RISC-V

Realtime Clock

Introduction

This application note shows usage examples of the fixed-cycle interrupt function and the alarm interrupt function of the realtime clock (RTC).

The fixed-cycle interrupt function inverts the outputs of output ports and displays the clock time on the LCD. The alarm interrupt function generates an alarm interrupt five seconds after the set clock time.

Continuous RTC operation can be monitored with the clock time display on the LCD even during a reset period.

Target Device

RISC-V

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

Contents

1. Specifications

1.1 Overview of Specifications

The fixed-cycle interrupt function inverts the outputs of output ports and displays the clock time on the LCD. The alarm interrupt function generates an alarm interrupt five seconds after the set clock time.

Table 1.1 lists peripheral functions to be used and Figure 1.1 shows an overview of sample code operation.

Table 1.1 Peripheral Function and Use

Peripheral Function	Use
Realtime clock	RTC interrupt (INTRTC)
P107	Set to port output in the fixed-cycle interrupt processing (inverted output)
P100	Set to port output in the alarm interrupt processing (high level output)
Serial Interface IICA1 P011/SCLA1, P010/SDAA1	I^{2}C communication with the LCD module (LCM)
$\overline{\text{RES}}$	External reset input
P3/XT1	RTC operating clock

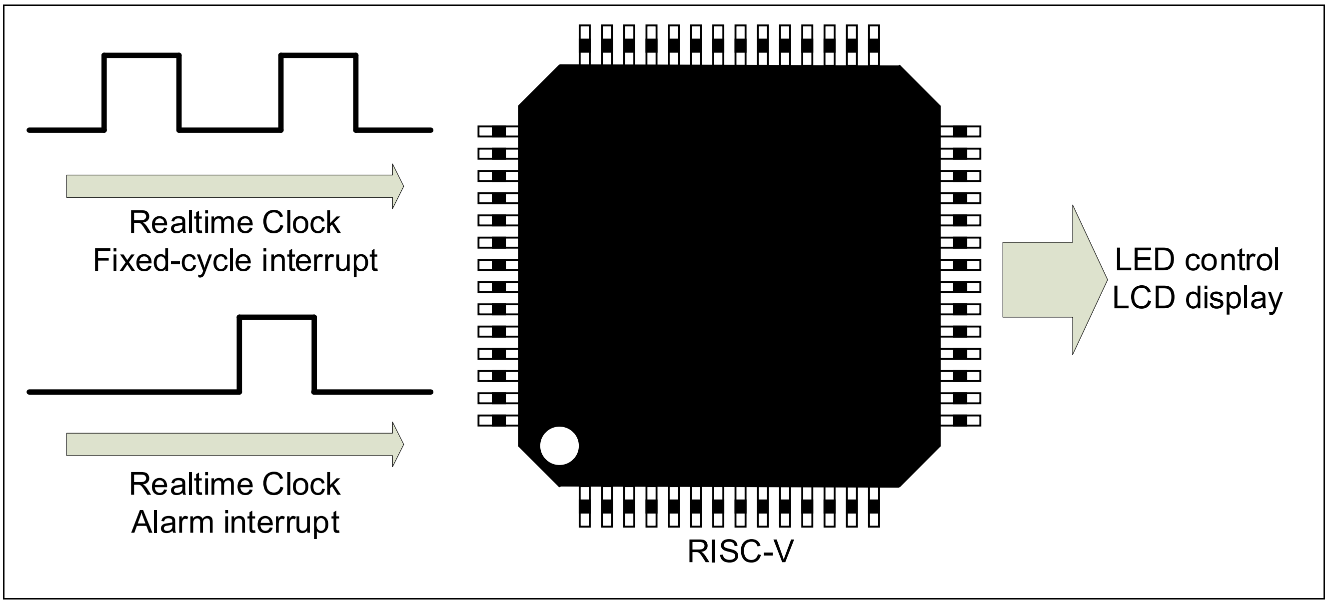

Figure 1.1 Overview of Sample Code Operation

1.2 Outline of operation

In this application note, the clock time of the RTC is set to “2024/1/1 (Mon) 15:59:55” and the alarm time is set to “16:00:00 every day”. Furthermore, the following interrupt processing is performed.

- P107 output inversion and clock time display on the LCD in the fixed-cycle interrupt processing
- High-level output of P100 (LED ON) in the alarm interrupt processing

(1) Initialize the realtime clock (RTC)
- Select the subsystem clock (fsosc) at the RTC operation clock.
- Present the time in 24-hour system.
- Disable the RTC1HZ pin output.
- Enable fixed-cycle interrupt and set their cycle time to 0.5 second.
- Enable alarm interrupt.
- Enable INTRTC interrupts.

(2) Initialize the I/O ports.
- Set P107 to the output port for the fixed-cycle interrupt processing. (The LED is lit when the initial value is high level.)
- Set P100 to the output port for the alarm interrupt processing. (The LED is unlit when the initial value is low level.)

(3) Initialize the serial interface IICA
- Use IICA1 (P011 set to SCLA1 and P010 set to SDAA1).
- Select PCLKB/2 at the IICA1 operation clock.
- Set the local address to 0x10.
- Set the standard mode as the operation mode.
- Set the transfer clock to 80000 bps.
- Enable IIC1_ENDI/IIC1_WUI interrupt.

(4) Initialize the LCD module
- Set to 8 bits, bus mode, 2-line display, and font type 5x8 dots.
- Make settings to enable display indication, disable cursor display, and disable cursor blinking.
- Set the cursor shift direction to right.

(5) Control the LEDs and perform communication with the LCD module according to interrupts of the RTC.

Note 1. Refer to the RISC-V User’s Manual: Hardware for usage notes concerning this device.

Note 2. Make initial settings for the RTC only when a power-on reset is generated. To enable this, some codes generated by the smart configurator are modified for source codes. Note that, if the smart configurator re-generates codes, this modification becomes invalid. The constants defined in r_cg_userdefine.h are used for the RTC’s clock time and the initial set value of alarm time.

2. Operation Confirmation Conditions

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

Table 2.1 Operation Confirmation Conditions

Item	Description
MCU used	RISC-V (R9A02G021)
Board used	RISC-V-48p Fast Prototyping Board (RTK9FPG021S000W0BJ)
Operating frequency	High-speed on-chip oscillator clock (HOCO): 32 MHz Subsystem clock (XT1 clock (f_{XT})): 32.768 kHz
Operating voltage	3.3 V (can be operated at 1.6 V to 5.5 V)
Integrated development environment (e^2studio)	e^2studio V2024-01.1 (24.1.1) from Renesas Electronics Corp.
C compiler (e^2studio)	LLVM for RISC-V 17.0.2.202401
Smart configurator (SC)	Smart Configurator for RISC-V V24.1.1.v20240125-1623
Board support package (BSP)	V1.00 from Renesas Electronics Corp.
LCD module	WayinTop 1602

3. Hardware Descriptions

3.1 Example of Hardware Configuration

Figure 3.1 shows an example of the hardware configuration used in the application note.

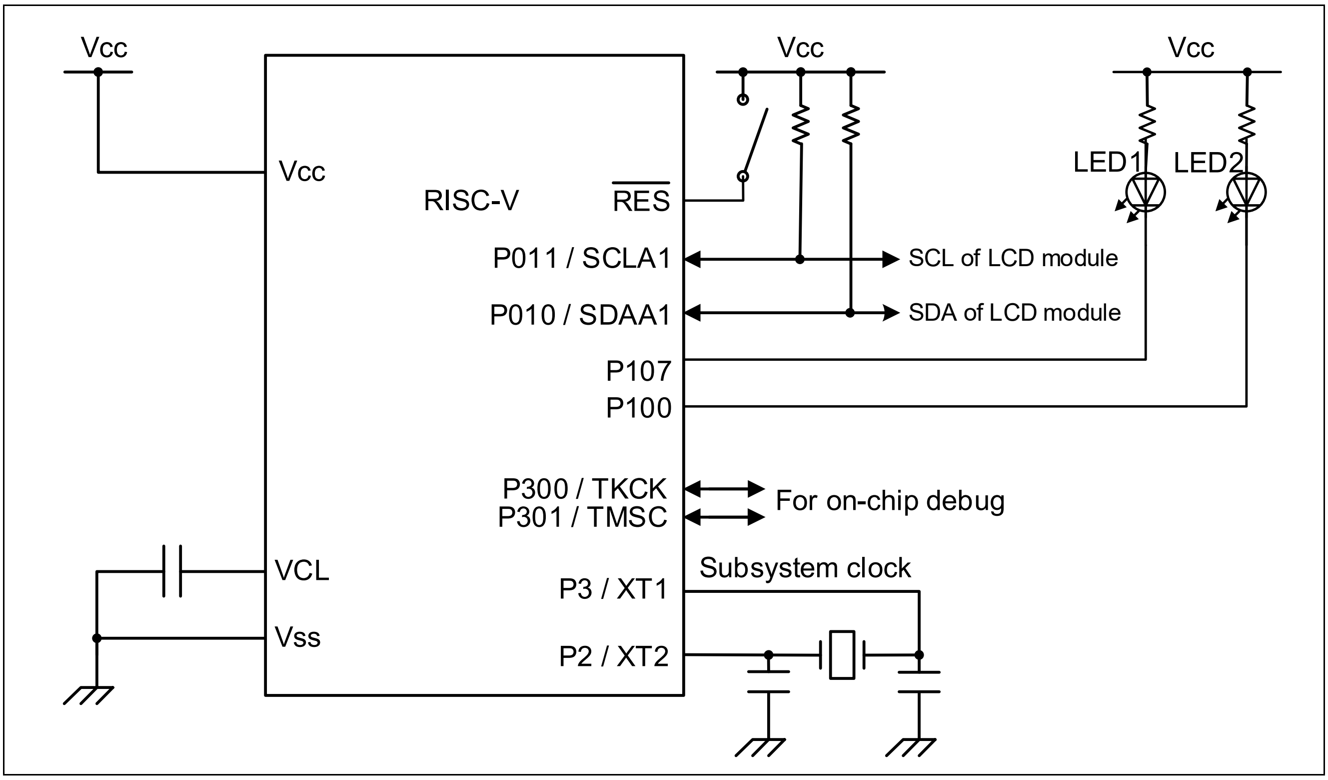

Figure 3.1 Hardware Configuration

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

Note 2. V_{CC} must not be lower than the reset release voltage (V_{LVD0}) that is specified for the LVD0.

3.2 List of Pins to be Used

Table 3.1 lists the pins to be used and their functions.

Table 3.1 Pins to be Used and Their Functions

Pin name	I/O	Function
P107	Output	LED1 control
P100	Output	LED2 control
P011 / SCLA1, P010 / SDAA1	Input/ Output	I^2C communication with LCD module
$\overline{\text{RES}}$	Input	External reset input
P3 / XT1, P2 / XT2	Input	RTC operating clock

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

4. Software Explanation

4.1 Setting of Option Byte

Table 4.1 shows the option byte settings.

Table 4.1 Option Byte Settings

Address	Setting Value	Contents
0000_0400H	FFFF_FFFFH	Disables the watchdog timer. (Counting stopped after reset)
0000_0404H	FFFF_AFFFH	High-speed on-chip oscillator clock : 32 MHz
0101_0008H	FFFF_FFFFH	Enables on-chip debugging

4.2 List of Constants

Table 4.2 and lists the constants that are used in the sample code.

Table 4.2 Constants

Constant Name	Setting Value	Description
SLAVE_ADDRESS_LCD	0x27	Slave address for LCM commands
R_RTC_INIT_SEC	0x55	Initial value of the current clock time (second)
R_RTC_INIT_MIN	0x59	Initial value of the current clock time (minute)
R_RTC_INIT_HOUR	0x15	Initial value of the current clock time (hour)
R_RTC_INIT_WEEK	0x00	Initial value of the current clock time (day of the week)
R_RTC_INIT_DAY	0x01	Initial value of the current clock time (day)
R_RTC_INIT_MONTH	0x01	Initial value of the current clock time (month)
R_RTC_INIT_YEAR	0x24	Initial value of the current clock time (year)
R_RTC_ALARM_MIN	0x00	Set value of alarm occurrence time (minute)
R_RTC_ALARM_HOUR	0x16	Set value of alarm occurrence time (hour)
R_RTC_ALARM_WEEK	0x7F	Set value of alarm occurrence time (day of the week)
R_INTERRUPT_OFF	0	Interrupt flag cleared
R_INTERRUPT_ON	1	Interrupt flag ON

4.3 List of Variables

Table 4.3 lists global variables.

Table 4.3 Global Variables

Type	Variable Name	Description	Function Used
uint8_t	g_rtc_constperiod	Fixed-cycle interrupt flag	r_Config_RTC_Create_UserInit, r_Config_RTC_callback_constperiod, r_rtc_is_constperiod_flag_on, r_rtc_clear_constperiod_flag

4.4 List of Functions

Table 4.4 shows a list of functions.

Table 4.4 Functions

Function Name	Outline
R_Config_RTC_Create_UserInit()	User-specified RTC initialization processing
r_Config_RTC_callback_alarm()	Alarm interrupt processing.
r_Config_RTC_callback_constperiod()	Fixed-cycle interrupt processing.
r_rtc_init_current_time()	Initial setting for the current clock time
r_rtc_init_alarm_time()	Initial setting for alarm time
r_rtc_is_constperiod_flag_on()	Check the fixed-cycle interrupt flag.
r_rtc_clear_constperiod_flag()	Clear the fixed-cycle interrupt flag.
r_rtc_display_current_time()	Display the current clock time on the LCD.
convert_bcd_to_2chars()	Convert 2-digit bcd to two characters.
convert_week_to_3chars()	Convert day-of-the-week code to three characters.
r_Config_IICA1_callback_master_sendend()	IICA1 send end callback processing.
r_Config_IICA1_callback_master_error()	IICA1 error callback processing.
R_Config_PORT_Create_UserInit()	User-specified port initialization processing
i2c_lcd_init()	LCD module initialization.
lcd_clear()	LCD module display clear processing
i2c_lcd_send_string()	LCD module character string transmission processing
lcd_send_cmd()	LCD module command transmission processing
lcd_send_data()	LCD module data transmission processing
2c_lcd_put_cur()	LCD module communication end flag setting

4.5 Specification of Function

The function specifications of the sample code are shown below.

R_Config_RTC_Create_UserInit()	
Outline	User-specified RTC initialization processing
Header	r_cg_userdefine.h
Declaration	void R_Config_RTC_Create_UserInit(void);
Description	Perform user-specified processing among initialization required before starting the RTC.
Argument	None
Return Value	None

r_Config_RTC_callback_alarm()	
Outline	Alarm interrupt processing.
Header	r_cg_userdefine.h
Declaration	void r_Config_RTC_callback_alarm(void);
Description	This callback function is called when an RTC alarm interrupt is generated.
Argument	None
Return Value	None

r_Config_RTC_callback_constperiod()	
Outline	Fixed-cycle interrupt processing.
Header	r_cg_macrodriver.h, r_cg_userdefine.h
Declaration	void r_Config_RTC_callback_constperiod(void);
Description	This callback function is called when an RTC fixed-cycle interrupt is generated.
Argument	None
Return Value	None

r_rtc_init_current_time()	
Outline	Initial setting for the current clock time
Header	r_cg_userdefine.h
Declaration	void r_rtc_init_current_time(void);
Description	Set the current clock time in the RTC register.
Argument	None
Return Value	None

r_rtc_init_alarm_time()	
Outline	Initial setting for the alarm clock time
Header	r_cg_userdefine.h, Config_RTC.h
Declaration	void r_rtc_init_alarm_time(void);
Description	Set the alarm time in the RTC register.
Argument	None
Return Value	None

r_rtc_is_constperiod_flag_on()

Outline	Check the fixed-cycle interrupt flag.
Header	r_cg_userdefine.h
Declaration	uint8_t r_rtc_is_constperiod_flag_on(void);
Description	Check g_ rtc_constperiod and return the check result.
Argument	None
Return Value	1: g_rtc_constperiod is R_INTTERUPT_ON 0: g_ rtc_constperiod is R_INTERRUPT_OFF

r_rtc_clear_constperiod_flag()

Outline	Clear the fixed-cycle interrupt flag.
Header	r_cg_userdefine.h
Declaration	void r_rtc_clear_constperiod_flag(void);
Description	Clear g_ rtc_constperiod (set to R_INTERRUPT_OF).
Argument	None
Return Value	None

convert_bcd_to_2chars()

Outline	Conversion of two-digit bcd value to two characters
Header	Config_RTC.h
Declaration	static void convert_bcd_to_2chars(uint8_t bcd, uint8_t * const str);
Description	Convert a two-digit bcd value to two characters.
Argument	uint8_t bcd: Two-digit bcd value to be converted (0x00 to 0x99) uint8_t *str: Area to store converted characters ('\0' is not added to the end.)
Return Value	None

convert_week_to_3chars()

Outline	Conversion of day-of-the-week value to three characters
Header	Config_RTC.h
Declaration	static void convert_week_to_3chars(uint8_t week, uint8_t * const str);
Description	Convert a day-of-the-week value to three characters. The conversion result is as follows:

Source value	0	1	2	3	4	5	6
Conversion result	Sun	Mon	Tue	Wed	Thu	Fri	Sat

Argument	uint8_t week: Day-of-the-week value to be converted (0 to 6) uint8_t *str: Area to store converted characters ('\0' is not added to the end.)
Return Value	None

r_Config_IICA1_callback_master_sendend()	
Outline	IICA1 send end callback processing.
Header	r_cg_macrodriver.h, Config_IICA1.h
Declaration	static void r_Config_IICA1_callback_master_receiveend(void);
Description	Callback function called when an IICA1 transmission completion interrupt is generated Generate stop conditions, and then call the LCD module communication end flag setting function.
Argument	None
Return Value	None

r_Config_IICA1_callback_master_error()	
Outline	IICA1 error callback processing.
Header	r_cg_macrodriver.h
Declaration	static void r_Config_IICA1_callback_master_error(MD_STATUS flag);
Description	Callback function called when an IICA1 transmission error interrupt is generated Call the LCD module communication end flag setting function.
Argument	MD_STATUS flag : error type
Return Value	None

R_Config_PORT_Create_UserInit()	
Outline	User-specified port initialization processing
Header	r_cg_macrodriver.h, r_cg_userdefine.h
Declaration	void R_Config_PORT_Create_UserInit(void);
Description	Perform user-specified processing among initialization required before using the port.
Argument	None
Return Value	None

i2c_lcd_init()	
Outline	LCD module Initialization
Header	simple_i2c_lcd.h
Declaration	void i2c_lcd_init(void);
Description	Initializes LCD module.
Argument	None
Return Value	None

lcd_clear()	
Outline	LCD module display clear processing
Header	simple_i2c_lcd.h
Declaration	void lcd_clear(void);
Description	Transmit the display clear command to the LCD module.
Argument	None
Return Value	None

i2c_lcd_send_string()	
Outline	LCD module character string transmission processing
Header	simple_i2c_lcd.h
Declaration	void i2c_lcd_send_string (uint8_t *str);
Description	Display the character string transferred with str on the LCD module. uint8_t *str: Character string to be displayed.
Argument	uint8_t *str: string_time[20]
Return Value	None

lcd_send_cmd()	
Outline	LCD module command transmission processing
Header	simple_i2c_lcd.h, Config_IICA1.h, r_cg_macrodriver.h
Declaration	void lcd_send_cmd (uint8_t cmd);
Description	Send the command transferred with command to the LCD module.
Argument	uint8_t cmd: Command to be sent to the LCD module
Return Value	None

lcd_send_data()	
Outline	LCD module data transmission processing
Header	simple_i2c_lcd.h, Config_IICA1.h, r_cg_macrodriver.h
Declaration	void lcd_send_data (uint8_t data);
Description	Send the data transferred with data to the LCD module.
Argument	uint8_t data: Data to be sent to the LCD module
Return Value	None

i2c_lcd_put_cur()	
Outline	LCD module communication end flag setting
Header	simple_i2c_lcd.h
Declaration	void i2c_lcd_put_cur (int row, int col);
Description	Send data to the LCD module
Argument	None
Return Value	None

4.6 Flowcharts

4.6.1 Main Processing

Figure 4.1 shows the flowchart of the main processing.

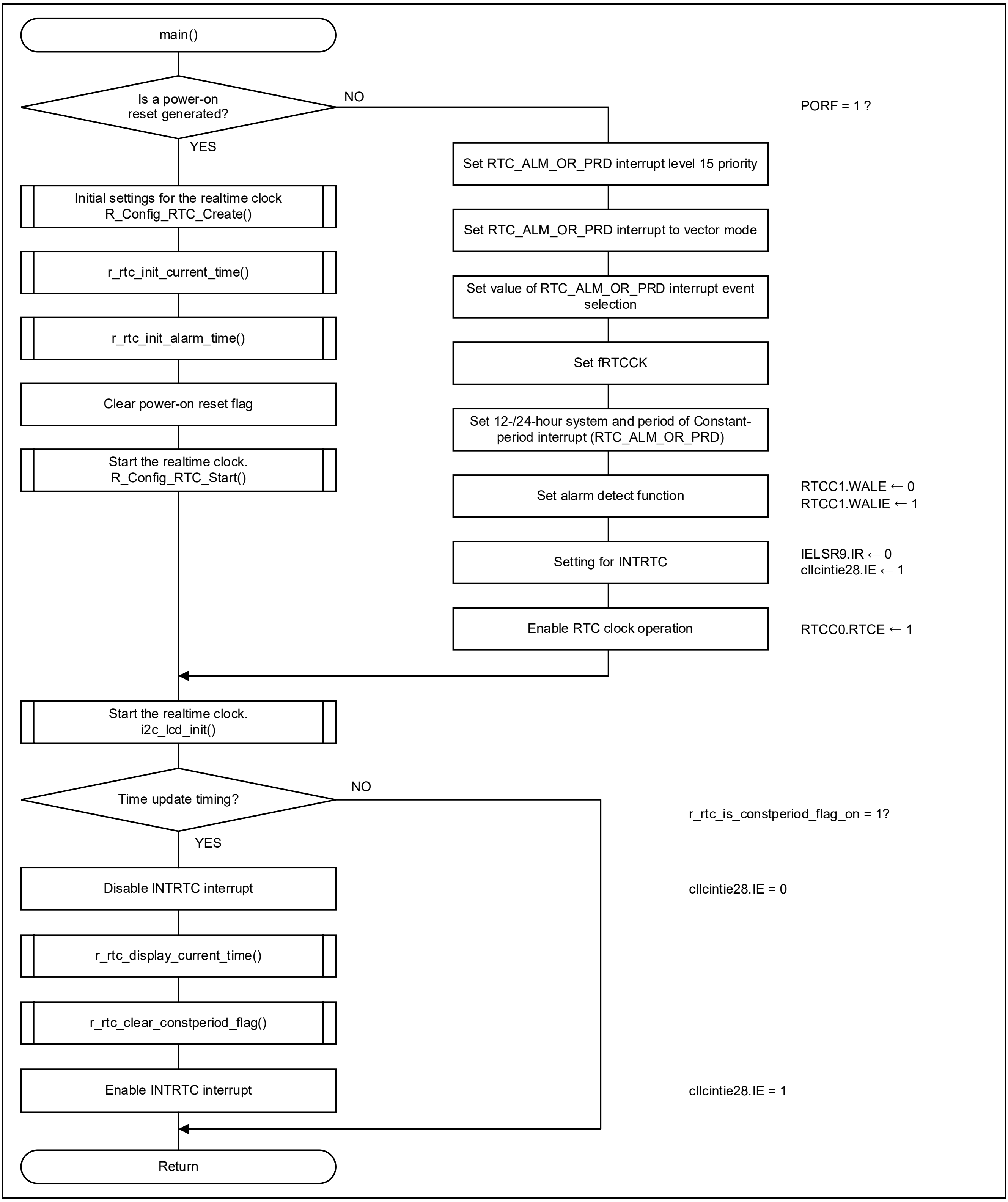

Figure 4.1 main Processing

4.6.2 RTC Initialization Processing (User-Defined)

Figure 4.2 shows the flowchart of the user-defined RTC initialization processing.

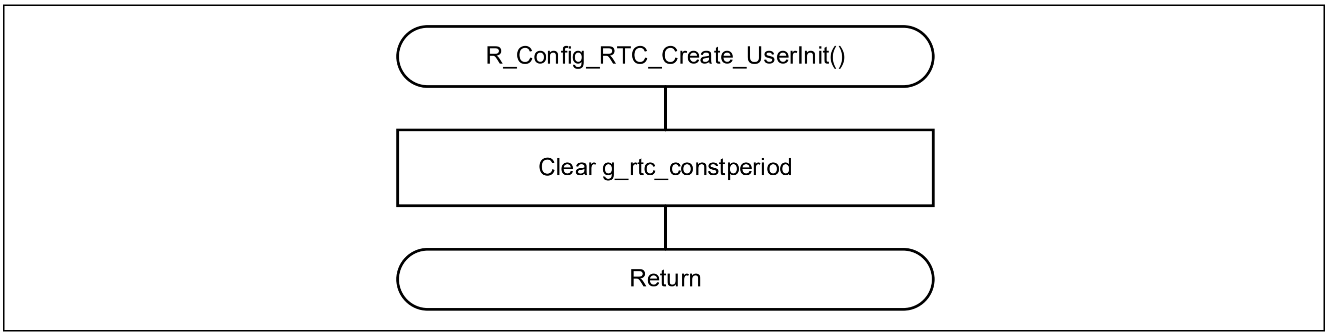

Figure 4.2 RTC Initialization Processing (User-Defined)

4.6.3 Alarm Interrupt Processing

Figure 4.3 shows the flowchart of the alarm interrupt processing.

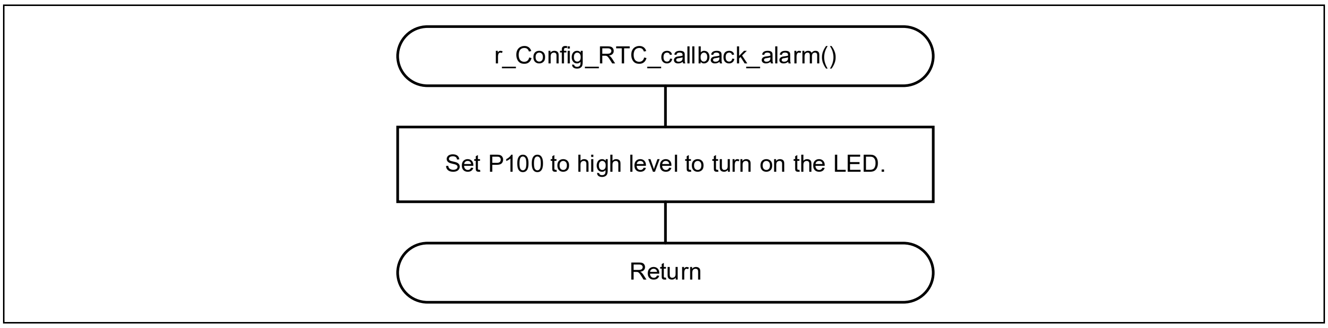

Figure 4.3 Alarm Interrupt Processing

4.6.4 Fixed-cycle Interrupt Processing

Figure 4.4 shows the flowchart of the fixed-cycle interrupt processing.

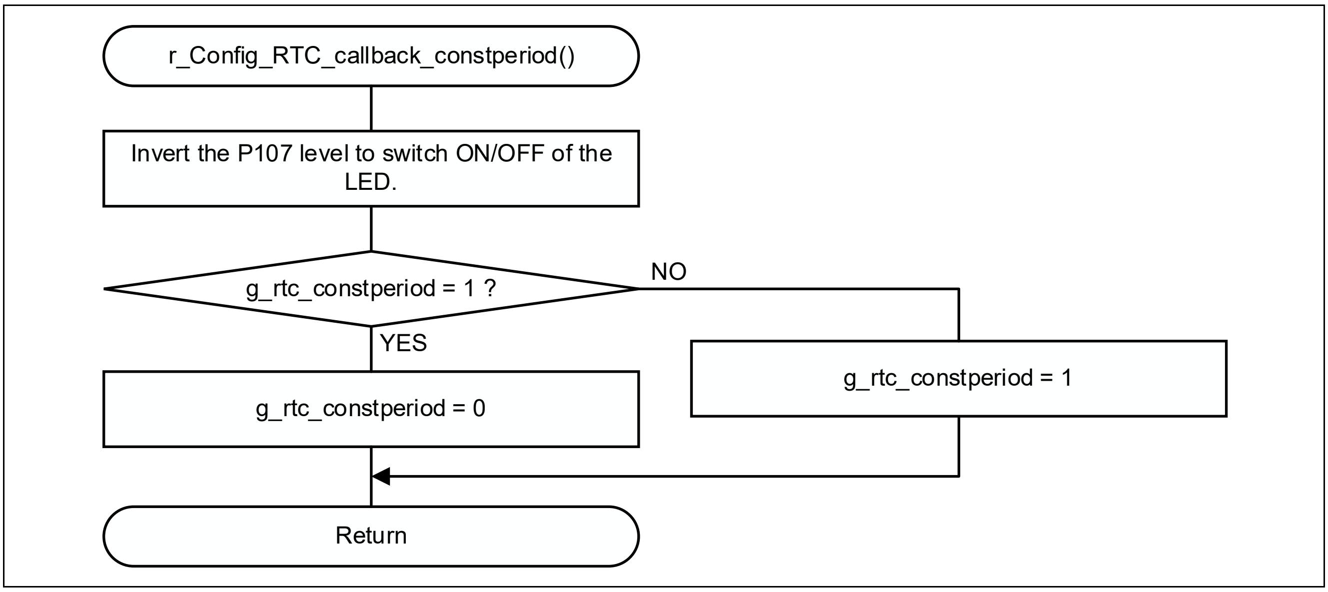

Figure 4.4 Fixed-cycle Interrupt Processing

4.6.5 Initial Settings for the Current Clock Time

Figure 4.5 shows the flowchart of the initial settings for the current clock time.

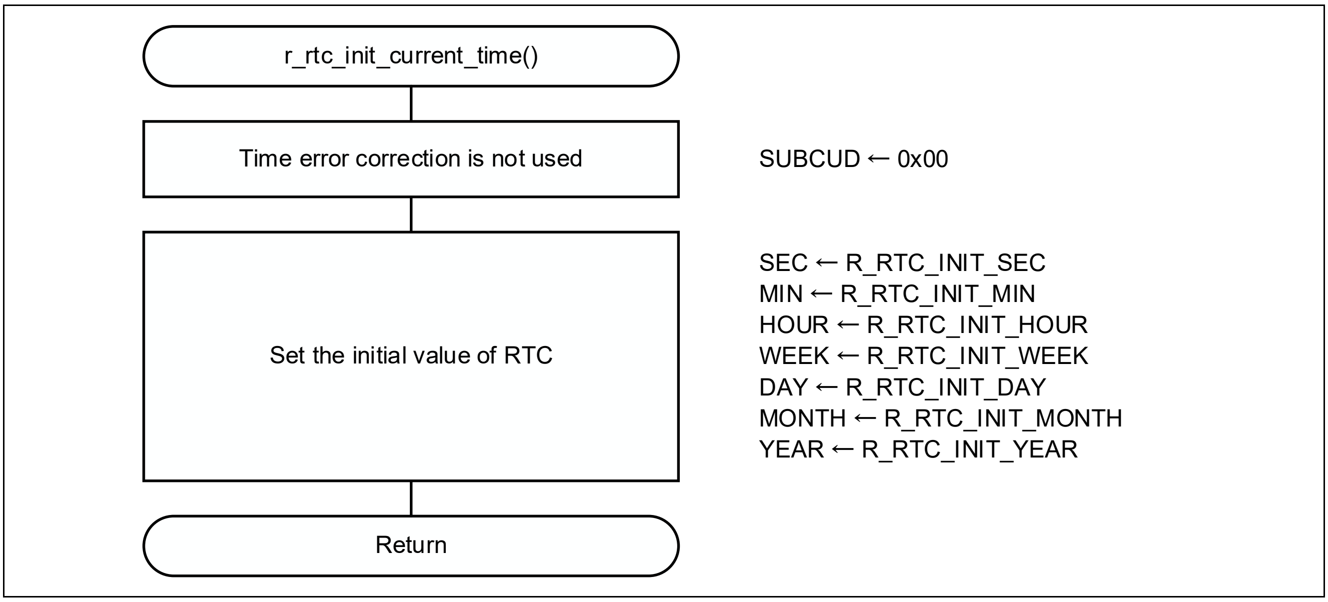

Figure 4.5 Initial Settings for the Current Clock Time

4.6.6 Initial Settings for Alarm Time

Figure 4.6 shows the flowchart of the initial settings for alarm time.

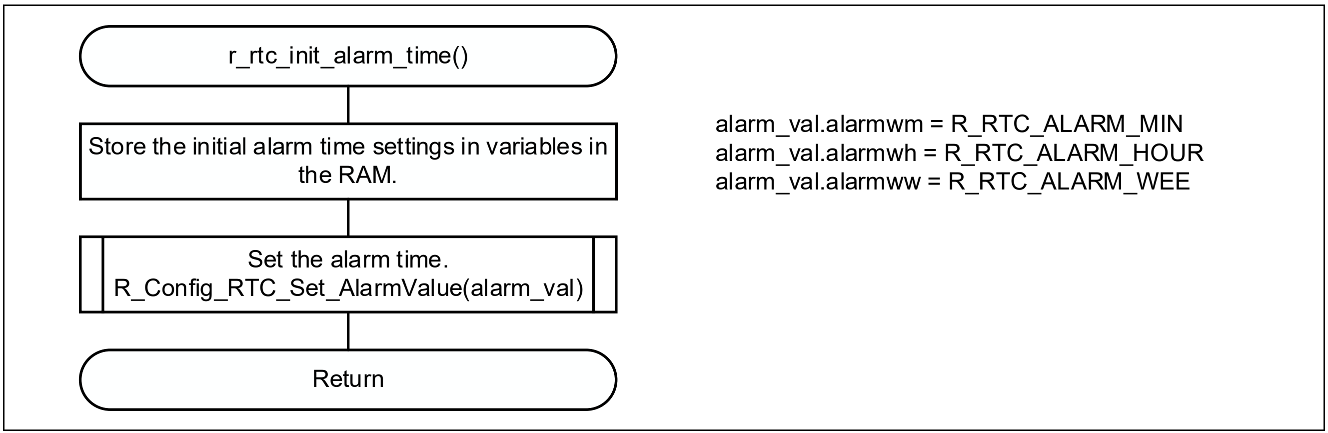

Figure 4.6 Initial Settings for Alarm Time

4.6.7 Check the Fixed-Cycle Interrupt Flag

Figure 4.7 shows the flowchart of the check the fixed-cycle interrupt flag.

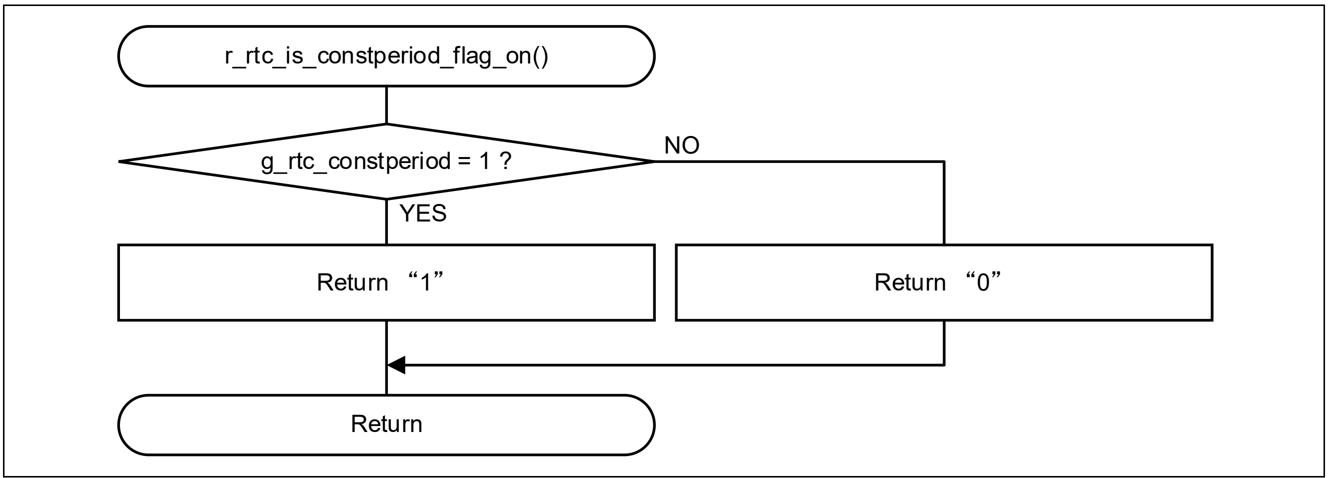

Figure 4.7 Check the Fixed-Cycle Interrupt Flag

4.6.8 Clear the Fixed-Cycle Interrupt Flag

Figure 4.8 shows the flowchart of the clear the fixed-cycle interrupt flag.

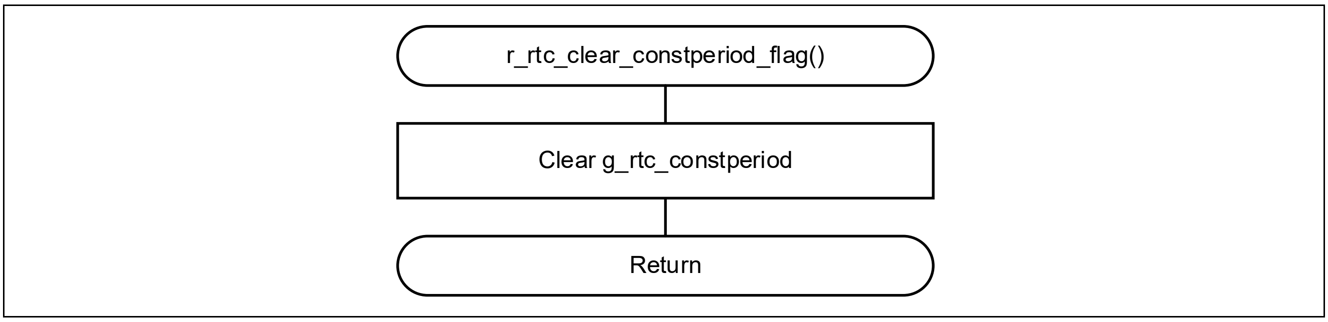

Figure 4.8 Clear the Fixed-Cycle Interrupt Flag

4.6.9 Current Clock Time Display Processing

Figure 4.9 and Figure 4.10 show the flowcharts of the current clock time display processing.

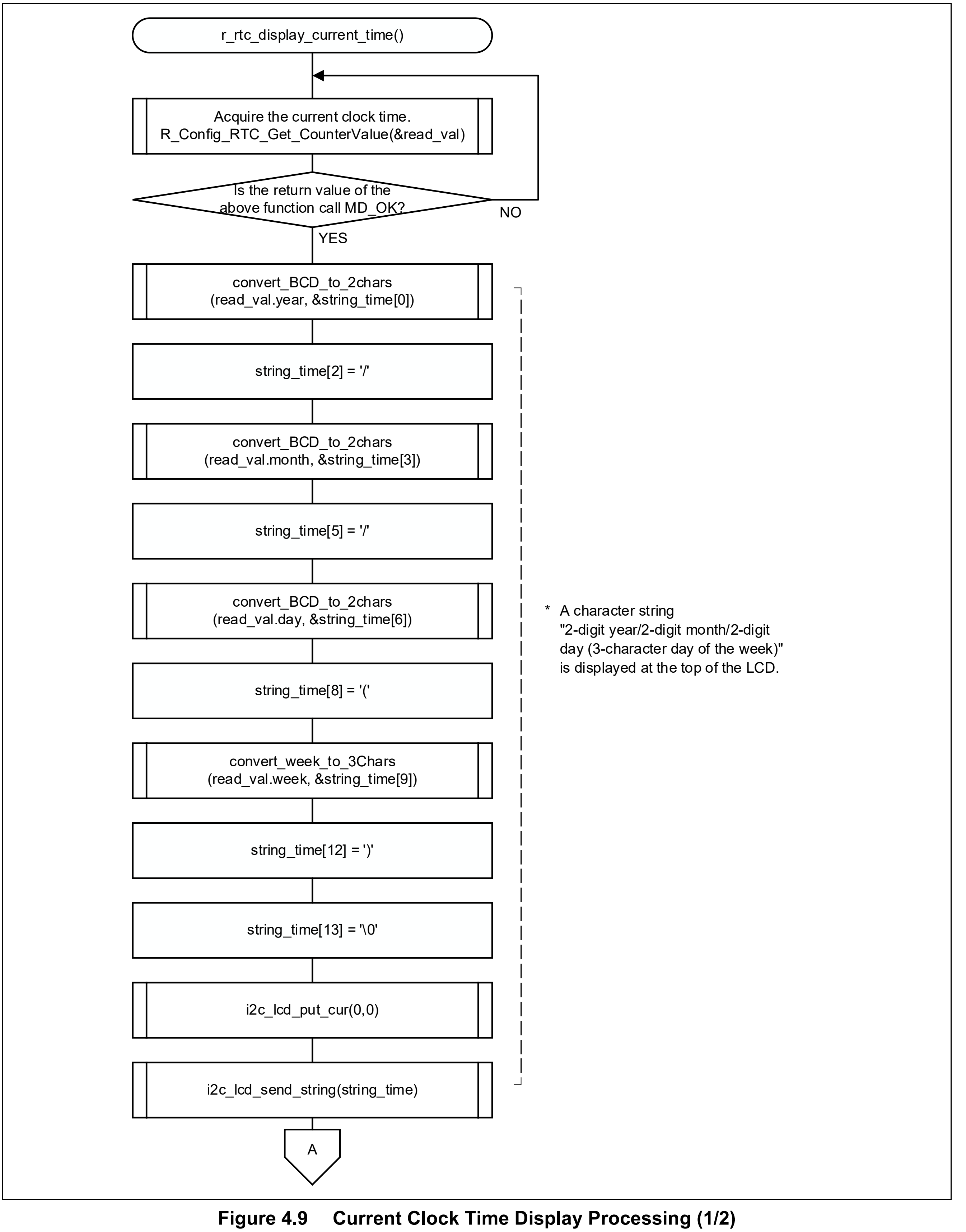

Figure 4.9 Current Clock Time Display Processing (1/2)

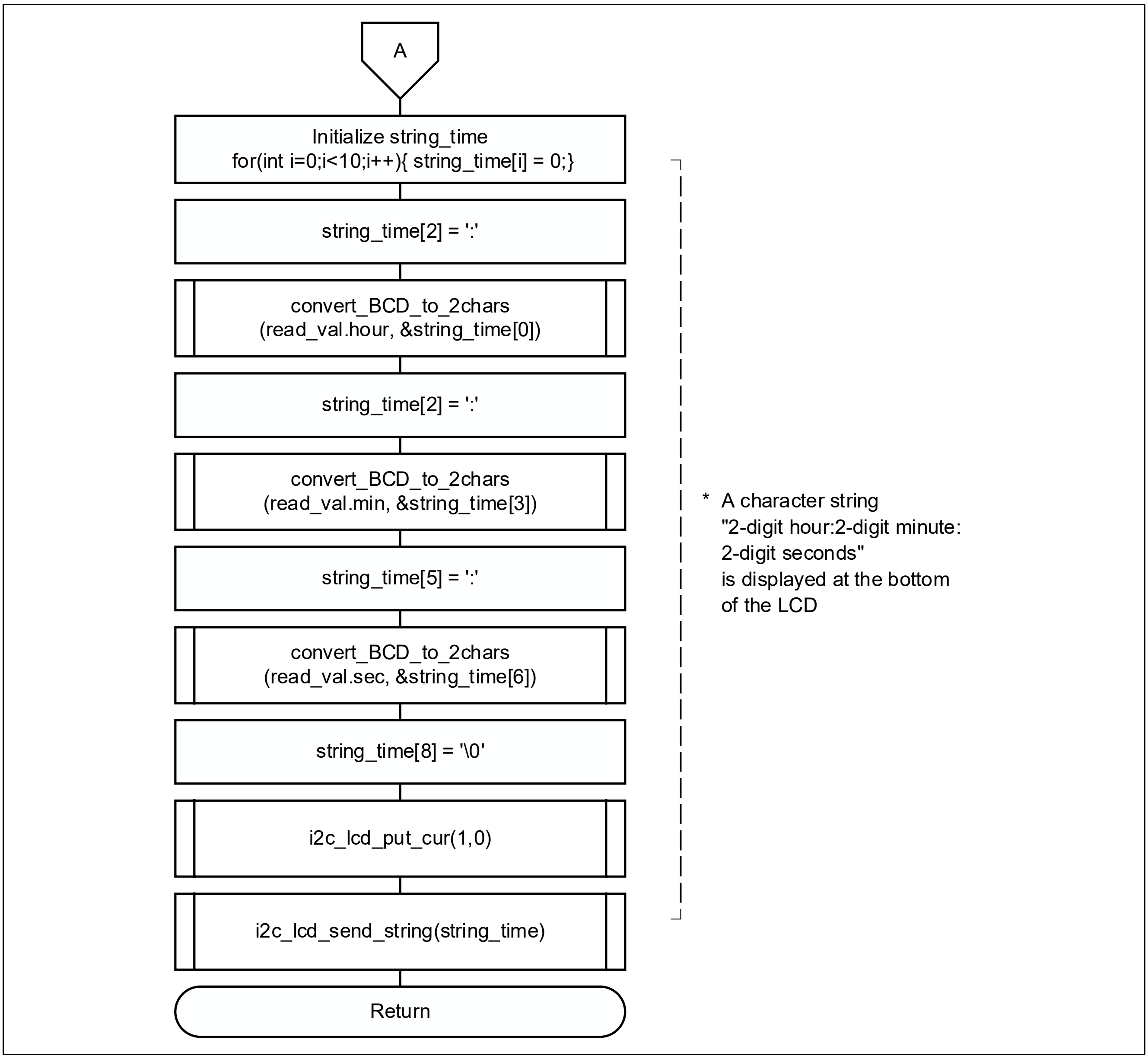

Figure 4.10 Current Clock Time Display Processing (2/2)

4.6.10 Bcd-to-Character Conversion Processing

Figure 4.11 shows the flowchart of the processing for conversion from bcd to character.

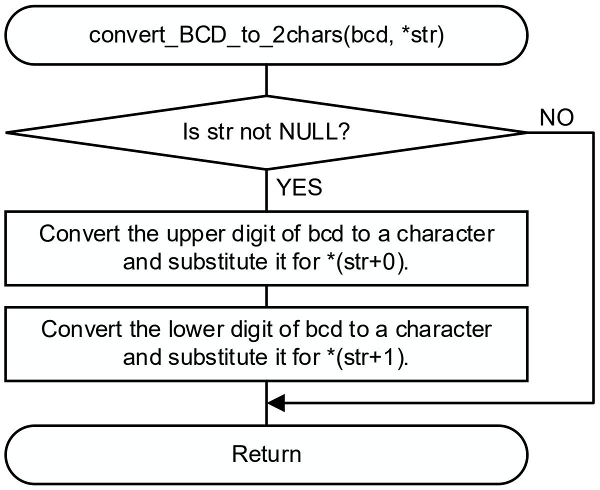

Figure 4.11 Bcd-to-Character Conversion Processing

4.6.11 Day of the Week-to-Character Conversion Processing

Figure 4.12 shows the flowchart of the processing for conversion from day of the week to character.

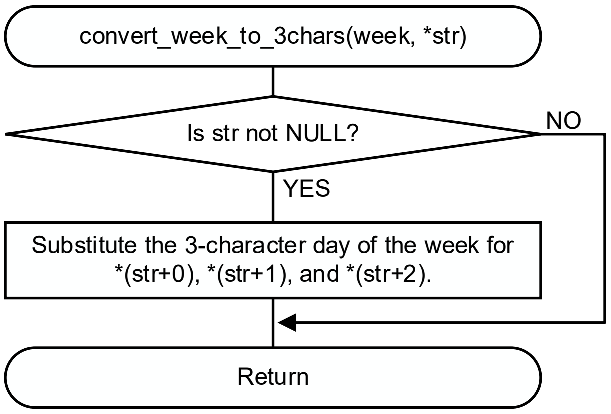

Figure 4.12 Day of the Week-to-Character Conversion Processing

4.6.12 IICA1 Send End Callback Processing

Figure 4.13 shows the flowchart of the IICA1 send end callback processing.

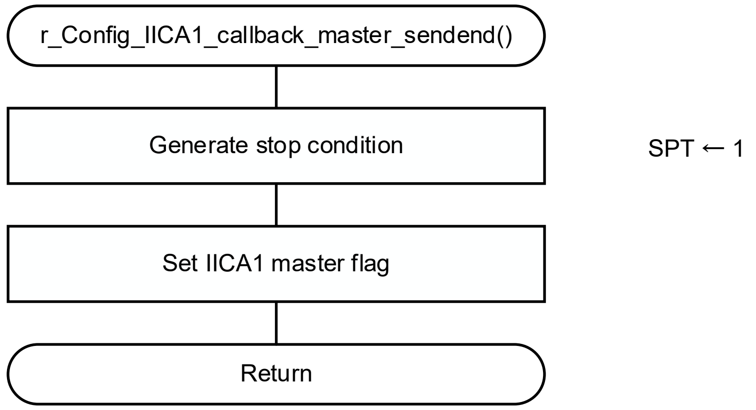

Figure 4.13 IICA1 Send End Callback Processing

4.6.13 IICA1 Error Callback Processing

Figure 4.14 shows the flowchart of the IICA1 error callback processing.

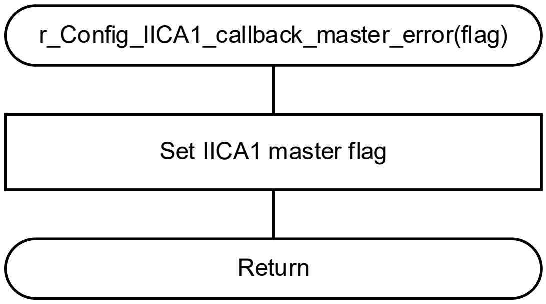

Figure 4.14 IICA1 Error Callback Processing

4.6.14 Port Initialization Processing (User-Defined)

Figure 4.15 shows the flowchart of the user-defined port initialization processing.

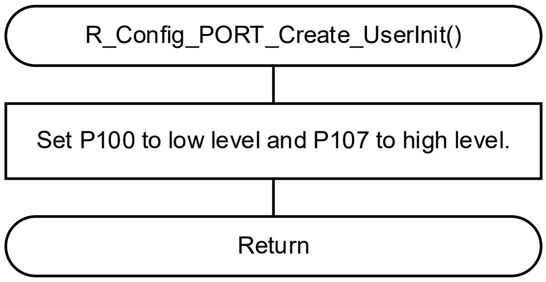

Figure 4.15 Port Initialization Processing (User-Defined)

4.6.15 LCD Module Initialization

Figure 4.16 shows the flowchart of the LCD module initialization.

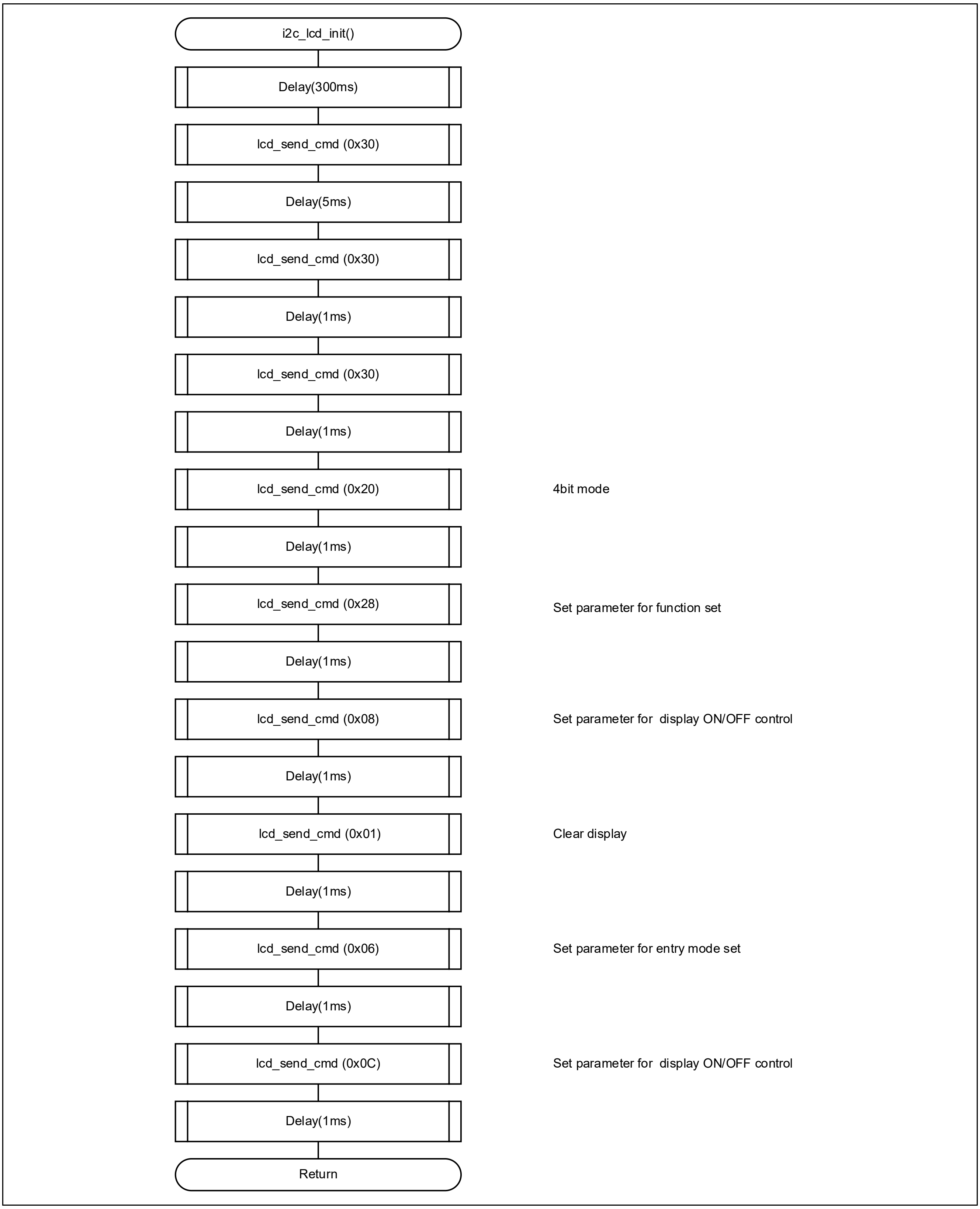

Figure 4.16 LCD Module Initialization

4.6.16 LCD Module Display Clear Processing

Figure 4.17 shows the flowchart of the display clear processing for the LCD module.

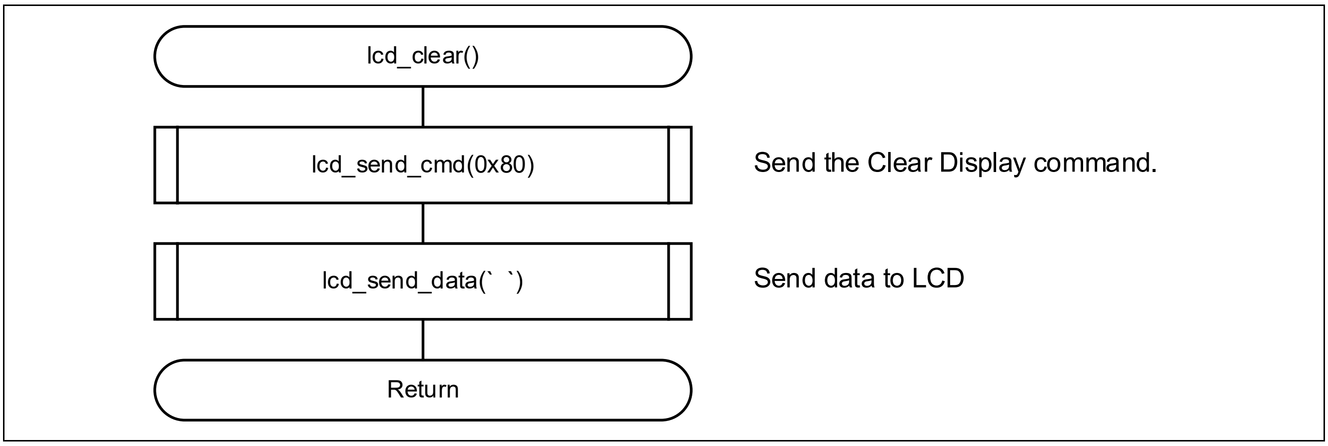

Figure 4.17 LCD Module Display Clear Processing

4.6.17 LCD Module Character String Transmission Processing

Figure 4.18 shows the flowchart of the character string transmission processing for the LCD module.

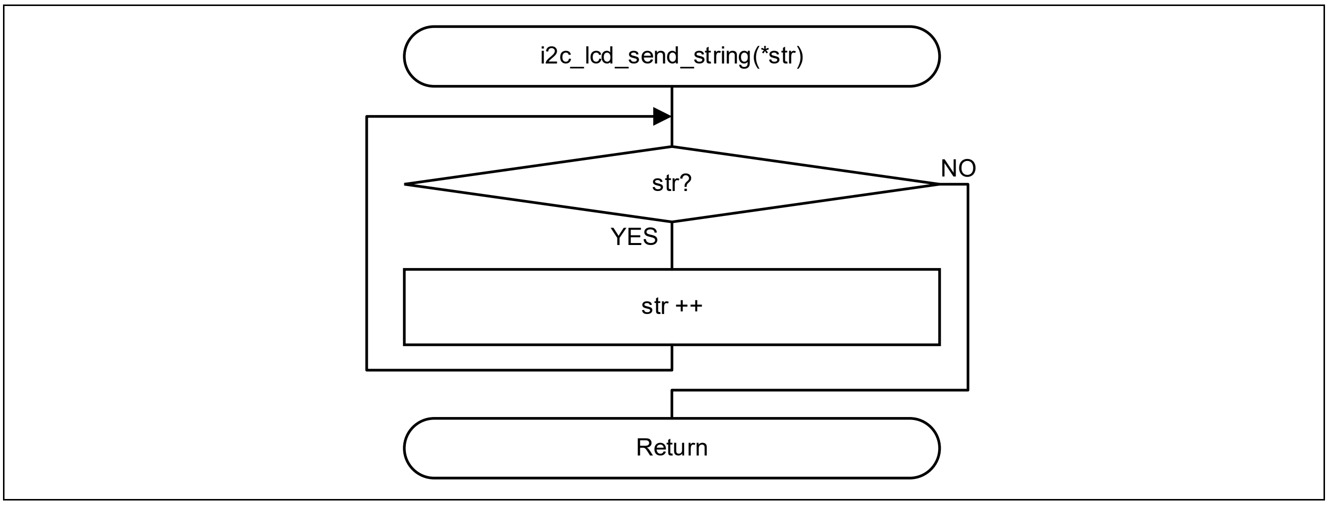

Figure 4.18 LCD Module Character String Transmission Processing

4.6.18 LCD Module Command Transmission Processing

Figure 4.19 shows the flowchart of the command transmission processing for the LCD module.

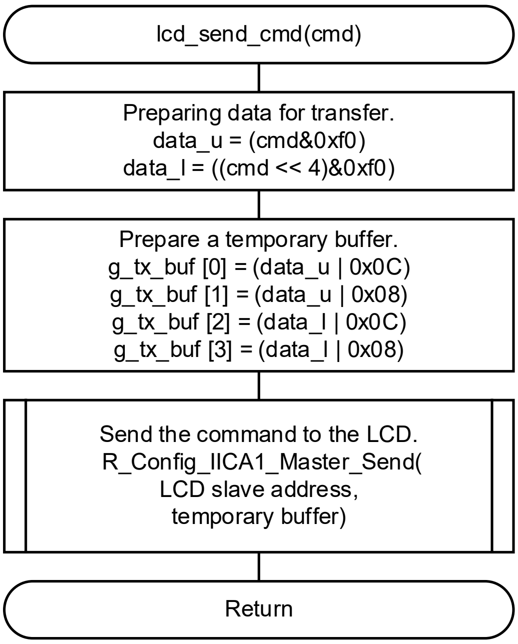

Figure 4.19 LCD Module Command Transmission Processing

4.6.19 LCD Module Data Transmission Processing

Figure 4.20 shows the flowchart of the data transmission processing for the LCD module.

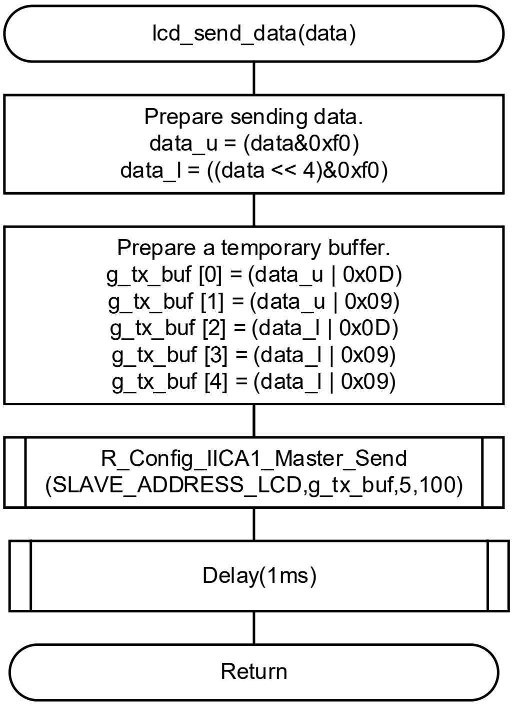

Figure 4.20 LCD Module Data Transmission Processing

4.6.20 LCD Module Communication End Flag Setting

Figure 4.21 shows the flowchart for setting the communication end flag for the LCD module.

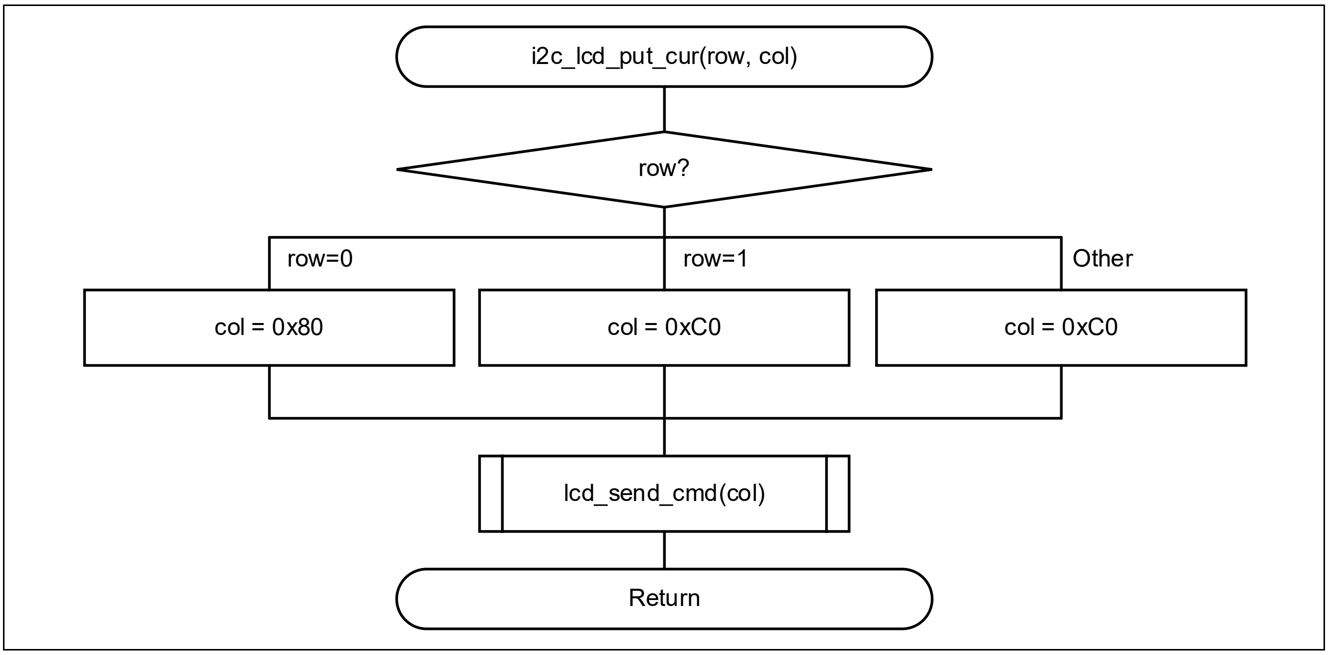

Figure 4.21 LCD Module Communication End Flag Setting

5. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

6. Reference Documents

RISC-V User's Manual: Hardware (R01UH1036EJ)
The latest versions can be downloaded from the Renesas Electronics website.

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar.18.24	—	Initial release

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Corporate Headquarters

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

Trademarks

Contact information

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