

## RL78/G13

R01AN0956EJ0100

Rev. 1.00

## Safety Function (Frequency Detection)

Feb. 27, 2012

### 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 be selected as a timer input for channel 5 of TAU0.

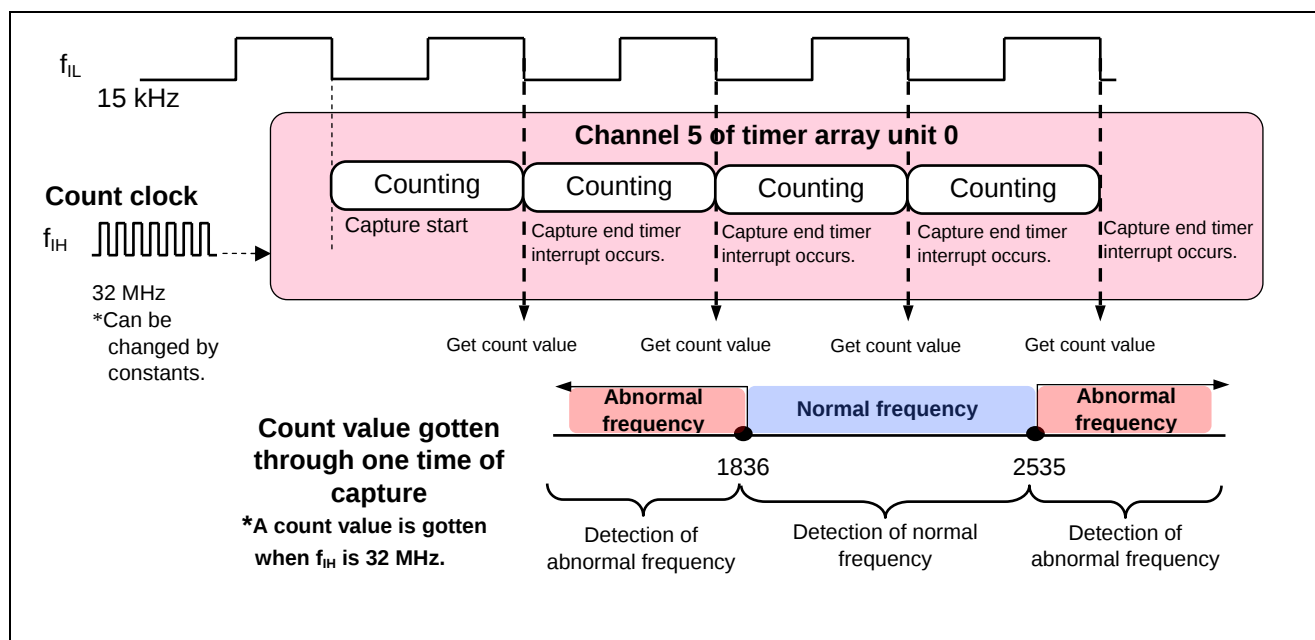
To judge whether the frequency is normal or abnormal, the pulse interval is checked to see if it is within the tolerable range specified by constants. If the frequency is normal, the LED turns off. If not, the LED blinks.

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<br>Changes the count clock (HOCO clock) frequency for TAU0. |
| Channel 5 of timer array unit 0  | Low-speed on-chip oscillator clock<br>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.                             |



**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                | <ul style="list-style-type: none"><li>CPU/peripheral hardware clock:<br/>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.</li></ul> |
| Operating voltage                  | 5.0 V (Operation is possible at 2.9 V to 5.5 V.)<br>LVD operation (VLV): Reset mode 2.81 V (2.76 V to 2.87 V)                                                                                                                       |
| Integrated development environment | CubeSuite+ V1.01.00 from Renesas Electronics Corp.                                                                                                                                                                                  |
| C compiler                         | CA78K0R V1.30 from Renesas Electronics Corp.                                                                                                                                                                                        |
| Board used                         | RL78/G13 target board (QB-R5F100LE-TB)                                                                                                                                                                                              |

## 3. Related Application Note

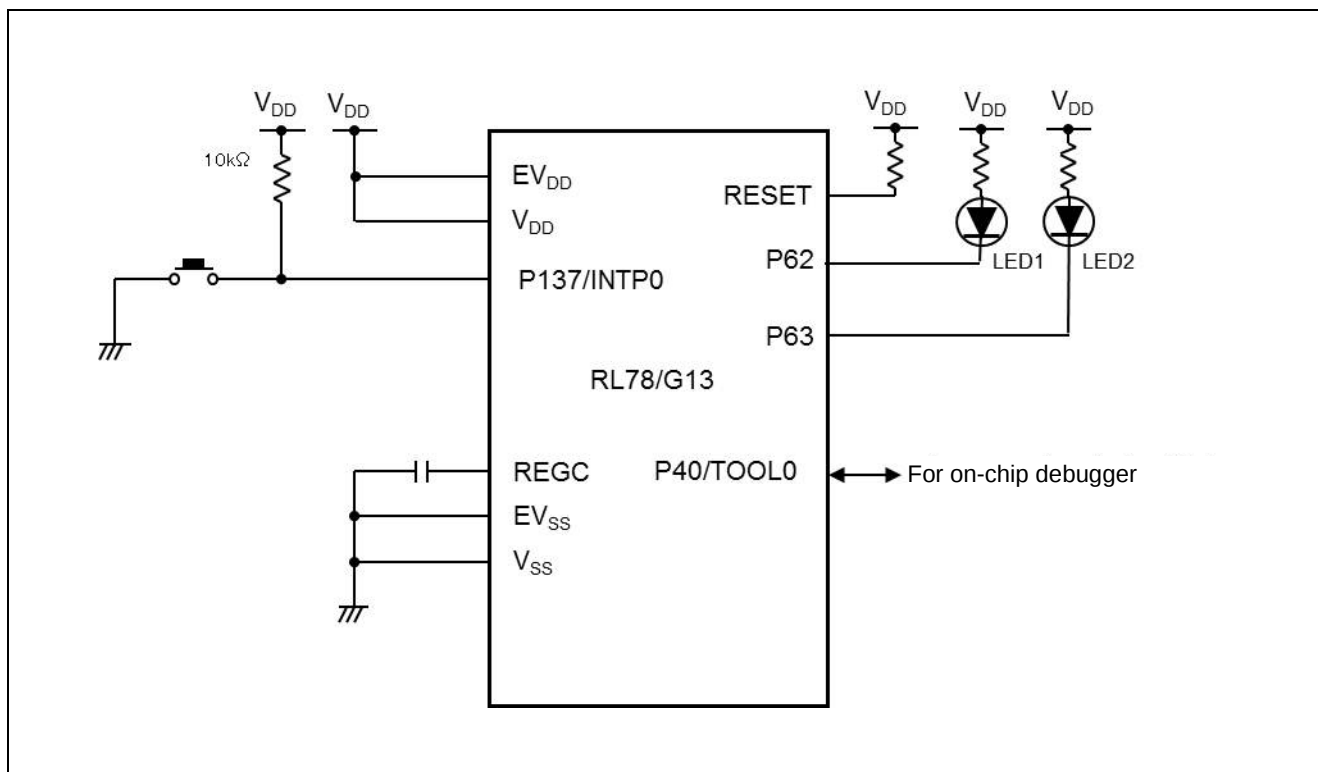
The application note that is related to this application note is listed below for reference.

RL78/G13 Initialization (R01AN0451E) 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 ( $V_{LVI}$ ) 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<br>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:<br>R_Main_Get_PulseWidthMeasureResult,<br>R_Main_fCLK_Change,<br>R_Main_HOCO_Change,<br>R_Main_Start_LedBlink,<br>R_Main_Stop_LedBlink |
| r_cg_intc_user.c  | External interrupt input module<br>INTP0 external interrupt | Additional functions:<br>R_INTC0_Get_INTP0_Flag,<br>R_INTC0_Clear_INTP0_Flag                                                                                 |
| r_cg_timer_user.c | TAU module<br>Capture end timer interrupt (INTTM05)         | Additional functions:<br>R_TAU0_Channel5_MeasureStart,<br>R_TAU0_Channel5_Get_MeasureStatus,<br>R_TAU0_Channel5_Clear_MeasureStatus                          |
| r_cg_it_user.c    | Interval timer module<br>Interval timer interrupt           | Additional functions:<br>R_IT_Get_fCLK_ChangeFlag,<br>R_IT_Clear_fCLK_ChangeFlag,<br>R_IT_Get_INTIT_Flag<br>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.<br>(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.<br>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.<br>It is predefined as one of these values:<br>0x00: 32 MHz<br>0x01: 16 MHz<br>0x02: 8 MHz<br>0x03: 4 MHz<br>0x04: 2 MHz<br>0x05: 1 MHz |
| TAU0_COUNT_CLOCK_2        | 0x01    | Count clock 2 for TAU0.<br>Same as TAU0_COUNTCLOCK_1.                                                                                                           |

Note 1: The above PULSEWIDTH\_RANGE\_MIN and PULSEWIDTH\_RANGE\_MAX are calculated as follows.

$$\text{PULSEWIDTH\_RANGE\_MIN} = \text{HOCO\_FREQUENCY\_MIN} / \text{LOCO\_FREQUENCY\_MAX}$$

$$\text{PULSEWIDTH\_RANGE\_MAX} = \text{HOCO\_FREQUENCY\_MAX} / \text{LOCO\_FREQUENCY\_MIN}$$

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,<br>R_TAU0_Channel5_Get_MeasureStatus,<br>R_TAU0_Channel5_Clear_MeasureStatus |
| uint8_t | g_fCLKChangeFlag | Count clock change request flag          | r_it_interrupt,<br>R_IT_Get_fCLK_ChangeFlag,<br>R_IT_Clear_fCLK_ChangeFlag                              |
| uint8_t | g_intp0_flag     | INTP0 external interrupt occurrence flag | r_intc0_interrupt,<br>R_INTC0_Get_INTP0_Flag,<br>R_INTC0_Clear_INTP0_Flag                               |
| uint8_t | g_intit_flag     | Interval timer interrupt occurrence flag | r_it_interrupt,<br>R_IT_Get_INTIT_Flag,<br>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<br>r_cg_intc.h<br>r_cg_userdefine.h |
| Declaration  | void R_INTC0_Start(void)                               |
| Explanation  | This function unmask 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<br>r_cg_timer.h<br>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<br>r_cg_timer.h<br>r_cg_userdefine.h                    |
| Declaration  | void R_TAU0_Channel5_Start(void)                                           |
| Explanation  | This function unmask 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<br>r_cg_timer.h<br>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<br>r_cg_it.h<br>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 | <ul style="list-style-type: none"><li>• When count clock change is not requested: 0x00</li><li>• When count clock change is requested: 0x01</li></ul> |
| Remarks      | None                                                                                                                                                  |

---

[Function Name] R\_TAU0\_Channel5\_Get\_MeasureStatus

---

|              |                                                                                                                                                               |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis     | Gets the pulse interval measurement end flag.                                                                                                                 |
| Header       | r_cg_macrodriver.h<br>r_cg_timer.h<br>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 | <ul style="list-style-type: none"><li>• When pulse interval measurement is not ended: 0x00</li><li>• When pulse interval measurement is ended: 0x01</li></ul> |
| Remarks      | None                                                                                                                                                          |

## [Function Name] R\_Main\_Get\_PulseWidthMeasureResult

|              |                                                                                                                                                                                                                                                       |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis     | Judges the pulse interval measurement value.                                                                                                                                                                                                          |
| Header       | r_cg_macrodriver.h<br>r_cg_cgc.h<br>r_cg_port.h<br>r_cg_intc.h<br>r_cg_timer.h<br>r_cg_rtc.h<br>r_cg_it.h<br>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 | <ul style="list-style-type: none"> <li>When the pulse interval measurement value is within the tolerable range: 0x00</li> <li>When the pulse interval measurement value is not within the tolerable range: 0x01</li> </ul>                            |
| Remarks      | None                                                                                                                                                                                                                                                  |

## [Function Name] R\_TAU0\_Channel5\_Get\_PulseWidth

|              |                                                                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis     | Gets the pulse interval measurement value.                                                                                                                     |
| Header       | r_cg_macrodriver.h,<br>r_cg_timer.h,<br>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    | <ul style="list-style-type: none"> <li>width                                      Address of the area to store the pulse interval measurement value</li> </ul> |
| 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,<br>r_cg_timer.h<br>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<br>r_cg_cgc.h<br>r_cg_port.h<br>r_cg_intc.h<br>r_cg_timer.h<br>r_cg_rtc.h<br>r_cg_it.h<br>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<br>r_cg_cgc.h<br>r_cg_port.h<br>r_cg_intc.h<br>r_cg_timer.h<br>r_cg_rtc.h<br>r_cg_it.h<br>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    | <ul style="list-style-type: none"> <li>clock TAU0_COUNT_CLOCK1 or TAU0_COUNT_CLOCK2</li> </ul>                                                               |
|              | 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<br>r_cg_cgc.h<br>r_cg_port.h<br>r_cg_intc.h<br>r_cg_timer.h<br>r_cg_rtc.h<br>r_cg_it.h<br>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<br>r_cg_cgc.h<br>r_cg_port.h<br>r_cg_intc.h<br>r_cg_timer.h<br>r_cg_rtc.h<br>r_cg_it.h<br>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<br>r_cg_rtc.h<br>r_cg_userdefine.h                             |
| Declaration  | void R_RTC_Start(void)                                                            |
| Explanation  | This function unmaskes 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<br>r_cg_rtc.h<br>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<br>r_cg_rtc.h<br>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    | <ul style="list-style-type: none"> <li>period</li> </ul> <div> Period between fixed-frequency interrupts<br/> HALFSEC: 0.5 second <math>\times (f_{SUB}/f_{IL})</math><br/> ONESEC: 1 second <math>\times (f_{SUB}/f_{IL})</math><br/> ONEMIN: 1 minute <math>\times (f_{SUB}/f_{IL})</math><br/> ONEHOUR: 1 hour <math>\times (f_{SUB}/f_{IL})</math><br/> ONEDAY: 1 day <math>\times (f_{SUB}/f_{IL})</math><br/> ONEMONTH: 1 year <math>\times (f_{SUB}/f_{IL})</math> </div> |
| Return value | <ul style="list-style-type: none"> <li>Normal end: MD_OK (0x00)</li> <li>Argument specification error: MD_ARGERROR (0x01)</li> </ul>                                                                                                                                                                                                                                                                                                                                             |
| Remarks      | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

**[Function Name] R\_RTC\_Set\_ConstPeriodInterruptOff**


---

|              |                                                         |
|--------------|---------------------------------------------------------|
| Synopsis     | Stops real-time clock fixed-frequency interrupts.       |
| Header       | r_cg_macrodriver.h,<br>r_cg_rtc.h,<br>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<br>r_cg_it.h<br>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<br>r_cg_it.h<br>r_cg_userdefine.h                                                                                  |
| Declaration  | __interrupt void 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<br>r_cg_timer.h<br>r_cg_userdefine.h                                                                                               |
| Declaration  | __interrupt void 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<br>r_cg_intc.h<br>r_cg_it.h<br>r_cg_userdefine.h                                 |
| Declaration  | __interrupt void 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<br>r_cg_rtc.h<br>r_cg_userdefine.h                         |
| Declaration  | __interrupt static void 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<br>r_cg_rtc.h<br>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<br>r_cg_it.h<br>r_cg_userdefine.h                          |
| Declaration  | void R_IT_Start(void)                                                         |
| Explanation  | This function starts the interval timer and unmask 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<br>r_cg_it.h<br>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<br>r_cg_intc.h<br>r_cg_it.h<br>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 | <ul style="list-style-type: none"> <li>When an INT0 external interrupt has not occurred: 0x00</li> <li>When an INT0 external interrupt has occurred: 0x01</li> </ul> |
| Remarks      | None                                                                                                                                                                 |

---

[Function Name] R\_INTC0\_Clear\_INTP0\_Flag

---

|              |                                                                     |
|--------------|---------------------------------------------------------------------|
| Synopsis     | Clears the INTP0 external interrupt occurrence flag.                |
| Header       | r_cg_macrodriver.h<br>r_cg_intc.h<br>r_cg_it.h<br>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<br>r_cg_it.h<br>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 | <ul style="list-style-type: none"> <li>When an interval timer interrupt has not occurred: 0x00</li> <li>When an interval timer interrupt has occurred: 0x01</li> </ul> |
| Remarks      | None                                                                                                                                                                   |

---

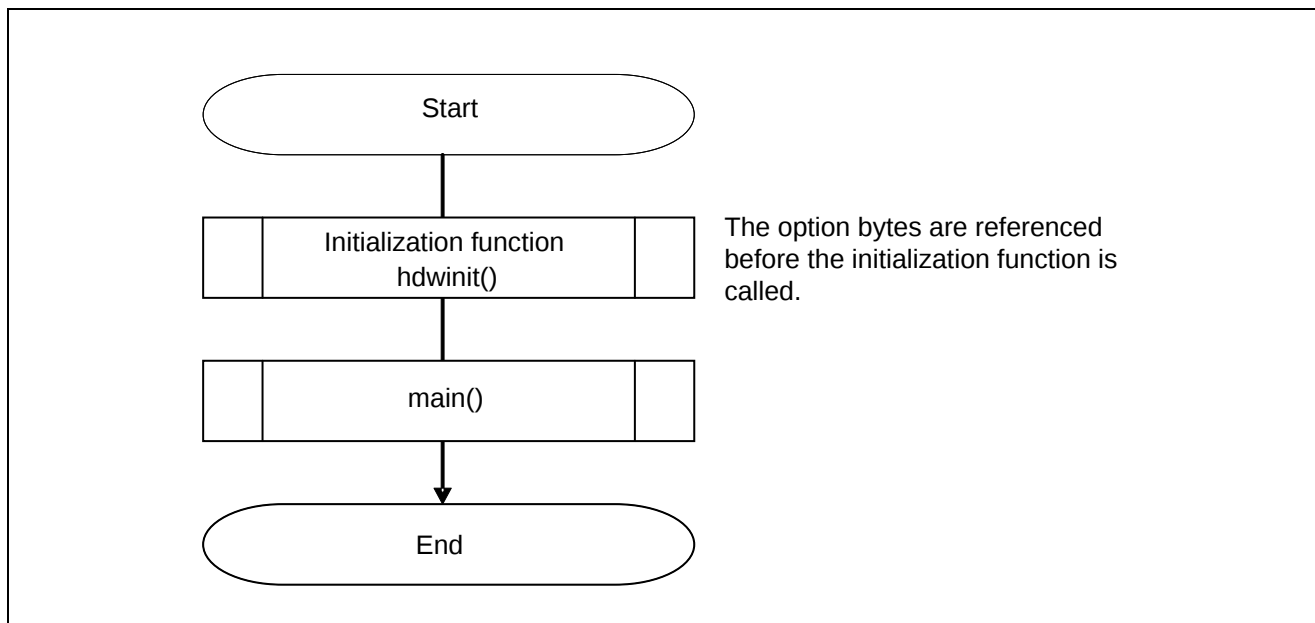
[Function Name] R\_IT\_Clear\_INTIT\_Flag

---

|              |                                                                    |
|--------------|--------------------------------------------------------------------|
| Synopsis     | Clears the interval timer interrupt occurrence flag.               |
| Header       | r_cg_macrodriver.h<br>r_cg_it.h<br>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.



**Figure 5.1 Overall Flow**

### 5.8.1 Initialization Function

Figure 5.2 shows the flowchart for the initialization function.

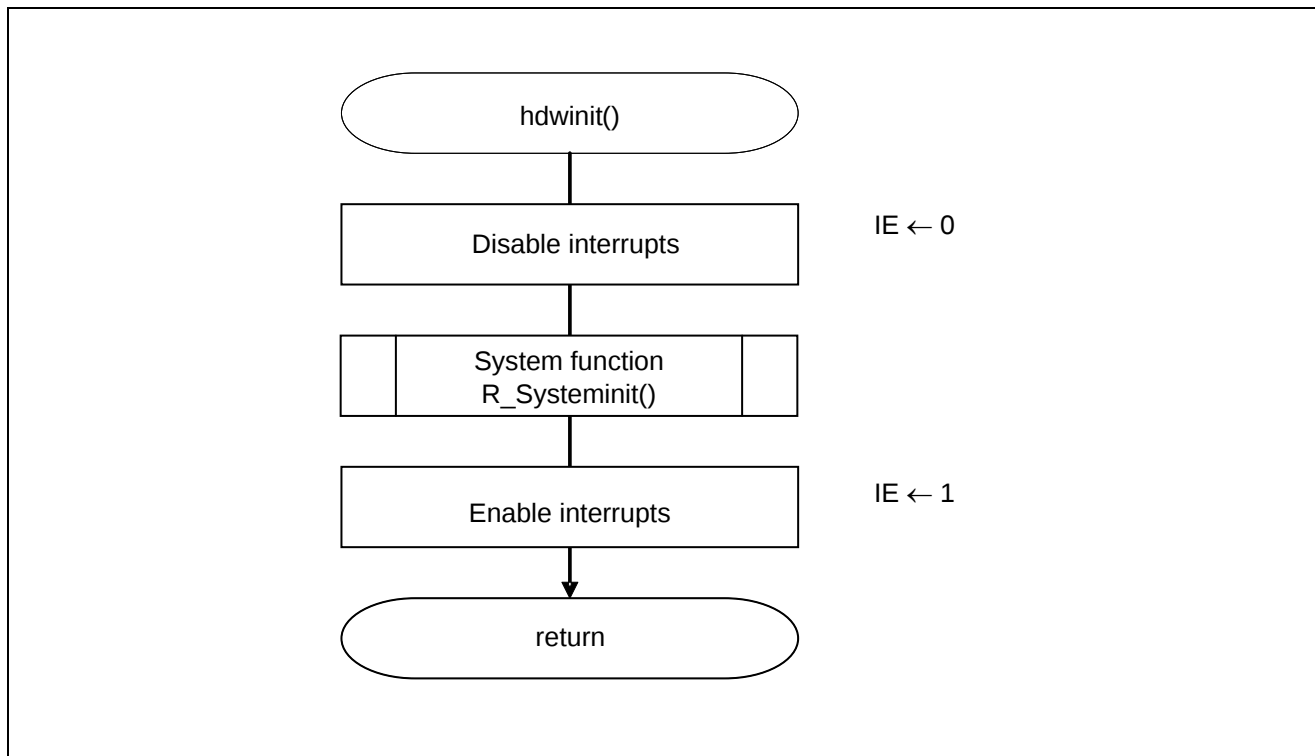


Figure 5.2 Initialization Function

### 5.8.2 System Function

Figure 5.3 shows the flowchart for the system function.

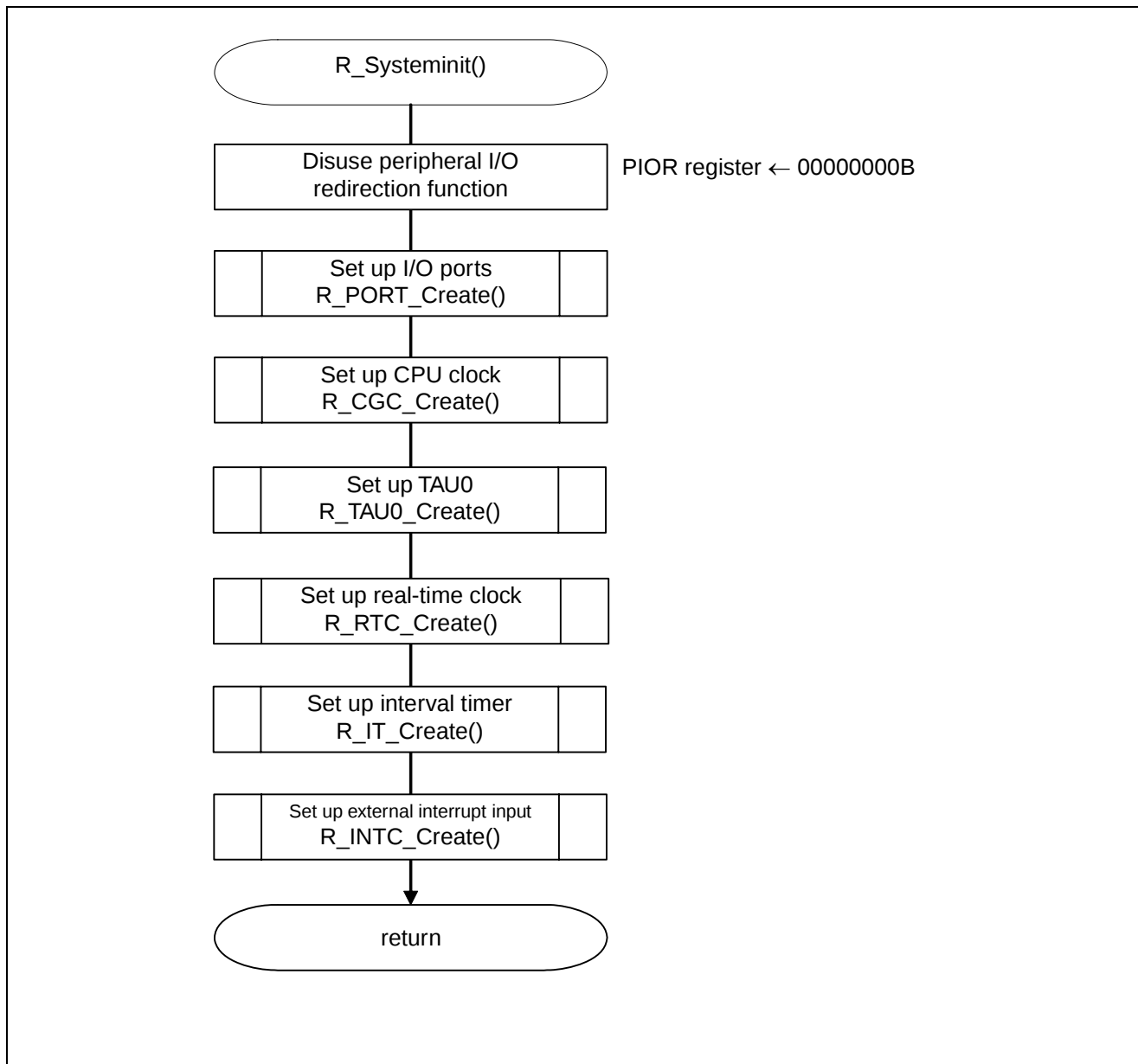
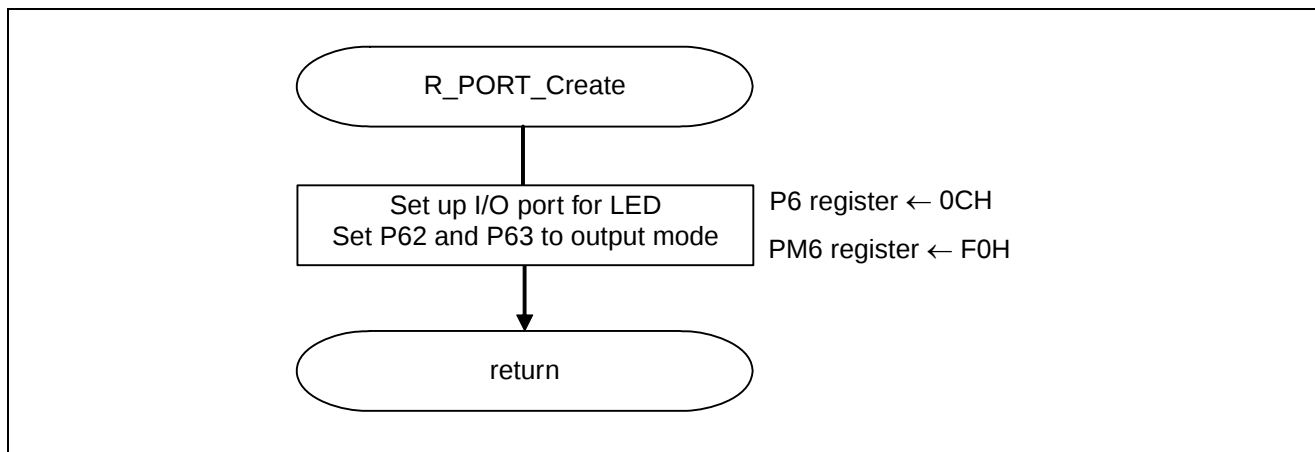


Figure 5.3 System Function

### 5.8.3 I/O Port Setup

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

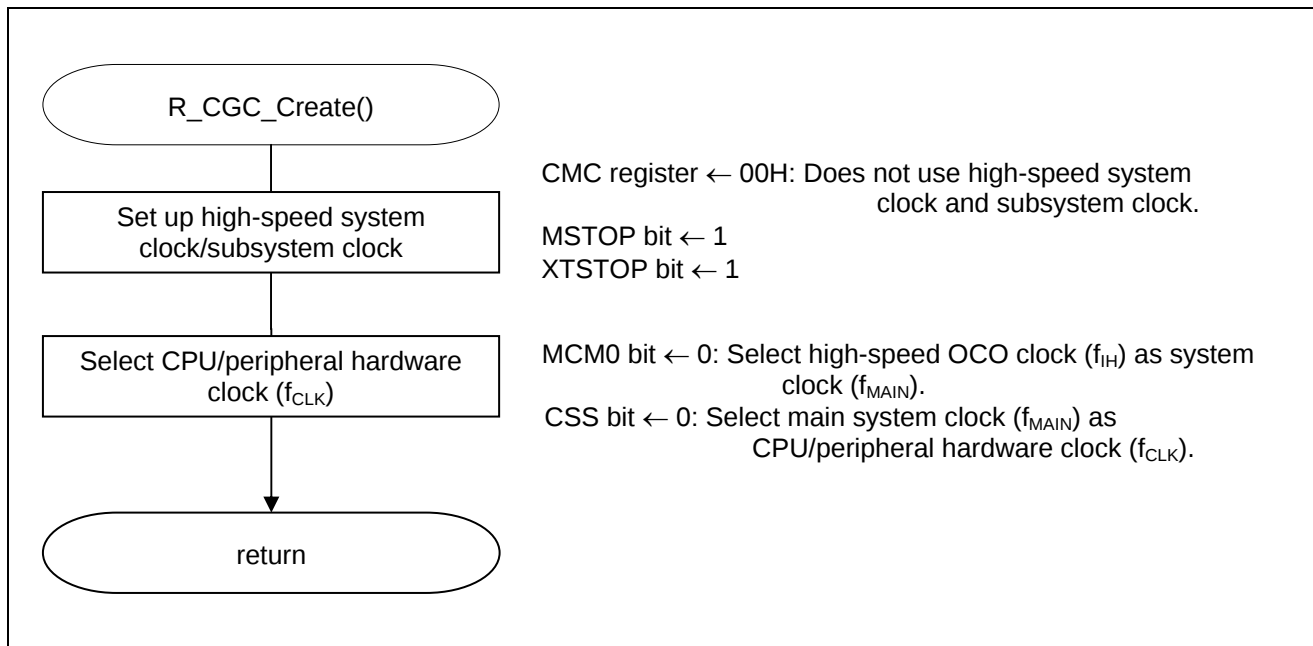


**Figure 5.4 I/O Port Setup**

- Cautions:
1. Refer to the section entitled "Flowcharts" in RL78/G13 Initialization Application Note (R01AN0451E) 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.



**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 (R01AN0451E).

### 5.8.5 TAU0 Setup

Figure 5.6 shows the flowchart for setting up the TAU0.

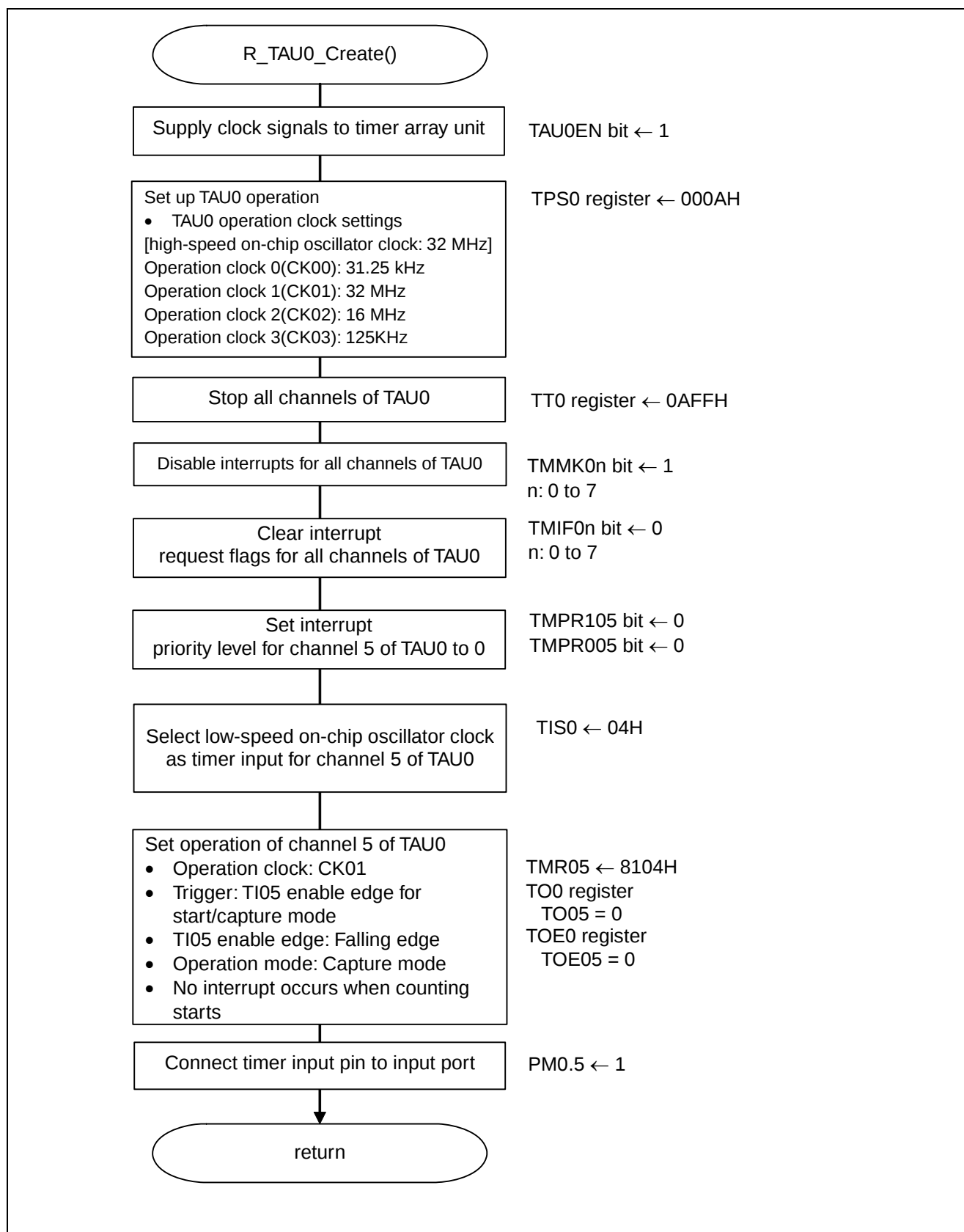


Figure 5.6 TAU 0 Setup

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      | <b>1</b> |

Bit 0

| TAU0EN   | Control of timer array unit input clock supply |
|----------|------------------------------------------------|
| 0        | Stops input clock supply.                      |
| <b>1</b> | <b>Enables input clock supply.</b>             |

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<br>031 | PRS<br>030 | 0  | 0  | PRS<br>021 | PRS<br>020 | PRS<br>013 | PRS<br>012 | PRS<br>011 | PRS<br>010 | PRS<br>003 | PRS<br>002 | PRS<br>001 | PRS<br>000 |
| 0  | 0  | 0          | 0          | 0  | 0  | 0          | 0          | 0          | 0          | 0          | 0          | 1          | 0          | 1          | 0          |

| PRS<br>003 | PRS<br>002 | PRS<br>001 | PRS<br>000 | Selection of operation clock (CK00) |                      |                      |                       |                       |                       |
|------------|------------|------------|------------|-------------------------------------|----------------------|----------------------|-----------------------|-----------------------|-----------------------|
|            |            |            |            |                                     | $f_{CLK}$<br>= 2 MHz | $f_{CLK}$<br>= 5 MHz | $f_{CLK}$<br>= 10 MHz | $f_{CLK}$<br>= 20 MHz | $f_{CLK}$<br>= 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          | $f_{CLK}/2^{10}$                    | 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<br>013 | PRS<br>012 | PRS<br>011 | PRS<br>010 | Selection of operation clock (CK01) |                      |                      |                       |                       |                       |
|------------|------------|------------|------------|-------------------------------------|----------------------|----------------------|-----------------------|-----------------------|-----------------------|
|            |            |            |            |                                     | $f_{CLK}$<br>= 2 MHz | $f_{CLK}$<br>= 5 MHz | $f_{CLK}$<br>= 10 MHz | $f_{CLK}$<br>= 20 MHz | $f_{CLK}$<br>= 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          | $f_{CLK}/2^{10}$                    | 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<br>H03 | 0  | TT<br>H01 | 0 | TT<br>07 | TT<br>06 | TT<br>05 | TT<br>04 | TT<br>03 | TT<br>02 | TT<br>01 | TT<br>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<br>051 | CKS<br>050 | 0  | CCS<br>05 | 0  | STS<br>052 | STS<br>051 | STS<br>050 | CIS<br>051 | CIS<br>050 | 0 | 0 | MD<br>053 | MD<br>052 | MD<br>051 | MD<br>050 |
| 1          | 0          | 0  | 0         | 0  | 0          | 0          | 1          | 0          | 0          | 0 | 0 | 0         | 1         | 0         | 0         |

Bits 3 to 0

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

Bits 7 and 6

| CIS<br>001 | CIS<br>000 | Selection of TI00 pin input valid edge         |
|------------|------------|------------------------------------------------|
| 0          | 0          | <b>Falling edge</b>                            |
| 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<br>051 | CKS<br>050 | 0  | CCS<br>05 | 0  | STS<br>052 | STS<br>051 | STS<br>050 | CIS<br>051 | CIS<br>050 | 0 | 0 | MD<br>053 | MD<br>052 | MD<br>051 | MD<br>050 |
| 1          | 0          | 0  | 0         | 0  | 0          | 0          | 1          | 0          | 0          | 0 | 0 | 0         | 1         | 0         | 0         |

Bits 10 to 8

| STS<br>002 | STS<br>001 | STS<br>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          | <b>Valid edge of the TI00 pin input is used as both the start trigger and capture trigger.</b>                                                 |
| 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        | <b>Operates in independent channel operation function or as slave channel 0 simultaneous channel operation function.</b> |
| 1        | Operates as master channel in simultaneous channel operation function.                                                   |

Bit 12

| CCS00 | Selection of count clock ( $f_{TCLK}$ ) of channel 0                                |
|-------|-------------------------------------------------------------------------------------|
| 0     | <b>Operation clock <math>f_{MCK}</math> specified by the CKS000 and CKS001 bits</b> |
| 1     | Valid edge of a signal input to the TI00 pin                                        |

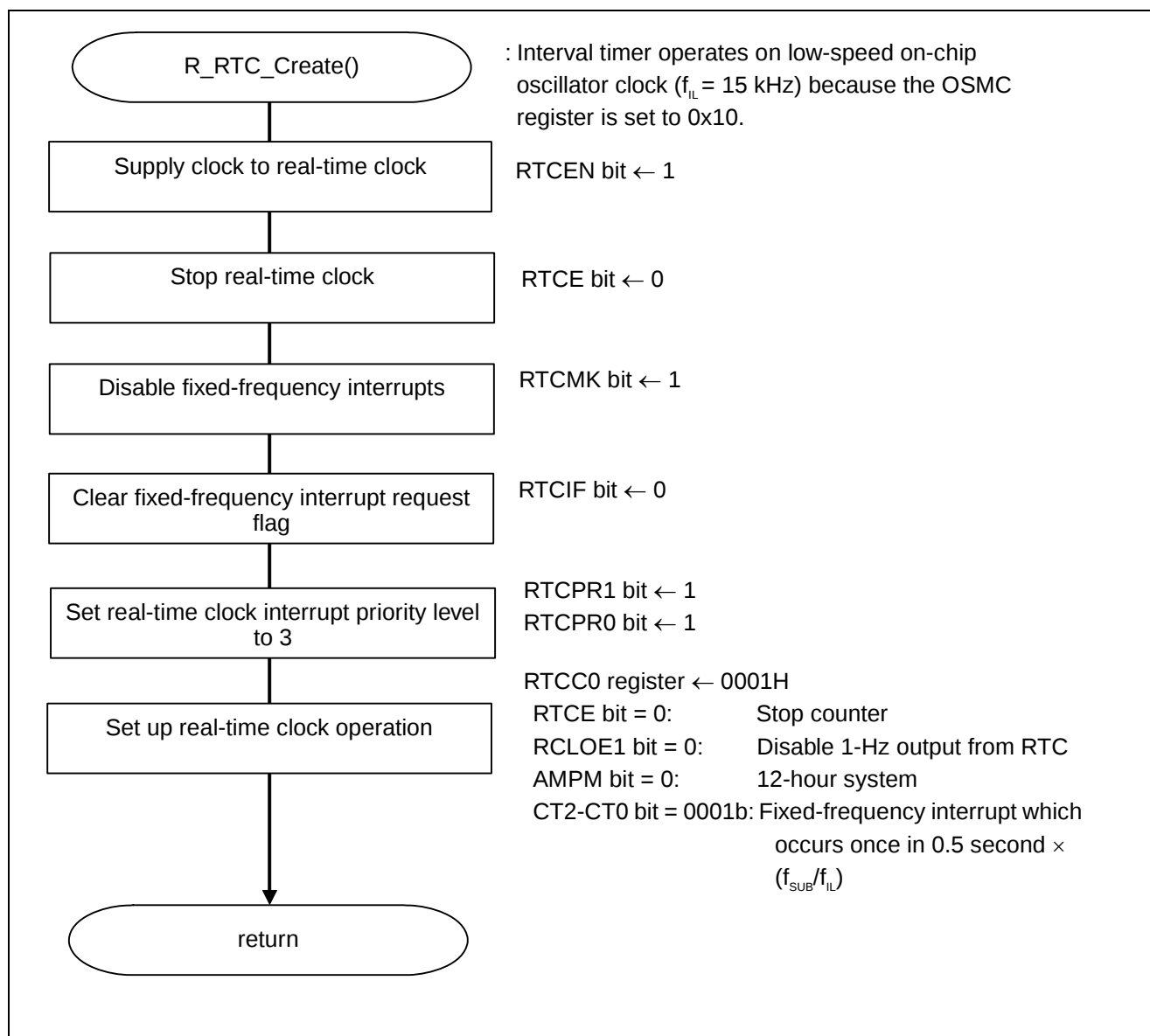
Bits 15 and 14

| CKS001 | CKS000 | Selection of operation clock ( $f_{MCK}$ ) of channel 0 |
|--------|--------|---------------------------------------------------------|
| 0      | 0      | Operation clock CK00 set by the PRS register            |
| 1      | 0      | <b>Operation clock CK01 set by the PRS register</b>     |

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

### 5.8.6 Real-Time Clock Setup

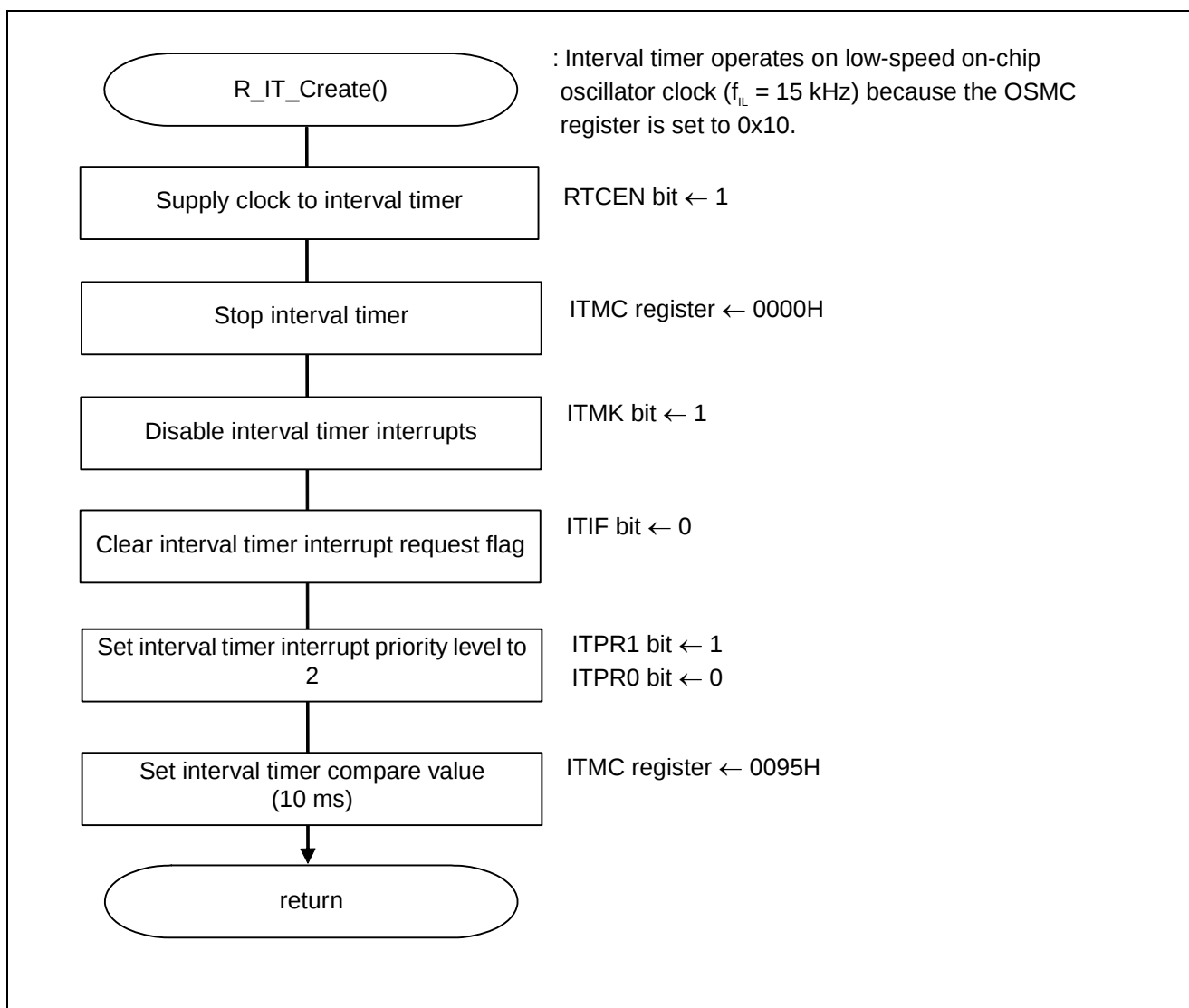
Figure 5.7 shows the flowchart for setting up the real-time clock.



**Figure 5.7 Real-Time Clock Setup**

### 5.8.7 Interval Timer Setup

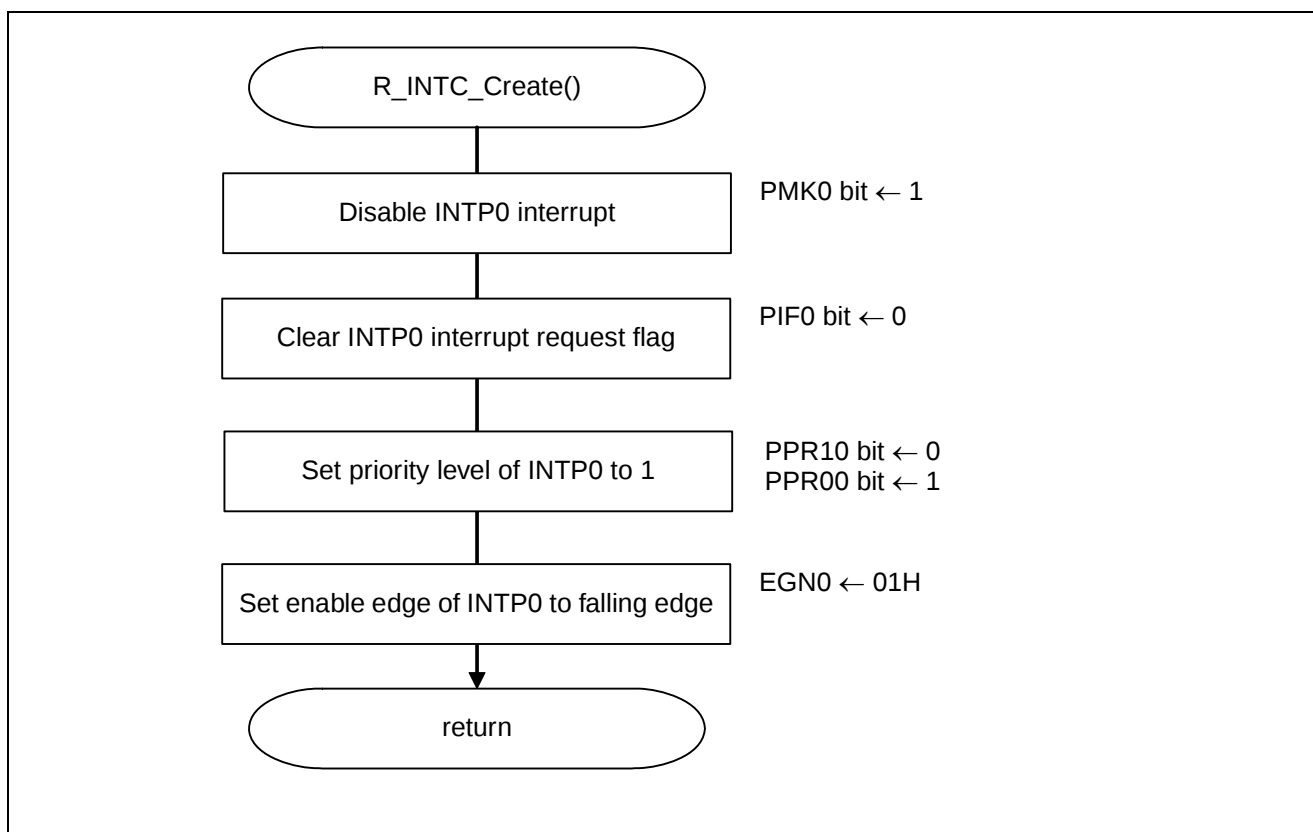
Figure 5.8 shows the flowchart for setting up the interval timer.



**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.



**Figure 5.9 External Interrupt Input Setup**

### 5.8.9 Main Processing

Figure 5.10 shows the flowchart for the main processing.

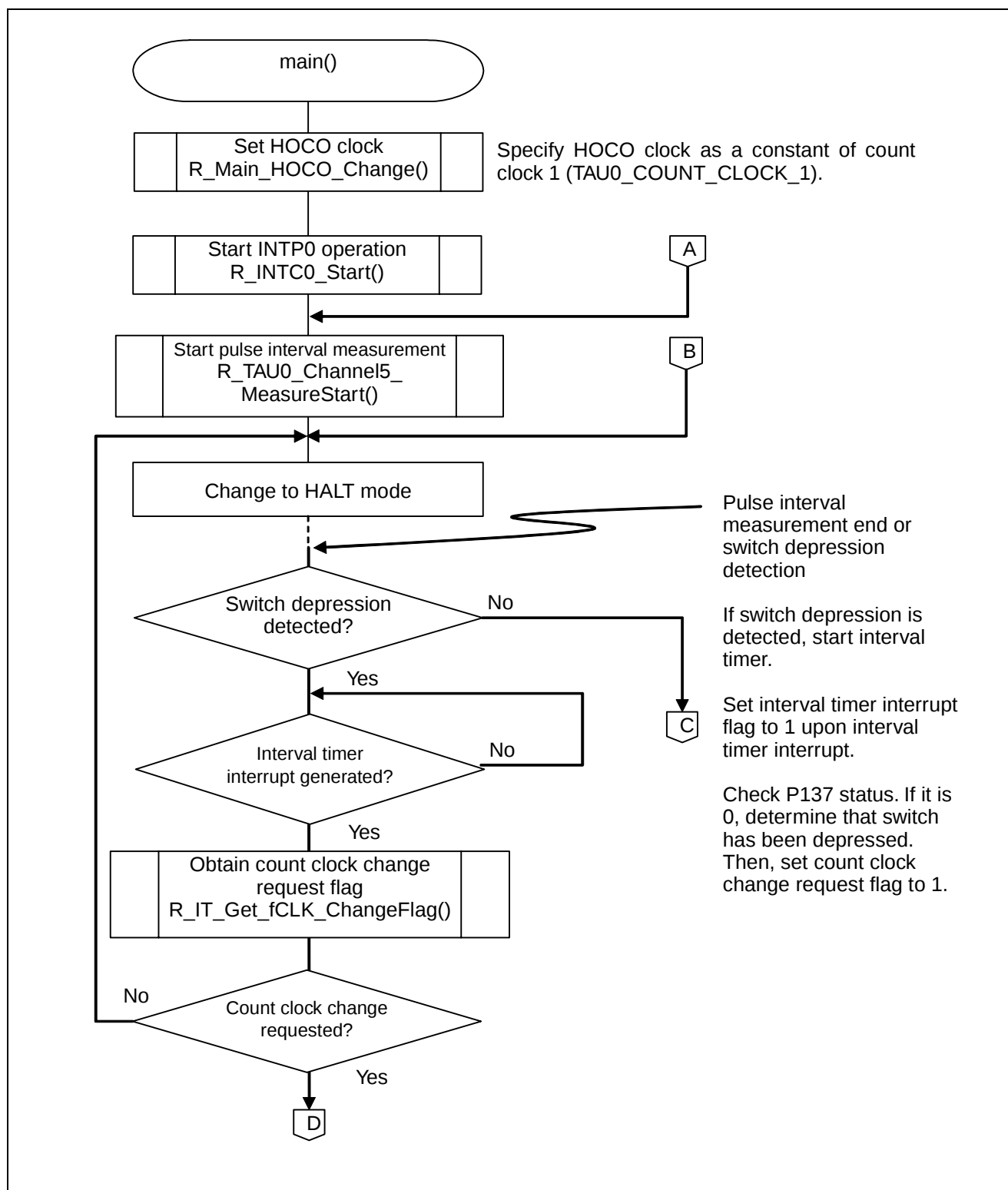


Figure 5.10 Main Processing (1/3)

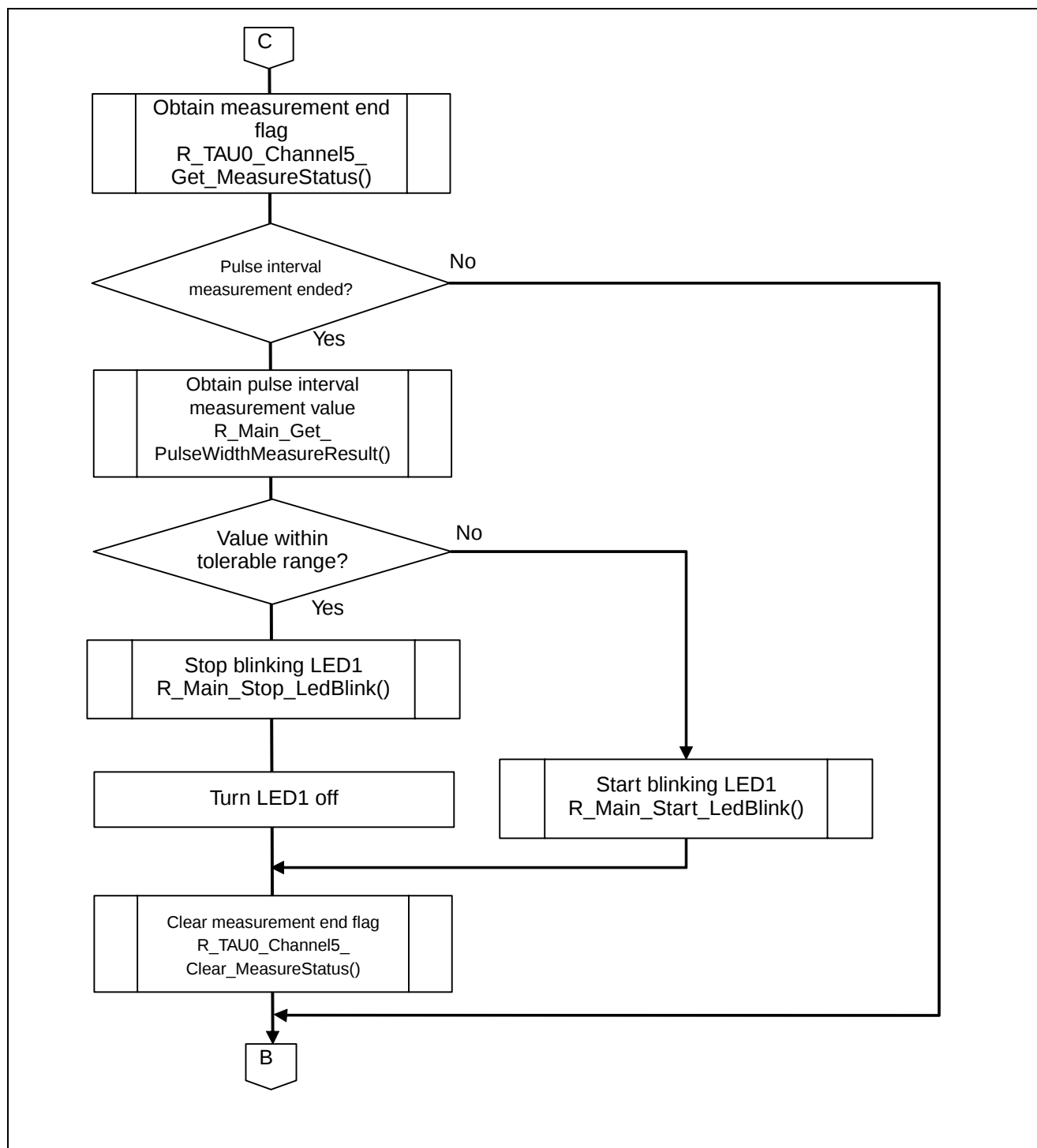
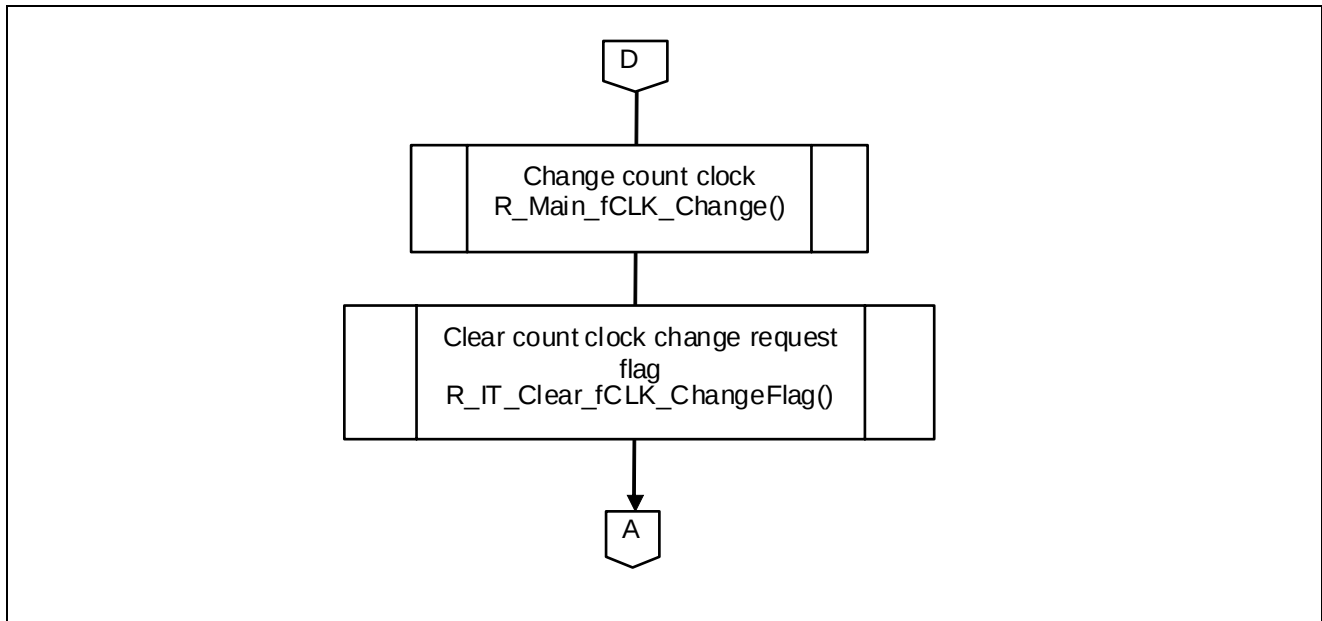
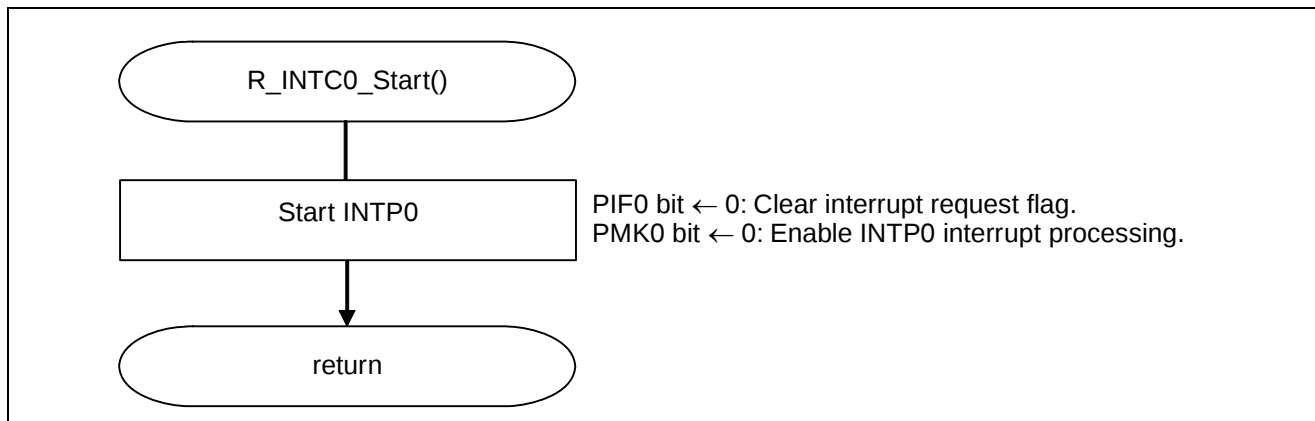


Figure 5.11 Main Processing (2/3)

**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.



**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.

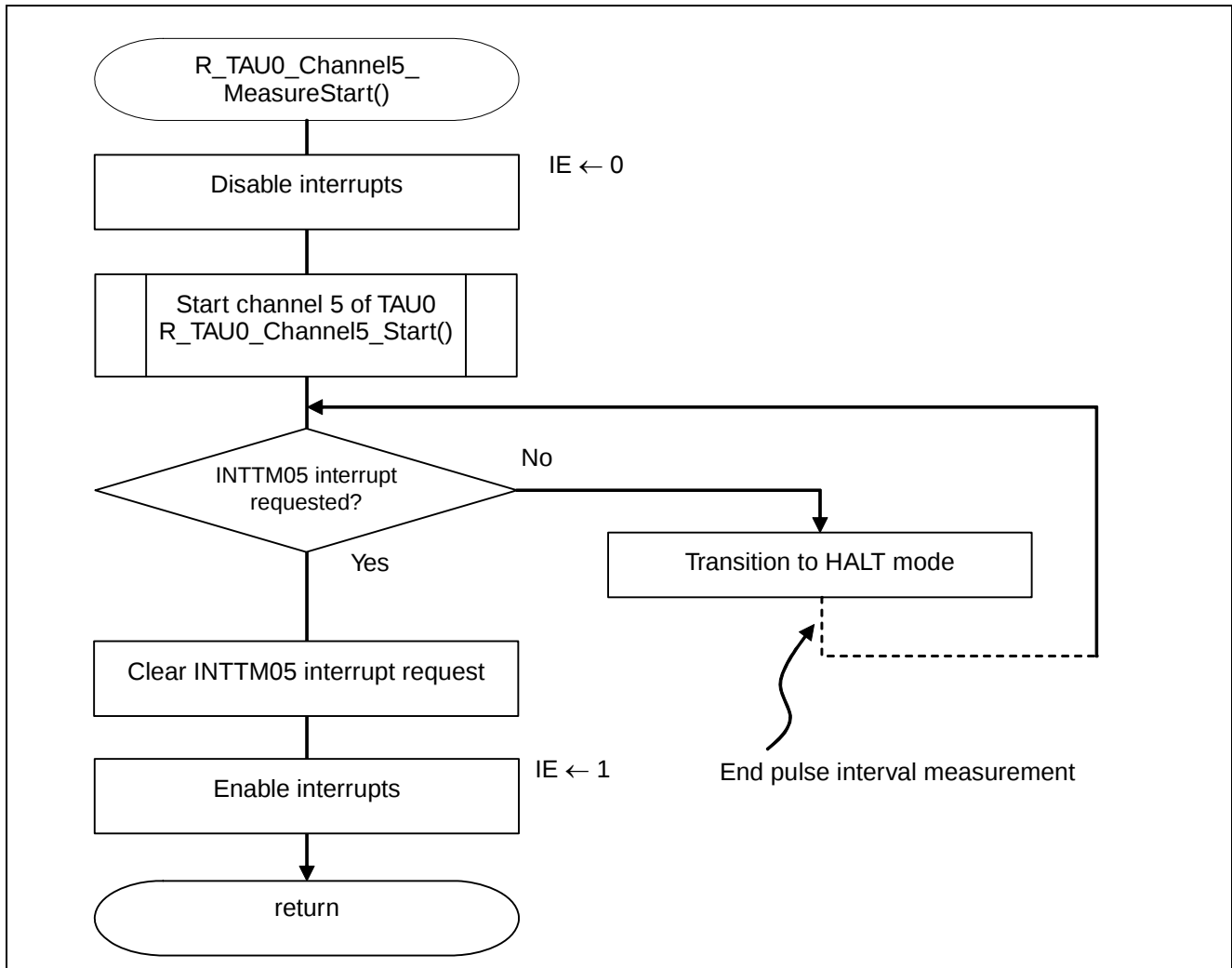
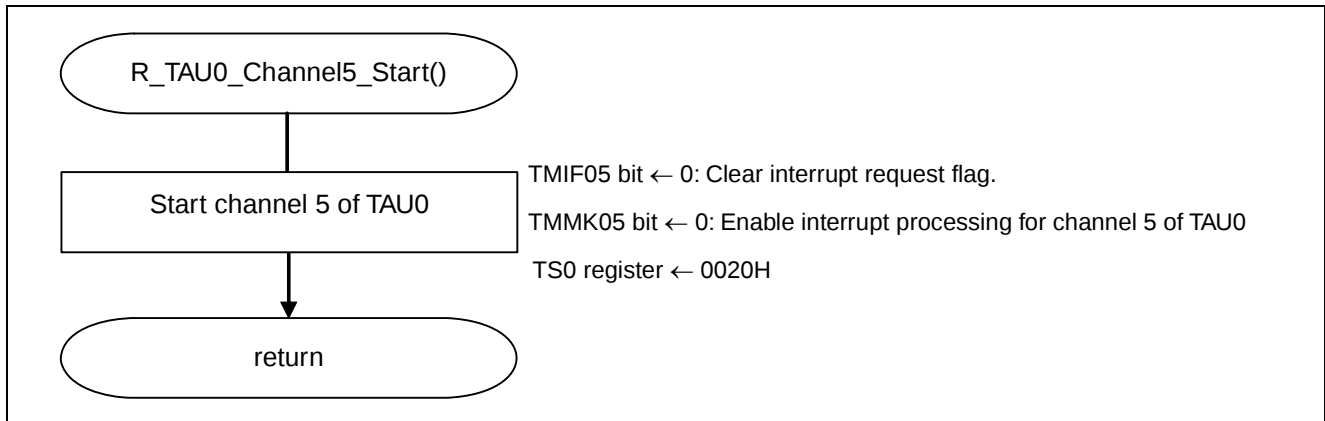


Figure 5.14 Starting Pulse Interval Measurement

### 5.8.12 Starting Channel 5 of TAU0

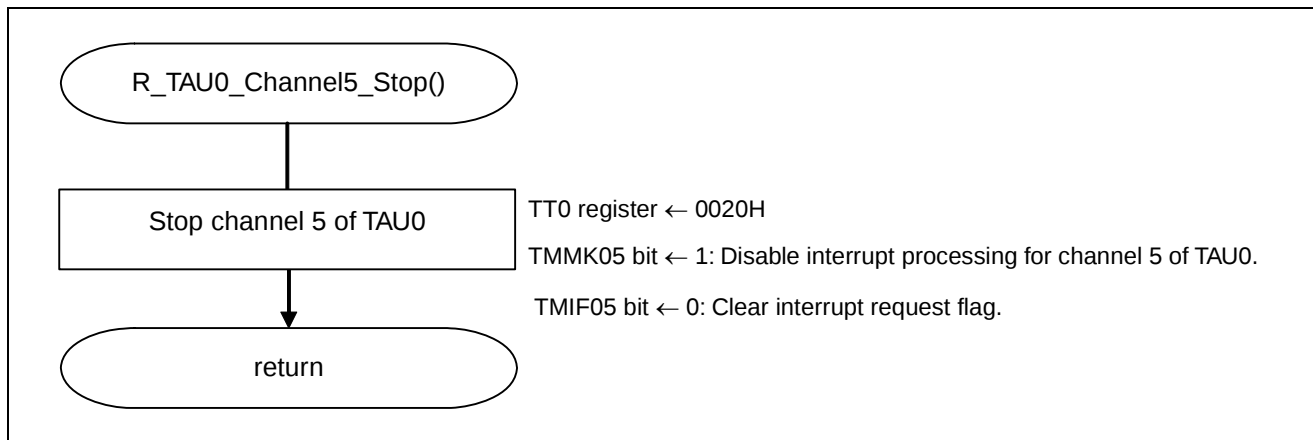
Figure 5.15 shows the flowchart for starting channel 5 of TAU0.



**Figure 5.15 Starting Channel 5 of TAU0**

### 5.8.13 Stopping Channel 5 of TAU0

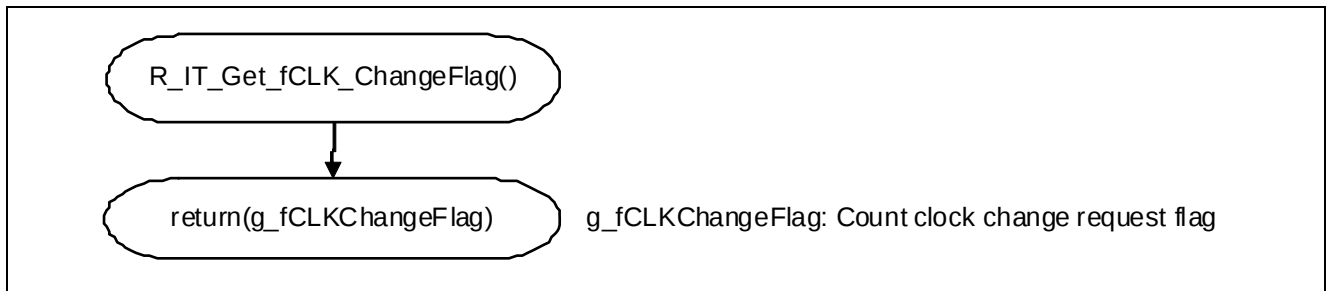
Figure 5.16 shows the flowchart for stopping channel 5 of TAU0.



**Figure 5.16 Stopping Channel 5 of TAU0**

#### 5.8.14 Getting Count Clock Change Request Flag

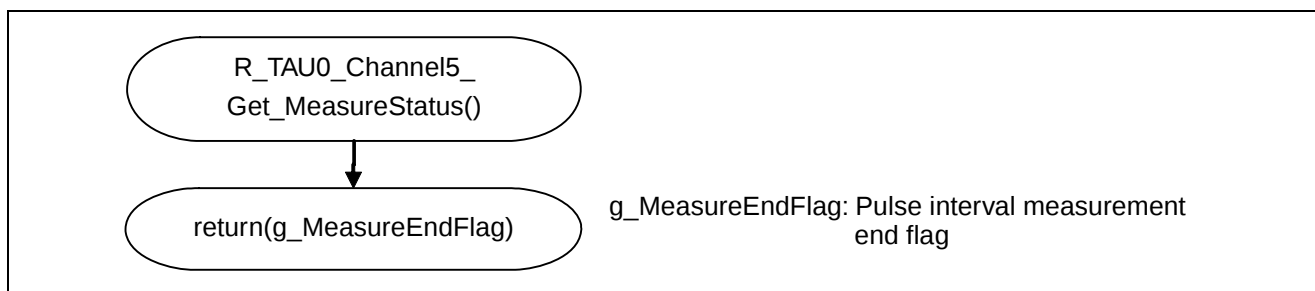
Figure 5.17 shows the flowchart for getting the count clock change request flag. This function does nothing but returns global variable g\_fCLKChangeFlag as a return value.



**Figure 5.17** Getting Count Clock Change Request Flag

### 5.8.15 Getting Pulse Interval Measurement End Flag

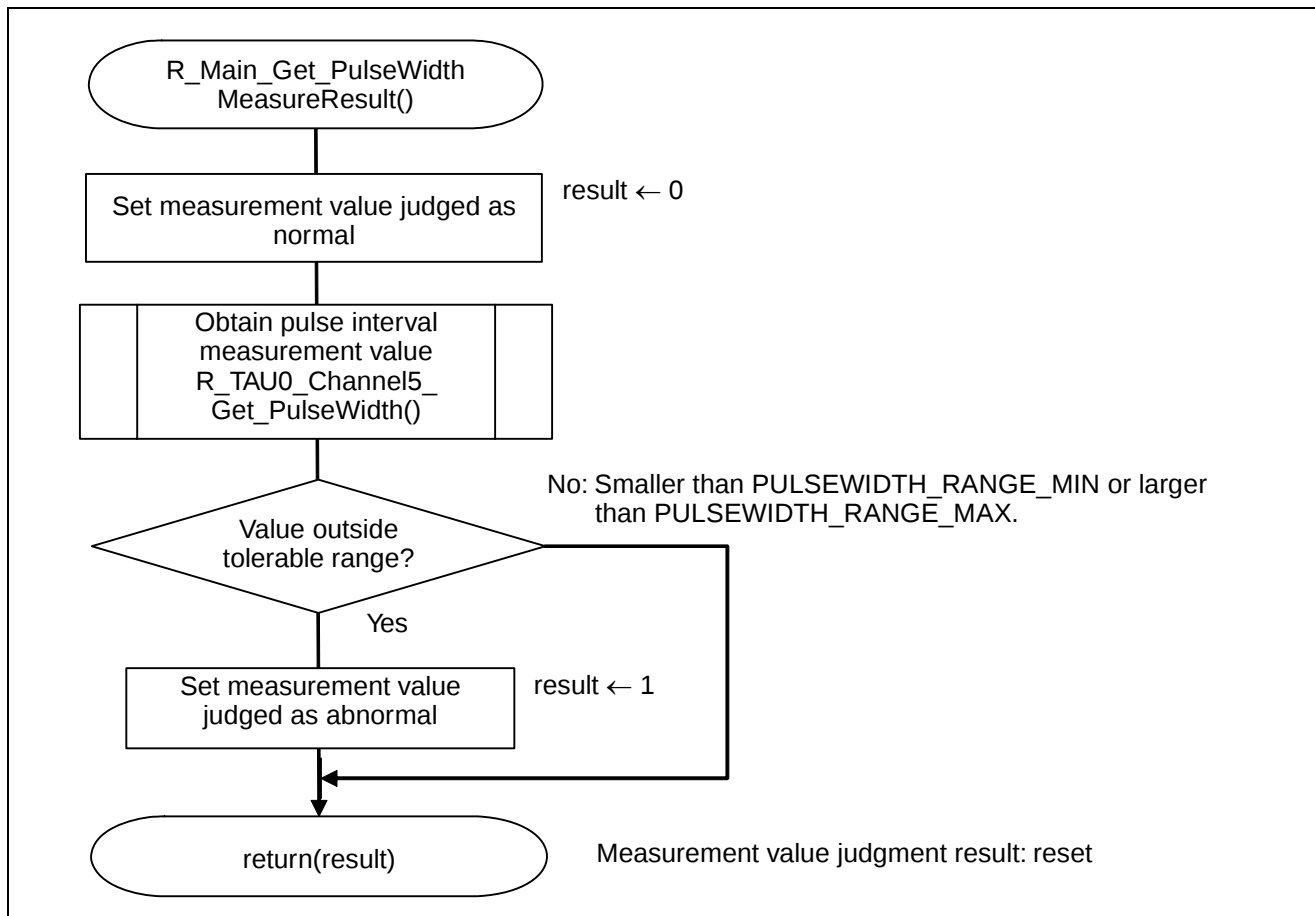
Figure 5.18 shows the flowchart for getting the pulse interval measurement end flag. This function does nothing but returns global variable `g_fCLKChangeFlag` as a return value.



**Figure 5.18** Getting Pulse Interval Measurement End Flag

### 5.8.16 Judging Pulse Interval Measurement Value

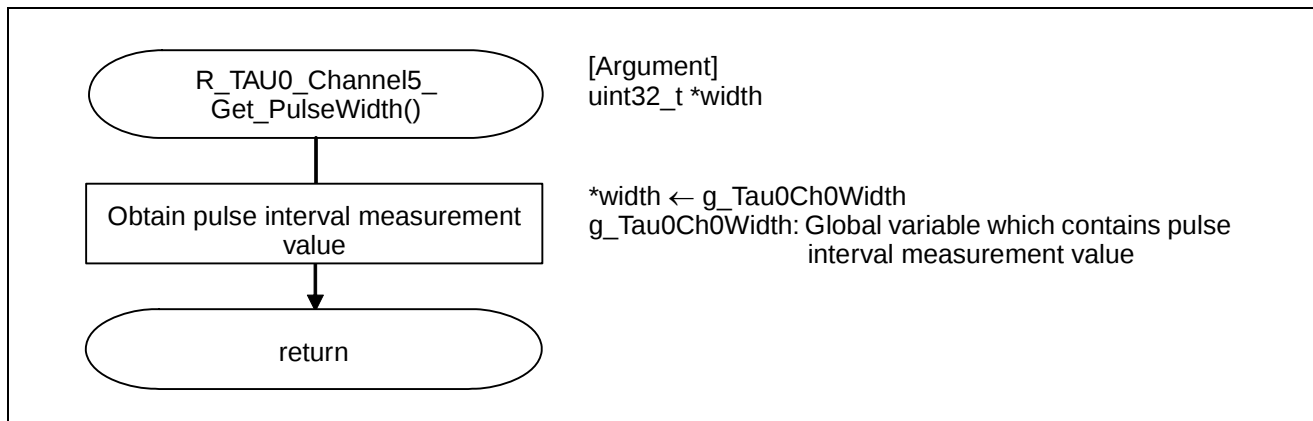
Figure 5.19 shows the flowchart for judging the pulse interval measurement value.



**Figure 5.19 Judging Pulse Interval Measurement Value**

### 5.8.17 Getting Pulse Interval Measurement Value

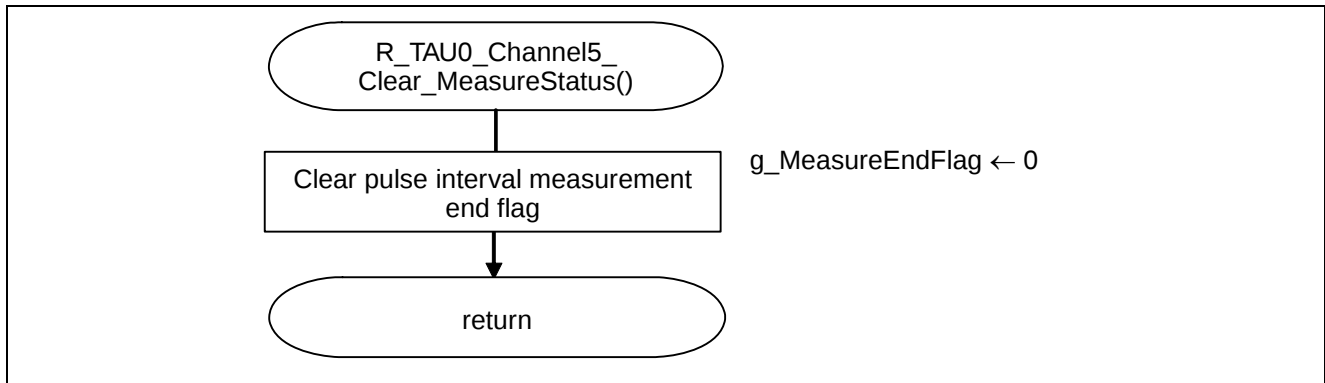
Figure 5.20 shows the flowchart for getting the pulse interval measurement value.



**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.

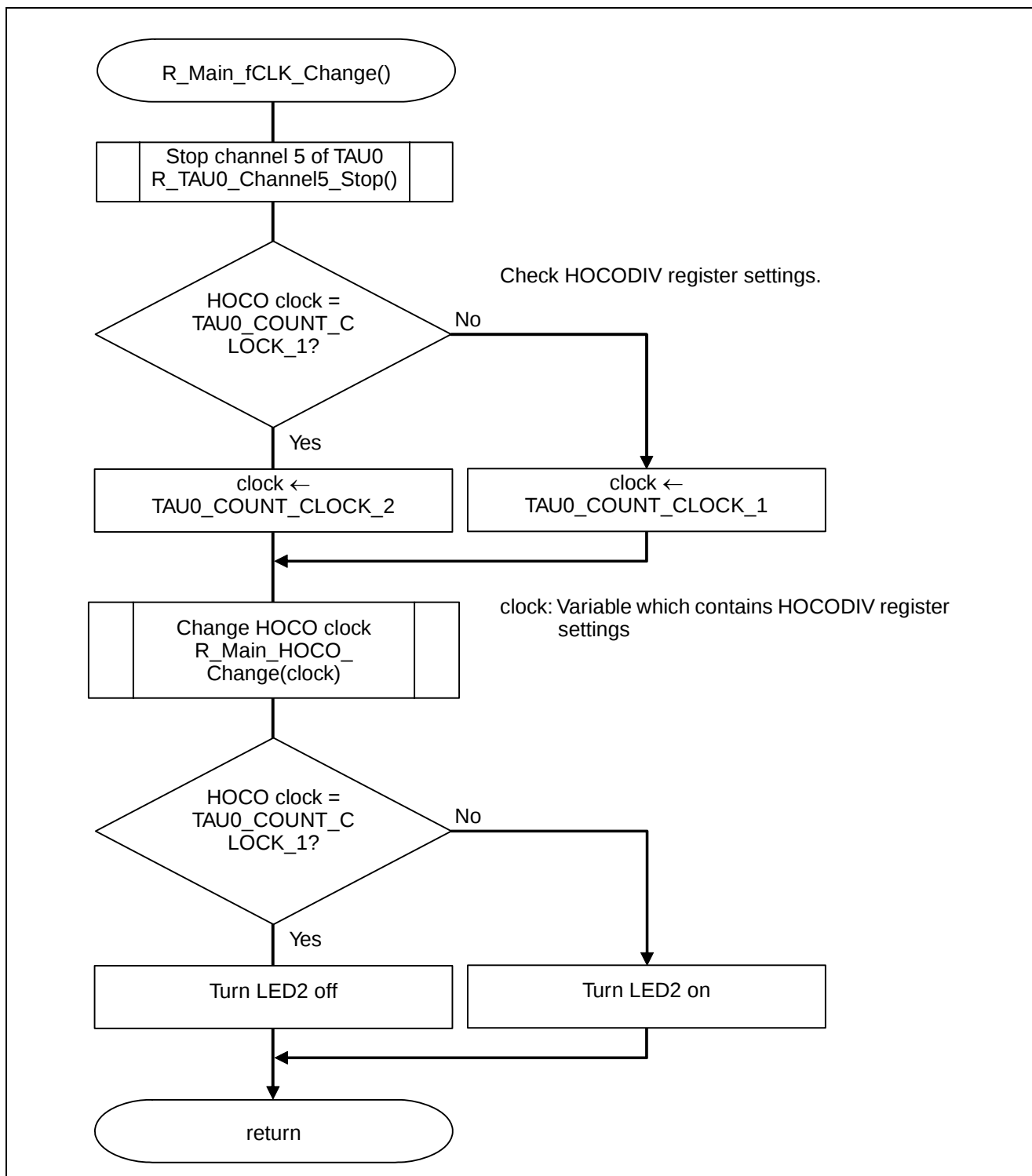


Figure 5.22 Changing Count Clock

### 5.8.20 Changing HOCO Clock

Figure 5.23 shows the flowchart for changing the HOCO clock.

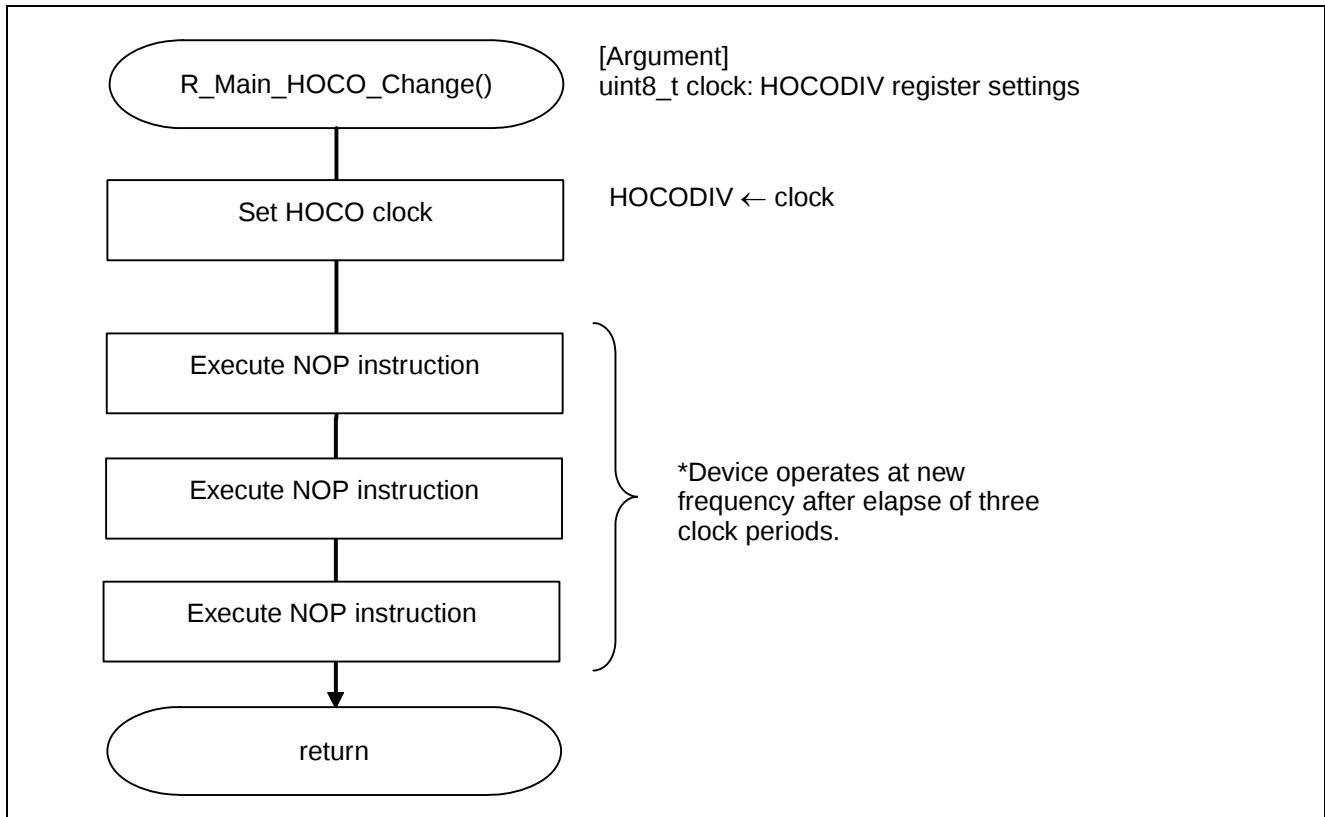


Figure 5.23 Changing HOCO Clock

### 5.8.21 Starting LED Blink

Figure 5.24 shows the flowchart for starting the LED blink.

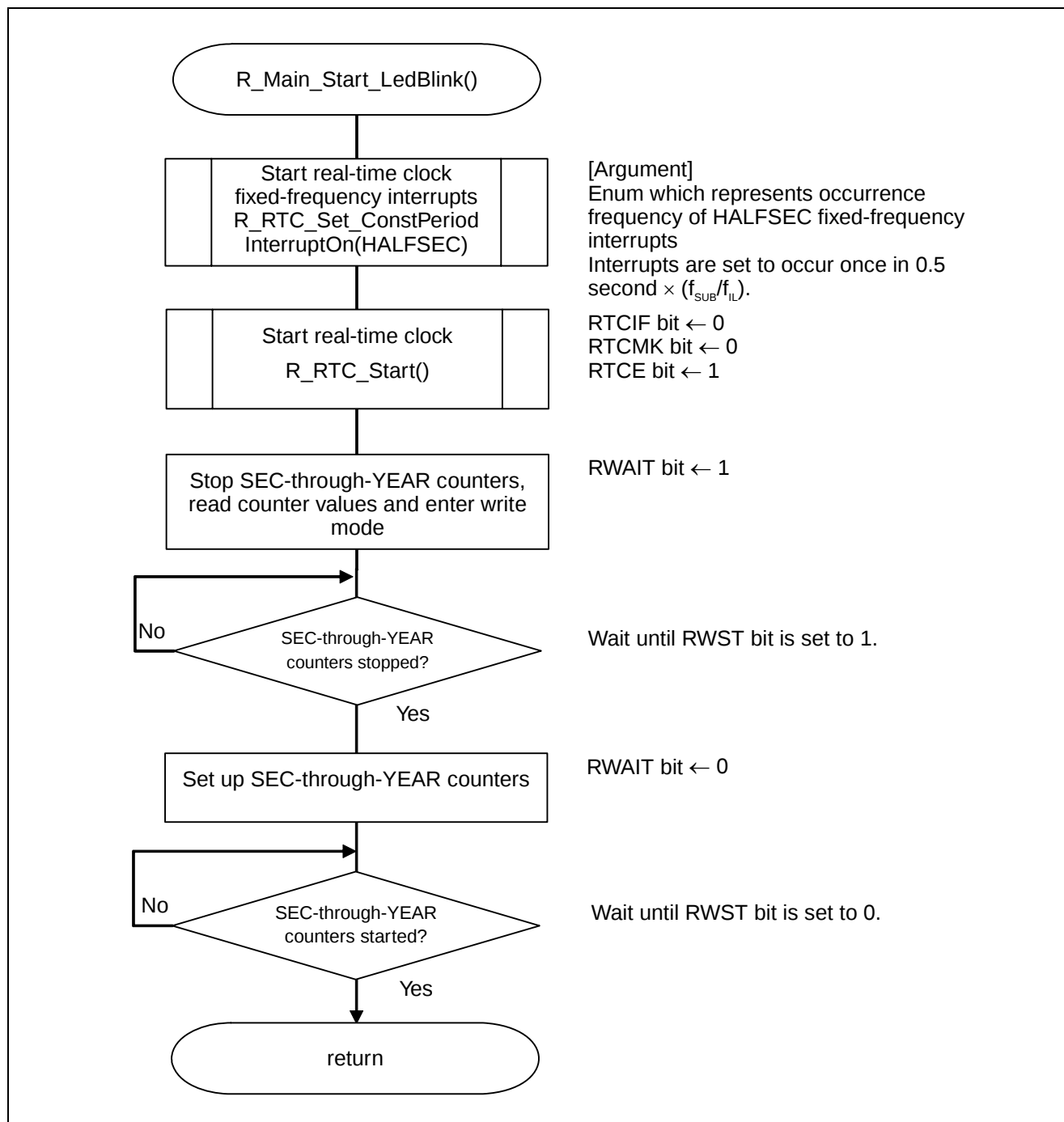


Figure 5.24 Starting LED Blink

### 5.8.22 Stopping LED Blink

Figure 5.25 shows the flowchart for stopping the LED blink.

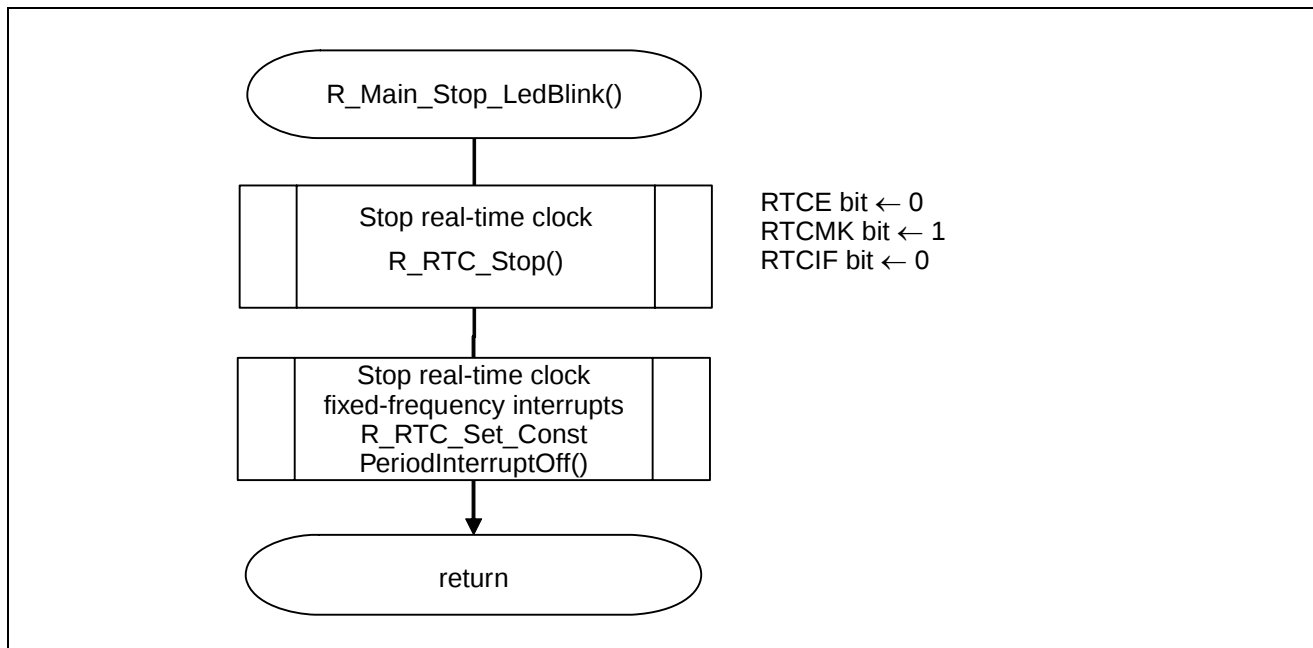
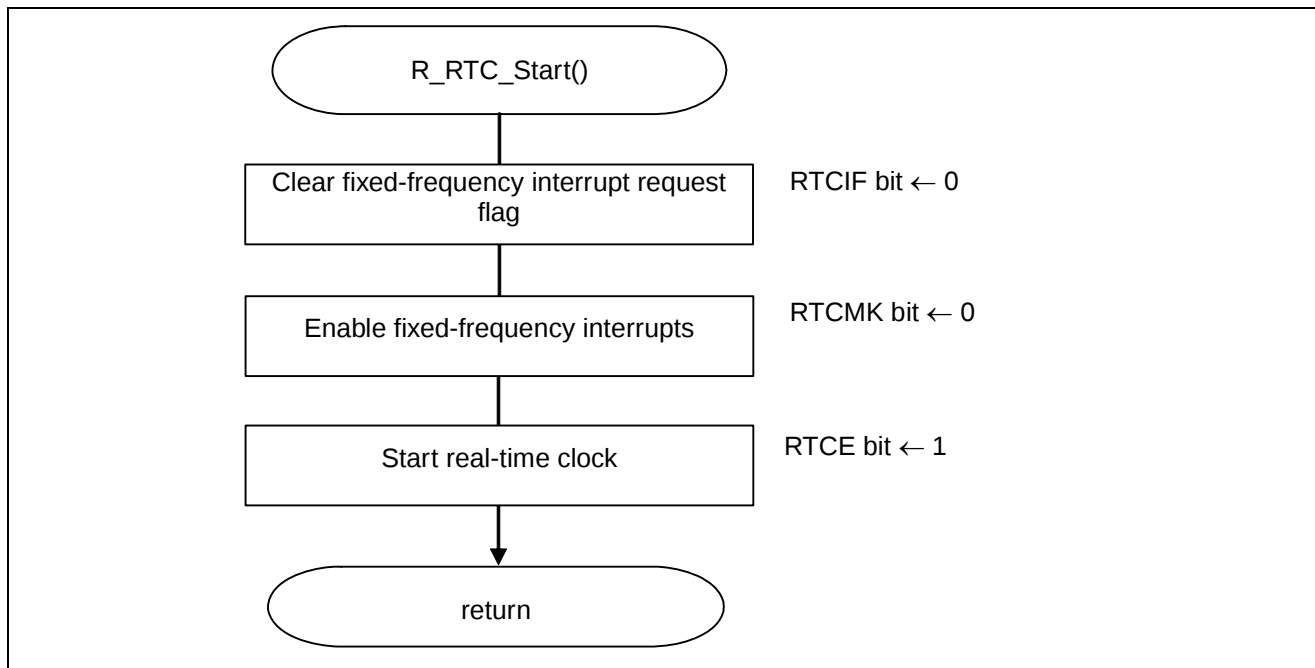


Figure 5.25 Stopping LED Blink

### 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.

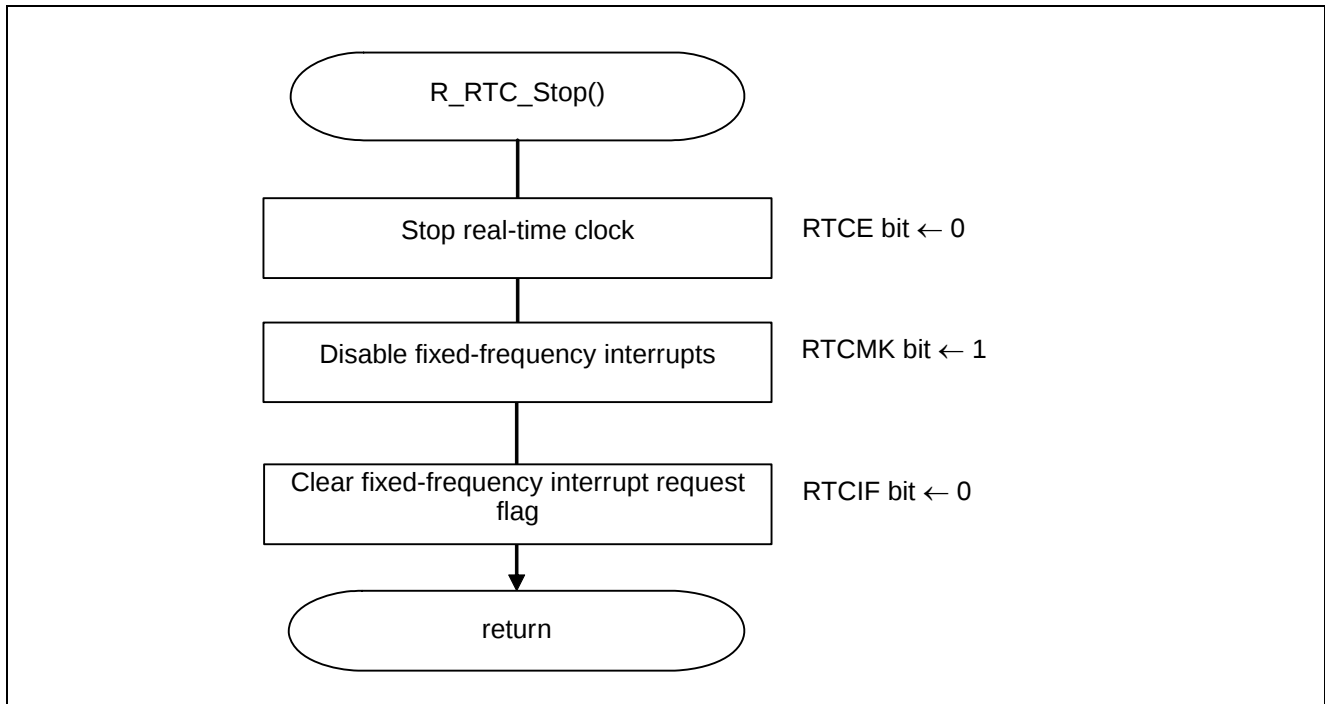
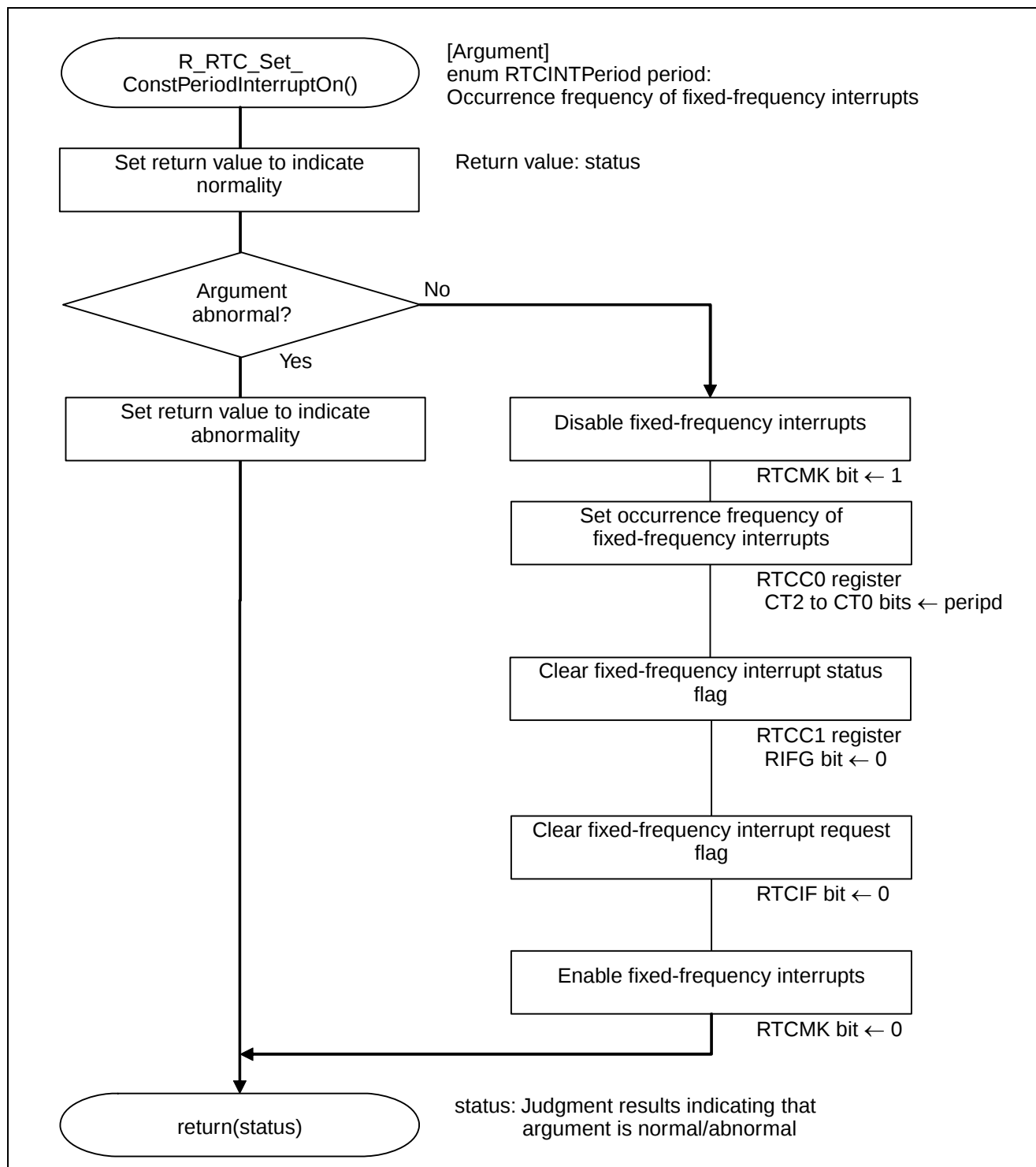


Figure 5.27 Stopping Real-Time Clock

### 5.8.25 Starting Real-Time Clock Fixed-Frequency Interrupt

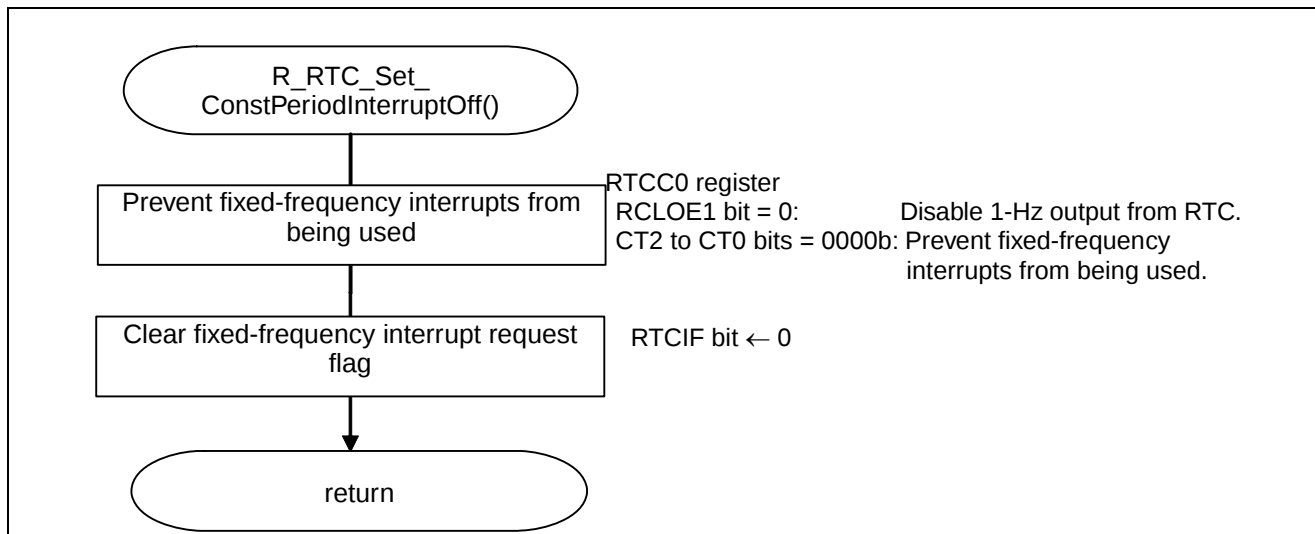
Figure 5.28 shows the flowchart for starting real-time clock fixed-frequency interrupts.



**Figure 5.28 Starting Real-Time Clock Fixed-Frequency Interrupt**

### 5.8.26 Stopping Real-Time Clock Fixed-Frequency Interrupt

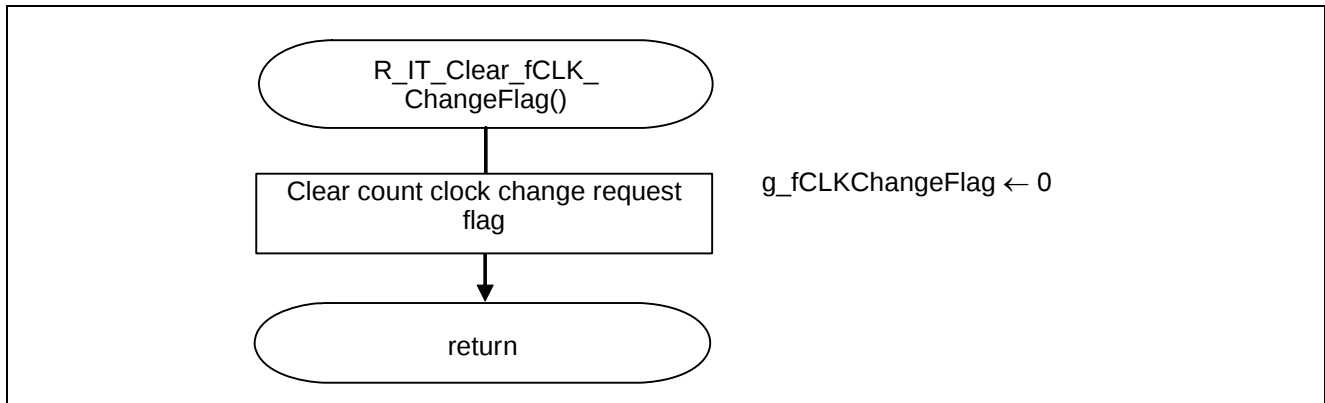
Figure 5.29 shows the flowchart for stopping real-time clock fixed-frequency interrupts.



**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.



**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.

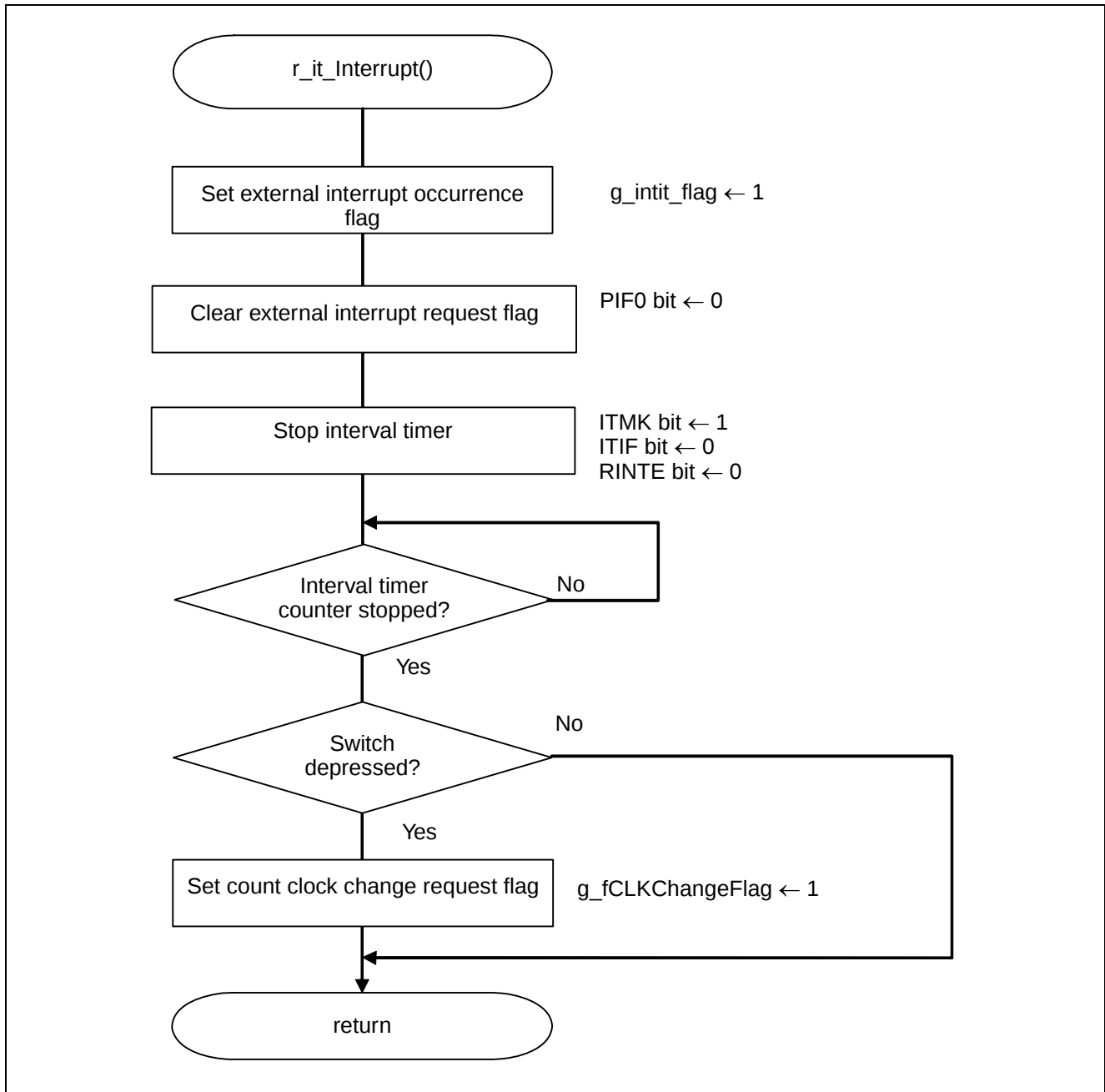


Figure 5.31 Interval Timer Interrupt

### 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.

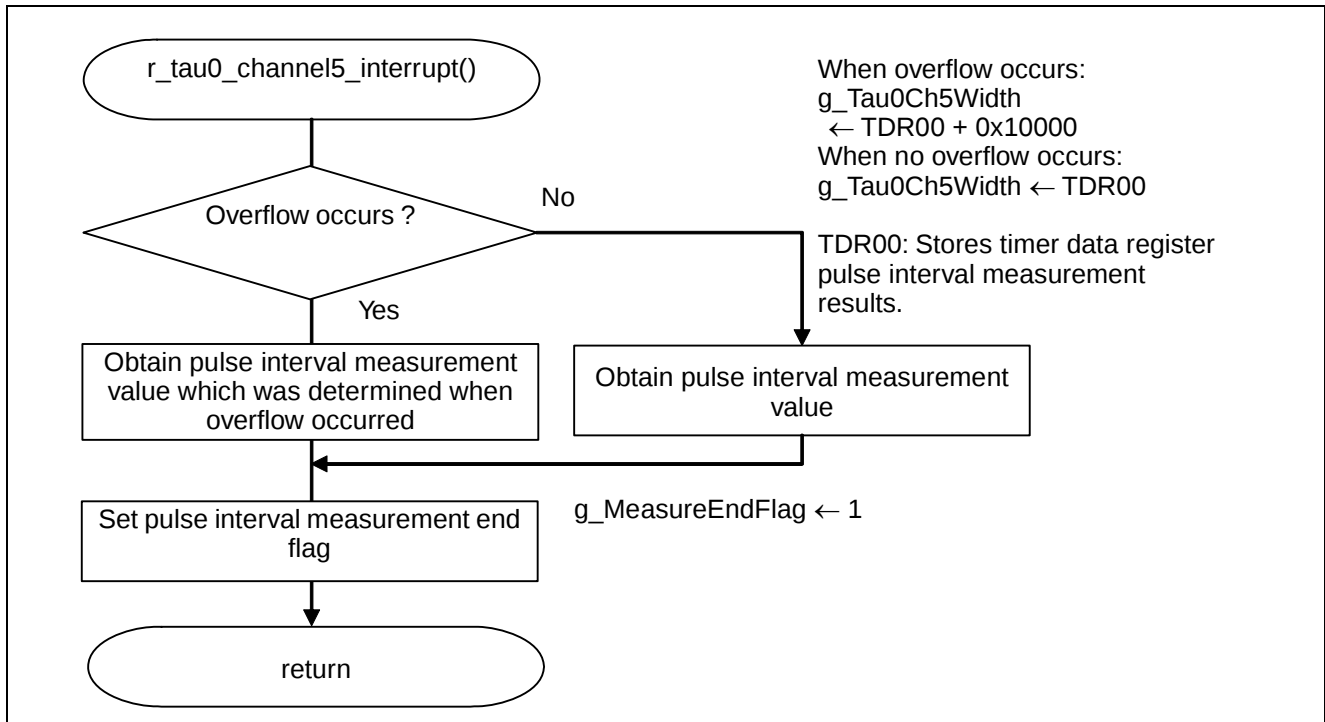


Figure 5.32 Capture End Interrupt for Channel 5 of TAU0

### 5.8.30 INTP0 External Interrupt

Figure 5.33 shows the flowchart for an INTP0 external interrupt.

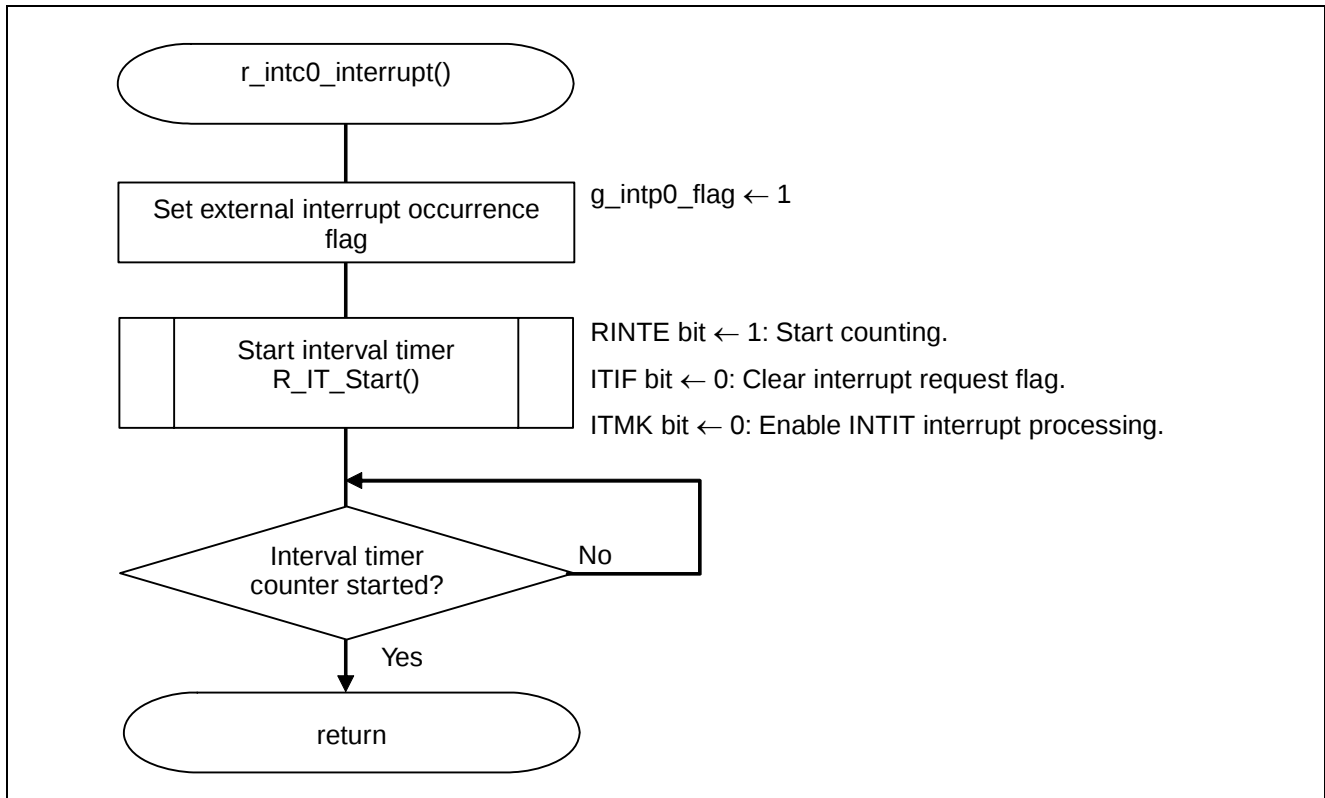
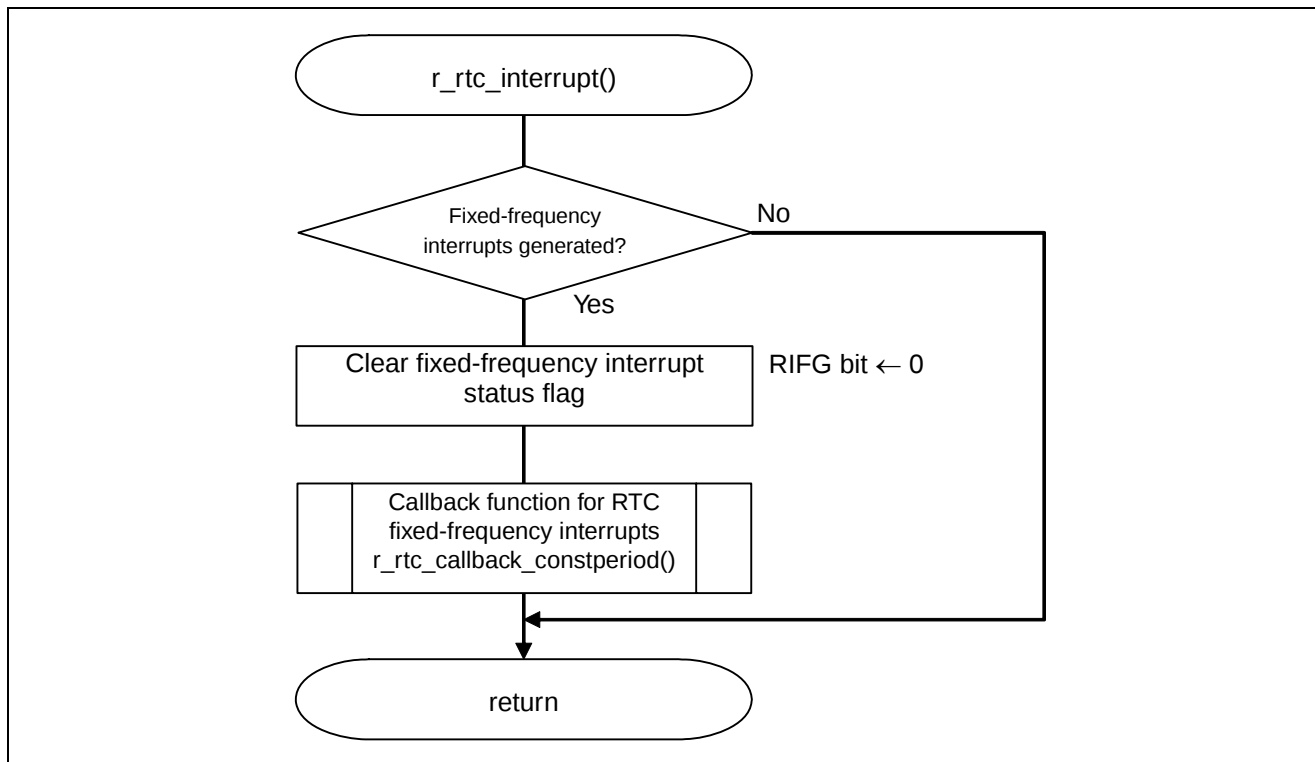


Figure 5.33 INTP0 External Interrupt

### 5.8.31 RTC Fixed-Frequency Interrupt

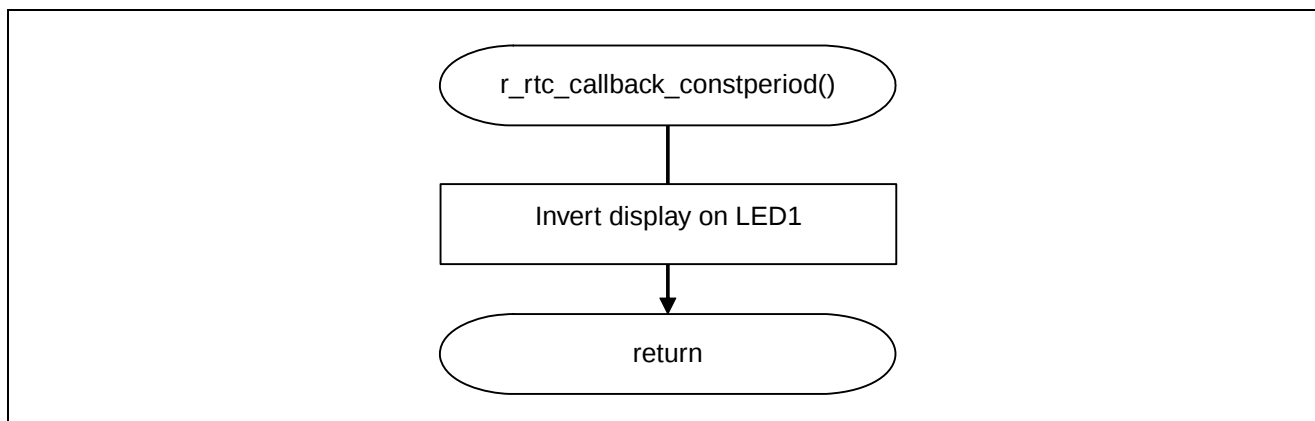
Figure 5.34 shows the flowchart for an RTC fixed-frequency interrupt.



**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 starting the interval timer.

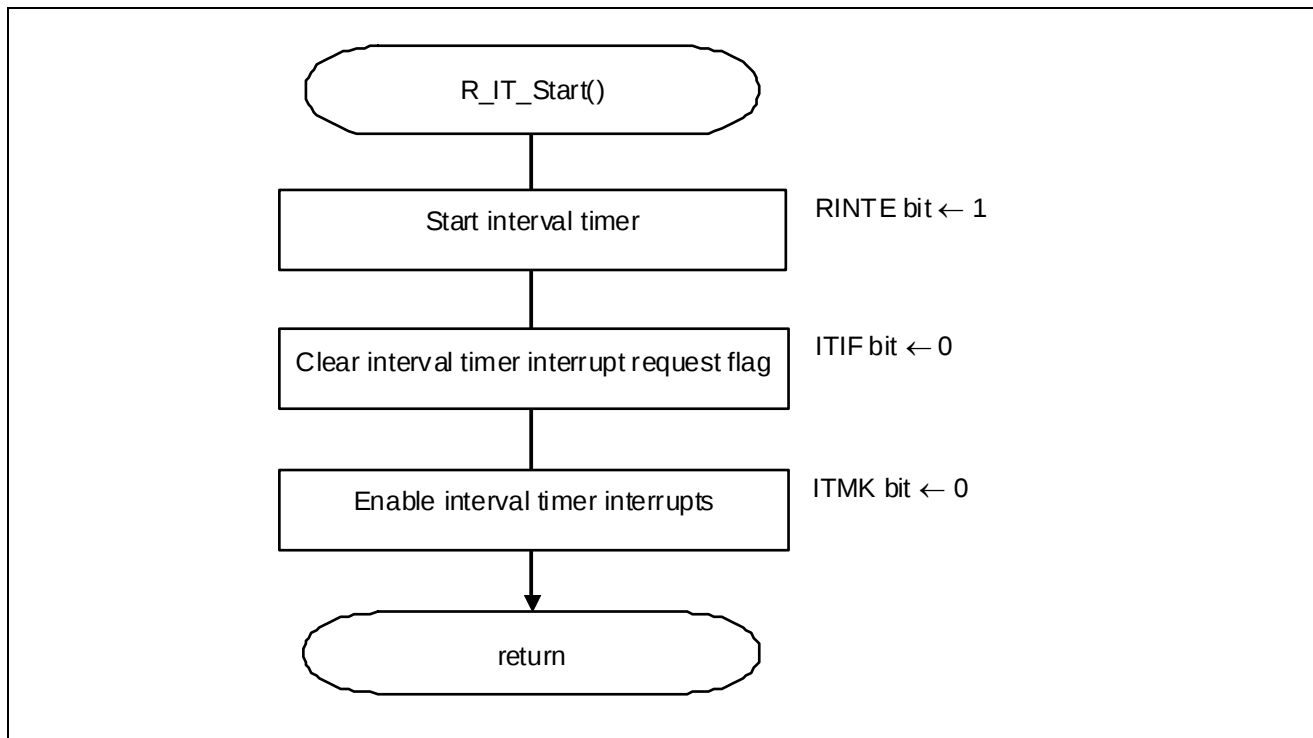


Figure 5.36 Interval Timer Start

### 5.8.34 Stopping Interval Timer

Figure 5.37 shows the flowchart for stopping the interval timer.

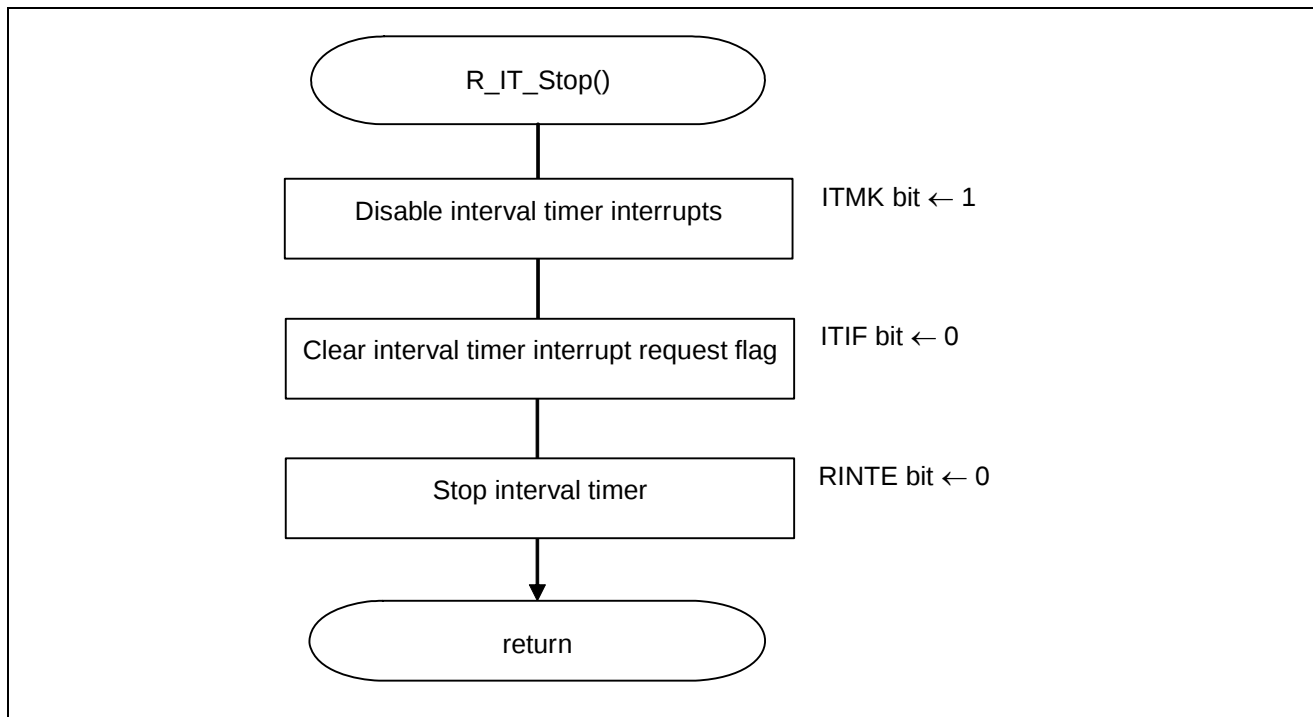
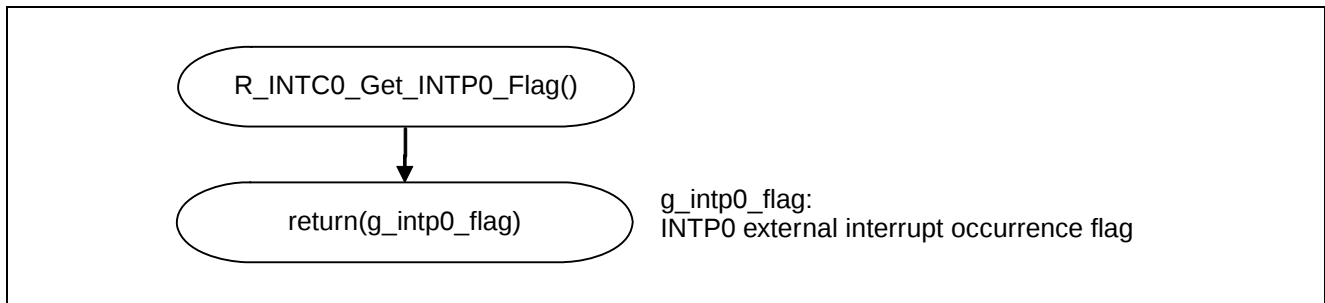


Figure 5.37 Stopping Interval Timer

### 5.8.35 Getting INTP0 External Interrupt Occurrence Flag

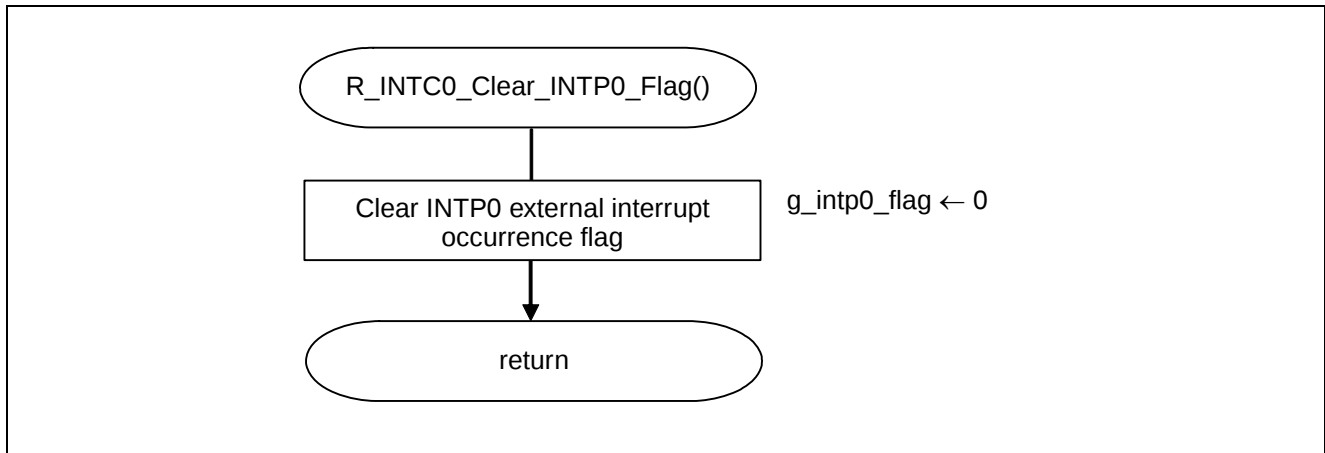
Figure 5.38 shows the flowchart for getting the INTP0 external interrupt occurrence flag. This function does nothing but returns global variable g\_intp0\_flag as a return value.



**Figure 5.38** Getting INTP0 External Interrupt Occurrence Flag

### 5.8.36 Clearing INTP0 External Interrupt Occurrence Flag

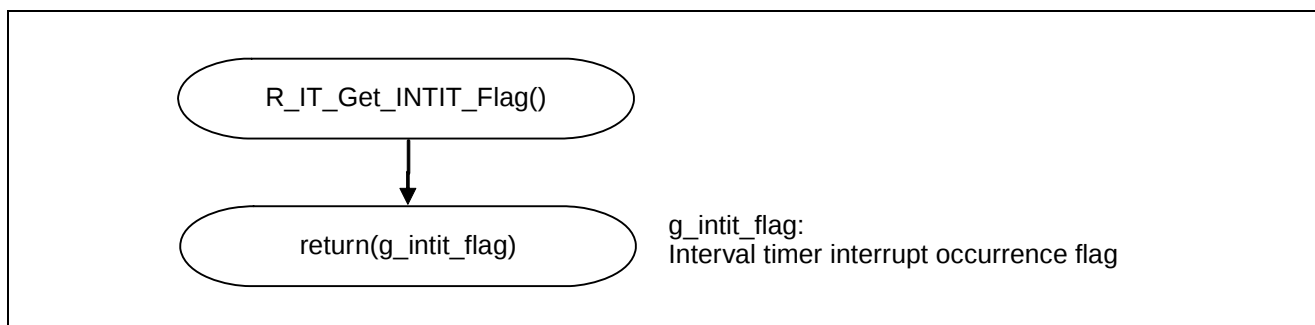
Figure 5.39 shows the flowchart for clearing the INTP0 external interrupt occurrence flag.



**Figure 5.39** Clearing INTP0 External Interrupt Occurrence Flag

### 5.8.37 Getting Interval Timer Interrupt Occurrence Flag

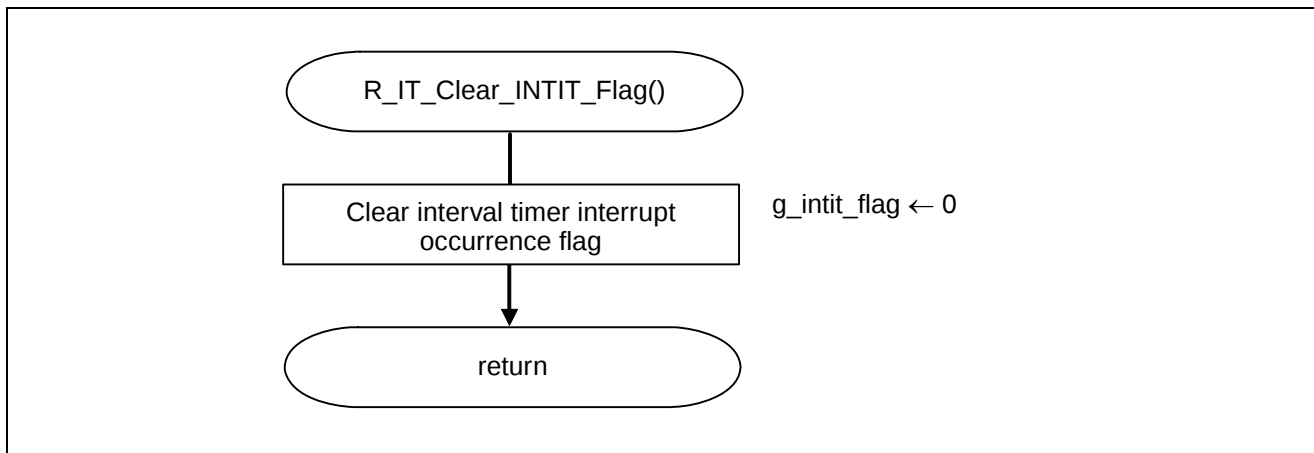
Figure 5.40 shows the flowchart for getting the interval timer interrupt occurrence flag. This function does nothing but returns global variable `g_intit_flag` as a return value.



**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 | Feb. 27, 2012 | —           | 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 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.



### 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