

RL78/G22

LTE MQTT 通信

要旨

本アプリケーションノートでは、RL78/G22 と RYZ024A を使用して、LTE 通信を行う方法を説明します。RYZ024A は LTE Cat M1/NB1/NB2 通信が可能なモジュールであり、ホスト MCU と UART 通信を介して接続され、AT コマンドによって制御されます。RL78/G22 は RYZ024A のホスト MCU として動作します。提供されるサンプルアプリケーションでは、RL78/G22 をホスト MCU として活用し、RYZ024A を制御して MQTT 通信や低消費電力動作（eDRX、PSM）を行うためのプログラムを提供します。また、AT コマンドマネジメントフレームワークを使用して、RL78/G22 から RYZ024A へ AT コマンドを送信するプログラムを実装しています。AT コマンドマネジメントフレームワークを利用することで、RYZ024A の LTE Cat M1 通信機能がサポートする様々な通信プロトコルを活用するアプリケーションを実装することが可能です。本アプリケーションノートでは、本サンプルアプリケーションに実装された MQTT 通信アプリケーションと AT コマンドマネジメントフレームワークの説明を詳しく行います。

PMOD Expansion Board for RYZ024A ([RTKYZ024A0B00000BE](#)) (以後、PMOD-RYZ024A) に付属の Truphone SIM カードをご使用の場合は、SIM カードの Activation が必要です。「RA6M5 Group RYZ024A PMOD LTE Connectivity with RA6M5 MCU Quick Start Guide」(R21QS0007)を参照して SIM カードの Activation を行ってください。

動作確認デバイス

RL78/G22

RYZ024A

関連ドキュメント

- RL78/G22 ユーザーズマニュアル ハードウェア編 (R01UH0978)
- RL78/G22 Fast Prototyping Board ユーザーズマニュアル (R20UT5121)
- RL78/G22 Fast Prototyping Board Quick Start Guide (R20UT5123)
- RYZ024 Use Case for AT Commands(R19AN103)
- RYZ024 Modules AT Command User's Manual(R11UZ0110)
- RYZ024 Module System Integration Guide (R19AN0101)
- RA6M5 Group RYZ024A PMOD LTE Connectivity with RA6M5 MCU Quick Start Guide (R21QS0007)

Pmod™ は、Digilent Inc.の商標です。

目次

1. 概要	4
1.1 動作概要	4
1.2 ソフトウェア説明	5
1.2.1 サンプルプログラムの構成	5
1.2.2 サンプルプログラムの階層	5
1.2.3 使用する周辺機能	6
1.2.4 オプション・バイトの設定一覧	6
1.2.5 フォルダ/ファイル構成	7
1.2.6 コードサイズ	7
2. MQTT 通信アプリケーションの動作	8
2.1 アプリケーションの動作環境	8
2.2 アプリケーションの動作概要	16
2.2.1 1セットでの動作	16
2.2.2 2セットでの動作	18
3. AT コマンドマネジメントフレームワーク	21
3.1 フレームワーク概要	21
3.2 API 関数	23
3.2.1 マネジメント API	23
3.2.1.1 R_LTE_Init	24
3.2.1.2 R_LTE_Execute	24
3.2.2 AT コマンド API	25
3.2.2.1 R_LTE_OM_Config	26
3.2.2.2 R_LTE_NWK_Connect	27
3.2.2.3 R_LTE_NWK_Disconnect	27
3.2.2.4 R_LTE_MQTT_Connect	28
3.2.2.5 R_LTE_MQTT_Subscribe	28
3.2.2.6 R_LTE_MQTT_Publish	29
3.2.2.7 R_LTE_MQTT_RcvMessage	30
3.2.2.8 R_LTE_MQTT_Disconnect	30
3.2.2.9 R_LTE_SEC_CertificateAdd	31
3.2.2.10 R_LTE_SEC_CertificateRemove	31
3.2.2.11 R_LTE_SEC_PrivateKeyAdd	32
3.2.2.12 R_LTE_SEC_PrivateKeyRemove	32
3.2.2.13 R_LTE_NWK_ConnectionConfig	33
3.2.2.14 R_LTE_eDRX_Config	34
3.2.2.15 R_LTE_PSM_Config	36
3.3 コールバック関数	38
3.4 ユーザ固有値の設定	44
3.5 フレームワーク内で使用されるスマート・コンフィグレータモジュール	47
3.5.1 UARTA モジュール	47
3.5.2 TAU モジュール	48
3.5.3 割り込み機能	48
3.5.4 UART0 モジュール	48

3.6	低消費電力動作	49
3.6.1	RL78/G22 の低消費電力動作制御	49
3.6.2	RYZ024A の低消費電力動作制御	50
4.	AT コマンドマネジメントフレームワークを利用したアプリケーション開発	52
4.1	アプリケーション開発の概要	52
4.2	AT コマンド API の追加	54
4.2.1	データ受信を伴う AT コマンド API	58
4.3	エラー発生処理のガイドライン	59
4.4	PMOD-RYZ024A 固有の処理	62
4.5	PMOD-RYZ024A の初期化	64
	改訂記録	65

1. 概要

1.1 動作概要

RYZ024A は LTE Cat M1 通信機能を持つモジュールです。この機能は RL78/G22 から UART 経由で AT コマンドを文字列として入力することで制御することができます。

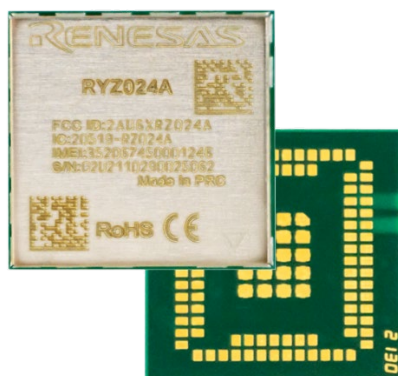


図 1-1 RYZ024A

本サンプルアプリケーションは、ホスト MCU として RL78/G22 を使用し RYZ024A の LTE Cat M1 通信を制御するソフトウェアです。RL78/G22 は UART 通信で AT コマンドを文字列として RYZ024A へ送信します。AT コマンドに対する応答文字列も UART 通信で受信します。これらのやり取りを介して RL78/G22 は RYZ024A の LTE Cat M1 通信機能を利用します。

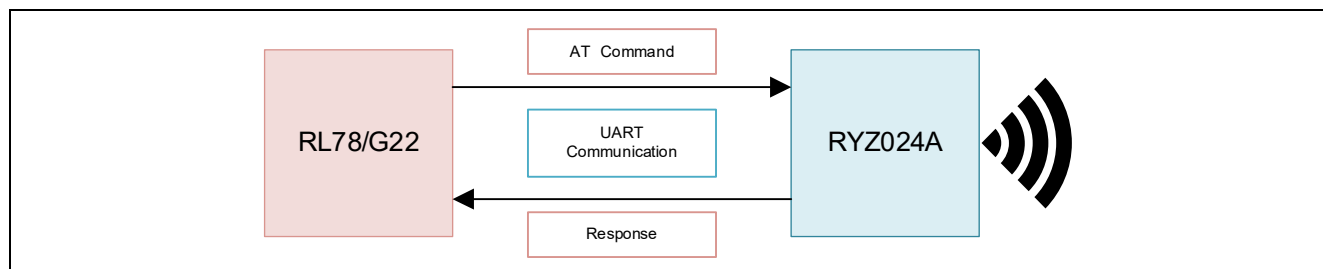


図 1-2 RL78/G22 と RYZ024A の通信

※ PMOD-RYZ024A のご使用について：

RYZ024A はディープスリープに移行すると、CTS 端子は Hi-Z 状態となります。しかし、PMOD-RYZ024A では、使用しているレベルシフタの特性によりレベルシフタからホスト MCU への CTS 信号が Low レベルのままになります。(RXD 端子も Low のままになります。) よって、RYZ024A をディープスリープから起床させてマイコンから UART 送信する時には注意が必要です。
本サンプルアプリケーションでは PMOD-RYZ024A で動作するように専用の処理を追加しています。詳しくは「4.4 PMOD-RYZ024A 固有の処理」を参照してください。

1.2 ソフトウェア説明

このサンプルアプリケーションでは、Publish/Subscribe の動作を確認するためのサンプルプログラムを提供しています。このサンプルプログラムは、1つのセットのみで Publish/Subscribe の動作を確認することもできます（2.2.1 1 セットでの動作）し、2つのセットを使用し、お互いに Publish/Subscribe の動作を確認することもできます（2.2.2 2 セットでの動作）。

1.2.1 サンプルプログラムの構成

本サンプルプログラムの構成を示します。

表 1-1 本アプリケーションノートの内容物

ファイル名	説明
r01an6951xxrrrr--lte-sample.pdf xx: 言語、作成地域 rrrr: リビジョン番号	本ドキュメント
sample_rl78g22_ryz024a	publish/subscribe サンプルアプリケーションプログラム

1.2.2 サンプルプログラムの階層

サンプルアプリケーションのソフトウェア階層は以下の通りです。

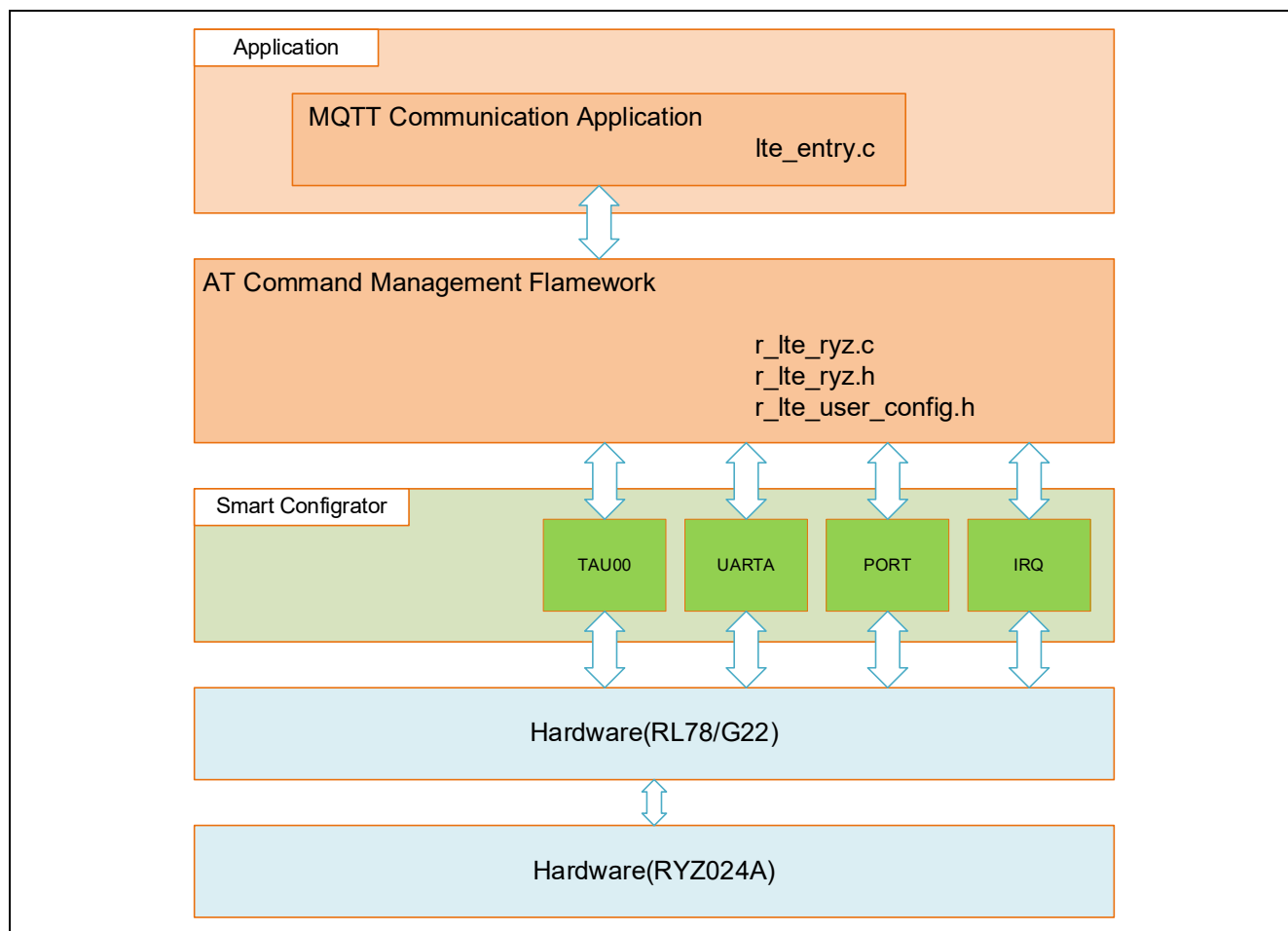


図 1-3 ソフトウェア構成

本サンプルアプリケーションは RL78/G22 から RYZ024A を制御して MQTT 通信を行うプログラムです。このプログラムは RYZ024A へ AT コマンドを送信するための AT コマンドマネジメントフレームワークと MQTT 通信するための API をコールする MQTT 通信アプリケーションの 2 つで構成されています。

MQTT 通信アプリケーションは MQTT サーバに接続した後、ユーザスイッチを押下することでデータを送信するプログラムです。MQTT 通信アプリケーションは AT コマンドマネジメントフレームワークの API を利用して作成されています。MQTT 通信アプリケーションの詳しい説明については「2 MQTT 通信アプリケーション」を参照してください。

AT コマンドマネジメントフレームワークは RYZ024A への AT コマンド送信と RYZ024A から受信したレスポンスを処理するためのフレームワークです。アプリケーションから AT コマンドマネジメントフレームワークの API 関数をコールすることで複数の AT コマンドを RYZ024A へ送信し、実行結果をコールバック関数でアプリケーションへ通知します。

本サンプルアプリケーションでは RL78/G22 が RYZ024A を通じて MQTT 通信ができるように AT コマンドマネジメントフレームワークを使用してフレームワークベースプログラムを作成しています。なお、AT コマンドマネジメントフレームワークは MQTT 通信以外の RYZ024A の機能を利用する場合も、そのアプリケーション開発を行う場合のベースとして利用されることを想定しています。AT コマンドマネジメントフレームワークの詳しい説明については「3 AT コマンドマネジメントフレームワーク」を参照して下さい。

1.2.3 使用する周辺機能

本サンプルプログラムで使用する周辺機能を以下に示します。

表 1-2 使用する周辺機能一覧

周辺機能	用途
TAU00	AT コマンド通信タイムアウト
UARTA(P72/TxDA0)	UART TX
UARTA(P71/RxDA0)	UART RX
PORT (P70)	UART 用 RTS 信号
PORT (P50)	UART 用 CTS 信号
PORT (P17)	RYZ024A RESET 制御
SW(P137/INTP0)	ユーザスイッチの外部端子割り込み
INTP(P51/INTP2)	RYZ024A RING 信号の外部端子割り込み
UART(P12/TxD0)	モニタログ出力用 UART TX

1.2.4 オプション・バイトの設定一覧

本サンプルプログラムのオプション・バイト設定を以下に示します。

表 1-3 オプション・バイト設定

アドレス	設定値	内容
000C0H / 020C0H	11101111B	ウォッチドッグ・タイマ動作停止 (リセット解除後、カウント停止)
000C1H / 020C1H	11111100B	LVDD 検出電圧：リセット・モード 立ち上がり時 TYP. 2.67V(2.59V ~ 2.75V) 立ち下がり時 TYP. 2.62V(2.54V ~ 2.70V)
000C2H / 020C2H	11101000B	HS モード、 高速オンチップ・オシレータ・クロック：32MHz
000C3H / 020C3H	10000100B	オンチップ・デバッグ許可

1.2.5 フォルダ/ファイル構成

本アプリケーションノートで提供するサンプルアプリケーションプログラムのファイル構成は以下の通りです。

表 1-4 ファイル構成

フォルダ名、ファイル名	説明
r01an6951_rl78g22_itemqtt	プログラム格納用フォルダ
└ src	ソースファイル
├ └ ryz	RYZ024A 関連ソースファイル
├ └ smc_gen	スマート・コンフィグレータ生成
├ └ └ Config_INTC	
├ └ └ Config_PORT	
├ └ └ Config_TAU0_1	
├ └ └ Config_UART0	
├ └ └ Config_UARTA0	
├ └ └ general	
├ └ └ r_bsp	
├ └ └ r_config	
├ └ └ r_pincfg	
├ └ lte_entry.c	サンプルアプリケーションのメインプログラム
├ └ lte_entry.h	
├ └ main.c	
├ └ main.h	

1.2.6 コードサイズ

本アプリケーションノートで提供するサンプルアプリケーションプログラムの ROM/RAM サイズを示します。

表 1-5 コードサイズ

リソース	サイズ
ROM	23,700 Byte
RAM	2,394 Byte

2. MQTT 通信アプリケーションの動作

2.1 アプリケーションの動作環境

MQTT 通信アプリケーションを動作させるための環境について説明します。
本サンプルアプリケーションプログラムは以下のハードウェア環境で動作します。

表 2-1 サンプルアプリケーションのハードウェア環境

ハードウェア名	説明
RL78/G22 Fast Prototyping Board (RL78/G22 FPB)	RL78/G22 搭載評価ボード (RTK7RLG220C00000BJ)
PMOD Expansion Board for RYZ024A (PMOD-RYZ024A)	RYZ024A モジュール搭載 PMOD ボード (RTKY024A0B00000BE)
Windows PC	RL78/G22 のアプリ開発環境及び動作確認用デバッグコンソール

※ PMOD-RYZ024A のご使用について：

RYZ024A はディープスリープに移行すると、CTS 端子は Hi-Z 状態となります。しかし、PMOD-RYZ024A では、使用しているレベルシフタの特性によりレベルシフタからホスト MCU への CTS 信号が Low レベルのままになります。(RXD 端子も Low のままになります。) よって、RYZ024A をディープスリープから起床させてマイコンから UART 送信する時には注意が必要です。
本サンプルアプリケーションでは PMOD-RYZ024A で動作するように専用の処理を追加しています。詳しくは「4.4 PMOD-RYZ024A 固有の処理」を参照してください。

本サンプルアプリケーションは以下のソフトウェア環境で開発と動作確認を行っています。

表 2-2 サンプルアプリケーションのソフトウェア環境

項目	内容
統合開発環境 (e ² studio)	ルネサス エレクトロニクス製 e ² studio V2024-01 (24.1.0)
C コンパイラ (e ² studio)	ルネサス エレクトロニクス製 CC-RL V1.13.00
統合開発環境 (CS+)	ルネサス エレクトロニクス製 CS+ V8.11
C コンパイラ (CS+)	ルネサス エレクトロニクス製 CC-RL V1.13.00
スマート・コンフィギュレータ(SC)	ルネサス エレクトロニクス製 V1.9.0
ボードサポートパッケージ (BSP)	ルネサス エレクトロニクス製 V1.62
Renesas Flash Programmer (RFP)	ルネサス エレクトロニクス製 V3.14.00

以下の手順でサンプルアプリケーションの実行準備を行います。

1. RL78/G22 FPB と PMOD-RYZ024A を PMOD コネクタで接続します。
この時、RL78/G22 FPB の PMOD2 に接続します。

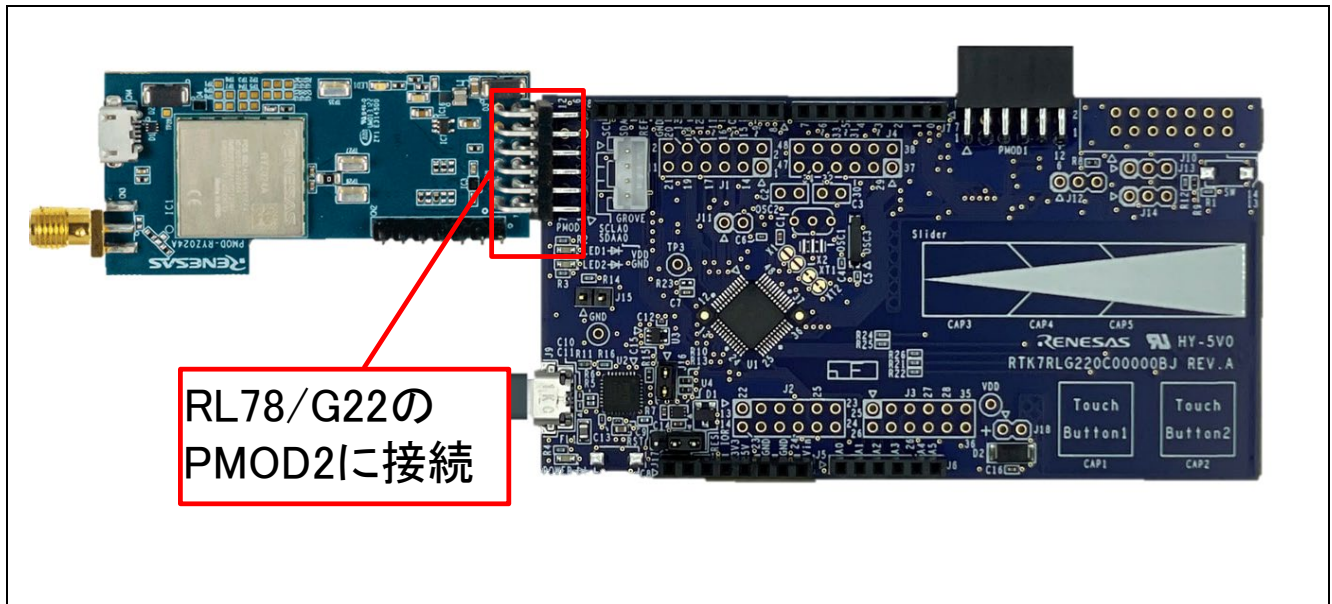


図 2-1 RL78/G22 FPB と PMOD-RYZ024A を接続

2. RL78/G22 FPB と PMOD-RYZ024A に USB ケーブルを接続します。
また、PMOD-RYZ024A にアンテナを接続します

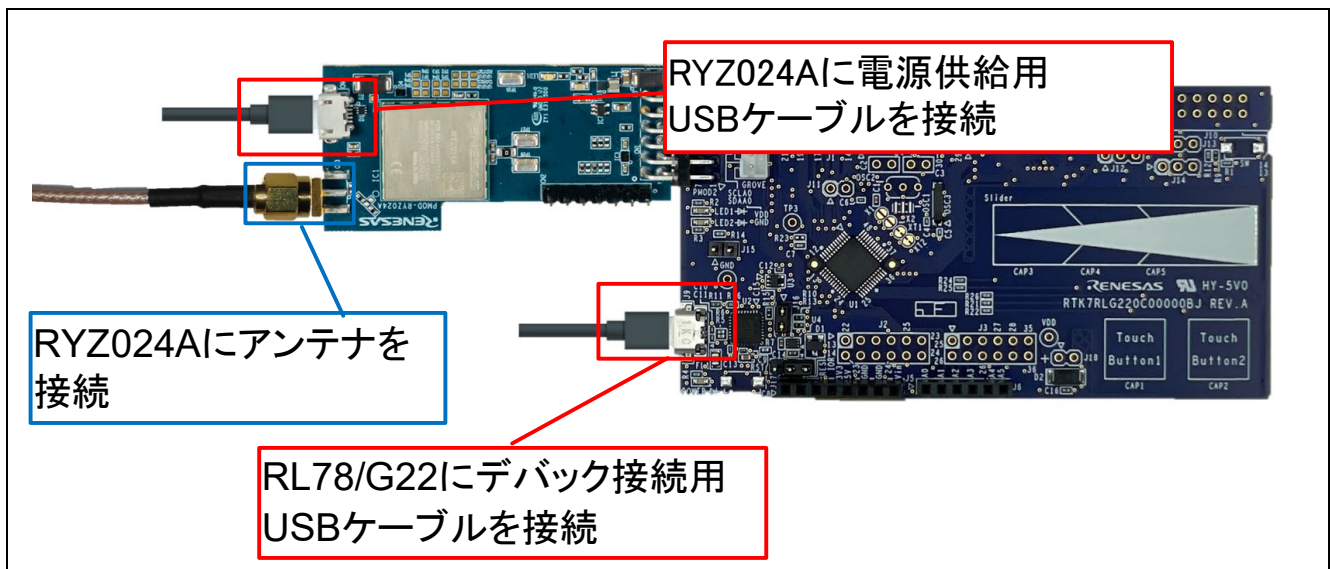


図 2-2 USB ケーブルとアンテナの接続

3. サンプルプロジェクトをインポートします。
 サンプルプログラムは e² studio のプロジェクト形式で提供しています。e² studio および CS+へプロジェクトをインポートする方法を示します。

・ e² studio での手順

e² studio でご使用になる際は、以下の手順で e² studio にインポートしてください。なお、e² studio で管理するプロジェクトのフォルダ名、およびそのフォルダに至るファイルパスには、空白文字の他、半角カナ文字、全角文字、半角記号 (特に '\$', '#', '%') が混じらないようにしてください (使用する e2 studio のバージョンによっては画面が異なる場合があります)。

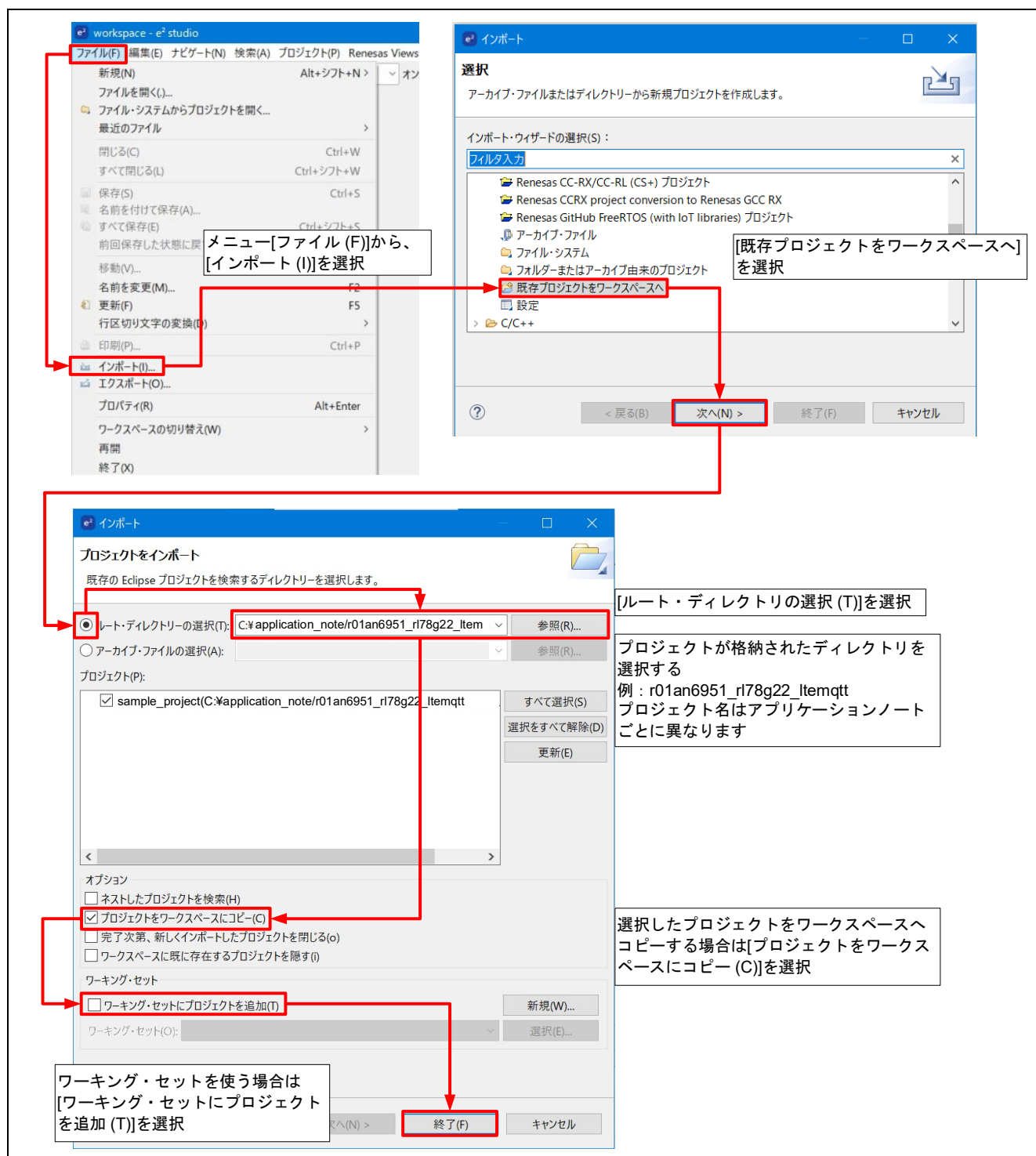


図 2-3 プロジェクトを e² studio にインポートする方法

・ CS+での手順

CS+ でご使用になる際は、以下の手順で CS+ にインポートしてください。

なお、CS+で管理するプロジェクトのフォルダ名、およびそのフォルダに至るファイルパスには、空白文字の他、半角カナ文字、全角文字、半角記号 (特に'\$','#','%') が混じらないようにしてください (使用する CS+ のバージョンによっては画面が異なる場合があります)。プロジェクトに必要なファイルを生成します。

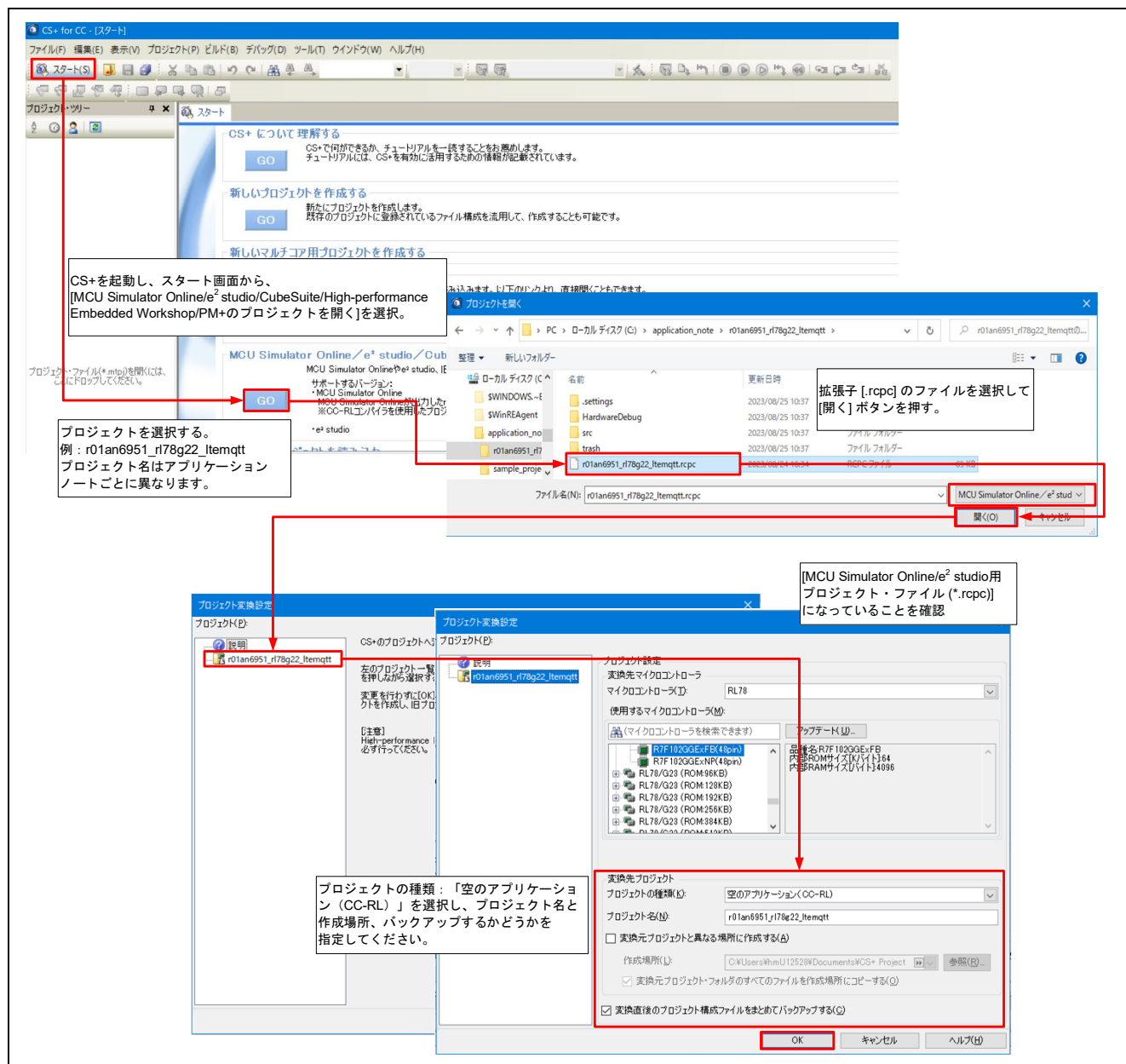


図 2-4 プロジェクトを CS+ にインポートする方法

4. アクセスポイント名、認証プロトコル、ユーザ名、パスワード、LTE バンドを変更します。LTE ネットワークへ接続する際に使用するアクセスポイント名、認証プロトコル、ユーザ名、パスワード、LTE バンドは文字列データとして指定されています。ユーザのアプリケーションに合わせてこれらの値を変更してください。変更するファイルは以下の通りです。

・ lte_entry.c

アクセスポイント名 (APN)、認証プロトコル、ユーザ名、パスワード

設定する文字列データは基本的に使用する SIM によって異なります。使用できる APN については SIM の提供元に問い合わせてください。ユーザ名、パスワードは省略されることもあります。ルネサスから提供されるキットに付属する SIM のアクティベート及び APN については各キットのマニュアルを参照してください。

LTE バンド

LTE バンドは使用する地域やオペレータなどによって異なります。使用される LTE バンドがわかっている場合そのバンドを指定してください。以下に LTE バンドの例を示します。

- “1,19”
 - DOCOMO バンドを指定する場合。
- “2,4,12”
 - AT&T バンドを指定する場合。
- “1,2,3,4,5,8,12,13,17,18,19,20,25,26,28,66”
 - 使用するバンドが不明である場合。

※注意：SIM のアクティベート時や、初回にネットワークに接続するとき、接続までに数分の時間がかかる場合があります。なお、使用するバンドを制限することは、接続までの時間を短くすることに有効です。

本アプリケーションでは以下の設定で動作確認を行っています。

- APN : ppsim.jp
- 認証プロトコル : 0
- ユーザ名 : -
- パスワード : -
- LTE バンド : 1,19

```

36
37 * @addtogroup r01an6951_rl78g22_ltemqtt
38 * @{
39
40
41 /* Application value definition */
42 extern uint8_t g_rbuf_guard[10];
43
44 uint8_t g_sw1_push_flag = 0;
45 uint8_t g_sw2_push_flag = 0;
46
47 static uint8_t s_sbuf[100];
48 static uint8_t s_sw1_count = 0;
49 static uint8_t s_reinitialize_flag = 0;
50 static uint8_t s_mqtt_conn_state = 0;
51
52 /* Application specific string data for network connection */
53 static uint8_t s_str_pdp_type[] = "IPv4V6";
54
55 /* Access Point Name using in network connection. Please select depending on SIM card */
56 static uint8_t s_str_pdp_apn[] = "ppsim.jp"; // ex. globaldata.iot , ibasis.iot , soracom.io
57
58 /* Authentication protocol, user id, password. Please modify depending on using LTE band */
59 static uint8_t s_str_pdp_protocol[] = "2";
60 static uint8_t s_str_pdp_userid[] = "pp@sim";
61 static uint8_t s_str_pdp_password[] = "jpn";
62
63 /* Band List for network connection. Please select depending on using LTE band */
64 static uint8_t s_str_lte_bandlist[] = "1,19"; // Docomo bands in JP Re
65
66 /* static uint8_t s_str_lte_bandlist[] = "2,4,12"; AT&T bands in US Region */
67 /* static uint8_t s_str_lte_bandlist[] = "1,2,3,4,5,8,12,13,17,18,19,20,25,26,28,66"; All selectable bands. */

```

APN、プロトコル、ユーザ名、パスワードを変更する。

LTEバンドを変更する

図 2-5 APN、認証プロトコル、ユーザ名、パスワード、LTE バンドを変更 (lte_entry.c)

5. RL78/G22 FPB の低消費電力動作を無効にします。本サンプルプログラムの動作をモニタリングする dynamic printf を使用する際は、RL78/G22 FPB の低消費電力モードの使用が推奨されていないため、低消費電力モードを実行する API(ryz_lte_lowpower_devspe)をコメントアウトします。以下に示す関数の中で ryz_lte_lowpower_devspe() を呼び出している行をコメントアウトしてください。

ファイル名/関数名 : r_lte_ryz.c / void R_LTE_Execute(void)

```

1237     }
1238     ryz_lte_lowpower_devspe(lpm_mode);
1239 }

```

図 2-6 低消費電力無効化

6. モトローラ・Sタイプ・ファイル出力指定を行います。
 - (1): プロジェクト上でマウスを右クリック
 - (2): プロパティを選択
 - (3): 設定⇒Converter の出力を選択し、「コード・モジュール・コンバータを実行する」にチェックを入れ、モトローラ・Sタイプ・ファイルを出力するが選択されていることを確認し、適用して閉じるをクリック

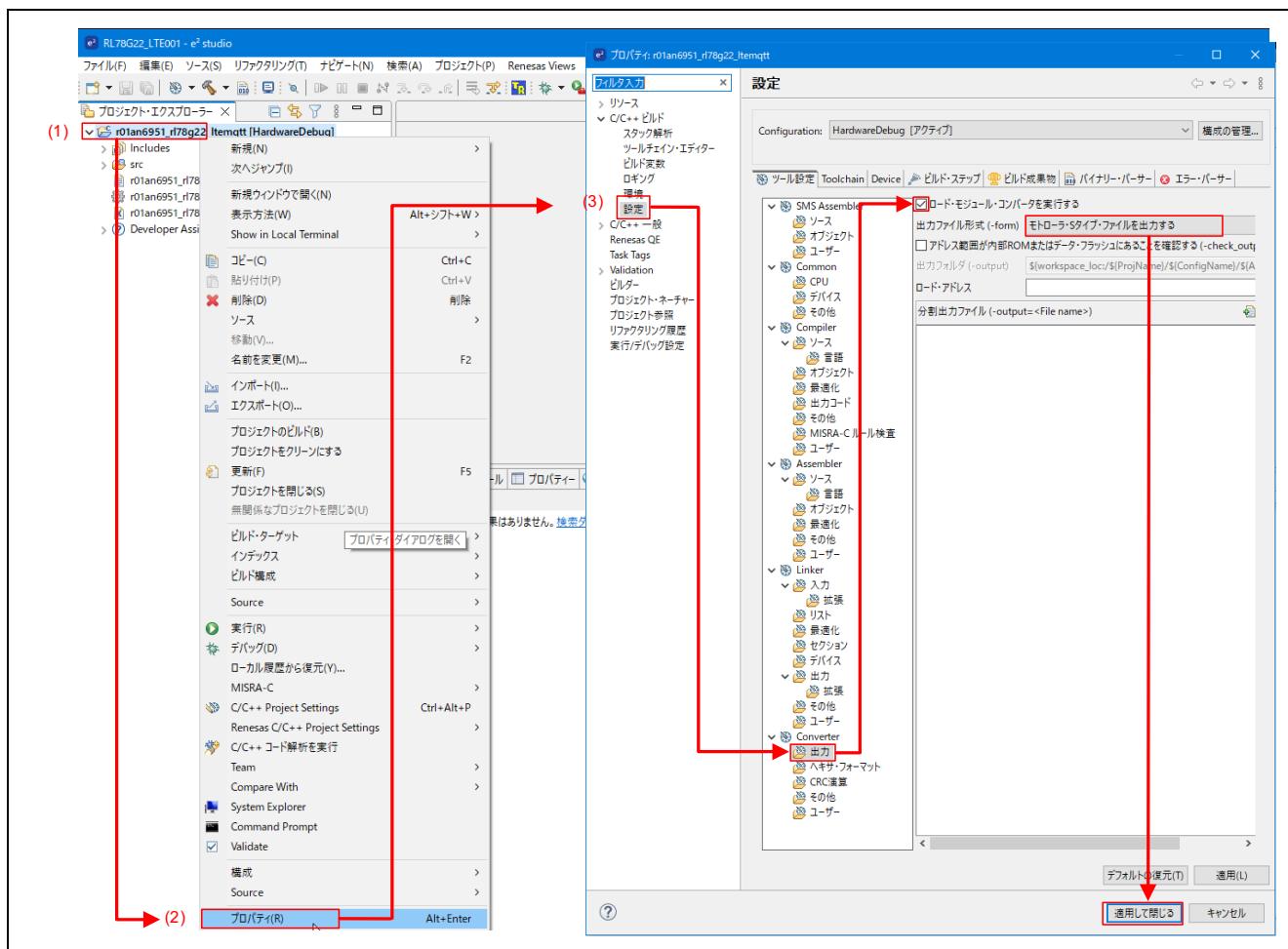


図 2-7 モトローラ・Sタイプ・ファイル出力設定

7. サンプルプロジェクトをビルドします。ビルドすると、Debug フォルダの下にモトローラ・Sタイプ・ファイル (.mot) が出力されます。

8. Renesas Flash Programmer を使用し、RL78/G22 FPB にモトローラ・S タイプ・ファイルを書き込みます。
 - (1) : ファイル⇒新しいプロジェクトを選択
 - (2) : プルダウンメニューより、RL78/G2x を選択
 - (3) : プロジェクト名、及び、作成場所に任意の名称、任意の作業場所を指定
 - (4) : プルダウンメニューより、COM port、及び、2wire UART を選択
 - (5) : ツール詳細をクリック
 - (6) : RL78/G22 FPB の COM ポートを指定
 - (7) : OK をクリック
 - (8) : 接続をクリック（正常に接続できると、(9)に進みます。エラーの場合は、(6)以前の設定をご確認ください
 - (9) : ビルドで生成したモトローラ・S タイプ・ファイルを指定
 - (10) : スタートをクリックすると、書き込みを開始

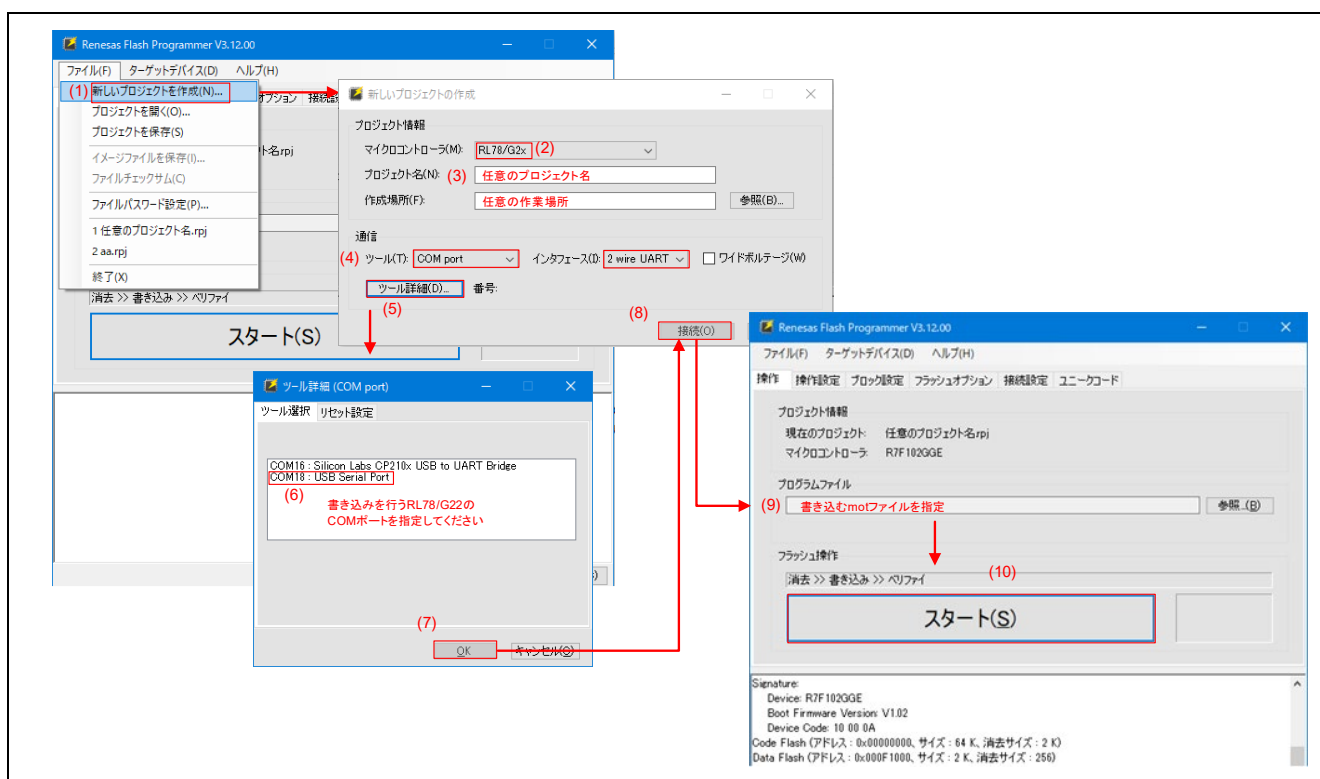


図 2-8 ファイルの書き込み

9. モニタリングの準備をします。
 サンプルプログラムの実行結果は、RL78/G22 FPB の USB から出力されます。サンプルプロジェクトの書き込みが終わったら、TeraTerm などのターミナルソフトで、RL78/G22 FPB の COM ポートを指定してください。

ボーレートは 115200bps、データは 8bit、パリティはなし、ストップビットは 1bit です。
 改行コードの受信設定は「LF」です。

2.2 アプリケーションの動作概要

本サンプルプログラムは、公開されている MQTT サーバ「test.mosquitto.org」を利用してメッセージの送受信を行います。この章では提供されるプログラムの動作について説明します。本サンプルプログラムは、1 セットのみで Publish/Subscribe の動作を確認することもできますし、2 セットを用いて、お互いに Publish/Subscribe の動作を確認することもできます。

2.2.1 1 セットでの動作

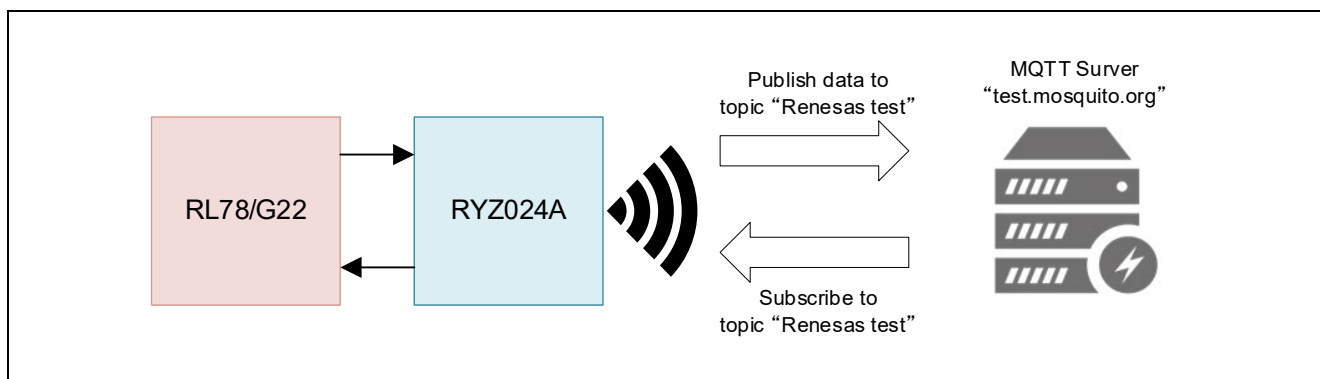


図 2-9 1 セットでのサンプルプログラムシステム構成

サンプルプログラムが実行されるとまず、RL78/G22 は、RYZ024A モジュールをリセットします。RYZ024A のリセット後、LTE 経由でネットワークに接続し、続いて MQTT サーバに接続します。次に MQTT サーバに Subscribe 要求を行った後、「SW READY」の文字列を表示します。この状態でボード上のユーザスイッチで操作可能となります。

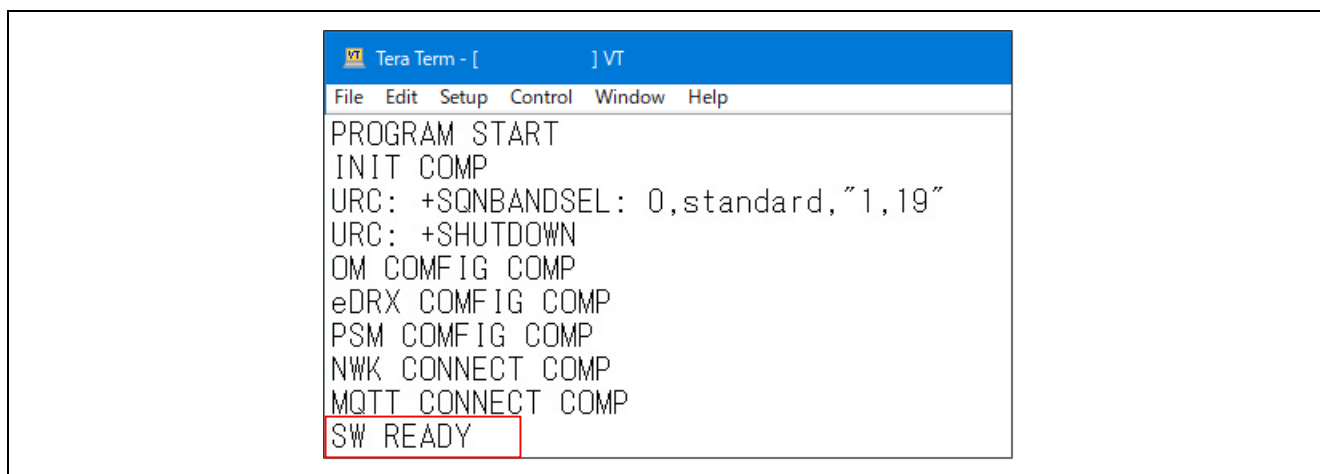
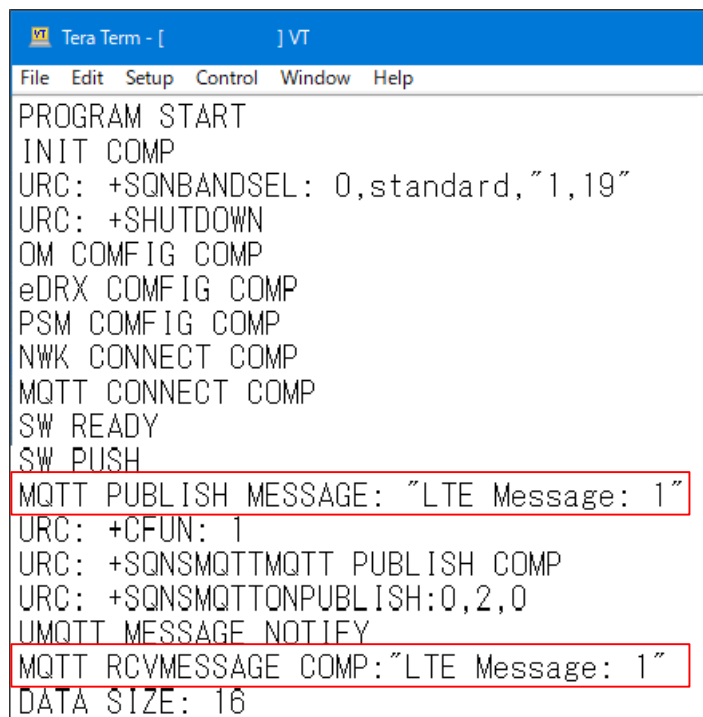


図 2-10 MQTT サーバへ接続

「SW READY」が表示されたあと RL78/G22 FPB のユーザスイッチを押すことで MQTT サーバへ Publish によりメッセージを送信します。RL78/G22 は、事前に MQTT サーバには Subscribe 要求を行っています。MQTT サーバは、Publish を受信すると、Subscriber である RL78/G22 にメッセージの ID などを送信します。RL78/G22 はメッセージ受信要求を送信し、受信したメッセージをコンソールに表示します。ユーザスイッチを押す回数によって送信する文字列データは変化します。

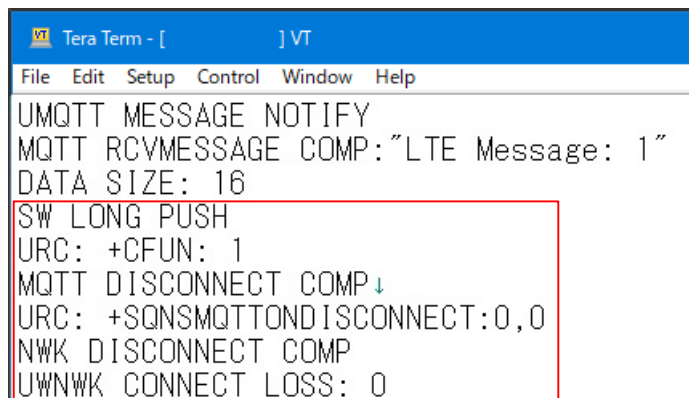


```

Tera Term - [ ] VT
File Edit Setup Control Window Help
PROGRAM START
INIT COMP
URC: +SQNBANDSEL: 0,standard,"1,19"
URC: +SHUTDOWN
OM CONFIG COMP
eDRX CONFIG COMP
PSM CONFIG COMP
NWK CONNECT COMP
MQTT CONNECT COMP
SW READY
SW PUSH
MQTT PUBLISH MESSAGE: "LTE Message: 1"
URC: +CFUN: 1
URC: +SQNSMQTTMQTT PUBLISH COMP
URC: +SQNSMQTTTONPUBLISH:0,2,0
UMQTT MESSAGE NOTIFY
MQTT RCVMESAGE COMP:"LTE Message: 1"
DATA SIZE: 16
  
```

図 2-11 ユーザスイッチを押下

ユーザスイッチを長押し（1 秒以上）すると MQTT サーバから切断した後、ネットワークからも切断します。



```

Tera Term - [ ] VT
File Edit Setup Control Window Help
UMQTT MESSAGE NOTIFY
MQTT RCVMESAGE COMP:"LTE Message: 1"
DATA SIZE: 16
SW LONG PUSH
URC: +CFUN: 1
MQTT DISCONNECT COMP↓
URC: +SQNSMQTTTNDISCONNECT:0,0
NWK DISCONNECT COMP
UWNWK CONNECT LOSS: 0
  
```

図 2-12 ユーザスイッチを長押し（1 秒以上）

なお、電波状況の悪化などの理由でネットワークや MQTT サーバから切断された場合、本サンプルアプリケーションでは再度 MQTT サーバへ接続しようとします。そのため電波状況が回復した後、ボタンなどを押すことなく MQTT サーバへ再接続し、Subscribe 要求を行った後「SW READY」が表示されます。この後スイッチの操作が可能になります。

2.2.2 2セットでの動作

2セットでの動作確認を行う場合も、同じサンプルプログラムを使用します。

ただし、lte_entry.cの「s_str_mqtt_username」を、renesas_device_001以外の値（例えば、renesas_device_002等）に変更した後、ビルドを行ってください。

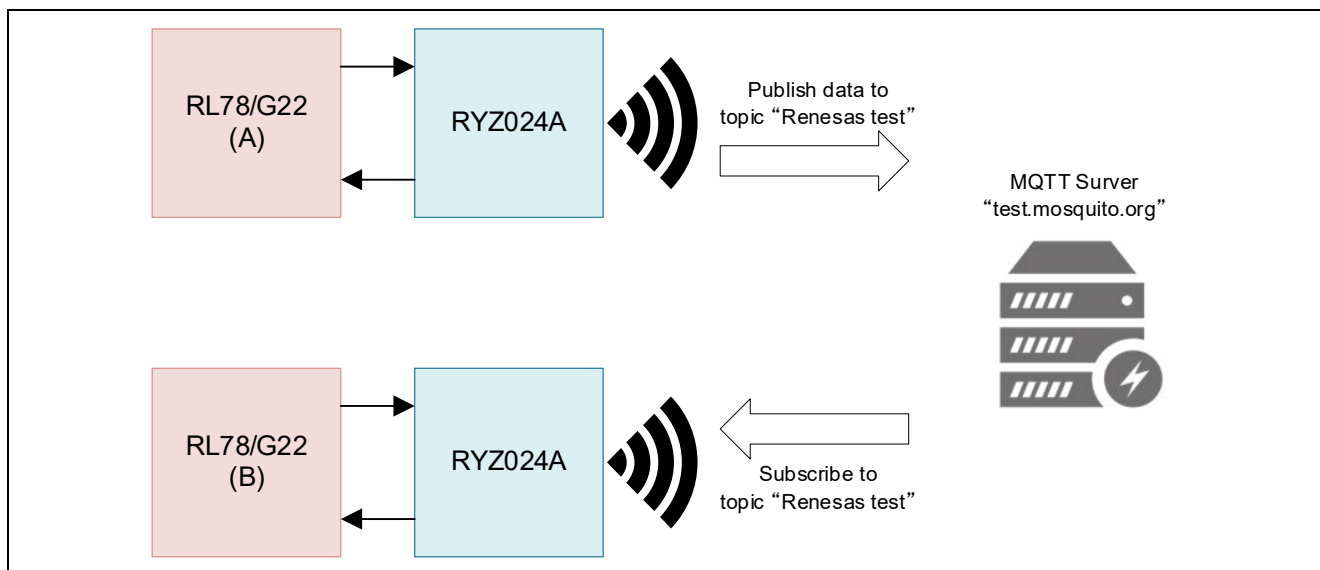


図 2-13 2 セットでのサンプルプログラムシステム構成

コードの変更箇所 (lte_entry.c)

```
/* Application specific string data for MQTT Connection */
static uint8_t s_str_mqtt_serveraddress[] = "test.mosquitto.org";
static uint8_t s_str_mqtt_serverport[] = "1883";
static uint8_t s_str_mqtt_username[] = "renesas_device_001";
static uint8_t s_str_mqtt_topic[] = "renesas_test";
```

サンプルプログラムが実行されるとまず、RL78/G22 は、RYZ024A モジュールをリセットします。RYZ024A のリセット後、LTE 経由でネットワークに接続し、続いて MQTT サーバに接続します。次に MQTT サーバに Subscribe 要求を行った後、「SW READY」の文字列を表示します。この状態でボード上のユーザスイッチで操作可能となります。

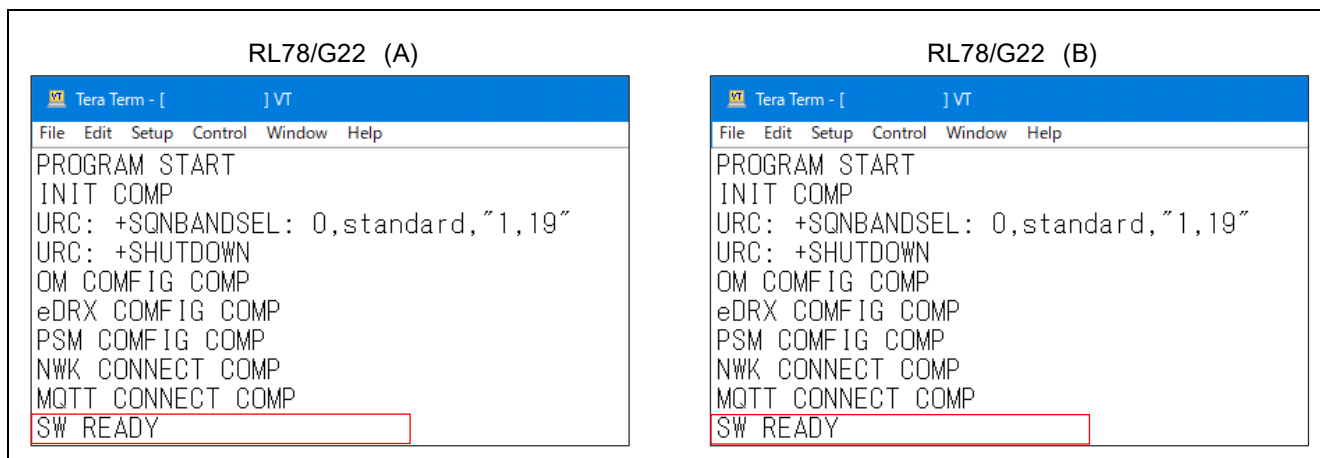


図 2-14 MQTT サーバへ接続 (2 セット)

「SW READY」と表示された後、RL78/G22 (A)のユーザスイッチを押すことで、MQTT サーバにメッセージを Publish により送信します。RL78/G22 (A)は、事前に MQTT サーバに Subscribe 要求を行っています。MQTT サーバは、Publish を受信すると Subscriber にそのメッセージ ID などを送信します。Subscriber はメッセージの受信要求を送信し、受信したメッセージをコンソールに表示します。同様に、RL78/G22 (B)も MQTT サーバに Subscribe 要求を行っているため、RL78/G22(A)と同じように、受信したメッセージをコンソールに表示します。ユーザスイッチを押す回数に応じ、送信する文字列データが変化します。

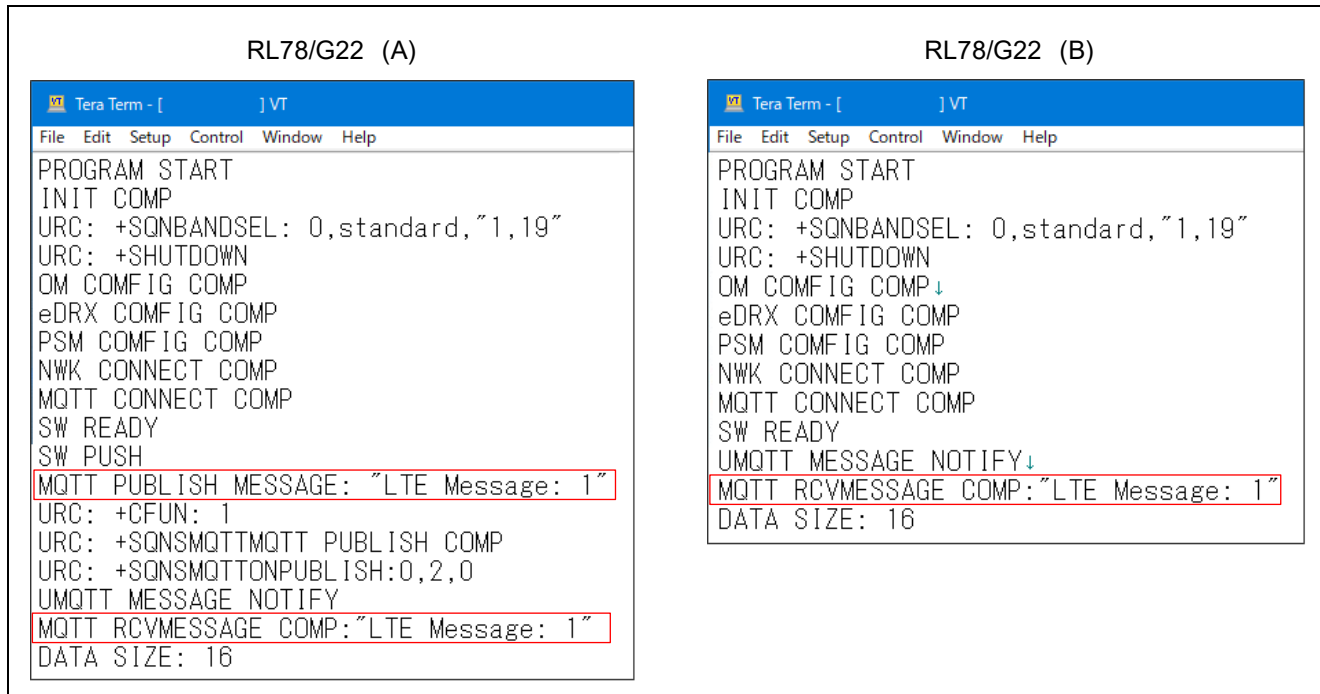


図 2-15 ユーザスイッチを押下 (2 セット)

RL78/G22 (A)のユーザスイッチを押すたびに、RL78/G22 (A)から RL78/G22 (B)にメッセージが送信されます。

RL78/G22 (A)	RL78/G22 (B)
<pre> Tera Term - [] VT File Edit Setup Control Window Help MQTT RCVMESSAGE COMP:"LTE Message: 1" DATA SIZE: 16 SW PUSH MQTT PUBLISH MESSAGE: "LTE Message: 2" URC: +CFUN: 1 URC: +SQNSMQTTMQTT PUBLISH COMP URC: +SQNSMQTTTONPUBLISH:0,3,0 UMQTT MESSAGE NOTIFY MQTT RCVMESSAGE COMP:"LTE Message: 2" DATA SIZE: 16 SW PUSH MQTT PUBLISH MESSAGE: "LTE Message: 3" URC: +CFUN: 1↓ URC: +SQNSMQTTMQTT PUBLISH COMP URC: +SQNSMQTTTONPUBLISH:0,4,0 UMQTT MESSAGE NOTIFY MQTT RCVMESSAGE COMP:"LTE Message: 3" DATA SIZE: 16 </pre>	<pre> Tera Term - [] VT File Edit Setup Control Window Help MQTT RCVMESSAGE COMP:"LTE Message: 1" DATA SIZE: 16 UMQTT MESSAGE NOTIFY MQTT RCVMESSAGE COMP:"LTE Message: 2" DATA SIZE: 16 UMQTT MESSAGE NOTIFY MQTT RCVMESSAGE COMP:"LTE Message: 3" DATA SIZE: 16 </pre>

図 2-16 RL78/G22 (A)から RL78/G22 (B)への publish (2 セット)

RL78/G22 (B)のユーザスイッチを押すと、RL78/G22 (B)から、RL78/G22 (A)にメッセージを送信することができます。

RL78/G22 (A)	RL78/G22 (B)
<pre> Tera Term - [] VT File Edit Setup Control Window Help UMQTT MESSAGE NOTIFY MQTT RCVMESSAGE COMP:"LTE Message: 3" DATA SIZE: 16 UMQTT MESSAGE NOTIFY MQTT RCVMESSAGE COMP:"LTE Message: 1" DATA SIZE: 16 UMQTT MESSAGE NOTIFY MQTT RCVMESSAGE COMP:"LTE Message: 2" DATA SIZE: 16 </pre>	<pre> Tera Term - [] VT File Edit Setup Control Window Help UMQTT MESSAGE NOTIFY MQTT RCVMESSAGE COMP:"LTE Message: 3" DATA SIZE: 16 SW PUSH MQTT PUBLISH MESSAGE: "LTE Message: 1" URC: +CFUN: 1 URC: +SQNSMQTTMQTT PUBLISH COMP URC: +SQNSMQTTTONPUBLISH:0,2,0 UMQTT MESSAGE NOTIFY MQTT RCVMESSAGE COMP:"LTE Message: 1" DATA SIZE: 16 SW PUSH MQTT PUBLISH MESSAGE: "LTE Message: 2" URC: +CFUN: 1 URC: +SQNSMQTTMQTT PUBLISH COMP URC: +SQNSMQTTTONPUBLISH:0,3,0 UMQTT MESSAGE NOTIFY MQTT RCVMESSAGE COMP:"LTE Message: 2" DATA SIZE: 16 </pre>

図 2-17 RL78/G22 (B)から RL78/G22 (A)への publish (2 セット)

3. AT コマンドマネジメントフレームワーク

3.1 フレームワーク概要

RL78/G22 から RYZ024A を操作するためには UART 通信を使用して AT コマンドを送信し、レスポンスを受信することで行います。AT コマンドマネジメントフレームワークは AT コマンドとレスポンスの送受信を効率よく実装するためのフレームワークです。本サンプルプログラムでは AT コマンドマネジメントフレームワークを使用して MQTT 通信をするためのフレームワークベースプログラムを作成しています。

本サンプルプログラムのフレームワークベースプログラムで作成されている API は、マネジメント API と AT コマンド API の 2 つに分類されます。マネジメント API はフレームワークベースプログラムの初期化や、一連の AT コマンドをレスポンスに応じて送信するための API です。AT コマンド API は AT コマンドを送信するための API です。AT コマンド API で送信された AT コマンドの実行結果はコールバック関数としてアプリケーションに通知されます。

AT コマンドマネジメントフレームワークはスマート・コンフィグレータで生成された TAU、UARTA、PORT、及び、割り込みコントローラを使用して作成されています。

- UARTA は、RYZ024A への AT コマンド送信と RYZ024A からのレスポンスの受信で使します。
- TAU は、AT コマンド実行後のタイムアウトを計測するために使します。
- 割り込みコントローラは、RYZ024A からの URC があることを通知する RING 信号の割り込みで使します。
- Low Power Module は、RL78/G22 が IDLE 状態の時や、AT コマンド API の中で AT コマンドを送信して応答を待っている状態の時に使します。

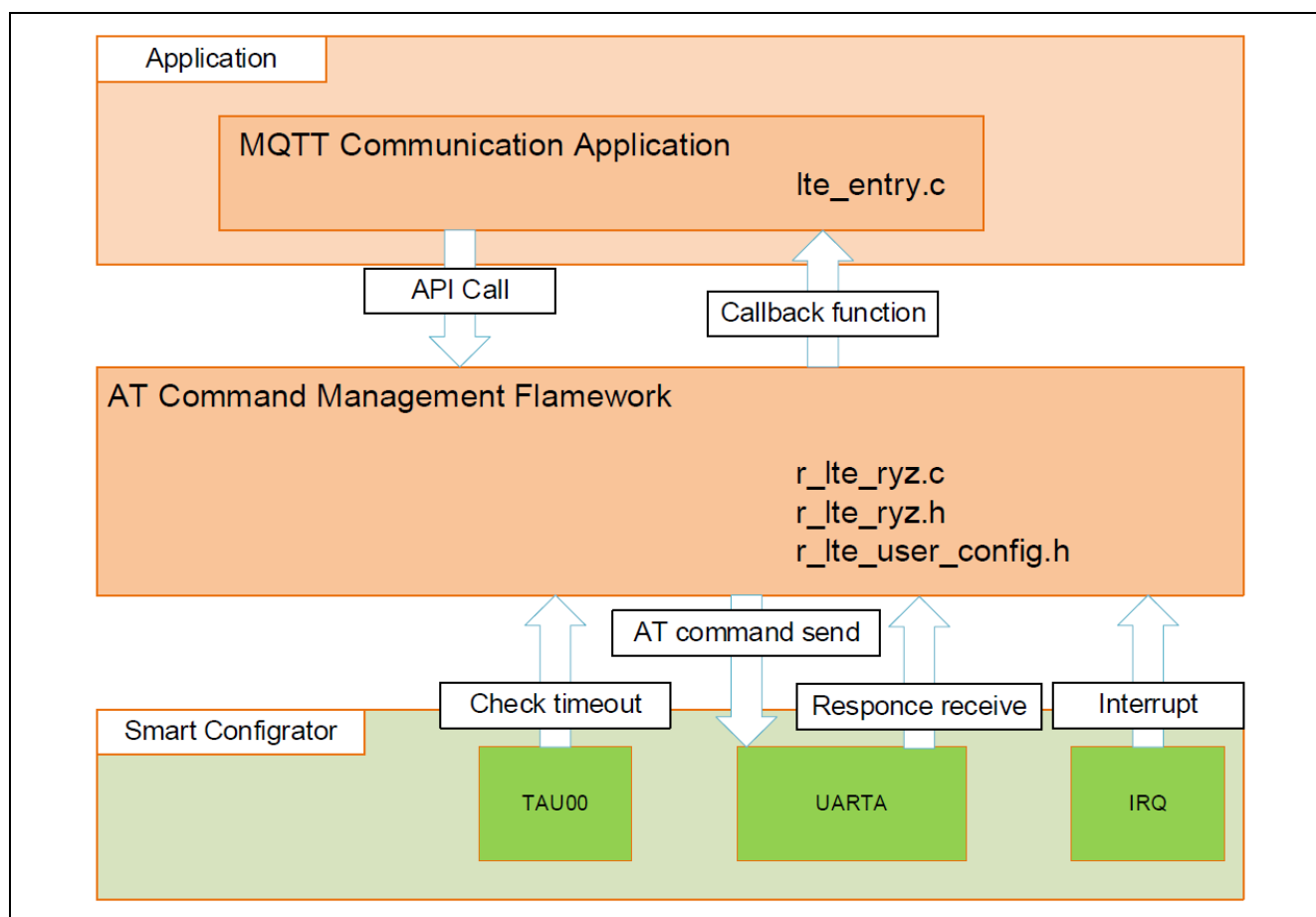


図 3-1 AT コマンドマネジメントフレームワーク

AT コマンドマネジメントフレームワークでは RYZ024A の LTE 通信機能を利用するために AT コマンドを送信する AT コマンド API を呼び出しています。アプリケーションで AT コマンド API を呼び出すことで実行したい動作を実施するために必要な一連の AT コマンドが送信待機リストに追加されます。送信待機リストに追加された AT コマンドは順番に RYZ024A へ送信されます。

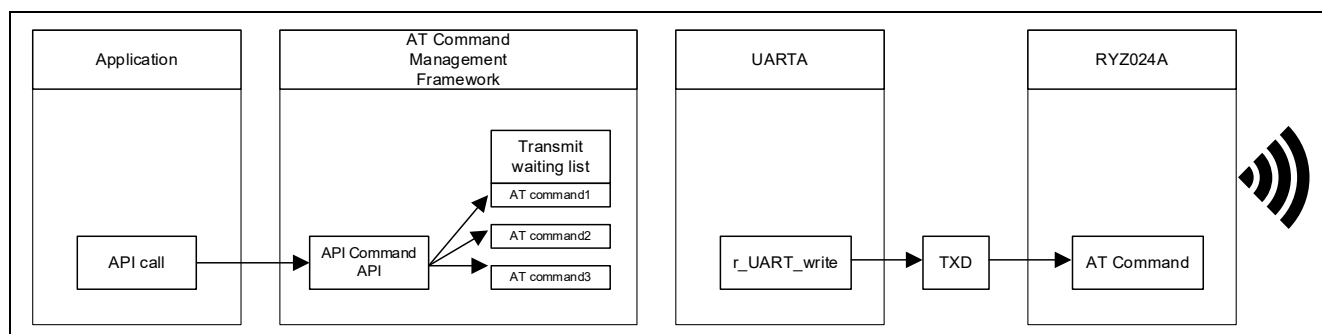


図 3-2 AT コマンド API の呼出し

RYZ024A での AT コマンドの実行結果はレスポンスとして送信されます。このレスポンスデータを UART モジュールのコールバック関数で受け取り、R_LTE_Execute 関数で解析されます。実行結果が正しい場合、R_LTE_Execute 関数は送信待機リストに追加されている次の AT コマンドを送信します。この手順は送信待機リストに追加されたすべての AT コマンドが送信されるまで繰り返されます。

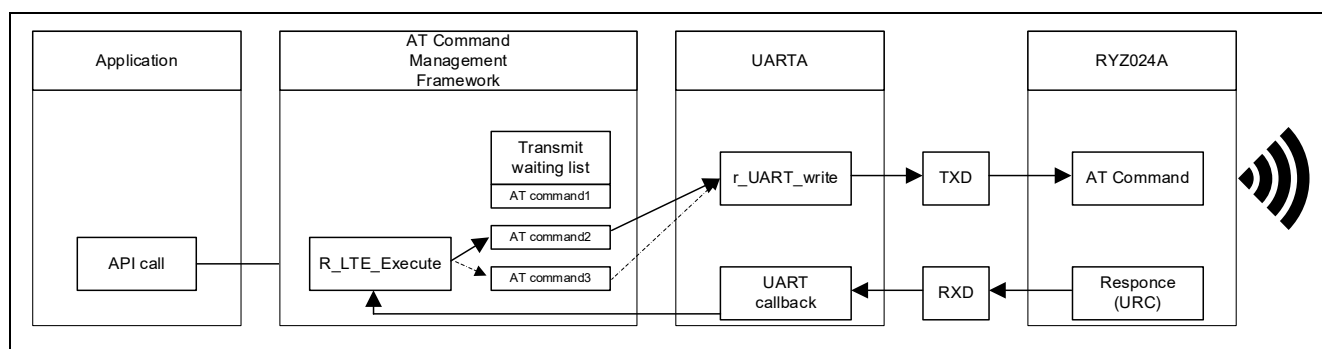


図 3-3 レスポンスの受信と AT コマンドの送信

送信待機リストに追加されたすべての AT コマンドを送信した、RYZ024A からのレスポンスがエラーだった、その他 AT コマンドと関係のないデータを受信したなどの場合、R_LTE_Execute 関数はそれらをアプリケーションにコールバック関数として通知します。

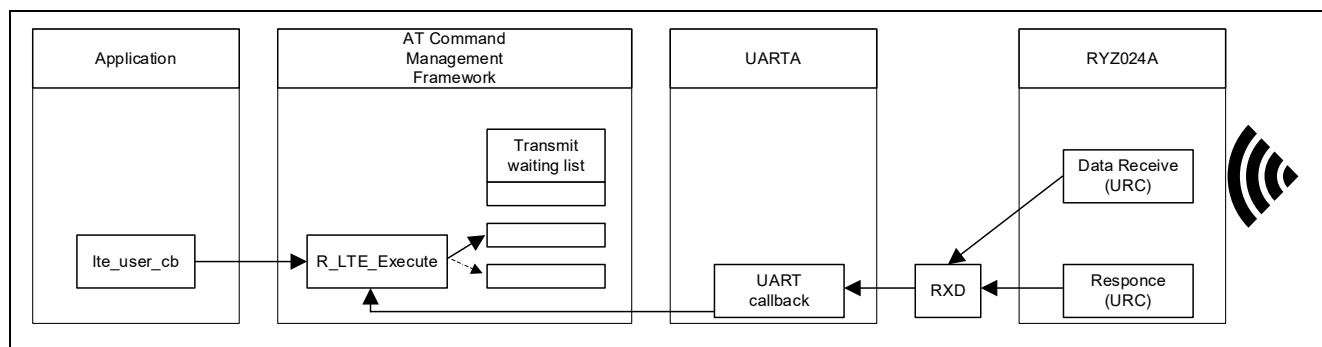


図 3-4 コールバック関数での通知

本サンプルプログラムのフレームワークベースプログラムから提供される API 関数は「3.2 API 関数」を参照してください。

AT コマンド API の実行結果はアプリケーションヘコールバック関数で通知されます。コールバック関数と通知されるデータについては「3.3 コールバック関数」を参照してください。

また AT コマンドマネジメントフレームワークを他の RL78 マイコンで使用する場合は「r_lte_user_config.h」を編集することで使用できます。設定可能な値は「3.4 ユーザ固有値の設定」を参照してください。

3.2 API 関数

本サンプルプログラムのフレームワークベースプログラムを使用して実装されている API 関数はマネジメント API と AT コマンド API の 2 種類に分類されます。マネジメント API はフレームワークベースプログラムの初期化や、一連の AT コマンドをレスポンスに応じて送信するための API です。AT コマンド API は AT コマンドを送信するための API です。マネジメント API は「3.2.1 マネジメント API」、AT コマンド API は「3.2.2 AT コマンド API」で説明します。

3.2.1 マネジメント API

マネジメント API はフレームワークベースプログラムの初期化や、一連の AT コマンドをレスポンスに応じて送信するための API です。マネジメント API はアプリケーションのメインループでコールする必要があります。本サンプルプログラムのフレームワークベースプログラムをベースに機能追加等を行う場合でも、基本的にマネジメント API のプログラム(r_lte_ryz.c, r_lte_ryz.h, r_lte_user_config.c)は変更不要です。

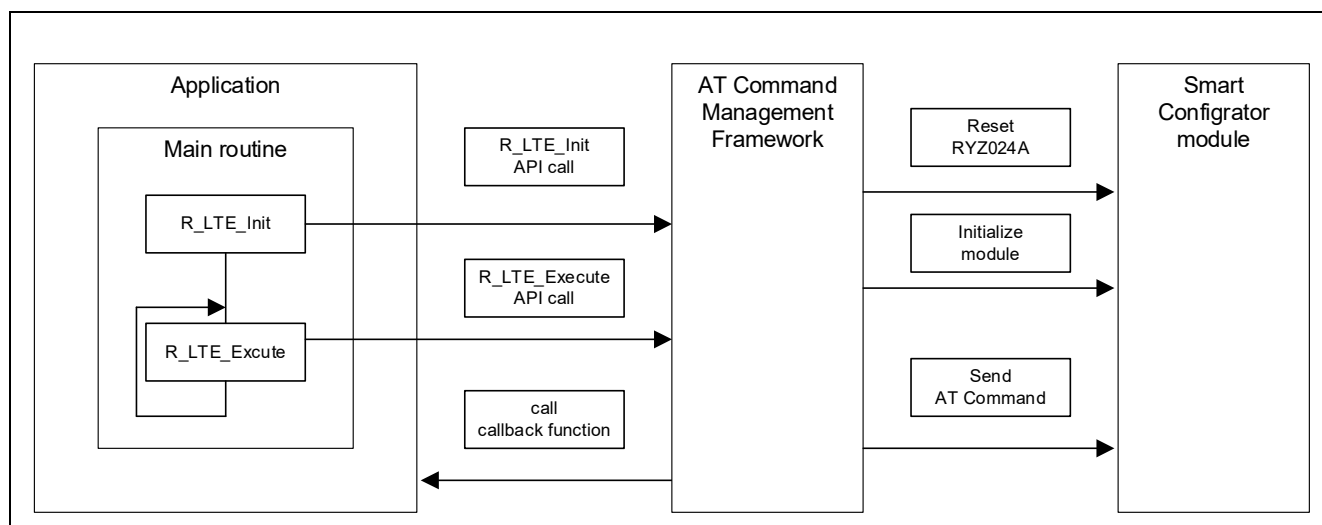


図 3-5 マネジメント API

3.2.1.1 R_LTE_Init

関数名	R_LTE_Init	
概要	フレームワークベースプログラムの初期化を行う	
引数	lte_cb_t * p_callback_fun (IN)	登録するコールバック関数。 コールバック関数の型については「3.3 コールバック関数」を参照。
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_POINTER_NULL (0x0001)	引数のポインタが NULL になっている
詳細機能	初期化として、以下を実行します。 使用するスマート・コンフィグレータ module の初期化 RYZ024A のハードウェアリセット API 関数の実行結果やイベントをアプリケーションに通知するコールバック関数の登録 本 API を実行後、RYZ024A をリセットするための AT コマンドが送信されます。 この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_INIT (0xFF) 本 API はアプリケーションのメインループ前に必ずコールしてください。	

3.2.1.2 R_LTE_Execute

関数名	R_LTE_Execute	
概要	フレームワークベースプログラムの実行を行う	
引数	void	なし
返り値	void	なし
詳細機能	フレームワークベースプログラムで実行する各種処理を実行します。 本関数では以下の動作を実行します。 RYZ024A から送信されるデータの受信と解析 AT コマンドの送信待機リストに登録されたデータの順次送信 受信したデータに合わせたコールバック関数のコール 本関数はアプリケーションのメインループ内で必ず繰り返しコールしてください。	

3.2.2 AT コマンド API

AT コマンド API は実行する動作に合わせて一連の AT コマンドを送信するための API です。アプリケーションから AT コマンド API をコールすることで送信待機リストに 1 つあるいは複数の AT コマンドが追加されます。送信待機リストに追加された AT コマンドは RYZ024A からのレスポンスに応じて順次 RYZ024A に送信されます。AT コマンド API で指定したすべての AT コマンドが送信されたら AT コマンドの送信結果をコールバック関数としてアプリケーションに通知します。

AT コマンド API の呼出し後、コールバック関数で結果が通知される前に次の AT コマンド API をコールすることはできません。また、AT コマンド API は割り込みハンドラからコールすることはできません。メインルーチン(AT コマンドマネジメントフレームワークのコールバック関数を含む)から実行してください。

本サンプルプログラムのフレームワークベースプログラムでは RYZ024A で MQTT 通信を行うために必要な API を実装しています。MQTT 通信アプリケーションで利用していない AT コマンドを用いた機能を実装したい場合、AT コマンドマネジメントフレームワークを使用して新しく AT コマンド API をユーザが追加し、アプリケーションを開発することを想定しています。

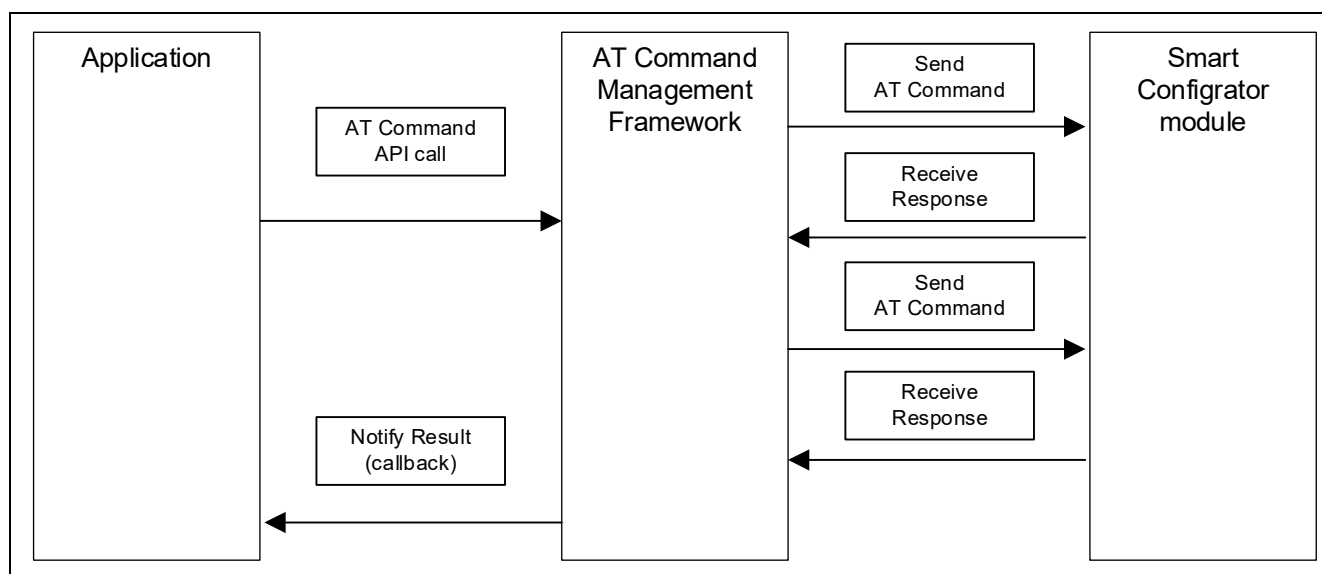


図 3-6 AT コマンド API

3.2.2.1 R_LTE_OM_Config

関数名	R_LTE_OM_Config	
機能概要	オペレータモードを設定する	
引数	uint8_t * p_pdp_type (IN)	PDP コンテキストの種類 例 : "IPv4V6"
	uint8_t * p_pdp_apn (IN)	PDP コンテキストのアクセスポイント名 例 : " ppsim.jp"
	uint8_t * p_pdp_protocol (IN)	PDP コンテキスト認証のプロトコル 例 : "0"
	uint8_t * p_pdp_userid (IN)	PDP コンテキスト認証のユーザ ID 例 : ""
	uint8_t * p_pdp_password (IN)	PDP コンテキスト認証のパスワード 例 : ""
	uint8_t * p_bandlist (IN)	LTE のバンド 例 : "1,2,3,4,5,8,12,13,17,18,19,20,25,26,28,66"
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_POINTER_NULL (0x0001)	引数のポインタが NULL になっている
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
	LTE_ERR_DATASIZE_OVERFLOW (0x0003)	引数のデータサイズが送信待機リストに登録できるサイズを超過した
詳細機能	以下の AT コマンドを順番に送信します。 "AT+CFUN=0" "AT+CGDCONT=1,[p_pdp_type],[p_pdp_apn]" "AT+CGAUTH=1,[p_pdp_protocol],[p_pdp_userid],[p_pdp_password]" または "AT+CGAUTH=1,0" (注) "AT+SQNCTM="standard" "AT+SQNBANDSEL=0,"standard",[p_bandlist]" "AT^RESET" "AT+CMEE=1" この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_OM_CONFIG (0x01) 注) p_pdp_protocol に"0"を指定すると"AT+CGAUTH=1,0"を送信します。	

3.2.2.2 R_LTE_NWK_Connect

関数名	R_LTE_NWK_Connect	
機能概要	ネットワークに接続する	
引数	uint8_t mode	送信する AT コマンドを選択する
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
詳細機能	Mode の値に応じて以下の AT コマンドを順番に送信します。(現在は 0 のみ使用可能) Mode = 0 “AT+CEREG=5” “AT+CFUN=1” この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_NWK_CONNECT (0x02)	

3.2.2.3 R_LTE_NWK_Disconnect

関数名	R_LTE_NWK_Disconnect	
機能概要	ネットワークから切断する	
引数	uint8_t mode	送信する AT コマンドを選択する
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
詳細機能	Mode の値に応じて以下の AT コマンドを送信します。(現在は 0 のみ使用可能) Mode = 0 “AT+CFUN=0” この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_NWK_DISCONNECT (0x03)	

3.2.2.4 R_LTE_MQTT_Connect

関数名	R_LTE_MQTT_Connect	
機能概要	MQTT 接続の設定を行い、サーバに接続する	
引数	uint8_t * p_username (IN)	MQTT 通信で使用するユーザ名 例: "renesas_device_001"
	uint8_t * p_host (IN)	接続する MQTT サーバのアドレス 例: "test.mosquitto.org"
	uint8_t * p_port (IN)	接続する MQTT サーバのポート 例: "1883"
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_POINTER_NULL (0x0001)	引数のポインタが NULL になっている
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
	LTE_ERR_DATASIZE_OVERFLOW (0x0003)	引数のデータサイズが送信待機リストに登録できるサイズを超過した
詳細機能	以下の AT コマンドを順番に送信します。 "AT+SQNSMQTTTCFG=0,[p_username]" "AT+SQNSMQTTCONNECT=0,[p_host],[p_port]" この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_MQTT_CONNECT (0x04)	

3.2.2.5 R_LTE_MQTT_Subscribe

関数名	R_LTE_MQTT_Subscribe	
機能概要	サブスクライブするトピックを指定する	
引数	uint8_t * p_topic (IN)	MQTT 通信でサブスクライブするトピック 例: "renesas_test"
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_POINTER_NULL (0x0001)	引数のポインタが NULL になっている
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
	LTE_ERR_DATASIZE_OVERFLOW (0x0003)	引数のデータサイズが送信待機リストに登録できるサイズを超過した
詳細機能	以下の AT コマンドを送信します。 AT+SQNSMQTTSUBSCRIBE=0,[p_topic],1" この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_MQTT_SUBSCRIBE (0x05)	

3.2.2.6 R_LTE_MQTT_Publish

関数名	R_LTE_MQTT_Publish	
機能概要	MQTT のトピックにメッセージをパブリッシュする	
引数	uint8_t * p_topic (IN)	メッセージをパブリッシュするトピック 例 : "renesas_test"
	uint16_t length (IN)	パブリッシュするメッセージのサイズ
	uint8_t * p_message (IN)	パブリッシュするメッセージ
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_POINTER_NULL (0x0001)	引数のポインタが NULL になっている
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
	LTE_ERR_DATASIZE_OVERFLOW (0x0003)	引数のデータサイズが送信待機リストに登録できるサイズを超過した
詳細機能	以下の AT コマンドを送信します。 "AT+SQNSMQTTPUBLISH=0,[p_topic],1,[length]" この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_MQTT_PUBLISH (0x06)	

3.2.2.7 R_LTE_MQTT_RcvMessage

関数名	R_LTE_MQTT_RcvMessage	
機能概要	MQTT からメッセージを受信する	
引数	uint8_t * p_topic (IN)	受信するメッセージのトピック 例 : "renesas_test"
	uint8_t message_id (IN)	受信するメッセージの ID
	uint16_t message_size (IN)	受信するメッセージのサイズ
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_POINTER_NULL (0x0001)	引数のポインタが NULL になっている
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
	LTE_ERR_DATASIZE_OVERFLOW (0x0003)	引数のデータサイズが送信待機リストに登録できるサイズを超過した
詳細機能	message_id の値に応じて以下の AT コマンドを送信します。 Message_id = 0 "AT+SQNSMQTTRCVMESSAGE=0,[p_topic]" Message_id = 0 以外 "AT+SQNSMQTTRCVMESSAGE=0,[p_topic],[message_id]" この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 また受信したメッセージもコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_MQTT_RCVMESSAGE (0x07) 本 API は通常、"+SQNSMQTTTONMESSAGE" の URC を受信した後に実行します。 引数に指定した message_id に該当する MQTT メッセージをまだ受信していない場合、コールバック関数でエラー "LTE_CME_ERR_OPERATION_NOT_SUPPORTED" が通知されます。	

3.2.2.8 R_LTE_MQTT_Disconnect

関数名	R_LTE_MQTT_Disconnect	
機能概要	MQTT サーバから切断する	
引数	void	なし
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
詳細機能	以下の AT コマンドを送信します。 "AT+SQNSMQTTDISCONNECT=0" この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_MQTT_DISCONNECT (0x08)	

3.2.2.9 R_LTE_SEC_CertificateAdd

関数名	R_LTE_SEC_CertificateAdd	
機能概要	証明書を追加する	
引数	uint8_t cet_id	追加する証明書の ID
	uint16_t cet_len	追加する証明書のデータ長
	uint8_t * p_cet_data	追加する証明書の文字列データ
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_POINTER_NULL (0x0001)	引数のポインタが NULL になっている
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
	LTE_ERR_DATASIZE_OVERFLOW (0x0003)	引数のデータサイズが送信待機リストに登録できるサイズを超過した
詳細機能	以下の AT コマンドを送信します。 "AT+SQNSNVW="certificate",[cet_id],[cet_len]" この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_SEC_CERTIFICATEADD (0x09)	

3.2.2.10 R_LTE_SEC_CertificateRemove

関数名	R_LTE_SEC_CertificateRemove	
機能概要	証明書を削除する	
引数	uint8_t cet_id	削除する証明書の ID
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
詳細機能	以下の AT コマンドを送信します。 "AT+SQNSNVW="certificate",[cet_id],0" この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_SEC_CERTIFICATEREMOVE (0x0A)	

3.2.2.11 R_LTE_SEC_PrivateKeyAdd

関数名	R_LTE_SEC_PrivateKeyAdd	
機能概要	プライベートキーを追加する	
引数	uint8_t prk_id	追加するプライベートキーの ID
	uint16_t prk_len	追加するプライベートキーのデータ長
	uint8_t * p_prk_data	追加するプライベートキーの文字列データ
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_POINTER_NULL (0x0001)	引数のポインタが NULL になっている
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
	LTE_ERR_DATASIZE_OVERFLOW (0x0003)	引数のデータサイズが送信待機リストに登録できるサイズを超過した
詳細機能	以下の AT コマンドを送信します。 "AT+SQNSNVW="privatekey",[prk_id],[prk_len]" この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_SEC_PRIVATEKEYADD (0x0B)	

3.2.2.12 R_LTE_SEC_PrivateKeyRemove

関数名	R_LTE_SEC_PrivateKeyRemove	
機能概要	プライベートキーを削除する	
引数	uint8_t prk_id	削除するプライベートキーの ID
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
詳細機能	以下の AT コマンドを送信します。 "AT+SQNSNVW="privatekey",[prk_id],0" この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_SEC_PRIVATEKEYREMOVE (0x0C)	

3.2.2.13 R_LTE_NWK_ConnectionConfig

関数名	R_LTE_NWK_ConnectionConfig	
機能概要	接続とセキュリティを設定する	
引数	uint8_t ca_cer_id	CA 証明書の ID
	uint8_t client_cer_id	クライアント証明書の ID
	uint8_t prk_id	プライベートキーの ID
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
詳細機能	以下の AT コマンドを順番に送信します。 "AT+SQNSCFG=1,1,1" "AT+SQNSPCFG=1,2,,5,[ca_cer_id],[client_cer_id],[prk_id],""" この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_NWK_CONNECTIONCONFIG (0x0D)	

3.2.2.14 R_LTE_eDRX_Config

関数名	R_LTE_eDRX_Config	
機能概要	eDRX の動作とパラメータを設定する	
引数	uint8_t mode (IN)	eDRX 動作モード
	uint8_t edrx_time_value (IN)	eDRX 周期の設定値
	uint8_t ptw_time_value (IN)	PTW 時間の設定値
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
	LTE_ERR_DATASIZE_OVERFLOW (0x0003)	引数のデータサイズが送信待機リストに登録できるサイズを超過した
詳細機能	以下のように、指定された引数から AT コマンド文字列を生成し送信します。 AT+SQNEDRX=2,4,"0001","0000" この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_EDRX_CONFIG (0x0E)	

(1) mode パラメータ

```
typedef enum
{
    LTE_EDRX_MODE_DISABLE = 0,
    LTE_EDRX_MODE_ENABLE,
    LTE_EDRX_MODE_ENABLE_WITH_URC,
    LTE_EDRX_MODE_RESET_PARAM,
} e_lte_edrx_mode_t;
```

(2) edrx_time_value パラメータ

```
typedef enum
{
    LTE_EDRX_TIME_VAL_5_SEC = 0,
    LTE_EDRX_TIME_VAL_10_SEC,
    LTE_EDRX_TIME_VAL_20_SEC,
    LTE_EDRX_TIME_VAL_40_SEC,
    LTE_EDRX_TIME_VAL_61_SEC,
    LTE_EDRX_TIME_VAL_81_SEC,
    LTE_EDRX_TIME_VAL_102_SEC,
    LTE_EDRX_TIME_VAL_122_SEC,
    LTE_EDRX_TIME_VAL_143_SEC,
    LTE_EDRX_TIME_VAL_163_SEC,
    LTE_EDRX_TIME_VAL_327_SEC,
    LTE_EDRX_TIME_VAL_655_SEC,
    LTE_EDRX_TIME_VAL_1301_SEC,
    LTE_EDRX_TIME_VAL_2621_SEC,
} e_lte_edrx_time_value_t;
```

(3) ptw_time_value パラメータ

```
typedef enum
{
    LTE_EDRX_PTW_TIME_VAL_1_SEC = 0,
    LTE_EDRX_PTW_TIME_VAL_2_SEC,
    LTE_EDRX_PTW_TIME_VAL_3_SEC,
    LTE_EDRX_PTW_TIME_VAL_5_SEC,
    LTE_EDRX_PTW_TIME_VAL_6_SEC,
    LTE_EDRX_PTW_TIME_VAL_7_SEC,
    LTE_EDRX_PTW_TIME_VAL_8_SEC,
    LTE_EDRX_PTW_TIME_VAL_10_SEC,
    LTE_EDRX_PTW_TIME_VAL_11_SEC,
    LTE_EDRX_PTW_TIME_VAL_12_SEC,
    LTE_EDRX_PTW_TIME_VAL_14_SEC,
    LTE_EDRX_PTW_TIME_VAL_15_SEC,
    LTE_EDRX_PTW_TIME_VAL_16_SEC,
    LTE_EDRX_PTW_TIME_VAL_17_SEC,
    LTE_EDRX_PTW_TIME_VAL_19_SEC,
    LTE_EDRX_PTW_TIME_VAL_20_SEC,
} e_lte_edrx_ptw_time_value_t;
```

3.2.2.15 R_LTE_PSM_Config

関数名	R_LTE_PSM_Config	
機能概要	PSM の動作とパラメータを設定する	
引数	uint8_t mode (IN)	PSM 動作モード
	uint8_t tau_time_value (IN)	TAU 時間の設定値
	uint8_t tau_multiplier (IN)	TAU 時間に対する乗数
	uint8_t active_time_value (IN)	Active 時間の設定値
	uint8_t active_multiplier (IN)	Active 時間に対する乗数
返り値	LTE_SUCCESS (0x0000)	成功
	LTE_ERR_IN_PROCESS (0x0002)	他の LTE API が実行中である
	LTE_ERR_DATASIZE_OVERFLOW (0x0003)	引数のデータサイズが送信待機リストに登録できるサイズを超過した
詳細機能	以下のように指定された引数から AT コマンド文字列を生成し送信します。 AT+CPSMS=1,,,"10000010","00001111" この API で送信された AT コマンドの実行結果はコールバック関数で通知されます。 以下の API_ID で通知されます。 LTE_API_PSM_CONFIG (0x0F)	

(1) mode パラメータ

```
typedef enum
{
    LTE_PSM_MODE_DISABLE = 0,
    LTE_PSM_MODE_ENABLE,
    LTE_PSM_MODE_RESET_PARAM,
} e_lte_psm_mode_t;
```

(2) tau_time_value パラメータ

```
typedef enum
{
    LTE_PSM_TAU_TIME_VAL_10_MIN = 0,
    LTE_PSM_TAU_TIME_VAL_1_HOUR,
    LTE_PSM_TAU_TIME_VAL_10_HOUR,
    LTE_PSM_TAU_TIME_VAL_2_SEC,
    LTE_PSM_TAU_TIME_VAL_30_SEC,
    LTE_PSM_TAU_TIME_VAL_1_MIN,
    LTE_PSM_TAU_TIME_VAL_320_HOUR,
} e_lte_psm_tau_time_value_t;
```

(3) active_time_value パラメータ

```
typedef enum
{
    LTE_PSM_ACTIVE_TIME_VAL_2_SEC = 0,
    LTE_PSM_ACTIVE_TIME_VAL_1_MIN,
    LTE_PSM_ACTIVE_TIME_VAL_6_MIN,
    LTE_PSM_ACTIVE_TIME_VAL_NONE = 7,
} e_lte_psm_active_time_value_t;
```

(4) tau_multiplier、active_multiplier パラメータ

```
typedef enum
{
    LTE_PSM_MULTIPLIER_0 = 0,
    LTE_PSM_MULTIPLIER_1,
    LTE_PSM_MULTIPLIER_2,
    LTE_PSM_MULTIPLIER_3,
    LTE_PSM_MULTIPLIER_4,
    LTE_PSM_MULTIPLIER_5,
    LTE_PSM_MULTIPLIER_6,
    LTE_PSM_MULTIPLIER_7,
    LTE_PSM_MULTIPLIER_8,
    LTE_PSM_MULTIPLIER_9,
    LTE_PSM_MULTIPLIER_10,
    LTE_PSM_MULTIPLIER_11,
    LTE_PSM_MULTIPLIER_12,
    LTE_PSM_MULTIPLIER_13,
    LTE_PSM_MULTIPLIER_14,
    LTE_PSM_MULTIPLIER_15,
    LTE_PSM_MULTIPLIER_16,
    LTE_PSM_MULTIPLIER_17,
    LTE_PSM_MULTIPLIER_18,
    LTE_PSM_MULTIPLIER_19,
    LTE_PSM_MULTIPLIER_20,
    LTE_PSM_MULTIPLIER_21,
    LTE_PSM_MULTIPLIER_22,
    LTE_PSM_MULTIPLIER_23,
    LTE_PSM_MULTIPLIER_24,
    LTE_PSM_MULTIPLIER_25,
    LTE_PSM_MULTIPLIER_26,
    LTE_PSM_MULTIPLIER_27,
    LTE_PSM_MULTIPLIER_28,
    LTE_PSM_MULTIPLIER_29,
    LTE_PSM_MULTIPLIER_30,
    LTE_PSM_MULTIPLIER_31,
} e_lte_psm_tau_multiplier_t;
```

3.3 コールバック関数

RYZ024A へ AT コマンドを送るとレスポンスが返ってきます。また、RYZ024A の状態が変化した場合は RYZ024A から Unsolicited Response Code (URC) が送信されます。本サンプルプログラムのフレームワークベースプログラムでは RYZ024A からそれらのデータを受信した後、R_LTE_Execute 関数内で解析します。アプリケーションに通知する必要がある場合、R_LTE_Execute 関数内でコールバック関数をコールしてアプリケーションに通知します。これにより、アプリケーションは AT コマンド API の実行結果を確認したり、RYZ024A の URC を確認したりすることができます。ここではコールバック関数の構造やコールバック関数によって通知されるイベントやデータについて説明します。

コールバック関数は以下の構造になっています。

型名	void * lte_cb_t	
引数	uint16_t event_type (In)	通知されるイベントの ID。 設定される値は表 3-1 を参照してください。
	uint16_t api_id (In)	フレームワークベースプログラムがどの API を実行しているかを示す ID。 設定される値は表 3-2 を参照してください。
	uint16_t data_len (in)	p_data で通知されるデータのサイズ。
	void * p_data (out)	アプリケーションに通知するデータのポインタ。 データの内容はイベント種類に応じて変化します。 格納されるデータはコールバック関数がコールされるたびに上書きされるので必要に応じてアプリケーションで保持して下さい。

event_type と api_id の値はフレームワークベースプログラム内のマクロ形式で定義された値を使用します。以下にそれぞれの値を示します。

表 3-1 event_type の値

定義名	値	説明
LTE_EVENT_API_COMPLETE	0x0000	API 関数で指定した動作が正常に完了したことを通知するイベント。 p_data にはコールした API に合わせたデータが設定される。
LTE_EVENT_ERROR	0x0001	API 関数で指定した動作でエラーが発生したことを通知するイベント。 p_data にはエラーが数値データとして設定される。
LTE_EVENT_RCVURC	0x0002	URC を受信したことを通知するイベント。 p_data には URC が文字列データとして設定される。
LTE_EVENT_TIMEOUT_ERROR	0x0003	AT コマンドの送信後、レスポンスの受け取りまでにタイムアウトが発生したことを通知するイベント。 AT コマンドの送信後、60s 経過するとタイムアウトが発生する。
LTE_EVENT_FATAL_ERROR	0x0004	致命的なエラーが発生した場合に通知されるイベント。 意図しないタイミングで URC"+SYSSTART"を受信したときにコールバック関数をコールする。

表 3-2 api_id の値

定義名	値	対応する API
LTE_API_NO_CURRENT_API	0x0000	なし
LTE_API_OM_CONFIG	0x0001	R_LTE_OM_Config
LTE_API_NWK_CONNECT	0x0002	R_LTE_NWK_Connect
LTE_API_NWK_DISCONNECT	0x0003	R_LTE_NWK_Disconnect
LTE_API_MQTT_CONNECT	0x0004	R_LTE_MQTT_Connect
LTE_API_MQTT_DISCONNECT	0x0005	R_LTE_MQTT_Disconnect
LTE_API_MQTT_SUBSCRIBE	0x0006	R_LTE_MQTT_Subscribe
LTE_API_MQTT_PUBLISH	0x0007	R_LTE_MQTT_Publish
LTE_API_MQTT_RCVMESSAGE	0x0008	R_LTE_MQTT_RcvMessage
LTE_API_SEC_CERTIFICATEADD	0x0009	R_LTE_SEC_CertificateAdd
LTE_API_SEC_CERTIFICATEREMOVE	0x000A	R_LTE_SEC_CertificateRemove
LTE_API_SEC_PRIVATEKEYADD	0x000B	R_LTE_SEC_PrivateKeyAdd
LTE_API_SEC_PRIVATEKEYREMOVE	0x000C	R_LTE_SEC_PrivateKeyRemove
LTE_API_NWK_CONNECTIONCONFIG	0x000D	R_LTE_NWK_ConnectionConfig
LTE_API_EDRX_CONFIG	0x000E	R_LTE_eDRX_Config
LTE_API_PSM_CONFIG	0x000F	R_LTE_PSM_Config
LTE_API_INIT	0x00FF	R_LTE_Init

コールバック関数は特定の状況で R_LTE_Execute 関数からコールされます。以下にコールバック関数がコールされる状況とその時に設定されるデータを示します。

コールバック関数が記載されているソースコードを以下に示します

プログラム : lte_entry.c

- AT コマンド API で送信する AT コマンドとそれに対するレスポンスがすべて送受信でき、レスポンスにエラーがない場合 :
 - “event_type”に”LTE_EVENT_API_COMPLETE”が設定されます。
 - “p_data”には実行する AT コマンドに合わせてデータが設定されます。
 - ✧ AT コマンドの実行結果として URC を受信する場合、受信した URC の文字列データが登録されます。通知される文字列データのサイズは”data_len”に設定されています。
 - ✧ R_LTE_MQTT_RcvMessage など別途データを受信する AT コマンド API をコールしている場合、受信した文字列データが登録されます。受信したデータサイズが”LTE_DATA_STR_SIZE”を超過する場合、超過した分のデータは破棄され、前半部分のデータが登録されます。通知される文字列データのサイズは”data_len”に設定されています。
 - ✧ 上記以外の場合、データは設定されません。”data_len”は 0 に設定されています。

```
void lte_user_cb(uint16_t event_type, uint16_t api_id, uint16_t data_len, uint8_t *
p_data)
{
    if(LTE_EVENT_API_COMPLETE == event_type)
    {
        switch (api_id)
        {
            case LTE_API_OM_CONFIG:
                /* Connect to network after configuration of operation mode complete
                */
                sprintf(sbuf, "OM CONFIG COMPLETE");
                debug_printf(sbuf);
                R_LTE_NWK_Connect(1);
                break;

            /* 省略 */

            case LTE_API_MQTT_RCVMESSAGE:
                /* Display received message */
                sprintf(sbuf, "MQTT RCVMESSAGE COMPLETE");
                for(uint8_t i = 0; i < data_len; i++)
                {
                    sbuf[21+i] = p_data[i];
                }
                sbuf[21+data_len] = '\0';
                sbuf[21+data_len+1] = '\0';
                break;
        }
    }
}
```

LTE_EVENT_API_COMPLETE のイベント通知

api_id でどの AT コマンド API の結果か判断する

データを受信する場合 p_data に登録されている

図 3-7 LTE_EVENT_API_COMPLETE のイベント通知

- RYZ024A に送信した AT コマンドに対するレスポンスがエラーだった場合 :
 - “event_type”に”LTE_EVENT_ERROR”が設定されます。
 - “p_data”にはエラーを示す値が登録されています。この値を確認するため、LTE_ERROR_DECODE 関数を使用して 16bit の値として確認してください。

```
void lte_user_cb(uint16_t event_type, uint16_t api_id, uint16_t data_len, uint8_t *
p_data)
{
    /* 省略 */

    if(LTE_EVENT_ERROR == event_type)
    {
        /* Display API ID and Error code when error occurs */
        uint16_t err_code;
        LTE_ERROR_DECODE(&err_code, p_data);
        sprintf(sbuf,"ERROR RESPONSE\n");
        debug_printf(sbuf);
        sprintf(sbuf,"API ID: %d, ERROR CODE: %d\n", api_id, err_code);
        debug_printf(sbuf);
    }
}
```

LTE_EVENT_ERROR のイベント通知

LTE_ERROR_DECODE で
エラーコードを解析

図 3-8 LTE_EVENT_ERROR のイベント通知

- RYZ024A に送信した AT コマンドがタイムアウトした場合 :
 - “event_type”に”LTE_EVENT_TIMEOUT_ERROR”が設定されます。
 - “p_data”にはデータは設定されていません。
 - タイムアウトが発生した場合、多くの場合 RYZ024A の動作でエラーが発生したことが想定されます。そのため、初期化を実行することを推奨しています。

```
void lte_user_cb(uint16_t event_type, uint16_t api_id, uint16_t data_len, uint8_t *
p_data)
{
    /* 省略 */
    if(LTE_EVENT_TIMEOUT_ERROR == event_type)
    {
        /* Set flag to initialize in main program */
        gs_reinitate_flag = 1;
    }
    /* 省略 */

    void lte_entry(void)
    {
        /* 省略 */
        if(1 == gs_reinitate_flag)
        {
            /* Initialize when timeout occurs */
            R_LTE_Init(lte_user_cb);
            gs_reinitate_flag = 0;
        }
    }
}
```

LTE_EVENT_TIMEOUT_ERROR のイベント通知

フレームワークベースプログラムの初期化

図 3-9 LTE_EVENT_TIMEOUT_ERROR のイベント通知

- RYZ024A から意図しないタイミングで URC"+SYSSTART"を受信した場合：
 - "event_type"に"LTE_EVENT_FATAL_ERROR"が設定されます。
 - "p_data"にはデータは設定されていません。
 - 本イベントが発生した場合、RYZ024A が再起動したことが想定されます。そのため、初期化からユーザのアプリケーションを再度開始することを推奨します。

【注】 RYZ024A の通常の動作において意図しないタイミングで URC"+SYSSTART"を受信することはありません。本ケースは万一の発生に備えてフェールセーフの目的で実装しています。

```
void lte_user_cb(uint16_t event_type, uint16_t api_id, uint16_t data_len, uint8_t *
p_data)
{
/* 省略 */
    if(LTE_EVENT_FATAL_ERROR == event_type)
    {
        /* Set flag to initialize in main loop */
        gs_reinitialize_flag = 1;
    }
}
/* 省略 */

void lte_entry(void)
{
/* 省略 */
    if(1 == gs_reinitialize_flag )
    {
        /* Initialize to restart user application */
        R_LTE_Init(lte_user_cb);
        gs_reinitialize_flag = 0;
    }
}
```

LTE_EVENT_FATAL_ERROR のイベント通知

フレームワークベースプログラムの初期化

図 3-10 LTE_EVENT_FATAL_ERROR のイベント通知

- RYZ024A から URC を受信した場合 :
 - “event_type”に”LTE_EVENT_RCVURC”が設定されます。
 - “p_data”には受信した URC の文字列データが設定されます。URC に合わせて処理を実行してください。受信した URC のデータサイズが”LTE_DATA_STR_SIZE”を超過する場合、超過した分のデータは破棄され、前半部分のデータが登録されます。通知される文字列データのサイズは”data_len”に設定されています。

```

void lte_user_cb(uint16_t event_type, uint16_t api_id, uint16_t data_len, uint8_t *
p_data)
{
    /* 省略 */

    if(LTE_EVENT_RCVURC == event_type)
    {
        /* Receive message from MQTT server when RYZ024A received subscribe
        notification */
        const uint8_t str_onmessage[] = "+SQNSMQTTONMESSAGE";

        /* Display received URC */
        sprintf(sbuf,"URC: %s",p_data);
        debug_printf(sbuf);

        if(0 == memcmp(p_data, str_onmessage, (sizeof(str_onmessage) - 1)))
        {
            uint8_t rcv_id = 0;
            char * ptr;
            uint8_t msg_count = 0;

            /* Display subscribed notification */
            sprintf(sbuf,"MQTT MESSAGE NOTIFY\n");
            debug_printf(sbuf);

            /* Get message ID from received URC string data */
            ptr = strtok((char *)p_data, ",");
            while(ptr != NULL)
            {
                ptr = strtok(NULL, ",");
                if(ptr != NULL)
                {
                    msg_count++;
                    if(2 == msg_count)
                    {
                        mqtt_rcvdata_len = (uint8_t )atoi(ptr);
                    }
                    if(4 == msg_count)
                    {
                        rcv_id = (uint8_t )atoi(ptr);
                    }
                }
            }
            /* request message receive */
            R_LTE_MQTT_RcvMessage(str_MQTT_topic, rcv_id);
        }
    }
}

```

LTE_EVENT_RCVURC のイベント通知

**受信 URC の確認
“+SQNSMQTTONMESSAGE”の場合、処理を実行**

URC のパラメータを解析

**解析したパラメータを使用して
AT コマンド API をコール**

図 3-11 LTE_EVENT_RCVURC のイベント通知

3.4 ユーザ固有値の設定

本サンプルアプリケーションをベースにしてアプリケーションを開発する場合、ユーザが使用する RL78 マイコンに応じて一部の設定値を変更します。AT コマンドマネジメントフレームワークではこれらのユーザ固有の設定値を設定するプログラムを「r_lte_ryz.c」に定義しています。ユーザはこのファイルを変更することで環境にあった設定で AT コマンドマネジメントフレームワークを使用することができます。ここでは設定できる値について説明します。

表 3-3 に RYZ024A の各端子に接続する RL78/G22 の端子操作関数を示します。ホスト MCU として使用するボードを変更するなどの場合にご確認ください。

表 3-3 RYZ024A の端子操作関数

関数名	説明	使用端子
<code>void ryz_lte_rts_low_devspe(void)</code>	RYZ024A の RTS 端子を Low にします。	P70
<code>void ryz_lte_rts_high_devspe(void)</code>	RYZ024A の RTS 端子を High にします。	P70
<code>void ryz_lte_reset_low_devspe(void)</code>	RYZ024A の RESET 端子を Low にします。	P17
<code>void ryz_lte_reset_high_devspe(void)</code>	RYZ024A の RESET 端子を High にします。	P17
<code>uint8_t ryz_lte_cts_read_devspe(void)</code>	RYZ024A の CTS 端子をリードします。	P50
<code>uint8_t ryz_lte_ring_read_devspe(void)</code>	RYZ024A の RING 端子をリードします。	P51

表 3-4 に AT コマンドマネジメントフレームワーク内でスマート・コンフィグレータを使用するための設定項目を示します。スマート・コンフィグレータ編集する、使用する RL78 マイコンを変更するなどの場合にご確認ください

表 3-4 RL78/G22 のスマート・コンフィグレータ設定

タグ名	コンポーネント	内容
クロック	-	動作モード：高速メインモード 1.8 (V) ~5.5 (V) 高速オンチップ・オシレータ：32MHz foco 開始設定：通常 f _{IHP} ：32MHz f _{MAIN} ：32MHz f _{CLK} ：32000kHz
システム	-	オンチップ・デバッグ動作設定：エミュレータを使う エミュレータ設定：E2 エミュレータ Lite 疑似 RRM/DMM 機能設定：使用する Start/Stop 関数機能設定：使用しない セキュリティ ID 設定：セキュリティ ID を設定する セキュリティ ID：0x00000000000000000000 セキュリティ ID 認証失敗時の設定：フラッシュ・メモリのデータを消去する

タグ名	コンポーネント	内容
コンポーネント	r_bsp	Start up select : Enable (use BSP startup) Control of invalid memory access detection : Disable RAM guard space (GRAM0-1) : Disabled Guard of control registers of port function (GPORT) : Disabled Guard of registers of interrupt function (GINT) : Disabled Guard of control registers of clock control function, voltage detector, and RAM parity error detection function (GCSC) : Disabled Data flash access control (DFLEN) : Disables Initialization of peripheral functions by Code Generator/Smart Configurator : Enable API functions disable : Enable Parameter check enable : Enable Setting for starting the high-speed on-chip oscillator at the times of release from STOP mode and of transitions to SNOOZE mode : High-speed Enable user warm start callback (PRE) : Unused Enable user warm start callback (POST) : Unused Watchdog Timer refresh enable : Unused
	Config_TAU0_1	コンポーネント : インターバル・タイマ 動作モード : 16 ビット・カウンタ・モード リソース : TAU0_1 動作クロック : CK00 クロックソース : $f_{CLK}/2$ インターバル時間 : 1ms 割り込み設定 : 使用する 優先順位 : レベル 3
	Config_UART0	コンポーネント : UART 通信 動作モード : 送受信 リソース : UART0 動作クロック : CK00 クロックソース : $f_{CLK}/2$ 転送モード : シングル転送 データ・ビット長設定 : 8bit データ転送方向設定 : LSB パリティ設定 : なし ストップビット長設定 : 1bit 送信データ・レベル設定 : 非反転 転送レート : 115200bps 優先順位 : レベル 3 コールバック機能設定 : 送信完了

タグ名	コンポーネント	内容
コンポーネント	Config_UARTA0	コンポーネント：UART 通信 動作モード：送受信 リソース：UARTA0 動作クロック：f _{SEL} クロックソース：f _{SEL} クロック選択 f _{IHP} データ・ビット長設定：8bit データ転送方向設定：LSB パリティ設定：なし ストップビット長設定：1bit 送信データ・レベル設定：非反転 送信モード設定：ポーリングによる連続送信 受信エラーの設定：INTUR 割り込み発生 転送レート：115200bps 受信完了割り込み設定(INTUR0)：レベル 3 コールバック機能設定：受信完了、受信エラー
	Config_INTC	INTP0：立下りエッジ、レベル 3（低優先順位） INTP2：立下りエッジ、レベル 3（低優先順位）
	Config_PORT	PORT1、PORT5、PORT7 にチェック ポートモード設定：Pmn レジスタ値を読み出す P17：出力 P50：入力 P70：出力

表 3-5 に AT コマンドマネジメントフレームワーク内で使用する各種データのサイズ設定項目を示します。AT コマンドマネジメントフレームワークではこれらの設定値を「r_lte_user_config.h」に定義しています。アプリケーションで使用する AT コマンドや文字列のデータサイズ、使用する MCU のスタックサイズに合わせて変更してください。

表 3-5 AT コマンド送信待機リストのサイズ設定

定義名	デフォルト値	説明
LTE_ATC_STR_SIZE	100	AT コマンドの文字列の最大長。
LTE_DATA_STR_SIZE	100	RYZ024A から受信するデータの最大長。 受信するデータがこのサイズを超過する場合、超過する分のデータは破棄される。
LTE_ATC_LIST_SIZE	8	送信待機リストに登録することのできる AT コマンドの数。 "登録する最大 AT コマンド数 + 1"を定義してください。

3.5 フレームワーク内で使用されるスマート・コンフィグレータモジュール

AT コマンドマネジメントフレームワークではスマート・コンフィグレータモジュールを使用して機能を実装しています。スマート・コンフィグレータモジュールはコードだけでなくスマート・コンフィグレータで設定しています。ここでは AT コマンドマネジメントフレームワークで使用するスマート・コンフィグレータモジュールについて利用方法や設定方法について説明します。

3.5.1 UARTA モジュール

AT コマンドマネジメントフレームワークでは UARTA モジュールを使用して RL78/G22 と RYZ024A の間の UART 通信を行います。

RL78/G22 から RYZ024A へ AT コマンドを送信するとき、UARTA モジュールの書き込み関数 (R_Config_UARTA0_Send) を使用しています。アプリケーションから AT コマンド API をコールした後、フレームワーク内の送信待機リストに一連の AT コマンドが登録されます。送信待機リストの先頭から書き込み関数を使用して順番に送信されます。

RYZ024A から RL78/G22 へレスポンスを送信するとき、UARTA モジュールのコールバック関数を使用してデータを受信します。このコールバック関数では 1 文字ずつ文字データを受信し、フレームワーク内のリングバッファに格納されます。リングバッファに格納された文字データは R_LTE_Execute 関数のコールするたびに 1 文字ずつ処理されます。

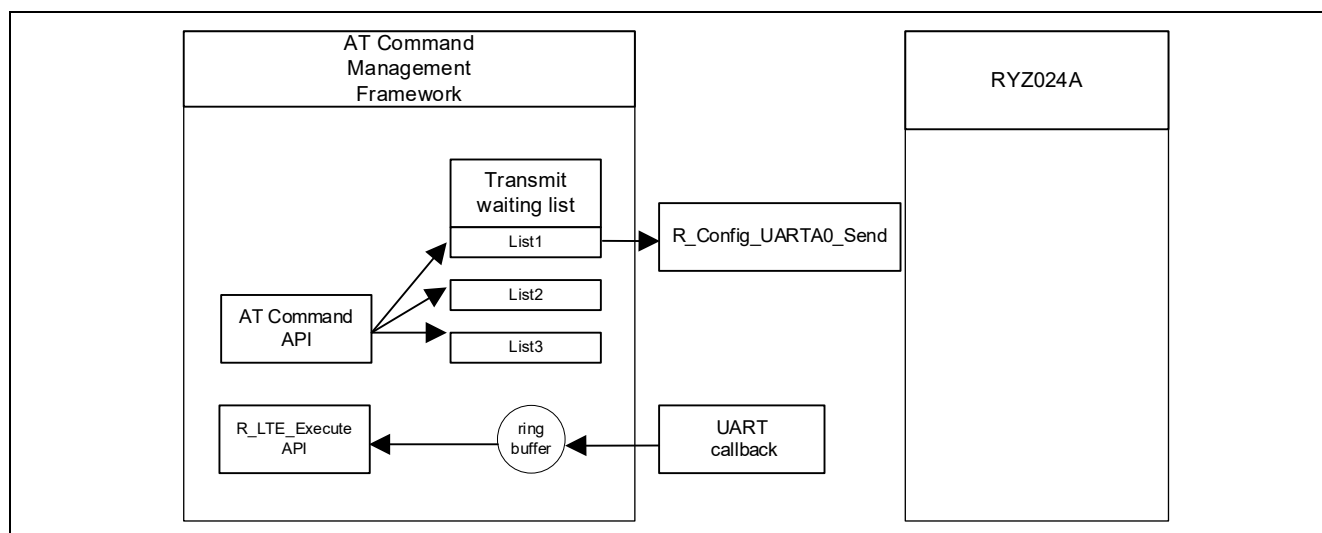


図 3-12 UARTA モジュールの利用

3.5.2 TAU モジュール

AT コマンドマネジメントフレームワークではタイムアウト機能に TAU モジュールを使用します。AT コマンドを送信した後 60 秒間応答がない場合タイムアウトが発生します。タイムアウト時間を設定する定義を「表 3-6 AT コマンドタイムアウト時間設定(r_lte_ryz.c)」に示します。

表 3-6 AT コマンドタイムアウト時間設定(r_lte_ryz.c)

定義名	値	説明
AT_COMMAND_TIMETOUT	30	AT コマンドのタイムアウトカウント。 単位: 2sec 例) 2sec * 30 = 60sec

AT コマンドを送信するタイミングでタイマを開始します。このタイマは comp_msg に指定したレスポンスを受信したとき、もしくはエラーレスポンスを受信したときに停止します。AT コマンド送信後、一定時間レスポンスを受け取れない場合、タイマのコールバック関数でタイムアウトの判定を行います。タイムアウトが判定されると、R_LTE_Execute 関数でタイムアウトが発生したことをアプリケーションに通知するためにユーザのコールバック関数をコールします。

スマート・コンフィグレータでタイマのカウント時間を設定しています。タイムアウト時間を変更する場合はスマート・コンフィグレータでタイマカウントの時間と、「表 3-6 AT コマンドタイムアウト時間設定(r_lte_ryz.c)」を変更してください。



図 3-13 TAU モジュールの設定

3.5.3 割り込み機能

割り込み機能を使用して、RYZ024A から URC があることを通知する RING 信号の割り込みを発生させます。この RING 割り込みを検出するために、INTP2 を使用しています。

また、SW が押下されたことを検出するために、INTP0 を使用しています。

3.5.4 UART0 モジュール

サンプルプログラムの実行結果を出力するために、UART0 モジュールを使用します。サンプルプログラムの実行結果は、RL78/G22 FPB の USB (COM ポート) 経由で出力されます。

3.6 低消費電力動作

本サンプルアプリケーションでは RL78/G22 と RYZ024A の低消費電力機能を利用した動作に対応しています。

3.6.1 RL78/G22 の低消費電力動作制御

RL78/G22 は、RL78/G22 が IDLE 状態の時に SNOOZE モード、AT コマンド API の中で AT コマンドを送信して応答を待っている状態の時に HALT モードの低消費電力動作状態になります。低消費電力動作を実行しているファイルと関数を以下に示します。

アプリケーションプログラム : r_lte_ryz.c、R_LTE_Execute()

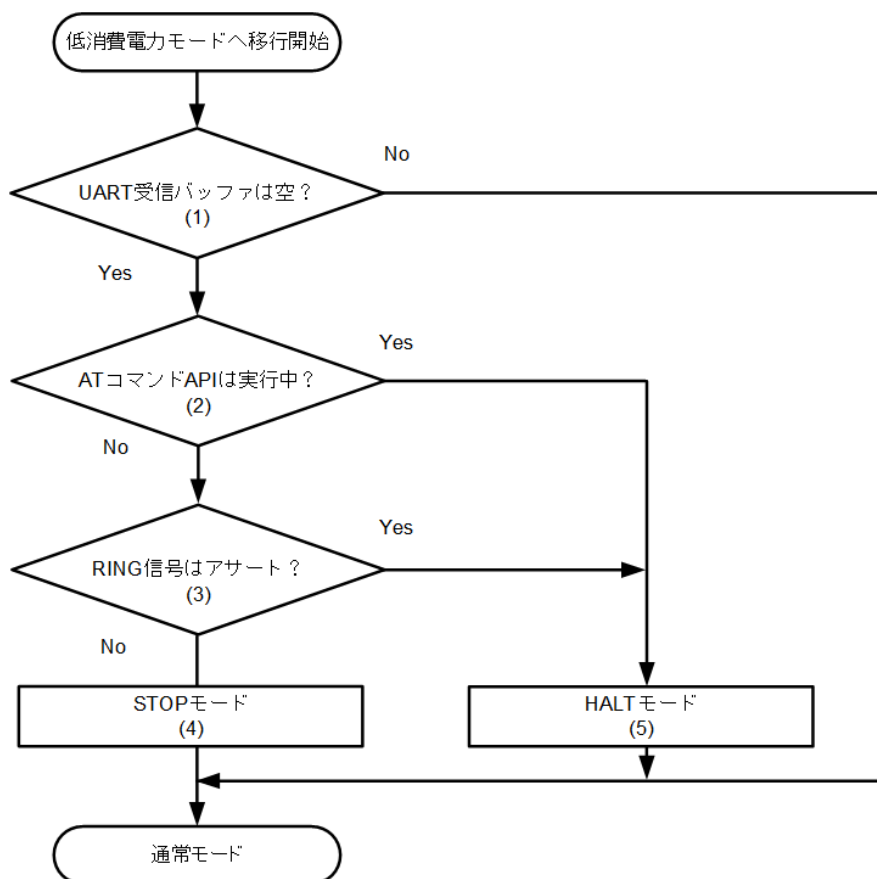


図 3-14 RL78/G22 低消費電力動作フローチャート

- (1) UART の受信バッファに文字列やデータが格納されている場合は、低消費電力モードへ移行せず R_LTE_Execute()関数で解析処理を行います。
- (2) AT コマンド API の実行中は送信した AT コマンドのレスポンスを受信した際に UART 受信割り込みで通常モードへ復帰できるように HALT へ移行します。
- (3) RING 信号がアサート中は RYZ024A からの URC が受信できるように HALT へ移行します。
- (4) STOP モードからは、SW の押下、または RING 信号アサートによる外部端子割り込みの発生により通常モードへ復帰します。
- (5) 送信した AT コマンドのレスポンス受信による UART 受信割り込みの発生により通常モードへ復帰します。

3.6.2 RYZ024A の低消費電力動作制御

RYZ024A が eDRX や PSM を有効にしている時、RTS 信号を制御することで RYZ024A を低消費電力動作させることができます。

RTS=L : 低消費電力動作禁止

RTS=H : 低消費電力動作許可

RYZ024A の低消費電力動作については「RYZ024 Power Consumption Measurements on RYZ024-Based Modules」(R19AN0167)も参照してください。

RL78/G22 から AT コマンドを送信する際、低消費電力モードになっている RYZ024A を起床させるために、RTS 信号の制御を AT コマンドマネジメントフレームワークの中で行っています。具体的には、AT コマンド API の中で RTS=Low にし、処理終了後に High にしています。また、RING 割り込みが発生した際、RYZ024A から URC が送信されるよう RTS=Low にし、必要な処理終了後に High にしています。

(1) AT コマンド API がコールされた場合

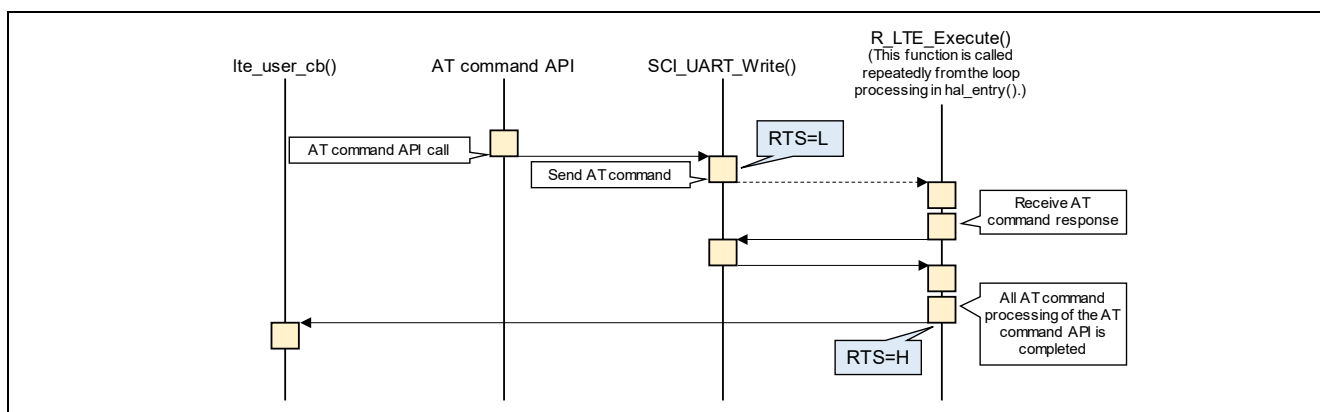


図 3-15 RTS 信号制御シーケンス(AT コマンド API がコールされた場合)

(2) RING 割り込みが発生した場合

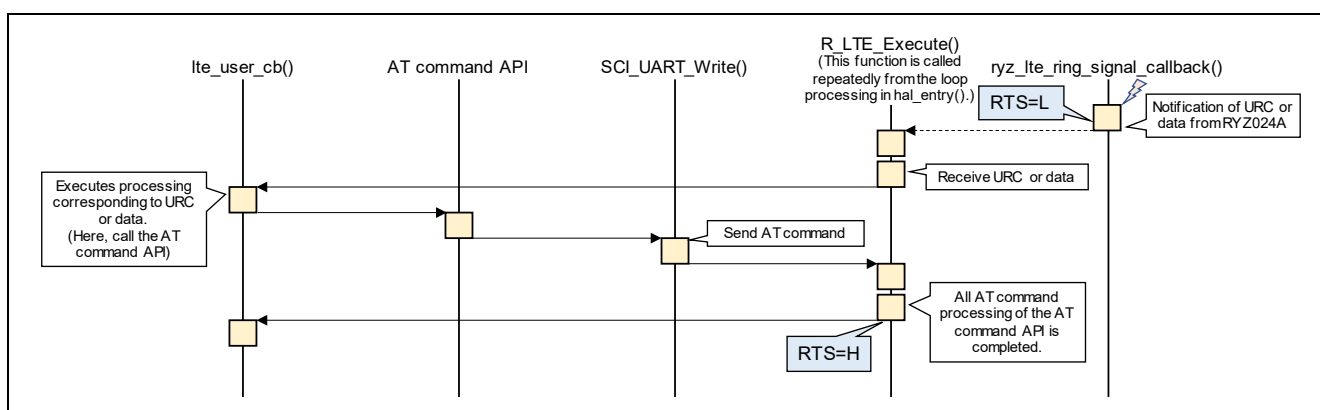


図 3-16 RTS 信号制御シーケンス(RING 割り込みが発生した場合)

eDRX と PSM の動作設定は R_LTE_EDRX_Config 関数、R_LTE_PSM_Config 関数で行います。本サンプルアプリではコールバック関数内で実行しています。そのソースコードを図 3-17 に示します。

プログラム : lte_entry.c

eDRX と PSM の動作は、デフォルトでは動作禁止になっています。有効にする場合は、「3.2.2.14 R_LTE_eDRX_Config」と「3.2.2.15 R_LTE_PSM_Config」を参照して各 API の第一引数を変更してください。

```
void lte_user_cb(uint16_t event_type, uint16_t api_id, uint16_t data_len, uint8_t *
p_data)
{
    /* 省略 */
    case LTE_API_OM_CONFIG_COMPLETE: eDRX 動作設定
    {
        /* Connect to network after configuration of operation mode complete */
        sprintf(sbuf, "OM CONFIG COMP\n");
        debug_printf(sbuf);
        R_LTE_EDRX_Config(LTE_EDRX_MODE_DISABLE, LTE_EDRX_TIME_VAL_81_SEC,
                        LTE_EDRX_PTW_TIME_VAL_10_SEC);
    } break;

    case LTE_API_EDRX_CONFIG_COMPLETE: PSM 動作設定
    {
        /* Configure PSM after configuration of eDRX complete */
        sprintf(sbuf, "eDRX CONFIG COMP\n");
        debug_printf(sbuf);
        R_LTE_PSM_Config(LTE_PSM_MODE_DISABLE, LTE_PSM_TAU_TIME_VAL_30_SEC,
                        LTE_PSM_MULTIPLIER_6, LTE_PSM_ACTIVE_TIME_VAL_2_SEC, LTE_PSM_MULTIPLIER_8);
    } break;
}
```

図 3-17 eDRX, PSM 動作設定

4. AT コマンドマネジメントフレームワークを利用したアプリケーション開発

AT コマンドマネジメントフレームワークはユーザのアプリケーション開発のベースとして利用されることを想定しています。AT コマンドマネジメントフレームワークを使用することで RL78/G22 と RYZ024A の通信アプリケーションを効率的に実装することができます。ここでは本サンプルアプリケーションを例にアプリケーションを開発する方法について説明します。

4.1 アプリケーション開発の概要

AT コマンドマネジメントフレームワークはフレームワーク内に API を効率的に追加実装できる仕様になっています。フレームワーク内に実装した API をアプリケーションプログラムでコールし、ユーザの求める動作を実現します。本サンプルアプリケーションでは以下のファイルで動作を実現しています。

- フレームワークベースプログラム
 - r_lte_ryz.c
 - r_lte_ryz.h
 - r_lte_user_config.h
- ベアメタルアプリケーションプログラム
 - lte_entry.c

フレームワークベースプログラムに実装されている API はマネジメント API と AT コマンド API の 2 種類に分類されます。

マネジメント API

マネジメント API は RYZ024A とのやり取りを管理するための API です。アプリケーションプログラムの適切な場所にコールする必要があります。また、ユーザはアプリケーション開発の際に変更する必要はありません。

マネジメント API には以下の 2 つの API が実装されています。

- R_LTE_Init
フレームワークベースプログラムの初期化を行う関数です。本関数では RYZ024A のハードウェアリセットなどが実行されます。RYZ024A は初期化が完了し、AT コマンドを受け付けることが可能になったときに "+SYSSTART" の URC を送信します。本関数では "+SYSSTART" の後、詳細なエラーレスポンスを受け取るための AT コマンド "AT+CMEE=1" を送信します。すべての AT コマンドの実行が完了した後、引数に指定したコールバック関数に API_ID = "LTE_API_INIT" のイベントが通知されます。この関数はフレームワークに実装された API の中で最初に実行して下さい。
- R_LTE_Execute
RYZ024A から受信したデータを保持・解析し、データに合わせてコールバック関数をコールしたり、AT コマンドを送信したりする関数です。本関数はコールする度にリングバッファに格納された 1 文字ずつ処理を行うので繰り返しコールする必要があります。

```

void lte_entry(void)
{
    sprintf(sbuf, "PROGRAM START\n");
    debug_printf(sbuf);

    /* SW inter */
    R_ICU_ExtIntInit(g_external_irq10.p_cfg);
    R_ICU_ExtIntInit(g_external_irq9.p_ctrl, g_external_irq9.p_cfg);

    /* Initialize framework-based program and register callback function */
    R_LTE_Init(lte_user_cb);

    while(1)
    {
        /* Execute variable process in framework-based program */
        R_LTE_Execute();
    }
}
/* 省略 */

```

R_LTE_Init を最初にコール

R_LTE_Execute をメインループで繰り返しコール

図 4-1 マネジメント API の実装 (lte_entry.c)

AT コマンド API

AT コマンド API をコールすることで実行したい動作に必要な一連の AT コマンドが送信待機リストに追加されます。登録された AT コマンドは RYZ024A からのレスポンスに対応して順次送信されます。一連の AT コマンドの実行結果はコールバック関数でアプリケーションに通知されます。ユーザは望む順番で AT コマンド API をコールしたり、コールバック関数に対応した処理を実装したりすることでアプリケーションを開発します。またユーザは、自身で新たな AT コマンド API を追加し、本サンプルアプリケーションでは利用していない AT コマンドを利用することが可能です。

```

void lte_user_cb(uint16_t event_type, uint16_t api_id, uint16_t data_len, uint8_t *
p_data)
{
    if(LTE_EVENT_API_COMPLETE == event_type)
    {
        switch (api_id)
        {
            case LTE_API_INIT:
            {
                /* Configure operation mode after initiation complete */
                sprintf(sbuf, "INIT COMP\n");
                debug_printf(sbuf);
                R_LTE_OM_Config(str_PDP_type, str_PDP_APN, str_LTE_bandlist);
            } break;

            case LTE_API_OM_CONFIG:
            {
                /* Connect to network after configuration of operation mode
complete */
                sprintf(sbuf, "OM CONFIG COMP\n");
                debug_printf(sbuf);
                R_LTE_NWK_Connect(1);
            } break;
        }
    }
}

```

1. AT コマンド API をコール

2. コールバック関数に結果が通知される

3. 次の AT コマンド API をコール

図 4-2 AT コマンド API の実装 (lte_entry.c)

4.2 AT コマンド API の追加

本フレームワークではユーザのアプリケーションに合わせて AT コマンド API を追加することを想定しています。ここでは本サンプルアプリケーションに実装されている AT コマンド API の解説と、新しい AT コマンド API の追加実装方法について説明します。

AT コマンド API を追加するときは以下の手順で追加します。

1. API ID と関数のプロトタイプ宣言の追加

追加した AT コマンド API がコールバック関数で識別できるように API ID を追加します。また AT コマンド API がアプリケーションプログラムから実行できるようにヘッダファイル(r_lte_ryz.h)にプロトタイプ宣言を追加します。

```
typedef enum
{
    LTE_API_NO_CURRENT_API = 0,
    LTE_API_OM_CONFIG,
    LTE_API_NWK_CONNECT,
    LTE_API_NWK_DISCONNECT,
    LTE_API_MQTT_CONNECT,
    LTE_API_MQTT_DISCONNECT,
    LTE_API_MQTT_SUBSCRIBE,
    LTE_API_MQTT_PUBLISH,
    LTE_API_MQTT_RCVMESSAGE,
    LTE_API_SEC_CERTIFICATEADD,
    LTE_API_SEC_CERTIFICATEREMOVE,
    LTE_API_SEC_PRIVATEKEYADD,
    LTE_API_SEC_PRIVATEKEYREMOVE,
    LTE_API_NWK_CONNECTIONCONFIG,
    LTE_API_EDRX_CONFIG,
    LTE_API_PSM_CONFIG,
    LTE_API_INIT = 0xff,
} e_lte_api_id_t;
```

図 4-3 本サンプルアプリケーションの API ID (r_lte_ryz.h)

2. AT コマンド API の実装

AT コマンド API の実体をソースファイル(r_lte_ryz.c)に実装します。本サンプルアプリケーションの AT コマンド API は以下の構成で実装されています。

引数の確認・実行中の AT コマンド API の確認

引数にポインタがある場合、NULL を指定していないことを確認します。また"gs_process_api"を確認して他の AT コマンド API が実行中でないことを確認します。実行中の場合、AT コマンド送信待機リストを変更すると、実行中の AT コマンド API が正常に動作することができないため何も実行せずにエラー"LTE_ERR_IN_PROCESS"を返却します。その後、本 AT コマンド API が実行中であることを示すため、"gs_process_api"に API_ID を登録します。

```
e_lte_err_t R_LTE_OM_Config(uint8_t* p_pdp_type, uint8_t* p_pdp_apn, uint8_t*
p_bandlist)
{
    /* Check Argument and current state */
    if((NULL == p_pdp_type) || (NULL == p_pdp_apn) || (NULL == p_bandlist))
    {
        return LTE_ERR_POINTER_NULL;
    }

    if(LTE_API_NO_CURRENT_API != gs_process_api)
    {
        return LTE_ERR_IN_PROCESS;
    }

    /* Clear ATC list and set processing API ID */
    ryz_lte_clear_atc_list();
    gs_process_api = LTE_API_OM_CONFIG;

    /* 省略 */
}
```

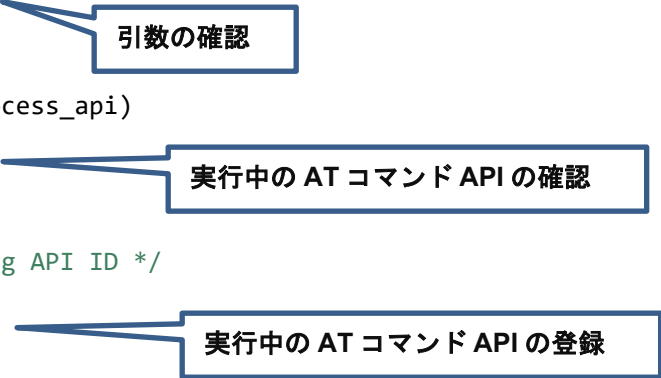


図 4-4 R_LTE_OM_Config の引数確認・実行中の AT コマンド API の確認 (r_lte_ryz.c)

送信待機リストへATコマンドの登録

送信待機リスト"gs_atc_list"に AT コマンドを文字列として登録します。送信待機リスト"gs_atc_list"には 1 つの AT コマンドに対して以下を登録する必要があります。

- **atcommand :**
実行したい AT コマンドの文字列を登録します。文字列の長さは"atcommand_size"に登録してください。登録できる文字列の最大長は 256 文字です。さらに大きい AT コマンド文字列を使用する場合はユーザ設定ファイル(r_lte_user_config.h)の"LTE_ATC_STR_SIZE"を変更してください。
- **data :**
AT コマンドによって処理したいデータ文字列のアドレスを登録するポインタです。データを送信する AT コマンドなどで必要になります。ここに登録されたデータ文字列は"> "のレスポンスに対応して送信されます。文字列の長さは"data_size"に登録してください。本ポインタに登録する文字列の実体はアプリケーションに実装することを想定しています。
- **comp_msg :**
実行したい AT コマンドが完了したと見なせるレスポンス文字列を登録します。"OK"もしくは URC を指定して下さい。文字列の長さは"comp_msg_size"に登録してください。comp_msg で指定した文字列と前方一致するレスポンスを受信した後、送信待機リストの次に登録されている AT コマンドが送信されます。"OK"と URC が連続で送信される場合、最後に送信されるレスポンスを登録してください。また送信待機リストに登録する一連の AT コマンドの最後の comp_msg によってコールバック関数で通知されるデータが変化します。詳しくは「3.3 コールバック関数」を参照してください。
- **data_exist_flag :**
送信したい AT コマンドが設定されていることを示すフラグです。R_LTE_Execute 関数ではこの値を確認することで AT コマンドが登録されていることを確認します。AT コマンドを設定した場合は"1"を設定してください。

送信待機リスト"gs_atc_list"は固定長の配列によって文字列データを保持します。そのため、登録する文字列データが送信待機リストに登録できる最大長を超過するとバッファオーバーフローによるエラーが発生する可能性があります。入力されるデータサイズが登録できる文字列の最大長を超過することが予想される場合、データサイズを確認するための処理を追加してください。

```

e_lte_err_t R_LTE_MQTT_Publish(uint8_t* p_topic, uint16_t length, uint8_t* p_message)
{
    /* 省略 */

    /* Set AT command to ATC list */
    gs_atc_list[0].atcommand_size = (uint16_t)snprintf((char*)gs_atc_list[0].atcommand,
    LTE_ATC_STR_SIZE, "AT+SQNSMQTTPUBLISH=%s,\"%s\",%s,%d\r", "0",p_topic,"0",length);
    gs_atc_list[0].comp_msg_size = (uint16_t)snprintf((char*)gs_atc_list[0].comp_msg,
    LTE_ATC_STR_SIZE, "%s", "OK");
    gs_atc_list[0].data_exist_flag = 1;

    gs_atc_list[0].data = p_message;
    gs_atc_list[0].data_size = length;

    if(gs_atc_list[0].atcommand_size > LTE_ATC_STR_SIZE)
    {
        ryz_lte_clear_atc_list();
        return LTE_ERR_DATASIZE_OVERFLOW;
    }
}

```

送信待機リストの先頭に以下を追加

atcommand =
"AT+SQNSMQTTPUBLISH=0,"[p_topic]"0,[length] "
comp_msg = "OK"
data_exist_flag = 1

data には送信したいデータのアドレスを設定

引数を送信待機リストに登録するため、
データサイズの確認

図 4-5 R_LTE_MQTT_Publish の AT コマンドの登録 (r_lte_ryz.c)

先頭の AT コマンドの送信

登録した送信待機リストの先頭の AT コマンドを送信します。以降の AT コマンドの送信はレスポンスに対応して R_LTE_Execute 関数で行われます。

```

e_lte_err_t R_LTE_OM_Config(uint8_t* p_pdp_type, uint8_t* p_pdp_apn, uint8_t*
p_bandlist)
{
    /* 省略 */

    /* Send first AT command from ATC list */
    ryz_lte_transmit_atc_list(LTE_TRANSMIT_ATCOMMAND);

    return LTE_SUCCESS;
}

```

送信待機リストの先頭の AT コマンドを送信

図 4-6 R_LTE_OM_Config の先頭の AT コマンドの送信 (r_lte_ryz.c)

4.2.1 データ受信を伴う AT コマンド API

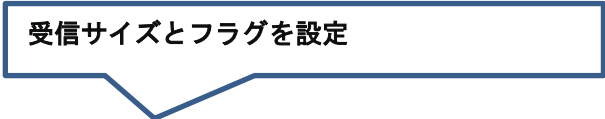
本サンプルアプリケーションでコールされている R_LTE_MQTT_RcvMessage 関数のように AT コマンド送信後に任意のデータ受信を行う AT コマンド API をコールするにはフレームワーク内のグローバル変数の書き換えが必要になります。

データ受信を行う場合、グローバル変数"gs_ryz_lte_receive_size"と"gs_ryz_lte_receive_flag"を変更する必要があります。"gs_ryz_lte_receive_size"には受信したいデータのサイズを、"gs_ryz_lte_receive_flag"はマクロ"LTE_RCV_DATA_FLAG_ON"を設定します。

```
e_lte_err_t R_LTE_MQTT_RcvMessage(uint8_t* p_topic, uint8_t message_id, uint16_t
message_size)
{
    /* 省略 */

    /* Set receive flag and size for data receive operation */
    gs_ryz_lte_receive_size = message_size;
    gs_ryz_lte_receive_flag = LTE_RCV_DATA_FLAG_ON;

    /* 省略 */
}
```



受信サイズとフラグを設定

図 4-7 R_LTE_MQTT_RcvMessage のグローバル変数設定 (r_lte_ryz.c)

この時に受信したデータは、アプリケーションへコールバック関数で通知されます。コールバック関数はデータの後に RYZ024A から送信される「OK」の応答を受け取ったタイミングでコールされます。

※注意：受信データ内の「\r」または「\n」は RYZ024A 内で「\r\n」に変換されて RL78/G22 に送信されます。そのため、受信データの内容やサイズの一部が変更される場合があります。

4.3 エラー発生処理のガイドライン

通信制御のシステムでは、通信コントローラの制御やネットワーク動作における様々なエラー発生を想定してアプリケーションを開発する必要があります。ここではその検出と処理について、本 AT コマンドマネジメントフレームワークを使用してアプリケーション開発を行う場合のガイドラインを示します。実際には応用製品への要求事項によって処理は変わりますので、参考情報の扱いをお願いします。

また、「RYZ024 Module System Integration Guide」(R19AN0101)に記載されている「Connection Manager」の説明も参照してください。

UART 通信及び RYZ024A の動作エラー

不具合の状況	フレームワークの挙動	アプリケーションの対応
意図せず RYZ024A が再起動される	“+SYSSTART”を受信する。意図しない“+SYSSTART”を受信するため、アプリケーションにコールバック関数で通知する。	イベント "LTE_EVENT_FATAL_ERROR"でコールバック関数がコールされる。本イベントに R_LTE_Init 関数を使用して初期化することが推奨される。
RYZ024A から RL78/G22 への UART 通信でビットエラーあるいは文字の受信ミスが起きる	文字列が comp_msg で指定した文字列と合致しない場合もしくは文字列が“\n”で終了しない場合、タイムアウトが発生し、コールバック関数でアプリケーションに通知する。	イベント "LTE_EVENT_TIMEOUT_ERROR"でコールバック関数がコールされる。本イベントに対しては R_LTE_Init 関数を使用して初期化することが推奨される。
	受信文字列が comp_msg で指定した URC と前方一致する場合、コールバック関数にてアプリケーションに通知される。	イベント "LTE_EVENT_RCVURC"でコールバック関数がコールされる。P_data に受信した文字列が登録されているため、データを確認する。
RL78/G22 から RYZ024A への UART 通信でビットエラーあるいは文字の受信ミスが起きる	送信した文字列に対するレスポンスがない場合、タイムアウトが発生し、コールバック関数でアプリケーションに通知する。	イベント "LTE_EVENT_TIMEOUT_ERROR"でコールバック関数がコールされる。本イベントに対しては R_LTE_Init 関数を使用して初期化することが推奨される。
	送信した AT コマンドの一部が間違っている場合、エラーのレスポンスを受信する。エラーレスポンスを受信後、コールバック関数でアプリケーションに通知する。	イベント "LTE_EVENT_ERROR"でコールバック関数がコールされる。エラーコード LTE_CME_ERR_OPERATION_NOT_SUPPORTED"(0x04)が p_data にて通知されるので対応した処理を追加する。
MCU の送信と RYZ024A の送信タイミングが重なり、RYZ024A が期待した動作をしない	動作が停止することでタイムアウトが発生する。タイムアウト発生時にコールバック関数でアプリケーションに通知する。	イベント "LTE_EVENT_TIMEOUT_ERROR"でコールバック関数がコールされる。本イベントに対しては R_LTE_Init 関数を使用して初期化することが推奨される。

RYZ024A からの CTS が長時間イネーブルにならない	長時間 RYZ024A からレスポンスを受け取ることができず、タイムアウトが発生する。タイムアウト発生時にコールバック関数でアプリケーションに通知する。	イベント "LTE_EVENT_TIMEOUT_ERROR"でコールバック関数がコールされる。本イベントに対しては R_LTE_Init 関数を使用して初期化することが推奨される。
--------------------------------	--	---

ネットワーク通信エラー

不具合の状況	フレームワークの挙動	アプリケーションの対応
電波状況の悪化などでネットワークが切断される	"+CEREG"の URC を受信する。コールバック関数でアプリケーションに通知される。	イベント "LTE_EVENT_RCVURC"でコールバック関数がコールされる。URC のパラメータを確認し、対応した処理を実行する。URC のパラメータは AT コマンドマニュアルを確認する。
ネットワークに接続しようとしたが APN の間違いなどによって接続できなかった	"+CEREG"の URC を受信する。コールバック関数でアプリケーションに通知される。	イベント "LTE_EVENT_RCVURC"でコールバック関数がコールされる。URC のパラメータを確認し、対応した処理を実行する。URC のパラメータは AT コマンドマニュアルを確認する。
その他、何らかの理由でコネクションが切断される。	"+SQNSH"の URC を受信する。コールバック関数でアプリケーションに通知される。	イベント "LTE_EVENT_RCVURC"でコールバック関数がコールされる。URC のパラメータを確認し、対応した処理を実行する。URC のパラメータは AT コマンドマニュアルを確認する。

通信状況は URC"+CEREG"などによって通知されます。以下に RYZ024A から通知される通信状況の URC 例を説明します。

「RYZ024 Module System Integration Guide」(R19AN0101)に記載されている「Connection Manager」の説明も併せて参照してください。

- URC"+CEREG: 80"または"+CEREG: 4"を受信：
 - 一時的にネットワークから切断されたときに通知される URC です。再度ネットワークに接続しようとしているため、電波の状況などが改善すれば、AT コマンド API を実行することなくネットワークに再接続することができます。この時、RYZ024A 内で MQTT 通信は保持されているため、ネットワークに再接続した際に R_LTE_MQTT_Connect 関数を実行することなく MQTT 通信を再開することができます。
- URC"+CEREG: 0"を受信：
 - ネットワークから切断されたときに通知される URC です。電波の状況などが改善すれば自動的に再接続します。なお上位層においては、例えば TCP ソケットが切断されると+SQNSH 等の URC が通知されますので、必要に応じて TCP 接続等の処理が必要です。
- URC" +SQNSMQTTTONCONNECT: 0,-7"を受信
 - ネットワークから切断されて一定時間たった後 MQTT 通信も切断したときに通知される URC です。ネットワークに再接続しても MQTT 通信は保持されていないので再度 R_LTE_MQTT_Connect 関数を実行する必要があります。

4.4 PMOD-RYZ024A 固有の処理

RYZ024A はディープスリープ状態になると UART の CTS 信号が Hi-Z となります。通常の回路では、CTS 信号をプルアップし High レベルにすることで、RYZ024A がディープスリープ状態の時は HW フロー制御により RL78/G22 から AT コマンドを送信できないように制御できます。

しかしながら PMOD-RYZ024A では、使用しているレベルシフタの特性により RYZ024A がディープスリープに移行した状態でもレベルシフタから RL78/G22 への CTS 信号が Low レベルのままになり HW フロー制御ができない状態になります。

そのため、本サンプルアプリケーションでは RYZ024A がディープスリープ状態のときに RL78/G22 から AT コマンドを送信する場合、最初に"AT+CFUN?"コマンドを送信し RYZ024A が起床したこと確認します。"AT+CFUN?"コマンドの応答が返らない時は数回リトライを行います。"OK"を受信した後に目的の AT コマンド(例えば"AT+SQNSMQTTPUBLISH")を送信します。

「図 4-8 AT+CFUN?コマンドの登録 (r_lte_ryz.c)」に、送信待機リストへ"AT+CFUN?"コマンドを追加する方法を示します。

```
e_lte_err_t R_LTE_MQTT_Publish(uint8_t* p_topic, uint16_t length, uint8_t* p_message)
{
    /* 省略 */

    /* Set AT command to ATC list */
    gs_atc_list[0].atcommand_size = (uint16_t)snprintf((char*)gs_atc_list[0].atcommand,
    LTE_ATC_STR_SIZE, "AT+CFUN?\r");
    gs_atc_list[0].comp_msg_size = (uint16_t)snprintf((char *)gs_atc_list[0].comp_msg,
    LTE_ATC_STR_SIZE, "OK");
    gs_atc_list[0].data_exist_flag = 1;
    gs_atc_list[0].at_polling_flag = 1;

    gs_atc_list[1].atcommand_size = (uint16_t)snprintf((char*)gs_atc_list[1].atcommand,
    LTE_ATC_STR_SIZE, "AT+SQNSMQTTPUBLISH=%s,\"%s\",%s,%d\r", "0",p_topic,"0",length);
    gs_atc_list[1].comp_msg_size = (uint16_t)snprintf((char*)gs_atc_list[1].comp_msg,
    LTE_ATC_STR_SIZE, "%s", "OK");
    gs_atc_list[1].data_exist_flag = 1;
    gs_atc_list[1].data = p_message;
    gs_atc_list[1].data_size = length;

    if(gs_atc_list[1].atcommand_size > LTE_ATC_STR_SIZE)
    {
        ryz_lte_clear_atc_list();
        return LTE_ERR_DATASIZE_OVERFLOW;
    }

    /* 省略 */
}
```

送信待機リストの先頭に“AT+CFUN?”を追加

atcommand = “AT+CFUN?”
comp_msg = “OK”
data_exist_flag = 1
at_polling_flag=1

次の送信待機リストに
“AT+SQNSMQTTPUBLISH”を追加

図 4-8 AT+CFUN?コマンドの登録 (r_lte_ryz.c)

なお、適切なレベルシフタを用いることにより、RYZ024A がディープスリープ状態になった場合も CTS 信号で HW フロー制御が可能なボードではこの処理は不要となります。

PMOD-RYZ024A 固有の処理を有効または無効にする定義を「表 4-1 PMOD-RYZ024A 固有処理」に示します。

表 4-1 PMOD-RYZ024A 固有処理の定義(r_lte_ryz.c)

定義名	値	説明
PMOD_RYZ024A	1	1: AT+CFUN?コマンドを送信する PMOD-RYZ024A 固有の処理を有効化します。 0: PMOD-RYZ024A 固有の処理を無効化します。

"AT+CFUN?"コマンドの動作を設定する定義を「表 4-2 AT+CFUN?コマンドの動作設定」に示します。

表 4-2 AT+CFUN?コマンドの動作設定(r_lte_ryz.c)

定義名	値	説明
AT_POLLING_TIMETOUT	2	"AT+CFUN?"コマンドのタイムアウトカウント。 単位: 2sec 例) 2sec * 2 = 4sec
AT_POLLING_RETRY_COUNT	2	"AT+CFUN?"コマンドのリトライ回数。

4.5 PMOD-RYZ024A の初期化

本アプリケーションノートのサンプルアプリケーションの実行の前に PMOD-RYZ024A を使用されていた場合、RYZ024A の不揮発メモリに設定が保存されている場合があります。以下の AT コマンドを実行することにより、初期化することが出来ます。

```
AT+CGDCONT=1,"IPV4V6"," ppsim.jp"
```

```
OK
```

```
AT+CGDCONT?
```

```
+CGDCONT: 1,"IPV4V6","ppsim.jp",,,,0,0,0,0,0,0,0,,0
```

後で初期化されたことを確認するために設定します。

```
OK
```

```
AT+SQNSFACTORYRESET
```

```
ERROR
```

```
AT^RESET
```

```
OK
```

応答が **ERROR** でない場合もあります。

```
+SHUTDOWN
```

```
+SYSSTART
```

```
OK
```

```
AT+CGDCONT?
```

```
+CGDCONT: 1,"IPV4V6","",,,,0,0,0,0,0,0,1,,0
```

設定が初期化され、デフォルト値が表示されることを確認します。

```
OK
```

図 4-9 PMOD-RYZ024A の初期化

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2023/ 8/30	—	第一版
1.10	2023/10/30	13	2.1 アプリケーションの動作環境 5 項 低消費電力モードを実行する API 名を変更
1.20	2024/02/09	7 8 45	ポート設定と RTS 設定条件を変更 1.2.6 コードサイズを更新 2.1 アプリケーションの動作環境 表 2 2 サンプルアプリケーションのソフトウェア環境を更新 3.4 ユーザ固有値の設定 表 3 4 RL78/G22 のスマート・コンフィグレータ設定 Config_INTC, Config_PORT を変更 (サンプルプログラム(ryz_lte_rts_high_devspe) の変更)

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くと、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。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、これに限られません。）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た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しします。
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。