RL78/G1D ビーコンスタック
マルチホップ機能の実装(セキュリティ機能なし)

R01AN4375JJ0100
Rev.1.00
2018.10.31

要旨

本アプリケーションノートは、Bluetooth® Low Energy に対応した RL78/G1D 上で動作し、RL78/G1D ビーコンスタックを利用してマルチホップ機能を実装した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は、アプリケーションレイヤとマルチホップレイヤで構成されます。

マルチホップレイヤは、Bluetooth Low Energy の Advertising 動作と Scan 動作を制御して、フラッディング方式でマルチホップフレームの送信と受信、中継を行います。またマルチホップフレームの改竄検出と、データ内容を秘匿するためのセキュリティ機能の有無を選択できます。

アプリケーションレイヤには、RL78/G1D 評価ボードを使用して、マルチホップフレームの送受信を評価するための仕組みが実装されています。RL78/G1D 評価ボードのプッシュスイッチを押下すると、マルチホップフレームを一定周期で送信します。また自分宛のマルチホップフレームを受信すると、UART からログを出力し、マルチホップフレームの受信結果を確認できます。

なお同梱されたサンプルプログラムの実装内容(セキュリティ機能の有無)によって、アプリケーションノートの番号が異なります。

マルチホップ機能の実装(セキュリティ機能あり)	R01AN4466
マルチホップ機能の実装(セキュリティ機能なし)	R01AN4375

動作確認デバイス

RL78/G1D 評価ボード(RTK0EN0001D01001BZ)

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 V3.04 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	R20UT4206
CC-RL コンパイラ	
ユーザーズマニュアル	R20UT3123
RL78/G1D ビーコンスタック	
ユーザーズマニュアル	R01UW0171

目次

1. 概説

1.1 ユースケース

マルチホップ機能のユースケースを示します。

1.1.1 設定情報の伝送

図 1-1 にマルチホップ機能を利用して、照明機器の設定を変更するユースケースを示します。

照明リモコンが近くの照明に、特定の照明の照度を変更する設定を送信します。この照明は、この設定情報をマルチホップ機能で周囲の照明に送信します。周囲の照明は、さらにこの設定情報をマルチホップ機能で中継します。各照明が中継を繰り返すことで、設定情報は特定の照明に到達し、設定が反映されます。

照明リモコンから設定情報を直接受信できない照明も、マルチホップ機能を使用することで、設定情報を受け取ることができます。

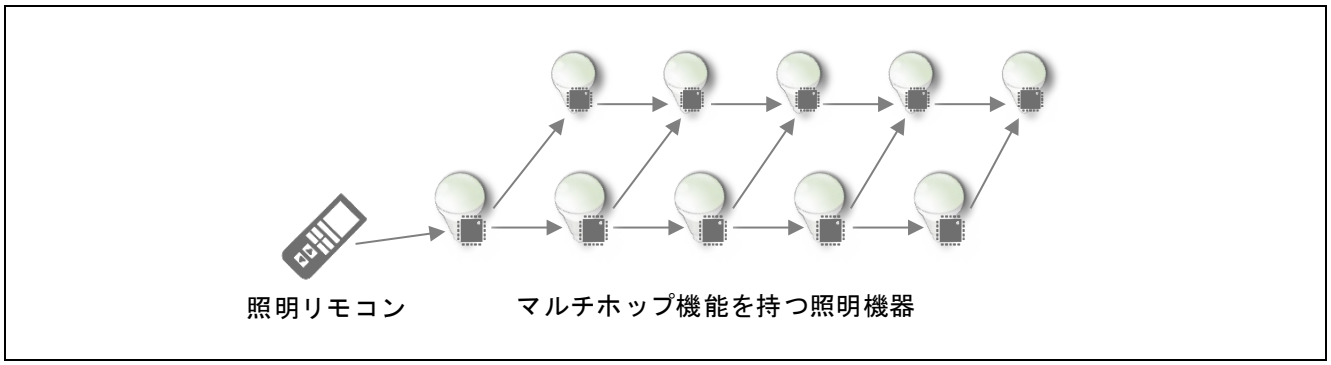


図 1-1　設定情報の伝送

1.1.2 測定データの収集

図 1-2 にマルチホップ機能を利用して、センサ機器のデータを収集するユースケースを示します。

各センサは、測定データをマルチホップ機能で周囲のセンサに送信します。周囲のセンサは、測定データをマルチホップ機能で中継します。ただしセンサが密集している場合は、一部のセンサのみ中継します。各センサが中継を繰り返すことで、センサデータはコレクタに到達し、監視や制御に利用されます。

コレクタに対して測定データを直接送信できないセンサも、マルチホップ機能を利用することで、測定データを通知することができます。また測定データの送信のみ行い中継を行わないセンサでは、送信時以外は RF トランシーバがスリープすることで、消費電力を低減することができます。

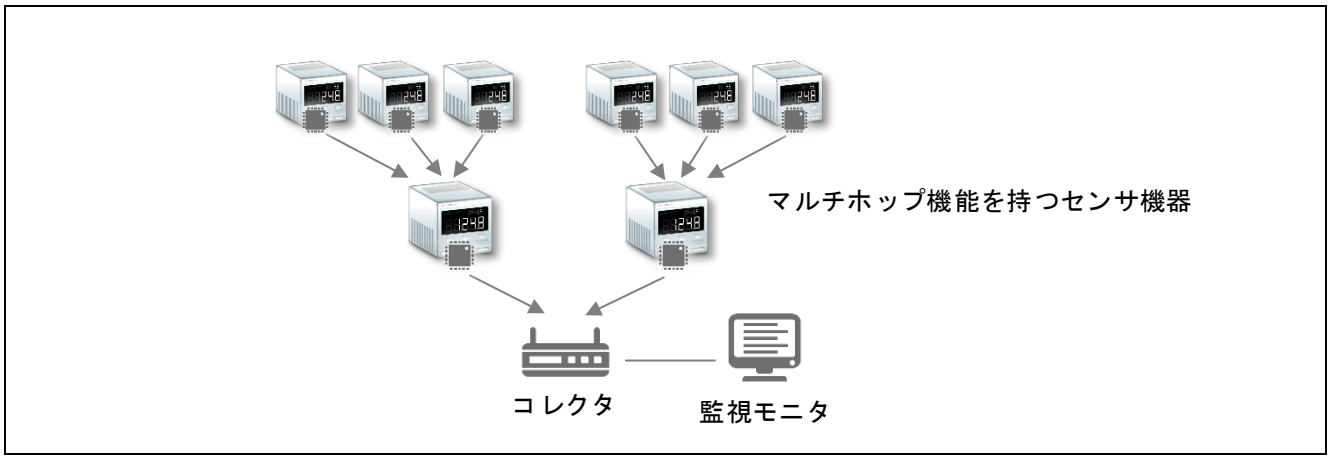


図 1-2　測定データの収集

2. システム構成

図 2-1 にサンプルプログラムのシステム構成を示します。

- アプリケーションレイヤ：

  マルチホップフレームの送受信を実行

- マルチホップレイヤ：

  Advertising 動作と Scan 動作を制御し、マルチホップフレームの送受信を管理

- ビーコンスタック：

  RL78/G1D の RF 部を制御し、Bluetooth Low Energy の Advertising 動作と Scan 動作を実行

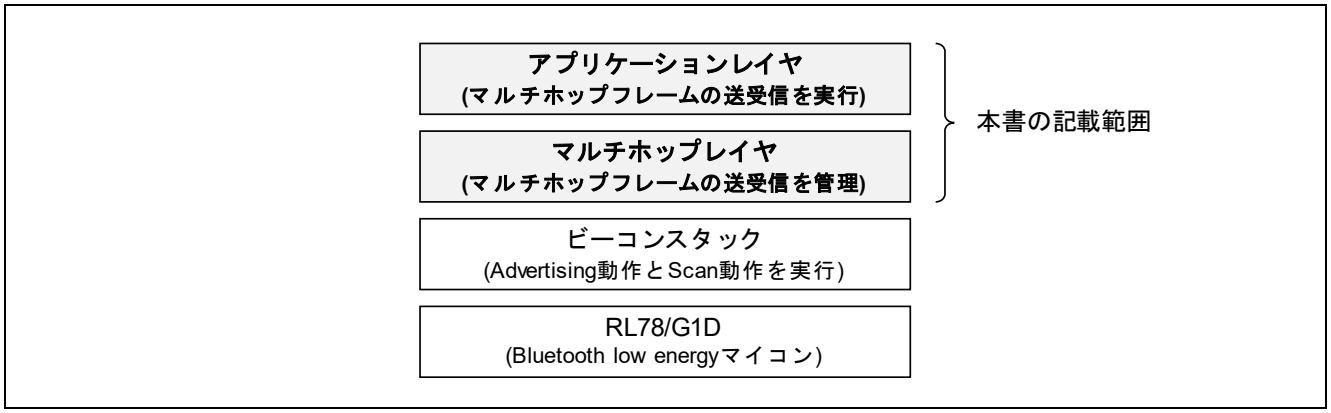


図 2-1　システム構成

アプリケーションレイヤとマルチホップレイヤはソースコードファイルで提供されます。各レイヤに対応するファイルを以下に示します。

アプリケーションレイヤ:　Project_Source¥application¥src¥r_node.c

マルチホップレイヤ:　Project_Source¥application¥src¥r_multihop.c

アプリケーションレイヤは、マルチホップレイヤの API を利用して周期的にフレームを送受信します。こちらをカスタマイズすることで、マルチホップ機能を利用したソフトウェアを開発頂けます。

マルチホップレイヤはマルチホップ機能が実装されており、そのままご使用頂くことを想定しています。

ビーコンスタックの仕様については『RL78/G1D ビーコンスタック ユーザーズマニュアル』(R01UW0171)を参照してください。

RL78/G1D の詳細については『RL78/G1D ユーザーズマニュアル ハードウェア編』(R01UH0515)を参照してください。

注意：本版のサンプルプログラムに同梱されているビーコンスタックは、マルチホップ対応のための暫定的な変更(R_BLE_StartAdvScan に対する機能追加)が含まれます。ルネサスの WEB サイトで公開されているビーコンスタックとは異なるため、ビーコンスタックライブラリは同梱されているものをお使いください。

3.　マルチホップレイヤ仕様

本章ではサンプルプログラムのマルチホップレイヤの仕様を示します。マルチホップレイヤの API 仕様については 8 章「マルチホップレイヤ API」を参照してください。

3.1　基本動作

図 3-1 にマルチホップの基本動作を示します。

ここではネットワークを構成する各デバイスをノードと呼びます。また各ノードが情報を伝送するためのデータユニットを、マルチホップフレームと呼びます。

マルチホップフレームは、各ノードが以下のような動作を行うことで、発信ノードから宛先ノードまで伝送されます。なお図中の円は、各ノードの送信したフレームが到達する範囲を示します。

1. 発信ノードは、宛先ノードに対してフレームを送信します。
2. 発信ノードからフレームを受信した中継ノードは、受信したフレームを中継します。
3. 各中継ノードが中継されたフレームをさらに中継することで、宛先ノードに到達します。
   また各中継ノードはランダム時間の経過後に送信することで、フレームの衝突を回避します。
4. 中継ノードは既に中継済みのフレームを破棄します。
5. 宛先ノードは既に受信済みのフレームを破棄します。

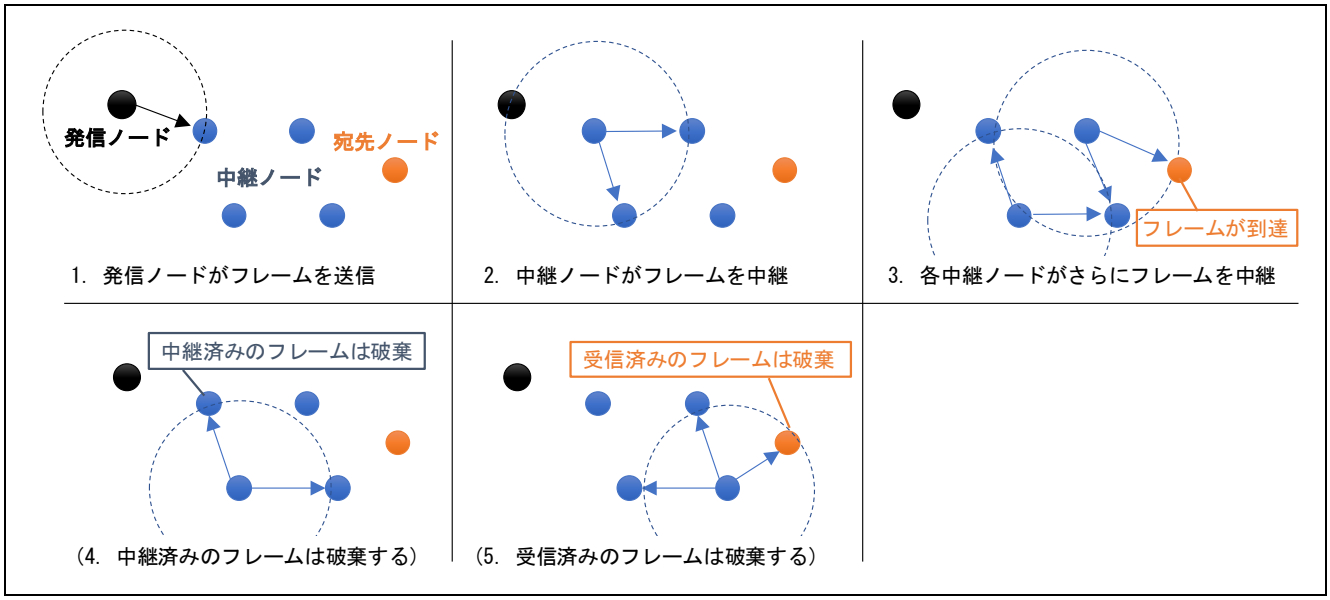


図 3-1　マルチホップのフレーム伝送

3.2　ネットワークとノード

図 3-2 にマルチホップのネットワークとノードについて示します。

ネットワークは、フレームを相互に送受信するノードの集団であり、ネットワーク ID で識別されます。各ノードは、同一ネットワークのノード間でのみフレームを送受信し、異なるネットワークのノード間ではフレームの送受信を行いません。

各ノードは、ノード ID で識別されます。同一ネットワークでは異なるノード ID を持つ必要があります。異なるネットワークではノード ID が重複しても問題ありません。

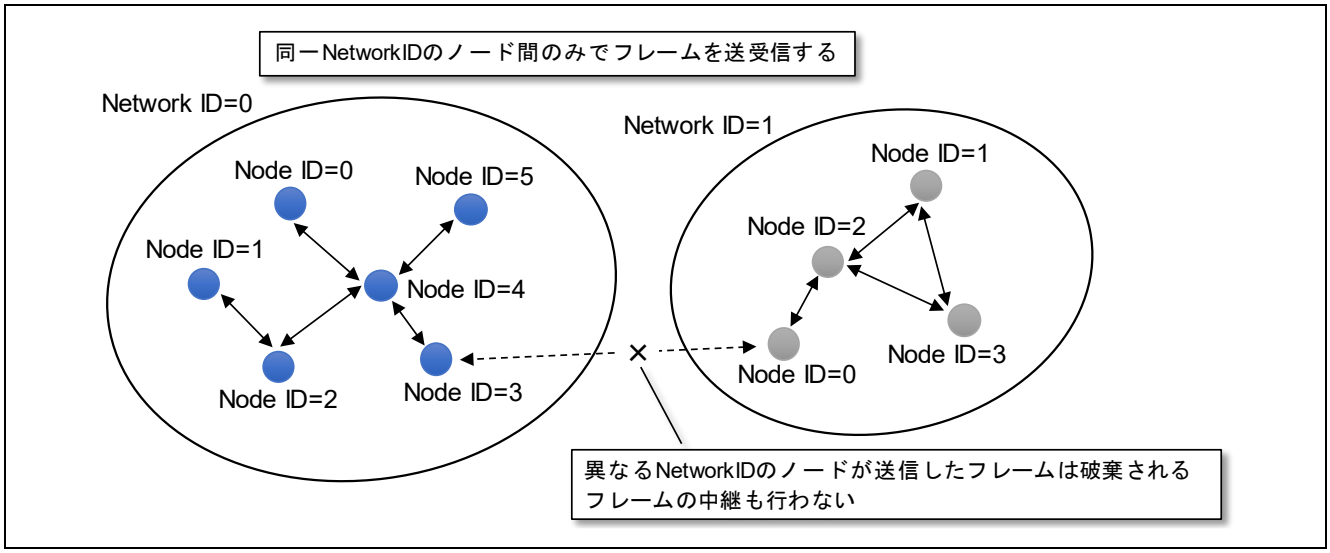


図 3-2　ネットワークとノード

サンプルプログラムでは、各デバイスに個別のネットワーク ID とノード ID を設定する仕組みとして、システム動作設定が実装されています。詳細は 4.3 項「システム動作設定」を参照してください。

3.3　セキュリティ機能

マルチホップレイヤは、フレーム改竄を検出するための認証と、データ内容を秘匿するための暗号化を行うセキュリティ機能を持ちます(※R01AN4466 のみ)。この認証と暗号化には AES-CCM*を使用し、フレームの暗号化と復号を行うため、各ノードに 128bit の共通暗号鍵を事前に設定する必要があります。

セキュリティ機能はそれぞれ下記の処理を実行します。

- フレームの発信時

  データを暗号化し、フレーム認証のための MIC を付加した暗号化フレームを発信する。

- 他ノード宛フレームの中継時

  暗号化フレームの MIC を計算して、付加された MIC と照合し、フレーム認証を行う。

- 自ノード宛フレームの受信時

  暗号化フレームの MIC 計算によるフレーム認証と、暗号化データの復号を行う。

セキュリティ機能のデータ暗号化とフレーム認証により、下記のような効果があります。

- データの暗号化

  データが暗号化されているため、傍受されても暗号鍵を持たないデバイスはデータを解読できない。

- フレームの認証

  暗号鍵を持たないデバイスがフレームを改竄しても、フレーム認証に失敗するため破棄される。
  またカウンタ値も改竄できないため、フレームを故意に再送するリプレイ攻撃が無効となる。

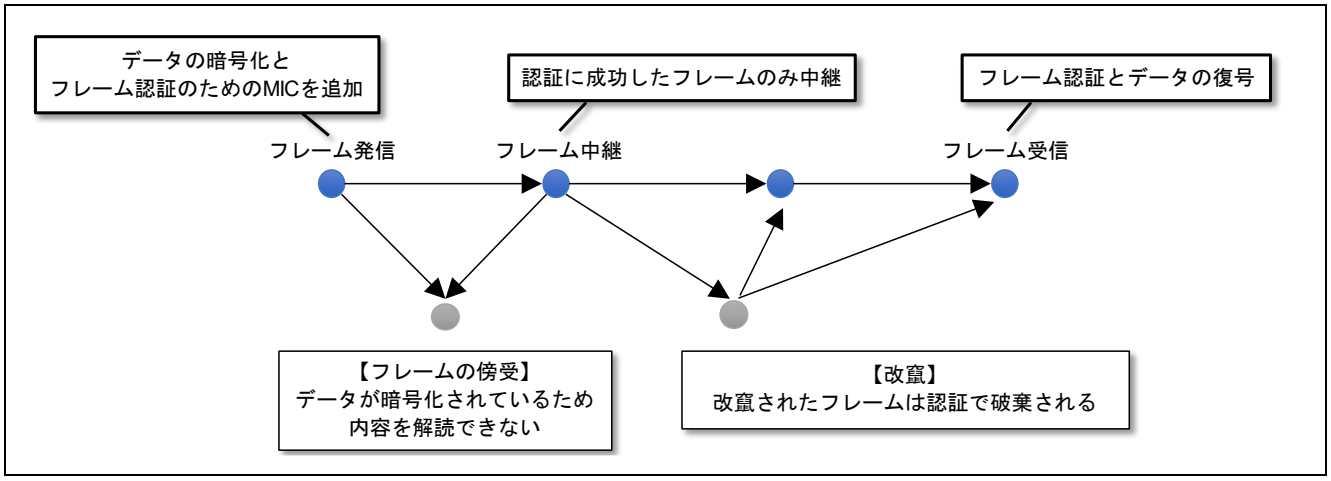


図 3-3　暗号化と認証によるセキュリティ

なおマルチホップレイヤはセキュリティ機能の無効時、受信した暗号化フレームを破棄します。またセキュリティ機能の有効時、受信した非暗号化フレームを破棄します。

フレームの暗号化と認証には AES ライブラリを使用します。詳細は下記を参照してください。

RL78 ファミリ用 AES ライブラリ
https://www.renesas.com/software-tool/crypto-library

注意：サンプルプログラムのセキュリティ機能は、システムの安全性を保証するものではありません。必要に応じて、お客様のアプリケーションによるセキュリティを追加実装してください。

3.4　マルチホップフレーム

図 3-4 と**表 3-1** にマルチホップフレームの構成を示します。

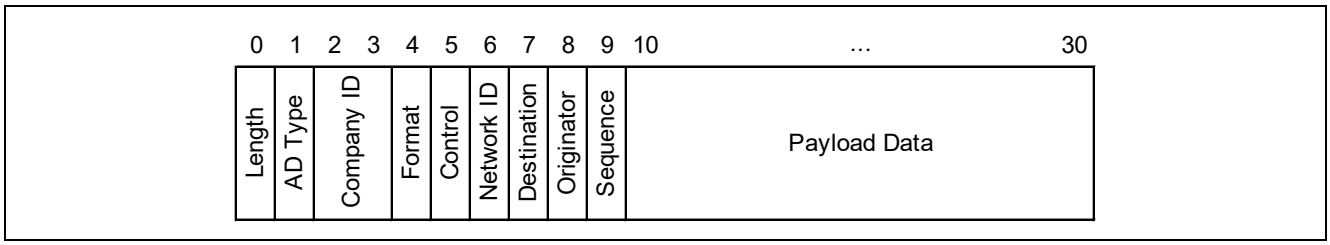


図 3-4　マルチホップフレームのフィールド構成

マルチホップフレームは Bluetooth Low Energy の Advertising パケットで、Manufacturer Specific Data として送信されます。Manufacturer Specific Data の仕様については『Supplement to the Bluetooth Core Specification | CSS Version 7』の Part A, Section 1.4 "MANUFACTURER SPECIFIC DATA"を参照してください。

前述のネットワーク ID は Network ID フィールドに格納され、発信ノード ID と宛先ノード ID はそれぞれ Originator、Destination フィールドに格納されます。また発信ノードが宛先ノードに伝送するデータは Payload Data フィールドに格納されます。

表 3-1　マルチホップフレームのフィールド構成

オフセット	サイズ (byte)	フィールド	説明
0	1	Length	AdvData 長(byte) AD Type フィールドから Payload Data フィールドまでの長さ
1	1	AD Type	AdvData 0xFF: <<Manufacturer Specific Data>>
2	2	Company ID	Bluetooth 企業 ID (LSO(Least Significant Octet) First) https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers
4	1	Format	マルチホップフレームのフォーマットバージョン 0x01: フレームフォーマットバージョン ver.1.00
5	1	Control	マルチホップ制御

ビット	フィールド	説明
0:3	hop limit	残り最大ホップ回数 0～15 中継時に 1 減算されてフレーム送信される hop limit が 0 のフレームは中継されない
4:5	counter	フレーム送信カウンタ 同一フレームの送信回数 0: フレーム送信 1 回目 1: フレーム送信 2 回目 2: フレーム送信 3 回目 ※フレーム到達率向上のため、発信ノードも中継ノードも同一フレームを 3 回ずつ送信する
6	encrypted	暗号化フラグ ペイロードデータの暗号化の有無 0: 暗号化なし 1: 暗号化あり (※R01AN4466 のみ対応)
7	optional	オプションデータフラグ マルチホップの解析などを行うためのオプションデータの有無 0: オプションデータなし 1: オプションデータあり

6	1	Network ID	ネットワーク ID 0x00～0xFF
7	1	Destination	宛先ノード ID 0x00～0xFE は個別ノードへの送信、0xFF は全ノードへの送信
8	1	Originator	発信ノード ID 0x00～0xFE のフレームを発信するノードの ID
9	1	Sequence	シーケンス番号 フレームを識別するための番号、同一フレームの重複受信判定に使用される 0x00～0xFF フレームの送信毎に 1 加算され、ラップアラウンドする
10	最大 21	Payload Data	ペイロードデータ

3.4.1　暗号化フレーム

図 3-5 と**表 3-2** に暗号化フレームの構成を示します。

データ暗号化時、マルチホップフレームの Payload Data フィールドは、Counter、Encrypted Data、MIC の 3 つのサブフィールドで構成されます。また Control フィールドの encrypted ビットには 1 がセットされます。

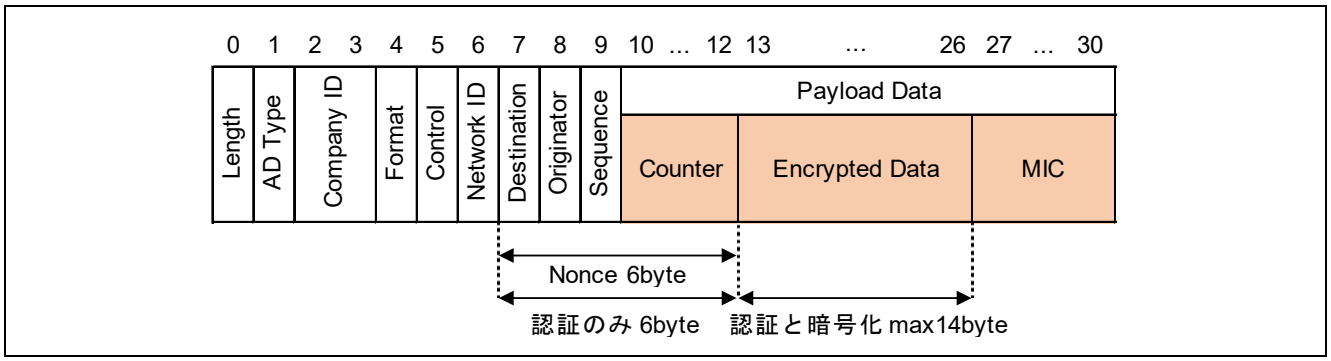


図 3-5　暗号化時の Payload Data のサブフィールド構成

表 3-2　暗号化時の Payload Data のサブフィールド構成

オフセット	サイズ (byte)	フィールド	説明
10	3	Counter	カウンタ値 Sequence と Counter の合計 4byte で LSO(Least Significant Octet)-First の Nonce カウンタ値を示す AES-CCM の Nonce として使用される Nonce カウンタ値は各ノードで独立した値を持ち、フレーム送信時に毎回インクリメントされる (Sequence がラップアラウンドすると Counter が 1 加算される)
13	最大 14	Encrypted Data	暗号化データ
最大 27	4	MIC	メッセージ完全性符号 Message Integrity Code

暗号化データのランダム性を確保するための Nonce の一部として、Destination から Counter までのフィールドが使用されます。

データ改竄を検出するための認証は、Destination から Encrypted Data までのフィールドが対象です。またデータ内容を秘匿するための暗号化は、Encrypted Data フィールドが対象です。

フレームの最後尾には、メッセージを認証するためのメッセージ完全性符号(MIC:Message Integrity Code)が付加されます。

3.4.2　オプションデータフレーム

マルチホップレイヤには通信解析のための機能として、フレームの中継経路を確認する機能(詳細は 9.1 節を参照)が実装されています。本確認機能は、下記に示すオプションデータフレームを使用します。

図 3-6、**図 3-7** と**表 3-3** にオプションデータフレームの構成を示します。

オプションデータを含むマルチホップフレームのデータフィールドは ID、Length、Option Data の 3 つのサブフィールドで構成されます。また Control フィールドの optional ビットには 1 がセットされます。

マルチホップレイヤはオプションデータフレームを受信すると、オプションデータ ID に従ってオプション機能を実行します。

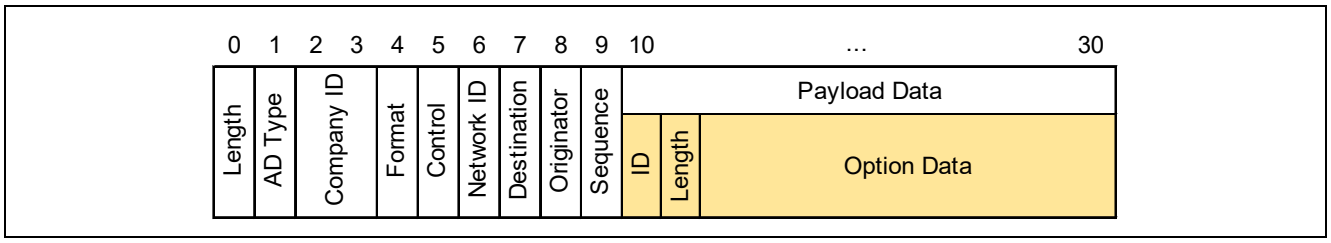


図 3-6　オプションデータフレームのサブフィールド構成

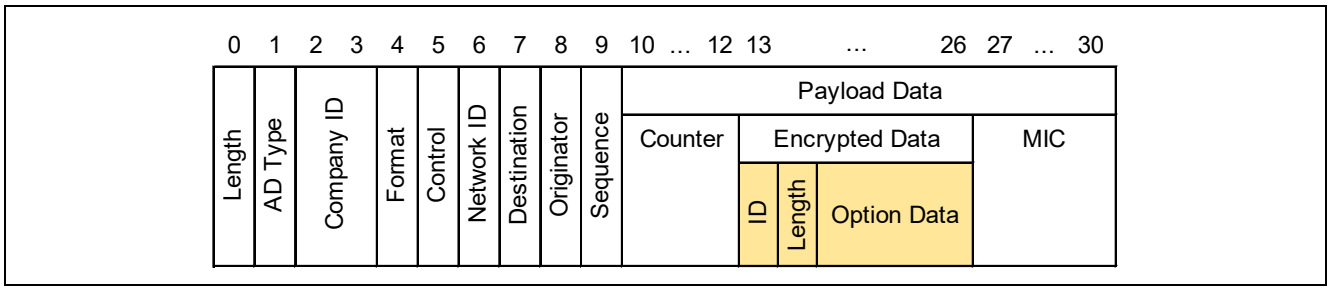


図 3-7　オプションデータフレームのサブフィールド構成(暗号化時)

表 3-3　オプションデータフレームのサブフィールド構成

オフセット	サイズ (byte)	フィールド	説明
10	1	ID	オプションデータ ID
11	1	Length	オプションデータ長
12	最大 19 (暗号化時は最大 12)	Option Data	オプションデータ

オプションデータの構造体定義については 8.3.5「オプションデータ構造体」を参照してください。

3.5　送受信動作

各ノードは、マルチホップフレームを Bluetooth Low Energy の Advertising 動作で送信し Scan 動作で受信します。**図 3-8** に例として、ID＝0～3 の各ノードが ID=3→ID=2→ID=1→ID=0 の経路でフレーム中継した場合の送受信動作を示します。

マルチホップフレームは、各ノードが以下のような送受信動作を実行することで、発信ノードから宛先ノードまで伝送されます。

1. 全ノードは、他ノード宛のフレーム中継と自ノード宛のフレーム受信のため、起動時に受信動作を開始します。
2. ID=3 のノードが ID=0 宛にフレームを送信します。

   このときフレームの到達率向上のため、本ノードは同一フレームを 3 回ずつ送信します。
3. ID=2 と ID=1 のノードは、ID=0 宛の新規フレームを受信すると、フレームを中継します。

   このときフレームの到達率向上のため、各ノードは受信したフレームを 3 回ずつ中継送信します。また送信タイミングの衝突低減のため、各ノードはランダム時間後にフレームを中継送信します。その後、同一フレームを再受信した場合は、重複フレームとして破棄し、中継されません。
4. ID=0 のノードは、ID=0 宛の新規フレームを受信すると、フレームデータがアプリケーションに通知されます。

   その後、同一フレームを再受信した場合は、重複フレームとして破棄し、アプリケーションには通知されません。

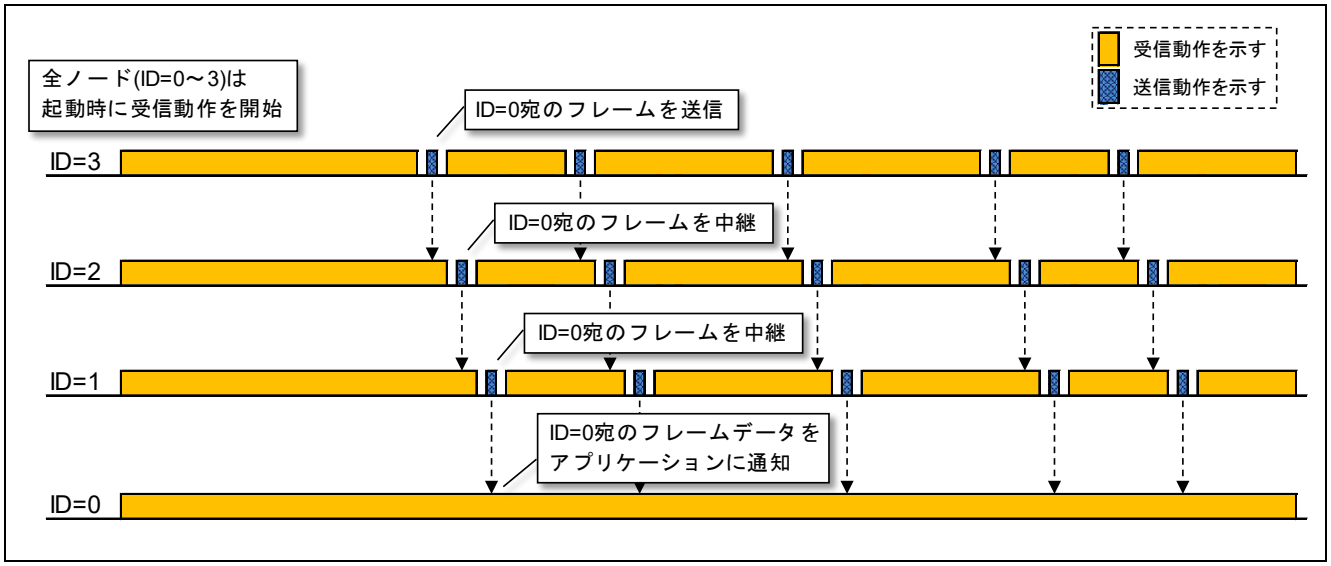


図 3-8　フレームの送受信動作例

送信動作ではフレームの到達率向上のため、全 Advertising チャネルに対してフレームを 3 回送信します。詳細は 8.8 節「フレーム送信動作」を参照してください。

受信動作では Advertising チャネルを周期的に切り替えて受信します。詳細は 8.7 節「フレーム受信動作」を参照してください。

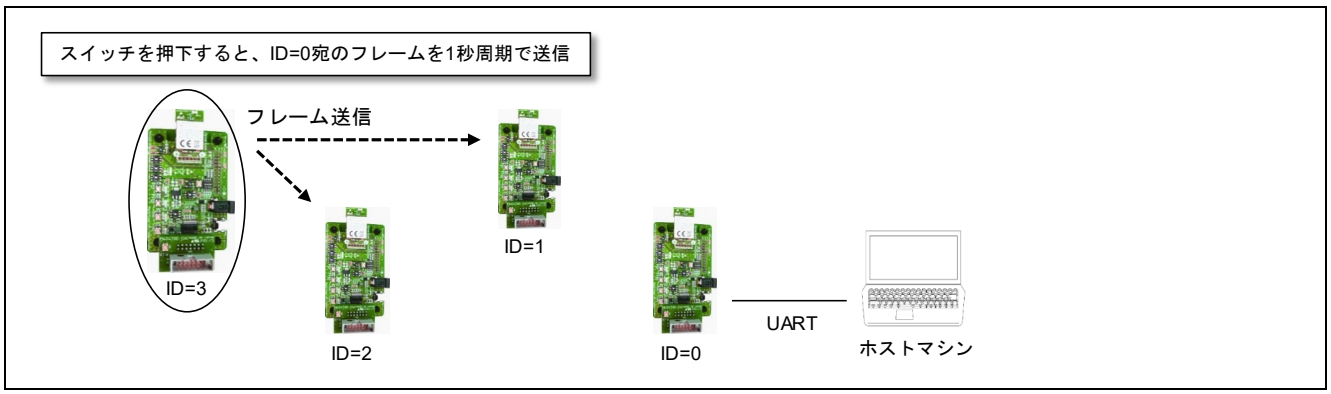
4. アプリケーションレイヤ仕様

本章ではサンプルプログラムのアプリケーションレイヤの仕様を示します。サンプルプログラムの操作方法については 5 章「操作方法」を参照してください。

4.1 通信動作

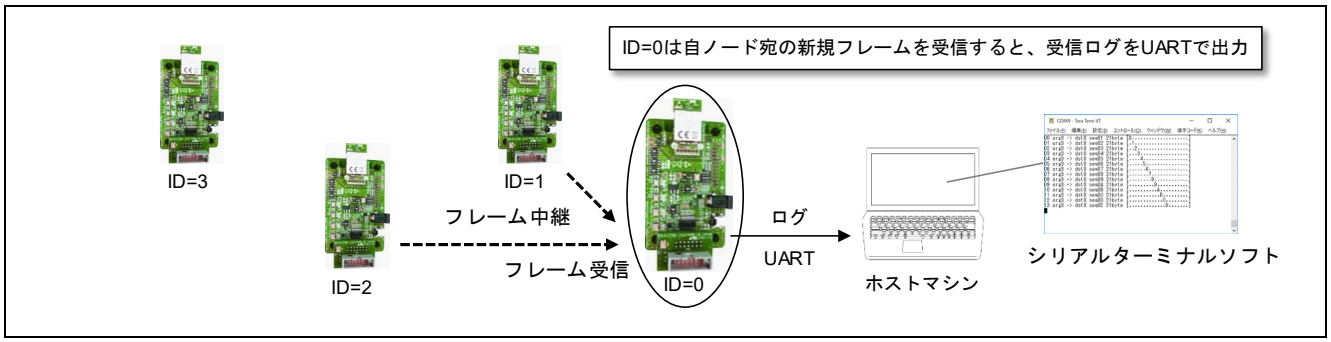
サンプルプログラムを書き込んだ評価ボードを起動すると、各評価ボードはフレーム受信動作を開始します。また各評価ボード上のスイッチを押下すると、ID=0 に対してフレームを送信します。

例として、ID＝0～3 までの評価ボードがあるとします。ID=3 の評価ボード上のスイッチを押下すると、ID=3 は ID=0 に対してフレームを 1 秒周期で送信します。このとき、1 秒おきに送信されるフレームは異なるシーケンス番号を持ちます。



ID=1～2 のノードは他ノード宛のフレームを受信すると、受信したフレームを中継します。なおフレームの中継はマルチホップレイヤが行い、アプリケーションレイヤにはフレームの中継は通知されません。

ID=0 は自ノード宛のフレームを受信すると、受信したフレームデータのログを UART で出力します。その後、同一シーケンス番号のフレームを再受信した場合は、重複フレームとして破棄し、ログは出力されません。



4.2　フレームデータ

スイッチ SW2 を押下すると、アプリケーションは ID=0 に対して周期的にフレームを送信します。

送信するデータは、下記のソースコードファイルに配列として実装されています。サンプルプログラムは本データを順番に繰り返し送信します。

- Project_Source¥application¥src¥r_node.c

r_node.c (line.101-119)

```
101:    static RMH_DATA demo_payload_data[] =
102:    {
103:        {{"0.....................", 21, 0},
104:        {{".1....................", 21, 0},
105:        {{"..2...................", 21, 0},
106:        {{"...3..................", 21, 0},
107:        {{"....4.................", 21, 0},
108:        {{".....5................", 21, 0},
109:        {{"......6...............", 21, 0},
110:        {{".......7..............", 21, 0},
111:        {{"........8.............", 21, 0},
112:        {{".........9............", 21, 0},
113:        {{"..........A...........", 21, 0},
114:        {{"...........B..........", 21, 0},
115:        {{"............C.........", 21, 0},
116:        {{".............D........", 21, 0},
117:        {{"..............E......", 21, 0},
118:        {{"...............F.....", 21, 0},
119:    };
```

node: 宛先ノード ID(0x00～0xFF)

data: フレームデータ

len: フレームデータサイズ(最大 21byte)

また ID=0 のノードは受信したフレームデータのログを下記のフォーマットで UART から出力します。

通し番号 ORG 番号 -> *DST 番号 SEQ 番号 フレームデータサイズ* [*フレームデータ(ASCII)*]

例として、ID=3 からフレームを送信します。ID=0 はフレームを受信すると、**図 4-1** のようなデータログを出力します。本ログにより、ID=3 が送信したフレームを ID=0 が順番に受信したことが確認できます。

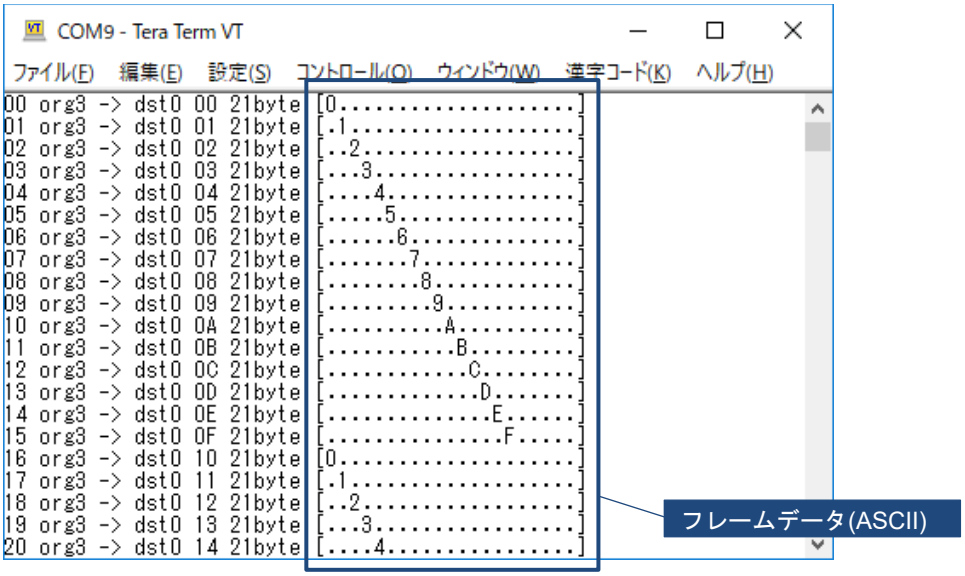


図 4-1　受信フレームのデータログ例

セキュリティ機能の有効時に送信するデータも、下記のソースコードファイルに配列として実装されています(※R01AN4466 のみ)。サンプルプログラムは本データを暗号化フレームで順番に繰り返し送信します。

- Project_Source¥application¥src¥r_node.c

r_node.c (line.125-141)

```
125:    static RMH_DATA demo_sec_data[] =
126:    {
127:        {{"0............"}, 14, 0},
128:        {{".1............"}, 14, 0},
129:        {{"..2............"}, 14, 0},
130:        {{"...3............"}, 14, 0},
131:        {{"....4............"}, 14, 0},
132:        {{".....5............"}, 14, 0},
133:        {{"......6............"}, 14, 0},
134:        {{".......7............"}, 14, 0},
135:        {{"........8............"}, 14, 0},
136:        {{".........9............"}, 14, 0},
137:        {{"..........A............"}, 14, 0},
138:        {{"...........B............"}, 14, 0},
139:        {{"............C............"}, 14, 0},
140:        {{".............D............"}, 14, 0},
141:    };
```

node: 宛先ノード ID(0x00～0xFF)

data: フレームデータ

len: フレームデータサイズ(最大 14byte)

例として、ID=0 から暗号化フレームを送信します。同梱されたスキャンプログラム(5.4 節を参照)で受信すると、**図 4-2** のようなフレームログを出力します。フレームデータが暗号化されているため、内容を解読することはできません。

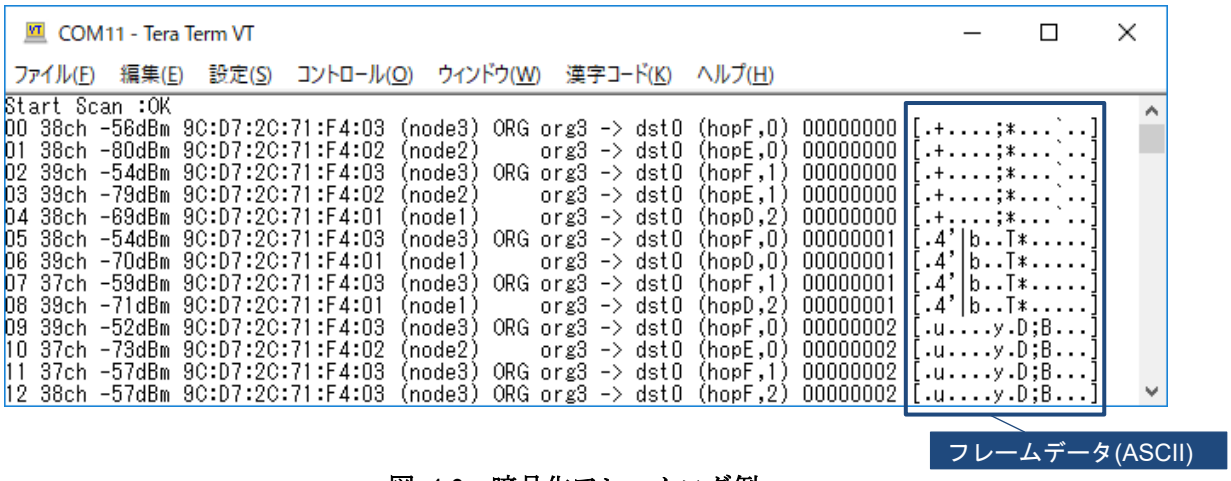


図 4-2　暗号化フレームログ例

また ID=0 は暗号化フレームを受信すると、**図 4-3** のようなデータログを出力します。

本ログにより、暗号化フレームのデータが正しく復号されたことが確認できます。

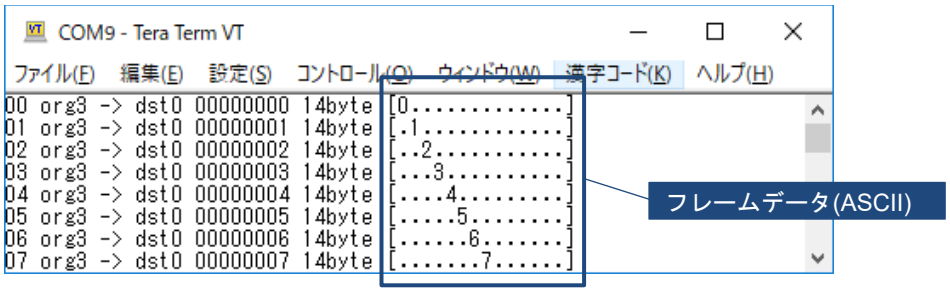


図 4-3　暗号化フレームのデータログ例

4.3　システム動作設定

マルチホップを構成する各ノードデバイスには、各デバイスを識別するための異なるノード ID を割り当てる必要があります。そこでサンプルプログラムでは、各デバイスに異なるパラメータ（システム動作設定）を設定する仕組みを提供します。

図 4-4 にシステム動作設定の概要を示します。

システム動作設定の書き込みには、フラッシュ書き込みソフトウェア Renesas Flash Programmer のユニークコード機能を利用します。またシステム動作設定は、ユニークコードと呼ばれるデータファイルに記述します。

Renesas Flash Programmer でフラッシュ書き込みを実行すると、全デバイスで共通のファームウェアと、各デバイスで異なるシステム動作設定を順番に書き込むことができます。

サンプルプログラムを実行すると、ファームウェアはシステム動作設定からノード ID といったパラメータを読み込み、マルチホップの設定値として使用します。

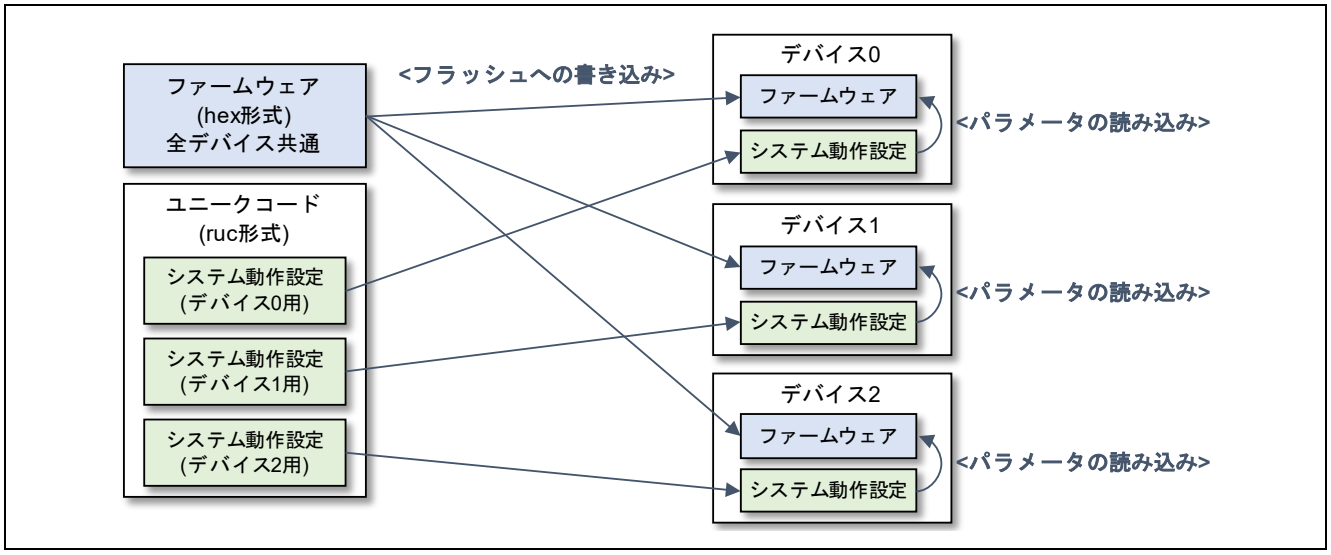


図 4-4　システム動作設定

表 4-1 にシステム動作設定の先頭アドレスを示します。

システム動作設定は、コードフラッシュメモリのプログラム外の領域に配置されます。ソースコードを変更することにより、システム動作設定の先頭アドレスは変更できます。

表 4-1　システム動作設定の先頭アドレス

RL78/G1D	配置アドレス
R5F11AGG	0x1FC00
R5F11AGH	0x2FC00
R5F11AGJ	0x3F400

表 4-2 にシステム動作設定の仕様を示します。ソースコードを変更することにより、システム動作設定を構成するパラメータは追加や変更ができます。

※RL78/G1D モジュール(RY7011)をご使用の場合、サンプルプログラムはシステム動作設定のデバイスアドレスを無視し、出荷時にコードフラッシュの Block255 に書き込まれたデバイスアドレスを優先して使用します。

表 4-2　システム動作設定の構成

オフセット	サイズ (byte)	データ
0	6 (RBLE_BD_ADDR 構造体)	デバイスアドレス ※1
6	1 (uint8_t 型)	デバイスアドレスタイプ: 0x00: public, 0x01: random
7	1 (uint8_t 型)	ネットワーク ID 0x00～0xFF
8	1 (uint8_t 型)	ノード ID 0x00～0xFE
9	1 (bool 型)	セキュリティフラグ 0x01: セキュリティ機能の有効化 (※R01AN4466 のみ対応) 0x00: セキュリティ機能の無効化
10	16 (uint8_t[16]型)	128bit 暗号鍵 ※セキュリティ機能の無効時は不要

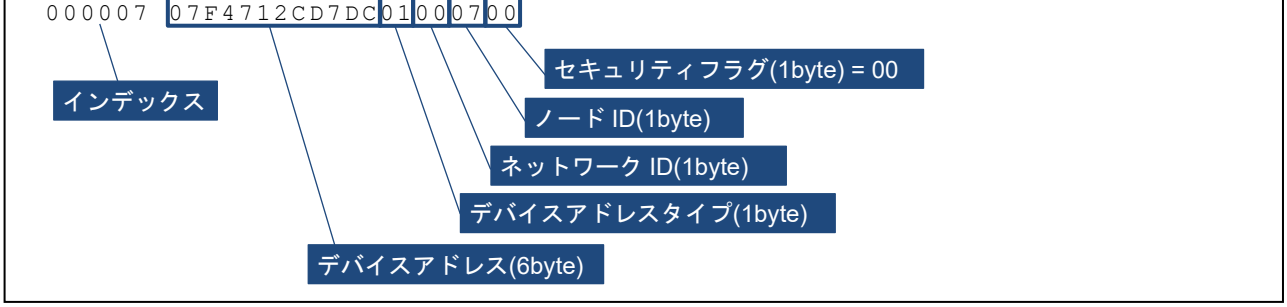
システム動作設定が記述されたユニークコードファイルのパスを以下に示します。

- RUC_File¥r5f11agj_syscfg.ruc

ユニークコードファイルの内容を以下に示します。各インデックスのシステム動作設定データが、各 RL78/G1D に順番に書き込まれます。なおユニークコードファイルはテキストエディタで編集できます。

r5f11agj_syscfg.ruc

```
// ---------------------------------------------------------------
// -- System Configuration for RL78/G1D Multi Hop Sample Program --
// --   Device Part Number : R5F11AGJ                           --
// ---------------------------------------------------------------
format hex
area user flash
address 0x3f400
size 10
index data
//     |----------|       uint8_t[6]: Device Address (LSB-first)
//                ||      uint8_t: Device Address Type (00:Public, 01:Random)
//                 ||     uint8_t: Network ID (0x00 to 0xFF)
//                  ||    uint8_t: Node ID (0x00 to 0xFE)
//                   ||   bool: Security Flag (00:disable, 01:enable)
000000 00F4712CD7DC010000
000001 01F4712CD7DC010001
000002 02F4712CD7DC010002
000003 03F4712CD7DC010003
000004 04F4712CD7DC010004
000005 05F4712CD7DC010005
000006 06F4712CD7DC010006
000007 07F4712CD7DC010007
```



セキュリティ機能を有効にするシステム動作設定が記述されたユニークコードファイルのパスを以下に示します(※R01AN4466 のみ同梱)。

- RUC_File¥r5f11agj_syscfg_sec.ruc

ユニークコードファイルの内容を以下に示します。前述の r5f11agj_syscfg.ruc に対する差分として、セキュリティフラグに 1 に設定され、最後尾に 128bit 暗号鍵が付加されています。

r5f11agj_syscfg_sec.ruc

```
// ---------------------------------------------------------------
// -- System Configuration for RL78/G1D Multi Hop Sample Program --
// --   Device Part Number : R5F11AGJ                           --
// ---------------------------------------------------------------
format hex
area user flash
address 0x3f400
size 10
index data
//    |----------|       uint8_t[6]: Device Address (LSB-first)
//               ||      uint8_t: Device Address Type (00:Public, 01:Random)
//                ||     uint8_t: Network ID (0x00 to 0xFF)
//                 ||    uint8_t: Node ID (0x00 to 0xFE)
//                  ||   bool: Security Flag (00:disable, 01:enable)
//               |-------------------------------| uint8_t[16]: Encryption Key (128bit)
000000 00F4712CD7DC010000 01010203040506070809 0A0B0C0D0E0F10
000001 01F4712CD7DC010001 01010203040506070809 0A0B0C0D0E0F10
000002 02F4712CD7DC010002 01010203040506070809 0A0B0C0D0E0F10
000003 03F4712CD7DC010003 01010203040506070809 0A0B0C0D0E0F10
000004 04F4712CD7DC010004 01010203040506070809 0A0B0C0D0E0F10
000005 05F4712CD7DC010005 01010203040506070809 0A0B0C0D0E0F10
000006 06F4712CD7DC010006 01010203040506070809 0A0B0C0D0E0F10
000007 07F4712CD7DC010007 01010203040506070809 0A0B0C0D0E0F10
```

暗号鍵(16byte)

セキュリティフラグ(1byte) = 01

4.4　シーケンス

図 4-5 にサンプルプログラムのアプリケーションレイヤのシーケンスを示します。

サンプルプログラムを起動すると、初期化関数 node_init()が R_MH_Init()をコールしてマルチホップを初期化し、さらに R_MH_Receive()をコールしてフレーム受信動作を開始します。

スイッチを押下すると、外部入力割り込み INTP5 が発生し、割り込みハンドラ input_callback()が R_MH_Send()をコールしてフレームを送信します。またここで 12 ビット・インターバル・タイマを起動します。

さらに 12 ビット・インターバル・タイマからのインターバル割り込み INTIT が発生して 1 秒周期となる毎に、割り込みハンドラ it_callback()が R_MH_Send()をコールしてフレームを送信します。

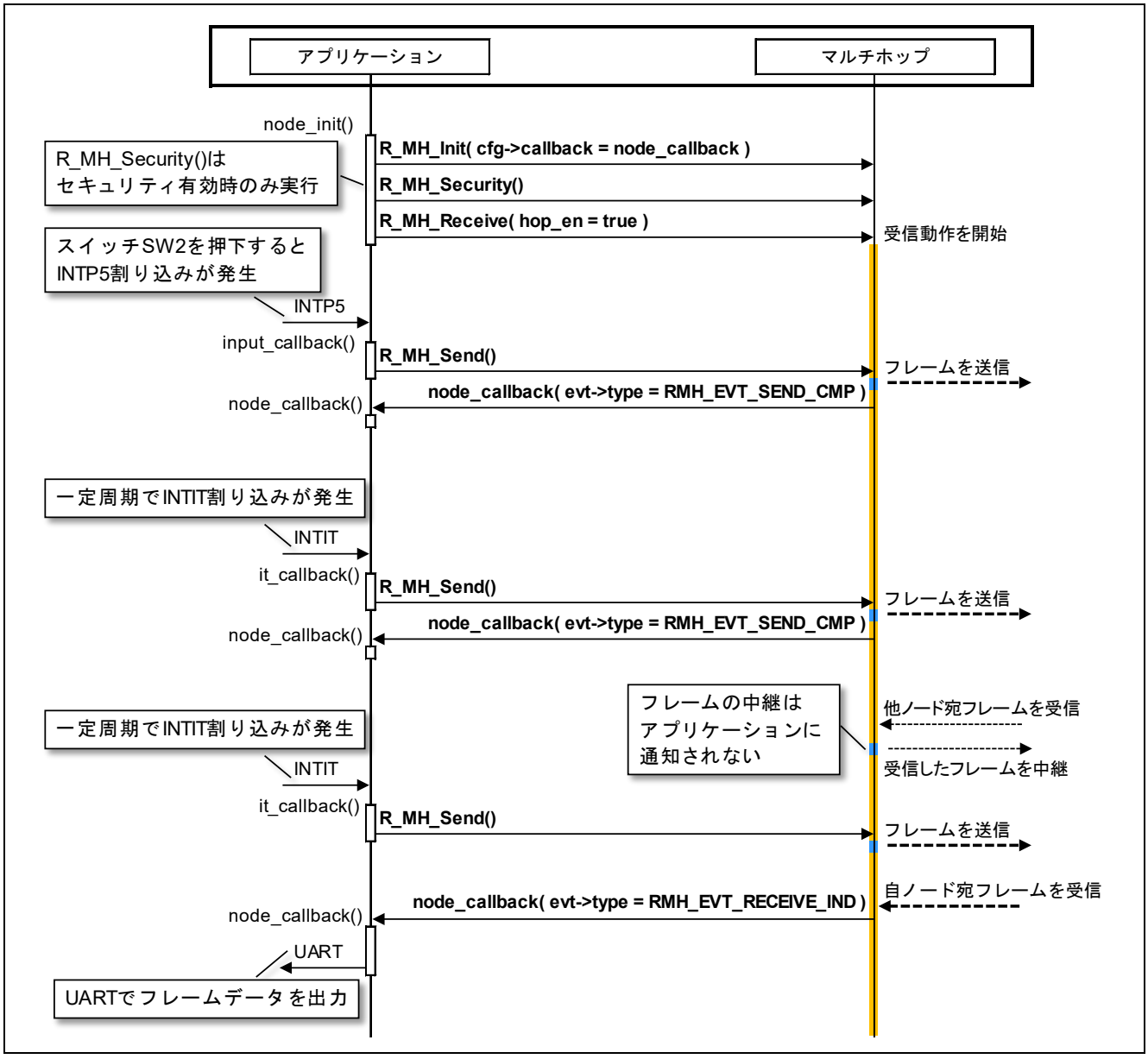


図 4-5　サンプルプログラムのシーケンス

マルチホップレイヤの API 仕様については 8 章「マルチホップレイヤ API」を参照してください。

またマルチホップレイヤがコールバック関数 node_callback()によるイベントの通知と、受信した他ノード宛のフレームの中継を実行するため、アプリケーションレイヤは R_MH_Proc()を定期的に実行する必要があります。詳細は 8.4.2 項「R_MH_Proc」を参照してください。

5. 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。

5.1 動作環境

サンプルプログラムのファームウェアの書き込みには、フラッシュ書き込みソフトウェア Renesas Flash Programmer を使用します。

Renesas Flash Programmer (Programming GUI)
https://www.renesas.com/software-tool/renesas-flash-programmer-programming-gui

サンプルプログラムの動作確認に必要な環境を示します。

- 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) ：2 台以上
    - USB ケーブル(A タイプ オス / mini-B タイプ オス) ：2 本以上
  - 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 Flash Programmer V3.04.00**
    - 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 (※)

※評価ボードとホストマシンを接続する際に、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。その際に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

5.2　スライドスイッチの設定

図 5-1 に RL78/G1D 評価ボードのスライドスイッチを示します。

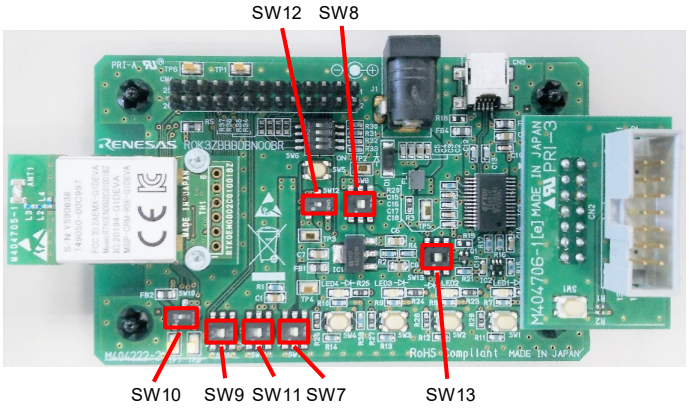


図 5-1　RL78/G1D 評価ボードのスライドスイッチ

表 5-1 にサンプルプログラムを動作確認する際の評価ボードのスライドスイッチ設定を示します。

表 5-1　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)	(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。

5.3　サンプルプログラムの書き込み

図 5-2 にサンプルプログラムの書き込み方法を示します。

サンプルプログラム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 Reneas Flash Programmer を実行します。また Renesas Flash Programer のユニークコード機能を利用することで、RL78/G1D 評価ボードに異なる ID を設定することができます。

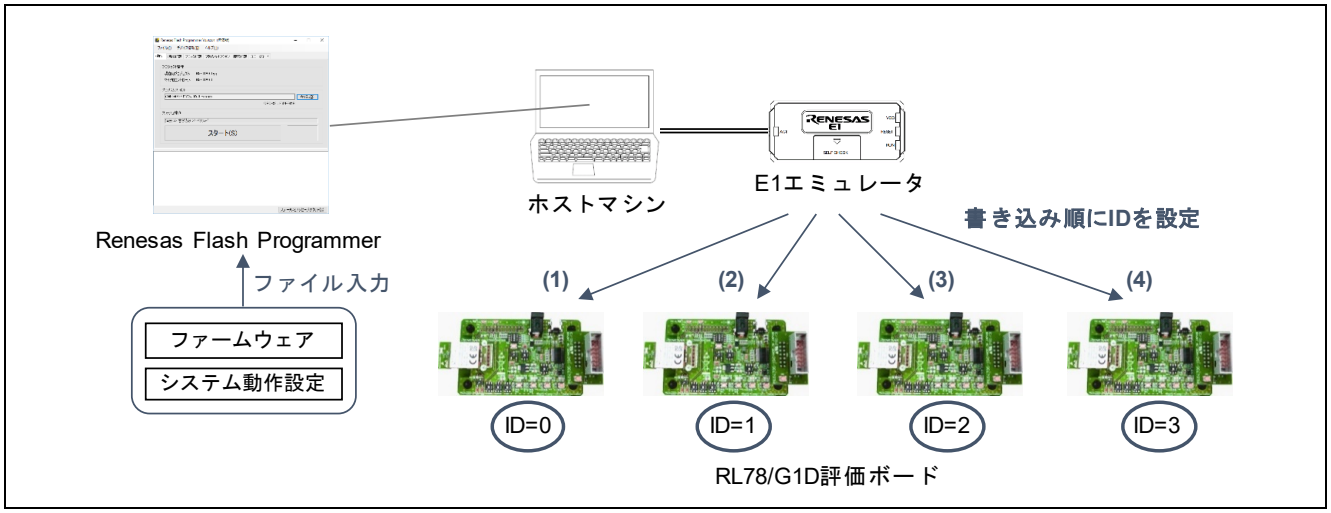


図 5-2　ファームウェア書き込み時の評価ボード接続

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。

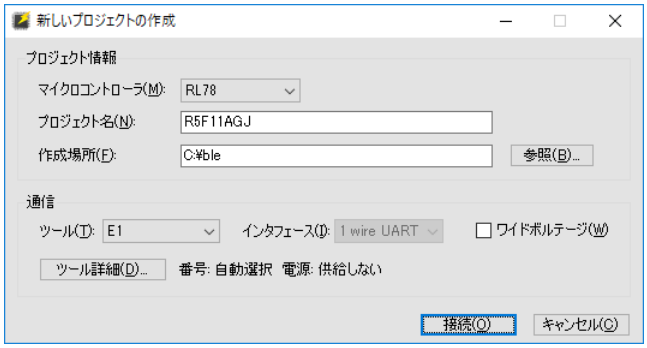
サンプルプログラムファームウェアの RL78/G1D 評価ボードへの書き込み手順を以下に示します。

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。

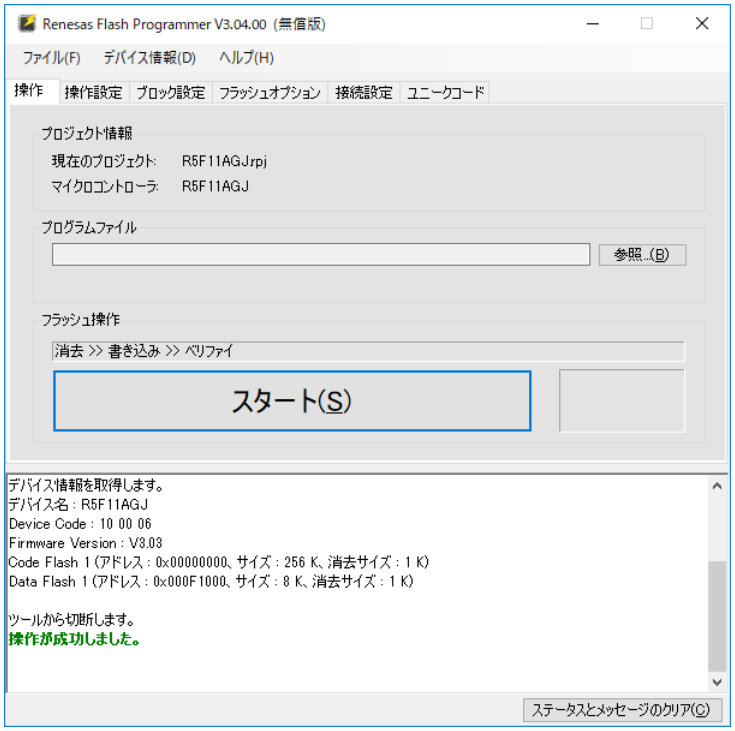
   ※プロジェクト作成後は、作成したプロジェクトを使用することで次回から本手順を省略可能です。

   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。



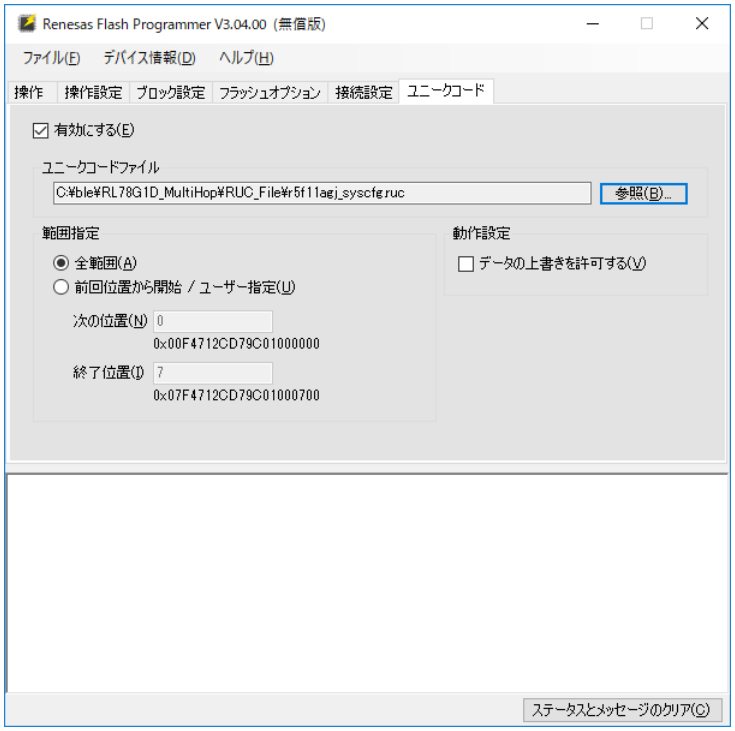
4. 下記の手順でシステム動作設定が記述されたユニークコードファイルを指定します。

4-1. [ユニークコード設定]タブを選択します。

4-2. [有効にする]にチェックを入れます。

4-3. [ユニークコードファイル]で下記のユニークコードファイルを指定します。

- (R01AN4375) RUC_File¥r5f11agj_syscfg.ruc
- (R01AN4466) RUC_File¥r5f11agj_syscfg_sec.ruc



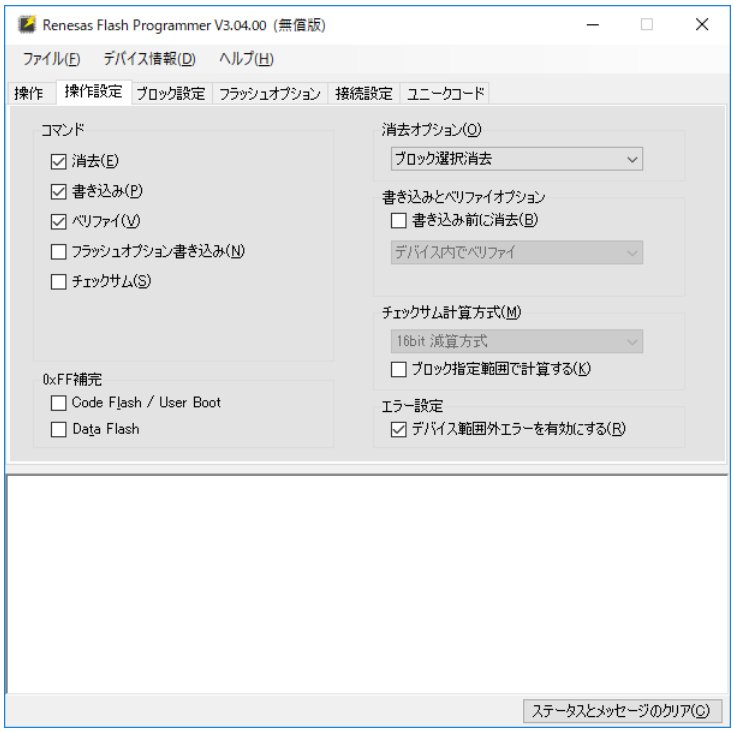
システム動作設定については 4.3 項「システム動作設定」を参照してください。

ユニークコード機能については『Renesas Flash Programmer V3.04 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』(R20UT4206)の 2.3.6 項「[ユニークコード]タブ」を参照してください。

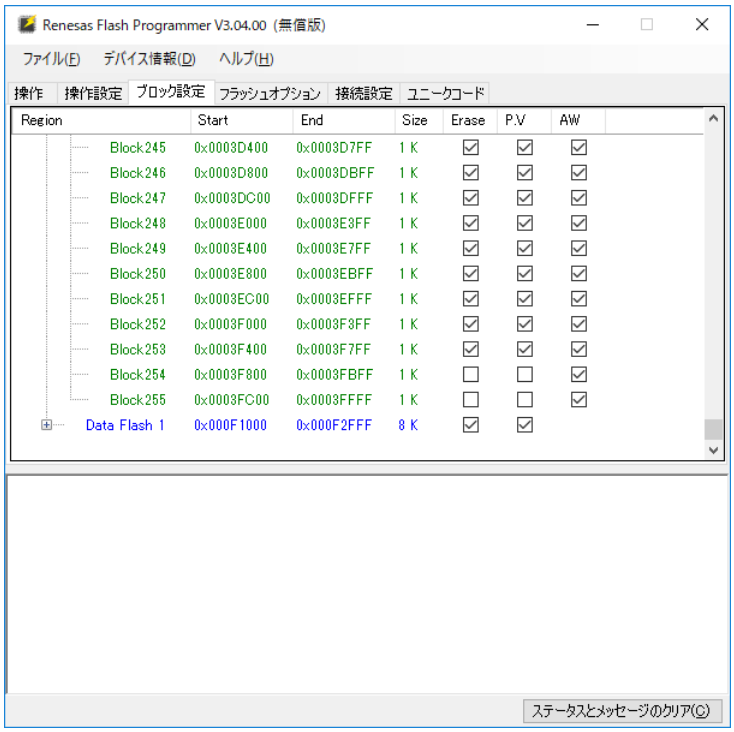
5. 下記の手順でコードフラッシュメモリの Block254,255 消去を防止します。

※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。

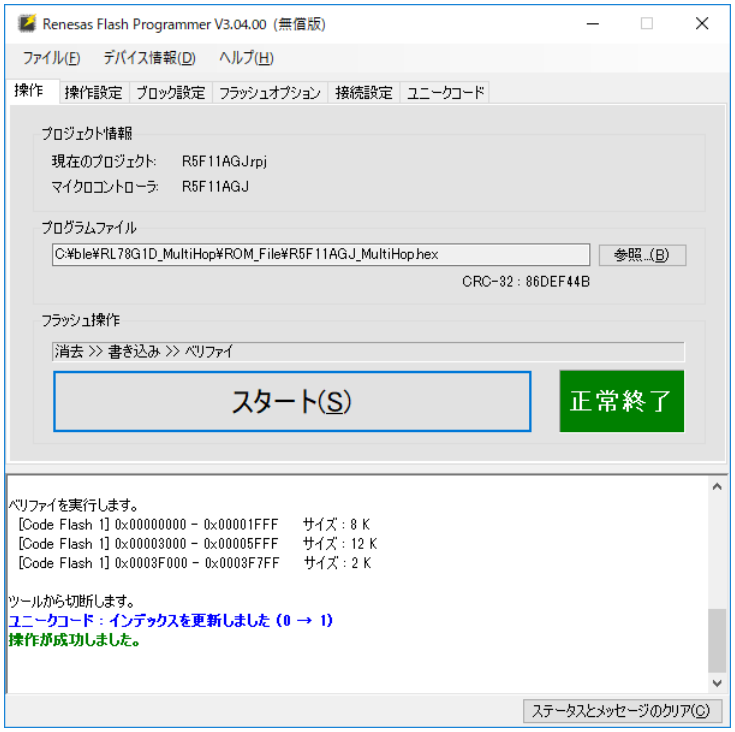
5-1. [操作設定]タブを選択し、[消去オプション]で[ブロック選択消去]を選択します。



5-2. [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。



6. [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。
   - (R01AN4375) ROM_File¥R5F11AGJ_MultiHop.hex
   - (R01AN4466) ROM_File¥R5F11AGJ_MultiHopSEC.hex
7. [スタート]を押下して書き込みを実行します。
8. 「操作が成功しました」と表示されることを確認します。



9. 電源、E1 エミュレータを評価ボードから取り外します。
10. 評価に使用する次の評価ボードに E1 エミュレータと電源を接続し、手順 7 から再度実行します。これを全ての評価ボードへの書き込みが完了するまで繰り返します。

5.4　マルチホップ通信の実行

図 5-3 にサンプルプログラムの概要を示します。ID=0 以外の評価ボードでマルチホップ送受信を実行し、ID=0 の評価ボードにフレームが到達することを確認します。

なおサンプルプログラムにはデモ用として、特定のデバイスが送信したフレームのみ受信するためのデバイスフィルタが実装されています。デバイスフィルタを使用すると、ID=3－ID=2－ID=1－ID=0 といった特定の中継経路に限定したフレーム送受信動作を確認できます。詳細は 9.2 節「デバイスフィルタ」を参照してください。

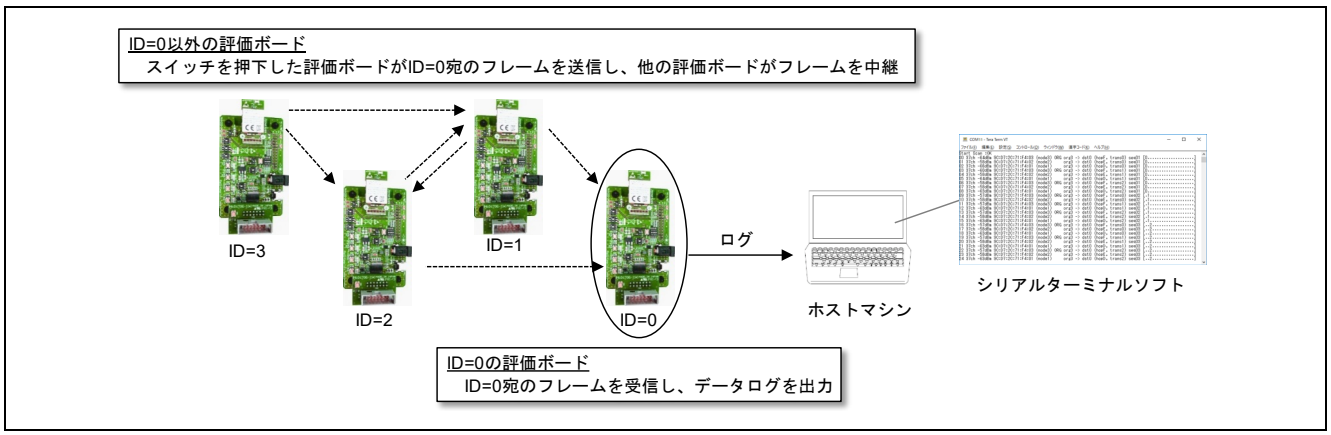


図 5-3　サンプルプログラムの概要

サンプルプログラムのフレーム送受信の実行手順を以下に示します。

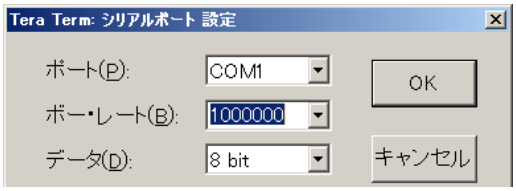
ID=0 の評価ボード

1. ID=0 の評価ボードとホストマシンを USB ケーブルで接続します。また USB から電源供給します。
2. シリアルターミナルソフトを起動し、**表 5-2** を参照して評価ボードのシリアルポートと接続します。

表 5-2　ターミナルソフトのシリアルポート設定

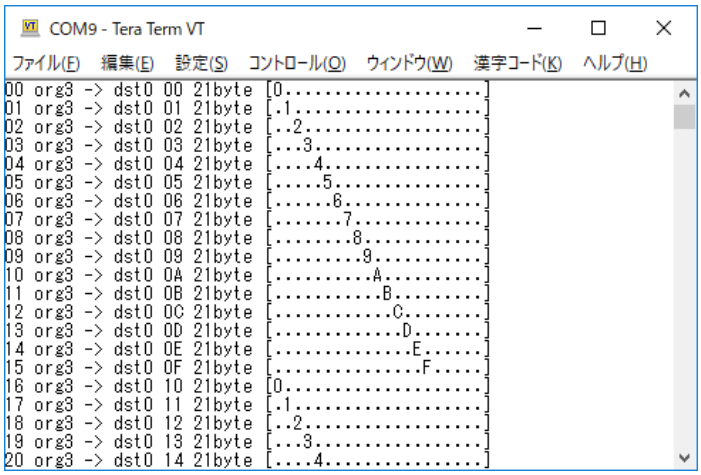
設定項目		設定値
シリアルポート	ポート	USB シリアルポート ※COM 番号は評価ボードごとに異なる
	ボーレート	1,000,000bps
	データ長	8bit
	パリティ	None
	ストップビット	1bit
	フロー制御	None
改行コード	受信	LF
	送信	LF
端末サイズ	横幅	128 文字以上

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 は含まれていません。「ボー・レート」欄に直接"1000000"と入力してください。



ID=0 以外の評価ボード

1. ID=0 以外の評価ボードに 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。
2. まずは 1 台の評価ボードのスイッチ SW2 を押下します。例として ID=3 のスイッチを押下します。
3. スイッチを押下した評価ボードから ID=0 宛のフレームが 1 秒ごとに送信され、ID=0 の評価ボードがフレームを受信すると、1 秒おきに下記のようなログがターミナルソフトに表示されます。



※評価ボードを使用しない等の状況でスイッチ SW2 を使用できない場合は、下記のファームウェアを ID=0 以外の RL78/G1D、例として ID=3 に書き込んでください。本ファームウェアは起動すると、スイッチを押下することなく、1 秒周期のフレーム送信を自動で開始します。

- ROM_File¥R5F11AGJ_MultiHop(NO_SW).hex

上記のファームウェアは下記ファイルの TEST_NO_SW マクロを(1)に変更してビルドしたものです。

- Project_Source¥application¥src¥r_node.c

r_node.c (line.57-59)

```
54:    /* 1: start to sends frame immediately after application initialization */
55:    /* 0: start to sends frame when switch is pushed */
56:    #define TEST_NO_SW            (0)
                                     (1)に変更
```

サンプルプログラムのパッケージには、マルチホップフレームをキャプチャするためのスキャンプログラムが含まれます。

下記のファームウェアを評価ボードに書き込み、ID=0 と同様の手順で実行すると、本評価ボードが受信したマルチホップフレームのログを確認することができます。

- ROM_File¥R5F11AGJ_Scan.hex

図 5-4 にスキャンプログラムの概要を示します。スキャンプログラムが出力するログの確認には、ID=0 のログ確認手順と同様にターミナルソフトを使用します。

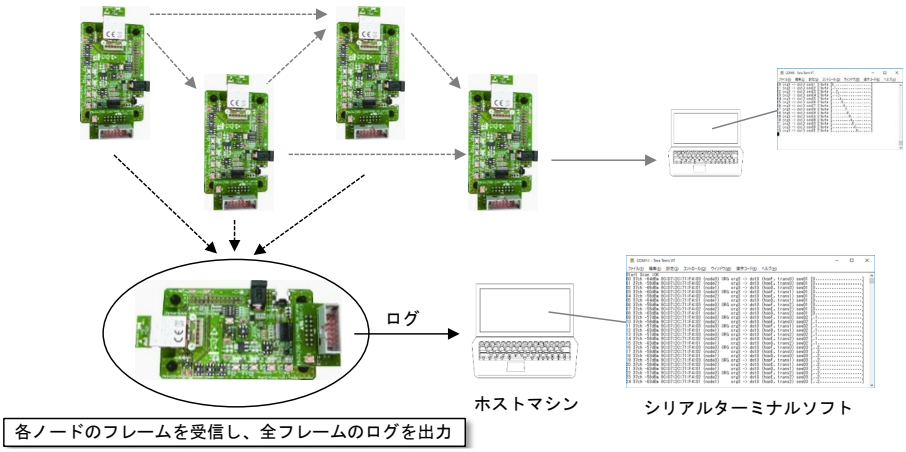


図 5-4　スキャンプログラムの概要

スキャンプログラムは受信したフレームのログを下記のフォーマットで UART から出力します。

通し番号 *チャネル* *RSSI* *デバイスアドレス* *デバイスアドレスタイプ* (*ノード ID*) *ORG 判定*
ORG 番号 -> *DST 番号* (*ホップリミット*, *送信回数*) *SEQ 番号* *フレームデータサイズ* [*データ(ASCII)*]

図 5-5 にスキャンプログラムが出力するフレームログの例を示します。

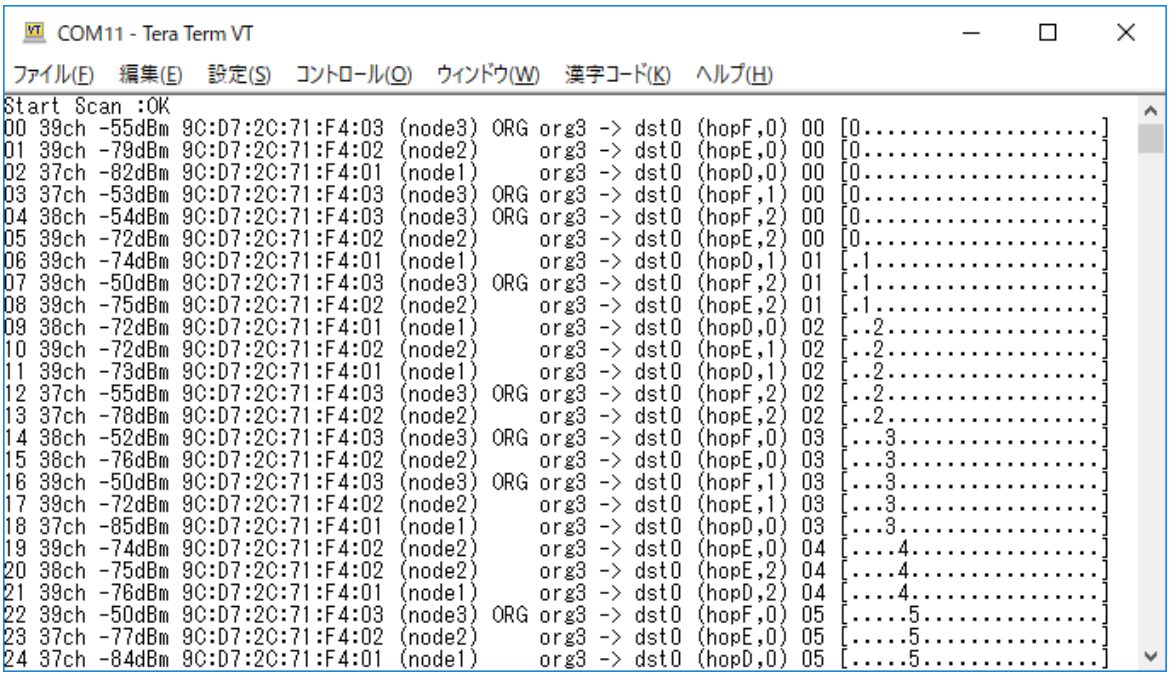


図 5-5　スキャンプログラムのフレームログ例

6. 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。

6.1 開発環境

プログラムの開発とビルドには、統合開発環境 CS+ for CC を使用します。

統合開発環境 CS+
https://www.renesas.com/software-tool/cs

サンプルプログラムのビルドに必要な開発環境を示します。

- 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) ：2 台以上
    - USB ケーブル(A タイプ オス / mini-B タイプ オス) ：2 本
  - 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 V3.04.00
    - 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 (※)

※評価ボードとホストマシンを接続する際に、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。その際に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

サンプルプログラムのパッケージには下記のライブラリが含まれます(※R01AN4466 のみ)。これらのライブラリは、セキュリティ機能に対応したサンプルプログラムのビルドに必要となります。

AES ライブラリ：RL78 ファミリ用 AES ライブラリ V1.05 Release 00

データフラッシュライブラリ：EEPROM Emulation Library Pack02 for CC-RL Compiler Ver1.01

6.2　ファイル構成

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以下に示します。

```
RL78G1D_MultiHop
  ├ROM_File
  │      R5F11AGJ_MultiHop.hex                          サンプルプログラムファームウェア(※R01AN4375 のみ)
  │      R5F11AGJ_MultiHop(DEV_FILTER).hex                (DEV_ADDR_FILTER=1)
  │      R5F11AGJ_MultiHop(NO_SW).hex                     (TEST_NO_SW=1)
  │      R5F11AGJ_MultiHopSEC.hex                        サンプルプログラムファームウェア(※R01AN4466 のみ)
  │      R5F11AGJ_MultiHop(DEV_FILTER).hex                (DEV_ADDR_FILTER=1)
  │      R5F11AGJ_MultiHop(NO_SW).hex                     (TEST_NO_SW=1)
  │      R5F11AGJ_Scan.hex                               スキャンプログラムファームウェア
  ├RUC_File
  │      r5f11agj_syscfg.ruc                            システム動作設定
  │      r5f11agj_syscfg_sec.ruc                        システム動作設定(※R01AN4466 のみ)
  └Project_Source
      ├library
      │      r_arch.h
      │      r_compiler.h
      │      r_iodefine.h
      │      r_ll.h                                     ビーコンスタックライブラリ
      │      r_port.h
      │      r_bcn_api.h
      │      BLE_BEACON_CC.lib
      └application
          ├src
          │   │  cstart.asm                             スタートアップルーチン
          │   │  r_config.h                             ビーコンスタックコンフィグレーション
          │   │  r_interrupt.c                          ビーコンスタック割り込みハンドラ
          │   │  r_main.c                               アプリケーションエントリポイント
          │   │  r_multihop.c                           マルチホップレイヤ
          │   │  r_multihop.h
          │   │  r_node.c                               ノードアプリケーション
          │   │  r_node.h
          │   │  r_utility.c                            ユーティリティ
          │   │  r_utility.h
          │   ├aes
          │   │   │  P_AesProto.h                       AES-CCM 変換
          │   │   │  P_Ccmz.c                           (※R01AN4466 のみ)
          │   │   └library
          │   │          r_aes.h
          │   │          r_mw_version.h                 AES ライブラリ
          │   │          r_stdint.h                     (※R01AN4466 のみ)
          │   │          aes_rl78_s2_m_ccrl.lib
          │   └driver
          │       ├dataflash
          │       │   │  r_dataflash.c                  データフラッシュドライバ
          │       │   │  r_dataflash.h                  (※R01AN4466 のみ)
          │       │   │  r_eel_descriptor_t02.c         EEPROM エミュレーションライブラリデスクリプタ
          │       │   │  r_eel_descriptor_t02.h         (※R01AN4466 のみ)
          │       │   │  r_fdl_descriptor_t02.c         データフラッシュライブラリデスクリプタ
          │       │   │  r_fdl_descriptor_t02.h         (※R01AN4466 のみ)
          │       │   └library
          │       │          eel.h
          │       │          eel.lib                     EEPROM エミュレーションライブラリ
          │       │          eel_types.h                 (※R01AN4466 のみ)
          │       │          fdl.h
          │       │          fdl.lib                     データフラッシュライブラリ
          │       │          fdl_types.h                 (※R01AN4466 のみ)
          │       ├input
          │       │   r_input.c                          外部入力割り込みドライバ
          │       │   r_input.h
```

```
│       ├it                                  -
│       │   r_it.c                           ┐ 12 ビット・インターバル・タイマドライバ
│       │   r_it.h                           ┘
│       ├led
│       │   r_led.c                          ┐ LED ドライバ
│       │   r_led.h                          ┘
│       ├plf
│       │   r_plf.c                          ┐ プラットフォームドライバ
│       │   r_plf.h                          ┘
│       ├timer
│       │   r_timer.c                        ┐ タイマ・アレイ・ユニットドライバ
│       │   r_timer.h                        ┘
│       │   r_timerx.c                       インデックスタイマ機能
│       └uart
│           r_log.c                           ログ送信機能
│           r_uart.c                          ┐ UART ドライバ
│           r_uart.h                          ┘
└project
    └cs_cc
        └BLE_Software
            │   BLE_Software.mtpj              CS+ for CC プロジェクトファイル
            ├R5F11AGJ_MultiHop
            │       R5F11AGJ_MultiHop.mtsp     CS+ for CC サブプロジェクトファイル(※R01AN4375 のみ)
            └R5F11AGJ_MultiHopSEC
                    R5F11AGJ_MultiHopSEC.mtsp  CS+ for CC サブプロジェクトファイル(※R01AN4466 のみ)
```

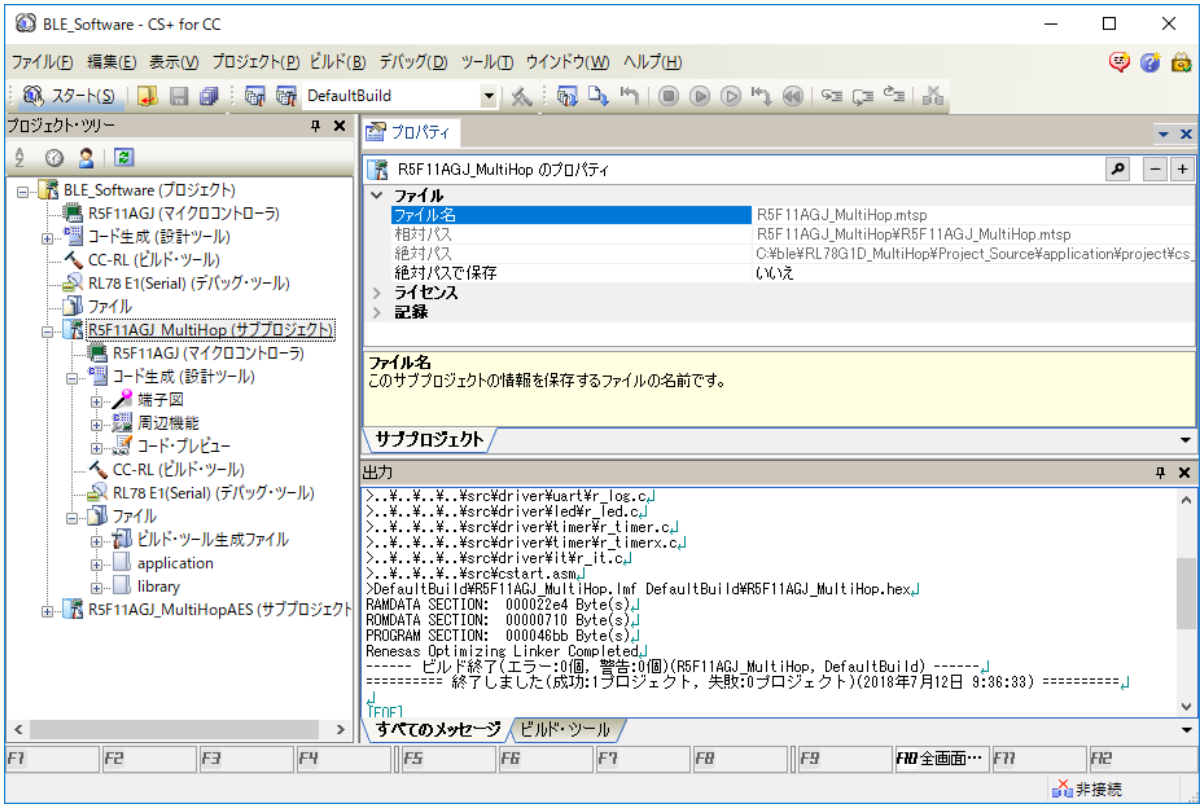
注意：サンプルプログラムに同梱されているビーコンスタックは、マルチホップ対応のための暫定的な機能追加が含まれます。ルネサスの WEB サイトで公開されているビーコンスタックとは異なるため、ビーコンスタックライブラリは差し替えないでください。

6.3　ファームウェアのビルド

サンプルプログラムのビルドには、統合開発環境 CS+ for CC を使用します。

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 BLE_Software.mtpj プロジェクトファイルを開きます。
   - Project_Source¥application¥project¥cs_cc¥BLE_Software¥
2. (R01AN4375)[プロジェクト・ツリー]ウィンドウで R5F11AGJ_Multihop(サブプロジェクト)を右クリックし、右クリックメニューで[R5F11AGJ_Multihop をリビルド]を選択してファームウェアをビルドします。

   (R01AN4466)[プロジェクト・ツリー]ウィンドウで R5F11AGJ_MultihopSEC(サブプロジェクト)を右クリックし、右クリックメニューで[R5F11AGJ_MultihopSEC をリビルド]を選択してファームウェアをビルドします。
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. (R01AN4375)下記のパスに R5F11AGJ_MultiHop.hex が生成されていることを確認します。
   - Project_Source¥application¥project¥cs_cc¥BLE_Software¥R5G11AGJ_MultiHop¥DefaultBuild¥

   (R01AN4466)下記のパスに R5F11AGJ_MultiHopSEC.hex が生成されていることを確認します。
   - Project_Source¥application¥project¥cs_cc¥BLE_Software¥R5G11AGJ_MultiHopSEC¥DefaultBuild¥

6.4 Company ID について

図 6-1 にマルチホップフレームの Company ID フィールドを示します。

マルチホップフレームは Bluetooth Low Energy の Advertising パケットで、Manufacturer Specific Data として送信されます。Manufacturer Specific Data には Bluetooth 企業 ID (Company ID)が必要です。

サンプルプログラムを開発する際は、下記の手順にて Bluetooth 企業 ID を設定してください。

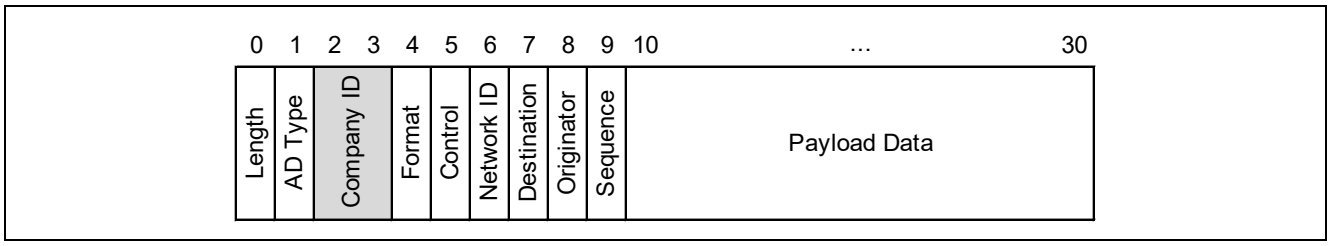


図 6-1　マルチホップフレームの Company ID フィールド

Bluetooth 企業 ID を設定するには、下記ファイルの COMPANY_ID マクロの値を変更します。

- Project_Source¥application¥src¥r_node.c

r_node.c (line.50-52)

```
50:    /* Bluetooth Company ID */
51:    /* https://www.bluetooth.com/specifications/assigned-numbers/company-identifie
52:    #define COMPANY_ID              (0xFFFF)
```

Bluetooth 企業 ID に変更

Bluetooth 企業 ID は下記の WEB サイトで確認できます。ID を取得するには、Bluetooth SIG への登録申請が必要です。

https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers

7. 使用ハードウェアリソース

ビーコンスタックとマルチホップレイヤは、下記のハードウェアリソースを使用します。

MCU 部		
	クロック発生回路	MCU 部メイン・システム・クロック(f_{MAIN})は、高速オンチップ・オシレータ発振クロック(f_{IH})の下記周波数のみ選択可能 • 4MHz • 8MHz • 16MHz • 32MHz ※MCU 部メイン・システム・クロック(f_{MAIN})は高速オンチップ・オシレータ・クロック(f_{IH})のみ使用可能 ※外部メイン・システム・クロック(f_{EX})は使用不可
		XT1 発振クロック(f_{XT})の使用・不使用を選択可能 • XT1 発振使用 • XT1 発振不使用（RF 部内蔵オシレータ使用） ※XT1 発振を使用する場合、PCLBUZ0 端子からクロックを出力し、RF 部スロー・クロック用クロックとして EXSLK_RF 端子に入力する ※XT1 発振には XT1,XT2 端子への 32.768kHz 水晶振動子の接続が必須
	クロック出力／ブザー出力	RF 部スロー・クロックに、PCLBUZ0 端子からの出力クロックを選択可能 • PCLBUZ0 からの 16.384kHz 出力 (XT1 発振の使用が必須) • PCLBUZ0 からの 32.768kHz 出力 (XT1 発振の使用が必須) • PCLBUZ0 不使用 ※PCLBUZ0 の不使用時、RF 部内蔵オシレータの使用が必須
	タイマ･アレイ･ユニット	動作クロック CK00 を 1MHz に設定 動作クロック CK01 を 250kHz に設定 TM00 使用、動作クロック CK00、ビーコンスタックで使用 TM01 使用、動作クロック CK01、マルチホップレイヤで使用 TM02 使用、動作クロック CK02、マルチホップレイヤで使用
	シリアル･アレイ･ユニット	CSI21 使用
	DMA コントローラ	DMA2 使用 DMA3 使用
	割り込み	INTRF 使用 INTDMA2 使用 INTDMA3 使用 INTTM00 使用
RF 部		
	DC-DC コンバータ	RF 部内蔵 DC-DC コンバータの使用・不使用を選択可能 • RF 部内蔵 DC-DC コンバータ使用 • RF 部内蔵 DC-DC コンバータ不使用 ※DC-DC コンバータの使用時、外付けの DC-DC コンバータ用フィードバック回路が必須
	スロー・クロック用 オシレータ	RF 部内蔵オシレータの使用・不使用を選択可能 • RF 部内蔵オシレータ使用 • RF 部内蔵オシレータ不使用 ※RF 部内蔵オシレータの不使用時、EXSLK_RF 端子への 16.384kHz クロックまたは 32.768kHz クロックの入力が必須
	クロック出力	CLKOUT_RF 端子から RF 基準クロック(32MHz)の分周クロックを出力可能 • クロック出力なし • 16MHz クロック出力 • 8MHz クロック出力 • 4MHz クロック出力 ※クロック出力なしを選択時、CLKOUT_RF 端子は入力モードとなる

8. マルチホップレイヤ API

本章ではマルチホップレイヤが提供する API の仕様を示します。

8.1 型

型名	基本型	説明
uint8_t	unsigned char	符号なし 8bit 整数型
uint16_t	unsigned short	符号なし 16bit 整数型
uint32_t	unsigned long	符号なし 32bit 整数型
int8_t	signed char	符号付き 8bit 整数型
int16_t	signed short	符号付き 16bit 整数型
int32_t	signed long	符号付き 32bit 整数型
bool	unsigned char	ブール型
int_t	signed int	符号付き int 型
uint_t	unsigned int	符号なし int 型
char_t	char	文字型
RBLE_STATUS	unsigned char	マルチホップ関数返却値

8.2 マクロ

8.2.1 ステータスマクロ

マクロ名	値	説明
RBLE_OK	0x00	正常終了
RBLE_ERR_PARAM	0x01	エラー終了：パラメータが不正
RBLE_ERR_WL	0x02	エラー終了：White List が空
RBLE_ERR_PWRDOWN	0x03	エラー終了：RF 部の電源供給が停止状態
RBLE_ERR_PWRUP	0x04	エラー終了：RF 部の電源供給が開始状態
RBLE_ERR_RFRX	0x05	エラー終了：RF 部の受信機能が無効状態
RBLE_ERR_START	0x06	エラー終了：送受信機能が実行状態
RBLE_ERR_STOP	0x07	エラー終了：送受信機能が停止状態
RBLE_ERR_HW_STANDBY	0x08	エラー終了：RF 部の異常、STANDBY_RF モード遷移時
RBLE_ERR_HW_STANDBYRX	0x09	エラー終了：RF 部の異常、STANDBY_RF モード遷移時、受信有効時
RBLE_ERR_HW_IDLE	0x0A	エラー終了：RF 部の異常、IDLE_RF モード遷移時

8.2.2 デバイスアドレスタイプマクロ

マクロ名	値	説明
RBLE_ADDR_PUBLIC	0x00	Public Device Address
RBLE_ADDR_RANDOM	0x01	Random Device Address

8.2.3 イベントマクロ

マクロ名	値	説明
RMH_EVT_RECEIVE_IND	0x01	フレーム受信通知
RMH_EVT_OPTION_IND	0x02	フレーム受信通知(オプションデータ)
RMH_EVT_STOP_CMP	0x03	フレーム受信停止完了通知
RMH_EVT_SEND_CMP	0x04	フレーム送信完了通知
RMH_EVT_ENCCNT_WRN	0x05	暗号化フレームカウンタ値の枯渇の警告
RMH_EVT_HOP_WRN	0x06	中継フレームバッファのフル状態による中継フレーム破棄の警告
RMH_EVT_DUP_WRN	0x07	重複判定バッファのフル状態による新規フレーム破棄の警告

8.3　構造体

8.3.1　デバイスアドレス構造体

メンバ名	型	オフセット	説明
struct RBLE_BD_ADDR			
addr	uint8_t[6]	0	デバイスアドレス

8.3.2　マルチホップ設定構造体

メンバ名	型	オフセット	説明
struct RMH_CFG			
own_addr	RBLE_BD_ADDR	0	ローカルデバイスのデバイスアドレス
own_addr_type	uint8_t	6	ローカルデバイスのデバイスアドレスタイプ
reserved	uint8_t	7	(予約)
company	uint16_t	8	ローカルデバイスのカンパニーID
network	uint8_t	10	ローカルデバイスのネットワーク ID
node	uint8_t	11	ローカルデバイスのノード ID
callback	void (*)(RMH_EVT*)	12	マルチホップコールバック関数

8.3.3　セキュリティ設定構造体

メンバ名	型	オフセット	説明
struct RMH_SEC			
enable	bool	0	セキュリティ機能の有効化
reserved	uint8_t	1	(予約)
key	uint8_t[16]	2	暗号鍵
counter	uint32_t	18	Nonce カウンタ値

8.3.4　フレームデータ構造体

メンバ名	型	オフセット	説明
struct RMH_DATA			
data	uint8_t[21]	0	フレームデータ
len	uint8_t	21	フレームデータ長
node	uint8_t	22	ノード ID
reserved	uint8_t	23	(予約)

8.3.5　オプションデータ構造体

メンバ名	型	オフセット	説明
struct RMH_OPTION			
id	uint8_t	0	オプションデータ ID
len	uint8_t	1	オプションデータ長
union data			
buf	uint8_t[19]	2	オプションデータ
rootcheck	ROOTCHECK	2	フレーム中継ログ

8.3.6　フレーム中継ログ構造体

メンバ名	型	オフセット	説明
struct ROOTCHECK			
counter	uint16_t	0	送信カウンタ
rootlog	uint8_t[15]	2	フレーム中継ログ
tail	uint8_t	17	フレーム中継ログの最後尾

8.3.7　マルチホップイベント構造体

メンバ名	型	オフセット	説明
struct RMH_EVT			
type	uint8_t	0	イベントタイプ
reserved	uint8_t	1	(予約)
union param			
send	RMH_SEND	2	フレーム送信通知
receive	RMH_RECEIVE	2	フレーム受信通知
discard	RMH_DISCARD	2	フレーム破棄警告

8.3.8　フレーム送信通知構造体

メンバ名	型	オフセット	説明
struct RMH_SEND			
union num			
seq	uint8_t	0	非暗号化フレーム送信時：シーケンス番号
counter	uint32_t	0	暗号化フレーム送信時：Nonce カウンタ値
enc	bool	4	暗号化フレームフラグ

8.3.9　フレーム受信通知構造体

メンバ名	型	オフセット	説明
struct RMH_RECEIVE			
data	uint8_t[21]	0	フレームデータ
len	uint8_t	21	フレームデータ長
dst	uint8_t	22	宛先ノード ID
org	uint8_t	23	発信ノード ID
union num			
seq	uint8_t	24	非暗号化フレーム受信時：シーケンス番号
counter	uint32_t	24	暗号化フレーム受信時：Nonce カウンタ値
enc	bool	28	暗号化フレームフラグ

8.3.10　フレーム破棄警告構造体

メンバ名	型	オフセット	説明
struct RMH_DISCARD			
dst	uint8_t	0	宛先ノード ID
org	uint8_t	1	発信ノード ID
union num			
seq	uint8_t	2	非暗号化フレーム受信時：シーケンス番号
counter	uint32_t	2	暗号化フレーム受信時：Nonce カウンタ値
enc	bool	6	暗号化フレームフラグ

8.4　関数

表 8-1 にマルチホップフレームの送受信を実行するための関数を示します。

表 8-1　マルチホップ関数

関数	動作
R_MH_Init()	マルチホップ機能の初期化
R_MH_Proc()	マルチホップ機能の実行
R_MH_Security()	セキュリティ機能の有効化 (※R01AN4466 のみ)
R_MH_Receive()	フレーム受信動作の実行
R_MH_Stop()	フレーム受信動作の停止
R_MH_Send()	フレームの送信
R_MH_CheckRoot()	経路確認用オプションフレームの送信

8.4.1　R_MH_Init

void R_MH_Init(RMH_CFG* cfg);

マルチホップ機能を初期化します。
マルチホップフレームを送受信するため、起動時に 1 回だけ呼び出してください。

引数 cfg->callback に設定するコールバック関数の実装例は、Usage を参照してください。

Parameters:

*cfg	own_addr	自デバイスのデバイスアドレス
	own_addr_type	自デバイスのデバイスアドレスタイプ 設定マクロは、8.2.2 項「デバイスアドレスタイプマクロ」を参照
	company	自デバイスのカンパニーID https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers
	network	自デバイスのネットワーク ID 0x00～0xFF
	node	自デバイスのノード ID 0x00～0xFE
	callback	マルチホップイベント通知コールバック関数 typedef void (*RMH_CALLBACK)(RMH_EVT* evt);

Return:

None

Usage:

引数 cfg->callback に登録するコールバック関数の実装例を示します。

```
static void node_callback(RMH_EVT* evt)
{
   switch (evt->type)
   {
      case RMH_EVT_SEND_CMP:
         /* マルチホップフレームの送信完了後はここに到達 */
         break;

      case RMH_EVT_RECEIVE_IND:
         /* マルチホップフレームの受信時はここに到達 */
         break;

      case RMH_EVT_STOP_CMP:
         /* マルチホップフレームの受信停止後はここに到達 */
         break;

      default:
         break;
   }
}
```

8.4.2　R_MH_Proc

void R_MH_Proc(void);

マルチホップ機能を実行するための関数です。

本関数はビーコンスタックのイベントを取得し、以下の動作を実行します。

- フレーム送信動作の完了後、フレーム送信完了をアプリケーションに通知
- フレーム受信動作の完了後、フレーム受信停止完了をアプリケーションに通知
- 自ノード宛フレームの受信時、フレーム受信をアプリケーションに通知
- 他ノード宛フレームの受信時、フレームの中継送信を開始

ビーコンスタックのイベントに対する処理が全て完了後、本関数は終了します。
また実行すべきビーコンスタックのイベントが無い場合、本関数は即時終了します。

本関数はアプリケーションのメインループで定期的に呼び出す必要があります。
実装例は、Usage を参照してください。

Parameters:

None

Return:

None

Usage:

本関数はアプリケーションのメインループで呼び出し、ビーコンスタックのイベント発生時に実行する必要があります。
メインループへの実装例を示します。

```
while (1)
{
   /* マルチホップ機能を実行 */
   R_MH_Proc();
}
```

メインループで HALT 命令または STOP 命令を実行することで、MCU 部消費電流の低減が可能です。
なお一部の MCU 周辺機能は STOP モードで動作が停止します(※)。このため周辺機能の動作状態に応じて、HALT 命令または STOP 命令を切り替えて使用してください。
HALT 命令、STOP 命令の実装例を示します。

```
while (1)
{
   /* マルチホップ機能を実行 */
   R_MH_Proc();

   __disable_interrupt();
   if(R_TIMER_IsActive())
   {
      /* マスクされていない割り込みの発生で HALT モードが解除される */
      __halt();
   }
   else
   {
      /* マスクされていない割り込みの発生で STOP モードが解除される */
      __stop();
   }
   __enable_interrupt();
}
```

※STOP モードで動作が停止する MCU 周辺機能については『RL78/G1D ユーザーズマニュアル ハードウェア編』(R01UH0515)の 19.3.2 項「STOP モード」を参照してください。

8.4.3 R_MH_Security

void R_MH_Security(RMH_SEC* sec);			
マルチホップのセキュリティ機能を有効化し、共通暗号鍵とカウンタ値を設定します。 セキュリティ機能を使用しない場合、本関数は使用しません。 ※本関数は R01AN4466 のみ実装されています。 暗号鍵はフレームの発信時、フレームの中継時、フレームの受信時に使用します。 このため同一ネットワーク ID の全てのノードに同じ暗号化を設定する必要があります。 セキュリティ機能を有効化後、R_MH_Send()を実行するとマルチホップレイヤは暗号化フレームを送信し、 R_MH_Receive()を実行するとマルチホップレイヤは暗号化フレームの中継と受信を行います。			
Parameters:			
	sec	enable	セキュリティ機能の有効化 true：有効 false：無効
		key	128bit 共通暗号鍵 暗号鍵は同一ネットワーク ID の発信ノード、中継ノード、受信ノードで共有
		counter	32bit Nonce カウンタ初期値 暗号化データのランダム性を確保するための Nonce として使用
Return:			
	RBLE_OK		正常終了
	RBLE_ERR_START		エラー終了：フレーム受信動作を実行中
	RBLE_ERR_PARAM		エラー終了：パラメータが不正

8.4.4 R_MH_Receive

RBLE_STATUS R_MH_Receive(bool hop_en);		
マルチホップフレームの受信動作を開始します。 引数 hop_en に true を設定した場合、他ノード宛フレームの中継動作を実行します。 自ノード宛または全ノード宛のフレームを受信すると、コールバック関数で RMH_EVT_RECEIVE_IND イベントが通知されます。 他ノード宛フレームの中継動作は、コールバック関数で通知されません。 フレームの受信動作を停止するには、R_MH_Stop()を実行します。 フレームの受信動作中に R_MH_Send()を実行することで、自ノードからフレームを送信できます。		
Parameters:		
	hop_en	他ノード宛マルチホップフレームの中継動作 true：中継動作を有効化 false：中継動作を無効化
Return:		
	RBLE_OK	正常終了
	RBLE_ERR_START	エラー終了：フレーム受信動作を実行中
	RBLE_ERR_PARAM	エラー終了：パラメータが不正

8.4.5 R_MH_Stop

RBLE_STATUS R_MH_Stop(void);		
R_MH_Receive()で開始したマルチホップフレーム受信動作を停止します。 マルチホップフレーム受信動作の停止完了は、コールバック関数で RMH_EVT_STOP_CMP イベントが通知されます。		
Parameters:		
	None	
Return:		
	RBLE_OK	正常終了
	RBLE_ERR_STOP	エラー終了：フレーム受信動作は停止中

8.4.6 R_MH_Send

RBLE_STATUS R_MH_Send(RMH_DATA* data);			
マルチホップフレームの送信動作を実行します。 マルチホップフレームの送信完了は、コールバック関数で RMH_EVT_SEND_CMP イベントが通知されます。 フレームを連続送信する場合は、RMH_EVT_SEND_CMP イベントの通知後、本関数を再実行してください。 フレームの送信中に R_MH_Receive()を実行することはできません。			
Parameters:			
	*data	data	マルチホップフレームのペイロードデータ
		len	マルチホップフレームのペイロードデータ長 (byte) 非暗号化フレーム時：最大 21byte 暗号化フレーム時：最大 14byte (※R01AN4466 のみ)
		node	マルチホップフレームの宛先ノード ID 0x00～0xFE　個別ノード 0xFF　全ノード
Return:			
	RBLE_OK		正常終了
	RBLE_ERR_START		エラー終了：フレーム送信動作を実行中
	RBLE_ERR_PARAM		エラー終了：パラメータが不正

8.4.7　R_MH_CheckRoot

RBLE_STATUS R_MH_CheckRoot(uint8_t dst, uint16_t counter);		
経路確認のためのオプションデータフレームを送信します。 ※本関数は経路確認機能が有効時のみ使用できます。経路確認機能の有効化については 9.1 節「経路確認機能」を参照してください。 フレームの送信完了は、コールバック関数で RMH_EVT_SEND_CMP イベントが通知されます。 また経路確認用オプションデータフレームを受信したノードでは、コールバック関数で RMH_EVT_OPTION_IND イベントが通知され、通知パラメータのフレームデータにはフレーム中継ログ構造体(8.3.6 項を参照)が格納されます。		
Parameters:		
	dst	経路確認機能用オプションデータフレームの宛先ノード ID 0x00～0xFE　個別ノード 0xFF　全ノード
	counter	フレームを識別するための任意のカウンタ値
Return:		
	RBLE_OK	正常終了
	RBLE_ERR_START	エラー終了：フレーム送信動作を実行中
	RBLE_ERR_PARAM	エラー終了：パラメータが不正

8.5　イベント

表 8-2 にマルチホップイベントを示します。

マルチホップレイヤは各マルチホップイベントを、R_MH_Init()で登録されたコールバック関数でアプリケーションに通知します。またコールバック関数は R_MH_Proc()から実行されます。

表 8-2　マルチホップイベント

イベント	通知内容
RMH_EVT_RECEIVE_IND	自ノード宛フレームの受信通知
RMH_EVT_OPTION_IND	自ノード宛フレーム(オプションデータ)の受信通知
RMH_EVT_STOP_CMP	フレーム受信の停止完了通知
RMH_EVT_SEND_CMP	フレーム送信の完了通知
RMH_EVT_ENCCNT_WRN	暗号化フレームカウンタ値の 0 リセット直前の警告 (※R01AN4466 のみ)
RMH_EVT_HOP_WRN	中継フレームバッファのフル状態による中継フレーム破棄の警告
RMH_EVT_DUP_WRN	重複判定バッファのフル状態による新規フレーム破棄の警告

8.5.1　RMH_EVT_RECEIVE_IND

RMH_EVT_RECEIVE_IND	
自ノード宛フレームの受信を通知します。 R_MH_Receive()でフレーム受信動作を実行中、自ノード ID 宛または全ノード宛のフレームを受信すると、コールバック関数で本イベントが通知されます。 暗号化フレームを受信した場合、復号後のフレームデータがパラメータ data に格納されます。	
Parameters:	
data[21]	フレームデータ
len	フレームデータ長(byte)
dst	宛先ノード ID
org	発信ノード ID
seq	非暗号化フレーム受信時：シーケンス番号
counter	暗号化フレーム受信時：Nonce カウンタ値
enc	暗号化フレームフラグ true：暗号化フレーム false：非暗号化フレーム

8.5.2　RMH_EVT_OPTION_IND

RMH_EVT_RECEIVE_IND	
自ノード宛フレーム(オプションデータ)の受信を通知します。 R_MH_Receive()でフレーム受信動作を実行中、自ノード ID 宛または全ノード宛の Control-optional ビットがセットされたフレームを受信すると、コールバック関数で本イベントが通知されます。 暗号化フレームを受信した場合、復号後のオプションデータがパラメータ data に格納されます。	
Parameters:	
data[21]	オプションデータ
len	オプションデータ長(byte)
dst	宛先ノード ID
org	発信ノード ID
seq	非暗号化フレーム受信時：シーケンス番号
counter	暗号化フレーム受信時：Nonce カウンタ値
enc	暗号化フレームフラグ true：暗号化フレーム false：非暗号化フレーム

8.5.3 RMH_EVT_STOP_CMP

RMH_EVT_STOP_CMP
フレーム受信動作の停止完了を通知します。 R_MH_Stop()でフレーム受信動作の停止が完了すると、コールバック関数で本イベントが通知されます。
Parameters: None

8.5.4 RMH_EVT_SEND_CMP

RMH_EVT_SEND_CMP	
フレームの送信完了を通知します。 R_MH_Send()でフレーム送信動作を実行後、コールバック関数で本イベントが通知されます。	
Parameters:	
seq	非暗号化フレーム送信時：シーケンス番号
counter	暗号化フレーム送信時：Nonce カウンタ値
enc	暗号化フレームフラグ true：暗号化フレーム false：非暗号化フレーム

8.5.5 RMH_EVT_ENCCNT_WRN

RMH_EVT_ENCCNT_WRN	
暗号化フレームにセットされる Nonce カウンタ値がまもなく一巡して 0 にリセットされることを警告します。 (※R01AN4466 のみ) R_MH_Security()でセキュリティ機能を有効化後、R_MH_Send()で暗号化フレームを送信して Nonce カウンタ値が枯渇警告の閾値と同一または超えた場合、コールバック関数で本イベントが通知されます。 本イベントが通知される場合、他ノードに対して Nonce カウンタ値のリセットを通知する必要があります。 (※本版では未対応)	
Parameters:	
counter	暗号化フレーム受信時：Nonce カウンタ値
enc	暗号化フレームフラグ true：暗号化フレーム
Supplementation: Nonce カウンタ値の閾値は、下記ファイルの ENCCNT_THRESHOLD マクロで定義されます。 － Project_Source¥application¥src¥r_multihop.c 106: /* encryption counter warning threshold to notify application */ 107: #define ENCCNT_THRESHOLD (0xFFFFFFF1UL)	

8.5.6　RMH_EVT_HOP_WRN

RMH_EVT_HOP_WRN

中継フレームの破棄を通知します。
R_MH_Receive()でフレーム受信動作を実行中、中継すべきフレームを格納するバッファに空きがなく、中継フレームが破棄されると、コールバック関数で本イベントが通知されます。

本イベントが通知される場合、下記のような要因で中継動作の負荷が大きい状態にあり、フレームの到達率が低下する可能性があります。

- マルチホップネットワーク上でフレームを送信するノード数が多い
- マルチホップネットワーク上の各ノードがフレームを送信する周期が短い

本イベントが通知される場合は、アプリケーションにてフレーム送信周期を調整するなどにより、マルチホップネットワーク上でのフレーム中継負荷を低減してください。また Supplementation を参照し、マルチホップレイヤに実装された中継フレーム格納バッファ数を調整してください。

Parameters:

dst	宛先ノード ID
org	発信ノード ID
seq	非暗号化フレーム受信時：シーケンス番号
counter	暗号化フレーム受信時：Nonce カウンタ値
enc	暗号化フレームフラグ true：暗号化フレーム false：非暗号化フレーム

Supplementation:

中継フレームバッファ数は、下記ファイルの HOP_BUFFER_NUM マクロで定義されます。

- Project_Source¥application¥src¥r_multihop.c

中継動作ではタイマを使用し、タイマドライバの時間管理バッファ数が中継フレームバッファ数となります。

```
95:   /* hop frame buffer size */
96:   #define HOP_BUFFER_NUM       (INDEXTIMER_NUM)
```

上記の時間管理バッファ数である INDEXTIMER_NUM マクロは、下記ファイルで定義されます。

- Project_Source¥application¥src¥driver¥timer¥r_timer.h

本マクロの値を、中継すべき全フレームを格納可能なバッファ数(2 のべき乗であること)に変更してください。

```
45:   /* the number of timer index (power of 2 (2^N)) */
46:   #define INDEXTIMER_NUM            (0x10)
```

8.5.7　RMH_EVT_DUP_WRN

RMH_EVT_DUP_WRN

新規宛先である受信フレームの破棄を通知します。
R_MH_Receive()でフレーム受信動作を実行中、重複フレームを判定するためのバッファに空きがなく、新規フレームが破棄されると、コールバック関数で本イベントが通知されます。

本イベントが通知される場合、重複フレーム判定バッファ数が不足しています。Supplementation を参照し、マルチホップレイヤに実装された重複フレーム判定バッファ数を調整してください。

Parameters:

dst	宛先ノード ID
org	発信ノード ID
seq	非暗号化フレーム受信時：シーケンス番号
counter	暗号化フレーム受信時：Nonce カウンタ値
enc	暗号化フレームフラグ true：暗号化フレーム false：非暗号化フレーム

Supplementation:

重複フレーム判定バッファ数は、下記ファイルの MH_NODE_NUM マクロで定義されます。

- Project_Source¥application¥src¥r_multihop.c

本マクロの値を、同一ネットワーク上に存在するノード数(最大 0xFE)に変更してください。

```
48:   /* the number of nodes existing in the network */
49:   #define MH_NODE_NUM         (0x40)
```

8.6　シーケンス

マルチホップの API シーケンスを示します。

アプリケーションは、R_MH_Init()でマルチホップレイヤを初期化し、R_MH_Receive()でフレーム受信動作と R_MH_Send()でフレーム送信動作を実行します。セキュリティ機能を使用する場合、R_MH_Security()でマルチホップレイヤのセキュリティ機能を有効化して暗号鍵を設定します(※R01AN4466 のみ)。

マルチホップレイヤは、R_MH_Init()で登録されたコールバック関数でイベントを通知します。

またマルチホップレイヤがコールバック関数によるイベントの通知と、受信フレームの中継を実行するために、アプリケーションは R_MH_Proc()を実行する必要があります。R_MH_Proc()の実装方法については 8.4.2 項「R_MH_Proc」を参照してください。

8.6.1　フレーム受信動作

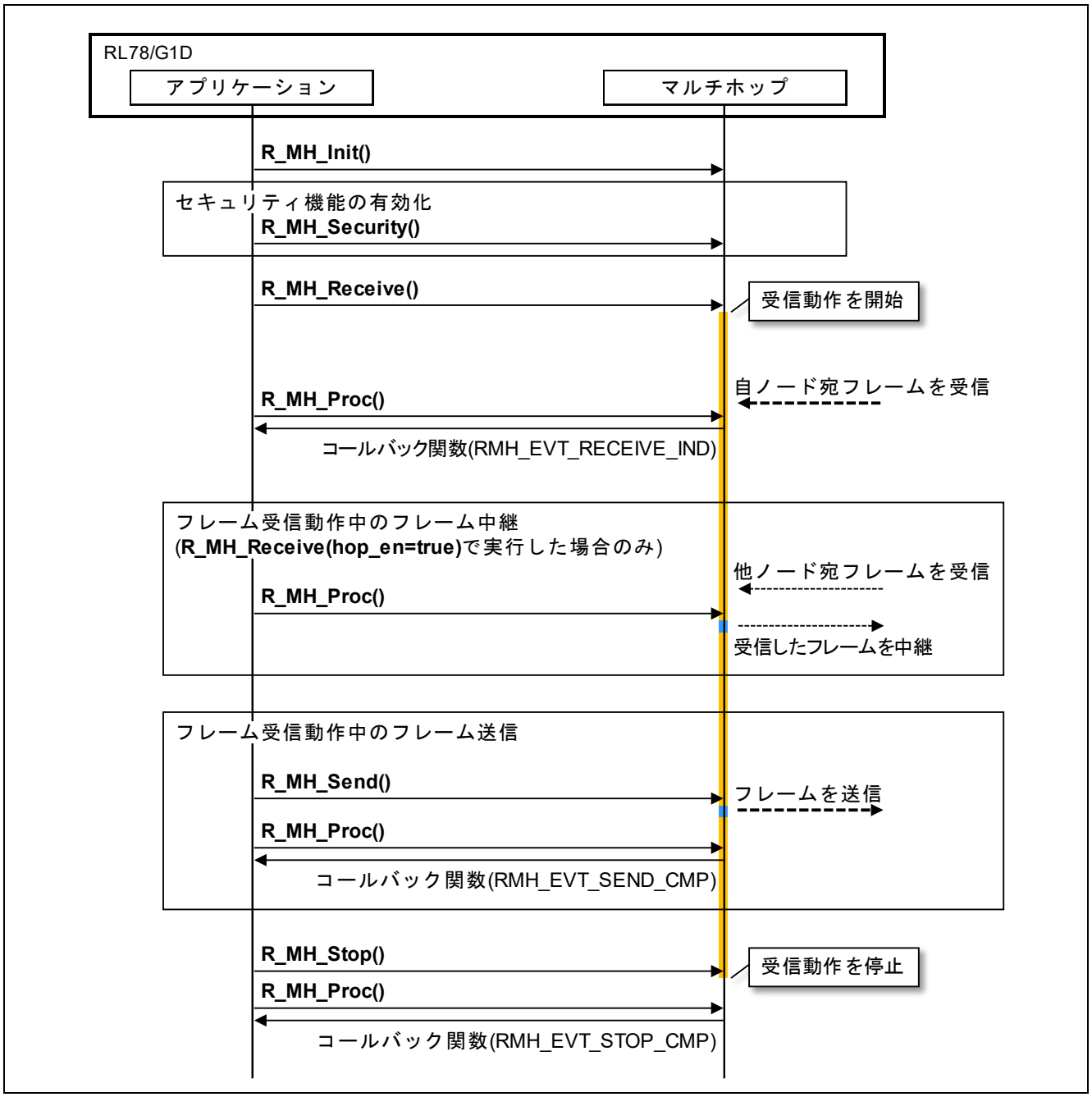


図 8-1　フレーム受信動作

8.6.2　フレーム送信動作

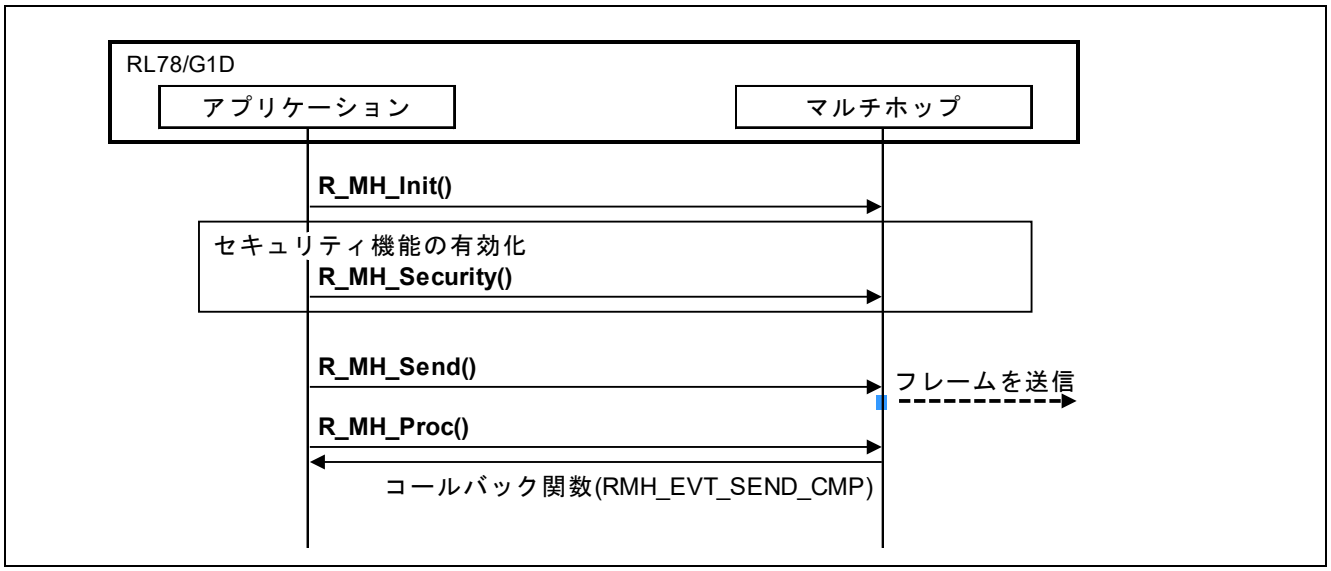


図 8-2　フレーム送信動作

8.7　フレーム受信動作

フレーム受信動作を開始するには R_MH_Receive()を実行します。

受信動作では Advertising チャネルの 37、38、39ch を周期的に切り替えてフレームを受信します。

フレーム受信動作を停止するには R_MH_Stop()を実行します。フレーム受信動作の停止完了後、コールバック関数が実行されて RMH_EVT_STOP_CMP イベントが通知されます。

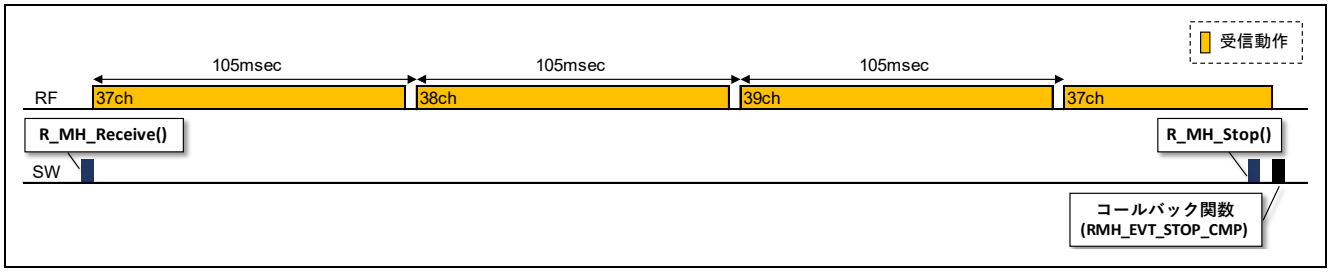


図 8-3　フレーム受信動作

8.7.1　自ノード宛フレームの受信

フレーム受信動作中に自ノード宛のフレームを受信すると、コールバック関数が実行されて RMH_EVT_RECEIVE_IND イベントが通知されます。

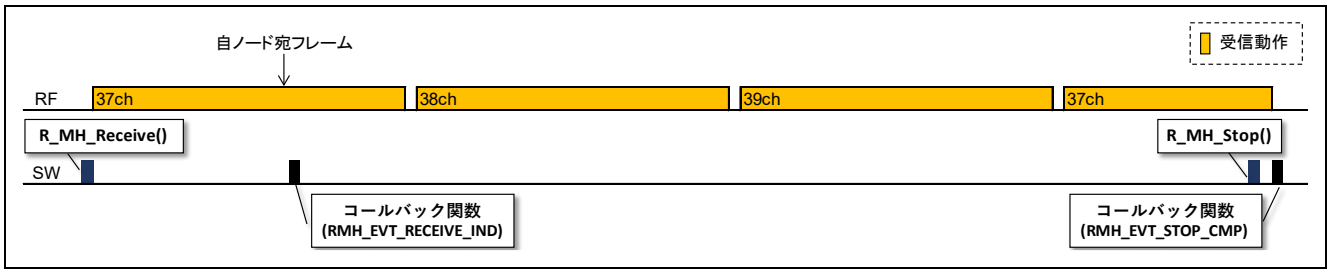


図 8-4　自ノード宛フレームの受信

8.7.2　他ノード宛フレームの中継

R_MH_Receive()を引数 hop_en=true で実行すると、他ノード宛フレームの中継動作が有効となります。

中継動作が有効でフレーム受信動作中に他ノード宛のフレームを受信すると、受信したフレームの中継動作が実行されます。

中継では、周囲のノードが中継したフレームとの衝突を避けるため、受信からランダム時間の経過後に送信されます。また他のノードへの到達率向上のため、各チャネルに３回ずつフレームを送信します。

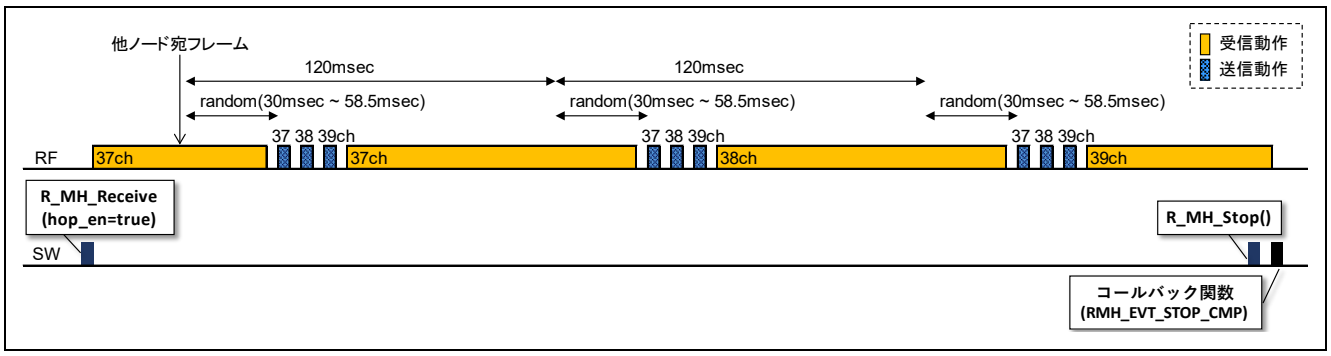


図 8-5　他ノード宛フレームの中継

8.8　フレーム送信動作

フレームを送信するには R_MH_Send()を実行します。

他のノードへの到達率向上のため、Advertising チャネルの 37、38、39ch に 3 回ずつフレームを送信します。また周囲のノードが送信したフレームとの衝突を避けるため、ランダム時間の経過後に再送信します。

フレーム送信の完了後、コールバック関数が実行されて RMH_EVT_SEND_CMP イベントが通知されます。

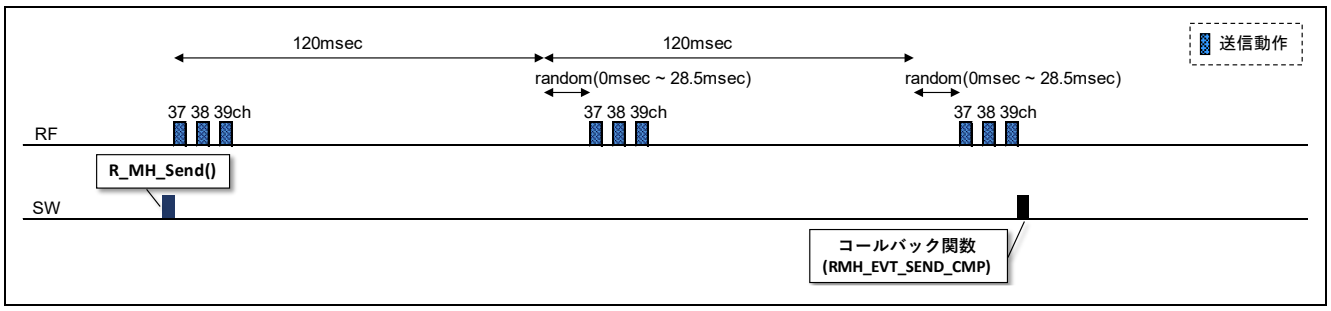


図 8-6　フレーム送信動作

8.8.1　フレーム受信動作中の送信

フレームの受信動作中でもフレームの送信が可能です。

フレームの受信動作中に R_MH_Send()を実行すると、フレームを送信します。フレーム送信の完了後、コールバック関数が実行されて RMH_EVT_SEND_CMP イベントが通知されます。

また RMH_EVT_SEND_CMP イベントが通知されるまでの受信動作で、自ノード宛のフレームを受信した場合は、RMH_EVT_RECEIVE_IND イベントが通知されます。

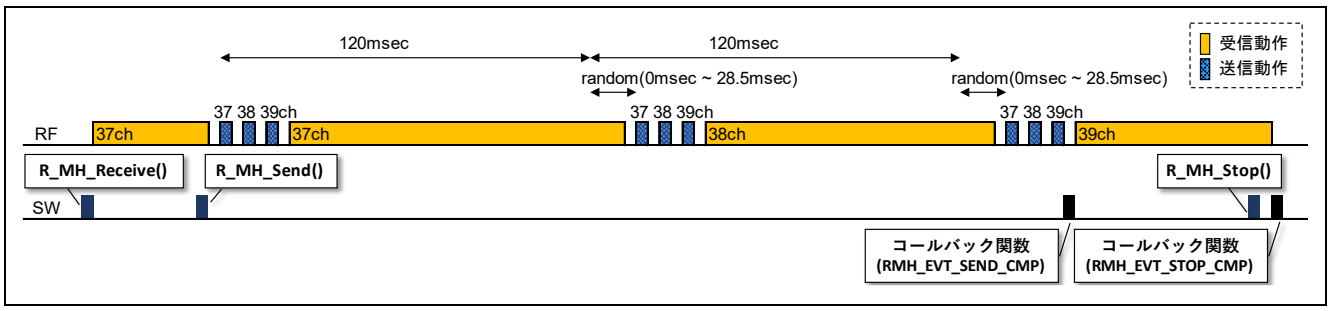


図 8-7　フレーム受信動作中のフレーム送信

8.9　送受信チャネル

8.7 節と 8.8 節で示したように、マルチホップレイヤは全 Advertising チャネルを使用してフレームを送受信します。

ここでサンプルプログラムでは、マルチホップで使用するチャネルを 1 つに限定することができます。使用チャネルを 1 つに限定するには、下記ファイルの MH_ALL_CH_EN マクロの値を(0)に、MH_SINGLE_CH マクロの値を使用するチャネルに変更します。

- Project_Source¥application¥src¥r_multihop.c

r_multihop.c (line.51-56)

```
51:    /* 1: uses ALL    advertising channels for multi-hop transmission/reception. *
52:    /* 0: uses SINGLE advertising channel  for multi-hop transmission/reception. *
53:    #define MH_ALL_CH_EN           (1)
54:    #if !MH_ALL_CH_EN
55:        #define MH_SINGLE_CH       (RBLE_ADV_CHANNEL_37)
56:    #endif
```

(0)に変更

37 or 38 or 39 に変更

図 8-8 に単一チャネルでの受信動作を示します。常に指定チャネルで受信します。

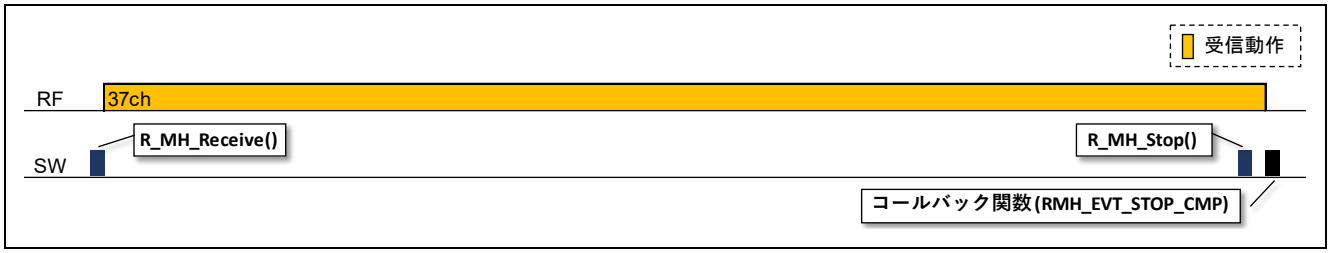


図 8-8　単一チャネルのフレーム受信動作

図 8-9 に単一チャネルでの送信動作を示します。同一フレームを指定チャネルに各 3 回送信します。

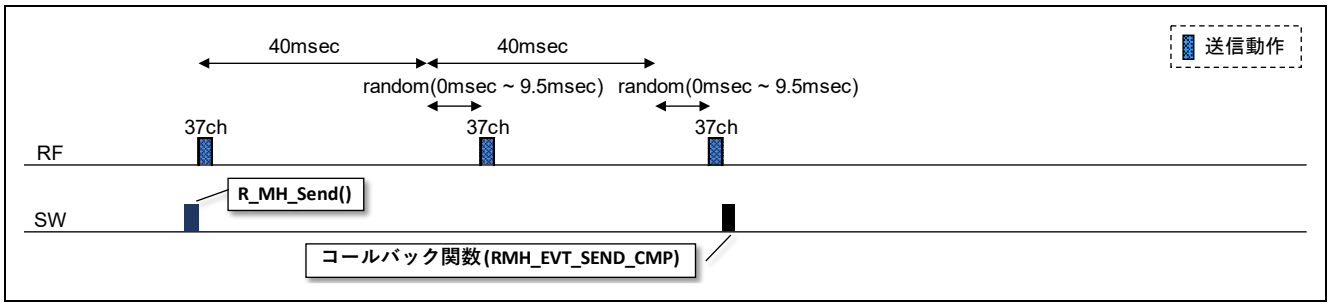


図 8-9　単一チャネルのフレーム送信動作

9. Appendix

9.1 経路確認機能

サンプルプログラムには、フレームの中継経路を確認するための機能が実装されています。本機能を使用することで、どのノードによってフレームが中継されたかを確認することができます。

経路確認機能を有効にするには、下記ファイルの HOP_ROOT_CHECK マクロの値を(1)に変更します。

なお本マクロの値を(1)に変更すると、セキュリティ機能は無効となります。

- Project_Source¥application¥src¥r_multihop.h

r_multihop.h (line.44-46)

```
44:    /* 1: enables the function to check hopping root */
45:    /* 0: normal operation; disable the function to check */
46:    #define HOP_ROOT_CHECK          (1)
```

(1)に変更

上記の設定でビルドしたファームウェアを全ての評価ボードに書き込んでプログラムを実行し、スイッチ SW2 を押下すると、サンプルプログラムは ID=0 に対して周期的にフレームを送信します。

また ID=0 のノードは受信したフレームの中継経路ログを下記のフォーマットで UART から出力します。

通し番号 ORG 番号 -> *DST 番号 SEQ 番号 フレームデータサイズ* [*送信番号*] *中継経路*

図 9-1 に、ID=3 からフレームを送信した場合の中継経路ログの例を示します。

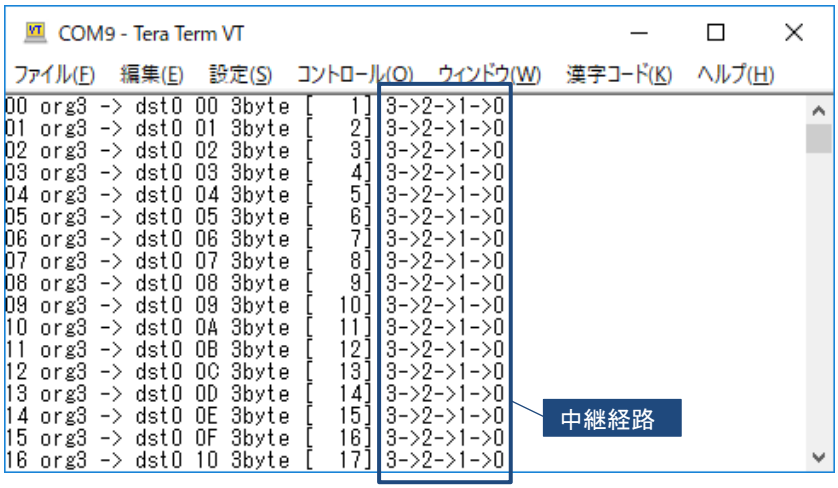


図 9-1　受信フレームの中継経路ログ例

本ログにより、下記の経路でフレームが中継されたことを確認できます。

ID=3→ID=2→ID=0　　：フレーム 0006,0015
ID=3→ID=1→ID=2→ID=0　　：フレーム 0012,0013
ID=3→ID=1→ID=0　　：上記を除くフレーム

9.2　デバイスフィルタ

サンプルプログラムにはデモ用に、特定のデバイスが送信した Advertising パケットのみ受信し、他のデバイスから送信された Advertising パケットは破棄するためのデバイスフィルタが実装されています。デバイスフィルタを使用することで、特定の中継経路に限定したフレーム送受信動作を確認することができます。

デバイスフィルタを有効にするには、下記ファイルの DEV_ADDR_FILTER マクロの値を(1)に変更します。

- Project_Source¥application¥src¥r_multihop.h

r_multihop.h (line.48-50)

```
48:    /* 1: receives the frame from only the device addresses registered in addr_fil
49:    /* 0: normal operation; receives the frame without filtering. */
50:    #define DEV_ADDR_FILTER        (0)
                                      (1)に変更
```

サンプルプログラムに同梱されたシステム動作設定で書き込まれるアドレスは、ランダムデバイスアドレス DC:D7:2C:71:F4:xx であり、xx の値はノード ID と同一の値です。

例)　ノード ID=0：　DC:D7:2C:71:F4:00
　　ノード ID=1：　DC:D7:2C:71:F4:01

デバイスフィルタは、デバイスアドレスでフィルタリングを行い、自ノード ID±１のデバイスから送信された Advertising データのみマルチホップの送受信対象となります。これにより、ノード ID の昇順または降順でフレームの送受信と中継が実行されます。

図 9-2 にデバイスフィルタ有効時のフレームの伝送経路を示します。

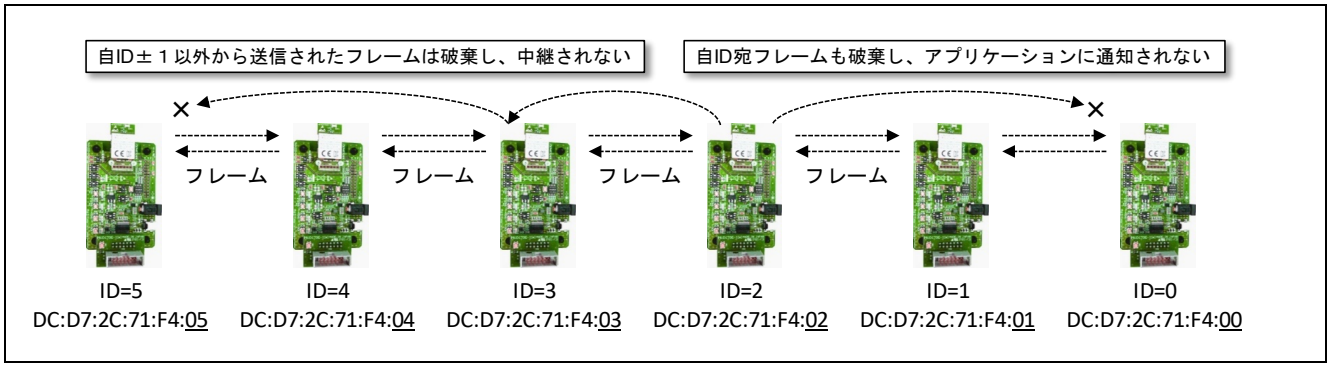


図 9-2　デバイスフィルタによる伝送経路

なおサンプルプログラムに同梱されている下記のファームウェアは、DEV_ADDR_FILTER マクロを(1)に変更してビルドしたものです。

- ROM_File¥R5F11AGJ_MultiHop(DEV_FILTER).hex

9.3　デバイスドライバ

本サンプルプログラムに実装されたデバイスドライバの関数仕様を示します。

デバイスドライバはソースコード形式で提供されるため、必要に応じてカスタマイズが可能です。

9.3.1　プラットフォーム（クロック、ポート）

アプリケーションは起動時にプラットフォーム（クロック、ポート）を初期化します。

プラットフォームドライバのソースコードは下記のフォルダに格納されています。

- Project_Source¥application¥src¥driver¥plf

プラットフォームドライバの関数仕様を以下に示します。

bool R_PLF_Init(void);	
プラットフォーム（クロック、ポート）を初期化します。 - 高速オンチップ・オシレータ - XT1 発振回路または低速オンチップ・オシレータ - サブ・クロック出力 - 入出力ポート 本関数は、RL78/G1D 評価ボード(RTK0EN0001D01001BZ)を想定した設定を実行します。 上記と異なる RL78/G1D ボード上で本サンプルプログラムを実行する場合、本関数のポート設定を見直してください。	
Parameters:	
None	
Return:	
true	プラットフォームの初期化に成功
false	プラットフォームの初期化に失敗、ユーザ・オプション・バイト設定に誤りあり

ユーザ・オプション・バイトについては『RL78/G1D ユーザーズマニュアル ハードウェア編』(R01UH0515)の第 25 章「オプション・バイト」を参照してください。

9.3.2　12 ビット・インターバル・タイマ

アプリケーションは一定周期でフレームを送信するために 12 ビット・インターバル・タイマを使用します。

12 ビット・インターバル・タイマドライバのソースコードは下記のフォルダに格納されています。

- Project_Source¥application¥src¥driver¥it

12 ビット・インターバル・タイマドライバの関数仕様を以下に示します。

void R_IT_Init(void);	
12 ビット・インターバル・タイマを初期化します。	
Parameters:	
None	
Return:	
None	

void R_IT_Start(uint16_t time, IT_CALLBACK callback);	
12 ビット・インターバル・タイマの動作を開始します。 12 ビット・インターバル・タイマは MCU が STOP モードでも動作が可能です。 タイマが満了すると、引数 callback で指定したコールバック関数を実行します。	
Parameters:	
time	コンペア値 0x0001～0x0FFF サブシステム・クロック(32.768kHz)：(time+1)×30.5usec 低速オンチップ・オシレータ・クロック(15kHz)：(time+1)×66.6usec
callback	タイマ満了通知コールバック関数 typedef void (*IT_CALLBACK)(void);
Return:	
None	

void R_IT_Stop(void);
12 ビット・インターバル・タイマの動作を停止します。
Parameters:
None
Return:
None

9.3.3　タイマ・アレイ・ユニット

マルチホップレイヤはフレームの送信間隔を制御するためにタイマ・アレイ・ユニットを使用します。またアプリケーションは未使用のチャネルを使用することができます。

チャネル 0　：ビーコンスタックが使用

チャネル 1,2 ：マルチホップが使用

チャネル 3　：UART ドライバが使用

チャネル 4～7：アプリケーションで使用可能

タイマ・アレイ・ユニットドライバのソースコードは下記のフォルダに格納されています。

- Project_Source¥application¥src¥driver¥timer

タイマ・アレイ・ユニットドライバの関数仕様を以下に示します。

void R_TIMER_Init(void);	
タイマ・アレイ・ユニットを初期化します。 タイマ・アレイ・ユニット 0 のチャネル 1～7 が使用可能となります。	
Parameters:	
None	
Return:	
None	

bool R_TIMER_IsActive(void);	
タイマ・アレイ・ユニットの動作状態を確認します。	
Parameters:	
None	
Return:	
true	タイマ・アレイ・ユニットのいずれかのチャネルが動作状態
false	タイマ・アレイ・ユニットのすべてのチャネルが停止状態

void R_TIMER1_Start(uint16_t time, TIMER_CALLBACK callback);
:
void R_TIMER7_Start(uint16_t time, TIMER_CALLBACK callback);	

タイマ・アレイ・ユニット 0 のチャネル 1～7 のいずれかの動作を開始します。 タイマ・アレイ・ユニットは MCU が STOP モードでは動作が停止します。 タイマが満了すると、引数 callback で指定したコールバック関数を実行します。	
Parameters:	
time	コンペア値 1～500 1 クロックあたりの時間は 0.5msec
callback	タイマ満了通知コールバック関数 typedef void (*TIMER_CALLBACK)(void);
Return:	
None	

void R_TIMER1_Stop(void);
:
void R_TIMER7_Stop(void);

タイマ・アレイ・ユニット 0 のチャネル 1～7 のいずれかの動作を停止します。
Parameters:
None
Return:
None

void R_TIMER1_Elapsed(void);
:
void R_TIMER7_Elapsed(void);	

タイマ・アレイ・ユニット 0 のチャネル 1～7 のいずれかの動作開始からの経過時間を取得します。	
Parameters:	
None	
Return:	
経過時間 1～500 1 クロックあたりの時間は 0.5msec	

void R_TIMERX_Init(TIMERX_CALLBACK callback);	
インデックスタイマを初期化します。 インデックスタイマはタイマ・アレイ・ユニット 0 のチャネル 1 を利用した、同時動作が可能な複数のタイマです。 インデックスタイマはインデックスで識別し、指定可能なインデックスは 0～15 です。 満了したタイマのインデックスは、引数 callback の引数で通知されます。	
Parameters:	
callback	タイマ満了通知コールバック関数 typedef void (*TIMERX_CALLBACK)(uint8_t); 満了したタイマのインデックスをコールバックの引数で通知
Return:	
None	

void R_TIMERX_Start(uint8_t idx, uint16_t time);	
インデックスタイマの動作を開始します。 インデックスタイマはタイマ・アレイ・ユニットを使用するため、MCU が STOP モードでは動作が停止します。 インデックスタイマが満了すると、R_TIMERX_Init()の引数 callback で指定したコールバック関数を実行します。	
Parameters:	
idx	インデックスタイマを識別するインデックス 0～15
time	コンペア値 1～500 1 クロックあたりの時間は 0.5msec
Return:	
None	

9.3.4　データフラッシュ

アプリケーションはセキュリティ機能の Nonce カウンタ値を保持するためにデータフラッシュを使用します。データフラッシュドライバのソースコードは下記のフォルダに格納されています(※R01AN4466 のみ)。

- Project_Source¥application¥src¥driver¥dataflash

本ドライバは下記のデータフラッシュライブラリを使用します。

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/software-tool/data-flash-libraries

データフラッシュドライバの関数仕様を以下に示します。

uint8_t R_DFL_Init(void);	
データフラッシュドライバを初期化します。	
Parameters:	
None	
Return:	
DFL_OK	正常終了
上記以外	エラー終了

uint8_t R_DFL_Start(void);	
データフラッシュの操作を開始します。 本関数の実行でデータフラッシュ操作の開始状態となり、下記の関数が実行可能となります。 - R_DFL_Read() - R_DFL_Write() - R_DFL_Refresh() - R_DFL_Format() データフラッシュ操作の開始状態では、CPU を HALT モードや STOP モードに遷移させることはできません。 CPU を HALT モードや STOP モードに遷移させる場合は、R_DFL_Stop()を実行してデータフラッシュの操作を停止状態にしてください。	
Parameters:	
None	
Return:	
DFL_OK	正常終了
上記以外	エラー終了

uint8_t R_DFL_Stop(void);	
データフラッシュの操作を停止します。 本関数の実行でデータフラッシュ操作の停止状態となります。	
Parameters:	
None	
Return:	
DFL_OK	正常終了
上記以外	エラー終了

uint8_t R_DFL_Read(const uint8_t id, void* addr);

データフラッシュに保持されている引数 id のデータを読み込みます。
データ ID はあらかじめ下記のものが定義されています。
4byte データ用 ID： EEL_ID_DATA01、EEL_ID_DATA02
8byte データ用 ID： EEL_ID_DATA03、EEL_ID_DATA04
16byte データ用 ID： EEL_ID_DATA05、EEL_ID_DATA06
32byte データ用 ID： EEL_ID_DATA07、EEL_ID_DATA08

本関数はデータフラッシュ操作の開始状態で実行できます。

Parameters:

id	データ ID
addr	データの読み込み先

Return:

DFL_OK	正常終了
DFL_ERR_PARAMETER	エラー終了：引数に誤りがある
上記以外	エラー終了：上記以外

uint8_t R_DFL_Write(const uint8_t id, void* addr);

データフラッシュに引数 id としてデータを書き込みます。
データ ID はあらかじめ下記のものが定義されています。
4byte データ用 ID： EEL_ID_DATA01、EEL_ID_DATA02
8byte データ用 ID： EEL_ID_DATA03、EEL_ID_DATA04
16byte データ用 ID： EEL_ID_DATA05、EEL_ID_DATA06
32byte データ用 ID： EEL_ID_DATA07、EEL_ID_DATA08

本関数はデータフラッシュ操作の開始状態で実行できます。

Parameters:

id	データ ID
addr	書き込みデータ

Return:

DFL_OK	正常終了
DFL_ERR_PARAMETER	エラー終了：引数に誤りがある
DFL_ERR_NO_INSTANCE	エラー終了：データが存在しない
DFL_ERR_POOL_FULL	エラー終了：書き込み可能な領域がない
上記以外	エラー終了：上記以外

uint8_t R_DFL_Refresh(void);

データフラッシュのブロック状態をリフレッシュします。データフラッシュ内のデータは保持されます。

本関数はデータフラッシュ操作の開始状態で実行できます。

Parameters:

None

Return:

DFL_OK	正常終了
上記以外	エラー終了

uint8_t R_DFL_Format(void);

データフラッシュをフォーマットします。データフラッシュ内の全データは消去されます。

本関数はデータフラッシュ操作の開始状態で実行できます。

Parameters:

None

Return:

DFL_OK	正常終了
上記以外	エラー終了

9.3.5 UART

アプリケーションは受信したフレームのデータログを出力するために UART を使用します。

UART ドライバのソースコードは下記のフォルダに格納されています。

- Project_Source¥application¥src¥driver¥uart

UART ドライバの関数仕様を以下に示します。

bool R_UART_Init(UART_INFO* info);			
シリアル・アレイ・ユニットの UART0 を初期化します。			
Parameters:			
	info	baudrate	UART ボーレート UART_BAUDRATE_9600_BPS　9,600bps UART_BAUDRATE_115200_BPS　115,200bps UART_BAUDRATE_1M_BPS　1,000,000bps
		rx_tout	UART 受信タイムアウト 1～100msec データ先頭の受信開始から、本タイムアウト時間内に R_UART_Rx()の引数 size で指定したデータサイズを受信できない場合、受信したデータを破棄 0 を設定すると受信タイムアウトは無効
		rx_callback	R_UART_Rx()を実行後、データ受信完了を通知するコールバック関数
		tx_callback	R_UART_Tx()を実行後、データ送信完了を通知するコールバック関数
		err_callback	R_UART_Rx()を実行後、UART 通信のエラーを通知するコールバック関数
Return:			
	true		UART の初期化に成功
	false		UART の初期化に失敗

bool R_UART_IsActive(void);		
UART の動作状態を確認します。		
Parameters:		
	None	
Return:		
	true	UART が動作状態
	false	UART が停止状態

void R_UART_Rx(__near uint8_t *rxbuf, const uint16_t size);		
UART の受信動作を開始します。 受信したデータを引数 rxbuf の受信バッファに格納します。 引数 size で指定したサイズのデータを受信完了すると、R_UART_Init()の引数 info->rx_callback で指定したコールバック関数を実行します。		
Parameters:		
	rxbuf	受信データの格納バッファ
	size	受信データサイズ 1～1024byte
Return:		
	None	

void R_UART_Tx(__near const uint8_t *txbuf, const uint16_t size);		
UART の送信動作を開始します。 引数 txbuf で指定された送信バッファのデータを送信します。 引数 size で指定したサイズのデータを送信完了すると、R_UART_Init()の引数 info->tx_callback で指定したコールバック関数を実行します。		
Parameters:		
	txbuf	送信データの格納バッファ
	size	送信データサイズ 1～1024byte
Return:		
	None	

void R_LOG_Init(uint8_t baudrate);		
ログ送信機能を初期化します。 ログ送信機能は複数の送信バッファを持ち、UART を利用して送信バッファに格納されたログを送信します。 本関数は UART を初期化するため、R_UART_Init()を実行します。		
Parameters:		
	baudrate	UART ボーレート UART_BAUDRATE_9600_BPS　9,600bps UART_BAUDRATE_115200_BPS　115,200bps UART_BAUDRATE_1M_BPS　1,000,000bps
Return:		
	true	ログ送信機能の初期化に成功
	false	ログ送信機能の初期化に失敗

bool R_LOG_Send(char_t* string);		
ログを送信します。 引数 string で指定されたログを送信バッファに格納し、UART でログを送信します。 本関数は UART でログを送信するため、R_UART_Tx()を実行します。		
Parameters:		
	string	ログ文字列 1～200byte
Return:		
	true	ログを送信バッファに格納成功、ログは順次送信される
	false	送信バッファに空きが無く、ログの格納に失敗

9.3.6　外部入力割り込み

アプリケーションは評価ボードのスイッチ押下を検出するために外部入力割り込みを使用します。

外部入力割り込みドライバのソースコードは下記のフォルダに格納されています。

- Project_Source¥application¥src¥driver¥input

外部入力割り込みドライバの関数仕様を以下に示します。

void R_INPUT_Init(input_callback_t callback);		
外部入力割り込みを初期化します。 評価ボードの SW2 に接続された INTP5/P16 の立ち上がりエッジ検出を有効化します。 外部入力割り込みが発生すると、引数 callback で指定したコールバック関数を実行します。		
Parameters:		
	callback	外部入力通知コールバック関数 typedef void (*input_callback_t)(void);
Return:		
	None	

9.3.7　LED

アプリケーションは自ノード宛のフレームを受信すると、評価ボードの LED の点灯状態を変更します。またマルチホップレイヤは中継すべきフレームを受信すると、評価ボードの LED を点灯させます。

LED ドライバのソースコードは下記のフォルダに格納されています。

- Project_Source¥application¥src¥driver¥led

LED ドライバの関数仕様を以下に示します。

void R_LED_Init(void);	
評価ボードの LED 操作を有効化します。 評価ボードの LED1-4 に接続された P120, P147, P03, P60 を初期化します。	
Parameters:	
None	
Return:	
None	

void R_LED_Set(uint8_t led);		
評価ボードの LED の点灯状態を操作します。		
Parameters:		
	led	LED の点灯状態 bit0：LED1 (0: 消灯、1: 点灯) bit1：LED2 (0: 消灯、1: 点灯) bit2：LED3 (0: 消灯、1: 点灯) bit3：LED4 (0: 消灯、1: 点灯)
Return:		
None		

9.4　マルチホップのフレーム到達率評価例

マルチホップ機能によるフレーム到達率について、弊社環境における評価結果を示します。

9.4.1　注意事項

記載したフレーム到達率は、弊社環境での測定結果を示す参考値であり、製品の保証値ではありません。製品ご利用を検討の際は、お客様の想定する動作条件と環境においての評価をお願いします。

9.4.2　評価内容

本評価は、屋内(弊社事業所内)で実施しました。無線 LAN デバイスや他の Bluetooth デバイスが多数動作している環境となります。

本評価の構成を**図 9-3** に示します。RL78/G1D 評価ボードを 6.5m おきに配置します。評価ボードにはフレーム到達率を測定するためのプログラムを書き込み、ノード ID をノード 0 から順番に割り当てます。ノード 0 の評価ボードは USB で PC と接続し、その他のノードの評価ボードにはバッテリを接続します。

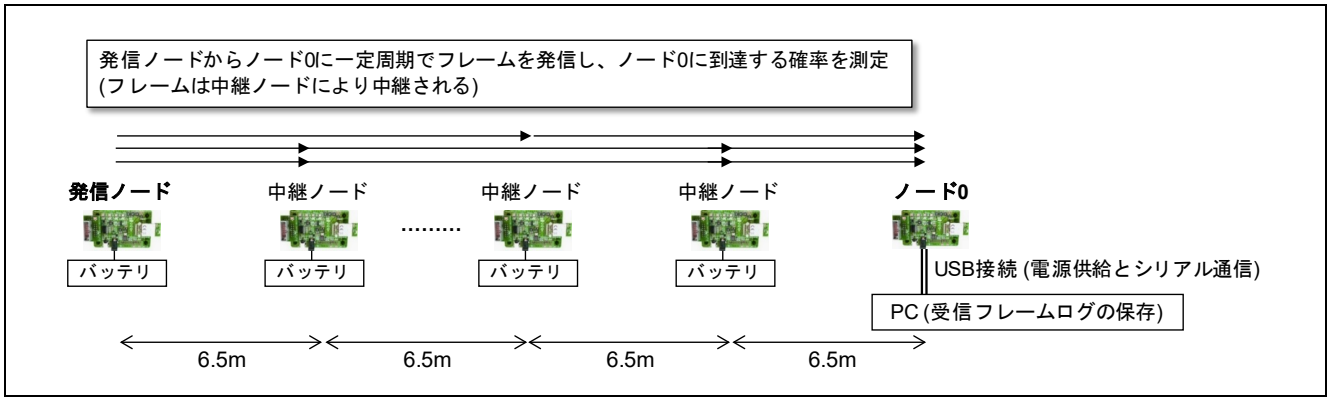


図 9-3　フレーム到達率の測定構成

本評価では、発信ノードからノード 0 に対してマルチホップフレームを発信し、最終的にノード 0 が受信する割合をフレーム到達率として測定しました。

測定プログラムの動作設定を**表 9-1** に示します。測定プログラムでは、マルチホップレイヤのデフォルト実装から送信パワーを変更しました。

表 9-1　測定プログラムの動作設定

分類	測定条件
MCU 動作周波数	8MHz
DC-DC コンバータ	RF 内蔵 DC-DC コンバータ使用
RF スロー・クロック供給源	RF 内蔵オシレータ使用、キャリブレーション実行
送信パワー	-15dBm (デフォルトの 0dBm から変更) ※室内環境で各ノードからの 1 ホップでの通信距離を抑制し、マルチホップ動作させるため
送信チャネル数	3ch 送信 (37,38,39ch)
マルチホップ動作	サンプルプログラムの経路確認機能を有効化、セキュリティ機能は無効 ※詳細は 9.1「経路確認機能」を参照

測定プログラムによるフレーム到達率の測定シーケンスを図 **9-4** に示します。

発信ノードはノード 0 に対してフレームを発信します。発信ノード、ノード 0 以外のノードは、中継ノードとして本フレームを中継し、最終的にノード 0 にフレームが到達します。ただし、各フレームの中継経路はそれぞれ異なる可能性があります。

ノード 0 は、フレーム到達率を下記により計算しました。

(フレーム到達率%)=(ノード 0 が受信したフレーム数)／(発信ノードが送信したフレーム数)×100

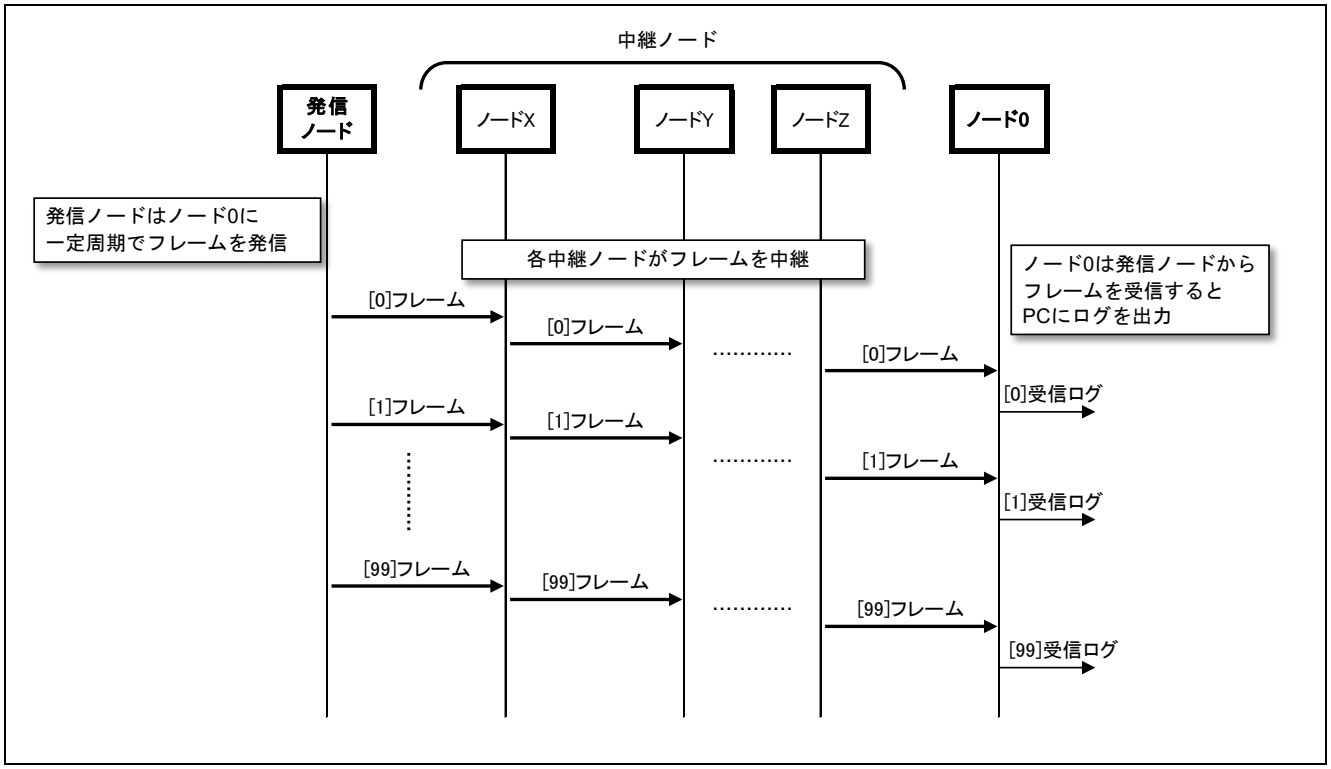


図 **9-4**　フレーム到達率の測定シーケンス

9.4.3　中継ノードの有無によるフレーム到達率

(1)　測定手順

マルチホップのノード構成にて、中継ノードがある場合とない場合のフレーム到達率をそれぞれ測定しました。これにより、マルチホップ機能のフレーム中継によるフレーム到達率の向上効果を確認しました。

中継ノードがある構成を**図 9-5** に、中継ノードがない構成を**図 9-6** に示します。

どちらの構成においても発信ノードは、ノード 0 に対して 500 ミリ秒間隔でフレームを送信しました。また発信ノードがノード 1 からノード 6 までの場合のそれぞれのフレーム到達率を測定しました。

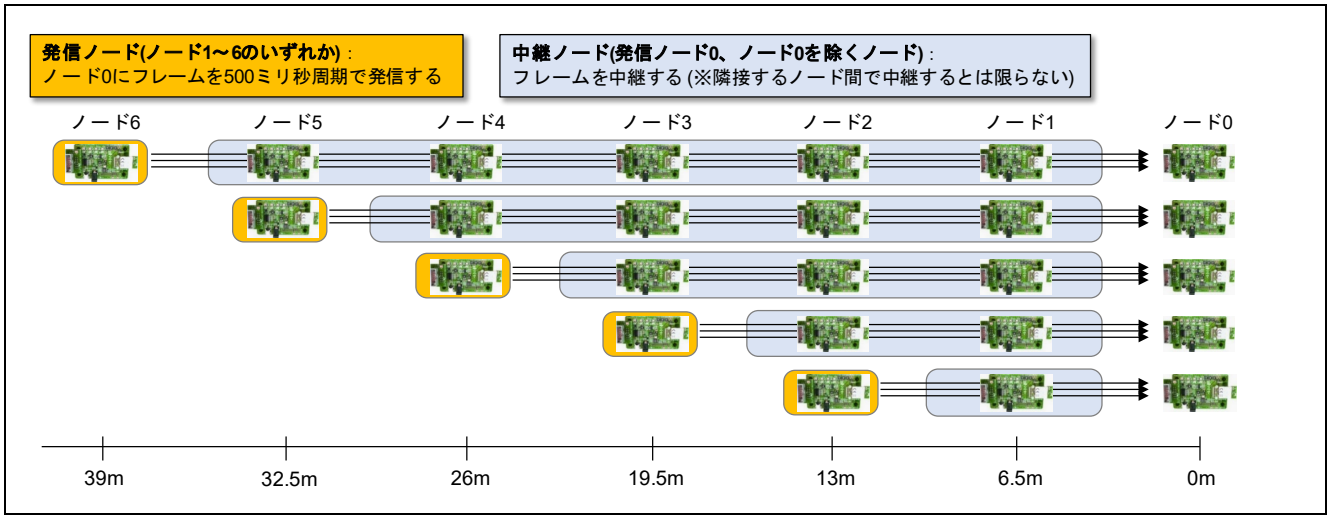


図 9-5　中継ノードがある構成

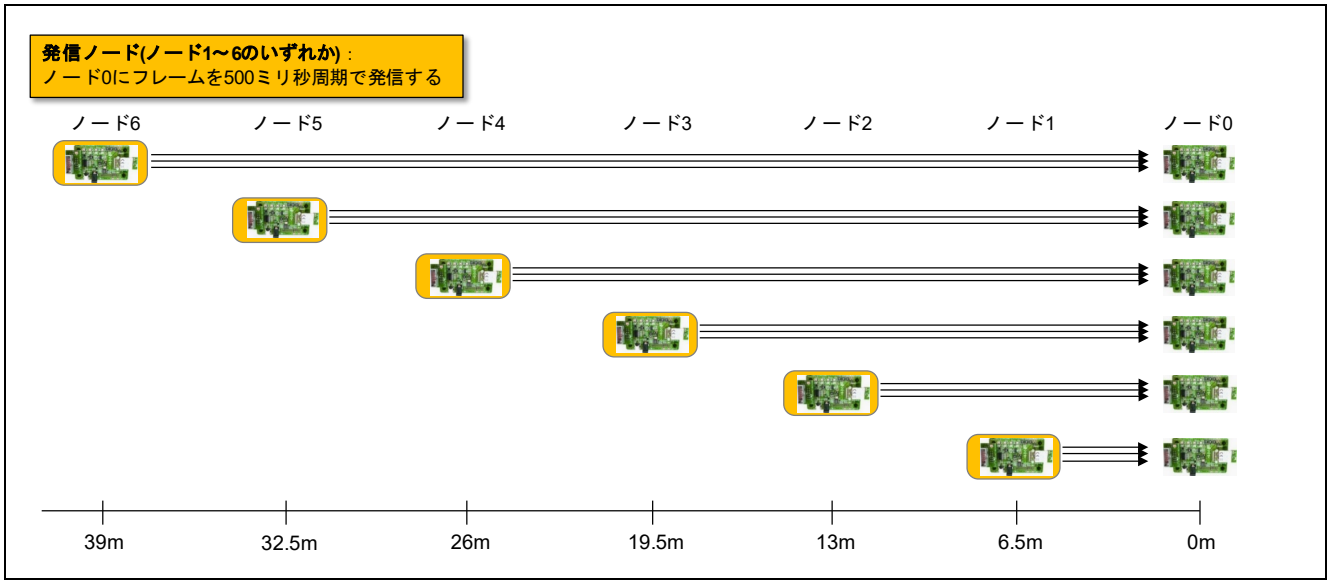


図 9-6　中継ノードがない構成

(2)　測定結果

発信ノード 0 から 500 ミリ秒周期でノード 0 にフレームを送信し、中継ノードがない構成とある構成でのフレーム到達率の測定結果を**表 9-2** と**図 9-7** に示します。

表 9-2　中継ノードの有無によるフレーム到達率

宛先ノード	中継ノードなし	中継ノードあり
ノード 1	99.5%	
ノード 2	100.0%	100.0%
ノード 3	70.5%	100.0%
ノード 4	0.0%	100.0%
ノード 5	0.0%	100.0%
ノード 6	1.0%	100.0%

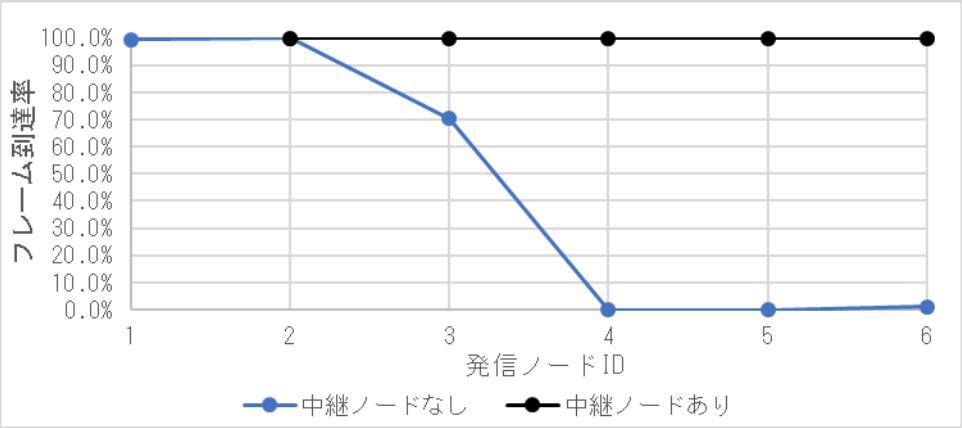


図 9-7　中継ノードの有無によるフレーム到達率

本測定結果において中継ノードがない構成では、発信ノードとノード 0 の距離が長くなるにつれ、フレームの到達率が低下し、ノード 4～ノード 6 から発信した場合ではフレーム到達率がほぼ 0%となりました。

これに対して中継ノードがある構成では、フレームが中継されることにより、ノード 3～ノード 6 から発信した場合でもフレーム到達率は 100%となりました。

9.4.4　フレーム送信周期の変更によるフレーム到達率

前項では発信ノードからの送信周期を 500 ミリ秒として測定しました。本項ではフレーム送信周期を短縮してデータの転送レートを向上させる変更例を示します。

(1)　測定手順

9.4.3 項に示した測定条件において、発信ノードのフレーム送信周期を変更した場合のフレーム到達率を測定しました。測定では各発信ノードからのフレーム送信周期を 100 ミリ秒、250 ミリ秒、500 ミリ秒周期としました。なお本測定に使用したサンプルプログラムでは、フレーム発信、フレーム中継の再送間隔を短縮するため、下記の変更を行いました。

- Project_Source¥application¥src¥r_multihop.c

変更前の r_multihop.c

```
 71:    /* unit:0.5msec value:120msec */
 72:    #define MH_TX_INTERVAL      (240)
 73:
 74:    /* unit:0.5msec resolution:1.5msec range:30msec to 58.5msec */
 75:    #define MH_RAND_RANGE       (57)
 76:    #define MH_RAND_OFFSET      (60)
 77:    #define MH_RAND_TIME()      (((((uint16_t)rand()) & 0xFF) * MH_RAND_RANGE * 3)
 78:    #define MH_RAND_MAX         (MH_RAND_RANGE + MH_RAND_OFFSET)
-------------------------------------------------------------------------------
 724:    R_HopTimer_Start(idx, gs_hop_buf[idx].delay);
```

変更後の r_multihop.c

```
 71:    /* unit:0.5msec value:10msec */
 72:    #define MH_TX_INTERVAL      (20)
 73:
 74:    /* disable random time */
 75:    #define MH_RAND_RANGE       (0)     /* 0msec */
 76:    #define MH_RAND_OFFSET      (0)     /* 0msec */
 77:    #define MH_RAND_TIME()      (0)     /* 0msec */
 78:    #define MH_RAND_MAX         (0)
-------------------------------------------------------------------------------
 724:    R_HopTimer_Start(idx, MH_TX_INTERVAL);
```

(2)　測定結果

発信ノードから 100 ミリ秒、250 ミリ秒、500 ミリ秒周期でフレームを送信した場合のフレーム到達率の測定結果を**表 9-3** に示します。

表 9-3　フレーム送信周期の変更によるフレーム到達率

発信ノード	送信周期 100 ミリ秒	送信周期 250 ミリ秒	送信周期 500 ミリ秒
ノード 2	99.7%	100.0%	100.0%
ノード 3	99.7%	100.0%	100.0%
ノード 4	99.3%	99.7%	100.0%
ノード 5	100.0%	100.0%	100.0%
ノード 6	100.0%	100.0%	100.0%

本測定結果では発信ノードからのフレーム送信周期を短縮した場合、フレーム到達率に若干の低下がありました。送信周期は 250 ミリ秒以上の時間を空ける必要がある結果となります。

フレームを短時間に送信すると、発信ノードや中継ノードが送信するフレーム同士の衝突や、同一周波数帯を使用した無線規格によって連続送信されたフレームとの衝突の確率が上昇し、結果としてフレーム到達率が低下した可能性があります。

また本測定結果は小数ノードを一列に配置して測定したものですが、大多数ノードが高密度で配置されるネットワークでは、各ノードが送信するフレーム同士が衝突しやすくなるため、フレーム到達率が低下する可能性があることに留意が必要です。

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	2018.10.31	・新規発行

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/