RX23W グループ

Bluetooth Mesh ネットワークによる温湿度センサデータ通信サンプルコード

要旨

本アプリケーションノートでは、RX23W 用 Bluetooth® Mesh モジュールに含まれる Sensor Model を利用した Mesh 温湿度センサデータ通信デモ（以下、Mesh センサデモと略します）について説明します。

Bluetooth Low Energy 無線通信の Mesh ネットワーク(MtoMt 通信)を利用することで、従来の PtoP 通信よりも遙かに遠い距離までデータを送ったり、ノード追加でネットワークを容易に拡張したり、また、通信障害に強いネットワークを構築できます。

なお、「RX23W グループ　Bluetooth Mesh スタック　開発者ガイド(R01AN4875)」を参考にし、本アプリケーションノートとプログラムを作成しています。

動作確認デバイス

- RX23W グループ
- 温湿度センサ
  - ルネサス エレクトロニクス製 HS300x 相対湿度／温度センサ（以下、HS300x センサと略します）

動作確認ボード

- Target Board for RX23W（以下、TB-RX23W ボードと略します）
- HS3001 センサボード(※)：Relative Humidity Sensor Pmod™ Board (US082-HS3001EVZ)
- 変換ボード(※)：Interposer Board to convert Type2/3 to Type 6A PMOD standard (US082-INTERPEVZ)

※　入手方法は、弊社にお問い合わせ願います。
　　また、HS3001 センサボードが未接続の場合でも、疑似データ送信が可能なため Sensor Model を使った通信を確認できます。

関連文書

本アプリケーションノートは、以下の文書を参照し、説明しています。また、文書番号の下 6 桁を省略して記載します。文書更新の場合に章構成等が変わる場合があります。参照の際に注意してください。

文書名	文書番号
RX Family Renesas Sensor Control Modules Firmware Integration Technology	R01AN5892EJ0110
RX Family Renesas HS300x Sensor Control Module Firmware Integration Technology	R01AN5893EJ0110
RX Family Renesas Sensor I2C Communication Middleware Control Module Firmware Integration Technology	R01AN5895EJ0110
RX23W グループ Bluetooth Mesh スタック　スタートアップガイド	R01AN4874JJ0120
RX23W グループ Bluetooth Mesh スタック　開発ガイド	R01AN4875JJ0120
RX23W グループ Bluetooth Mesh モジュール Firmware Integration Technology	R01AN4930JJ0120
RX23W グループ BLE モジュール Firmware Integration Technology	R01AN4860JJ0230

目次

1.　概要

1.1　システム概要

Mesh ネットワークで使用するノードは全て RX23W を使用します。Sensor Server ノード、Sensor Client ノード、Relay 機能を持つノード、Friend 機能を持つノードで構成された Bluetooth Mesh ネットワークを構成し、Sensor Server ノードに接続された HS300x センサの温湿度データを Sensor Client ノードに送信します。さらに Sensor Client ノードから Sensor Server ノードに対して、センサデータの送信周期（デフォルト 2 秒⇔5 秒）の変更要求ができます。

また、Sensor Server ノードは Low Power 機能を持たせ、低消費電力化を実現しています。

スマートフォン向け Mesh ノード設定用スマートフォンアプリケーションを使って、全ノードのプロビジョニングとコンフィグレーションが可能です。

1.1.1　Mesh ネットワーク構成とデータの流れ

(1)　Mesh センサデモのネットワーク構成

Mesh センサデモで使用するノード一覧を表 1-1 に、その Mesh ネットワーク構成を図 1-1 に示します。

各ノードのエレメント／モデルの構成については、「4.2 Mesh Model」を参照してください。

表 1-1　ノード一覧

ノード	内容
Sensor Client	Sensor Client Model として動作するノード 以下の動作が可能 • センサデータの受信 • センサデータの送信周期変更要求値の送信
Sensor Server _Low Power	Sensor Server Model として動作し、オプション機能の Low Power 機能を持つノード 説明上、Sensor Server ノードもしくは Low Power ノードと略す場合があります。 以下の動作が可能 • 接続されたセンサデバイスからセンサデータの取得と送信 • センサデータの送信周期変更要求値の受信と送信周期の切り替え
Friend_Relay	オプション機能の Friend 機能と Relay 機能を持つノード 説明上、Friend ノードと略す場合があります。
Relay	オプション機能の Relay 機能を持つノード

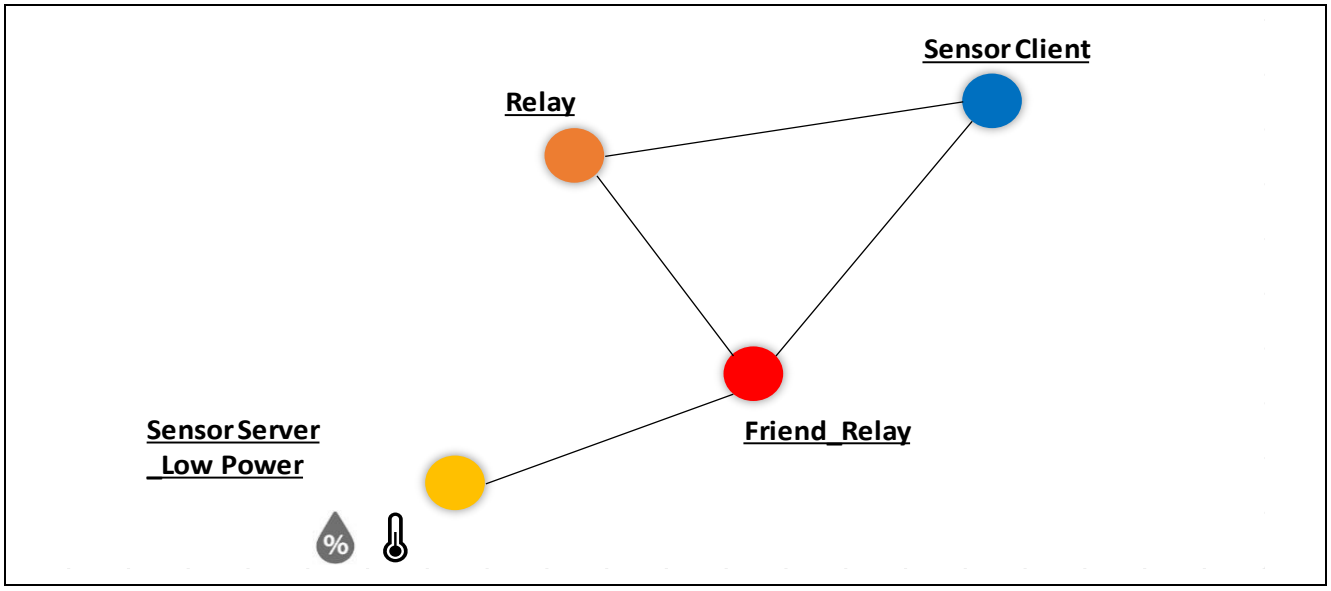

図 1-1　Mesh センサデモの Mesh ネットワーク構成

(2)　Sensor Server ノードから Sensor Client ノードへのメッセージ（センサデータ）の流れ

図 1-2 に Sensor Server が測定したデータを Sensor Client で受信する際のセンサデータの流れを示します。Sensor Client ノードが Sensor Server ノードの無線到達範囲外であっても、Sensor Client ノードは Relay 機能を持つノード経由でセンサデータを受け取ることができます。

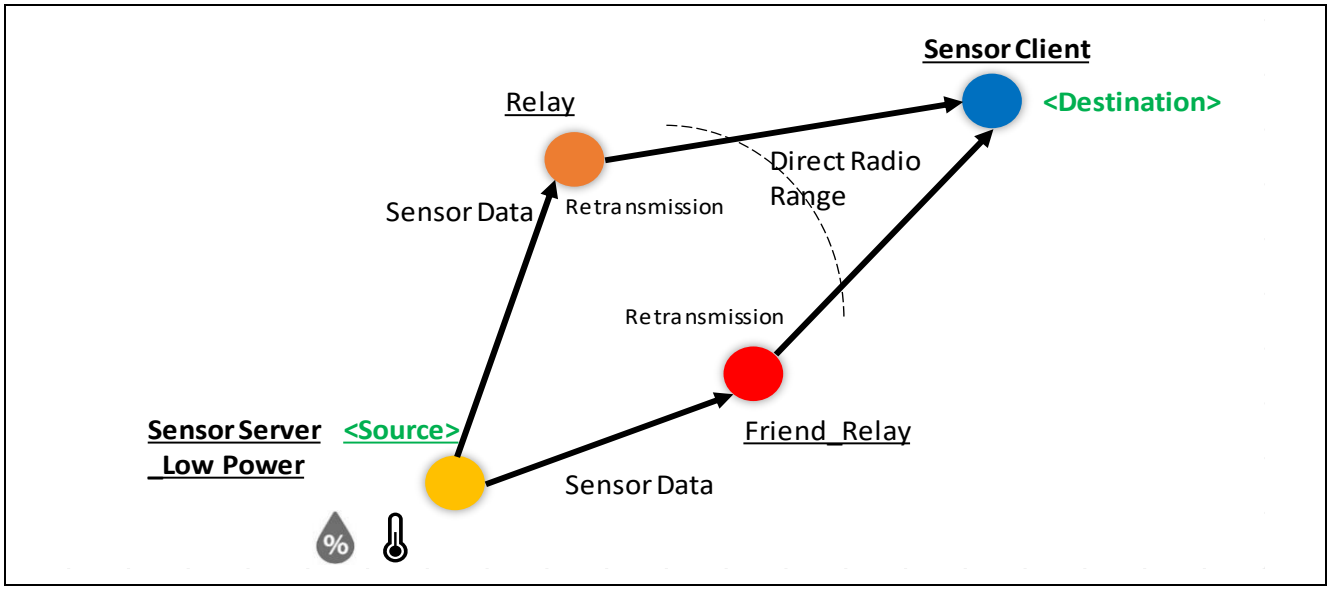

図 1-2　Sensor Client ノードが Sensor Server ノードの無線到達範囲外の場合のデータの流れ

(3)　Sensor Client ノードから Sensor Server ノードへのメッセージ（センサデータの送信周期変更要求）の流れ

Low Power 機能を持つ Sensor Server（Sensor Server_Low Power）ノードは、Friend ノードとフレンドシップ確立により Low Power 機能が稼働します。そのため、Sensor Client ノードからのメッセージは Friend ノードに保管されます。Sensor Server ノードは Scan 再開時に Friend ノードから保管メッセージを受け取ることができます。

図 1-3 にメッセージ（センサデータの送信周期変更要求値）の流れを示します。

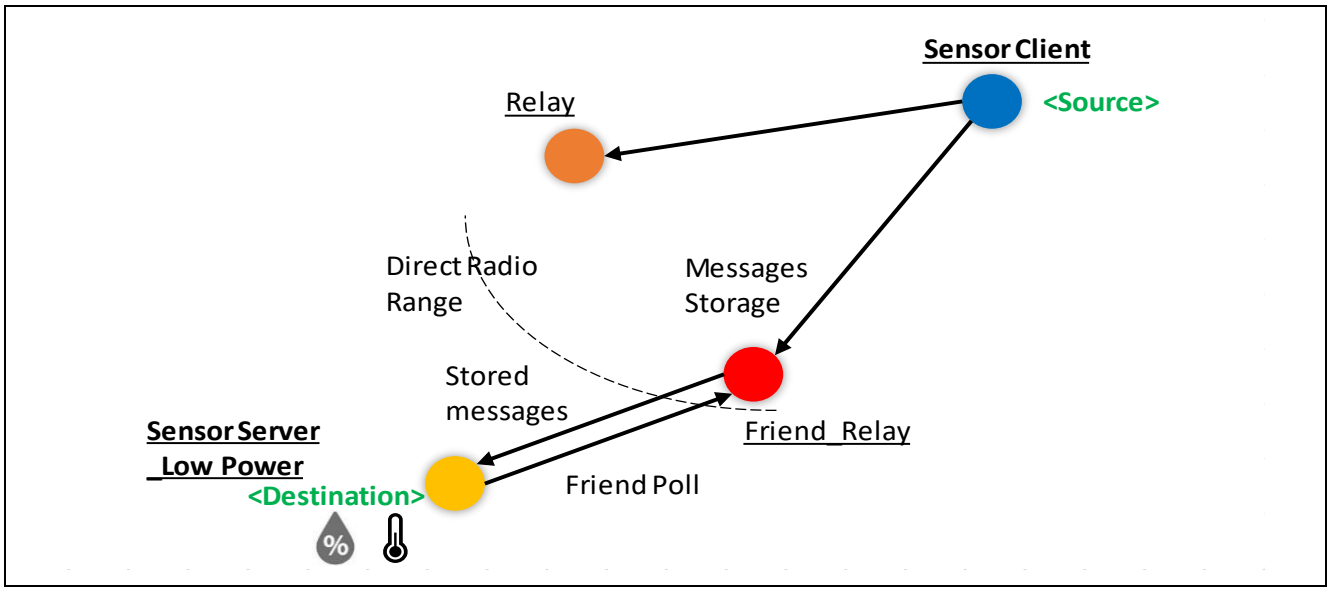

図 1-3　Sensor Client ノードからのメッセージの流れ

1.1.2　Mesh センサデモパッケージ内の MOT ファイルと e2 studio プロジェクトについて

以下にパッケージ内の MOT ファイルと e2 studio プロジェクトの概要を示します。

(1)　MOT ファイル

パッケージに、表 1-2 に示すプロジェクトの MOT ファイルを格納しています。

MOT ファイルを TB-RX23W ボードに書き込むことで、直ぐにデモが可能です。

書き込み方法は、「2.1 Renesas Flash Programmer を使った書き込み」を参照してください。

また、MOT ファイル作成時のプロジェクト設定については、「1.3 動作確認環境」を参照してください。

表 1-2　MOT ファイル一覧

MOT ファイル名	内容【注 1】
tbrx23w_mesh_**sensor**_server.mot	TB-RX23W ボード用 Sensor Server Model
tbrx23w_mesh_**sensor**_client.mot	TB-RX23W ボード用 Sensor Client Model
tbrx23w_mesh_**sensor**_friend.mot 【注 2】	TB-RX23W ボード用 Friend_Relay ノード TB-RX23W ボード用 Relay ノード

注 1：格納先フォルダ名は、共に「FITDemos\ROM_Files」です。
注 2：Friend_Relay ノードおよび Relay ノード用 MOT ファイルは同じです。

(2)　e2 studio プロジェクト

パッケージに、表 1-3 に示す Sensor Model プロジェクトを格納しています。

e2 studio プロジェクトをインポートし TB-RX23W ボードに書き込むことで、デモが可能です。

また、e2 studio プロジェクトの設定については、「1.3 動作確認環境」を参照してください。

表 1-3　Sensor Model プロジェクト一覧

プロジェクト名	内容および格納先フォルダ名
tbrx23w_mesh_**sensor**_server	TB-RX23W ボード用 Sensor Server Model プロジェクト フォルダ名：tbrx23w_mesh_**sensor**_server
tbrx23w_mesh_**sensor**_client	TB-RX23W ボード用 Sensor Client Model プロジェクト フォルダ名：tbrx23w_mesh_**sensor**_client
tbrx23w_mesh_**sensor**_friend	TB-RX23W ボード用 Friend_Relay ノード、Relay ノードプロジェクト フォルダ名：tbrx23w_mesh_**sensor_friend**

注：各プロジェクトの BLE FIT モジュールは Ver.2.30 を使用しています。

上記プロジェクトは Bluetooth Mesh スタックパッケージ(R01AN4930)に含まれる tbrx23w_mesh_server プロジェクトおよび tbrx23w_mesh_client プロジェクトをベースに作成したプロジェクトです。

プロジェクトの以下の差分については、「4.6 プログラム差分」を参照ください。

1.2　フォルダ／ファイル構成

1.2.1　フォルダ構成

以下にフォルダ／ファイル構成を示します。一部のフォルダとファイルの記載を省略しています。

(1)　r01an6129xx0101-rx23w-blemesh-sensor.zip

```
r01an6129xx0101-rx23w-blemesh-sensor
├──FITDemos
│   ├──ROM_Files
│   │   ├──tbrx23w_mesh_sensor_client.mot   MOT File for Sensor Client Node
│   │   ├──tbrx23w_mesh_sensor_friend.mot   MOT File for Friend_Relay Node
│   │   └──tbrx23w_mesh_sensor_server.mot   MOT File for Sensor Server Node
│   ├──tbrx23w_mesh_sensor_client.zip       Project Zip file for Sensor Client Node
│   ├──tbrx23w_mesh_sensor_friend.zip       Project Zip file for Friend_Relay Node
│   └──tbrx23w_mesh_sensor_server.zip       Project Zip file for Sensor Server Node
├──r01an6129ej0101-rx23w-blemesh-sensor.pdf
└──r01an6129jj0101-rx23w-blemesh-sensor.pdf
```

図 1-4　パッケージのフォルダ／ファイル構成

(2)　tbrx23w_mesh_sensor_client.zip

```
tbrx23w_mesh_sensor_client
├─.cproject
├─.project
├─.settings
├─src
│   ├─main.c
│   ├─mesh_appl.h
│   ├─mesh_core.c
│   ├─mesh_model.c
│   ├─smc_gen
│   │   ├─general
│   │   ├─r_ble_rx23w
│   │   ├─r_bsp
│   │   ├─r_byteq
│   │   ├─r_cmt_rx
│   │   ├─r_config
│   │   ├─r_flash_rx
│   │   ├─r_gpio_rx
│   │   ├─r_irq_rx
│   │   ├─r_lpc_rx
│   │   ├─r_mesh_rx23w
│   │   ├─r_pincfg
│   │   └─r_sci_rx
│   └─vendor_model
├─tbrx23w_mesh_client HardwareDebug.launch
├─tbrx23w_mesh_client.rcpc
└─tbrx23w_mesh_client.scfg
```

図 1-5　tbrx23w_mesh_sensor_client.zip のフォルダ／ファイル構成

(3) tbrx23w_mesh_sensor_friend.zip

```
tbrx23w_mesh_sensor_friend
├──.cproject
├──.project
├──.settings
├──src
│   ├──main.c
│   ├──mesh_appl.h
│   ├──mesh_core.c
│   ├──mesh_model.c
│   ├──smc_gen
│   │   ├──general
│   │   ├──r_ble_rx23w
│   │   ├──r_bsp
│   │   ├──r_byteq
│   │   ├──r_cmt_rx
│   │   ├──r_config
│   │   ├──r_flash_rx
│   │   ├──r_gpio_rx
│   │   ├──r_irq_rx
│   │   ├──r_lpc_rx
│   │   ├──r_mesh_rx23w
│   │   ├──r_pincfg
│   │   └──r_sci_rx
│   └──vendor_model
│       ├──vendor_api.h
│       ├──vendor_client.c
│       └──vendor_server.c
├──tbrx23w_mesh_client HardwareDebug.launch
├──tbrx23w_mesh_client.rcpc
└──tbrx23w_mesh_client.scfg
```

図 1-6　tbrx23w_mesh_sensor_friend.zip のフォルダ／ファイル構成

(4) tbrx23w_mesh_sensor_server.zip

```
tbrx23w_mesh_sensor_server
├──.cproject
├──.project
├──.settings
├──src
│   ├──hs300x
│   │   ├──RX_HS300X.c
│   │   └──RX_HS300X.h
│   ├──main.c
│   ├──mesh_appl.h
│   ├──mesh_core.c
│   ├──mesh_model.c
│   ├──smc_gen
│   │   ├──general
│   │   ├──r_ble_rx23w
│   │   ├──r_bsp
│   │   ├──r_byteq
│   │   ├──r_cmt_rx
│   │   ├──r_comms_i2c_rx    : I2C COMM Module
│   │   ├──r_config
│   │   ├──r_flash_rx
│   │   ├──r_gpio_rx
│   │   ├──r_hs300x_rx       : HS300x Module
│   │   ├──r_irq_rx
│   │   ├──r_lpc_rx
│   │   ├──r_mesh_rx23w
│   │   ├──r_pincfg
│   │   ├──r_riic_rx         : RIIC Module (Not used)
│   │   ├──r_sci_iic_rx      : SCI Simple I2C Mode
│   │   └──r_sci_rx
│   └──vendor_model
│       ├──vendor_api.h
│       ├──vendor_client.c
│       └──vendor_server.c
├──tbrx23w_mesh_sensor_server.x.launch
├──tbrx23w_mesh_server HardwareDebug.launch
├──tbrx23w_mesh_server.rcpc
└──tbrx23w_mesh_server.scfg
```

図 1-7　tbrx23w_mesh_sensor_server.zip のフォルダ／ファイル構成

1.2.2　Mesh センサデモプロジェクトで使用する主要 FIT モジュール

図 1-8 に、Sensor Client プロジェクトと Sensor Server プロジェクトのコンポーネント設定を示します。

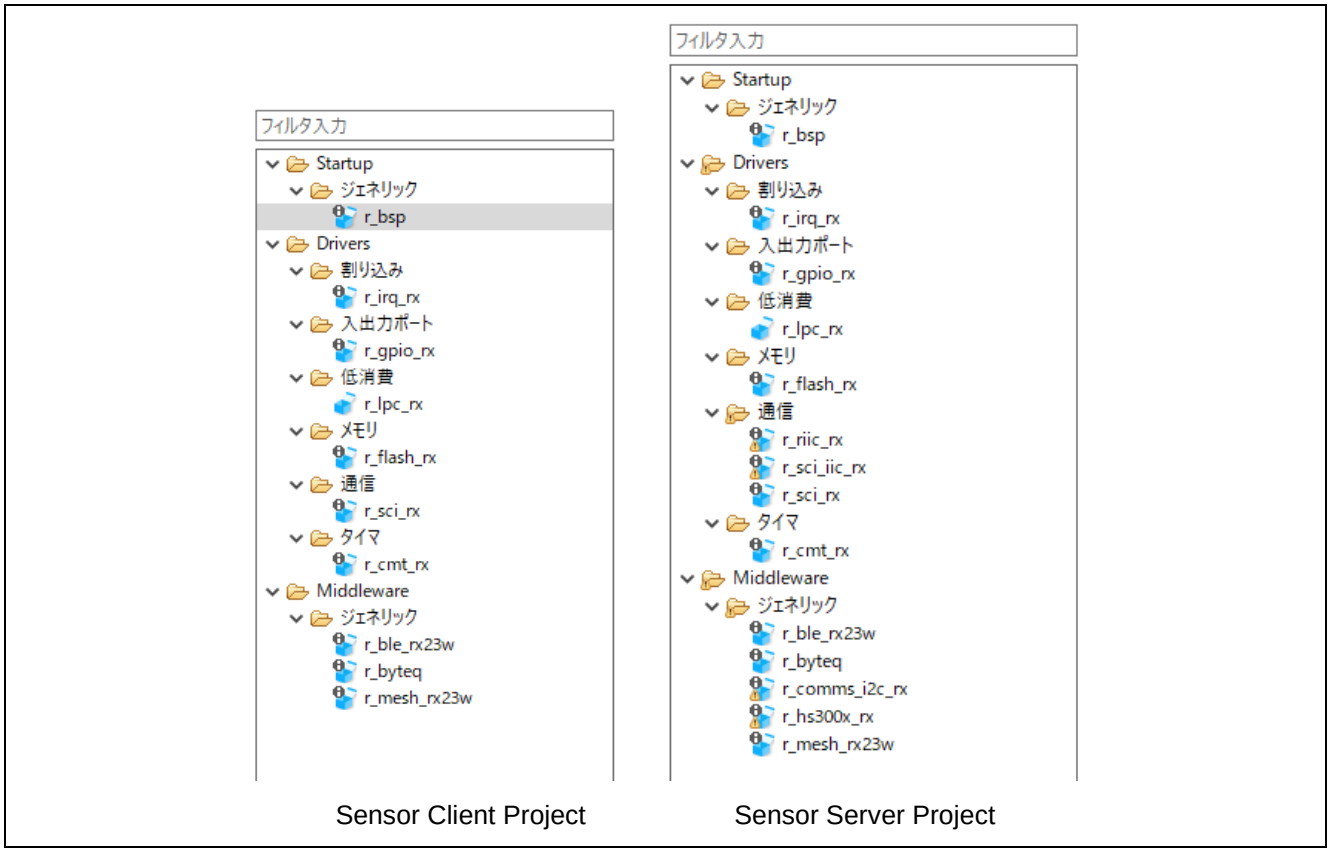

図 1-8　プロジェクトのコンポーネント設定

表 1-4 に使用する主要 FIT モジュールを示します。

FIT モジュール使用時に必要となる他 FIT モジュール（例：BSP）については、各 FIT モジュールのドキュメントを参照してください。

また、FIT モジュールのソフトウェアレイヤ構成については、「4.1 ソフトウェア構成」を参照してください。「1.3 動作確認環境」も参照してください。

表 1-4　使用する主要 FIT モジュール

モジュール名	モジュール Ver.	内容
Bluetooth Mesh モジュール（Mesh FIT モジュール）	Ver.1.20	Bluetooth Mesh スタック層と Bluetooth ベアラーラッパー層を含むモジュールです。 Bluetooth Mesh スタックパッケージ(R01AN4930xx0120)に、tbrx23w_mesh_server プロジェクトおよび tbrx23w_mesh_client プロジェクト等のデモプロジェクトやスマートフォンアプリケーションが含まれています。
BLE モジュール（BLE FIT モジュール）	Ver.2.30	Bluetooth Low Energy プロトコルスタック層を含むモジュールです。
HS300x モジュール	Ver.1.10	I2C バス上に接続された HS300x センサのデータを取得するためのモジュールです。 本モジュールは I2C COMMS モジュールを介して、I2C モジュールを制御します。
I2C COMMS モジュール	Ver.1.10	HS300x モジュールと I2C モジュールを接続するモジュールです。
I2C モジュール	Ver.2.49	I2C バス制御用モジュールです。RIIC モジュールもしくは SCI Simple I2C Mode モジュールが対象です。 Mesh センサデモでは SCI Simple I2C Mode モジュールを使用します。 RIIC モジュールを含みますが、センサを接続するコネクタの都合上、RIIC モジュールを使用しません。

1.2.3　スマートフォンアプリケーション「Renesas Bluetooth Mesh Mobile」

Bluetooth Mesh スタックパッケージ(R01AN4930)に含まれる「Renesas Bluetooth Mesh Mobile」（以下、「Mesh Mobile」と略します）を使用します。

また、「1.3 動作確認環境」も参照してください。

Mesh Mobile はデモ用サンプルアプリケーションです。各ノードのプロビジョニングとコンフィグレーションが可能です。

1.3　動作確認環境

(1)　動作確認条件

本 Mesh センサデモは以下に示す条件で動作を確認しています。

表 1-5　動作確認条件

項目	概要
MCU	RX23W
MCU ボード	Target Board for RX23W（TB-RX23W ボード）
センサ	HS3001
センサボード	Relative Humidity Sensor Pmod Board (US082-HS3001EVZ)
I/F 変換ボード	Interposer Board to convert Type2/3 to Type 6A PMOD standard (US082-INTERPEVZ) MCU ボード上の PMOD コネクタとセンサボードを接続するための変換ボード
IDE	e2 studio 2021-07 もしくはそれ以降
Toolchain	CC-RX V2.08.01
ファームウェア書き込みツール	Renesas Flash Programmer V3.08.03 もしくはそれ以降
スマートフォン	Android™ OS 8 もしくはそれ以降を搭載した端末
Mesh Mobile アプリケーション	Renesas Bluetooth Mesh Mobile Bluetooth Mesh スタックパッケージ(R01AN4930)に含まれています。 R01AN4930xx0120 に含まれるもので動作を確認済です。

(2)　ノードの機能設定

「図 1-1　Mesh センサデモの Mesh ネットワーク」に示す 4 種類のノードを使用します。

表 1-6 に各ノードの機能設定を示します。

ボード台数が少ない場合のデモについては、「3.2 ハードウェア環境」を参照してください。

表 1-6　ノードの機能の設定一覧

ノード	機能
Sensor Client ノード	Sensor Client 機能
Sensor Server_Low Power ノード	Sensor Server 機能 Low Power 機能を有効化設定
Friend_Relay ノード【注 1】	Friend 機能と Relay 機能を有効化設定
Relay ノード【注 1】	Relay 機能を有効化設定

注：上記以外にノードに必須の機能があります。「4.2 Mesh Model」も参照してください。
注 1：後述のコンフィグレーションにより、Friend_Relay ノードもしくは Relay ノードとして稼働させます。

1.4　コードサイズ

表 1-7 にコードサイズを示します。

表 1-7　コードサイズ一覧

プロジェクト	全体	BLE セクションと MESH セクションの合計
tbrx23w_mesh_**sensor**_server	ROM：315KB RAM：45KB	ROM：227KB RAM：21KB
tbrx23w_mesh_**sensor**_client	ROM：298KB RAM：44KB	ROM：227KB RAM：21KB

2.　ファームウェア書き込み

以下に、Renesas Flash Programmer を使った書き込み方法を示します。

2.1　Renesas Flash Programmer を使った書き込み手順

Renesas Flash Programmer を使って、MOT ファイルをボード上の RX23W に書き込むことができます。同梱の MOT ファイルについては、「表 1-2　MOT ファイル一覧」を参照してください。

Renesas Flash Programmer V3.08.03 もしくはそれ以降を以下から入手してください。

https://www.renesas.com/software-tool/renesas-flash-programmer-programming-gui

以下に手順を示します。

(1)　TB-RX23W ボードへの書き込み時の ESW1 スイッチ設定

以下を設定してください。

- ファームウェアの書き込みやデバッグの場合：ESW1 の 2-4 を ON に設定

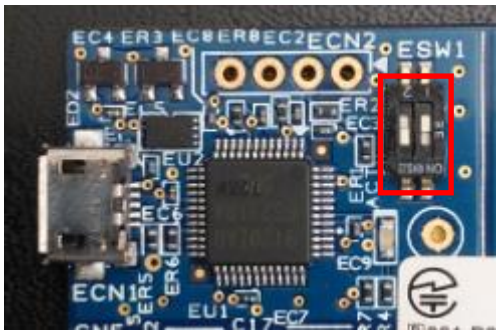

図 2-1　TB-RX23W ボードの書き込み時の ESW1 スイッチ設定

(2)　PC と TB-RX23W ボードの接続

図 2-2 に示すように、PC とエミュレータ用コネクタ(ECN1)を USB ケーブルで接続してください。

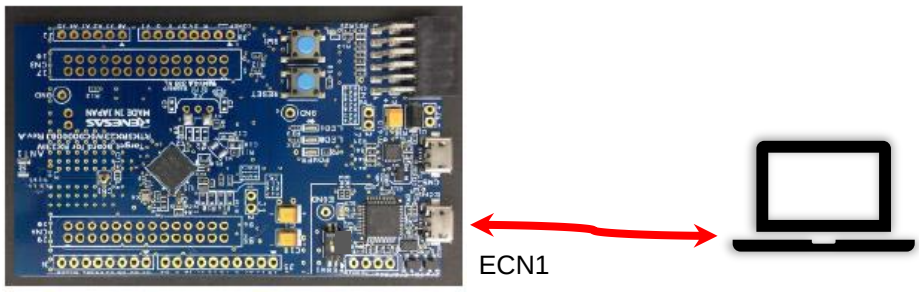

図 2-2　PC と TB-RX23W ボードとの接続

(3)　Renesas Flash Programmer の起動と接続処理

Renesas Flash Programmer を起動させてください。

Renesas Flash Programmer プロジェクトの有無により手順が異なります。以下に示す 2 通りについて説明します。

(a) Renesas Flash Programmer プロジェクトが無い場合
(b) 既に Renesas Flash Programmer プロジェクト作成済の場合

(a)　Renesas Flash Programmer プロジェクトが無い場合

1.　新規プロジェクト作成

「ファイル」の「新しいプロジェクトを作成(N)…」をクリックしてください。

以下を設定してください。

- 「プロジェクト情報」の「マイクロコントローラ(M)」：RX200
- 「通信」の「ツール(T)」：E2 emulator Lite（TB-RX23W ボードのオンボードエミュレータの指定）
- 「通信」の「インタフェース(I)」：FINE

「通信」の「電源：供給しない」を確認してください。

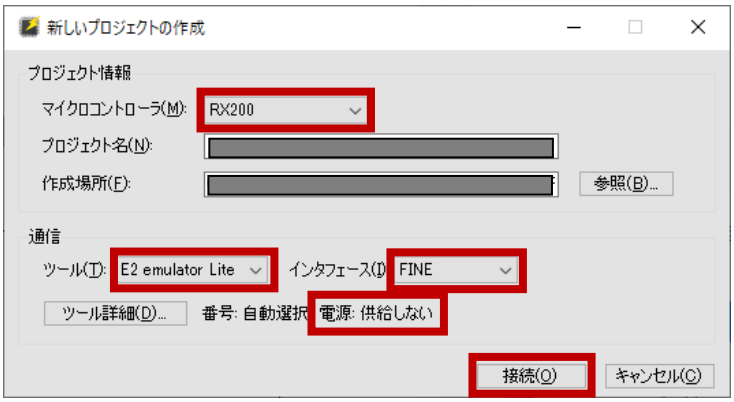

図 2-3　「プロジェクト情報」と「通信」の設定

2.　接続処理

図 2-3 に示す「接続」をクリックしてください。Renesas Flash Programmer は接続処理を開始します。

ログ出力ウィンドウに、以下が表示されることを確認してください。

```
ツールから切断します。
操作が成功しました。
```

図 2-4　正常接続時のログ出力ウィンドウ表示

(b)　既に Renesas Flash Programmer プロジェクト作成済の場合

1.　プロジェクトのオープン

「ファイル」の「プロジェクトを開く(O)…」をクリックし、プロジェクトファイルを選択してください。

「接続設定」タブをクリックし、「通信」の以下の設定を確認してください。

- 「ツール(T)」設定：E2 emulator Lite（TB-RX23W ボードのオンボードエミュレータの指定）
- 「インタフェース(I)」設定：FINE
- 「電源：供給しない」

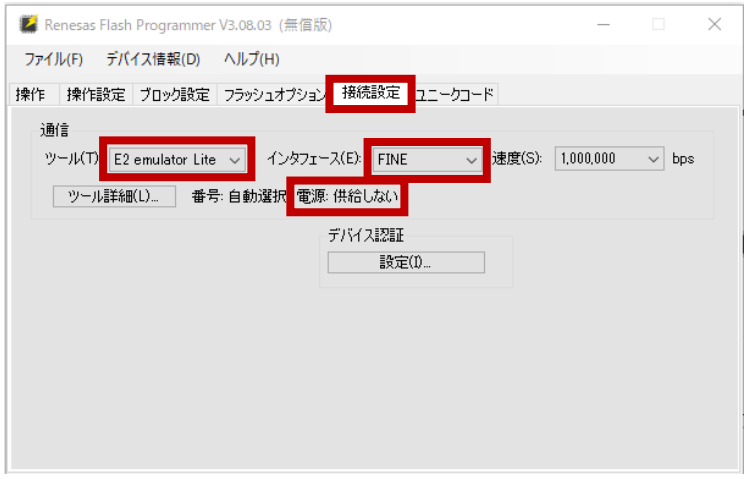

図 2-5　「接続設定」タブの設定

(4)　TB-RX23W ボードへの書き込み

Renesas Flash Programmer の操作手順にしたがって、ファームウェアを書き込んでください。

(5)　TB-RX23W ボードへの書き込み後の ESW1 スイッチ設定

デモ実行のために、以下を設定してください。

- オンボードエミュレータを使用せずファームウェアを実行する場合：ESW1 の 2-4 を OFF に設定

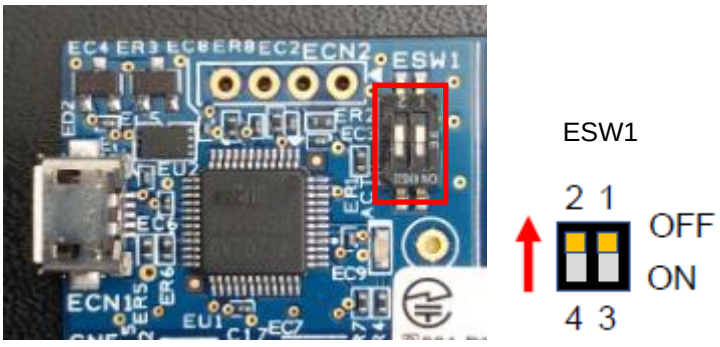

図 2-6　TB-RX23W ボードの書き込み後の ESW1 スイッチ設定

3.　デモ

Mesh センサデモの実行方法を示します。

3.1　デモ概要

図 3-1 に示すノードおよびネットワークの構成を使って、表 3-1 に示す Sensor Server ノード⇒Sensor Client ノードのセンサデータ通信、および Sensor Client ノード⇒Sensor Server ノードのデータ送信周期変更要求通信のデモが可能です。

また、ノードの状態変化をシリアル通信経由でログ出力できます。

表 3-1　デモ内容概要

ノード	デモ内容概要
Sensor Server _Low Power	• センサデータの取得と送信、および Friend ノードとの Friendship の確立：「3.4 センサデータ通信のデモ」を参照 • センサデータ送信周期の変更要求の受信と周期変更処理：「3.5 センサデータの送信周期変更のデモ」を参照
Sensor Client	• センサデータの受信：「3.4 センサデータ通信のデモ」を参照 • センサデータ送信周期の変更要求：「3.5 センサデータの送信周期変更のデモ」を参照
Friend_Relay	• 有無による Low Power ノードの動作変化確認「3.6 Friend ノードの有無による動作変化のデモ」を参照 • 有無による Mesh ネットワーク通信変化確認【注 1】
Relay	• 有無による Mesh ネットワーク通信変化確認【注 1】

注：全ノード共に RX マイコンの低消費電力低減機能のソフトウェアスタンバイモードには遷移しません。
注 1：全ノードを近距離に設置した場合、有無による Mesh ネットワークによる変化を確認できません。

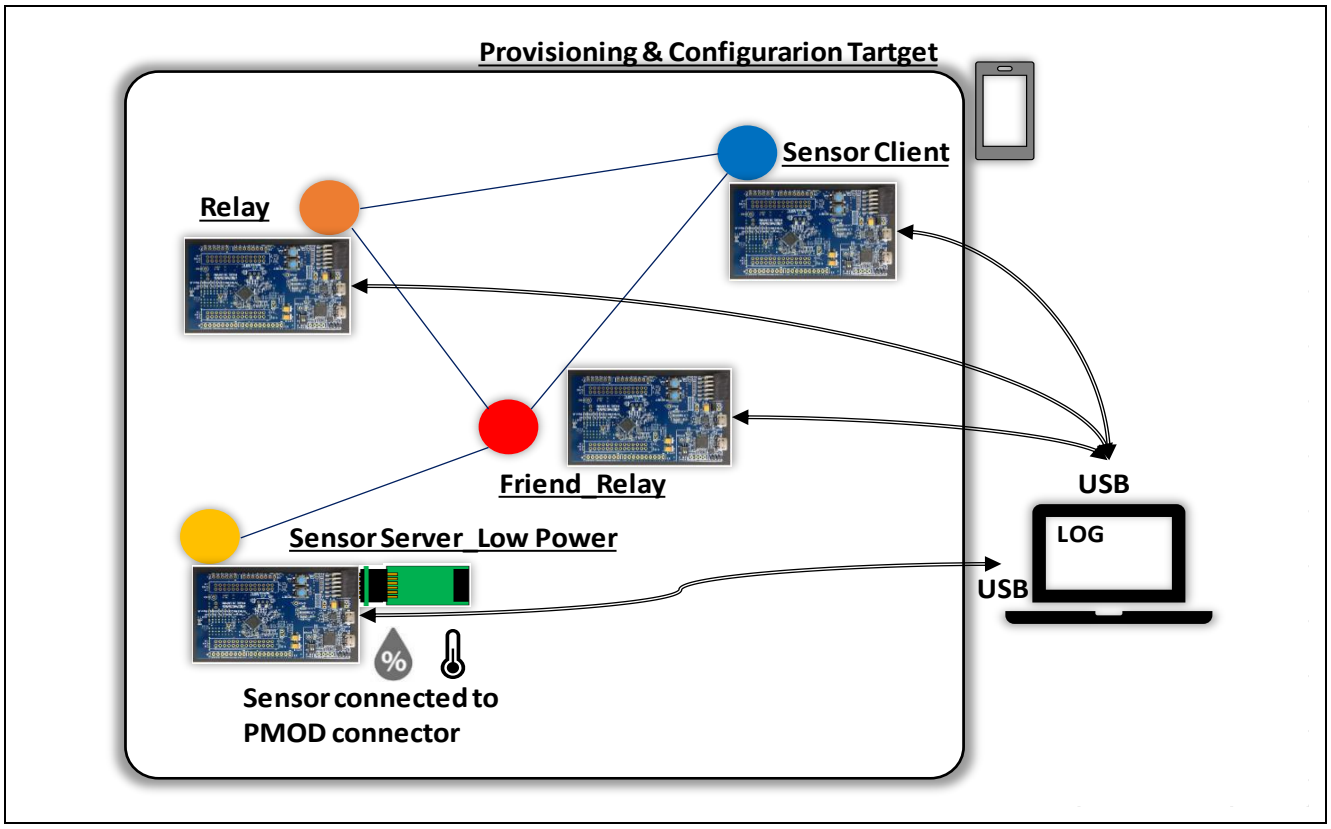

図 3-1　デモ全体の構成

3.2　ハードウェア環境

以下にデモ時のハードウェア構成を示します。

(1)　TB-RX23W ボード

「図 3-1　デモ全体の構成」に示す 4 種類のノードの TB-RX23W ボードを使用します。

「2 ファームウェア書き込み」を参照し、各ボード用のファームウェアを事前に書き込んでください。

ボード台数を確保できない場合、最小 2 台の構成による Sensor Model 機能の確認が可能です。

表 3-2　TB-RX23W ボードの構成

ノード	台数	備考
Sensor Client	1（必須）	—
Sensor Server _Low Power	1（必須）	—
Friend_Relay	1（オプション） 準備優先度：高	本ノードが無い場合、Sensor Server_Low Power ノードの Low Power 機能は動作しません。 本ノードと Relay ノードが共に無い場合、Sensor Client ノード⇔Sensor Server_Low Power ノード間で 1 対 1 通信可能な距離に置いてください。
Relay	1（オプション） 準備優先度：低	本ノードが無い場合、Sensor Client ノード⇔Sensor Server_Low Power ノード間で 1 対 1 通信可能な環境もしくは他 Relay 機能を有するノード利用による通信可能な距離に置いてください。

(2)　TB-RX23W ボードの ESW1 スイッチ設定確認

デモ実行のために、以下を設定してください。

- オンボードエミュレータを使用せずファームウェアを実行する場合：ESW1 の 2-4 を OFF に設定

ESW1 スイッチ設定が間違っている場合、正常動作しません。

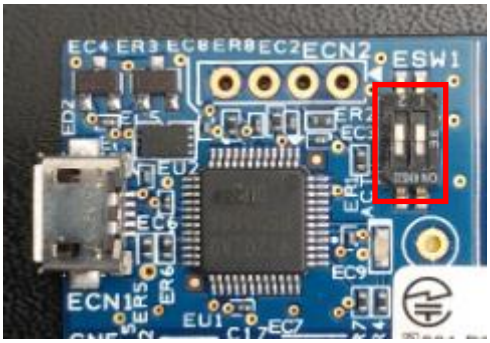

図 3-2　TB-RX23W ボードのデモ時の ESW1 スイッチ設定

(3)　Sensor Server_Low Power ノード用 TB-RX23W ボードと HS3001 センサボードの接続

図 3-3 に示すように、PMOD コネクタ(CN2)に、変換ボード(US082-INTERPEVZ)を介して、HS3001 センサボード(US082-HS3001EVZ)を多段接続方法で接続しください。

変換ボード(US082-INTERPEVZ)の 1 ピンマークを確認し、PMOD コネクタ(CN2)に正しく接続してください。

また、I2C バス信号をプルアップ処理させるため、HS3001 センサボード(US082-HS3001EVZ)上のジャンパ端子(J4, J5)を 2 つ共にショートさせてください。

なお、HS3001 センサボードが無い場合、センサデータの代替として、疑似データを生成します。

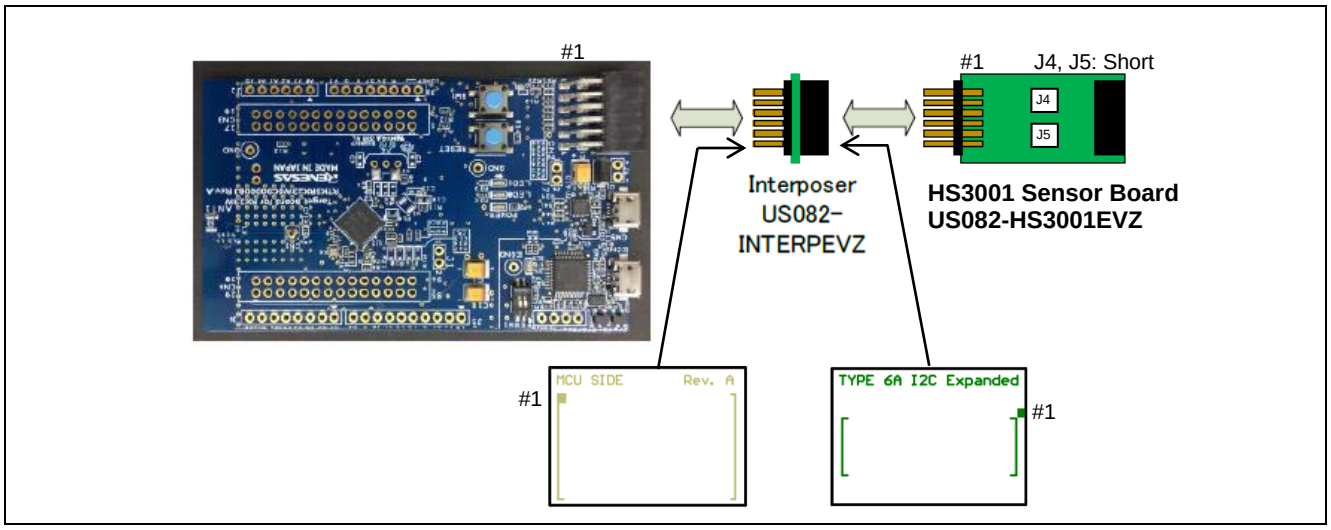

図 3-3　Sensor Server_Low Power ノード用 TB-RX23W ボードと HS3001 センサボードの接続

(4)　PC と TB-RX23W ボードのログ出力用シリアルケーブルの接続

各 TB-RX23W ボードからログ出力が可能です。

図 3-4 に示すように、PC と USB シリアル変換コネクタ(CN5)を USB ケーブルで接続してください。

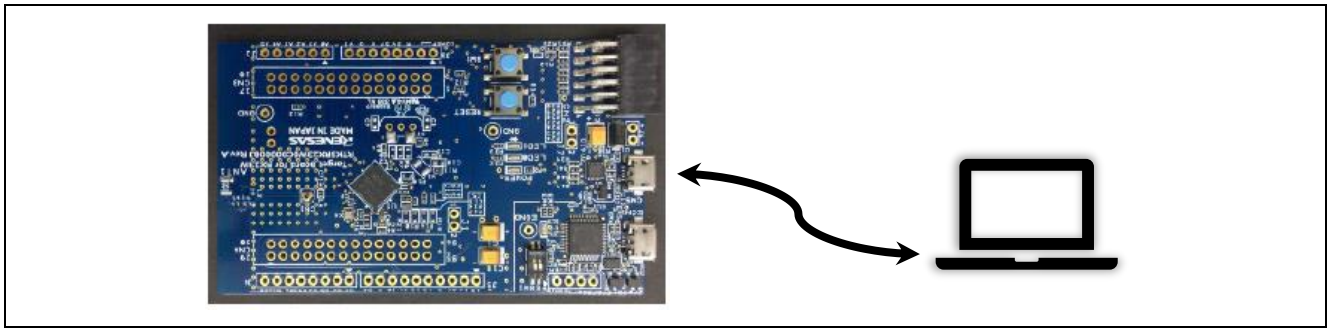

図 3-4　PC と TB-RX23W ボードとシリアルケーブルの接続

(5)　スマートフォン

各ノードのプロビジョニングおよびコンフィグレーションの目的で使用します。

利用可能なスマートフォン条件については、「1.3 動作確認環境」を参照してください。

3.3　ソフトウェア環境

事前に、「2 ファームウェア書き込み」を参照し、ファームウェアを書き込んでください。

3.3.1　ターミナルソフトの設定

シリアル通信を使ったログ出力を利用時にターミナルソフト（例：TeraTerm 等）が必要です。なお、ログ出力用のシリアル通信チャネル（TB-RX23W ボードの場合、SCI チャネル 8）を有効化する必要があります。

設定方法は、「RX23W グループ　Bluetooth Mesh モジュール Firmware Integration Technology (R01AN4930)」の「5.4.2 r_ble_rx23w」、「5.4.3 r_sci_rx」を参照してください。

(1)　シリアルポートの設定

表 3-3 にシリアルポートの設定を示します。

表 3-3　シリアルポート設定

項目	設定
ボーレート	115,200 bps
データ	8-bit
パリティ	なし
ストップ	1-bit
フロー制御	なし

3.3.2　Mesh Mobile のインストール手順

Mesh Mobile は、Mesh ノードのプロビジョニングとコンフィグレーションを行うためのツールで、Bluetooth Mesh スタックパッケージ(R01AN4930)に含まれています。

Mesh Mobile をインストールしていない場合はインストールしてください。以下にインストール手順例を示します。

1. Bluetooth Mesh スタックパッケージ(R01AN4930)の以下フォルダに格納されたパッケージファイル（apk ファイル）を USB 経由で PC からスマートフォンにコピー
   FITDemos\mesh_mobile\android-debug.apk

2. スマートフォン上の任意のファイルマネージャーアプリケーションで apk ファイルを実行

3.4　センサデータ通信のデモ

以下の処理順番で、センサデータの通信を開始します。「RX23W グループ　Bluetooth Mesh スタックスタートアップガイド」(R01AN4874)の「7. デモの実行」も参照してください。

1. Mesh Mobile を使ったプロビジョニングとコンフィグレーション
2. Low Power（Low Power 機能有効化済の Sensor Server_Low Power）ノードと Friend (Friend_Relay) ノードの Friendship 確立
3. Sensor Server_Low Power ノードのセンサデータ送信
4. Sensor Client ノードのセンサデータ受信

3.4.1　Mesh Mobile を使ったプロビジョニングとコンフィグレーション

全てのボードに電源を供給し、Mesh Mobile を起動します。

注: Android 端末向け Mesh Mobile 利用時は以下の許可設定が必要です。

- 位置情報
- ストレージ

以下に、プロビジョニングとコンフィグレーションの設定手順を示します。

設定手順とおりに進まない場合、「5 トラブル事例」も参照してください。

(1)　プロビジョニング

デバイスを Mesh ネットワークに追加し、ノードとして登録します。

全てのボードに対して、以下の手順を実行してください。

1. プロビジョニングされていないデバイスを探索するため、「PROVISION」タブに移動して「SCAN」ボタンをタップします。
2. 探索結果からプロビジョニングする任意のデバイスを選択します。
3. 接続の確立後、プロビジョニングが実行されます。

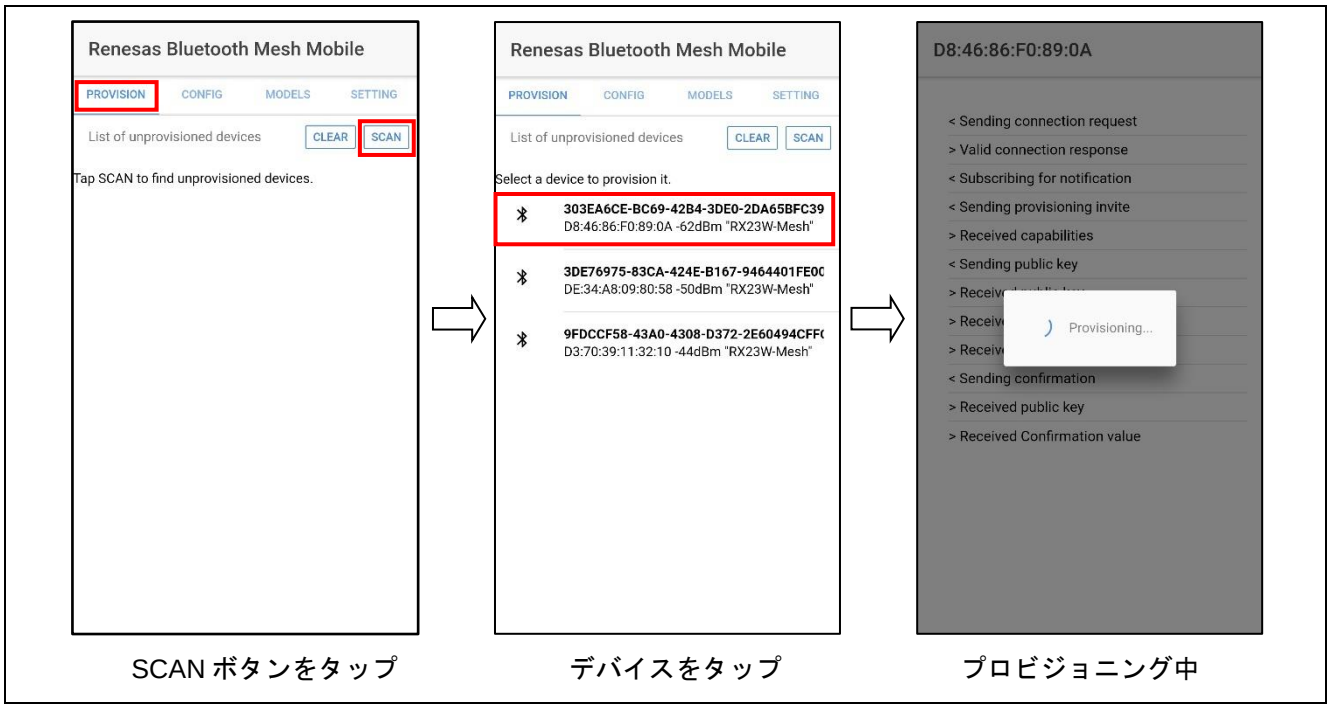

図 3-5　プロビジョニング操作画面

(2)　コンフィグレーション

プロビジョニングされたノードをコンフィグレーションします。

コンフィグレーションにより、各ノードが稼働し、Mesh Model 通信が可能になります。

以下の設定ができます。

— 複数ノードのグループ化設定
　設定したグループは複数のノードをグループ化して操作する際に利用できます。
　Mesh Model に関係なく、グループ作成が可能です。
— ノードに実装されたオプション機能（ファームウェア依存）の設定

(a)　グループ追加設定

使用制限につき、以下のグループ追加設定手順を省略してください。以降では、「Demo」グループを利用します。なお、次 Bluetooth Mesh スタックパッケージ以降で修正予定です。

以下の手順により、任意のグループを追加することができます。

1. 「MODELS」タブに移動し「ADD GROUP」ボタンをタップします。
   Mesh Model に依存しないため、「GENERIC ONOFF」タブもしくは「VENDOR STRING」タブのどちらか一方を選択状態でグループ追加ができます。
2. 「ADD GROUP」ダイアログ上で、"Kitchen"のような任意のグループ名を入力します。
3. グループが追加されたことを確認します。

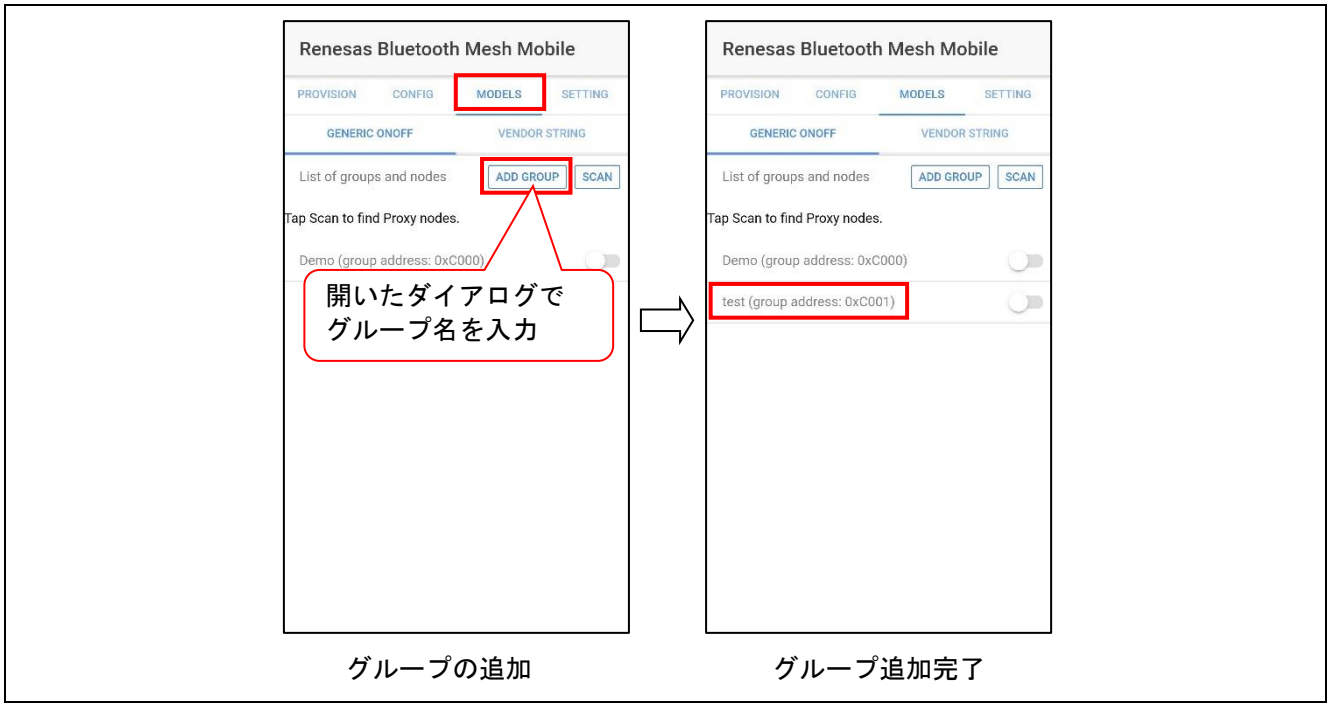

図 3-6　コンフィグレーション（グループの追加）操作画面

(b)　各ボードのグループ登録設定とノード設定

全てのボードに対して、以下の手順を実行することでボードのグループ登録ができます。

使用制限につき、「Demo」グループに登録してください。

1. ノードを探索するため、「CONFIG」タブに移動し「SCAN」ボタンをタップします。
2. 接続可能なノードは緑色で表示されます。接続を確立してコンフィグレーションを実行するため、緑色で表示された任意のノードを選択します。
3. 構成情報の表示後、「CONFIGURATION」タブに移動します。

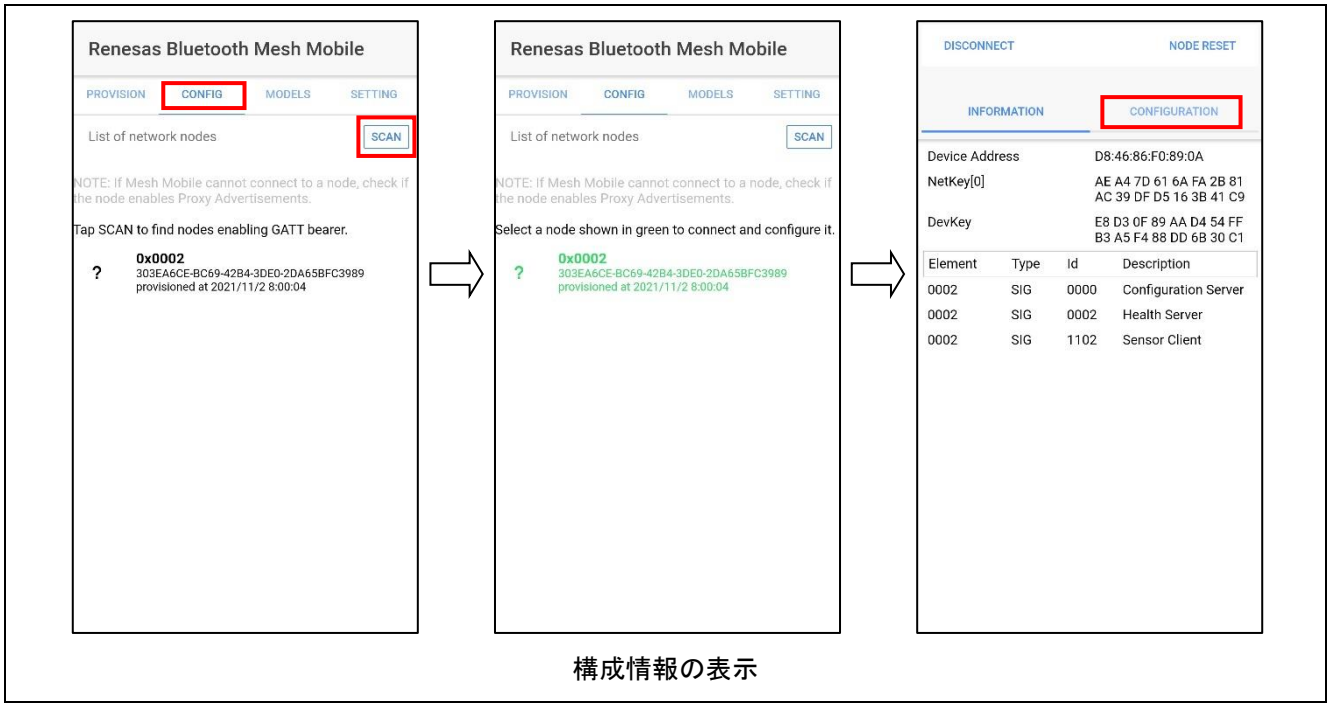

図 3-7　コンフィグレーション（グループ登録設定とノード設定）操作画面(1/2)

4. 「CONFIGURATION」タブ上でグループ設定とノード設定を選択します。
5. 「APPLY」ボタンをタップします。

表 3-4 にデモ時の各ノードの設定を示します。図 3-8 に設定時の画面を示します。

表 3-4　ノード一覧

Sensor Client ノード	Sensor Server _Low Power ノード	Friend_Relay ノード	Relay ノード
• 「Group」を選択		• 「Friend」と「Relay」を設定 • 「Group」設定不要	• 「Relay」を設定 • 「Group」設定不要

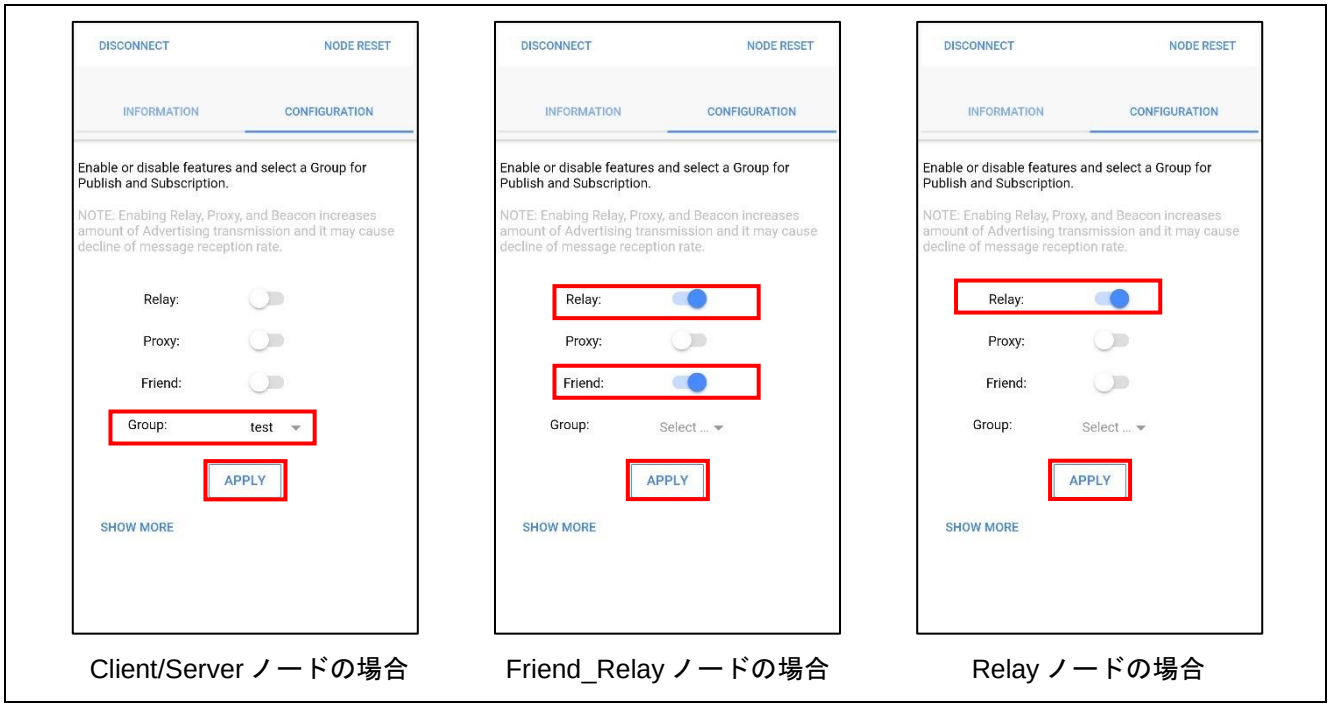

図 3-8　コンフィグレーション（グループ登録設定とノード設定）操作画面(2/2)

6. コンフィグレーションの完了後、「DISCONNECT」ボタンをタップすると、「CONFIG」タブに移動します。

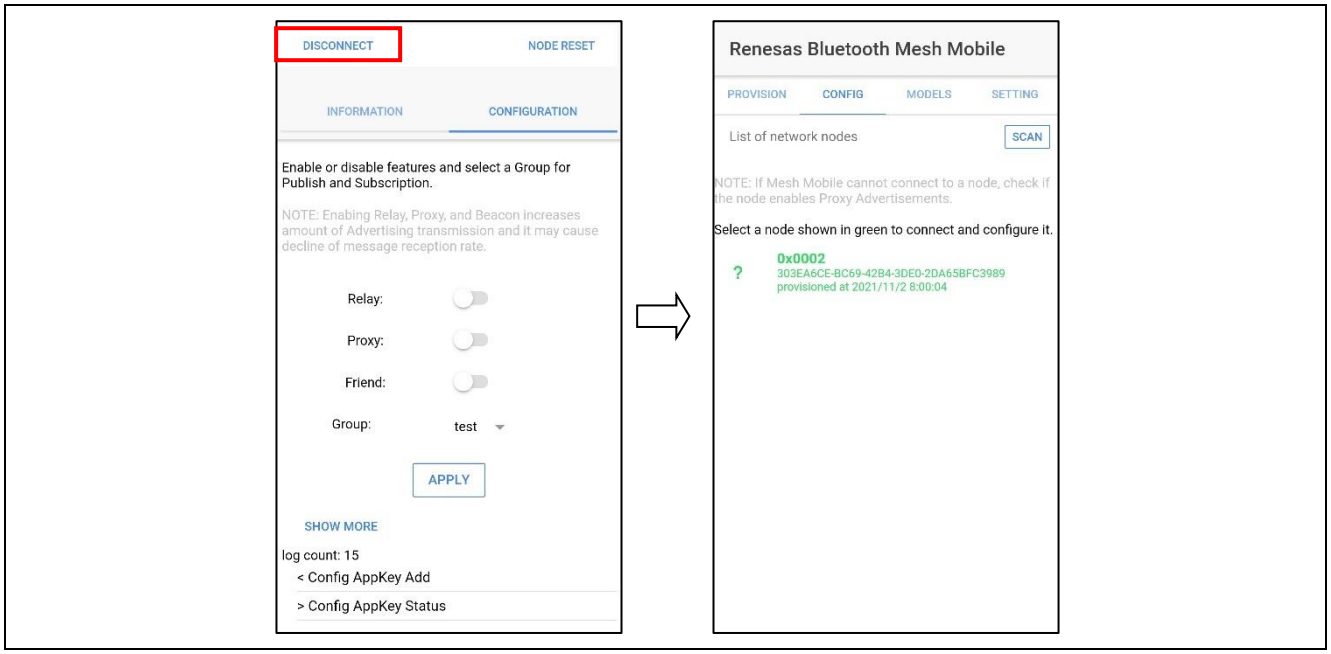

図 3-9　コンフィグレーション操作画面（DISCONNECT）

3.4.2　Low Power（Low Power 機能有効化済の Sensor Server_Low Power）ノードと Friend (Friend_Relay)ノードの Friendship 確立

プロビジョニング／コンフィグレーション済の Low Power ノードは、Friend ノードとの間で Friendship 確立処理を開始します。

Friendship 確立後、Low Power ノードは Scan の休止と再開と Friend ノードへの問い合わせを自動で実行します。「図 4-6　Lower Power ノードと Friend ノードの動作」も参照してください。

図 3-10 に Low Power ノードの Friendship 確立処理時のログ出力を示します。

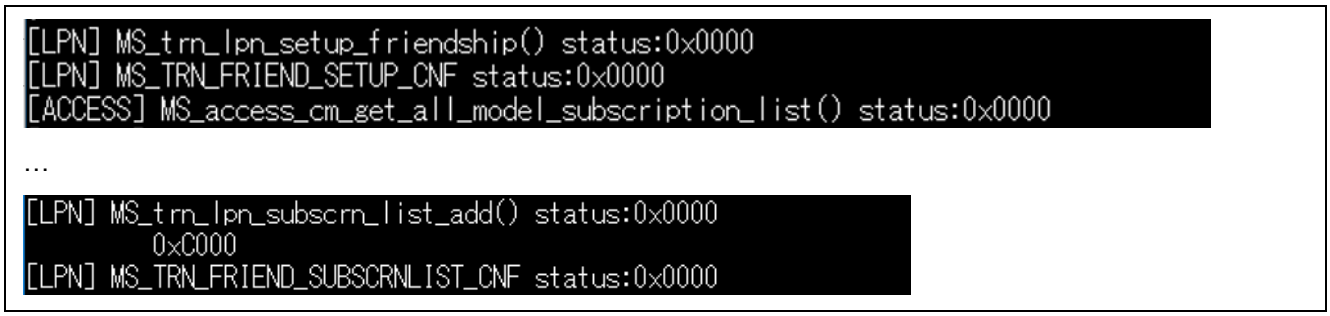

図 3-10　Low Power ノードの Friendship 確立処理時のログ出力

以下にログを順に説明します。

(1)　Low Power ノードのログ

表 3-5 に Low Power ノードのログを示します。

「RX23W グループ　Bluetooth Mesh スタック　開発ガイド(R01AN4875)」の「3.5.2 ローパワーノードのシーケンス」および「3.5.3 ローパワーノードのシーケンス」も参照してください。

表 3-5　Low Power ノードのログ内容

ログ	内容
[LPN] MS_trn_lpn_setup_friendship()	Friend ノードとの間で Friendship を確立するための Friend Request 送信開始を示します。
[LPN] MS_TRN_FRIEND_SETUP_CNF	Friendship 確立完了のイベントです。
[ACCESS] MS_access_cm_get_all_model_subscription_list()	Low Power ノードの全サブスクリプションアドレス取得を示します。
[LPN] MS_trn_lpn_subscrn_list_add()【注 1】	Friend ノードのフレンドサブスクリプションリストへの、Low Power ノードの全サブスクリプションアドレス追加を示します。
[LPN] MS_TRN_FRIEND_SUBSCRNLIST_CNF	Friend Subscription List に全てのサブスクリプションアドレス登録イベントです。

注 1：これにより、Friend ノードは Low Power ノード向けのメッセージを保管できます。

(2)　Friend ノードのログ

Friendship 確立時のログはありません。

3.4.3　Sensor Server (Sensor Server_Low Power)ノードのセンサデータ送信

Low Power 機能が有効化された Sensor Server ノードは定期的に以下の 1-4 の処理を繰り返します。

1. センシング実行によりセンサデータを取得します。
2. センサデータの送信
3. Low Power 機能により Scan 再開と Friend ノードへの問い合わせを実行します。
4. Low Power 機能により Scan を休止します。

図 3-11 に Sensor Server ノードのセンサデータ送信時のログ出力を示します。

なお、図中の表示値は、HS3001 センサボード未接続時の疑似データ値です。この送信値の場合の Sensor Client ノードでの受信時のログ出力を「図 3-12　Sensor Client ノードのセンサデータ（疑似データ）の受信時のログ出力」に示します。

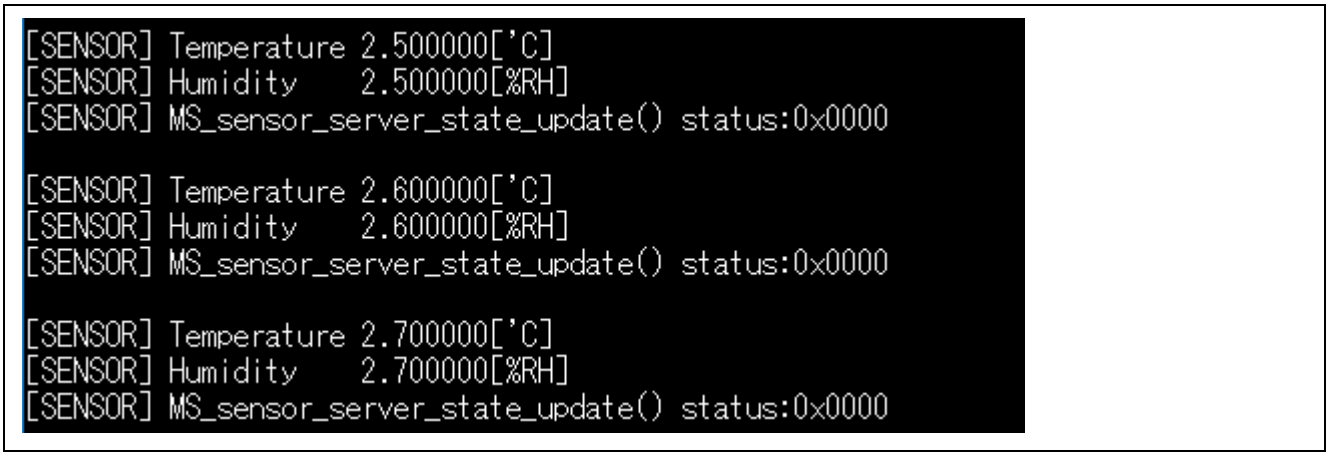

図 3-11　Sensor Server ノードのセンサデータ（疑似データ）の送信時のログ出力

HS3001 センサボード接続時には、HS300x センサから取得した測定値が出力されます。

(1)　Sensor Server ノードのログ

表 3-6 に Sensor Server ノードのログを示します。

表 3-6　Sensor Server ノードのログ内容

ログ	内容
[SENSOR] Temperature XX.XX[‘C]	温度[℃]を示します。
[SENSOR] Humidity XX.XX[%RH]	相対湿度[%RH]を示します。
[SENSOR] MS_sensor_server_state_update()	測定データの送信を示します。

3.4.4　Sensor Client ノードのセンサデータ受信

Sensor Client ノードは、センサデータの受信を繰り返します。

図 3-12 に Sensor Client ノードのセンサデータ受信時のログ出力を示します。「図 3-11　Sensor Server ノードのセンサデータ（疑似データ）の送信時のログ出力」に示すデータを送った場合の例です。

図 3-12　Sensor Client ノードのセンサデータ（疑似データ）の受信時のログ出力

HS3001 センサボード接続時には、HS300x センサから取得した測定値が出力されます。

(1)　Sensor Client ノードのログ

表 3-7 に Sensor Client ノードのログを示します。

表 3-7　Sensor Client ノードのログ内容

ログ	内容
[SENSOR] Temperature XX.XX[‘C]	温度[℃]を示します。
[SENSOR] Humidity XX.XX[%RH]	相対湿度[%RH]を示します。

3.5　センサデータの送信周期変更のデモ

Sensor Client ノードの TB-RX23W ボード上の SW1 押下により、センサデータ送信周期を 2 秒⇔5 秒（デフォルト：注）で切り替えることができます。

注：設定値については、「4.2.3(3) センサデータの送信周期の変更」を参照してください。

図 3-13 に Sensor Client ノードと Sensor Server ノードのセンサデータの送信周期変更時のログ出力を示します。

図 3-13 センサデータの送信周期変更時のログ出力

(1)　Sensor Client ノードの動作説明

ボード上の SW1 押下により、Sensor Cadence Set Unacknowledged メッセージを使ったセンサデータ送信周期変更要求を送信します。

(2)　Sensor Server ノードの動作説明

Sensor Cadence Set Unacknowledged メッセージを使ったセンサデータ送信周期変更要求を受信すると、センサデータ送信周期を切り替えます。

3.6　Friend ノードの有無による動作変化のデモ

図 3-14 に Friendship 確立後、Friend ノードの電源切断による Friendship 切断が発生した時のログ出力を示します。Friendship 切断発生後、Friendship を要求します。

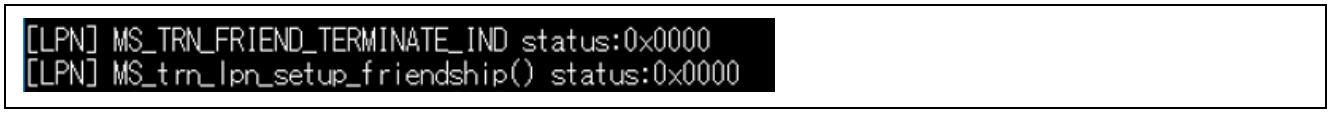

図 3-14 Friend ノードの電源を切断した際の Low Power (Sensor Server_Low Power)ノードのログ出力

図 3-15 に、Friend ノードの電源再投入による Friendship 再確立が発生した時のログ出力を示します。

図 3-15 Friend ノードの電源を再投入した際の Low Power (Sensor Server_Low Power)ノードのログ出力

表 3-8 に Low Power ノードのログを示します。

「RX23W グループ　Bluetooth Mesh スタック　開発ガイド(R01AN4875)」の「3.5.2 ローパワーノード」および「3.5.3 ローパワーノードのシーケンス」も参照してください。

表 3-8　Low Power ノードのログ内容

ログ	内容
[LPN] MS_TRN_FRIEND_TERMINATE_IND	Friendship の切断イベントです。
[LPN] MS_trn_lpn_setup_friendship()	Friend ノードとの間で Friendship を確立ための Friend Request 送信開始を示します。
[ACCESS] MS_access_cm_get_all_model_subscription_list()	Low Power ノードの全サブスクリプションアドレス取得を示します。
[LPN] MS_trn_lpn_subscrn_list_add() 【注 1】	Friend ノードのフレンドサブスクリプションリストへの、Low Power ノードの全サブスクリプションアドレス追加を示します。
[LPN] MS_TRN_FRIEND_SUBSCRNLIST_CNF	Friend Subscription List に全てのサブスクリプションアドレス登録イベントです。

注 1：これにより、Friend ノードは Low Power ノード向けのメッセージを保管できます。

3.7　Relay ノードの有無による動作

Relay ノードの電源の切断と供給を実行することで、Relay ノードの有無の動作を確認できます。

全ノードを近距離に設置した場合、有無による Mesh ネットワークによる変化を確認できません。

そのため、図 3-16 に示すように Sensor Client ノードと Friend ノード無線到達範囲外に設置し、それらの間に Relay ノードを設置することで、Relay ノードの動作を確認できます。

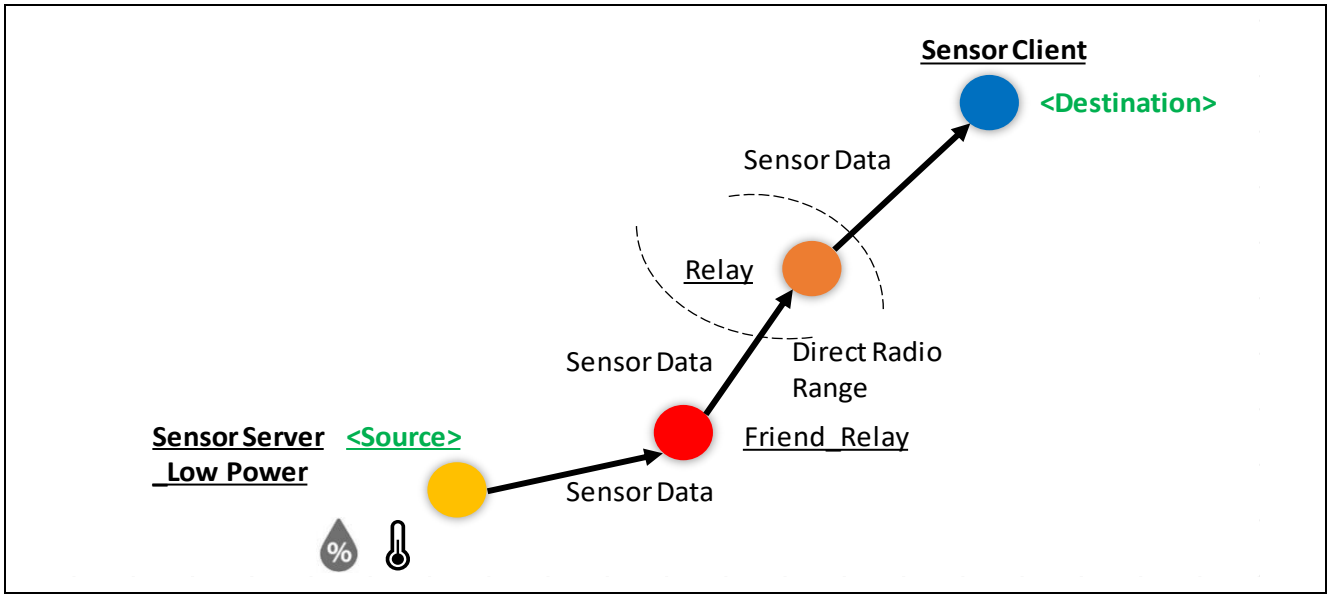

図 3-16　Sensor Client ノードと Friend ノードの無線到達範囲外に、それらの間に Relay ノードを設置した場合のデータの流れ

4.　プログラム説明

Mesh センサデモのプログラムについて説明します。

4.1　ソフトウェア構成

4.1.1　Sensor Sever_Low Power ノードのソフトウェア構成

以下に Sensor Sever のソフトウェア構成を示します。

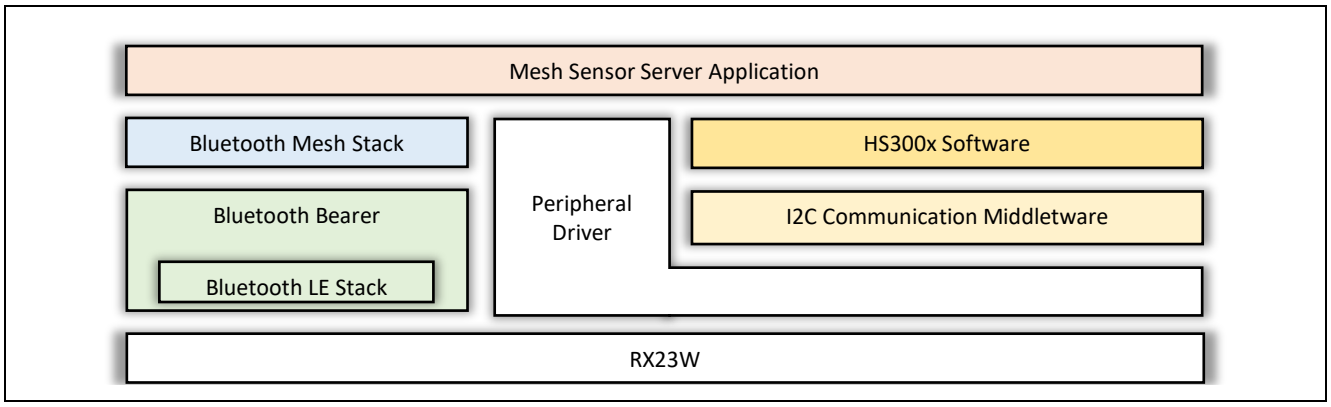

図 4-1　Sensor Server_Low Power ノードのソフトウェア構成

- Mesh Sensor Server Application

Bluetooth Mesh スタックおよび HS300x モジュールを制御します。他センサ制御の追加が容易にできます。

上記以外のソフトウェアについては、それぞれのドキュメントを参照してください。

4.1.2　Sensor Client ノードのソフトウェア構成

以下に Sensor Client のソフトウェア構成を示します。

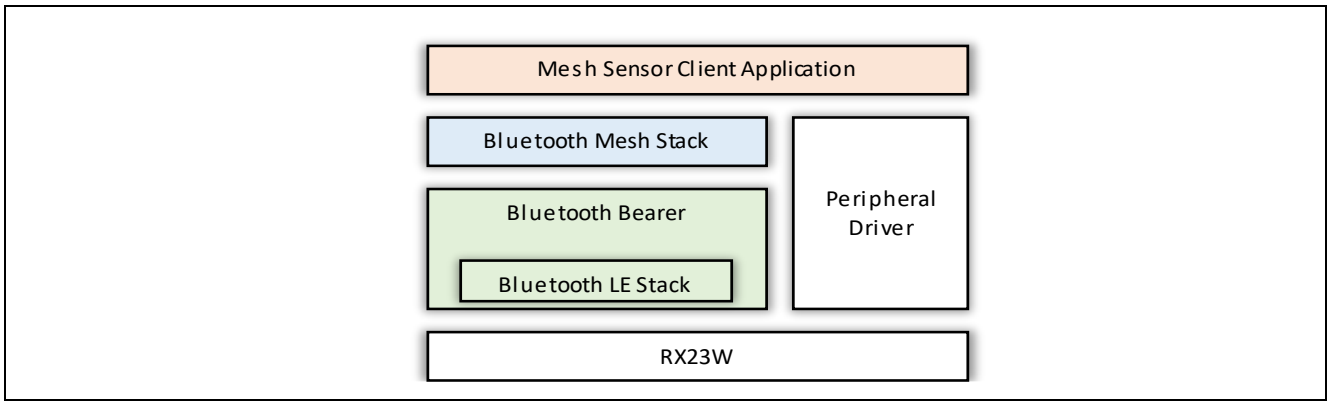

図 4-2　Sensor Client ノードのソフトウェア構成

4.2　Mesh Model

本 Mesh センサデモで使用する Mesh Model は、mesh_model.c ファイルに実装されています。

本章では、Sensor Model を主に説明します。Configuration Server Model と Health Sever Model については、「RX23W グループ　Bluetooth Mesh スタック　開発ガイド(R01AN4875)」を参照してください。

4.2.1　ノード毎の Mesh Model 構成

各ノードは、Foundation Model の Configuration Server/Health Server が必要です。

また、オプション機能である Relay/Proxy/Friend/Low Power 機能を有効化できます。「4.3 ノードのオプション機能設定」も参照してください。

Mesh センサデモで利用する各ノードの Mesh Model の構成を図 4-3 に示します。

Sensor Server ノードは、Mesh Model の Sensor Server Model を使用します。

Sensor Client ノードは、Mesh Model の Sensor Client Model を使用します。

本 Mesh センサデモでは、Relay ノードや Friend_Relay ノードを他目的で使用しないため、Configuration Server/Health Server 以外の Mesh Model の組み込みは不要です。

なお、Relay ノードと Friend_Relay ノードのプロジェクトは同じです。コンフィグレーションにより、ノード毎に機能を稼働させます。

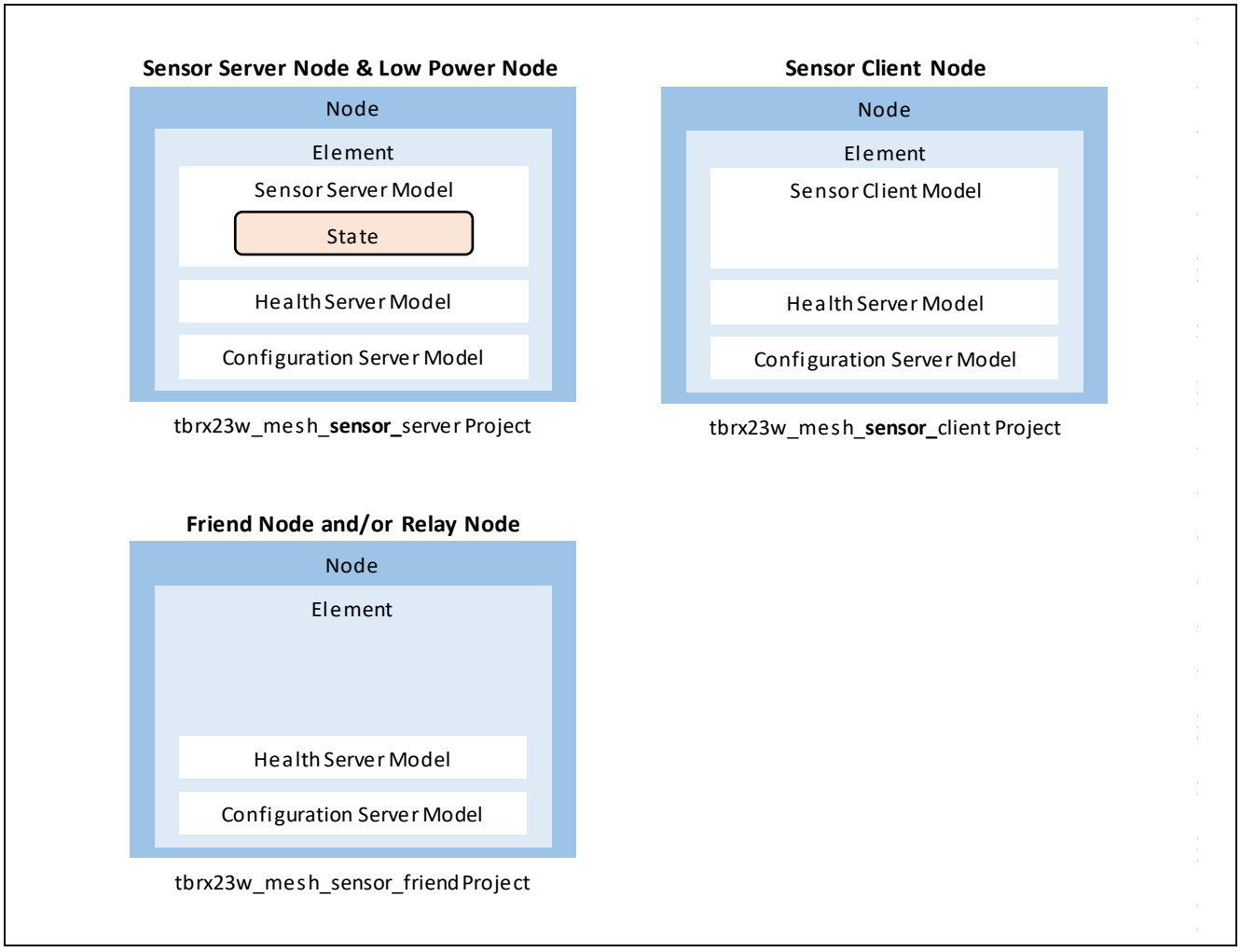

図 4-3　Mesh センサデモで利用するノードの Mesh Model の構成

4.2.2　Sensor Model (Sensor Server/Sensor Client)の設定方法

Bluetooth Mesh Model 仕様の Sensor Model を使用します。設定方法を表 4-1 に示します。

表 4-1　Sensor Model 設定方法

Model (Project Name)	ビルドオプション時のマクロ定義
Sensor Server Model (tbrx23w_mesh_sensor_server Project)	DEMO_SENSOR_SERVER_MODEL
Sensor Client Model (tbrx23w_mesh_sensor_client Project)	DEMO_SENSOR_CLIENT_MODEL

4.2.3　Sensor Model

Sensor Model には、4 つの Sensor State (Sensor Descriptor state, Sensor Setting and Sensor Cadence states, Sensor Data state, Sensor Series Column state)があります。

本 Mesh センサデモでは、Sensor Setting and Sensor Cadence states と Sensor Data state を使用します。

(1)　Sensor Server ノード

Sensor Server ノード用サンプルプログラムは、以下の動作が可能です。

- センサデータ送信周期で、HS300x センサの温湿度データを測定し、温湿度を表示します。
- センサデータを Sensor Client ノードに送信します。
  Sensor Status メッセージを使ったセンサデータ送信として Sensor Data state を使用します。
- Sensor Client ノードからセンサデータ送信周期の変更要求を受信し、センサデータ送信周期を更新します。
  Sensor Cadence Set Unacknowledged メッセージを使ったセンサデータ送信周期の変更要求として Sensor Setting and Sensor Cadence states を使用します。
  この State の Status Min Interval field の値によってセンサデータ送信周期が設定されます。

(a)　センサデータ送信

以下にセンサデータ送信時の Sensor Model の動作を示します。

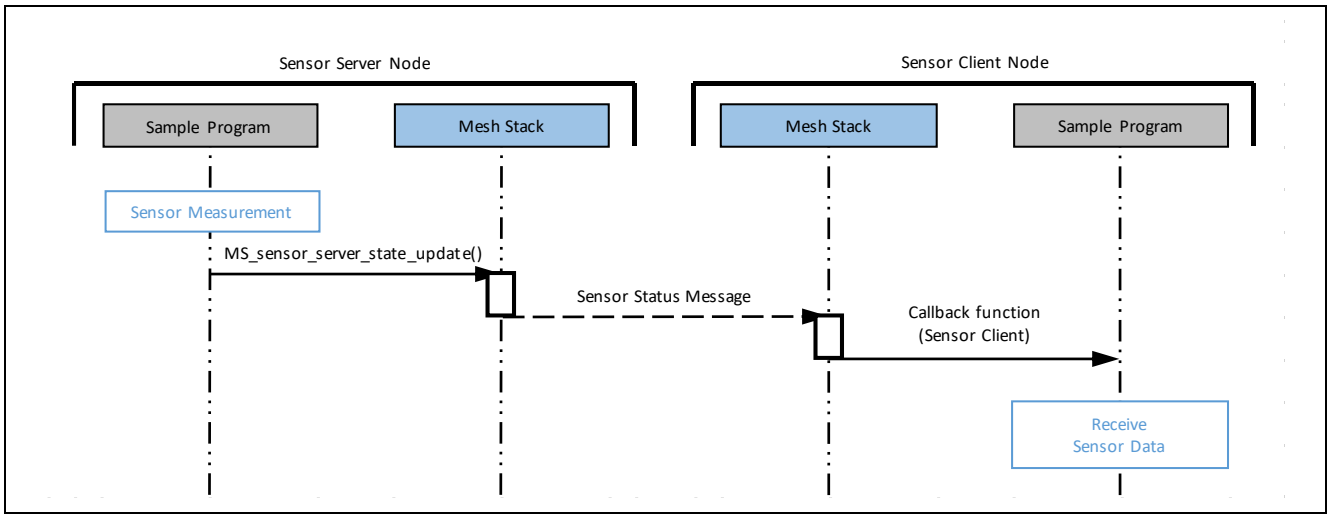

図 4-4　センサデータ送信時の Sensor Model 動作

(b)　センサデータ送信周期の変更要求の受信

以下にセンサデータ送信周期の変更要求時の Sensor Model の動作を示します。

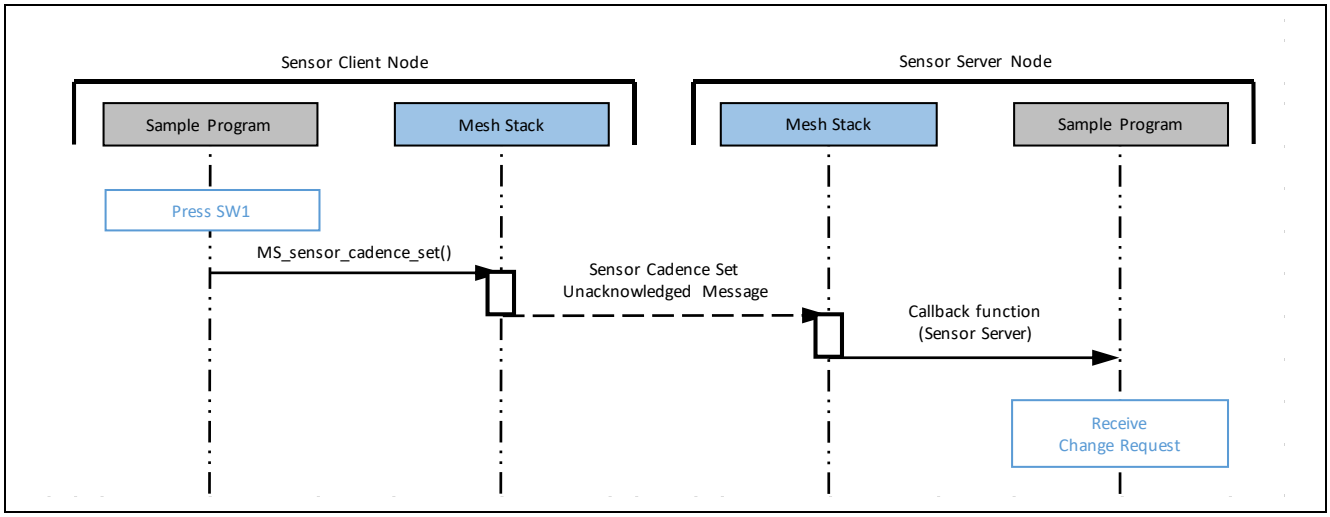

図 4-5　センサデータ送信周期の変更要求時の Sensor Model 動作

(2)　Sensor Client ノード

Sensor Client ノード用サンプルプログラムは、以下の動作が可能です。

— Sensor Server ノードからセンサデータを受信します。
— Sensor Client ノードのボード上の SW1 押下により、Sensor Cadence Set Unacknowledged メッセージを使ったセンサデータ送信周期の変更要求を送信します。

(a)　センサデータ受信

センサデータ受信時の Sensor Model の動作については、「図 4-4　センサデータ送信時の Sensor Model 動作」を参照してください。

(b)　センサデータ送信周期の変更要求の送信

センサデータ送信周期の変更要求の送信時の Sensor Model の動作については、「図 4-5　センサデータ送信周期の変更要求時の Sensor Model 動作」を参照してください。

(3)　センサデータの送信周期の変更

センサデータの送信周期を 2 つ定義できます。Sensor Client ノードのボード上の SW1 押下により送信周期が交互に切り替わります。

周期を変更したい場合は、Sensor Client プロジェクト内の mesh_appl.h ファイルの以下のマクロを変更してください。

表 4-2　センサデータ送信周期の設定マクロ

マクロ名	デフォルト値（単位：秒）
SENSOR_DATA_SEND_INTRERVAL01	2
SENSOR_DATA_SEND_INTRERVAL02	5

4.3　ノードのオプション機能設定

本 Mesh センサデモで使用するオプション機能(Low Power/Friend/Relay)を持つノードについて説明します。

各機能については、「RX23W グループ　Bluetooth Mesh スタック　開発ガイド(R01AN4875)」を参照してください。

4.3.1　Relay ノード

(1)　Relay 機能の有効化

Configuration Client である Mesh Mobile からのコンフィグレーションにより、Relay ノードとし稼働します。

4.3.2　Low Power ノード

(1)　Low Power 機能の有効化

Low Power 機能の有効化／無効化は、mesh_appl.h ファイルの LOW_POWER_FEATURE_EN マクロの値で設定してください。

本 Sensor Server プロジェクトでは、Low Power 機能を有効に設定しています。「4.6.2.1(2) mesh_appl.h」も参照してください。

フレンドシップ確立ができた場合、Low Power ノードは、Low Power 機能が稼働します。

表 4-3　LOW_POWER_FEATURE_EN マクロ

マクロ名	設定値	プロジェクト
LOW_POWER_FEATURE_EN	1	Sensor Server プロジェクト
	0	上記以外のプロジェクト

(2)　Scan 休止間隔

Scan の休止と再開と Friend ノードへの問い合わせを自動で実行します。Scan 再開タイミングで Friend ノードに問い合わせ、保管データがあればデータの受信を行います。データ送信後、再度 Scan 休止します。この Scan 休止から Scan 再開までの間隔を以下のマクロで制御しています。

「図 4-6　Lower Power ノードと Friend ノードの動作」も参照してください。

表 4-4　Scan 休止間隔マクロ

マクロ名	デフォルト値 (単位：100ms)	最小値	最大値
CORE_FRIEND_POLLTIMEOUT	50	1	345,600 (96 時間)

Scan 休止間隔を長く設定すると、Scan の停止時間が長くなり、消費電力を低減できますが、Client ノードからの要求を受け取るまでの時間が長くなります。そのため、Sensor Client ノードから要求された動作の開始まで時間がかかります。

「RX23W グループ　Bluetooth Mesh スタック　開発ガイド(R01AN4875)」の「3.5.3 ローパワーノードのシーケンス」も参照してください。

4.3.3　Friend ノード

(1)　Friend 機能の有効化

Configuration Client である Mesh Mobile からのコンフィグレーションにより、Friend ノードとして稼働します。

(2)　Friend ノードの動作

図 4-6 に、Mesh センサデモの Low Power ノードと Friend ノードの動作を示します。

Friend ノードの動作については、「RX23W グループ　Bluetooth Mesh スタック　開発ガイド (R01AN4875)」の「1.10.3 フレンドシップ」および「3.5 フレンドシップ」を参照してください。

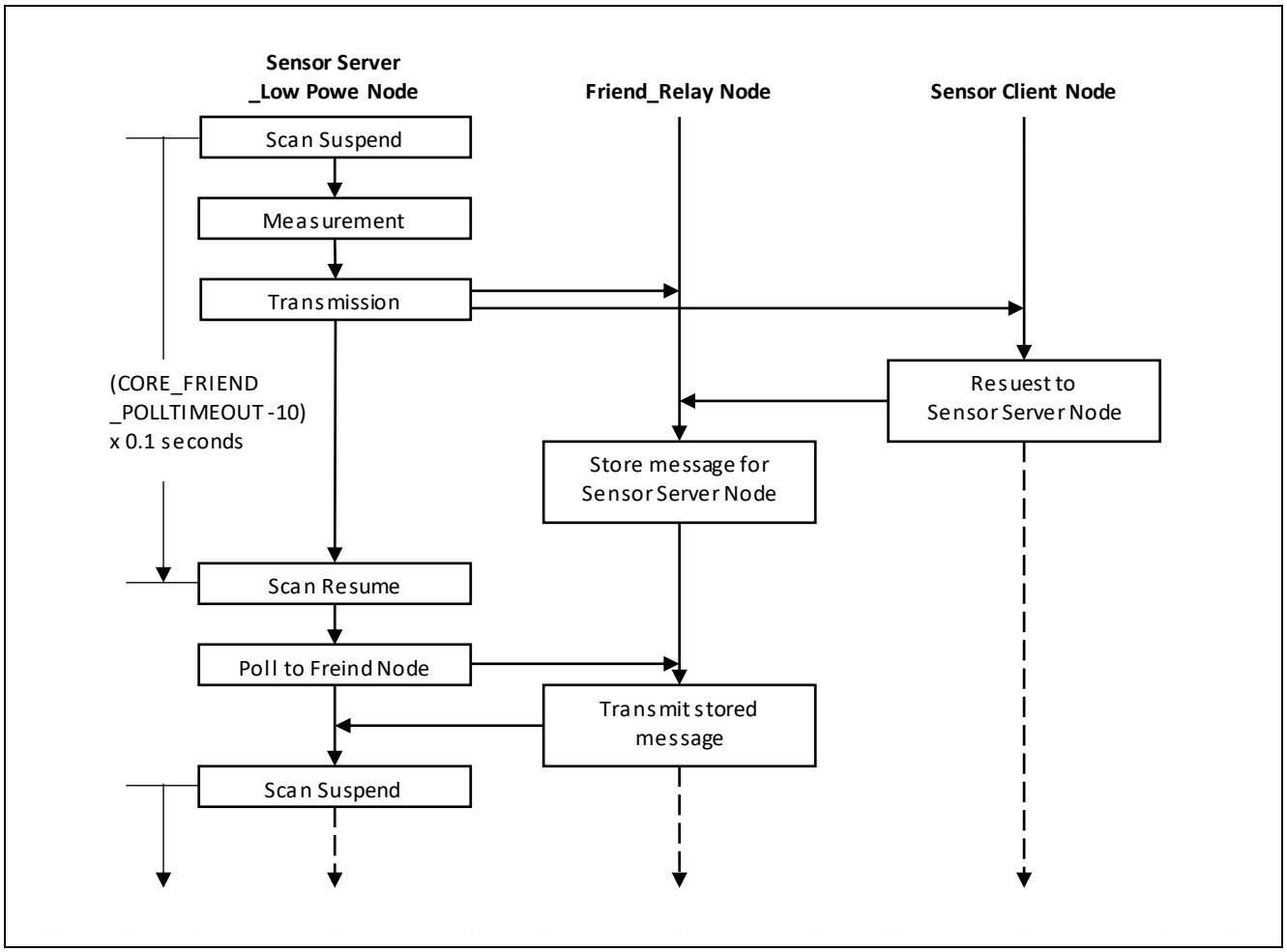

図 4-6　Lower Power ノードと Friend ノードの動作

4.4　ノード構成の設定

ノード構成の設定を以下に示します。本実装例は Mesh サンプルプログラムの mesh_model.c ファイルに実装されています。ノード構成を設定するためには、少なくとも次の実装が必要です。

(1) ノードの生成: MS_access_create_node()
(2) エレメントの登録: MS_access_register_element()
(3) Configuration Server Model の追加: MS_config_server_init()
(4) Health Server Model の追加: MS_health_server_init()
(5) Application Model 追加

(1)～(4)については、「RX23W グループ　Bluetooth Mesh スタック　開発ガイド(R01AN4875)」アプリケーションノートの「3.2 ノード構成」を参照ください。

(5) Application Model の追加の具体的内容については、「4.6.2.1(3) mesh_model.c」、「4.6.2.2(3) mesh_model.c」を参照してください。

4.5　各ノードのプロジェクトと Mesh サンプルヘッダ(mesh_appl.h)マクロ設定

以下に、Mesh サンプルヘッダ(mesh_appl.h)マクロ設定の違いを示します、

表 4-5　ノードと使用するプロジェクトの一覧

Node	Sensor Server _Low Power	Sensor Client	Friend_Relay	Relay
Project	tbrx23w_mesh_**sensor**_server	tbrx23w_mesh_**sensor**_client	tbrx23w_mesh_**sensor_friend**	

表 4-6　プロジェクトの Mesh サンプルヘッダ(mesh_appl.h)のマクロ設定

マクロ名	tbrx23w_mesh _**sensor**_server	tbrx23w_mesh _**sensor**_client	tbrx23w_mesh _**sensor_friend**
IV アップデート開始処理有効化 IV_UPDATE_INITIATION_EN	(1)	(1)	(1)
ローパワー機能有効化 LOW_POWER_FEATURE_EN	(1)	(0)	(0)
Mesh モニタ設定 CONSOLE_MONITOR_LOG	(0)	(0)	(0)
コンソール出力設定 CONSOLE_OUT_EN	(1)	(1)	(1)
コンソールへの ANSI CSI 出力設定 ANSI_CSI_EN	(1)	(1)	(1)
CPU 使用率測定の有効化 CPU_USAGE_EN	(0)	(0)	(0)

4.6　プログラム差分

各プロジェクトの差分を説明します。

表 4-7 に各プロジェクトとベースプロジェクトを示します。「1.1.2 Mesh センサデモパッケージ内の MOT ファイルと e2 studio プロジェクトについて」も参照してください。

表 4-7　プロジェクトとベースプロジェクトの一覧

プロジェクト名	ノード	ベースプロジェクト
tbrx23w_mesh_**sensor**_**server**	Sensor Server_Low Power	tbrx23w_mesh_server
tbrx23w_mesh_**sensor**_**client**	Sensor Client	tbrx23w_mesh_client
tbrx23w_mesh_**sensor**_**friend**	Friend_Relay Relay	

また、全プロジェクト共通で、以下を変更しています。

— BLE モジュールを Ver.2.30 に更新

4.6.1　Mesh FIT モジュール関連の r_xxx_config.h ファイル設定

全プロジェクト共通で、以下を設定しています。ベースのプロジェクトの設定値と同じです。

(1)　r_mesh_rx23w_config.h

Mesh FIT モジュール(R01AN4930)の「3.1 Mesh FIT モジュール」に示すデフォルト値を設定しています。

(2)　r_bsp_config.h

Mesh FIT モジュール(R01AN4930)の「3.2 BSP FIT モジュール」に示す Mesh FIT 向けの値を設定しています。

(3)　r_ble_rx23w_config.h

Mesh FIT モジュール(R01AN4930)の「3.3 BLE FIT モジュール」に示す Mesh FIT 向けの値を設定しています。

4.6.2　tbrx23w_sensor_mesh_server / tbrx23w_sensor_mesh_client プロジェクト

Sensor Model および HS300x モジュール関連の組み込みによる差分を以下に示します。

Sensor Model 制御追加時に簡素化を図るため、Vendor Model 用に定義された構造体や define を利用しています。

以下の識別子の箇所が、変更箇所です。

- DEMO_SENSOR_MODEL　：Sensor Model 用マクロ
- DEMO_SENSOR_SERVER_MODEL　：Sensor Server Model 用マクロ
- DEMO_SENSOR_CLIENT_MODEL　：Sensor Client Model 用マクロ

表 4-8 に変更したファイルの一覧を示します。以降にプロジェクト毎の変更内容を示します。

表 4-8　変更ファイルの一覧

ファイル名	tbrx23w_mesh_**sensor_server**	tbrx23w_mesh_**sensor_client**
maim.c	Sensor Server と Sensor Client で共通のファイル	
mesh_appl.h	Sensor Server 専用 （LOW_POWER_FEATURE_EN マクロの値以外は同じ）	Sensor Client 専用
meshu_model.c	Sensor Server と Sensor Client で共通のファイル	
r_irq_rx.c		
Pin.c	Sensor Server 専用	

4.6.2.1　tbrx23w_sensor_mesh_server プロジェクト

Sensor Server と Sensor Client で共通のファイルの場合、Sensor Server 用変更箇所のみ説明します。

(1)　main.c

HS300x モジュールのヘッダファイルの読み込み処理です。

48～50 行目

```
#ifdef DEMO_SENSOR_SERVER_MODEL
#include "RX_HS300X.h"
#endif
```

Sensor Server Model 用の変数宣言および変数の追加の処理です。

114～121 行目

```
#ifdef DEMO_SENSOR_SERVER_MODEL
#define SENSOR_DEMO_ADD (0.1)
static UINT32 gs_send_interval_timer_hdl;
static float g_demo_value_cnt = 0.0;
UINT16 g_send_Interval = 2000;
bool   g_interval_change_flg = false;
bool   g_send_flg = false;
#endif /* DEMO_SENSOR_SERVER_MODEL */
```

Sensor Model 用構造体の追加の処理です。

164～171 行目

```
#ifdef DEMO_SENSOR_SERVER_MODEL
static void mesh_sensor_server_set_cb
                (
                    MS_ACCESS_MODEL_REQ_MSG_CONTEXT * ctx,
```

```
                    MS_STATE_VENDOR_STRUCT          * state
                );
MS_ACCESS_MODEL_REQ_MSG_CONTEXT g_ctx;
#endif /* DEMO_SENSOR_SERVER_MODEL */
```

Sensor Model 用コールバック関数の設定処理です。

206～208 行目

```
#ifdef DEMO_SENSOR_SERVER_MODEL
.sensor_server_set_cb = mesh_sensor_server_set_cb,
#endif /* DEMO_SENSOR_SERVER_MODEL */
```

Sensor Server Model 用のセンサデータの送信処理です。

なお、HS3001 センサボード未接続の場合、疑似データ値を生成します。0 から始まり、0.1 毎インクリメントしたデータを送信します。float データのため少数部に誤差が発生することがあります。

299～357 行目

```
#ifdef DEMO_SENSOR_SERVER_MODEL
static void sever_send_data_timer_cb(UINT32 timer_hdl)
{
    API_RESULT  retval;
    static MS_STATE_SENSOR_DATA_STRUCT  param;
    static MS_ACCESS_MODEL_STATE_PARAMS current_state_params;
    UCHAR  dummy_data[10] = {0};
    UINT16 dummy_len;
    float consol_dummy_01;
    float consol_dummy_02;
    uint8_t demo_sensro_less_value[sizeof(float)] = {0};

    if (true == g_send_flg)
    {
        if (0 == *(const float*)g_sensor_temp_value)
        {
       < Operation without HS300x Sensor >
        }
        else
        {
       < Operation with HS300x Sensor >
        }
        dummy_len = 8;
        param.property_id_1 = 0xAA;
        param.raw_value_1 = dummy_data;
        param.raw_value_1_len = dummy_len;
        current_state_params.state_type = MS_STATE_SENSOR_DATA_T;
        current_state_params.state = &param;

        retval = MS_sensor_server_state_update(&g_ctx, &current_state_params, NULL,
0, NULL, 0, 0);
        CONSOLE_OUT ("[SENSOR] Temperature %f['C]\n", consol_dummy_01);
        CONSOLE_OUT ("[SENSOR] Humidity    %f[%RH]\n", consol_dummy_02);
        CONSOLE_STATUS("[SENSOR] MS_sensor_server_state_update()", retval);
        CONSOLE_OUT ("\n");
    }
}
#endif /* DEMO_SENSOR_SERVER_MODEL */
```

Sensor Model 用の処理です。

583～599 行目

```
#ifdef DEMO_SENSOR_SERVER_MODEL
/**************************************************************************
* @brief Callback function to receive a new Vendor state
**************************************************************************/
static void mesh_sensor_server_set_cb
                (
                    MS_ACCESS_MODEL_REQ_MSG_CONTEXT * ctx,
                    MS_STATE_VENDOR_STRUCT          * state
                )
{
    --- Omitted ---
}
#endif /* DEMO_SENSOR_SERVER_MODEL */
```

Sensor Server Model 用のセンサデータ送信用タイマの設定処理です。

758～761 行目

```
int main(void)
{
    --- Omitted ---
#ifdef DEMO_SENSOR_SERVER_MODEL
    R_BLE_TIMER_Create(&gs_send_interval_timer_hdl, g_send_Interval,
BLE_TIMER_PERIODIC, sever_send_data_timer_cb);
    R_BLE_TIMER_Start(gs_send_interval_timer_hdl);
#endif /* DEMO_SENSOR_SERVER_MODEL */
    --- Omitted ---
}
```

Sensor Server Model 用のセンサデータの取得処理です。

780～783 行目

```
int main(void)
{
    --- Omitted ---
#ifdef DEMO_SENSOR_SERVER_MODEL
    /* Initialize HS3001 Sensor Fit */
    sensor_hs3001_init();
#endif /* DEMO_SENSOR_SERVER_MODEL */
    --- Omitted ---
}
```

Sensor Client ノードの要求により、Sensor Server ノードがセンサデータ送信周期を更新する処理です。

sensor_hs3001_main()によるセンサデータの取得処理です。

840～850 行目

```
int main(void)
{
    --- Omitted ---
    /* main loop */
    while (1)
    {
        --- Omitted ---
#ifdef DEMO_SENSOR_SERVER_MODEL
```

```
        if (true == g_interval_change_flg)
        {
            R_BLE_TIMER_Stop(gs_send_interval_timer_hdl);
            R_BLE_TIMER_Create(&gs_send_interval_timer_hdl, g_send_Interval,
BLE_TIMER_PERIODIC, sever_send_data_timer_cb);
            R_BLE_TIMER_Start(gs_send_interval_timer_hdl);
            g_interval_change_flg = false;
        }

        sensor_hs3001_main();
#endif /* DEMO_SENSOR_SERVER_MODEL */
    }
    --- Omitted ---
}
```

(2)　mesh_appl.h

以下のヘッダファイルの読み込み処理です。

- Vendor Model API：一部 Vendor Model 用に定義された構造体や define を利用するため
- Sensor Model API：
- Access Model API：サブスクリプションアドレス取得等のログ出力目的で使用

35～39 行目

```
#if defined(DEMO_SENSOR_SERVER_MODEL) || defined(DEMO_SENSOR_CLIENT_MODEL)
#include "vendor_model/vendor_api.h"
#include "MS_sensor_api.h"
#include "MS_access_api.h"
#endif /* defined(VENDOR_SERVER_MODEL) || defined(VENDOR_CLIENT_MODEL) */
```

Low Power 機能有効化設定です。本 Sensor Server プロジェクトでは、Low Power 機能を有効に設定しています。「4.3.2(1) Low Power 機能の有効化」も参照してください。

53 行目

```
#define LOW_POWER_FEATURE_EN            (1)
```

コールバック関数の関数宣言です。

217～221 行目

```
#ifdef DEMO_SENSOR_SERVER_MODEL
void (*sensor_server_set_cb)(
                MS_ACCESS_MODEL_REQ_MSG_CONTEXT * ctx,
                MS_STATE_VENDOR_STRUCT          * state);
#endif /* DEMO_SENSOR_SERVERT_MODEL */
```

(3)　mesh_model.c

ベースのプロジェクトの設定と同様に共通であり、Sensor Server Model と Sensor Client Model の処理が含まれています。Vendor Model を参考に、Sensor Model 処理を追加しています。また、一部 Vendor Model 用に定義された構造体や define を利用しています。

Sensor Model 用の変数宣言の処理です。

62～69 行目

```
#ifdef DEMO_SENSOR_SERVER_MODEL
static MS_ACCESS_MODEL_HANDLE   gs_sensor_server_model_handle;
```

```
static MS_ACCESS_MODEL_HANDLE   gs_sensor_setup_server_model_handle;
extern UINT16                   g_send_Interval;
extern bool                     g_interval_change_flg;
extern bool                     g_send_flg;
extern MS_ACCESS_MODEL_REQ_MSG_CONTEXT g_ctx;
#endif /* DEMO_SENSOR_SERVER_MODEL */
```

Sensor Model 用の構造体宣言の処理です。

113～122 行目

```
#ifdef DEMO_SENSOR_SERVER_MODEL
static API_RESULT mesh_model_sensor_server_cb
                (
                    MS_ACCESS_MODEL_REQ_MSG_CONTEXT * ctx,
                    MS_ACCESS_MODEL_REQ_MSG_RAW     * msg_raw,
                    MS_ACCESS_MODEL_REQ_MSG_T       * req_type,
                    MS_ACCESS_MODEL_STATE_PARAMS    * state_params,
                    MS_ACCESS_MODEL_EXT_PARAMS      * ext_params
                );
#endif /* DEMO_SENSOR_SERVER_MODEL */
```

Sensor Server Model 用の受信処理です。

607～769 行目

```
#ifdef DEMO_SENSOR_SERVER_MODEL
/*****************************************************************************
* @brief Callback function to receive events for Vendor Server model
*****************************************************************************/
static API_RESULT mesh_model_sensor_server_cb
                (
                    MS_ACCESS_MODEL_REQ_MSG_CONTEXT * ctx,
                    MS_ACCESS_MODEL_REQ_MSG_RAW     * msg_raw,
                    MS_ACCESS_MODEL_REQ_MSG_T       * req_type,
                    MS_ACCESS_MODEL_STATE_PARAMS    * state_params,
                    MS_ACCESS_MODEL_EXT_PARAMS      * ext_params
                )
{
    --- Omitted ---
}

#define MS_MAX_NUM_STATES    3
#define MS_MAX_SENSORS       1
#define CONSOLE_PRINT(...)

typedef struct _MS_SENSOR_STRUCT
{
    --- Omitted ---
} MS_SENSOR_STRUCT;

bool debug_sensor_led = false;
/**
 * Check if the sensor data to be published, based on the current value of the sensor
data,
 * Fast Cadence High, Low range
 */
API_RESULT appl_handle_sensor_publish_timeout(/* IN */ MS_ACCESS_MODEL_HANDLE *
handle)
{
```

```
    --- Omitted ---
}

/**
 * \brief Access Layer Model Publication Timeout Callback.
 *
 * \par Description
 * Access Layer calls the registered callback to indicate Publication Timeout
 * for the associated model.
 *
 * \param [in]  handle        Model Handle.
 * \param [out] blob          Blob if any or NULL.
 */
API_RESULT mesh_sensor_server_publish_timout_cb
           (
                /* IN */ MS_ACCESS_MODEL_HANDLE * handle,
                /* IN */ void                   * blob
           )
{
    --- Omitted ---
}
#endif /* DEMO_SENSOR_SERVER_MODEL */
```

Sensor Server Model 用のコンフィグレーション処理です。

897～926 行目

```
static API_RESULT mesh_model_config_server_cb
                (
                    MS_ACCESS_MODEL_REQ_MSG_CONTEXT * ctx,
                    UINT32                    opcode,
                    UCHAR                   * data_param,
                    UNIT16                    data_len
                )
{
    --- Omitted ---
        #ifdef DEMO_SENSOR_SERVER_MODEL
        if (opcode == gs_config_opcode_string_table[20].opcode)
        {
            g_send_flg = true;

            /* make MS_ACCESS_MODEL_REQ_MSG_CONTEXT for sending Sensor Status message
*/
            /** Model Handle - for which request is received */
            g_ctx.handle = gs_sensor_server_model_handle;

            /** Source Address - originator of request */
            MS_ACCESS_PUBLISH_INFO  publish_info;
            MS_access_cm_get_model_publication(gs_sensor_server_model_handle,
&publish_info);
            g_ctx.saddr = publish_info.addr.addr; // It assumes that Publish Address
is a address other than Virtual Address

            /** Destination Address - of the request */
            MS_access_cm_get_primary_unicast_address(&g_ctx.daddr);

            /** Associated Subnet Identifier */
            g_ctx.subnet_handle = 0x0000; // hard-coded: primary subnet

            /** Associated AppKey Identifier */
```

```
            g_ctx.appkey_handle = 0x0000; // hard-coded: first appkey

            CONSOLE_OUT("g_ctx.handle = 0x%04X\n", g_ctx.handle);
            CONSOLE_OUT("g_ctx.saddr = 0x%04X\n", g_ctx.saddr);
            CONSOLE_OUT("g_ctx.daddr = 0x%04X\n", g_ctx.daddr);
            CONSOLE_OUT("g_ctx.subnet_handle = 0x%04X\n", g_ctx.subnet_handle);
            CONSOLE_OUT("g_ctx.appkey_handle = 0x%04X\n", g_ctx.appkey_handle);
        }
        #endif /* DEMO_SENSOR_SERVER_MODEL */
```

Sensor Model 用モデルの登録処理です。

1146～1163 行目

```
static API_RESULT mesh_application_model_register(void)
{
    --- Omitted ---

    #ifdef DEMO_SENSOR_SERVER_MODEL
    retval = MS_sensor_server_init
                (
                    gs_element_handle,
                    &gs_sensor_server_model_handle,
                    mesh_model_sensor_server_cb,
                    mesh_sensor_server_publish_timout_cb
                );
    CONSOLE_STATUS("[SENSOR] MS_sensor_server_init()", retval);

    retval = MS_sensor_setup_server_init
                (
                    gs_element_handle,
                    &gs_sensor_setup_server_model_handle,
                    mesh_model_sensor_server_cb
                );
    CONSOLE_STATUS("[SENSOR] MS_sensor_setup_server_init()", retval);
    #endif /* DEMO_SENSOR_SERVER_MODEL */
```

(4)　Pin.c

Sensor Server ノード上の HS300x センサを I2C バスに接続するための設定ファイルで、「コードの生成」実行により生成されたファイルです。

そのため、識別子 DEMO_SENSOR_SERVER_MODEL はありません。

Sensor Server Mode プロジェクトの Pin.c ファイルが変更対象です。

68～74 行目

```
    /* Set SSCL1 pin */
    MPC.P30PFS.BYTE = 0x0AU;
    PORT3.PMR.BYTE |= 0x01U;

    /* Set SSDA1 pin */
    MPC.P26PFS.BYTE = 0x0AU;
    PORT2.PMR.BYTE |= 0x40U;
```

(5)　r_irq_rx.c

Sensor Client Model 用の SW1 押下時の検出フラグ処理です。

Sensor Client Model 用ですが、Sensor Server Mode プロジェクトも同じファイルを使用しています。Sensor Server では機能しません。

4.6.2.2　tbrx23w_sensor_mesh_client プロジェクト

Sensor Server と Sensor Client で共通のファイルの場合、Sensor Client 用変更箇所のみ説明します。

(1)　main.c

HS300x モジュールのヘッダファイルの読み込み処理です。

48～50 行目

```
#ifdef DEMO_SENSOR_SERVER_MODEL
#include "RX_HS300X.h"
#endif
```

Sensor Server Model 用の変数宣言および変数の追加の処理です。

106～112 行目

```
#ifdef DEMO_SENSOR_CLIENT_MODEL
static UCHAR  gs_sci_buf[MS_VENDOR_VALUE_MAX_SIZE];
static UINT16 gs_sci_max_len;
static UINT16 gs_sci_rcv_len;
static sci_comp_cb gs_sci_comp_cb = NULL;
extern unsigned char  g_server_setting_flg;
#endif /* DEMO_SENSOR_CLIENT_MODEL */
```

Sensor Model 用構造体の追加の処理です。

172～178 行目

```
#ifdef DEMO_SENSOR_CLIENT_MODEL
static void mesh_sensor_client_status_cb
                (
                    MS_ACCESS_MODEL_REQ_MSG_CONTEXT * ctx,
                    MS_VENDOR_STATUS_STRUCT         * status
                );
#endif /* DEMO_SENSOR_CLIENT_MODEL */
```

Sensor Model 用コールバック関数の設定処理です。

209～211 行目

```
#ifdef DEMO_SENSOR_CLIENT_MODEL
.sensor_client_status_cb = mesh_sensor_client_status_cb,
#endif /* DEMO_SENSOR_CLIENT_MODEL */
```

Sensor Model 用の処理です。

601～677 行目

```
#ifdef DEMO_SENSOR_CLIENT_MODEL
/*****************************************************************************
* @brief Callback function to receive a state from Vendor Server model
*****************************************************************************/
static void mesh_sensor_client_status_cb
                (
                    MS_ACCESS_MODEL_REQ_MSG_CONTEXT * ctx,
                    MS_VENDOR_STATUS_STRUCT         * status
                )
{
    --- Omitted ---
```

```
}

/**************************************************************************
* @brief Callback function for receiving and handling string from SCI
**************************************************************************/
static void sci_rcv_string(void)
{
    --- Omitted ---
}
#endif /* DEMO_SENSOR_CLIENT_MODEL */
```

Vendor Model 用の処理を Sensor Model センサ用に置換する処理です。

796～799 行目

```
int main(void)
{
    --- Omitted ---
    /* main loop */
    while (1)
    {
        --- Omitted ---
        #ifdef DEMO_SENSOR_CLIENT_MODEL
        /* Receive and Handle String from SCI */
        sci_rcv_string();
        #endif /* DEMO_SENSOR_CLIENT_MODEL */
        --- Omitted ---
    }
    --- Omitted ---
}
```

Sensor Client Model 用の SW1 押下による、センサデータ送信周期の変更を要求する処理です。

808～839 行目

```
int main(void)
{
    --- Omitted ---
    /* main loop */
    while (1)
    {
        --- Omitted ---
#ifdef DEMO_SENSOR_CLIENT_MODEL
        MS_SENSOR_CADENCE_SET_STRUCT param;
        static UCHAR  dummy_data = SENSOR_DATA_SEND_INTRERVAL01;
        API_RESULT    retval;
        if (1 == g_server_setting_flg)
        {
            if (SENSOR_DATA_SEND_INTRERVAL02 == dummy_data)
            {
                dummy_data = SENSOR_DATA_SEND_INTRERVAL01;
            }
            else if (SENSOR_DATA_SEND_INTRERVAL01 == dummy_data)
            {
                dummy_data = SENSOR_DATA_SEND_INTRERVAL02;
            }
            --- Omitted ---
        }
#endif /* DEMO_SENSOR_CLIENT_MODEL */
        --- Omitted ---
```

```
    }
    --- Omitted ---
}
```

(2)　mesh_appl.h

以下のヘッダファイルの読み込み処理です。

- Vendor Model API：一部 Vendor Model 用に定義された構造体や define を利用するため
- Sensor Model API：
- Access Model API：サブスクリプションアドレス取得等のログ出力目的で使用

35～39 行目

```
#if defined(DEMO_SENSOR_SERVER_MODEL) || defined(DEMO_SENSOR_CLIENT_MODEL)
#include "vendor_model/vendor_api.h"
#include "MS_sensor_api.h"
#include "MS_access_api.h"
#endif /* defined(DEMO_SENSOR_SERVER_MODEL) || defined(DEMO_SENSOR_CLIENT_MODEL) */
```

センサデータの送信周期の 2 つ定義です。「4.2.3(3) センサデータの送信周期の変更」も参照してください。

179～182 行目

```
#ifdef DEMO_SENSOR_CLIENT_MODEL
#define SENSOR_DATA_SEND_INTRERVAL01 (2)
#define SENSOR_DATA_SEND_INTRERVAL02 (5)
#endif /* DEMO_SENSOR_CLIENT_MODEL */
```

コールバック関数の関数宣言です。

223～227 行目

```
#ifdef DEMO_SENSOR_CLIENT_MODEL
void (*sensor_client_status_cb)(
                MS_ACCESS_MODEL_REQ_MSG_CONTEXT * ctx,
                MS_VENDOR_STATUS_STRUCT         * state);
#endif /* DEMO_SENSOR_CLIENT_MODEL */
```

Sensor Client 用関数宣言です。

261～265 行目

```
#ifdef DEMO_SENSOR_CLIENT_MODEL
API_RESULT mesh_model_sensor_client_get(void);
API_RESULT mesh_model_sensor_client_set(UCHAR * value, UINT16 len);
API_RESULT mesh_model_sensor_client_set_unack(UCHAR * value, UINT16 len);
#endif /* DEMO_SENSOR_CLIENT_MODEL */
```

(3)　mesh_model.c

ベースのプロジェクトの設定と同様に共通であり、Sensor Server Model と Sensor Client Model の処理が含まれています。Vendor Model を参考に、Sensor Model 処理を追加しています。また、一部 Vendor Model 用に定義された構造体や define を利用しています。

Sensor Model 用の変数宣言の処理です。

70～72 行目

```
#ifdef DEMO_SENSOR_CLIENT_MODEL
```

```
static MS_ACCESS_MODEL_HANDLE   gs_sensor_client_model_handle;
#endif /* DEMO_SENSOR_CLIENT_MODEL */
```

Sensor Model 用の構造体宣言の処理です。

123～131 行目

```
#ifdef DEMO_SENSOR_CLIENT_MODEL
static API_RESULT mesh_model_sensor_client_cb
                (
                    MS_ACCESS_MODEL_REQ_MSG_CONTEXT * ctx,
                    UINT32                    opcode,
                    UCHAR                   * data_param,
                    UINT16                    data_len
                );
#endif /* DEMO_SENSOR_CLIENT_MODEL */
```

Sensor Client Model 用の受信処理です。

770～872 行目

```
#ifdef DEMO_SENSOR_CLIENT_MODEL
/*****************************************************************************
* @brief Callback function to receive events for Vendor Client model
*****************************************************************************/
static API_RESULT mesh_model_sensor_client_cb
                (
                    MS_ACCESS_MODEL_REQ_MSG_CONTEXT * ctx,
                    UINT32                    opcode,
                    UCHAR                   * data_param,
                    UINT16                    data_len
                )
{
    --- Omitted ---
}

/*****************************************************************************
 * @brief Sends Vendor Get message
*****************************************************************************/
API_RESULT mesh_model_sensor_client_get(void)
{
    --- Omitted ---
}

/*****************************************************************************
 * @brief Sends Vendor Set message
*****************************************************************************/
API_RESULT mesh_model_sensor_client_set(UCHAR * value, UINT16 len)
{
    --- Omitted ---
}

/*****************************************************************************
* @brief Sends Vendor Set Unacknowledged message
*****************************************************************************/
API_RESULT mesh_model_sensor_client_set_unack(UCHAR * value, UINT16 len)
{
    --- Omitted ---
}
#endif /* DEMO_SENSOR_CLIENT_MODEL */
```

Sensor Model 用モデルの登録処理です。

1165～1173 行目

```
static API_RESULT mesh_application_model_register(void)
{
    --- Omitted ---

    #ifdef DEMO_SENSOR_CLIENT_MODEL
    retval = MS_sensor_client_init
                (
                    gs_element_handle,
                    &gs_sensor_client_model_handle,
                    mesh_model_sensor_client_cb
                );
    CONSOLE_STATUS("[SENSOR] MS_sensor_client_init()", retval);
    #endif /* DEMO_SENSOR_CLIENT_MODEL */

    --- Omitted ---
}
```

(4)　r_irq_rx.c

Sensor Client Model 用の SW1 押下時の検出フラグ処理です。

Sensor Client Model 用ですが、Sensor Server Mode プロジェクトも同じファイルを使用しています。

1015～1017 行目、および 1027～1029 行目

```
#ifdef DEMO_SENSOR_CLIENT_MODEL
unsigned char g_server_setting_flg = 0;
#endif
R_BSP_ATTRIB_INTERRUPT void irq5_isr(void)
{
    /* check callback address */
    if((FIT_NO_FUNC != (*(g_irq5_handle.pirq_callback))) && (NULL !=
(*(g_irq5_handle.pirq_callback))))
    {
        /* casting void * type to callback* type is valid */
        (*(g_irq5_handle.pirq_callback))((void*)&(g_irq5_handle.irq_num));
    }

#ifdef DEMO_SENSOR_CLIENT_MODEL
    g_server_setting_flg = 1;
#endif

} /* End of function irq5_isr */
```

4.6.3　tbrx23w_sensor_mesh_friend プロジェクト

以下のノード用プロジェクトです。

- TB-RX23W ボード用 Friend_Relay ノード
- TB-RX23W ボード用 Relay ノード

Sensor Model 用の変更はありません。

4.7　グローバル変数

表 4-9 に Sensor Model 用に追加したグローバル変数を示します。

表 4-9　グローバル変数一覧

変数名	型	備考
g_send_Interval	UINT16	Sensor Client ノードからのセンサデータの送信周期変更要求値を格納します。
g_interval_change_flg	bool	Sensor Client ノードからのセンサデータの送信周期変更要求値を受信したときに「true」がセットされます。
g_send_flg	bool	スマートフォンからの Configuration 設定完了時に「true」が入力されます。

4.8　メイン処理

以下にメイン処理のフローチャートを示します。

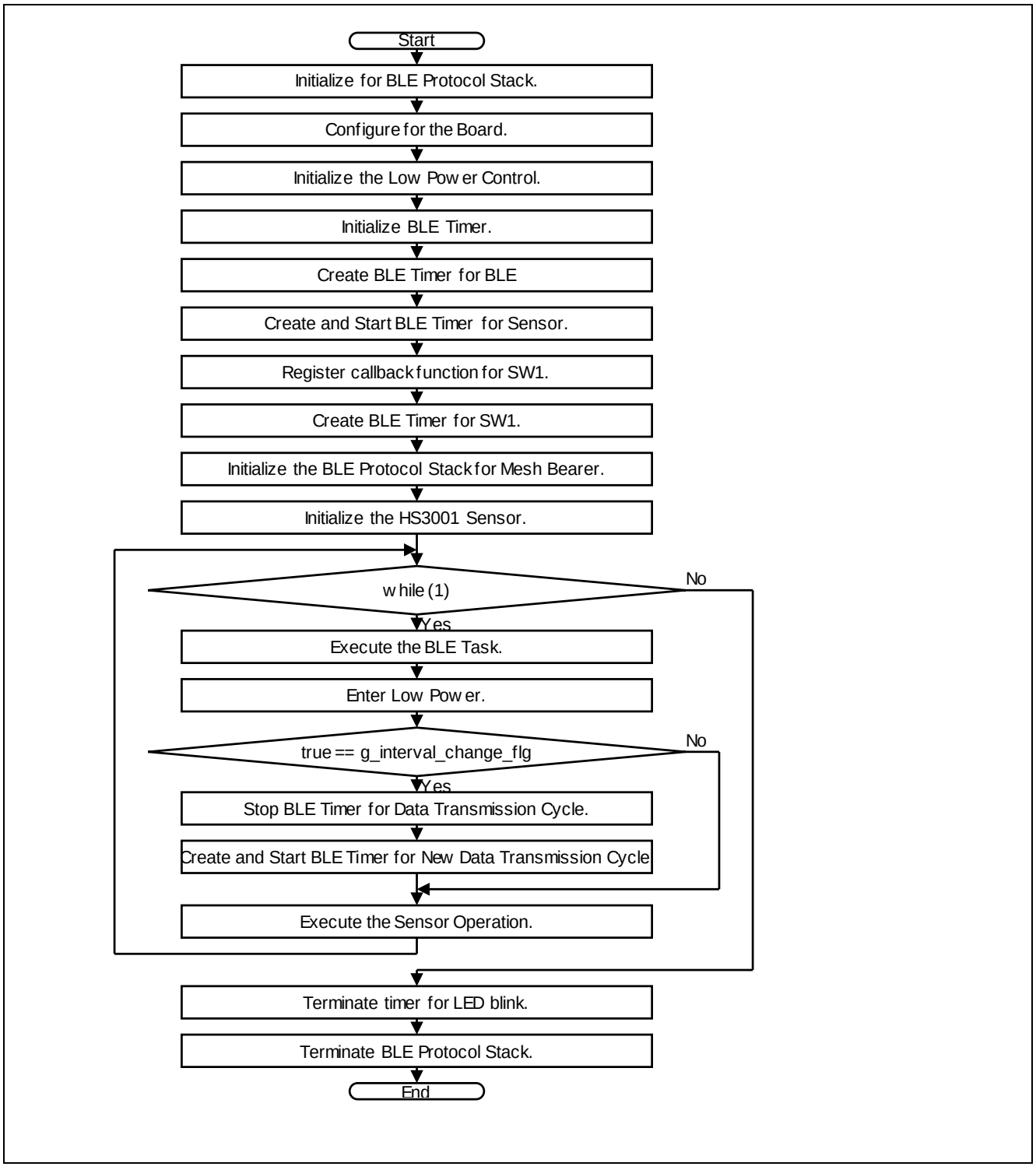

図 4-7　メイン処理のフローチャート

5. トラブル事例

トラブル事例を以下に示します。デモ時に参考にしてください。

1	• プロビジョニング時、デバイスが見つからない
	— 「SCAN」を再度タップしてください。
2	• 1 ノードのコンフィグレーショング完了の「DISCONNENT」タップ時、次の未コンフィグレーション・ノードが見つからない
	— 「SCAN」を再度タップしてください。

商標権および著作権

Bluetooth®のワードマークおよびロゴは登録商標であり、Bluetooth SIG, Inc が所有権を有します。ルネサス エレクトロニクス株式会社は使用許諾の下でこれらのマークおよびロゴを使用しています。その他の商標および登録商標は、それぞれの所有者の商標および登録商標です。

RX23W グループ Bluetooth Mesh スタックは次のオープンソースソフトウェアを使用します。

crackle; AES-CCM, AES-128bit 機能

BSD 2-Clause License

Copyright (c) 2013-2018, Mike Ryan
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Android™ は Google LLC の商標です。

Pmod™ は米国 Digilent Inc.の商標です。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2021/12/21	ー	新規作成
1.01	2022/2/7	1	要旨　MtoM を MtoM 通信に変更
		1	関連文書　「文章」を「文書」に変更
		5	1.1.1 Mesh ネットワーク構成とデータの流れ (2)、(3)　文章に図番号を追加
		12	1.3 動作確認環境　表 1-6 ノードの機能設定一覧　にて、 「Relay」を「Relay 機能」に変更
		16	3.1 デモ概要 「データ送信周期要求」を「データ送信周期変更要求」に変更
		25	3.4.2 Low Power（Low Power 機能有効化済の Sensor Server_Low Power）ノードと Friend (Friend_Relay)ノードの Friendship 確立 表 3-5 Low Power ノードのログ内容　の注 1 を修正
		29	3.6 Friend ノードの有無による動作変化のデモ 「3.5.2 ローパワーノードのシーケンス」を「3.5.2 ローパワーノード」に修正
		29	3.6 Friend ノードの有無による動作変化のデモ 表 3-6 Low Power ノードのログ内容　の注 1 を修正
		32	4.2.1 ノード毎の Mesh Model 構成 「センサ Mesh センサデモ」を「Mesh センサデモ」に修正
		39	4.6.2 tbrx23w_sensor_mesh_server / tbrx23w_sensor_mesh_client プロジェクト 「DEMO_SENSOR_SERVER_MODEL」を「DEMO_SENSOR_MODEL」に修正
		39	4.6.2 tbrx23w_sensor_mesh_server / tbrx23w_sensor_mesh_client プロジェクト 「表 4-8 変更ファイルの一覧と変更」を「表 4-8 変更ファイルの一覧」に修正
		46	4.6.2.2 tbrx23w_sensor_mesh_client プロジェクト 説明文を追加

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. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   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. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「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。
5. クロックについて
   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。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   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、将来の拡張機能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。
8. 製品間の相違について
   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、フラッシュメモリ、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、ノイズ幅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。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、お客様の責任において、お客様の機器・システムを設計ください。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. 当社製品または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. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 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。
6. 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。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 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。
9. 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。
10. 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。
11. 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。
12. 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。
13. 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。
14. 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.5.0-1 2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。