RL78/G23
I2C Supporting Multiple Slave Addresses (Slave)

Introduction
This application note describes how to use the slave function of the I2C bus that supports multiple slave addresses by using the all addresses matching function of the RL78/G23. This application note enables a function of four serial memory areas (256 bytes × 4) designated by respective slave addresses.

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 Basic Specifications of the I2C Bus (Slave) ................................................................................. 3  
   1.2 Outline of Operation ................................................................................................................. 4  
   1.3 Identifying Slave Address ......................................................................................................... 4  
   1.4 RAM Function ......................................................................................................................... 5  

2. Operation Confirmation Conditions .......................................................................................... 6  

3. Hardware Descriptions ............................................................................................................ 7  
   3.1 Example of Hardware Configuration ......................................................................................... 7  
   3.2 List of Pins to be Used ............................................................................................................ 8  

4. Software Explanation ............................................................................................................... 9  
   4.1 Setting of Option Byte ............................................................................................................ 9  
   4.2 List of Constants ................................................................................................................... 9  
   4.3 List of Variables .................................................................................................................... 10  
   4.4 List of Functions ................................................................................................................... 10  
   4.5 Specification of Functions ..................................................................................................... 11  
   4.6 Flowcharts ........................................................................................................................... 14  
       4.6.1 Main Processing ............................................................................................................. 14  
       4.6.2 Variables Initialization Processing ................................................................................. 14  
       4.6.3 LED Display Initialization Function .............................................................................. 15  
       4.6.4 Interval Timer (5 ms) Interrupt Processing Function ................................................. 16  
       4.6.5 IICA0 Variables Initialization Processing Function ................................................. 17  
       4.6.6 Slave Address Read Processing Function ...................................................................... 17  
       4.6.7 IICA0 Communication Status Read Processing Function ...................................... 18  
       4.6.8 IICA0 Interrupt Processing Function ............................................................................ 18  
       4.6.9 IICA0 Slave Interrupt Processing Function ............................................................... 19  
       4.6.10 RAM Data Transmission Processing Function ....................................................... 20  

5. Sample code ............................................................................................................................ 25  

6. Reference Documents ............................................................................................................. 25  

Revision History ......................................................................................................................... 26
1. Specifications

1.1 Basic Specifications of the I2C Bus (Slave)

The following lists the basic specifications of the I2C bus.

- I2C bus to be connected: Fast mode (up to 400 kbps)
- Slave address 1: 0010000B (First serial memory)
- Slave address 2: 0100100B (Second serial memory)
- Slave address 3: 1011010B (Third serial memory)
- Slave address 4: 1101011B (Fourth serial memory)
- Extension code: Not supported (Ignored and escapes from communication)

Note: In the RL78 family, the own address (7 bits) is expressed by the upper 7 bits of the SVA0 register.

The least significant bit (LSB) of the SVA0 register is always 0.

Table 1-1 lists peripheral functions to be used.

<table>
<thead>
<tr>
<th>Peripheral Function</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>IICA0</td>
<td>Slave function of the I2C bus</td>
</tr>
<tr>
<td>TM07</td>
<td>5 ms interval timer interrupt</td>
</tr>
</tbody>
</table>

The following describes the main settings.

(1) Initial settings for IICA0

Table 1-2 lists Initial Settings for IICA0.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Setting Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IICCTL01</td>
<td>4DH</td>
<td>All addresses match function enabled, fast mode, digital filter on, IICA0 operating clock fCLK/2</td>
</tr>
<tr>
<td>IICWL0</td>
<td>15H</td>
<td>SCLA0 low width</td>
</tr>
<tr>
<td>IICWH0</td>
<td>14H</td>
<td>SCLA0 high width</td>
</tr>
<tr>
<td>SVA0</td>
<td>50H</td>
<td>Slave address 50H</td>
</tr>
<tr>
<td>IICF0</td>
<td>03H</td>
<td>Bus released, communication reservation disabled</td>
</tr>
<tr>
<td>IICCTL00</td>
<td>1CH</td>
<td>SPD interrupt enabled, wait at the 9th clock, ACK response enabled</td>
</tr>
<tr>
<td>P6</td>
<td>00H</td>
<td>SCL/SDA signal multiplexed pins: Set to 0</td>
</tr>
<tr>
<td>PM6</td>
<td>00H</td>
<td>SCL/SDA signal pins: Set to output</td>
</tr>
</tbody>
</table>
(2) Initial settings for TM07

Table 1-3 lists the Initial Settings for TM07.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Setting Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPS0</td>
<td>0020H</td>
<td>CK00/CK02/CK03: 32 MHz, CK01: 8 MHz</td>
</tr>
<tr>
<td>TT0</td>
<td>0AFFH</td>
<td>Count operation of all channels are stopped</td>
</tr>
<tr>
<td>TMPR007</td>
<td>00H</td>
<td>Interrupt priority: Level 2</td>
</tr>
<tr>
<td>TMR07</td>
<td>8000H</td>
<td>TM07 count clock: CK01 (8 MHz)</td>
</tr>
<tr>
<td>TDR07</td>
<td>40000</td>
<td>Interval time: 5 ms</td>
</tr>
<tr>
<td>TOM0</td>
<td>00H</td>
<td>TM07: Master mode</td>
</tr>
<tr>
<td>TOL0</td>
<td>00H</td>
<td>TM07 output: Positive logic</td>
</tr>
<tr>
<td>T00</td>
<td>00H</td>
<td>TM07 output: 0</td>
</tr>
<tr>
<td>TOE0</td>
<td>00H</td>
<td>TM07 output: Disabled</td>
</tr>
</tbody>
</table>

1.2 Outline of Operation

Four 256-byte serial memory (serial RAM) areas are enabled. The internal address of each serial RAM is designated by the 1-byte data (register address) next to the slave address. The received slave address is indicated by LEDs.

1.3 Identifying Slave Address

When the all addresses match function is enabled (SVADIS0 = 1), the RL78/G23 operates in the same way as extension code reception. An INTIICA0 interrupt of the slave is generated at the falling edge of the 8th clock during reception of the slave address. When the received slave address is valid, an ACK response (ACKE0 = 1) is returned. If it is invalid, the RL78/G23 escapes from communication (LREL0 = 1).

In this application note, WTIM0 is set to 1 to generate an interrupt request at the falling edge of the 9th clock. Therefore, an INTIICA0 interrupt is generated twice during reception of the slave address.
1.4 RAM Function
Data can be written to or read from the slave address and the serial RAM address designated by the next 1-byte data (register address).

When the transfer direction (W) following the valid slave address is specified, the slave stores the next receive data (register address), and then writes received data to the designated register addresses sequentially.

**Figure 1-1** Continuous Data Write to Designated Register Addresses

![Diagram of continuous data write](image)

When the transfer direction (W) following the valid slave address is specified, the slave stores the next receive data (register address). After that, when the slave receives start conditions, valid slave address, and transfer direction (R), the slave sequentially reads data from the previously designated register address, and then sends the data to the master.

**Figure 1-2** Continuous Data Read from Designated Register Addresses

![Diagram of continuous data read](image)

Upon completion of data read or write, the slave increments the designated address to prepare for the next data read or write. After the last address (FFH) is accessed, the next address becomes the first address (00H). The selected serial RAM is not switched by incrementing the address.
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 Fast Prototyping Board (RTK7RLG230CLG000BJ)</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>High-speed on-chip oscillator clock: 32 MHz</td>
</tr>
<tr>
<td></td>
<td>CPU/peripheral hardware clock: 32 MHz</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>3.3 V (can be operated at 1.8 V to 5.5 V)</td>
</tr>
<tr>
<td></td>
<td>LVD0 detection voltage: Reset mode</td>
</tr>
<tr>
<td></td>
<td>At rising edge TYP. 3.96 V (3.84 V to 4.08 V)</td>
</tr>
<tr>
<td></td>
<td>At falling edge TYP. 3.88 V (3.76 V to 4.00 V)</td>
</tr>
<tr>
<td>Integrated development environment (CS+)</td>
<td>CS+ V8.05.00 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>e2 studio 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 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>
</tbody>
</table>
3. Hardware Descriptions

3.1 Example of Hardware Configuration

Figure 3-1 Hardware Configuration shows an example of the hardware configuration used in the application note.

Note 1. This schematic circuit diagram is simplified to show the outline of connections. When creating actual circuits, design them using appropriate pin processing so that the circuits meet electrical characteristics. (Connect input-only ports to VDD or VSS individually through a resistor.)

Note 2. Connect pins (with a name beginning with EVSS), if any, to VSS, and connect pins (with a name beginning with EVDD), if any, to VDD.

Note 3. Set VDD to a voltage not less than the reset release voltage (VLVD0) set by the LVD.
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>P03 (D0)</td>
<td>Output</td>
<td>Drives LED1 and LED5.</td>
</tr>
<tr>
<td>P02 (D1)</td>
<td>Output</td>
<td>Drives LED2 and LED6.</td>
</tr>
<tr>
<td>P43 (D2)</td>
<td>Output</td>
<td>Drives LED3 and LED7.</td>
</tr>
<tr>
<td>P42 (D3)</td>
<td>Output</td>
<td>Drives LED4 and LED8.</td>
</tr>
<tr>
<td>P77 (D4)</td>
<td>Output</td>
<td>Lights the lower 4 bits of LEDs.</td>
</tr>
<tr>
<td>P41 (D5)</td>
<td>Output</td>
<td>Lights the upper 4 bits of LEDs.</td>
</tr>
<tr>
<td>P61 (D14)</td>
<td>Input / Output</td>
<td>SDA signal</td>
</tr>
<tr>
<td>P60 (D15)</td>
<td>Input / Output</td>
<td>SCL signal</td>
</tr>
<tr>
<td>P137 (D18)</td>
<td>Input</td>
<td>Switch (SW1) input</td>
</tr>
</tbody>
</table>
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>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$000C0H$ / $010C0H$</td>
<td>$11101111B$</td>
<td>Disables the watchdog timer. (Counting stopped after reset)</td>
</tr>
<tr>
<td>$000C1H$ / $010C1H$</td>
<td>$1111010B$</td>
<td>LVD0 detection voltage: reset mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At rising edge TYP. 3.96 V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At falling edge TYP. 3.88 V</td>
</tr>
<tr>
<td>$000C2H$ / $010C2H$</td>
<td>$1101000B$</td>
<td>HS mode, High-speed on-chip oscillator clock ($f_{IH}$): 32 MHz</td>
</tr>
<tr>
<td>$000C3H$ / $010C3H$</td>
<td>$1000100B$</td>
<td>Enables on-chip debugging</td>
</tr>
</tbody>
</table>

4.2 List of Constants

Table 4-2 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>D0_pin</td>
<td>P0_bit.no3</td>
<td>Port corresponding to the D0 pin</td>
</tr>
<tr>
<td>D1_pin</td>
<td>P0_bit.no2</td>
<td>Port corresponding to the D1 pin</td>
</tr>
<tr>
<td>D2_pin</td>
<td>P4_bit.no3</td>
<td>Port corresponding to the D2 pin</td>
</tr>
<tr>
<td>D3_pin</td>
<td>P4_bit.no2</td>
<td>Port corresponding to the D3 pin</td>
</tr>
<tr>
<td>SEL_L</td>
<td>P7_bit.no7</td>
<td>Lower LED select pin</td>
</tr>
<tr>
<td>SEL_U</td>
<td>P4_bit.no1</td>
<td>Upper LED select pin</td>
</tr>
<tr>
<td>SW_IN</td>
<td>P13_bit.no7</td>
<td>Onboard switch input pin</td>
</tr>
<tr>
<td>LED_ON</td>
<td>0</td>
<td>Common data value for turning on LEDs</td>
</tr>
<tr>
<td>LED_OFF</td>
<td>1</td>
<td>Common data value for turning off LEDs</td>
</tr>
<tr>
<td>RAM1</td>
<td>0b00100000</td>
<td>7-bit slave address of RAM1</td>
</tr>
<tr>
<td>RAM2</td>
<td>0b0100100</td>
<td>7-bit slave address of RAM2</td>
</tr>
<tr>
<td>RAM3</td>
<td>0b1011010</td>
<td>7-bit slave address of RAM3</td>
</tr>
<tr>
<td>RAM4</td>
<td>0b1101011</td>
<td>7-bit slave address of RAM4</td>
</tr>
<tr>
<td>PAGE_SIZE</td>
<td>256</td>
<td>RAM size</td>
</tr>
<tr>
<td>HIGH</td>
<td>0x01</td>
<td>High level</td>
</tr>
<tr>
<td>LOW</td>
<td>0x00</td>
<td>Low level</td>
</tr>
</tbody>
</table>
4.3 List of Variables
Table 4-3 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_ram_[4][256]</td>
<td>RAM data area (256 bytes × 4)</td>
<td>R_IICA0_Init(), iica0_slave_handler()</td>
</tr>
<tr>
<td>uint16_t</td>
<td>gp_ram[4]</td>
<td>Register address</td>
<td>R_IICA0_Init(), iica0_slave_handler()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_prev_status</td>
<td>Immediately previous communication status</td>
<td>R_IICA0_Init(), iica0_slave_handler()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_status</td>
<td>Current communication status</td>
<td>R_IICA0_Init(), iica0_slave_handler()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00: Successful completion (waiting for the next communication)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x01: Register address is being received</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x02: Data is being transmitted</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x03: Data is being received</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_sl_addr</td>
<td>Received slave address</td>
<td>R_IICA0_Init(), iica0_slave_handler()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_selRAM</td>
<td>Selected RAM</td>
<td>iica0_slave_handler()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_disp_data</td>
<td>LED display data</td>
<td>R_LED_Init(), r_tau0_channel7_interrupt()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g_5ms</td>
<td>5 ms counter</td>
<td>R_LED_Init(), r_tau0_channel7_interrupt()</td>
</tr>
</tbody>
</table>

4.4 List of Functions
Table 4-4 shows a list of functions.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_Config_IICA0_Create()</td>
<td>IICA0 initial setting processing</td>
</tr>
<tr>
<td>R_Config_IICA0_Create_UserInit()</td>
<td>Enabling all addresses matching function</td>
</tr>
<tr>
<td>R_IICA0_Init()</td>
<td>IICA0 variables initialization processing</td>
</tr>
<tr>
<td>r_Config_IICA0_interrupt()</td>
<td>IICA0 interrupt processing</td>
</tr>
<tr>
<td>r_IICA0_slave_handler()</td>
<td>IICA0 slave interrupt processing</td>
</tr>
<tr>
<td>r_sl_addr_get()</td>
<td>IICA0 reception slave address read processing</td>
</tr>
<tr>
<td>r_status_chk()</td>
<td>IICA0 communication status read processing</td>
</tr>
<tr>
<td>R_Config_PORT_Create()</td>
<td>Port initial setting processing</td>
</tr>
<tr>
<td>R_LED_Init()</td>
<td>LED display initialization processing</td>
</tr>
<tr>
<td>R_Config_TAU0_7_Create()</td>
<td>TM07 initial setting processing</td>
</tr>
<tr>
<td>r_Config_TAU0_7_interrupt()</td>
<td>Interval timer (5 ms) interrupt processing</td>
</tr>
</tbody>
</table>
4.5 Specification of Functions

The function specifications of the sample code are shown below.

### R_Config_IICA0_Create()

<table>
<thead>
<tr>
<th>Outline</th>
<th>IICA0 initial setting processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_Config_IICA0_Create(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Makes initial settings for the IICA0 function.</td>
</tr>
<tr>
<td>Argument</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### R_Config_IICA0_Create_UserInit()

<table>
<thead>
<tr>
<th>Outline</th>
<th>Enabling all addresses matching function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_Config_IICA0_Create_UserInit(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Enables the all addresses matching function of IICA0.</td>
</tr>
<tr>
<td>Argument</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### R_IICA0_Init()

<table>
<thead>
<tr>
<th>Outline</th>
<th>IICA0 variables initialization processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_IICA0_Init(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Initializes variables and registers used for IICA0.</td>
</tr>
<tr>
<td>Argument</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### r_Config_IICA0_interrupt()

<table>
<thead>
<tr>
<th>Outline</th>
<th>IICA0 interrupt processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>static void __near r_Config_IICA0_interrupt(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Receives an IICA0 interrupt request and performs required processing.</td>
</tr>
<tr>
<td>Argument</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>

### r_IICA0_slave_handler()

<table>
<thead>
<tr>
<th>Outline</th>
<th>IICA0 slave interrupt processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cg_macrodriver.h, Config_IICA0.h, LED_LIB.h, r_cg_userdefine.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>static void r_IICA0_slave_handler(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Checks whether the slave address is valid upon receiving a slave interrupt from IICA0. When the address is valid, a response is returned according to the request from the master. If the address is invalid, the interrupt processing is exited.</td>
</tr>
<tr>
<td>Argument</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>
### r_sl_addr_get()

**Outline**  
Slave address read processing

**Header**  
r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h

**Declaration**  
uint8_t r_sl_addr_get(void)

**Description**  
Returns the received slave address.

**Argument**  
None

**Return Value**  
uint8_t  
The latest received slave address

### r_status_chk()

**Outline**  
IICA0 communication status read processing

**Header**  
r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h

**Declaration**  
uint8_t r_status_chk(void)

**Description**  
Reads the IICA0 communication status (g_prev_status), and then clears it.

**Argument**  
None

**Return Value**  
uint8_t  
g_prev_status value

### R_Config_PORT_Create()

**Outline**  
Port initial setting processing

**Header**  
r_cg_macrodriver.h, Config_PORT.h, r_cg_userdefine.h

**Declaration**  
void R_Config_PORT_Create(void)

**Description**  
Makes initial settings for ports used for the LED display.

**Argument**  
None

**Return Value**  
uint8_t

### R_LED_Init()

**Outline**  
LED display initialization processing

**Header**  
r_cg_macrodriver.h, Config_TAU0_7.h, r_cg_userdefine.h

**Declaration**  
void R_LED_Init(void)

**Description**  
Initializes variables used for the LED display.

**Argument**  
None

**Return Value**  
uint8_t

### R_Config_TAU0_7_Create()

**Outline**  
Timer array unit initialization processing

**Header**  
r_cg_macrodriver.h, Config_TAU0_7.h, r_cg_userdefine.h

**Declaration**  
void R_Config_TAU0_7_Create(void)

**Description**  
Makes initial settings for the TAU0 function.

**Argument**  
None

**Return Value**  
None
### r_Config_TAU0_7_interrupt ()

<table>
<thead>
<tr>
<th>Outline</th>
<th>Slave address read processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cg_macrodriver.h, Config_TAU0_7.h, r_cg_userdefine.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>static void __near r_Config_TAU0_7_interrupt (void)</td>
</tr>
<tr>
<td>Description</td>
<td>Switches the LED display (between upper 4 bits and lower 4 bits) at intervals of 5 ms.</td>
</tr>
<tr>
<td>Argument</td>
<td>None</td>
</tr>
<tr>
<td>Return Value</td>
<td>None</td>
</tr>
</tbody>
</table>
4.6 Flowcharts

4.6.1 Main Processing

Figure 4-1 shows the flowchart of the main processing.

Figure 4-1  Main Processing

main()

- Initialize variables, and start operation. 
  R_MAIN_UserInit()

- Processing for initializing LEDs and IICA0 variables and for activating timer for LED display

- Transition to HALT mode

Waiting for an interrupt request

4.6.2 Variables Initialization Processing

Figure 4-2 shows the flowchart of the variables initialization function.

Figure 4-2  Variables Initialization Function

R_MAIN_UserInit()

- Initialization of LED control variables and activating timer for LED display

- Initialization of IICA0 control variables

- Enable vector interrupts.

IE bit ← 1

return
4.6.3 LED Display Initialization Function

Figure 4-3 shows the flowchart of the LED display initialization function.

Figure 4-3  LED Display Initialization Function

```
R_LED_Init()

Initialize display data.
Variable g_disp_data ← 0x00

Clear the 5-ms counter.
Variable g_5ms ← 0x00

Activate the 5-ms interval timer.
TMIF07 bit ← 0: INTTM07 request cleared
TMMK07 bit ← 0: INTTM07 request enabled
TS0L register ← 0x80: TM07 activated

return
```
4.6.4 Interval Timer (5 ms) Interrupt Processing Function

Figure 4-4 shows the flowchart of the interval timer (5 ms) interrupt processing function.

**Figure 4-4  Interval Timer (5 ms) Interrupt Processing Function**

```plaintext
r_Config_TAU0_7_interrupt()

Turn off LEDs.

Set data for LED1 and LED5.

Set data for LED2 and LED6.

Set data for LED3 and LED7.

Set data for LED4 and LED8.

Display for lower LEDs?

Yes

Turn on lower LEDs.

Turn on upper LEDs.

Read the next display data.

r_sl_addr_get()

Count 5 ms.

RETI

No

SEL_L bit ← LED_OFF
SEL_U bit ← LED_OFF

D0_pin bit ← (g Disp_data & 0x01)
One-bit right shift of variable g Disp_data

D1_pin bit ← (g Disp_data & 0x01)
One-bit right shift of variable g Disp_data

D2_pin bit ← (g Disp_data & 0x01)
One-bit right shift of variable g Disp_data

D3_pin bit ← (g Disp_data & 0x01)
One-bit right shift of variable g Disp_data

Determine LEDs to be activated by checking whether the variable g_5ms is an even or odd value.

SEL_L bit ← LED_ON

SEL_U bit ← LED_ON

Variable g Disp_data ← variable g Sl_addr

Variable g_5ms ← variable g_5ms + 1
```
4.6.5 IICA0 Variables Initialization Processing Function

Figure 4-5 shows the flowchart of the IICA0 variables initialization processing function.

**Figure 4-5** IICA0 Variables Initialization Processing Function

```
R_IICA0_Init()

Initialize RAM data
i = 0, i = PAGE_SIZE + 1

Set RAM1 data.

Set RAM2 data.

Set PAGE2 data.

Set PAGE3 data.

Initialize RAM data.

Initialize the RAM pointer.

Initialize the last communication status.

Initialize the slave address.

Initialize communication status.

return
```

- Variable g_ram[0][i] ← 0x00
- Variable g_ram[1][i] ← 0x01
- Variable g_ram[2][i] ← 0x02
- Variable g_ram[3][i] ← 0x03
- Variable gp_ram[] ← 0x00
- Variable g_prev_status ← 0x00
- Variable g_sl_addr ← 0x00
- Variable g_status ← 0x00
4.6.6 Slave Address Read Processing Function
Figure 4-6 shows the flowchart of the slave address read processing function.

**Figure 4-6 Slave Address Read Processing Function**

```
r_sl_addr_get()

Read the slave address.

return
```

Return value ← variable g_sl_addr

4.6.7 IICA0 Communication Status Read Processing Function
Figure 4-7 shows the flowchart of the IICA0 communication status read processing function.

**Figure 4-7 IICA0 Communication Status Read Processing Function**

```
r_status_chk()

Enable exclusive control.  IICAMK0 bit ← 1

Read communication status.  Local variable work ← variable g_prev_status

Clear communication status.  Variable g_prev_status ← 0x00

Disable exclusive control.  IICAMK0 bit ← 0

return
```

Return value ← local variable work
4.6.8 IICA0 Interrupt Processing Function

Figure 4-8 shows the flowchart of the IICA0 interrupt processing function.

**Figure 4-8 IICA0 Interrupt Processing Function**

- **r_Config_IICA0_interrupt()**
  - Slave operation?
    - No
    - Yes
      - IICA0 slave interrupt processing
        - r_IICA0_slave_handler()
      - RETI

When MSTS0 = 1 (master operation), this interrupt processing is skipped.
4.6.9 IICA0 Slave Interrupt Processing Function

Figure 4-9 to Figure 4-12 show the flowchart of the IICA0 slave interrupt processing function.

Figure 4-9 IICA0 Slave Interrupt Processing Function (1/4)
Figure 4-10  IICAO Slave Interrupt Processing Function (2/4)

Processing for checking address during normal address reception

Acquire the 7-bit address.

Variable \( s_{\text{addr}} \) \( \leftarrow \) \((g_{\text{sl_addr}} >> 1)\)

LREL0 bit \( \leftarrow \) 1: Escape from communication

Select RAM1.

Select RAM2.

Select RAM3.

Select RAM4.

Processing for starting register address reception

Variable \( g_{\text{prev_status}} \) \( \leftarrow \) 0x00

Clear the last communication status.

\( \text{TRC0} = 0? \)

Yes

Update communication status.

Start receiving the register address.

No

Receiving state?

Variable \( g_{\text{status}} \) = 0x01:

Update communication status to register address reception.

WREL0 bit \( \leftarrow \) 1: Wait released (communication continues)

Processing for starting RAM data transmission

Variable \( g_{\text{status}} \) = 0x02:

Update communication status to data transmission.

Perform transmission processing for data at the designated address.

0b0010000

0b0100100

0b1011010

0b1101011

Others

0b0010000

0b0100100

0b1011010

0b1101011

Others

Variable \( s_{\text{addr}} \) \( \leftarrow \) \((g_{\text{sl_addr}} >> 1)\)

RAM data transmission processing

\( r_{\text{ram_data_send}}() \)
Figure 4-11  I2C0 Slave Interrupt Processing Function (3/4)

Variable g_status = 0x00:
Initialize communication status. (Normal completion)

Variable g_status = 0x02:
Update communication status to data transmission.
**Figure 4-12** IICA0 Slave Interrupt Processing Function (4/4)

When variable \( g\_status = 0x01 \), set the register address.

Variable \( g\_status = 0x03 \): Change to data reception.

Variable \( gp\_ram[g\_selRAM] = \) IICA0 register

WREL0 bit \( \leftarrow 1 \): Wait released (communication continued)

\[
g\_ram[g\_selRAM] \leftarrow \text{g}_\text{ram} \leftarrow \text{IICA0 register (Read received data from IICA0, and then store it.)}
\]

Increment variable \( gp\_ram[g\_selRAM] \).

WREL0 bit \( \leftarrow 1 \): Wait released (communication continued)
4.6.10 RAM Data Transmission Processing Function
Figure 4-13 shows the flowchart of the RAM data transmission processing function.

**Figure 4-13** RAM Data Transmission Processing Function

```
<table>
<thead>
<tr>
<th>r_ram_data_send()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send RAM data.</td>
</tr>
<tr>
<td>IICA0 register ← g_ram[g_selRAM] [gp_ram[g_selRAM]]</td>
</tr>
<tr>
<td>(Write RAM data to IICA0.)</td>
</tr>
<tr>
<td>return</td>
</tr>
</tbody>
</table>
```
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.
## Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Apr.01.21</td>
<td>—</td>
<td>First Edition</td>
</tr>
<tr>
<td>1.01</td>
<td>May.18.21</td>
<td>6</td>
<td>Updated the Operation Confirmation Conditions</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 \( V_{IL} \) (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.
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 such 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, conditions are within the ranges specified by
Renesas Electronics products 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
appliance electronics; machine tools; personal electronic equipment; industrial robots; etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key
financial terminal systems; safety control 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: underground 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.

6. 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 INVULNERABLE 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.

7. 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.

8. 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 malfunctions 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 danger 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.

9. 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.

10. 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.

11. 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.

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

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

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled
subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters
TOYOUSU 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.