RX ファミリ

SX-ULPGN-2000 Wi-Fi モジュール制御モジュール

Firmware Integration Technology

R01AN4664JJ0116
Rev.1.16
2022.5.27

要旨

本アプリケーションノートは、Firmware Integration Technology (FIT)に準拠した SX-ULPGN-2000 Wi-Fi モジュール制御モジュールの使用方法について説明します。

以降、SX-ULPGN-2000 Wi-Fi モジュール制御モジュールのソフトウェアを総じて”SX-ULPGN Wi-Fi FIT モジュール”、または”本 FIT モジュール”と称します。

本 FIT モジュールがサポートしている Wi-Fi モジュールは以下です。

Silex 社製 ULPGN（SX-ULPGN-2000）

以降、Silex 社製 ULPGN（SX-ULPGN-2000）を”Wi-Fi モジュール”と称します。

なお、本 FIT モジュールでは、RTOS の機能を使用しています。RTOS と合わせて使用してください。また、MCU のシリアル通信機能を制御するデバイスドライバは含まないため、別途、以下のアップリケーションノートを取得してください。

- SCI モジュール Firmware Integration Technology
  R01AN1815

対象デバイス

RX65N グループ

RX671 グループ

RX72N グループ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

関連ドキュメント

Firmware Integration Technology ユーザーズマニュアル(R01AN1833)
ボードサポートパッケージモジュール Firmware Integration Technology(R01AN1685)
e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)
CS+に組み込む方法 Firmware Integration Technology (R01AN1826)
Renesas e^2 studio スマート・コンフィグレータ ユーザーガイド(R20AN0451)
RX ファミリ SCI モジュール Firmware Integration Technology (R01AN1815)
RX ファミリ バイト型キューバッファ（BYTEQ）モジュール Firmware Integration Technology (R01AN1683)

目次

1. 概要

1.1 SX-ULPGN Wi-Fi FIT モジュールとは

本 FIT モジュールは API として、プロジェクトに組み込んで使用します。本 FIT モジュールの組み込み方については、「2.10 FIT モジュールの追加方法」を参照してください。

1.2 SX-ULPGN Wi-Fi FIT モジュールの概要

本 FIT モジュールは SX-ULPGN の Transparent mode（1CH 通信モード）と Separate port mode（2CH 通信モード）をサポートします。

1.2.1 SX-ULPGN との接続

SX-ULPGN の接続例を示します。

図 1.1 1CH 通信モード、図 1.2 2CH 通信モード時の接続になります。

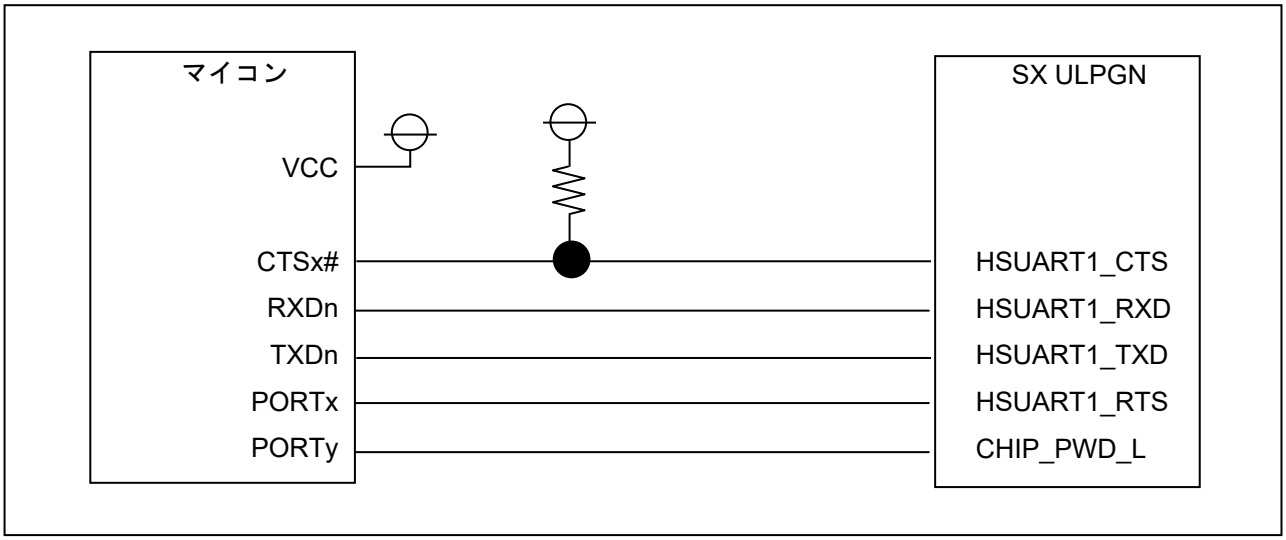

図 1.1　1CH 通信モード時の接続例

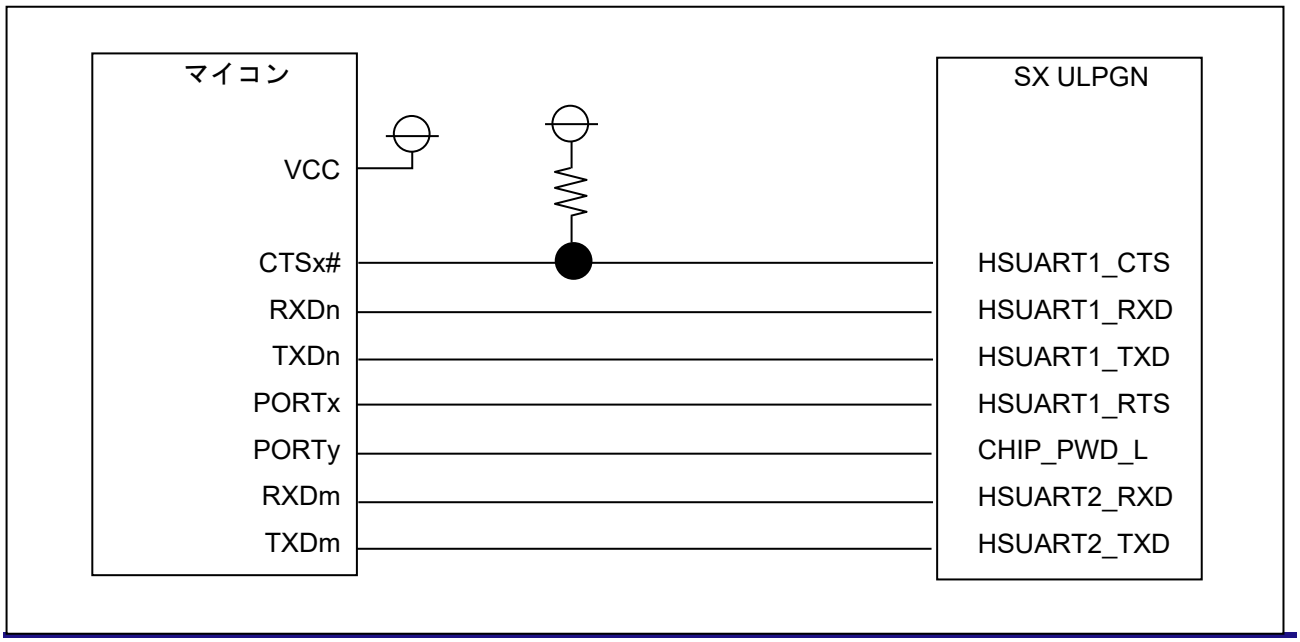

図 1.2　2CH 通信モード時の接続例

1.2.2　ソフトウェア構成

図 1.3 にソフトウェア構成を示します。

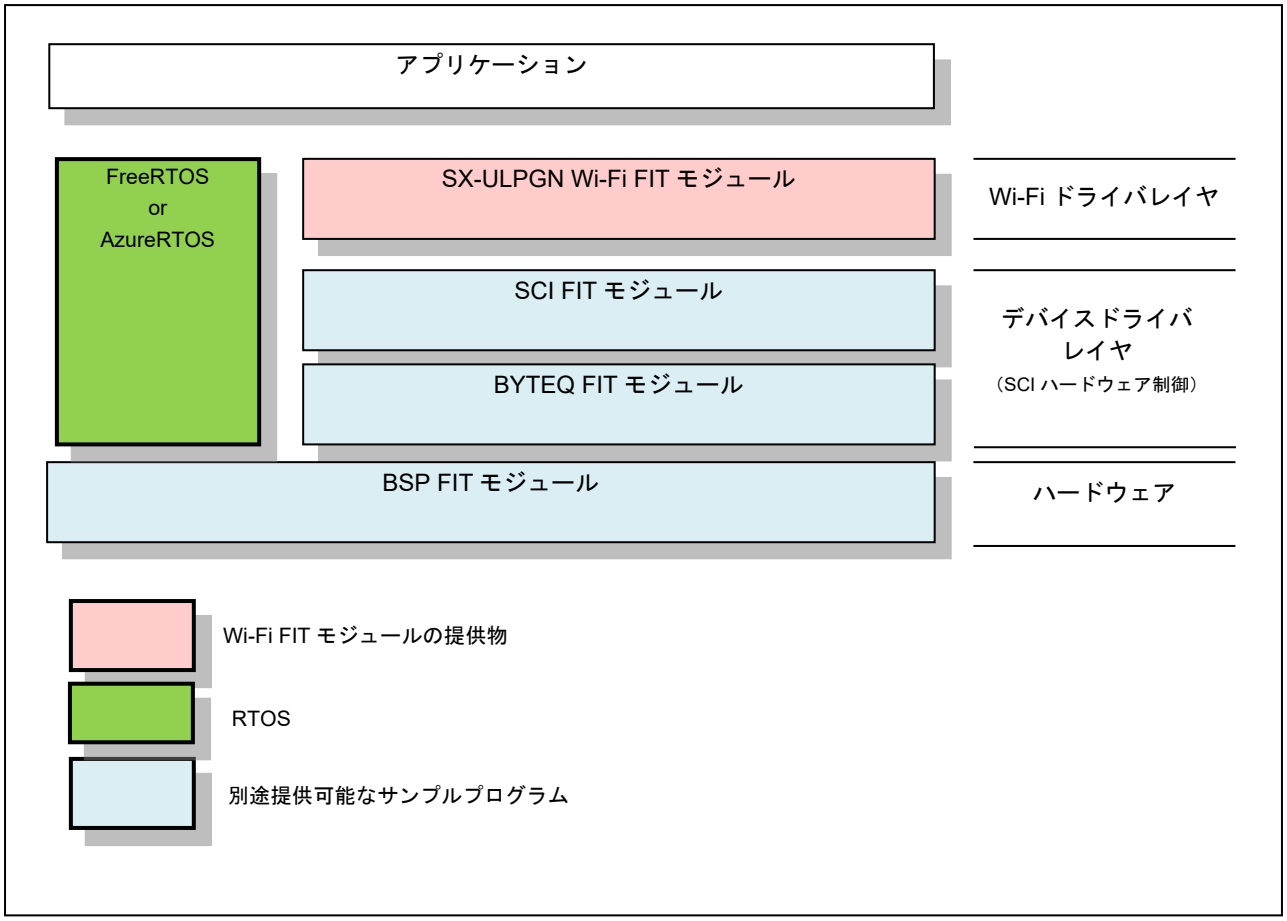

図 1.3　アプリケーション構成図

(1)　SX-ULPGN Wi-Fi FIT モジュール

本 FIT モジュールです。Wi-Fi モジュールを制御するために使用するソフトウェアです。

(2)　SCI FIT モジュール

Wi-Fi モジュールと MCU 間の通信を行います。サンプルプログラムが入手可能です。先頭ページの「関連ドキュメント」を参照し、入手してください。

(3)　周辺機能モジュール

タイマ制御とバッファ管理を行うソフトウェアです。サンプルプログラムが入手可能です。先頭ページの「関連ドキュメント」を参照し、入手してください。

(4)　RTOS

RTOS がシステム全体を管理します。本 FIT モジュールは FreeRTOS と AzureRTOS に対応します。

1.2.3　API の概要

表 1.1 に本 FIT モジュールに含まれる API 関数を示します。

表 1.1　API 関数一覧

関数	関数説明
R_WIFI_SX_ULPGN_Open()	Wi-Fi モジュールの初期化を行います。
R_WIFI_SX_ULPGN_Close()	Wi-Fi モジュールをクローズします。
R_WIFI_SX_ULPGN_SetDnsServerAddress()	DNS サーバアドレスを設定します。
R_WIFI_SX_ULPGN_Scan()	アクセスポイントの一覧を取得します。
R_WIFI_SX_ULPGN_Connect()	アクセスポイントに接続します。
R_WIFI_SX_ULPGN_Disconnect()	アクセスポイントを切断します。
R_WIFI_SX_ULPGN_IsConnected()	アクセスポイントの接続状態を取得します。
R_WIFI_SX_ULPGN_GetMACaddress()	Wi-Fi モジュールの MAC アドレスを取得します。
R_WIFI_SX_ULPGN_GetIPaddress()	Wi-Fi モジュールの IP アドレスを取得します。
R_WIFI_SX_ULPGN_CreateSocket()	ソケットを作成します。
R_WIFI_SX_ULPGN_ConnectSocketct()	ソケット通信を開始します。
R_WIFI_SX_ULPGN_SendSocket()	データ送信を行います。
R_WIFI_SX_ULPGN_ReceiveSocket()	データ受信を行います。
R_WIFI_SX_ULPGN_ShutdownSocket()	ソケット通信を終了します。
R_WIFI_SX_ULPGN_CloseSocket()	ソケットをクローズします。
R_WIFI_SX_ULPGN_GetTcpSocketStatus()	ソケット状態を取得します。
R_WIFI_SX_ULPGN_DnsQuery()	DNS クエリを実行します。
R_WIFI_SX_ULPGN_Ping()	指定した IP アドレスに Ping を送信します。
R_WIFI_SX_ULPGN_GetVersion()	本モジュールのバージョン情報を返します。
Wi-Fi モジュール SSL 機能使用に関する関数	
R_WIFI_SX_ULPGN_RequestTlsSocket()	作成済の TCP ソケットを SSL 通信用に割り当てます。
証明書格納に関する関数	
R_WIFI_SX_ULPGN_WriteServerCertificate()	Wi-Fi モジュールに証明書を書き込みます。
R_WIFI_SX_ULPGN_EraseServerCertificate()	Wi-Fi モジュールの証明書を消去します。
R_WIFI_SX_ULPGN_GetServerCertificate()	Wi-Fi モジュールの証明書情報を取得します。
R_WIFI_SX_ULPGN_EraseAllCertificate()	Wi-Fi モジュールの証明書を全て消去します。
R_WIFI_SX_ULPGN_SetCertificateProfile()	Wi-Fi モジュールの証明書とサーバ情報を紐づけます。

1.2.4　状態遷移図

図 1.4 に本 FIT モジュールの通信状態までの状態遷移図を示します。

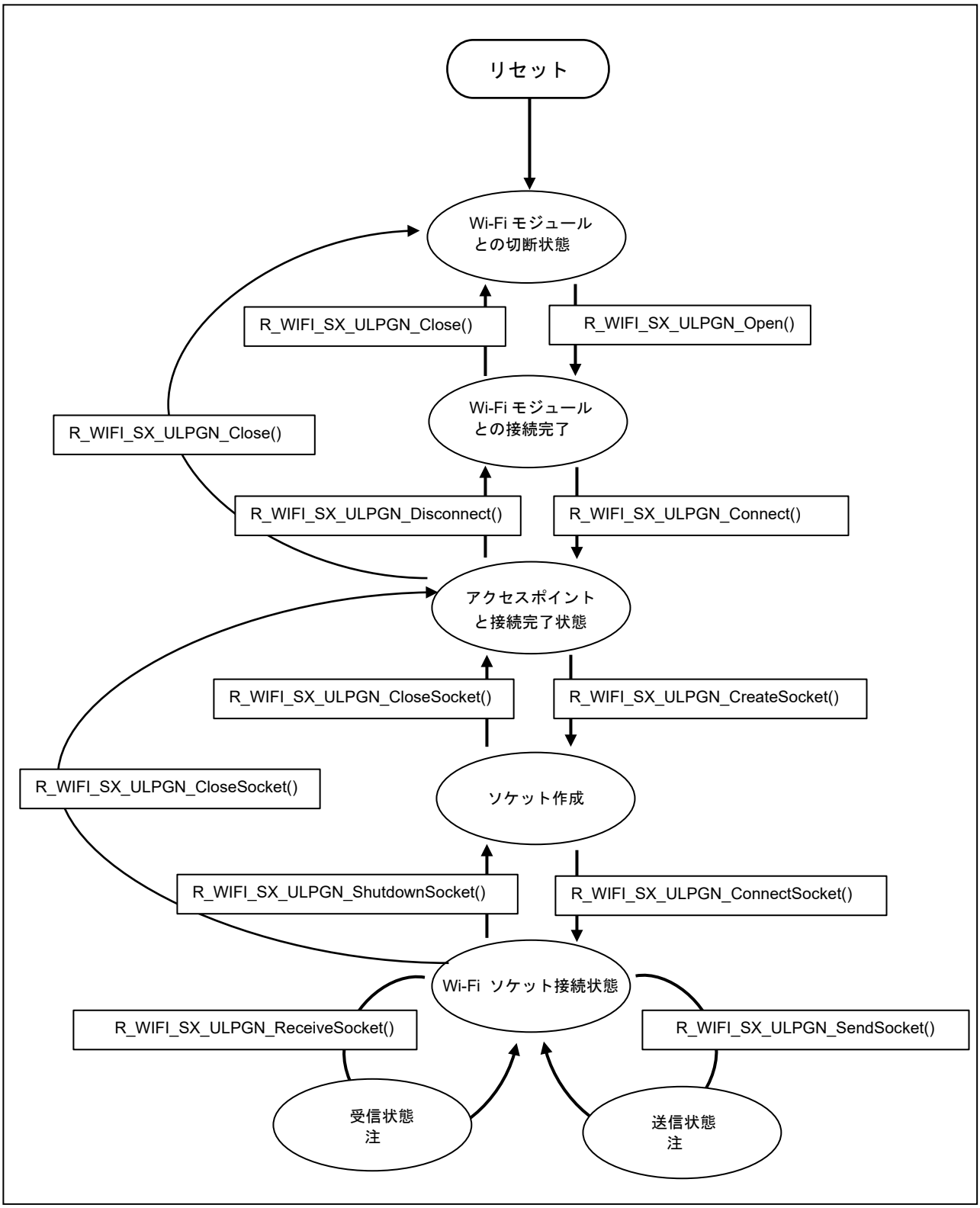

図 1.4 状態遷移図

2. API 情報

本 FIT モジュールは、下記条件で動作を確認しています。

2.1 ハードウェアの要求

ご使用になるマイコンが以下の機能をサポートしている必要があります。

- シリアル通信
- I/O ポート

2.2 ソフトウェアの要求

このドライバは以下のパッケージに依存しています。

- r_bsp
- r_sci_rx
- r_byteq_rx
- FreeRTOS
- AzureRTOS

2.3 サポートされているツールチェーン

本 FIT モジュールは 5.1 動作確認環境に示すツールチェーンで動作確認を行っています。

2.4 使用する割り込みベクタ

なし

2.5 ヘッダファイル

すべての API 呼び出しとそれをサポートするインタフェース定義は r_wifi_sx_ulpgn_if.h に記載しています。

2.6 整数型

Wi-Fi FIT モジュールは ANSI C99 を使用しています。これらの型は stdint.h で定義されています。

2.7　コンパイル時の設定

本 FIT モジュールのコンフィギュレーションオプションの設定は、r_wifi_sx_ulpgn_config.h と r_sci_rx_config.h で行います。

オプション名および設定値に関する説明を下表に示します。

表 2.1 Configuration options(r_wifi_sx_ulpgn_config.h)

Configuration options in r_wifi_sx_ulpgn_config.h	
WIFI_CFG_SCI_CHANNEL ※デフォルトは“0”	HSUART1 に割り当てる SCI チャネル番号を指定します。
WIFI_CFG_SCI_SECOND_CHANNEL ※デフォルトは“1”	HSUART2 に割り当てる SCI チャネル番号を指定します。 WIFI_CFG_SCI_CHANNEL と同じ番号を指定すると 1ch モードで動作します
WIFI_CFG_SCI_INTERRUPT_LEVEL ※デフォルトは“4”	SCI モジュールの割り込み優先度を設定します。システムに合わせて設定してください。
WIFI_CFG_SCI_PCLK_HZ ※デフォルトは“60000000”	SCI の PLCK クロックを Hz 単位で指定します。 システムに合わせて設定してください。 ※1ch モード時のみ有効です。
WIFI_CFG_SCI_BAUDRATE ※デフォルトは“460800”	SCI のボーレートを Hz 単位で指定します。
WIFI_CFG_SCI_USE_FLOW_CONTROL ※デフォルトは“1”	HSUART1 のハードウェアフロー制御の有効無効を設定します。 1：有効、0：無効
WIFI_CFG_RESET_PORT ※デフォルトは“D”	PWD_L 端子に割り当てる GPIO を設定します。 WIFI_CFG_RESET_PORT と WIFI_CFG_RESET_PIN を組み合わせてポート番号を指定します。 例）PD0 #define WIFI_CFG_RESET_PORT　D #define WIFI_CFG_RESET_PIN　0
WIFI_CFG_RESET_PIN ※デフォルトは“0”	
WIFI_CFG_RTS_PORT ※デフォルトは“2”	RTS 端子に割り当てる GPIO を設定します。 WIFI_CFG_RTS_PORT と WIFI_CFG_ RTS_PIN を組み合わせてポート番号を指定します。 例）P22 #define WIFI_CFG_RTS_PORT　2 #define WIFI_CFG_ RTS_PIN　2
WIFI_CFG_ RTS_PIN ※デフォルトは“2”	
WIFI_CFG_CREATABLE_SOCKETS ※デフォルトは“4”	作成可能なソケット数を設定します。最大数は 4 です。 システムに合わせて設定してください。 ※1ch モード時は、常に 1 となります。
WIFI_CFG_SOCKETS_RECEIVE_BUFFER_SIZE ※デフォルトは“8192”	ソケットの受信バッファサイズを設定します。メモリ使用量、データ受信量に合わせて設定してください。
WIFI_CFG_USE_CALLBACK_FUNCTION ※デフォルトは“0”	ユーザコールバック関数の有効無効を設定します。 1=有効、0=無効
WIFI_CFG_CALLBACK_FUNCTION_NAME ※デフォルトは“NULL”	ユーザコールバック関数名を登録します。 コールバック関数の実装方法は 4 章を参照してください。 WIFI_CFG_USE_CALLBACK_FUNCTION が 0 の場合、本項目は無効です。

表 2.2 Configuration options(r_sci_rx_config.h)

Configuration options in r_sci_rx_config.h	
#define SCI_CFG_CHx_INCLUDED ※1. CHx = CH0～CH12 ※2. 各デフォルト値は以下のとおり: CH0、CH2～CH12: 0、CH1: 1	チャネルごとに送受信バッファ、カウンタ、割り込み、その他のプログラム、RAM などのリソースを持ちます。このオプションを“1”に設定すると、そのチャンネルに関連したリソースが割り当てられます。
#define SCI_CFG_CHx_TX_BUFSIZ ※1. CHx = CH0～CH12 ※2. 各デフォルト値は(80)	チャネルごとの送信バッファサイズを指定します。 WIFI_CFG_SCI_CHANNEL で指定したチャネルに対応するバッファサイズを 2048 にしてください。
#define SCI_CFG_CHx_RX_BUFSIZ ※1. CHx = CH0～CH12 ※2. 各デフォルト値は(80)	チャネルごとの受信バッファサイズを指定します。 WIFI_CFG_SCI_CHANNEL で指定したチャネルに対応するバッファサイズを 2048 にしてください。
#define SCI_CFG_TEI_INCLUDED ※デフォルト値は″0″	シリアル送信の送信完了割り込みを有効にします。”1”を設定してください。

表 2.3 Configuration options(r_byteq_config.h)

Configuration options in r_byteq_config.h	
#define BYTEQ_CFG_MAX_CTRL_BLKS	WIFI_CFG_CREATABLE_SOCKETS で指定した値を加算してください。

表 2.4 Configuration options(r_bsp_config.h)

Configuration options in r_byteq_config.h	
#define BSP_CFG_RTOS_USED ※デフォルト値は″0″	リアルタイム OS の種類を選択します。 本 FIT モジュールを使用する場合は以下を設定してください。 FreeRTOS の場合：”1” AzureRTOS の場合：”5”

2.8　コードサイズ

本 FIT モジュールのコードサイズを下表に示します。

表 2.5 コードサイズ

ROM、RAM およびスタックのコードサイズ			
デバイス	分類	使用メモリ	備考
RX65N	ROM	8729 バイト	
	RAM	4759 バイト	ソケットのバッファ(8192*ソケット数)を除いたサイズです。
	最大使用スタックサイズ	214 バイト	多重割り込み禁止のため 1 チャネル使用時の最大値のみを記載しています。

2.9　戻り値

以下は API 関数が返すエラーコードです。戻り値の列挙型は、API 関数の宣言とともに r_wifi_sx_ulpgn_if.h に含まれます。

```
/*  WiFi API error code */
typedef enum
{
    WIFI_SUCCESS                = 0,  // success
    WIFI_ERR_PARAMETER          = -1, // invalid parameter
    WIFI_ERR_ALREADY_OPEN       = -2, // already WIFI module opened
    WIFI_ERR_NOT_OPEN           = -3, // WIFI module is not opened
    WIFI_ERR_SERIAL_OPEN        = -4, // serial open failed
    WIFI_ERR_MODULE_COM         = -5, // cannot communicate WiFi module
    WIFI_ERR_NOT_CONNECT        = -6, // not connect to access point
    WIFI_ERR_SOCKET_NUM         = -7, // no available sockets
    WIFI_ERR_SOCKET_CREATE      = -8, // create socket failed
    WIFI_ERR_CHANGE_SOCKET      = -9, // cannot change socket
    WIFI_ERR_SOCKET_CONNECT     = -10,// cannot connect socket
    WIFI_ERR_BYTEQ_OPEN         = -11,// cannot assigned BYTEQ
    WIFI_ERR_SOCKET_TIMEOUT     = -12,// socket timeout
    WIFI_ERR_TAKE_MUTEX         = -13 // cannot take mutex
} wifi_err_t;

/* Security type */
typedef enum
{
    WIFI_SECURITY_OPEN = 0,           // Open
    WIFI_SECURITY_WEP,                // WEP
    WIFI_SECURITY_WPA,                // WPA
    WIFI_SECURITY_WPA2,               // WPA2
    WIFI_SECURITY_UNDEFINED           // Undefined
} wifi_security_t;

/* Query current socket status */
typedef enum
{
    ULPGN_SOCKET_STATUS_CLOSED = 0,  // "CLOSED"
    ULPGN_SOCKET_STATUS_SOCKET,      // "SOCKET"
    ULPGN_SOCKET_STATUS_BOUND,       // "BOUND"
    ULPGN_SOCKET_STATUS_LISTEN,      // "LISTEN"
    ULPGN_SOCKET_STATUS_CONNECTED,   // "CONNECTED"
    ULPGN_SOCKET_STATUS_BROKEN,      // "BROKEN"
    ULPGN_SOCKET_STATUS_MAX          // Stopper
} sx_ulpgn_socket_status_t;

/* Error event for user callback */
typedef enum
{
    WIFI_EVENT_WIFI_REBOOT = 0,      // reboot WIFI
    WIFI_EVENT_WIFI_DISCONNECT,      // disconnected WIFI
    WIFI_EVENT_SERIAL_OVF_ERR,       // serial : overflow error
    WIFI_EVENT_SERIAL_FLM_ERR,       // serial : flaming error
    WIFI_EVENT_SERIAL_RXQ_OVF_ERR,   // serial : receiving queue overflow
    WIFI_EVENT_RCV_TASK_RXB_OVF_ERR, // receiving task : receive buffer overflow
    WIFI_EVENT_SOCKET_CLOSED,        // socket is closed
    WIFI_EVENT_SOCKET_RXQ_OVF_ERR    // socket : receiving queue overflow
} wifi_err_event_enum_t;
```

```
typedef struct
{
    wifi_err_event_enum_t event;     // Error event
    uint8_t socket_number;           // Socket number
} wifi_err_event_t;

/* AP scan result */
typedef struct
{
    uint8_t  ssid[33];               // SSID
    uint8_t  bssid[6];               // BSSID
    wifi_security_t security;        // kinds of security
    int8_t   channel;                // Channel
    int8_t   rssi;                   // RSSI
    uint8_t  hidden;                 // Hidden channel
} wifi_scan_result_t;

/* IP configurations */
typedef struct
{
    uint32_t ipaddress;              // IP address
    uint32_t subnetmask;             // subnet mask
    uint32_t gateway;                // gateway
} wifi_ip_configuration_t;

/* Certificate information */
typedef struct {
    uint8_t  num_of_files;           // certificate number
    struct {
        uint8_t  file_name[20];      // certificate file name
    } cert[10];
} wifi_certificate_infomation_t;
```

2.10　FIT モジュールの追加方法

本 FIT モジュールは、使用するプロジェクトごとに追加する必要があります。ルネサスでは、Smart Configurator を使用した(1)、(3)の追加方法を推奨しています。ただし、Smart Configurator は、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)、(4)の方法を使用してください。

(1) e² studio 上で Smart Configurator を使用して FIT モジュールを追加する場合
e² studio の Smart Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(2) e² studio 上で FIT Configurator を使用して FIT モジュールを追加する場合
e² studio の FIT Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加することができます。詳細は、アプリケーションノート「RX ファミリ e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)」を参照してください。

(3) CS+上で Smart Configurator を使用して FIT モジュールを追加する場合
CS+上で、スタンドアロン版 Smart Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(4) CS+上で FIT モジュールを追加する場合
CS+上で、手動でユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」を参照してください。

2.11　RTOS の使用要件

本 FIT モジュールでは RTOS の機能を使用しています。

2.12　制限事項

本 FIT モジュールには以下の制限があります。

- WIFI_ERR_SERIAL_OPEN が発生した際は R_WIFI_SX_ULPGN_Close()で Wi-Fi FIT モジュールをクローズしてください。
- R_WIFI_SX_ULPGN_WriteServerCertificate()でエラーが発生した際は R_WIFI_SX_ULPGN_EraseAllCertificate()で Wifi モジュールに格納された証明書をすべて削除した後、再度 R_WIFI_SX_ULPGN_WriteServerCertificate()で証明書を書き込んでください。

3. API 関数

3.1 R_WIFI_SX_ULPGN_Open()

本 FIT モジュールと Wi-Fi モジュールの初期化を行います。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_Open (
      void
)
```

Parameters

なし

Return Values

WIFI_SUCCESS	*正常終了*
WIFI_ERR_TAKE_MUTEX	*Mutex 取得失敗*
WIFI_ERR_SERIAL_OPEN	*シリアルの初期化失敗*
WIFI_ERR_SOCKET_BYTEQ	*BYTEQ の割り当てに失敗*
WIFI_ERR_ALREADY_OPEN	*Wi-Fi モジュールが初期化済*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

本 FIT モジュールと Wi-Fi モジュールを初期化します。

Reentrant

不可

Example

```
R_WIFI_SX_ULPGN_Open();
```

Special Notes:

WIFI_ERR_SERIAL_OPEN が発生した場合は R_WIFI_SX_ULPGN_Close()を実行してください。

3.2　R_WIFI_SX_ULPGN_Close()

Wi-Fi モジュールをクローズします。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_Close (
      void
)
```

Parameters

なし

Return Values

WIFI_SUCCESS　　*正常終了*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

Wi-Fi モジュールをクローズします。

アクセスポイント接続中に本 API を実行した場合はアクセスポイント切断と Wi-Fi モジュールをクローズします。

Reentrant

不可

Example

```
R_WIFI_SX_ULPGN_Open();
R_WIFI_SX_ULPGN_Close();
```

Special Notes:

なし

3.3　R_WIFI_SX_ULPGN_SetDnsServerAddress()

DNS サーバの IP アドレスを設定します。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_SetDnsServerAddress (
      uint32_t dns_address1,
      uint32_t dns_address2
)
```

Parameters

dns_address1	*DNS サーバ IP アドレス 1（0：本パラメータ無効）*
dns_address2	*DNS サーバ IP アドレス 2（0：本パラメータ無効）*

Return Values

WIFI_SUCCESS	*正常終了*
WIFI_ERR_NOT_OPEN	*Wi-Fi モジュールが初期化されていない*
WIFI_ERR_TAKE_MUTEX	*Mutex 取得失敗*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*

Description

dns_address1、dns_address2 で指定したアドレスを DNS サーバアドレスに設定します。

DHCP 無効設定で R_WIFI_SX_ULPGN_Connect()を実行した場合に本 API で設定した DNS サーバアドレスが適用されます。

本関数は R_WIFI_SX_ULPGN_Connect()を実行前に呼び出してください。

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Example

```
R_WIFI_SX_ULPGN_Open();
R_WIFI_SX_ULPGN_SetDnsServerAddress(0xc0a80105, 0xc0a80106);
```

Special Notes:

なし

3.4　R_WIFI_SX_ULPGN_Scan()

アクセスポイントをスキャンします。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_Scan (
      wifi_scan_result_t *ap_results,
      uint32_t max_networks,
      uint32_t *exist_ap_count
)
```

Parameters

**ap_results*	*スキャン結果を格納する構造体のポインタ*
max_networks	*ap_results に格納するアクセスポイントの最大数*
exist_ap_count	*存在するアクセスポイントの個数*

Return Values

WIFI_SUCCESS	*正常終了*
WIFI_ERR_PARAMETER	*不正な引数*
WIFI_ERR_NOT_OPEN	*Wi-Fi モジュールが初期化されていない*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

Wi-Fi モジュール周辺に存在するアクセスポイントをスキャンします。

スキャン結果は ap_results で指定された領域に最大 maxnetworks 分を格納します。

また検出したアクセスポイント数を exist_ap_count に格納します。

Example

```
wifi_scan_result_t scan_rslt[5];
uint32_t max_networks = 5;
uint32_t exist_ap_count;
uint32_t max_ap;

R_WIFI_SX_ULPGN_Scan(scan_rslt, max_networks, &exist_ap_count);
printf("Found access point(s) : %d¥n", exist_ap_count);
if (exist_ap_count >= max_networks)
{
   max_ap = max_networks;
}
else
{
   max_ap = exist_ap_count;
}
for (int i = 0; i < max_ap; i++ )
{
   printf("  --------------------¥n");
   printf("  ssid    : %s¥n", p[i].ssid);
   printf("  channel  : %d¥n", p[i].channel);
   printf("  rssi    : %d¥n", p[i].rssi);
   printf("  security : %d¥n", p[i].security);
}
```

Special Notes:

なし

3.5　R_WIFI_SX_ULPGN_Connect()

指定のアクセスポイントに接続します。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_Connect (
      const uint8_t *ssid,
      const uint8_t *pass,
      uint32_t security,
      uint8_t dhcp_enable
      wifi_ip_configuration_t *ip_config
)
```

Parameters

**ssid*	*アクセスポイントの SSID*
**pass*	*アクセスポイントのパスワード*
security	*セキュリティタイプ情報（WIFI_SECURITY_WPA、WIFI_SECURITY_WPA2）*
dhcp_enable	*IP アドレス自動割り当て（0: 無効、1: 有効)*
ip_config	*IP コンフィグレーション構造体のポインタ*

Return Values

WIFI_SUCCESS	*正常終了*
WIFI_ERR_NOT_OPEN	*Wi-Fi モジュールが初期化されていない*
WIFI_ERR_PARAMETER	*不正な引数*
WIFI_ERR_TAKE_MUTEX	*Mutex 取得失敗*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

ssid で指定したアクセスポイントへ接続します。

dhcp_enable=0 の場合、ip_config に IP アドレスを設定します。

dhcp_enable=1 の場合、DHCP で割り当てられた IP アドレスが ip_config に格納されます。

Reentrant

不可

Example

```
int32_t sock;
uint32_t ipadr = 0xc0a8010a; /* 192.168.1.10 */
uint16_t port = 80;         /* Port 80 */
wifi_ip_configuration_t ip_cfg;


R_WIFI_SX_ULPGN_Open();

/* DHCP 有効の場合 */
R_WIFI_SX_ULPGN_Connect("ssid", "passwd", WIFI_SECURITY_WPA2, 1, &ip_cfg);

/* DHCP 無効の場合 */
ip_cfg.ipaddr = 0xc0a80003;  //192.168.0.3
ip_cfg.subnetmask = 0xffffff00;   //255.255.255.0
ip_cfg.gateway = 0xc0a80001;   //192.168.0.1
R_WIFI_SX_ULPGN_Connect("ssid", "passwd", WIFI_SECURITY_WPA2, 0, &ip_cfg);

sock = R_WIFI_SX_ULPGN_CreateSocket(WIFI_SOCKET_IP_PROTOCOL_TCP,
WIFI_SOCKET_IP_VERSION_4);
R_WIFI_SX_ULPGN_RequestTlsSocket(sock);
R_WIFI_SX_ULPGN_ConnectSocket(sock, ipadr, port, NULL);
```

Special Notes:

なし

3.6　R_WIFI_SX_ULPGN_Disconnect()

接続中のアクセスポイントを切断します。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_Disconnect (
      void
)
```

Parameters

なし

Return Values

WIFI_SUCCESS	*正常終了*
WIFI_ERR_NOT_OPEN	*Wi-Fi モジュールが初期化されていない*
WIFI_ERR_TAKE_MUTEX	*Mutex 取得失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

現在接続中のアクセスポイントを切断します。

Reentrant

不可

Example

```
int32_t sock;
wifi_ip_configuration_t ip_cfg;

R_WIFI_SX_ULPGN_Open();
R_WIFI_SX_ULPGN_Connect("ssid", "passwd", WIFI_SECURITY_WPA2, 1, &ip_cfg);
R_WIFI_SX_ULPGN_Disconnect();
```

Special Notes:

なし

3.7　R_WIFI_SX_ULPGN_IsConnected()

Wi-Fi モジュールとアクセスポイントの接続状態を取得します。

Format

```
int32_t R_WIFI_SX_ULPGN_IsConnected (
      void
)
```

Parameters

なし

Return Values

0　　*アクセスポイントに接続中*

-1　　*アクセスポイント未接続*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

Wi-Fi モジュールとアクセスポイントの接続状態を返します。

Reentrant

不可

Example

```
if (0 == R_WIFI_SX_ULPGN_IsConnected())
{
   printf("connected ¥n");
}
else
{
   printf("not connect ¥n");
}
```

Special Notes:

なし

3.8　R_WIFI_SX_ULPGN_GetMacAddress()

この関数は、Wi-Fi モジュールの MAC アドレス値を取得する関数です。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_GetMacAddress (
      uint8_t *mac_address
)
```

Parameters

**mac_address*　　*MAC アドレスの格納領域へのポインタ（6 バイト）*

Return Values

WIFI_SUCCESS	*正常終了*
WIFI_ERR_NOT_OPEN	*Wi-Fi モジュールが初期化されていない*
WIFI_ERR_TAKE_MUTEX	*Mutex 取得失敗*
WIFI_ERR_PARAMETER	*不正な引数*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

Wi-Fi モジュールの MAC アドレス値を取得し、mac_address にバイナリ形式で格納します。

例）MAC アドレス　11:22:33:44:55:66

mac_address[0] = 0x11, mac_address [1] = 0x22,　…, mac_address [5] = 0x66

Reentrant

不可

Example

```
uint8_t mac[6];

R_WIFI_SX_ULPGN_Open();
R_WIFI_SX_ULPGN_GetMacAddress(mac);
printf("— MAC adr : %lx:%lx:%lx:%lx:%lx:%lx¥r¥n",
           mac[0],mac[1],mac[2],mac[3],mac[4],mac[5]);
```

Special Notes:

なし

3.9　R_WIFI_SX_ULPGN_GetIpAddress()

Wi-Fi モジュールに割り当てられた IP アドレスを取得します。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_GetIpAddress (
      wifi_ip_configuration_t *ip_config
)
```

Parameters

** ip_config　IP アドレス格納領域へのポインタ*

Return Values

WIFI_SUCCESS	*正常終了*
WIFI_ERR_NOT_OPEN	*Wi-Fi モジュールが初期化されていない*
WIFI_ERR_TAKE_MUTEX	*Mutex 取得失敗*
WIFI_ERR_PARAMETER	*不正な引数*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

Wi-Fi モジュールに割り当てられた、IP アドレス、サブネットマスク、ゲートウェイを取得し ip_config に格納します。

Reentrant

不可

Example

```
wifi_ip_configuration_t ip_cfg;
R_WIFI_SX_ULPGN_GetIpAddress(&ip_cfg);
```

Special Notes:

なし

3.10　R_WIFI_SX_ULPGN_CreateSocket()

ソケットタイプと IP タイプを指定しソケットを生成します。

Format

```
int32_t R_WIFI_SX_ULPGN_CreateSocket (
      uint32_t type,
      uint32_t ip_version
)
```

Parameters

type　　*ソケットタイプ*

WIFI_SOCKET_IP_PROTOCOL　　：　TCP

WIFI_SOCKET_IP_PROTOCOL_UDP　：UDP

ip_version　*IP バージョン（WIFI_SOCKET_IP_VERSION_4）*

Return Values

正の値	*正常終了（作成したソケットの番号）*
WIFI_ERR_PARAMETER	*不正な引数*
WIFI_ERR_NOT_CONNECT	*アクセスポイントに未接続*
WIFI_ERR_SOCKET_CREATE	*ソケット作成失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

作成したソケットの番号を整数値で返します。

Reentrant

不可

Example

```
int32_t sock_tcp;
int32_t sock_udp;
uint32_t ipadr = 0xc0a8010a; /* 192.168.1.10 */
uint16_t port = 80;         /* Port 80 */
wifi_ip_configuration_t ip_cfg;

R_WIFI_SX_ULPGN_Open();
R_WIFI_SX_ULPGN_Connect("ssid", "passwd", WIFI_SECURITY_WPA2, 1, &ip_cfg);
Sock_tcp = R_WIFI_SX_ULPGN_CreateSocket(WIFI_SOCKET_IP_PROTOCOL_TCP,
WIFI_SOCKET_IP_VERSION_4);
Sock_udp = R_WIFI_SX_ULPGN_CreateSocket(WIFI_SOCKET_IP_PROTOCOL_UDP,
WIFI_SOCKET_IP_VERSION_4);
```

Special Notes:

なし

3.11　R_WIFI_SX_ULPGN_ConnectSocket()

作成したソケットに接続します。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_ConnectSocket (
      uint8_t socket_number,
      uint32_t ip_address,
      uint16_t port,
      char    *destination
)
```

Parameters

socket_number	*ソケット番号*
ip_address	*通信相手の IP アドレス*
port	*通信相手のポート番号*
destination	*通信相手のサーバ名*

Return Values

WIFI_SUCCESS	*正常終了*
WIFI_ERR_PARAMETER	*不正な引数*
WIFI_ERR_SOCKET_NUM	*利用可能なソケット無し*
WIFI_ERR_TAKE_MUTEX	*Mutex 取得失敗*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*
WIFI_ERR_NOT_CONNECT	*アクセスポイントに未接続*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

R_WIFI_SX_ULPGN_CreateSocket()で作成したソケットに接続します。

R_WIFI_SX_ULPGN_RequestTlsSocket()を実行した場合は SSL で接続します。

存在しないソケットを指定した場合は WIFI_ERR_SOCKET_NUM を返します。

Reentrant

不可

Example

```
int32_t sock;
uint32_t ipadr = 0xc0a8010a; /* 192.168.1.10 */
uint16_t port = 80;         /* Port 80 */
wifi_ip_configuration_t ip_cfg;


R_WIFI_SX_ULPGN_Open();
R_WIFI_SX_ULPGN_Connect("ssid", "passwd", WIFI_SECURITY_WPA2, 1, &ip_cfg);
sock = R_WIFI_SX_ULPGN_CreateSocket(WIFI_SOCKET_IP_PROTOCOL_TCP,
WIFI_SOCKET_IP_VERSION_4);
R_WIFI_SX_ULPGN_RequestTlsSocket(sock);
R_WIFI_SX_ULPGN_ConnectSocket(sock, ipadr, port, NULL);
```

Special Notes:

なし

3.12　R_WIFI_SX_ULPGN_SendSocket()

指定したソケットでデータ送信します。

Format

```
int32_t R_WIFI_SX_ULPGN_SendSocket (
      uint8_t socket_number,
      uint8_t *data,
      uint32_t length,
      uint32_t timeout_ms
)
```

Parameters

socket_number	*ソケット番号*
**data*	*送信データ格納領域へのポインタ*
length	*送信したいデータバイト数*
timeout_ms	*送信のタイムアウト時間 [ms] （未使用）*

Return Values

正の値	*正常終了(送信完了したバイト数)*
WIFI_ERR_PARAMETER	*不正な引数*
WIFI_ERR_SOCKET_NUM,	*利用可能なソケット無し*
WIFI_ERR_NOT_CONNECT,	*アクセスポイントに未接続*
WIFI_ERR_TAKE_MUTEX	*Mutex 取得失敗*
WIFI_ERR_CHANGE_SOCKET	*ソケット切り替えの失敗*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

指定したソケットから data に格納されたデータを length で指定したバイト数を送信します。

Reentrant

不可

Example

```
int32_t xReturned;
xReturned = R_WIFI_SX_ULPGN_SendSocket(sock, buffer, sizeof(buffer), 1000);
```

Special Notes:

なし

3.13　R_WIFI_SX_ULPGN_ReceiveSocket()

指定したソケットからデータを受信します。

Format

```
int32_t R_WIFI_SX_ULPGN_ReceiveSocket (
      uint8_t socket_number,
      uint8_t *data,
      int32_t length,
      uint32_t timeout_ms
)
```

Parameters

socket_number	*ソケット番号*
**data*	*受信データ格納領域へのポインタ*
data_length	*受信したいデータバイト数*
timeout_ms	*受信のタイムアウト時間 [ms]*

Return Values

正の値	*正常終了(受信完了したバイト数)*
WIFI_ERR_PARAMETER	*不正な引数*
WIFI_ERR_NOT_CONNECT,	*アクセスポイントに未接続*
WIFI_ERR_SOCKET_NUM	*利用可能なソケット無し*
WIFI_ERR_TAKE_MUTEX	*Mutex 取得失敗*
WIFI_ERR_CHANGE_SOCKET	*ソケット切り替えの失敗*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

指定ソケットから length 分のデータを受信します。

タイムアウトが発生した場合、タイムアウト時点の受信データサイズを返します。

Reentrant

不可

Example

```
int32_t xReturned;
xReturned = R_WIFI_SX_ULPGN_ReceiveSocket(sock, buffer, sizeof(buffer), 1000);
```

Special Notes:

なし

3.14　R_WIFI_SX_ULPGN_ShutdownSocket()

指定したソケットの通信を切断します。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_ShutdownSocket (
      uint8_t socket_number
)
```

Parameters

socket_number　　*ソケット番号*

Return Values

WIFI_SUCCESS	*正常終了*
WIFI_ERR_NOT_CONNECT,	*アクセスポイントに未接続*
WIFI_ERR_SOCKET_NUM	*利用可能なソケット無し*
WIFI_ERR_TAKE_MUTEX	*Mutex 取得失敗*
WIFI_ERR_CHANGE_SOCKET	*ソケット切り替えの失敗*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

指定したソケットの通信を切断します。

ソケット自体は削除されないので R_WIFI_SX_ULPGN_ConnectSocket()で再度接続可能です。

Reentrant

不可

Example

```
int32_t sock;
wifi_ip_configuration_t ip_cfg;

R_WIFI_SX_ULPGN_Open();
R_WIFI_SX_ULPGN_Connect("ssid", "passwd", WIFI_SECURITY_WPA2, 1, &ip_cfg);
sock = R_WIFI_SX_ULPGN_CreateSocket(WIFI_SOCKET_IP_PROTOCOL_TCP,
WIFI_SOCKET_IP_VERSION_4);
R_WIFI_SX_ULPGN_ConnectSocket(sock, ipadr, port, NULL);
R_WIFI_SX_ULPGN_ShutdownSocket(sock);
R_WIFI_SX_ULPGN_ConnectSocket(sock, ipadr, port, NULL);
```

Special Notes:

なし

3.15　R_WIFI_SX_ULPGN_CloseSocket()

指定したソケットの通信を切断し、ソケットも削除します。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_CloseSocket (
      uint8_t socket_number
)
```

Parameters

socket_number　　*ソケット番号*

Return Values

WIFI_SUCCESS	*正常終了*
WIFI_ERR_NOT_CONNECT	*アクセスポイントに未接続*
WIFI_ERR_SOCKET_NUM	*利用可能なソケット無し*
WIFI_ERR_TAKE_MUTEX	*Mutex 取得失敗*
WIFI_ERR_CHANGE_SOCKET	*ソケット切り替えの失敗*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

指定したソケットの通信を切断およびソケットを削除します。

既に R_WIFI_SX_ULPGN_ShutdownSocket()が実行されている場合はソケット削除を行います。

Reentrant

不可

Example

```
ソケット切断済の場合
R_WIFI_SX_ULPGN_ConnectSocket(sock, ipadr, port, NULL);
R_WIFI_SX_ULPGN_ShutdownSocket(sock);
R_WIFI_SX_ULPGN_CloseSocket(sock);

ソケット未切断の場合
R_WIFI_SX_ULPGN_ConnectSocket(sock, ipadr, port, NULL);
R_WIFI_SX_ULPGN_CloseSocket(sock);
```

Special Notes:

なし

3.16　R_WIFI_SX_ULPGN_DnsQuery()

DNS クエリを実行します。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_DnsQuery (
      uint8_t *domain_name,
      uint32_t *ip_address
)
```

Parameters

** domain_name*　　*ドメイン名*

**ip_address*　　*IP アドレス格納領域領域*

Return Values

WIFI_SUCCESS	*正常終了*
WIFI_ERR_NOT_CONNECT	*アクセスポイントに未接続*
WIFI_ERR_PARAMETER	*不正な引数*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗または存在しないドメイン*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

DNS クエリを実行し、指定したドメインの IP アドレスを取得します。

Reentrant

不可

Example

```
uint32_t ipadr;
R_WIFI_SX_ULPGN_DnsQuery(“hostname”, &ipadr);
```

Special Notes:

なし

3.17　R_WIFI_SX_ULPGN_Ping()

指定した IP アドレスに ping を送信します。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_Ping (
      uint32_t ip_address,
      uint16_t count,
      uint32_t interval_ms
)
```

Parameters

ip_address　*送信先の IP アドレス*

count　*ping 送信回数*

interval_ms　*ping 送信間隔 [ms]*

Return Values

WIFI_SUCCESS	*正常終了*
WIFI_ERR_PARAMETER	*不正な引数*
WIFI_ERR_NOT_CONNECT	*アクセスポイントに未接続*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗または応答なし*
WIFI_ERR_TAKE_MUTEX	*Mutex の取得に失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

ip_address で指定した IP アドレスに ping を送信します。パラメータ（count、interval_ms）で送信回数と送信間隔を指定します。

Reentrant

不可

Example

```
uint32_t ip_addr = 0xc8a8010a; /* 192.168.1.10 */

R_WIFI_SX_ULPGN_Ping(ip_addr, 1, 1000);
```

Special Notes:

なし

3.18　R_WIFI_SX_ULPGN_GetVersion()

この関数は、本 FIT モジュールのバージョン情報を取得します。

Format

```
uint32_t R_WIFI_SX_ULPGN_GetVersion (
      void
)
```

Parameters

なし

Return Values

本 FIT モジュールのバージョン

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

本 FIT モジュールのバージョンを返します。

上位の 2 バイトがメジャーバージョン、下位 2 バイトがマイナーバージョンを示します。

Reentrant

不可

Example

```
uint32_t ver;
ver = R_WIFI_SX_ULPGN_GetVersion();
printf("Version V%d.%2d¥n", ((ver >> 16) & 0x0000FFFF), (ver & 0x0000FFFF));
```

Special Notes:

なし

3.19　R_WIFI_SX_ULPGN_RequestTlsSocket()

作成済の TCP ソケットを TLS 通信用に割り当てます。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_RequestTlsSocket (
      uint8_t socket_number
)
```

Parameters

socket_number　　*TCP ソケット番号*

Return Values

WIFI_SUCCESS	*正常終了*
WIFI_ERR_SOCKET_NUM	*利用可能なソケット無し*
WIFI_ERR_NOT_CONNECT	*アクセスポイントに未接続*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

R_WIFI_SX_ULPGN_CreateSocket()で作成した TCP ソケットを TLS 通信用に割り当てます。

R_WIFI_SX_ULPGN_ConnectSocket()を実行前に呼びだしてください。

Reentrant

不可

Example

```
int32_t sock;
uint32_t ipadr = 0xc0a8010a; /* 192.168.1.10 */
uint16_t port = 80;          /* Port 80 */
wifi_ip_configuration_t ip_cfg;

R_WIFI_SX_ULPGN_Open();
R_WIFI_SX_ULPGN_Connect("ssid", "passwd", WIFI_SECURITY_WPA2, 1, &ip_cfg);
sock=R_WIFI_SX_ULPGN_CreateSocket(WIFI_SOCKET_IP_PROTOCOL_TCP,WIFI_SOCKET_IP_VERSION_4);
R_WIFI_SX_ULPGN_RequestTlsSocket(sock);
R_WIFI_SX_ULPGN_ConnectSocket(sock, ipadr, port, NULL);
```

Special Notes:

なし

3.20　R_WIFI_SX_ULPGN_WriteServerCertificate()

この関数は、Wi-Fi モジュールに証明書を格納する関数です。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_WriteServerCertificate (
      uint32_t data_id,
      uint32_t data_type,
      const uint8_t *certificate,
      uint32_t certificate_length
)
```

Parameters

data_id	*証明書 ID (0~4)*
data_type	*証明書タイプ(0：certificate、1：CA list)*
certificate	*証明書データのポインタ*
certificate_length	*証明書サイズ*

Return Values

WIFI_SUCCESS	*正常終了*
WIRI_ERR_PARAMETER	*証明書データが正しくセットされていない*
WIFI_ERR_NOT_OPEN	*Wi-Fi モジュールが初期化されていない*
WIFI_ERR_TAKE_MUTEX	*Mutex の取得に失敗*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

証明書ファイル名は、証明書 ID コード、証明書タイプにより以下のように設定されます。

証明書タイプ 0 の場合：cert<証明書 ID＞.crt

証明書タイプ 1 の場合：calist<証明書 ID＞.crt

Wi-Fi モジュールには 5 セットまで書き込むことができます。

証明書データはバイナリフォーマットに変換する必要があります。詳細は 5.3 章を参照してください。

Reentrant

不可

Example

```
const uint8_t cert_data[1053] =
{
    ...
};

const uint8_t ca_data[1053] =
{
    ...
};

cert_info_t cert_info;

/* Write certificate0 : cert0.crt */
R_WIFI_SX_ULPGN_WriteServerCertificate(0, 0, cert_data, sizeof(cert_data));

/* Write CA list0 : calist0.crt */
R_WIFI_SX_ULPGN_WriteServerCertificate(0, 1, ca_data, sizeof(ca_data));

/* Write certificate1 : cert1.crt */
R_WIFI_SX_ULPGN_WriteServerCertificate(1, 0, cert_data, sizeof(cert_data));

/* Write CA list1 : calist1.crt */
R_WIFI_SX_ULPGN_WriteServerCertificate(1, 1, ca_data, sizeof(ca_data));

/* Get certificate list */
/* cert_info.cert[0].file_name = cert0.crt */
/* cert_info.cert[1].file_name = calist0.crt */
/* cert_info.cert[2].file_name = cert1.crt */
/* cert_info.cert[3].file_name = calist1.crt */
R_WIFI_SX_ULPGN_GetServerCertificate(&cert_info);

/* Erase cert0.crt */
R_WIFI_SX_ULPGN_EraseServerCertificate(cert_info.cert[0].file_name);

/* Erase all */
R_WIFI_SX_ULPGN_EraseAllServerCertificate();
```

Special Notes:

なし

3.21　R_WIFI_SX_ULPGN_EraseServerCertificate()

この関数は、Wi-Fi モジュールに格納されている証明書を削除する関数です。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_EraseServerCertificate (
      uint8_t *certificate_name
)
```

Parameters

certificate_name　　*証明書ファイル名のポインタ*

Return Values

WIFI_SUCCESS	*正常終了*
WIRI_ERR_PARAMETER	*証明書ファイル名が正しくセットされていない*
WIFI_ERR_TAKE_MUTEX	*Mutex 取得失敗*
WIFI_ERR_NOT_OPEN	*Wi-Fi モジュールが初期化されていない*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

この関数は、証明書ファイル名を指定し、WiFI モジュールに格納されている証明書を消去します。

本関数を実行する前に R_WIFI_SX_ULPGN_Open()を実行しておく必要があります。

WiFI モジュールに格納されている証明書は R_WIFI_SX_ULPGN_GetServerCertificate()で確認できます。

Reentrant

不可

Example

R_WIFI_SX_ULPGN_WriteServerCertificate()を参照

Special Notes:

なし

3.22　R_WIFI_SX_ULPGN_GetServerCertificate()

この関数は、Wi-Fi モジュールに格納されている証明書ファイル名を取得する関数です。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_GetServerCertificate (
      wifi_certificate_infomation_t *wifi_certificate_information
)
```

Parameters

wifi_certificate_information　*証明書情報格納領域へのポインタ*

Return Values

WIFI_SUCCESS	*正常終了*
WIRI_ERR_PARAMETER	*証明書ファイル名が正しくセットされていない*
WIFI_ERR_TAKE_MUTEX	*Mutex の取得に失敗*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

この関数は Wi-Fi モジュールに格納された証明書情報を取得し、証明書情報の先頭アドレスを wifi_certificate_information に返します。

本関数を実行する前に R_WIFI_SX_ULPGN_Open()を実行しておく必要があります。

Reentrant

不可

Example

`R_WIFI_SX_ULPGN_WriteServerCertificate()`を参照

Special Notes:

なし

3.23　R_WIFI_SX_ULPGN_EraseAllServerCertificate()

この関数は、Wi-Fi モジュールに格納されている証明書をすべて削除する関数です。

Format

```
wifi_err_t R_WIFI_SX_ULPGN_EraseAllServerCertificate (
      void
)
```

Parameters

なし

Return Values

WIFI_SUCCESS	*正常終了*
WIFI_ERR_NOT_OPEN	*Wi-Fi モジュールが初期化されていない*
WIFI_ERR_TAKE_MUTEX	*Mutex の取得に失敗*
WIFI_ERR_MODULE_COM	*Wi-Fi モジュールとの通信に失敗*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

この関数は Wi-Fi モジュールに格納されている、すべての証明書を消去します。

本関数を実行する前に R_WIFI_SX_ULPGN_Open()を実行しておく必要があります。

Reentrant

不可

Example

`R_WIFI_SX_ULPGN_WriteServerCertificate()`を参照

Special Notes:

なし

3.24　R_WIFI_SX_ULPGN_SetCertificateProfile()

この関数は、WiFI モジュールに格納した証明書とサーバ情報の紐づけを行います。

Format

```
void R_WIFI_SX_ULPGN_SetCertificateProfile
      uint8_t  certificate_id,
      uint32_t ip_address,
      char     *sever_name
)
```

Parameters

certificate_id	*証明書 ID 番号*
ip_address	*サーバ IP アドレス*
server_name	*サーバ名のポインタ*

Return Values

WIFI_SUCCESS	*正常終了*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

この関数は WiFI モジュールに格納した証明書とサーバ情報の紐づけを行います。

証明書 ID は必須の指定項目です。サーバ IP アドレス、サーバ名はどちらか一方を指定してください。

両方に設定がある場合はサーバ IP アドレスが優先されます。

Reentrant

不可

Example

```
uint32_t ip_addr = 0xc0a80105; /* 192.168.1.5 */

/* 証明書 ID0 に IP アドレスを紐づけ */
R_WIFI_SX_ULPGN_SetCertificateProfile(0, ip_addr, NULL);

/* 証明書 ID1 にサーバ名を紐づけ */
R_WIFI_SX_ULPGN_SetCertificateProfile(1, 0, “ServerName”);
```

Special Notes:

なし

3.25　R_WIFI_SX_ULPGN_GetTcpSocketStatus()

指定ソケットのステータスを取得します。

Format

```
int32_t R_WIFI_SX_ULPGN_GetTcpSocketStatus (
      uint8_t socket_number
)
```

Parameters

socket_number　　*ソケット番号*

Return Values

0	*CLOSED*
1	*SOCKET*
2	*BOUND*
3	*LISTEN*
4	*CONNECTED*
5	*BROKEN*
-1	*指定ソケットが存在しない*

Properties

r_wifi_sx_ulpgn_if.h にプロトタイプ宣言されています。

Description

引数で指定したソケットのステータスを返します。

Reentrant

不可

Example

```
int32_t sock_status;
int32_t sock;
wifi_ip_configuration_t ip_cfg;

R_WIFI_SX_ULPGN_Open();
R_WIFI_SX_ULPGN_Connect("ssid", "passwd", WIFI_SECURITY_WPA2, 1, &ip_cfg);
sock = R_WIFI_SX_ULPGN_CreateSocket(WIFI_SOCKET_IP_PROTOCOL_TCP,
WIFI_SOCKET_IP_VERSION_4);
sock_status = R_WIFI_SX_ULPGN_GetTcpSocketStatus(sock);
```

Special Notes:

なし

4. コールバック関数

4.1 callback()

Wi-Fi モジュールのエラーをユーザアプリケーションに通知します。

Format

```
void * callback(
      void * pevent
)
```

Parameters

pevent　　*エラー情報の領域を指すポインタ*

Return Values

なし

Properties

本関数はユーザにて実装します。

Description

以下のコンフィグレーションで本 API を有効にします。関数名は”callback”である必要はありません。

```
#define WIFI_CFG_USE_CALLBACK_FUNCTION  (1)
#define WIFI_CFG_CALLBACK_FUNCTION_NAME (wifi_callback)
```

イベントが void ポインタ型で通知されるので wifi_err_event_t 型にキャストしてから参照します。

```
void wifi_callback(void *p_args)
{
   wifi_err_event_t *pevent;
   pevent = (wifi_err_event_t *)p_args;

   switch(pevent->event)
   {
      case WIFI_EVENT_SERIAL_OVF_ERR:
         break;
・・・
   }
}
```

通知イベントは以下の通りです。

- WIFI_EVENT_SERIAL_OVF_ERR
  SCI モジュールの受信オーバーフローエラーを通知します。
- WIFI_EVENT_SERIAL_FLM_ERR
  SCI モジュールの受信フレーミングエラーを通知します。
- WIFI_EVENT_SERIAL_RXQ_OVF_ERR
  SCI モジュールの受信キュー（BYTEQ）オーバーフローを通知します
- WIFI_EVENT_RCV_TASK_RXB_OVF_ERR
  本 FIT モジュールの AT コマンド受信バッファのオーバーフローを通知します。
- WIFI_EVENT_SOCKET_RXQ_OVF_ERR
  ソケットの受信キュー（BYTEQ）オーバーフローを通知します。

Reentrant

不可

Example

```
[r_wifi_sx_ulpgn_config.h]
#define WIFI_CFG_USE_CALLBACK_FUNCTION  (1)
#define WIFI_CFG_CALLBACK_FUNCTION_NAME (wifi_callback)

[xxx.c]
void wifi_callback(void *p_args)
{
   wifi_err_event_t *pevent;
   pevent = (wifi_err_event_t *)p_args;

   switch(pevent->event)
   {
      case WIFI_EVENT_SERIAL_OVF_ERR:
         break;
      case WIFI_EVENT_SERIAL_FLM_ERR:
         break;
      case WIFI_EVENT_SERIAL_RXQ_OVF_ERR:
         break;
      case WIFI_EVENT_RCV_TASK_OVF_ERR:
         break;
      case WIFI_EVENT_SOCKET_RXQ_OVF_ERR:
         switch(pevent->socket_number)
         {
            case 0:
               break;
            case 1:
               break;
            case 2:
               break;
            case 3:
               break;
         }
         break;
      default:
         break;
   }
}
```

Special Notes:

コールバック関数内で 3.API 関数で示す関数を実行しないでください。

5. 付録

5.1 動作確認環境

本 FIT モジュールの動作確認環境を以下に示します。

表 5.1　動作確認環境 (Ver.1.15)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio 2022.04
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.04.00
	コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev1.15
使用ボード	Renesas RX65N Cloud Kit (型名：RTK5RX65N0SxxxxxBE) RX72N Envision Kit (型名：RTK5RX72N0C00000BJ) Renesas Starter Kit+ for RX671 (型名：RTK55671EHSxxxxxBE)

5.2　トラブルシューティング

(1) Q：本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「Could not open source file "platform.h"」エラーが発生します。

A：FIT モジュールがプロジェクトに正しく追加されていない可能性があります。プロジェクトへの追加方法をご確認ください。

- CS+を使用している場合
  アプリケーションノート RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」
- e^2 studio を使用している場合
  アプリケーションノート RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)」

また、本 FIT モジュールを使用する場合、ボードサポートパッケージ FIT モジュール(BSP モジュール)もプロジェクトに追加する必要があります。BSP モジュールの追加方法は、アプリケーションノート「ボードサポートパッケージモジュール(R01AN1685)」を参照してください。

(2) Q：本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「[コンフィグ設定が間違っている場合のエラーメッセージ]」エラーが発生します。

A：“r_wifi_sx_ulpgn_config.h”ファイルの設定値が間違っている可能性があります。“r_wifi_sx_ulpgn_config.h”ファイルを確認して正しい値を設定してください。詳細は「2.7 コンパイル時の設定」を参照してください。

(3) Q：[端子設定していない場合発生する現象]が発生します。

A：正しく端子設定が行われていない可能性があります。本 FIT モジュールを使用する場合は端子設定が必要です。詳細は「4 端子設定」を参照してください。

(4) Q：SX-ULPGN を Transparent mode（1CH 通信モード）で使うと、ターミナルソフトにエラーログが出力されます。

A：r_wifi_sx_ulpgn_config.h の内容を、以下を参考に設定してください。

例）RX72N Enviosion Kit の場合

・SCI7 を使用する場合

```
#define WIFI_CFG_SCI_CHANNEL (7)
```

・Separate port mode（2CH 通信モード）は未使用（WIFI_CFG_SCI_CHANNEL と同じ Ch を設定）

```
#define WIFI_CFG_SCI_SECOND_CHANNEL (7)
```

・WIFI_CFG_SCI_CHANNEL に、SCI0～6,12 を設定している場合は PCLKB、SCI7～11 を設定している場合は PCLKA の周波数を設定。
クロックの詳細は、「RX72N グループ ユーザーズマニュアル ハードウェア編」を参照。

```
#define WIFI_CFG_SCI_PCLK_HZ (120000000)
```

5.3　Appendix（証明書データの組み込み手順）

TLS 通信を行うために Wi-Fi モジュールに書き込む証明書の作成手順を示します。

5.3.1　証明書の作成

証明書は OpenSSL で作成します。PC に OpenSSL をインストールしてください。
証明書作成手順は以下の通りです。

```
Openssl genrsa -out certs/client.key 2048

openssl req -new -key certs/client.key -out certs/client.csr ¥
-subj "/C=JP/L=<States>/O=<Company>/OU=<Department>/CN=<Object>/email=<EmailAddress>"

openssl x509 -req -in certs/client.csr -CA certs.server.pem -Cakey certs/server.key ¥
-Cacreateserial -out certs/client.pem -days 365 -sha256"
```

5.3.2　フォーマットの変換

Wi-Fi モジュールへ書き込みを行う証明書データは SharkSSLParseCert バイナリフォーマット、CA リストは SharkSSLPerseCAList バイナリフォーマットに変換しておく必要があります。
フォーマット変換は下記のフリーソフトウェアで行うことができます。

SharkSSL　<https://realtimelogic.com/downloads/sharkssl/>

ダウンロード、インストールはソフトウェアの指示に従ってください。

フォーマット変換時の出力ファイルは、変換した証明書をプログラムに組み込む場合と、

PC から直接書き込みを行う場合で 2 通りの出力が行えます。

証明書のフォーマット変換方法は以下の通りです。

① ルート証明書（Class 2 Root CA）を入手する

② ルート証明書を SharkSSL バイナリフォーマットに変換する

（プログラムに組み込む場合）

> SharkSSLParseCAList.exe xxxx.cer > starfield.c

（PC から直接書き込みを行う場合）

> SharkSSLParseCAList.exe xxxx.cer –b xxxx.bin

③ クライアント証明書と秘密鍵を SharkSSL バイナリフォーマットに変換する

（プログラムに組み込む場合）

> SharkSSLParseCert XXXX-certificate.pem.crt XXXX-private.pem.key > mycert.c

（PC から直接書き込みを行う場合）

> SharkSSLParseCert XXXX-certificate.pem.crt XXXX-private.pem.key –b XXXX-certific
ate.bin

5.3.3　Wi-Fi ドライバへの証明書の登録

5.3.2 章で変換したルート証明書、クライアント証明書、秘密鍵を、

本 API で証明書の書き込みを行う場合はご使用のプロジェクトに変換したファイルを組み込んでご使用ください。API を使った証明書の書き込みは、3 API 関数を参照ください。

PC から直接 Wi-Fi モジュールに書き込む場合、PC と Wi-Fi モジュールの TX0、RX0 端子を USB シリアル変換経由で接続し下記の AT コマンドを用いて書き込みを行ってください。
ボーレートは 115200bps としてください。
以下に証明書書き込みを行う場合の AT コマンド例を記載します。

（AT コマンド例）

- ATNSSLCERT=<証明書ファイル名>,<証明書サイズ>

  AT コマンド送信後 30 秒以内にバイナリファイルを送信してください。

証明書ファイル名：Wi-Fi モジュールに登録する証明書ファイル名です。20 文字以内で設定してください。
CA リストの場合は”calist＜番号>.crt“
クライアント証明書の場合は”cert<番号>.crt”

証明書サイズ　　：バイナリデータのサイズ（バイト数）を設定してください。

6. 参考ドキュメント

ユーザーズマニュアル：ハードウェア
（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート／テクニカルニュース
（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：開発環境
RX ファミリ CC-RX コンパイラ ユーザーズマニュアル（R20UT3248）
（最新版をルネサス エレクトロニクスホームページから入手してください。）

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2021/7/21	-	新規作成
1.13	2021/11/5	-	・全体のスタイルを見直し（フォント、語句の統一） ・対応 OS に AzureRTOS を追加 1.3.2 ソフトウェア構成 2.2 ソフトウェアの要求 図 1.3　アプリケーション構成図 表 2.4 Configuration options(r_bsp_config.h) ・2.9 戻り値　のスタイルを修正 ・最新のコンフィグに合わせて表を更新 表 2.1 Configuration options(r_wifi_sx_ulpgn_config.h) ・表 5.1　動作確認環境（Ver.1.00）の IDE、コンパイラバージョン、モジュールのリビジョンを修正 ・新規 API を追加 3.25 R_WIFI_SX_ULPGN_GetTcpSocketStatus () ・API 引数の型を修正 3.5　R_WIFI_SX_ULPGN_Connect() 3.11 R_WIFI_SX_ULPGN_ConnectSocket () 3.12 R_WIFI_SX_ULPGN_SendSocket () 3.13 R_WIFI_SX_ULPGN_ReceiveSocket () 3.14 R_WIFI_SX_ULPGN_ShutdownSocket () 3.16 R_WIFI_SX_ULPGN_DnsQuery() 3.19 R_WIFI_SX_ULPGN_RequestTlsSocket () 3.20 R_WIFI_SX_ULPGN_WriteServerCertificate() ・UDP 関連の記述を追加 3.10 R_WIFI_SX_ULPGN_CreateSocket () 3.19 R_WIFI_SX_ULPGN_RequestTlsSocket () ・3.1～3.25 全 API の Description、Example を見直し ・4.1 コールバック関数の記述を見直し ・5.3.3 章のタイトルを修正 ・表 2.5 コードサイズ表を更新
1.14	2022/3/4	プログラム	・以下の通り、FIT モジュールを修正。 ■内容 下記 API で使用している標準関数 vsscanf の変数指定を見直し ・R_WIFI_SX_ULPGN_Scan() ・R_WIFI_SX_ULPGN_GetMacAddress() ・R_WIFI_SX_ULPGN_GetTcpSocketStatus() ・R_WIFI_SX_ULPGN_ConnectSocket() ・R_WIFI_SX_ULPGN_GetServerCertificate() ・R_WIFI_SX_ULPGN_EraseAllServerCertificate()
1.15	2022/5/20	-	・RX671 のサポートを追加。
		-	・表 2.1 に WIFI_CFG_SCI_BAUDRATE を記載。
		プログラム	・以下の通り、FIT モジュールを修正。 ■内容 ・内部バッファクリアを追加。 ・ビッグエンディアンでの IP アドレス処理を修正。

1.16	2022/5/27	プログラム	・以下の通り、FIT モジュールを修正。 ■内容 ・FIT モジュールバージョン情報マクロを修正。

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。