Bluetooth® Low Energy プロトコルスタック センサアプリケーション

R01AN4159JJ0103
Rev.1.03
2018.12.21

要旨

本アプリケーションノートは、Bluetooth® Low Energy (以降、BLE と表記)に対応した RL78/G1D 上で動作し、センサの測定データをリモートデバイスに通知するサンプルプログラムについて説明します。

サンプルプログラムには、RL78/G1D で動作するセンサアプリケーションのソースコードとファームウェアに加え、センサの測定データを Android デバイスで確認するための Android アプリが含まれます。

サンプルプログラムのセンサアプリケーションは、GATT ベースプロファイルのサーバロールとして動作します。また Android アプリは、GATT ベースプロファイルのクライアントロールとして動作します。

サンプルプログラムの Android アプリを使用することで、センサアプリケーションの GATT データベースにアクセスし、RL78/G1D の GPIO 操作や、RL78/G1D に接続されたセンサの測定データをグラフで確認できます。

動作確認デバイス

RL78/G1D (R5F11AGJ)

関連資料

資料名	資料番号
RL78/G1D	
ユーザーズマニュアル ハードウェア編	R01UH0515
RL78/G1D 評価ボード	
ユーザーズマニュアル	R30UZ0048
E1 エミュレータ	
ユーザーズマニュアル	R20UT0398
ユーザーズマニュアル別冊 (RL78 接続時の注意事項)	R20UT1994
Renesas Flash Programmer V3.05 フラッシュ書き込みソフトウェア	
ユーザーズマニュアル	R20UT4307
CC-RL コンパイラ	
ユーザーズマニュアル	R20UT3123
Bluetooth Low Energy プロトコルスタック	
ユーザーズマニュアル	R01UW0095
API リファレンスマニュアル 基本編	R01UW0088
セキュリティライブラリ アプリケーションノート	R01AN3777

目次

1. 概要

図 1-1 にサンプルプログラムの概要を示します。

サンプルプログラムには RL78/G1D の GPIO 制御や A/D 変換、I²C 通信を実行する機能があります。また本アプリケーションノートで独自に定義した GATT ベースのセンサプロファイルが実装されています。RL78/G1D と Bluetooth Low Energy で接続したリモートデバイスは、このセンサプロファイルに従ってデータ通信することで、RL78/G1D の GPIO とセンサを制御できます。

サンプルプログラムの動作確認では、RL78/G1D 評価ボードを使用し、アナログ出力のセンサや I²C インタフェースを持つセンサを接続します。またリモートデバイスとして Android デバイスを使用し、同梱された Android アプリの BleSensor をインストールして実行します。

サンプルプログラムを起動すると自動で Advertising を実行します。Android デバイス上で BleSensor を操作することで、RL78/G1D との接続確立から GPIO の制御やセンサ測定値の確認まで行うことができます。

BleSensor の GPIO 制御画面では、出力ポートの出力信号の High/Low 変更や、入力ポートの入力信号の High/Low 確認ができます。

BleSensor のセンサ測定画面では、A/D 変換の結果や、I²C で接続したセンサの測定値をグラフで確認できます。

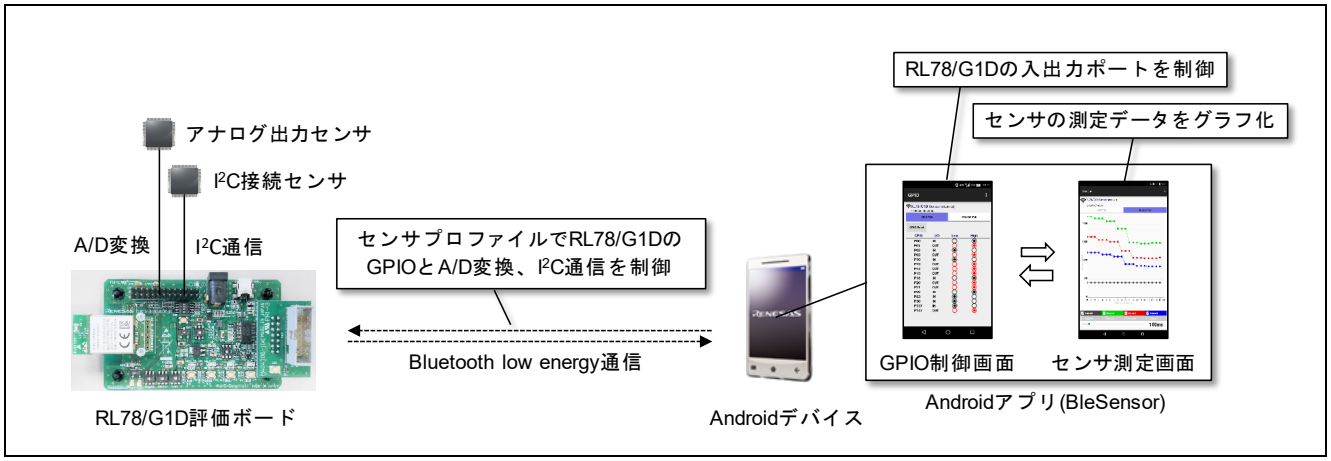


図 1-1　サンプルプログラムの概要

RL78/G1D の GPIO やセンサを制御するセンサプロファイルの仕様は、下記を参照してください。

- 2.3 節「センサプロファイル」

RL78/G1D へのファームウェア書き込みやセンサの接続、Android アプリのインストールといった、サンプルプログラムの動作確認の手順は、下記を参照してください。

- 3 章「操作方法」

サンプルプログラムには、A/D コンバータドライバが実装されています。測定結果をアナログ出力するセンサを RL78/G1D に接続することで、プログラムを変更することなく測定結果を確認できます。

A/D コンバータドライバの関数仕様は、下記を参照してください。

- 6.4 節「A/D コンバータドライバ」

サンプルプログラムには、I²C インタフェースを持つデバイスのレジスタを読み書きする I²C ドライバが実装されています。I²C インタフェースのセンサデバイスを RL78/G1D に接続する場合は、本ドライバを利用することで、センサデバイスの動作制御や測定結果の取得ができます。なお各センサデバイスのレジスタ仕様や制御方法はそれぞれ異なるため、各センサデバイスの仕様書を参照してください。

I²C ドライバの関数仕様は、下記を参照してください。

- 6.3 節「I²C ドライバ」

またサンプルプログラムには、I²C インタフェースの RGB ライトセンサである Renesas ISL29125 のデバイスドライバがあらかじめ実装されています。他のセンサデバイスを接続する場合は、本ドライバの実装内容を基に、使用するセンサデバイスのドライバを追加実装してください。

ISL29125 デバイスドライバの関数仕様は、下記を参照してください。

- 6.2 節「デバイスドライバ」

ISL29125 デバイスドライバの実行シーケンスは、下記を参照してください。

- 5 章「センサ制御」

2. 仕様

2.1 ソフトウェア構成

サンプルプログラムのソフトウェア構成を示します。

- BLE アプリケーション　: BLE 無線通信の管理
- センサアプリケーション　: GPIO・センサの管理
- セキュリティライブラリ　: BLE 無線通信のセキュリティ制御
- センサプロファイル　: BLE 無線通信の GATT 制御
- BLE プロトコルスタック　: BLE プロトコル機能の提供
- カーネル　: カーネル機能の提供
- データフラッシュライブラリ　: データフラッシュの制御
- デバイスドライバ　: I^2C スレーブデバイスの制御
- 周辺機能ドライバ　: RL78/G1D 周辺機能の制御

BLE プロトコルスタック、カーネル、データフラッシュライブラリはライブラリ形式で提供されます。

BLE アプリケーション、センサアプリケーションに加え、セキュリティライブラリ、センサプロファイル、デバイスドライバ、コード生成プラグインが自動生成した周辺機能ドライバはソースコード形式で提供され、必要に応じてカスタマイズが可能です。

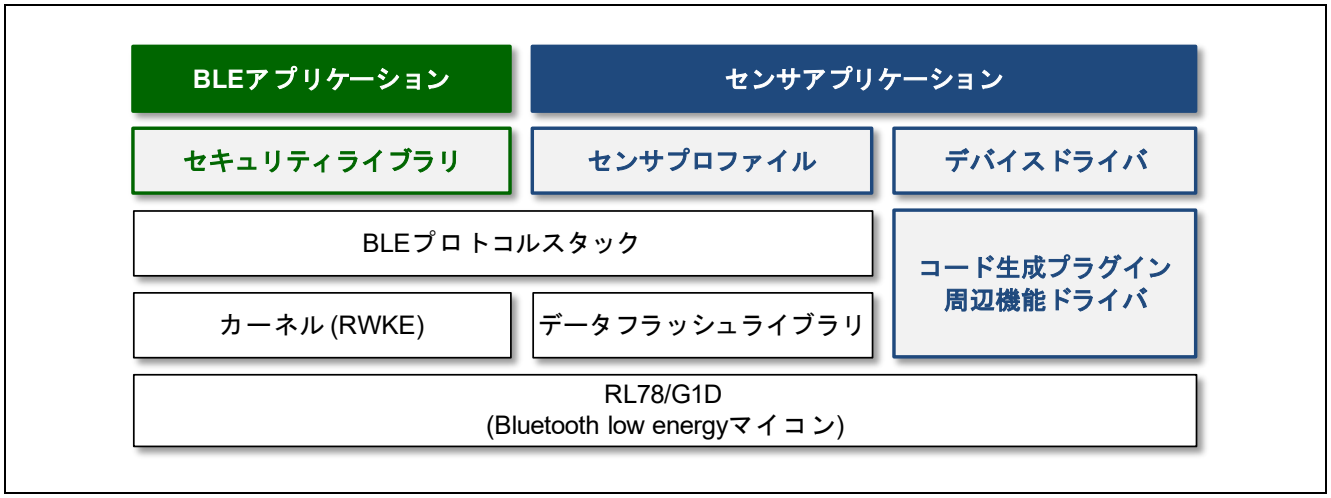


図 2-1　ソフトウェア構成

本アプリケーションノートのサンプルプログラムは、以下のサンプルプログラムを応用したものです。BLE 無線通信の制御シーケンスやセキュリティライブラリの仕様については、下記のサンプルプログラムを参照してください。

Bluetooth Low Energy プロトコルスタック Embedded 構成サンプルプログラム (R01AN3319)
https://www.renesas.com/document/scd/bluetooth-low-energy-protocol-stack-embedded-configuration-sample-program

サンプルプログラムのパッケージには、動作確認のためのライブラリが同梱されています。アプリケーション開発の際は 4.2 節「ライブラリの入手」を参照し、最新版のライブラリを入手してください。

2.2　デジタル・アナログインタフェース

図 2-2 にサンプルプログラムが使用する RL78/G1D のデジタル・アナログインタフェースを示します。

- I²C マスタ　：I²C スレーブデバイスの制御とデータ取得
- A/D 変換　：アナログ出力デバイスの信号レベル取得
- GPIO 出力　：簡易デジタル入力デバイスの制御
- GPIO 入力　：簡易デジタル出力デバイスの信号取得
- 外部入力割り込み　：簡易デジタル出力デバイスの信号エッジ検出
- デバッグ用 UART 送信　：ホストマシンへのデバッグ用メッセージを送信

なお後述するコード生成プラグインの設定を変更することで、使用するインタフェースは変更できます。

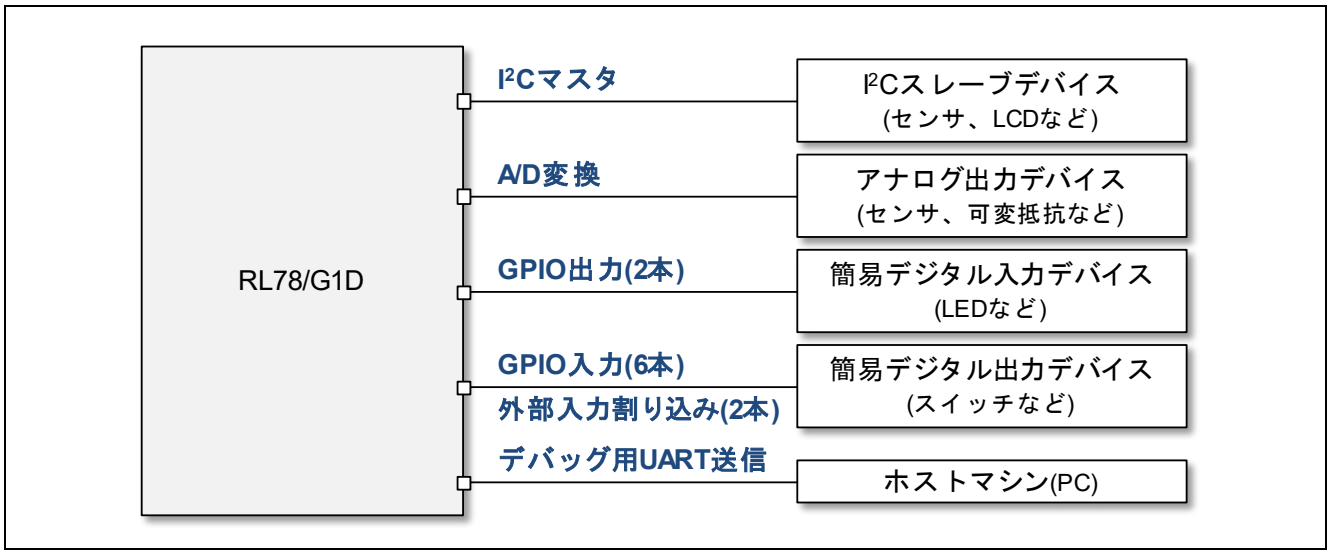


図 2-2　RL78/G1D のデジタル・アナログインタフェース

2.3　センサプロファイル

RL78/G1D の GPIO やセンサを制御するための独自の GATT ベースプロファイルを定義します。

サンプルプログラムに実装されたセンサプロファイルの仕様を以下に示します。

■ロールについて

- GPIO やセンサを持つデバイスを、センサプロファイルのサーバとする

  サーバは、センササービスを保持する

  本アプリケーションノートでは RL78/G1D がサーバとなる

- センサプロファイルサーバと接続して GPIO やセンサを制御するリモートデバイスを、センサプロファイルのクライアントとする

  クライアントは、サーバのセンササービスにアクセスする

  本アプリケーションノートでは Android デバイスがクライアントとなる

■サービスとキャラクタリスティックについて

- センササービスは、GPIO やセンサを制御するための複数のキャラクタリスティックで構成される
- クライアントは、Characteristic Value Read でキャラクタリスティック値を取得し、Characteristic Value Write でキャラクタリスティック値を更新する
- サーバは Notification と Indication によってキャラクタリスティック値をクライアントに通知する

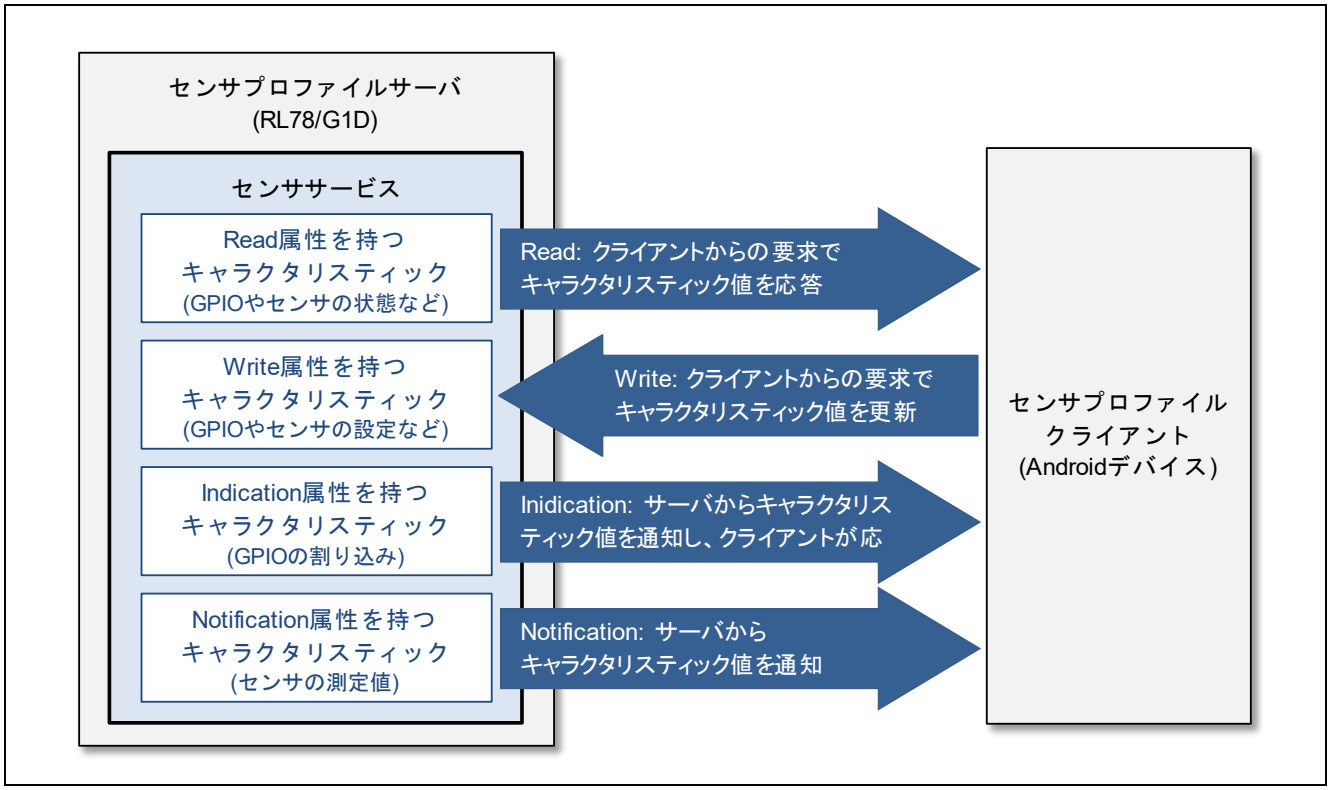


図 2-3　センサプロファイル

2.3.1　サービス仕様

表 2-1 にサンプルプログラムのセンササービス仕様を示します。

表 2-1　センササービス仕様

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 入出力モード(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 入出力値(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 割り込み通知(1byte)**
0x0013	Client Characteristic Configuration Descriptor (0x2902)	Properties: Read, Write (0x0A) **Indication 設定値(2byte)**
Sensor Availability Characteristic		
0x0014	Characteristic Declaration (0x2803)	Properties: Read (0x02) Value Handle: 0x0015 UUID: 7C570005-1449-4D27-9206-BCFDEA46A0FF
0x0015	Sensor Availability	**センサ利用可否(1byte)**
Sensor Operation Characteristic		
0x0016	Characteristic Declaration (0x2803)	Properties: Read, Write (0x0A) Value Handle: 0x0017 UUID: 7C570006-1449-4D27-9206-BCFDEA46A0FF
0x0017	Sensor Operation	**センサ動作(1byte)**
Sensor Notification Interval Characteristic		
0x0018	Characteristic Declaration (0x2803)	Properties: Read, Write (0x0A) Value Handle: 0x0019 UUID: 7C570007-1449-4D27-9206-BCFDEA46A0FF
0x0019	Sensor Notificatioin Interval	**センサ通知間隔(2byte)**
Sensor Value Characteristic		
0x001A	Characteristic Declaration (0x2803)	Properties: Notification (0x10) Value Handle: 0x001B UUID: 7C570008-1449-4D27-9206-BCFDEA46A0FF
0x001B	Sensor Value	**センサ測定値(16byte)**
0x001C	Client Characteristic Configuration Descriptor (0x2902)	Properties: Read, Write (0x0A) **Notification 設定値(2byte)**

GPIO Mode

各 GPIO のデジタル入出力モードを示します。なお未使用設定のポートは常に 0 となります。

0: Output

1: Input

表 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

各 GPIO のデジタル入出力値を示します。なお未使用設定のポートは常に 0 となります。本キャラクタリスティック値に書き込むことで出力ポートの値を変更できます。入力ポートの値を読み込む場合は、本キャラクタリスティック値への書き込み後、読み込んでください。

0: Low

1: High

表 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

外部入力割り込みを通知します。割り込み発生時、GPIO Value の値も更新されます。

0: No Interrupt

1: Interrupt Generated

表 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

外部入力割り込みの通知を制御します。

0x0000: Indication 通知停止

0x0002: Indication 通知開始

表 2-5 GPIO Interrupt Input Indication Configuration

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

	b0:7
[0]	Indication 設定値 (LSB)
[1]	Indication 設定値 (MSB)

Sensor Availability

各センサの利用可否を示します。

0: Not Available
1: Available

表 2-6　Sensor Availability

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

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

Sensor Operation

各センサの停止/開始のいずれかの動作状態を示します。本キャラクタリスティック値に書き込むことで各センサの動作を変更できます。なお本キャラクタリスティック値は Sensor Value の Notification 通知が停止中に書き込んでください。Notification 通知の開始中の書き込みは無視されます。

0: Stop
1: Start

表 2-7　Sensor Operation

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

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

Sensor Notification Interval

センサ測定値の通知間隔を 10msec 単位で示します。本キャラクタリスティック値に書き込むことで Sensor Value の Notification 間隔を変更できます。なお通知間隔は接続インターバル以上を設定してください。接続インターバル未満の通知間隔を設定した場合、接続インターバル値に切り上げられます。

表 2-8　Sensor Notification Interval

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

	b0:7
[0]	センサ測定間隔 (LSB)
[1]	センサ測定間隔 (MSB)

Sensor Value

各センサの測定値を示します。なお未使用センサの測定値は常に 0 となります。

表 2-9　Sensor Value

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

	b0:7
[0]	センサ 0 測定値 (LSB)
[1]	センサ 0 測定値 (MSB)
[2]	センサ 1 測定値 (LSB)
[3]	センサ 1 測定値 (MSB)
:	:
[12]	センサ 6 測定値 (LSB)
[13]	センサ 6 測定値 (MSB)
[14]	センサ 7 測定値 (LSB)
[15]	センサ 7 測定値 (MSB)

Sensor Value Notification Configuration

Sensor Value の Notification 通知設定を示します。

0x0000: Notification 通知停止

0x0001: Notification 通知開始

表 2-10　Sensor Value Notification Configuration

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

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

2.3.2　サービスへのアクセス

RL78/G1D の GPIO を制御する場合のセンササービスへのアクセス例を**図 2-4** に示します。

初期化処理として、GPIO の入出力モードと入出力値を取得後、割り込み通知を開始します。

GPIO の出力値の変更は GPIO Value キャラクタリスティック値に書き込み、入力値の取得は GPIO Value キャラクタリスティック値を読み込みます。

GPIO の入力割り込みの発生は、GPIO Interrput Input キャラクタリスティック値で通知されます。

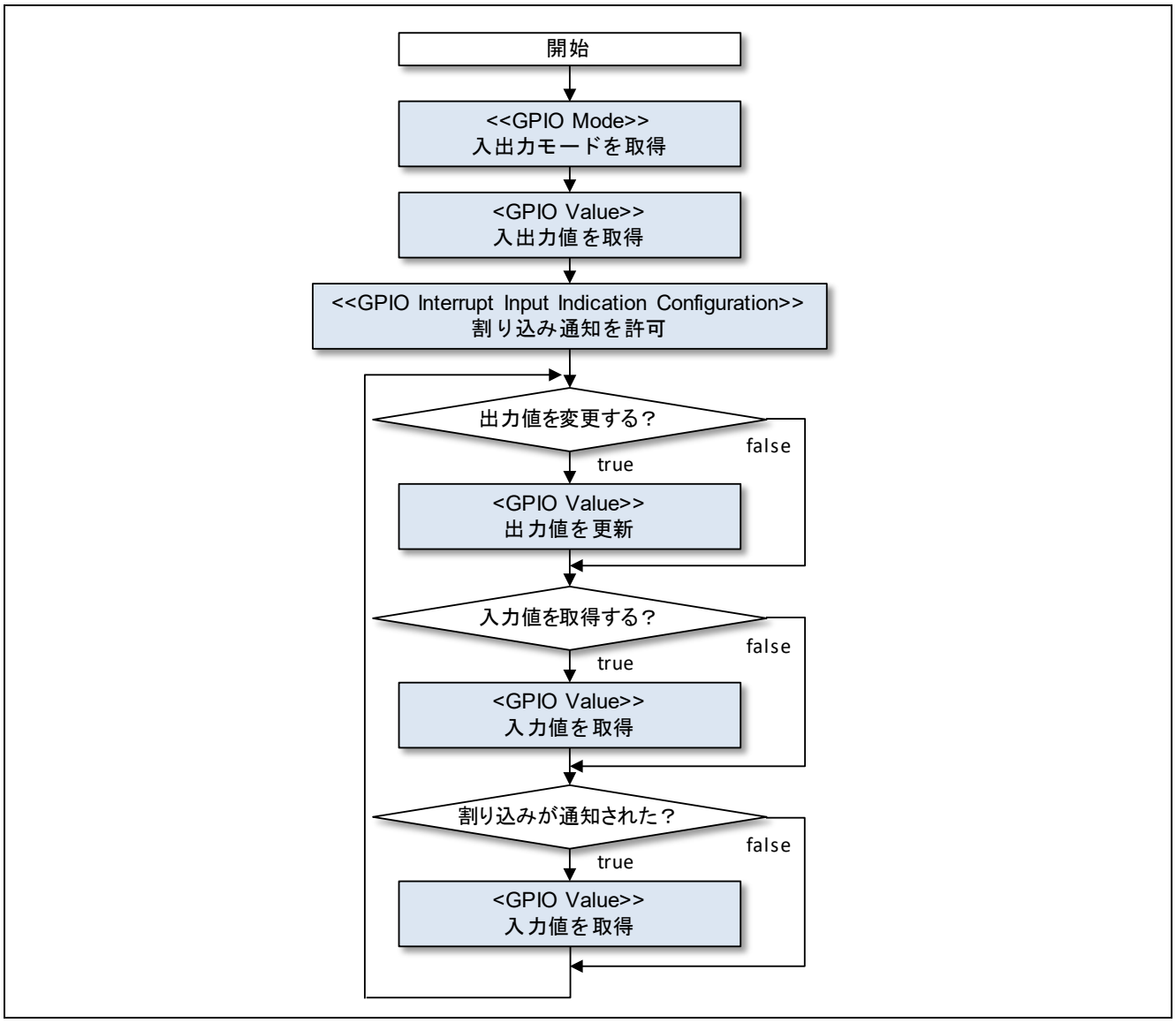


図 2-4　GPIO の制御例

センサの測定値を取得する場合のセンササービスへのアクセス例を**図 2-5** に示します。

初期化処理として、有効なセンサを取得し、センサの動作を開始後、センサ測定値の通知を開始します。また必要に応じて、センサ測定値の通知間隔を設定します。

センサの測定値は、Sensor Value キャラクタリスティック値で定期的に通知されます。

終了処理として、センサ測定の通知を停止し、センサの動作を停止します。

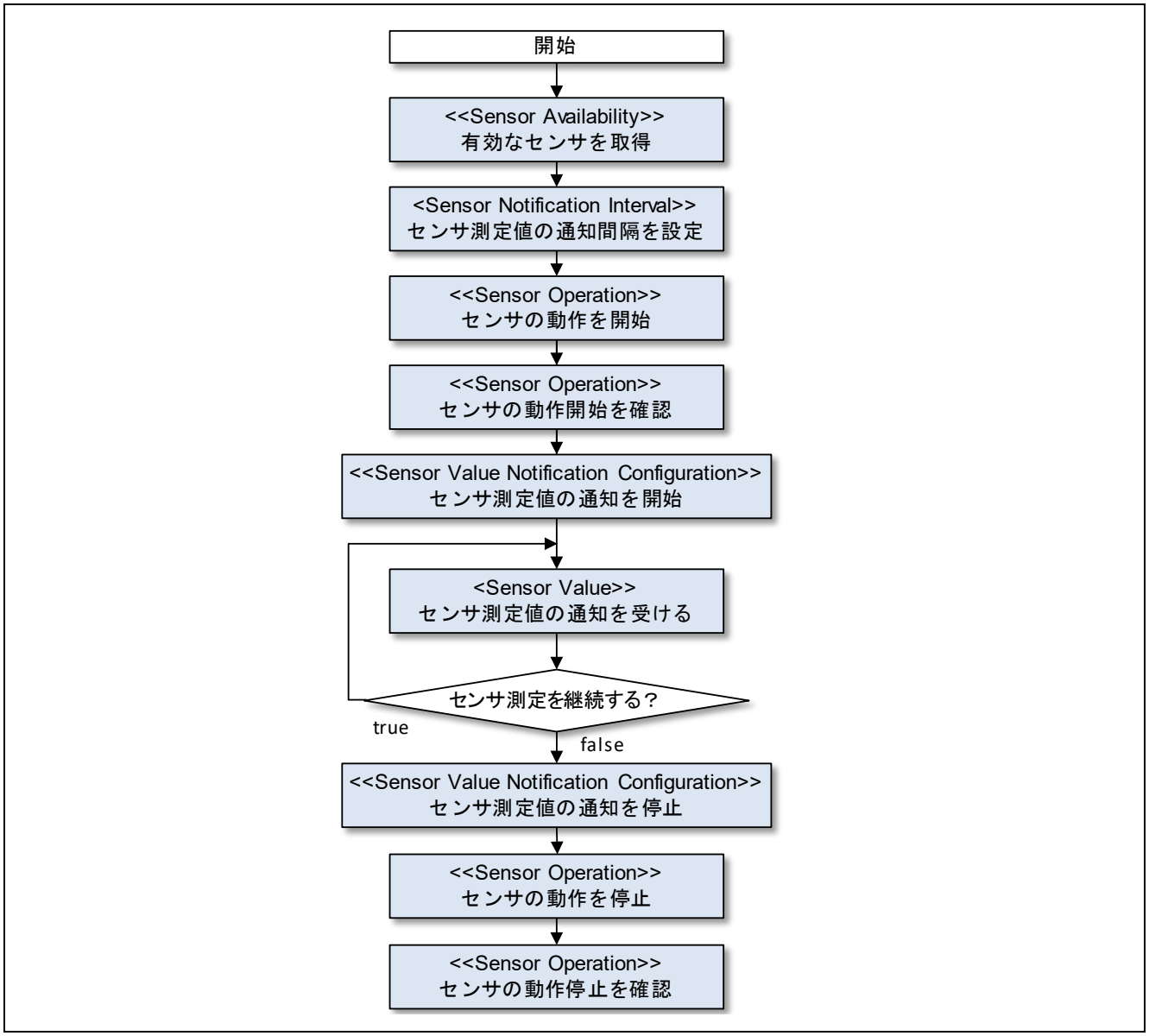


図 2-5　センサ測定値の取得例

3. 操作方法

本章ではサンプルプログラムの操作方法を示します。

3.1 動作環境

サンプルプログラムのビルドと動作確認で使用する環境を示します。

- ハードウェア環境
  - ホストマシン
    - PC/AT™ 互換機
  - デバイス
    - RL78/G1D 評価ボード(RTK0EN0001D01001BZ)
    - Android デバイス (Version 4.4 KitKat 以降)
    - アナログ出力センサ※
    - I²C スレーブ・センサ※

    ※動作確認に使用するセンサについては 3.4 節「センサの接続」を参照してください。
  - ツール
    - Renesas オンチップデバッギングエミュレータ E1 (R0E000010KCE00)

- ソフトウェア環境
  - Windows®10
  - Renesas CS+ for CC V6.01.00 / Renesas CC-RL V1.06.00
  - Renesas Flash Programmer v3.05.00
  - Tera Term Pro (またはシリアルポートと接続可能なターミナルソフト)
  - UART-USB 変換デバイスドライバ※

  ※RL78/G1D 評価ボードと PC を接続する際に、UART-USB 変換 IC「FT232RL」のデバイスドライバを要求される場合があります。その際はドライバを以下から入手してください。
  - FTDI (Future Technology Devices International) - Drivers
    http://www.ftdichip.com/Drivers/D2XX.htm

- ソフトウェアライブラリ※
  - BLE プロトコルスタック : Bluetooth Low Energy Protocol Stack V1.21
  - データフラッシュライブラリ : EEPROM Emulation Library Pack02 for CC-RL Compiler Ver1.01

  ※サンプルプログラムのパッケージには、上記のソフトウェアライブラリが同梱されています。また各ライブラリはルネサスの WEB サイトからダウンロード可能です。ライブラリの入手については後述の 4.2 節「ライブラリの入手」を参照してください。

3.2　スライドスイッチの設定

図 3-1 に RL78/G1D 評価ボードのスライドスイッチを示します。

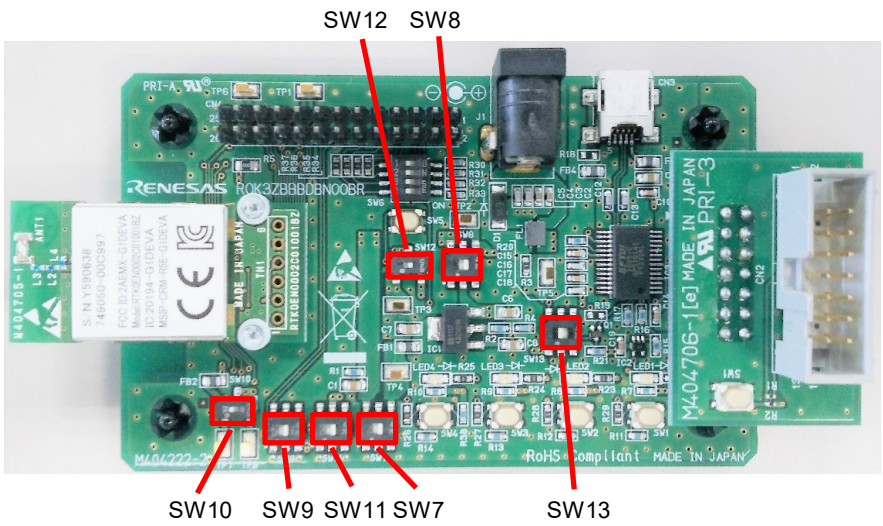


図 3-1　RL78/G1D 評価ボードのスライドスイッチ

表 3-1 に RL78/G1D 評価ボードのスライドスイッチ設定を示します。

表 3-1　スライドスイッチ設定

スイッチ	設定	説明
SW7	2-3 接続(右側)	DC ジャック(J1)または USB インタフェース(CN3)からレギュレータ経由で電源供給 ※DC ジャック(J1)とバッテリを接続して直接供給する場合は 1-2 接続(左側)
SW8	2-3 接続(右側)	USB インタフェース(CN3)をレギュレータに接続して電源供給 ※DC ジャック(J1)をレギュレータに接続して電源供給する場合は 1-2 接続(左側)
SW9	2-3 接続(右側)	USB と接続
SW10	1-2 接続(左側)	モジュールに電源供給
SW11	2-3 接続(右側)	E1 エミュレータ 3.3V 以外から電源供給
SW12	2-3 接続(右側)	(デフォルト固定)
SW13	1-2 接続(左側)	USB 接続

評価ボードの電源に関するスライドスイッチ設定については『RL78/G1D 評価ボード ユーザーズマニュアル』(R30UZ0048)の 6.1 節「電源系統」を参照してください。

3.3　ファームウェアの書き込み

図 3-2 にファームウェアの書き込み方法を示します。

ファームウェアの書き込みは、ホストマシンと接続した E1 エミュレータを使用し、ホストマシン上で Renesas Flash Programmer を実行します。

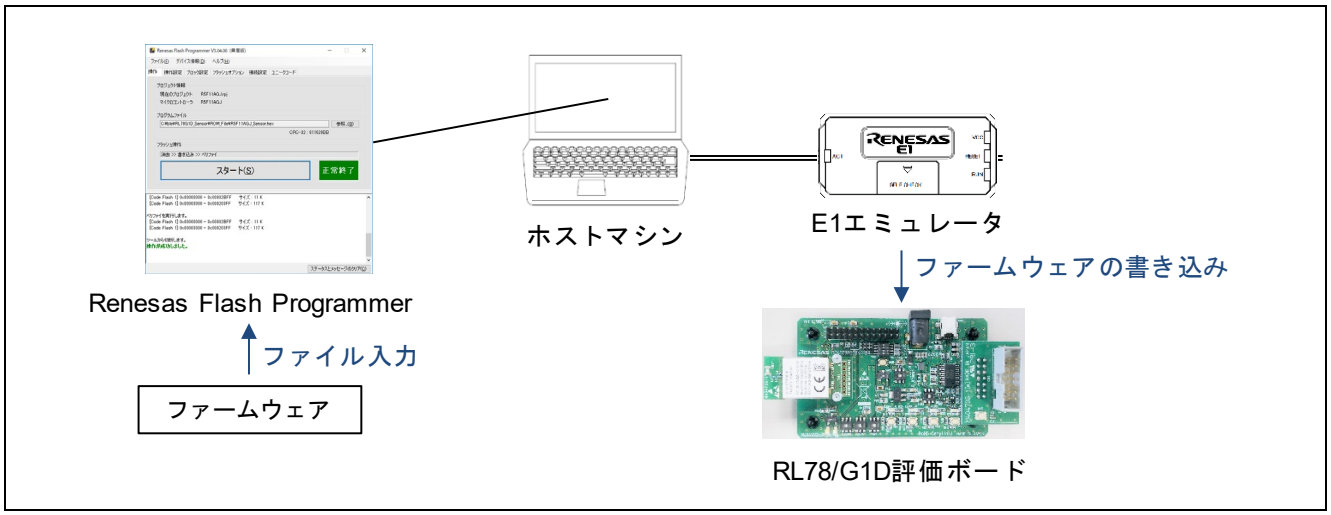


図 3-2　ファームウェア書き込み時の RL78/G1D 評価ボード操作

E1 エミュレータの詳細については『E1/E20 エミュレータ ユーザーズマニュアル』(R20UT0398)および『E1/E20 エミュレータ, E2 エミュレータ Lite ユーザーズマニュアル別冊 (RL78 接続時の注意事項)』(R20UT1994)を参照してください。

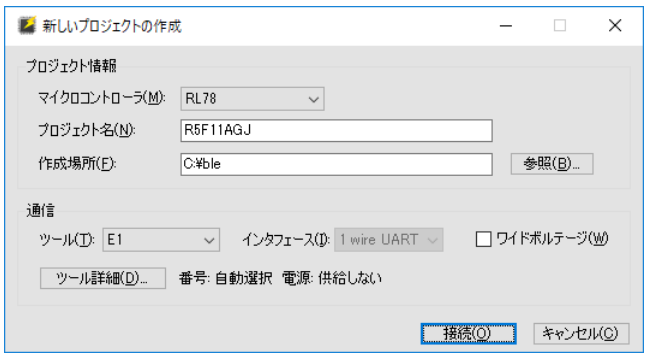
ファームウェアの RL78/G1D 評価ボードへの書き込み手順を以下に示します。

1. E1 エミュレータを評価ボードに接続後、E1 エミュレータとホストマシンを接続します。
2. 評価ボードに DC ジャックまたは USB インタフェースから電源を供給します。
3. Renesas Flash Programmer を起動し、下記の手順でプロジェクトを作成します。

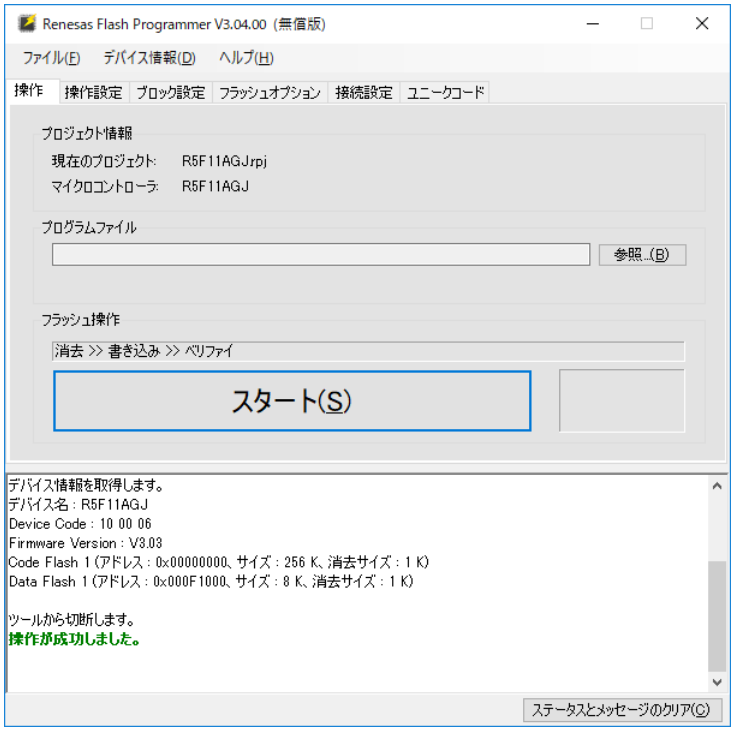
   ※プロジェクト作成後は、作成したプロジェクトを使用することで本手順を省略可能です。

   3-1.　[ファイル]→[新しいプロジェクトを作成]を選択します。

   3-2.　[新しいプロジェクトの作成]ダイアログの[プロジェクト情報]で[RL78]を選択し、任意のプロジェクト名を入力後、[接続]をクリックします。



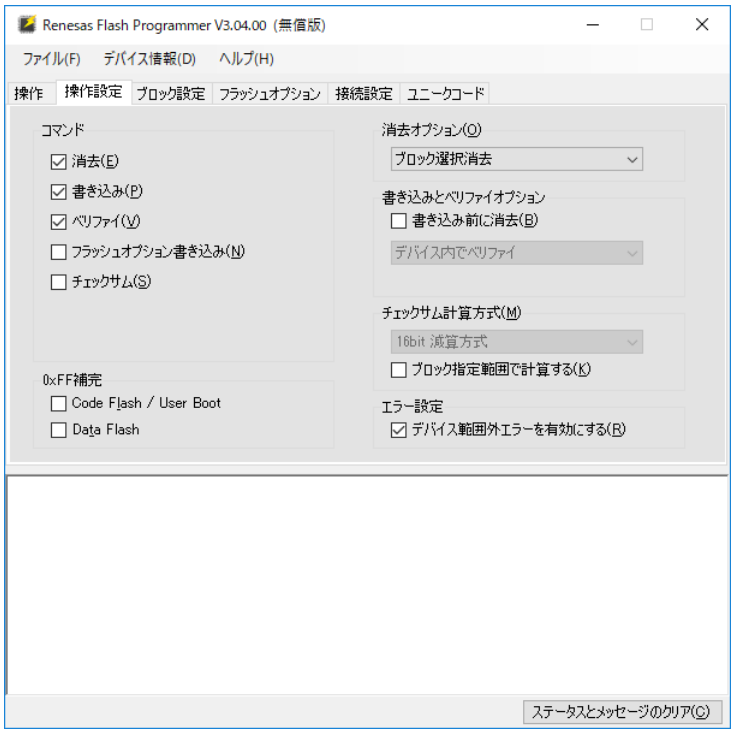
3-3.　ログ出力ウィンドウに「操作が成功しました」と表示されることを確認します。



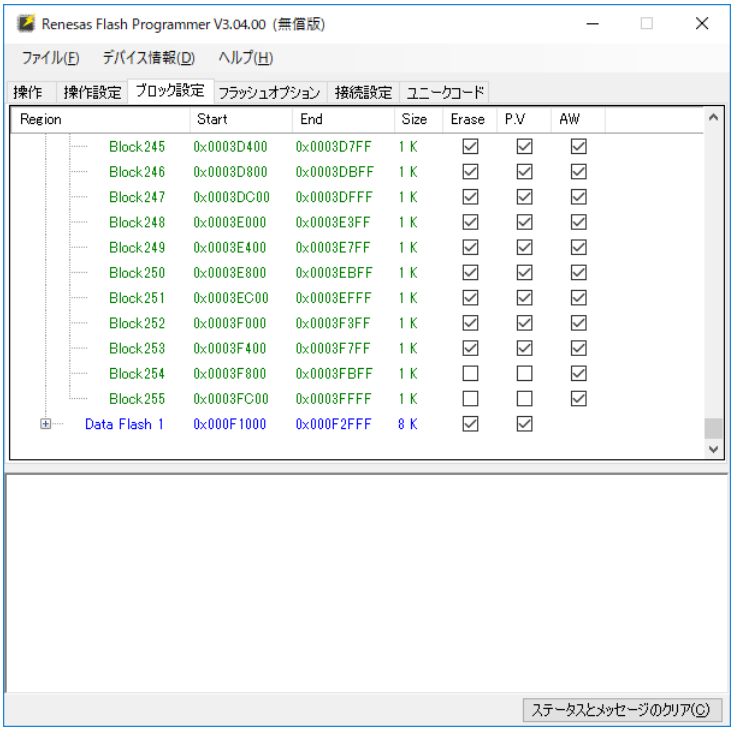
4.　下記の手順でコードフラッシュメモリの Block254, 255 消去を防止します。

※RL78/G1D モジュール(RY7011)では Block254 に出荷時検査フラグ、Block255 にデバイスアドレスが書き込まれています。

4-1.　[操作設定]タブを選択し、[消去オプション]で[ブロック選択消去]を選択します。



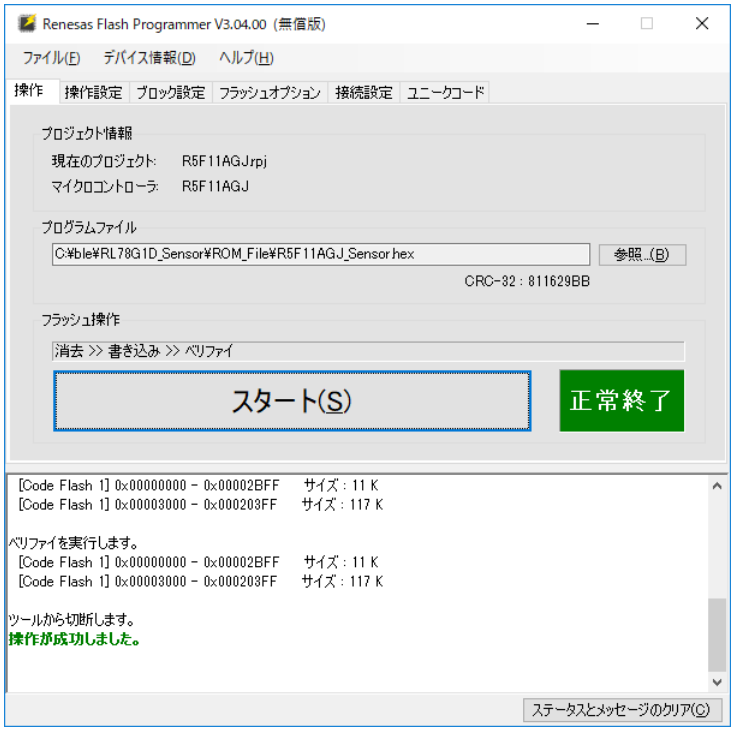
4-2.　[ブロック設定]タブを選択し、Block254, 255 の[Erase]、[P.V]のチェックを外します。



5.　[操作]タブを選択し、[プログラムファイル]で下記のファームウェアファイルを指定します。

- ROM_File/R5F11AGJ_Sensor.hex

6.　[スタート]を押下して書き込み開始後、「正常終了」と表示されることを確認します。



7.　電源、E1 エミュレータを評価ボードから取り外します。

3.4　センサの接続

図 3-3 に RL78/G1D 評価ボードの外部拡張端子 CN4 を示します。CN4 には Pin1～26 があります。

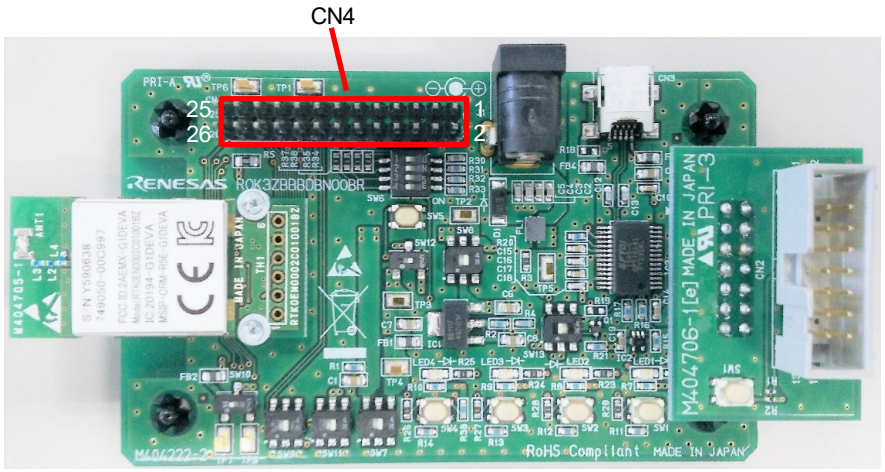


図 3-3　RL78/G1D 評価ボードの外部拡張端子

表 3-2 に RL78/G1D 評価ボードの外部拡張端子を示します。

サンプルプログラムはデジタル入出力ポートや A/D 変換のアナログ入力、I²C マスタのシリアル・データ・バスを使用します。動作確認では、評価ボードの LED やスイッチが使用できます。

表 3-2　RL78/G1D 評価ボードの外部拡張端子

Pin	RL78/G1D の端子	評価ボードの I/O	サンプルプログラムが使用する RL78/G1D の機能	
1	P30/INTP3	FT232RL	P30/INTP3	デジタル入力/外部入力割り込み
2	VCC	-	-	-
3	P61/SDAA0	-	SDAA0	I²C マスタのデータ・バス
4	GND	-	-	-
5	P23/ANI3	SW3 ※2	P23	デジタル入力
6	P10/SCK00/SCL00	SW6-1	P10	デジタル入力
7	P147/ANI18	LED2	P147	デジタル出力
8	GPIO1/TXSELL_RF	SW6-2 ※1	-	-
9	P03/ANI16/RxD1	LED3	P03	デジタル出力
10	GPIO0/TXSELH_RF	SW6-3 ※1	-	-
11	P60/SCLA0	LED4	SCLA0	I²C マスタのシリアル・クロック
12	P02/ANI17/TxD1	SW6-4	P02	デジタル入力
13	P22/ANI2	SW4	P22	デジタル入力
14	P12/SO00/TxD0/TOOLTxD	-	TxD0	デバッグ用 UART
15	P120/ANI19	LED1	ANI19	A/D 変換のアナログ入力
16	P11/SI00/RxD0/TOOLRxD/SDA00	FT232RL	-	-
17	VCC	-	-	-
18	-	SW1 ※1	-	-
19	GND	-	-	-
20	P16/TI01/TO01/INTP5	SW2	P16/INTP5	デジタル入力/入力割り込み
21	P40/TOOL0	-	-	-
22	RESET	-	-	-
23	-	-	-	-
24	5V	-	-	-
25	GND	-	-	-
26	GND	-	-	-

※1：SW1,SW3,SW6-2,SW6-3 は RL78/G1D 端子と未接続のため使用できません。
※2：SW3 の使用時は外部プルアップが必要です。

RL78/G1D 評価ボードとセンサを接続します。なお一部のセンサを接続しない場合でも、動作確認はできます。

- I²C スレーブデバイス・センサ ※

  RGB ライトセンサ - 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

  例) ISL29125 RGB ライトセンサ - スイッチサイエンス
  https://www.switch-science.com/catalog/1928/

- アナログ信号出力デバイス

  可変抵抗 50kΩ

※サンプルプログラムでは、RGB ライトセンサ ISL29125 のデバイスドライバが実装されています。

他の I²C デバイスを使用する場合は、対象デバイスを制御するためのデバイスドライバを実装し、ISL29125 ドライバと置き換えてください。

サンプルプログラムのセンサ制御に関する設計情報は、下記を参照してください。

- 5 章「センサ制御」
- 6 章「関数仕様」

1. **図 3-4** を参照し、RL78/G1D 評価ボードに ISL29125 と可変抵抗を接続します。

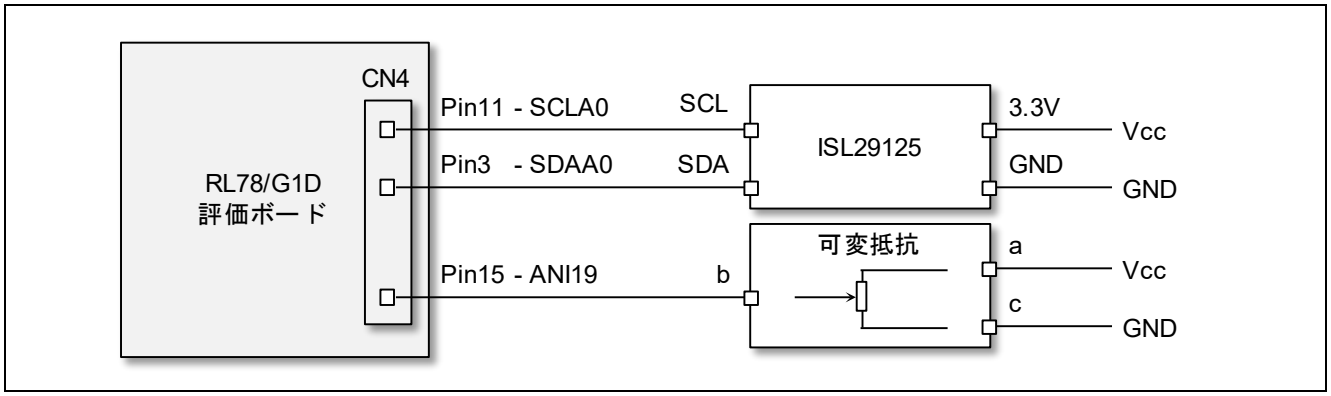


図 3-4　RL78/G1D 評価ボードへのセンサ接続

3.5　アプリのインストール

Android アプリの BleSensor を Android デバイスにインストールします。

1. Android デバイスの「設定」→「セキュリティ」→「提供元不明のアプリ」で、提供元不明アプリのインストールを許可します。
2. 下記のパッケージファイルを PC から Android デバイスにメールに添付して転送します。
   - Android_File/BleSensor.apk
3. Android デバイスで上記のメールを受信し、添付のパッケージファイルを実行します。
4. BleSensor のインストール画面でインストールを実行します。

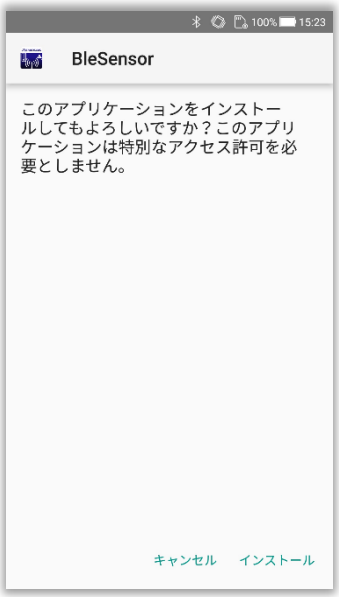


図 3-5　Android アプリのインストール

5. BleSensor のインストールが完了することを確認します。
6. Android OS 6 以降をご使用の場合、Android デバイスの「設定」→「アプリと通知」→「アプリ情報」→「BleSensor」→「権限」で、ストレージと位置情報の権限を付与します。

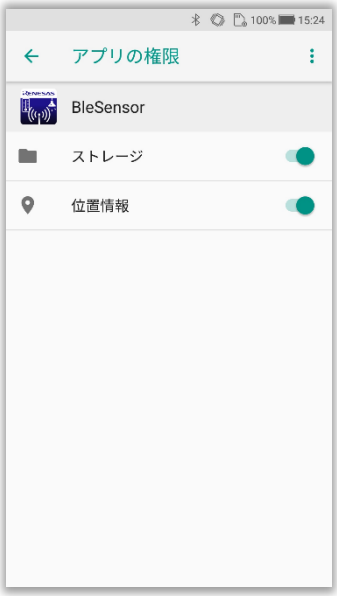


図 3-6　権限の設定

3.6　接続の確立

Android デバイスと RL78/G1D で Bluetooth Low Energy 接続を確立します。

1. Android デバイスの「設定」→「Bluetooth」で、Bluetooth 機能を ON にします。

   ※Android デバイスには、Bluetooth 端末を"信頼できる端末"として登録し、画面ロックを自動解除する機能(Smart Lock 機能)があります。サンプルプログラムの動作を確認する際は、RL78/G1D を Smart Lock 機能の信頼できる端末として追加しないでください。

2. 3.5 節でインストールしたアプリを起動します。

   Android アプリはデバイス探索画面を表示し、デバイスの探索(Scan)を自動で開始します。

   デバイス探索画面には、接続可能なデバイスのデバイス名と受信強度(RSSI)が表示されます。

図 3-7　デバイス探索画面

3. デバイスの探索結果から"RL78/G1D Sensor"を選択し、接続を確立します。

   ※センササービスを持たないデバイスに接続した場合、自動で切断しデバイスの探索を再開します。

3.7　GPIO の制御

RL78/G1D の GPIO を Android デバイスで制御します。

1. RL78/G1D と接続を確立すると、Android アプリは GPIO 制御画面に遷移します。

   GPIO 制御画面には RL78/G1D のポート名、入出力モード、デジタル入出力値が表示されます。

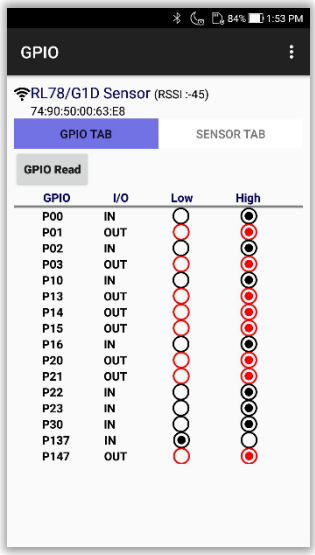
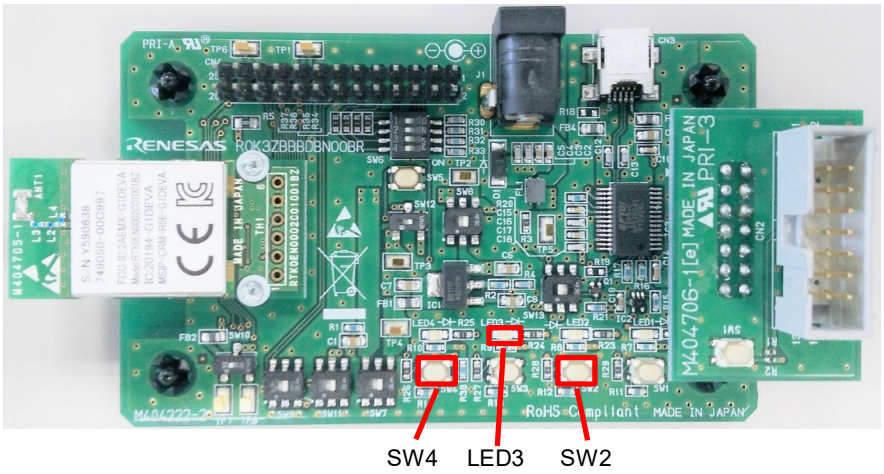


図 3-8　GPIO 制御画面

2. GPIO 制御画面で出力ポートである"P03"のデジタル出力値を変更すると、評価ボードの LED3 の点灯状態が変化します。



3. SW2 を押下すると、GPIO 制御画面で入力ポートである"P16"の入力値が変化します。

   SW2 を押下した状態では P16 は Low、SW2 を押下しない状態では P16 は High となります。

4. SW4 を押下した状態で GPIO 制御画面の Read ボタンを押下すると、入力ポートである"P22"の入力値が Low に変化します。

   SW4 を押下しない状態で GPIO 制御画面の Read ボタンを押下すると、入力ポートである"P22"の入力値が High に変化します。

3.8　センサ測定値の確認

RL78/G1D 評価ボードに接続したセンサの測定値を Android デバイスで確認します。

1. GPIO 制御画面で SENSOR TAB を選択すると、センサ測定画面に遷移します。

   センサ測定画面にはセンサの測定値を表示するためのグラフと、各センサの停止・開始を制御するチェックボックス、センサ測定値の通知間隔を設定するためのスライダーが表示されます。

   各センサのチェックボックスは下記のセンサに対応します。

   - センサ 0　　: A/D コンバータ
   - センサ 1　　: ISL29125 RGB ライトセンサ (Green)
   - センサ 2　　: ISL29125 RGB ライトセンサ (Red)
   - センサ 3　　: ISL29125 RGB ライトセンサ (Blue)

   またセンサの測定値は自動で CSV(Comma Separated Values)形式のログファイルに出力されます。

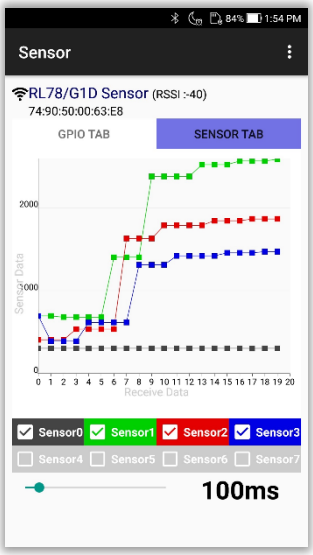


図 3-9　センサ測定画面

2. センサ測定画面でセンサ 0 にチェックを入れると、RL78/G1D の A/D 変換が開始されます。A/D 変換の結果はグラフに黒線で表示されます。

   評価ボードに接続した可変抵抗を回すと、グラフに表示された A/D 変換の結果が変化します。

3. センサ測定画面でセンサ 1、センサ 2、センサ 3 にチェックを入れると、RGB ライトセンサのそれぞれ G (緑)、R(赤)、B(青)の測定が開始されます。それぞれの測定結果はグラフに表示されます。

   RGB ライトセンサ表面を明るくしたり暗くしたりすると、RGB の測定結果が変化します。

4. センサ測定画面で通知間隔スライダーを操作すると、RL78/G1D からのセンサ測定値の通知間隔が変更されます。

5. 画面上の GPIO TAB を選択すると、再度 GPIO 制御画面に遷移します。

6. Android デバイスの戻るボタンを押下すると、接続が切断され、デバイス探索画面に遷移します。

3.9　センサ測定ログの確認

Android デバイスに保存されたセンサ測定値のログファイルを PC で確認します。

1. Android デバイスを PC に接続します。接続方式はファイル転送が可能な MTP 形式を選択します。
2. PC で Explorer を立ち上げ、Android デバイスの内部ストレージに BleSensor フォルダがあり、下記のログファイルが生成されていることを確認します。なおログファイル名の Y,M,D,H,M,S は、接続確立の日時を示します。

   ログファイル　：log_YYYY_MM_DD_HH_MM_SS.csv
3. ログファイルは下記のフォーマットで記録されています。ログ内容は PC にコピー後、テキストエディタ、表計算ソフトなどで確認できます。なおログフォーマットの timestamp はセンサ測定値の通知日時、sensor0-sensor7 は各センサの符号なし 2byte 測定値を示します。

   ログフォーマット：timestamp,sensor0,sensor1,sensor2,sensor3,sensor4,sensor5, sensor6,sensor7

図 3-10 に BleSensor が出力するセンサ測定ログの例を示します。

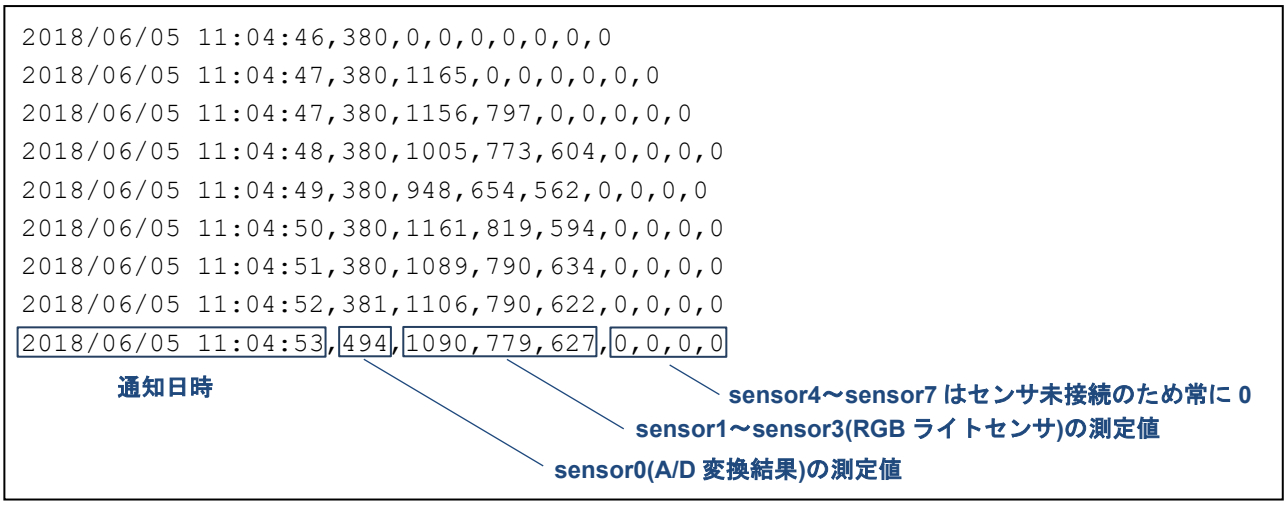


図 3-10　センサ測定ログ例

4. ビルド方法

本章ではサンプルプログラムのビルド方法を示します。

4.1 ファイル構成

サンプルプログラムのパッケージには、RL78/G1D ソフトウェアのソースコードと動作確認のためのファームウェアに加え、Android アプリのパッケージファイルとプロジェクトが同梱されます。

またサンプルプログラムのパッケージには、動作確認のための BLE プロトコルスタックライブラリとデータフラッシュライブラリも同梱されますが、アプリケーション開発の際は最新版のライブラリを入手し、プロジェクトフォルダに配置してください。

ライブラリの入手につきましては、4.2 節「ライブラリの入手」を参照してください。

サンプルプログラムのパッケージに含まれるファイルとフォルダの構成を示します。

```
Android_BleSensor
 ├──Android_BleSensor_V1_0_4.pdf          Android アプリ"BleSensor" アプリケーションノート
 └──Android_BleSensor_V1_0_4.zip          Android アプリ"BleSensor" プロジェクト
RL78G1D_Sensor
 ├──Android_File
 │     BleSensor.apk                      Android アプリ"BleSensor" パッケージファイル
 ├──ROM_File
 │     R5F11AGJ_Sensor.hex                動作確認用ファームウェア
 │     R5F11AGJ_Sensor(console_lvl4).hex   動作確認用ファームウェア(デバッグ UART 出力あり)
 └──Project_Source
   ├──bleip
   │   └──src
   │      ├──common
   │      │    co_bt.h
   │      └──rwble
   │           rwble.h
   │           rwble_config.h
   ├──rBLE
   │   └──src
   │      ├──include
   │      │    rble.h                     ┐ BLE プロトコルスタック
   │      │    rble_api.h                 │ ※動作確認のためのライブラリファイルを同梱
   │      │    rble_rwke.h                ┘ 開発時は最新版を入手して本フォルダに配置する
   │      ├──sample_app
   │      │   │  console.c
   │      │   │  console.h
   │      │   │  rble_sample_app_peripheral.c   ┐ BLE アプリケーション
   │      │   │  rble_sample_app_peripheral.h   ┘
   │      │   │  rble_sample_app_sensor.c       ┐ センサアプリケーション
   │      │   │  rble_sample_app_sensor.h       ┘ ※新規の I2C デバイスを追加する場合、実装変更
   │      │   └──seclib
   │      │        secdb.c                 ┐ セキュリティライブラリ
   │      │        secdb.h                 │
   │      │        seclib.c                │
   │      │        seclib.h                ┘
   │      └──sample_profile
   │          └──sen
   │               sens.c                  ┐ センサプロファイル
```

```
│           sens.h
└──renesas
   ├──lib
   │     BLE_CONTROLLER_LIB_CCRL.lib
   │     BLE_HOST_lib_CCRL.lib
   │     BLE_rBLE_lib_CCRL.lib
   ├──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
   │   │     r_cg_adc.h
   │   │     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
```

BLE プロトコルスタック
※動作確認のためのライブラリファイルを同梱
開発時は最新版を入手して本フォルダに配置する

周辺機能ドライバ
※コード生成プラグインの自動生成ソースコード

```
│   │   │   │   eel_descriptor_t02.h
│   │   │   │   fdl_descriptor_t02.c
│   │   │   │   fdl_descriptor_t02.h
│   │   │   └──cc_rl
│   │   │       eel.h
│   │   │       eel.lib
│   │   │       eel_types.h
│   │   │       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
│       ISL29125.h
└──tools
    └──project
        └──CS_CCRL
            └──BLE_Peripheral
                │   BLE_Peripheral.mtpj
                └──R5F11AGJ_Sensor
                    R5F11AGJ_Sensor.mtsp
```

データフラッシュライブラリ
※動作確認のためのライブラリファイルを同梱
開発時は最新版を入手して本フォルダに配置する

デバイスドライバ
※新規の I^2C デバイスを追加する場合、追加実装

4.2　ライブラリの入手

ファームウェアのビルドには下記のライブラリが必要です。サンプルプログラムのパッケージには動作確認のためのライブラリが同梱されていますが、アプリケーション開発の際は最新版のライブラリを入手してください。

1. 下記の WEB サイトからライブラリをダウンロードします。

   BLE プロトコルスタック:

   Bluetooth Low Energy Protocol Stack V1.21
   https://www.renesas.com/document/lbr/bluetooth-low-energy-protocol-stack-ver121

   データフラッシュライブラリ:

   RL78 ファミリ EEPROM エミュレーションライブラリ Pack02 パッケージ Ver.2.00
   (CA78K0R/CC-RL コンパイラ用)
   https://www.renesas.com/document/upr/eeprom-emulation-library-pack02-package-ver200for-ca78k0rcc-rl-compiler-rl78-family

2. ライブラリのダウンロード後、下記のファイルをコピーします。

   BLE プロトコルスタック:

   - 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

   データフラッシュライブラリ:

   - 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/fdl.lib
   - EEL/CCRL_100/FDL/lib/fdl.h
   - EEL/CCRL_100/FDL/lib/fdl_types.h

3. 上記のファイルをサンプルプログラムの下記のライブラリフォルダに配置します。

```
Project_Source
├─rBLE
│  └─src
│     └─include
│
└─renesas
   ├─lib
   │
   └─src
      └─driver
         └─dataflash
            └─cc_rl
```

rble.h	Protocol Stack rBLE definitions - header file
rble_api.h	Protocol Stack rBLE API - header file

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

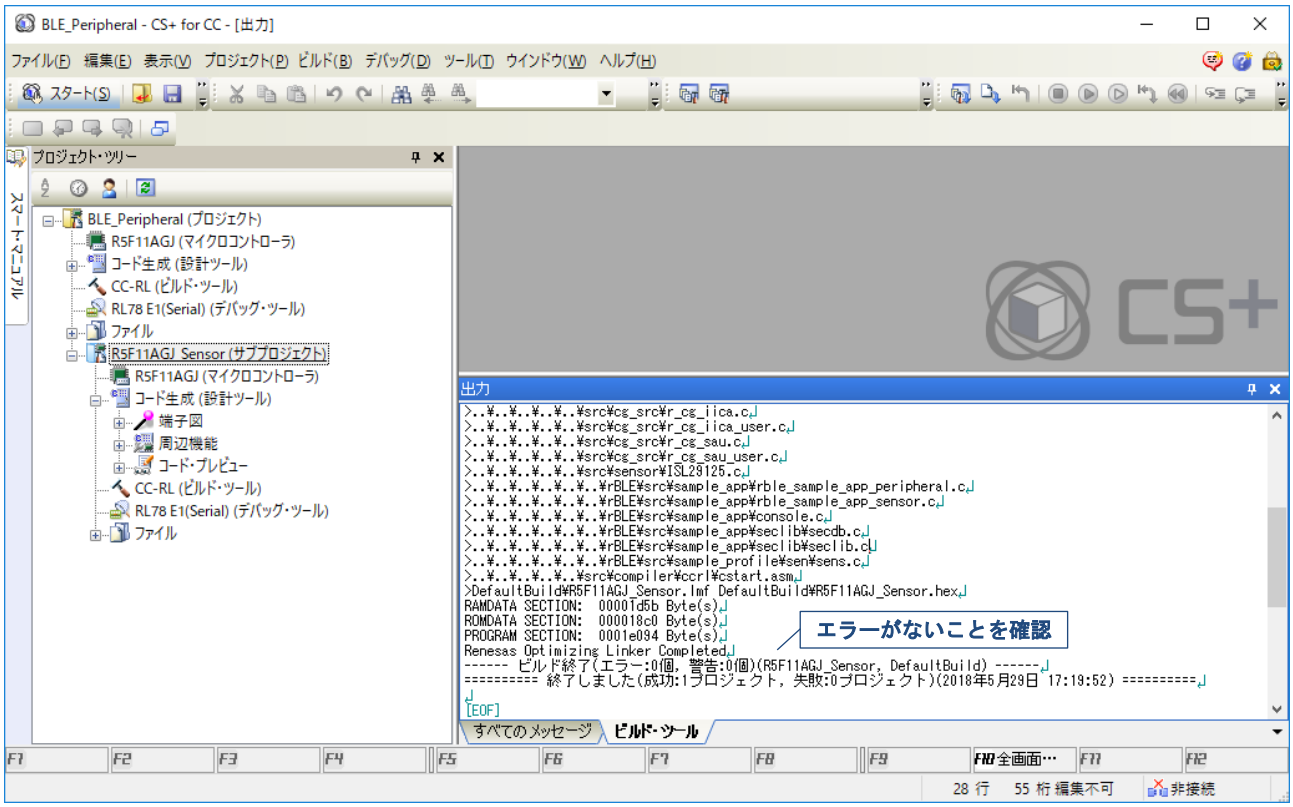
eel.h	Data Flash Library EEPROM Emulation - header file
eel.lib	Data Flash Library EEPROM Emulation - library file
eel_types.h	Data Flash Library EEPROM Emulation type definition -
fdl.h	Data Flash Library - header file
fdl.lib	Data Flash Library - library file

fdl_types.h	Data Flash Library type definition - header file

4.3　ファームウェアのビルド

RL78/G1D で動作するファームウェアのビルドには統合開発環境 CS+ for CC を使用します。ビルドすると、HEX 形式のファームウェアファイル R5F11AGJ_Sensor.hex が生成されます。

1. CS+ for CC を起動し、[ファイル]→[ファイルを開く]を選択して下記のパスにあるプロジェクトファイル BLE_Peripheral.mtpj を開きます。
   - Project_Source/renesas/tools/project/CS_CCRL/BLE_Peripheral
2. [ビルド]→[リビルド・プロジェクト]を選択してファームウェアのビルドを実行します。
3. [すべてのメッセージ]ウィンドウでエラーが無く、ビルドが成功したことを確認します。



4. 下記のパスに R5F11AGJ_Sensor.hex が生成されていることを確認します。
   - Project_Source/renesas/tools/project/CS_CCRL/BLE_Peripheral/R5F11AGJ_Sensor/DefaultBuild

4.4　周辺機能設定

RL78/G1D の周辺機能の制御には、統合開発環境 CS+ for CC のコード生成プラグインが自動生成する周辺機能ドライバを使用できます。

サンプルプログラムが使用する周辺機能は以下の通りです。

- 共通/クロック発生回路　：動作モード設定、高速オンチップ・オシレータ設定
- ポート機能　：入出力モード、デフォルト出力値、内部プルアップ設定
- 割り込み機能　：外部入力割り込みのエッジ設定
- A/D コンバータ　：アナログ入力端子設定、VREF(+,-)設定、分解能設定
- シリアル・インタフェース IICA　：転送クロック設定
- シリアル・アレイ・ユニット　：送受信動作設定、ボーレート設定

図 4-1 に CS+ for CC のコード生成プラグインを示します。

プロジェクト・ツリーで各周辺機能を選択し、表示される各周辺機能タブで設定を変更できます。設定変更後は「コードを生成する」を押下し、設定をソースコードに反映してください。

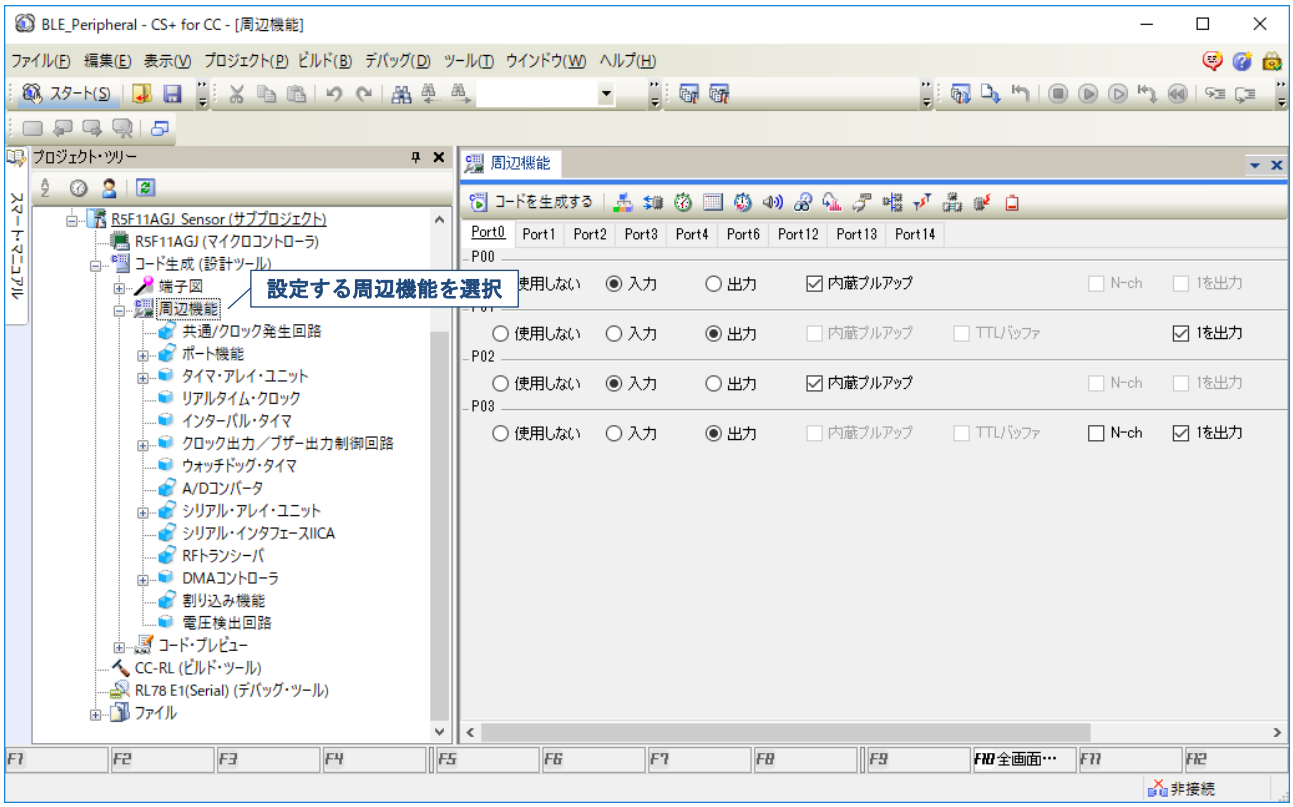


図 4-1　コード生成プラグイン（ポート機能設定）

生成された周辺機能ドライバの関数仕様は、CS+ for CC のスマートマニュアルを参照してください。

スマートマニュアルを表示するには、CS+ for CC で[表示]→[スマートマニュアル]を選択します。

5. センサ制御

本章ではセンサ制御に関連する下記モジュールの動作を示します。

各モジュールのソースコードは、下記のファイルを参照してください。

- BLE アプリケーション　: Project_Source/rBLE/src/sample_app/rble_sample_app_peripheral.c
- セキュリティライブラリ　: Project_Source/rBLE/src/sample_app/seclib/seclib.c
- センサアプリケーション　: Project_Source/rBLE/src/sample_app/r_sample_app_sensor.c
- センサプロファイル　: Project_Source/rBLE/src/sample_profile/sen/sens.c
- ISL29125 ドライバ　: Project_Source/renesas/src/sensor/ISL29125.c
- 周辺機能ドライバ(IICA0)　: Project_Source/renesas/src/cg_src/r_cg_iica.c, r_cg_iica_user.c
- 周辺機能ドライバ(ADC)　: Project_Source/renesas/src/cg_src/r_cg_adc.c, r_cg_adc_user.c

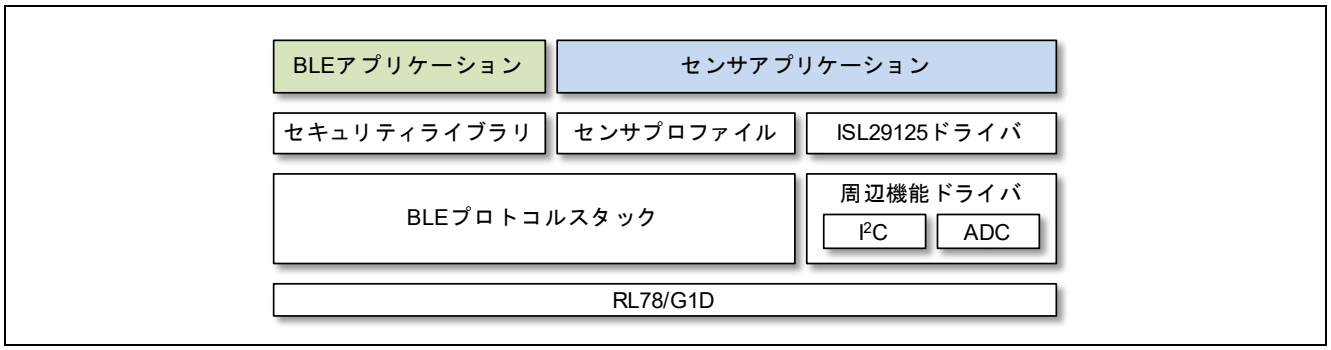


図 5-1　センサ制御に関連するモジュール

図 5-2 にサンプルプログラムのフローチャートを示します。

BLE アプリケーションは BLE 接続の確立と切断、データの暗号化に関する処理を実行します。

センサアプリケーションが実行する処理については、後述のシーケンスを参照してください。

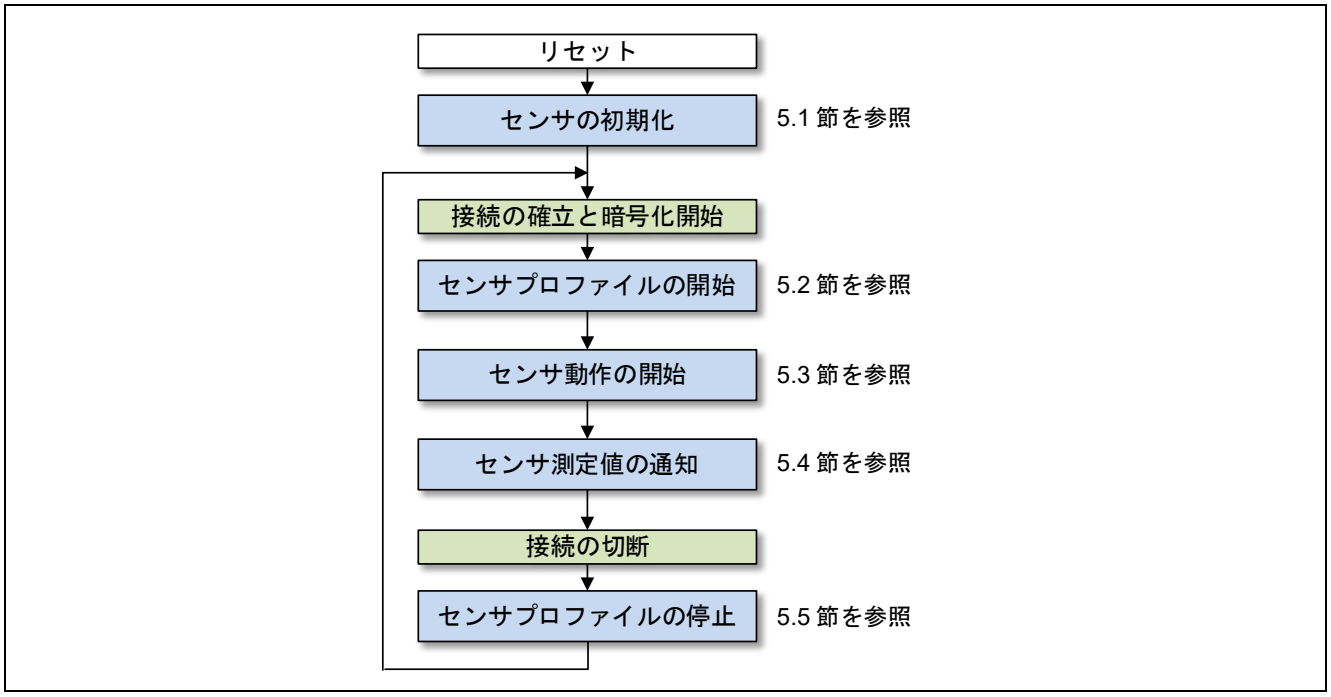


図 5-2　センサアプリケーションのフローチャート

5.1　センサの初期化

図 5-3 にセンサの初期化シーケンスを示します。

本シーケンスは、RL78/G1D のリセット後に 1 回だけ実行され、A/D 変換や I²C 通信などの RL78/G1D の周辺機能を初期化します。

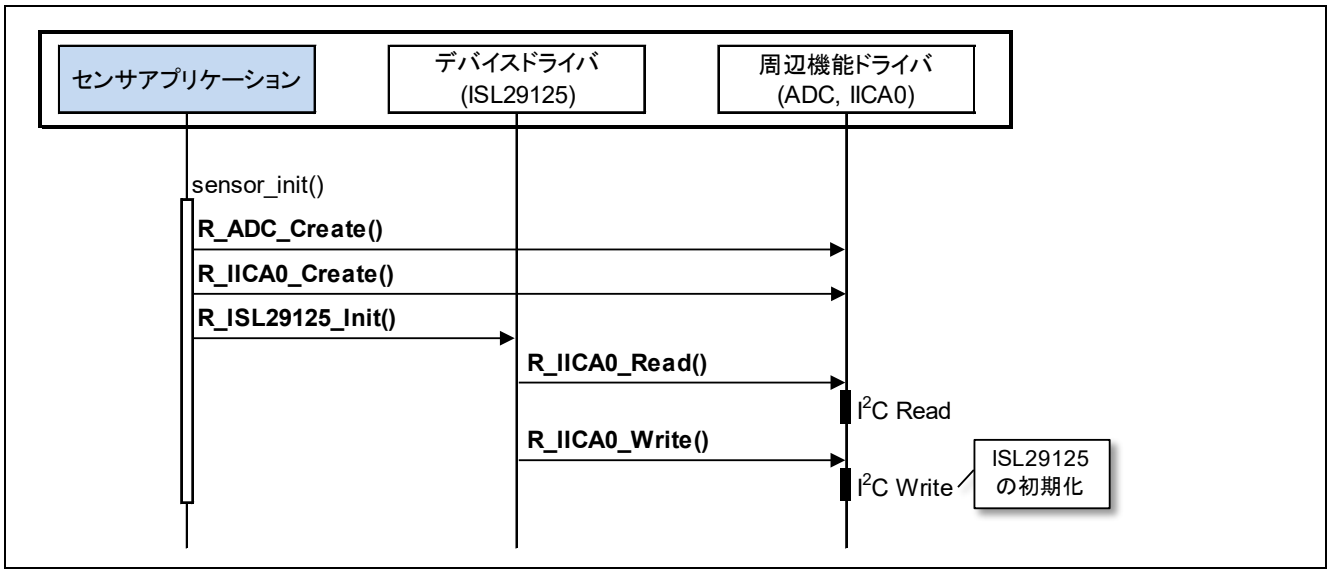


図 5-3　センサ初期化のシーケンス

サンプルプログラムでは、RGB ライトセンサ ISL29125 のデバイスドライバが実装されています。

他の I²C デバイスを使用する場合は、対象デバイスを制御するためのデバイスドライバを実装し、シーケンス上の ISL29125 ドライバと置き換えてください。

5.2　センサプロファイルの開始

図 5-4 にセンサプロファイルの開始シーケンスを示します。

本シーケンスは、リモートデバイスと接続を確立すると実行されます。

センサアプリケーションはセンサプロファイルを開始し、センササービスのキャラクタリスティック値を最新の状態に更新します。

またリモートデバイスは暗号化の開始後、センササービスへのアクセスが可能となります。

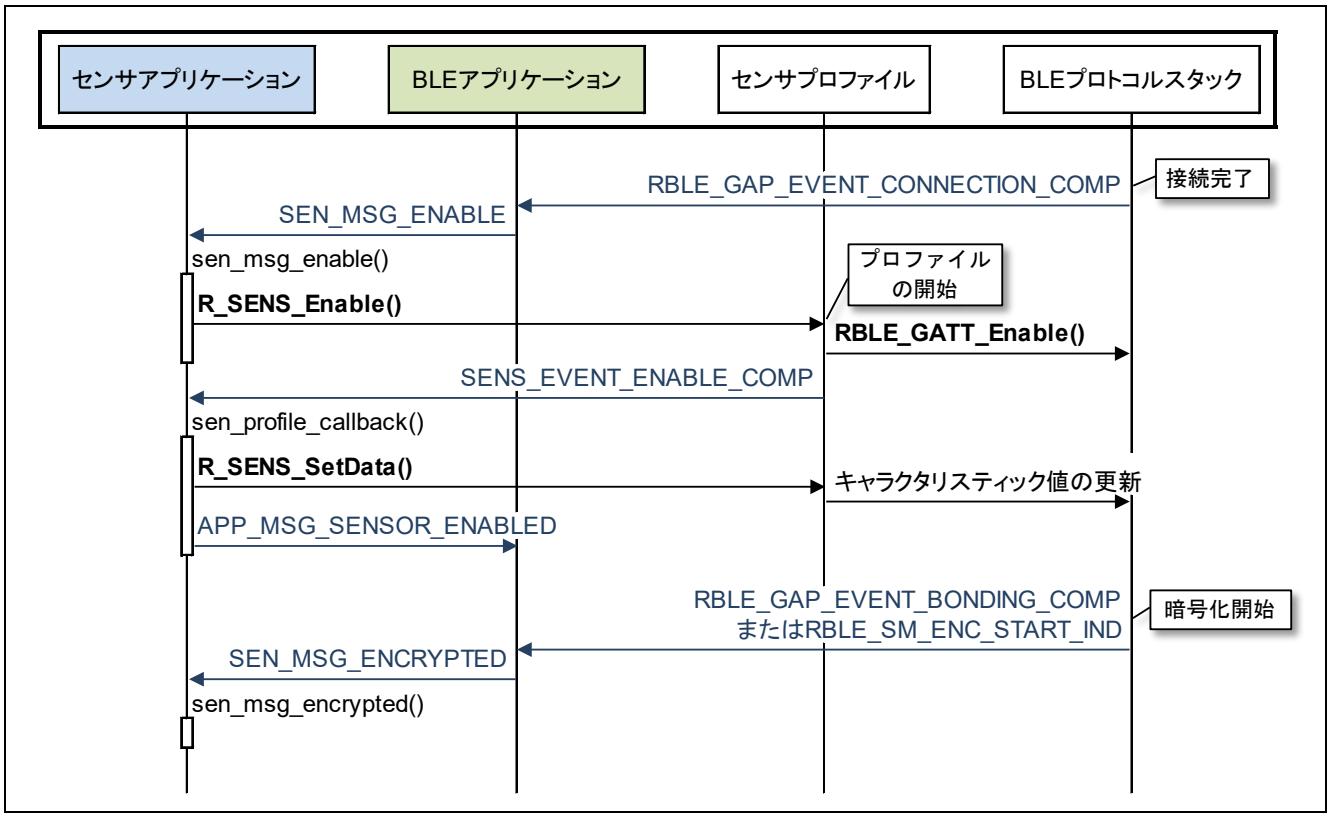


図 5-4　センサプロファイルの開始シーケンス

5.3　センサ動作の開始

図 5-5 にセンサの動作開始シーケンスを示します。

本シーケンスは、リモートデバイスがセンサ動作の開始を要求することで実行されます。

センサアプリケーションはリモートデバイスからの要求により、ISL29125 の測定動作を開始します。また ISL29125 の動作開始後、動作状態を示すキャラクタリスティック値を更新します。

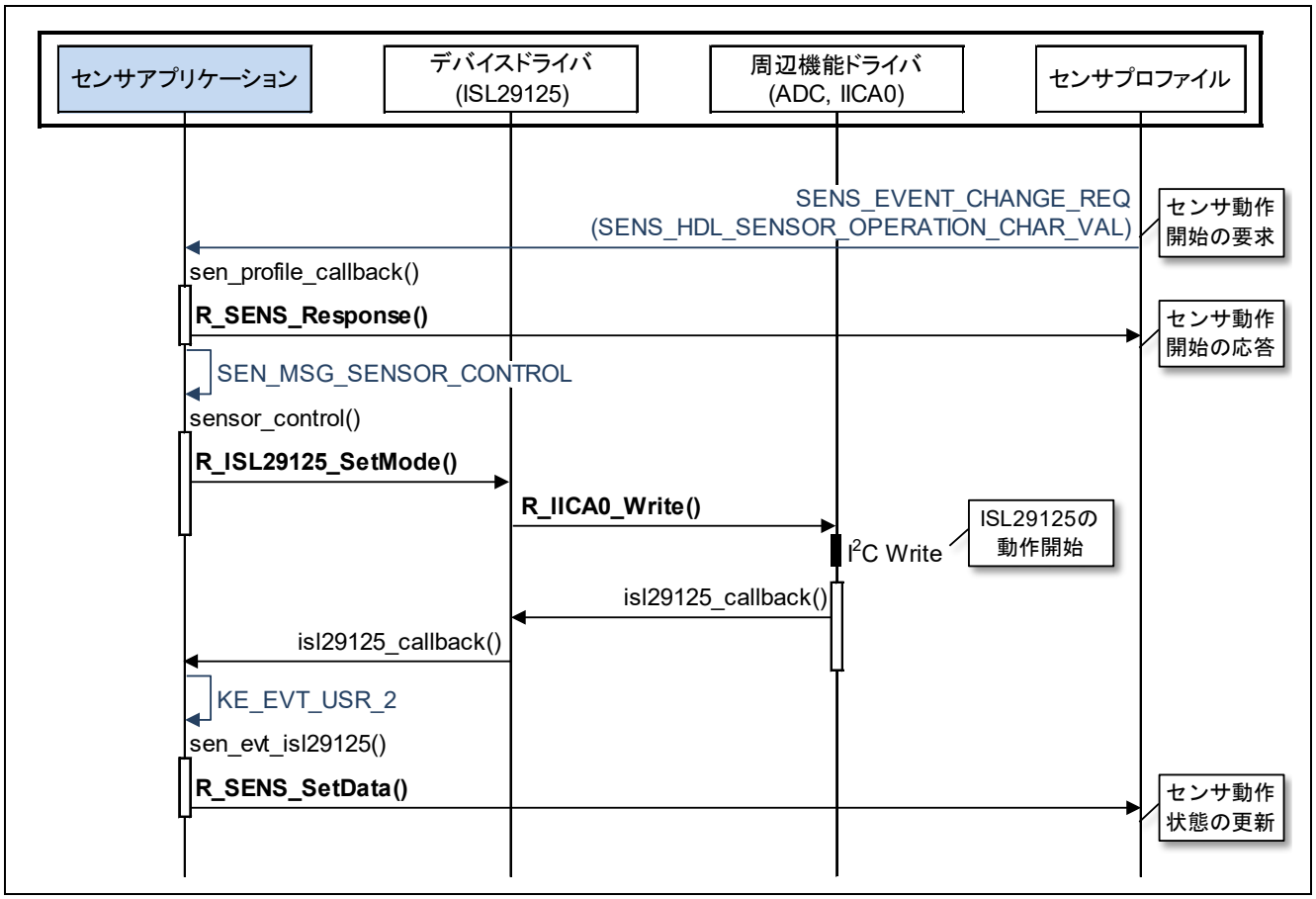


図 5-5　センサ動作の開始シーケンス

サンプルプログラムでは、RGB ライトセンサ ISL29125 のデバイスドライバが実装されています。

他の I²C デバイスを使用する場合は、対象デバイスを制御するためのデバイスドライバを実装し、シーケンス上の ISL29125 ドライバと置き換えてください。

5.4　センサ測定値の通知

図 5-6 にセンサ測定値の通知シーケンスを示します。

本シーケンスは、リモートデバイスがセンサ測定値の通知を許可すると実行されます。

センサアプリケーションはリモートデバイスからの要求により、A/D 変換結果や ISL29125 の測定値をリモートデバイスに周期的に通知します。

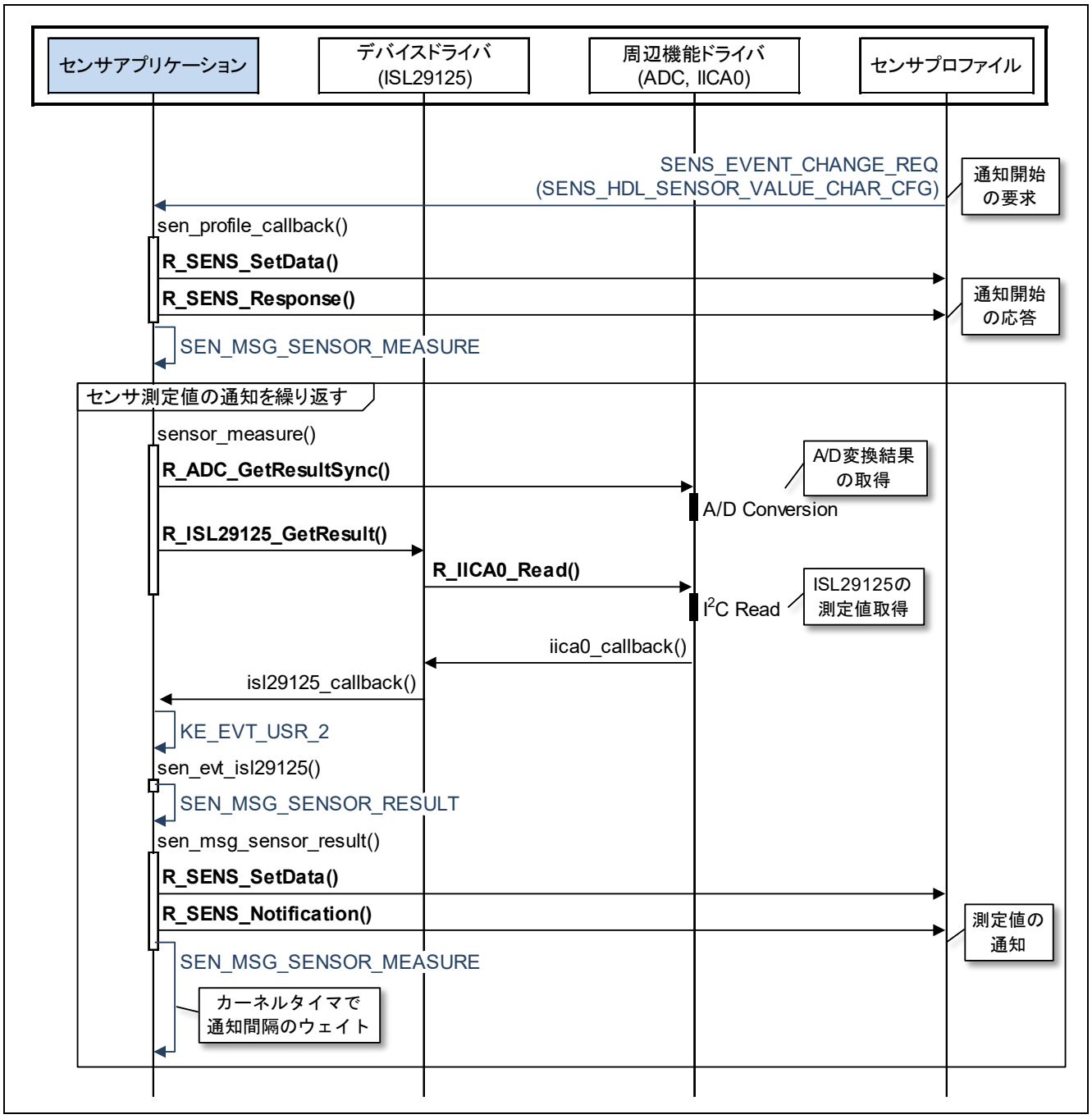


図 5-6　センサの測定値通知シーケンス

サンプルプログラムでは、RGB ライトセンサ ISL29125 のデバイスドライバが実装されています。

他の I²C デバイスを使用する場合は、対象デバイスを制御するためのデバイスドライバを実装し、シーケンス上の ISL29125 ドライバと置き換えてください。

5.5　センサプロファイルの停止

図 5-7 にセンサプロファイルの停止シーケンスを示します。

本シーケンスは、接続の切断時に実行されます。

センサアプリケーションはセンサプロファイルを停止し、センサプロファイルの停止完了を BLE アプリケーションに通知します。

BLE アプリケーションは、センサプロファイルの停止完了により、Advertising を再開します。

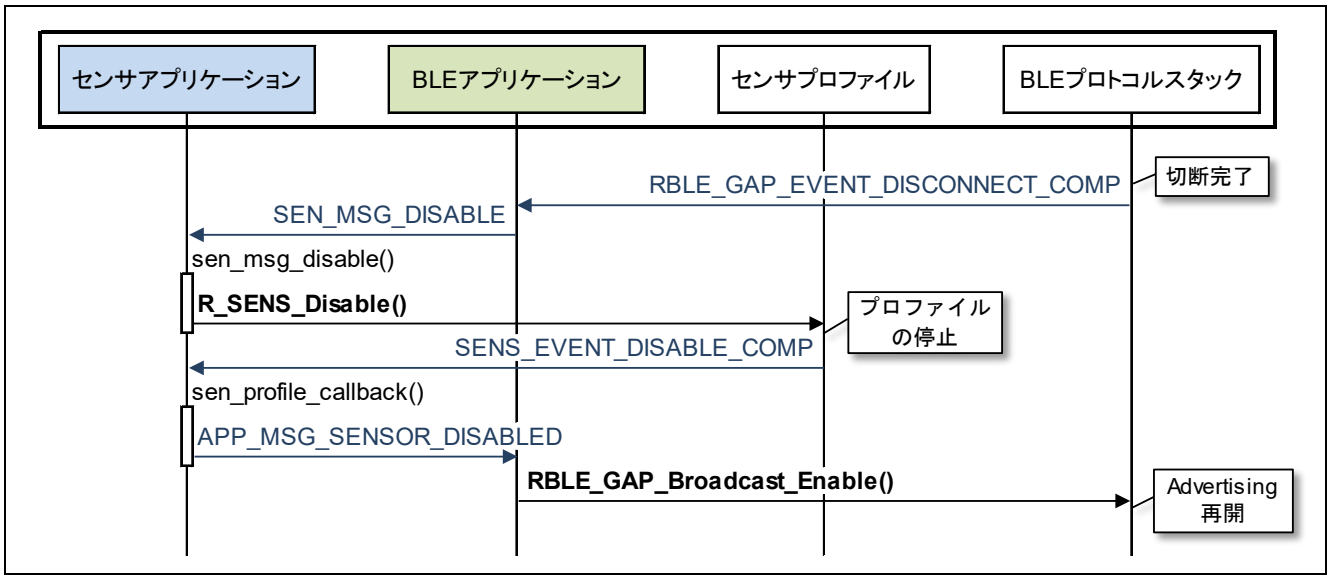


図 5-7　センサプロファイルの停止シーケンス

6. 関数仕様

本章では、サンプルプログラムに実装されたモジュールであるセンサプロファイル、デバイスドライバ、I²C ドライバの関数仕様を示します。

他の I²C デバイスを制御するためのデバイスドライバを実装する際に参照してください。

6.1 センサプロファイル

センサプロファイルの関数仕様を示します。

センサプロファイルのソースコードは下記のファイルを参照してください。

- センサプロファイル　：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);		
センサプロファイルサーバを有効化します。 接続の確立後に毎回実行してください。 本関数で登録したコールバック関数は、センサプロファイルサーバの各イベント発生時に実行されます。		
Parameters:		
	conhdl	コネクションハンドル RBLE_GAP_EVENT_CONNECTION_COMP イベントで通知された値を設定
	callback	イベント通知のためのコールバック関数 void (*SENS_EVENT_HANDLER)(SENS_EVENT *event);
		event：センサプロファイルサーバイベント SENS_EVENT 構造体の定義は sens.h を参照
Return:		
	RBLE_OK	正常終了
	上記以外	エラーコードの定義は rble.h の RBLE_STATUS_enum を参照

6.1.2 R_SENS_Disable

RBLE_STATUS **R_SENS_Disable**(uint16_t conhdl);		
センサプロファイルサーバを無効化します。 接続の切断後に毎回実行してください。		
Parameters:		
	conhdl	コネクションハンドル R_SENS_Enable()で設定した値を設定
Return:		
	RBLE_OK	正常終了
	上記以外	エラーコードの定義は rble.h の RBLE_STATUS_enum を参照

6.1.3 R_SENS_SetData

void **R_SENS_SetData**(uint16_t charhdl, void* charval);		
センササービスの各キャラクタリスティック値を変更します。		
Parameters:		
	charhdl	変更するキャラクタリスティック値のアトリビュートハンドル
	charval	変更後のキャラクタリスティック値
Return:		
	None	

6.1.4 R_SENS_Indication

void **R_SENS_Indication**(uint16_t charhdl);	
リモートデバイスに対してインディケーションを送信します。 リモートデバイスからインディケーション送信が許可された後、本関数を実行することができます。	
Parameters:	
charhdl	送信するキャラクタリスティック値のアトリビュートハンドル
Return:	
None	

6.1.5 R_SENS_Notification

void **R_SENS_Notification**(uint16_t charhdl);	
リモートデバイスに対してノーティフィケーションを送信します。 リモートデバイスからノーティフィケーション送信が許可された後、本関数を実行することができます。	
Parameters:	
charhdl	送信するキャラクタリスティック値のアトリビュートハンドル
Return:	
None	

6.1.6 R_SENS_Response

void **R_SENS_Response**(uint16_t charhdl, uint8_t status);	
リモートデバイスからのキャラクタリスティック値への書き込み要求に対する応答を送信します。 リモートデバイスからキャラクタリスティック値への書き込みを要求された時、本関数を実行してください。	
Parameters:	
charhdl	書き込み要求されたキャラクタリスティック値のアトリビュートハンドル
status	書き込み要求に対するステータス ステータスコードの定義は rble_api.h の RBLE_ATT_ERR_CODE_enum を参照
Return:	
None	

6.2 デバイスドライバ

サンプルプログラムには、RGB ライトセンサ ISL29125 を制御するためのデバイスドライバが実装されています。ISL29125 ドライバは I²C ドライバを使用し、I²C 通信で ISL29125 のレジスタにアクセスします。

他の I²C デバイスを使用する場合は、対象デバイスを制御するためのデバイスドライバを実装し、本 ISL29125 ドライバと置き換えてください。

デバイスドライバの関数仕様を以下に示します。ソースコードは下記のファイルを参照してください。

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

6.2.1 R_ISL29125_Init

uint8_t **R_ISL29125_Init**(r_isl29125_calback_t callback);

ISL29125 を初期化します。
制御対象のデバイスが I²C に接続されていることを確認し、デバイスリセットとコンフィグレーションを行います。
その他、キャリブレーションなどのデバイス仕様で定義された初期化シーケンスを実行します。
本関数で登録したコールバック関数は、非同期でのデバイス制御の完了後に割り込みハンドラから実行されます。

Parameters:

callback	非同期でのデバイス制御の完了通知のためのコールバック関数 void (*r_isl29125_calback_t) (r_isl29125_opcode_t opcode, uint8_t status, void* data);

opcode	実行したデバイス制御処理を識別するためのオペレーションコード
status	デバイス制御処理の完了ステータス 0　正常終了 上記以外　エラー終了
data	デバイス制御処理の結果データ

Return:

0	正常
上記以外	デバイスが I²C バス上に存在しない、デバイス異常、I²C 異常、その他のエラー

6.2.2 R_ISL29125_SetModeSync

uint8_t **R_ISL29125_SetModeSync**(uint8_t mode);	
ISL29125 の動作設定レジスタを設定します。 引数で指定されたモードに応じて、デバイスの測定状態(アクティブ、ランなど)または停止状態(スタンバイ、アイドルなど)といった動作モードを変更します。 その他、デバイスの仕様に応じて動作設定を実行します。 本関数はデバイス動作の設定完了後に返ります。また R_ISL29125_Init()で登録したコールバック関数は実行されません。	
Parameters:	
mode	デバイスの動作設定値
Return:	
0	正常
上記以外	デバイス異常、I²C 異常、その他のエラー

6.2.3 R_ISL29125_SetMode

uint8_t **R_ISL29125_SetMode**(uint8_t mode);	
ISL29125 の動作設定レジスタを設定します。 引数で指定されたモードに応じて、デバイスの動作モードを測定状態(アクティブ、ランなど)または停止状態(スタンバイ、アイドルなど)を変更します。 その他、デバイスの仕様に応じて動作設定を実行します。 本関数はデバイス動作の設定完了を待たずに返り、設定完了は R_ISL29125_Init()で登録したコールバック関数で通知されます。	
Parameters:	
mode	デバイスの動作設定値
Return:	
0	正常
上記以外	デバイス異常、I²C 異常、その他のエラー

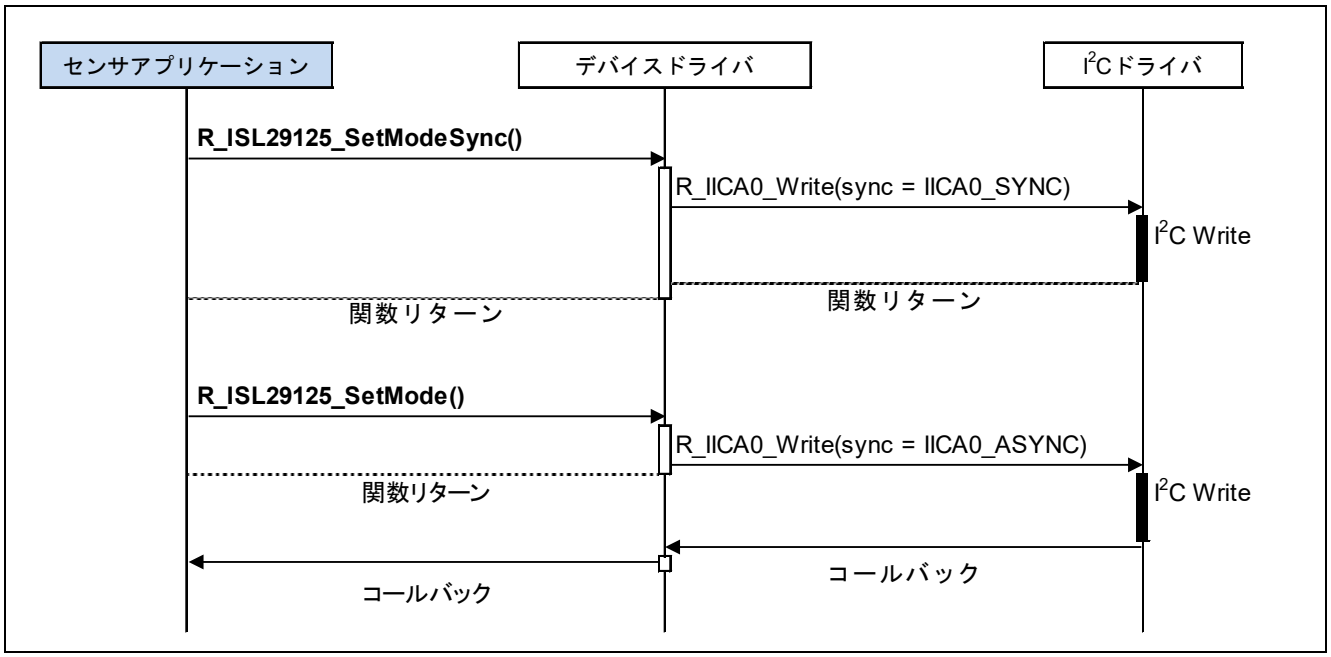


図 6-1　R_ISL29125_SetModeSync()と R_ISL29125_SetMode()のシーケンス

6.2.4 R_ISL29125_GetResultSync

uint8_t **R_ISL29125_GetResultSync**(r_isl29125_result_t* result);	
ISL29125 のセンサ測定値を取得します。 デバイスのセンサ測定値を取得し、引数で指定した変数に格納します。 本関数はデバイスのセンサ測定値取得の完了後に返ります。また R_ISL29125_Init()で登録したコールバック関数は実行されません。	
Parameters:	
result	デバイスのセンサ測定値
Return:	
0	正常
上記以外	デバイス異常、I²C 異常、その他のエラー

6.2.5 R_ISL29125_GetResult

uint8_t **R_ISL29125_GetResult**(void);	
ISL29125 のセンサ測定値を取得します。 本関数はデバイスのセンサ測定値取得を待たずに返り、取得結果は R_ISL29125_Init()で登録したコールバック関数で通知されます。	
Parameters:	
None	
Return:	
0	正常
上記以外	デバイス異常、I²C 異常、その他のエラー

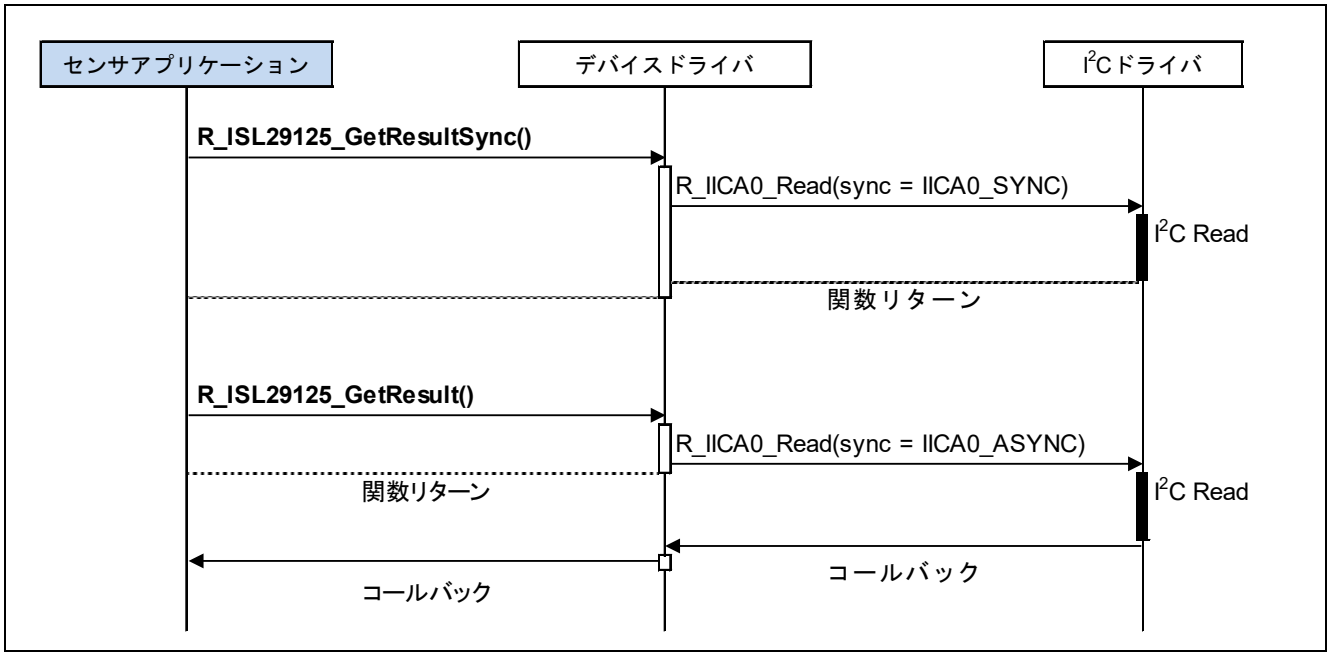


図 6-2　R_ISL29125_GetResultSync()と R_ISL29125_GetResult()のシーケンス

6.3 I²C ドライバ

サンプルプログラムは RL78/G1D のシリアル・インタフェース IICA を使用するための I²C ドライバが実装されています。本ドライバを使用すると、RL78/G1D が I²C マスタ、センサが I²C スレーブとして動作します。

I²C ドライバの関数仕様を以下に示します。ソースコードは下記のファイルを参照してください。

- I²C ドライバ　：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);			
RL78/G1D のシリアル・インタフェース IICA を初期化します。			
Parameters:			
	None		
Return:			
	None		

6.3.2 R_IICA0_RegisterCallback

void **R_IICA0_RegisterCallback**(iica0_user_calback_t callback);				
IICA0 の動作完了を通知するためのコールバック関数を登録します。 本関数で登録したコールバック関数は、以下のいずれかのタイミングで割り込みハンドラから実行されます。 - I²C 書き込み完了 - I²C 読み込み完了 - I²C エラー完了				
Parameters:				
	callback	I²C 完了通知のためのコールバック関数 void (*iica0_user_calback_t)(iica0_rw_calltype_t type, uint8_t flag);		
			type	IICA0_SENDEND　I²C 書き込み完了 IICA0_RECEIVEEND　I²C 読み込み完了 IICA0_ERROR　I²C エラー
			flag	r_cg_macrodriver.h を参照 MD_OK　正常 MD_OK 以外　エラー
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);			
I²C スレーブデバイスのレジスタにデータを書き込みます。 デバイスアドレス adr の buf[0]で指定したレジスタに、(len-1)byte のデータを buf[1]から書き込みます。 本関数は割り込み許可状態で実行してください。また割り込みハンドラからの本関数の実行はできません。			
Parameters:			
	adr	デバイスアドレス	7bit デバイスアドレスを設定
	buf	データバッファ	buf[0]にレジスタアドレス、buf[1]以降にデータを設定
	len	アクセス長(byte)	レジスタアドレス長(1)＋データ長を設定 (len >= 2)
	sync	同期設定 IICA0_SYNC IICA0_ASYNC	 I²C アクセス完了後に関数が返る I²C アクセス完了を待たずに関数が返る コールバック関数が I²C アクセス完了を通知する
Return:			
	r_cg_macrodriver.h を参照		
	MD_OK	正常	
	MD_OK 以外	エラー	

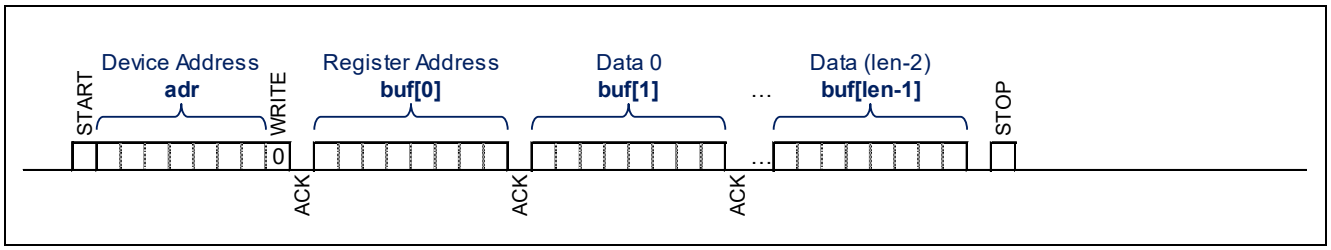


図 6-3　R_IICA0_Write の I²C アクセス

6.3.4 R_IICA0_Read

MD_STATUS **R_IICA0_Read**(uint8_t adr, void* buf, uint16_t len, iica0_rw_sync_t sync);			
I²C スレーブデバイスのレジスタからデータを読み込みます。 デバイスアドレス adr の buf[0]で指定したレジスタから、(len-1)byte のデータを buf[1]に読み込みます。 本関数は割り込み許可状態で実行してください。また割り込みハンドラからの本関数の実行はできません。			
Parameters:			
	adr	デバイスアドレス	7bit デバイスアドレスを設定
	buf	データバッファ	buf[0]にレジスタアドレスを設定
	len	アクセス長(byte)	レジスタアドレス長(1)＋データ長を設定 (len >= 2)
	sync	同期設定 IICA0_SYNC IICA0_ASYNC	 I²C アクセス完了後に関数が返る I²C アクセス完了を待たずに関数が返る コールバック関数が I²C アクセス完了を通知する
Return:			
	r_cg_macrodriver.h を参照		
	MD_OK	正常	
	MD_OK 以外	エラー	

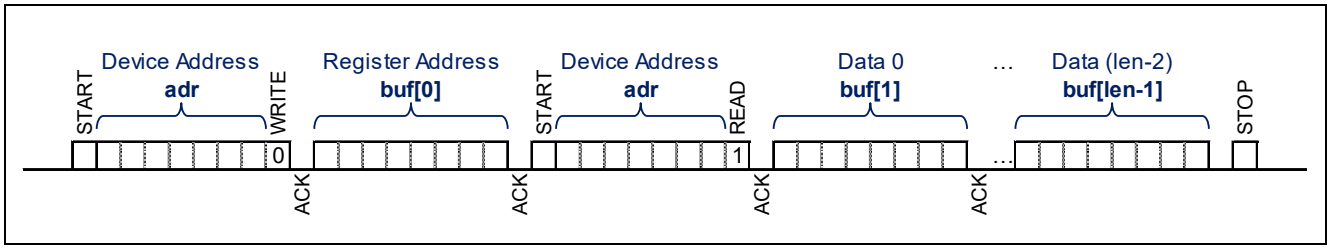


図 6-4　R_IICA0_Read の I²C アクセス

6.4 A/D コンバータドライバ

サンプルプログラムは RL78/G1D の A/D コンバータを使用するための A/D コンバータドライバが実装されています。A/D コンバータの設定は、統合開発環境 CS+ for CC のコード生成プラグインで変更することができます。

A/D コンバータドライバの関数仕様を以下に示します。ソースコードは下記のファイルを参照してください。

- A/D コンバータドライバ ：Project_Source/renesas/src/cg_src/r_cg_adc.c, r_cg_adc_user.c

6.4.1 R_ADC_Create

void **R_ADC_Create**(void);	
RL78/G1D の A/D コンバータを初期化します。	
Parameters:	
None	
Return:	
None	

6.4.2 R_ADC_GetChannel

uint8_t **R_ADC_GetChannel**(void);	
A/D コンバータで選択されたアナログ入力チャネルを取得します。	
Parameters:	
None	
Return:	
アナログ入力チャネル 返却されるアナログ入力チャネルの値は『RL78/G1D ユーザーズマニュアル ハードウェア編』(R01UH0515)の 12.3.7 項「アナログ入力チャネル指定レジスタ（ADS）」を参照。	

6.4.3 R_ADC_GetResultSync

uint8_t **R_ADC_GetResultSync**(uint16_t* result);	
A/D コンバータの変換を実行し、結果を引数 result で返却します。	
Parameters:	
result	A/D 変換結果
Return:	
0	正常終了
上記以外	エラー終了

7. Appendix

7.1 デバッグ用 UART

サンプルプログラムにはデバッグ向けに、UART からメッセージを出力する関数が実装されています。

表 7-1 にデバッグ用 UART 関数を示します。必要に応じて下記の関数を使用してください。

表 7-1　デバッグ用 UART 関数

関数名	用途例
PrintError	実装に問題があるエラーを送信
PrintWarning	想定しない動作の警告を送信
PrintInfo	パラメータ確認のための情報を送信
PrintLog	動作シーケンスのログを送信

デバッグ用 UART 関数を有効化するには、下記ファイルの CONSOLE_LVL マクロの値を変更します。

Project_Source/rBLE/src/sample_app/console.h

console.h (line 63)

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

1~4 のいずれかの値に変更

CONSOLE_LVL マクロの値により、**表 7-2** が示すデバッグ用 UART 関数のみ有効となります。

表 7-2　CONSOLE_LVL

CONSOLE_LVL	有効となるデバッグ用 UART 送信関数
0	デバッグ用 UART は無効
1	PrintError のみ
2	PrintError、PrintWarning
3	PrintError、PrintWarning、PrintInfo
4	PrintError、PrintWarning、PrintInfo、PrintLog

デバッグ用 UART が送信するメッセージは、PC のターミナルソフトで確認します。**表 7-3** にターミナルソフトのシリアルポート設定を示します。

表 7-3　ターミナルソフトのシリアルポート設定

設定項目		設定値
シリアルポート	ポート	USB シリアルポート ※COM 番号は評価ボードごとに異なる
	ボーレート	1,000,000bps
	データ長	8bit
	パリティ	None
	ストップビット	1bit
	フロー制御	None
改行コード	受信	LF
端末サイズ	横幅	128 文字以上

※ターミナルソフトとして Tera Term をご使用の場合、「ボー・レート」のドロップダウンリストに 1,000,000bps は含まれません。「ボー・レート」欄に直接"1000000"と入力してください。

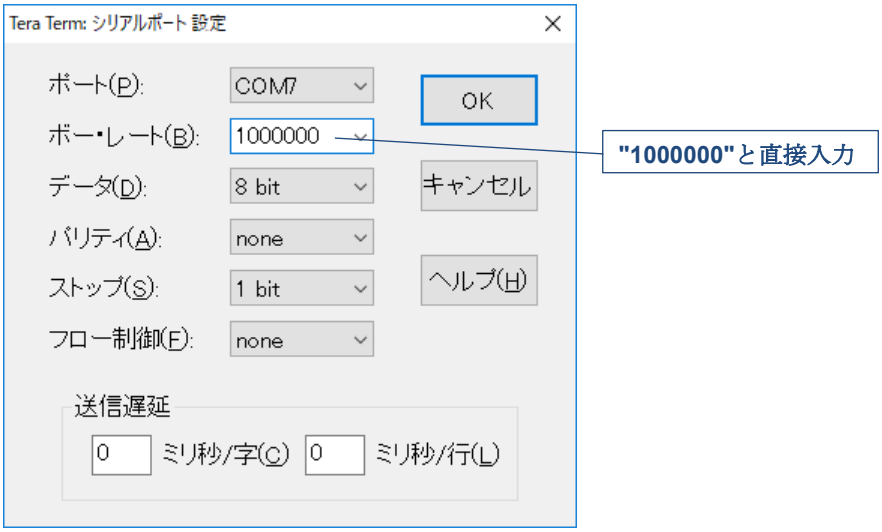


図 7-1 に CONSOLE_LVL=4 でサンプルプログラムを実行した場合のメッセージ例を示します。

メッセージの追加や削除、内容の変更は必要に応じてカスタイマイズできます。

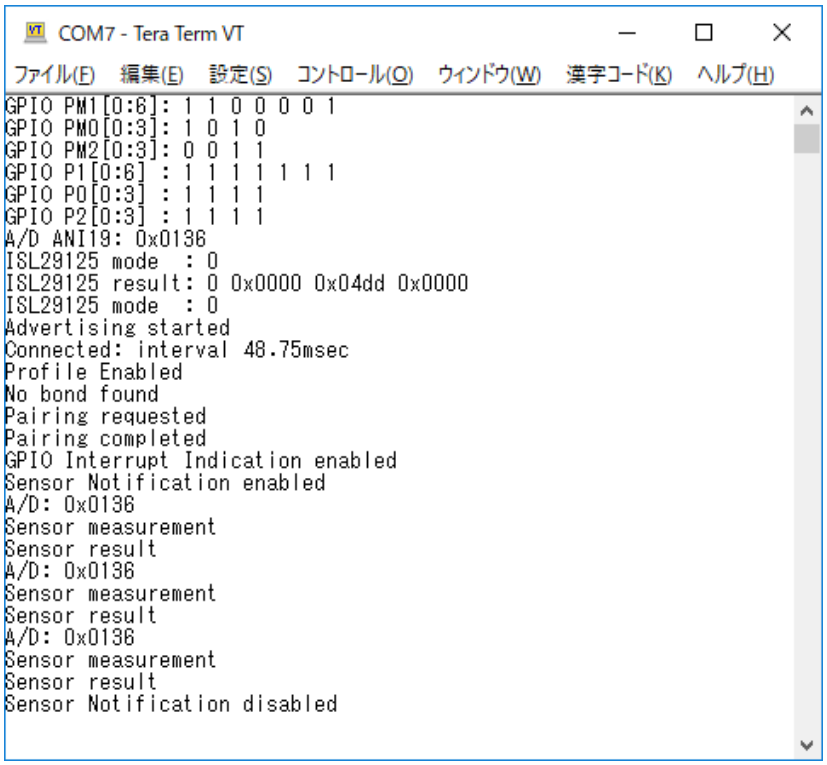


図 7-1　デバッグ用 UART のメッセージ例

本メッセージ例では、"Connected"が接続の確立、"Pairing completed"がペアリングの完了、"Sensor Notification Enabled"がセンサ測定値の通知開始を示します。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
https://www.renesas.com/

お問合せ先
https://www.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容
1.00	2018.06.15	新規発行
1.01	2018.06.26	1.概要：想定するユースケースを削除
		4.1.ファイル構成：Android アプリ"BleSensor"のプロジェクトとアプリケーションノートを同梱
1.02	2018.11.5	4.1.ファイル構成：Android アプリ"BleSensor"のプロジェクトとアプリケーションノートのバージョンを更新
1.03	2018.12.21	3.5.アプリのインストール：手順 6 を追加 4.1.ファイル構成：Android アプリ"BleSensor"のプロジェクトとアプリケーションノートのバージョンを更新

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：　輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注1.　本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注2.　本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1　2017.11)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口　http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/