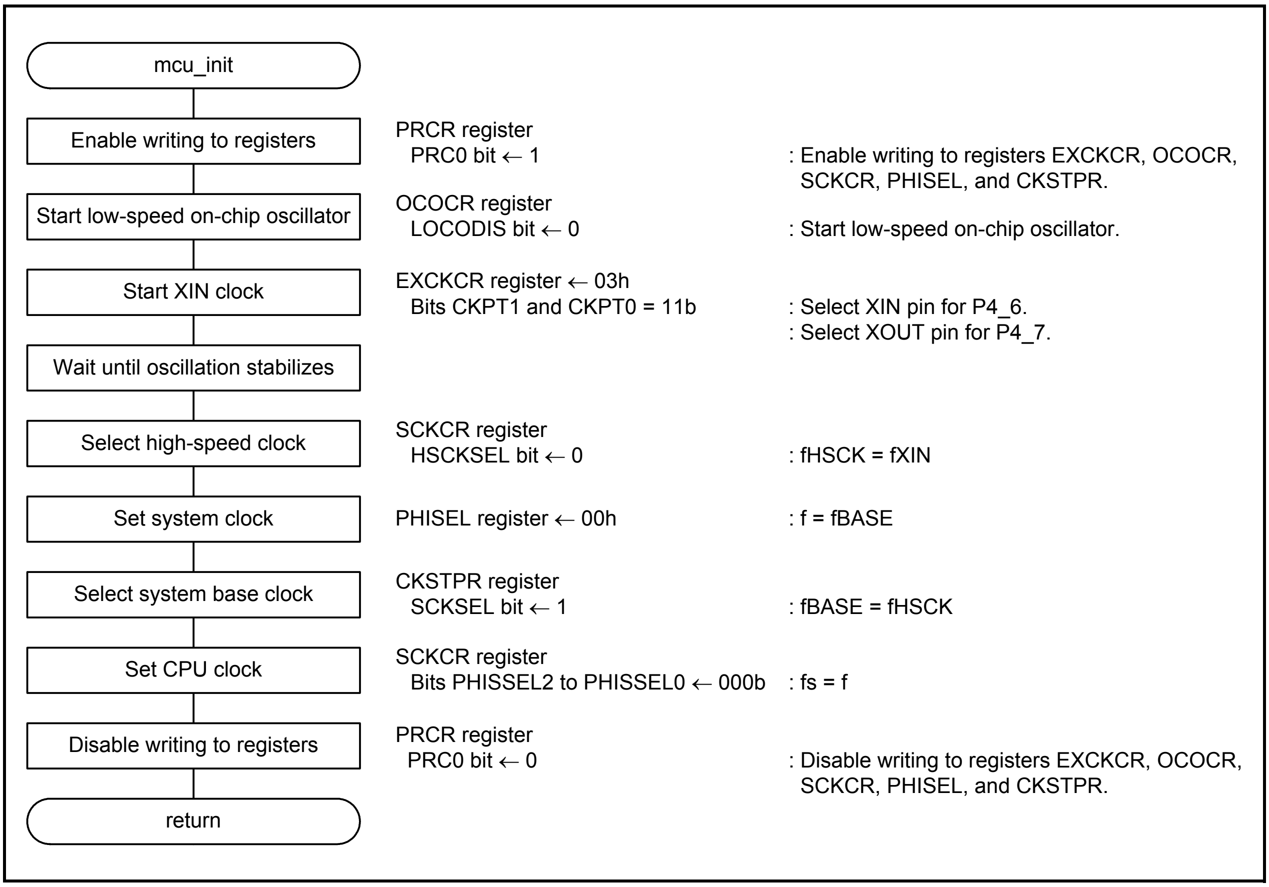


Figure 4.3 System Clock Setting

4.7.3 Initial Setting of the $\overline{\mathrm{INT0}}$ Interrupt

Figure 4.4 shows the Initial Setting of $\overline{\mathrm{INT0}}$ Interrupt.

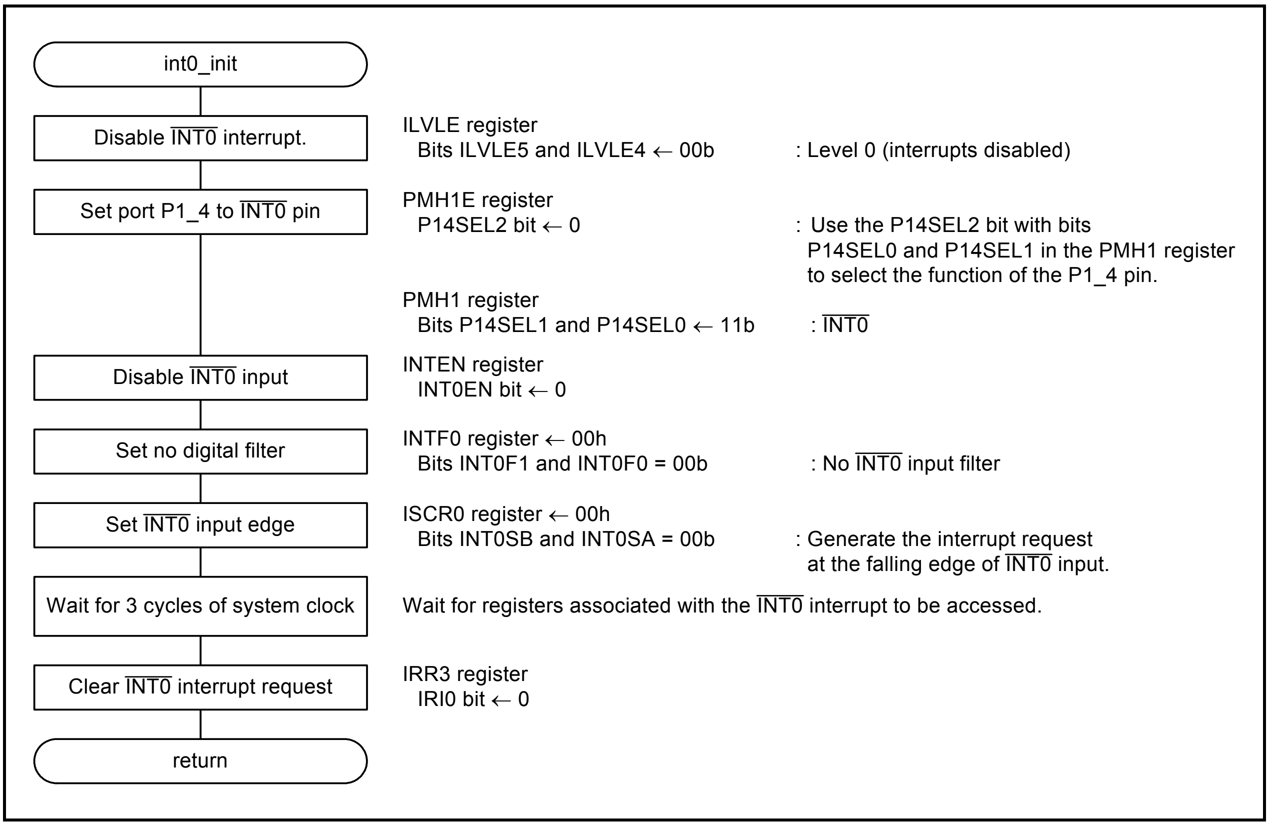


Figure 4.4 Initial Setting of the $\overline{\mathrm{INT0}}$ Interrupt

4.7.4 Initial Setting of Timer RJ2

Figure 4.5 shows the Initial Setting of Timer RJ2.

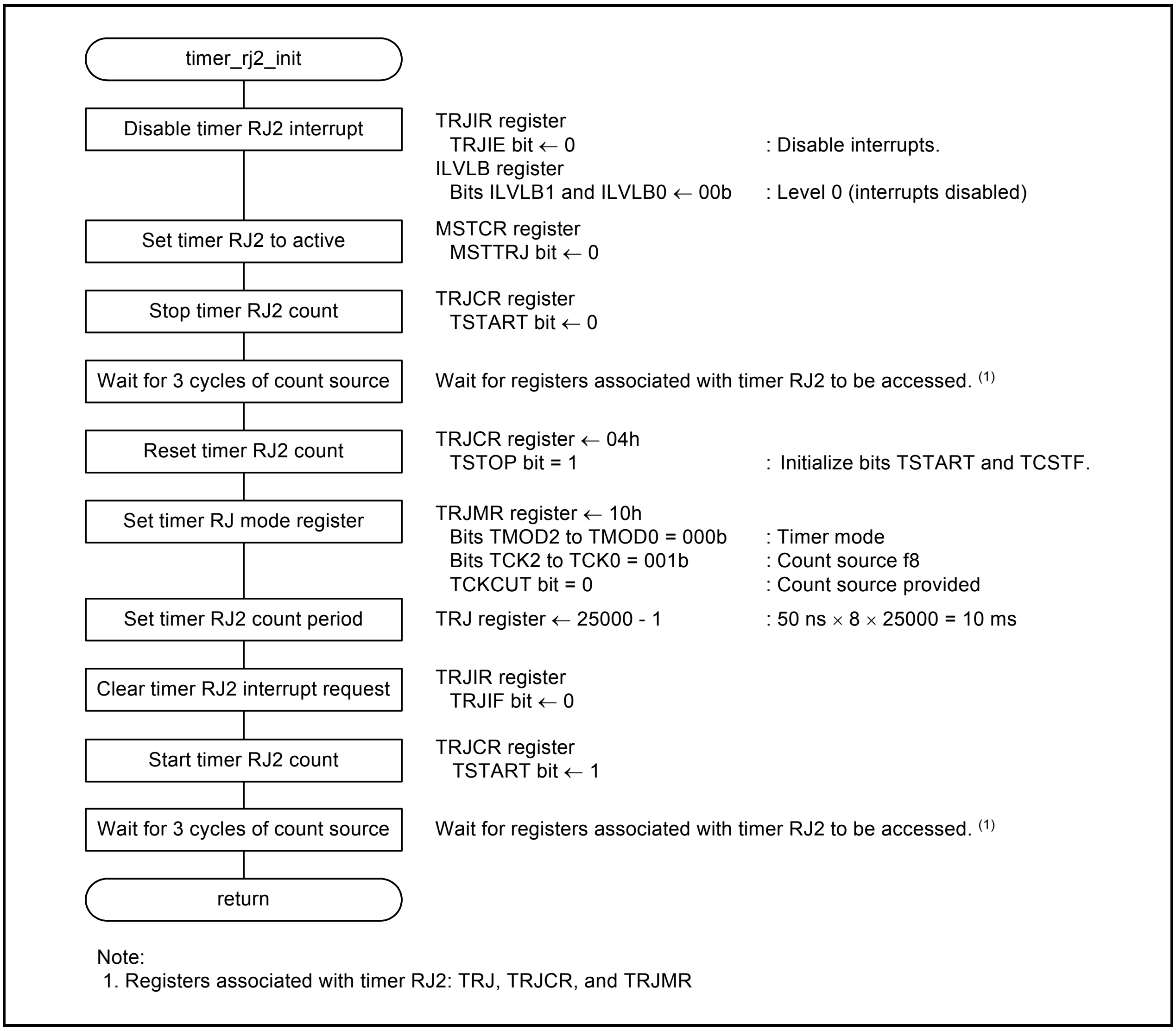


Figure 4.5 Initial Setting of Timer RJ2

4.7.5 Input Processing of Stop Mode Enter Signal

Figure 4.6 shows the Input Processing of Stop Mode Enter Signal.

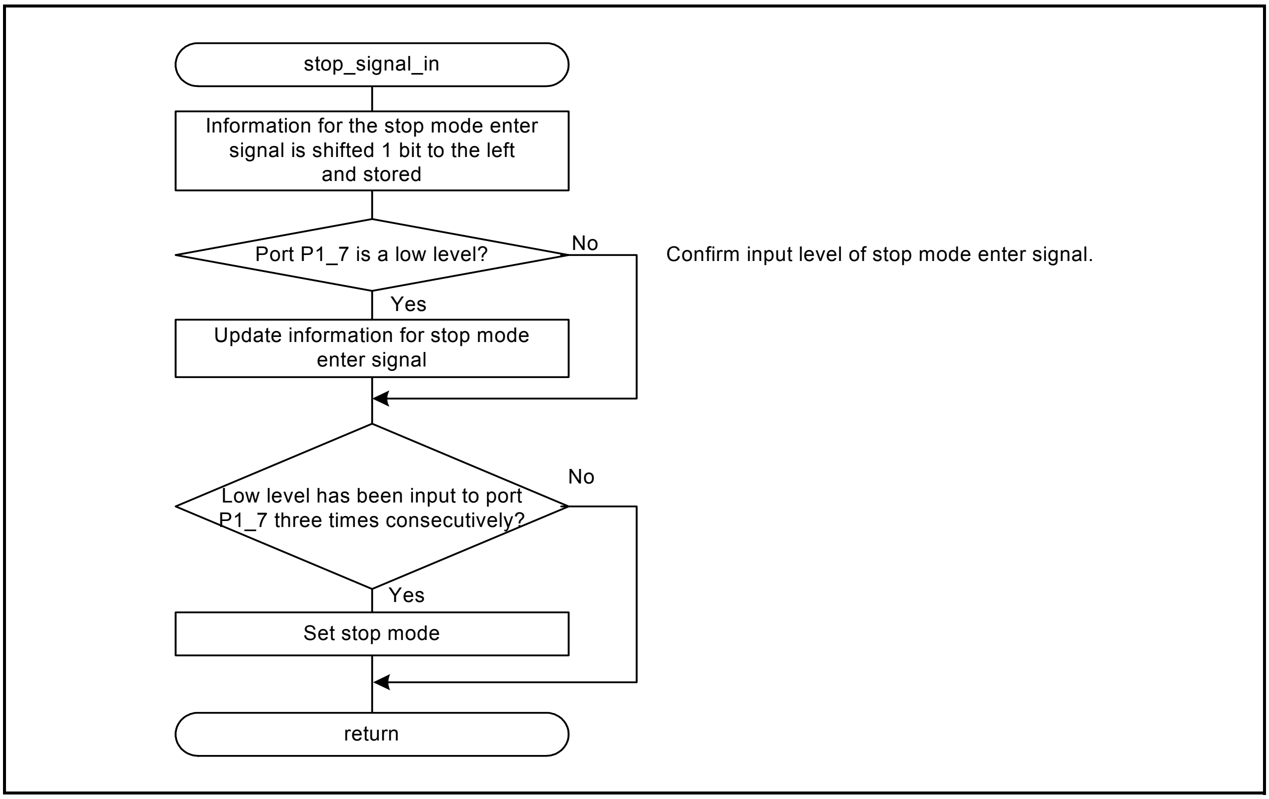


Figure 4.6 Input Processing of Stop Mode Enter Signal

4.7.6 Stop Mode Processing

Figures 4.7 and 4.8 show the Stop Mode Processing.

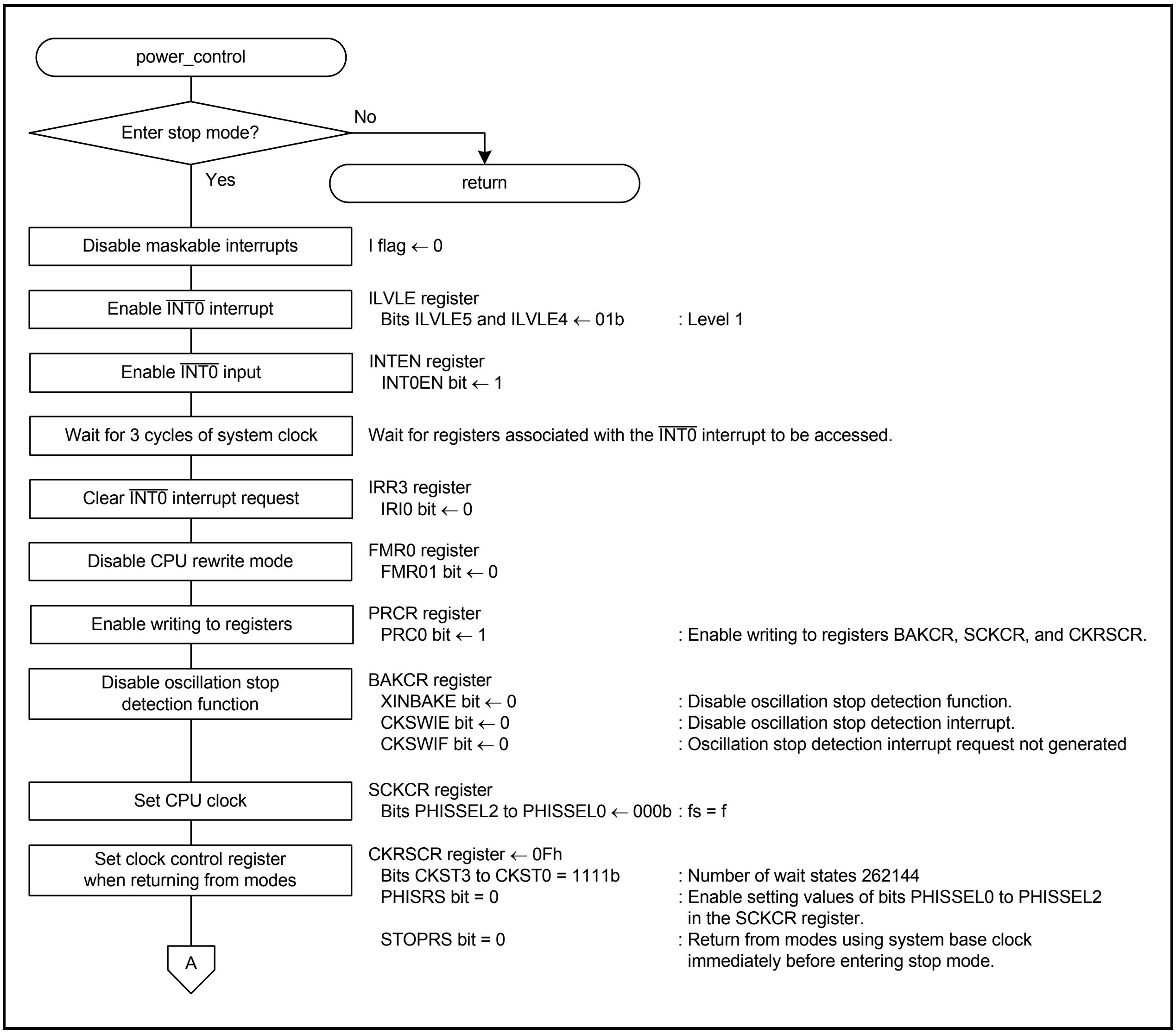
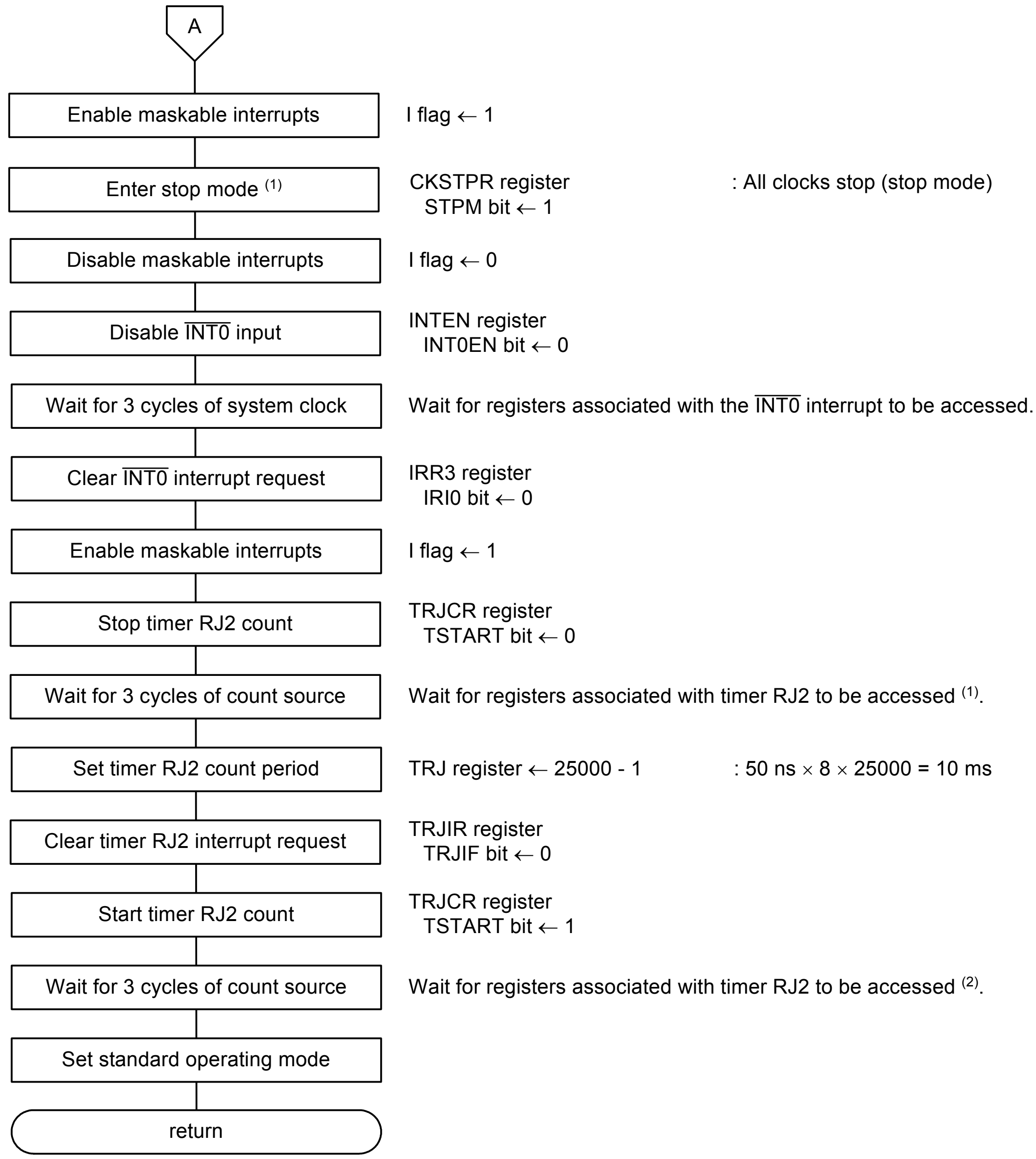


Figure 4.7 Stop Mode Processing (1/2)



Notes:
1. Since the 4 bytes of instruction data following the instruction that sets the STPM bit to 1 are prefetched and then the program stops, insert at least four NOP instructions following the JMP.B instruction immediately after the instruction that sets the STPM bit to 1 as a program restriction when entering wait mode.
2. Registers associated with timer RJ2: TRJ, TRJCR, TRJIOC, and TRJMR

Figure 4.8 Stop Mode Processing (2/2)

4.7.7 INT0 Interrupt Handling

Figure 4.9 shows the INT0 Interrupt Handling.

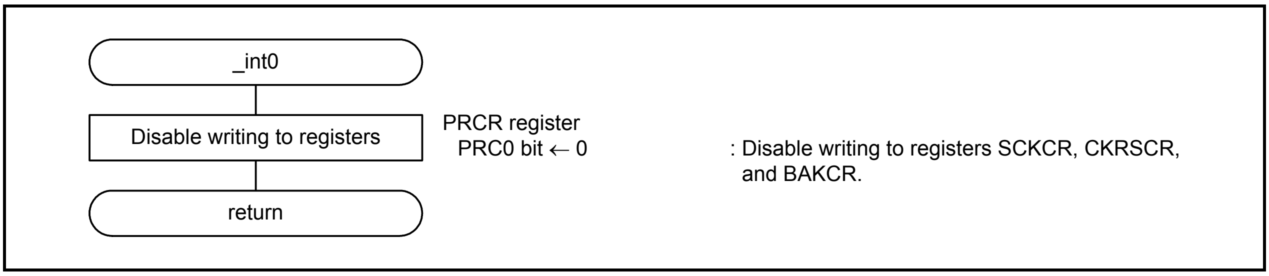


Figure 4.9 INT0 Interrupt Handling

5. Software for Sample Program 2

5.1 Operation Overview of Sample Program 2

After reset, the MCU enters low-speed on-chip oscillator mode (divided-by-8) from low-speed on-chip oscillator mode (no division) by a program. Set the interval of refresh operations in low-current-consumption read mode to the FREFR register, and set the FMR27 bit in the FMR2 register to 1 (low-current-consumption read mode enabled) to use low-current-consumption read mode. Then the timer RJ2 count set to a 10 ms period starts. Read the P1_7 pin every 10 ms and determine if the MCU enters stop mode. When the state is held low three times consecutively, write 1 (stop mode) to the variable (mode). Disable maskable interrupts, enable the $\overline{\text{INT0}}$ interrupt input used to return from stop mode, disable CPU rewrite mode, disable the oscillation stop detection function, set a clock after returning from stop mode, and disable low-current-consumption read mode. Then enable maskable interrupts, set the STPM bit in the CKSTPR register to 1 (all clocks stop (stop mode)) to enter stop mode.

When applying the falling edge to the $\overline{\text{INT0}}$ pin, the MCU returns from stop mode. High-speed clock mode (divided-by-8) is automatically selected for the operating mode when returning from stop mode. Then set the opeating mode to high-speed clock mode (no division) by a program.

(1) Set low-speed on-chip oscillator mode (divided-by-8) by a program after reset.
(2) After setting the interval of refresh operations in low-current-consumption read mode, set the FMR27 bit in the FMR2 register to 1 (low-current-consumption read mode enabled).
(3) Start counting timer RJ2. After the count starts, read the P1_7 pin every 10 ms.
(4) Read the P1_7 pin as a low level three times consecutively, disable maskable interrupts, and perform the settings described below.

Settings
- Enable the INT0 interrupt input.
- Disable CPU rewrite mode.
- Disable oscillation stop detection.
- Set wait states.
- Set the PHISRS bit in the CKRSCR register to 0 (setting values of bits PHISSEL0 to PHISSEL2 in the SCKCR register are enabled).
- Set the STOPRS bit in the CKRSCR register to 1 (fHSCK).
- Set bits CKPT1 and CKT0 in the EXCKCR register to 11b (P4_6: XIN, P4_7: XOUT).
- Set the SCKSEL bit in the CKSTPR register to 1 (fHSCK).
- Set the FMR27 bit to 0 (low-current-consumption read mode disabled).

(5) Enable maskable interrupts, set the STPM bit in the CKSTPR register to 1 to enter stop mode.
(6) Return from stop mode using the INT0 interrupt (falling edge signal). Clocks set in (4) are selected for the CPU clock when returning from stop mode.
(7) Set the operating mode to high-speed clock mode (no division) by a program.

Figure 5.1 shows the Stop Mode Operating Example.

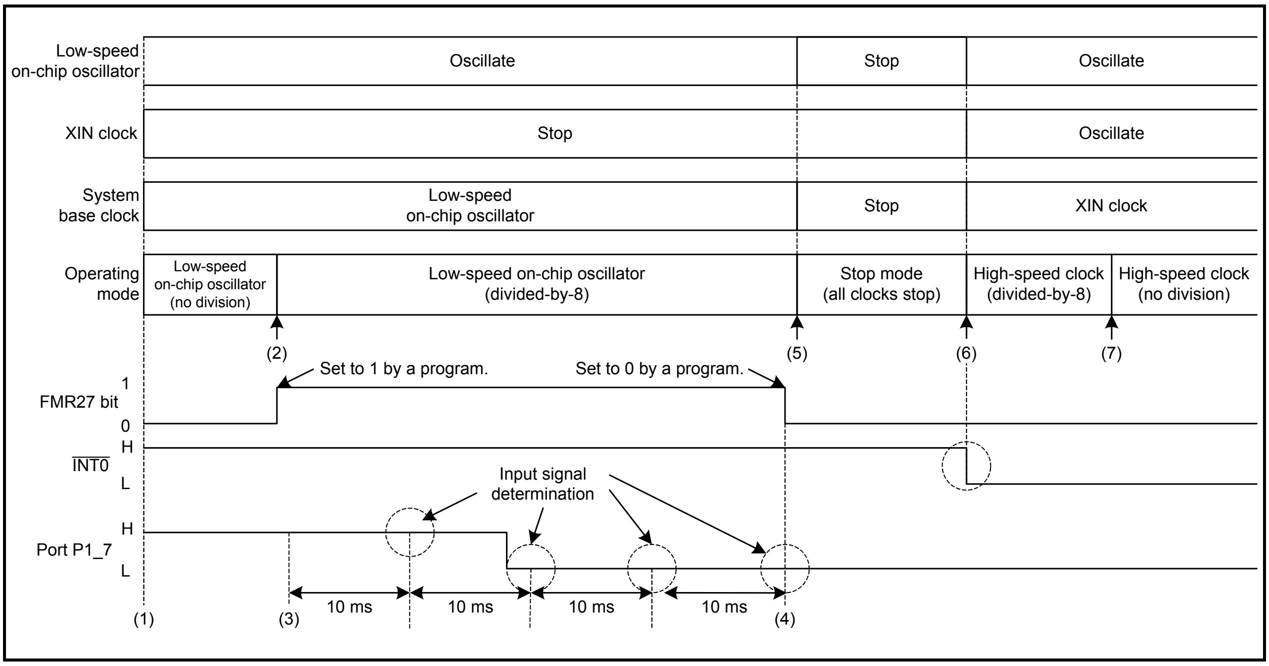


Figure 5.1 Stop Mode Operating Example

5.2 Required Memory Size

Table 5.1 lists the Required Memory Size.

Table 5.1 Required Memory Size for Sample Program 2

Memory Used	Size	Remarks
ROM	405 bytes	In the r01an0369_src_sample2.c module
RAM	2 bytes	In the r01an0369_src_sample2.c module
Maximum user stack usage	6 bytes	
Maximum interrupt stack usage	18 bytes	

The required memory size varies depending on the C compiler version and compile options.

5.3 Constant

Table 5.2 lists the Constant Used in the Sample Code.

Table 5.2 Constant Used in the Sample Code

Constant Name	Setting Value	Contents
LOW	0	Port P1_7 input level low

5.4 Variables

Table 5.3 lists the Global Variable, and Table 5.4 lists the static Variable.

Table 5.3 Global Variable

Type	Variable Name	Contents	Function Used
unsigned char	mode	Select to enter stop mode	main, stop_signal_in, power_control

Table 5.4 static Variable

Type	Variable Name	Contents	Function Used
static unsigned char	stp_sig_bit	Input information for stop mode enter signal	stop_signal_in

5.5 Functions

Table 5.5 lists the Functions.

Table 5.5 Functions

Function Name	Outline
mcu_init	System clock setting
int0_init	Initial setting of $\overline{\text{INT0}}$ interrupt
timer_rj2_init	Initial setting of timer RJ2
stop_signal_in	Input processing of stop mode enter signal
power_control	Stop mode processing
_int0	$\overline{\text{INT0}}$ interrupt handling

5.6 Function Specifications

The following tables list the sample code function specifications.

mcu_init	
Outline	System clock setting
Header	None
Declaration	void mcu_init(void)
Description	Set the system clock.
Argument	None
Returned value	None
Remark	—

int0_init	
Outline	Initial setting of $\overline{INT0}$ interrupt
Header	None
Declaration	void int0_init(void)
Description	Perform initial setting to use the $\overline{INT0}$ interrupt.
Argument	None
Returned value	None
Remark	—

timer_rj2_init	
Outline	Initial setting of timer RJ2
Header	None
Declaration	void timer_rj2_init(void)
Description	Perform initial setting to use timer RJ2 in timer mode.
Argument	None
Returned value	None
Remark	—

stop_signal_in	
Outline	Input processing of stop mode enter signal
Header	None
Declaration	void stop_signal_in(void)
Description	Perform stop mode enter determination.
Argument	None
Returned value	None
Remark	—

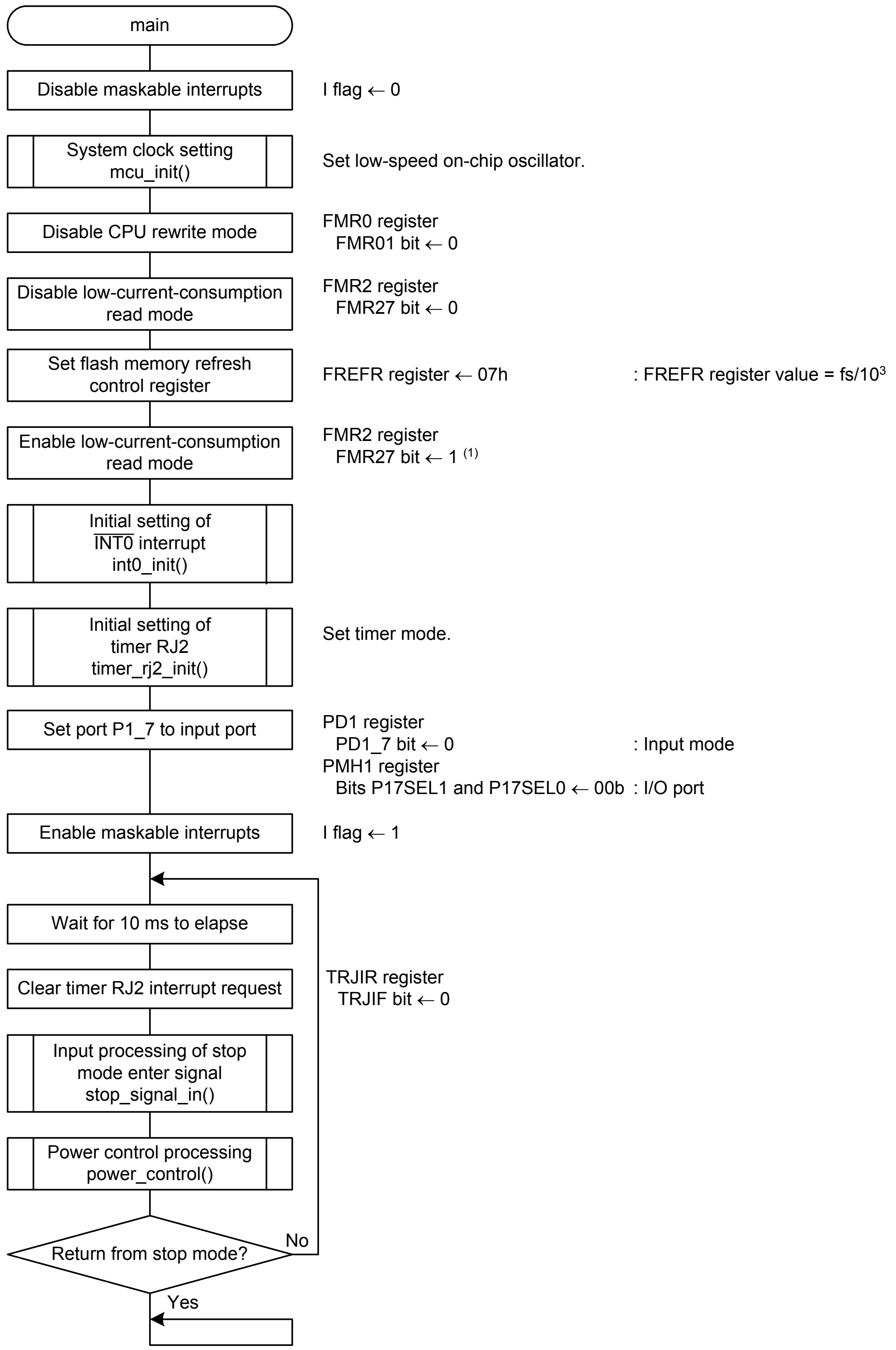
power_control	
Outline	Stop mode processing
Header	None
Declaration	void power_control(void)
Description	Enter stop mode.
Argument	None
Returned value	None
Remark	—

_int0	
Outline	$\overline{\text{INT0}}$ interrupt handling
Header	None
Declaration	void _int0(void)
Description	Perform $\overline{\text{INT0}}$ interrupt handling
Argument	None
Returned value	None
Remark	—

5.7 Flowcharts

5.7.1 Main Processing

Figure 5.2 shows the Main Processing.



Note:
1. To set the FMR27 bit to 1, first write 0 and then write 1 immediately.

Figure 5.2 Main Processing

5.7.2 System Clock Setting

Figure 5.3 shows the System Clock Setting.

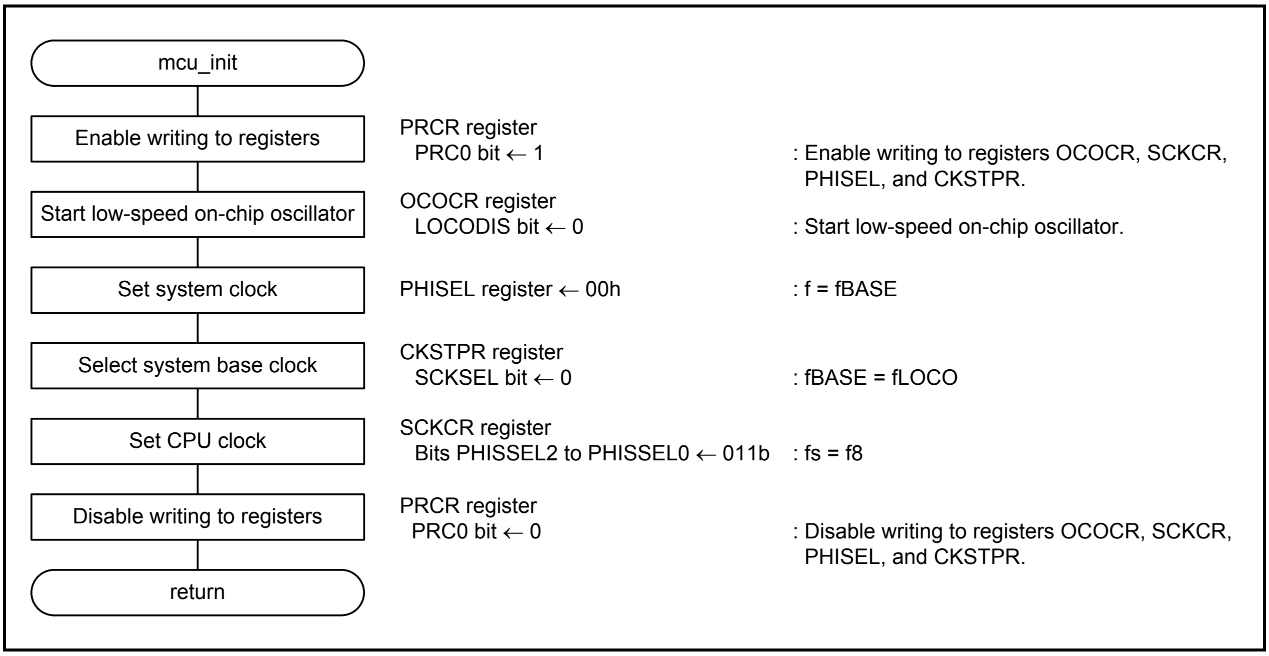


Figure 5.3 System Clock Setting

5.7.3 Initial Setting of the $\overline{\text{INT0}}$ Interrupt

Figure 5.4 shows the Initial Setting of the $\overline{\text{INT0}}$ Interrupt.

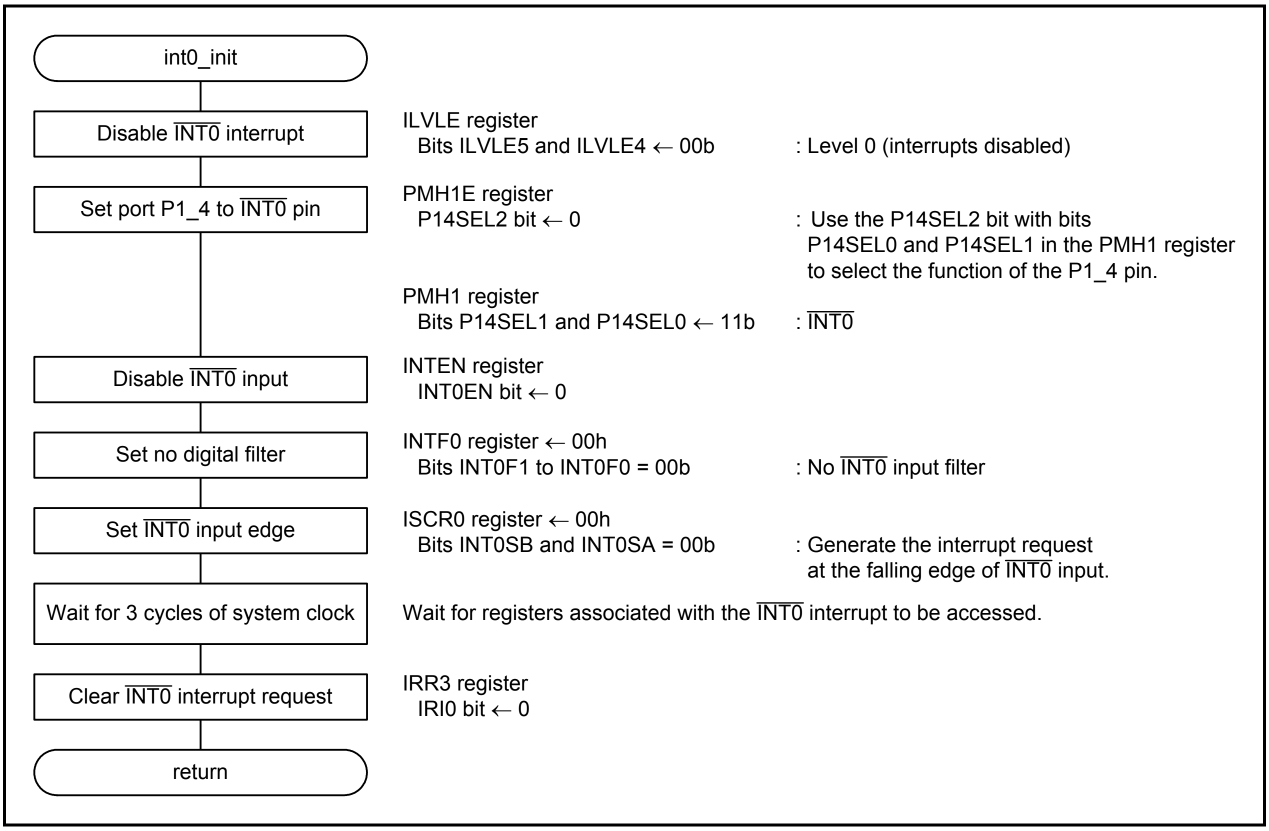
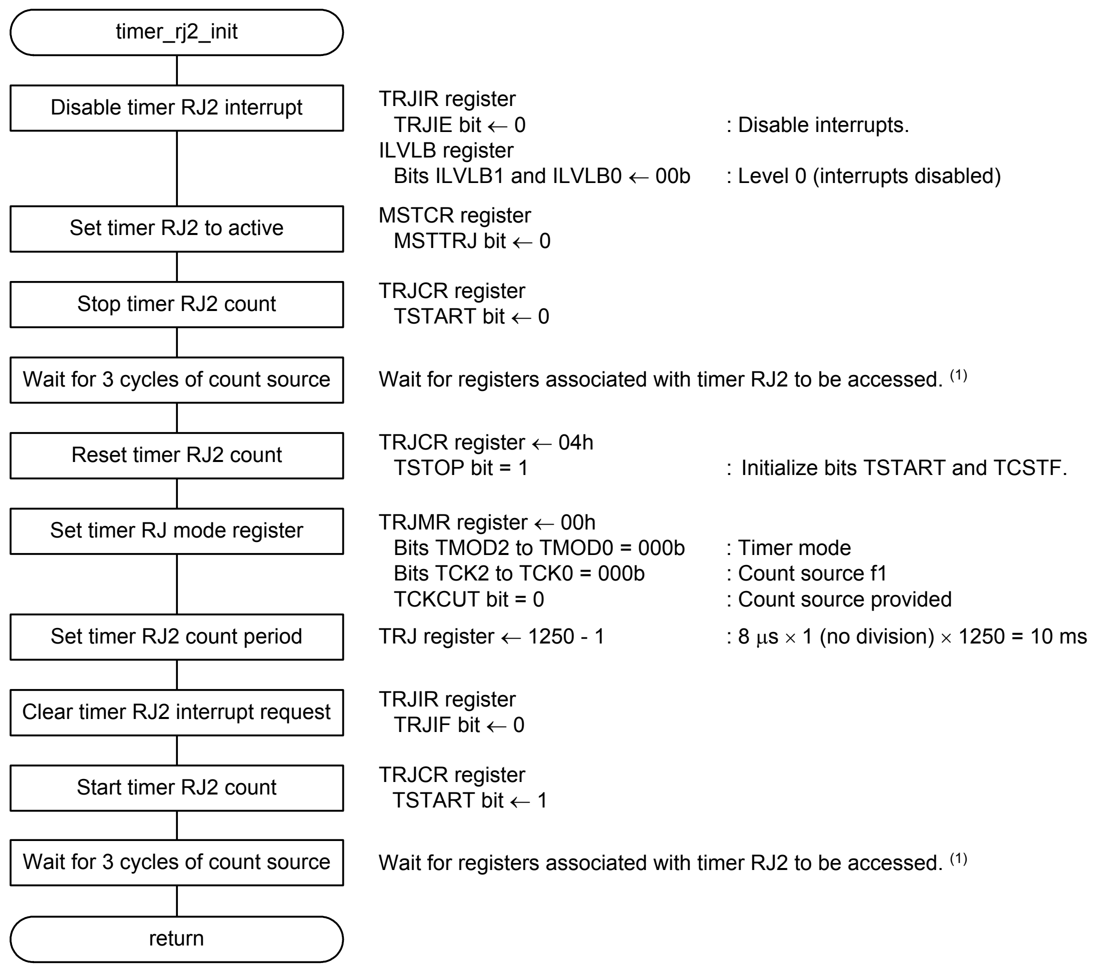


Figure 5.4 Initial Setting of the $\overline{\text{INT0}}$ Interrupt

5.7.4 Initial Setting of Timer RJ2

Figure 5.5 shows the Initial Setting of Timer RJ2.



Note:
1. Registers associated with timer RJ2: TRJ, TRJCR, and TRJMR

Figure 5.5 Initial Setting of Timer RJ2

5.7.5 Input Processing of the Stop Mode Enter Signal

Figure 5.6 shows the Input Processing of the Stop Mode Enter Signal.

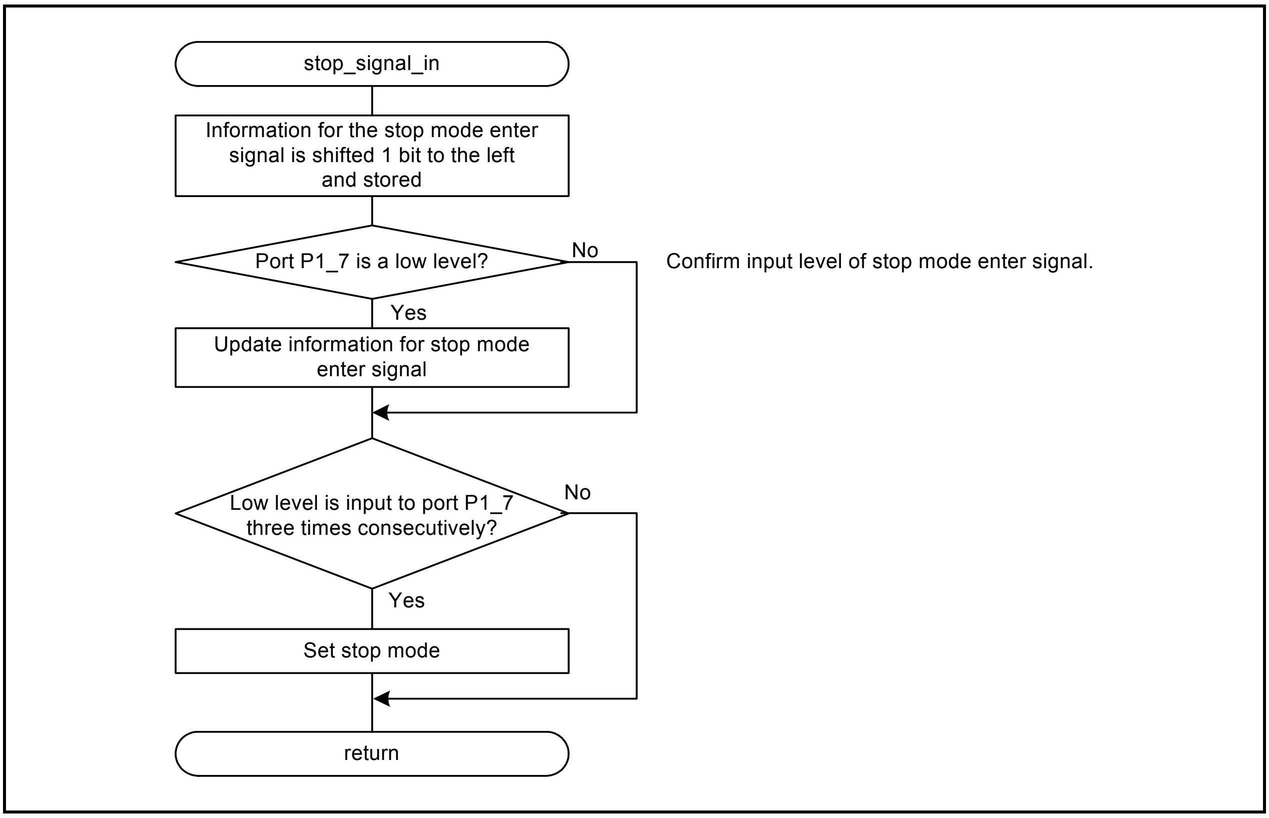


Figure 5.6 Input Processing of the Stop Mode Enter Signal

5.7.6 Stop Mode Processing

Figures 5.7 and 5.8 show Stop Mode Processing.

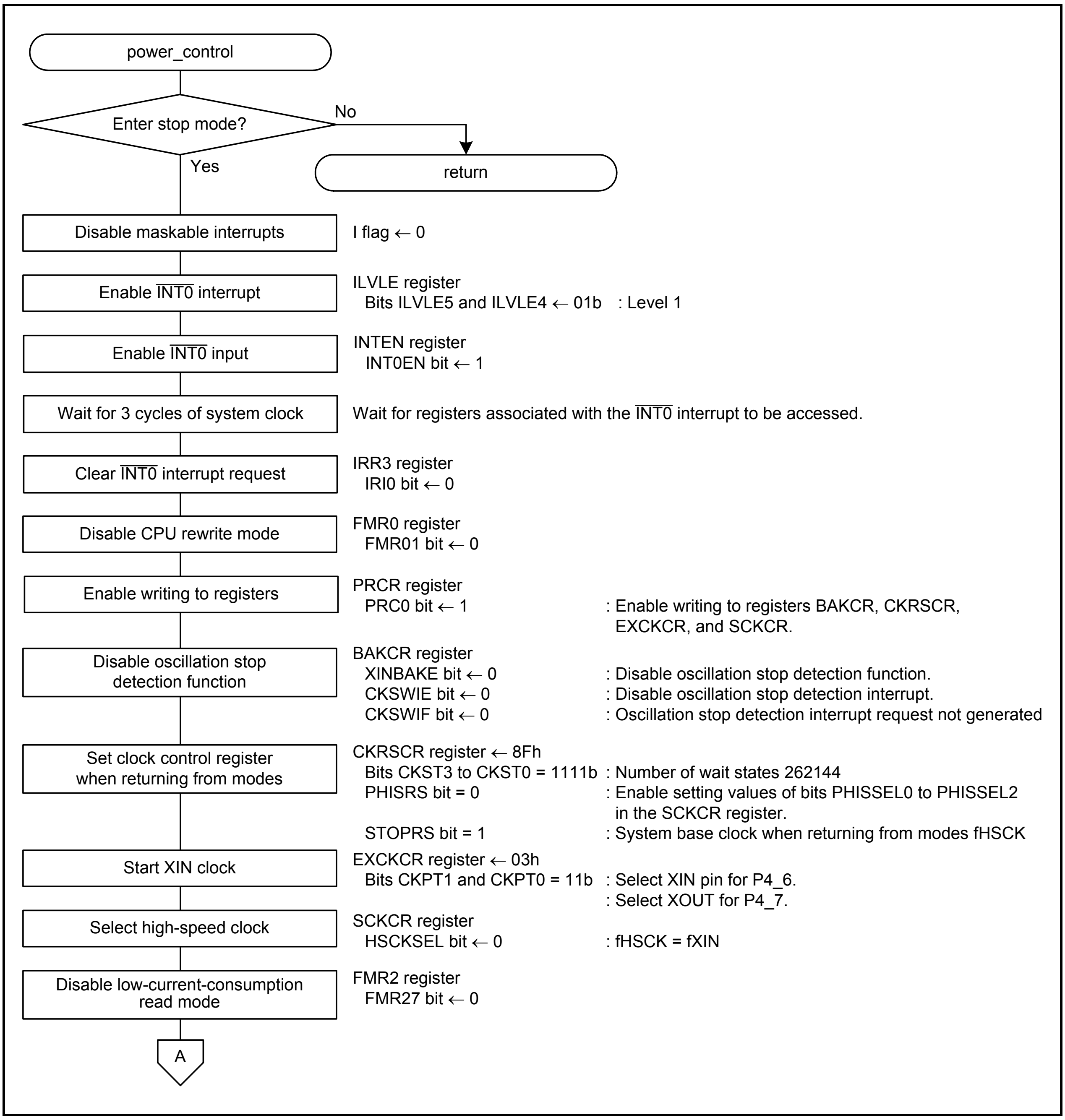
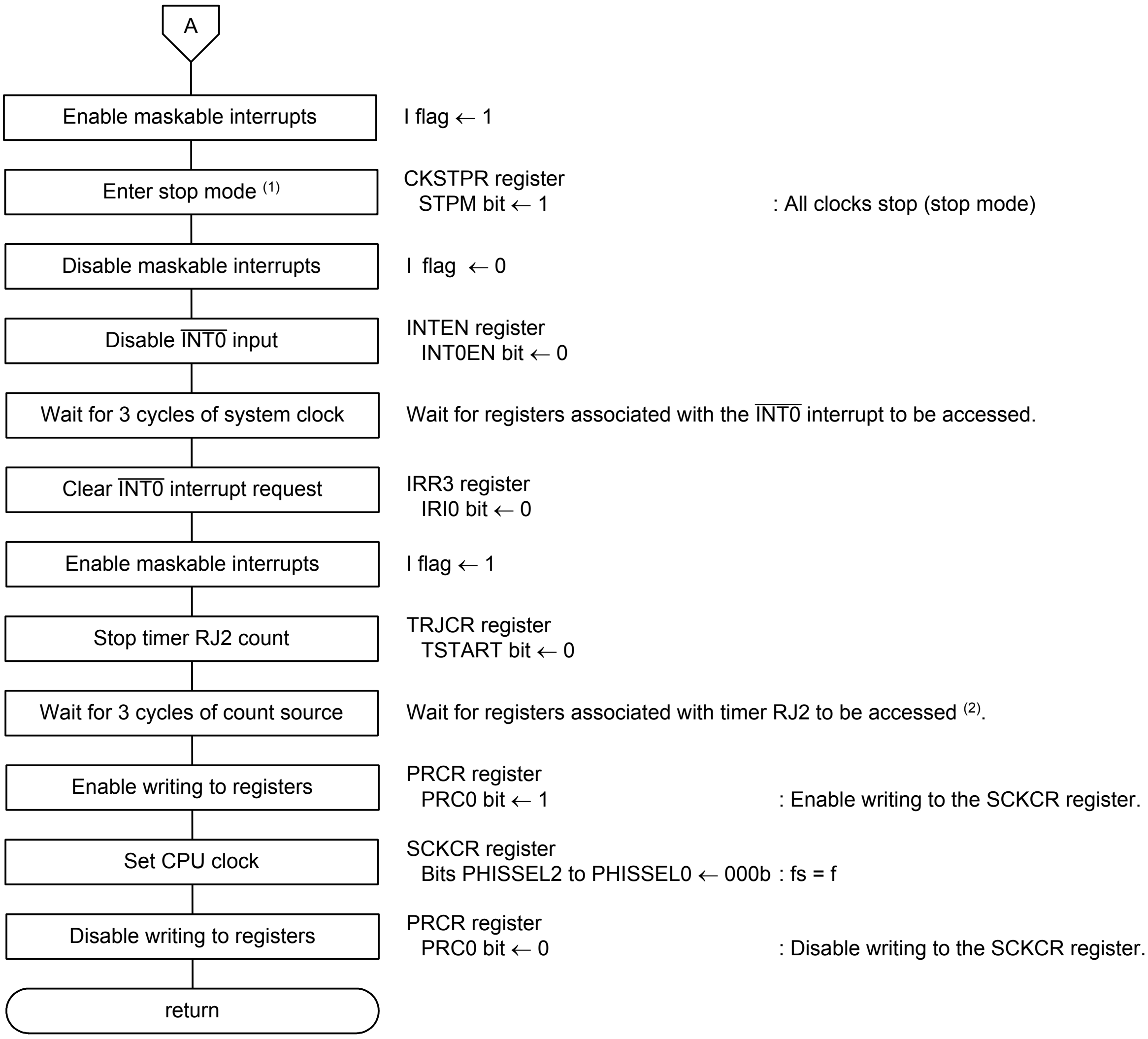


Figure 5.7 Stop Mode Processing (1/2)



Notes:
1. Since the 4 bytes of instruction data following the instruction that sets the STPM bit to 1 are prefetched and then the program stops, insert at least four NOP instructions following the JMP.B instruction immediately after the instruction that sets the STPM bit to 1 as a program restriction when entering wait mode.
2. Registers associated with timer RJ2: TRJ, TRJCR, TRJIOC, and TRJMR

Figure 5.8 Stop Mode Processing (2/2)

5.7.7 $\overline{\text{INT0}}$ Interrupt Handling

Figure 5.9 shows the $\overline{\text{INT0}}$ Interrupt Handling.

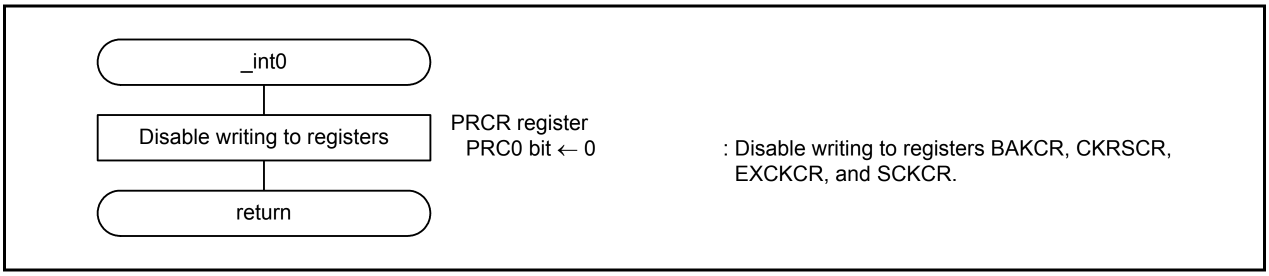


Figure 5.9 $\overline{\text{INT0}}$ Interrupt Handling

6. Software for Sample Program 3

6.1 Operation Overview of Sample Program 3

After reset, the MCU enters low-speed on-chip oscillator mode (divided-by-8) from low-speed on-chip oscillator mode (no division) by a program. Set the interval of refresh operations in low-current-consumption read mode to the FREFR register and the FMR27 bit in the FMR2 register to 1 (low-current-consumption read mode enabled) to use low-current-consumption read mode. Then the timer RJ2 count set to a 10 ms period starts. Read the P1_7 pin every 10 ms and determine if the MCU enters stop mode. When the state is held low three times consecutively, write 1 (stop mode) to the variable (mode). Disable maskable interrupts, enable the $\overline{\text{INT0}}$ interrupt input used to return from stop mode, disable CPU rewrite mode, disable the oscillation stop detection function, set a clock after returning from stop mode, and disable low-current-consumption read mode. Then enable maskable interrupts, set the STPM bit in the CKSTPR register to 1 (all clocks stop (stop mode)) to enter stop mode.
When applying the falling edge to the $\overline{\text{INT0}}$ pin, the MCU returns from stop mode. Low-speed on-chip oscillator mode (divided-by-8) is automatically selected for the operating mode when returning from stop mode. Then set the operating mode to low-speed on-chip oscillator mode (no division) by a program.

(1) After reset, set low-speed on-chip oscillator mode (divided-by-8) by a program.
(2) After setting the interval of refresh operations in low-current-consumption read mode, set the FMR27 bit in the FMR2 register to 1 (low-current-consumption read mode enabled).
(3) Start counting timer RJ2. After the count starts, read the port P1_7 pin every 10 ms.
(4) Read the P1_7 pin as a low level three times consecutively, disable maskable interrupts, and perform the settings described below.

Settings
- Enable the $\overline{\text{INT0}}$ interrupt input.
- Disable CPU rewrite mode.
- Disable oscillation stop detection.
- Set the PHISRS bit in the CKRSCR register to 0 (setting values of bits PHISSEL0 to PHISSEL2 in the SCKCR register are enabled).
- Set the STOPRS bit in the CKRSCR register to 0 (return from stop mode using the system base clock immediately before entering stop mode).
- Set the FMR27 bit to 0 (low-current-consumption read mode disabled).

(5) Enable maskable interrupts, set the STPM bit in the CKSTPR register to 1 to enter stop mode.
(6) Return from stop mode using the $\overline{\text{INT0}}$ interrupt (falling edge signal). Clocks set in (4) are selected for the CPU clock when returning from stop mode.
(7) Set the operating mode to low-speed on-chip oscillator mode (no division) by a program.

Figure 6.1 shows the Stop Mode Operating Example.

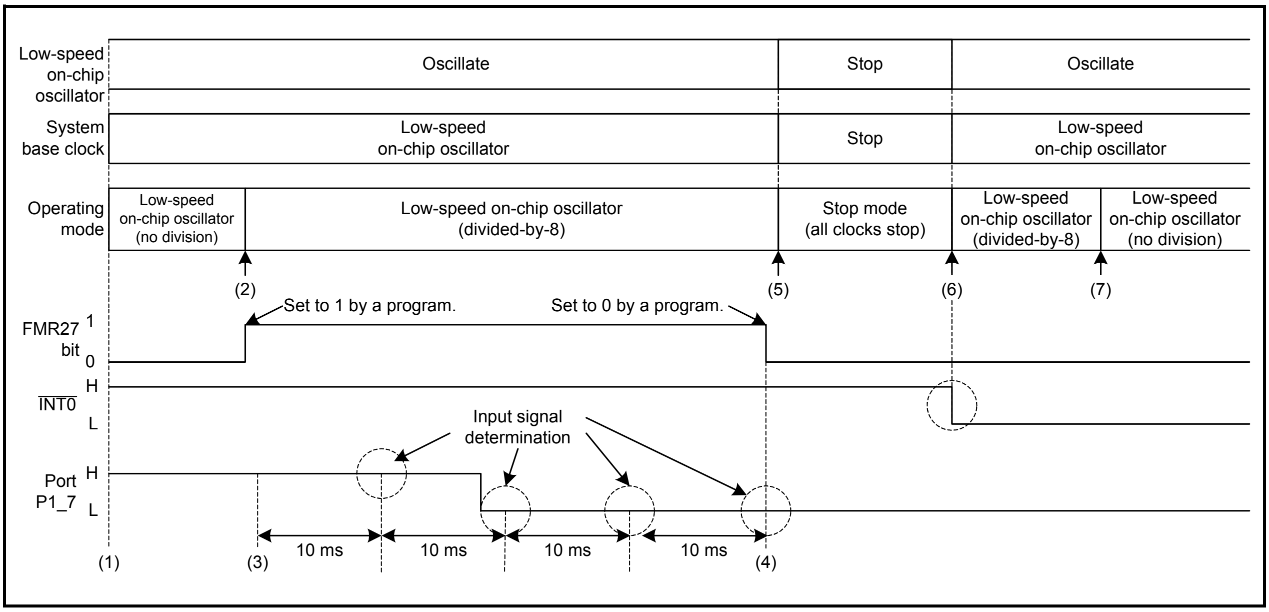


Figure 6.1 Stop Mode Operating Example

6.2 Required Memory Size

Table 6.1 lists the Required Memory Size.

Table 6.1 Required Memory Size for Sample Program 3

Memory Used	Size	Remarks
ROM	395 bytes	In the r01an0369_src_sample3.c module
RAM	2 bytes	In the r01an0369_src_sample3.c module
Maximum user stack usage	6 bytes	
Maximum interrupt stack usage	18 bytes	

The required memory size varies depending on the C compiler version and compile options.

6.3 Constant

Table 6.2 lists the Constant Used in the Sample Code.

Table 6.2 Constant Used in the Sample Code

Constant Name	Setting Value	Contents
LOW	0	Port P1_7 input level low

6.4 Variables

Table 6.3 lists the Global Variable, and Table 6.4 lists the static Variable.

Table 6.3 Global Variable

Type	Variable Name	Contents	Function Used
unsigned char	mode	Select to enter stop mode	main, stop_signal_in, power_control, _int0

Table 6.4 static Variable

Type	Variable Name	Contents	Function Used
static unsigned char	stp_sig_bit	Input information for stop mode enter signal	stop_signal_in

6.5 Functions

Table 6.5 lists the Functions.

Table 6.5 Functions

Function Name	Outline
mcu_init	System clock setting
int0_init	Initial setting of $\overline{\text{INT0}}$ interrupt
timer_rj2_init	Initial setting of timer RJ2
stop_signal_in	Input processing of stop mode enter signal
power_control	Stop mode processing
_int0	$\overline{\text{INT0}}$ interrupt handling

6.6 Function Specifications

The following tables list the sample code function specifications.

mcu_init	
Outline	System clock setting
Header	None
Declaration	void mcu_init(void)
Description	Set the system clock.
Argument	None
Returned value	None
Remark	—

int0_init	
Outline	Initial setting of $\overline{INT0}$ interrupt
Header	None
Declaration	void int0_init(void)
Description	Perform initial setting to use the $\overline{INT0}$ interrupt.
Argument	None
Returned value	None
Remark	—

timer_rj2_init	
Outline	Initial setting of timer RJ2
Header	None
Declaration	void timer_rj2_init(void)
Description	Perform initial setting to use timer RJ2 in timer mode.
Argument	None
Returned value	None
Remark	—

stop_signal_in	
Outline	Input processing of stop mode enter signal
Header	None
Declaration	void stop_signal_in(void)
Explanation	Perform stop mode enter determination.
Argument	None
Returned value	None
Remark	—

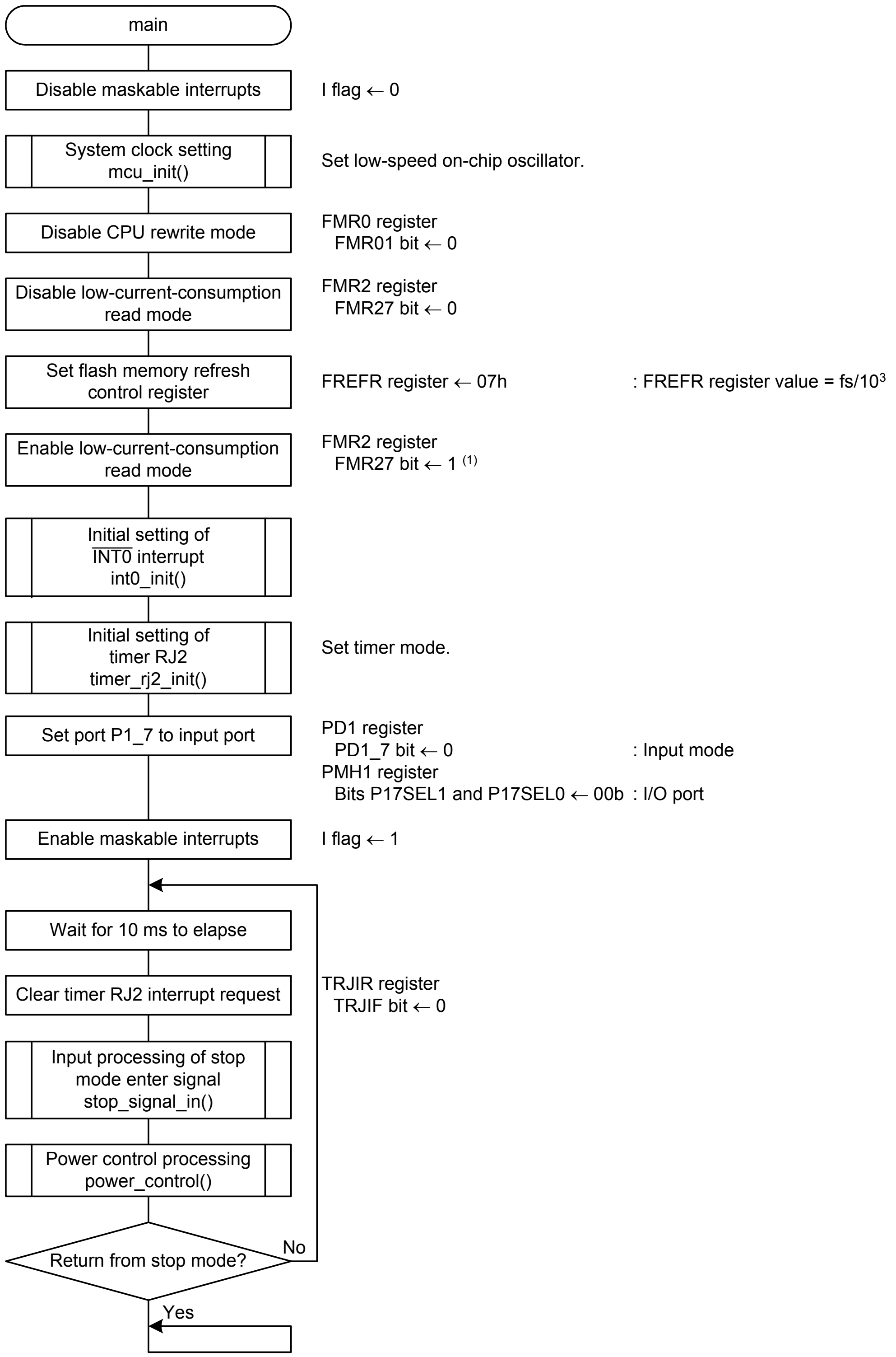
power_control	
Outline	Stop mode processing
Header	None
Declaration	void power_control(void)
Description	Enter stop mode.
Argument	None
Returned value	None
Remark	—

_int0	
Outline	$\overline{\text{INT0}}$ interrupt handling
Header	None
Declaration	void _int0(void)
Description	Perform $\overline{\text{INT0}}$ interrupt handling
Argument	None
Returned value	None
Remark	—

6.7 Flowcharts

6.7.1 Main Processing

Figure 6.2 shows the Main Processing.



Note:
1. To set the FMR27 bit to 1, first write 0 and then write 1 immediately.

Figure 6.2 Main Processing

6.7.2 System Clock Setting

Figure 6.3 shows the System Clock Setting.

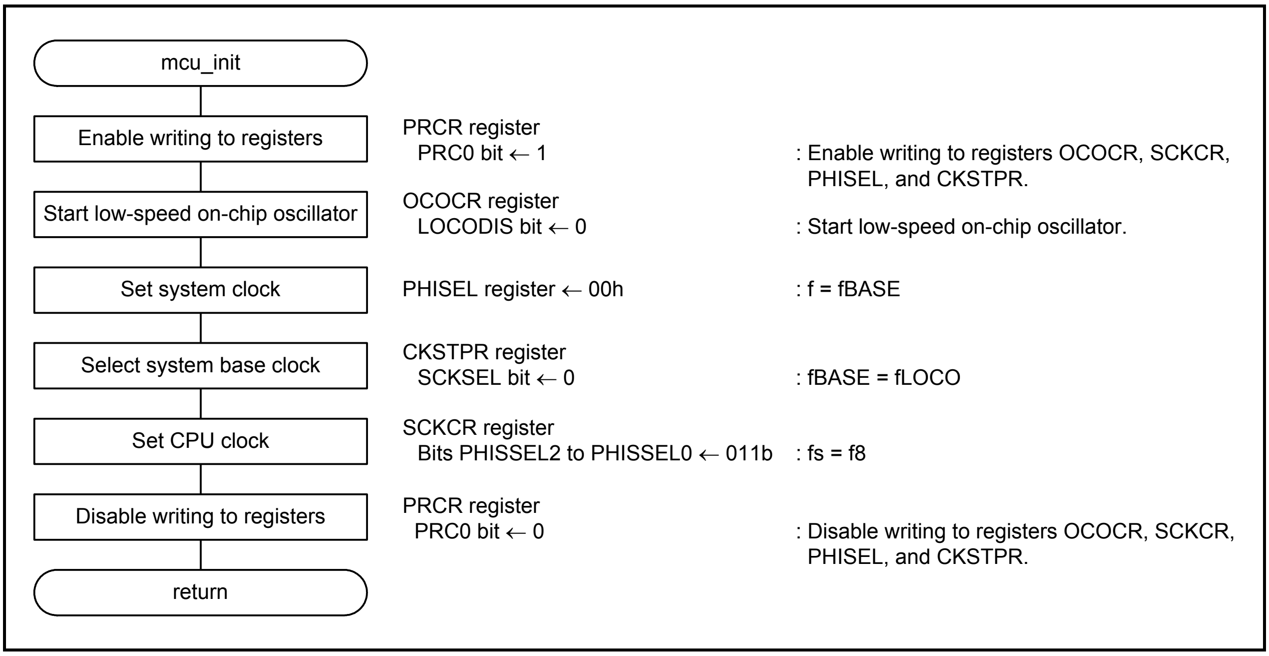


Figure 6.3 System Clock Setting

6.7.3 Initial Setting of the $\overline{\text{INT0}}$ Interrupt

Figure 6.4 shows the Initial Setting of the $\overline{\text{INT0}}$ Interrupt.

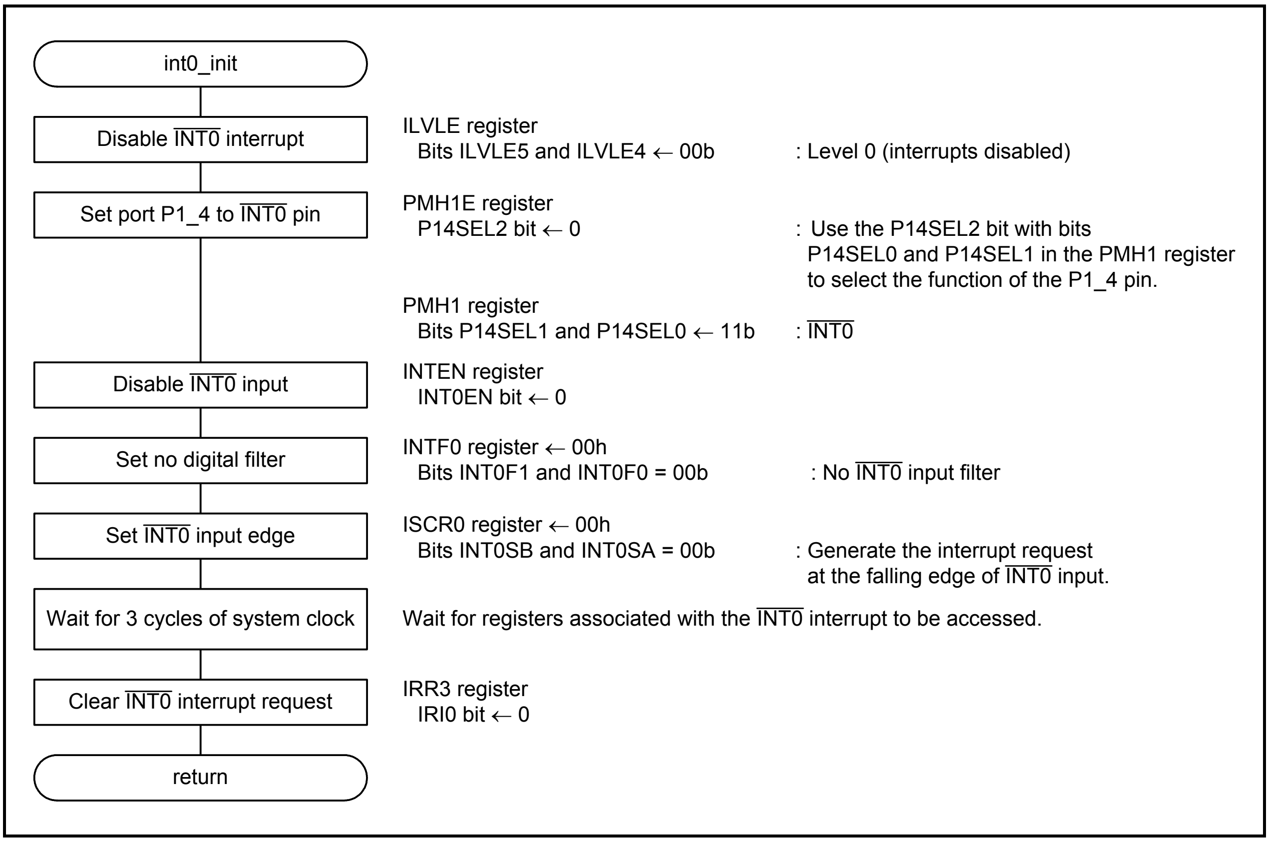
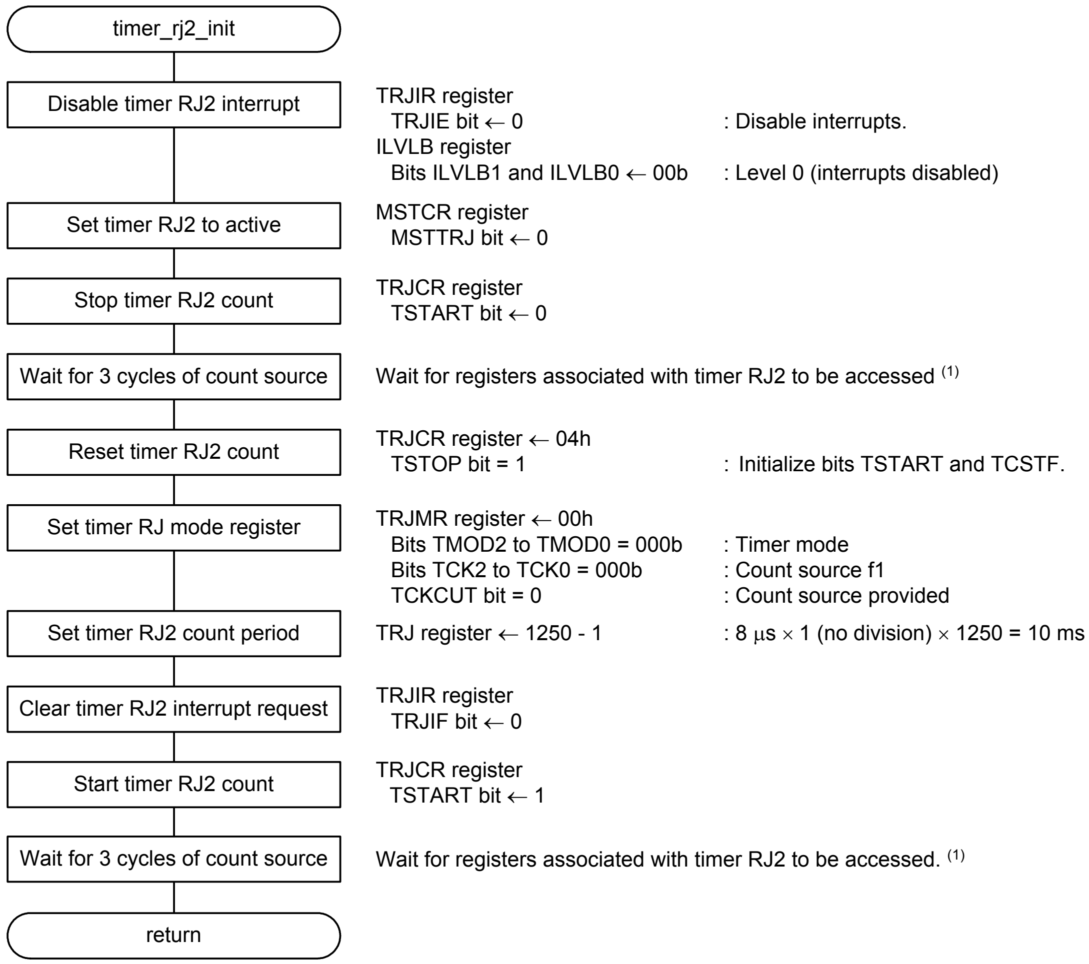


Figure 6.4 Initial Setting of the $\overline{\text{INT0}}$ Interrupt

6.7.4 Initial Setting of Timer RJ2

Figure 6.5 shows the Initial Setting of Timer RJ2.



Note:
1. Registers associated with timer RJ2: TRJ, TRJCR, and TRJMR

Figure 6.5 Initial Setting of Timer RJ2

6.7.5 Input Processing of the Stop Mode Enter Signal

Figure 6.6 shows the Input Processing of the Stop Mode Enter Signal.

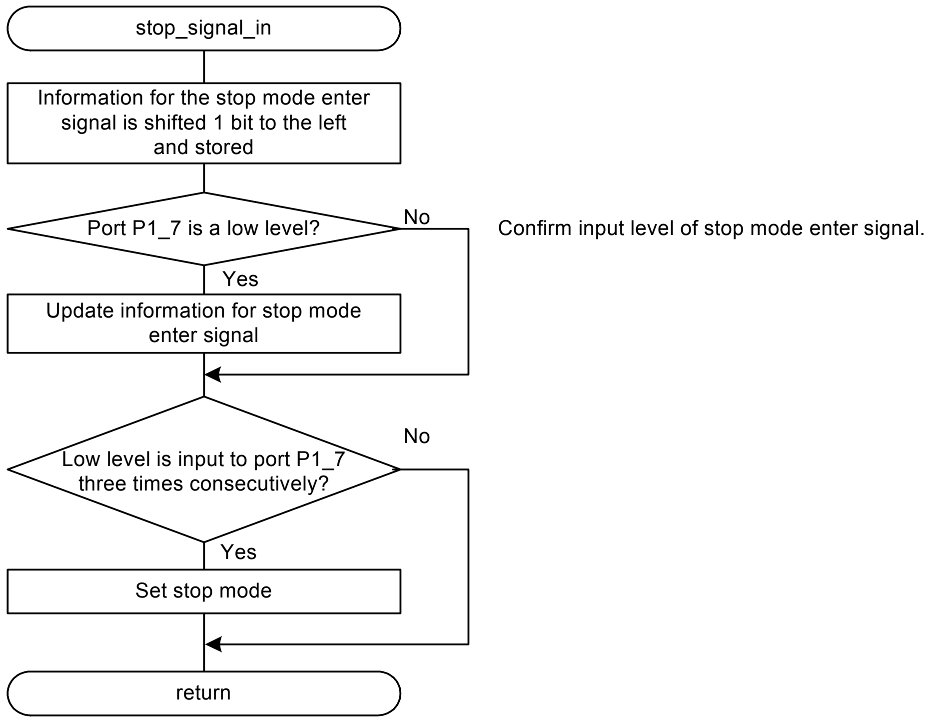


Figure 6.6 Input Processing of the Stop Mode Enter Signal

6.7.6 Stop Mode Processing

Figures 6.7 and 6.8 show Stop Mode Processing.

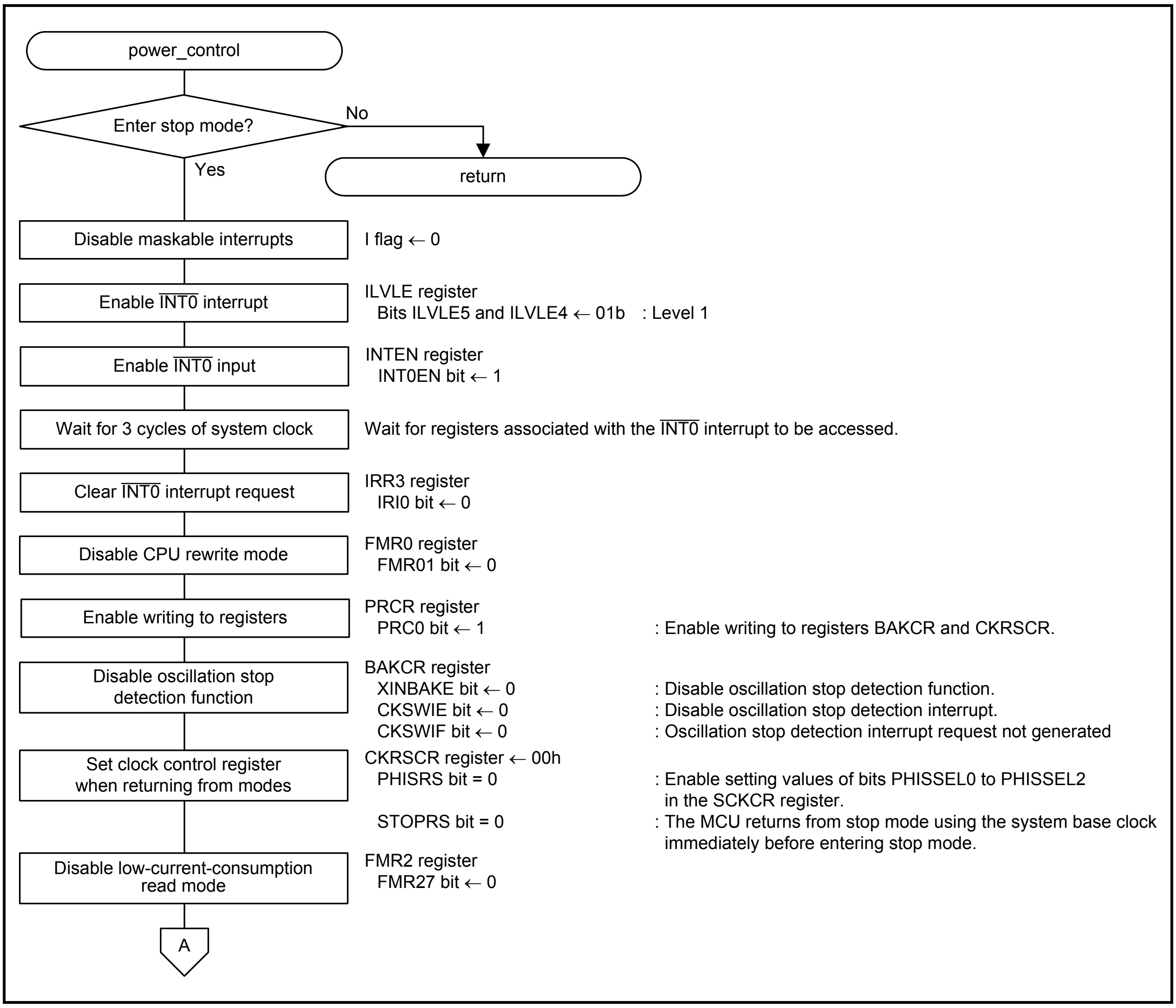
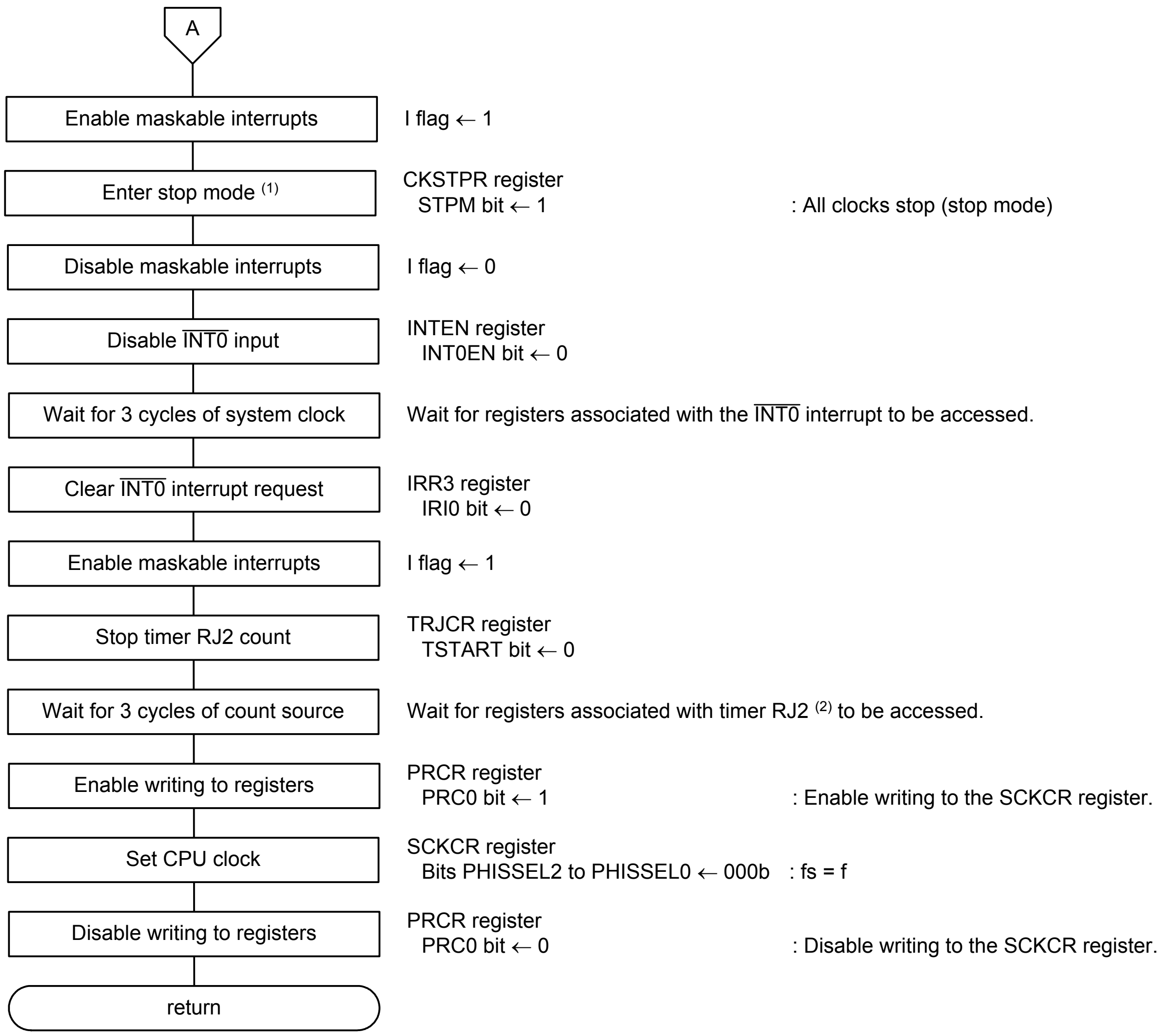


Figure 6.7 Stop Mode Processing (1/2)



Notes:
1. Since the 4 bytes of instruction data following the instruction that sets the STPM bit to 1 are prefetched and then the program stops, insert at least four NOP instructions following the JMP.B instruction immediately after the instruction that sets the STPM bit to 1 as a program restriction when entering wait mode.
2. Registers associated with timer RJ2: TRJ, TRJCR, TRJIOC, and TRJMR

Figure 6.8 Stop Mode Processing (2/2)

6.7.7 $\overline{\text{INT0}}$ Interrupt Handling

Figure 6.9 shows the $\overline{\text{INT0}}$ Interrupt Handling.

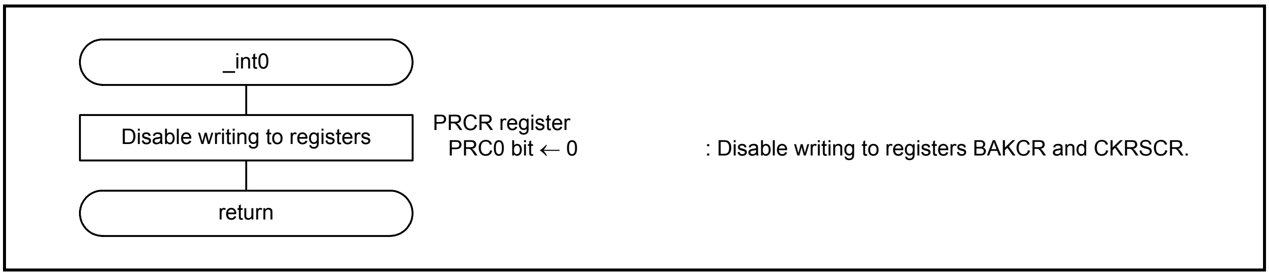


Figure 6.9 $\overline{\text{INT0}}$ Interrupt Handling

7. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

8. Reference Documents

R8C/M12A Group User's Manual: Hardware Rev.1.00
The latest version can be downloaded from the Renesas Electronics website.

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

C Compiler Manual
M16C Series, R8C Family C Compiler Package V.5.45
C Compiler User's Manual Rev.2.00
The latest version can be downloaded from the Renesas Electronics website.

Website and Support

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

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

Revision History	R8C/M12A Group Power Control in Stop Mode

Rev.	Date	Description	
		Page	Summary
1.01	Aug. 29, 2011	—	First edition issued

All trademarks and registered trademarks are the property of their respective owners.

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

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

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

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

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

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

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

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

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

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

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

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

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

© 2011 Renesas Electronics Corporation. All rights reserved.

Colophon 1.1