Bluetooth® Low Energy Protocol Stack

Sensor Application

R01AN4159EJ0103
Rev.1.03
Dec 21, 2018

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

Document Name		Document No.
RL78/G1D		
	User's Manual: Hardware	R01UH0515
RL78/G1D Evaluation Board		
	User's Manual	R30UZ0048
E1 Emulator		
	User's Manual	R20UT0398
	Additional Document for User's Manual (Notes on Connection of RL78)	R20UT1994
Renesas Flash Programmer V3.05 Flash memory programming software		
	User's Manual	R20UT4307
CC-RL Compiler		
	User's Manual	R20UT3123
Bluetooth Low Energy Protocol Stack		
	User's Manual	R01UW0095
	API Reference Manual: Basics	R01UW0088
	Security Library Application Note	R01AN3777

Contents

1. Overview

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

The sample program is capable of controlling GPIO, A/D conversion, and I^2C 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 I^2C 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 I^2C with a line graph.

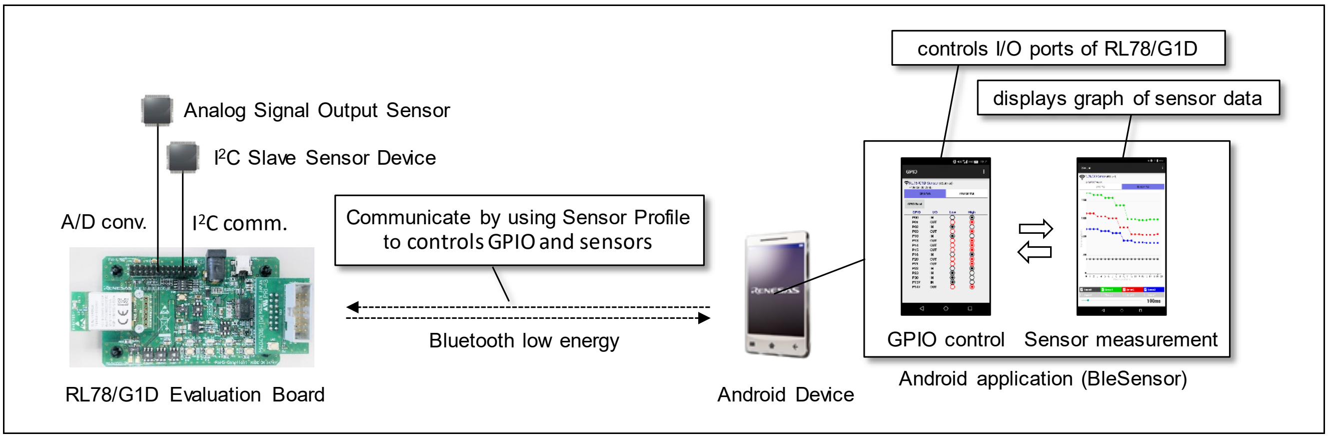


Figure 1-1 Overview of the sample program

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"

I^2C driver is implemented in the sample program. This driver provides a function to access registers of device having I^2C 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 I^2C converter driver, refer to the following section.

section 6.3 "I^2C 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.

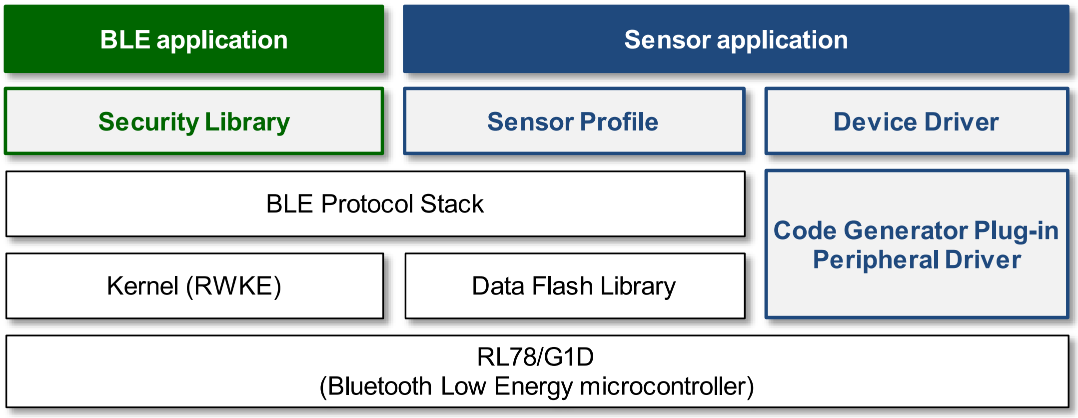


Figure 2-1 Overview of The Sample Program

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.

Bluetooth Low Energy Protocol Stack Embedded Configuration Sample Program (R01AN3319)
https://www.renesas.com/document/scd/bluetooth-low-energy-protocol-stack-embedded-configuration-sample-program

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.

I^2C master:	control and get status of I^2C 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.

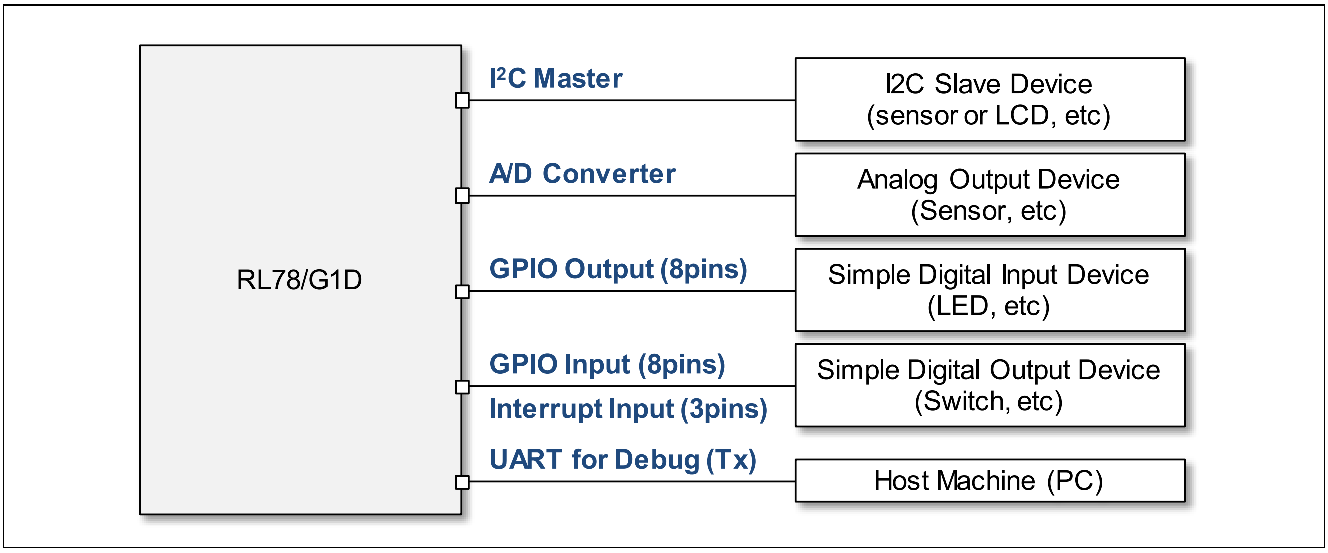


Figure 2-2 Digital and Analog Interface Used

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.

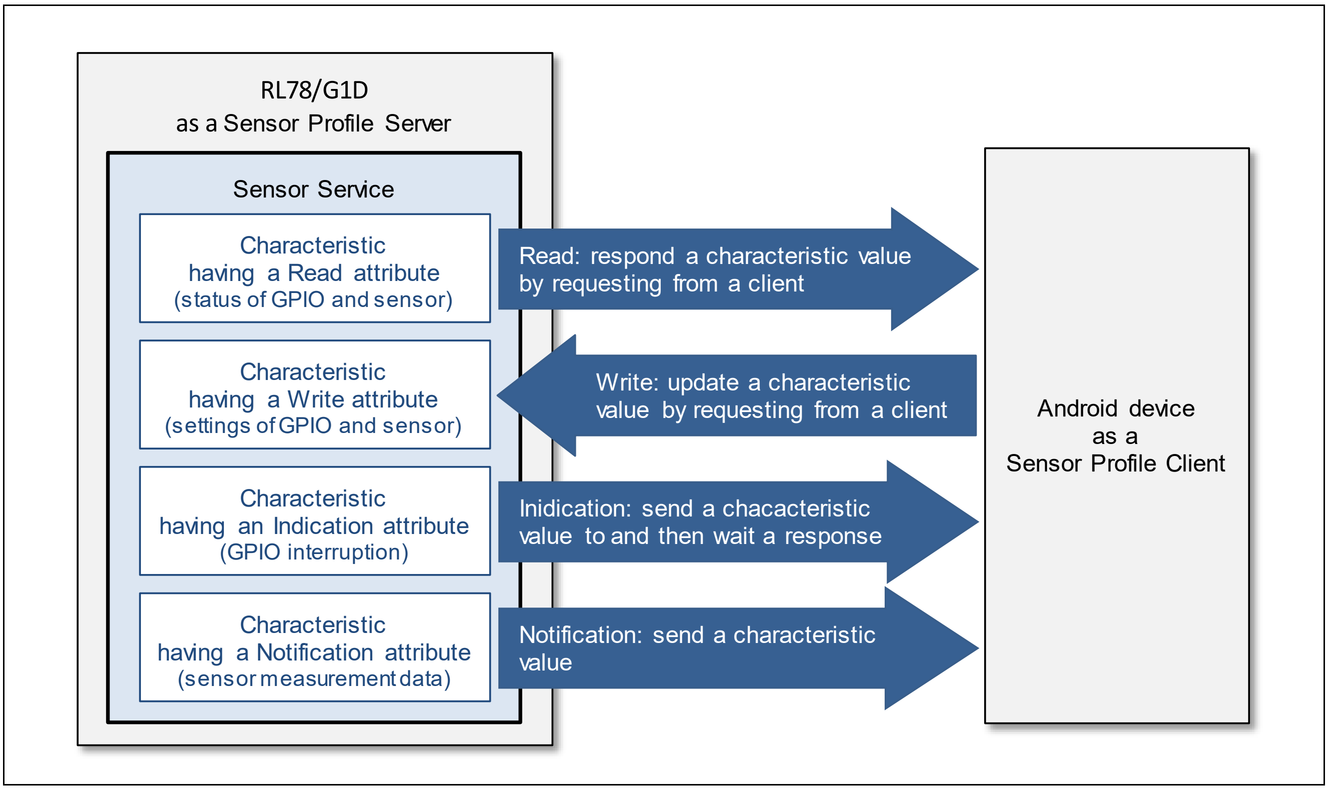


Figure 2-3 Sensor Profile

2.3.1 Sensor Service

Table 2-1 shows the specification of Sensor Service.

Table 2-1 Sensor Service Specification

Attribute Handle	Attribute Type	Attribute Value
Renesas Sensor Service		
0x000C	Primary Service Declaration (0x2800)	UUID: 7C570001-1449-4D27-9206-BCFDEA46A0FF
GPIO Mode Characteristic		
0x000D	Characteristic Declaration (0x2803)	Properties: Read (0x02) Value Handle: 0x000E UUID: 7C570002-1449-4D27-9206-BCFDEA46A0FF
0x000E	GPIO Mode	**GPIO Mode (4byte)**
GPIO Value Characteristic		
0x000F	Characteristic Declaration (0x2803)	Properties: Read, Write (0x0A) Value Handle: 0x0010 UUID: 7C570003-1449-4D27-9206-BCFDEA46A0FF
0x0010	GPIO Value	**GPIO Value (4byte)**
GPIO Interrupt Input Characteristic		
0x0011	Characteristic Declaration (0x2803)	Properties: Indication (0x20) Value Handle: 0x0012 UUID: 7C570004-1449-4D27-9206-BCFDEA46A0FF
0x0012	GPIO Interrupt Input	**GPIO Interrupt Input (1byte)**
0x0013	Client Characteristic Configuration Descriptor (0x2902)	Properties: Read, Write (0x0A) **Indication Configuration (2byte)**
Sensor Availability Characteristic		
0x0014	Characteristic Declaration (0x2803)	Properties: Read (0x02) Value Handle: 0x0015 UUID: 7C570005-1449-4D27-9206-BCFDEA46A0FF
0x0015	Sensor Availability	**Sensor Availability (1byte)**
Sensor Operation Characteristic		
0x0016	Characteristic Declaration (0x2803)	Properties: Read, Write (0x0A) Value Handle: 0x0017 UUID: 7C570006-1449-4D27-9206-BCFDEA46A0FF
0x0017	Sensor Operation	**Sensor Operation (1byte)**
Sensor Notification Interval Characteristic		
0x0018	Characteristic Declaration (0x2803)	Properties: Read, Write (0x0A) Value Handle: 0x0019 UUID: 7C570007-1449-4D27-9206-BCFDEA46A0FF
0x0019	Sensor Notification Interval	**Sensor Notification Interval (2byte)**
Sensor Value Characteristic		
0x001A	Characteristic Declaration (0x2803)	Properties: Notification (0x10) Value Handle: 0x001B UUID: 7C570008-1449-4D27-9206-BCFDEA46A0FF
0x001B	Sensor Value	**Sensor Value (16byte)**
0x001C	Client Characteristic Configuration Descriptor (0x2902)	Properties: Read, Write (0x0A) **Notification Configuration (2byte)**

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 2-2 GPIO Mode

Attribute Handle: 0x000E Properties: Read Size: 4byte

	b0	b1	b2	b3	b4	b5	b6	b7
[0]	PM10	PM11	PM12	PM13	PM14	PM15	PM16	reserved
[1]	PM00	PM01	PM02	PM03	PM20	PM21	PM22	PM23
[2]	PM30	PM40	PM60	PM61	reserved	reserved	reserved	reserved
[3]	PM120	PM121	PM122	PM123	PM124	reserved	PM137	PM147

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 2-3 GPIO Value

Attribute Handle: 0x0010 Properties: Read, Write Size: 4byte

	b0	b1	b2	b3	b4	b5	b6	b7
[0]	P10	P11	P12	P13	P14	P15	P16	reserved
[1]	P00	P01	P02	P03	P20	P21	P22	P23
[2]	P30	P40	P60	P61	reserved	reserved	reserved	reserved
[3]	P120	P121	P122	P123	P124	reserved	P137	P147

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 2-4 GPIO Interrupt Input

Attribute Handle: 0x0012 Properties: Indication Size: 1byte

	b0	b1	b2	b3	b4	b5	b6	b7
[0]	INTP0 (P137)	reserved	reserved	INTP3 (P30)	reserved	INTP5 (P16)	INTP6 (P140)	reserved

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 2-5 GPIO Interrupt Input Indication Configuration

Attribute Handle: 0x0013 Properties: Read, Write Size: 2byte

	b0:7
[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 2-6 Sensor Availability

Attribute Handle: 0x0015 Properties: Read Size: 1byte

	b0	b1	b2	b3	b4	b5	b6	b7
[0]	Sensor 0	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6	Sensor 7

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 2-7 Sensor Operation

Attribute Handle: 0x0017 Properties: Read, Write Size: 1byte

	b0	b1	b2	b3	b4	b5	b6	b7
[0]	Sensor 0	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6	Sensor 7

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 2-8 Sensor Notification Interval

Attribute Handle: 0x0019 Properties: Read, Write Size: 2byte

	b0:7
[0]	Sensor Notification Interval (LSB)
[1]	Sensor Notification Interval (MSB)

Sensor Value

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

Table 2-9 Sensor Value

Attribute Handle: 0x001B Properties: Notification Size: 16byte

	b0:7
[0]	measurement value of sensor 0 (LSB)
[1]	measurement value of sensor 0 (MSB)
[2]	measurement value of sensor 1 (LSB)
[3]	measurement value of sensor 1 (MSB)
:	:
[12]	measurement value of sensor 6 (LSB)
[13]	measurement value of sensor 6 (MSB)
[14]	measurement value of sensor 7 (LSB)
[15]	measurement value of sensor 7 (MSB)

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

Attribute Handle: 0x001C Properties: Read, Write Size: 2byte

	b0:7
[0]	Notification Configuration (LSB)
[1]	Notification Configuration (MSB)

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.

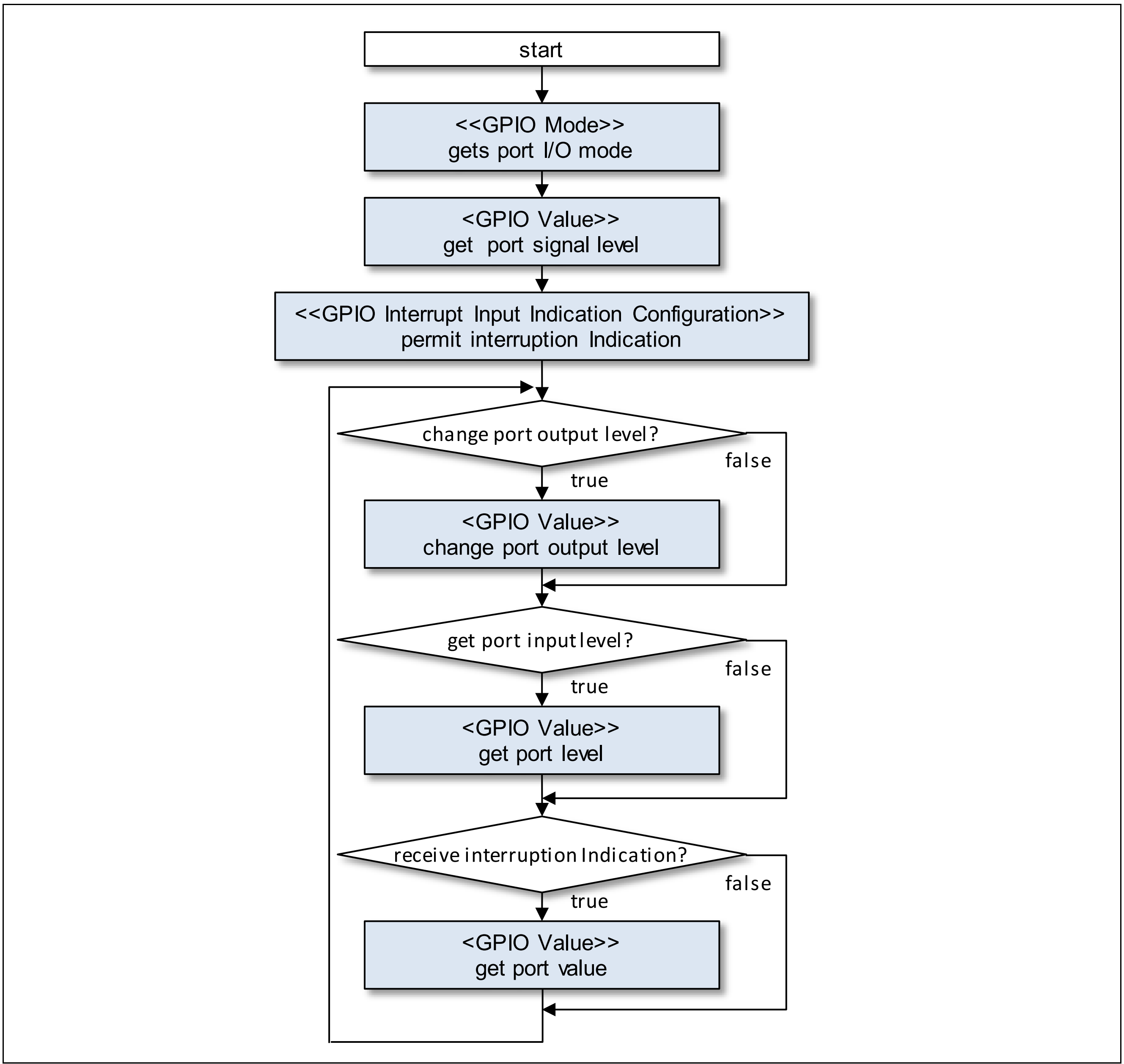


Figure 2-4 Example of Controlling GPIO

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.

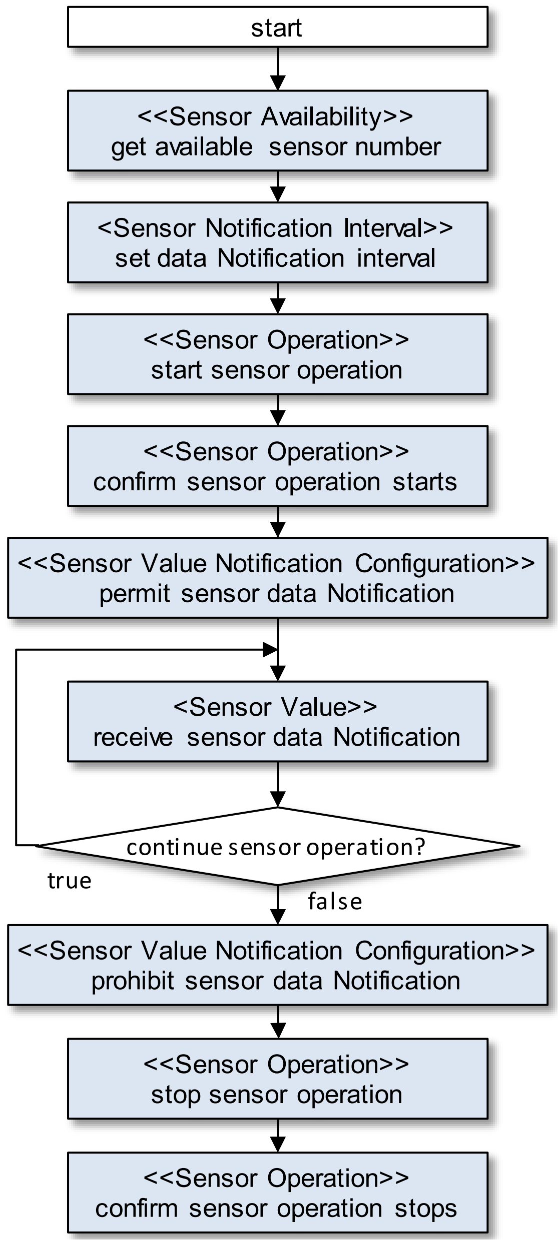


Figure 2-5 Example of Controlling Sensor

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 Note
    - I²C slave sensor device Note

    Note: Regarding sensors used for evaluation , refer to section 3.4 "Connecting Sensors".
  - Tool
    - ✧ Renesas On-chip Debugging Emulator E1 (R0E000010KCE00)

- 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.

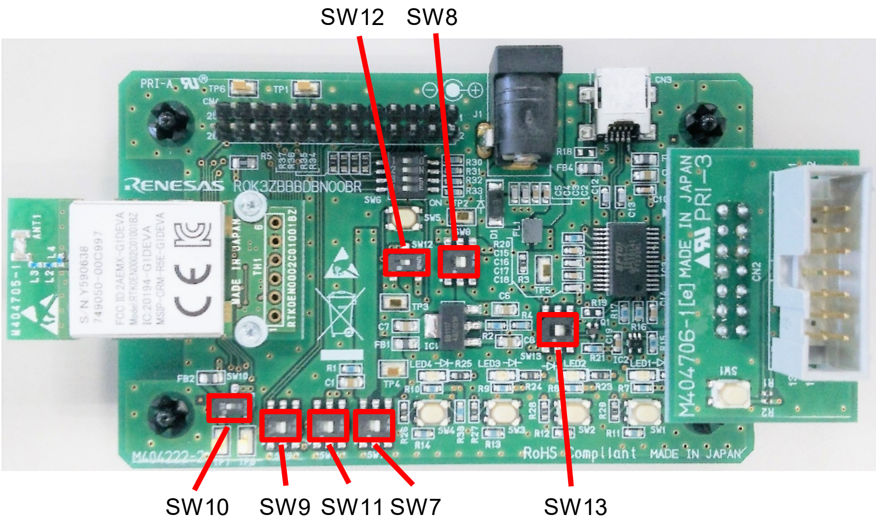


Figure 3-1 Slide Switches of RL78/G1D Evaluation Board

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

Table 3-1 Slide Switch Settings

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

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.

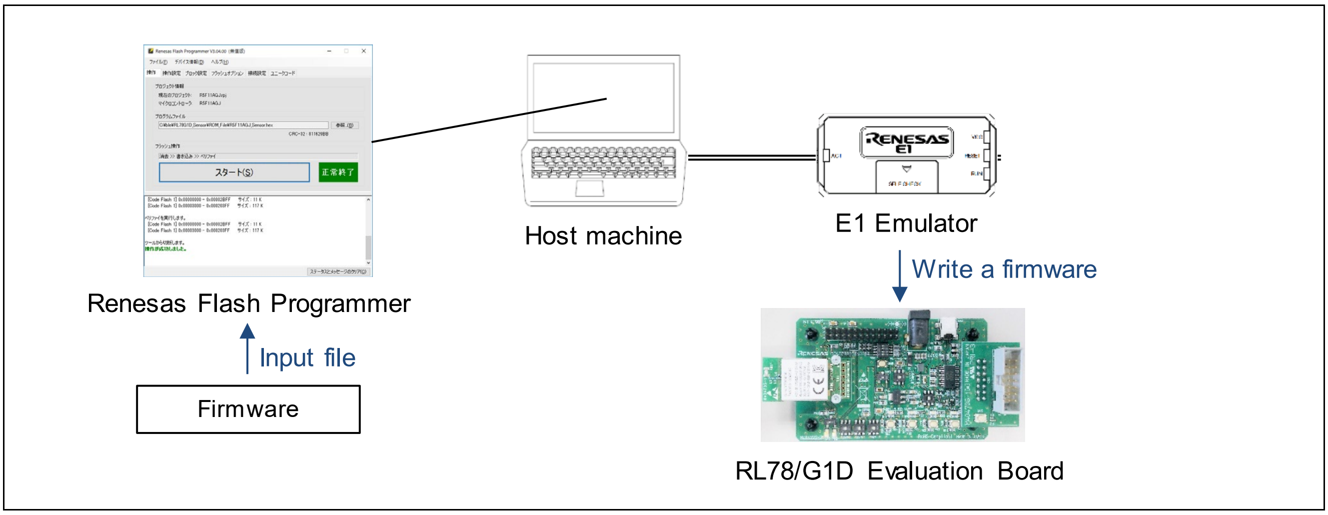


Figure 3-2 Overview of writing a firmware to RL78/G1D

Regarding the details of E1 Emulator, refer to E1 Emulator User's Manual (R20UT0398) and E1 Emulator Additional Document for User's Manual (Notes on Connection of RL78) (R20UT1994).

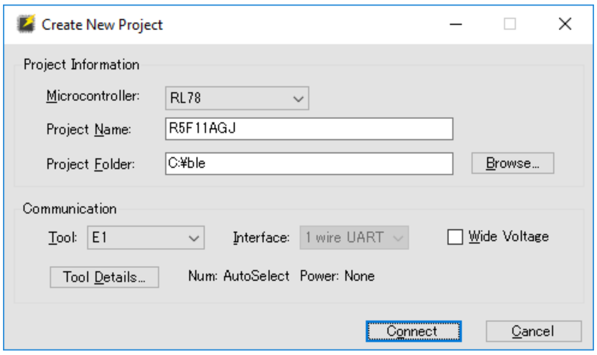
How to write a firmware to RL78/G1D evaluation board is shown below.

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.

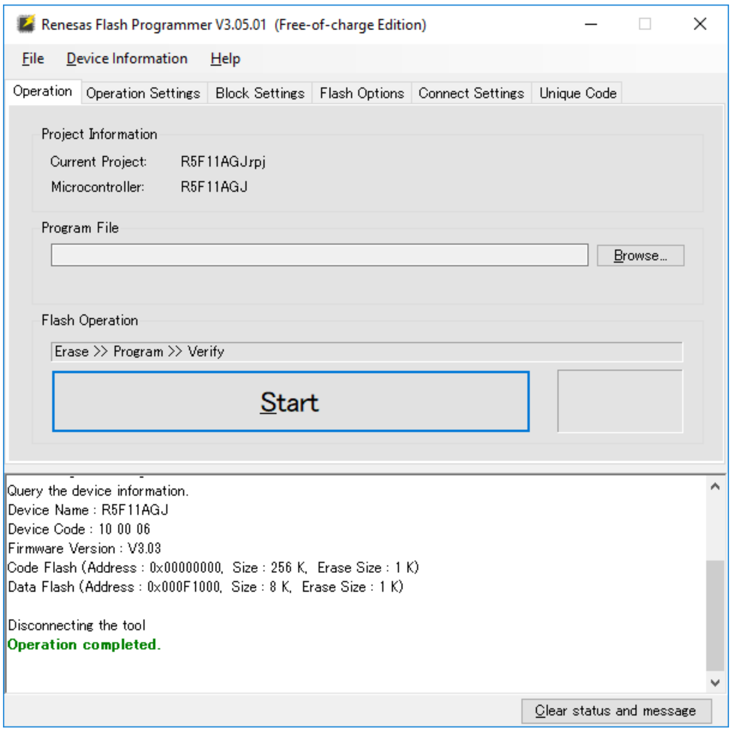
   Once you created a project, you can skip to execute these steps.

   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.



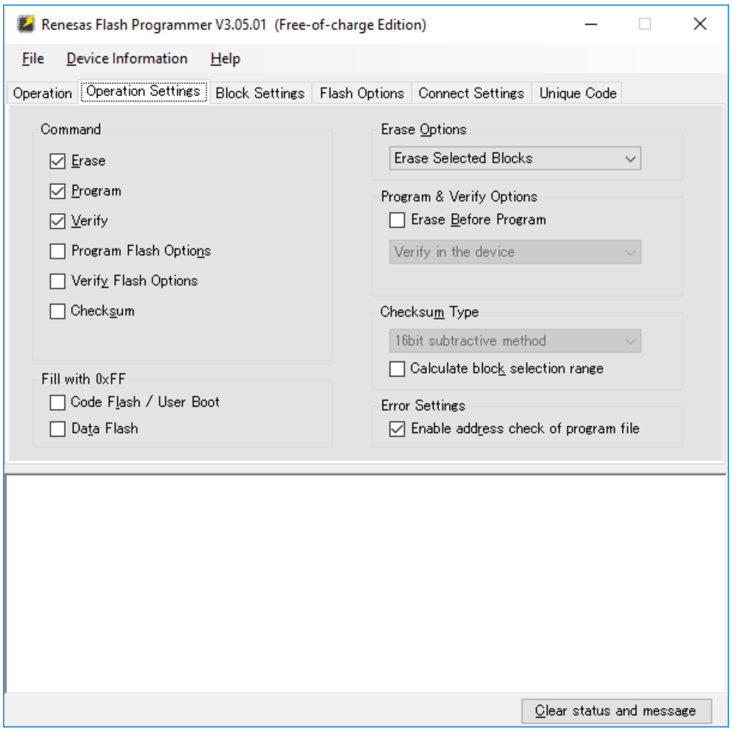
3-3. Confirm [Operation completed] message in Log output panel.



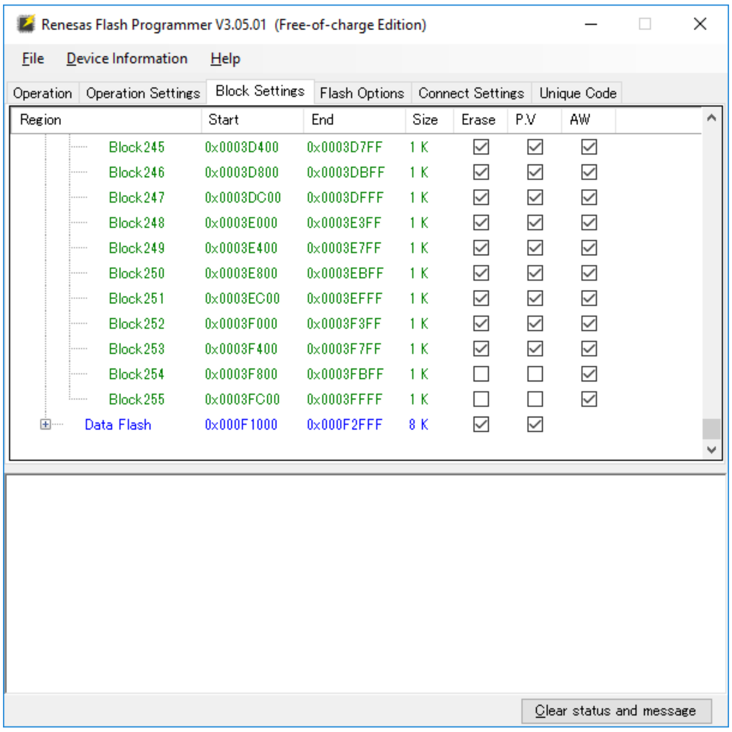
4. Prevent erasing Block 254, 255 in Code Flash memory according to the following steps.

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].



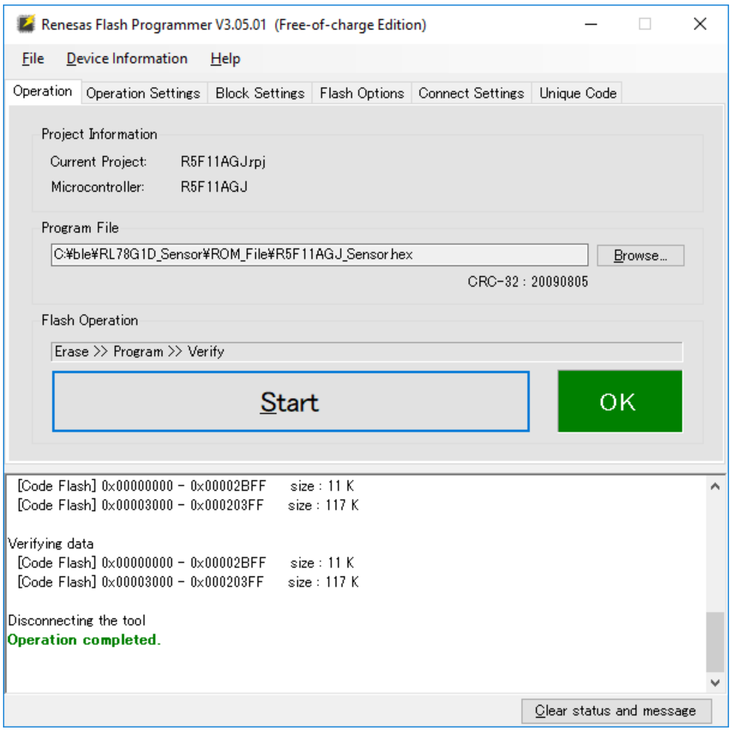
4-2. Select [Block Setting] tab and uncheck each [Erase], [P.V] of Block254, 255.



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.

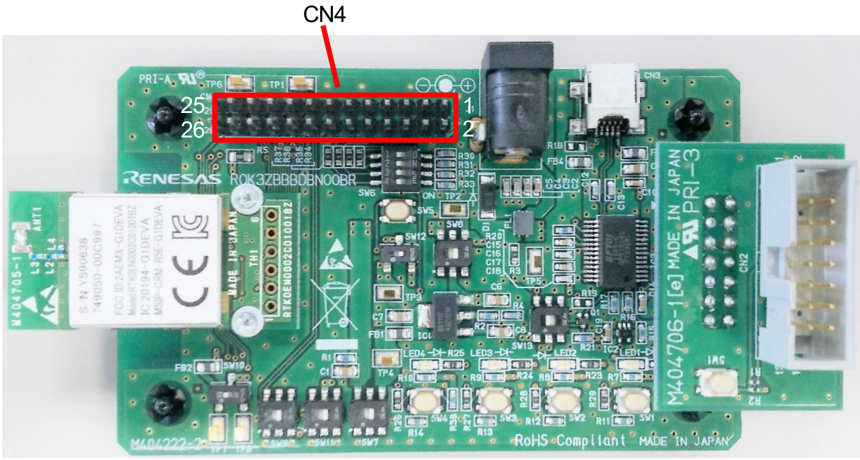


Figure 3-3 External Extension Interface of RL78/G1D Evaluation Board

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 I^2C 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

Pin	port of RL78/G1D	I/O of Board	RL78/G1D Functionality used by the program	
1	P30/INTP3	FT232RL	P30/INTP3	Digital Input, Interrupt Input
2	VCC	-	-	-
3	P61/SDAA0	-	SDAA0	Serial Data Bus for I^2C master
4	GND	-	-	-
5	P23/ANI3	SW3 Note2	P23	Digital Input
6	P10/SCK00/SCL00	SW6-1	P10	Digital Input
7	P147/ANI18	LED2	P147	Digital Output
8	GPIO1/TXSELL_RF	SW6-2 Note1	-	-
9	P03/ANI16/RxD1	LED3	P03	Digital Output
10	GPIO0/TXSELH_RF	SW6-3 Note1	-	-
11	P60/SCLA0	LED4	SCLA0	Serial Clock for I^2C master
12	P02/ANI17/TxD1	SW6-4	P02	Digital Input
13	P22/ANI2	SW4	P22	Digital Input
14	P12/SO00/TxD0/TOOLTxD	-	TxD0	UART for debug
15	P120/ANI19	LED1	ANI19	Analog Input for A/D conversion
16	P11/SI00/RxD0/TOOLRxD/SDA00	FT232RL	-	-
17	VCC	-	-	-
18	-	SW1 Note1	-	-
19	GND	-	-	-
20	P16/TI01/TO01/INTP5	SW2	P16/INTP5	Digital Input, Interrupt Input
21	P40/TOOL0	-	-	-
22	RESET	-	-	-
23	-	-	-	-
24	5V	-	-	-
25	GND	-	-	-
26	GND	-	-	-

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.

- I^2C slave sensor device [Note]

  RGB Light sensor - Renesas ISL29125
  https://www.renesas.com/products/sensor-products/light-proximity-sensors/ambient-light-sensors/ambient-light-digital-sensors/isl29125-digital-red-green-and-blue-color-light-sensor-ir-blocking-filter

  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 degign 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.

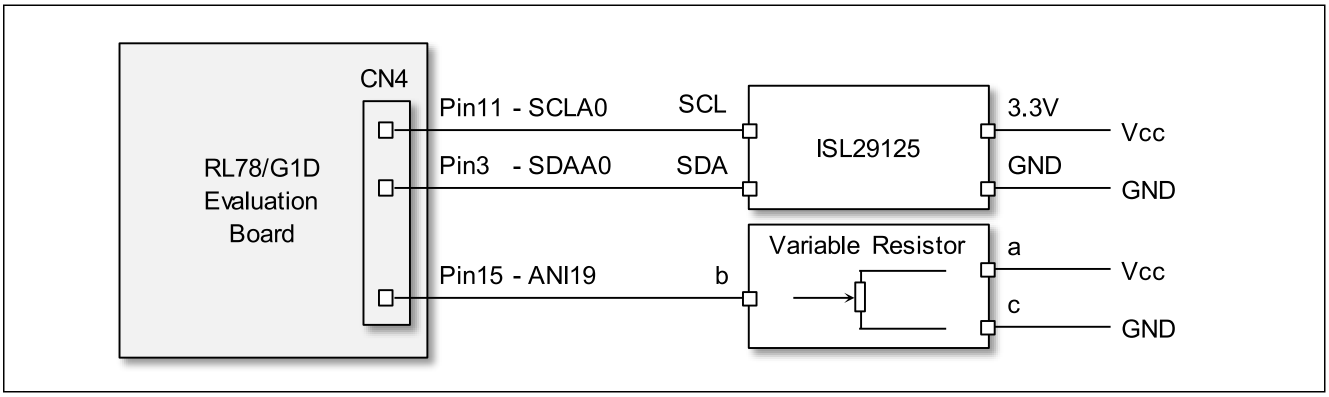


Figure 3-4 Connecting Sensors with RL78/G1D Evaluation Board

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*.

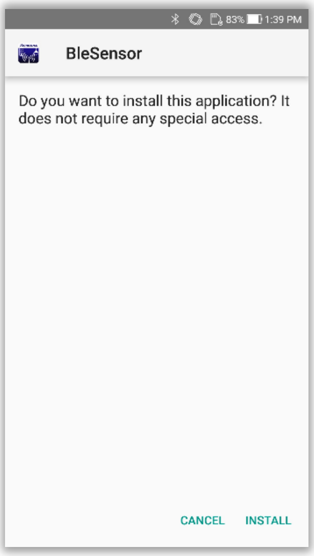


Figure 3-5 Installing Android Application

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".

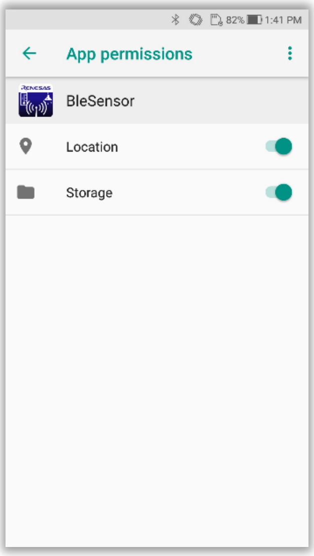


Figure 3-6 Permission Settings

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 deices and their RSSI: Received Signal Strength Indicator are displayed.

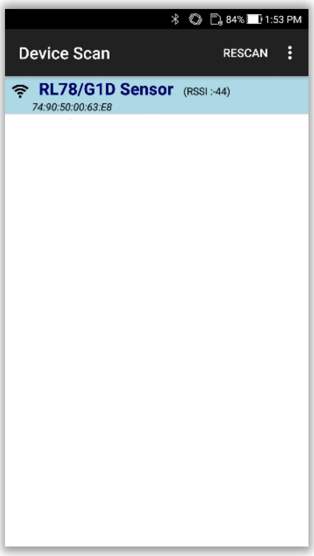


Figure 3-7 Device Search Display

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.

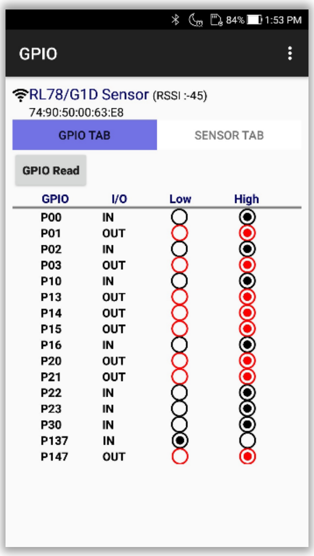
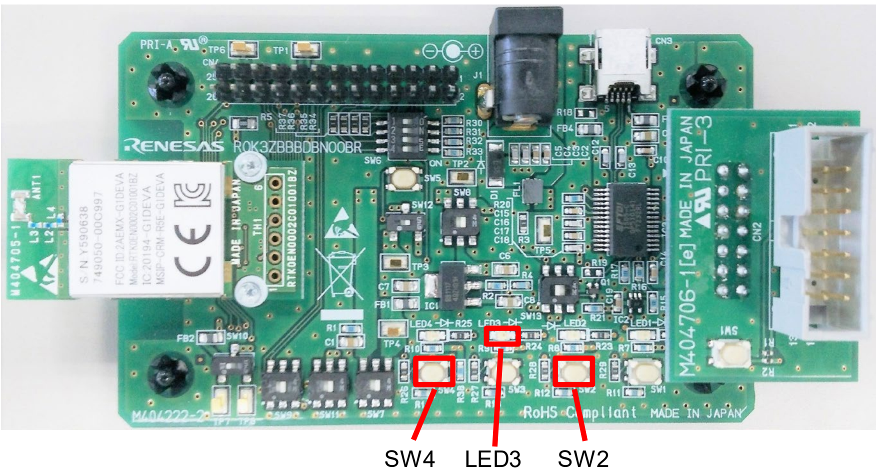


Figure 3-8 GPIO Control Display

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.

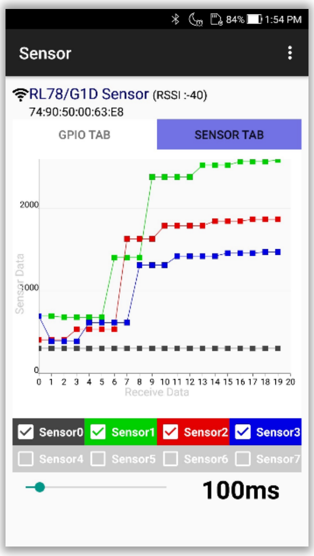


Figure 3-9 Sensor Measurement Display

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 *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, Y,M,D,H,M, and S are the date and time of stablishing a connection.

   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.

   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 *BleSensor*.

```
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
```

date and time
sensor4 to sensor7: not connected, always 0
sensor1～sensor3: RGB light sensor measurement result
sensor0: A/D conversion result

Figure 3-10 Example Log of Sensor Measurement Data

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          document for Android application BleSensor
└─Android_BleSensor_V1_0_5.zip          project for Android application BleSensor
RL78G1D_Sensor
├─Android_File
│   BleSensor.apk                       installation package of Android application BleSensor
├─ROM_File
│   R5F11AGJ_Sensor.hex                 firmware for evaluation
│   R5F11AGJ_Sensor(console_lvl4).hex   firmware for evaluation, UART for debug is enabled
└─Project_Source
   ├─bleip
   │  └─src
   │     ├─common
   │     │   co_bt.h
   │     └─rwble
   │         rwble.h
   │         rwble_config.h
   ├─rBLE
   │  └─src
   │     ├─include
   │     │   rble.h                     ] BLE Protocol Stack
   │     │   rble_api.h                 ] Its library files are included.
   │     │   rble_rwke.h                  When you develop, get the latest libraries.
   │     ├─sample_app
   │     │  │ console.c
   │     │  │ console.h
   │     │  │ rble_sample_app_peripheral.c ] BLE Application
   │     │  │ rble_sample_app_peripheral.h ]
   │     │  │ rble_sample_app_sensor.c     ] Sensor Application
   │     │  │ rble_sample_app_sensor.h     ] If you use other I2C device, modify this.
   │     │  └─seclib
   │     │      secdb.c                 ] Security Library
   │     │      secdb.h                 ]
   │     │      seclib.c                ]
   │     │      seclib.h                ]
   │     └─sample_profile
   │        └─sen
   │           sens.c                   ] Sensor Profile
   │           sens.h                   ]
   └─renesas
      ├─lib
      │   BLE_CONTROLLER_LIB_CCRL.lib   ] BLE Protocol Stack
```

```
|     BLE_HOST_lib_CCRL.lib              Its library files are included.
|     BLE_rBLE_lib_CCRL.lib              When you develop, get the latest libraries.
├─src
|  |  types.h
|  ├─arch
|  |  └─rl78
|  |      |  arch.h
|  |      |  arch_main.c
|  |      |  config.h
|  |      |  db_handle.h
|  |      |  hw_config.h
|  |      |  ke_conf.c
|  |      |  main.c
|  |      |  prf_config.c
|  |      |  prf_config.h
|  |      |  prf_config_host.c
|  |      |  prf_sel.h
|  |      |  rble_core_config.c
|  |      |  rble_core_config.h
|  |      |  rwble_mem.c
|  |      |  rwble_mem.h
|  |      |  rwke_api.h
|  |      └─ll
|  |          ll.h
|  ├─cg_src
|  |    r_cg_adc.c                       Peripheral Driver
|  |    r_cg_adc.h                       These are generated by Code Generator Plug-in
|  |    r_cg_adc_user.c
|  |    r_cg_iica.c
|  |    r_cg_iica.h
|  |    r_cg_iica_user.c
|  |    r_cg_intp.c
|  |    r_cg_intp.h
|  |    r_cg_macrodriver.h
|  |    r_cg_port.c
|  |    r_cg_port.h
|  |    r_cg_sau.c
|  |    r_cg_sau.h
|  |    r_cg_sau_user.c
|  |    r_cg_userdefine.h
|  ├─compiler
|  |  |  compiler.h
|  |  |  iodefine.h
|  |  └─ccrl
|  |      cstart.asm
|  ├─driver
|  |  ├─dataflash
|  |  |  |  dataflash.c
|  |  |  |  dataflash.h
|  |  |  |  eel_descriptor_t02.c
|  |  |  |  eel_descriptor_t02.h
|  |  |  |  fdl_descriptor_t02.c
|  |  |  |  fdl_descriptor_t02.h
|  |  |  └─cc_rl
|  |  |      eel.h                       Data Flash Library
```

```
│   │   │        eel.lib                     Its library files are included.
│   │   │        eel_types.h                 When you develop, get the latest libraries.
│   │   │        fdl.h
│   │   │        fdl.lib
│   │   │        fdl_types.h
│   │   ├─peak
│   │   │    peak.h
│   │   │    peak_isr.c
│   │   ├─pktmon
│   │   │    pktmon.h
│   │   ├─plf
│   │   │    plf.c
│   │   │    plf.h
│   │   ├─port
│   │   │    port.h
│   │   ├─rf
│   │   │    rf.h
│   │   └─serial
│   │        serial.h
│   └─sensor
│        ISL29125.c                          Device Driver
│        ISL29125.h                          If you use other I2C device, add driver for it.
└─tools
   └─project
      └─CS_CCRL
         └─BLE_Peripheral
            │  BLE_Peripheral.mtpj
            └─R5F11AGJ_Sensor
                 R5F11AGJ_Sensor.mtsp
```

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
   https://www.renesas.com/document/lbr/bluetooth-low-energy-protocol-stack-ver121

   Data Flash Library:

   EEPROM Emulation Library Pack02 Package Ver.2.00(for CA78K0R/CC-RL Compiler) for RL78 Family
   https://www.renesas.com/document/upr/eeprom-emulation-library-pack02-package-ver200for-ca78k0rcc-rl-compiler-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/BLE_CONTROLLER_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.

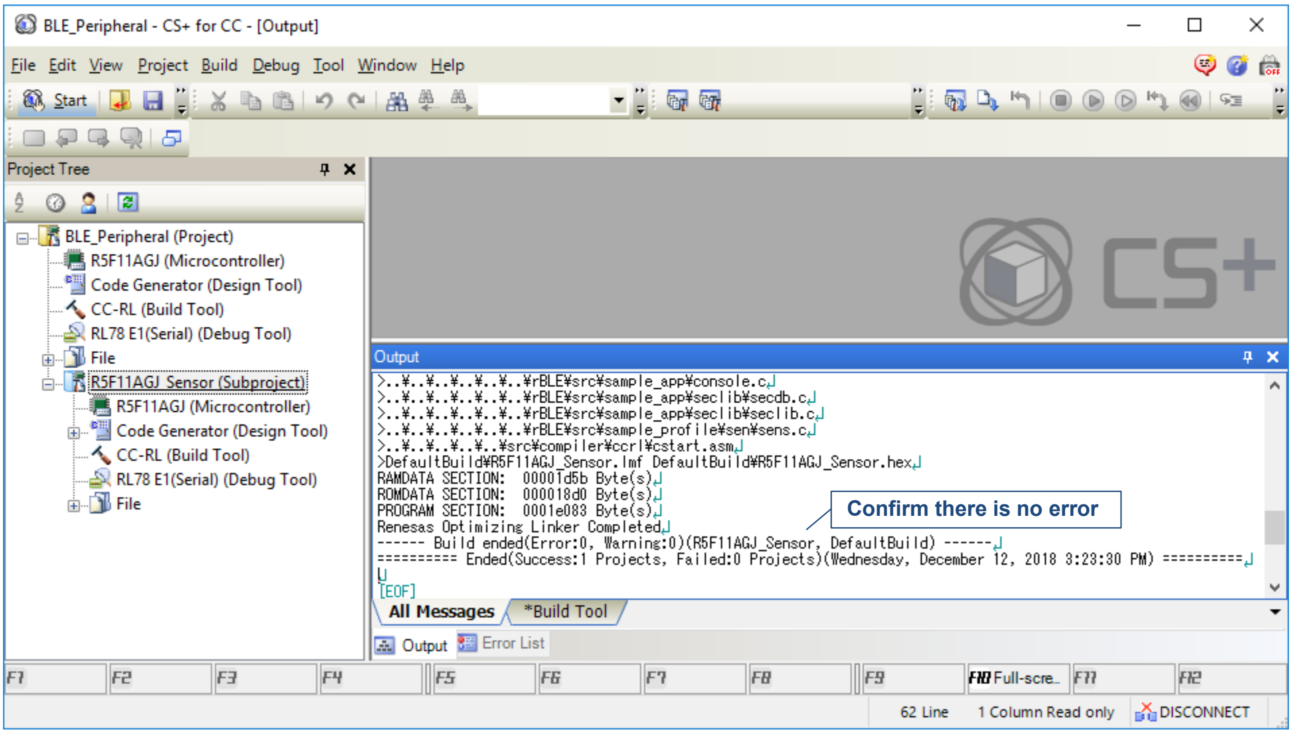
```
Project_Source
├─rBLE
│ └─src
│   └─include
│           rble.h                      Protocol Stack rBLE definitions - header file
│           rble_api.h                  Protocol Stack rBLE API - header file
└─renesas
  ├─lib
  │         BLE_CONTROLLER_LIB_CCRL.lib Protocol Stack Controller Layer - library file
  │         BLE_HOST_lib_CCRL.lib       Protocol Stack Host Layer - library file
  │         BLE_rBLE_lib_CCRL.lib       Protocol Stack rBLE Layer - library file
  └─src
    └─driver
      └─dataflash
        └─cc_rl
            eel.h                       Data Flash Library EEPROM Emulation - header file
            eel.lib                     Data Flash Library EEPROM Emulation - library file
            eel_types.h                 Data Flash Library EEPROM Emulation type definition -
            fdl.h                       Data Flash Library - header file
            fdl.lib                     Data Flash Library - library file
            fdl_types.h                 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_CCRL/BLE_Peripheral
2. To build a firmware, select [Build]→[Rebuild Project].
3. Confirm that no error occurs, it succeeds in building a firmware.



4. Confirm that the firmware named *R5F11AGJ_Sensor.hex* is generated in the following folder.
   - Project_Source/renesas/tools/project/CS_CCRL/BLE_Peripheral/R5F11AGJ_Sensor/DefaultBuild

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.

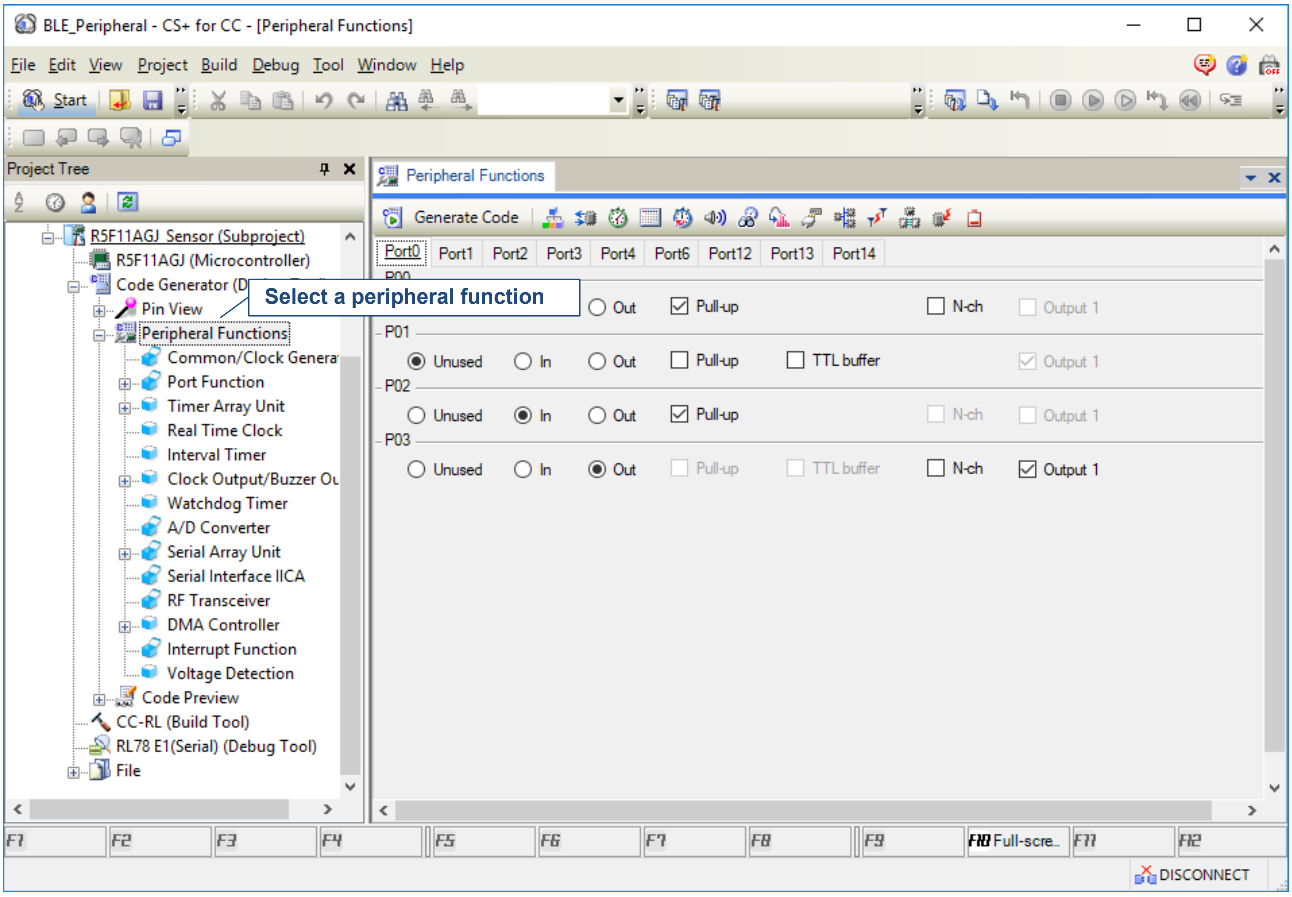


Figure 4-1 Code Generator Plug-In

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/renesas/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/renesas/src/cg_src/r_cg_adc.c, r_cg_adc_user.c

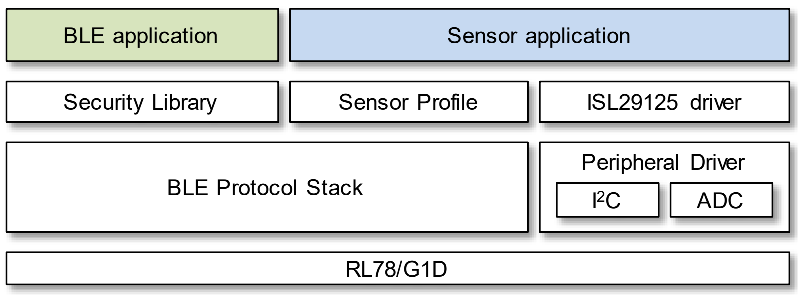


Figure 5-1 Module Related to Control Sensor

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.

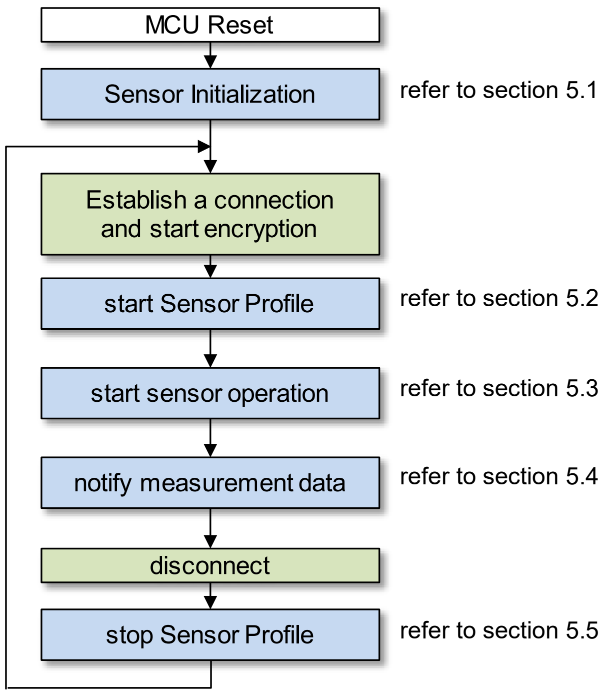


Figure 5-2 Flow Chart of Sensor Application

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 I^2C.

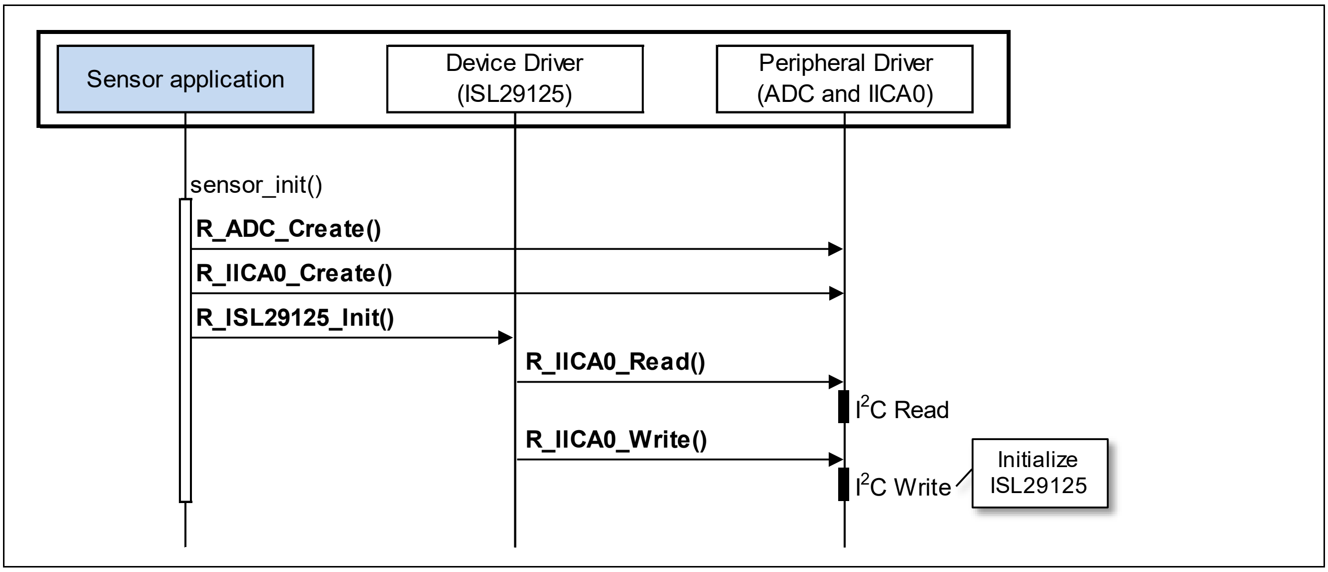


Figure 5-3 Sensor Initialization

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

If you use another I^2C 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.

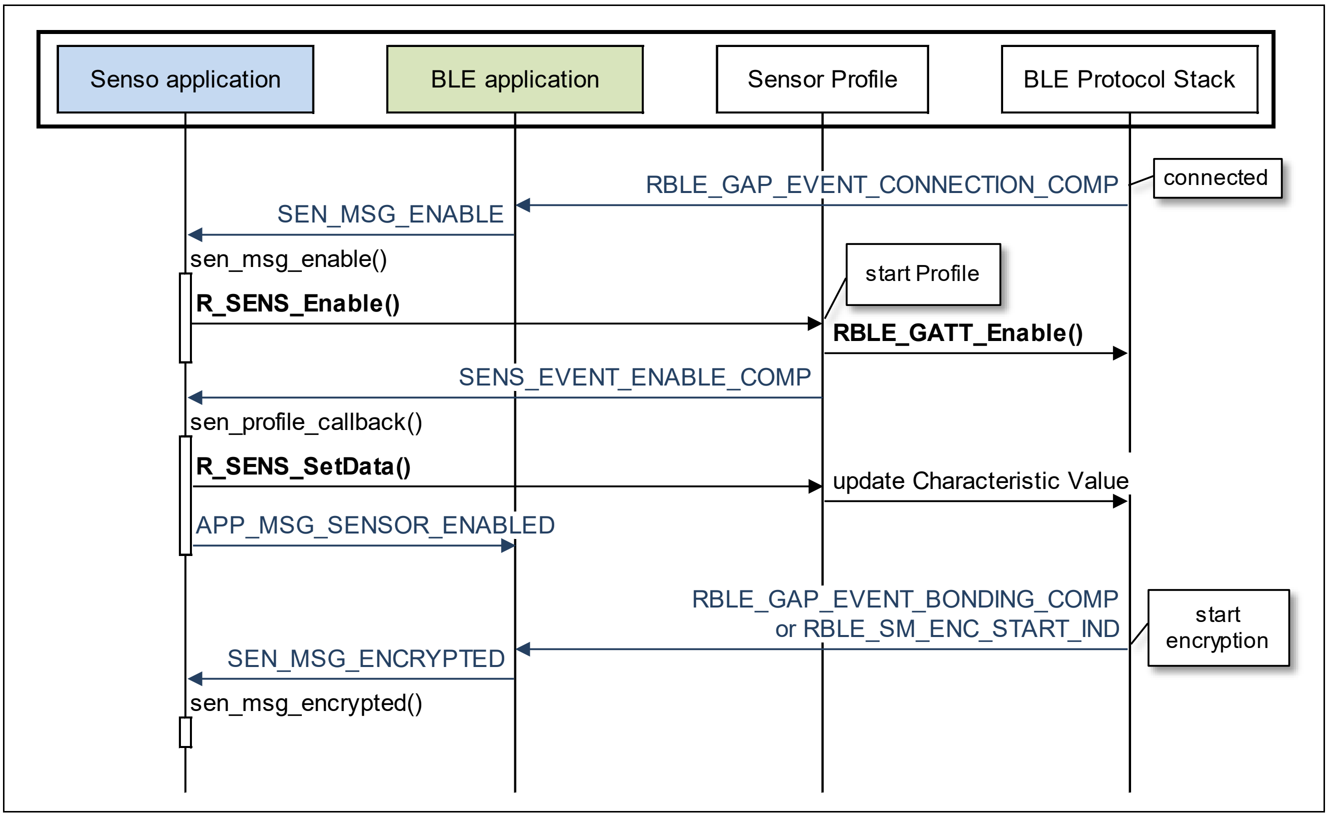


Figure 5-4 Sensor Profile Start

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.

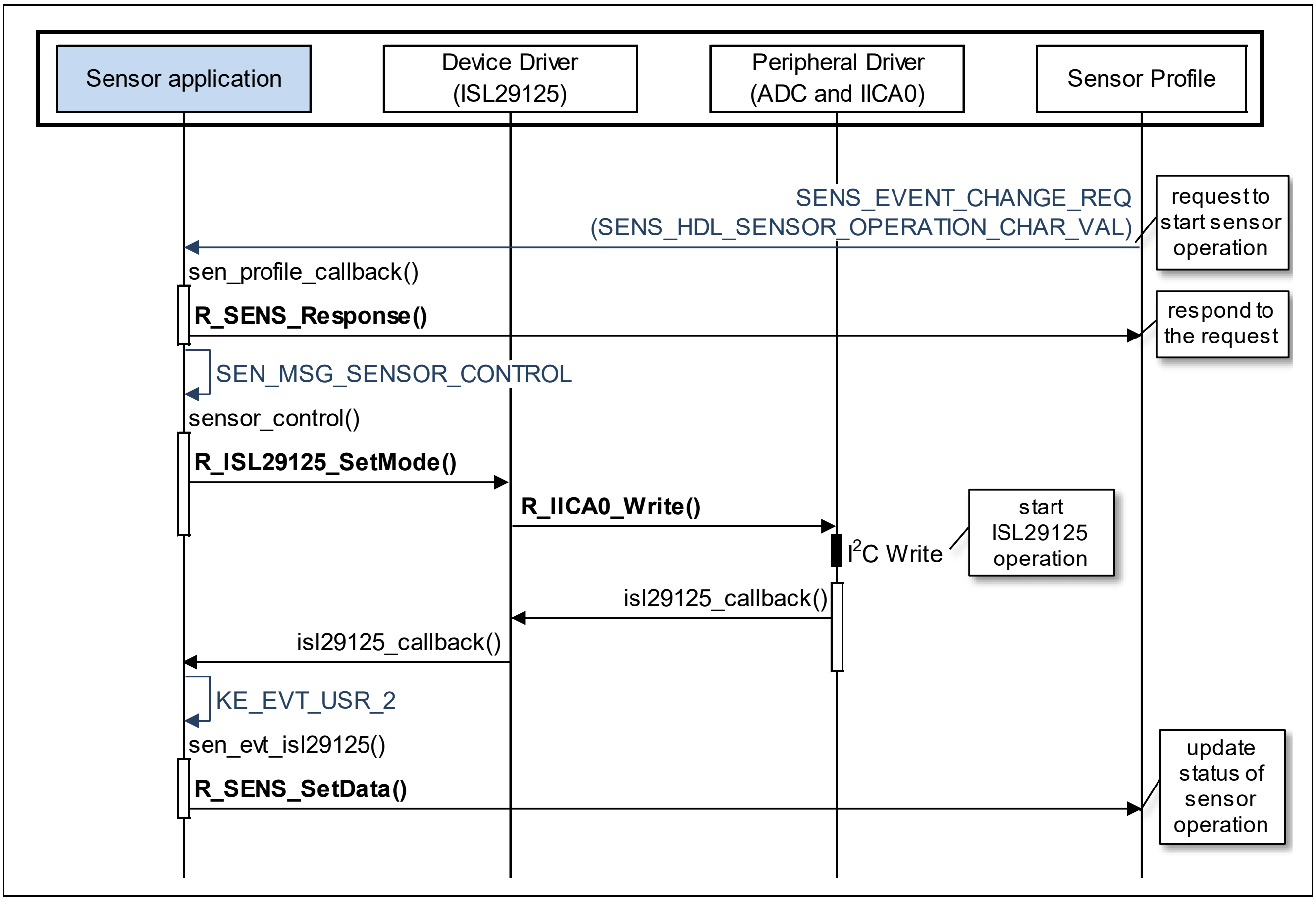


Figure 5-5 Sensor Operation Start

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

If you use another I^2C 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.

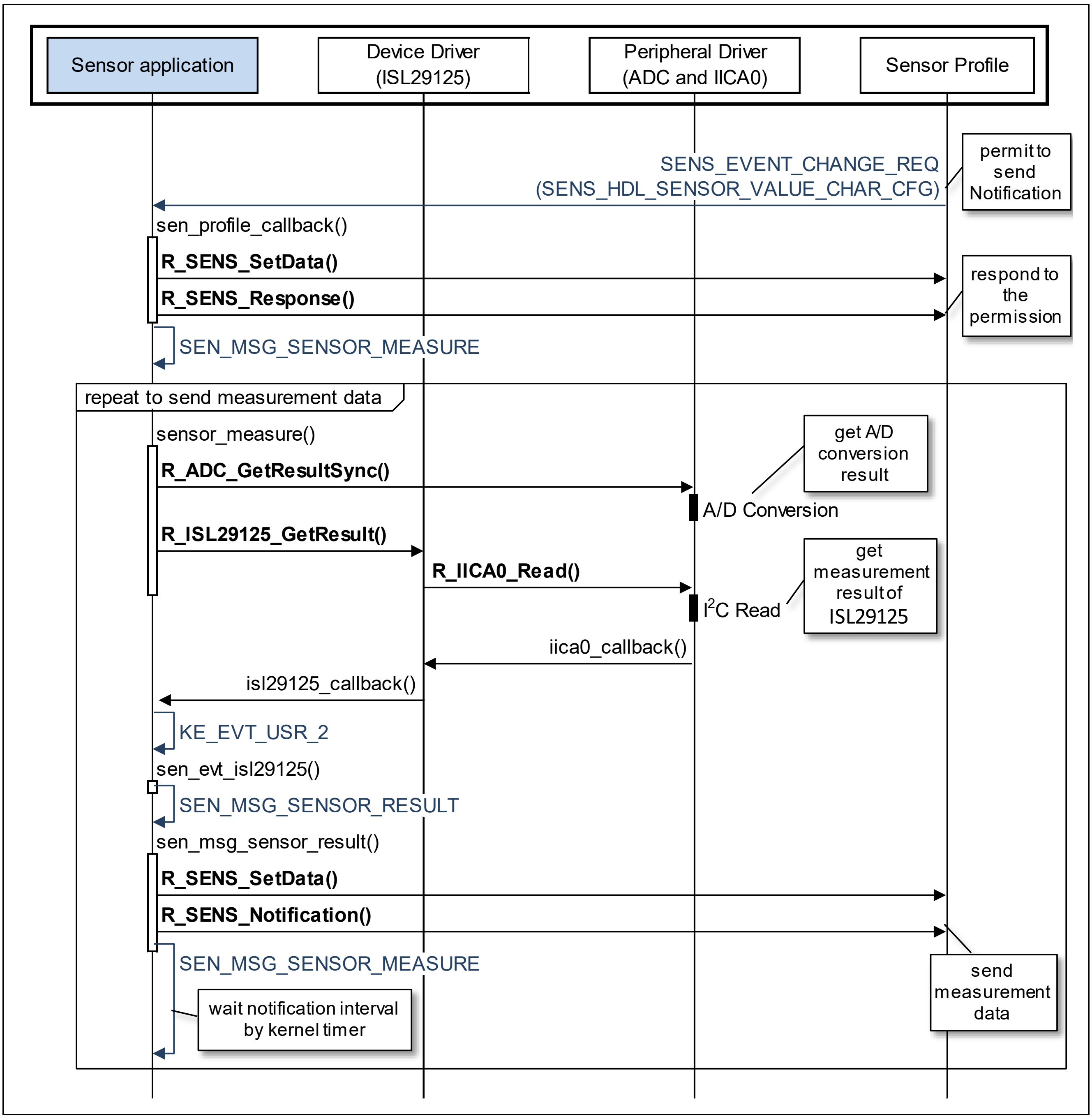


Figure 5-6 Sensor Measurement Data Notification

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.

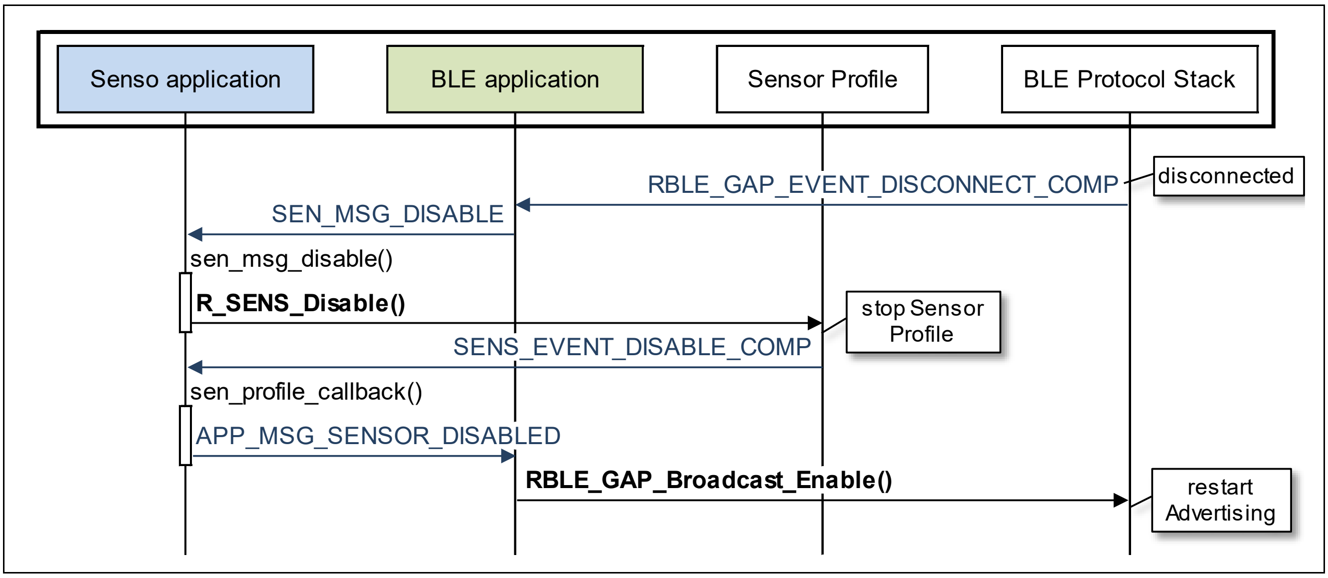


Figure 5-7 Sensor Profile Stop

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

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

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

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

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

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

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

<table>
<tr><td colspan="5">uint8_t R_ISL29125_Init(r_isl29125_calback_t callback);</td></tr>
<tr><td colspan="5">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.</td></tr>
<tr><td colspan="5">Parameters:</td></tr>
<tr><td rowspan="5"></td><td rowspan="5">callback</td><td colspan="3">Callback function to notify that asynchronous device control is finished.
void (*r_isl29125_calback_t)
(r_isl29125_opcode_t opcode, uint8_t status, void* data);</td></tr>
<tr><td rowspan="4"></td><td>opcode</td><td>Operation Code to identify each device control operation</td></tr>
<tr><td rowspan="2">status</td><td>status of device control operation</td></tr>
<tr><td>0 Success
others Failed</td></tr>
<tr><td>data</td><td>returned data from device control operation</td></tr>
<tr><td colspan="5">Return:</td></tr>
<tr><td></td><td>0
others</td><td colspan="3">Success
Device not present, Device or I²C error, or other error</td></tr>
</table>

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

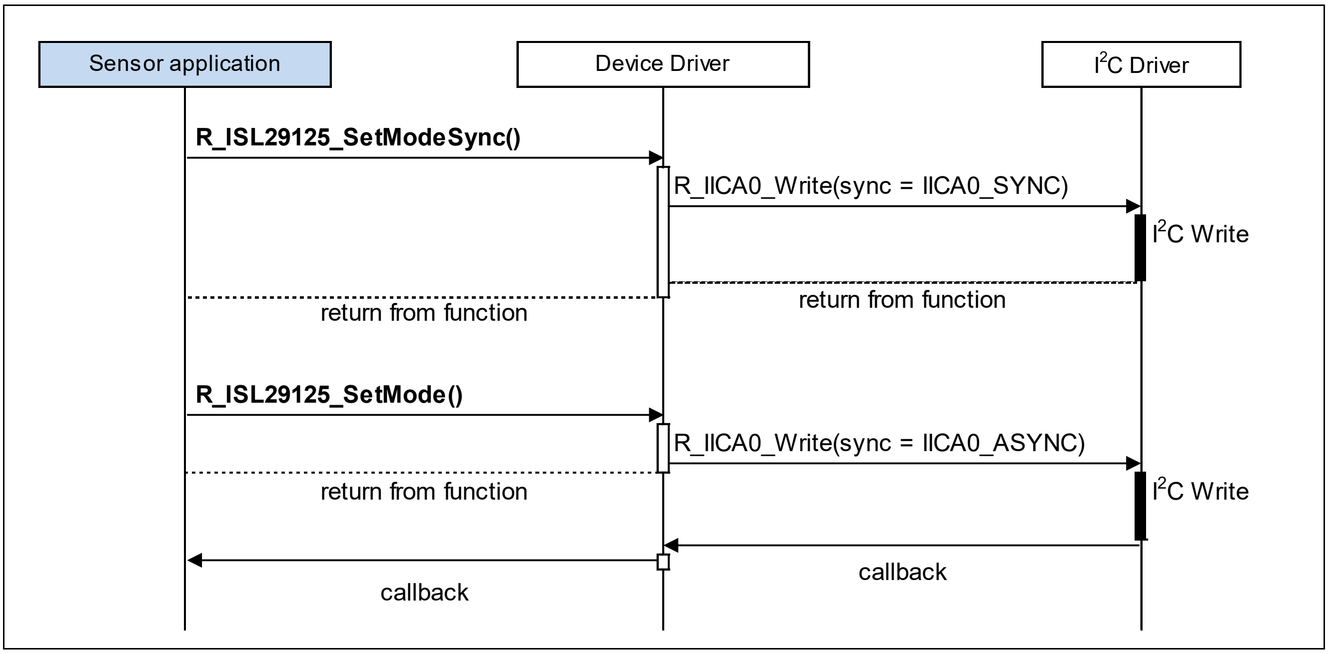


Figure 6-1 Sequence of R_ISL29125_SetModeSync() and R_ISL29125_SetMode()

6.2.4 R_ISL29125_GetResultSync

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

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

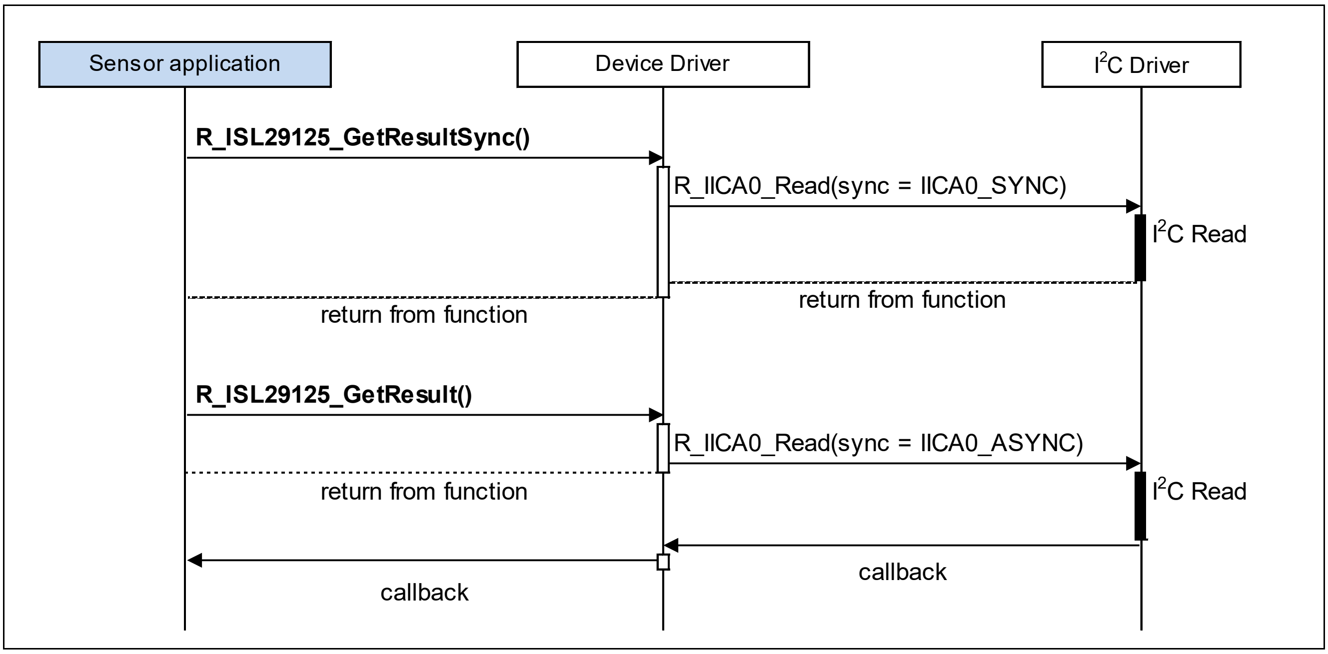


Figure 6-2 Sequence of R_ISL29125_GetResultSync() and R_ISL29125_GetResult()

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 a 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

void **R_IICA0_Create**(void);	
This function initializes Serial Interface IICA of RL78/G1D.	
Parameters:	
	None
Return:	
	None

6.3.2 R_IICA0_RegisterCallback

void **R_IICA0_RegisterCallback**(iica0_user_calback_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 void (*iica0_user_calback_t)(iica0_rw_calltype_t type, uint8_t flag);			
		type	IICA0_SENDEND IICA0_RECEIVEEND IICA0_ERROR	I²C write completion I²C read completion I²C error
		flag	refer to r_cg_macrodriver.h MD_OK other than MD_OK	 Success 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 I^{2}C slave device.
The address of I^{2}C 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 I^{2}C communication, refer to Figure 6-3.

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

Parameters:

adr	Device Address	set 7bit device address
buf	Data Buffer	set register address to *buf*[0] set data to *buf*[1] and the following
len	Access Length (byte)	set the sum of register address (1byte) and data length (*len* >= 2)
sync	Synchronous Setting	
	IICA0_SYNC	This function returns after completion of I^{2}C access.
	IICA0_ASYNC	This function returns without waiting completion of I^{2}C access. Completion of I^{2}C access is notified by callback function.

Return:

refer to r_cg_macrodriver.h

MD_OK	Success
other than MD_OK	Error

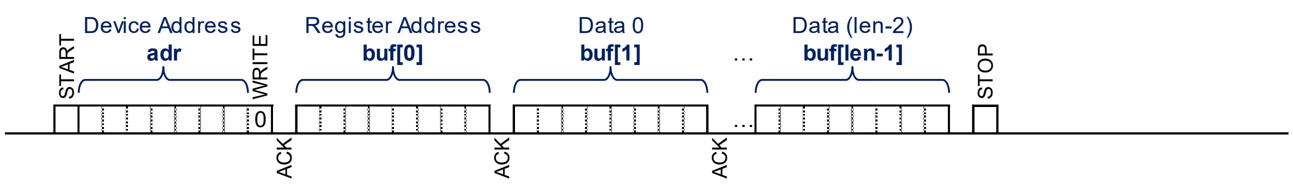


Figure 6-3 I2C Accessing by R_IICA0_Write

6.3.4 R_IICA0_Read

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 I^2C slave device. The address of I^2C 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 I^2C communication, refer to Figure 6-4. This function should be executed when interrupt enables. And this function cannot be executed by interrupt handler.		
Parameters:		
adr	Device Address	set 7bit device address
buf	Data Buffer	set register address to *buf*[0]
len	Access Length (byte)	set the sum of register address (1byte) and data length (*len* >= 2)
sync	Synchronous Setting IICA0_SYNC IICA0_ASYNC	 This function returns after completion of I^2C access. This function returns without waiting completion of I^2C access. Completion of I^2C access is notified by callback function.
Return:		
refer to r_cg_macrodriver.h		
MD_OK	Success	
other than MD_OK	Error	

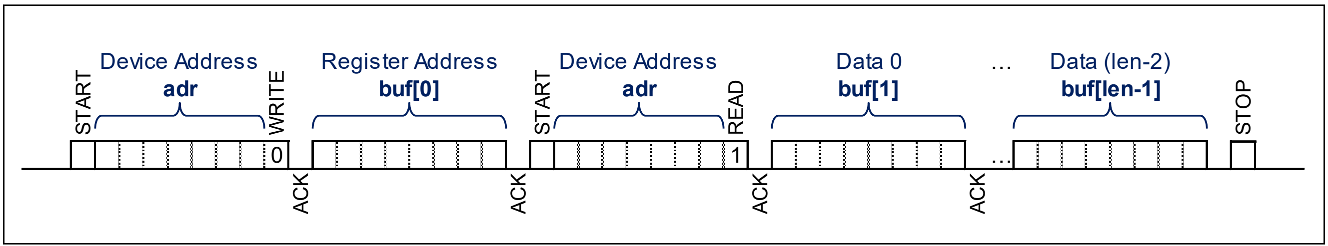


Figure 6-4 I2C Accessing by R_IICA0_Read

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

void **R_ADC_Create**(void);	
This function initializes the A/D converter of RL78/G1D.	
Parameters:	
	None
Return:	
	None

6.4.2 R_ADC_GetChannel

uint8_t **R_ADC_GetChannel**(void);	
This function gets a selected analog input channel of A/D converter.	
Parameters:	
	None
Return:	
	Analog Input Channel 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

uint8_t **R_ADC_GetResultSync**(uint16_t* result);		
This function A/D conversion and return the result by argument.		
Parameters:		
	result	A/D conversion result
Return:		
	0	Success
	others	Error

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 7-1 Functions of UART for Debug

Function	Example Use-case
PrintError()	for reporting implementation problem
PrintWarning()	for warning unexpected operation
PrintInfo()	for checking parameters used by application
PrintLog()	for checking sequence of application

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)
```

Change either 1 to 4

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

Table 7-2 CONSOLE_LVL

CONSOLE_LVL	Usable function of UART for debug
0	UART for debug is disabled
1	PrintError() only
2	PrintError() and PrintWarning
3	PrintError(), PrintWarning() and PrintInfo()
4	PrintError(), PrintWarning(), PrintInfo() andPrintLog()

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 7-3 Serial Communication Setting for Terminal Software

Item		Setting
Serial Port	Port	USB Serial Port Note that COM number is different from each evaluation board
	Baud rate	1,000,000bps
	Data Bit Length	8bit
	Parity	None
	Stop Bit Length	1bit
	Flow Control	None
New Line	Receive	LF
Terminal Size	Horizontal	over than 128 characters

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.

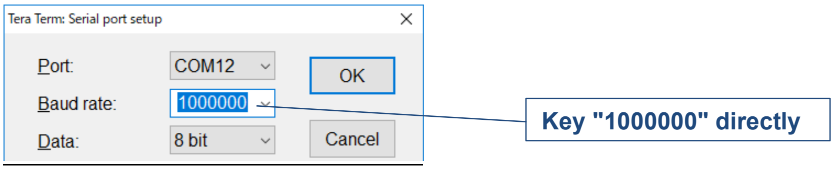


Figure 7-1 shows an example message that is output by the sample program built with CONSOLE_LVL=4.

You can customize message content as necessary.

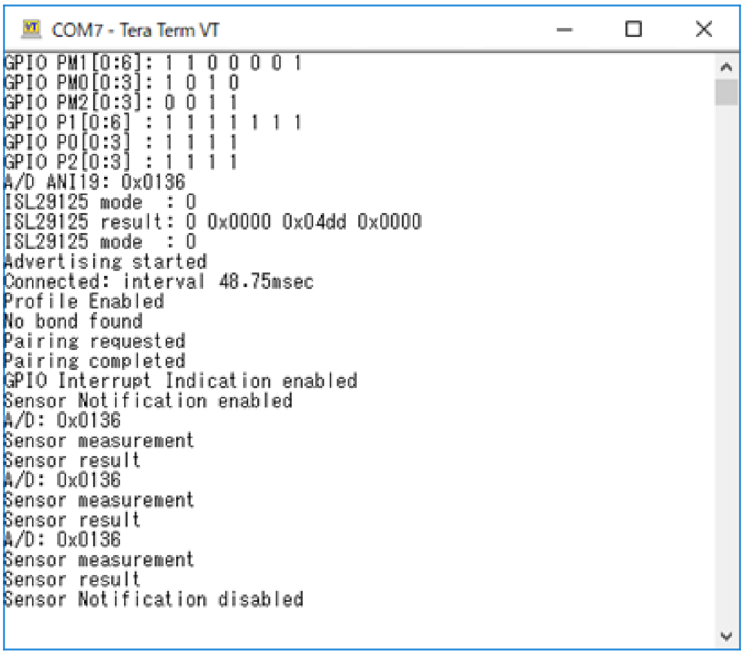


Figure 7-1 Example Message of UART for Debug

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/

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.

   ¾ 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.

   ¾ 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.

   ¾ 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.

   ¾ 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.

   ¾ 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.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

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

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

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

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700

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

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

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

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

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

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

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

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.1