RL78/G22

DA16200/DA16600 Wi-Fi 通信 (Soft AP モード)

要旨

Wi-Fi 機器のステーション(STA)モードはアクセスポイント(AP)モードと接続して使用します。従って、スマートフォン等のおもにステーション(STA)モードで動作するデバイスは、一般的に Wi-Fi ルータのアクセスポイント(AP)に接続をして使用します。Wi-Fi 機器をアクセスポイント(AP)モードで動作させることで Wi-Fi ルータを使用せずダイレクト接続できます。これにより、Wi-Fi ルータが不要でスマートフォン等の TCP クライアントと 1 対 1 で通信が可能になり、スマートフォン等からの機器制御、機器状態をスマートフォン等への通知をすることに利用できます。この機能で、” Wi-Fi ガレージシャッターコントローラー “に応用できます。

本アプリケーションノートでは、RL78/G22 と DA16200 / DA16600 を使用して、AP モードの Wi-Fi 通信を行う方法を説明します。DA16200 / DA16600 は STA モードと AP モード の Wi-Fi 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 DA16200 / DA16600 のホスト MCU として動作します。サンプルプログラムでは、ホスト MCU の RL78/G22 から DA16200 / DA16600 の制御し、TCP 通信を行います。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 DA16200 / DA16600 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、DA16200 / DA16600 の Wi-Fi 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、本サンプルプログラムに実装された Wi-Fi 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。

動作確認デバイス

RL78/G22

DA16200/DA16600

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)
- DA16200 / DA16600 Host Interface and AT Command User Manual (UW-WI-0003)
- DA16200 / DA16600 SDK Update Guide(R12AN0129)

目次

1. 概要

1.1 動作概要

DA16200 / DA16600 は Wi-Fi 通信機能を持つモジュールです。Wi-Fi 通信機能は RL78/G22 から UART 経由で AT コマンドによって文字列を入力することで制御することができます。

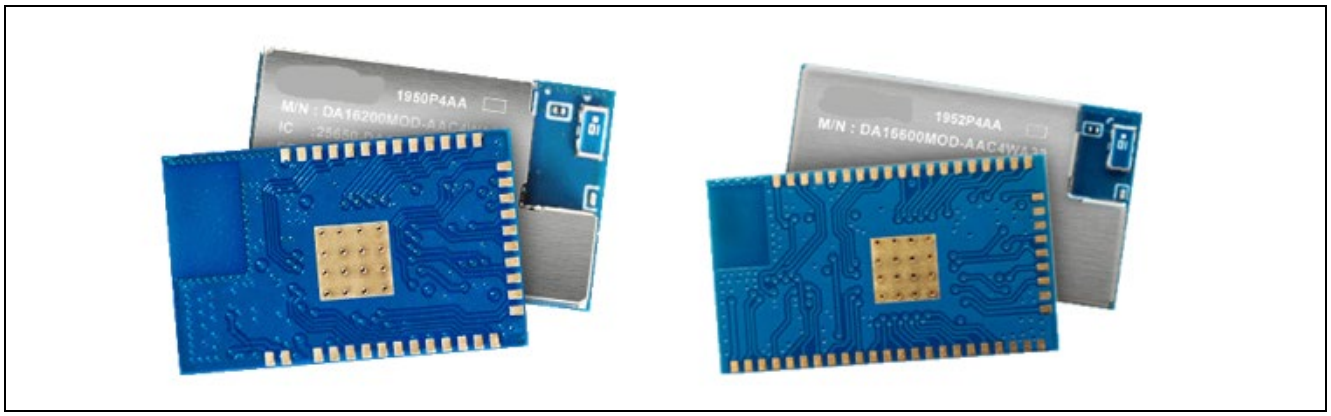

図 1-1 DA16200 /DA16600

本サンプルプログラムは、ホスト MCU として RL78/G22 を使用し DA16200 / DA16600 の Wi-Fi 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 DA16200 / DA16600 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 DA16200 / DA16600 の TCP 通信機能を利用します。

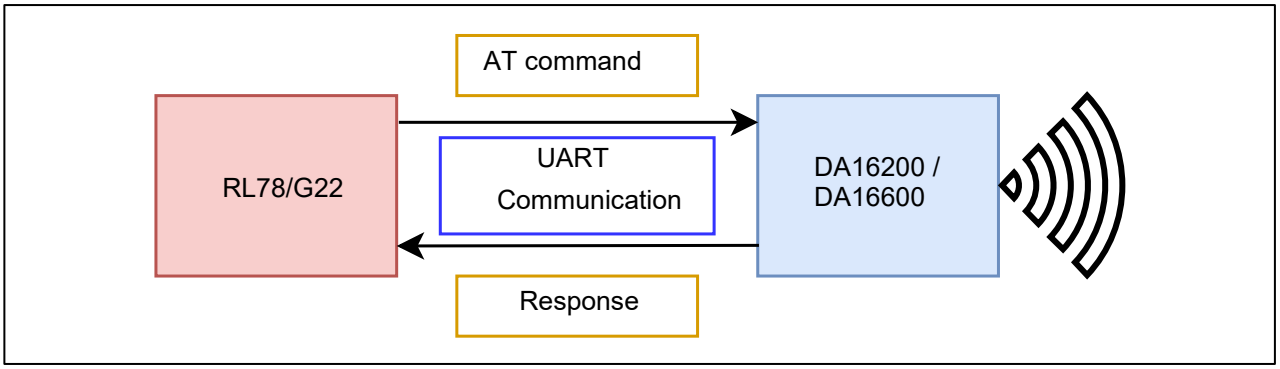

図 1-2 RL78/G22 と DA16200 / DA16600 の通信

1.2 ソフトウェア説明

このサンプルプログラムでは、Wi-Fi の Soft AP モードの TCP Server として動作をするように初期化をします。スマートフォン等の Wi-Fi の STA モードの接続と TCP Client からの接続後、TCP Client が ASCII データで”on”と送ると RL78/G22 Fast Prototyping Board (RL78/G22 FPB) の LED をオン、そして、ASCII データで”off”と送ると RL78/G22 FPB の LED をオフにします。RL78/G22 FPB のスイッチ(SW)を押すことで TCP Client に文字列を送ります。

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/デレクトリ名	説明
r01an7287xxrrrr-rl78g22_wifi.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_DA16200 / DA16600	Soft AP モード動作の サンプルプログラム

1.2.2 サンプルプログラムの階層

サンプルプログラムのソフトウェア階層は以下の通りです。

Smart Configurator は自動コード生成ツールで、緑の 3 つの関数については Smart Configurator で生成されたプログラムを使用しています。

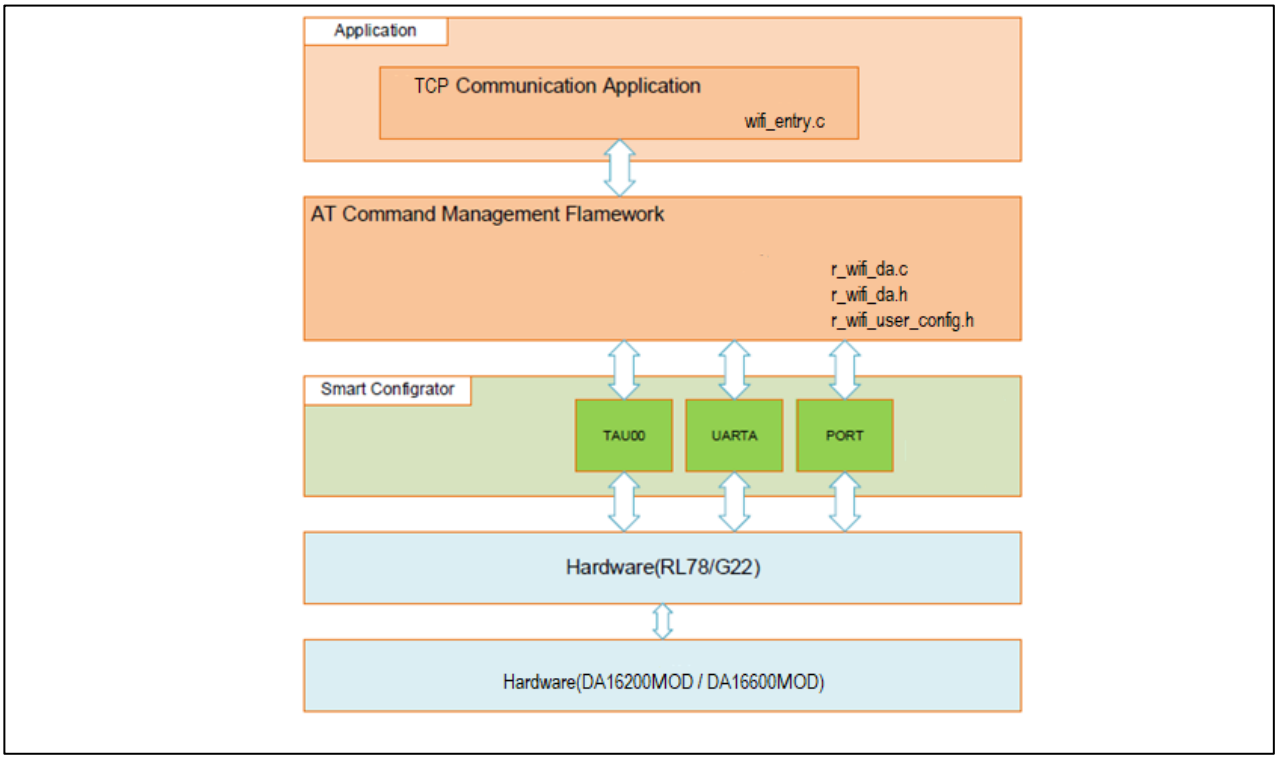

図 1-3 ソフトウェア構成

本サンプルプログラムは RL78/G22 から DA16200 / DA16600 を制御して TCP 通信を行うプログラムです。このプログラムは DA16200 / DA16600 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 TCP 通信するための API をコールする TCP 通信アプリケーションの 2 つで構成されています。

TCP 通信アプリケーションは TCP クライアントからデータを受信した後、LED をオン/オフにするプログラムです。TCP 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 URC 受信を利用して作成されています。TCP 通信アプリケーションの詳しい説明については「2 TCP 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 DA16200 / DA16600 への AT コマンド送信と DA16200 / DA16600 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 DA16200 / DA16600 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。

本サンプルプログラムでは RL78/G22 が DA16200 / DA16600 を通じて TCP 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 TCP 通信以外の DA16200 / DA16600 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 (未使用)
PORT (P50)	UART 用 CTS 信号 (未使用)
PORT (P17)	DA16200 / DA16600 RESET 制御 (未使用)
SW(P137/INTP0)	ユーザスイッチの外部端子割り込み
INTP(P51/INTP2)	DA16200 / DA16600 INT 信号の外部端子割込み (未使用)
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	LVD0 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
r01an7287_rl78g22_wifi	プログラム格納用フォルダ
├ src	ソースファイル
│ ├ da16	DA16200 / DA16600 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	
│ ├ wifi_entry.c	サンプルプログラムのメインプログラム
│ ├ wifi_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 RL78/G22 のサンプルプログラムの 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	17,898 Byte
RAM	1,390 Byte

2. TCP 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

TCP 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。
本サンプルプログラム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 サンプルプログラム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)
US159-DA16200EVZ DA16200 Wi-Fi Pmod™ board *	DA16200 モジュール搭載 PMOD ボード (US159-DA16200MEVZ)
US159-DA16600EVZ Wi-Fi + Bluetooth® Low Energy Combo Pmod™ Board *	DA16600 モジュール搭載 PMOD ボード (US159-DA16600EVZ)
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

* DA16200 / DA16600 のソフトウェアは、DA16200 DA16600 FreeRTOS SDK Image v3.2.8.1 を使用。

本サンプルプログラム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 サンプルプログラム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プログラムの実行準備を行います。

1. US159-DA16200EVZ / US159-DA16600EVZ のファームウェアを念のため、最新版に更新します。
   更新方法は、DA16200/DA16600 SDK Update Guide (renesas.com)を参照してください。
   動作確認は、DA16200 DA16600 FreeRTOS SDK Image v3.2.8.1 でおこなっています。

2. RL78/G22 FPB と US159-DA16200EVZ / US159-DA16600EVZ 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。また、J17 2-3 ショートをして 3.3V 電源を選択します。

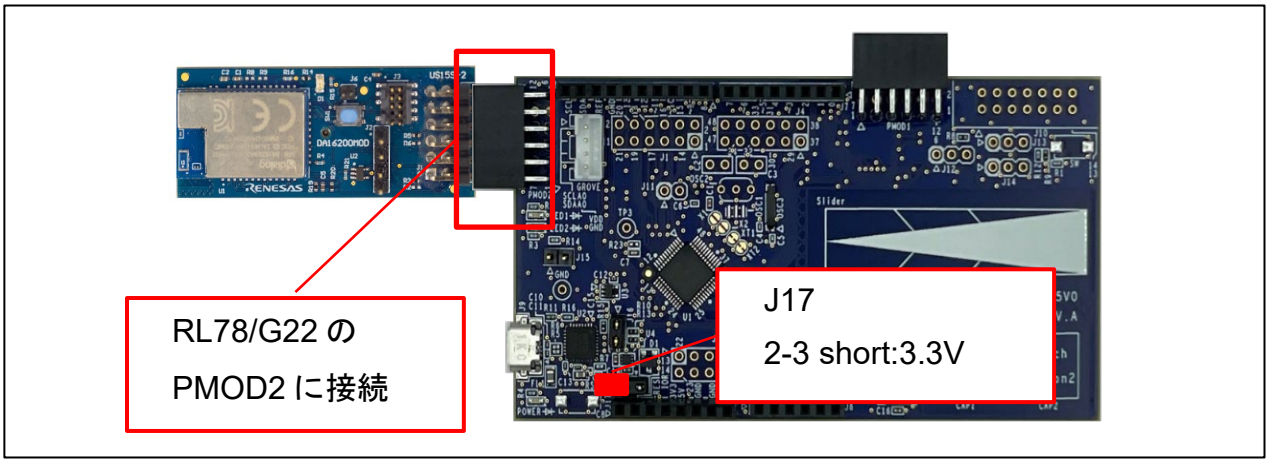

図 2-1 RL78/G22 FPB と US159-DA16200EVZ / US159-DA16600EVZ を接続

3. RL78/G22 FPB に USB ケーブルを接続します。

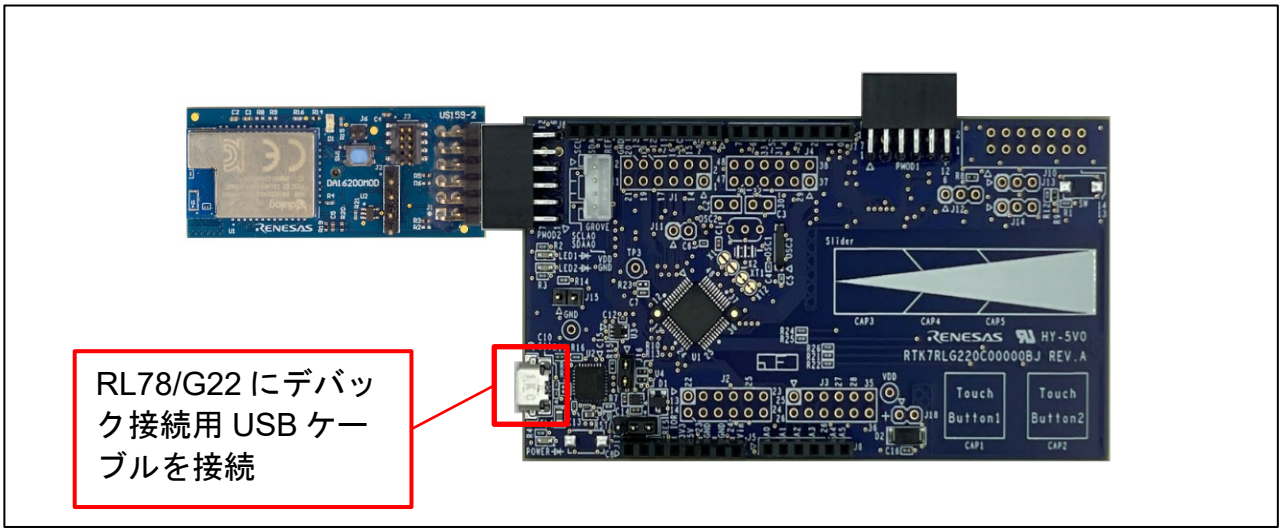

図 2-2 USB ケーブルの接続

4. 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）。

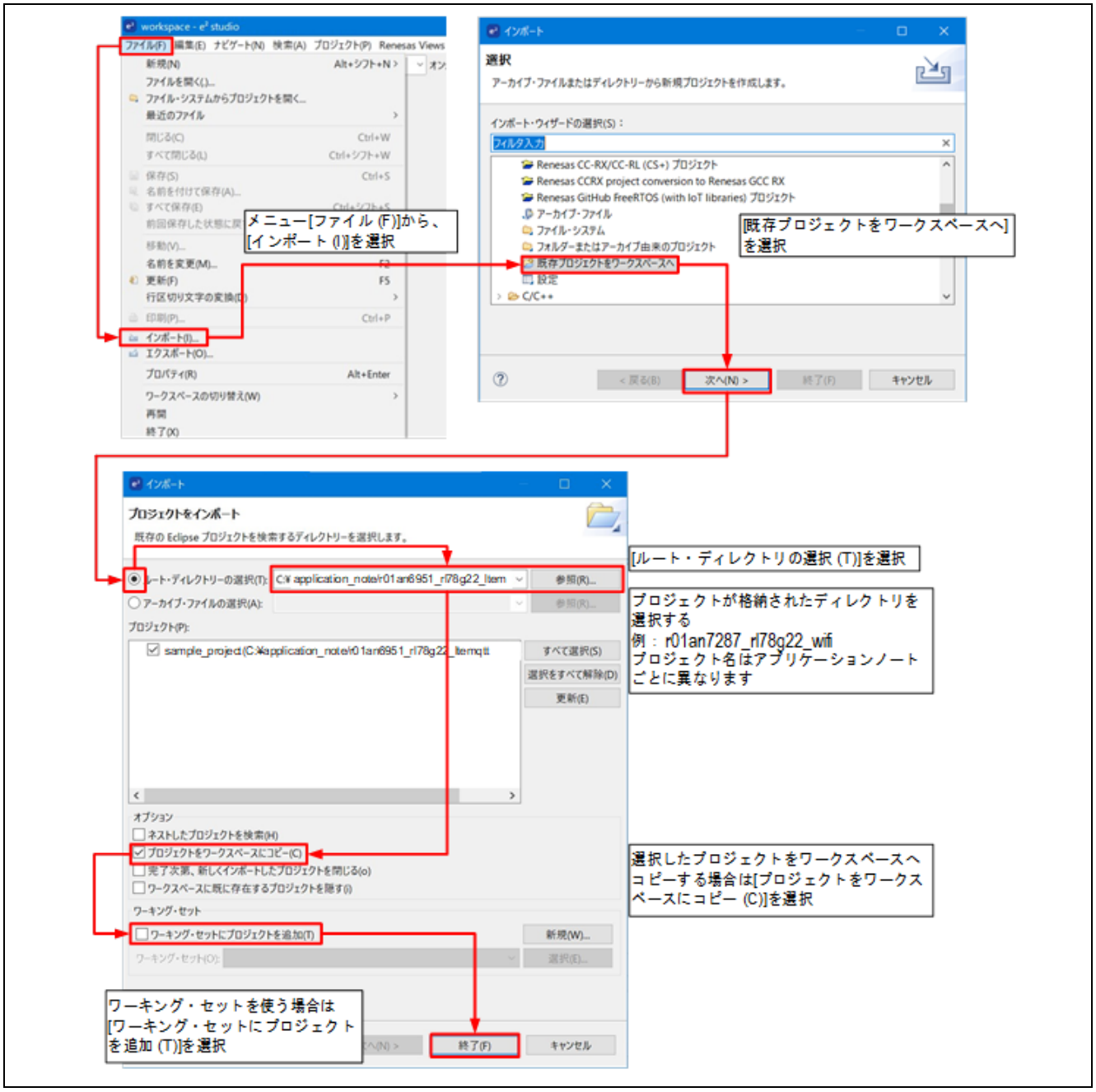

図 2-3 プロジェクトを e² studio にインポート 例: r01an7287_rl78g22_wifi

・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。

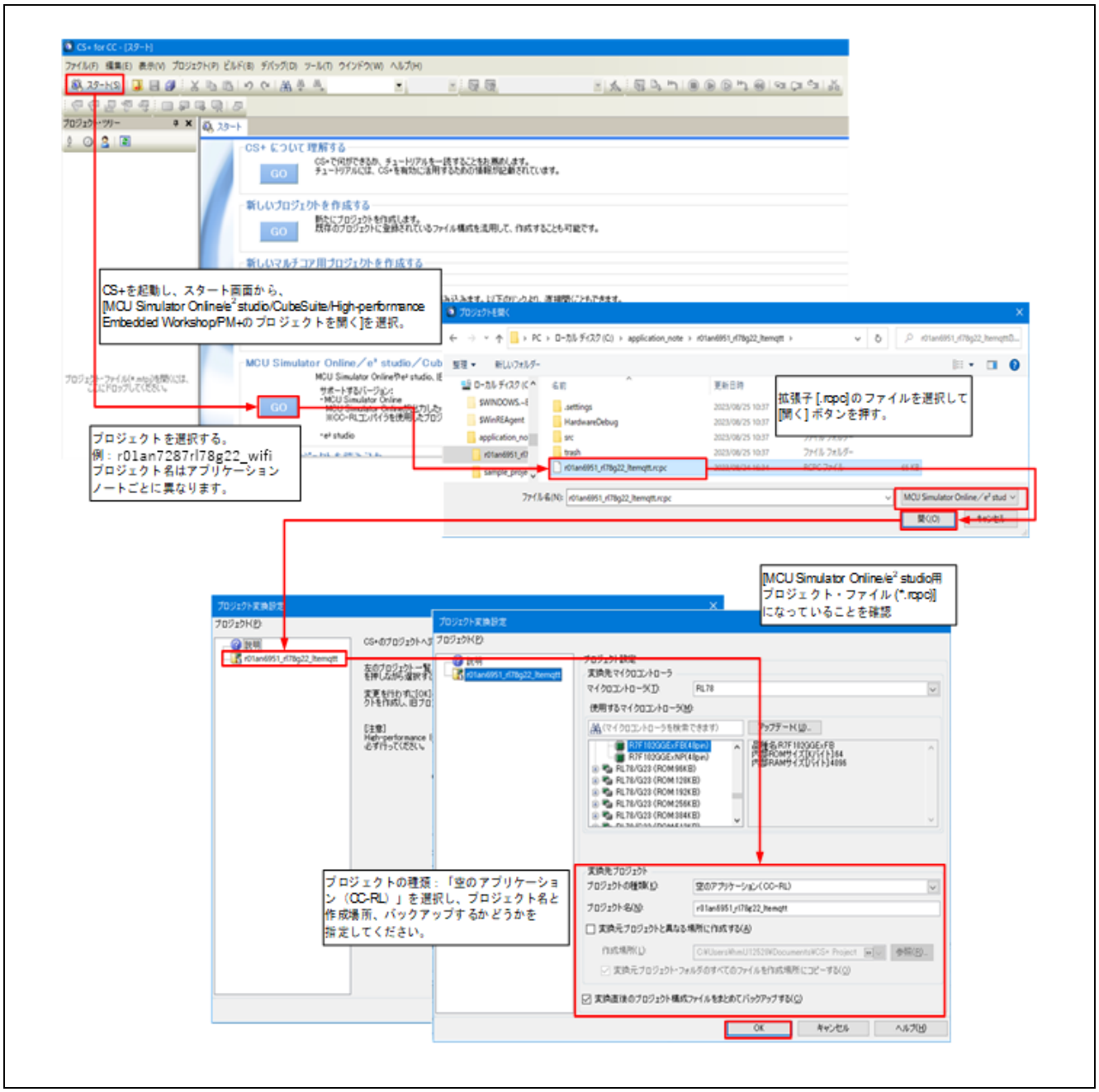

図 2-4　プロジェクトを CS+ にインポートする方法

5. SSID、パスワード、カントリ名、セキュリティプロトコル、暗号種類を変更します。Wi-Fi ネットワークへ接続する際に使用する IP アドレス、サブネットマスク。DHCP サーバ設定範囲のスタート IP、エンド IP。そして、TCP サーバのローカルポートを指定されています。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。

   ・wifi_entry.c

   設定内容の詳細は、下記ドキュメントを参照してください。

   UM-WI-003 DA16200 DA16600 Host Interface and AT Command User Manual (renesas.com)

```
50
51          /* Application specific string data for AP */
52          static uint8_t s_str_ap_ssid[]     = "test1234";
53          static uint8_t s_str_ap_password[] = "12345678";
54
55          /* Country code (optional). If exists, code is essential */
56          static uint8_t s_str_ap_country[]  = "JP";
57
58          /* Operating channel (optional). Default is 1 or uses the current channel if Soft AP is operating */
59          static uint8_t s_str_ap_ch[]       = "0";
60
61          /* Security protocol. 0 (OPEN), 2 (WPA), 3 (WPA2), 4 (WPA+WPA2) ),
62             5 (WPA3 OWE), 6 (WPA3 SAE), 7 (WPA2 RSN & WPA3 SAE) */
63          static uint8_t s_str_ap_sec[]      = "3";
64
65          /* Encryption. 0 (TKIP), 1 (AES), 2 (TKIP+AES) */
66          static uint8_t s_str_ap_enc[]      = "1";
67
68          /* IP Address Setting */
69          static uint8_t s_str_ip_addr[]     = "10.0.0.1";
70          static uint8_t s_str_ip_netmask[]  = "255.255.255.0";
71
72          /*Network DHCP Server setting */
73          static uint8_t s_str_dhcp_staip[]   = "10.0.0.2";
74          static uint8_t s_str_dhcp_endip[]   = "10.0.0.11";
75
76          /* TCP server socket setting */
77          static uint8_t s_str_local_port[]   = "10194";
78
```

図 2-5 SSID,パスワード等の設定 (wifi_entry.c)

この設定のもとに、下記 AT コマンドを実行しています。

```
ATF
+INIT:DONE,0
AT+TMRFNOINIT=0
OK
AT+WFMODE=1
OK
AT+WFSAP=test1234,3,1,12345678,0,JP
+WFSAP:test1234
OK
AT+RESTART
OK
+INIT:DONE,1
(The received URC is different between DA16200 and DA16600.)
AT+NWIP=1,10.0.0.1,255.255.255.0,10.0.0.1
OK
AT+NWDHS=1,10.0.0.2,10.0.0.11,1800
OK
AT+TRTS=10194
+TRTS:0
OK
AT+TRTRM=0
OK
AT+TRTS=10194
+TRTS:0
OK
```

図 2-6 AT コマンドの実行内容（Soft AP モード設定）

6. Wi-Fi モジュールの DA16200 と DA16600 はリセット後、もしくはリスタート後の初期化時間が異なり、異なるウエイト時間を設定ができるようになっています。DA16600 は、初期化時間が長いため DA16600 に設定している場合、DA16200 と DA16600 は同じプログラムで動作できます。サンプルプログラムのデフォルトでは DA16600 に設定しています。変更するファイルは以下の通りです。

・da16\r_wifi_da.c

```
46  /**********************************************************************************
47   * Macro definition
48   **********************************************************************************/
49  /* 1: DA16600 setting, 0: DA16200 (Adjust period)  */
50  /* Setting due to different initialization times of DA16200 and DA16600 */
51  #define DA16600         (1)
52
```

図 2-7 DA16200 / DA16600 の選択

7. 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

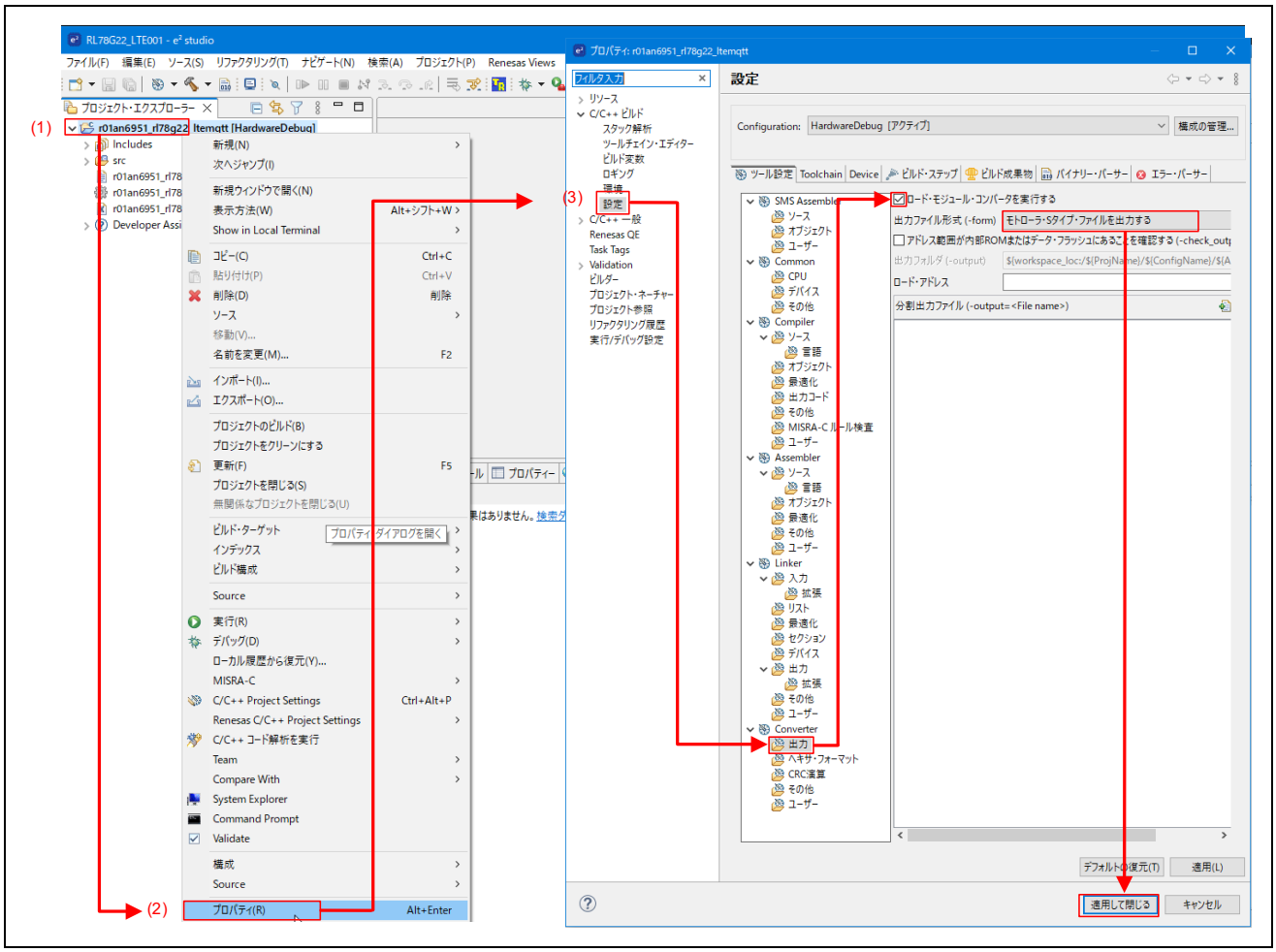

図 2-8　モトローラ・S タイプ・ファイル出力設定

8. 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。

9. 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

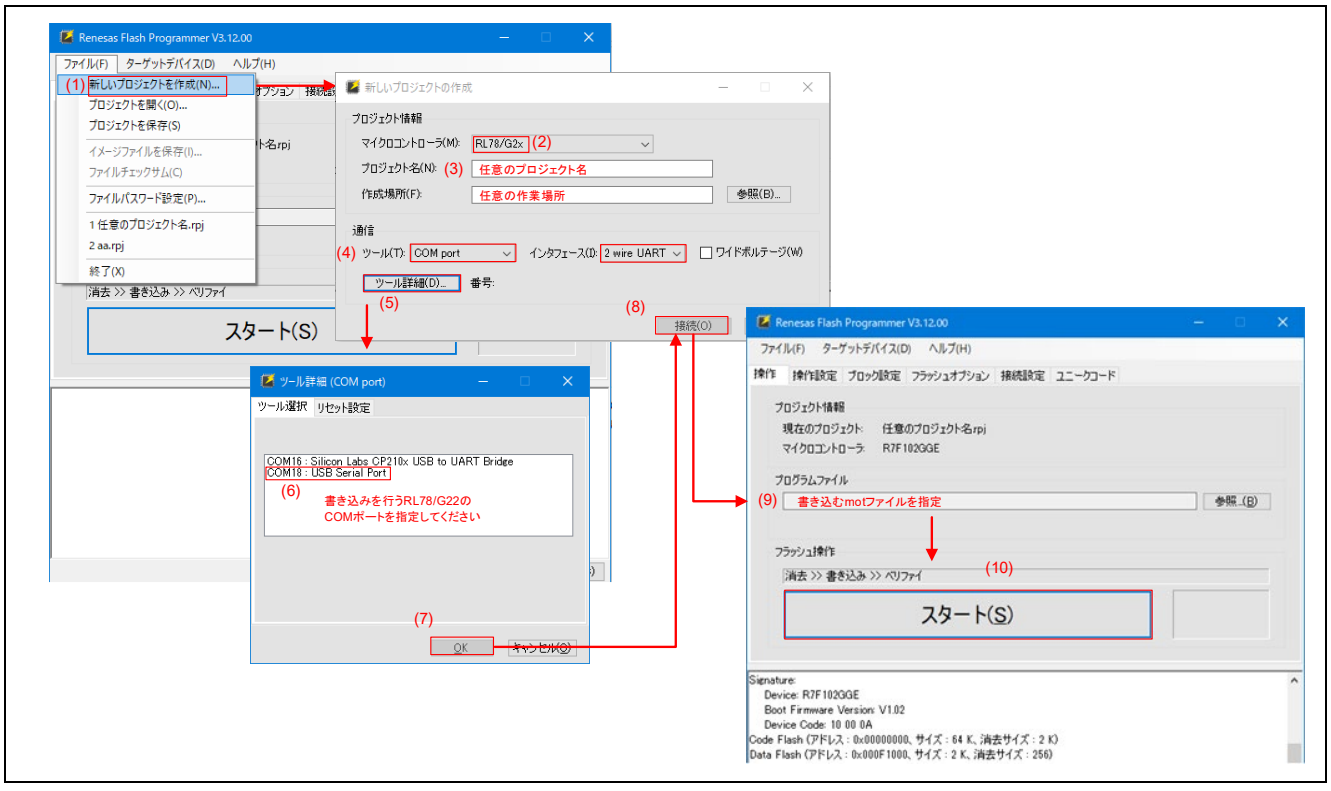

図 2-9　ファイルの書き込み

10. 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ログは、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、Wi-Fi の Soft AP モードの TCP Server として動作をするように初期化をして、スマートフォン等の Wi-Fi の STA モードのコネクトと TCP Client からコネクト後、ASCII データで”on”と送ると RL78/G22 FPB の LED をオン、ASCII データで”off”と送ると ASCIIRL78/G22 FPB の LED をオフにします。また、RL78/G22 FPB のスイッチを押すことで、スマートフォンに文字列を送ります。

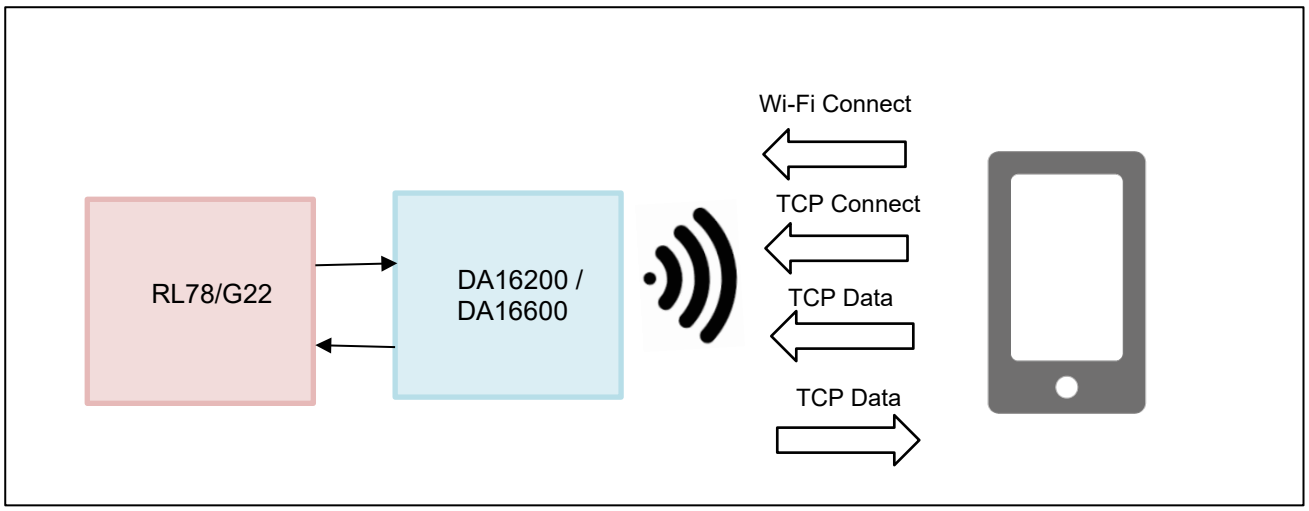

図 2-10 サンプルプログラムシステム構成

サンプルプログラムが実行されると、RL78/G22 は DA16200 / DA16600 モジュールをリセットします。DA16200 / DA16600 のリセット後、Operation Modo の設定で、Soft-AP モードの設定をます。その後、モジュールのリスタートを行います。Network の設定で、IP の設定、DHCP の設定、TCP サーバでの動作設定、ローカルポートの設定を行います。ターミナルソフトの実行ログを下記に示します。

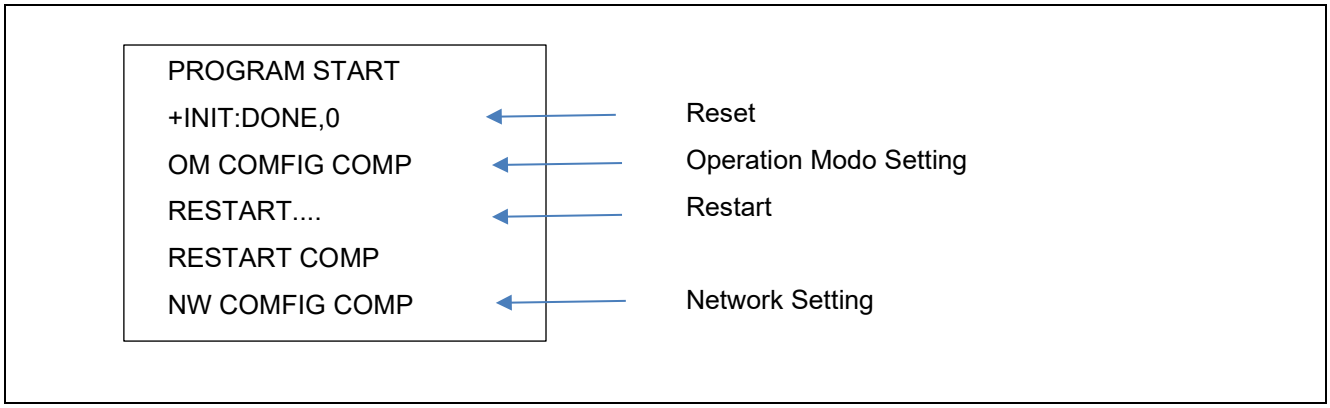

図 2-11 RL78/G22 FPB のログ表示(ターミナルソフト) 設定

動作設定後の動作例を示します。

TCP Client は、TCP Client - Apps on Google Play 等の TCP Client ソフトウェアを使用します。スマートフォンの例は TCP Client - Apps on Google Play で説明します。

（1） スマートフォンから Wi-Fi 接続を行います。RL78/G22 FPB に接続のターミナルソフトでは” URC：+WFCST:XX:XX:XX:XX:XX:XX ”を表示します。

（2） スマートフォンで TCP Address を”10.0.0.1”、Port を”10194”に設定をして、TCP の接続をします。TCP の接続をすると、RL78/G22 FPB に接続のターミナルソフトでは“TCP connected”を表示します。また、TCP 接続の間、RL78/G22 FPB の LED1 を点灯します。

（3） スマートフォンから”on\n”を送ると、RL78/G22 FPB に接続のターミナルソフトでは“LED on”を表示します。また、RL78/G22 FPB の LED2 を点灯します。

（4） スマートフォンから”off\n”を送ると、RL78/G22 FPB に接続のターミナルソフトでは“LED off”を表示します。また、RL78/G22 FPB の LED2 を消灯します。

（5） RL78/G22 FPB のスイッチ(SW)を押すと、” Switch: X”(X はインクリメント)を RL78/G22 FPB からを送ります。スマートフォンにデータを表示します。

（6） スマートフォンで TCP 切断をします。TCP の切断をすると、RL78/G22 FPB に接続のターミナルソフトでは“TCP disconnected”を表示します。また、TCP 接続の間、RL78/G22 FPB の LED1 を消灯します。

（7） スマートフォンから Wi-Fi 切断を行います。RL78/G22 FPB に接続のターミナルソフトでは” URC：+WFDST:XX:XX:XX:XX:XX:XX ”を表示します。

動作例のターミナルソフトの実行ログを下記に動作例の項番で示します。

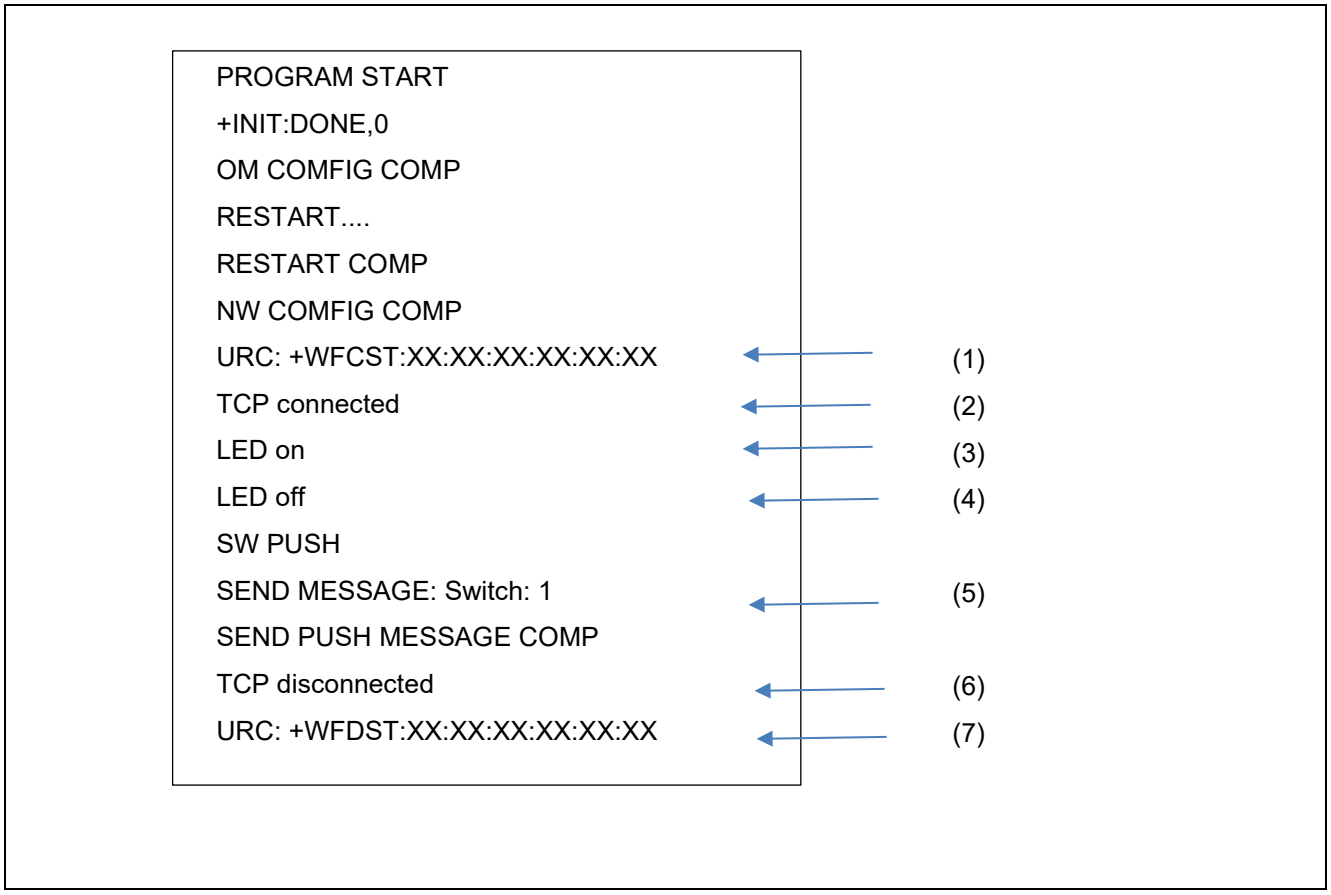

図 2-12 RL78/G22 FPB のログ表示(ターミナルソフト) TCP 動作

動作例のスマートフォンの動作を下記に動作例の項番で示します。

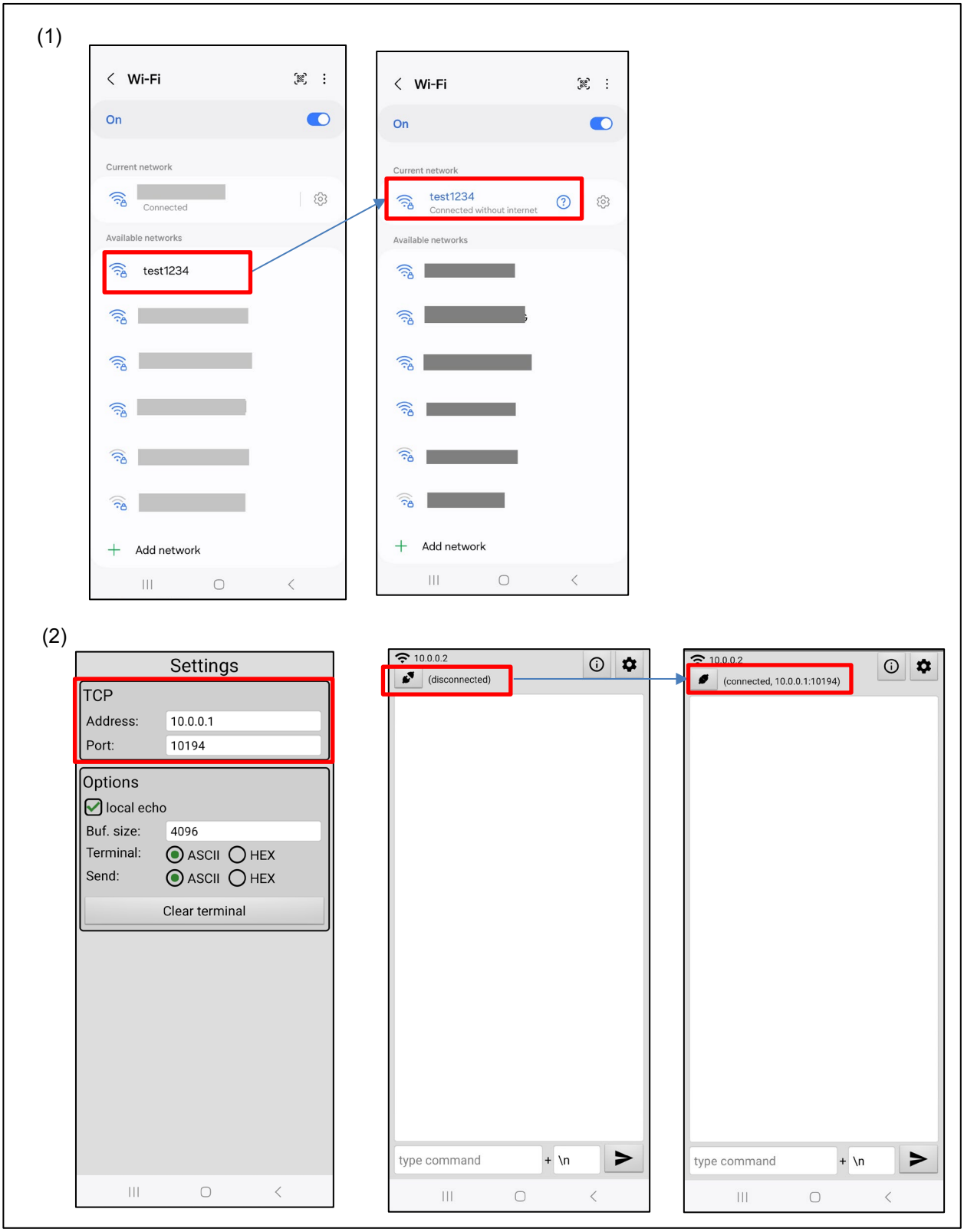

図 2-13　スマートフォンの表示(1)

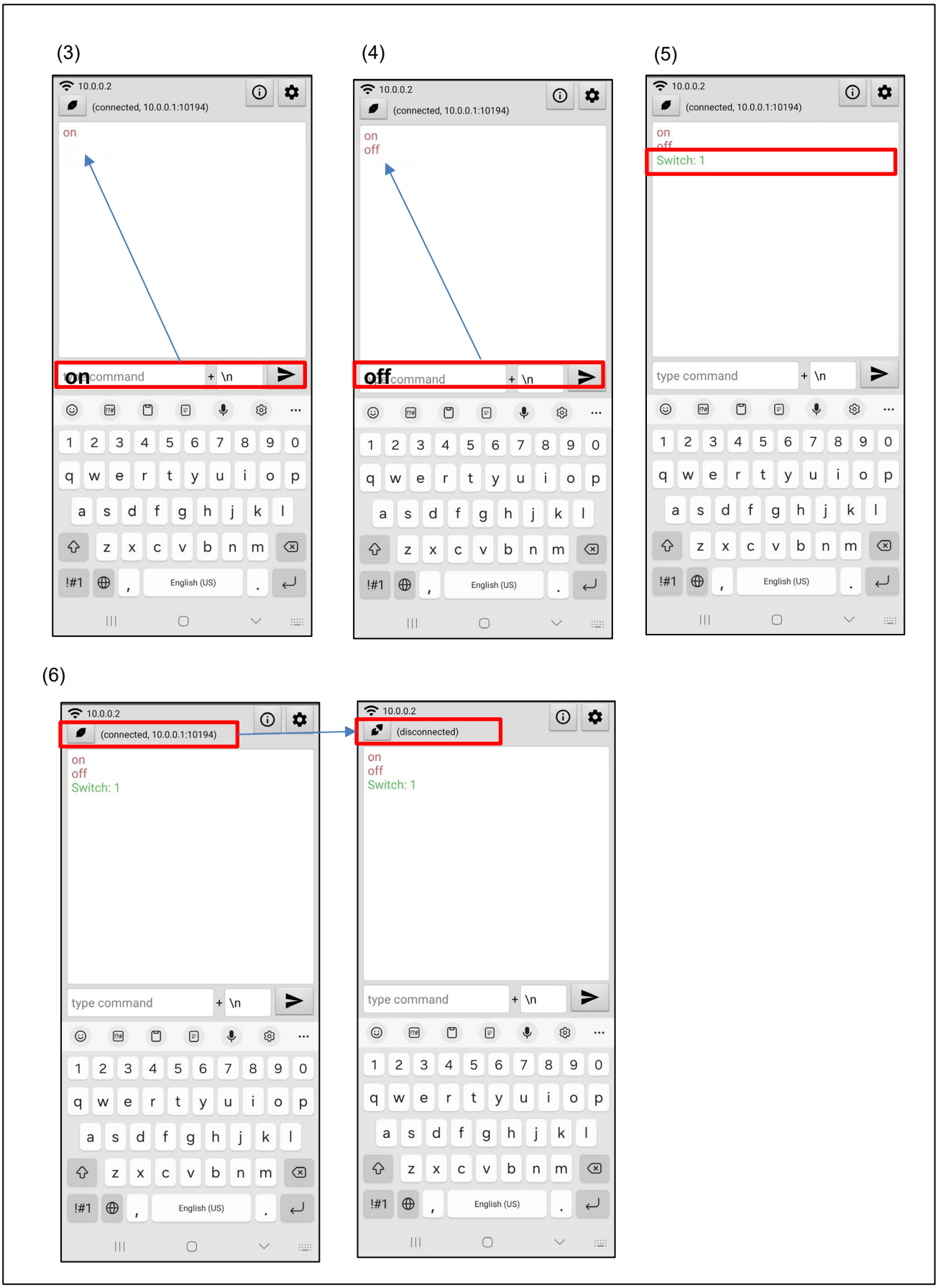

図 2-14　スマートフォンの表示(2)

動作例での RL78/G22 FPB の LED とスイッチ(SW)の位置を示します。

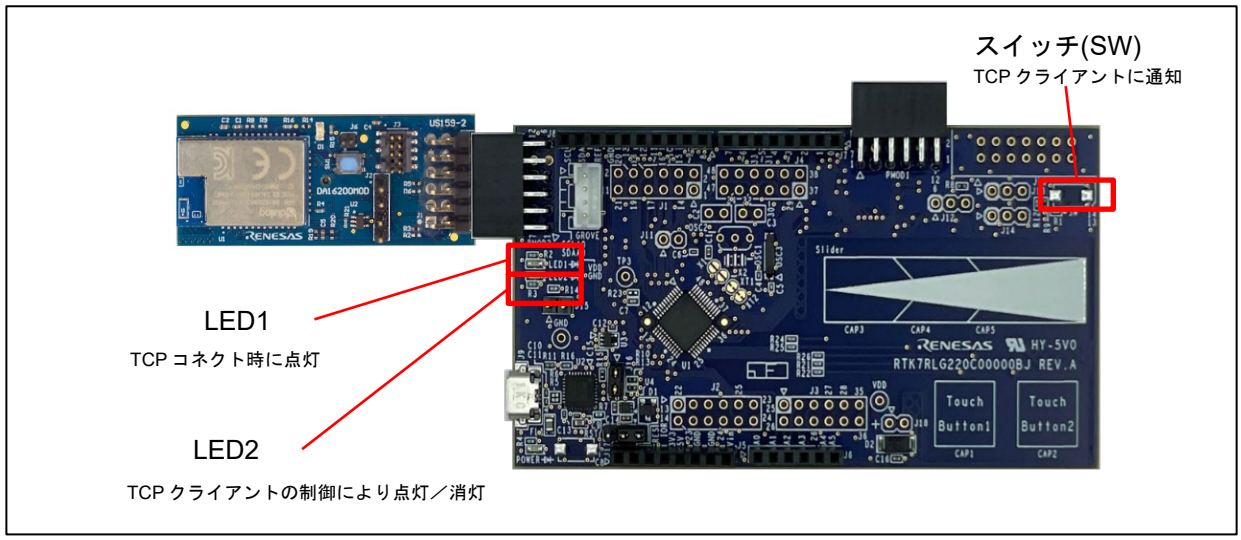

図 2-15　RL78/G22 FPB の LED／スイッチ

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 DA16200 / DA16600 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 TCP 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、DA16200 / DA16600 への AT コマンド送信と DA16200 / DA16600 からのレスポンスの受信で使用します。
- 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用します。
- 割り込みコントローラは、DA16200 / DA16600 の割込みには使用していません。ただし、スイッチ(SW)を押されたことを通知する割り込みで使用します。

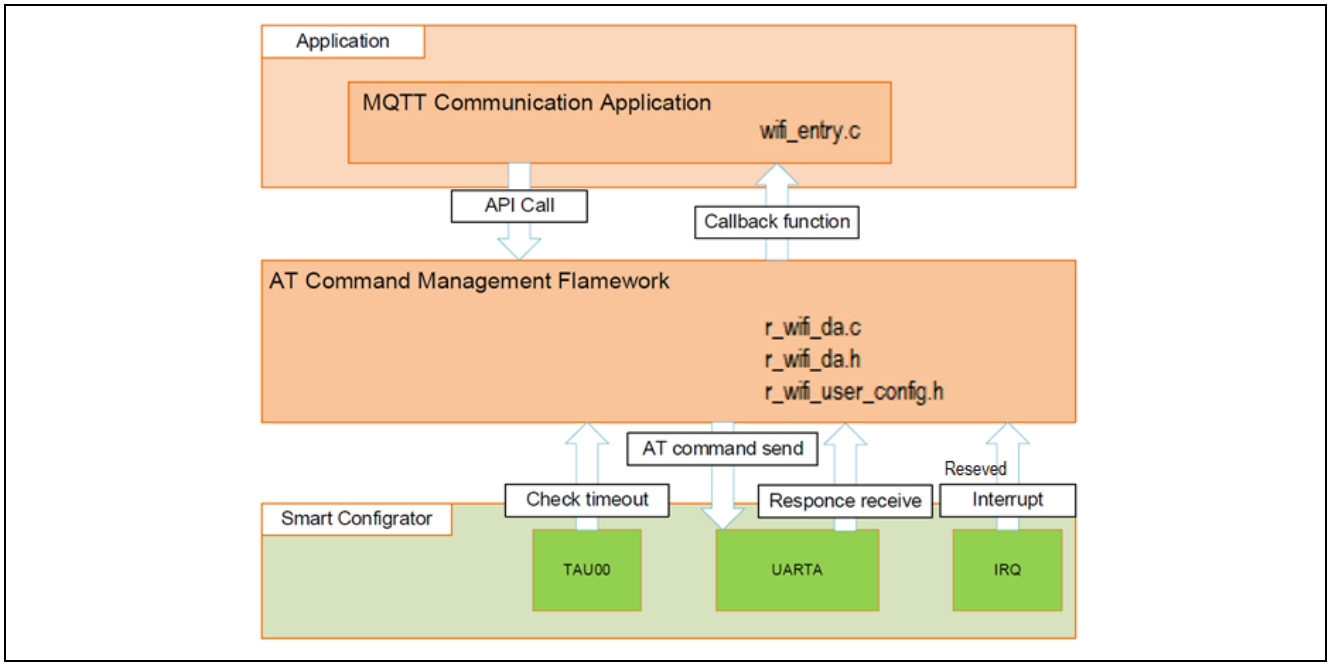

図 3-1 AT コマンドマネジメントフレームワーク

AT コマンドマネジメントフレームワークでは DA16200 / DA16600 の Wi-Fi 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 をコールしています。アプリケーションで AT コマンド API をコールする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 DA16200 / DA16600 へ送信されます。

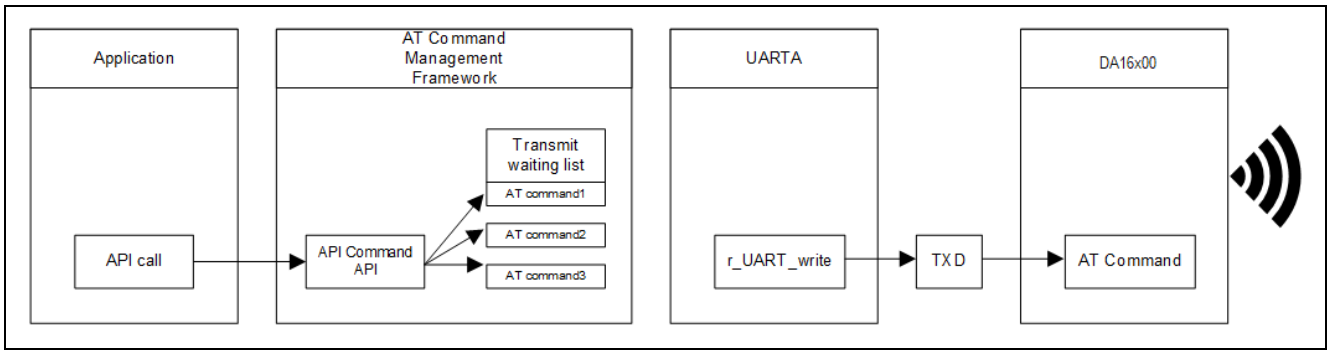

図 3-2 AT コマンド API の呼出し

DA16200 / DA16600 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_WIFI_Execute 関数で解析されます。実行結果が正しい場合、R_WIFI_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。

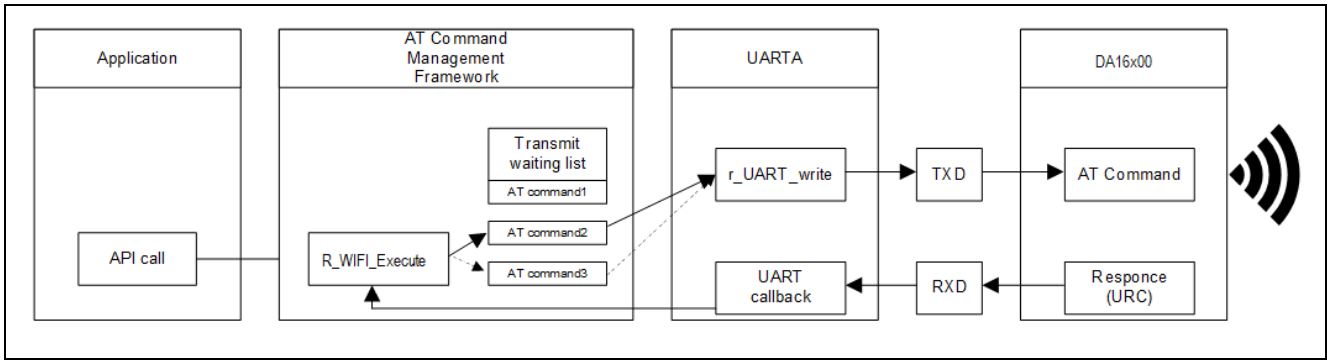

図 3-3　レスポンスの受信と AT コマンドの送信

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、DA16200 / DA16600 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_WIFI_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。

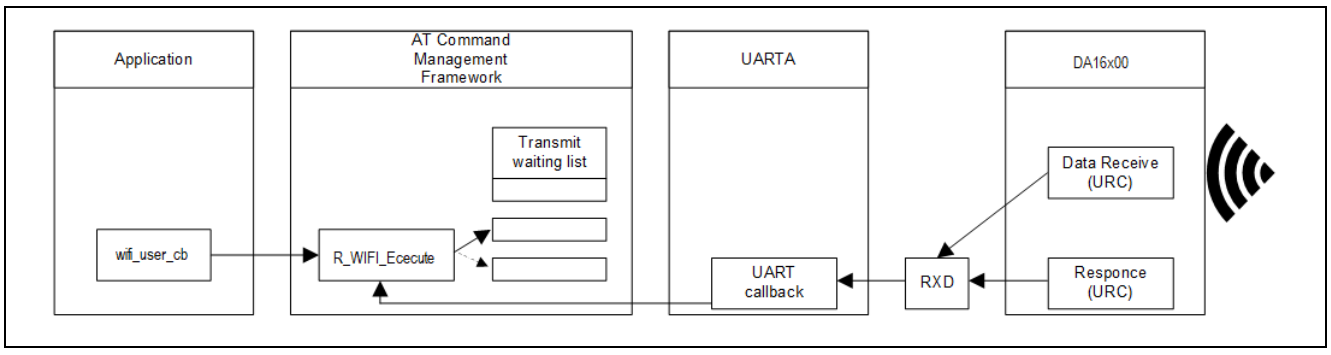

図 3-4　コールバック関数での通知

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へ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_wifi_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_wifi_da.c, r_wifi_da.h, r_wifi_user_config.c)は変更不要です。

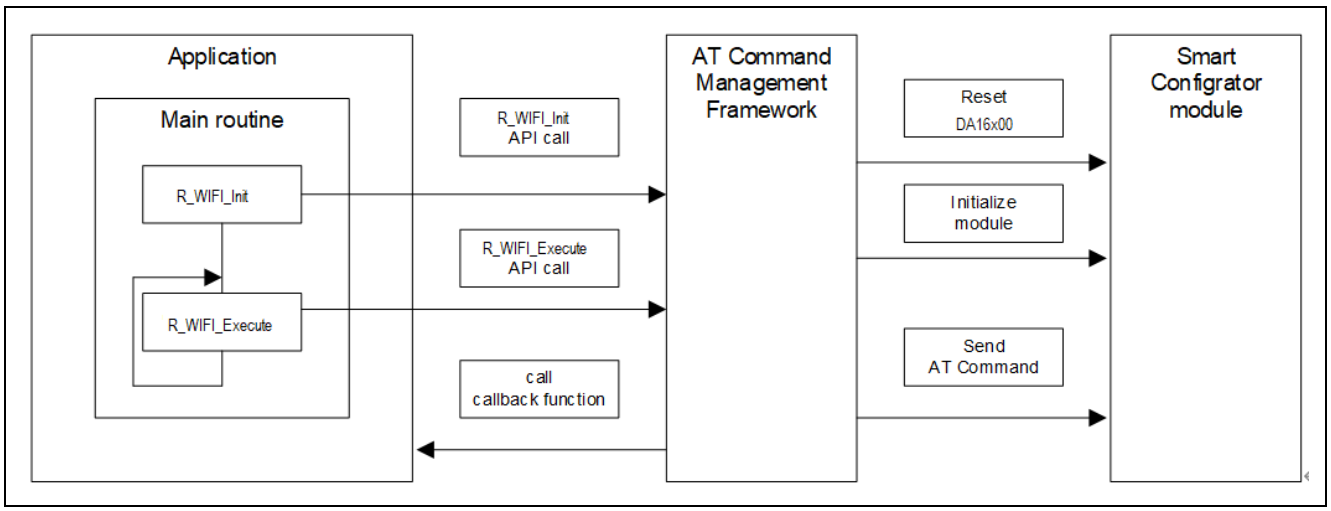

図 3-5 マネジメント API

3.2.1.1 R_WIFI_Init

関数名	R_WIFI_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	wifi_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	WIFI_SUCCESS (0x0000)	成功
	WIFI_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 DA16200 / DA16600 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、DA16200 / DA16600 をリセットするための AT コマンドが送信されます。 DA16200 と DA16600 で初期化時間が異なるため、異なるウエイト時間が設定しています。 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。 WIFI_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_WIFI_Execute

関数名	R_WIFI_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。 DA16200 / DA16600 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 DA16200 / DA16600 からのレスポンスに応じて順次 DA16200 / DA16600 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 DA16200 / DA16600 で TCP 通信を行うために必要な API を実装しています。TCP 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。

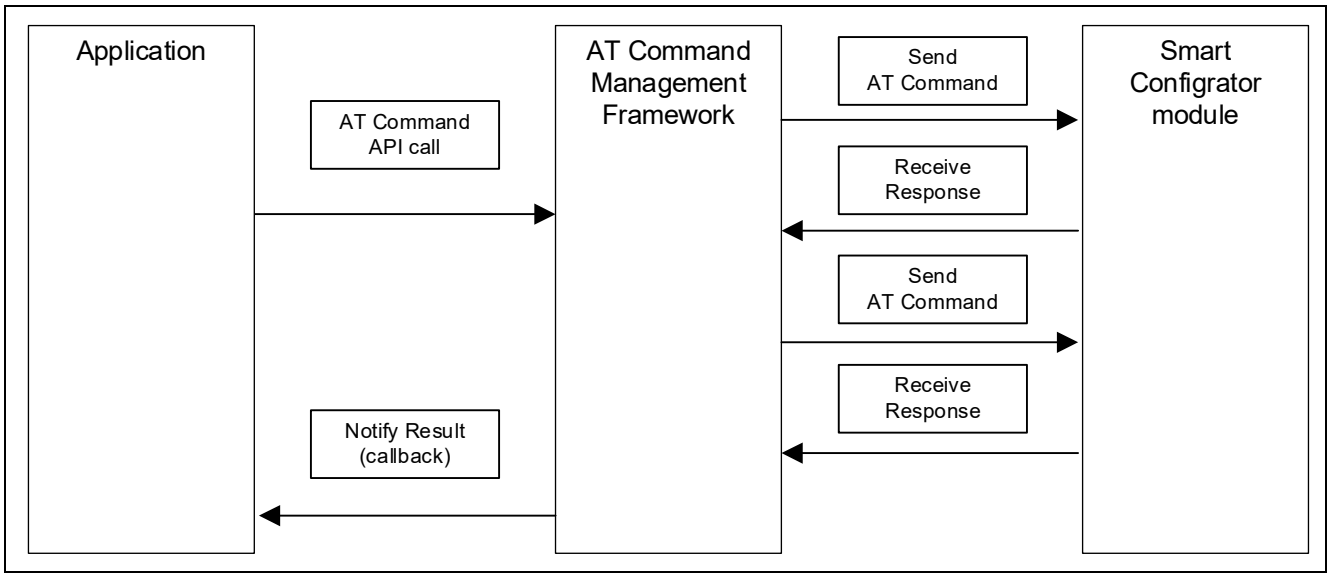

図 3-6 AT コマンド API

3.2.2.1 R_WIFI_OM_Config

関数名	R_WIFI_OM_Config	
機能概要	オペレーションモードを設定する	
引数	uint8_t * p_ap_ssid (IN)	AP mode の SSID 例：”test1234”
	uint8_t * p_ap_password (IN)	AP mode の Password 例：” 12345678”
	uint8_t * p_ap_country (IN)	AP mode の国名 例："JP"
	uint8_t * p_ap_ch (IN)	Operating channel 例：”0”
	uint8_t * p_ap_sec (IN)	Security protocol 例：” 3” (WPA2)
	uint8_t * p_ap_enc (IN)	Encryption mode 例：”1” (AES)
返り値	WIFI_SUCCESS (0x0000)	成功
	WIFI_ERR_POINTER_NULL (0x0001)	引数のポインタが NULL になっている
	WIFI_ERR_IN_PROCESS (0x0002)	他の WIFI 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+TMRFNOINIT=0 AT+WFMODE=1 AT+WFSAP=[p_ap_ssid], [p_ap_sec], [p_ap_enc], [p_ap_password], [p_ap_ch], [p_ap_country] 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。 WIFI_API_OM_CONFIG (0x01)	

3.2.2.2 R_WIFI_Restart

関数名	R_WIFI_Restart	
機能概要	モジュールをリスタートする	
引数	void	なし
返り値	WIFI_SUCCESS (0x0000)	成功
詳細機能	以下の AT コマンドを送信します。 AT+RESTART 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。 WIFI_API_RESTART (0x02)	

3.2.2.3 R_WIFI_RestartWait

関数名	R_WIFI_RestartWait	
機能概要	モジュールのリスタート後の復帰待ちをする	
引数	void	なし
返り値	WIFI_SUCCESS (0x0000)	成功
詳細機能	AT+RESTART コマンドからの復帰のウエイトをする DA16200 と DA16600 で初期化時間が異なるため、異なるウエイト時間が設定しています。 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。 WIFI_API_RESTART_WAIT (0x03)	

3.2.2.4 R_WIFI_NW_Config

関数名	R_WIFI_NW_Config	
機能概要	ネットワークの設定をする	
引数	uint8_t * p_ip_addr (IN)	IP アドレス 例：” 10.0.0.1”
	uint8_t * p_ip_netmask (IN)	IP サブネットマスク 例：” 255.255.255.0”
	uint8_t * p_dhcp_staip (IN)	DHCP スタート IP 例："10.0.0.2"
	uint8_t * p_dhcp_endip (IN)	DHCP エンド IP 例：” 10.0.0.11”
	uint8_t * p_local_port (IN)	ローカルポート 例：”10194”
返り値	WIFI_SUCCESS (0x0000)	成功
	WIFI_ERR_IN_PROCESS (0x0002)	他の WIFI 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+NWIP=1, [p_ip_addr], [p_ip_netmask], [p_ip_addr] AT+NWDHS=1, [p_dhcp_staip], [p_dhcp_endip],1800 AT+TRTS=[p_local_port] AT+TRTRM=0 AT+TRTS=[p_local_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。 WIFI_API_NW_CONFIG (0x04)	

3.2.2.5 R_WIFI_PUSH_Message

関数名	R_WIFI_PUSH_Message	
機能概要	TCP クライアントにデータを送る	
引数	uint8_t* p_tcp_socket	データを送る TCP ソケット
	uint16_t length	送るデータのレングス
	uint8_t* p_message	データ
返り値	WIFI_SUCCESS (0x0000)	成功
	WIFI_ERR_POINTER_NULL (0x0001)	引数のポインタが NULL になっている
	WIFI_ERR_IN_PROCESS (0x0002)	他の WIFI API が実行中である
	WIFI_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	以下のデータを送信します。 <ESC>S0[ASCII(length)],[p_tcp_socket],[p_message] 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。 WIFI_API_PUSH_MESSAGE (0x05)	

3.3 コールバック関数

DA16200 / DA16600 へ AT コマンドを送るとレスポンスが返ってきます。また、DA16200 / DA16600 の状態が変化した場合は DA16200 / DA16600 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 DA16200 / DA16600 からそれらのデータを受信した後、R_WIFI_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_WIFI_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、DA16200 / DA16600 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 * wifi_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
WIFI_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。
WIFI_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。
WIFI_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。
WIFI_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。
WIFI_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。 (未使用)

表 3-2 api_id の値

定義名	値	対応する API
WIFI_API_NO_CURRENT_API	0x0000	なし
WIFI_API_OM_CONFIG	0x0001	R_WIFI_OM_Config
WIFI_API_RESTART	0x0002	R_WIFI_Restart
WIFI_API_RESTART_WAIT	0x0003	R_WIFI_RestartWait
WIFI_API_NW_CONFIG	0x0004	R_WIFI_NW_Config
WIFI_API_PUSH_MESSAGE	0x0005	R_WIFI_PUSH_Message
WIFI_API_INIT	0x00FF	R_WIFI_Init

コールバック関数は特定の状況で R_WIFI_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：wifi_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”WIFI_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。
    - 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”WIFI_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 wifi_user_cb(uint16_t event_type, uint16_t api_id, uint16_t data_len, uint8_t *
p_data)
{
    if(WIFI_EVENT_API_COMPLETE == event_type)
    {
        switch (api_id)
        {
              case WIFI_API_INIT:
            {
                /* Configure operation
                if (0 != data_len)
                {
                    /* URC after ATF */
                    sprintf((char *)s_sbuf, (char *)p_data);
                }
                else
                {
/* 省略 */
                }
                debug_printf(s_sbuf);
                R_WIFI_OM_Config(s_str_ap_ssid, s_str_ap_password,
                     s_str_ap_country, s_str_ap_ch, s_str_ap_sec, s_str_ap_enc);
                break;
            }

            case WIFI_API_OM_CONFIG:
            {
                /* Restart after configuration of operation mode complete */
                sprintf((char *)s_sbuf, "OM COMFIG COMP\n");
                debug_printf(s_sbuf);
                R_WIFI_Restart();
                break;
            }
/* 省略 */
        }
```

WIFI_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

AT コマンドの実行結果として URC を受信する場合、
URC データを受信する場合 p_data に登録されている

図 3-7 WIFI_EVENT_API_COMPLETE のイベント通知

- DA16200 / DA16600 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”WIFI_EVENT_ERROR”が設定されます。

```
void wifi_user_cb(uint16_t event_type, uint16_t api_id, uint16_t data_len, uint8_t *
p_data)
{

/* 省略 */

    if(WIFI_EVENT_ERROR == event_type)
    {
        sprintf(sbuf,"ERROR RESPONSE\n");
        debug_printf(sbuf);
        r_wifi_delay_devspe(5);
     }

/* 省略 */
```

WIFI_EVENT_ERROR のイベント通知

図 3-8 WIFI_EVENT_ERROR のイベント通知

- DA16200 / DA16600 に送信した AT コマンドがタイムアウトした場合：
  - “event_type”に”WIFI_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 DA16200 / DA16600 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 wifi_user_cb(uint16_t event_type, uint16_t api_id, uint16_t data_len, uint8_t *
p_data)
{
/* 省略 */
    if(WIFI_EVENT_TIMEOUT_ERROR == event_type)
    {
        /* Set flag to initialize in mai       n */
        s_reinitate_flag = 1;
    }
}
/* 省略 */

void wifi_entry(void)
{
/* 省略 */
       if (1 == s_reinitate_flag)
       {
           /* Initialize when timeout oc      */
           R_WIFI_Init(wifi_user_cb);
           s_reinitate_flag = 0;
       }
```

WIFI_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 WIFI_EVENT_TIMEOUT_ERROR のイベント通知

- DA16200 / DA16600 から URC を受信した場合：
  - “event_type”に”WIFI_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” WIFI_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 wifi_user_cb(uint16_t event_type, uint16_t api_id, uint16_t data_len, uint8_t *
p_data)
{
/* 省略 */

   if (WIFI_EVENT_RCVURC == event_type)
    {
        const uint8_t p_str_onconnect[] = "+TRCTS";
        const uint8_t p_str_onmessage[] = "+TRDTS";
        const uint8_t p_str_ondisconn[] = "+TRXTS";
        uint8_t i;

        /* Check TCP data  */
        if (0 == memcmp(p_data, p_str_onmessage, ((sizeof(p_str_onmessage)) - 1)))
        {
            if(0 == memcmp(p_data + 26,
            {
                sprintf((char *)s_sbuf,
                debug_printf(s_sbuf);
                PIN_WRITE(LED2) = LED_ON;
            }

            if(0 == memcmp(p_data + 26, "off", 3))
            {
                sprintf((char *)s_sbuf, "LED off\n");
                debug_printf(s_sbuf);
                PIN_WRITE(LED2) = LED_OFF;
            }
        }
```

WIFI_EVENT_RCVURC のイベント通知

受信 URC の確認
“+TRDTS”の場合、処理を実行

URC のパラメータを解析

図 3-10 WIFI_EVENT_RCVURC のイベント通知

3.4 ユーザ固有値の設定

本サンプルプログラム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_wifi_da.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 DA16200 / DA16600 の各端子に接続する RL78/G22 の端子操作関数を示します。サンプルプログラムは RTS/CTS および INT 端子は利用していません。したがって、RTS/CTS および INT 端子に関連する関数は未使用です。このためホスト 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 DA16200 / DA16600 の端子操作関数

関数名	説明	使用端子
void r_wifi_reset_low_devspe(void)	DA16200 / DA16600 の RESET 端子を Low にします。	P17
void r_wifi_reset_high_devspe(void)	DA16200 / DA16600 の RESET 端子を High にします。	P17
void r_wifi_rts_low_devspe(void) (未使用)	DA16200 / DA16600 の RTS 端子を Low にします。	P70
void r_wifi_rts_high_devspe(void) (未使用)	DA16200 / DA16600 の RTS 端子を High にします。	P70
uint8_t r_wifi_cts_read_devspe(void) (未使用)	DA16200 / DA16600 の CTS 端子をリードします。	P50
uint8_t r_wifi_int_read_devspe(void) (未使用)	DA16200 / DA16600 の INT 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 f_{OCO} 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_wifi_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
WIFI_ATC_STR_SIZE	60	AT コマンドの文字列の最大長。
WIFI_DATA_STR_SIZE	60	DA16200 / DA16600 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。
WIFI_ATC_LIST_SIZE	6	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 DA16200 / DA16600 の間の UART 通信を行います。

RL78/G22 から DA16200 / DA16600 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。

DA16200 / DA16600 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_WIFI_Execute 関数のコールするたびに 1 文字ずつ処理されます。

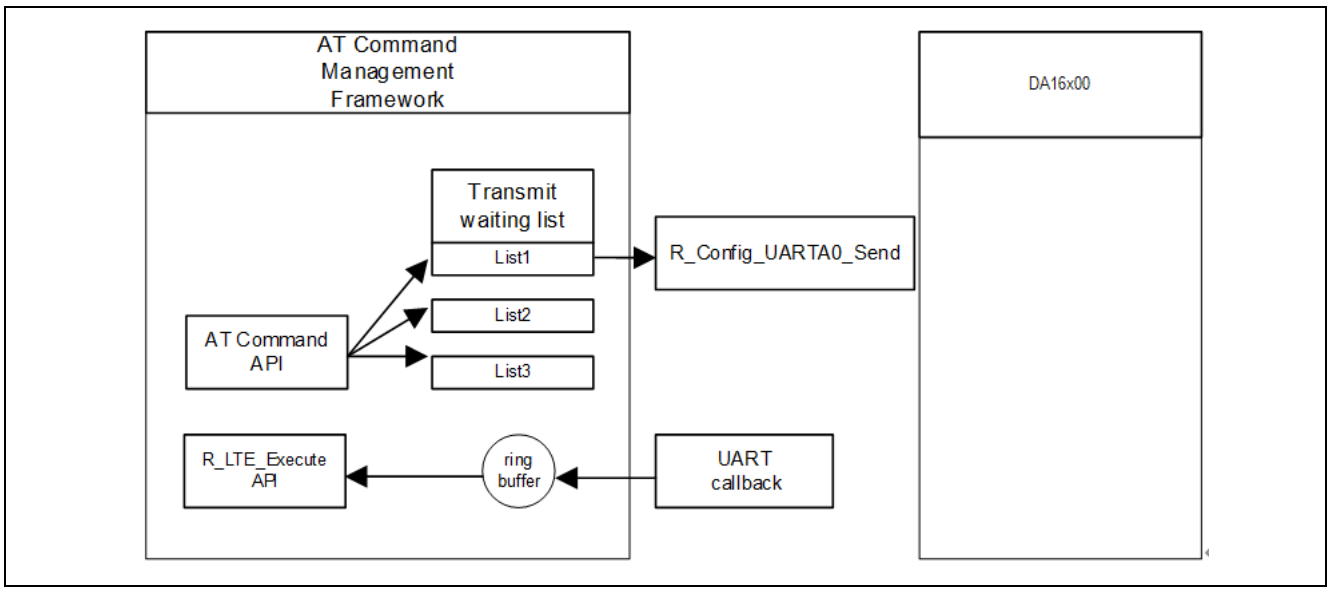

図 3-11 UARTA モジュールの利用

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_wifi_da.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_wifi_da.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_WIFI_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_wifi_da.c)」を変更してください。

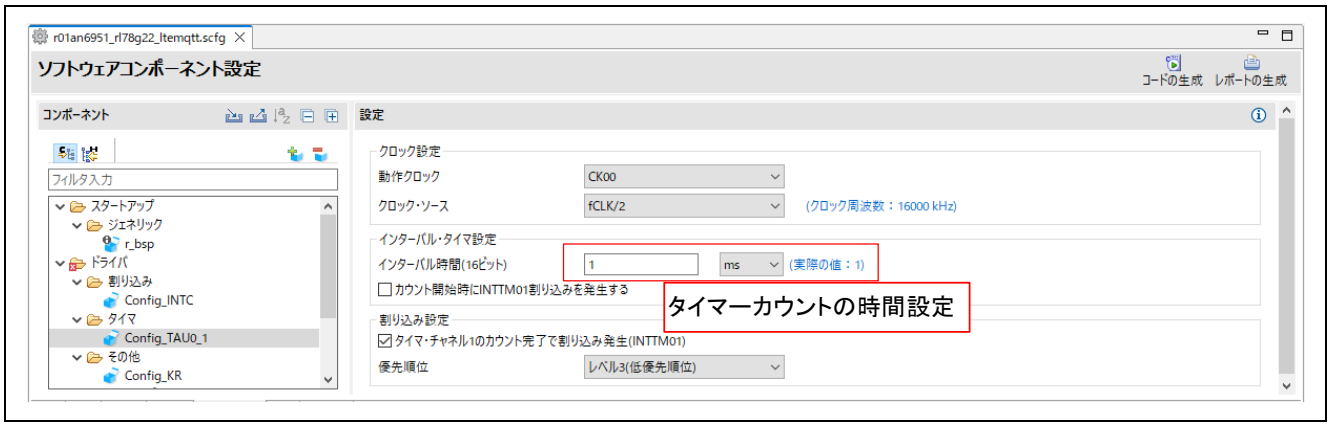

図 3-12 TAU モジュールの設定

3.5.3 割り込み機能

割り込み機能を使用して、DA16200 / DA16600 からの INT 信号の割り込みは INTP2 に接続されています。この INT 割り込みはサンプルプログラムでは使用していません。

また、スイッチ(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。

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 DA16200 / DA16600 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プログラム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。本サンプルプログラム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_wifi_da.c
  - r_wifi_da.h
  - r_wifi_user_config.h
- ベアメタルアプリケーションプログラム
  - wifi_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 DA16200 / DA16600 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_WIFI_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 DA16200 / DA16600 のハードウェアリセットなどが実行されます。DA16200 / DA16600 は初期化が完了後、AT コマンドを受け付けられるようになります。その後 ATF を送信してファクトリリセットを実行しています。
  +INIT:DONE,0 が返ってくるのを待ち、引数に指定したコールバック関数に API_ID = ”WIFI_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_WIFI_Execute
  DA16200 / DA16600 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 wifi_entry(void)
{
    sprintf(sbuf,"PROGRAM START\n");
    debug_printf(sbuf);
    wifi_user_sw_enable_devspe();
    PIN_WRITE(LED1) = LED_OFF;
    PIN_WRITE(LED2) = LED_OFF;

    /* SW  INT interrupt start */
    R_Config_INTC_INTP0_Start();
    /* Initialize framework-based program and register callback function */
    R_WIFI_Init(lte_user_cb);

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

/* 省略 */
   }
  }
```

R_WIFI_Init を最初にコール

R_WIFI_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 (wifi_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 DA16200 / DA16600 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、本サンプルプログラム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 wifi_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 WIFI_API_INIT:
            {
                /* Configure operation mode after initiation complete */
                if (0 != data_len)
                {
                    /* URC after ATF */
                    sprintf((char *)s_sbuf, (char *)p_data);
                }
                else
                {
                    /* If URC is not received, this mes
                    sprintf((char *)s_sbuf, "INIT COMP\n
                }
                debug_printf(s_sbuf);
                R_WIFI_OM_Config(s_str_ap_ssid, s_str_ap_password,
                        s_str_ap_country, s_str_ap_ch, s_str_ap_sec, s_str_ap_enc);
                break;
            }

            case WIFI_API_OM_CONFIG:
            {
                /* Restart after configuration of operation mode complete */
                sprintf((char *)s_sbuf, "OM COMFIG COMP\n");
                debug_printf(s_sbuf);
                R_WIFI_Restart();
                break;
            }

/* 省略 */
```

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 (wifi_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プログラムに実装されている 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_wifi_da.h)にプロトタイプ宣言を追加します。

```
typedef enum
{
    WIFI_API_NO_CURRENT_API = 0,
    WIFI_API_OM_CONFIG,
    WIFI_API_RESTART,
    WIFI_API_RESTART_WAIT,
    WIFI_API_NW_CONFIG,
    WIFI_API_PUSH_MESSAGE,
    WIFI_API_INIT = 0xFF,
} e_lte_api_id_t;
```

図 4-3 本サンプルプログラムの API ID (r_wifi_da.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_wifi_da.c)に実装します。本サンプルプログラムの 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”s_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”WIFI_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、”s_process_api”に API_ID を登録します。

```
e_wifi_err_t R_WIFI_OM_Config(uint8_t* p_ap_ssid, uint8_t* p_ap_password, uint8_t*
p_ap_country, uint8_t* p_ap_ch,
                              uint8_t* p_ap_sec, uint8_t* p_ap_enc)
{

    /* Check SSID/PASSWORD */
    if ((NULL == p_ap_ssid) || (NULL == p_ap_password))
    {
        return WIFI_ERR_POINTER_NULL;
    }


    if (WIFI_API_NO_CURRENT_API != s_process_api)
    {
        return WIFI_ERR_IN_PROCESS;
    }
    /* Clear ATC list and set processing API ID */
    r_wifi_clear_atc_list();
    s_process_api = WIFI_API_OM_CONFIG;

/* 省略 */
```

引数の確認

実行中の AT コマンド API の確認

実行中の AT コマンド API の登録

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

送信待機リストへ AT コマンドの登録

送信待機リスト”s_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”s_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_wifi_user_config.h)の”WIFI_ATC_STR_SIZE”を変更してください。
- 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 で指定した文字列ト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_WIFI_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”s_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_wifi_err_t R_WIFI_PUSH_Message(uint8_t* p_tcp_socket, uint16_t length, uint8_t*
p_message)
{
/* 省略 */

    /* Set AT command to ATC list */
    s_atc_list[0].atcommand_size = (uint16_t)snprintf((char *)s_atc_list[0].atcommand,
                        WIFI_ATC_STR_SIZE, "%s%s%s,%s,%s\r",
                        "\x1b","S0", s_length, p_tcp_socket, p_message);

    s_atc_list[0].comp_msg_size = (uint16_t)snprintf(
                                                (char_t *)s_atc_list[0].comp_msg,
                                                WIFI_ATC_STR_SIZE, "%s", "OK");  //
    s_atc_list[0].data_exist_flag = 1;
    if(s_atc_list[0].atcommand_size > WIFI_ATC_STR_SIZE)
    {
        r_wifi_clear_atc_list();
        return WIFI_ERR_DATASIZE_OVERFLOW;
    }

/* 省略 */
```

送信待機リストの先頭に以下を追加

atcommand =
<ESC>S0[ASCII(length)],[p_tcp_socket],[p_message]
comp_msg = “OK”
data_exist_flag = 1

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_WIFI_PUSH_Message の AT コマンドの登録 (r_wifi_da.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_WIFI_Execute 関数で行われます。

```
e_wifi_err_t R_WIFI_OM_Config(uint8_t* p_ap_ssid, uint8_t* p_ap_password, uint8_t*
p_ap_country, uint8_t* p_ap_ch, uint8_t* p_ap_sec, uint8_t* p_ap_enc)
{

/* 省略 */

    /* Send first AT command from ATC list */
    ryz_wifi_transmit_atc_list(WIFI_TRANSMIT_ATCOMMAND);

      return WIFI_SUCCESS;
```

送信待機リストの先頭の AT コマンドを送信

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

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、参考情報の扱いでお願いします。

また、UM-WI-003 DA16200 DA16600 Host Interface and AT Command User Manual (renesas.com)の説明も参照してください。

UART 通信及び DA16200 / DA16600 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
DA16200 / DA16600 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 "WIFI_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_WIFI_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 "WIFI_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 DA16200 / DA16600 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 "WIFI_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_WIFI_Init 関数を使用して初期化することが推奨される。
MCU の送信と DA16200 / DA16600 の送信タイミングが重なり、DA16200 / DA16600 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 "WIFI_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_WIFI_Init 関数を使用して初期化することが推奨される。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2024/ 4/ 8	ー	第一版

製品ご使用上の注意事項

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

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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

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

(Rev.5.0-1 2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

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

www.renesas.com/contact/

商標について

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