# RL78/G13

## Low-power Consumption Operation (CSI in SNOOZE Mode)

## CC-RL

R01AN2762EJ0100
Rev. 1.00
Apr.16.2015

## Introduction

This application note explains how to make low-power consumption settings for CSI slave reception in SNOOZE mode. The sample application covered in this application note uses the SNOOZE mode to receive data through the CSI communication without starting the CPU. It compares the receive data with a preset expected value and displays the result on an LED.

## 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

This application note explains how to make low-power consumption settings for CSI slave reception in SNOOZE mode. The sample application covered in this application note sets up the serial array unit (SAU) for CSI reception, enables the SNOOZE mode, then sets the BUSY signal to 0. Subsequently, it executes the STOP instruction. When an input is detected at the SCK pin in STOP mode, the application starts data reception in SNOOZE mode. It compares the data received via CSI with the predefined data. Then, it turns on an LED if a match is found and turns off the LED otherwise.

The BUSY signal is used to indicate the busy state of the slave device. “0” indicates that the device is ready for communication. “1” indicates that the device is not ready for communication. The master device checks that the other device is not busy (handshakes) before sending its data.

Caution: The SNOOZE mode can be enabled only when the high-speed on-chip oscillator clock is selected as the CPU/peripheral hardware clock ($f_{CLK}$).

Table 1.1 lists the peripheral functions to be used for low-power consumption operation and their uses. Figure 1.1 shows the outline of the operation.

**Table 1.1 Peripheral Functions to be Used and their Uses**

| Peripheral Function | Use |
|---|---|
| Channel 0 of serial array unit 0 | Used for slave mode reception via CSI00. |
| External interrupt input (INTP0) | Switch input and start of CSI00 reception. |
| Interval timer | Generates a wait state till the switch state is set. |
| Port I/O | • LED1 lighting control<br>Turns on the LED1 with a switch input in HALT mode.<br>• LED2 lighting control<br>Turns on the LED2 when the receive data matches the expected value.<br>• BUSY signal output<br>0: Ready for communication,<br>1: Not ready for communication |

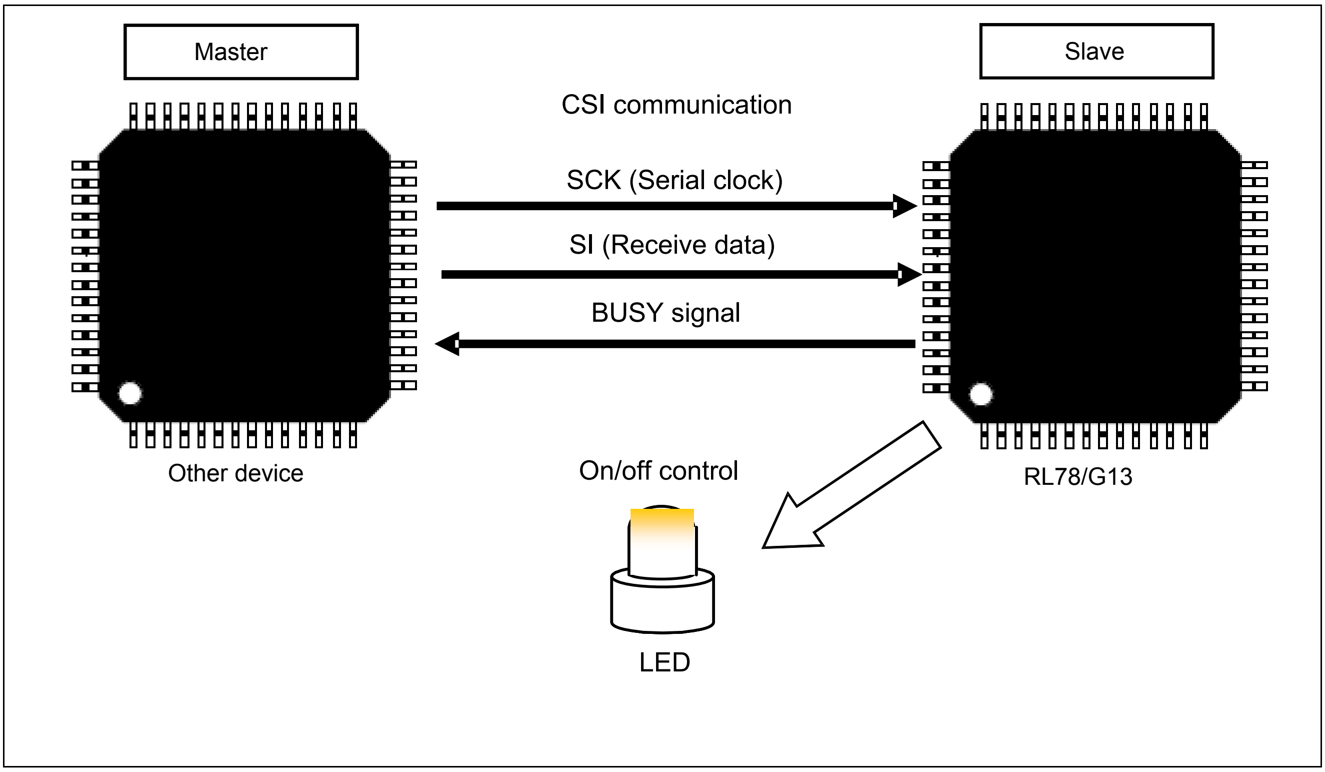
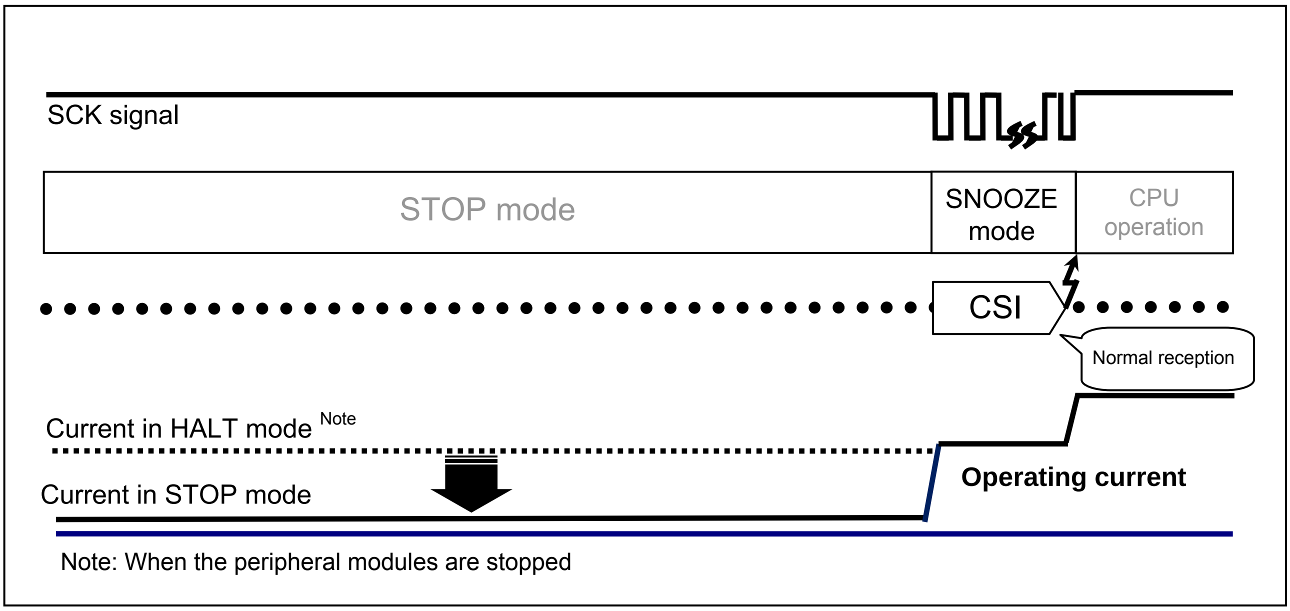


**Figure 1.1 Outline of the Operation**

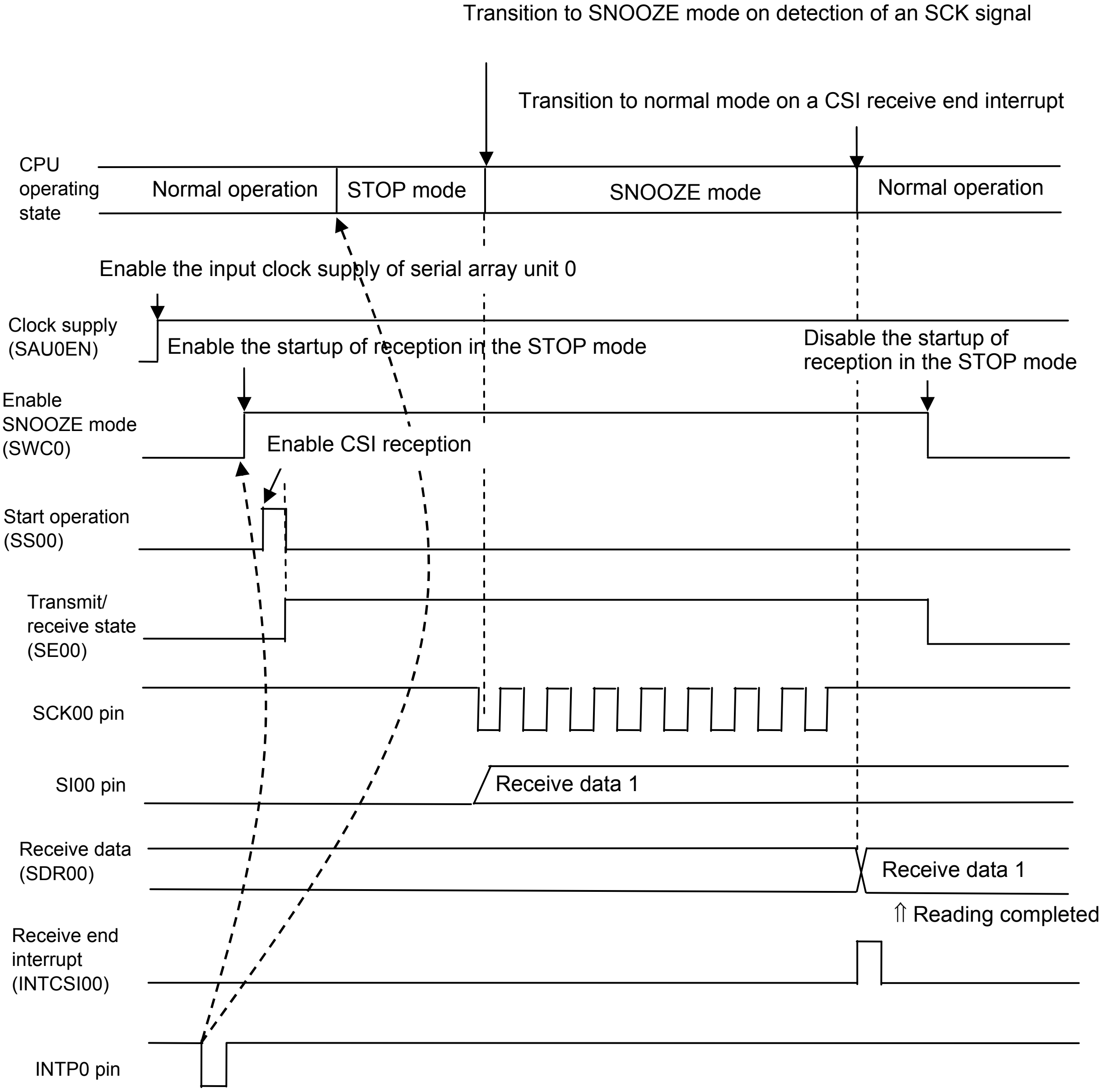
Figure 1.2 shows the outline of CSI slave reception in SNOOZE mode. The HALT mode is generally used to receive data via CSI. When a device which supports the SNOOZE mode is to be used, the STOP mode, which provides lower operating current than that in the HALT mode, is available in addition to the HALT mode. During CSI reception in STOP mode, the application switches from STOP mode to SNOOZE mode upon detection of the falling edge of the SCK signal and then receives data via CSI without starting the CPU. The SNOOZE mode can be set only when the high-speed on-chip oscillator clock is selected as the CPU/peripheral hardware clock ($f_{CLK}$).



Note: When the peripheral modules are stopped

**Figure 1.2 Outline of CSI Slave Reception in SNOOZE Mode**

Figure 1.3 shows the timing chart for the operation in the SNOOZE mode.


Transition to SNOOZE mode on detection of an SCK signal
Transition to normal mode on a CSI receive end interrupt
CPU operating state
Normal operation
STOP mode
SNOOZE mode
Normal operation
Enable the input clock supply of serial array unit 0
Clock supply (SAU0EN)
Enable the startup of reception in the STOP mode
Disable the startup of reception in the STOP mode
Enable SNOOZE mode (SWC0)
Enable CSI reception
Start operation (SS00)
Transmit/ receive state (SE00)
SCK00 pin
SI00 pin
Receive data 1
Receive data (SDR00)
Receive data 1
⇑ Reading completed
Receive end interrupt (INTCSI00)
INTP0 pin

Caution: Set the STm1 bit to 1 and clear the SEm1 bit (stop operation) when the reception in SNOOZE mode is completed. Also disable the startup of the reception in SNOOOZE mode by setting the SWCm bit to 0.

**Figure 1.3 Timing Chart of SNOOZE Mode**

## 2. Operation Check Conditions

The sample code described 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 | • High-speed on-chip oscillator (HOCO) clock: 32 MHz<br>• CPU/peripheral hardware module clock: 32 MHz |
| Operating voltage | 5.0V (Operation is possible at 2.9 V to 5.5 V.)<br>LVD operation ($V_{LVD}$): Reset mode 2.81 V (2.76 V to 2.87 V) |
| Integrated development environment (CS+) | CS+ V3.01.00 from Renesas Electronics Corp. |
| C compiler (CS+) | CC-RL V1.01.00 from Renesas Electronics Corp. |
| Integrated development environment ($e^2$ studio) | $e^2$ studio V4.0.0.26 from Renesas Electronics Corp. |
| C compiler ($e^2$ studio) | CC-RL V1.01.00 from Renesas Electronics Corp. |
| Board used | RL78/G13 target board (QB-R5F100LE-TB) |

## 3. Related Application Notes

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

RL78/G13 Initialization (R01AN2575E) Application Note

RL78/G13 Serial Array Unit for 3-Wire Serial I/O (Master Transmission/Reception) (R01AN2547E) Application Note

RL78/G13 Serial Array Unit for 3-Wire Serial I/O (Slave Transmission/Reception) (R01AN2711E) Application Note

# 4. Description of the Hardware

## 4.1 Hardware Configuration Example

Figure 4.1 shows an example of hardware configuration that is used for this application note.

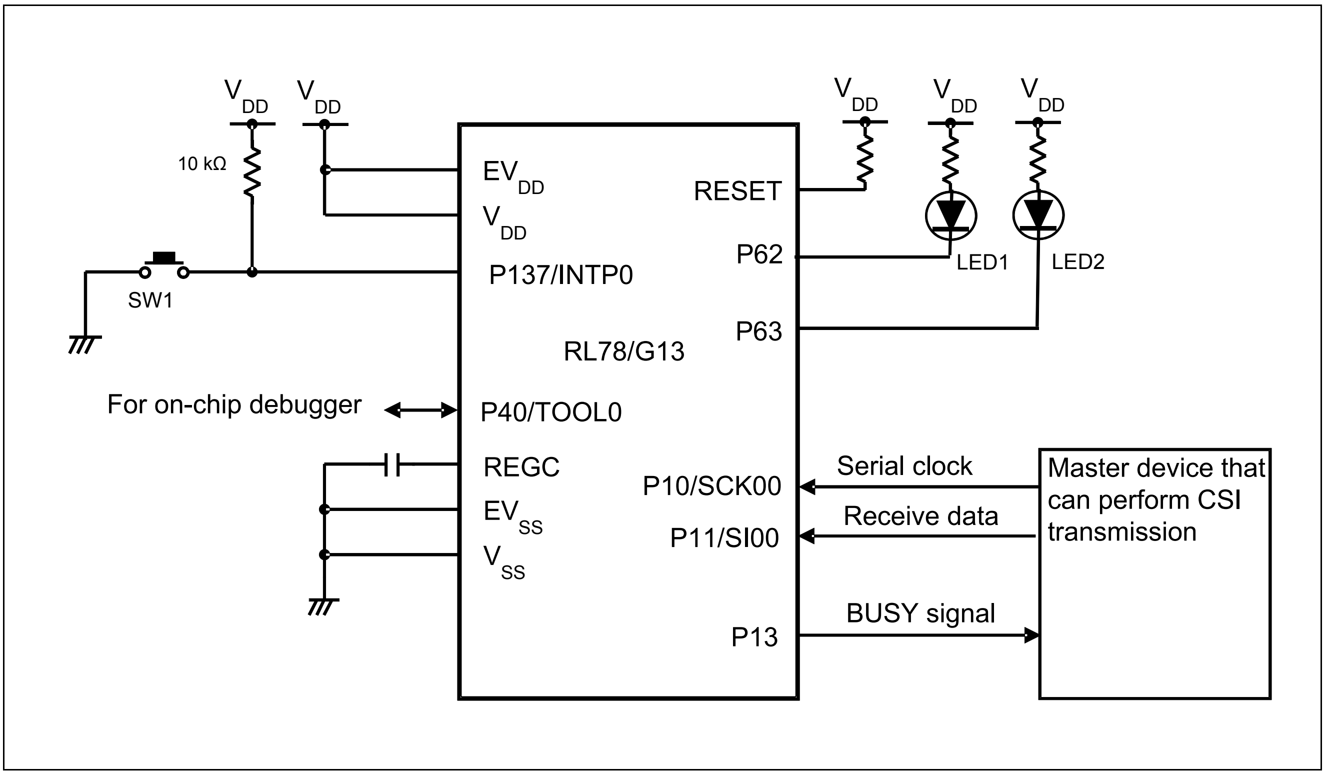


**Figure 4.1 Hardware Configuration**

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

## 4.2 List of Pins to be Used

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

**Table 4.1 Pins to be Used and their Functions**

| Pin Name | I/O | Description |
|---|---|---|
| P10/SCK00 | Input | CSI00 serial clock input pin |
| P11/SI00 | Input | CSI00 serial data receive pin |
| P13 | Output | BUSY signal output pin<br>0: Ready for communication<br>1: Not ready for communication |
| P137/INTP0 | Input | Switch input/CSI00 reception start trigger<br>Pressing the switch when LED1 is on causes the CPU to switch into STOP mode and starts CSI slave reception. |
| P62 | Output | LED1 lighting control port<br>Turns on when the CSI is in the switch input wait state in HALT mode. |
| P63 | Output | LED2 lighting control port<br>Turns on when the receive data matches the expected value. |

# 5. Software Description

## 5.1 Operation Outline

The sample application covered in this application note receives data via a CSI slave in SNOOZE mode, compares the receive data with the predefined data value, and turns on an LED when these data are matched.

The application turns on LED1, switches to HALT mode, and waits for a switch input. When a switch input is detected, the application turns off LED1 and LED2, sets CSI reception and SNOOZE mode, then transitions to the STOP mode.

The CPU is restored to the normal operation mode on a CSI slave receive end interrupt. After being restored into the normal operation mode, the application turns on LED2 if the receive data matches the expected value.

(1) Initialize the SAU0.
<Conditions for setting>

- Use channel 0 of SAU0 as the CSI.
- Set the serial clock to the input clock from the SCK00 pin (slave).
- Use single transfer mode as operation mode.
- Select type 1 as the phase between data and clock signals.
- Set the data transfer order to the MSB first.
- Set the length of data to 8 bits.
- Make sure that a serial transfer end interrupt (INTCSI00) should occur in single transfer mode.
- Set the priority level of the INTCSI00 interrupt to level 0.
- Use the P10/SCK00 pin for clock input.
- Use the P11/SI00 pin for data input.

(2) Setting up I/O ports
<Conditions for setting>

- LED lighting control port (LED1-LED2): Set ports P62 and P63 to output ports.
- CSI00 reception start switch: Set the P137/INTP0 pin for INTP0 interrupts in falling edge detection mode (using an external pull-up resistor).

(3) Take the following actions before switching into HALT mode (waiting for a switch input):

- Turn on LED1.
- Clear the INTP0 interrupt request flag.
- Enable the INTP0 interrupts.

(4) Transition to HALT mode and wait for a switch input.

(5) On detection of a switch input, the application exits the HALT mode and takes the following actions:

- To prevent chattering, take actions (A) through (E) below.
  - A) Using the interrupt handler for INTP0 interrupts, set the RINTE bit of the interval timer control register (ITMC) to 1 to start counting.
  - B) Wait until an interval timer interrupt occurs.
  - C) Check the switch status with the interval timer interrupt handler. That is, check the P137 status.
  - D) If this status is 1, determine that the switch has not been depressed. Then, return to step (4).
  - E) If it is 0, determine that the switch has been depressed. Perform the operations below.

- Disable the INTP0 interrupts.
- Clear the INTP0 interrupt request flag.
- Set the SWC0 bit in the serial standby control register 0 (SSC0) to 1 to enable reception to be started in STOP mode.
- Clear the INTCSI00 interrupt request flag.
- Enable the INTCSI00 interrupts.
- Set the SS00 bit in the serial channel start register 0 (SS0) to 1 to transition to the communication wait state.
- Turn off LED1 and LED2.
- Set the BUSY signal to 0.

(6) Transition to STOP mode and wait for SCK input.
In STOP mode, CSI reception is performed by hardware. When a serial clock is detected (SCK00 pin input), the CPU transitions to SNOOZE mode and starts CSI reception. When the reception ends successfully, the CPU exits the SNOOZE mode and enters the normal operation mode.

(7) After the transition to normal operation mode, the application reads the receive data.

- Set the BUSY signal to 1 after reading the receive data.

(8) Stop the CSI operation and disable transition to SNOOZE mode.

- Set the ST00 bit in the serial channel stop register (ST0) to 1 to stop communication.
- Disable the INTCSI00 interrupts.
- Clear the INTCSI00 interrupt request flag.
- Set the SWC0 bit in the serial standby control register 0 (SSC0) to 0 to disable reception in STOP mode.

(9) Check the receive data and turn on LED2 if it matches the expected value. Subsequently, repeat steps (3) through (9).

Caution: For information about the precautions in using the device, refer to RL78/G13 User’s Manual: Hardware.

## 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_cg_serial_user.c | SAU module<br>CSI00 receive end interrupt | Additional functions:<br>R_CSI00_Get_ReceiveEndFlag<br>R_CSI00_Clear_ReceiveEndFlag |
| r_cg_it_user.c | Interval timer module<br>Interval timer interrupt | Additional functions:<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 the Sample Program**

| Constant | Setting | Description |
|---|---|---|
| _0001_SAU_CH0_START_TRG_ON | 0x0001 | Serial channel enable status register 0 (SE0) value for enabling communication |
| _0001_SAU_CH0_STOP_TRG_ON | 0x0001 | Serial channel stop register 0 (ST0) value for stopping communication |
| _0001_SAU_CH0_SNOOZE_ON | 0x0001 | Serial standby control register 0 (SSC0) value for enabling reception in STOP mode |
| LED1_NOMAL | P6_bit.no2 | LED1 lighting control port |
| LED2_RCV_OK | P6_bit.no3 | LED2 lighting control port |
| BUSY_SIGNAL | P1_bit.no3 | BUSY signal output port |
| RECEIVE_OK_DATA | 0x5B | Expected value of receive data |

## 5.5 List of Variables

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

**Table 5.4 Global Variable**

| Type | Variable Name | Contents | Function Used |
|---|---|---|---|
| volatile uint16_t | g_csi00_rx_count | Number of CSI data bytes received | R_CSI00_Receive<br>r_csi00_interrupt |
| volatile uint16_t | g_csi00_rx_length | Number of CSI data bytes to receive | R_CSI00_Receive<br>r_csi00_interrupt |
| volatile uint8_t* | gp_csi00_rx_address | Address of location for storing the next CSI receive data | R_CSI00_Receive<br>r_csi00_interrupt |

**Table 5.5 Static Variables**

| Type | Variable Name | Contents | Function Used |
|---|---|---|---|
| uint8_t | g_ReceiveEndFlag | CSI00 receive end flag | r_csi00_callback_receiveend, R_CSI00_Get_ReceiveEndFlag, R_CSI00_Clear_ ReceiveEndFlag |
| uint8_t | g_sw_status | Switch state | r_it_interrupt, R_IT_Get_Switch_Status, R_IT_Clear_Switch_Status |
| uint8_t | g_intit_flag | Interval timer interrupt occurrence flag | r_it_interrupt, R_IT_Get_INTIT_Flag, R_IT_Clear_INTIT_Flag |

## 5.6 List of Functions

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

**Table 5.6 Functions**

| Function Name | Outline |
|---|---|
| R_SAU0_Set_SnoozeOn | Enables CSI00 in SNOOZE mode. |
| R_SAU0_Set_SnoozeOff | Disables CSI00 in SNOOZE mode. |
| R_CSI00_Start | Starts CSI00 reception. |
| R_CSI00_Stop | Stops CSI00 reception. |
| R_CSI00_Receive | Sets up CSI00 receive buffer. |
| r_csi00_interrupt | Processes CSI00 communication end interrupt. |
| r_csi00_callback_receiveend | Callback function for CSI00 communication end interrupts |
| R_CSI00_Get_ReceiveEndFlag | Gets CSI00 receive end flag. |
| R_CSI00_Clear_ReceiveEndFlag | Clears CSI00 receive end flag. |
| R_INTC0_Start | Starts INTP0 operation. |
| R_INTC0_Stop | Stops INTP0 operation. |
| r_intc0_interrupt | INTP0 interrupt |
| R_IT_Start | Starts the interval timer. |
| R_IT_Stop | Stops the interval timer operation. |
| r_it_interrupt | Interval timer interrupt |
| R_IT_Get_Switch_Status | Gets switch state. |
| R_IT_Clear_Swtich_Status | Clears switch state. |
| R_IT_Get_INTIT_Flag | Obtains 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 this sample program.

[Function Name] R_SAU0_Set_SnoozeOn

| | |
|---|---|
| Synopsis | Enables CSI00 in SNOOZE mode. |
| Header | r_cg_macrodriver.h<br>r_cg_serial.h<br>r_cg_userdefine.h |
| Declaration | void R_SAU0_SnoozeOn(void) |
| Explanation | Enables operation in SNOOZE mode (SWC0 = 1). |
| Arguments | None |
| Return value | None |
| Remarks | None |

[Function Name] R_SAU0_Set_SnoozeOff

| | |
|---|---|
| Synopsis | Disables CSI00 in SNOOZE mode. |
| Header | r_cg_macrodriver.h<br>r_cg_serial.h<br>r_cg_userdefine.h |
| Declaration | void R_SAU0_SnoozeOff(void) |
| Explanation | Disables operation in SNOOZE mode (SWC0 = 0). |
| Arguments | None |
| Return value | None |
| Remarks | None |

[ Function Name ] R_CSI00_Start

| | |
|---|---|
| Synopsis | Starts CSI00 reception. |
| Header | r_cg_macrodriver.h<br>r_cg_serial.h<br>r_cg_userdefine.h |
| Declaration | void R_CSI00_Start(void) |
| Explanation | Masks off CSI00 receive interrupts to enable CSI00 reception. |
| Arguments | None |
| Return value | None |
| Remarks | None |

[Function Name] R_CSI00_Stop

| | |
|---|---|
| Synopsis | Stops CSI00 reception. |
| Header | r_cg_macrodriver.h<br>r_cg_serial.h<br>r_cg_userdefine.h |
| Declaration | void R_CSI00_Stop(void) |
| Explanation | Masks on CSI00 receive interrupts to disable CSI00 reception. |
| Arguments | None |
| Return value | None |
| Remarks | None |

### [Function Name] R_CSI00_Receive

| | | |
|---|---|---|
| Synopsis | Sets up CSI00 receive buffer. | |
| Header | r_cg_macrodriver.h<br>r_cg_serial.h<br>r_cg_userdefine.h | |
| Declaration | MD_STATUS R_CSI00_Receive(uint8_t * const rx_buf, uint16_t rx_num) | |
| Explanation | Sets up the CSI00 receive buffer address and the number of data bytes to receive.<br>An argument error is returned if the receive data count is set to not more than 1. | |
| Arguments | rx_buf | Address of receive data buffer |
| | rx_num | Number of data bytes to receive |
| Return value | [MD_OK]: Setup completed<br>[MD_ARGERROR]: Argument error | |
| Remarks | None | |

### [Function Name] r_csi00_interrupt

| | |
|---|---|
| Synopsis | Processes CSI00 communication end interrupt. |
| Header | r_cg_macrodriver.h<br>r_cg_serial.h<br>r_cg_userdefine.h |
| Declaration | static void __near r_csi00_interrupt(void) |
| Explanation | Reads the receive data.<br>The function calls the callback function for CSI00 communication end interrupts. |
| Arguments | None |
| Return value | None |
| Remarks | None |

### [Function Name] r_csi00_callback_receiveend

| | |
|---|---|
| Synopsis | Callback function for CSI00 communication end interrupts |
| Header | r_cg_macrodriver.h<br>r_cg_serial.h<br>r_cg_userdefine.h |
| Declaration | static void r_csi00_callback_receiveend(void) |
| Explanation | Sets the CSI00 receive end flag (g_ReceiveEndFlag) to 1. |
| Arguments | None |
| Return value | None |
| Remarks | None |

### [Function Name] R_CSI00_Get_ReceiveEndFlag

| | |
|---|---|
| Synopsis | Gets receive end flag. |
| Header | r_cg_macrodriver.h<br>r_cg_serial.h<br>r_cg_userdefine.h |
| Declaration | uint8_t R_CSI00_Get_ReceiveEndFlag(void) |
| Explanation | Gets the CSI00 receive end flag (g_ReceiveEndFlag). |
| Arguments | None |
| Return value | • When reception is not ended: 0x00<br>• When reception is ended: 0x01 |
| Remarks | None |

[Function Name] R_CSI00_Clear_ReceiveEndFlag

| | |
|---|---|
| Synopsis | Clears receive end flag. |
| Header | r_cg_macrodriver.h<br>r_cg_serial.h<br>r_cg_userdefine.h |
| Declaration | void R_CSI00_Clear_ReceiveEndFlag(void) |
| Explanation | Clears the CSI receive end flag (g_ReceiveEndFlag) to 0. |
| Arguments | None |
| Return value | None |
| Remarks | None |

[Function Name] R_INTC0_Start

| | |
|---|---|
| Synopsis | Resets INTP0 interrupt mask. |
| Header | r_cg_macrodriver.h<br>r_cg_intc.h<br>r_cg_userdefine.h |
| Declaration | void R_INTC0_Start(void) |
| Explanation | Masks off INTP0 interrupts. |
| Arguments | None |
| Return value | None |
| Remarks | None |

[Function Name] R_INTC0_Stop

| | |
|---|---|
| Synopsis | Sets INTP0 interrupt mask. |
| Header | r_cg_macrodriver.h<br>r_cg_intc.h<br>r_cg_userdefine.h |
| Declaration | void R_INTC0_Stop(void) |
| Explanation | Masks on INTP0 interrupts. |
| Arguments | None |
| Return value | None |
| Remarks | None |

[Function Name] r_intc0_interrupt

| | |
|---|---|
| Synopsis | Processes INTP0 interrupt. |
| Header | r_cg_macrodriver.h<br>r_cg_intc.h<br>r_cg_it.h<br>r_cg_userdefine.h |
| Declaration | static void __near r_intc0_interrupt(void) |
| Explanation | Starts the interval timer. |
| 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 masks off 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 on interval timer interrupts and stops the interval timer. |
| 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 | static void __near r_it_interrupt(void) |
| Explanation | Sets the switch state (g_sw_status) to 1 if P137 carries a 0.<br>It sets the interval timer interrupt occurrence flag (g_intit_flag) to 1. |
| Arguments | None |
| Return value | None |
| Remarks | None |

[Function Name] R_IT_Get_Switch_Status

| | |
|---|---|
| Synopsis | Gets switch state. |
| Header | r_cg_macrodriver.h<br>r_cg_it.h<br>r_cg_userdefine.h |
| Declaration | uint8_t R_IT_Get_Switch_Status (void) |
| Explanation | Gets the switch state (g_sw_status). |
| Arguments | None |
| Return value | • When the switch is not pressed: 0x00<br>• When the switch is pressed: 0x01 |
| Remarks | None |

[Function Name] R_IT_Clear_Swtich_Status

| | |
|---|---|
| Synopsis | Clears switch state. |
| Header | r_cg_macrodriver.h<br>r_cg_it.h<br>r_cg_userdefine.h |
| Declaration | void R_IT_Clear_Switch_Status(void) |
| Explanation | Clears the switch state (g_sw_status) to 0. |
| Arguments | None |
| Return value | None |
| Remarks | None |

[Function Name] R_IT_Get_INTIT_Flag

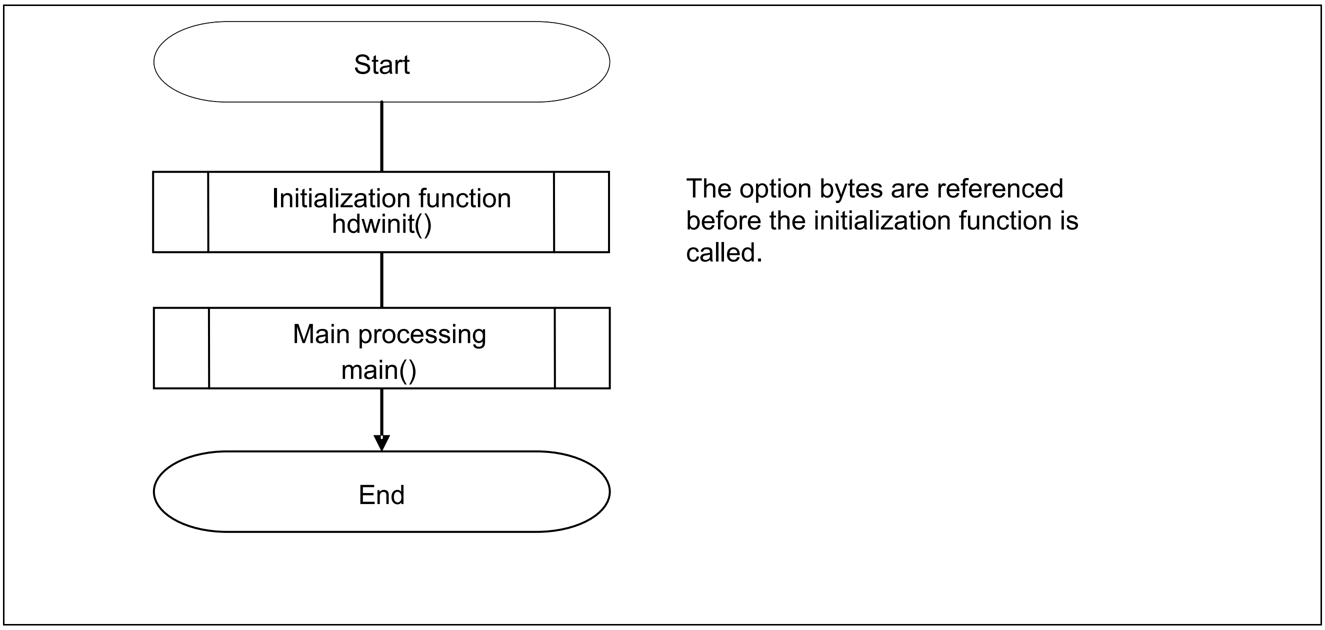
| | |
|---|---|
| Synopsis | Obtains the interval timer interrupt occurrence flag. |
| Header | r_cg_macrodriver.h<br>r_cg_it.h<br>r_cg_userdefine.h |
| Declaration | uint8_t R_IT_Get_INTIT_Flag(void) |
| Explanation | This function obtains the interval timer interrupt occurrence flag (g_intit_flag). |
| Arguments | None |
| Return value | • When an interval timer interrupt has not occurred: 0x00<br>• When an interval timer interrupt has occurred: 0x01 |
| Remarks | None |

[Function Name] R_IT_Clear_INTIT_Flag

| | |
|---|---|
| Synopsis | Clears the interval timer interrupt occurrence flag. |
| Header | r_cg_macrodriver.h<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. (g_intit_flag) to 0. |
| 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.



The option bytes are referenced before the initialization function is called.

**Figure 5.1 Overall Flow**

### 5.8.1 Initialization Function

Figure 5.2 shows the flowchart for the initialization function.

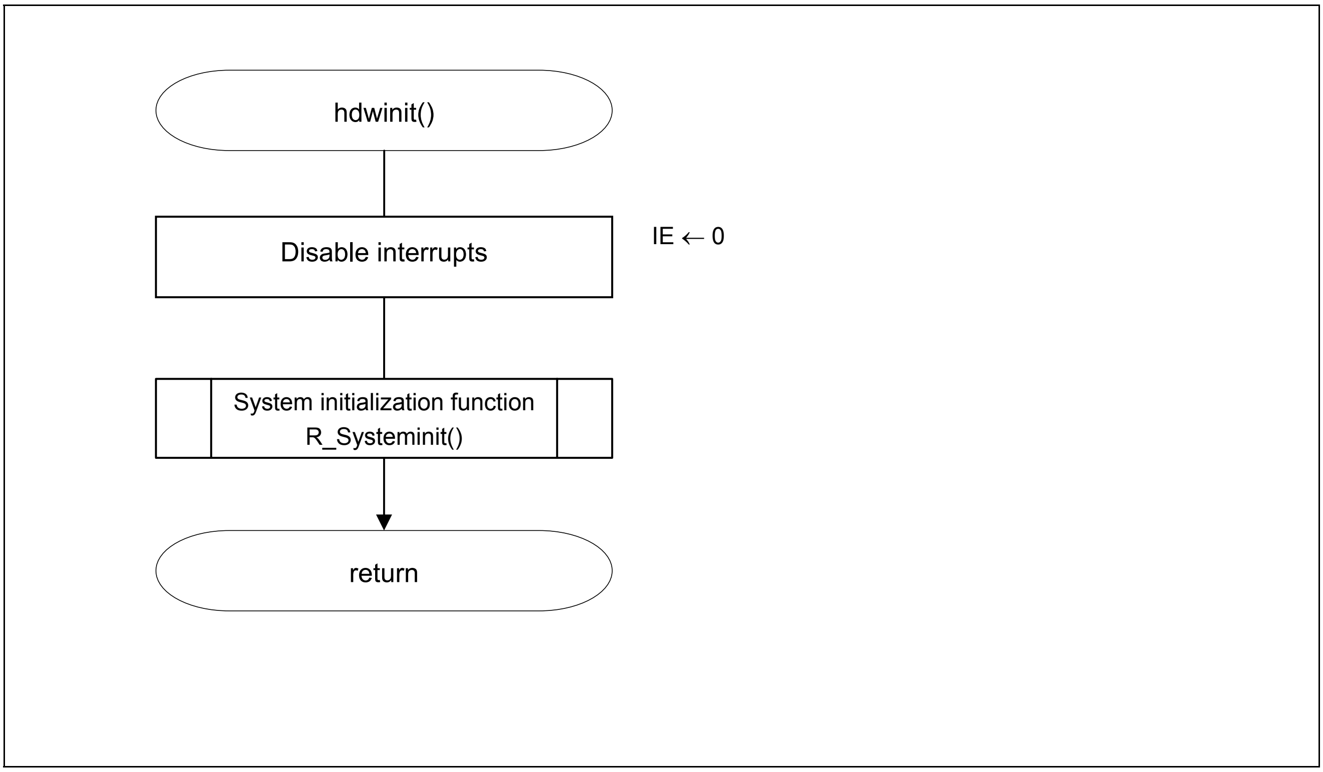


**Figure 5.2 Initialization Function**

### 5.8.2 System Initialization function

Figure 5.3 shows the flowchart for the system initialization function.

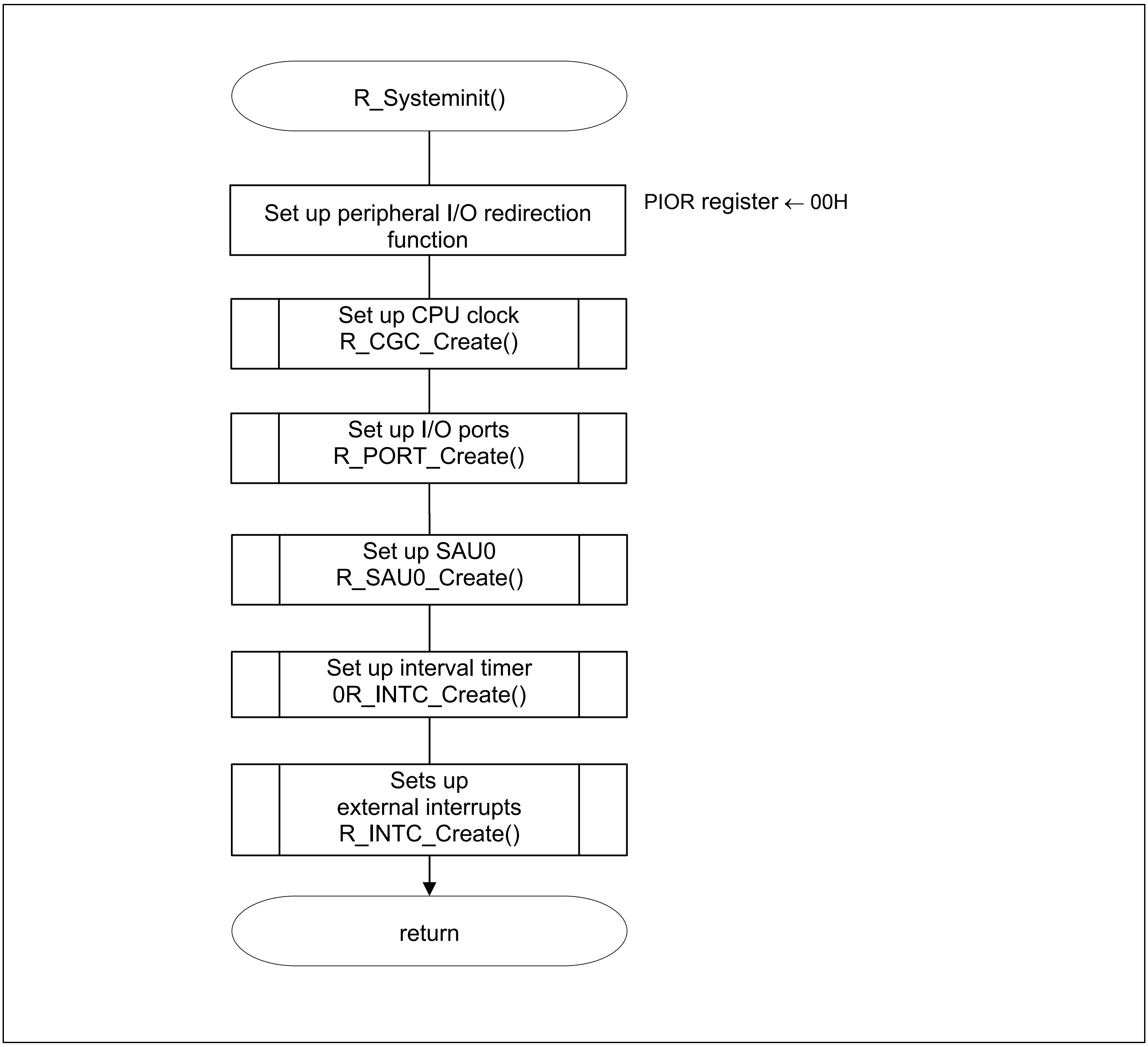


**Figure 5.3 System Initialization Function**

### 5.8.3 I/O Port Setup

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

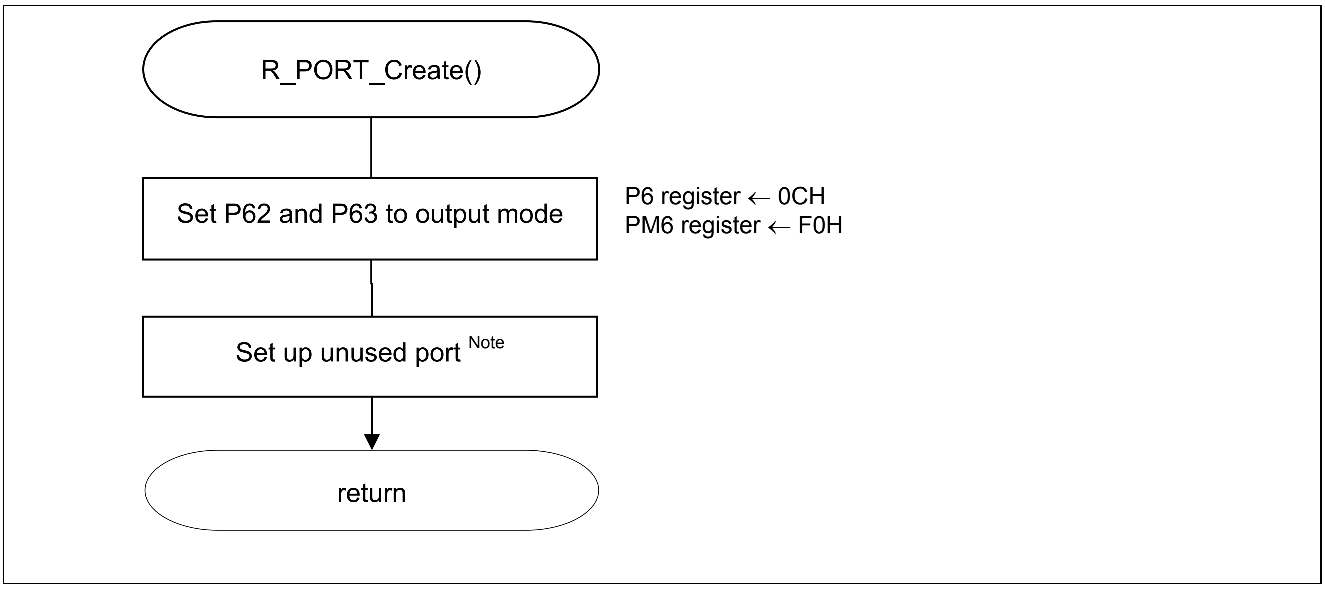


**Figure 5.4 I/O Port Setup**

Note: Refer to the section entitled "Flowcharts" in RL78/G13 Initialization Application Note (R01AN2575E) for the configuration of the unused ports.

Caution: 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.

Make settings for turning off LEDs

- Port register 0 (P6)
- Port mode register 6 (PM6)
  Select I/O mode and output latch of LED lighting control port

Symbol: P6

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | P63 | P62 | P61 | P60 |
| x | x | x | x | **1** | **1** | x | x |

Bits 3 and 2

| P62, P63 | P62 and P63 output selection |
|---|---|
| 0 | Output low level (LED on) |
| **1** | **Output high level (LED off)** |

Symbol: PM6

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | PM03 | PM02 | PM01 | PM00 |
| x | x | x | x | **0** | **0** | x | x |

Bits 3 and 2

| PM62,<br>PM63 | PM62 and PM63 I/O mode selection |
|---|---|
| **0** | **Output mode (output buffer on)** |
| 1 | Input mode (output buffer off) |

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

### 5.8.4 CPU Clock Setup

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

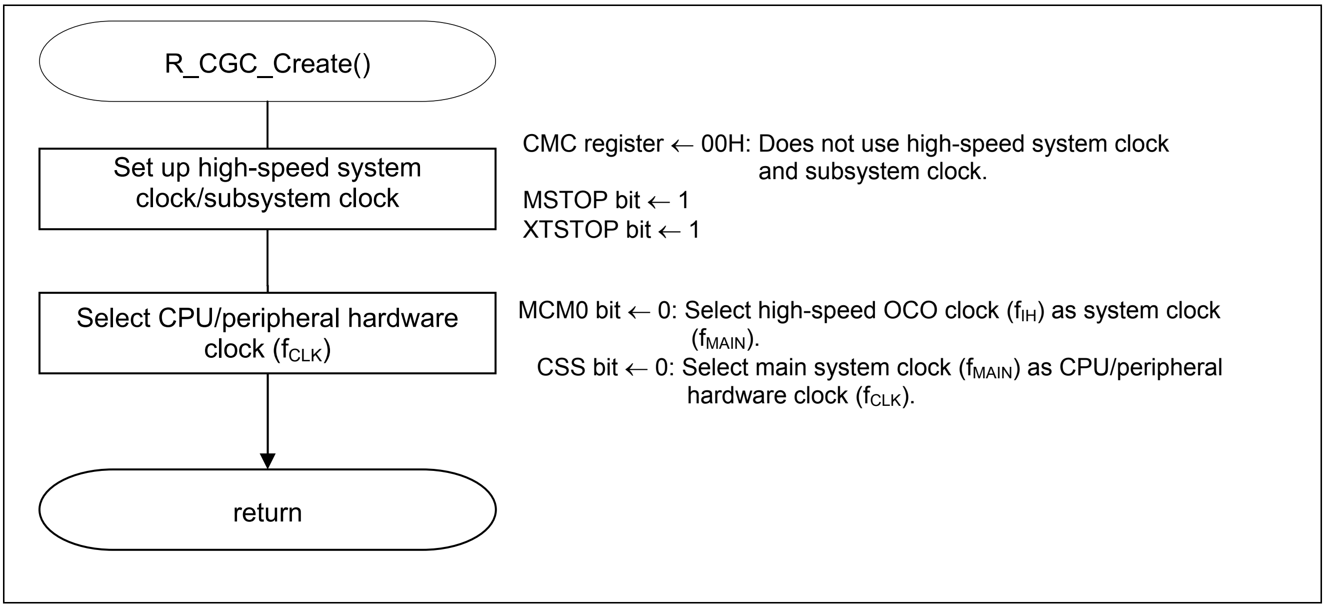

R_CGC_Create()
Set up high-speed system clock/subsystem clock
CMC register ← 00H: Does not use high-speed system clock and subsystem clock.
MSTOP bit ← 1
XTSTOP bit ← 1
Select CPU/peripheral hardware clock (fCLK)
MCM0 bit ← 0: Select high-speed OCO clock (fIH) as system clock (fMAIN).
CSS bit ← 0: Select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

**Figure 5.5 CPU Clock Setup**

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

### 5.8.5 SAU0 Setup

Figure 5.6 shows the flowchart for setting up the SAU0.

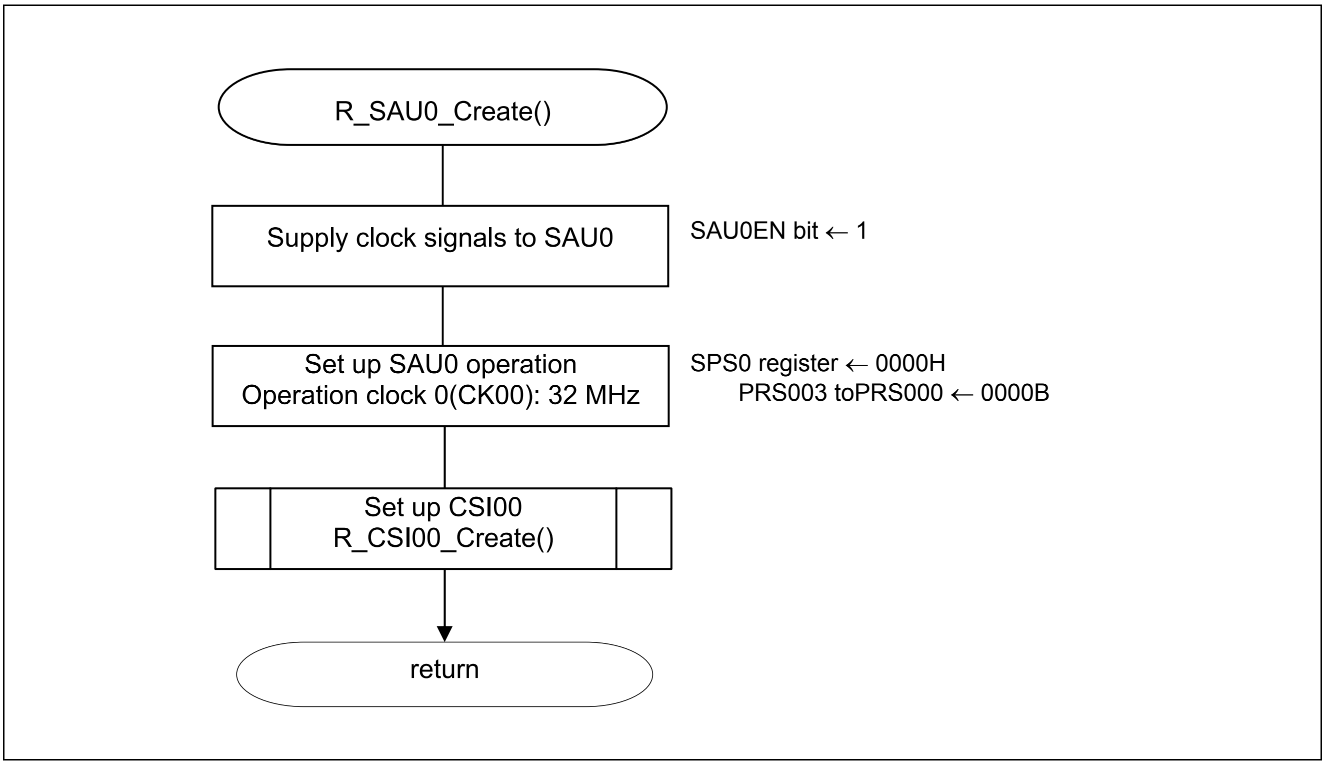


**Figure 5.6 SAU0 Setup**

Start supplying clock to the Serial Array Unit0

- Peripheral enable register 0 (PER0)
  Start supplying clock signals to Serial Array Unit 0.

Symbol: PER0

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

Bit 2

| SAU0EN | Control of serial array unit 0 input clock supply |
|---|---|
| 0 | Stops supply of input clock. |
| **1** | **Enables input clock supply.** |

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

Make settings for serial clock

- Serial clock select register (SPS0)
  Operation clock (CK00 = 32 MHz).

Symbol: SPS0

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

Bits 3 to 0

| PRS003 | PRS002 | PRS001 | PRS000 | Selection of operation clock (CK00) | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | $f_{CLK}$ = 2 MHz | $f_{CLK}$ = 5 MHz | $f_{CLK}$ = 10 MHz | $f_{CLK}$ = 20 MHz | **$f_{CLK}$ = 32 MHz** |
| **0** | **0** | **0** | **0** | **$f_{CLK}$** | 2 MHz | 5 MHz | 10 MHz | 20 MHz | **32 MHz** |
| 0 | 0 | 0 | 1 | $f_{CLK}$/2 | 1 MHz | 2.5 MHz | 5 MHz | 10 MHz | 16 MHz |
| 0 | 0 | 1 | 0 | $f_{CLK}$/22 | 500 kHz | 1.25 MHz | 2.5 MHz | 5 MHz | 8 MHz |
| 0 | 0 | 1 | 1 | $f_{CLK}$/23 | 250 kHz | 625 kHz | 1.25 MHz | 2.5 MHz | 4 MHz |
| 0 | 1 | 0 | 0 | $f_{CLK}$/24 | 125 kHz | 313 kHz | 625 kHz | 1.25 MHz | 2 MHz |
| 0 | 1 | 0 | 1 | $f_{CLK}$/25 | 62.5 kHz | 156 kHz | 313 kHz | 625 kHz | 1 MHz |
| 0 | 1 | 1 | 0 | $f_{CLK}$/26 | 31.3 kHz | 78.1 kHz | 156 kHz | 313 kHz | 500 kHz |
| 0 | 1 | 1 | 1 | $f_{CLK}$/27 | 15.6 kHz | 39.1 kHz | 78.1 kHz | 156 kHz | 250 kHz |
| 1 | 0 | 0 | 0 | $f_{CLK}$/28 | 7.81 kHz | 19.5 kHz | 39.1 kHz | 78.1 kHz | 125 kHz |
| 1 | 0 | 0 | 1 | $f_{CLK}$/29 | 3.91 kHz | 9.77 kHz | 19.5 kHz | 39.1 kHz | 62.5 kHz |
| 1 | 0 | 1 | 0 | $f_{CLK}$/210 | 1.95 kHz | 4.88 kHz | 9.77 kHz | 19.5 kHz | 31.3 kHz |
| 1 | 0 | 1 | 1 | $f_{CLK}$/211 | 977 Hz | 2.44 kHz | 4.88 kHz | 9.77 kHz | 15.6 kHz |
| 1 | 1 | 0 | 0 | $f_{CLK}$/212 | 488 Hz | 1.22 kHz | 2.44 kHz | 4.88 kHz | 7.81 kHz |
| 1 | 1 | 0 | 1 | $f_{CLK}$/213 | 244 Hz | 610 Hz | 1.22 kHz | 2.44 kHz | 3.91 kHz |
| 1 | 1 | 1 | 0 | $f_{CLK}$/214 | 122 Hz | 305 Hz | 610 Hz | 1.22 kHz | 1.95 kHz |
| 1 | 1 | 1 | 1 | $f_{CLK}$/215 | 61 Hz | 153 Hz | 305 Hz | 610 Hz | 977 Hz |

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

### 5.8.6 CSI00 Initialization

Figure 5.7 shows the flowchart for initializing the CSI00.

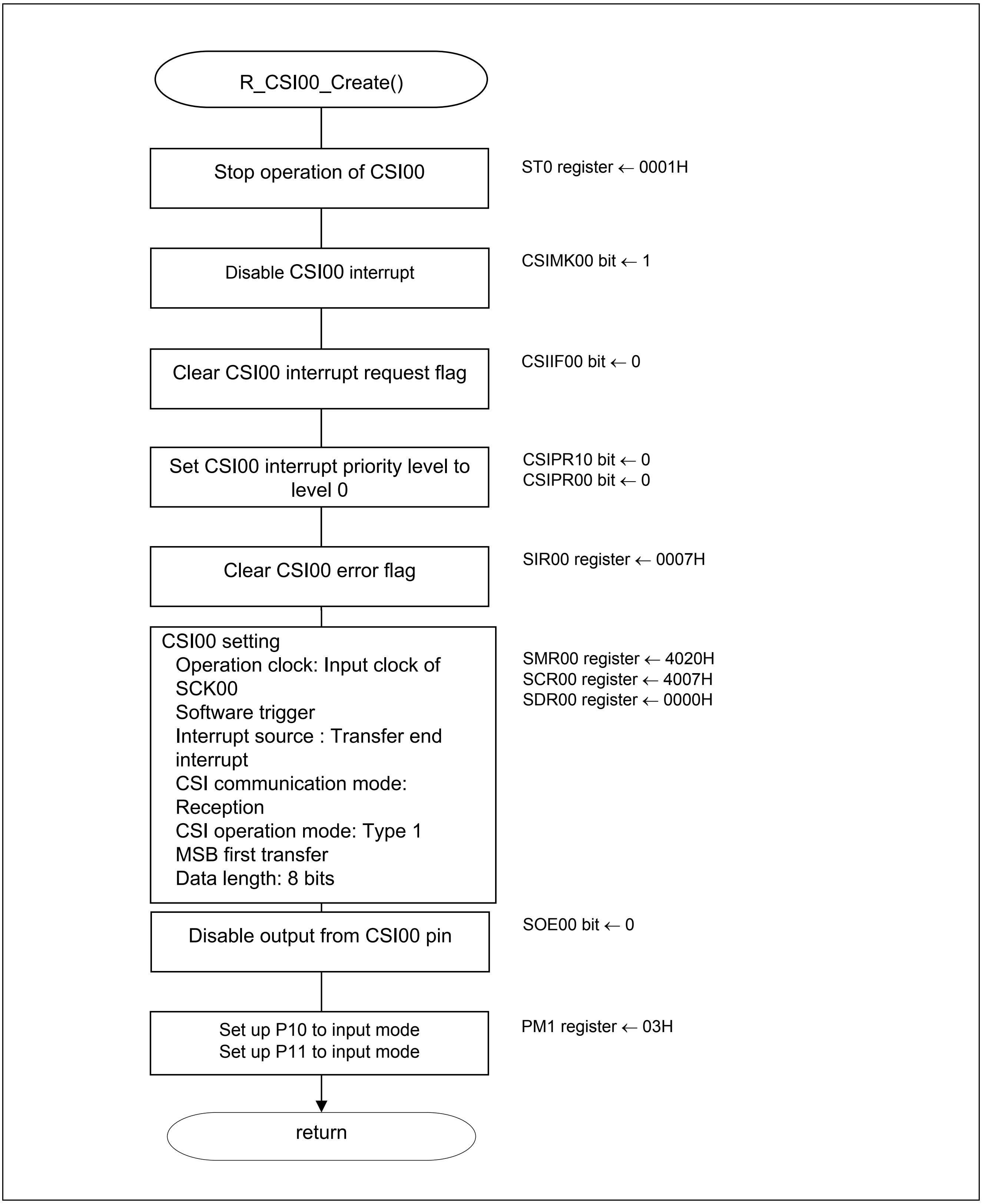


**Figure 5.7 CSI00 Initialization**

Deactivating serial channel 0

- Serial channel stop register 0 (ST0)
  Stop serial channel 0 communication operation.

Symbol: ST0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ST03 | ST02 | ST01 | ST00 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **0** | **0** | **0** | **1** |

Bits 3 to 0

| ST00 | Operation stop trigger of channel |
|---|---|
| 0 | Disables triggering. |
| **1** | **Clears SE00 bit to 0 to stop communication.** |

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

Setting up operation mode

- Serial mode register 00 (SMR00)
  Operation clock ($f_{MCK}$): Operation clock CK00 designated by SPS0 register
  Transfer clock ($f_{TCLK}$): Clock input $f_{SCK}$ from SCK00 pin(slave transfer in CSI mode)
  Operation mode: Select CSI.
  Interrupt source: Select transfer end interrupt.

Symbol: SMR00

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CKS<br>00 | CCS<br>00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | MD<br>002 | MD<br>001 | MD<br>000 |
| **0** | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | **0** | **0** | **0** |

Bit 15

| CKS00 | Selection of operation clock ($f_{MCK}$) of channel 0 |
|---|---|
| **0** | **Operation clock CK00 set by the SPS0 register** |
| 1 | Operation clock CK01 set by SPS0 register |

Bit 14

| CCS00 | Selection of transfer clock ($f_{TCLK}$) of channel 0 |
|---|---|
| 0 | Frequency-divided clock of operation clock $f_{MCK}$ designated by CKS00 bit |
| **1** | **Clock input $f_{SCK}$ from SCK00 pin(slave transfer in CSI mode)** |

Bits 2 and 1

| MD002 | MD001 | Setting of operation mode of channel 0 |
|---|---|---|
| **0** | **0** | **CSI mode** |
| 0 | 1 | UART mode |
| 1 | 0 | Simplified I$^2$C mode |
| 1 | 1 | Setting prohibited |

Bit 0

| MD000 | Selection of interrupt source of channel 0 |
|---|---|
| **0** | **Transfer end interrupt** |
| 1 | Buffer empty interrupt<br>(Occurs when data is transferred from the SDR00 register to the shift register.) |

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

Setting up communication format

- Serial communication operation setting register 00 (SCR00)
- Operation mode: Reception only
- CSI mode data clock and phase select: Type 1
- Data transfer sequence: MSB first
- Data length: 8 bits

Symbol: SCR00

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TXE00 | RXE00 | DAP00 | CKP00 | 0 | EOC00 | PTC001 | PTC000 | DIR00 | 0 | SLC001 | SLC000 | 0 | 1 | DLS001 | DLS000 |
| **0** | **1** | **0** | **0** | 0 | **0** | **0** | **0** | **0** | 0 | **0** | **0** | 0 | 1 | **1** | **1** |

Bits 15 and 14

| TXE00 | RXE00 | Setting of operation mode of channel 0 |
|---|---|---|
| 0 | 0 | Disable communication. |
| **0** | **1** | **Reception only** |
| 1 | 0 | Transmission only |
| 1 | 1 | Transmission/reception |

Bits 13 and 12

| DAP00 | CKP00 | Selection of data and clock phase in CSI mode |
|---|---|---|
| **0** | **0** | **Type 1** |
| 0 | 1 | Type 2 |
| 1 | 0 | Type 3 |
| 1 | 1 | Type 4 |

Set DAP00 and CKP00 to 0 in the UART or simplified I2C mode.

Bit 10

| EOC00 | Selection of masking of error interrupt signal (INTSRE0) |
|---|---|
| **0** | **Masks error interrupt INTSRE0 (INTSR0 is not masked).** |
| 1 | Enables generation of error interrupt INTSRE0 (INTSR0 is masked if an error occurs). |

Set EOC00 to 1 during UART reception.

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

Bits 9 and 8

| PTC001 | PTC000 | Setting of parity bit in UART mode | |
|---|---|---|---|
| | | Transmission | Reception |
| **0** | **0** | Does not output the parity bit. | **Receives without parity.** |
| 0 | 1 | Outputs 0 parity. | No parity judgement |
| 1 | 0 | Outputs even parity. | Judged as even parity. |
| 1 | 1 | Outputs odd parity. | Judged as odd parity. |

Bit 7

| DIR00 | Selection of data transfer sequence in CSI and UART modes |
|---|---|
| **0** | **Inputs/outputs data with MSB first.** |
| 1 | Inputs/outputs data with LSB first. |

Bits 5 and 4

| SLC001 | SLC000 | Setting of stop bit in UART mode |
|---|---|---|
| **0** | **0** | **No stop bit** |
| 0 | 1 | Stop bit length = 1 bit |
| 1 | 0 | Setting prohibited |
| 1 | 1 | Setting prohibited |

Set 1 bit (SLC001 = 0, SLC000 = 1) during UART reception mode and in the simplified $I^2C$ mode.

Bits 1 and 0

| DLS001 | DLS000 | Setting of data length in CSI and UART modes |
|---|---|---|
| 0 | 1 | 9-bit data length (stored in bits 0 to 8 of the SDR01 register)<br>(Selectable only in UART mode) |
| 1 | 0 | 7-bit data length (stored in bits 0 to 6 of the SDR01 register) |
| **1** | **1** | **8-bit data length (stored in bits 0 to 7 of the SDR01 register)** |
| Others | | Setting prohibited |

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

- Serial data register 00 (SDR00)
  Bits 15 to 9 must be set to 0 for slave receive mode.

Symbol: SDR00

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |

Bits 15 to 9

| SDR00[15:9] | | | | | | | Setting of transfer clock based on frequency division ratio of operation clock ($f_{MCK}$) |
|---|---|---|---|---|---|---|---|
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | $f_{MCK}$/2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | $f_{MCK}$/4 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | $f_{MCK}$/6 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | $f_{MCK}$/8 |
| •<br>•<br>• | •<br>•<br>• | •<br>•<br>• | •<br>•<br>• | •<br>•<br>• | •<br>•<br>• | •<br>•<br>• | •<br>•<br>• |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | $f_{MCK}$/254 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | $f_{MCK}$/256 |

For slave mode reception, set all of the transfer clock bits 15 to 9 in SDR00 to 0.

Bits 8 to 0

Bits 8 to 0 serve as a transmit/receive buffer register.

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

Enabling/disabling data output through target channel

- Serial output enable register 0 (SOE0)
  Disable output.

Symbol: SOE0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SOE 03 | SOE 02 | SOE 01 | SOE 00 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | **0** |

Bit 0

| SOE00 | Serial output enable/stop of channel 0 |
|---|---|
| **0** | **Stops output by serial communication operation.** |
| 1 | Enables output by serial communication operation. |

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

CSI SI00 and SCK00 pin settings

- Port mode register 1 (PM1)
  Select I/O mode of SI00 and SCK00 pins.

Symbol: PM1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PM17 | PM16 | PM15 | PM14 | PM13 | PM12 | PM11 | PM10 |
| x | x | x | x | x | x | **1** | **1** |

Bits 1 and 0

| PM11, PM10 | PM11 and PM10 I/O mode selection |
|---|---|
| 0 | Output mode (output buffer on) |
| **1** | **Input mode (output buffer off)** |

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

### 5.8.7 Interval Timer Setup

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

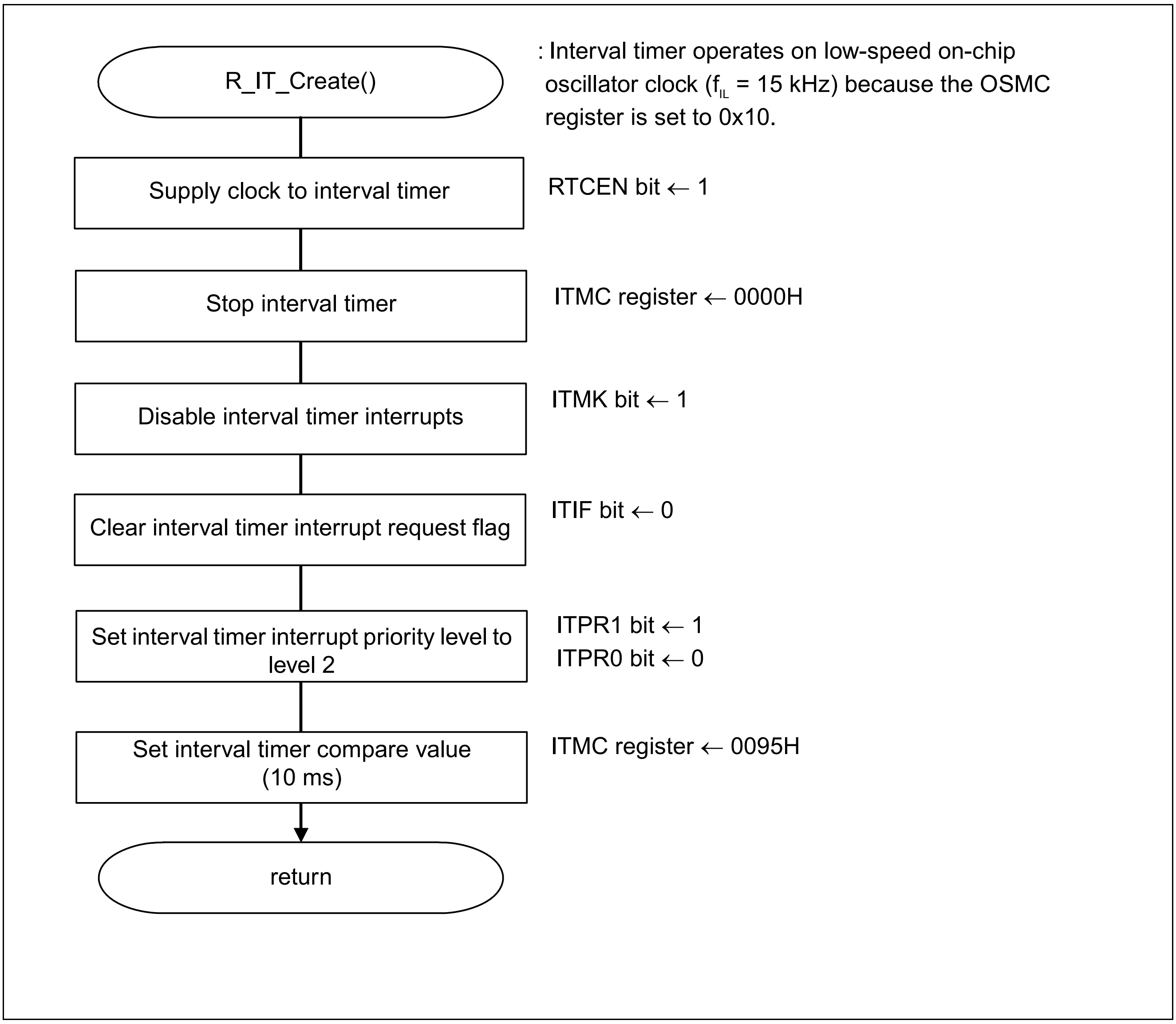

R_IT_Create()
: Interval timer operates on low-speed on-chip oscillator clock ($f_{IL}$ = 15 kHz) because the OSMC register is set to 0x10.
Supply clock to interval timer
RTCEN bit ← 1
Stop interval timer
ITMC register ← 0000H
Disable interval timer interrupts
ITMK bit ← 1
Clear interval timer interrupt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

**Figure 5.8 Interval Timer Setup**

### 5.8.8 External Interrupt Input Setup

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

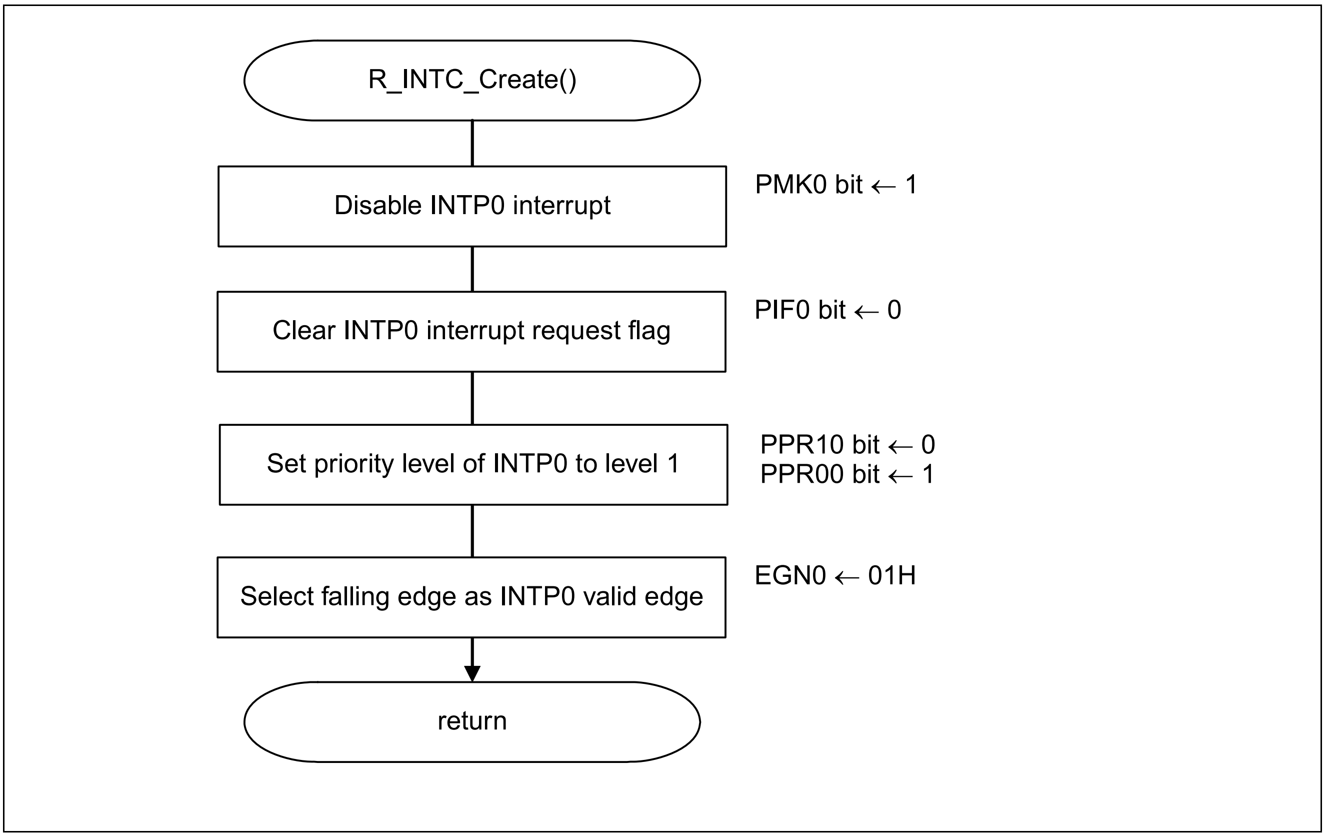

R_INTC_Create()
Disable INTP0 interrupt
PMK0 bit ← 1
Clear INTP0 interrupt request flag
PIF0 bit ← 0
Set priority level of INTP0 to level 1
PPR10 bit ← 0
PPR00 bit ← 1
Select falling edge as INTP0 valid edge
EGN0 ← 01H
return

**Figure 5.9 External Interrupt Input Setup**

### 5.8.9 Main Processing

Figure 5.10 shows the flowcharts for the main processing (1/2).

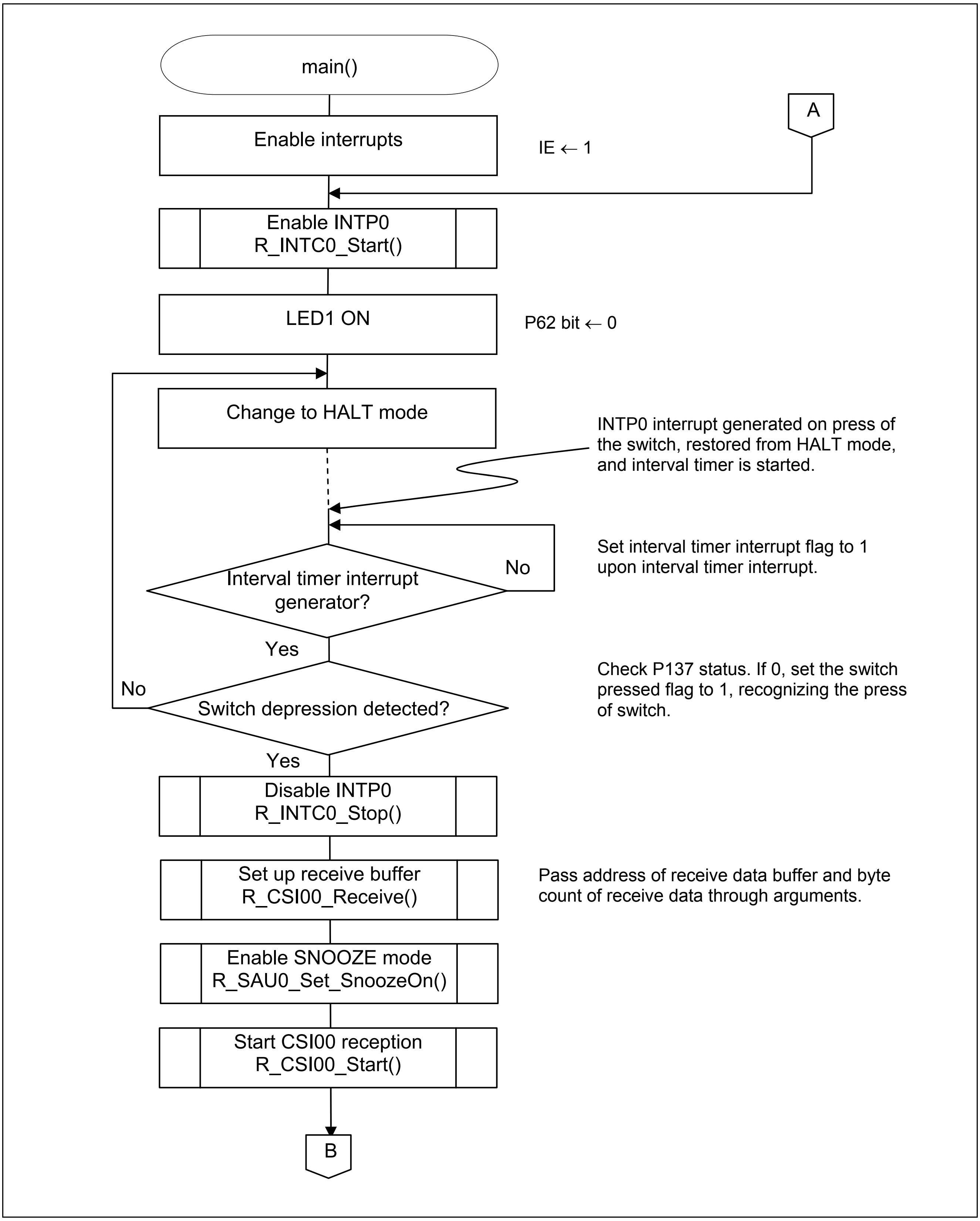

main()
Enable interrupts
IE ← 1
A
Enable INTP0
R_INTC0_Start()
LED1 ON
P62 bit ← 0
Change to HALT mode
INTP0 interrupt generated on press of the switch, restored from HALT mode, and interval timer is started.
Interval timer interrupt generator?
No
Set interval timer interrupt flag to 1 upon interval timer interrupt.
Yes
No
Switch depression detected?
Check P137 status. If 0, set the switch pressed flag to 1, recognizing the press of switch.
Yes
Disable INTP0
R_INTC0_Stop()
Set up receive buffer
R_CSI00_Receive()
Pass address of receive data buffer and byte count of receive data through arguments.
Enable SNOOZE mode
R_SAU0_Set_SnoozeOn()
Start CSI00 reception
R_CSI00_Start()
B

**Figure 5.10 Main Processing (1/2)**

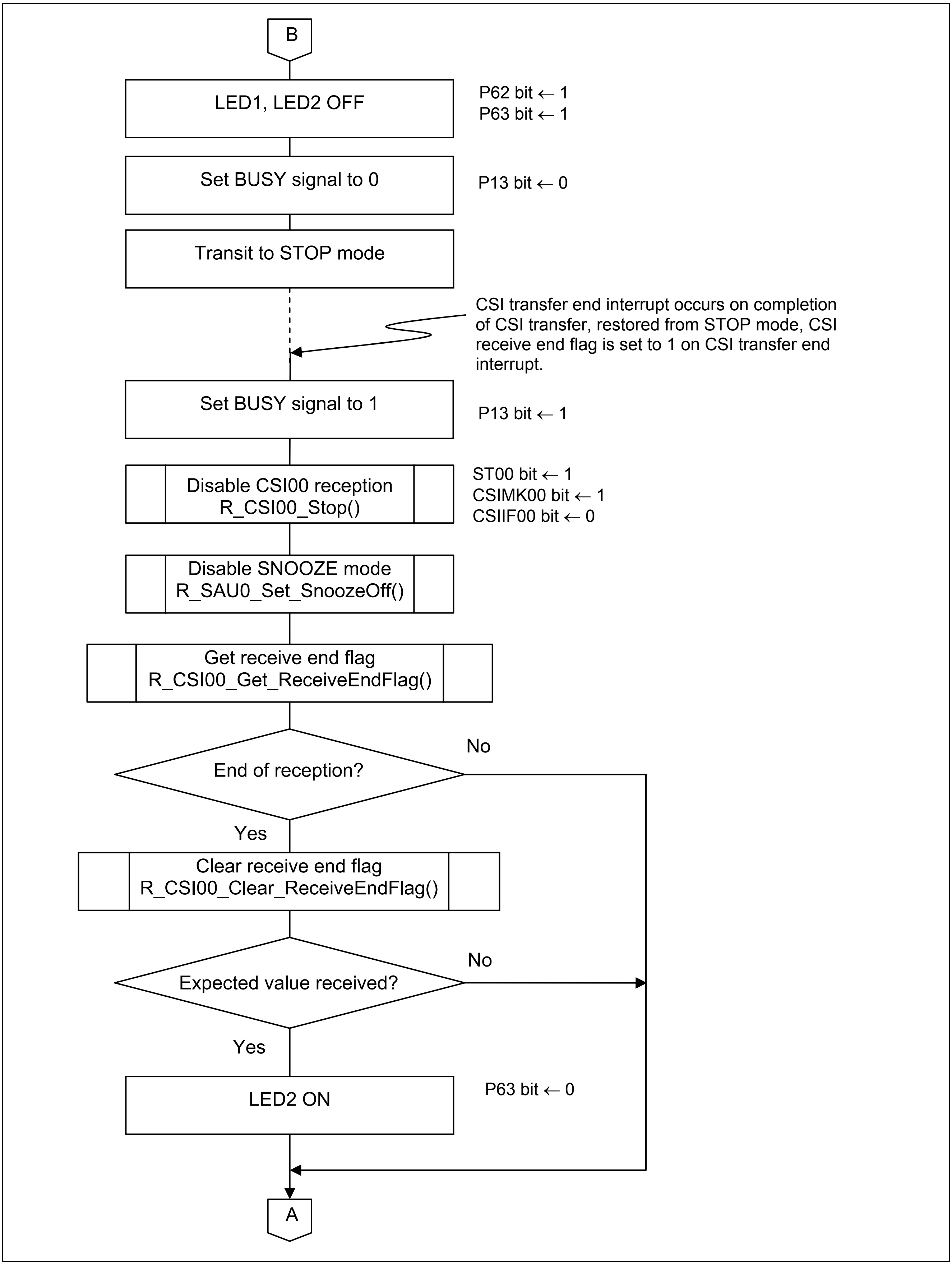


**Figure 5.11 Main Processing (2/2)**

### 5.8.10 Enabling the CSI00 in SNOOZE Mode

Figure 5.12 shows the flowchart for enabling the CSI00 in SNOOZE mode.

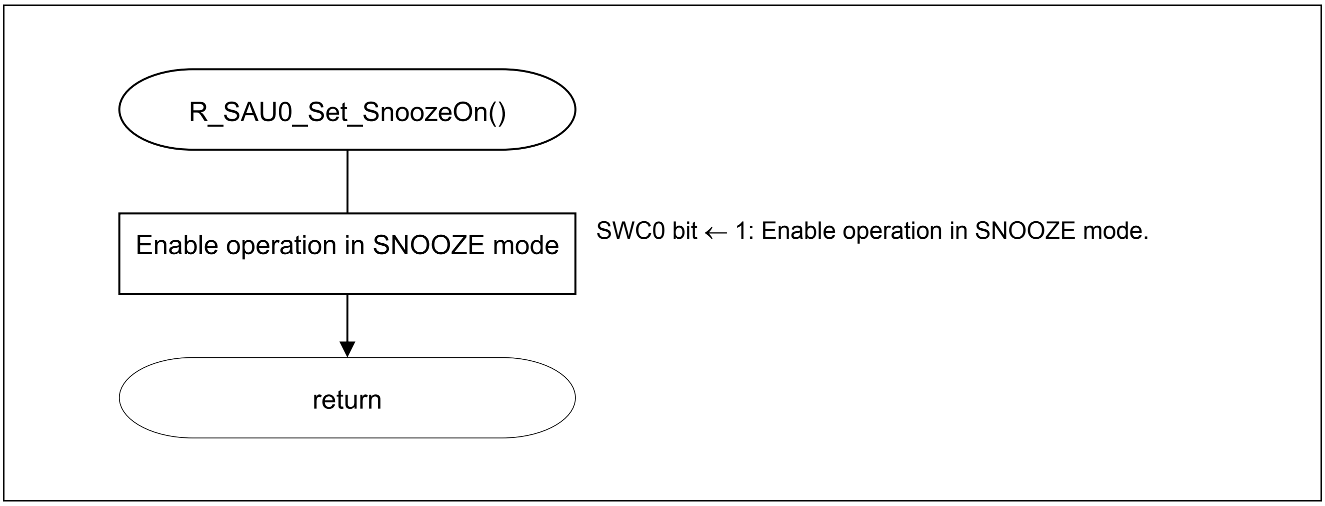

R_SAU0_Set_SnoozeOn()
Enable operation in SNOOZE mode
SWC0 bit ← 1: Enable operation in SNOOZE mode.
return

**Figure 5.12 Enabling CSI00 in SNOOZE Mode**

Controlling the SNOOZE mode

- Serial standby control register 0 (SSC0)
  Disable error interrupts (INTSRE0/INTSRE2).
  Enable starting reception in STOP mode.

Symbol: SSC0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SSEC0 | SWC0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **0** | **1** |

Bit 1

| SSEC0 | Selection of whether to enable or stop the generation of transfer end interrupts |
|---|---|
| **0** | **Enable the generation of error interrupts (INTSRE0/INTSRE2).**<br>**In the following cases, the clock request signal (an internal signal) to the clock generator is also cleared:**<br>• **When the SWC0 bit is cleared to 0**<br>• **When the UART reception start bit is mistakenly detected** |
| 1 | Stop the generation of error interrupts (INTSRE0/INTSRE2).<br>In the following cases, the clock request signal (an internal signal) to the clock generator is also cleared:<br>• When the SWC0 bit is cleared to 0<br>• When the UART reception start bit is mistakenly detected<br>• When the transfer end interrupt generation timing is based on a parity error or framing error |

Bit 0

| SWC0 | Selection of whether to enable or stop the startup of CSI00 or UART0 reception while in the STOP mode |
|---|---|
| 0 | Stop the startup of reception while in the STOP mode. |
| **1** | **Enable the startup of reception while in the STOP mode.**<br>**(During asynchronous CSI00/CSI20 reception or UART0/UART2 reception, the baud rate adjustment function is enabled.)** |

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

### 5.8.11 Disabling the CSI00 in SNOOZE Mode

Figure 5.13 shows the flowchart for disabling the CSI00 in SNOOZE mode.

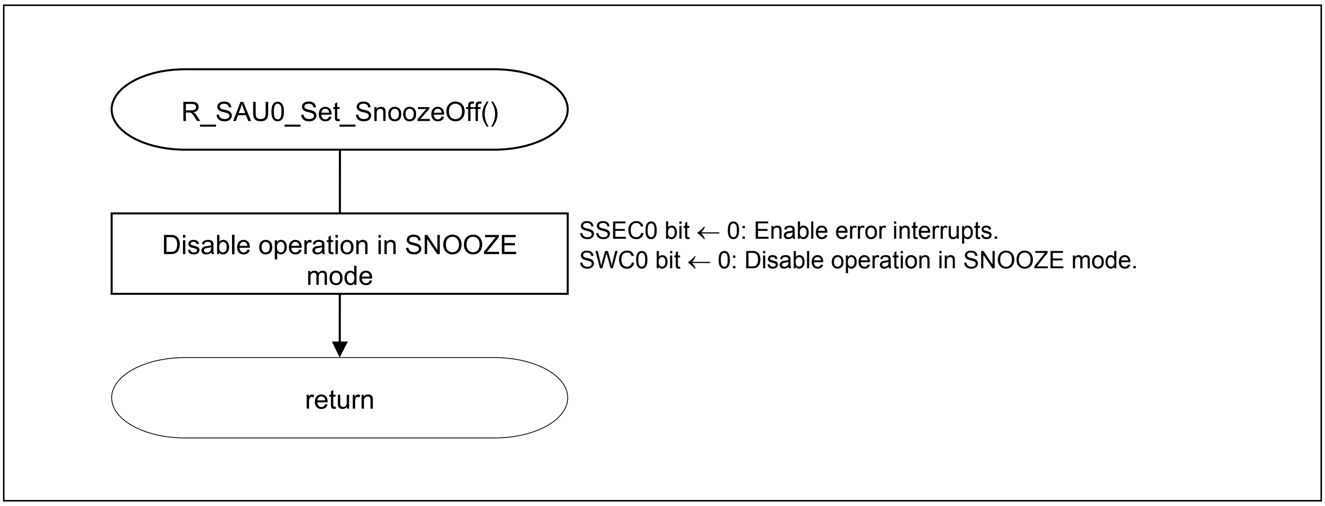

R_SAU0_Set_SnoozeOff()
Disable operation in SNOOZE mode
SSEC0 bit ← 0: Enable error interrupts.
SWC0 bit ← 0: Disable operation in SNOOZE mode.
return

**Figure 5.13 Disabling the CSI00 in SNOOZE Mode**

### 5.8.12 Starting CSI00 Reception

Figure 5.14 shows the flowchart for starting CSI00 reception.

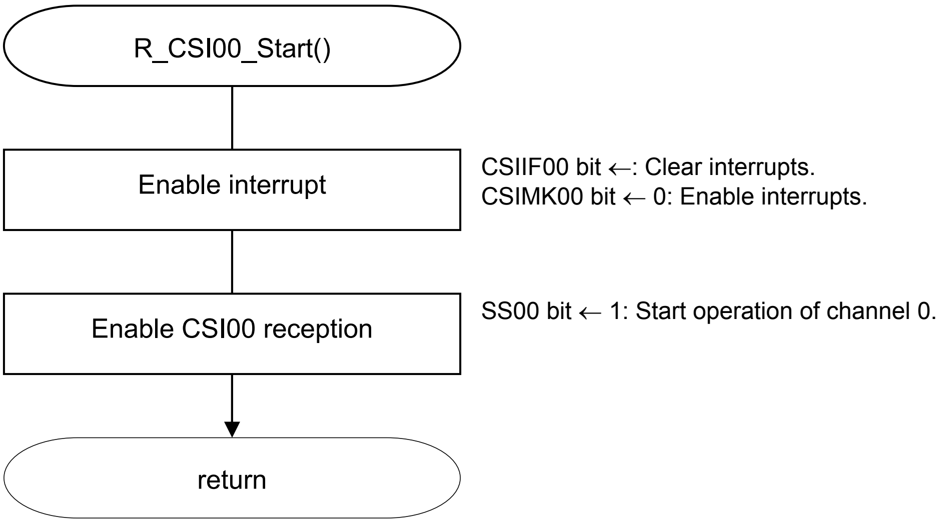

R_CSI00_Start()
Enable interrupt
CSIIF00 bit ←: Clear interrupts.
CSIMK00 bit ← 0: Enable interrupts.
Enable CSI00 reception
SS00 bit ← 1: Start operation of channel 0.
return

**Figure 5.14 Starting CSI00 Reception**

Making settings for awaiting the startup of CSI00 reception operation

- Serial channel start register 0 (SS0)
  Switch channel 0 to communication wait state.

Symbol: SS0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SS03 | SS02 | SS01 | SS00 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** |

Bit 0

| SS00 | Operation start trigger of channel 0 |
|---|---|
| 0 | No trigger operation |
| **1** | **Sets the SE00 bit to 1 and enters the communication wait status.** |

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

### 5.8.13 Stopping CSI00 Reception

Figure 5.15 shows the flowchart for stopping CSI00 reception.

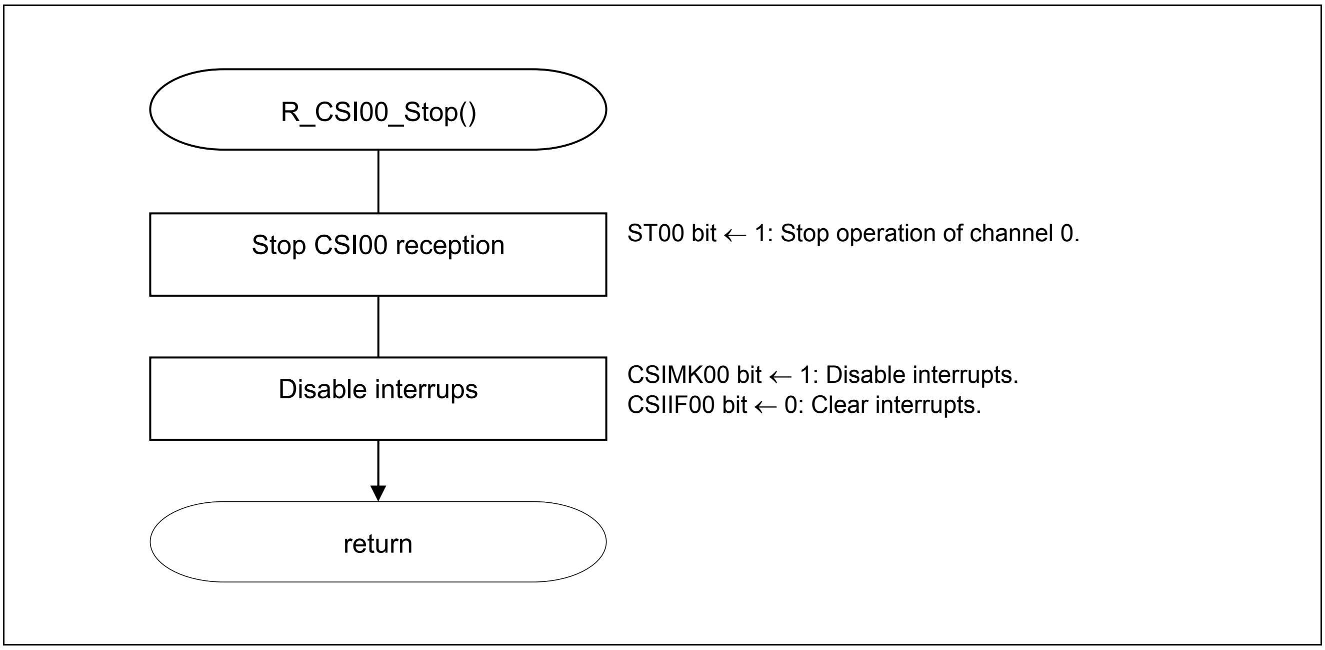

R_CSI00_Stop()
Stop CSI00 reception
ST00 bit ← 1: Stop operation of channel 0.
Disable interrups
CSIMK00 bit ← 1: Disable interrupts.
CSIIF00 bit ← 0: Clear interrupts.
return

**Figure 5.15 Stopping CSI00 Reception**

Making settings for stopping CSI00 reception operation.

- Serial channel stop register 0 (ST0)
  Enable stopping channel 0 communication.

Symbol: ST0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ST03 | ST02 | ST01 | ST00 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** |

Bit 0

| ST00 | Operation stop trigger of channel 0 |
|---|---|
| 0 | No trigger operation |
| **1** | **Clears SE00 bit to 0 and stops communication** Note. |

Note: Communication stops while holding the value of the control register and shift register, and the status of the serial clock I/O pin, serial data output pin, and the error flags (FEFmn: framing error flag, PEFmn: parity error flag, and OVFmn: overrun error flag).

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

### 5.8.14 Setting up the CSI00 Receive Buffer

Figure 5.16 shows the flowchart for setting up the CSI00 receive buffer.

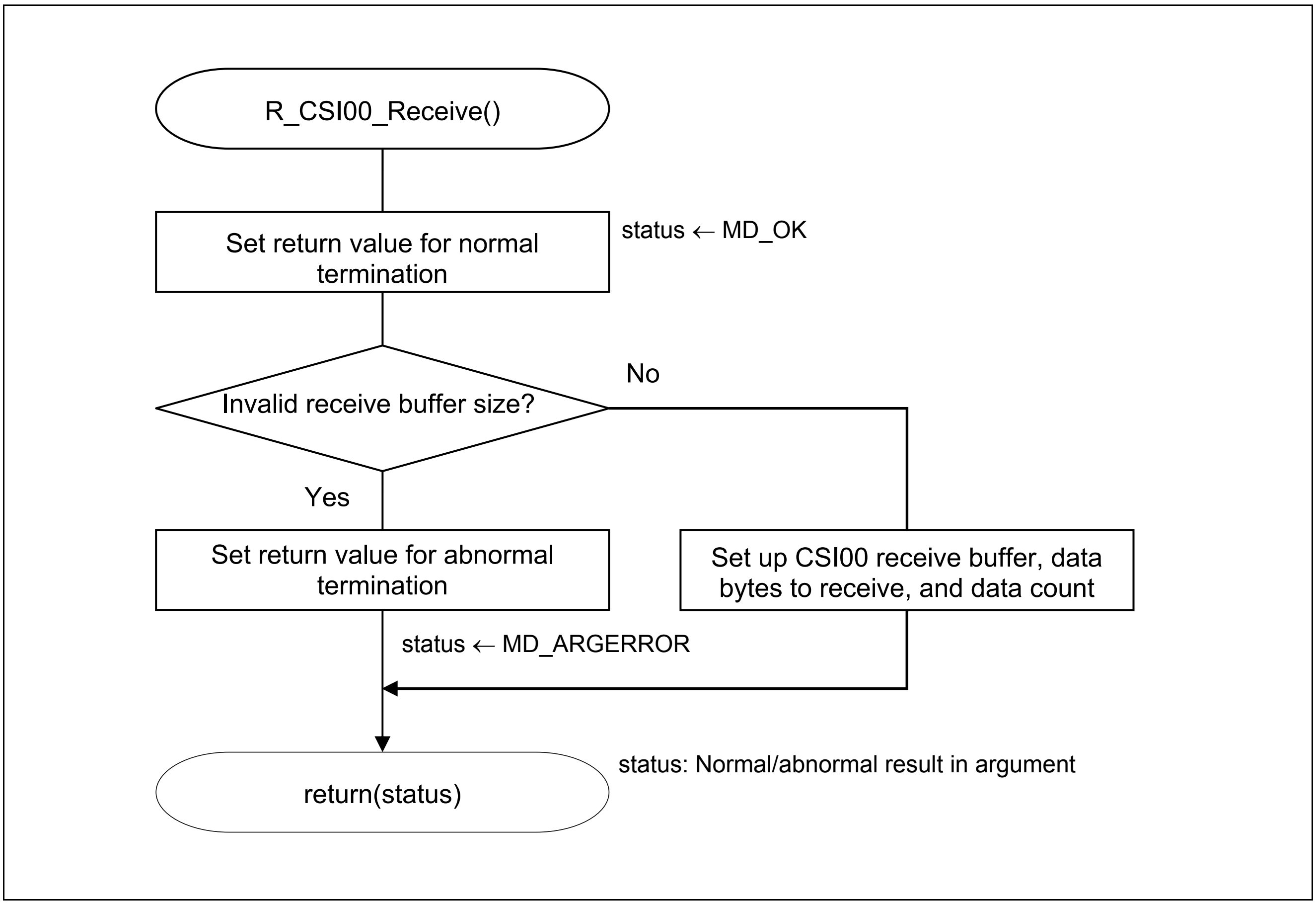

R_CSI00_Receive()
Set return value for normal termination
status ← MD_OK
Invalid receive buffer size?
No
Yes
Set return value for abnormal termination
Set up CSI00 receive buffer, data bytes to receive, and data count
status ← MD_ARGERROR
return(status)
status: Normal/abnormal result in argument

**Figure 5.16 Setting up the CSI00 Receive Buffer**

### 5.8.15 CSI00 Communication End Interrupts

Figure 5.17 shows the flowchart for CSI00 communication end interrupts.

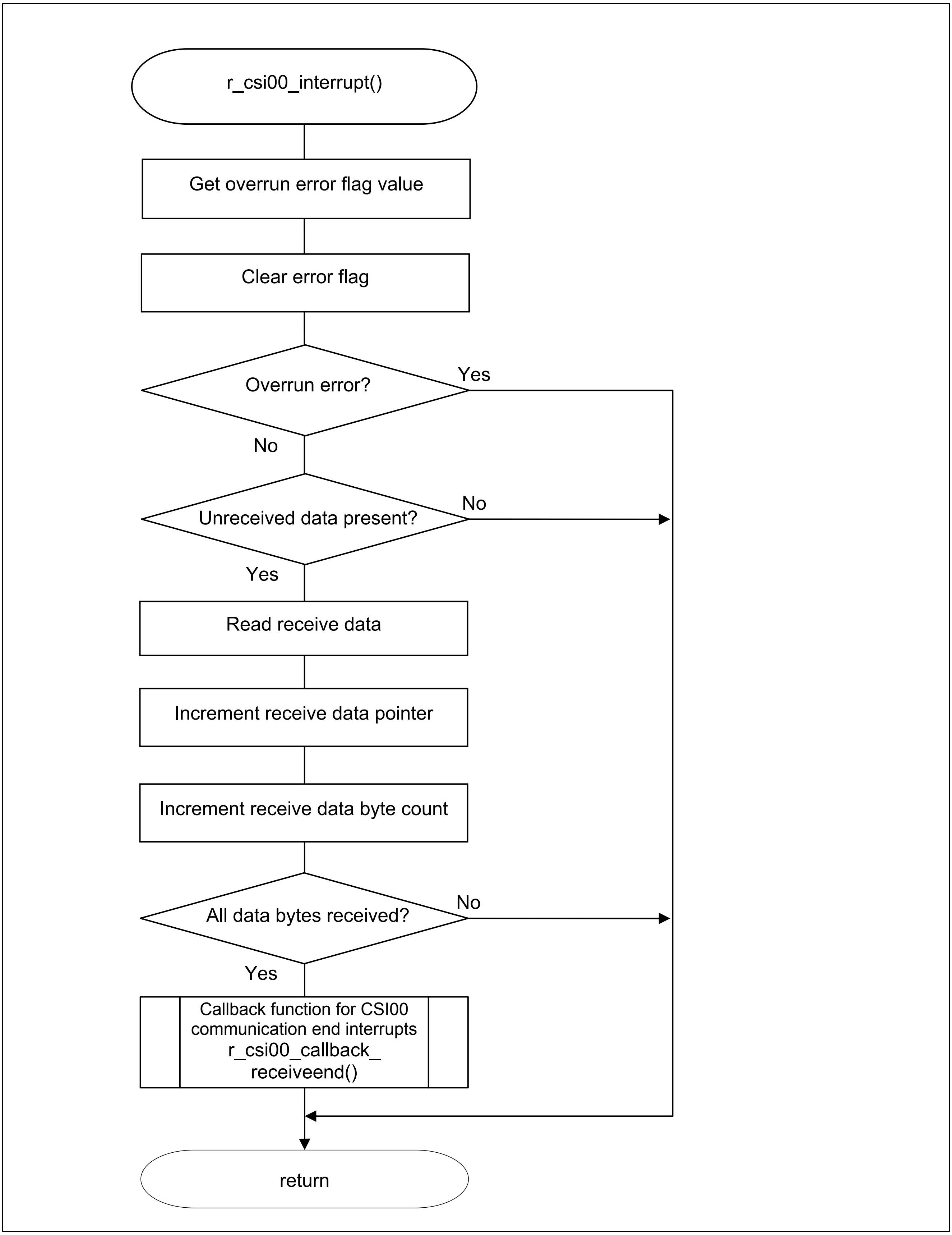

r_csi00_interrupt()
Get overrun error flag value
Clear error flag
Overrun error?
Yes
No
Unreceived data present?
No
Yes
Read receive data
Increment receive data pointer
Increment receive data byte count
All data bytes received?
No
Yes
Callback function for CSI00 communication end interrupts r_csi00_callback_ receiveend()
return

**Figure 5.17 CSI00 Communication End Interrupt**

### 5.8.16 Callback Function for CSI00 Communication End Interrupts

Figure 5.18 shows the flowchart for the callback function for callback function for CSI00 communication end interrupts.

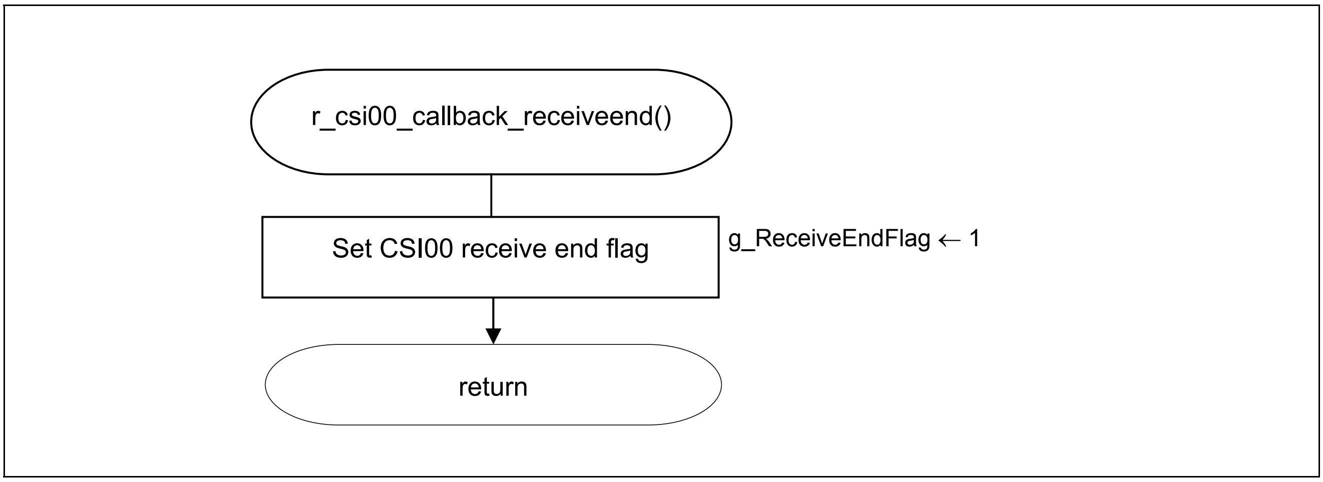


**Figure 5.18 Callback Function for SI00 Communication End Interrupts**

### 5.8.17 Getting Receive End Flag

Figure 5.19 shows the flowchart for getting the receive end flag. This function does nothing but returns global variable g_ReceiveEndFlag as a return value.

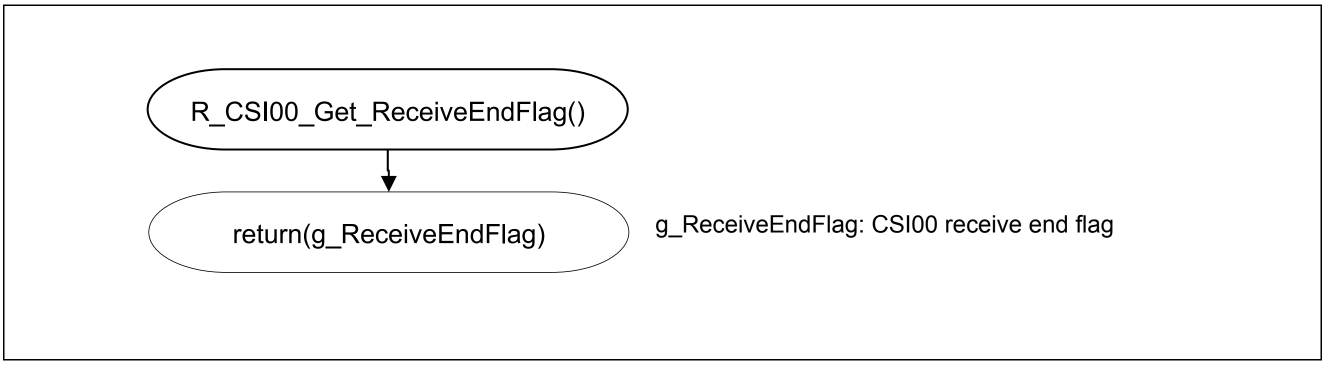

R_CSI00_Get_ReceiveEndFlag()
return(g_ReceiveEndFlag)
g_ReceiveEndFlag: CSI00 receive end flag

**Figure 5.19 Getting the Receive End Flag**

### 5.8.18 Clearing the Receive End Flag

Figure 5.20 shows the flowchart for clearing the receive end flag.

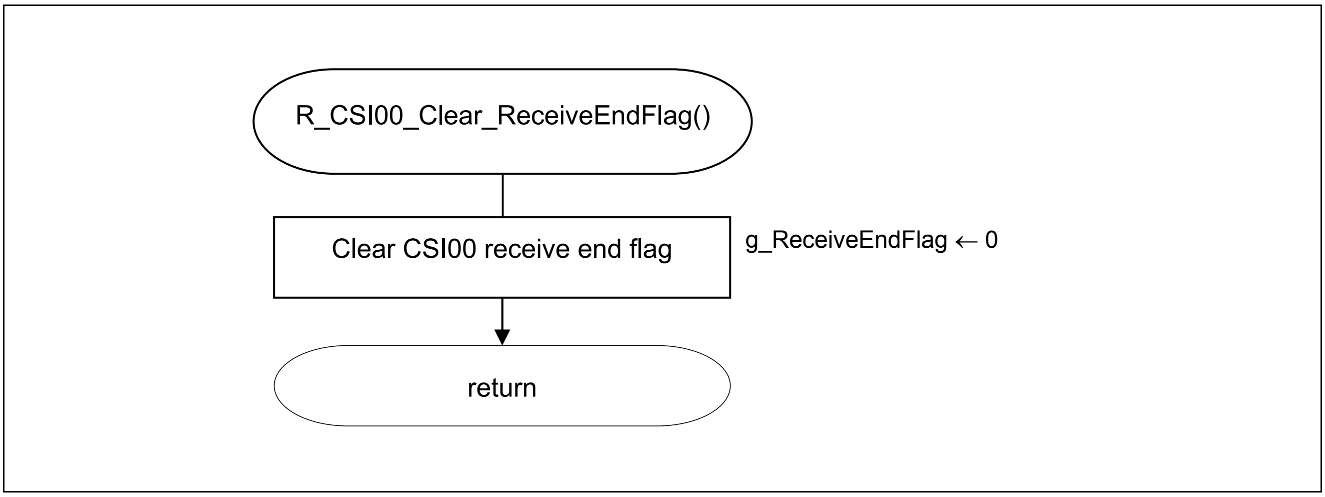


**Figure 5.20 Clearing the Receive End Flag**

### 5.8.19 Starting INTP0 Operation

Figure 5.21 shows the flowchart for starting INTP0 operation.

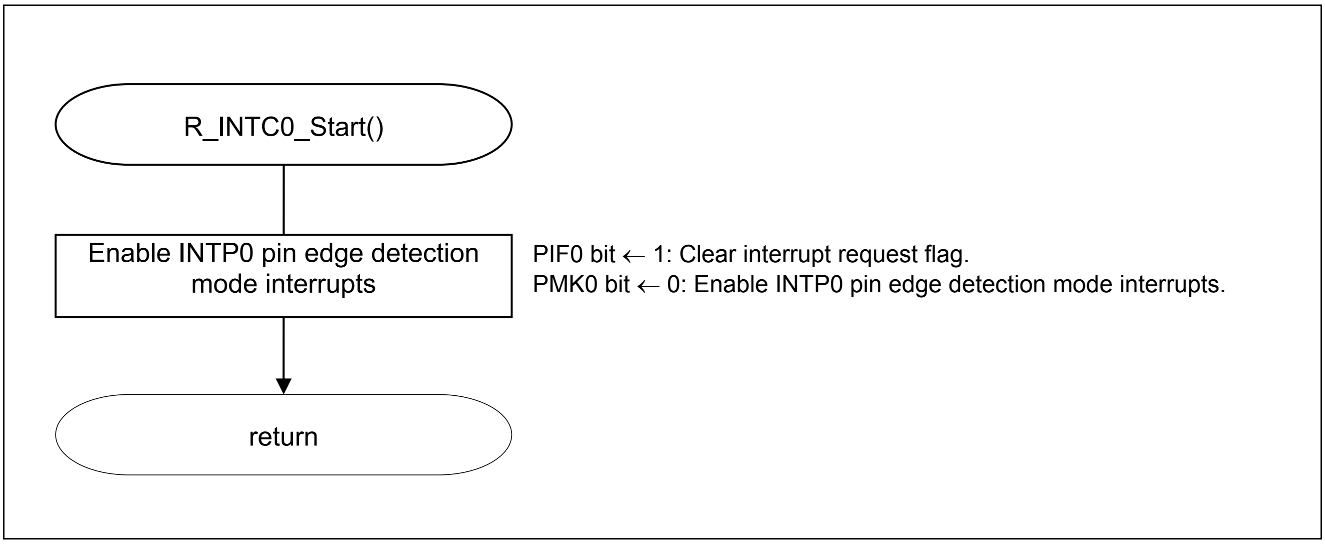

R_INTC0_Start()
Enable INTP0 pin edge detection mode interrupts
PIF0 bit ← 1: Clear interrupt request flag.
PMK0 bit ← 0: Enable INTP0 pin edge detection mode interrupts.
return

**Figure 5.21 Starting INTP0 Operation**

Setting up the INTP0 interrupt

- Interrupt request flag register (IF0L)
  Clear interrupt request flag.
- Interrupt mask flag register (MK0L)
  Clear interrupt mask.

Symbol: IF0L

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PIF5 | PIF4 | PIF3 | PIF2 | PIF1 | PIF0 | LVIIF | WDTIIF |
| x | x | x | x | x | **0** | x | x |

Bit 2

| PIF0 | Interrupt request flag |
|---|---|
| **0** | **No interrupt request signal is generated** |
| 1 | Interrupt request is generated, interrupt request status |

Symbol: MK0L

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PMK5 | PMK4 | PMK3 | PMK2 | PMK1 | PMK0 | LVIMK | WDTIMK |
| x | x | x | x | x | **0** | x | x |

Bit 2

| PMK0 | Interrupt processing control |
|---|---|
| **0** | **Interrupt processing enabled** |
| 1 | Interrupt processing disabled |

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

### 5.8.20 Stopping INTP0

Figure 5.22 shows the flowchart for stopping INTP0.

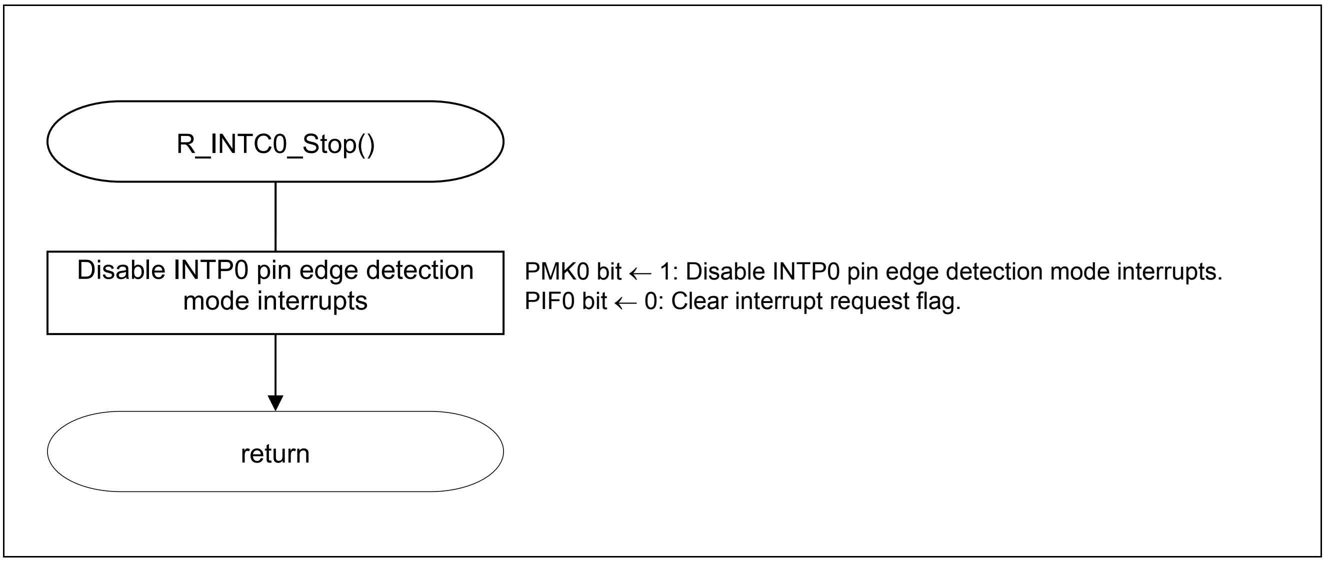
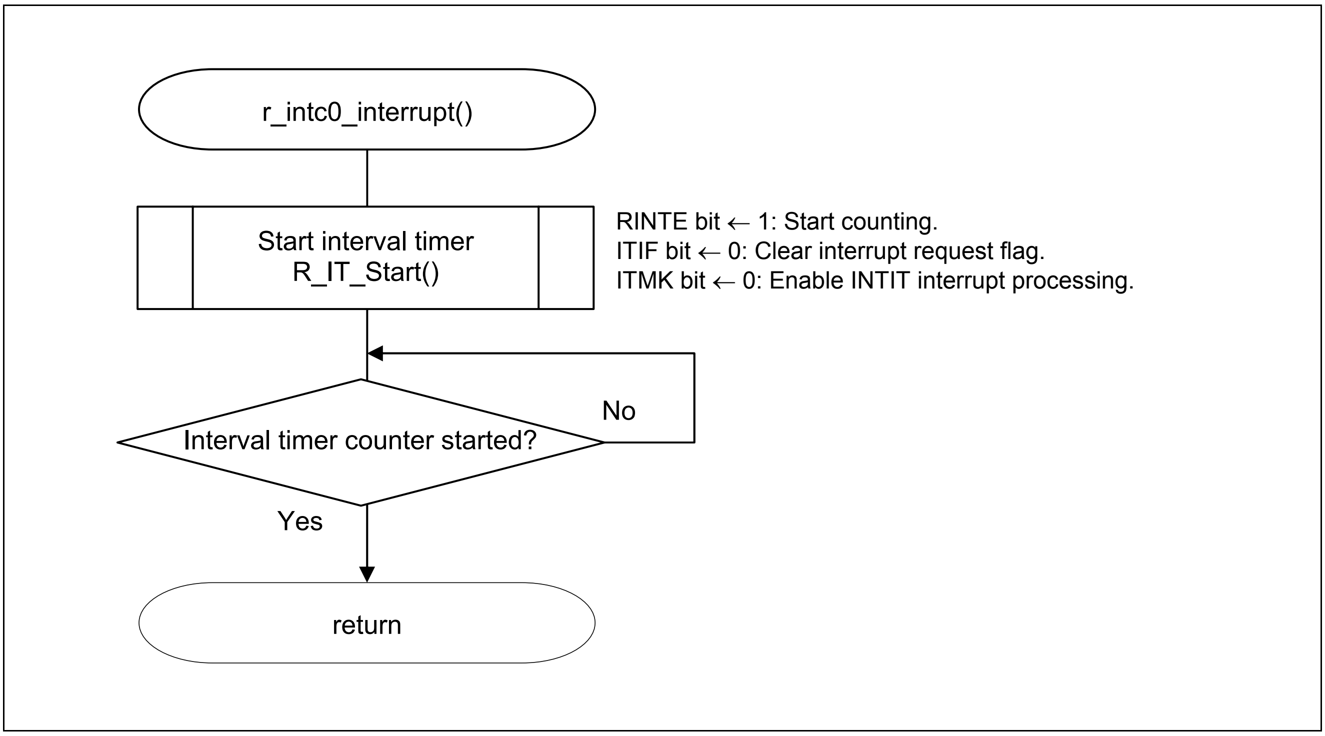

R_INTC0_Stop()
Disable INTP0 pin edge detection mode interrupts
PMK0 bit ← 1: Disable INTP0 pin edge detection mode interrupts.
PIF0 bit ← 0: Clear interrupt request flag.
return

**Figure 5.22 Stopping INTP0 Operation**

### 5.8.21 INTP0 Interrupt

Figure 5.23 shows the flowchart for an INTP0 interrupt.



RINTE bit ← 1: Start counting.
ITIF bit ← 0: Clear interrupt request flag.
ITMK bit ← 0: Enable INTIT interrupt processing.

**Figure 5.23 INTP0 Interrupt**

### 5.8.22 Starting Interval Timer

Figure 5.24 shows the flowchart for staring the interval timer.

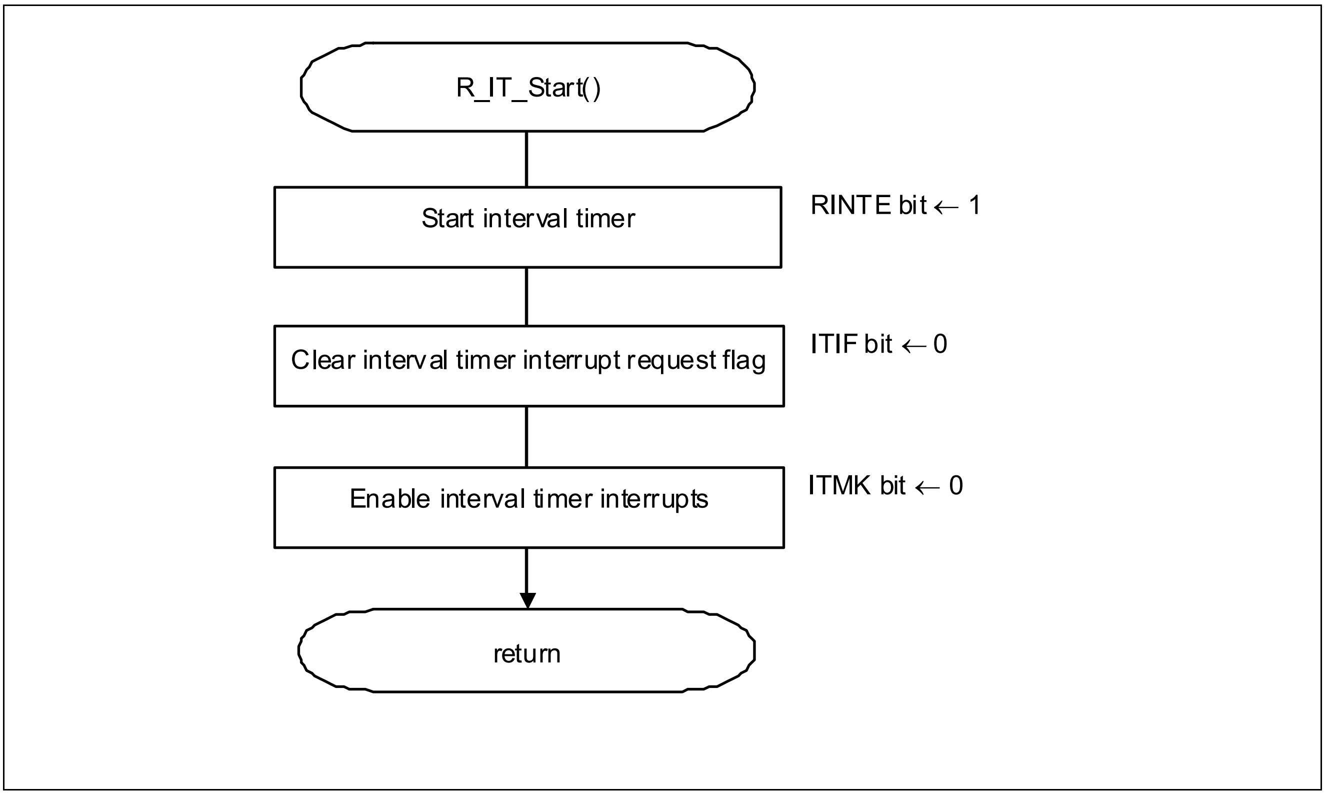


**Figure 5.24 Starting Interval Timer**

### 5.8.23 Stopping Interval Timer

Figure 5.25 shows the flowchart for stopping the interval timer.

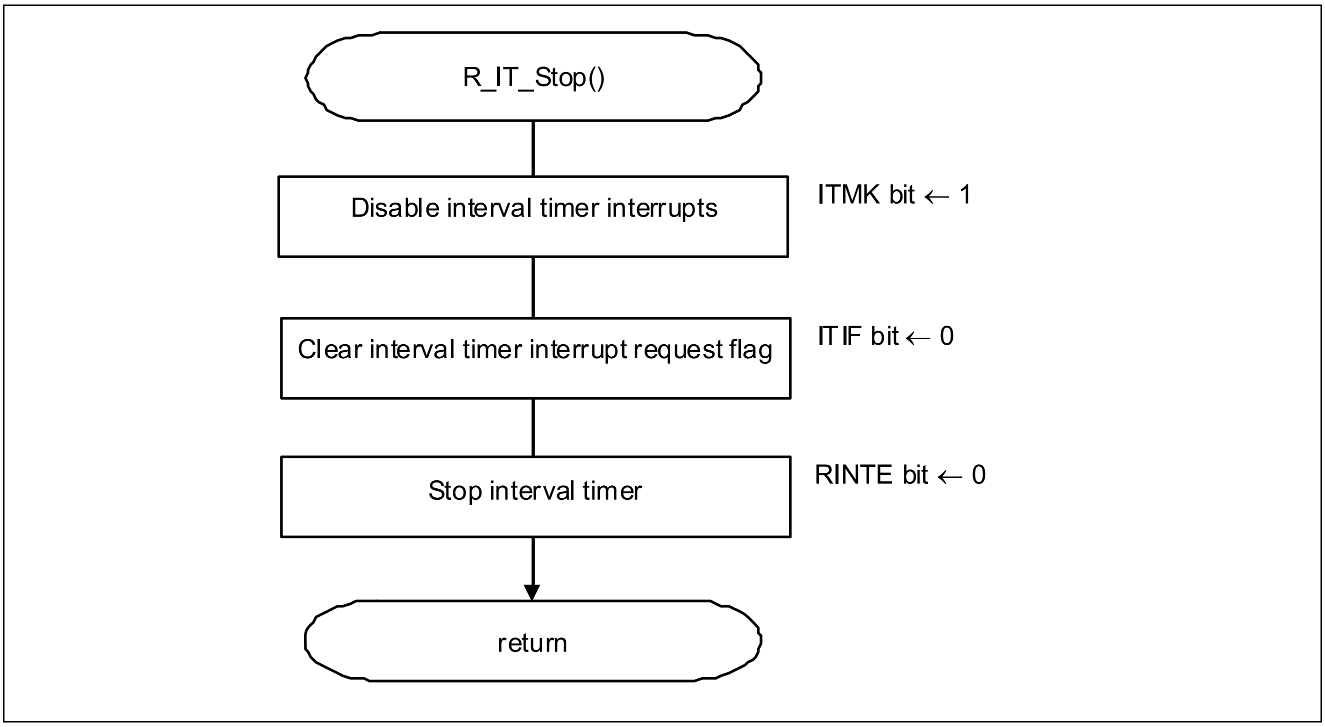

R_IT_Stop()
Disable interval timer interrupts
ITMK bit ← 1
Clear interval timer interrupt request flag
IT IF bit ← 0
Stop interval timer
RINTE bit ← 0
return

**Figure 5.25 Stopping Interval Timer**

### 5.8.24 Interval Timer Interrupt

Figure 5.26 shows the flowchart for an interval timer interrupt.

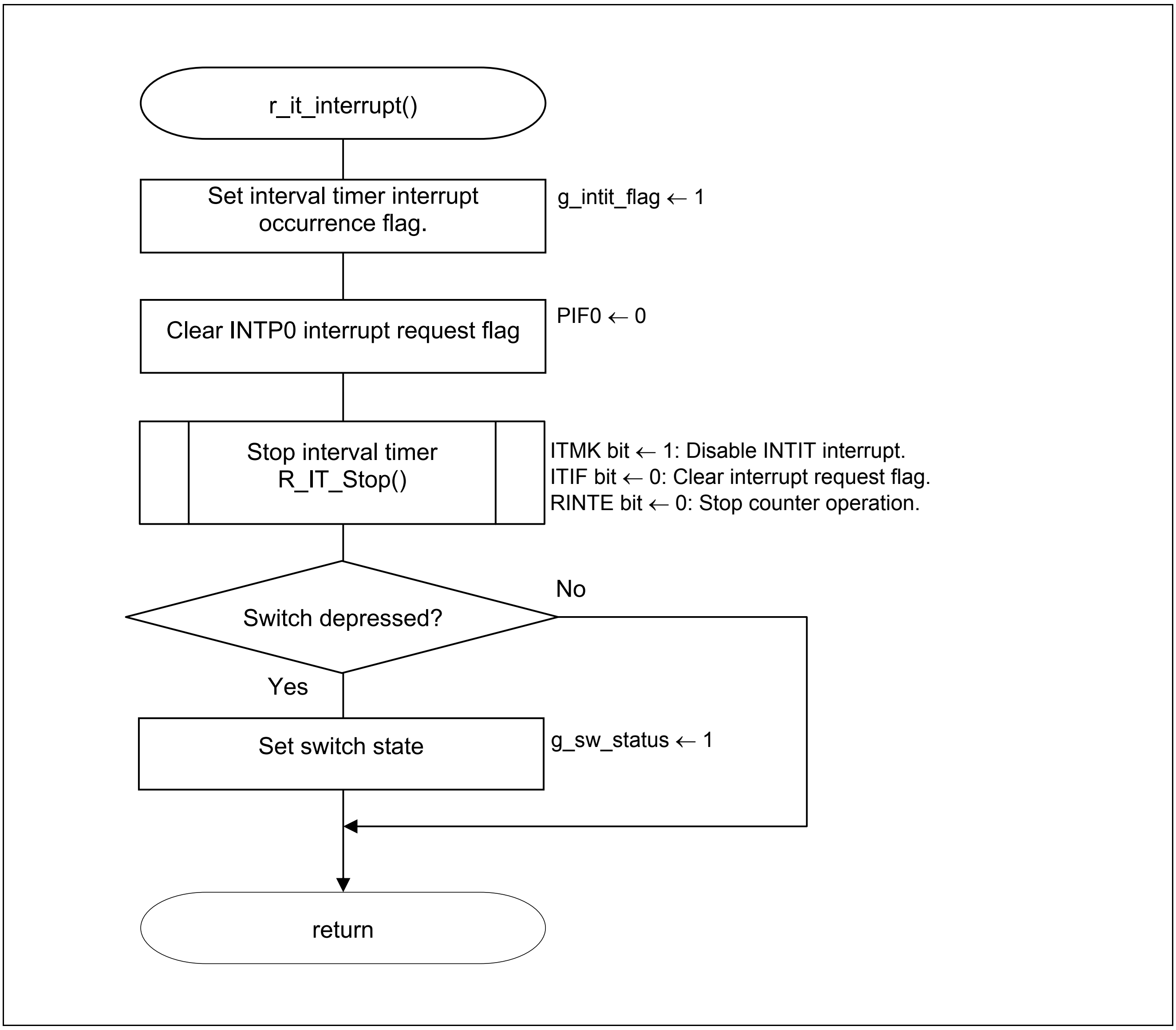

r_it_interrupt()
Set interval timer interrupt occurrence flag.
g_intit_flag ← 1
Clear INTP0 interrupt request flag
PIF0 ← 0
Stop interval timer
R_IT_Stop()
ITMK bit ← 1: Disable INTIT interrupt.
ITIF bit ← 0: Clear interrupt request flag.
RINTE bit ← 0: Stop counter operation.
Switch depressed?
No
Yes
Set switch state
g_sw_status ← 1
return

**Figure 5.26 Interval Timer Interrupt**

### 5.8.25 Getting the Switch State

Figure 5.27 shows the flowchart for getting the switch state. This function does nothing but returns global variable g_sw_status as a return value.

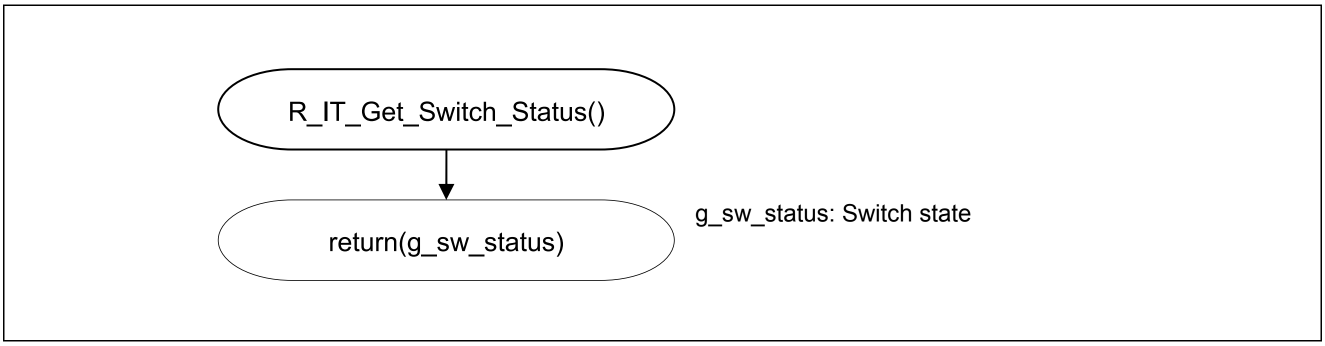


**Figure 5.27 Getting the Switch State**

### 5.8.26 Clearing the Switch State

Figure 5.28 shows the flowchart for clearing the switch state.

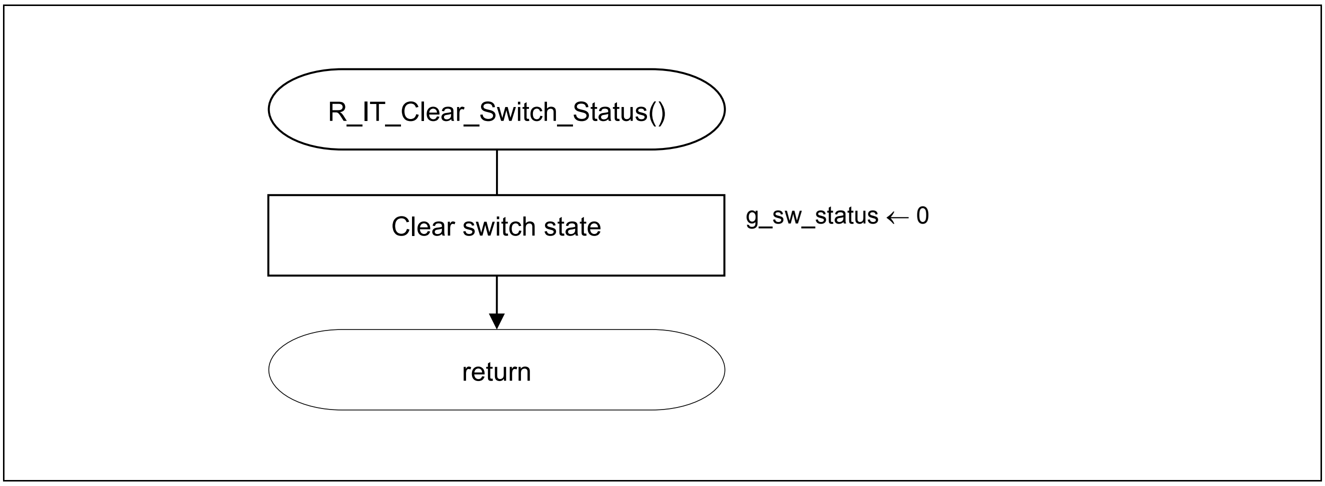


**Figure 5.28 Clearing the Switch State**

### 5.8.27 Getting the Interval Timer Interrupt Occurrence Flag

Figure 5.29 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.

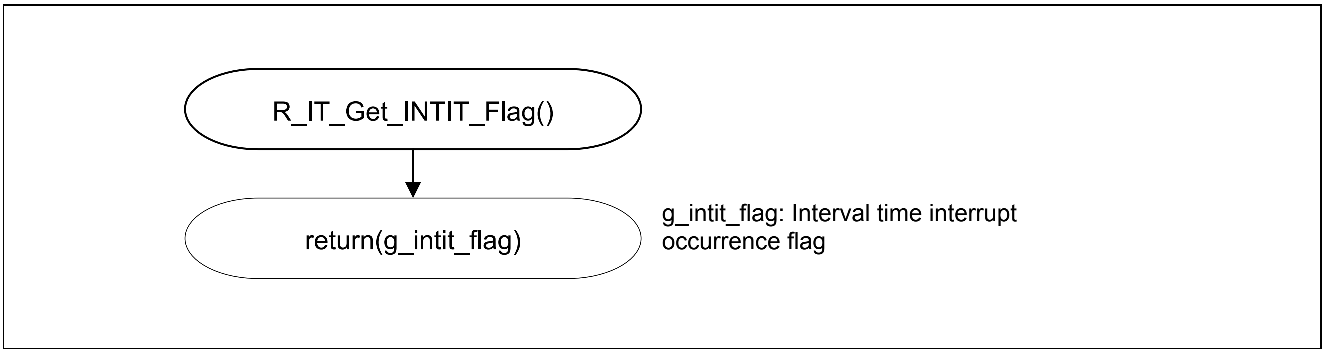


**Figure 5.29 Getting the Interval Timer Interrupt Occurrence Flag**

### 5.8.28 Clearing the Interval Timer Interrupt Occurrence Flag

Figure 5.30 shows the flowchart for clearing the interval timer interrupt occurrence flag.

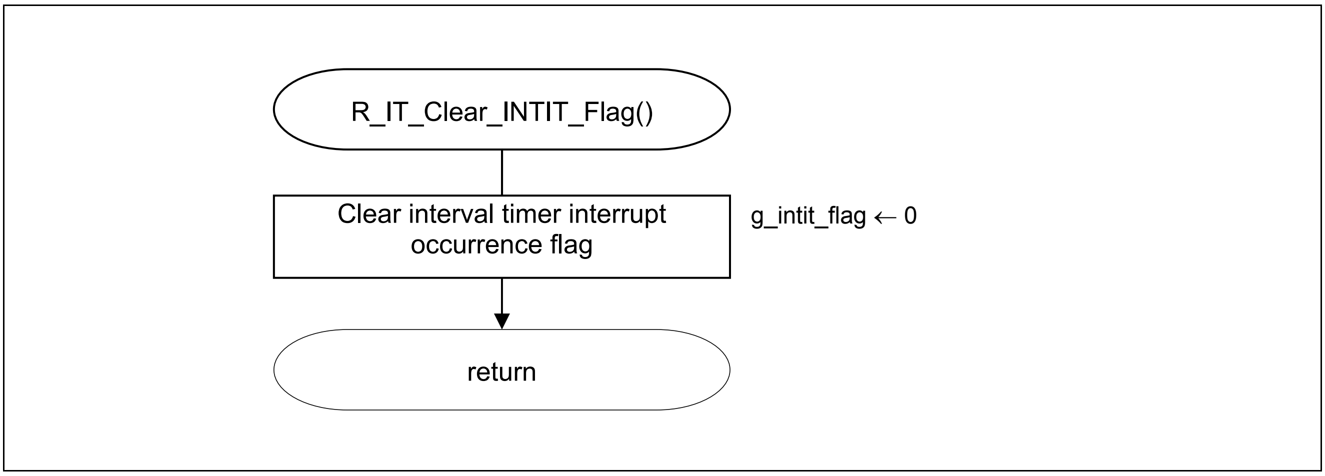


**Figure 5.30 Clearing the Interval Timer Interrupt Occurrence Flag**

## 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 Low-power Consumption Operation (CSI in SNOOZE Mode) |
|---|---|

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Apr. 16, 2015 | — | First edition issued |
| | | | |

## General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins

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

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

2. Processing at Power-on

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

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

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

4. Clock Signals

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

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

5. Differences between Products

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

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

**SALES OFFICES**

Renesas Electronics Corporation

http://www.renesas.com

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

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

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

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

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

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

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

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

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

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

**Renesas Electronics Malaysia Sdn.Bhd.**
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

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

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

Colophon 5.0