Bluetooth® Low Energy Protocol Stack

Sensor Application

Introduction

This application note explains the sample program, which runs on Bluetooth® Low Energy microcontroller RL78/G1D device and transmits sensor measured data to a remote device. The sample program contains not only the code files and firmware of the sensor application for RL78/G1D but also Android application to confirm sensor measured data transmitted by the sensor application. The sensor application works as a server role of GATT based profile. On the other hand, the Android application works as a client role of GATT based profile. By using the Android application, you can check sensor measured data with a line graph and control GPIO of RL78/G1D with a GUI.

Target Device

RL78/G1D (R5F11AGJ)

Related Documents

<table>
<thead>
<tr>
<th>Document Name</th>
<th>Document No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RL78/G1D</td>
<td></td>
</tr>
<tr>
<td>User's Manual: Hardware</td>
<td>R01UH0515</td>
</tr>
<tr>
<td>RL78/G1D Evaluation Board</td>
<td></td>
</tr>
<tr>
<td>User's Manual</td>
<td>R30UZ0048</td>
</tr>
<tr>
<td>E1 Emulator</td>
<td></td>
</tr>
<tr>
<td>User’s Manual</td>
<td>R20UT0398</td>
</tr>
<tr>
<td>Renesas Flash Programmer V3.05 Flash memory programming software</td>
<td></td>
</tr>
<tr>
<td>User's Manual</td>
<td>R20UT4307</td>
</tr>
<tr>
<td>CC-RL Compiler</td>
<td></td>
</tr>
<tr>
<td>User's Manual</td>
<td>R20UT3123</td>
</tr>
<tr>
<td>Bluetooth Low Energy Protocol Stack</td>
<td></td>
</tr>
<tr>
<td>User's Manual</td>
<td>R01UW0095</td>
</tr>
<tr>
<td>API Reference Manual: Basics</td>
<td>R01UW0088</td>
</tr>
<tr>
<td>Security Library Application Note</td>
<td>R01AN3777</td>
</tr>
</tbody>
</table>
Contents

1. Overview .......................................................................................................................... 4
2. Specification ....................................................................................................................... 6
   2.1 Software Composition ................................................................................................. 6
   2.2 Digital and Analog Interface ...................................................................................... 7
   2.3 Sensor Profile ............................................................................................................. 8
      2.3.1 Sensor Service ........................................................................................................ 9
      2.3.2 Accessing to Sensor Service .................................................................................. 13
3. Operating Procedure ....................................................................................................... 15
   3.1 Environment .................................................................................................................. 15
   3.2 Slide Switch Setting ..................................................................................................... 16
   3.3 Writing a Firmware ..................................................................................................... 17
   3.4 Connecting Sensors .................................................................................................... 20
   3.5 Installing Application ............................................................................................... 22
   3.6 Establishing a Connection .......................................................................................... 23
   3.7 Controlling GPIO ........................................................................................................ 24
   3.8 Confirming Sensor Measurement Data ..................................................................... 25
   3.9 Confirming Sensor Measurement Log ....................................................................... 26
4. Building Procedure ......................................................................................................... 27
   4.1 File Composition ......................................................................................................... 27
   4.2 Getting Libraries ......................................................................................................... 30
   4.3 Building a Firmware .................................................................................................. 31
   4.4 Configuring Peripherals ............................................................................................. 32
5. Sensor Control ................................................................................................................. 33
   5.1 Sensor Initialization ..................................................................................................... 34
   5.2 Sensor Profile Start .................................................................................................... 35
   5.3 Sensor Operation Start ............................................................................................... 36
   5.4 Sensor Measurement Data Notification ..................................................................... 37
   5.5 Sensor Profile Stop ..................................................................................................... 38
6. Functions .......................................................................................................................... 39
   6.1 Sensor Profile ............................................................................................................. 39
      6.1.1 R_SENS_Enable ..................................................................................................... 39
      6.1.2 R_SENS_Disable .................................................................................................. 39
      6.1.3 R_SENS_SetData ................................................................................................. 39
      6.1.4 R_SENS_Indication ............................................................................................. 40
      6.1.5 R_SENS_Notification ......................................................................................... 40
      6.1.6 R_SENS_Response .............................................................................................. 40
   6.2 Device Driver .............................................................................................................. 41
6.2.1 R_ISL29125_Init ................................................................. 41
6.2.2 R_ISL29125_SetModeSync ........................................... 42
6.2.3 R_ISL29125_SetMode ..................................................... 42
6.2.4 R_ISL29125_GetResultSync .......................................... 43
6.2.5 R_ISL29125_GetResult .................................................. 43
6.3 I²C Driver ........................................................................ 44
   6.3.1 R_IICA0_Create ............................................................ 44
   6.3.2 R_IICA0_RegisterCallback ........................................ 44
   6.3.3 R_IICA0_Write ............................................................. 45
   6.3.4 R_IICA0_Read .............................................................. 46
6.4 A/D Converter Driver ...................................................... 47
   6.4.1 R_ADC_Create ............................................................ 47
   6.4.2 R_ADC_GetChannel .................................................. 47
   6.4.3 R_ADC_GetResultSync ............................................. 47
7. Appendix ........................................................................... 48
   7.1 UART for Debug .......................................................... 48
1. Overview

Figure 1-1 shows the overview of the sample program.

The sample program is capable of controlling GPIO, A/D conversion, and I²C communication. In addition, it has a GATT based profile to control those operation. Remote device can control GPIO and sensor operation by communicating with RL78/G1D through the profile.

In evaluation, RL78/G1D Evaluation Board and sensors having analog output interface and/or I²C interface are used. Android device is also used as a remote device. Android application BleSensor for evaluation is included in this application note.

Upon start the sample program, RL78/G1D executes Advertising automatically. By operating BleSensor, you can establish a connection to RL78/G1D and then control GPIO and confirm measurement data of sensors.

By operating the GPIO control display of BleSensor, you can change a signal level of each output mode port and check a signal level of each input mode port.

By operating the sensor measurement display of BleSensor, you can check sensor measurement data both of A/D converter and sensors connected by I²C with a line graph.

Regarding the specification of a profile to control GPIO and sensors, refer to the following section.

section 2.3 "Sensor Profile"

Regarding the operation procedure such as writing a firmware to RL78/G1D, connecting sensors, and installing an Android application, refer to the following chapter.

chapter 3 "Operating Procedure"

A/D converter driver is implemented in the sample program. By connecting an analog output sensor to RL78/G1D, you can check the measurement result without modifying the sample program.

Regarding the specification of A/D converter driver, refer to the following section.

section 6.4 "A/D Converter Driver"

I²C driver is implemented in the sample program. This driver provides a function to access registers of device having I²C interface. You can connect various sensor device to RL78/G1D, and then control its operation and get its measurement data by using this driver. Note that the register specification of sensor device is different from each other, so it is necessary to refer its specification document.

Regarding the specification of I²C converter driver, refer to the following section.

section 6.3 "I²C Driver"
In the default implementation of the sample program, device drive for RGB light sensor Renesas ISL29125 is enabled. If you use another sensor device, it is necessary to implement a device driver to use it.

Regarding the specification of ISL29125 device driver, refer to the following section.

section 6.2 "Device Driver"

Regarding the operation sequence of ISL29125 device driver, refer to the following chapter.

chapter 5 "Sensor Control"
2. Specification

2.1 Software Composition

This section explains the software composition of the sample program.

- **BLE application**: manages BLE communication
- **Sensor application**: manages GPIO and sensors
- **Security Library**: controls security of BLE communication
- **Sensor Profile**: controls GATT of BLE communication
- **BLE Protocol Stack**: provides BLE protocol functionalities
- **Kernel**: provides Kernel functionalities
- **Data Flash Library**: controls Data Flash
- **Device Driver**: controls I2C slave device
- **Peripheral Drivers**: controls RL78/G1D peripheral functions

BLE Protocol Stack, Kernel, Data Flash Library are provided in library files.

BLE application and Sensor application as well as Security Library, Sensor Profile, Device Driver and Peripheral Drivers are provided in code files, and you can customize them if necessary.

---

**Figure 2-1 Overview of The Sample Program**

The sample program of this application note was made by customized the following sample program. Regarding the sequence of BLE communication and the specification of Security Library, refer to the following application note.


Libraries for evaluating the sample program are included in the package. It is recommended to get the latest libraries when you develop an application. Regarding the detail, refer to the section 4.2 "Getting Libraries".
2.2 Digital and Analog Interface

Figure 2-2 shows the digital and analog interface of RL78/G1D which is used by the sample program.

- **I2C master**: control and get status of I2C slave device
- **A/D converter**: get analog input signal level from sensor
- **GPIO output**: output digital signal
- **GPIO input**: get digital signal
- **External Input Interrupt**: detect edge of digital signal
- **UART for debug**: output message for debugging to a host machine

By changing a setting of Code Generator Plug-in described later, you can change interface used by the sample program.

![Digital and Analog Interface Used](image-url)
2.3 Sensor Profile

This section explains the GATT based profile to control GPIO and sensors.

The specification of the GATT based profile implemented in the sample program is shown below.

Roles:
- A device for controlling GPIO and/or sensor is a server role of the sensor profile.
  A server role has a sensor service.
  In this application note, RL78/G1D is the server role.
- A device for connecting to a sensor profile server is a client role of the sensor profile.
  A client role accesses to a sensor service of the server role to control GPIO and sensor.
  In this application note, Android device is the client role.

Service and Characteristic:
- Sensor Service consists of several characteristics to control GPIO and sensor.
- The client role gets a characteristic value by Characteristic Value Read and changes the value by Characteristic Value Write.
- The server role notifies a characteristic value to the client role by Notification or Indication.
### Table 2-1: Sensor Service Specification

<table>
<thead>
<tr>
<th>Attribute Handle</th>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Renesas Sensor Service</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x000C</td>
<td>Primary Service Declaration (0x2800)</td>
<td>UUID: 7C570001-1449-4D27-9206-BCFDEA46A0FF</td>
</tr>
<tr>
<td><strong>GPIO Mode Characteristic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x000D</td>
<td>Characteristic Declaration (0x2803) Properties: Read (0x02) Value Handle: 0x000E UUID: 7C570002-1449-4D27-9206-BCFDEA46A0FF</td>
<td></td>
</tr>
<tr>
<td>0x000E</td>
<td>GPIO Mode</td>
<td>GPIO Mode (4byte)</td>
</tr>
<tr>
<td><strong>GPIO Value Characteristic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x000F</td>
<td>Characteristic Declaration (0x2803) Properties: Read, Write (0x0A) Value Handle: 0x0010 UUID: 7C570003-1449-4D27-9206-BCFDEA46A0FF</td>
<td></td>
</tr>
<tr>
<td>0x0010</td>
<td>GPIO Value</td>
<td>GPIO Value (4byte)</td>
</tr>
<tr>
<td><strong>GPIO Interrupt Input Characteristic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0011</td>
<td>Characteristic Declaration (0x2803) Properties: Indication (0x20) Value Handle: 0x0012 UUID: 7C570004-1449-4D27-9206-BCFDEA46A0FF</td>
<td></td>
</tr>
<tr>
<td>0x0012</td>
<td>GPIO Interrupt Input</td>
<td>GPIO Interrupt Input (1byte)</td>
</tr>
<tr>
<td>0x0013</td>
<td>Client Characteristic Configuration Descriptor (0x2902) Properties: Read, Write (0x0A) Indication Configuration (2byte)</td>
<td></td>
</tr>
<tr>
<td><strong>Sensor Availability Characteristic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0014</td>
<td>Characteristic Declaration (0x2803) Properties: Read (0x02) Value Handle: 0x0015 UUID: 7C570005-1449-4D27-9206-BCFDEA46A0FF</td>
<td></td>
</tr>
<tr>
<td>0x0015</td>
<td>Sensor Availability</td>
<td>Sensor Availability (1byte)</td>
</tr>
<tr>
<td><strong>Sensor Operation Characteristic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0016</td>
<td>Characteristic Declaration (0x2803) Properties: Read, Write (0x0A) Value Handle: 0x0017 UUID: 7C570006-1449-4D27-9206-BCFDEA46A0FF</td>
<td></td>
</tr>
<tr>
<td>0x0017</td>
<td>Sensor Operation</td>
<td>Sensor Operation (1byte)</td>
</tr>
<tr>
<td><strong>Sensor Notification Interval Characteristic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0018</td>
<td>Characteristic Declaration (0x2803) Properties: Read, Write (0x0A) Value Handle: 0x0019 UUID: 7C570007-1449-4D27-9206-BCFDEA46A0FF</td>
<td></td>
</tr>
<tr>
<td>0x0019</td>
<td>Sensor Notification Interval</td>
<td>Sensor Notification Interval (2byte)</td>
</tr>
<tr>
<td><strong>Sensor Value Characteristic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x001A</td>
<td>Characteristic Declaration (0x2803) Properties: Notification (0x10) Value Handle: 0x001B UUID: 7C570008-1449-4D27-9206-BCFDEA46A0FF</td>
<td></td>
</tr>
<tr>
<td>0x001B</td>
<td>Sensor Value</td>
<td>Sensor Value (16byte)</td>
</tr>
<tr>
<td>0x001C</td>
<td>Client Characteristic Configuration Descriptor (0x2902) Properties: Read, Write (0x0A) Notification Configuration (2byte)</td>
<td></td>
</tr>
</tbody>
</table>
GPIO Mode
Each bit of this value indicates a digital input / output mode of port. A bit of unused port is always 0.

0: Output
1: Input

<table>
<thead>
<tr>
<th>b0</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
<th>b5</th>
<th>b6</th>
<th>b7</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM10</td>
<td>PM11</td>
<td>PM12</td>
<td>PM13</td>
<td>PM14</td>
<td>PM15</td>
<td>PM16</td>
<td>reserved</td>
</tr>
<tr>
<td>PM00</td>
<td>PM01</td>
<td>PM02</td>
<td>PM03</td>
<td>PM20</td>
<td>PM21</td>
<td>PM22</td>
<td>PM23</td>
</tr>
<tr>
<td>PM30</td>
<td>PM40</td>
<td>PM60</td>
<td>PM61</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
</tr>
<tr>
<td>PM120</td>
<td>PM121</td>
<td>PM122</td>
<td>PM123</td>
<td>PM124</td>
<td>reserved</td>
<td>PM137</td>
<td>PM147</td>
</tr>
</tbody>
</table>

GPIO Value
Each bit of this value indicates a digital input / output value of port. A bit of unused port is always 0. A client can change an output value of output mode port by writing to this value. To read an input value of input mode port, write and then read this value.

0: Low
1: High

<table>
<thead>
<tr>
<th>b0</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
<th>b5</th>
<th>b6</th>
<th>b7</th>
</tr>
</thead>
<tbody>
<tr>
<td>P10</td>
<td>P11</td>
<td>P12</td>
<td>P13</td>
<td>P14</td>
<td>P15</td>
<td>P16</td>
<td>reserved</td>
</tr>
<tr>
<td>P00</td>
<td>P01</td>
<td>P02</td>
<td>P03</td>
<td>P20</td>
<td>P21</td>
<td>P22</td>
<td>P23</td>
</tr>
<tr>
<td>P30</td>
<td>P40</td>
<td>P60</td>
<td>P61</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
</tr>
<tr>
<td>P120</td>
<td>P121</td>
<td>P122</td>
<td>P123</td>
<td>P124</td>
<td>reserved</td>
<td>P137</td>
<td>P147</td>
</tr>
</tbody>
</table>

GPIO Interrupt Input
Each bit of this value indicates interrupt input status. Upon occurring an interrupt input, GPIO Value is also updated.

0: No Interrupt
1: Interrupt Generated

<table>
<thead>
<tr>
<th>b0</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
<th>b5</th>
<th>b6</th>
<th>b7</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTP0</td>
<td>reserved</td>
<td>INTP3</td>
<td>reserved</td>
<td>INTP5</td>
<td>INTP6</td>
<td>reserved</td>
<td></td>
</tr>
</tbody>
</table>

GPIO Interrupt Input Indication Configuration
This value controls whether a server role notifies the GPIO Interrupt Input by Indication.

0x0000: stops Indication
0x0002: starts Indication

<table>
<thead>
<tr>
<th>b0:7</th>
</tr>
</thead>
</table>
| [0] Indication Configuration (LSB)
| [1] Indication Configuration (MSB) |
Sensor Availability
Each bit of this value indicates whether sensor is available or not.

0: Not Available
1: Available

<table>
<thead>
<tr>
<th>b0</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
<th>b5</th>
<th>b6</th>
<th>b7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sensor 0</td>
<td>Sensor 1</td>
<td>Sensor 2</td>
<td>Sensor 3</td>
<td>Sensor 4</td>
<td>Sensor 5</td>
<td>Sensor 6</td>
</tr>
</tbody>
</table>

Sensor Operation
Each bit of this value controls sensor operation. A client can change each sensor's operation by writing to this value. A client should write to this value only when Notification of Sensor Value is stopped. If a client writes to this value, it is ignored.

0: Stop
1: Start

<table>
<thead>
<tr>
<th>b0</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
<th>b5</th>
<th>b6</th>
<th>b7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sensor 0</td>
<td>Sensor 1</td>
<td>Sensor 2</td>
<td>Sensor 3</td>
<td>Sensor 4</td>
<td>Sensor 5</td>
<td>Sensor 6</td>
</tr>
</tbody>
</table>

Sensor Notification Interval
This value indicates a notification interval of a sensor measurement value in units of 10 milliseconds. A client can change the interval by writing to this value. A client should write an interval value greater than connection interval. If a client writes to a value less than connection interval, connection interval is set to this value.

<table>
<thead>
<tr>
<th>b0:7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

Sensor Value
This value indicates measurement value of each sensor. A measurement value of unused sensor is always 0.

<table>
<thead>
<tr>
<th>b0:7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>
Sensor Value Notification Configuration

This value controls whether a server role notifies the Sensor Value by Notification.

- 0x0000: stops Notification
- 0x0001: starts Notification

**Table 2-10  Sensor Value Notification Configuration**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>b0:7</td>
<td></td>
</tr>
<tr>
<td>[0]</td>
<td>Notification Configuration (LSB)</td>
</tr>
<tr>
<td>[1]</td>
<td>Notification Configuration (MSB)</td>
</tr>
</tbody>
</table>
2.3.2 Accessing to Sensor Service

Figure 2-4 shows an example flow chart of accessing sensor service to control GPIO of RL78/G1D by a remote device.

At first, a remote device gets I/O mode and signal level of ports and then permit RL78/G1D to send interruption.

To change signal level of each port, a remote device writes it to "GPIO Value" characteristic. Similarly, to get signal level of each port, a remote device reads it from "GPIO Value" characteristic.

When input interruption occurs, RL78/G1D notifies a remote device by "GPIO Interrupt Input" characteristic.
Figure 2-5 shows an example flow chart of accessing sensor service to get sensor measurement data by a remote device. At first, a remote device gets which sensor is available and starts sensor operation, and then permits to send measurement data. And if necessary, it changes notification interval of sending measurement data. Measurement data is sent by "Sensor Value" characteristic periodically. If a remote device does not need measurement data, it prohibits to send data and stops sensor operation.
3. Operating Procedure

This chapter explains the operating procedure of the sample program.

3.1 Environment

The necessary hardware and software environment for compiling and evaluating the sample program is as follow:

- **Hardware Environment**
  - **Host**
  - PC/AT™ compatible computer
  - **Device**
  - RL78/G1D Evaluation Board (RTK0EN0001D01001BZ)
    - Android device (Version 4.4 KitKat or later)
    - Analog signal output sensor
    - 1ªC slave sensor device
  
  Note: Regarding sensors used for evaluation, refer to section 3.4 "Connecting Sensors".

- **Tool**
  - Renesas On-chip Debugging Emulator E1 (ROE000010KCE00)

- **Software Environment**
  - Windows®10
  - Renesas CS+ for CC V6.01.00 / Renesas CC-RL V1.06.00
  - Renesas Flash Programmer v3.05.01
  - Tera Term Pro (or Terminal software which can connect to serial port)
  - UART-USB conversion device driver

Note: It may be that device driver for UART-USB conversion IC FT232RL is requested when you connect RL78/G1D Evaluation Board to Host first time. In this case, you can get the device driver from below website.

- FTDI (Future Technology Devices International) - Drivers
  http://www.ftdichip.com/Drivers/D2XX.htm

- **Software Library**
  - BLE Protocol Stack: Bluetooth Low Energy Protocol Stack V1.21
  - Data Flash Library: EEPROM Emulation Library Pack02 for CC-RL Compiler Ver1.01

Note: There software libraries are included in the package. And you can get them from Renesas web site. To get the libraries, refer to section 4.2 "Getting Libraries".
3.2 Slide Switch Setting

Figure 3-1 shows the slide switches of RL78/G1D Evaluation Board.

![Slide Switches of RL78/G1D Evaluation Board](image)

Table 3-1 shows the slide-switch setting to evaluate the sample program.

<table>
<thead>
<tr>
<th>Switch</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW7</td>
<td>2-3 connected (right)</td>
<td>Power is supplied from DC/USB VBUS via a regulator. If 1-2 is connected (left), power is directly supplied from a battery.</td>
</tr>
<tr>
<td>SW8</td>
<td>2-3 connected (right)</td>
<td>Power is supplied from USB VBUS to a regulator. If 1-2 is connected, power is supplied from DC to a regulator.</td>
</tr>
<tr>
<td>SW9</td>
<td>2-3 connected (right)</td>
<td>Connected to the USB device.</td>
</tr>
<tr>
<td>SW10</td>
<td>1-2 connected (left)</td>
<td>Power is supplied to the module.</td>
</tr>
<tr>
<td>SW11</td>
<td>2-3 connected (right)</td>
<td>Power is supplied from a source other than the E1 debugger (3.3V).</td>
</tr>
<tr>
<td>SW12</td>
<td>2-3 connected (right)</td>
<td>Unused</td>
</tr>
<tr>
<td>SW13</td>
<td>1-2 connected (left)</td>
<td>USB interface is connected</td>
</tr>
</tbody>
</table>

Regarding the slide-switch of the evaluation board, refer to the section 6.1 "Power Line System" in RL78/G1D Evaluation Board User's Manual (R30UZ0048).
3.3 Writing a Firmware

Figure 3-2 shows the overview of writing a firmware.

To write a firmware, use the E1 Emulator connected to host machine, and then execute Renesas Flash Programmer on the host machine.

![Figure 3-2 Overview of writing a firmware to RL78/G1D](image)


How to write a firmware to RL78/G1D evaluation board is shown below.

1. Connect E1 emulator to the evaluation board and to host machine.
2. Supply power to the evaluation board via a DC jack or USB interface.
3. Start Renesas Flash Programmer and create a project in accordance with the following steps.
   Once you created a project, you can skip to execute these steps.
   3-1. Select [File] → [Create a new project].
   3-2. Select [RL78] as a Microcontroller, input a project name and click [Connect] in [Create New Project] dialog.

4. Prevent erasing Block 254, 255 in Code Flash memory according to the following steps.
   In RL78/G1D Module, Shipping Check Flag is written in Block 254 and Device Address is written in Block 255 respectively.
   4-1. Select [Operation Setting] tab and select [Erase Selected Blocks] at [Erase Option].
4-2. Select [Block Setting] tab and uncheck each [Erase], [P.V] of Block254, 255.

5. Select [Operation] table and specify the following firmware at [Program File].
   - ROM_File/R5F11AGJ_Sensor.hex

6. Click [Start] button to start writing the firmware.

7. Disconnect E1 Emulator and Power Supply from the evaluation board.
3.4 Connecting Sensors

Figure 3-3 shows the external extension Interface CN4 of RL78/G1D Evaluation Board. And CN4 consists of Pin1 to Pin26.

![Figure 3-3   External Extension Interface of RL78/G1D Evaluation Board](image)

Table 3-2 shows the external extension interface of RL78/G1D Evaluation Board.

The sample program uses digital I/O ports, analog input port for A/D converter, and serial data bus and clock for I²C master of RL78/G1D. In addition, it uses LEDs and switches of the evaluation board.

### Table 3-2 External Extension Interface of RL78/G1D Evaluation Board

<table>
<thead>
<tr>
<th>Pin</th>
<th>port of RL78/G1D</th>
<th>I/O of Board</th>
<th>RL78/G1D Functionality used by the program</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P30/INTP3</td>
<td>FT232RL</td>
<td>P30/INTP3 Digital Input, Interrupt Input</td>
</tr>
<tr>
<td>2</td>
<td>VCC</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>P61/SDAA0</td>
<td>-</td>
<td>SDAA0 Serial Data Bus for I²C master</td>
</tr>
<tr>
<td>4</td>
<td>GND</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>P23/ANI3</td>
<td>SW3 Note2</td>
<td>P23 Digital Input</td>
</tr>
<tr>
<td>6</td>
<td>P10/SC00/SCL00</td>
<td>SW6-1</td>
<td>P10 Digital Input</td>
</tr>
<tr>
<td>7</td>
<td>P147/ANI18</td>
<td>LED2</td>
<td>P147 Digital Output</td>
</tr>
<tr>
<td>8</td>
<td>GPIO1/TXSELL_RF</td>
<td>SW6-2 Note1</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>P03/ANI16/Rx01</td>
<td>LED3</td>
<td>P03 Digital Output</td>
</tr>
<tr>
<td>10</td>
<td>GPIO0/TXSELH_RF</td>
<td>SW6-3 Note1</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>P60/SDA0</td>
<td>LED4</td>
<td>SCLA0 Serial Clock for I²C master</td>
</tr>
<tr>
<td>12</td>
<td>P02/ANI1/7/Tx01</td>
<td>SW6-4</td>
<td>P02 Digital Input</td>
</tr>
<tr>
<td>13</td>
<td>P22/ANI2</td>
<td>SW4</td>
<td>P22 Digital Input</td>
</tr>
<tr>
<td>14</td>
<td>P12/SI00/Tx00/TO00/TX00</td>
<td>-</td>
<td>Tx00 UART for debug</td>
</tr>
<tr>
<td>15</td>
<td>P120/ANI19</td>
<td>LED1</td>
<td>ANI19 Analog Input for A/D conversion</td>
</tr>
<tr>
<td>16</td>
<td>P11/SI00/Rx00/TO00/TX00/SDA00</td>
<td>FT232RL</td>
<td>-</td>
</tr>
<tr>
<td>17</td>
<td>VCC</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>SW1 Note1</td>
<td>-</td>
</tr>
<tr>
<td>19</td>
<td>GND</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>P16/TI01/TO01/INTP5</td>
<td>SW2</td>
<td>P16/INTP5 Digital Input, Interrupt Input</td>
</tr>
<tr>
<td>21</td>
<td>P40/TO05</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>RESET</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>24</td>
<td>5V</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>GND</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>26</td>
<td>GND</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Note1: SW1, SW3, SW6-2 and SW6-3 cannot be used, because they are not connected to RL78/G1D.
Note2: To use SW3, external pull-up resistor is required.
Connect sensor to RL78/G1D Evaluation Board. You can evaluate the sample program without sensor.

- I2C slave sensor device
  
  RGB Light sensor - Renesas ISL29125

  e.g. SparkFun RGB Light sensor ISL29125
  https://www.sparkfun.com/products/12829

- Analog signal output device
  variable resistor 50k ohm

Note: In the sample program, the device driver for RGB Light Sensor ISL29125 is implemented. If you use another device, it is necessary to replace it with new device driver.

Regarding the design information of controlling sensor, refer to the following chapters.

  chapter 5 "Sensor Control"

  chapter 6 "Functions"

Connect ISL29125 module and a variable resistor to RL78/G1D Evaluation Board in accordance with Figure 3-4.

![Figure 3-4 Connecting Sensors with RL78/G1D Evaluation Board](image-url)
3.5 Installing Application

Install Android application *BleSensor* to Android device.

How to install *BleSensor* is shown below.

1. To install *BleSensor*, allow installation of application from unknown sources in "Settings" → "Security" → "Unknown sources".
2. Send the following package file from PC to Android device by e-mail.
   - Android_File/BleSensor.apk
3. Receive the e-mail by Android device and execute the attached package file.
4. Start to install *BleSensor*.

![Figure 3-5 Installing Android Application](image)

5. Confirm that installing *BleSensor* is completed.
6. If you use Android OS 6 or later, you should give some permissions to *BleSensor*.

   Go to "Settings" → "Apps & notifications" → "App info" → "BleSensor" → "Permissions" and then enable "Location" and "Storage".

![Figure 3-6 Permission Settings](image)
3.6 Establishing a Connection
Establish a BLE connection between Android device and RL78/G1D by using BleSensor.

How to establish a connection is shown below.

1. Enable Bluetooth in "Settings" → "Bluetooth".
2. Start Android application installed in section 3.5.
   - It shows the device search display and starts Scan to search devices automatically.
   - In this display, connectable devices and their RSSI: Received Signal Strength Indicator are displayed.

   ![Device Search Display](image)

   **Figure 3-7 Device Search Display**

3. Select "RL78/G1D Sensor" in the result of searching device to establish a connection.
   - If a connection is established to a device which does not have the sensor service, the application disconnects and restarts to search devices.
3.7 Controlling GPIO

Control GPIO of RL78/G1D by operating Android device.

How to control GPIO is shown below.

1. Upon establishing a connection to RL78/G1D, Android application shows the GPIO control display.

   In this display, port name, I/O port mode, and digital signal level of each port are displayed respectively.

2. By changing signal level of output port "P03" in the GPIO control display, you can see that LED3 light state of the evaluation board is changed.

3. By pushing SW2 on the evaluation board, you can see that signal level of input port "P16" in the GPIO control display is changed.

   If you push the SW2, P16 becomes low level. If you release the SW2, P16 becomes high level.

4. By pushing SW4 on the board and tap Read button in the display, you can see that signal level of input port "P22" is changed to Low.

   Then, by releasing the SW4 and tap Read button, you can see that signal level of "P22" is High.
3.8 Confirming Sensor Measurement Data

Confirm measurement data of sensor connected to RL78/G1D evaluation board.

How to confirm sensor measurement data is shown below.

1. By selecting SENSOR TAB on the GPIO control display, Android application shows Sensor Measurement display.

   In this display, line graph of measurement data, check-box to control each sensor operation, slider to change a notification interval are displayed.

   The following sensors are assigned to each sensor number.

   Sensor 0: A/D converter
   Sensor 1: ISL29125 RGB Light Sensor (Green)
   Sensor 2: ISL29125 RGB Light Sensor (Red)
   Sensor 3: ISL29125 RGB Light Sensor (Blue)

   Sensor measurement data is saved as a CSV: Comma Separated Values formatted log file.

   ![Figure 3-9 Sensor Measurement Display](image)

   By selecting GPIO TAB, Android application shows GPIO control display again.

2. By putting a check in Sensor0, RL78/G1D starts A/D conversion. Conversion result is displayed by black line in the graph.

   By turning a variable resistor, you can see that the result of A/D conversion changes.

3. By putting a check in Sensor1, Sensor2, Sensor3 respectively, RGB Light sensor starts to measure each brightness of G: Green, R: Red, and B: Blue respectively. Each measurement result is displayed by green, red and blue line in the graph.

   By changing a brightness around the RGB light sensor, you can see that measurement result changes.

4. By moving slider, RL78/G1D changes the interval of sending measurement data.

5. By selecting GPIO TAB, Android application shows GPIO control display again.

6. If you push a back button of Android device, the application disconnects and goes back to the device search display.
3.9 Confirming Sensor Measurement Log

Confirm sensor measurement data log saved in Android device.

How to confirm sensor measurement data log is shown below.

1. Connect Android device to PC and select MTP format.
2. Start Explorer on PC. Confirm that there is a folder \textit{BleSensor} in an internal storage of Android device, and there may be a log file named with the following name format. In the name format, \textit{Y,M,D,H,M, and S} are the date and time of establishing a connection.

\textbf{File Name Format: log\_YYYY\_MM\_DD\_HH\_MM\_SS.csv}

3. Measurement data in the log file is recorded with the following format. You can confirm the log content by using text editor or spread sheet software. In data format, timestamp is a date and time of receiving measurement data, sensor0 to sensor7 are unsigned 2byte measurement data of each sensor.

\textbf{Data Format: timestamp,sensor0,sensor1,sensor2,sensor3,sensor4,sensor5, sensor6,sensor7}

Figure 3-10 shows an example of sensor measurement log which is output by \textit{BleSensor}.

\begin{verbatim}
2018/06/05 11:04:46,380,0,0,0,0,0,0,0
2018/06/05 11:04:47,380,1165,0,0,0,0,0,0
2018/06/05 11:04:47,380,1156,797,0,0,0,0,0
2018/06/05 11:04:48,380,1005,773,604,0,0,0,0
2018/06/05 11:04:49,380,948,654,562,0,0,0,0
2018/06/05 11:04:50,380,1161,819,594,0,0,0,0
2018/06/05 11:04:51,380,1089,790,634,0,0,0,0
2018/06/05 11:04:52,381,1106,790,622,0,0,0,0
2018/06/05 11:04:53,494,1090,779,627,0,0,0,0
\end{verbatim}

\textbf{Figure 3-10 Example Log of Sensor Measurement Data}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{example_log.png}
\caption{Example Log of Sensor Measurement Data}
\end{figure}
4. Building Procedure

This chapter explains the building procedure of the sample program.

4.1 File Composition

In the package of the sample program, not only code files and firmware for RL78/G1D but also package file and project for Android device are included.

Although libraries of BLE Protocol Stack and Data Flash Library are included in the package, it is recommended to use the latest libraries when you develop an application.

To get the libraries, refer to section 4.2 "Getting Libraries".

File and folder composition of the sample program is shown below.

```
Android_BleSensor
  ├─ Android_BleSensor_V1_0_5.pdf
  └─ Android_BleSensor_V1_0_5.zip

RL78G1D_Sensor
  └─ Android_BleSensor
      ├─ BleSensor.apk
      └─ ROM_File
          ├─ R5F11AGJ_Sensor.hex
          └─ R5F11AGJ_Sensor(console_lvl4).hex

Project_Source
  └─ bleip
      └─ src
          └─ common
              └─ co_bt.h
          └─ rwble
              └─ rwble.h
                  └─ rwble_config.h
  └─ rBLE
      └─ src
          └─ include
              └─ rble.h
                  └─ rble_api.h
                      └─ rble_rwlock.h
          └─ sample_app
              └─ console.c
                  └─ console.h
                      └─ rble_sample_app_peripheral.c
                          └─ rble_sample_app_peripheral.h
                              └─ rble_sample_app_sensor.c
                                  └─ rble_sample_app_sensor.h
                                      └─ secdb.c
                                          └─ secdb.h
                                              └─ secdb.c
                                                  └─ secdb.h
                                              └─ seclib.c
                                                  └─ seclib.h
                                                      └─ sample_profile
                                                          └─ sens
                                                              └─ sens.c
                                                                  └─ sens.h
                                                                  └─ renesas
                                                                      └─ lib
                                                                          └─ BLE_CONTROLLER_LIB_CCRL.lib

```

File and folder composition of the sample program is shown below.

Android_BleSensor
  └─ Android_BleSensor_V1_0_5.pdf
  └─ Android_BleSensor_V1_0_5.zip

RL78G1D_Sensor
  └─ Android_BleSensor
      ├─ BleSensor.apk
      └─ ROM_File
          ├─ R5F11AGJ_Sensor.hex
          └─ R5F11AGJ_Sensor(console_lvl4).hex

Project_Source
  └─ bleip
      └─ src
          └─ common
              └─ co_bt.h
          └─ rwble
              └─ rwble.h
                  └─ rwble_config.h
  └─ rBLE
      └─ src
          └─ include
              └─ rble.h
                  └─ rble_api.h
                      └─ rble_rwlock.h
          └─ sample_app
              └─ console.c
                  └─ console.h
                      └─ rble_sample_app_peripheral.c
                          └─ rble_sample_app_peripheral.h
                              └─ rble_sample_app_sensor.c
                                  └─ rble_sample_app_sensor.h
                                      └─ secdb.c
                                          └─ secdb.h
                                              └─ secdb.c
                                                  └─ secdb.h
                                              └─ seclib.c
                                                  └─ seclib.h
                                                      └─ sample_profile
                                                          └─ sens
                                                              └─ sens.c
                                                                  └─ sens.h
                                                                  └─ renesas
                                                                      └─ lib
                                                                          └─ BLE_CONTROLLER_LIB_CCRL.lib

BLE Protocol Stack
  Its library files are included.
  When you develop, get the latest libraries.

BLE Application

Sensor Application
  If you use other I2C device, modify this.

Security Library

Sensor Profile

BLE Protocol Stack
Its library files are included. When you develop, get the latest libraries.

Peripheral Driver
These are generated by Code Generator Plug-in

Data Flash Library
Its library files are included. When you develop, get the latest libraries.

Device Driver
If you use other I2C device, add driver for it.

Device Driver
If you use other I2C device, add driver for it.
4.2 Getting Libraries

To compile a firmware of the sample program, libraries are required. Although these libraries are included in the package, it is recommended to get the latest libraries when you develop an application.

How to get and set the latest libraries is shown below.

1. Download the library package from the following URL.

**BLE Protocol Stack:**

Bluetooth Low Energy Protocol Stack V1.21

**Data Flash Library:**

EEPROM Emulation Library Pack02 Package Ver.2.00(for CA78K0R/CC-RL Compiler) for RL78 Family

2. Copy the following files included in the downloaded library packages.

**BLE Protocol Stack:**

BLE_Software_Ver_x_xx/RL78_G1D/Project_Source/rBLE/src/include/rble.h
BLE_Software_Ver_x_xx/RL78_G1D/Project_Source/rBLE/src/include/rble_api.h
BLE_Software_Ver_x_xx/RL78_G1D/Project_Source/renesas/lib/BLController_LIB_CCRL.lib
BLE_Software_Ver_x_xx/RL78_G1D/Project_Source/renesas/lib/BLE_HOST_lib_CCRL.lib
BLE_Software_Ver_x_xx/RL78_G1D/Project_Source/renesas/lib/BLE_rBLE_lib_CCRL.lib

**Data Flash Library:**

EEL/CCRL_100/EEL/lib/eel.lib
EEL/CCRL_100/EEL/lib/eel.h
EEL/CCRL_100/EEL/lib/eel_types.h
EEL/CCRL_100/FDL/lib/librl78/fdl.lib
EEL/CCRL_100/FDL/lib/incrl78/fdl.h
EEL/CCRL_100/FDL/lib/incrl78/fdl_types.h

3. Place the above files to the following library folders.

```
Project_Source
    ├── BLE
        │   └── src
        │       └── include
        │               └── rble.h
        │               └── rble_api.h
        │               Protocol Stack rBLE definitions - header file
        │               Protocol Stack rBLE API - header file
        ├── renesas
        │   └── lib
        │       └── BLController_LIB_CCRL.lib
        │       └── BLE_HOST_lib_CCRL.lib
        │       └── BLE_rBLE_lib_CCRL.lib
        │       Protocol Stack Controller Layer - library file
        │       Protocol Stack Host Layer - library file
        │       Protocol Stack rBLE Layer - library file
        └── driver
            └── dataflash
                └── cc_rl
                    └── eel.h
                    └── eel.lib
                    └── eel_types.h
                    └── fdl.h
                    └── fdl.lib
                    └── fdl_types.h
                    Data Flash Library EEPROM Emulation - header file
                    Data Flash Library EEPROM Emulation - library file
                    Data Flash Library EEPROM Emulation type definition -
                    Data Flash Library - header file
                    Data Flash Library - library file
                    Data Flash Library type definition - header file
```
4.3 Building a Firmware

You can use CS+ for CC to build a firmware of the sample program. As a result of building, HEX-formatted firmware named R5F11AGJ_Sensor.hex is generated.

How to build a firmware by using CS+ for CC is shown below.

1. Start CS+ for CC and open the project named BLE_Peripheral.mtpj in the following folder by [File]→[Open…].
   - Project_Source/renesas/tools/project/CS_CCR/BL/Peripheral
2. To build a firmware, select [Build]→[Rebuild Project].
3. Confirm that no error occurs, it succeeds in building a firmware.

![Screenshot of CS+ for CC interface]

4. Confirm that the firmware named R5F11AGJ_Sensor.hex is generated in the following folder.
   - Project_Source/renesas/tools/project/CS_CCR/BL/Peripheral/R5F11AGJ_Sensor/DefaultBuild

Confirm there is no error.
4.4 Configuring Peripherals

To control peripheral functions of RL78/G1D, you can use drivers generated by Code Generator Plug-in of CS+ for CC. By default setting, the sample program uses the following peripheral functions.

- **Common/Clock Generator**: e.g. Operation mode setting and High-speed OCO clock setting
- **Port Function**: e.g. In/Out, Default output, and Pull-up setting
- **Interrupt Function**: e.g. Edge detection setting
- **A/D Converter**: e.g. Analog input selection, VREF(+-) setting, and Resolution setting
- **Serial Interface IICA**: e.g. Transfer clock setting
- **Serial Array Unit**: e.g. Transmit and Receive settings and Baud rate setting

Figure 4-1 shows Code Generator Plug-in of CS+ for CC.

There is each peripheral function setting in the Code Generator of Project Tree. You can change its settings. After changing the settings, click the "Generate Code" button to update code files.

![Figure 4-1 Code Generator Plug-In](image)

Regarding the specification of generated functions, refer to Smart Manual of CS+ for CC. To display Smart Manual, select [View] → [Smart Manual] in menu bar.
5. Sensor Control

This chapter explains the operation of the following modules.

Code files of each module are shown below.

- BLE application: Project_Source/rBLE/src/sample_app/rble_sample_app_peripheral.c
- Security Library: Project_Source/rBLE/src/sample_app/seclib/seclib.c
- Sensor application: Project_Source/rBLE/src/sample_app/r_sample_app_sensor.c
- Sensor Profile: Project_Source/rBLE/src/sample_profile/sen/sens.c
- ISL29125 driver: Project_Source/reneas/src/sensor/ISL29125.c
- Peripheral Driver (IICA0): Project_Source/renesas/src/cg_src/r_cg_iica.c, r_cg_iica_user.c
- Peripheral Driver (ADC): Project_Source/reneas/src/cg_src/r_cg_adc.c, r_cg_adc_user.c

Figure 5-1 Module Related to Control Sensor

Figure 5-2 show the flow chart of sensor application.

BLE application executes an operation to connect, disconnect, and encrypts and decrypts data.

Regarding operations of Sensor application, refer to the following pages.
5.1 Sensor Initialization

Figure 5-3 shows the sensor initialization sequence.

This sequence is executed only once after resetting RL78/G1D. Sensor application initializes peripherals of RL78/G1D such as A/D converter and I2C.

The device driver for RGB light sensor ISL29125 is implemented in the sample program.

If you use another I2C device, it is necessary to implement a device driver for each device and replace ISL29125 driver with it.
5.2 Sensor Profile Start

Figure 5-4 shows the sensor profile start sequence.

This sequence is executed after establishing a connection to a remote device.

Sensor application starts sensor profile and updates characteristics values of sensor service to the latest status. By starting sensor profile, a remote device can access to the sensor service.

![Diagram showing the sensor profile start sequence](image-url)
5.3 Sensor Operation Start

Figure 5-5 shows the sensor operation start sequence.

This sequence is executed by the request from a remote device.

Sensor application starts the measurement operation of ISL29125 by the request from a remote device. After starting ISL29125 operation, it updates a characteristic value for indicating a sensor operation status.

![Diagram of Sensor Operation Start](image)

The device driver for RGB light sensor ISL29125 is implemented in the sample program.

If you use another I²C device, it is necessary to implement a device driver for each device and replace ISL29125 driver with it.
5.4 Sensor Measurement Data Notification

Figure 5-6 shows the sensor measurement data notification sequence. This sequence is executed by the permission from a remote device.

Sensor application notifies measurement data of A/D conversion and ISL29125 periodically by the request from a remote device.

![Diagram of Sensor Measurement Data Notification]

The device driver for RGB light sensor ISL29125 is implemented in the sample program. If you use another I²C device, it is necessary to implement a device driver for each device and replace ISL29125 driver with it.
5.5 Sensor Profile Stop

Figure 5-7 shows the sensor profile stop sequence.

This sequence is executed after disconnection.

Sensor application stops sensor profile and notifies that sensor profile stopped. And then BLE application restarts Advertising.

![Sensor Profile Stop Diagram]

**Figure 5-7 Sensor Profile Stop**
6. Functions

This chapter explains the function specifications of each module such as sensor profile, device driver and I²C driver.

When you implement a device driver for using another I²C device, refer to this chapter as necessary.

6.1 Sensor Profile

The specification of sensor profile is shown below.

Regarding the implementation of sensor profile, refer to the following file.

Sensor Profile: Project_Source/rBLE/src/sample_profile/sen/sens.c

### 6.1.1 R_SENS_Enable

```c
RBLE_STATUS R_SENS_Enable( uint16_t conhdl, SENS_EVENT_HANDLER callback );
```

This function enables sensor profile server.

It is necessary to execute after each establishing a connection.

When sensor profile server event occurs, sensor Profile executes a callback function registered by this function.

**Parameters:**

- **conhdl** Connection Handle
  - Set a handle notified by RBLE_GAP_EVENT_CONNECTION_COMP event

- **callback** Callback function to notify that sensor profile server event occurs
  - `void (*SENS_EVENT_HANDLER)(SENS_EVENT *event);`

**Event**

- **Sensor Profile Server Event**
  - Regarding the definition of SENS_EVENT structure, refer to `sens.h`.

**Return:**

- **RBLE_OK** Success
- **others** Regarding the definitions of error codes, refer to `RBLE_STATUS_enum` in `rble.h`.

### 6.1.2 R_SENS_Disable

```c
RBLE_STATUS R_SENS_Disable( uint16_t conhdl );
```

This function disables sensor profile server.

It is necessary to execute after each disconnection.

**Parameters:**

- **conhdl** Connection Handle
  - Set the connection handle same as set by R_SENS_Enable()

**Return:**

- **RBLE_OK** Success
- **others** Regarding the definitions of error codes, refer to `RBLE_STATUS_enum` in `rble.h`.

### 6.1.3 R_SENS_SetData

```c
void R_SENS_SetData( uint16_t charhdl, void* charval );
```

This function changes a characteristic value of sensor service.

**Parameters:**

- **charhdl** Attribute Handle of Characteristic Value to be changed
- **charval** New Characteristic Value

**Return:**

- **None**
### 6.1.4 R_SENS_Indication

```c
void R_SENS_Indication( uint16_t charhdl );
```

This function sends an Indication to remote device. After remote device permits to send an Indication, own device can send it.

**Parameters:**
- `charhdl` Attribute Handle of Characteristic Value to be sent by Indication

**Return:**
- None

### 6.1.5 R_SENS_Notification

```c
void R_SENS_Notification( uint16_t charhdl );
```

This function sends a Notification to remote device. After remote device permits to send a Notification, own device can send it.

**Parameters:**
- `charhdl` Attribute Handle of Characteristic Value to be sent by Notification

**Return:**
- None

### 6.1.6 R_SENS_Response

```c
void R_SENS_Response( uint16_t charhdl, uint8_t status );
```

This function sends a Response for a Write Request from remote device. When a Write Request is received, it is necessary to send a Response by executing this function.

**Parameters:**
- `charhdl` Attribute Handle of Characteristic Value to be written
- `status` Status Code for a Write Request

Regarding the definitions, refer to `RBLE_ATT_ERR_CODE_enum` in `rble_api.h`.

**Return:**
- None
6.2 Device Driver

The device driver for controlling RGB light sensor ISL29125 is implemented in the sample program. It uses I²C driver and accesses registers of ISL29125 by I²C communication.

If you use another I²C device, it is necessary to implement a device driver for each device and replace ISL29125 driver with it.

The specification of ISL29125 driver is shown below.

Regarding the implementation of ISL29125 driver, refer to the following file.

| ISL29125: Project_Source/renesas/src/sensor/ISL29125.c |

6.2.1 R_ISL29125_Init

```c
uint8_t R_ISL29125_Init( r_isl29125_callback_t callback);
```

This function initializes ISL29125. This function check if the device is connected to RL78/G1D via I²C, and then executes device reset and configuration. Moreover, it executes the initialization sequence defined by the device specification such as a calibration. After an asynchronous device control finishes, callback function registered by this function is executed.

**Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>Callback function to notify that asynchronous device control is finished.</td>
</tr>
</tbody>
</table>

```c
def void (*r_isl29125_callback_t)( r_isl29125_opcode_t opcode, uint8_t status, void* data );
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>opcode</td>
<td>Operation Code to identify each device control operation</td>
</tr>
<tr>
<td>status</td>
<td>status of device control operation</td>
</tr>
<tr>
<td>data</td>
<td>returned data from device control operation</td>
</tr>
</tbody>
</table>

**Return:**

0 Success
others Device not present, Device or I²C error, or other error
6.2.2 R_ISL29125_SetModeSync

```
uint8_t R_ISL29125_SetModeSync( uint8_t mode );
```

This function sets the operation setting registers of ISL29125. Device operation mode such as run mode or standby mode is changed in accordance with the argument mode. And, additional settings to be required by the device specification are set. This function returns after completion of setting operation mode. A callback function registered by R_ISL29125_Init() is not executed.

Parameters:
- mode: device mode setting

Return:
- 0: Success
- others: Device or I²C error, or other error

6.2.3 R_ISL29125_SetMode

```
uint8_t R_ISL29125_SetMode( uint8_t mode );
```

This function sets the operation setting registers of ISL29125. Device operation mode such as run mode or standby mode is changed in accordance with the argument mode. And, additional settings to be required by the device specification are set. This function returns without waiting completion of setting operation mode. Completion of setting is notified by a callback function registered by R_ISL29125_Init().

Parameters:
- mode: device mode setting

Return:
- 0: Success
- others: Device or I²C error, or other error

---

**Figure 6-1** Sequence of R_ISL29125_SetModeSync() and R_ISL29125_SetMode()
### 6.2.4 R_ISL29125_GetResultSync

```c
uint8_t R_ISL29125_GetResultSync( r_isl29125_result_t* result );
```

This function gets the measurement result of ISL29125.
Measurement result is stored in a buffer specified by the argument `result`.
This function returns after completion of getting result. A callback function registered by `R_ISL29125_Init()` is not executed.

**Parameters:**
- `result`: sensor measurement result

**Return:**
- 0: Success
- `others`: Device or I2C error, or other error

### 6.2.5 R_ISL29125_GetResult

```c
uint8_t R_ISL29125_GetResult( void );
```

This function gets the measurement result of ISL29125.
This function returns without waiting completion of getting result. The measurement result is notified by a callback function registered by `R_ISL29125_Init()`.

**Parameters:**
- None

**Return:**
- 0: Success
- `others`: Device or I2C error, or other error

---

**Figure 6-2** Sequence of `R_ISL29125_GetResultSync()` and `R_ISL29125_GetResult()`
6.3 I²C Driver

The device driver to use serial interface IICA of RL78/G1D is implemented in the sample program. By using this driver, RL78/G1D works as an I²C master and accesses sensor working as an I²C slave.

The specification of I²C driver is shown below.

Regarding the implementation of I²C driver, refer to the following files.

I²C driver: Project_Source/renesas/src/cg_src/r_cg_iica.c, r_cg_iica_user.c

6.3.1 R_IICA0_Create

```c
void R_IICA0_Create( void );
```

This function initializes Serial Interface IICA of RL78/G1D.

**Parameters:**

- None

**Return:**

- None

6.3.2 R_IICA0_RegisterCallback

```c
void R_IICA0_RegisterCallback( iica0_user_callback_t callback );
```

This function registers a callback function to notify that IICA0 operation is completed. After the following timing, IICA interrupt handler executes a callback function registered by this function.

- I²C write completion
- I²C read completion
- I²C error

**Parameters:**

- `callback`: Callback function to notify that I²C access is completed
  ```c
  void (*iica0_user_callback_t)( iica0_rw_calltype_t type, uint8_t flag );
  ```

  - `type`:
    - IICA0_SENDEND: I²C write completion
    - IICA0_RECEIVEEND: I²C read completion
    - IICA0_ERROR: I²C error

  - `flag`:
    - MD_OK: Success
    - other than MD_OK: Error

**Return:**

- None
### 6.3.3 R_IICA0_Write

**MD_STATUS R_IICA0_Write** *(uint8_t adr, void* buf, uint16_t len, iica0_rw_sync_t sync );*

This function writes data to register of I²C slave device.

The address of I²C slave device is specified by `adr`.

Data in the buffer `buf[1]` and following is written to the register of address specified in `buf[0].`

Regarding I²C communication, refer to Figure 6-3.

This function should be executed when interrupt enables. And this function cannot be executed by interrupt handler.

**Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>adr</code></td>
<td>Device Address set 7bit device address</td>
</tr>
<tr>
<td><code>buf</code></td>
<td>Data Buffer set register address to <code>buf[0]</code> set data to <code>buf[1]</code> and the following</td>
</tr>
<tr>
<td><code>len</code></td>
<td>Access Length (byte) set the sum of register address (1byte) and data length (<code>len &gt;= 2</code>)</td>
</tr>
<tr>
<td><code>sync</code></td>
<td>Synchronous Setting IICA0_SYNC IICA0_ASYNC This function returns after completion of I²C access. This function returns without waiting completion of I²C access. Completion of I²C access is notified by callback function.</td>
</tr>
</tbody>
</table>

**Return:**

- `MD_OK` Success
- other than `MD_OK` Error

---

**Figure 6-3 I²C Accessing by R_IICA0_Write**

![I²C Accessing by R_IICA0_Write](image-url)
### 6.3.4 R_IICA0_Read

```c
MD_STATUS R_IICA0_Read( uint8_t adr, void* buf, uint16_t len, iica0_rw_sync_t sync );
```

This function reads data from register of I2C slave device.

- The address of I2C slave device is specified by `adr`.
- Data is read from the register of address specified in `buf[0]` and then stored to the buffer `buf[1]` and the following.

Regarding I2C communication, refer to Figure 6-4.

This function should be executed when interrupt enables. And this function cannot be executed by interrupt handler.

**Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>adr</code></td>
<td>Device Address set 7bit device address</td>
</tr>
<tr>
<td><code>buf</code></td>
<td>Data Buffer set register address to <code>buf[0]</code></td>
</tr>
<tr>
<td><code>len</code></td>
<td>Access Length (byte) set the sum of register address (1byte) and data length (<code>len &gt;= 2</code>)</td>
</tr>
<tr>
<td><code>sync</code></td>
<td>Synchronous Setting IICA0_SYNC IICA0_ASYNC This function returns after completion of I2C access. This function returns without waiting completion of I2C access. Completion of I2C access is notified by callback function.</td>
</tr>
</tbody>
</table>

**Return:**

- refer to `r_cg_macrodriver.h`
- MD_OK Success
- other than MD_OK Error

---

**Figure 6-4 I2C Accessing by R_IICA0_Read**

[Diagram of I2C access process shown]
6.4 A/D Converter Driver

The A/D converter driver to use A/D converter of RL78/G1D is implemented in the sample program. You can change a configuration of A/D converter by operating a Code Generator Plug-in of CS+ for CC.

The specification of A/D converter driver is shown below.

Regarding the implementation of A/D converter driver, refer to the following files.

- A/D converter : Project_Source/renesas/src/cg/src/r_cg_adc.c, and r_cg_adc_user.c

6.4.1 R_ADC_Create

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_ADC_Create</td>
<td>This function initializes the A/D converter of RL78/G1D.</td>
</tr>
<tr>
<td>Parameters:</td>
<td>None</td>
</tr>
<tr>
<td>Return:</td>
<td>None</td>
</tr>
</tbody>
</table>

6.4.2 R_ADC_GetChannel

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_ADC_GetChannel</td>
<td>This function gets a selected analog input channel of A/D converter.</td>
</tr>
<tr>
<td>Parameters:</td>
<td>None</td>
</tr>
<tr>
<td>Return:</td>
<td>Analog Input Channel</td>
</tr>
</tbody>
</table>

Regarding the returned analog input channel value, refer to subsection 12.3.7 "Analog input channel specification register (ADS)" in RL78/G1D User's Manual: Hardware (R01UH0515).

6.4.3 R_ADC_GetResultSync

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_ADC_GetResultSync</td>
<td>This function A/D conversion and return the result by argument.</td>
</tr>
<tr>
<td>Parameters:</td>
<td>result A/D conversion result</td>
</tr>
<tr>
<td>Return:</td>
<td>0 Success</td>
</tr>
<tr>
<td></td>
<td>others Error</td>
</tr>
</tbody>
</table>
7. Appendix

7.1 UART for Debug

Functions to output message for debugging are implemented in the sample program.

Table 7-1 shows the functions of UART for debug. You can use them as necessary.

<table>
<thead>
<tr>
<th>Function</th>
<th>Example Use-case</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrintError()</td>
<td>for reporting implementation problem</td>
</tr>
<tr>
<td>PrintWarning()</td>
<td>for warning unexpected operation</td>
</tr>
<tr>
<td>PrintInfo()</td>
<td>for checking parameters used by application</td>
</tr>
<tr>
<td>PrintLog()</td>
<td>for checking sequence of application</td>
</tr>
</tbody>
</table>

To enable these functions, change the macro `CONSOLE_LVL` defined in the following file.

- Project_Source/rBLE/src/sample_app/console.h

console.h (line 63):

```
63:    #define CONSOLE_LVL (0)  
```

Depending on the `CONSOLE_LVL`, each function displayed in Table 7-2 is enabled.

<table>
<thead>
<tr>
<th>CONSOLE_LVL</th>
<th>Usable function of UART for debug</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UART for debug is disabled</td>
</tr>
<tr>
<td>1</td>
<td>PrintError() only</td>
</tr>
<tr>
<td>2</td>
<td>PrintError() and PrintWarning</td>
</tr>
<tr>
<td>3</td>
<td>PrintError(), PrintWarning() and PrintInfo()</td>
</tr>
<tr>
<td>4</td>
<td>PrintError(), PrintWarning(), PrintInfo() and PrintLog()</td>
</tr>
</tbody>
</table>

You can confirm message output from UART for debug by a terminal software on PC. Table 7-3 shows the serial communication setting for a terminal software.

<table>
<thead>
<tr>
<th>Item</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial Port</td>
<td>USB Serial Port</td>
</tr>
<tr>
<td></td>
<td>Note that COM number is different from each</td>
</tr>
<tr>
<td></td>
<td>evaluation board</td>
</tr>
<tr>
<td>Port</td>
<td>1,000,000bps</td>
</tr>
<tr>
<td>Baud rate</td>
<td>8bit</td>
</tr>
<tr>
<td>Data Bit Length</td>
<td>None</td>
</tr>
<tr>
<td>Parity</td>
<td>1bit</td>
</tr>
<tr>
<td>Stop Bit Length</td>
<td>None</td>
</tr>
<tr>
<td>Flow Control</td>
<td>None</td>
</tr>
<tr>
<td>New Line</td>
<td>Receive LF</td>
</tr>
<tr>
<td>Terminal Size</td>
<td>over than 128 characters</td>
</tr>
</tbody>
</table>
When Tera Term is used as terminal software, there is no "1,000,000bps" in the drop-down list of Baud Rate. Thus, it is necessary to enter "1000000" to the input box of Baud Rate directly.

Figure 7-1 shows an example message that is output by the sample program built with CONSOLE_LVL=4. You can customize message content as necessary.

In this example message, "Connected" means that a connection is established, "Pairing completed" means that pairing is completed, and "Sensor Notification Enabled" means that operation to notify sensor measurement data is started.
Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

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

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

3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   - The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   - When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

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

5. 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”: Transportation equipment (automobiles, train, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

6. 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 implantation, etc.), or may cause serious property damage (space system; 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. 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. 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 applicable laws and regulations. Renesas Electronics disclaims any and all liability for any damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

9. Renesas Electronics products and technologies shall not be used or be 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.

10. 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.

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

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

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.