RL78/G13

Safety Function (Frequency Detection) CC-RL

R01AN2847EJ0100
Rev. 1.00
May 28, 2015

Introduction

This application note describes the frequency detection function which is one of the safety features offered by the RL78/G13.

The frequency detection function compares the high-speed on-chip oscillator clock or the external X1 oscillation clock and the low-speed on-chip oscillator clock. This allows detection of any abnormal clock frequencies.

Target Device

RL78/G13

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 4

2. Operation Check Conditions 5

3. Related Application Note 5

4. Description of the Hardware 6
4.1 Hardware Configuration Example 6
4.2 List of Pins to be used 7

5. Description of the Software 8
5.1 Operation Outline 8
5.2 File Configuration 10
5.3 List of Option Byte Settings 11
5.4 List of Constants 12
5.5 List of Variables 13
5.6 List of Functions 14
5.7 Function Specifications 15
5.8 Flowcharts 24
5.8.1 Initialization Function 25
5.8.2 System Function 26
5.8.3 I/O Port Setup 27
5.8.4 CPU Clock Setup 28
5.8.5 TAU0 Setup 29
5.8.6 Real-Time Clock Setup 35
5.8.7 Interval Timer Setup 36
5.8.8 External Interrupt Input Setup 37
5.8.9 Main Processing 38
5.8.10 Starting INTP0 External Interrupt Processing 41
5.8.11 Starting Pulse Interval Measurement 42
5.8.12 Starting Channel 5 of TAU0 43
5.8.13 Stopping Channel 5 of TAU0 44
5.8.14 Getting Count Clock Change Request Flag 45
5.8.15 Getting Pulse Interval Measurement End Flag 46
5.8.16 Judging Pulse Interval Measurement Value 47
5.8.17 Getting Pulse Interval Measurement Value 48
5.8.18 Clearing Pulse Interval Measurement End Flag 49
5.8.19 Changing Count Clock 50
5.8.20 Changing HOCO Clock 51
5.8.21 Starting LED Blink 52
5.8.22 Stopping LED Blink 53
5.8.23 Starting Real-Time Clock 54
5.8.24 Stopping Real-Time Clock 55
5.8.25 Starting Real-Time Clock Fixed-Frequency Interrupt 56
5.8.26 Stopping Real-Time Clock Fixed-Frequency Interrupt 57
5.8.27 Clearing Count Clock Change Request Flag 58
5.8.28 Interval Timer Interrupt 59

5.8.29 Capture End Interrupt for Channel 5 of TAU0 60
5.8.30 INTP0 External Interrupt 61
5.8.31 RTC Fixed-Frequency Interrupt 62
5.8.32 Callback Function for RTC Fixed-frequency Interrupts 63
5.8.33 Interval Timer Start 64
5.8.34 Stopping Interval Timer 65
5.8.35 Getting INTP0 External Interrupt Occurrence Flag 66
5.8.36 Clearing INTP0 External Interrupt Occurrence Flag 67
5.8.37 Getting Interval Timer Interrupt Occurrence Flag 68
5.8.38 Clearing Interval Timer Interrupt Occurrence Flag 69

6. Sample Code 70

7. Documents for Reference 70

Revision Record 71

General Precautions in the Handling of MPU/MCU Products 72

1. Specifications

The frequency detection function described in this application note compares the frequencies of the high-speed on-chip oscillator clock and the low-speed on-chip oscillator clock. This allows detection of any abnormal clock frequencies.

More specifically, this function measures the pulse interval under the conditions below to detect an abnormal frequency.

- The high-speed on-chip oscillator clock (HOCO clock) is selected as a count clock for timer array unit 0 (TAU0).
- The low-speed on-chip oscillator clock (15 kHz) should is selected as a timer input for channel 5 of TAU0.

To judge whether the frequency is normal or abnormal, the pulse interval is checked to see if it is within the tolerable range specified by constants. If the frequency is normal, the LED turns off. If not, the LED blinks.

The count clock for TAU0 is selected by setting the value of the high-speed on-chip oscillator frequency select register (HOCODIV) to one of predefined constants.

Table 1.1 lists the peripheral functions to be used and their uses. Figure 1.1 outlines the frequency detection.

Table 1.1 Peripheral Functions to be Used and their Uses

Peripheral Function	Use
External interrupt input (INTP0)	Switch input Changes the count clock (HOCO clock) frequency for TAU0.
Channel 5 of timer array unit 0	Low-speed on-chip oscillator clock Measures pulse intervals.
Bit 2 in port 6	Displays the frequency detection result on the LED.
Bit 3 in port 6	Displays the selected HOCO clock on the LED.

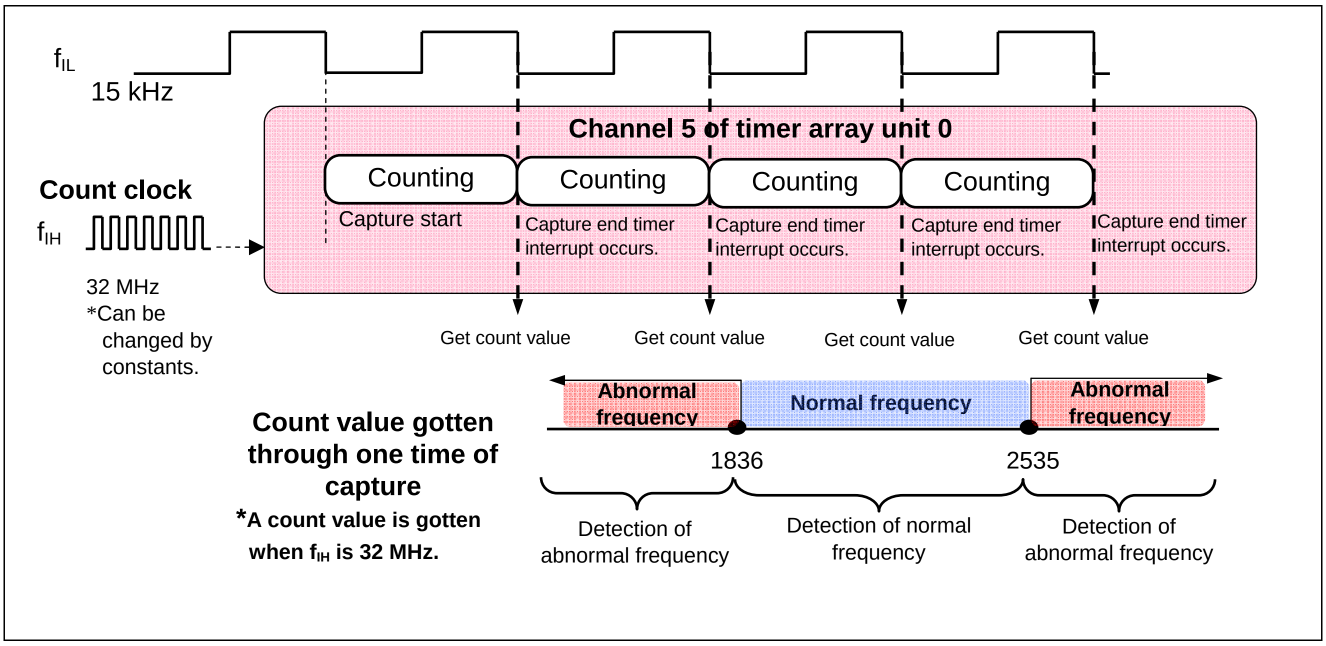

f_{IL}
15 kHz
Channel 5 of timer array unit 0
Count clock
f_{IH}
Counting
Counting
Counting
Counting
Capture start
Capture end timer interrupt occurs.
Capture end timer interrupt occurs.
Capture end timer interrupt occurs.
Capture end timer interrupt occurs.
32 MHz
*Can be changed by constants.
Get count value
Get count value
Get count value
Get count value
Abnormal frequency
Normal frequency
Abnormal frequency
1836
2535
Count value gotten through one time of capture
*A count value is gotten when f_{IH} is 32 MHz.
Detection of abnormal frequency
Detection of normal frequency
Detection of abnormal frequency

Figure 1.1 Outline of Frequency Detection

2. Operation Check Conditions

The sample code contained in this application note has been checked under the conditions listed in the table below.

Table 2.1 Operation Check Conditions

Item	Description
Microcontroller used	RL78/G13 (R5F100LEA)
Operating frequency	• CPU/peripheral hardware clock: The clock can be changed by depressing the target board's switch. Predefine each of the two constants for the clock as 32, 16, 8, 4, 2 or 1 MHz.
Operating voltage	5.0 V (Operation is possible at 2.9 V to 5.5 V.) LVD operation (V_{LVD}): Reset mode 2.81 V (2.76 V to 2.87 V)
Integrated development environment (CS+)	CS+ V3.01.00 from Renesas Electronics Corp.
C compiler (CS+)	CC-RL V1.01.00 from Renesas Electronics Corp.
Integrated development environment (e^2 studio)	e^2 studio V4.0.0.26 from Renesas Electronics Corp.
C compiler (e^2 studio)	CC-RL V1.01.00 from Renesas Electronics Corp.
Board used	RL78/G13 target board (QB-R5F100LE-TB)

3. Related Application Note

The application note that is related to this application note is listed below for reference.

RL78/G13 Initialization (R01AN2575E) Application Note

4. Description of the Hardware

4.1 Hardware Configuration Example

The example of configuration of the hardware that is used for this application note is shown below.

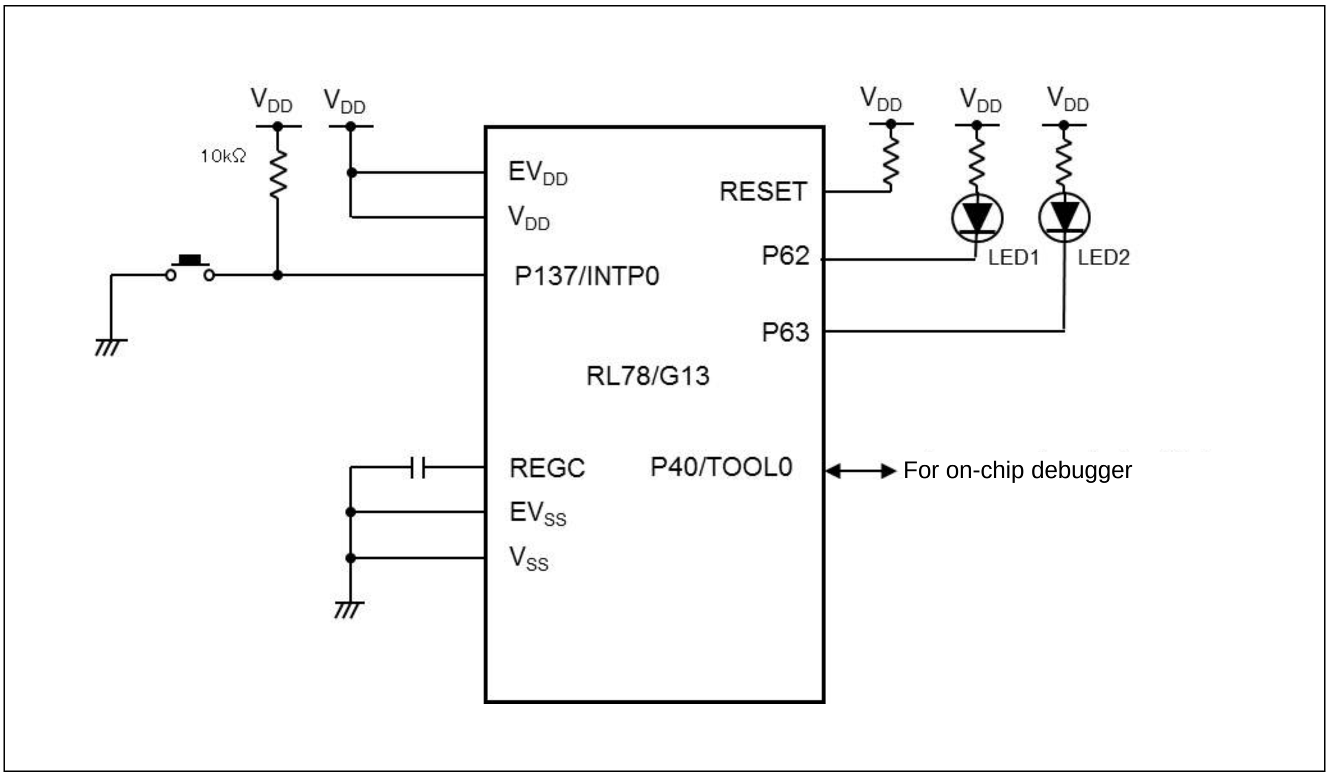


Figure 4.1 Hardware Configuration

Notes
1. The purpose of this circuit is only to provide the connection outline and the circuit is simplified accordingly. When designing and implementing an actual circuit, provide proper pin treatment and make sure that the hardware's electrical specifications are met (connect the input-only ports separately to V_{DD} or V_{SS} via a resistor).
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.
3. V_{DD} must be held at not lower than the reset release voltage (V_{LVD}) that is specified as LVD.

4.2 List of Pins to be used

Table 4.1 lists the pins to be used and their functions.

Table 4.1 Pins to be Used and their Functions

Pin Name	I/O	Function
P137/INTP0	Input	Switch input Changes the count clock for TAU0.
P62	Output	Displays the frequency detection result on the LED.
P63	Output	Displays the selected HOCO clock on the LED.

5. Description of the Software

5.1 Operation Outline

The sample code shown in this application note compares the frequencies of the high-speed on-chip oscillator clock and the low-speed on-chip oscillator clock. This enables detection of any abnormal clock frequencies.

More specifically, the sample code measures the pulse interval under the conditions below to detect an abnormal frequency.

- The high-speed on-chip oscillator clock (HOCO clock) is selected as a count clock for timer array unit 0 (TAU0).
- The low-speed on-chip oscillator clock (15 kHz) is selected as a timer input for channel 5 of TAU0.

To judge whether the frequency is normal or abnormal, the pulse interval is checked to see if it is within the tolerable range specified by constants. If the frequency is normal, the LED1 turns off. If not, the LED1 blinks.

The tolerable range above depends on PULSEWIDTH_RANGE_MIN and PULSEWIDTH_RANGE_MAX in Table 5.3.

The count clock frequency for TAU0 can be changed dynamically by using the switch. Its frequency depends on TAU0_COUNT_CLOCK_1 and TAU0_COUNT_CLOCK_2 in Table 5.3. The LED2 turns off when the device operates with the count clock set by TAU0_COUNT_CLOCK_1. The LED2 turns on when the device operates with the count clock set by TAU0_COUNT_CLOCK_2.

(1) Initialize the TAU.

Initialize the TAU.

<Conditions for setting>

- Select the HOCO clock as a count clock for TAU0.
- Select the low-speed on-chip oscillator clock (15 kHz) as a timer input for channel 5 of TAU0.

(2) Start the pulse interval measurement.

The value captured by a first capture end timer interrupt (INTTM05) is invalid. Thus, invalidate the first pulse interval measurement value according to the following processing:

- Disable interrupts.
- Set the TS05 bit of timer channel start register 0 (TS0) to 1 to enable counting. This clears the timer count register (TCR05) to 0000H and starts counting.
- Transition to the HALT status and wait until a capture end timer interrupt (INTTM05) occurs.
- When the timer input enable edge is detected, the timer counter register (TCR05) value is captured into the timer data register (TDR05) and then a capture end timer interrupt (INTTM05) will occur. (This results in a return from the HALT status). Then, clear the timer counter register (TCR05) to 0000H.
- Clear the interrupt request flag for INTTM05.
- Enable interrupts.

(3) Transition to HALT mode.

- Enter the HALT mode and wait until the next enable edge input.
- Return from the HALT mode by the processing of the second or subsequent capture end timer interrupt (INTTM05) or by that of switching external interrupt (INTP0).

(4) Check an interrupt source.

The processing differs depending on the interrupt source upon return from the HALT mode. Check the interrupt source after return from the HALT mode.

<When returning from the HALT mode upon completion of pulse interval measurement>

- Get a pulse interval measurement value.
- If the measurement value is within the tolerable range, LED1 turns off. Then, return to step (3).
- If it does not, LED1 blinks. Then, return to step (3).

<When returning from the HALT mode upon a switching external interrupt>

- To prevent chattering, take actions (A) through (F) below.
  - A) Using the interrupt handler for external interrupts to set the RINTE bit in the interval timer control register (ITMC) to 1. Then, start counting.
  - B) Wait until the value written to the RINTE bit is reflected.
  - C) Wait until an interval timer interrupt occurs.
  - D) Check the switch status with the interval timer interrupt handler. That is, check the P137 status.
  - E) If this status is 1, determine that the switch has not been depressed. Then, return to step (3).
  - F) If it is 0, determine that the switch has been depressed. Perform the operations below.
- Stop the timer of the TAU0.
- Change the HOCO clock which is the count clock for TAU0.
- If the current count clock for TAU0 is set by TAU0_COUNT_CLOCK_1 in Table 5.3, change to the count clock set by TAU0_COUNT_CLOCK_2 and turn on LED2. Then, return to step (2).
- If the current count clock for TAU0 is set by TAU0_COUNT_CLOCK_2 in Table 5.3, change to the count clock set by TAU0_COUNT_CLOCK_1 and turn off LED2. Then, return to step (2).

5.2 File Configuration

Table 5.1 lists the files that are used in this sample code. This table excludes files which are automatically generated by the integrated development environment.

Table 5.1 File Configuration

File name	Overview	Remarks
r_main.c	Main module	Additional functions: R_Main_Get_PulseWidthMeasureResult, R_Main_fCLK_Change, R_Main_HOCO_Change, R_Main_Start_LedBlink, R_Main_Stop_LedBlink
r_cg_intc_user.c	External interrupt input module INTP0 external interrupt	Additional functions: R_INTC0_Get_INTP0_Flag, R_INTC0_Clear_INTP0_Flag
r_cg_timer_user.c	TAU module Capture end timer interrupt (INTTM05)	Additional functions: R_TAU0_Channel5_MeasureStart, R_TAU0_Channel5_Get_MeasureStatus, R_TAU0_Channel5_Clear_MeasureStatus
r_cg_it_user.c	Interval timer module Interval timer interrupt	Additional functions: R_IT_Get_fCLK_ChangeFlag, R_IT_Clear_fCLK_ChangeFlag, R_IT_Get_INTIT_Flag R_IT_Clear_INTIT_Flag

5.3 List of Option Byte Settings

Table 5.2 summarizes the settings of the option bytes.

Table 5.2 Option Byte Settings

Address	Value	Description
000C0H/010C0H	11101111B	Disables the watchdog timer. (Stops counting after the release from the reset status.)
000C1H/010C1H	01111111B	LVD reset mode, 2.81 V (2.76 V to 2.87 V)
000C2H/010C2H	11101000B	HS mode, HOCO : 32 MHz
000C3H/010C3H	10000100B	Enables the on-chip debugger. Erases the data in the flash memory when on-chip debugging security ID authentication fails.

5.4 List of Constants

Table 5.3 lists the constants that are used in this sample program.

Table 5.3 Constants for Sample Program

Constant	Setting	Description
_0001_TAU_OVERFLOW_OCCURS	0x0001U	Overflow occurrence detection
PULSEWIDTH_RANGE_MIN	1836	Lower limit of the tolerable range for pulse interval measurement
PULSEWIDTH_RANGE_MAX	2535	Upper limit of the tolerable range for pulse interval measurement
TAU0_COUNT_CLOCK_1	0x00	Count clock 1 for TAU0. It is predefined as one of these values: 0x00: 32 MHz 0x01: 16 MHz 0x02: 8 MHz 0x03: 4 MHz 0x04: 2 MHz 0x05: 1 MHz
TAU0_COUNT_CLOCK_2	0x01	Count clock 2 for TAU0. Same as TAU0_COUNTCLOCK_1.

Note 1: The above PULSEWIDTH_RANGE_MIN and PULSEWIDTH_RANGE_MAX are calculated as follows.

PULSEWIDTH_RANGE_MIN = HOCO_FREQUENCY_MIN / LOCO_FREQUENCY_MAX

PULSEWIDTH_RANGE_MAX = HOCO_FREQUENCY_MAX / LOCO_FREQUENCY_MIN

HOCO_FREQUENCY_MIN:	Value (31.68 MHz) calculated by considering a tolerance of the HOCO frequency (–1 %)
LOCO_FREQUENCY_MAX:	Value (17.25 KHz) calculated by considering a tolerance of the low-speed on-chip oscillator frequency (+15 %)
HOCO_FREQUENCY_MAX:	Value (32.32 MHz) calculated by considering a tolerance of the HOCO frequency (+1 %)
LOCO_FREQUENCY_MIN:	Value (12.75 KHz) calculated by considering a tolerance of the low-speed on-chip oscillator frequency (-15 %)

Change the upper and lower limits of the tolerable range for pulse interval measurement depending on the system used.

5.5 List of Variables

Table 5.4 lists the global variables. Table 5.5 lists the static variables.

Table 5.4 Global Variables

Type	Variable Name	Contents	Function Used
volatile uint32_t	g_tau0_ch5_width	Buffer for pulse interval measurement values	r_tau0_channel5_interrupt

Table 5.5 Static Variables

Type	Variable Name	Contents	Function Used
uint8_t	g_MeasureEndFlag	Pulse interval measurement end flag	r_tau0_channel5_interrupt, R_TAU0_Channel5_Get_MeasureStatus, R_TAU0_Channel5_Clear_MeasureStatus
uint8_t	g_fCLKChangeFlag	Count clock change request flag	r_it_interrupt, R_IT_Get_fCLK_ChangeFlag, R_IT_Clear_fCLK_ChangeFlag
uint8_t	g_intp0_flag	INTP0 external interrupt occurrence flag	r_intc0_interrupt, R_INTC0_Get_INTP0_Flag, R_INTC0_Clear_INTP0_Flag
uint8_t	g_intit_flag	Interval timer interrupt occurrence flag	r_it_interrupt, R_IT_Get_INTIT_Flag, R_IT_Clear_INTIT_Flag

5.6 List of Functions

Table 5.6 lists the functions that are used in this sample program.

Table 5.6 List of Functions

Function Name	Outline
R_INTC0_Start	Starts INTP0 external interrupt processing.
R_TAU0_Channel5_MeasureStart	Starts pulse interval measurement.
R_TAU0_Channel5_Start	Starts channel 5 of TAU0.
R_TAU0_Channel5_Stop	Stops channel 5 of TAU0.
R_IT_Get_fCLK_ChangeFlag	Gets the count clock change request flag.
R_TAU0_Channel5_Get_MeasureStatus	Gets the pulse interval measurement end flag.
R_Main_Get_PulseWidthMeasureResult	Judges the pulse interval measurement value.
R_TAU0_Channel5_Get_PulseWidth	Gets the pulse interval measurement value.
R_TAU0_Channel5_Clear_MeasureStatus	Clears the pulse interval measurement end flag.
R_Main_fCLK_Change	Changes the count clock.
R_Main_HOCO_Change	Changes the HOCO clock.
R_Main_Start_LedBlink	Starts blinking the LED.
R_Main_Stop_LedBlink	Stops blinking the LED.
R_RTC_Start	Starts the real-time clock.
R_RTC_Stop	Stops the real-time clock.
R_RTC_Set_ConstPeriodInterruptOn	Starts real-time clock fixed-frequency interrupts.
R_RTC_Set_ConstPeriodInterruptOff	Stops real-time clock fixed-frequency interrupts.
R_IT_Clear_fCLK_ChangeFlag	Clears the count clock change request flag.
r_it_interrupt	Interval timer interrupt
r_tau0_channel5_interrupt	Capture end interrupt for channel 5 of TAU0
r_intc0_interrupt	INTP0 external interrupt
r_rtc_interrupt	RTC fixed-frequency interrupt
r_rtc_callback_constperiod	Callback function for RTC fixed-frequency interrupts
R_IT_Start	Starts the interval timer.
R_IT_Stop	Stops the interval timer.
R_INTC0_Get_INTP0_Flag	Gets the INTP0 external interrupt occurrence flag.
R_INTC0_Clear_INTP0_Flag	Clears the INTP0 external interrupt occurrence flag.
R_IT_Get_INTIT_Flag	Gets the interval timer interrupt occurrence flag.
R_IT_Clear_INTIT_Flag	Clears the interval timer interrupt occurrence flag.

5.7 Function Specifications

This section describes the specifications for the functions that are used in the sample code.

[Function Name] R_INTC0_Start

Synopsis	Starts INTP0 external interrupt processing.
Header	r_cg_macrodriver.h r_cg_intc.h r_cg_userdefine.h
Declaration	void R_INTC0_Start(void)
Explanation	This function unmasks INTP0 external interrupts.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_TAU0_Channel5_MeasureStart

Synopsis	Discards pulse interval measurement.
Header	r_cg_macrodriver.h r_cg_timer.h r_cg_userdefine.h
Declaration	void R_TAU0_Channel5_MeasureStart(void)
Explanation	This function waits until the first pulse interval measurement after the start of the timer (TM05). Then, it clears the interrupt flag.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_TAU0_Channel5_Start

Synopsis	Starts channel 5 of TAU0.
Header	r_cg_macrodriver.h r_cg_timer.h r_cg_userdefine.h
Declaration	void R_TAU0_Channel5_Start(void)
Explanation	This function unmasks interrupts for channel 5 of TAU0 and starts counting.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_TAU0_Channel5_Stop

Synopsis	Stops channel 5 of TAU0.
Header	r_cg_macrodriver.h r_cg_timer.h r_cg_userdefine.h
Declaration	void R_TAU0_Channel5_Stop(void)
Explanation	This function masks interrupts for channel 5 of TAU0 and stops counting.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_IT_Get_fCLK_ChangeFlag

Synopsis	Gets the count clock change request flag.
Header	r_cg_macrodriver.h r_cg_it.h r_cg_userdefine.h
Declaration	uint8_t R_IT_Get_fCLK_ChangeFlag(void)
Explanation	This function gets the count clock change request flag.
Arguments	None
Return value	• When count clock change is not requested: 0x00 • When count clock change is requested: 0x01
Remarks	None

[Function Name] R_TAU0_Channel5_Get_MeasureStatus

Synopsis	Gets the pulse interval measurement end flag.
Header	r_cg_macrodriver.h r_cg_timer.h r_cg_userdefine.h
Declaration	uint8_t R_TAU0_Channel5_Get_MeasureStatus
Explanation	This function gets the pulse interval measurement end flag.
Arguments	None
Return value	• When pulse interval measurement is not ended: 0x00 • When pulse interval measurement is ended: 0x01
Remarks	None

[Function Name] R_Main_Get_PulseWidthMeasureResult

Synopsis	Judges the pulse interval measurement value.
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_timer.h r_cg_rtc.h r_cg_it.h r_cg_userdefine.h
Declaration	uint8_t R_Main_Get_PulseWidthMeasureResult(void)
Explanation	This function judges the pulse interval measurement value. That is, it judges whether the global variable (g_Tau0Ch5Width) containing the pulse interval measurement value is within the tolerable range for pulse interval measurement in Table 5.3.
Arguments	None
Return value	• When the pulse interval measurement value is within the tolerable range: 0x00 • When the pulse interval measurement value is not within the tolerable range: 0x01
Remarks	None

[Function Name] R_TAU0_Channel5_Get_PulseWidth

Synopsis	Gets the pulse interval measurement value.
Header	r_cg_macrodriver.h, r_cg_timer.h, r_cg_userdefine.h
Declaration	void R_TAU0_Channel5_Get_PulseWidth(uint32_t *width)
Explanation	This function gets the pulse interval measurement value. That is, it stores the pulse interval measurement value in the global variable (g_Tau0Ch5Width).
Arguments	• width — Address of the area to store the pulse interval measurement value
Return value	None
Remarks	None

[Function Name] R_TAU0_Channel5_Clear_MeasureStatus

Synopsis	Clears the pulse interval measurement end flag.
Header	r_cg_macrodriver.h, r_cg_timer.h r_cg_userdefine.h
Declaration	void R_TAU0_Channel5_Clear_MeasureStatus(void)
Explanation	This function clears the pulse interval measurement end flag to 0.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_Main_fCLK_Change

Synopsis	Changes the count clock.
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_timer.h r_cg_rtc.h r_cg_it.h r_cg_userdefine.h
Declaration	void R_Main_fCLK_Change(void)
Explanation	This function stops channel 5 of TAU0. Then, it changes the HOCO clock which is a count clock for TAU0.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_Main_HOCO_Change

Synopsis	Changes the HOCO clock.
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_timer.h r_cg_rtc.h r_cg_it.h r_cg_userdefine.h
Declaration	void R_Main_HOCO_Change(uint8_t clock)
Explanation	This function changes the HOCO clock to the clock set by TAU0_COUNT_CLOCK_1 or the clock set by TAU0_COUNT_CLOCK_2. These constants are listed in Table 5.3.
Arguments	• clock — TAU0_COUNT_CLOCK1 or TAU0_COUNT_CLOCK2 0x00: 32 MHz 0x01: 16 MHz 0x02: 8 MHz 0x03: 4 MHz 0x04: 2 MHz 0x05: 1 MHz
Return value	None
Remarks	None

[Function Name] R_Main_Start_LedBlink

Synopsis	Starts blinking the LED.
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_timer.h r_cg_rtc.h r_cg_it.h r_cg_userdefine.h
Declaration	void R_Main_Start_LedBlink(void)
Explanation	This function starts blinking the LED.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_Main_Stop_LedBlink

Synopsis	Stops blinking the LED.
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_timer.h r_cg_rtc.h r_cg_it.h r_cg_userdefine.h
Declaration	void R_Main_Stop_LedBlink(void)
Explanation	This function stops blinking the LED.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_RTC_Start

Synopsis	Starts the real-time clock.
Header	r_cg_macrodriver.h r_cg_rtc.h r_cg_userdefine.h
Declaration	void R_RTC_Start(void)
Explanation	This function unmasks real-time clock interrupts and starts the real time clock.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_RTC_Stop

Synopsis	Stops the real-time clock.
Header	r_cg_macrodriver.h r_cg_rtc.h r_cg_userdefine.h
Declaration	void R_RTC_Stop(void)
Explanation	This function stops the real-time clock and masks real-time clock interrupts.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_RTC_Set_ConstPeriodInterruptOn

Synopsis	Starts real-time clock fixed-frequency interrupts.	
Header	r_cg_macrodriver.h r_cg_rtc.h r_cg_userdefine.h	
Declaration	MD_STATUS R_RTC_Set_ConstPeriodInterruptOn(enum RTCINTPeriod period)	
Explanation	This function sets the interrupt INTRTC occurrence frequency and then starts fixed-frequency interrupts.	
Arguments	• period	Period between fixed-frequency interrupts HALFSEC: 0.5 second x (f_{SUB}/f_{IL}) ONESEC: 1 second × (f_{SUB}/f_{IL}) ONEMIN: 1 minute × (f_{SUB}/f_{IL}) ONEHOUR: 1 hour × (f_{SUB}/f_{IL}) ONEDAY: 1 day × (f_{SUB}/f_{IL}) ONEMONTH: 1 year × (f_{SUB}/f_{IL})
Return value	• Normal end: MD_OK (0x00) • Argument specification error: MD_ARGERROR (0x01)	
Remarks	None	

[Function Name] R_RTC_Set_ConstPeriodInterruptOff

Synopsis	Stops real-time clock fixed-frequency interrupts.
Header	r_cg_macrodriver.h, r_cg_rtc.h, r_cg_userdefine.h
Declaration	void R_RTC_Set_ConstPeriodInterruptOff(void)
Explanation	This function stops fixed-frequency interrupts.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_IT_Clear_fCLK_ChangeFlag

Synopsis	Clears the count clock change request flag.
Header	r_cg_macrodriver.h r_cg_it.h r_cg_userdefine.h
Declaration	void R_IT_Clear_fCLK_ChangeFlag(void)
Explanation	This function clears the count clock change request flag to 0.
Arguments	None
Return value	None
Remarks	None

[Function Name] r_it_interrupt

Synopsis	Interval timer interrupt
Header	r_cg_macrodriver.h r_cg_it.h r_cg_userdefine.h
Declaration	static void __near r_it_interrupt(void)
Explanation	If P137 is 0, this function sets the count clock change request flag to 1. It sets the interval timer interrupt occurrence flag to 1.
Arguments	None
Return value	None
Remarks	None

[Function Name] r_tau0_channel5_interrupt

Synopsis	Capture end interrupt for channel 5 of TAU0
Header	r_cg_macrodriver.h r_cg_timer.h r_cg_userdefine.h
Declaration	static void __near r_tau0_channel5_interrupt(void)
Explanation	This function gets a pulse interval each time a transition to this interrupt processing occurs. It sets the pulse interval measurement end flag to 1.
Arguments	None
Return value	None
Remarks	None

[Function Name] r_intc0_interrupt

Synopsis	INTP0 external interrupt
Header	r_cg_macrodriver.h r_cg_intc.h r_cg_it.h r_cg_userdefine.h
Declaration	static void __near r_intc0_interrupt(void)
Explanation	This function starts the interval timer. It sets the INTP0 external interrupt occurrence flag to 1.
Arguments	None
Return value	None
Remarks	None

[Function Name] r_rtc_interrupt

Synopsis	RTC fixed-frequency interrupt
Header	r_cg_macrodriver.h r_cg_rtc.h r_cg_userdefine.h
Declaration	static void __near r_rtc_interrupt(void)
Explanation	This function calls the callback function for RTC fixed-frequency interrupts.
Arguments	None
Return value	None
Remarks	None

[Function Name] r_rtc_callback_constperiod

Synopsis	Callback function for RTC fixed-frequency interrupts
Header	r_cg_macrodriver.h r_cg_rtc.h r_cg_userdefine.h
Declaration	static void r_rtc_callback_constperiod(void)
Explanation	This function inverts the display on the LED1.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_IT_Start

Synopsis	Starts the interval timer.
Header	r_cg_macrodriver.h r_cg_it.h r_cg_userdefine.h
Declaration	void R_IT_Start(void)
Explanation	This function starts the interval timer and unmasks interval timer interrupts.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_IT_Stop

Synopsis	Stops the interval timer.
Header	r_cg_macrodriver.h r_cg_it.h r_cg_userdefine.h
Declaration	void R_IT_Stop(void)
Explanation	This function masks interval timer interrupts and stops the interval timer.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_INTC0_Get_INTP0_Flag

Synopsis	Gets the INTP0 external interrupt occurrence flag.
Header	r_cg_macrodriver.h r_cg_intc.h r_cg_it.h r_cg_userdefine.h
Declaration	uint8_t R_INTC0_Get_INTP0_Flag(void)
Explanation	This function gets the INTP0 external interrupt occurrence flag.
Arguments	None
Return value	• When an INT0 external interrupt has not occurred: 0x00 • When an INT0 external interrupt has occurred: 0x01
Remarks	None

[Function Name] R_INTC0_Clear_INTP0_Flag

Synopsis	Clears the INTP0 external interrupt occurrence flag.
Header	r_cg_macrodriver.h r_cg_intc.h r_cg_it.h r_cg_userdefine.h
Declaration	void R_INTC0_Clear_INTP0_Flag(void)
Explanation	This function clears the INTP0 external interrupt occurrence flag.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_IT_Get_INTIT_Flag

Synopsis	Gets the interval timer interrupt occurrence flag.
Header	r_cg_macrodriver.h r_cg_it.h r_cg_userdefine.h
Declaration	uint8_t R_IT_Get_INTIT_Flag(void)
Explanation	This function gets the interval timer interrupt occurrence flag.
Arguments	None
Return value	• When an interval timer interrupt has not occurred: 0x00 • When an interval timer interrupt has occurred: 0x01
Remarks	None

[Function Name] R_IT_Clear_INTIT_Flag

Synopsis	Clears the interval timer interrupt occurrence flag.
Header	r_cg_macrodriver.h r_cg_it.h r_cg_userdefine.h
Declaration	uint8_t R_IT_Clear_INTIT_Flag(void)
Explanation	This function clears the interval timer interrupt occurrence flag.
Arguments	None
Return value	None
Remarks	None

5.8 Flowcharts

Figure 5.1 shows the overall flow of the sample program described in this application note.

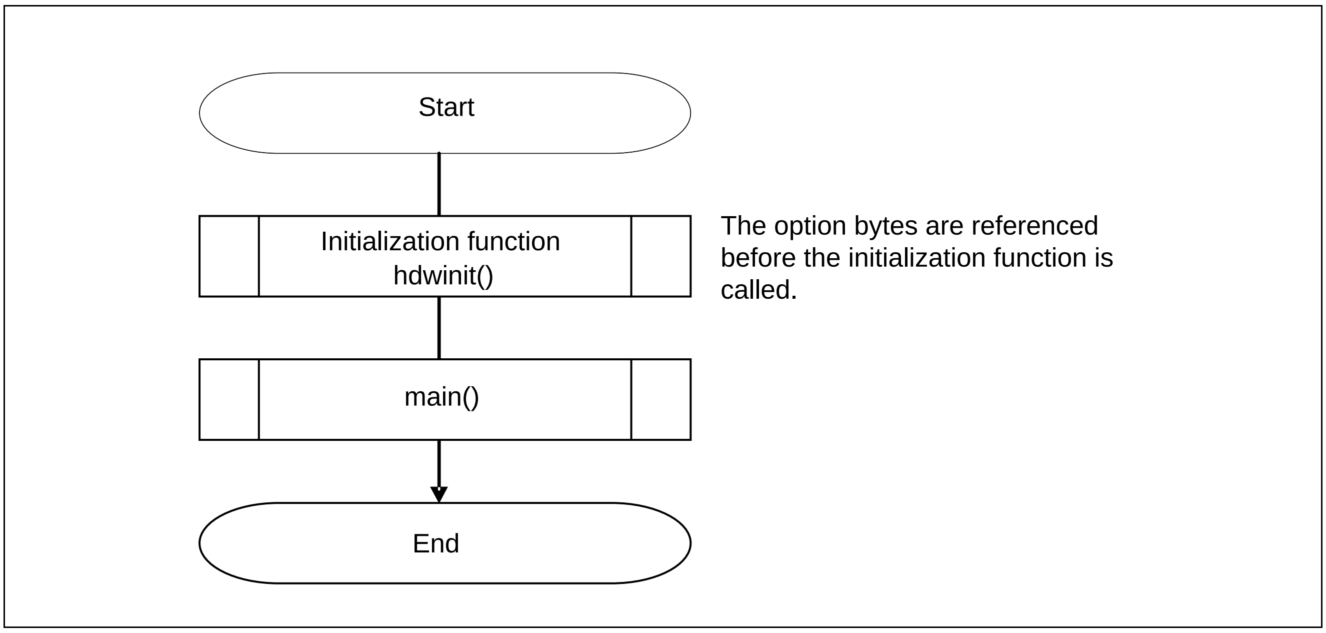

Start
Initialization function hdwinit()
The option bytes are referenced before the initialization function is called.
main()
End

Figure 5.1 Overall Flow

5.8.1 Initialization Function

Figure 5.2 shows the flowchart for the initialization function.

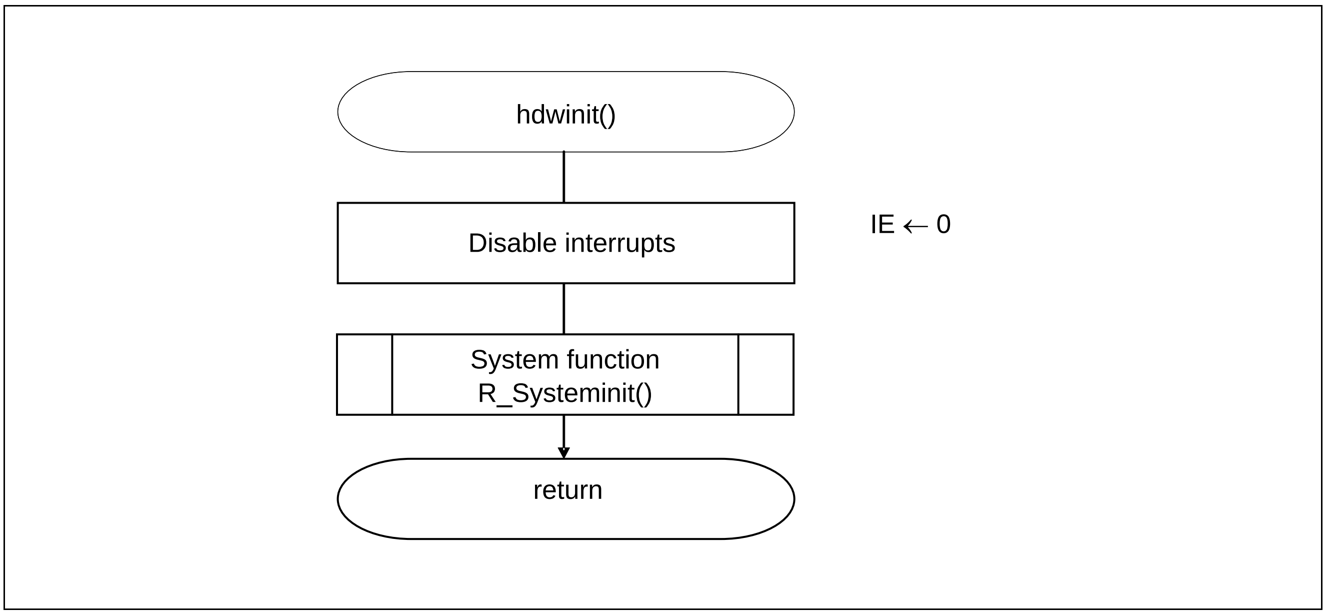


Figure 5.2 Initialization Function

5.8.2 System Function

Figure 5.3 shows the flowchart for the system function.

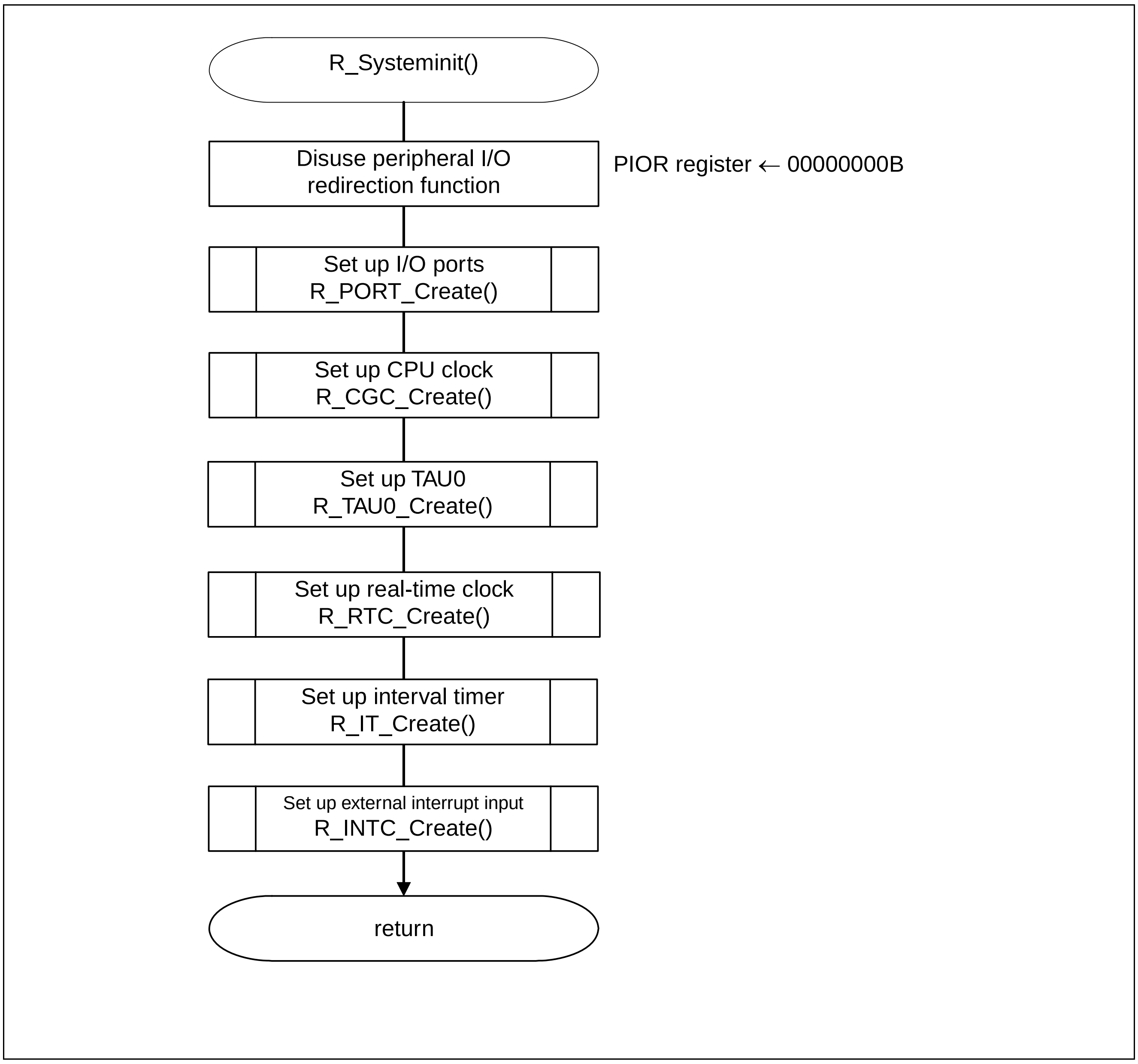

R_Systeminit()
Disuse peripheral I/O redirection function
PIOR register ← 00000000B
Set up I/O ports R_PORT_Create()
Set up CPU clock R_CGC_Create()
Set up TAU0 R_TAU0_Create()
Set up real-time clock R_RTC_Create()
Set up interval timer R_IT_Create()
Set up external interrupt input R_INTC_Create()
return

Figure 5.3 System Function

5.8.3 I/O Port Setup

Figure 5.4 shows the flowchart for I/O port setup.

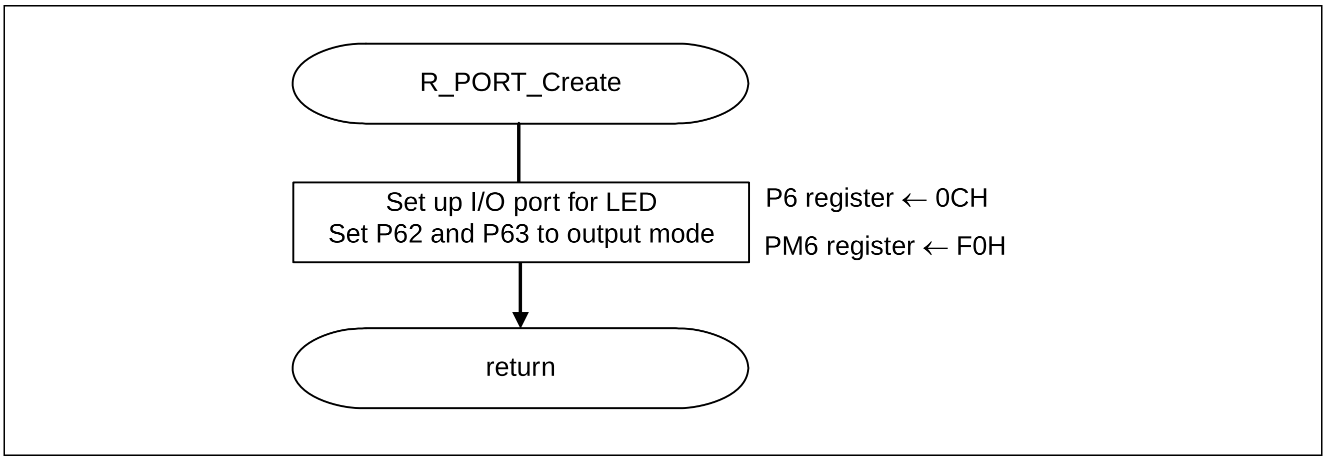


Figure 5.4 I/O Port Setup

Cautions:
1. Refer to the section entitled "Flowcharts" in RL78/G13 Initialization Application Note (R01AN2575E) for the configuration of the unused ports.
2. Provide proper treatment for unused pins so that their electrical specifications are met. Connect each of any unused input-only ports to V_{DD} or V_{SS} via a separate resistor.

5.8.4 CPU Clock Setup

Figure 5.5 shows the flowchart for setting up the CPU clock.

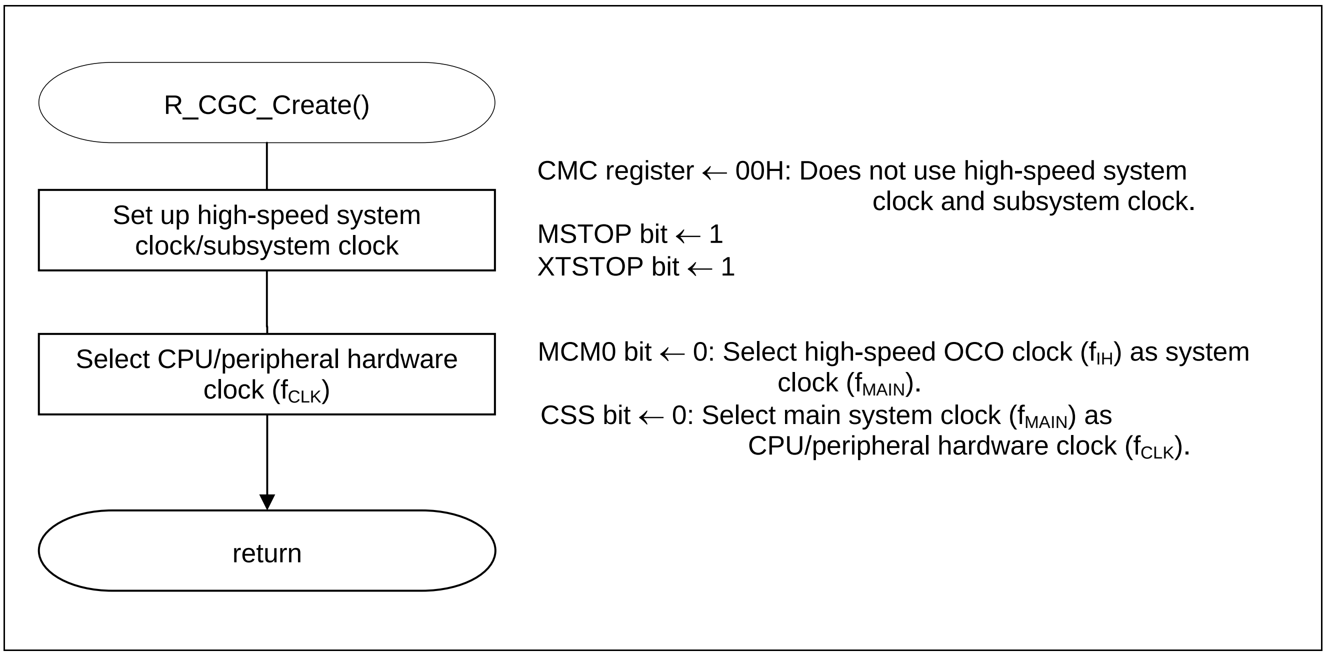

R_CGC_Create()
Set up high-speed system clock/subsystem clock
Select CPU/peripheral hardware clock (f_{CLK})
return
CMC register ← 00H: Does not use high-speed system clock and subsystem clock.
MSTOP bit ← 1
XTSTOP bit ← 1
MCM0 bit ← 0: Select high-speed OCO clock (f_{IH}) as system clock (f_{MAIN}).
CSS bit ← 0: Select main system clock (f_{MAIN}) as CPU/peripheral hardware clock (f_{CLK}).

Figure 5.5 CPU Clock Setup

Caution: For details on the procedure for setting up the CPU clock (R_CGC_Create ()), refer to the section entitled "Flowcharts" in RL78/G13 Initialization Application Note (R01AN2575E).

5.8.5 TAU0 Setup

Figure 5.6 shows the flowchart for setting up the TAU0.

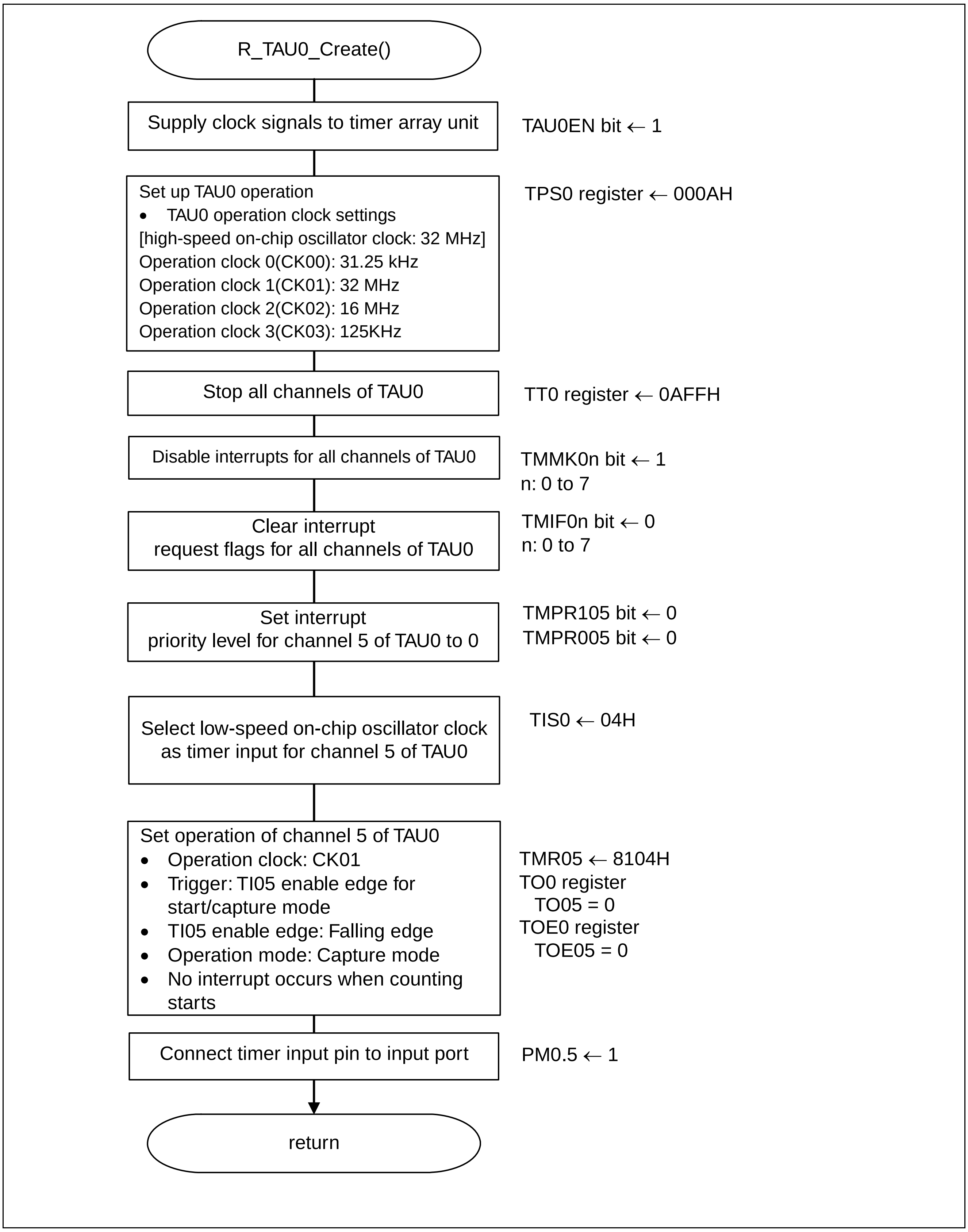

R_TAU0_Create()
Supply clock signals to timer array unit
TAU0EN bit ← 1
Set up TAU0 operation
• TAU0 operation clock settings
[high-speed on-chip oscillator clock: 32 MHz]
Operation clock 0(CK00): 31.25 kHz
Operation clock 1(CK01): 32 MHz
Operation clock 2(CK02): 16 MHz
Operation clock 3(CK03): 125KHz
TPS0 register ← 000AH
Stop all channels of TAU0
TT0 register ← 0AFFH
Disable interrupts for all channels of TAU0
TMMK0n bit ← 1
n: 0 to 7
Clear interrupt request flags for all channels of TAU0
TMIF0n bit ← 0
n: 0 to 7
Set interrupt priority level for channel 5 of TAU0 to 0
TMPR105 bit ← 0
TMPR005 bit ← 0
Select low-speed on-chip oscillator clock as timer input for channel 5 of TAU0
TIS0 ← 04H
Set operation of channel 5 of TAU0
• Operation clock: CK01
• Trigger: TI05 enable edge for start/capture mode
• TI05 enable edge: Falling edge
• Operation mode: Capture mode
• No interrupt occurs when counting starts
TMR05 ← 8104H
TO0 register
TO05 = 0
TOE0 register
TOE05 = 0
Connect timer input pin to input port
PM0.5 ← 1
return

Figure 5.6 TAU 0 Setup

Starting clock signal supply to the timer array unit

- Peripheral enable register 0 (PER0)
  : Supply clock signals to the timer array unit.

Symbol: PER0

7	6	5	4	3	2	1	0
RTCEN	IICA1EN	ADCEN	IICA0EN	SAU1EN	SAU0EN	TAU1EN	TAU0EN
0	0	0	0	0	0	0	**1**

Bit 0

TAU0EN	Control of timer array unit input clock supply
0	Stops input clock supply.
1	**Enables input clock supply.**

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

Setting the clock frequency

- Timer clock select register 0 (TPS0)
  Select the CK01 operation clock.

  Symbol: TPS0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	PRS 031	PRS 030	0	0	PRS 021	PRS 020	PRS 013	PRS 012	PRS 011	PRS 010	PRS 003	PRS 002	PRS 001	PRS 000
0	0	**0**	**0**	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**

PRS 003	PRS 002	PRS 001	PRS 000	Selection of operation clock (CK00)					
					f_{CLK} = 2 MHz	f_{CLK} = 5 MHz	f_{CLK} = 10 MHz	f_{CLK} = 20 MHz	f_{CLK} = 32 MHz
0	0	0	0	f_{CLK}	2 MHz	5 MHz	10 MHz	20 MHz	32 MHz
0	0	0	1	$f_{CLK}/2$	1 MHz	2.5 MHz	5 MHz	10 MHz	16 MHz
0	0	1	0	$f_{CLK}/2^{2}$	500 kHz	1.25 MHz	2.5 MHz	5 MHz	8 MHz
0	0	1	1	$f_{CLK}/2^{3}$	250 kHz	625 kHz	1.25 MHz	2.5 MHz	4 MHz
0	1	0	0	$f_{CLK}/2^{4}$	125 kHz	312.5 kHz	625 kHz	1.25 MHz	2 MHz
0	1	0	1	$f_{CLK}/2^{5}$	62.5 kHz	156.2 kHz	312.5 KHz	625 kHz	1 MHz
0	1	1	0	$f_{CLK}/2^{6}$	31.25 kHz	78.1 kHz	156.2 kHz	312.5 kHz	500 kHz
0	1	1	1	$f_{CLK}/2^{7}$	15.62 kHz	39.1 kHz	78.1 kHz	156.2 kHz	250 kHz
1	0	0	0	$f_{CLK}/2^{8}$	7.81 kHz	19.5 kHz	39.1 kHz	78.1 kHz	125 kHz
1	0	0	1	$f_{CLK}/2^{9}$	3.91 kHz	9.76 kHz	19.5 kHz	39.1 kHz	62.5 kHz
1	**0**	**1**	**0**	$\mathbf{f_{CLK}/2^{10}}$	**1.95 kHz**	**4.88 kHz**	**9.76 kHz**	**19.5 kHz**	**31.25 kHz**
1	0	1	1	$f_{CLK}/2^{11}$	976 Hz	2.44 kHz	4.88 kHz	9.76 kHz	15.63 kHz
1	1	0	0	$f_{CLK}/2^{12}$	488 Hz	1.22 kHz	2.44 kHz	4.88 kHz	7.81 kHz
1	1	0	1	$f_{CLK}/2^{13}$	244 Hz	610 Hz	1.22 kHz	2.44 kHz	3.91 kHz
1	1	1	0	$f_{CLK}/2^{14}$	122 Hz	305 Hz	610 Hz	1.22 kHz	1.95 kHz
1	1	1	1	$f_{CLK}/2^{15}$	61 Hz	153 Hz	305 Hz	610 Hz	976 Hz

PRS 013	PRS 012	PRS 011	PRS 010	Selection of operation clock (CK01)					
					f_{CLK} = 2 MHz	f_{CLK} = 5 MHz	f_{CLK} = 10 MHz	f_{CLK} = 20 MHz	f_{CLK} = 32 MHz
0	**0**	**0**	**0**	$\mathbf{f_{CLK}}$	**2 MHz**	**5 MHz**	**10 MHz**	**20 MHz**	**32 MHz**
0	0	0	1	$f_{CLK}/2$	1 MHz	2.5 MHz	5 MHz	10 MHz	16 MHz
0	0	1	0	$f_{CLK}/2^{2}$	500 kHz	1.25 MHz	2.5 MHz	5 MHz	8 MHz
0	0	1	1	$f_{CLK}/2^{3}$	250 kHz	625 kHz	1.25 MHz	2.5 MHz	4 MHz
0	1	0	0	$f_{CLK}/2^{4}$	125 kHz	312.5 kHz	625 kHz	1.25 MHz	2 MHz
0	1	0	1	$f_{CLK}/2^{5}$	62.5 kHz	156.2 kHz	312.5 KHz	625 kHz	1 MHz
0	1	1	0	$f_{CLK}/2^{6}$	31.25 kHz	78.1 kHz	156.2 kHz	312.5 kHz	500 kHz
0	1	1	1	$f_{CLK}/2^{7}$	15.62 kHz	39.1 kHz	78.1 kHz	156.2 kHz	250 kHz
1	0	0	0	$f_{CLK}/2^{8}$	7.81 kHz	19.5 kHz	39.1 kHz	78.1 kHz	125 kHz
1	0	0	1	$f_{CLK}/2^{9}$	3.91 kHz	9.76 kHz	19.5 kHz	39.1 kHz	62.5 kHz
1	0	1	0	$f_{CLK}/2^{10}$	1.95 kHz	4.88 kHz	9.76 kHz	19.5 kHz	31.25 kHz
1	0	1	1	$f_{CLK}/2^{11}$	976 Hz	2.44 kHz	4.88 kHz	9.76 kHz	15.63 kHz
1	1	0	0	$f_{CLK}/2^{12}$	488 Hz	1.22 kHz	2.44 kHz	4.88 kHz	7.81 kHz
1	1	0	1	$f_{CLK}/2^{13}$	244 Hz	610 Hz	1.22 kHz	2.44 kHz	3.91 kHz
1	1	1	0	$f_{CLK}/2^{14}$	122 Hz	305 Hz	610 Hz	1.22 kHz	1.95 kHz
1	1	1	1	$f_{CLK}/2^{15}$	61 Hz	153 Hz	305 Hz	610 Hz	976 Hz

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

Controlling the channel trigger operation

- Timer channel stop register 5 (TT05)
  Select the TAU0 stop trigger.

Symbol: TT0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	TT H03	0	TT H01	0	TT 07	TT 06	TT 05	TT 04	TT 03	TT 02	TT 01	TT 00
0	0	0	0	**1**	0	**1**	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Bit 5

TT05	Channel 5 stop trigger
0	No trigger operation
1	**Operation is stopped (stop trigger is generated).**

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

Setting up the channel 5 operation mode

- Timer mode register 05 (TMR05)
  : Specify the operation mode and start the software trigger.
  Select the operation clock.

Symbol: TMR05

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CKS 051	CKS 050	0	CCS 05	0	STS 052	STS 051	STS 050	CIS 051	CIS 050	0	0	MD 053	MD 052	MD 051	MD 050
1	**0**	0	**0**	**0**	**0**	**0**	**1**	**0**	**0**	0	0	**0**	**1**	**0**	**0**

Bits 3 to 0

MD 053	MD 052	MD 051	MD 050	Operation mode of channel 0
0	0	0	0	Interval timer mode. (Timer interrupt is not generated when counting is started)
			1	Interval timer mode. (Timer interrupt is generated when counting is started)
0	**1**	**0**	**0**	**Capture mode (Timer interrupt is not generated when counting is started)**
			1	Capture mode (Timer interrupt is generated when counting is started)
0	1	1	0	Event counter mode (Timer interrupt is not generated when counting is started)
1	0	0	0	One-count mode Start trigger is invalid during counting operation.
			1	One-count mode Start trigger is valid during counting operation.
1	1	0	0	Capture & one-count mode Timer interrupt is not generated when counting is started. Start trigger is invalid during counting operation.

Bits 7 and 6

CIS 001	CIS 000	Selection of TI00 pin input valid edge
0	**0**	**Falling edge**
0	1	Rising edge
1	0	Both edges (when low-level width is measured)
1	1	Both edges (when high-level width is measured)

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

Symbol: TMR05

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CKS 051	CKS 050	0	CCS 05	0	STS 052	STS 051	STS 050	CIS 051	CIS 050	0	0	MD 053	MD 052	MD 051	MD 050
1	**0**	0	**0**	**0**	**0**	**0**	**1**	**0**	**0**	0	0	**0**	**1**	**0**	**0**

Bits 10 to 8

STS 002	STS 001	STS 000	Setting of start trigger or capture trigger of channel 0
0	0	0	Only software trigger start is valid (other trigger sources are unselected).
0	**0**	**1**	**Valid edge of the TI00 pin input is used as both the start trigger and capture trigger.**
0	1	0	Both the edges of the TI00 pin input are used as a start trigger and a capture trigger.
1	0	0	Interrupt signal of the master channel is used (when the channel is used as a slave channel with the simultaneous channel operation function).

Bit 11

MASTER00	Selection between using channel n independently or simultaneously with another channel(as a slave or master)
0	**Operates in independent channel operation function or as slave channel 0 simultaneous channel operation function.**
1	Operates as master channel in simultaneous channel operation function.

Bit 12

CCS00	Selection of count clock (f_{TCLK}) of channel 0
0	**Operation clock f_{MCK} specified by the CKS000 and CKS001 bits**
1	Valid edge of a signal input to the TI00 pin

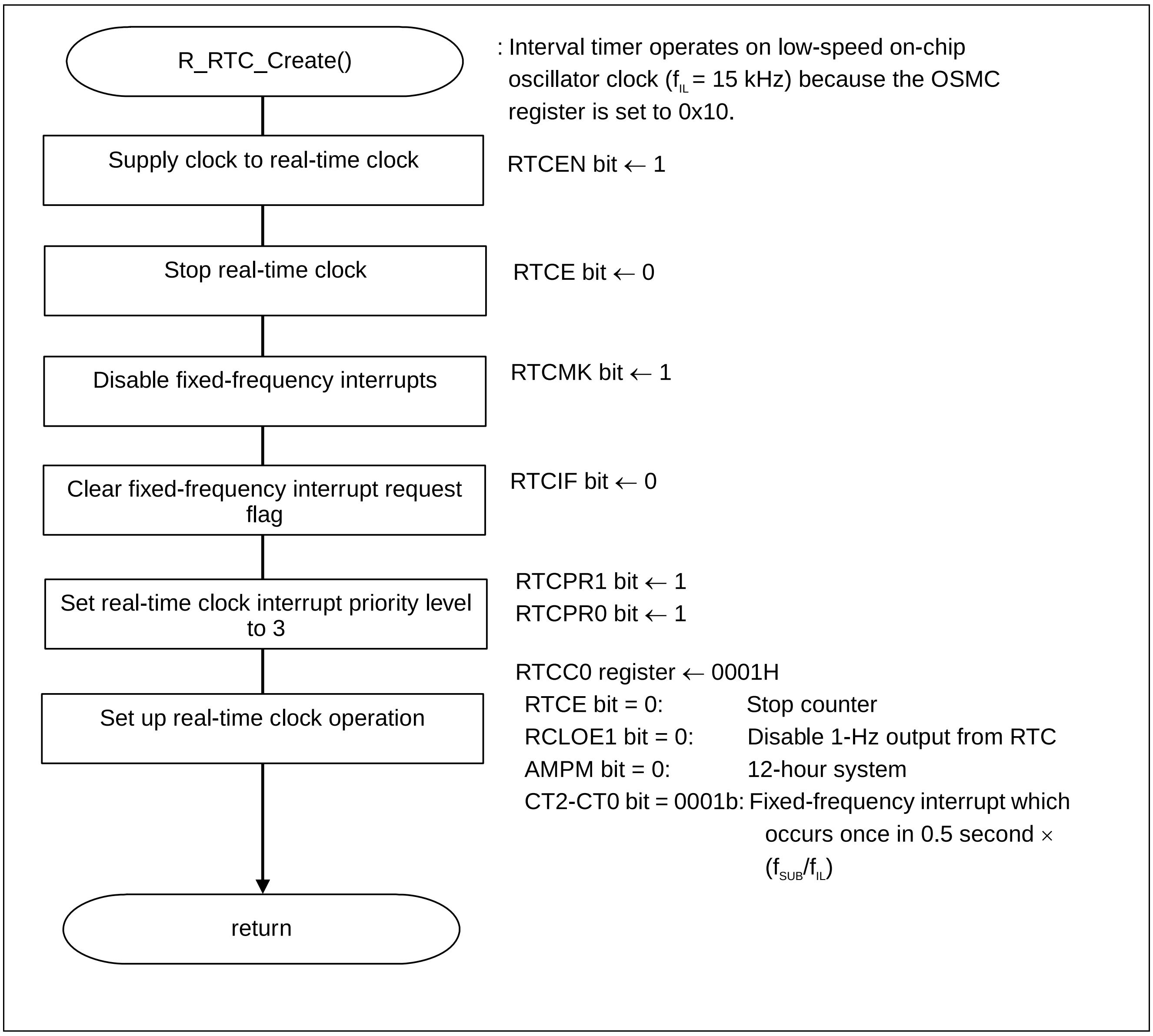
Bits 15 and 14

CKS001	CKS000	Selection of operation clock (f_{MCK}) of channel 0
0	0	Operation clock CK00 set by the PRS register
1	**0**	**Operation clock CK01 set by the PRS register**

Caution: For details on the register setup procedures, refer to RL78/G13 User's Manual: Hardware.

5.8.6 Real-Time Clock Setup

Figure 5.7 shows the flowchart for setting up the real-time clock.



R_RTC_Create()
: Interval timer operates on low-speed on-chip oscillator clock (f_{IL} = 15 kHz) because the OSMC register is set to 0x10.
Supply clock to real-time clock
RTCEN bit ← 1
Stop real-time clock
RTCE bit ← 0
Disable fixed-frequency interrupts
RTCMK bit ← 1
Clear fixed-frequency interrupt request flag
RTCIF bit ← 0
Set real-time clock interrupt priority level to 3
RTCPR1 bit ← 1
RTCPR0 bit ← 1
Set up real-time clock operation
RTCC0 register ← 0001H
RTCE bit = 0: Stop counter
RCLOE1 bit = 0: Disable 1-Hz output from RTC
AMPM bit = 0: 12-hour system
CT2-CT0 bit = 0001b: Fixed-frequency interrupt which occurs once in 0.5 second × (f_{SUB}/f_{IL})
return

Figure 5.7 Real-Time Clock Setup

5.8.7 Interval Timer Setup

Figure 5.8 shows the flowchart for setting up the interval timer.

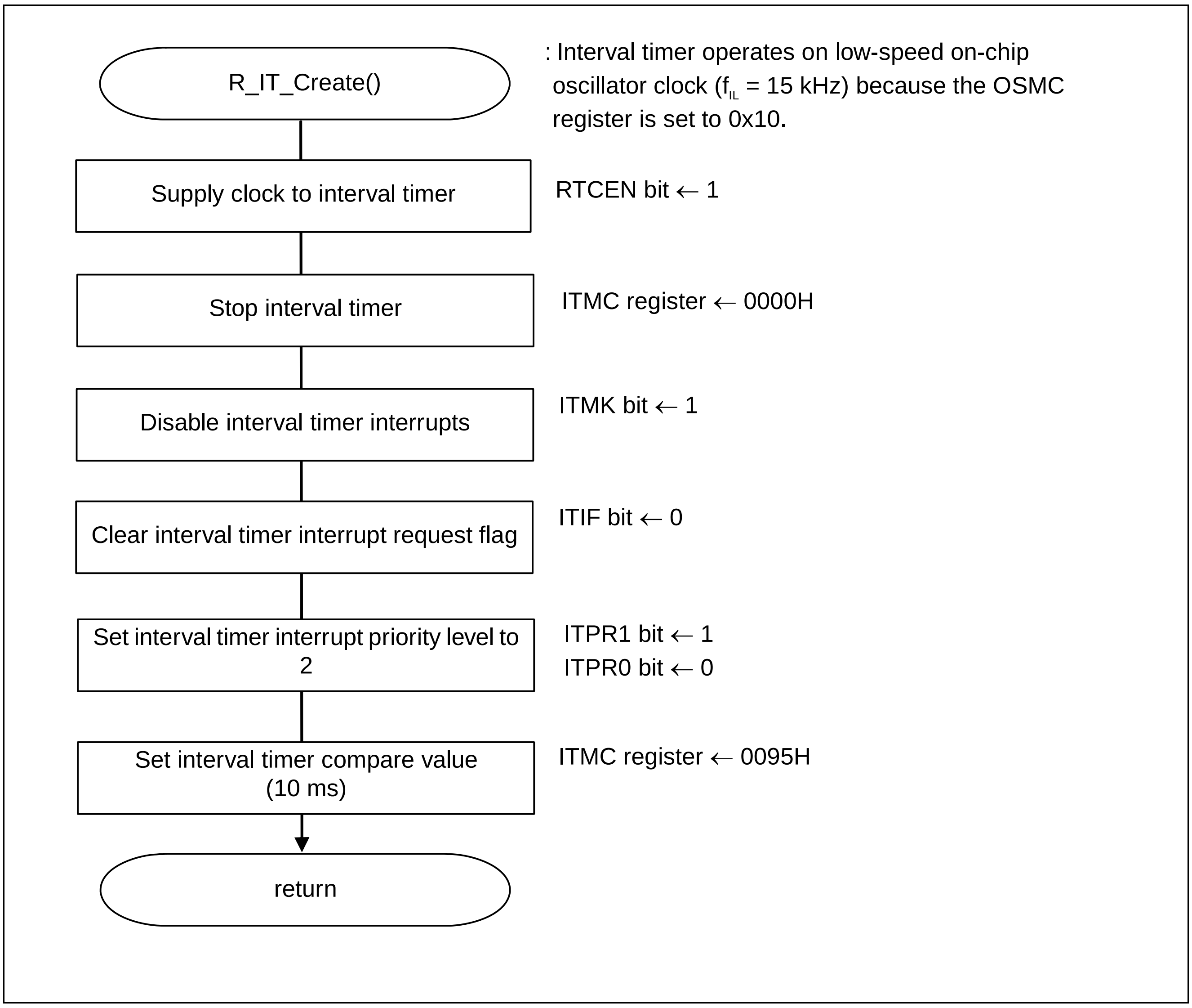

R_IT_Create()
: Interval timer operates on low-speed on-chip oscillator clock (f_{IL} = 15 kHz) because the OSMC register is set to 0x10.
Supply clock to interval timer
RTCEN bit ← 1
Stop interval timer
ITMC register ← 0000H
Disable interval timer interrupts
ITMK bit ← 1
Clear interval timer interrupt request flag
ITIF bit ← 0
Set interval timer interrupt priority level to 2
ITPR1 bit ← 1
ITPR0 bit ← 0
Set interval timer compare value (10 ms)
ITMC register ← 0095H
return

Figure 5.8 Interval Timer Setup

5.8.8 External Interrupt Input Setup

Figure 5.9 shows the flowchart for setting up the external interrupt input.

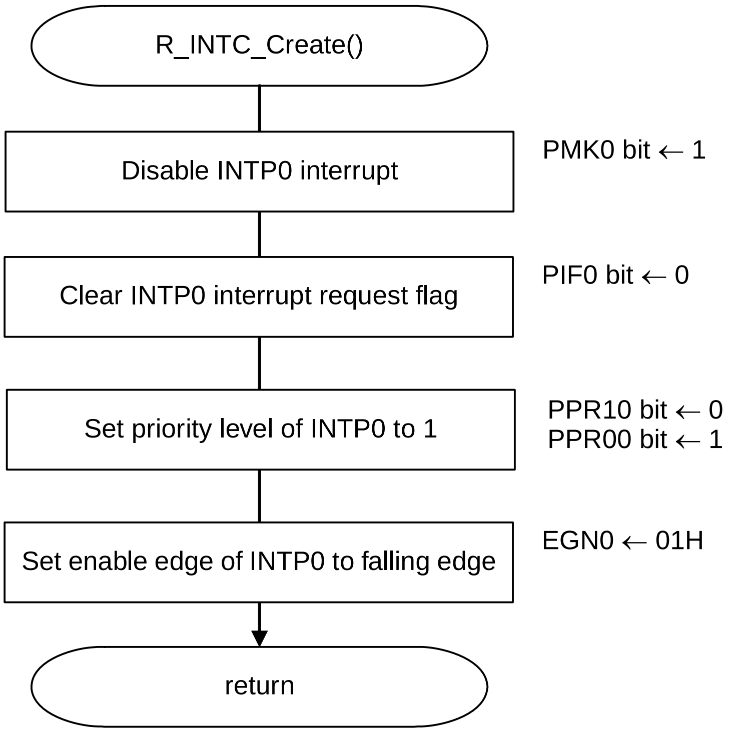

R_INTC_Create()
Disable INTP0 interrupt
PMK0 bit ← 1
Clear INTP0 interrupt request flag
PIF0 bit ← 0
Set priority level of INTP0 to 1
PPR10 bit ← 0
PPR00 bit ← 1
Set enable edge of INTP0 to falling edge
EGN0 ← 01H
return

Figure 5.9 External Interrupt Input Setup

5.8.9 Main Processing

Figure 5.10 shows the flowchart for the main processing.

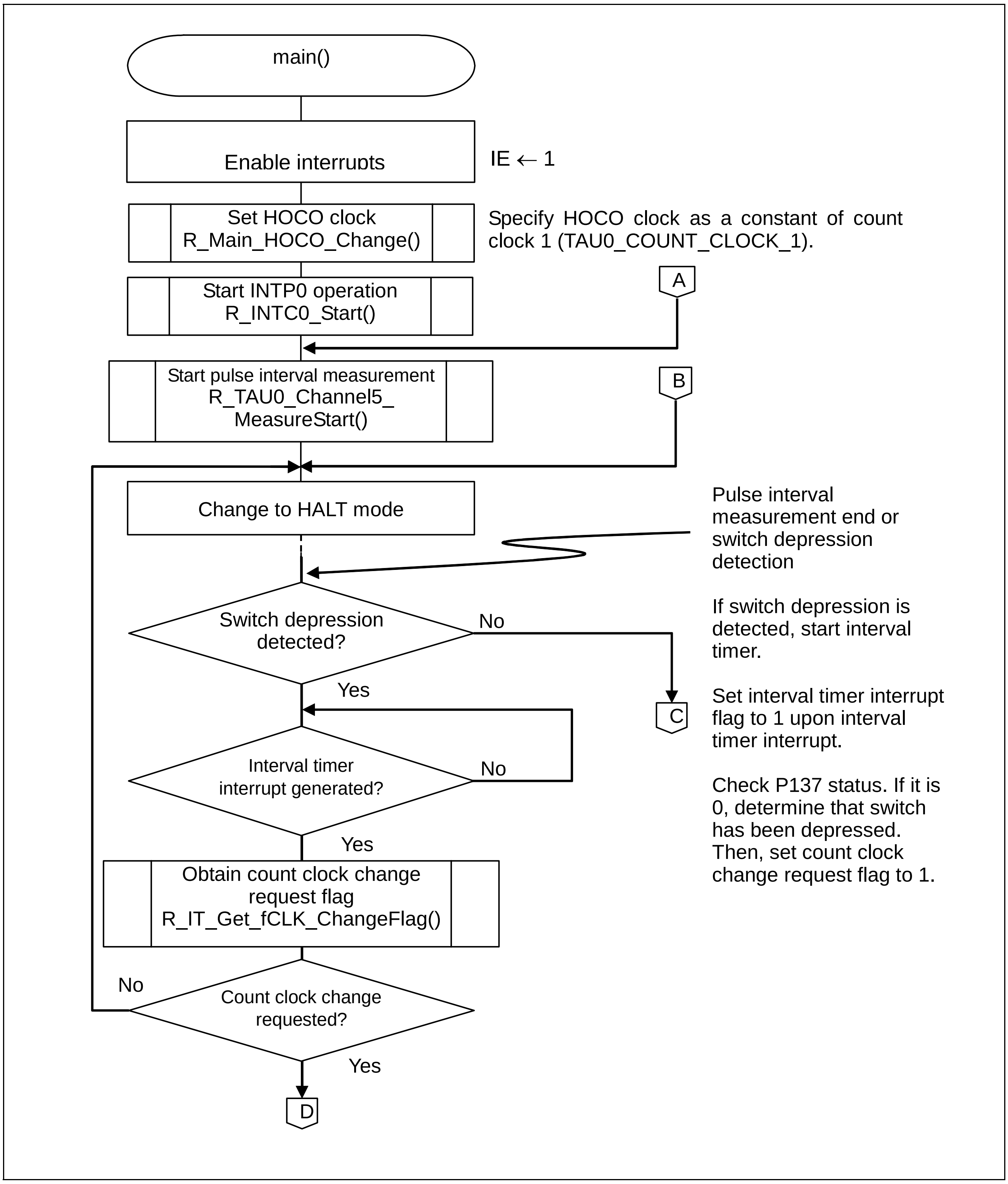

main()
Enable interrupts
IE ← 1
Set HOCO clock
R_Main_HOCO_Change()
Specify HOCO clock as a constant of count clock 1 (TAU0_COUNT_CLOCK_1).
Start INTP0 operation
R_INTC0_Start()
A
Start pulse interval measurement
R_TAU0_Channel5_
MeasureStart()
B
Change to HALT mode
Pulse interval measurement end or switch depression detection
Switch depression detected?
No
Yes
C
If switch depression is detected, start interval timer.
Interval timer interrupt generated?
No
Yes
Set interval timer interrupt flag to 1 upon interval timer interrupt.
Obtain count clock change request flag
R_IT_Get_fCLK_ChangeFlag()
Check P137 status. If it is 0, determine that switch has been depressed. Then, set count clock change request flag to 1.
No
Count clock change requested?
Yes
D

Figure 5.10 Main Processing (1/3)

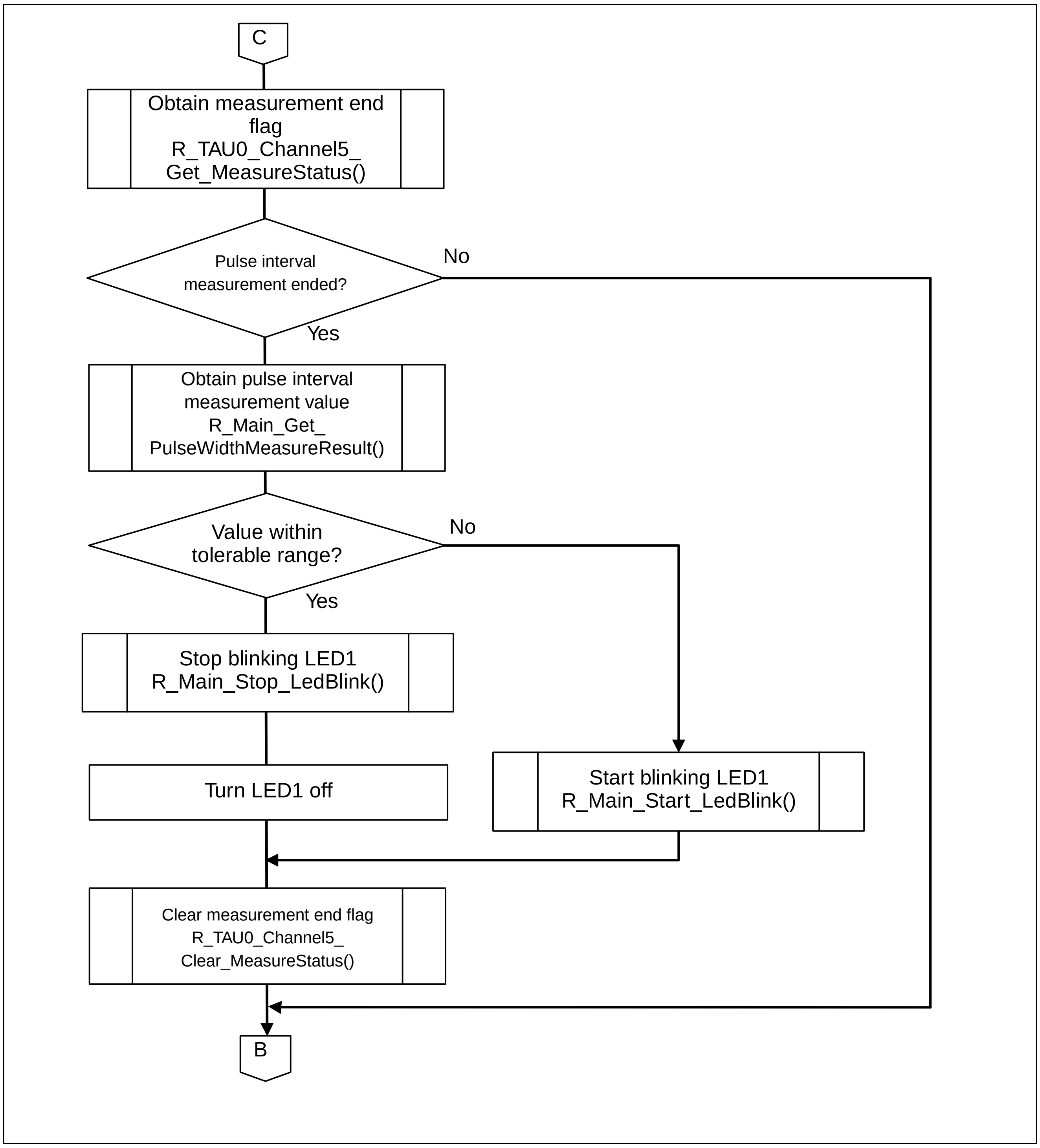


Figure 5.11 Main Processing (2/3)

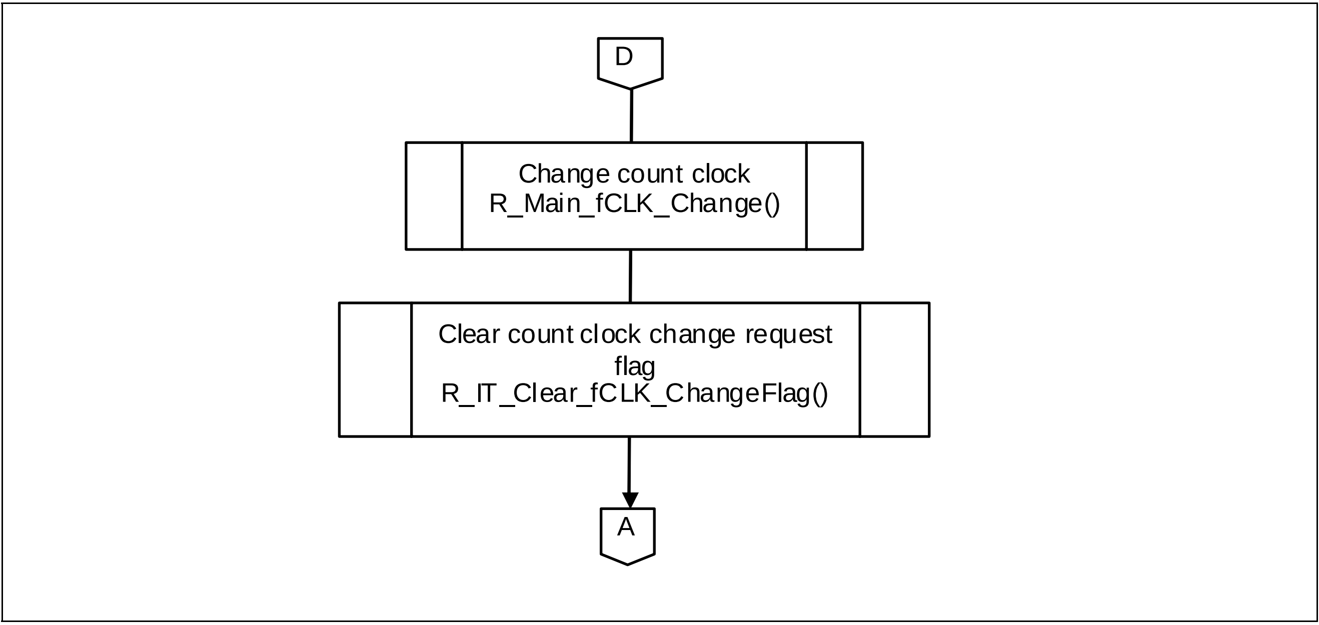

D
Change count clock
R_Main_fCLK_Change()
Clear count clock change request flag
R_IT_Clear_fCLK_ChangeFlag()
A

Figure 5.12 Main Processing (3/3)

5.8.10 Starting INTP0 External Interrupt Processing

Figure 5.13 shows the flowchart for starting INTP0 external interrupt processing.

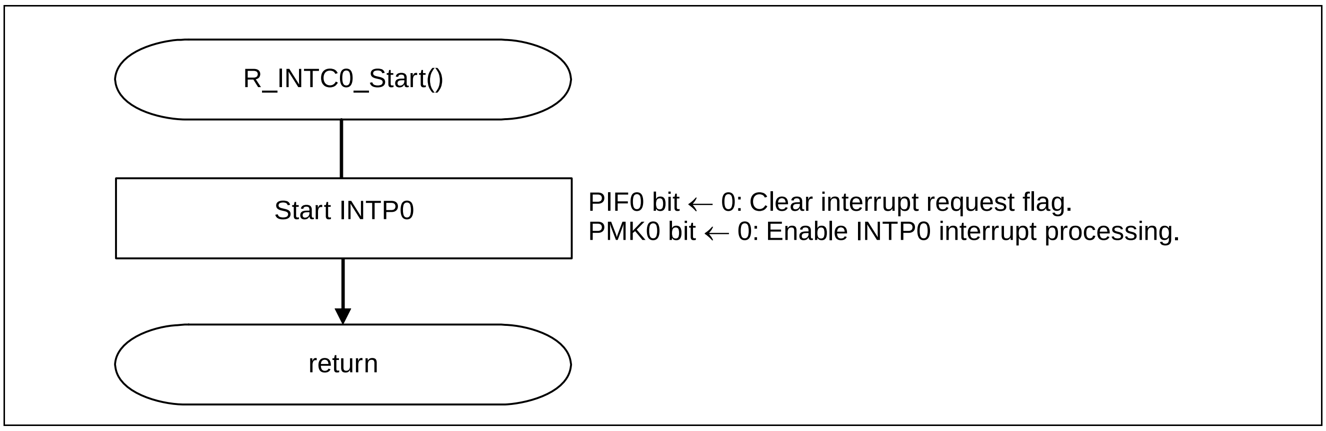

R_INTC0_Start()
Start INTP0
PIF0 bit ← 0: Clear interrupt request flag.
PMK0 bit ← 0: Enable INTP0 interrupt processing.
return

Figure 5.13 Starting INTP0 External Interrupt Processing

5.8.11 Starting Pulse Interval Measurement

Figure 5.14 shows the flowchart for starting pulse interval measurement.

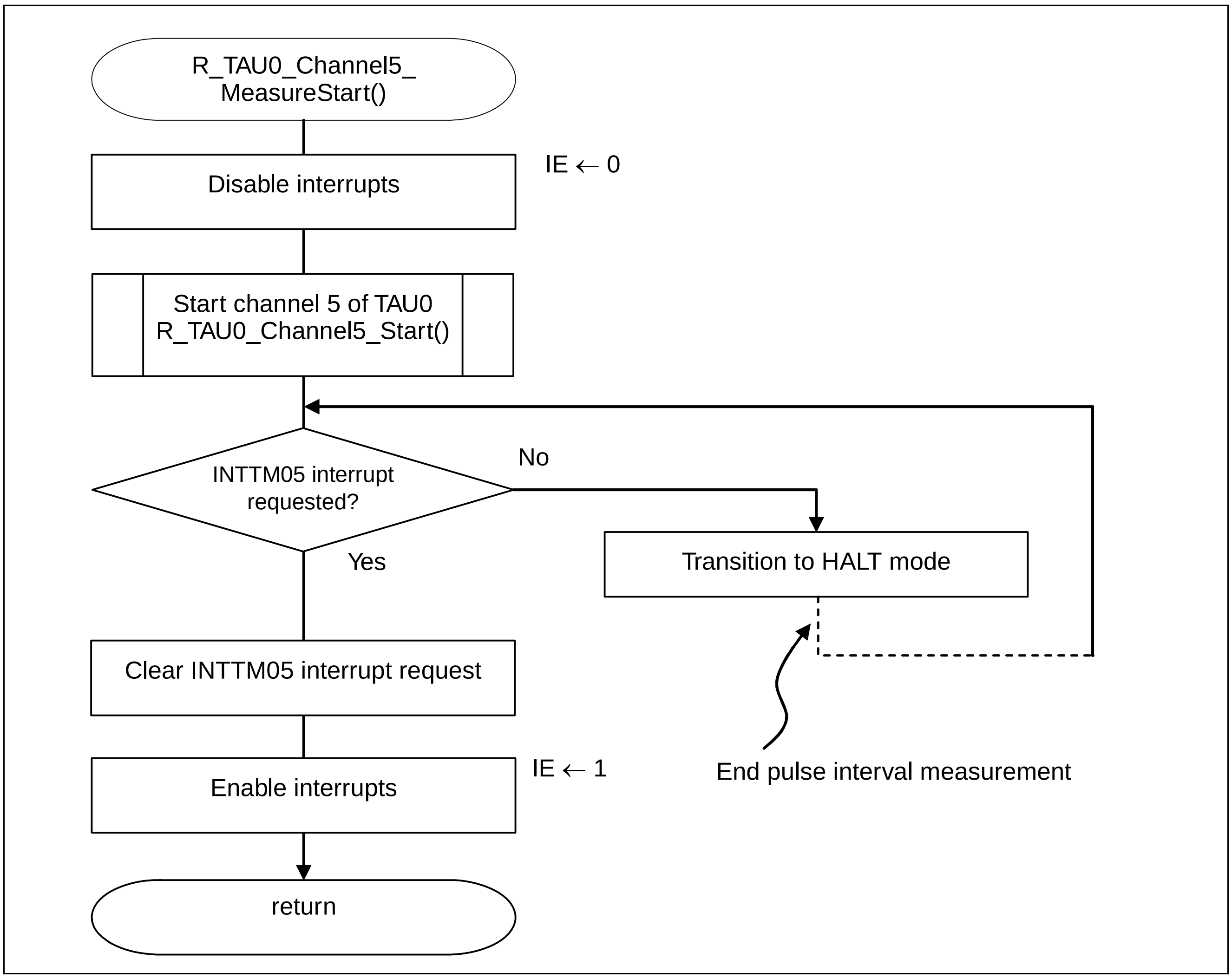


Figure 5.14 Starting Pulse Interval Measurement

5.8.12 Starting Channel 5 of TAU0

Figure 5.15 shows the flowchart for starting channel 5 of TAU0.

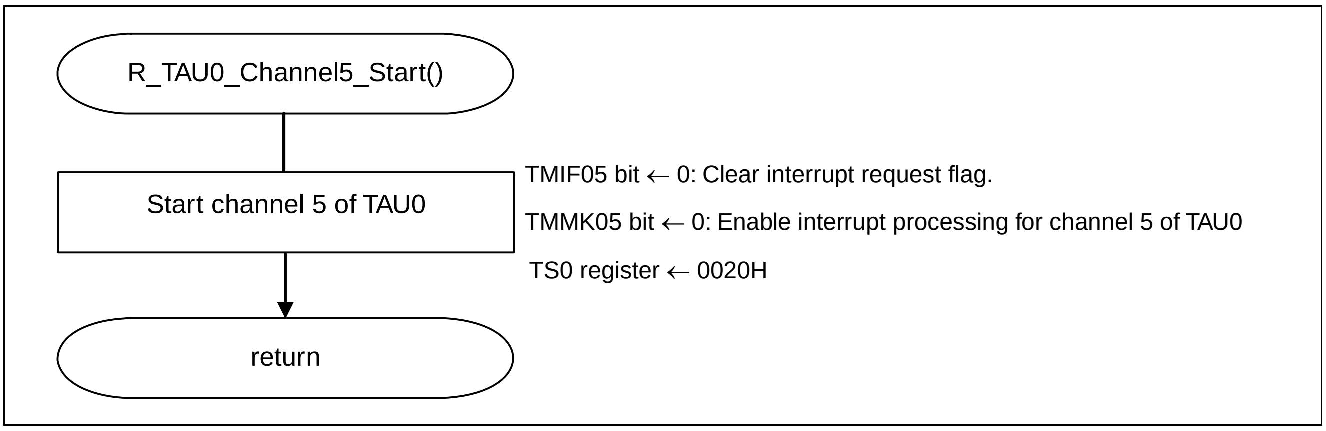

R_TAU0_Channel5_Start()
Start channel 5 of TAU0
TMIF05 bit ← 0: Clear interrupt request flag.
TMMK05 bit ← 0: Enable interrupt processing for channel 5 of TAU0
TS0 register ← 0020H
return

Figure 5.15 Starting Channel 5 of TAU0

5.8.13 Stopping Channel 5 of TAU0

Figure 5.16 shows the flowchart for stopping channel 5 of TAU0.

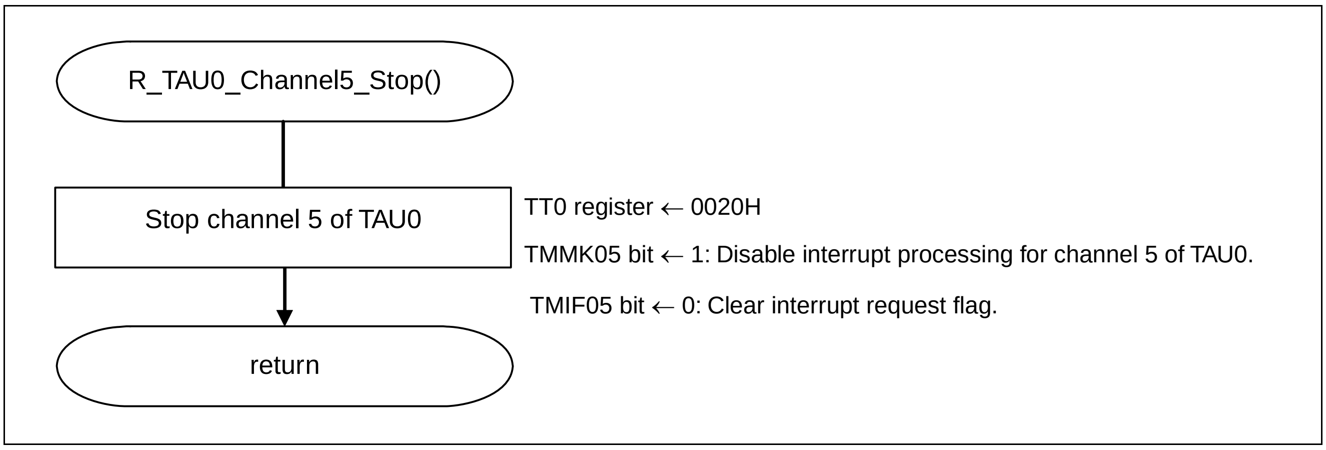

R_TAU0_Channel5_Stop()
Stop channel 5 of TAU0
TT0 register ← 0020H
TMMK05 bit ← 1: Disable interrupt processing for channel 5 of TAU0.
TMIF05 bit ← 0: Clear interrupt request flag.
return

Figure 5.16 Stopping Channel 5 of TAU0

5.8.14 Getting Count Clock Change Request Flag

Figure 5.17 shows the flowchart for getting the count clock change request flag. This function does nothing but returns global variable g_fCLKChangeFlag as a return value.

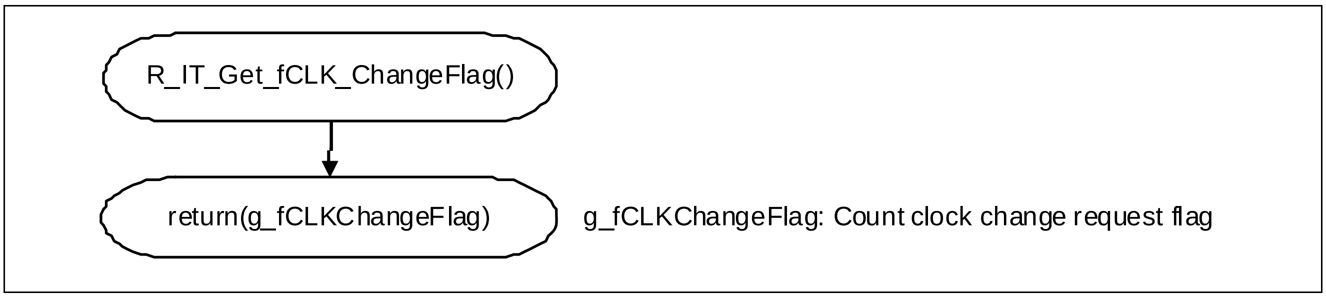

R_IT_Get_fCLK_ChangeFlag()
return(g_fCLKChangeFlag)
g_fCLKChangeFlag: Count clock change request flag

Figure 5.17 Getting Count Clock Change Request Flag

5.8.15 Getting Pulse Interval Measurement End Flag

Figure 5.18 shows the flowchart for getting the pulse interval measurement end flag. This function does nothing but returns global variable g_fCLKChangeFlag as a return value.

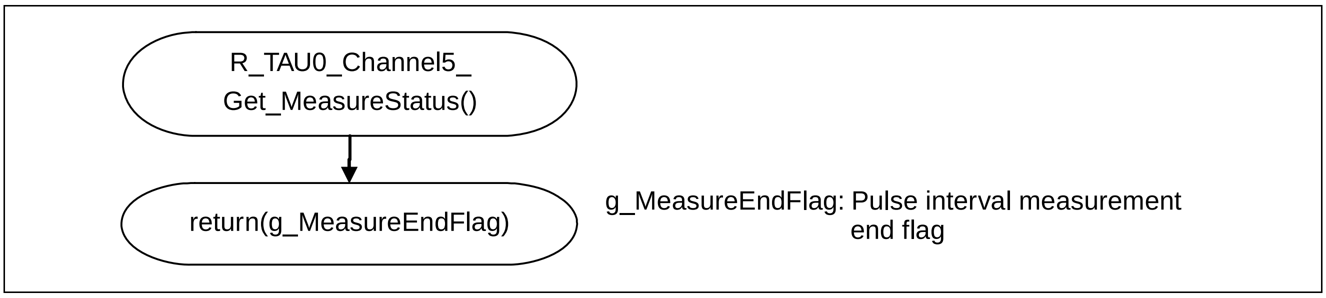


Figure 5.18 Getting Pulse Interval Measurement End Flag

5.8.16 Judging Pulse Interval Measurement Value

Figure 5.19 shows the flowchart for judging the pulse interval measurement value.

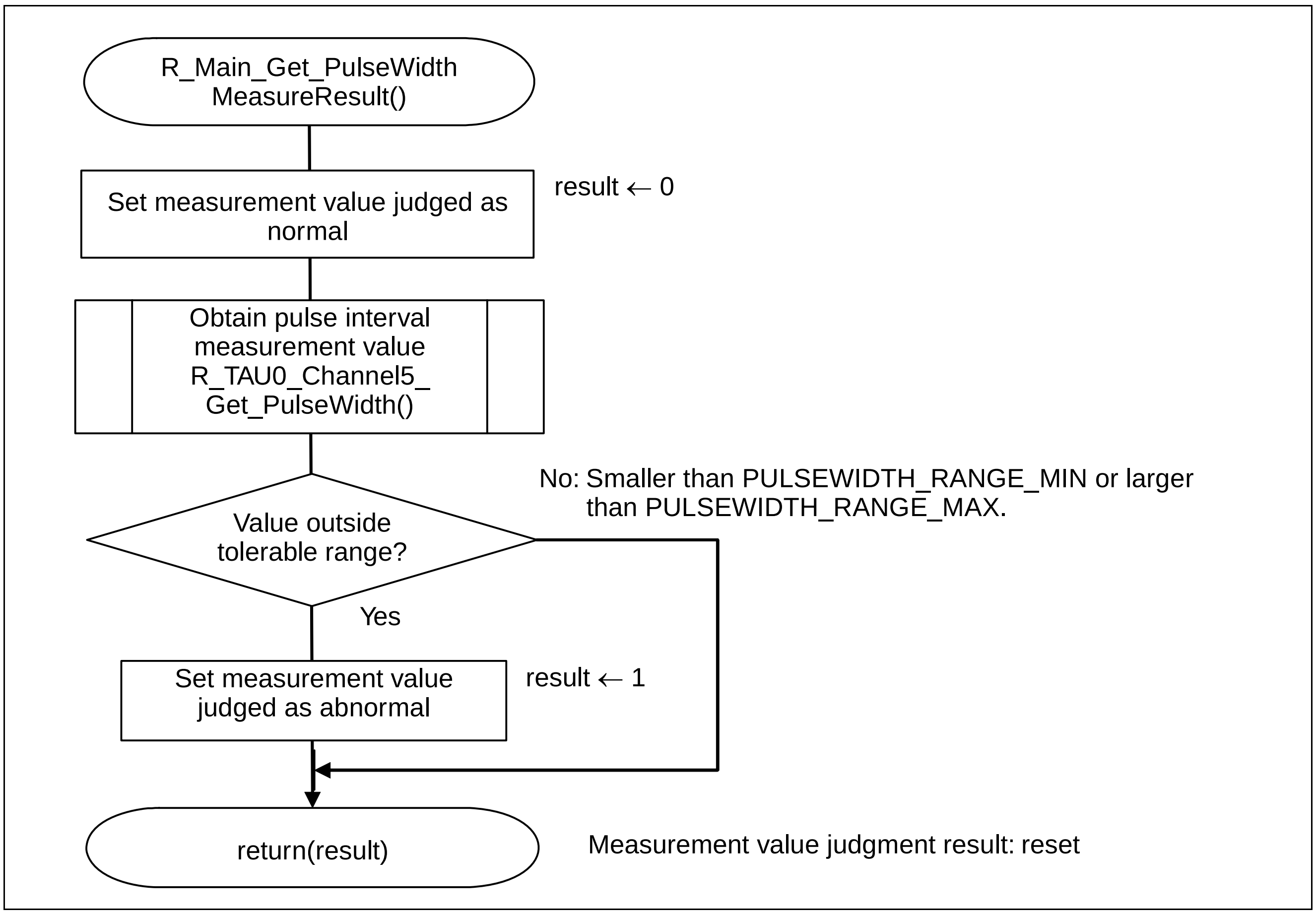

R_Main_Get_PulseWidth MeasureResult()
Set measurement value judged as normal
result ← 0
Obtain pulse interval measurement value R_TAU0_Channel5_ Get_PulseWidth()
Value outside tolerable range?
No: Smaller than PULSEWIDTH_RANGE_MIN or larger than PULSEWIDTH_RANGE_MAX.
Yes
Set measurement value judged as abnormal
result ← 1
return(result)
Measurement value judgment result: reset

Figure 5.19 Judging Pulse Interval Measurement Value

5.8.17 Getting Pulse Interval Measurement Value

Figure 5.20 shows the flowchart for getting the pulse interval measurement value.

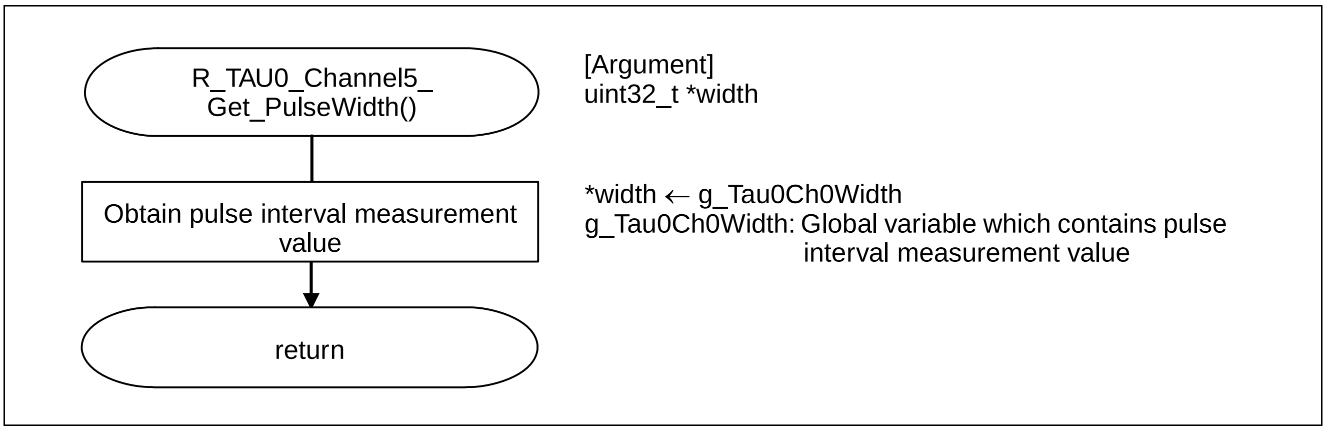

R_TAU0_Channel5_
Get_PulseWidth()
Obtain pulse interval measurement value
return
[Argument]
uint32_t *width
*width ← g_Tau0Ch0Width
g_Tau0Ch0Width: Global variable which contains pulse interval measurement value

Figure 5.20 Getting Pulse Interval Measurement Value

5.8.18 Clearing Pulse Interval Measurement End Flag

Figure 5.21 shows the flowchart for clearing the pulse interval measurement end flag.

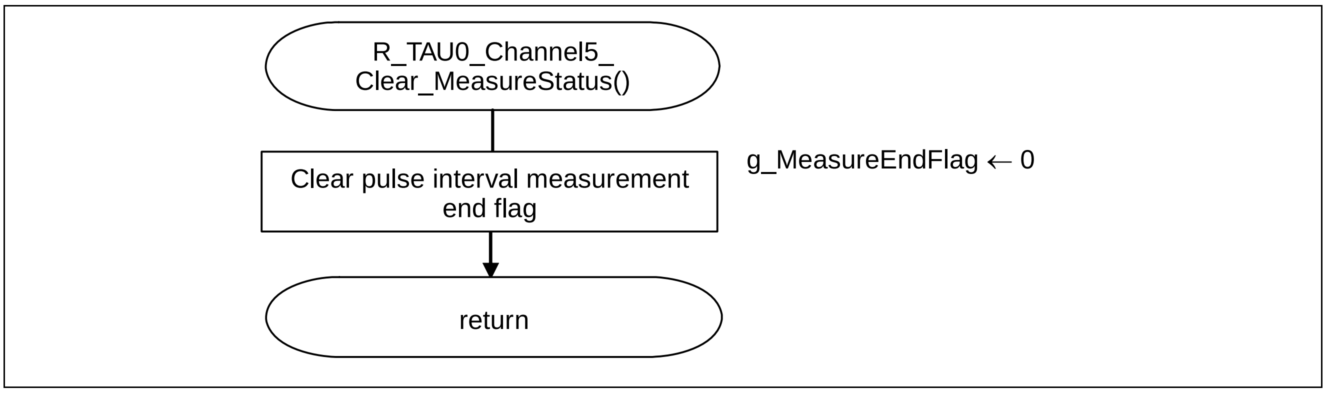


Figure 5.21 Clearing Pulse Interval Measurement End Flag

5.8.19 Changing Count Clock

Figure 5.22 shows the flowchart for changing the count clock.

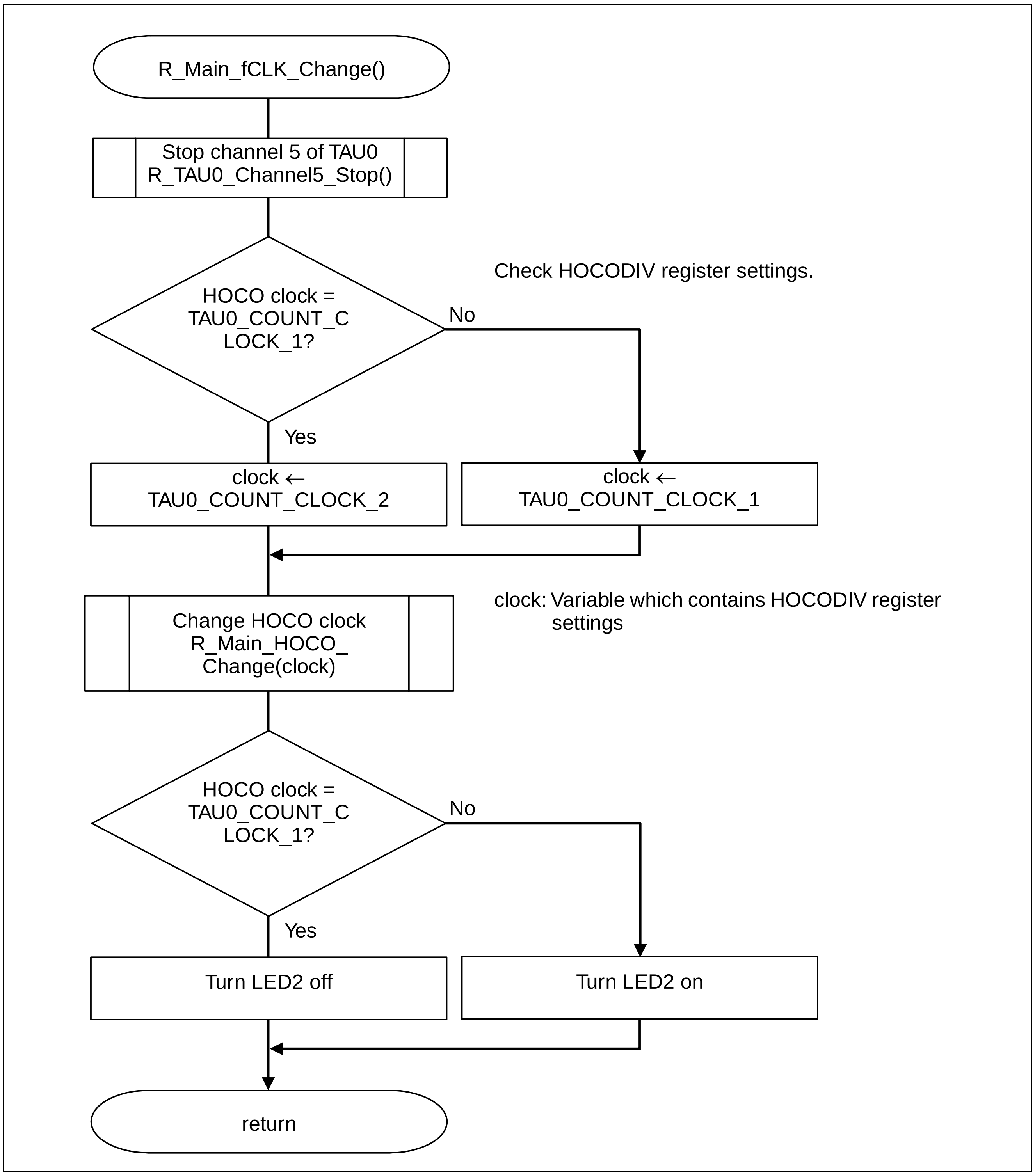

R_Main_fCLK_Change()
Stop channel 5 of TAU0
R_TAU0_Channel5_Stop()
HOCO clock = TAU0_COUNT_CLOCK_1?
Check HOCODIV register settings.
No
Yes
clock ← TAU0_COUNT_CLOCK_2
clock ← TAU0_COUNT_CLOCK_1
Change HOCO clock
R_Main_HOCO_Change(clock)
clock: Variable which contains HOCODIV register settings
HOCO clock = TAU0_COUNT_CLOCK_1?
No
Yes
Turn LED2 off
Turn LED2 on
return

Figure 5.22 Changing Count Clock

5.8.20 Changing HOCO Clock

Figure 5.23 shows the flowchart for changing the HOCO clock.

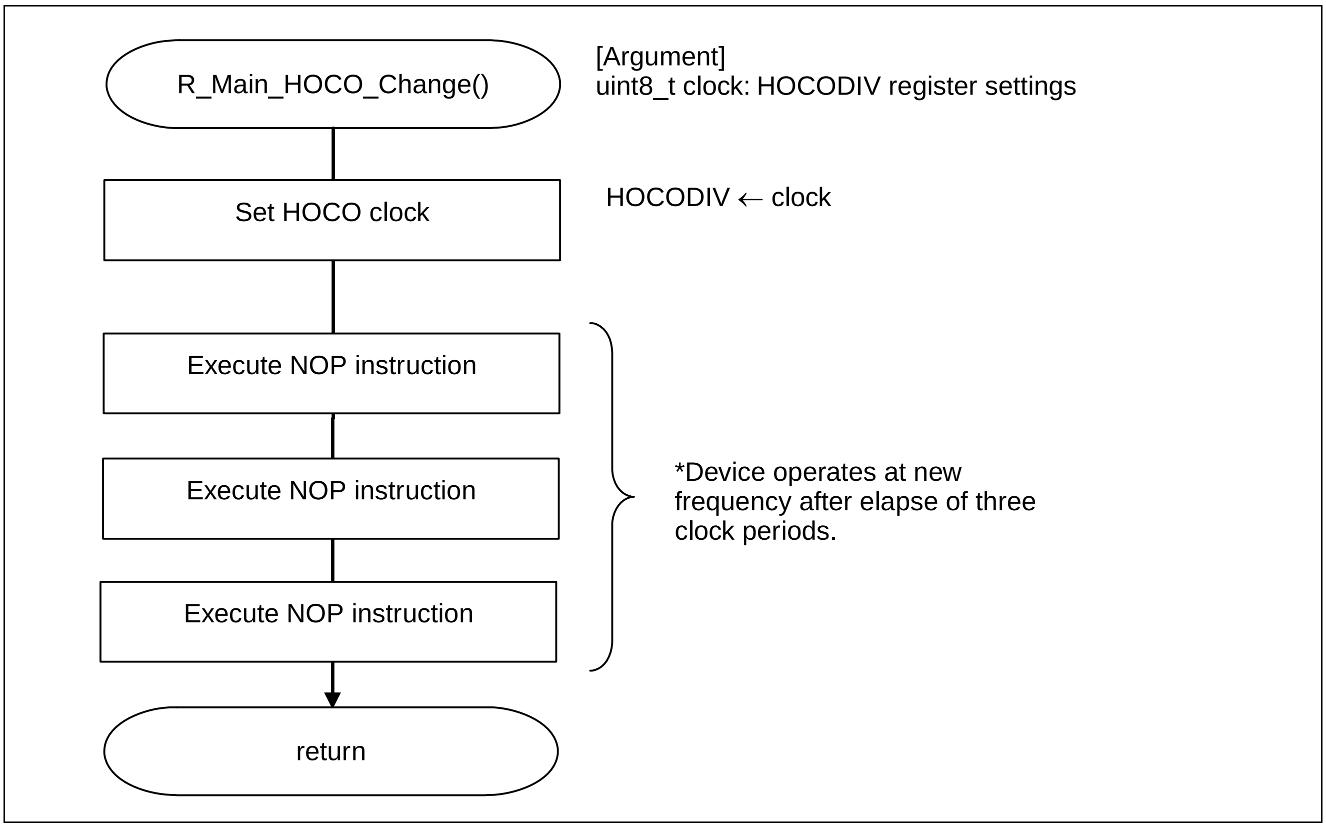


Figure 5.23 Changing HOCO Clock

5.8.21 Starting LED Blink

Figure 5.24 shows the flowchart for starting the LED blink.

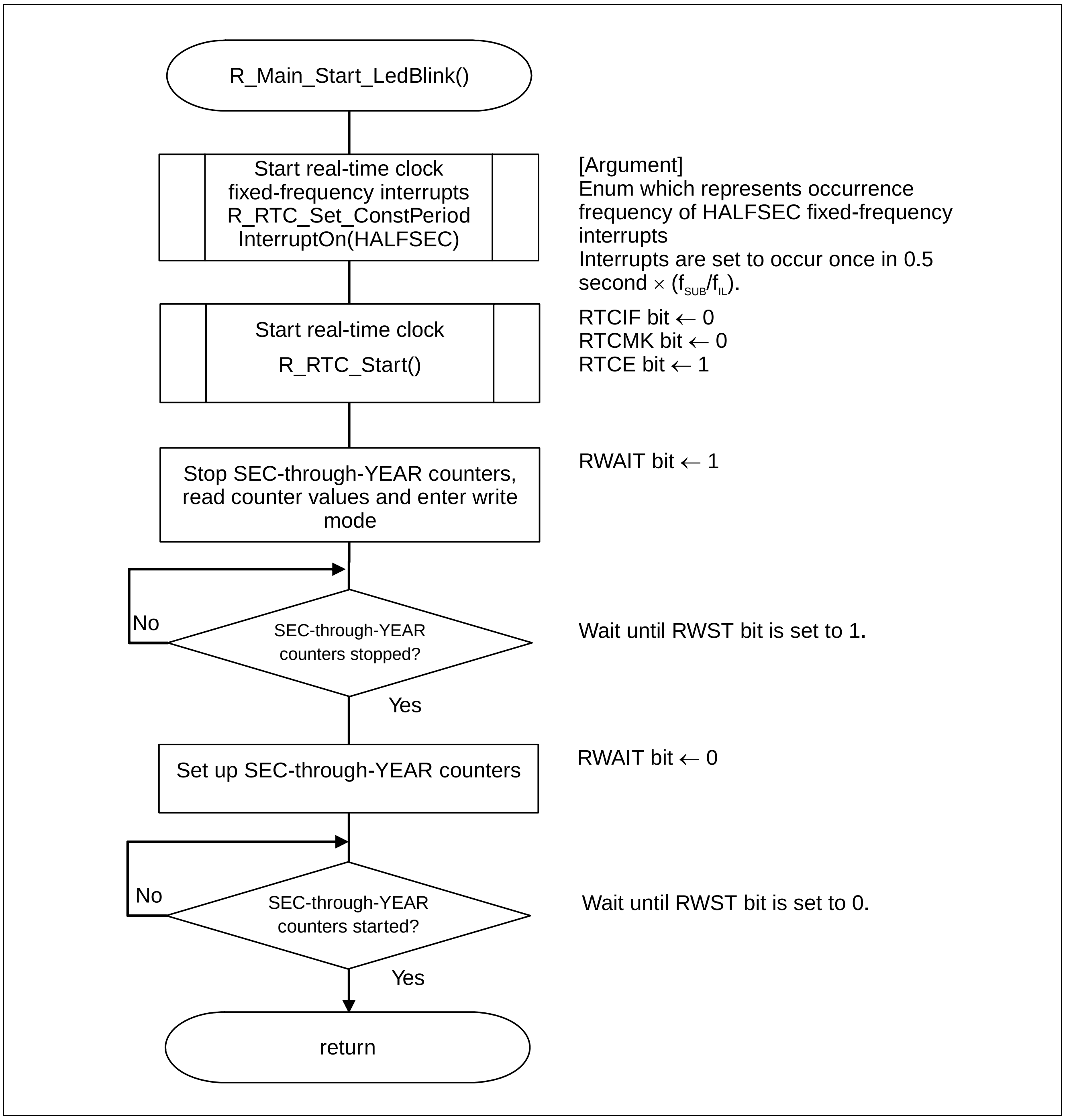

R_Main_Start_LedBlink()
Start real-time clock fixed-frequency interrupts R_RTC_Set_ConstPeriod InterruptOn(HALFSEC)
[Argument]
Enum which represents occurrence frequency of HALFSEC fixed-frequency interrupts
Interrupts are set to occur once in 0.5 second × (f_{SUB}/f_{IL}).
Start real-time clock
R_RTC_Start()
RTCIF bit ← 0
RTCMK bit ← 0
RTCE bit ← 1
Stop SEC-through-YEAR counters, read counter values and enter write mode
RWAIT bit ← 1
No
SEC-through-YEAR counters stopped?
Yes
Wait until RWST bit is set to 1.
Set up SEC-through-YEAR counters
RWAIT bit ← 0
No
SEC-through-YEAR counters started?
Yes
Wait until RWST bit is set to 0.
return

Figure 5.24 Starting LED Blink

5.8.22 Stopping LED Blink

Figure 5.25 shows the flowchart for stopping the LED blink.

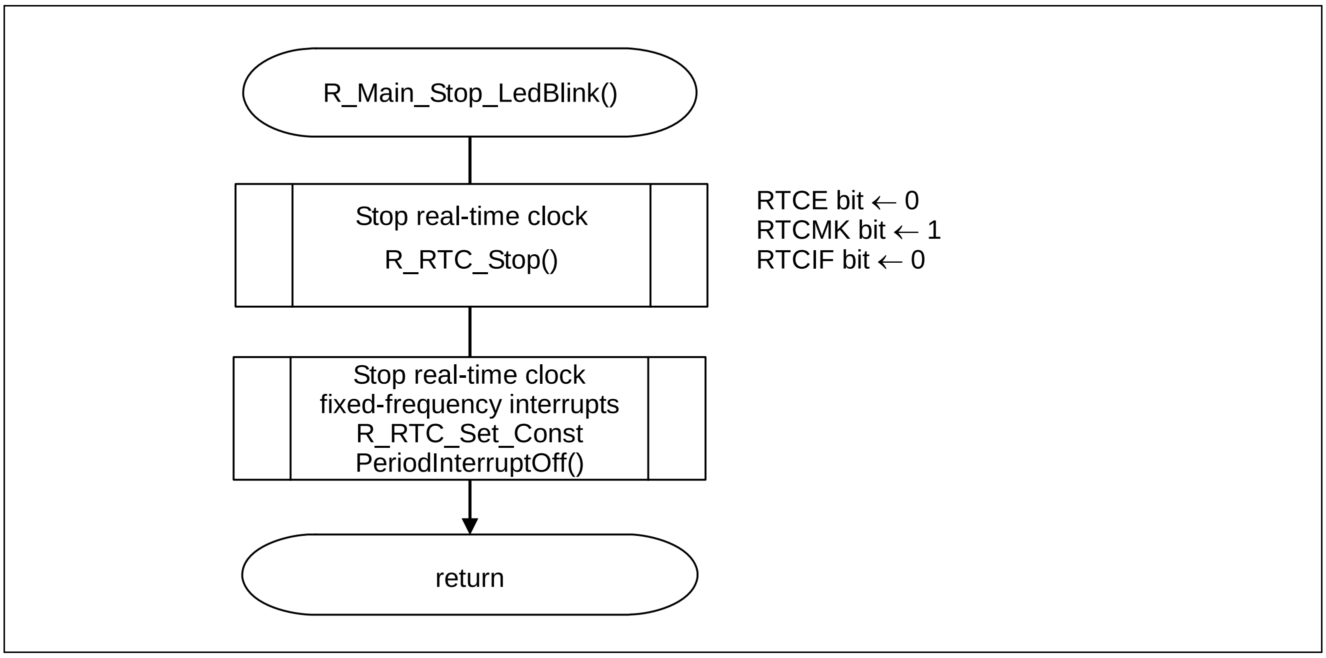


Figure 5.25 Stopping LED Blink

5.8.23 Starting Real-Time Clock

Figure 5.26 shows the flowchart for starting the real-time clock.

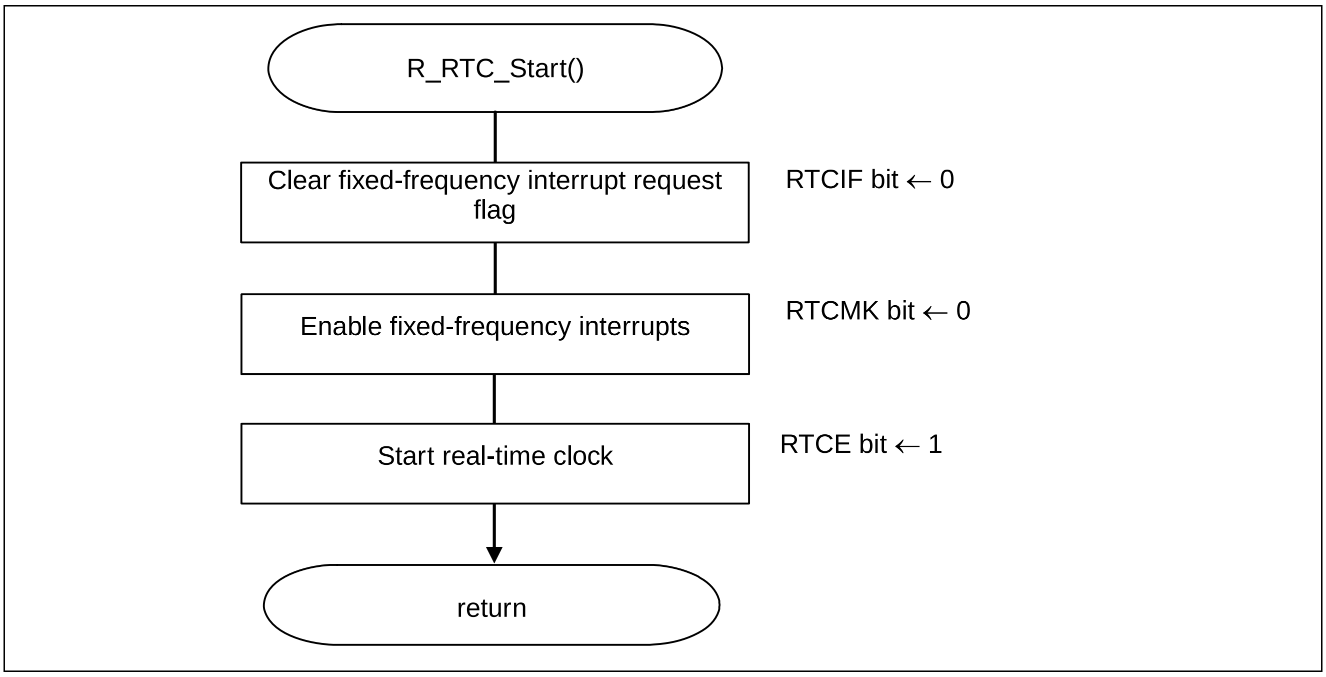


Figure 5.26 Starting Real-Time Clock

5.8.24 Stopping Real-Time Clock

Figure 5.27 shows the flowchart for stopping the real-time clock.

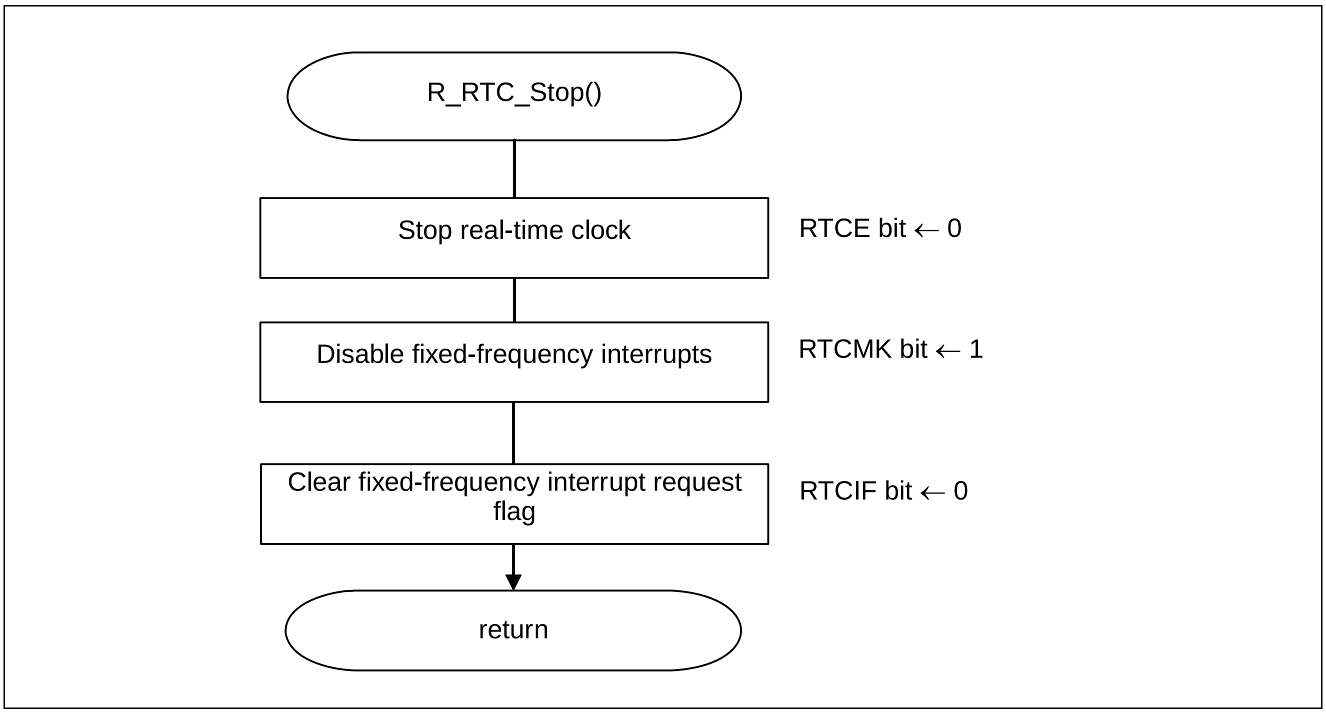

R_RTC_Stop()
Stop real-time clock
RTCE bit ← 0
Disable fixed-frequency interrupts
RTCMK bit ← 1
Clear fixed-frequency interrupt request flag
RTCIF bit ← 0
return

Figure 5.27 Stopping Real-Time Clock

5.8.25 Starting Real-Time Clock Fixed-Frequency Interrupt

Figure 5.28 shows the flowchart for starting real-time clock fixed-frequency interrupts.

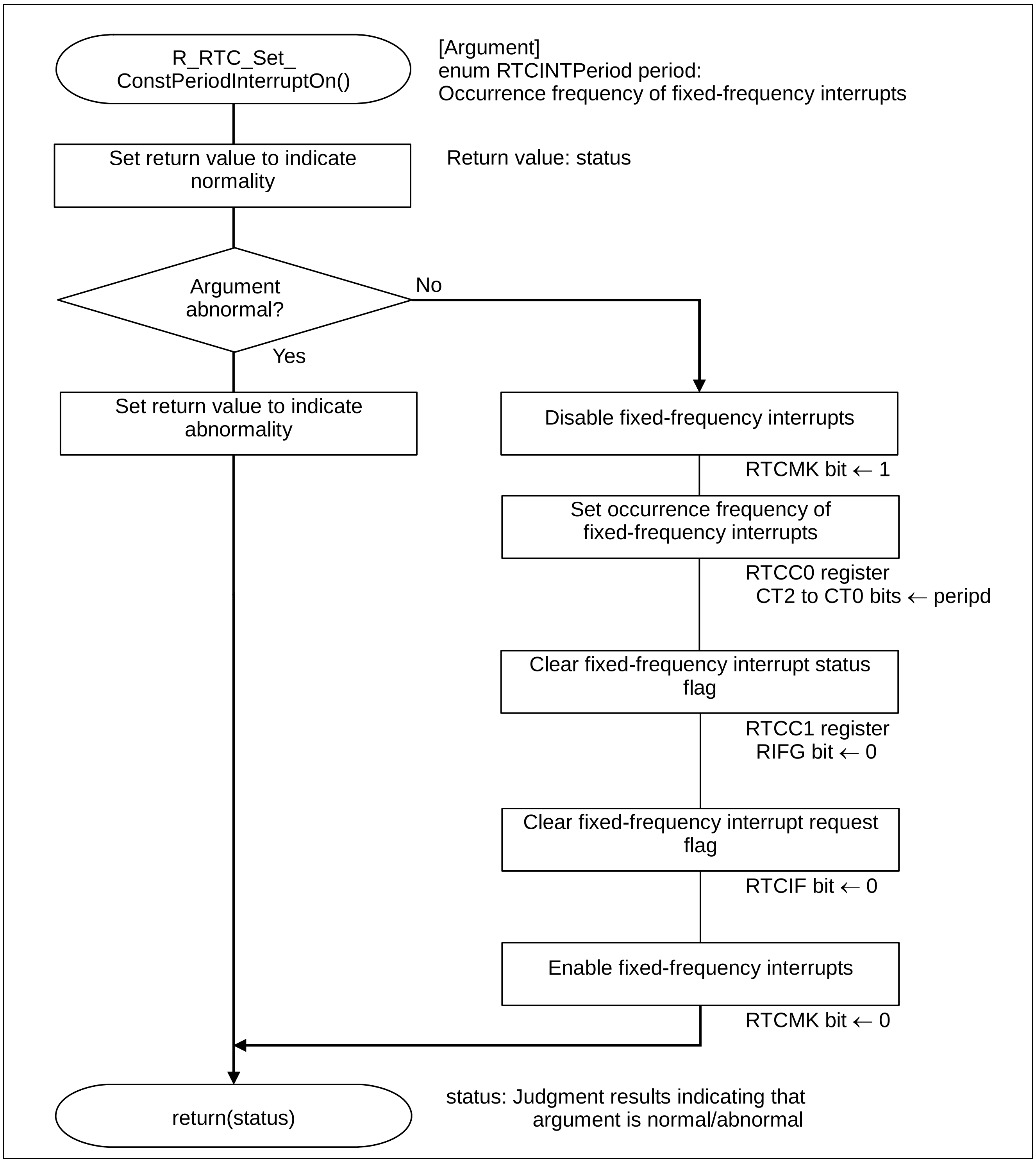

R_RTC_Set_ ConstPeriodInterruptOn()
[Argument]
enum RTCINTPeriod period:
Occurrence frequency of fixed-frequency interrupts
Set return value to indicate normality
Return value: status
Argument abnormal?
No
Yes
Set return value to indicate abnormality
Disable fixed-frequency interrupts
RTCMK bit ← 1
Set occurrence frequency of fixed-frequency interrupts
RTCC0 register
CT2 to CT0 bits ← peripd
Clear fixed-frequency interrupt status flag
RTCC1 register
RIFG bit ← 0
Clear fixed-frequency interrupt request flag
RTCIF bit ← 0
Enable fixed-frequency interrupts
RTCMK bit ← 0
return(status)
status: Judgment results indicating that argument is normal/abnormal

Figure 5.28 Starting Real-Time Clock Fixed-Frequency Interrupt

5.8.26 Stopping Real-Time Clock Fixed-Frequency Interrupt

Figure 5.29 shows the flowchart for stopping real-time clock fixed-frequency interrupts.

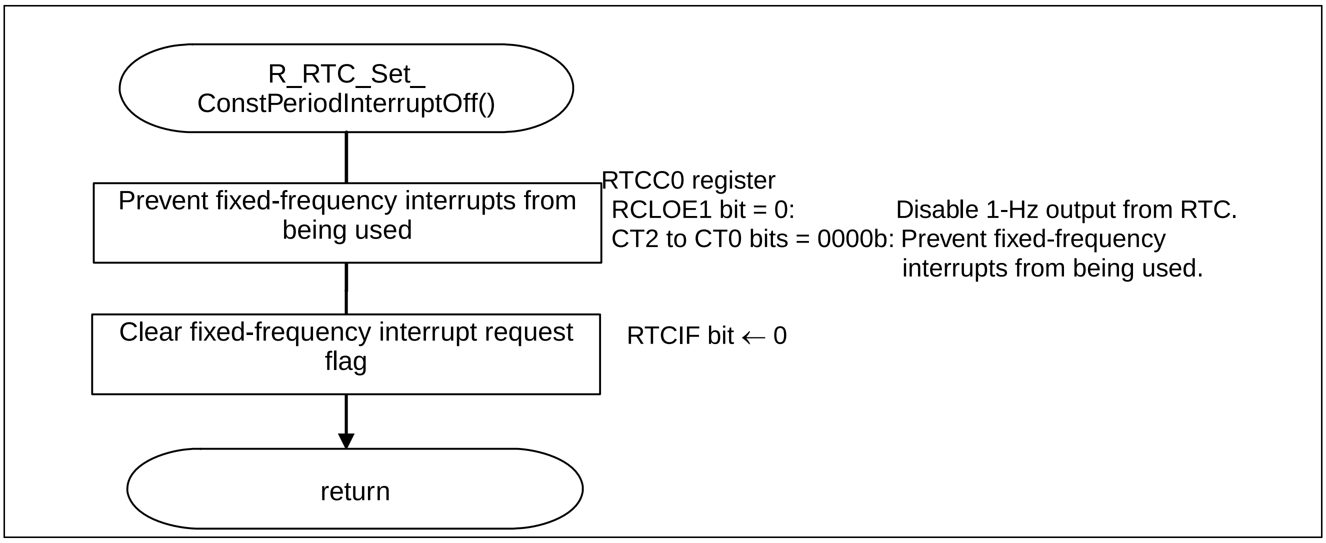

R_RTC_Set_ ConstPeriodInterruptOff()
Prevent fixed-frequency interrupts from being used
RTCC0 register
RCLOE1 bit = 0: Disable 1-Hz output from RTC.
CT2 to CT0 bits = 0000b: Prevent fixed-frequency interrupts from being used.
Clear fixed-frequency interrupt request flag
RTCIF bit ← 0
return

Figure 5.29 Stopping Real-Time Clock Fixed-Frequency Interrupt

5.8.27 Clearing Count Clock Change Request Flag

Figure 5.30 shows the flowchart for clearing the count clock change request flag.

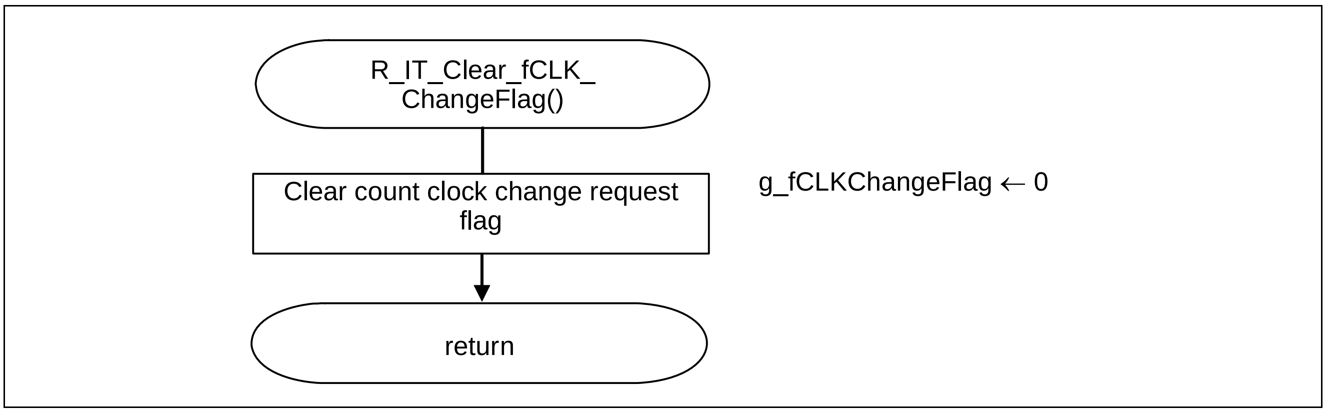

R_IT_Clear_fCLK_ ChangeFlag()
Clear count clock change request flag
g_fCLKChangeFlag ← 0
return

Figure 5.30 Clearing Count Clock Change Request Flag

5.8.28 Interval Timer Interrupt

Figure 5.31 shows the flowchart for an interval timer interrupt.

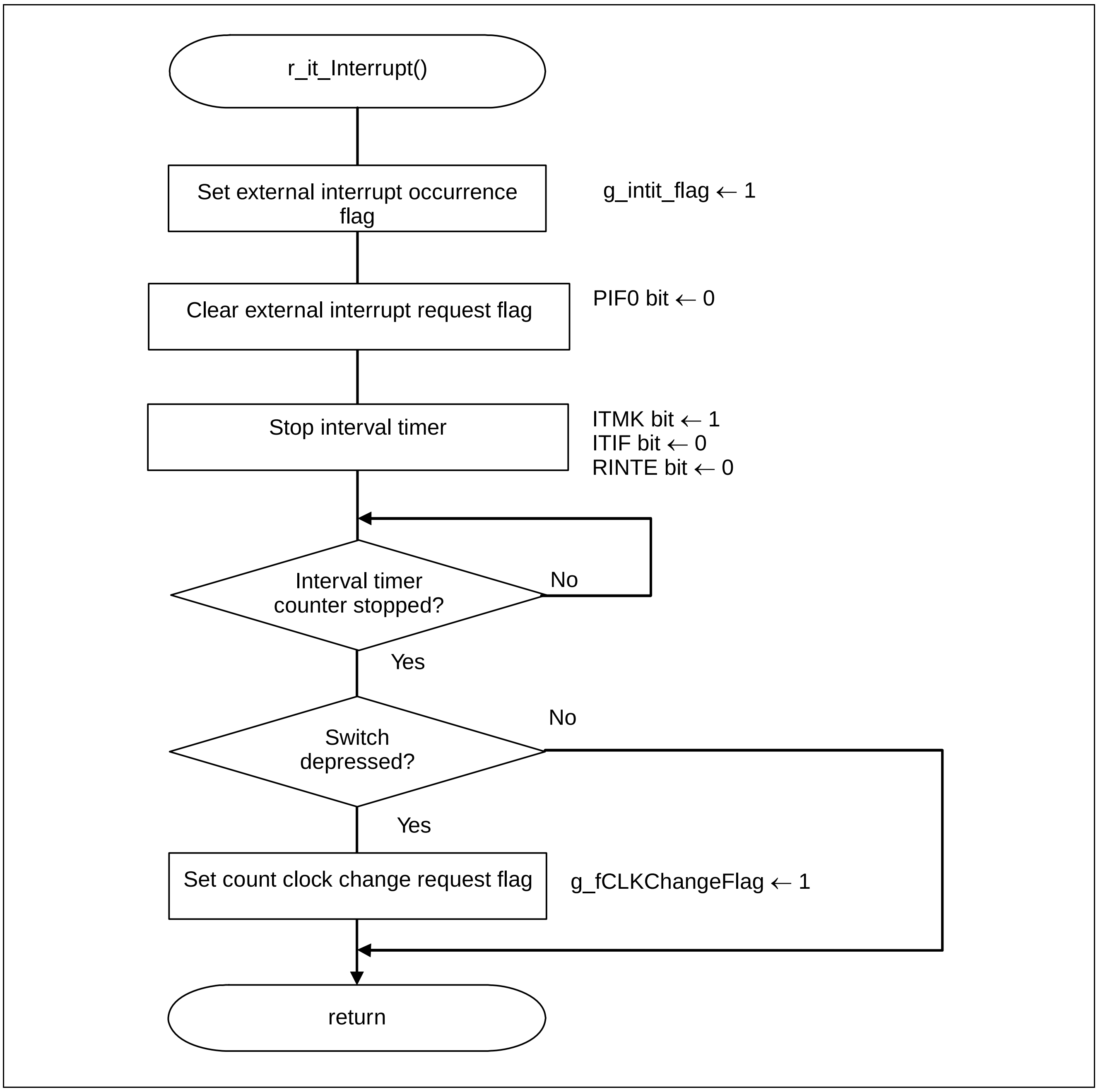


Figure 5.31 Interval Timer Interrupt

5.8.29 Capture End Interrupt for Channel 5 of TAU0

Figure 5.32 shows the flowchart for a capture end interrupt for channel 5 of TAU0.

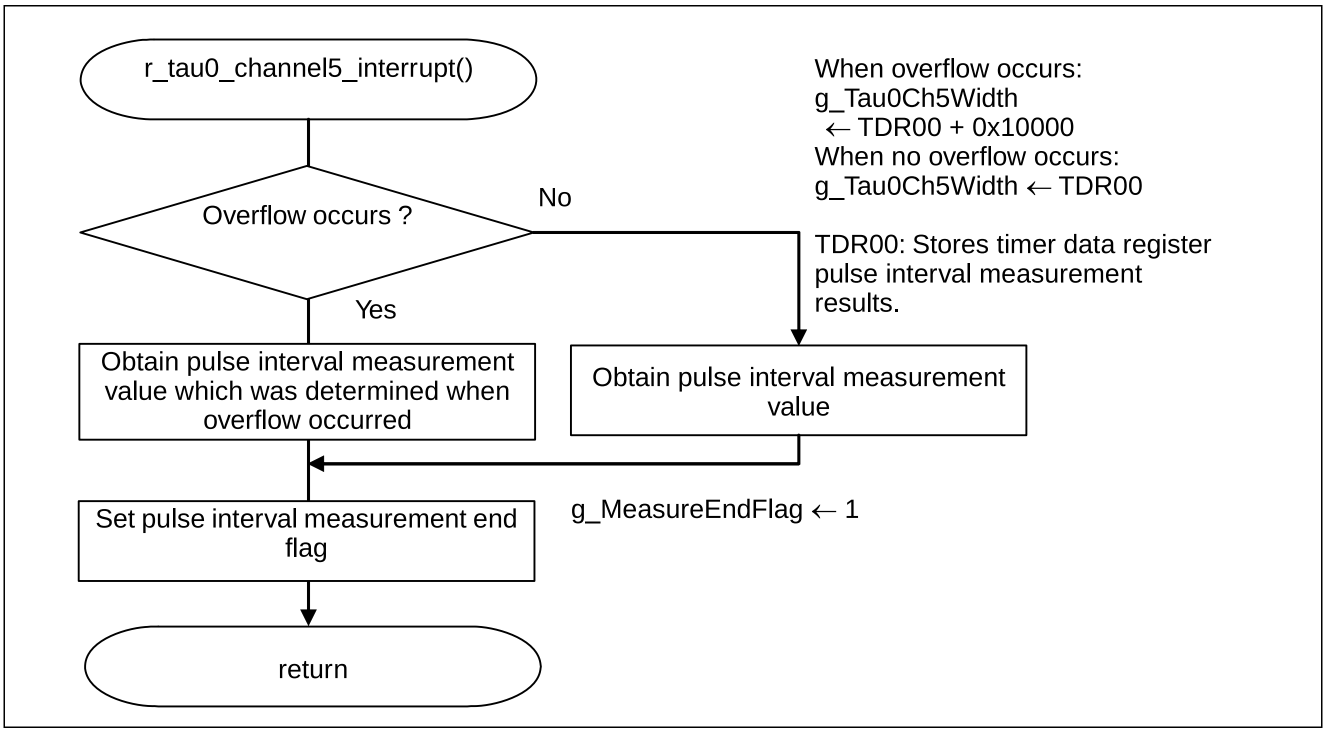

r_tau0_channel5_interrupt()
When overflow occurs:
g_Tau0Ch5Width
← TDR00 + 0x10000
When no overflow occurs:
g_Tau0Ch5Width ← TDR00
Overflow occurs ?
No
TDR00: Stores timer data register pulse interval measurement results.
Yes
Obtain pulse interval measurement value which was determined when overflow occurred
Obtain pulse interval measurement value
Set pulse interval measurement end flag
g_MeasureEndFlag ← 1
return

Figure 5.32 Capture End Interrupt for Channel 5 of TAU0

5.8.30 INTP0 External Interrupt

Figure 5.33 shows the flowchart for an INTP0 external interrupt.

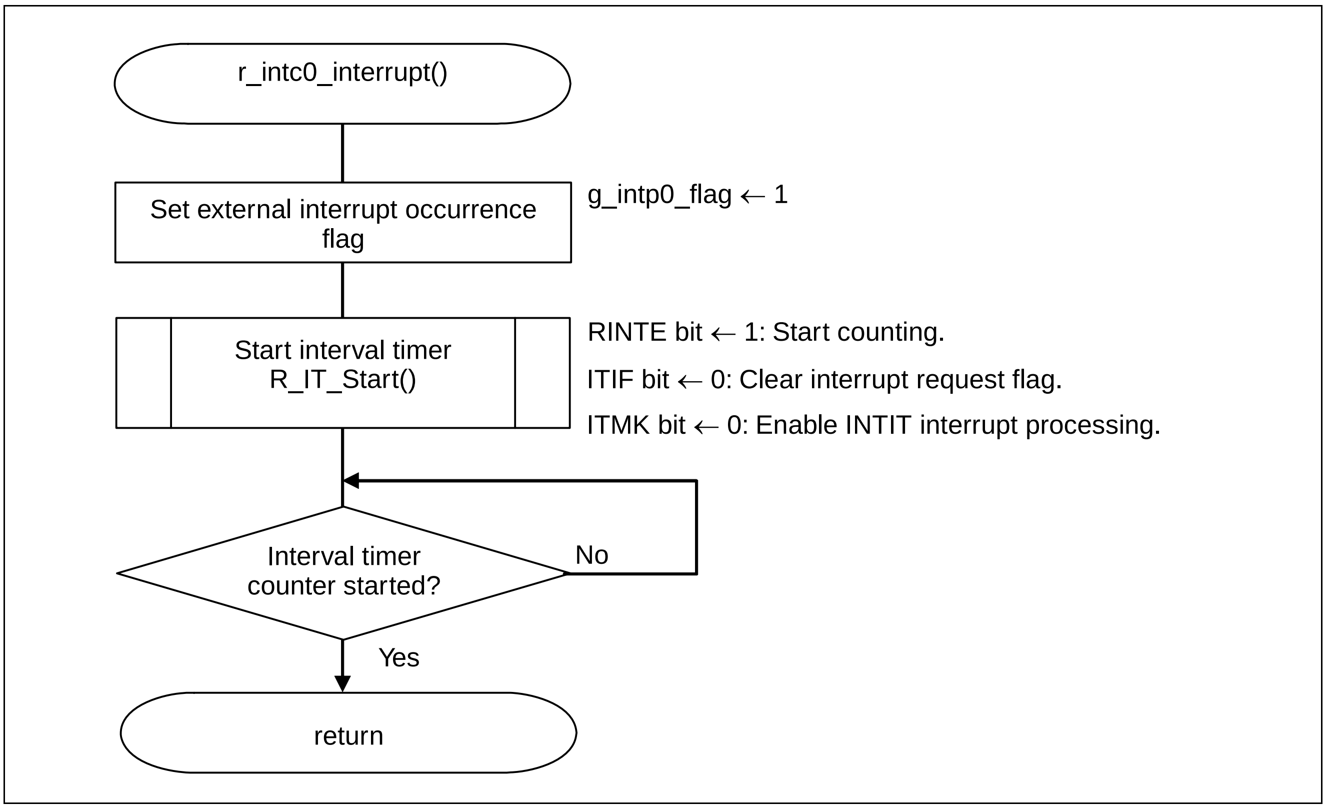

r_intc0_interrupt()
Set external interrupt occurrence flag
g_intp0_flag ← 1
Start interval timer R_IT_Start()
RINTE bit ← 1: Start counting.
ITIF bit ← 0: Clear interrupt request flag.
ITMK bit ← 0: Enable INTIT interrupt processing.
Interval timer counter started?
No
Yes
return

Figure 5.33 INTP0 External Interrupt

5.8.31 RTC Fixed-Frequency Interrupt

Figure 5.34 shows the flowchart for an RTC fixed-frequency interrupt.

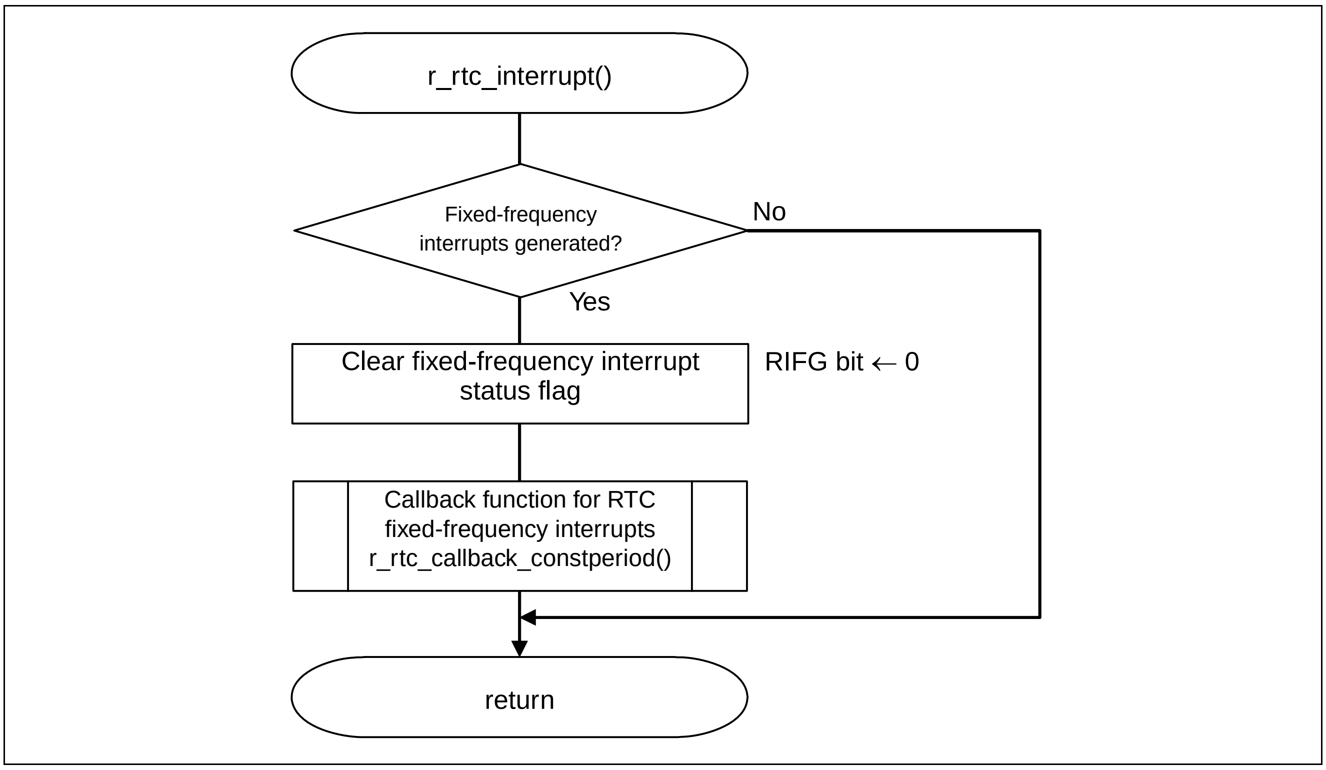

r_rtc_interrupt()
Fixed-frequency interrupts generated?
No
Yes
Clear fixed-frequency interrupt status flag
RIFG bit ← 0
Callback function for RTC fixed-frequency interrupts r_rtc_callback_constperiod()
return

Figure 5.34 RTC Fixed-Frequency Interrupt

5.8.32 Callback Function for RTC Fixed-frequency Interrupts

Figure 5.35 shows the flowchart for the callback function for RTC fixed-frequency interrupts.

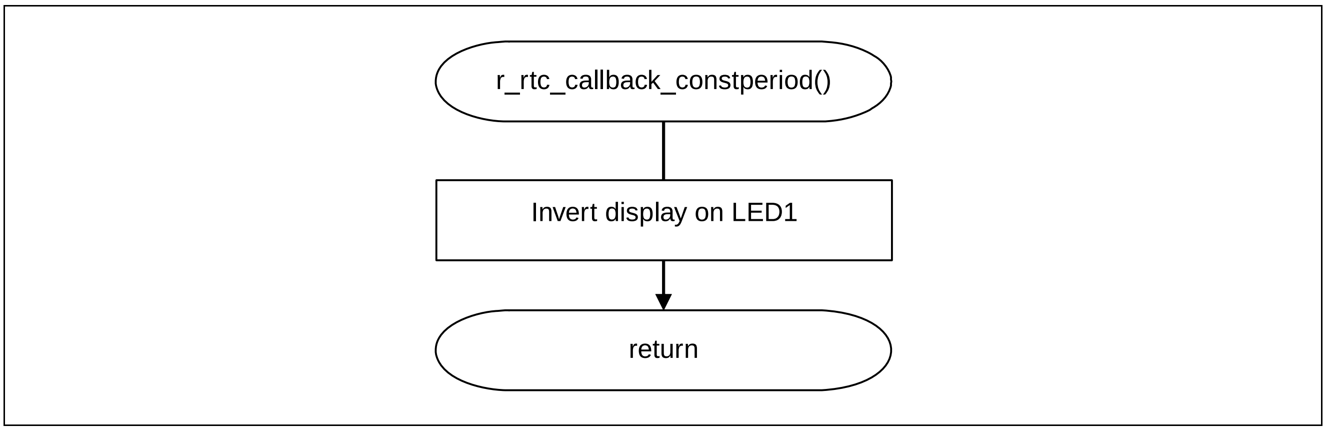


Figure 5.35 Callback Function for RTC Fixed-Frequency Interrupts

5.8.33 Interval Timer Start

Figure 5.36 shows the flowchart for staring the interval timer.

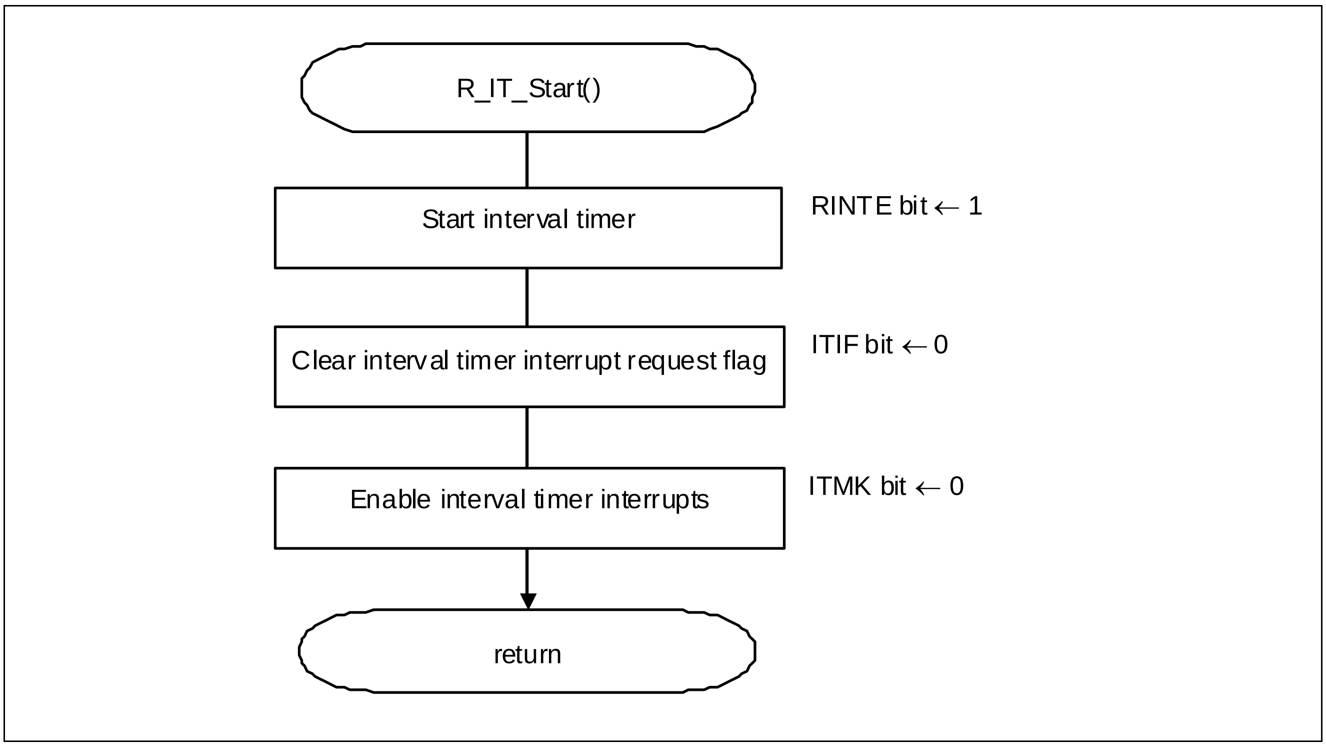

R_IT_Start()
Start interval timer
RINTE bit ← 1
Clear interval timer interrupt request flag
ITIF bit ← 0
Enable interval timer interrupts
ITMK bit ← 0
return

Figure 5.36 Interval Timer Start

5.8.34 Stopping Interval Timer

Figure 5.37 shows the flowchart for stopping the interval timer.

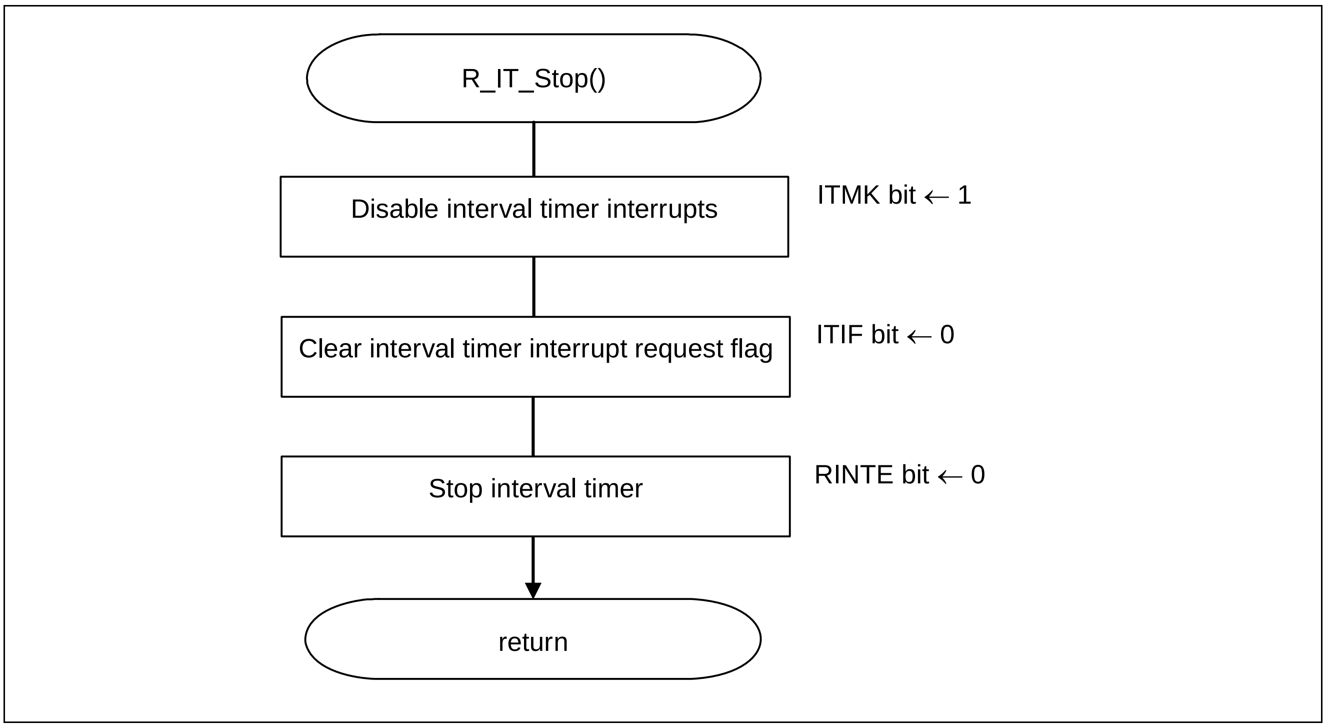

R_IT_Stop()
Disable interval timer interrupts
ITMK bit ← 1
Clear interval timer interrupt request flag
ITIF bit ← 0
Stop interval timer
RINTE bit ← 0
return

Figure 5.37 Stopping Interval Timer

5.8.35 Getting INTP0 External Interrupt Occurrence Flag

Figure 5.38 shows the flowchart for getting the INTP0 external interrupt occurrence flag. This function does nothing but returns global variable g_intop0_flag as a return value.

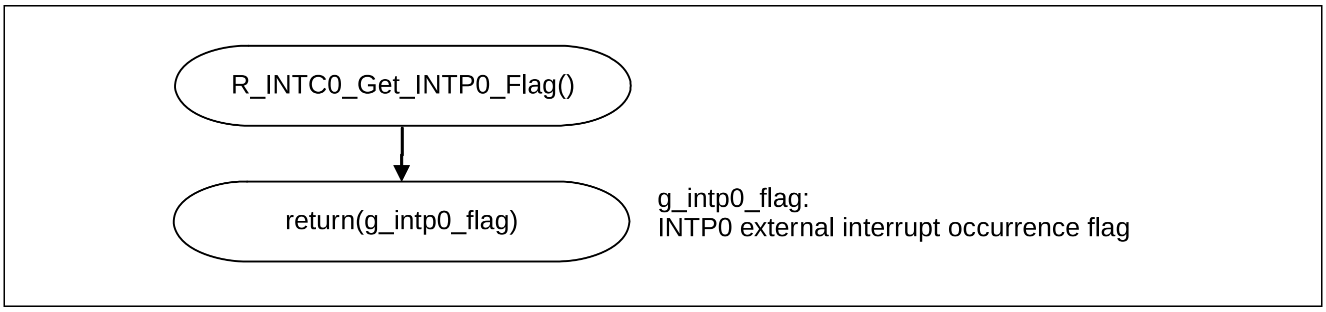


Figure 5.38 Getting INTP0 External Interrupt Occurrence Flag

5.8.36 Clearing INTP0 External Interrupt Occurrence Flag

Figure 5.39 shows the flowchart for clearing the INTP0 external interrupt occurrence flag.

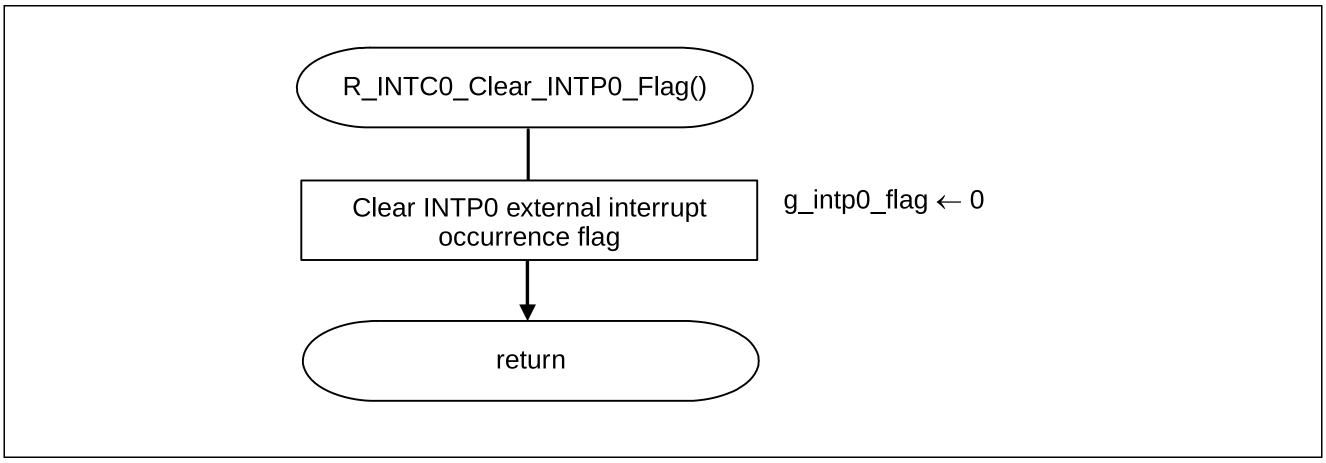

R_INTC0_Clear_INTP0_Flag()
Clear INTP0 external interrupt occurrence flag
g_intp0_flag ← 0
return

Figure 5.39 Clearing INTP0 External Interrupt Occurrence Flag

5.8.37 Getting Interval Timer Interrupt Occurrence Flag

Figure 5.40 shows the flowchart for getting the interval timer interrupt occurrence flag. This function does nothing but returns global variable g_intit_flag as a return value.

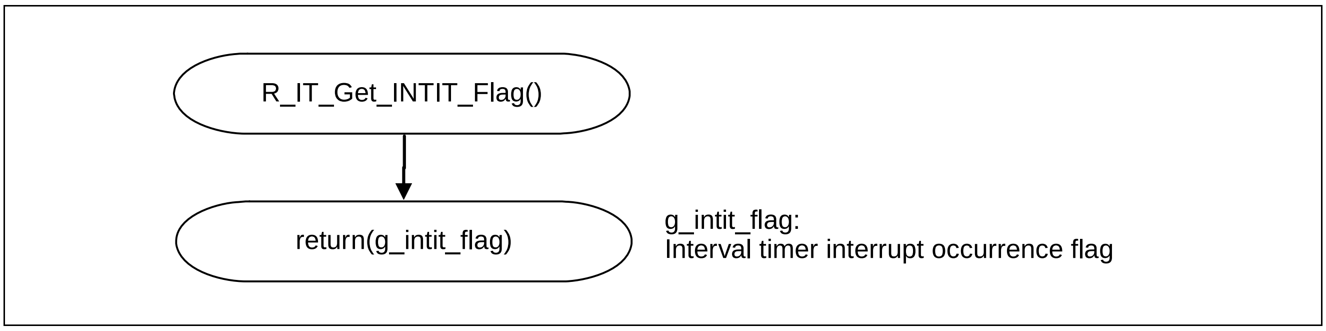


Figure 5.40 Getting Interval Timer Interrupt Occurrence Flag

5.8.38 Clearing Interval Timer Interrupt Occurrence Flag

Figure 5.41 shows the flowchart for clearing the interval timer interrupt occurrence flag.

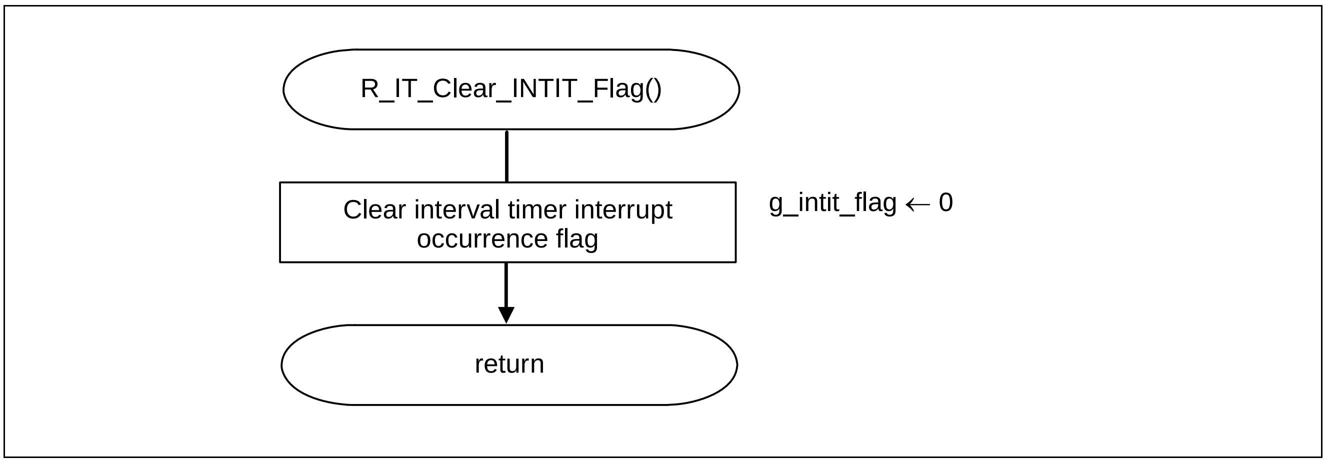


Figure 5.41 Clearing Interval Timer Interrupt Occurrence Flag

6. Sample Code

The sample code is available on the Renesas Electronics Website.

7. Documents for Reference

RL78/G13 User's Manual: Hardware (R01UH0146E)

RL78 Family User's Manual: Software (R01US0015E)

(The latest versions of the documents are available on the Renesas Electronics Website.)

Technical Updates/Technical Brochures

(The latest versions of the documents are available on the Renesas Electronics Website.)

Website and Support

Renesas Electronics Website

- http://www.renesas.com/index.jsp

Inquiries

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

Revision Record	RL78/G13 Safety Function (Frequency Detection)

Rev.	Date	Description	
		Page	Summary
1.00	May 28, 2015	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

RENESAS

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

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

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

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

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

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

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

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

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

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

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

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, 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 India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 5.0