RL78/G23

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
RL78/G23

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 .......................................................................................................................... 3
   1.1 Overview of Specifications ...................................................................................................................... 3
   1.2 Outline of operation ................................................................................................................................. 4

2. Operation Confirmation Conditions ..........................................................................................  5

3. Hardware Descriptions ............................................................................................................ 6
   3.1 Example of Hardware Configuration ....................................................................................................... 6
   3.2 List of Pins to be Used ............................................................................................................................ 6

4. Software Explanation ............................................................................................................... 7
   4.1 Setting of Option Byte ............................................................................................................................. 7
   4.2 List of Constants ...................................................................................................................................... 7
   4.3 List of Variables ....................................................................................................................................... 9
   4.4 List of Functions .................................................................................................................................... 9
   4.5 Specification of Function ....................................................................................................................... 10
   4.6 Flowcharts ............................................................................................................................................. 14
      4.6.1 Main Processing .................................................................................................................................. 14
      4.6.2 RTC Initialization Processing (User-Defined) ...................................................................................... 15
      4.6.3 Alarm Interrupt Processing .................................................................................................................. 15
      4.6.4 Fixed-cycle Interrupt Processing ......................................................................................................... 16
      4.6.5 Initial Settings for the Current Clock Time ........................................................................................... 16
      4.6.6 Initial Settings for Alarm Time ............................................................................................................. 17
      4.6.7 Check the Fixed-Cycle Interrupt Flag .................................................................................................. 17
      4.6.8 Clear the Fixed-Cycle Interrupt Flag ................................................................................................... 18
      4.6.9 Current Clock Time Display Processing .............................................................................................. 19
      4.6.10 BCD-to-Character Conversion Processing ......................................................................................... 20
      4.6.11 Day of the Week-to-Character Conversion Processing ........................................................................ 21
      4.6.12 IICA0 Send End Callback Processing ................................................................................................. 21
      4.6.13 IICA0 Error Callback Processing ......................................................................................................... 22
      4.6.14 Port Initialization Processing (User-Defined) ...................................................................................... 22
      4.6.15 LCD Module Initialization ..................................................................................................................... 23
      4.6.16 LCD Module Display Clear Processing ............................................................................................... 23
      4.6.17 LCD Module Character String Transmission Processing .................................................................... 24
      4.6.18 LCD Module Command Transmission Processing ............................................................................. 25
      4.6.19 LCD Module Data Transmission Processing ..................................................................................... 25
      4.6.20 LCD Module Communication End Flag Setting ................................................................................... 26
      4.6.21 LCD Module Communication End Wait Processing ............................................................................ 26

5. Sample Code ......................................................................................................................... 27

6. Reference Documents ............................................................................................................. 27

Revision History ............................................................................................................................ 28
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>
<thead>
<tr>
<th>Peripheral Function</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Realtime clock</td>
<td>RTC interrupt (INTRTC)</td>
</tr>
<tr>
<td>P53</td>
<td>Set to port output in the fixed-cycle interrupt processing (inverted output)</td>
</tr>
<tr>
<td>P52</td>
<td>Set to port output in the alarm interrupt processing (high level output)</td>
</tr>
<tr>
<td>Serial Interface IICA0</td>
<td>I2C communication with the LCD module (LCM)</td>
</tr>
<tr>
<td>P60/SCLA0, P61/SDAA0</td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>External reset input</td>
</tr>
<tr>
<td>P123/XT1</td>
<td>RTC operating clock</td>
</tr>
</tbody>
</table>

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 “2021/1/1 (Fri) 15:59:55” and the alarm time is set to “16:00:00 every day”. Furthermore, the following interrupt processing is performed.

- P53 output inversion and clock time display on the LCD in the fixed-cycle interrupt processing
- Low-level output of P52 (LED ON) in the alarm interrupt processing

(1) Initialize the realtime clock (RTC)
- Select the subsystem clock (fsXr) 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 P53 to the output port for the fixed-cycle interrupt processing. (The LED is lit when the initial value is low level.)
- Set P52 to the output port for the alarm interrupt processing. (The LED is unlit when the initial value is high level.)

(3) Initialize the serial interface IICA
- Use IICA0 (P60 set to SCLA0 and P61 set to SDAA0).
- Select fCLK/2 at the IICA0 operation clock.
- Set the local address to 0x10.
- Set the standard mode as the operation mode.
- Set the transfer clock to 80000 bps.
- Enable INTIICA0 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 RL78/G23 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

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU used</td>
<td>RL78/G23 (R7F100GLG)</td>
</tr>
<tr>
<td>Board used</td>
<td>RL78/G23-64p Fast Prototyping Board (RTK7RLG230CLG000BJ)</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>● High-speed on-chip oscillator clock (fIH): 32 MHz</td>
</tr>
<tr>
<td></td>
<td>● Subsystem clock (XT1 clock (fXT)): 32.768 kHz</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>3.3 V (can be operated at 3.1 V to 3.5 V)</td>
</tr>
<tr>
<td></td>
<td>LVD0 operations (VLVD0): Reset mode</td>
</tr>
<tr>
<td></td>
<td>At rising edge TYP. 1.90 V (1.84 V to 1.95 V)</td>
</tr>
<tr>
<td></td>
<td>At falling edge TYP. 1.86 V (1.80 V to 1.91 V)</td>
</tr>
<tr>
<td>Integrated development environment (CS+)</td>
<td>CS+ for CC E8.05.00 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>C compiler (CS+)</td>
<td>CC-RL V1.10.00 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>Integrated development environment (e2studio)</td>
<td>e2studio V2021-04 (21.4.0) from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>C compiler (e2studio)</td>
<td>CC-RL V1.10.00 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>Integrated development environment (IAR)</td>
<td>IAR Embedded Workbench for Renesas RL78 V4.21.1 from IAR Systems Corp.</td>
</tr>
<tr>
<td>C compiler (IAR)</td>
<td>IAR C/C++ Compiler for Renesas RL78 V4.21.1.2260 from IAR Systems Corp.</td>
</tr>
<tr>
<td>Smart configurator (SC)</td>
<td>V1.0.1 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>Board support package (BSP)</td>
<td>V1.00 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>LCD module</td>
<td>ACM1602NI-FLW-FBW-M01</td>
</tr>
</tbody>
</table>
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](image)

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 VDD or VSS through a resistor).

Note 2. Connect any pins whose name begins with EVSS to VSS, and any pins whose name begins with EVDD to VDD, respectively.

Note 3. VDD must not be lower than the reset release voltage (VLVD0) 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>
<thead>
<tr>
<th>Pin name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P53</td>
<td>Output</td>
<td>LED1 control</td>
</tr>
<tr>
<td>P52</td>
<td>Output</td>
<td>LED2 control</td>
</tr>
<tr>
<td>P60 / SCLA0, P61 / SDAA0</td>
<td>Input/Output</td>
<td>I2C communication with LCD module</td>
</tr>
<tr>
<td>RESET†</td>
<td>Input</td>
<td>External reset input</td>
</tr>
<tr>
<td>P123 / X T1, P124 / XT2</td>
<td>Input</td>
<td>RTC operating clock</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Address</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>000C0H / 040C0H</td>
<td>11101111B</td>
<td>Disables the watchdog timer. (Counting stopped after reset)</td>
</tr>
<tr>
<td>000C1H / 040C1H</td>
<td>11111110B</td>
<td>LVD0 detection voltage: reset mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At rising edge TYP. 1.90 V (1.84 V to 1.95 V)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At falling edge TYP. 1.86 V (1.80 V to 1.91 V)</td>
</tr>
<tr>
<td>000C2H / 040C2H</td>
<td>11101000B</td>
<td>HS mode, High-speed on-chip oscillator clock ($f_{IH}$): 32 MHz</td>
</tr>
<tr>
<td>000C3H / 040C3H</td>
<td>10000100B</td>
<td>Enables on-chip debugging</td>
</tr>
</tbody>
</table>

4.2 List of Constants

Table 4-2 and Table 4-3 lists the constants that are used in the sample code.

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA0 LCM_SLAVE_ADDR</td>
<td>0xA0</td>
<td>Slave address for LCM commands</td>
</tr>
<tr>
<td>0x00 LCM_SLAVE_ADDR_RW_LOW</td>
<td>0x00</td>
<td>Data write flag</td>
</tr>
<tr>
<td>0x00 LCM_CONTROL_BYTE</td>
<td>0x00</td>
<td>Control byte for LCM commands</td>
</tr>
<tr>
<td>0x80 LCM_CONTROL_BYTE_RS_HIGH</td>
<td>0x80</td>
<td>Control byte: Data transfer</td>
</tr>
<tr>
<td>0x00 LCM_CONTROL_BYTE_RS_LOW</td>
<td>0x00</td>
<td>Control byte: Command transfer</td>
</tr>
<tr>
<td>0x00 LCM_COMMAND_CLEAR_DISPLAY</td>
<td>0x00</td>
<td>Command: Clear Display</td>
</tr>
<tr>
<td>0x04 LCM_COMMAND_ENTRY_MODE_SET</td>
<td>0x04</td>
<td>Command: Entry Mode Set</td>
</tr>
<tr>
<td>0x02 LCM_COMMAND_ENTRY_MODE_SET_ID_HIGH</td>
<td>0x02</td>
<td>Entry Mode Set: Address increment</td>
</tr>
<tr>
<td>0x00 LCM_COMMAND_ENTRY_MODE_SET_S_LOW</td>
<td>0x00</td>
<td>Entry Mode Set: Display shift OFF</td>
</tr>
<tr>
<td>0x08 LCM_COMMAND_DISPLAY_ONOFF</td>
<td>0x08</td>
<td>Command: Display ON/OFF Control</td>
</tr>
<tr>
<td>0x04 LCM_COMMAND_DISPLAY_ONOFF_D_HIGH</td>
<td>0x04</td>
<td>Display ON/OFF control: Display ON</td>
</tr>
<tr>
<td>0x00 LCM_COMMAND_DISPLAY_ONOFF_C_LOW</td>
<td>0x00</td>
<td>Display ON/OFF control: Cursor display OFF</td>
</tr>
<tr>
<td>0x00 LCM_COMMAND_DISPLAY_ONOFF_B_LOW</td>
<td>0x00</td>
<td>Display ON/OFF control: Cursor blinking OFF</td>
</tr>
<tr>
<td>0x20 LCM_COMMAND_FUNCTION_SET</td>
<td>0x20</td>
<td>Command: Function Set</td>
</tr>
<tr>
<td>0x10 LCM_COMMAND_FUNCTION_SET_DL_HIGH</td>
<td>0x10</td>
<td>Function Set: Mpu 8-bit bus mode</td>
</tr>
<tr>
<td>0x08 LCM_COMMAND_FUNCTION_SET_N_HIGH</td>
<td>0x08</td>
<td>Function Set: 2-line display</td>
</tr>
<tr>
<td>0x00 LCM_COMMAND_FUNCTION_SET_F_LOW</td>
<td>0x00</td>
<td>Function Set: 5x8-dot font</td>
</tr>
<tr>
<td>0x80 LCM_COMMAND_SET_DDRAM_ADDRESS</td>
<td>0x80</td>
<td>Command: Set DDRAM Address</td>
</tr>
<tr>
<td>Constant Name</td>
<td>Setting Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LCM_COMMAND_EXEC_WAIT</td>
<td>26600</td>
<td>LCD module command execution wait time 5 ms (32 MHz operation)</td>
</tr>
<tr>
<td>LCM_CONFIG_FUNCTION_SET_PARAMS</td>
<td>0x18</td>
<td>Parameters for function set: 0x10_LCM_COMMAND_FUNCTION_SET_DL_HIGH</td>
</tr>
<tr>
<td>LCM_CONFIG_ENTRY_MODE_SET_PARAMS</td>
<td>0x20</td>
<td>Parameters for entry mode set: 0x02_LCM_COMMAND_ENTRY_MODE_SET_ID_HIGH</td>
</tr>
<tr>
<td>LCM_CONFIG_DISPLAY_ONOFF_PARAMS</td>
<td>0x40</td>
<td>Display ON/OFF control command parameters: 0x04_LCM_COMMAND_DISPLAY_ONOFF_D_HIGH</td>
</tr>
<tr>
<td>LCM_CONFIG_MAX_CHAR_PER_LINE</td>
<td>16</td>
<td>Maximum number of characters per line</td>
</tr>
<tr>
<td>LCM_CONFIG_WAIT_COUNT</td>
<td>13</td>
<td>II/A0 wait count</td>
</tr>
<tr>
<td>LCM_POSITION_TOP</td>
<td>0x00</td>
<td>LCD module display line (top)</td>
</tr>
<tr>
<td>LCM_POSITION_BOTTOM</td>
<td>0x40</td>
<td>LCD module display line (bottom)</td>
</tr>
<tr>
<td>R_RTC_INIT_SEC</td>
<td>0x55</td>
<td>Initial value of the current clock time (second)</td>
</tr>
<tr>
<td>R_RTC_INIT_MIN</td>
<td>0x59</td>
<td>Initial value of the current clock time (minute)</td>
</tr>
<tr>
<td>R_RTC_INIT HOUR</td>
<td>0x15</td>
<td>Initial value of the current clock time (hour)</td>
</tr>
<tr>
<td>R_RTC_INIT WEEK</td>
<td>0x06</td>
<td>Initial value of the current clock time (day of the week)</td>
</tr>
<tr>
<td>R_RTC_INIT_DAY</td>
<td>0x01</td>
<td>Initial value of the current clock time (day)</td>
</tr>
<tr>
<td>R_RTC_INIT_MONTH</td>
<td>0x01</td>
<td>Initial value of the current clock time (month)</td>
</tr>
<tr>
<td>R_RTC_INIT_YEAR</td>
<td>0x21</td>
<td>Initial value of the current clock time (year)</td>
</tr>
<tr>
<td>R_RTC_ALARM_MIN</td>
<td>0x00</td>
<td>Set value of alarm occurrence time (minute)</td>
</tr>
<tr>
<td>R_RTC_ALARM_HOUR</td>
<td>0x16</td>
<td>Set value of alarm occurrence time (hour)</td>
</tr>
<tr>
<td>R_RTC_ALARM_WEEK</td>
<td>0xFF</td>
<td>Set value of alarm occurrence time (day of the week)</td>
</tr>
<tr>
<td>R_INTERRUPT_OFF</td>
<td>0</td>
<td>Interrupt flag cleared</td>
</tr>
<tr>
<td>R_INTERRUPT_ON</td>
<td>1</td>
<td>Interrupt flag ON</td>
</tr>
</tbody>
</table>
4.3 List of Variables

Table 4-4 lists global variables.

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Description</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td>g_rtc_constperiod</td>
<td>Fixed-cycle interrupt flag</td>
<td>main, r_Config_RTC_Create_UserInit,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>r_Config_RTC_callback_constperiod,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>r_rtc_is_constperiod_flag_on,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>r_rtc_clear_constperiod_flag</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_LCM_is_sendend</td>
<td>I2C communication (with the LCD module) completion flag</td>
<td>r_LCM_init, r_LCM_turn_sendend_on, r_LCM_wait_sendend</td>
</tr>
</tbody>
</table>

4.4 List of Functions

Table 4-5 shows a list of functions.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_Config_RTC_Create_UserInit()</td>
<td>User-specified RTC initialization processing</td>
</tr>
<tr>
<td>r_Config_RTC_callback_alarm()</td>
<td>Alarm interrupt processing.</td>
</tr>
<tr>
<td>r_Config_RTC_callback_constperiod()</td>
<td>Fixed-cycle interrupt processing.</td>
</tr>
<tr>
<td>r_rtc_init_current_time()</td>
<td>Initial setting for the current clock time</td>
</tr>
<tr>
<td>r_rtc_init_alarm_time()</td>
<td>Initial setting for alarm time</td>
</tr>
<tr>
<td>r_rtc_is_constperiod_flag_on()</td>
<td>Check the fixed-cycle interrupt flag.</td>
</tr>
<tr>
<td>r_rtc_clear_constperiod_flag()</td>
<td>Clear the fixed-cycle interrupt flag.</td>
</tr>
<tr>
<td>r_rtc_display_current_time()</td>
<td>Display the current clock time on the LCD.</td>
</tr>
<tr>
<td>convert_BCD_to_2chars()</td>
<td>Convert 2-digit BCD to two characters.</td>
</tr>
<tr>
<td>convert_week_to_3chars()</td>
<td>Convert day-of-the-week code to three characters.</td>
</tr>
<tr>
<td>r_Config_IICA0_callback_master_sendend()</td>
<td>IICA0 send end callback processing.</td>
</tr>
<tr>
<td>r_Config_IICA0_callback_master_error()</td>
<td>IICA0 error callback processing.</td>
</tr>
<tr>
<td>R_Config_PORT_Create_UserInit()</td>
<td>User-specified port initialization processing</td>
</tr>
<tr>
<td>r_LCM_init()</td>
<td>LCD module initialization.</td>
</tr>
<tr>
<td>r_LCM_clear()</td>
<td>LCD module display clear processing.</td>
</tr>
<tr>
<td>r_LCM_send_string()</td>
<td>LCD module character string transmission processing</td>
</tr>
<tr>
<td>r_LCM_send_command()</td>
<td>LCD module command transmission processing</td>
</tr>
<tr>
<td>r_LCM_send_data()</td>
<td>LCD module data transmission processing</td>
</tr>
<tr>
<td>r_LCM_turn_sendend_on()</td>
<td>LCD module communication end flag setting</td>
</tr>
<tr>
<td>r_LCM_wait_sendend()</td>
<td>LCD module communication end wait processing</td>
</tr>
</tbody>
</table>
### 4.5 Specification of Function

The function specifications of the sample code are shown below.

<table>
<thead>
<tr>
<th>Function</th>
<th>Outline</th>
<th>Header</th>
<th>Declaration</th>
<th>Description</th>
<th>Argument</th>
<th>Return Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_Config_RTC_Create_UserInit()</td>
<td>User-specified RTC initialization processing</td>
<td>r_cg_userdefine.h</td>
<td>void R_Config_RTC_Create_UserInit(void);</td>
<td>Perform user-specified processing among initialization required before starting the RTC.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>r_Config_RTC_callback_alarm()</td>
<td>Alarm interrupt processing.</td>
<td>r_cg_macrodriver.h</td>
<td>static void r_Config_RTC_callback_alarm(void);</td>
<td>This callback function is called when an RTC alarm interrupt is generated.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>r_Config_RTC_callback_constperiod()</td>
<td>Fixed-cycle interrupt processing.</td>
<td>r_cg_macrodriver.h, r_cg_userdefine.h</td>
<td>static void r_Config_RTC_callback_constperiod(void);</td>
<td>This callback function is called when an RTC fixed-cycle interrupt is generated.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>r_rtc_init_current_time()</td>
<td>Initial setting for the current clock time</td>
<td>r_cg_macrodriver.h, r_cg_userdefine.h</td>
<td>void r_rtc_init_current_time(void);</td>
<td>Set the current clock time in the RTC register.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>r_rtc_init_alarm_time()</td>
<td>Initial setting for the alarm clock time</td>
<td>r_cg_macrodriver.h, r_cg_userdefine.h, Config_RTC.h</td>
<td>void r_rtc_init_alarm_time(void);</td>
<td>Set the alarm time in the RTC register.</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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_INTERRUPT_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_OFF).

Argument
None

Return Value
None

convert_BCD_to_2chars()

Outline
Conversion of two-digit BCD value to two characters

Header
r_cg_macrodriver.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
r_cg_macrodriver.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:

<table>
<thead>
<tr>
<th>Source value</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conversion result</td>
<td>Sun</td>
<td>Mon</td>
<td>Tue</td>
<td>Wed</td>
<td>Thu</td>
<td>Fri</td>
<td>Sat</td>
</tr>
</tbody>
</table>

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_IICA0_callback_master_sendend()  
Outline: IICA0 send end callback processing.  
Header: r_cg_macrodriver.h, Config_IICA0.h, LCM_driver.h  
Declaration: static void r_Config_IICA0_callback_master_sendend(void);  
Description: Callback function called when an IICA0 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_IICA0_callback_master_error()  
Outline: IICA0 error callback processing.  
Header: r_cg_macrodriver.h, Config_IICA0.h, LCM_driver.h  
Declaration: static void r_Config_IICA0_callback_master_error(MD_STATUS flag);  
Description: Callback function called when an IICA0 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, Config_PORT.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

r_LCM_init()  
Outline: LCD module Initialization  
Header: LCM_driver.h, Config_IICA0.h  
Declaration: void r_LCM_init(void);  
Description: Initializes LCD module.  
Argument: None  
Return Value: None

r_LCM_clear()  
Outline: LCD module display clear processing  
Header: LCM_driver.h, Config_IICA0.h  
Declaration: void r_LCM_clear(void);  
Description: Transmit the display clear command to the LCD module.  
Argument: None  
Return Value: None
### r_LCM_send_string()

**Outline**
LCD module character string transmission processing

**Header**
LCM_driver.h, Config_IICA0.h

**Declaration**
void r_LCM_send_string(uint8_t * const str, lcm_position_t pos);

**Description**
Display the character string transferred with str on the LCD module.
The line to display the character string is specified by pos.

**Argument**
- uint8_t * const str: Character string to be displayed
- lcm_position_t pos: LCM_POSITION_TOP: Displayed at the top
  LCM_POSITION_BOTTOM: Displayed at the bottom

**Return Value**
None

### r_LCM_send_command()

**Outline**
LCD module command transmission processing

**Header**
LCM_driver.h, Config_IICA0.h

**Declaration**
void r_LCM_send_command(uint8_t command);

**Description**
Send the command transferred with command to the LCD module.

**Argument**
- uint8_t command: Command to be sent to the LCD module

**Return Value**
None

### r_LCM_send_data()

**Outline**
LCD module data transmission processing

**Header**
LCM_driver.h, Config_IICA0.h

**Declaration**
void r_LCM_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

### r_LCM_turn_sendend_on()

**Outline**
LCD module communication end flag setting

**Header**
LCM_driver.h, Config_IICA0.h

**Declaration**
void r_LCM_turn_sendend_on(void);

**Description**
Set the I2C communication (with the LCD module) end flag for 
g_LCM_is_sendend.

**Argument**
None

**Return Value**
None

### r_LCM_wait_sendend()

**Outline**
LCD module communication end wait processing

**Header**
LCM_driver.h, Config_IICA0.h

**Declaration**
static void r_LCM_wait_sendend(void);

**Description**
Wait until the I2C communication (with the LCD module) ends, and then perform wait processing during the command execution wait time period (5 ms).

**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

main()

Enable interrupt

r_LCM_init()

Is a power-on reset generated?

YES

Initial settings for the realtime clock
R_Config_RTC_Create()

r rtc_init_current_time()

r rtc_init_alarm_time()

Setting without a power-on reset

Start the realtime clock.
R_Config_RTC_Start()

Time update timing?

YES

Disable INTRTC interrupt

r rtc_display_current_time()

r rtc_clear_constperiod_flag()

Enable INTRTC interrupt

IE ← 1

PORF = 0 ?

RCWEN ← 1
RTCPRI ← 1
RTCP0 ← 1
RTCIF ← 0
RTCMK ← 0

PORF ← 1

g rtc constperiod = 1 ?

RTCMK ← 1

RTCMK ← 0
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)

- R_Config_RTC_Create_UserInit()
- Clear g_rtc_constperiod_flag
- Return

4.6.3 Alarm Interrupt Processing
Figure 4-3 shows the flowchart of the alarm interrupt processing.

Figure 4-3 Alarm Interrupt Processing

- r_Config_RTC_callback_alarm()
- Set P52 to low level to turn on the LED.
- Return
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

```
[Diagram of flowchart]
```

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

```
[Diagram of flowchart]
```
4.6.6 Initial Settings for Alarm Time

Figure 4-6 shows the flowchart of the 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.
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

```
r_rtc_clear_constperiod_flag()

Clear g_rtc_constperiod

Return
```
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)

- `r_rtc_display_current_time()`

  - Acquire the current clock time. `R_Config_RTC_Get.CounterValue(&counter_val)`

  - NO

  - Is the return value of the above function call MD_OK?

  - YES

  - `convert_BCD_to_2chars` (read_val.year, &string_time[0])

  - `string_time[2] = '/'`

  - `convert_BCD_to_2chars` (read_val.month, &string_time[3])

  - `string_time[5] = '/'`

  - `convert_BCD_to_2chars` (read_val.day, &string_time[6])

  - `string_time[8] = '('`

  - `convert_week_to_3Chars` (read_val.week, &string_time[9])

  - `string_time[12] = ')'`

  - `string_time[13] = '\0'`

  - `r_LCM_send_string` (string_time, LCM_POSITION_TOP)

* A character string "2-digit year/2-digit month/2-digit day (3-character day of the week)" is displayed at the top of the LCD.
4.6.10 BCD-to-Character Conversion Processing

Figure 4-11 shows the flowchart of the processing for conversion from BCD to character.

```c
convert_BCD_to_2chars(bcd, *str)

Is str not NULL?
YES

Convert the upper digit of bcd to a character and substitute it for *(str+0).

Convert the lower digit of bcd to a character and substitute it for *(str+1).

NO

Return
```

* A character string “2-digit hour:2-digit minute: 2-digit seconds” is displayed at the bottom of the LCD.
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 IICA0 Send End Callback Processing

Figure 4-13 shows the flowchart of the IICA0 send end callback processing.

Figure 4-13  IICA0 Send End Callback Processing

convert_week_to_3chars(week, *str)

Is str not NULL?

NO

YES

Substitute the 3-character day of the week for *(str+0), *(str+1), and *(str+2).

Return

r_Config_IICA0_callback_master_sendend()

Generate stop condition

SPT0 ← 1

r_LCM_turn_sendend_on()

Return
4.6.13 IICA0 Error Callback Processing
Figure 4-14 shows the flowchart of the IICA0 error callback processing.

Figure 4-14 IICA0 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)

r_Config_IICA0_callback_master_error(flag)

r_LCM_turn_sendend_on()

Return

R_Config_PORT_Create_UserInit()

Set P52 to high level and P53 to low level.

Return
4.6.15 LCD Module Initialization

Figure 4-16 shows the flowchart of the LCD module initialization.

Figure 4-16 LCD Module Initialization

```
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_LCM_init()</td>
<td></td>
</tr>
<tr>
<td>r_LCM_send_command(0x20)</td>
<td>Set parameter for function set</td>
</tr>
<tr>
<td>r_LCM_send_command(0x08)</td>
<td>Set parameter for display ON/OFF</td>
</tr>
<tr>
<td>r_LCM_send_command(0x04)</td>
<td>Set parameter for entry mode set</td>
</tr>
<tr>
<td>Clear g_LCM_is_sendend</td>
<td></td>
</tr>
<tr>
<td>Return</td>
<td></td>
</tr>
</tbody>
</table>
```

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

```
r_LCM_clear()

r_LCM_send_command(0x00)  Send the Clear Display command.

Return
```

Send the Clear Display command.

Clear g_LCM_is_sendend ← 0
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**

```plaintext
r_LCD_send_string(str, pos)

R_LCD_send_command(_0x80_LCD_COMMAND_SET_DDRAM_ADDRESS | pos)

i = 0

Is i less than the number of str characters?

YES

Is i less than the number of displayable characters?

YES

r_LCD_send_data(The i-th character of str)

increment i.

NO

Return
```

Send the Set DDRAM Address command and designate the position specified by pos as the drawing start position.
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

```
<table>
<thead>
<tr>
<th>r_LCM_send_command(command)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare a temporary buffer and place</td>
</tr>
<tr>
<td>&quot;,_0x00_LCM_CONTROL_BYTE_RS_LOW&quot;</td>
</tr>
<tr>
<td>and command consecutively.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Send the command to the LCM.</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_Config_IICA0_Master_Send(</td>
</tr>
<tr>
<td>LCM slave address,</td>
</tr>
<tr>
<td>temporary buffer, temporary buffer size,</td>
</tr>
<tr>
<td>LCM_CONFIG_WAIT_COUNT)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r_LCM_wait_sendend()</th>
</tr>
</thead>
</table>

Return
```

LCD module specification:
Command is placed next to the control byte 0x00.

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

```
<table>
<thead>
<tr>
<th>r_LCM_send_data(data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare a temporary buffer and place</td>
</tr>
<tr>
<td>&quot;,_0x80_LCM_CONTROL_BYTE_RS_HIGH&quot;</td>
</tr>
<tr>
<td>and data consecutively.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Send data to the LCM.</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_Config_IICA0_Master_Send(</td>
</tr>
<tr>
<td>LCM slave address,</td>
</tr>
<tr>
<td>temporary buffer, temporary buffer size,</td>
</tr>
<tr>
<td>LCM_CONFIG_WAIT_COUNT)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r_LCM_wait_sendend()</th>
</tr>
</thead>
</table>

Return
```

LCD module specification:
Data is placed next to the control byte 0x80.
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**

```
\begin{itemize}
  \item r_LCM_turn_sendend_on()
  \item g_LCM_is_sendend = 1
  \item Return
\end{itemize}
```

4.6.21 LCD Module Communication End Wait Processing
Figure 4-22 shows the flowchart of the communication end flag processing for the LCD module.

**Figure 4-22  LCD Module Communication End Wait Processing**

```
\begin{itemize}
  \item r_LCM_wait_sendend(str, pos)
  \item \textbf{NO}
  \item g_LCM_is_sendend = 1 ?
  \item \textbf{YES}
  \item g_LCM_is_sendend = 0
  \item Wait processing
  \item Return
\end{itemize}
```

LCD module specification:
A wait time of at least 5 ms is required after transmission.
5. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

6. Reference Documents

RL78/G23 User's Manual: Hardware (R01UH0896)
RL78 family user's manual software (R01US0015)
The latest versions can be downloaded from the Renesas Electronics website.

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

LCD module datasheet
(ACM1602NI-FLW-FBW-M01 (ZETTLER DISPLAYS) CHARACTER MODULE VER1.4)

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

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Description</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>2021.04.13</td>
<td>-</td>
<td>First Edition</td>
<td></td>
</tr>
<tr>
<td>1.01</td>
<td>2021.07.12</td>
<td>5</td>
<td>Updated the Operation Confirmation Conditions</td>
<td></td>
</tr>
</tbody>
</table>
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 $V_{IL\text{ (Max.)}}$ and $V_{IH\text{ (Min.)}}$ 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 $V_{IL\text{ (Max.)}}$ and $V_{IH\text{ (Min.)}}$.

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.
Notice

1. 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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.

3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.

5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.

6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   - “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   - “High Quality”: Undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.) or may cause serious property damage (space systems; underwater repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.

7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INNUBBERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.

9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunction under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or damage to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating the safety of the final products or systems 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. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

11. Renesas Electronics products and technologies shall 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. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

© 2021 Renesas Electronics Corporation. All rights reserved.

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

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

Trademarks
Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.