R01AN6324JJ0112
Rev.1.12
2024.07.22

RX ファミリ

RYZ014A/GM01Q Cellular モジュール制御モジュール

Firmware Integration Technology

要旨

本アプリケーションノートは、Firmware Integration Technology (FIT)に準拠した RYZ014A/GM01Q Cellular モジュール制御 FIT モジュールについて説明します。

以降、RYZ014A/GM01Q Cellular モジュール制御 FIT モジュールのソフトウェアを総じて“RYZ014A Cellular 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する Cellular モジュールは以下です。

- Renesas Electronics 社製 RYZ014A Cellular モジュール (RYZ014A)
- Sequans 社製 GM01Q モジュール (GM01Q)

以降、上記 Cellular モジュールを“RYZ014A Cellular モジュール”、または“Cellular モジュール”と称します。

本 FIT モジュールは、リアルタイム OS (以下、RTOS)を使用します。RTOS と合わせて使用してください。また、本 FIT モジュールは以下の FIT モジュールを使用します。

- ボードサポートパッケージモジュール (R01AN1685)
- RX ファミリ SCI モジュール (R01AN1815)
- RX ファミリ バイト型キューバッファ(BYTEQ)モジュール (R01AN1683)
- RX ファミリ IRQ モジュール (R01AN1668)

動作確認デバイス

RX65N/RX651 グループ
RX66N グループ
RX72M グループ
RX72N グループ

本 FIT モジュールの動作確認環境については、4.1 節を参照してください。

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。

注意事項：当社での RYZ014A モジュールの取り扱いについて

当社は、RYZ014A (CAT-M1) 型名の既存 LTE モジュールの製造を中止し、製品の出荷を終了することを発表しました。

現在の設計または生産中に本製品を使用している場合、Sequans 社の製品型名 GM01Q (CAT-M1) が、RYZ014A とピン及び機能コンパティビリティ代替品となります。本ドライバは、以下の組み合わせにて継続してご利用可能です。

– RYZ014A Cellular モジュール制御モジュール： Sequans 社 GM01Q が互換のある代替モジュールとなります。

なお、RYZ014A の EOL 通知は製品ページ内の本リンクをご参照ください。

関連ドキュメント

[1] Firmware Integration Technology ユーザーズマニュアル(R01AN1833)
[2] RX ファミリ ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)
[3] e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)
[4] CS+に組み込む方法 Firmware Integration Technology (R01AN1826)
[5] Renesas e^2 studio スマート・コンフィグレータ ユーザーガイド(R20AN0451)
[6] RX ファミリ SCI モジュール Firmware Integration Technology (R01AN1815)
[7] RX ファミリ BYTEQ モジュール Firmware Integration Technology (R01AN1683)
[8] RX ファミリ IRQ モジュール Firmware Integration Technology (R01AN1668)

目次

1. 概要

1.1 RYZ014A Cellular 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、「2.11 FIT モジュールの追加方法」を参照してください。

1.2 RYZ014A Cellular FIT モジュールの概要

本 FIT モジュールは、Cellular モジュールとの UART 通信をサポートします。

Cellular モジュールドライバには下記 3 種類の実装タイプがあり、本 FIT モジュールは実装タイプ A のドライバです。

1. 実装タイプ A :
   Cellular モジュールの TCP/IP 通信機能に対応したドライバソフトウェア。
   Cellular モジュールの FOTA (Firmware upgrade Over-The-Air)を HTTPS で実行する場合の SSL 通信に必要なプロトコル制御や暗号化処理は Cellular モジュールが担当する。
2. 実装タイプ B :
   実装タイプ A に対して SSL 通信機能を追加したドライバソフトウェア。
   SSL 通信に必要なプロトコル制御や暗号化処理は Cellular モジュールが担当する。
3. 実装タイプ C :
   Cellular モジュールの PPP (Point to Point Protocol)通信機能に対応したドライバソフトウェア。

本 FIT モジュールは、PPP には対応していません。また、Cellular モジュールの WDT 機能を使用しません。

1.2.1　PMOD Expansion Board for RYZ014A との接続

PMOD Expansion Board for RYZ014A の接続例を、図 1.1 に示します。本 FIT モジュールは、PMOD_9 および PMOD_10 端子を使用しません。

本 FIT モジュールがサポートする接続タイプは、RYZ014 Module System Integration Guide の「1.1 Generic Rules」に記載されている Type 1 および Type2 です。

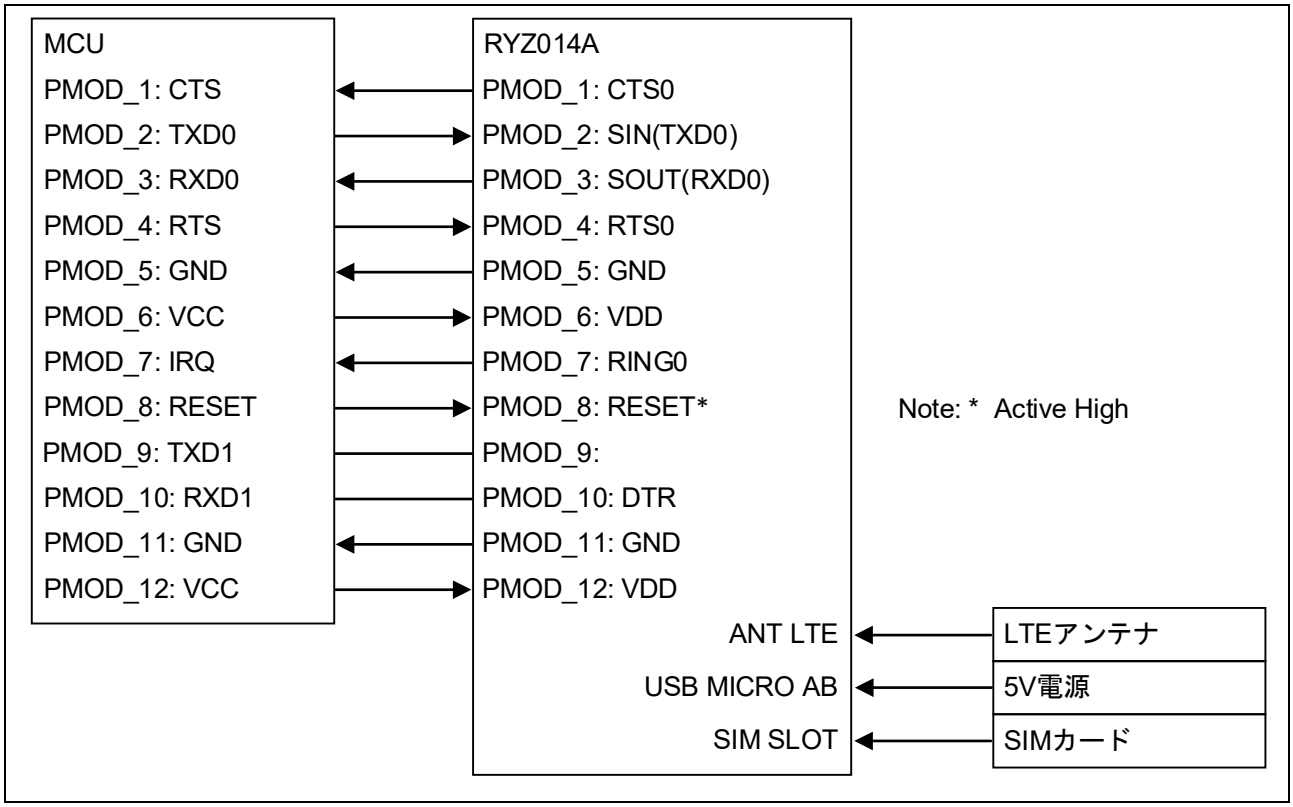

図 1.1 RX65N Cloud Kit と PMOD Expansion Board for RYZ014A の接続例

1.2.2　ソフトウェア構成

ソフトウェア構成を、図 1.2 に示します。

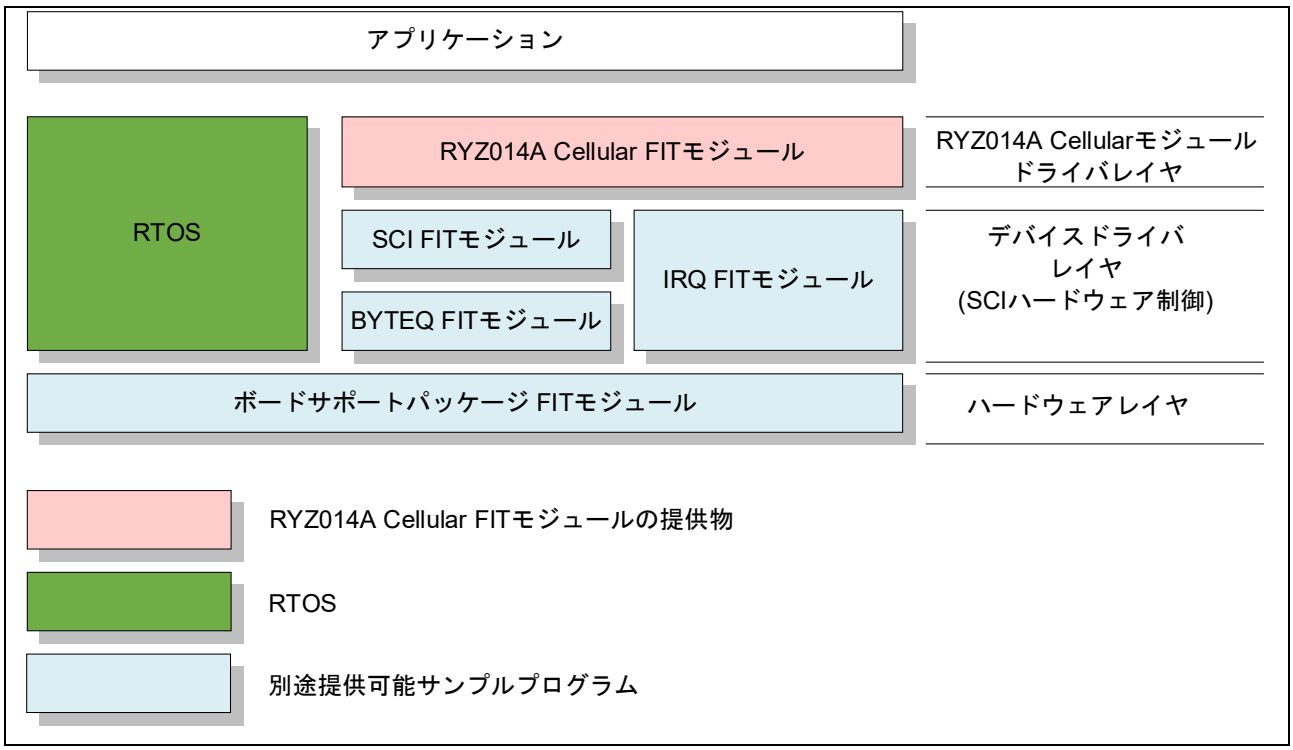

図 1.2 ソフトウェア構成図

(1) RYZ014A Cellular FIT モジュール
本 FIT モジュールです。RYZ014A Cellular 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
RYZ014A Cellular モジュールとマイコン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) IRQ FIT モジュール
RYZ014A Cellular モジュールからの特定の通知を割込みとして処理します。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。

(4) BYTEQ FIT モジュール
シリアルデータのバッファリングを行います。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。

(5) ボードサポートパッケージ FIT モジュール
マイコンの設定を行います。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。

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

1.2.3　API の概要

本 FIT モジュールに含まれる API 関数を、表 1.1 に示します。

表 1.1 API 関数一覧

関数	説明
R_CELLULAR_Open	本 FIT モジュールと Cellular モジュールの初期化を行います。
R_CELLULAR_Close	本 FIT モジュールと Cellular モジュールとの通信をクローズします。
R_CELLULAR_APConnect	Cellular モジュールをアクセスポイントに接続します。
R_CELLULAR_Disconnect	Cellular モジュールをアクセスポイントから切断します。
R_CELLULAR_IsConnected	本 FIT モジュールのアクセスポイント接続状態を取得します。
R_CELLULAR_CreateSocket	ソケットを作成します。
R_CELLULAR_ConnectSocket	ソケット通信を開始します。
R_CELLULAR_CloseSocket	ソケットをクローズします。
R_CELLULAR_ShutdownSocket	ソケット通信を終了します。
R_CELLULAR_SendSocket	ソケットのデータ送信を行います。
R_CELLULAR_ReceiveSocket	ソケットのデータ受信を行います。
R_CELLULAR_DnsQuery	DNS クエリを実行します。
R_CELLULAR_GetTime	Cellular モジュールに設定されている時刻情報を取得します。
R_CELLULAR_SetTime	Cellular モジュールへ時刻情報を設定します。
R_CELLULAR_SetEDRX	eDRX (extended Discontinuous Reception)の設定を行います。
R_CELLULAR_GetEDRX	eDRX (extended Discontinuous Reception)のパラメータを取得します。
R_CELLULAR_SetPSM	PSM (Power Saving Mode)の設定を行います。
R_CELLULAR_GetPSM	PSM (Power Saving Mode)のパラメータを取得します。
R_CELLULAR_GetICCID	ICCID (IC Card Identifier)を取得します。
R_CELLULAR_GetIMEI	IMEI (International Mobile Equipment Identifier)を取得します。
R_CELLULAR_GetIMSI	IMSI (International Mobile Subscriber Identity)を取得します。
R_CELLULAR_GetPhonenum	電話番号を取得します。
R_CELLULAR_GetRSSI	RSSI (Received Signal Strength Indicator)と、 BER (Bit Error Rate)を取得します。
R_CELLULAR_GetSVN	Cellular モジュールの SVN (Software Version Number)およびリビジョン情報を取得します。
R_CELLULAR_Ping	PING を実行します。
R_CELLULAR_GetAPConnectState	Cellular モジュールのアクセスポイント接続状態を取得します。
R_CELLULAR_GetCellInfo	セル情報を取得します。
R_CELLULAR_AutoConnectConfig	アクセスポイントへの自動接続の設定を行います。
R_CELLULAR_SetOperator	通信で使用するオペレータモードを設定します。
R_CELLULAR_SetBand	通信で使用する LTE バンドを設定します。
R_CELLULAR_GetPDPAddress	PDP アドレスを取得します。
R_CELLULAR_FirmUpgrade	FOTA (Firmware upgrade Over-The-Air)を実行します。
R_CELLULAR_FirmUpgradeBlocking	FOTA (Firmware upgrade Over-The-Air)をブロッキングモードで実行します。
R_CELLULAR_GetUpgradeState	FOTA (Firmware upgrade Over-The-Air)の状態を取得します。
R_CELLULAR_UnlockSIM	PIN コードを入力します。
R_CELLULAR_WriteCertificate	証明書または秘密鍵をモジュールの不揮発メモリへ書き込みます。
R_CELLULAR_EraseCertificate	証明書または秘密鍵をモジュールの不揮発メモリから削除します。

R_CELLULAR_GetCertificate	証明書または秘密鍵をモジュールの不揮発メモリから取得します。
R_CELLULAR_ConfigSSLProfile	SSL プロファイルを設定します。
R_CELLULAR_SoftwareReset	Cellular モジュールを AT コマンドでリセットします。
R_CELLULAR_HardwareReset	Cellular モジュールを RESET 端子でリセットします。
R_CELLULAR_FactoryReset	Cellular モジュールに設定されている値を工場出荷時の状態に戻します。
R_CELLULAR_RTS_Ctrl	Cellular モジュールの RTS0 端子の Low/High を切り替えます。

1.2.4　状態遷移図

本 FIT モジュールの状態遷移図を、図 1.3 に示します。図中の“R_CELLULAR_XXX”は表 1.1 に示す API 関数の呼び出しを示します。図中の「ユーザアプリケーションで実装」と記載している処理は、RYZ014 Module System Integration Guide の「2.2.8.1 RRC Idle」に記載されている Connection manager を実装するためにユーザアプリケーションへの実装が必要な処理です。

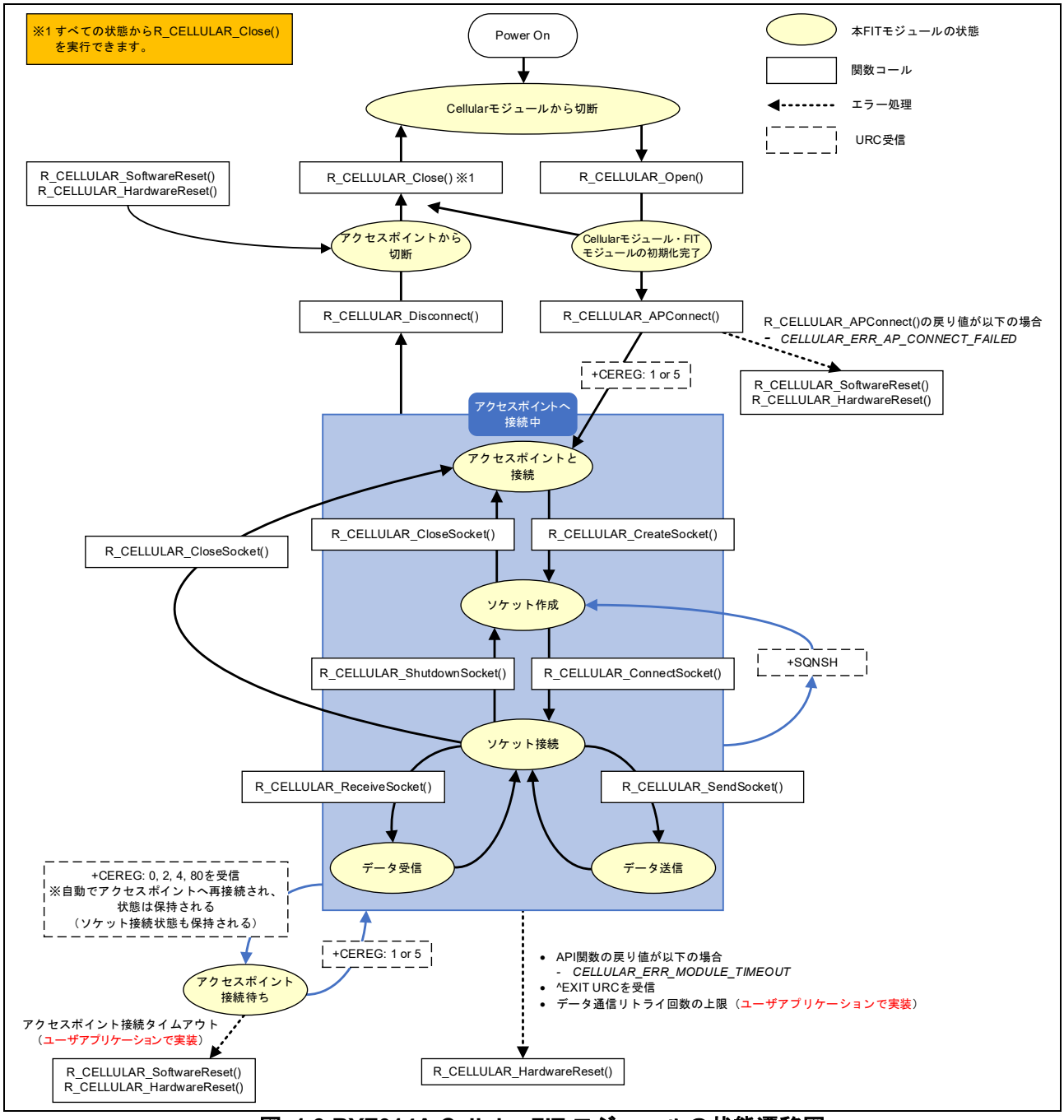

図 1.3 RYZ014A Cellular FIT モジュールの状態遷移図

2. API 情報

本 FIT モジュールは、下記の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
- IRQ
- 割り込み要因として設定できる 1 つ、または複数の IRQ 入力端子

2.2 ソフトウェアの要求

本 FIT モジュールは、以下の FIT モジュールに依存しています。

- r_bsp
- r_sci_rx
- r_byteq
- r_irq_rx

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、「4.1 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_cellular_if.h で定義されています。

2.6 整数型

本 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グレーションオプションの設定は、r_cellular_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、表 2.1 に示します。

表 2.1 Configuration options (r_cellular_config.h)

Configuration options in r_cellular _config.h	
CELLULAR_CFG_AP_NAME ※デフォルトは“iot.truphone.com”	接続するアクセスポイント名を指定します。 使用する SIM に合わせて設定してください。
CELLULAR_CFG_AP_USERID ※デフォルトは“”	接続するアクセスポイントのユーザ名を設定します。 使用する SIM に合わせて設定してください。 ユーザ名が無い場合は設定不要です。
CELLULAR_CFG_AP_PASSWORD ※デフォルトは“”	接続するアクセスポイントのパスワードを設定します。 使用する SIM に合わせて設定してください。 パスワードが無い場合は設定不要です。
CELLULAR_CFG_PIN_CODE ※デフォルトは“”	使用する SIM の PIN コードを設定します。 PIN コードが設定されていない場合は設定不要です。
CELLULAR_CFG_AUTH_TYPE ※デフォルトは“0”	接続の際に使用される認証プロトコルタイプを設定します。 使用する SIM に合わせて設定してください。 0: None, 1: PAP, 2: CHAP
CELLULAR_CFG_NETWORK_NOTIFY_LEVEL ※デフォルトは“2”	ネットワーク接続状態の自動通知設定を行います。 “0~5”の範囲で設定してください。ログ情報へ通知を出力する場合は、CELLULAR_CFG_DEBUGLOG を’4’に設定してください。 0: OFF, 1: 接続状態のみ通知, 2: 1 に追加で位置情報を通知, 3: 2 に追加で EMM 結果を通知, 4: 2 に追加で PSM 適用状態を通知 5: 3 に追加で PSM 適用状態を通知
CELLULAR_CFG_ATC_RETRY_CGATT ※デフォルトは“600”	アクセスポイントへの接続確認のリトライ回数です。 リトライ間隔は 1 秒です。”0~65535”の範囲で設定してください。
CELLULAR_CFG_EX_TIMEOUT ※デフォルトは“0”	TCP 接続タイムアウト時間を設定します。 単位は秒です。“0~65535”の範囲で設定してください。”0” はタイムアウトなしとなります。
CELLULAR_CFG_SCI_PRIORITY ※デフォルトは“4”	Cellular モジュールと通信を行うシリアルモジュールの割り込み優先度を設定します。システムの優先度に合わせて”2~15”の範囲で設定してください。
CELLULAR_CFG_SEMAPHORE_BLOCK_TIME ※デフォルトは“15000”	各関数の干渉を防ぐための、API の最大実行待ち時間を設定します。 単位はミリ秒です。“1~15000”の範囲で設定してください。
CELLULAR_CFG_PSM_PREPARATION_TIME ※デフォルトは“100”	Cellular モジュールが PSM へ移行するまでの時間です。 単位はミリ秒です。“100~10000”の範囲で設定してください。
CELLULAR_CFG_PSM_WAKEUP_LATENCY ※デフォルトは“5000”	PSM からの復帰完了までの最大時間です。 単位はミリ秒です。“0~10000”の範囲で設定してください。
CELLULAR_CFG_RING_LINE_ACTIVE_TIME ※デフォルトは“1000”	Cellular モジュールの RING 端子作動時間を設定します。 単位はミリ秒です。“1000~5000”の範囲で設定してください。
CELLULAR_CFG_UPGRADE_TIME ※デフォルトは“60”	R_CELLULAR_FirmUpgradeBlocking()実行時の最大 FW 更新完了待ち時間です。単位は分です。“1~600”の範囲で設定してください。
CELLULAR_CFG_URC_CHARGET_ENABLED ※デフォルトは“0”	Cellular モジュールから通知される Unsolicited Result Code（URC）を受信した時に呼び出されるコールバック関数の使用/不使用を設定します。 0: 使用しない, 1: 使用する
CELLULAR_CFG_URC_CHARGET_FUNCTION ※デフォルトは “my_sw_urc_charget_function”	Cellular モジュールから通知される URC を受信した時に呼び出されるコールバック関数名を設定します。

CELLULAR_CFG_DEBUGLOG ※デフォルトは “0”	ログ情報の出力設定を行います。ログ情報出力設定 1~4 は、FreeRTOS logging task で使用できます。必要に応じて”0~4”の範囲で設定してください。 0: OFF, 1: エラーログを出力, 2: 追加で警告を出力, 3: 追加で状態の通知を出力, 4: 追加で Cellular モジュールとの通信情報を出力
CELLULAR_CFG_UART_SCI_CH ※デフォルトは “6”	Cellular モジュールと通信をする SCI ポート番号を指定します。 デフォルト値は SCI ポート番号 6 を使用する場合です。制御する SCI ポートに合わせて設定してください。
CELLULAR_CFG_RESET_SIGNAL_LOGIC ※デフォルトは “1”	Cellular モジュールに対するリセット信号の出力形式を変更します。 デフォルト値はリセット信号を High 出力とする場合です。
CELLULAR_CFG_START_BIT_EDGE ※デフォルトは “0”	RXD 端子のスタートビット検出方法を設定します。 0: Low レベルで検出する, 1: 立ち下がりで検出する
CELLULAR_CFG_CTS_SW_CTRL ※デフォルトは “0”	Cellular モジュールの RTS 端子制御方法を設定します。 0: CTS 端子をハードウェアフロー制御、RTS 端子を本 FIT モジュールが GPIO によって制御する 1: RTS 端子をハードウェアフロー制御、CTS 端子を本 FIT モジュールが GPIO によって制御する
CELLULAR_CFG_CTS_PORT ※デフォルトは “J”	Cellular モジュールの CTS 端子を制御する汎用ポートの PDR(ポート方向レジスタ)を設定します。デフォルト値はポート J3 を使用する場合です。制御するポートに合わせて設定してください。本設定は、CELLULAR_CFG_CTS_SW_CTRL が 1 に設定されている場合のみ有効です。
CELLULAR_CFG_CTS_PIN ※デフォルトは “3”	Cellular モジュールの CTS 端子を制御する汎用ポートの PODR(ポート出力データレジスタ)を設定します。デフォルト値はポート J3 を使用する場合です。制御するポートに合わせて設定してください。本設定は、CELLULAR_CFG_CTS_SW_CTRL が 1 に設定されている場合のみ有効です。
CELLULAR_CFG_PFS_SET_VALUE ※デフォルトは “0x0BU”	Cellular モジュールの RTS 端子を制御するマイコン端子の周辺機能を選択する PFS(端子機能制御レジスタ)設定値です。使用する端子に合わせて設定してください。本設定は、CELLULAR_CFG_CTS_SW_CTRL が 1 に設定されている場合のみ有効です。
CELLULAR_CFG_RTS_PORT ※デフォルトは “0”	Cellular モジュールの RTS 端子を制御する汎用ポートの PDR(ポート方向レジスタ)を設定します。デフォルト値はポート 02 を使用する場合です。制御するポートに合わせて設定してください。
CELLULAR_CFG_RTS_PIN ※デフォルトは “2”	Cellular モジュールの RTS 端子を制御する汎用ポートの PODR(ポート出力データレジスタ)を設定します。デフォルト値はポート 02 を使用する場合です。制御するポートに合わせて設定してください。
CELLULAR_CFG_RESET_PORT ※デフォルトは “5”	Cellular モジュールの PWD_L 端子を制御する汎用ポートの PDR(ポート方向レジスタ)を設定します。デフォルト値はポート 55 を使用する場合です。制御するポートに合わせて設定してください。
CELLULAR_CFG_RESET_PIN ※デフォルトは “5”	Cellular モジュールの PWD_L 端子を制御する汎用ポートの PODR(ポート出力データレジスタ)を設定します。デフォルト値はポート 55 を使用する場合です。制御するポートに合わせて設定してください。
CELLULAR_CFG_IRQ_NUM ※デフォルトは “5”	Cellular モジュールの RING 端子の出力信号を割り込みとして使用する IRQ 番号を指定します。デフォルト値は、IRQ5 を使用します。使用するマイコンに合わせて設定してください。

本 FIT モジュールが使用する SCI FIT モジュールのコンフィグレーションオプションの設定は、r_sci_rx_config.h で行います。

SCI FIT モジュールに対する設定オプション名および設定値に関する説明を、表 2.2 に示します。オプションの詳細は、「RX ファミリ SCI モジュール Firmware Integration Technology (R01AN1815)」を参照してください。

表 2.2 Configuration options (r_sci_rx_config.h)

Configuration options in r_sci_rx_config.h	
SCI_CFG_CHx_INCLUDED ※1. CHx = CH0～CH12 ※2. 各デフォルト値は以下のとおり: CH0=1、CH1～CH12: 0	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。ボードに合わせて対応するチャネルに 1 を設定してください。
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。 CELLULAR_CFG_UART_SCI_CH で指定したチャネルに対応するバッファサイズを 2048 に設定してください。
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。 CELLULAR_CFG_UART_SCI_CH で指定したチャネルに対応するバッファサイズを 2048 に設定してください。
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。本 FIT モジュールではシリアル送信完了割り込みを使用するため、”1”を設定してください。

本 FIT モジュールが使用する IRQ FIT モジュールのコンフィグレーションオプションの設定は、r_irq_rx_config.h で行います。
IRQ FIT モジュールに対する設定オプション名および設定値に関する説明を、
表 2.3 に示します。オプションの詳細は、「RX ファミリ IRQ モジュール Firmware Integration Technology (R01AN1668)」を参照してください。

表 2.3 Configuration options (r_irq_rx_config.h)

Configuration options in r_sci_rx_config.h	
IRQ_CFG_FILT_EN_IRQx ※1. IRQx = IRQ0～IRQ15 ※2. 各デフォルト値は”0”	IRQ として使用するチャネルを指定します。 Cellular モジュールの RING 端子と接続するチャネルに”1”を設定してください。

本 FIT モジュールが使用する BSP FIT モジュールのコンフィグレーションオプションの設定は、r_bsp_config.h で行います。

BSP FIT モジュールに対するオプション名および設定値に関する説明を、表 2.4 に示します。オプションの詳細は、「RX ファミリ ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)」を参照してください。

表 2.4 Configuration options (r_bsp_config.h)

Configuration options in r_bsp_config.h	
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”

本 FIT モジュールが使用する RTOS のコンフィグレーションオプションの設定は、src/frtos_config/FreeRTOSConfig.h で行います。

FreeRTOS に対するオプション名および設定値に関する説明を表 2.5 に示します。使用する RTOS に応じた設定を行ってください。

表 2.5 Configuration options (FreeRTOSConfig.h)

Configuration options in FreeRTOSConfig.h	
configTICK_RATE_HZ ※デフォルト値は"(TickType_t)1000 "	RTOS tick 割り込み周期を設定します。 本 FIT モジュールを使用する場合は、"(TickType_t)1000"に設定してください。

2.8　コードサイズ

本 FIT モジュールの ROM サイズ、RAM サイズ、最大使用スタックサイズを表 2.6 に示します。

ROM (コードおよび定数) と RAM (グローバルデータ) のサイズは、ビルド時の「2.7 コンパイル時の設定」のコンフィギュレーションオプションによって決まります。

表 2.6 に示す値は、下記条件で確認しています。

FIT モジュールリビジョン: r_cellular rev1.12
コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.06.00
(統合開発環境のデフォルト設定に “-lang = c99“オプションを追加)
コンフィグレーションオプション: デフォルト設定

表 2.6 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 RX72N	ROM	約 36k バイト	-
	RAM	約 600 バイト	-
	最大使用スタックサイズ	約 700 バイト	-

2.9　引数

API 関数の引数に以下の構造体が使用されています。これらの構造体は、r_cellular_if.h で定義されています。

管理用構造体 (すべての API で使用)

- st_cellular_ctrl_t

コンフィグ構造体 (R_CELLULAR_Open()で使用)

- st_cellular_cfg_t (表 3.1 を参照)

アクセスポイント接続設定構造体 (R_CELLULAR_APConnect()で使用)

- st_cellular_ap_cfg_t (表 3.4 を参照)

IP アドレス取得構造体 (R_CELLULAR_DnsQuery()、R_CELLULAR_GetPDPAddress()で使用)

- st_cellular_ipaddr_t (表 3.5 を参照)

時刻設定・取得構造体 (R_CELLULAR_GetTime()、R_CELLULAR_SetTime()で使用)

- st_cellular_datetime_t (表 3.6 を参照)

eDRX 設定・取得構造体 (R_CELLULAR_SetEDRX()で使用)

- st_cellular_edrx_config_t (表 3.7 を参照)

PSM 設定・取得構造体 (R_CELLULAR_SetPSM()で使用)

- st_cellular_psm_config_t (表 3.8 を参照)

ICCID 取得構造体 (R_CELLULAR_GetICCID()で使用)

- st_cellular_iccid_t (表 3.9 を参照)

IMEI 取得構造体 (R_CELLULAR_GetIMEI()で使用)

- st_cellular_imei_t (表 3.10 を参照)

IMSI 取得構造体 (R_CELLULAR_GetIMSI()で使用)

- st_cellular_imsi_t (表 3.11 を参照)

電話番号取得構造体 (R_CELLULAR_GetPhonenum()で使用)

- st_cellular_phonenum_t (表 3.12 を参照)

RSSI 取得構造体 (R_CELLULAR_GetRSSI()で使用)

- st_cellular_rssi_t (表 3.13 を参照)

SVN 取得構造体 (R_CELLULAR_GetSVN()で使用)

- st_cellular_svn_t (表 3.14 を参照)

PING 設定構造体 (R_CELLULAR_Ping()で使用)

- st_cellular_ping_cfg_t (表 3.15 を参照)

アクセスポイント接続状態取得構造体 (R_CELLULAR_GetAPConnectState()で使用)

- st_cellular_notice_t (表 3.16 を参照)

FOTA 状態取得構造体 (R_CELLULAR_GetUpgradeState()で使用)

- st_cellular_updatestate_t (表 3.18 を参照)

証明書・秘密鍵取得構造体 (R_CELLULAR_GetCertificate()で使用)

- st_cellular_certificate_t (表 3.19 を参照)

2.10 戻り値

API 関数の戻り値に以下の列挙型が使用されています。この列挙型は、r_cellular_if.h で定義されています。

API エラーコード

- e_cellular_err_t

2.11 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)、(5)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) e^2 studio 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
e^2 studio のスマート・コンフィグレータを使用して、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「RX スマート・コンフィグレータ ユーザーガイド: e^2 studio 編 (R20AN0451)」を参照してください。

(2) e^2 studio 上で FIT コンフィグレータを使用して FIT モジュールを追加する場合
e^2 studio の FIT コンフィグレータを使用して、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 e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)」を参照してください。

(3) CS+上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
CS+上で、スタンドアロン版スマート・コンフィグレータを使用して、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「RX スマート・コンフィグレータ ユーザーガイド: CS+編 (R20AN0470)」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。

(5) IAREW 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
スタンドアロン版スマート・コンフィグレータを使用して、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「RX スマート・コンフィグレータ ユーザーガイド: IAREW 編 (R20AN0535)」を参照してください。

2.12 for 文、while 文、do while 文および FreeRTOS の portMAX_DELAY について

本 FIT モジュールでは、レジスタの反映待ち処理などで for 文、while 文、do while 文 (ループ処理) を使用しています。これらループ処理には、「WAIT_LOOP」をキーワードとしたコメントを記述しています。そのため、ループ処理にユーザがフェイルセーフの処理を組み込む場合は、「WAIT_LOOP」で該当の処理を検索できます。

また、FreeRTOS のセマフォなどのリソース確保やイベント待ちになどの処理完了までの待ち時間が発生する API に対して、「portMAX_DELAY」を使用してタイムアウトなしに設定している場合があります。これら API に対してタイムアウト時間を設定する場合は、「portMAX_DELAY」で検索して該当箇所を適切なタイムアウト時間に修正してください。

以下に記述例を示します。

while 文の例：

```
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}
```

for 文の例：

```
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}
```

do while 文の例：

```
/* Reset completion waiting */
do
{
    reg = phy_read(ether_channel, PHY_REG_CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

2.13 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。

3. API 関数

3.1 R_CELLULAR_Open()

RYZ014A Cellular FIT モジュールと Cellular モジュールの初期化を行います。

Format

```
e_cellular_err_t R_CELLULAR_Open (
    st_cellular_ctrl_t * const p_ctrl,
    st_cellular_cfg_t * const p_cfg
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
p_cfg (IN)	*ユーザが宣言した st_cellular_cfg_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_ALREADY_OPEN	*/* R_CELLULAR_Open が実行済み */*
CELLULAR_ERR_SERIAL_OPEN	*/* シリアルの初期化に失敗 */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_SEMAPHORE_INIT	*/* セマフォの初期化に失敗 */*
CELLULAR_ERR_EVENT_GROUP_INIT	*/* イベントグループの初期化に失敗 */*
CELLULAR_ERR_CREATE_TASK	*/* タスクの作成に失敗 */*
CELLULAR_ERR_MEMORY_ALLOCATION	*/* メモリの割り当てに失敗 */*
CELLULAR_ERR_RECV_TASK	*/* シリアル受信に失敗 */*

Properties

r_cellular_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 Cellular モジュールを初期化し、無線通信の準備を行います。

本 API の実行前に、引数 *p_ctrl* へ設定する st_cellular_ctrl_t 構造体をゼロで初期化してください。

引数 *p_cfg* へユーザが定義した st_cellular_cfg_t 構造体を設定した場合は、格納されているパラメータを使用します。st_cellular_cfg_t 構造体のメンバを、表 3.1 に示します。

引数 *p_cfg* へ NULL を設定した場合は、本 FIT モジュールが持つ初期値および Smart Configurator で設定した値を使用します。引数 *p_cfg* へ NULL を設定した場合に使用される初期値は、以下の通りです。

＜baud_rate = 921600 / ap_cgatt_retry_count = 600 / sci_timeout = 60000 / tx_process_size = 1500
/ rx_process_size = 1500 / packet_data_size = 0 / exchange_timeout = 0
/ connect_timeout = 200 / send_timeout = 10 / creatable_socket = 6＞

sim_pin_code の設定値は、CELLULAR_CFG_PIN_CODE が使用されます。

Smart Configurator でコールバック関数を使用するように設定している場合、コールバック関数で Cellular モジュールから通知される Unsolicited Result Code (URC)などの AT コマンドレスポンスを受け取ることができます。 (表 2.1 参照)

Cellular モジュールが予期せず再起動し、+SYSSTART URC を通知する場合があります。その場合は、ユーザは復帰処理を実行する必要があります。詳細は、4.3.2 節を参照してください。ユーザは、復帰処理を実行するためにコールバック関数で+SYSSTART URC を検出することを推奨します。

表 3.1 コンフィグ構造体のメンバ

Members in st_cellular_cfg_t structure		
uint8_t	sim_pin_code[]	SIM の PIN コード
uint32_t	baud_rate	モジュールとの通信ボーレート
uint16_t	ap_cgatt_retry_count	AP への接続リトライ上限回数
uint32_t	sci_timeout	マイコンとの通信タイムアウト設定
uint16_t	tx_process_size	Cellular モジュールへ 1 回で送るデータサイズ
uint16_t	rx_process_size	Cellular モジュールから 1 回で受け取るデータサイズ
uint16_t	packet_data_size	1 パケット当たりのデータサイズ
uint16_t	exchange_timeout	接続タイムアウト
uint16_t	connect_timeout	ソケット接続タイムアウト
uint8_t	send_timeout	パケット送信タイムアウト
uint8_t	creatable_socket	ソケット作成可能数

Reentrant

不可

Thread Safety

非対応

Examples

【デフォルトのコンフィグ値を使用する場合】

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
```

【コンフィグ値を設定する場合】

```
/* コールバック関数例 (関数名がデフォルト設定の場合) */
uint8_t portBuffer[2049] = {'¥0'};
void my_sw_urc_charget_function(void * p_arg)
{
    //URC を取得
    sprintf((char *)portBuffer, "%.2048s", (char *)p_arg);
}

e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_cfg_t cellular_cfg = {
                        "0000",      //SIM の PIN コード
                        921600,      //モジュールとの通信ボーレート(921600 を推奨)
                        600,         //AP への接続確認リトライ上限回数
                        0xffff,      //マイコンとの通信タイムアウト設定
                        100,         //Cellular モジュールへ 1 回で送るデータサイズ
                        100,         //Cellular モジュールから 1 回で受け取るデータサイズ
                        100,         //1 パケット当たりのデータサイズ
                        100,         //接続タイムアウト
                        100,         //ソケット接続タイムアウト
                        100,         //パケット送信タイムアウト
                        3};          //ソケット作成可能数

ret = R_CELLULAR_Open(&cellular_ctrl, &cellular_cfg);
```

Special Notes

なし

3.2　R_CELLULAR_Close()

RYZ014A Cellular モジュールとの通信を終了し、Cellular モジュールをシャットダウンします。

Format

```
e_cellular_err_t R_CELLULAR_Close (
    st_cellular_ctrl_t * const p_ctrl
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*
CELLULAR_ERR_RECV_TASK	*/* シリアル受信に失敗 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールとの通信を切断し、Cellular モジュールをシャットダウンします。

アクセスポイントへ接続していた場合は、アクセスポイントから切断します。ソケット通信中の場合は、ソケット接続を切断およびアクセスポイントから切断します。

本 API が戻り値でエラーを返した場合は、Cellular モジュールのシャットダウンが完了していません。その場合は、戻り値に応じて以下の処理を実行してください。

(1) 本 API が戻り値で CELLULAR_ERR_OTHER_API_RUNNING を返した場合は、本 API を再度実行してください。

(2) 本 API が戻り値で以下の値を返した場合は、R_CELLULAR_Open()を実行後に本 API を再度実行してください。

- CELLULAR_ERR_MODULE_COM
- CELLULAR_ERR_MODULE_TIMEOUT
- CELLULAR_ERR_RECV_TASK

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_Close(&cellular_ctrl);

/* R_CELLULAR_Close()が戻り値でエラーを返した場合 */
if ((CELLULAR_ERR_MODULE_COM == ret) ||
    (CELLULAR_ERR_MODULE_TIMEOUT == ret) ||
    (CELLULAR_ERR_RECV_TASK == ret))
{
    ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
    ret = R_CELLULAR_Close(&cellular_ctrl);
}
else if (CELLULAR_ERR_OTHER_API_RUNNING == ret)
{
    ret = R_CELLULAR_Close(&cellular_ctrl);
}
```

Special Notes

本 API は、AT コマンドを実行します。そのため、API でタイムアウトが発生した後は、本 API の処理が正常に完了しない場合があります。API でタイムアウトが発生した後は、本 API を実行する前に R_CELLULAR_HardwareReset()を実行してください。 (3.41 節および 4.3.3 節を参照)

3.3　R_CELLULAR_APConnect()

RYZ014A Cellular モジュールをアクセスポイントへ接続します。

Format

```
e_cellular_err_t R_CELLULAR_APConnect (
    st_cellular_ctrl_t * const p_ctrl,
    const st_cellular_ap_cfg_t * const p_ap_cfg
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
p_ap_cfg (IN)	*ユーザが宣言した st_cellular_ap_cfg_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_ALREADY_CONNECT	*/* アクセスポイントに接続済み */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*
CELLULAR_ERR_AP_CONNECT_FAILED	*/* アクセスポイントへの接続に失敗 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールをアクセスポイントへ接続します。

アクセスポイントへ接続するには、本関数実行前に、以下のいずれかの方法でアクセスポイント情報などの必要な情報を設定する必要があります。

(1) r_cellular_config.h の表 3.2 に示すマクロへ適切な値を設定する。(詳細は、表 2.1 を参照)

※ 引数 *p_ap_cfg* へ NULL を設定してください。
※ マクロの設定は Smart Configurator から実施し、マクロを直接編集しないでください。

(2) R_CELLULAR_Open()実行時に、表 3.3 に示すメンバへ適切な値を設定した st_cellular_cfg_t 構造体ポインタを第 2 引数へ設定する。そして、R_CELLULAR_APConnect()実行時に、表 3.4 に示すメンバへ適切な値を設定した st_cellular_ap_cfg_t 構造体ポインタを第 2 引数へ設定する。

アクセスポイントへの接続確認のリトライは、CELLULAR_CFG_ATC_RETRY_GATT マクロの値または st_cellular_cfg_t 構造体のメンバ ap_cgatt_retry_count の値の回数まで 1 秒間隔で実行されます。アクセスポイントへの接続完了まで、数分またはそれ以上かかる場合があります。接続完了までの時間は、使用する SIM および通信状況によって変わります。

アクセスポイントへ正常に接続した場合、自動的にネットワーク時刻が取得され、Cellular モジュールの不揮発メモリで保持されます。時刻の取得を行う場合は R_CELLULAR_GetTime()を実行してください。また、アクセスポイント接続時に割り当てられた IP アドレスが、引数 *p_ctrl* へ設定した st_cellular_ctrl_t 構造体のメンバ pdp_addr へ格納されます。

本 API を使用する前に R_CELLULAR_Open()を実行して本 FIT モジュールおよび Cellular モジュールを初期化してください。実行されていない場合は、戻り値で CELLULAR_ERR_NOT_OPEN を返します。

表 3.2 アクセスポイント接続に必要なマクロ

Configuration options in r_cellular_config.h	
CELLULAR_CFG_AP_NAME	接続先アクセスポイント名
CELLULAR_CFG_AP_USERID	接続先アクセスポイントのユーザ名
CELLULAR_CFG_AP_PASSWORD	接続先アクセスポイントのパスワード
CELLULAR_CFG_PIN_CODE	使用する SIM の PIN コード
CELLULAR_CFG_AUTH_TYPE	認証プロトコル (0: None、1: PAP、2: CHAP)
CELLULAR_CFG_ATC_RETRY_CGATT	アクセスポイントへの接続確認リトライ回数上限

表 3.3 アクセスポイント接続に必要なメンバ (R_CELLULAR_Open())

Members in st_cellular_cfg_t structure		
uint8_t	sim_pin_code[]	使用する SIM の PIN コード
uint16_t	ap_cgatt_retry_count	アクセスポイントへの接続確認リトライ回数上限

表 3.4 アクセスポイント接続に必要なメンバ (R_CELLULAR_APConnect())

Members in st_cellular_ap_cfg_t structure		
uint8_t	ap_name[]	接続先アクセスポイント名
uint8_t	ap_user_name[]	接続先アクセスポイントのユーザ名
uint8_t	ap_pass[]	接続先アクセスポイントのパスワード
e_cellular_auth_type_t	auth_type	認証プロトコル (0: None、1: PAP、2: CHAP)

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
```

Special Notes

R_CELLULAR_FactoryReset()を実行後または使用する SIM の交換後の初回接続時は、アクセスポイントへの接続まで 10 分以上かかる場合があります。本 API が戻り値で CELLULAR_ERR_AP_CONNECT_FAILED を返した場合は、CELLULAR_CFG_ATC_RETRY_CGATT または st_cellular_cfg_t 構造体のメンバ ap_cgatt_retry_count の値を大きくして接続完了までの待ち時間を長くしてください。

R_CELLULAR_SetBand()で使用する LTE バンドを設定により限定することで、アクセスポイントへの接続待ち時間を短くすることができます。詳細は、3.30 節を参照してください。

本関数でのアクセスポイントへの接続確認のリトライを中断する場合は、引数 *p_ctrl* へ設定されている st_cellular_ctrl_t 構造体のメンバ ap_connect_retry の値を、本関数を実行しているタスク以外のタスクから 0 に変更してください。Cellular モジュールおよび本 FIT モジュールを使用したシステムを非常停止する場合に使用できます。

3.4　R_CELLULAR_IsConnected()

本 FIT モジュールのアクセスポイント接続状態を取得します。

Format

```
e_cellular_err_t R_CELLULAR_IsConnected (
    st_cellular_ctrl_t * const p_ctrl
)
```

Parameters

p_ctrl (IN/OUT)　　*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* アクセスポイントへ接続中 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_NOT_CONNECT	*/* アクセスポイントに接続していない */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

本 FIT モジュールのアクセスポイントへの接続状態を確認します。

Reentrant

可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_IsConnect(&cellular_ctrl);
```

Special Notes

なし

3.5　R_CELLULAR_Disconnect()

Cellular モジュールをアクセスポイントから切断します。

Format

```
e_cellular_err_t R_CELLULAR_Disconnect (
    st_cellular_ctrl_t * const p_ctrl
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールをアクセスポイントから切断します。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
ret = R_CELLULAR_Disconnect(&cellular_ctrl);
```

Special Notes

本 API 実行後は、R_CELLULAR_Close()を実行してください。

3.6　R_CELLULAR_CreateSocket()

ソケットを生成します。

Format

```
int32_t R_CELLULAR_CreateSocket (
    st_cellular_ctrl_t * const p_ctrl,
    const uint8_t protocol_type,
    const uint8_t ip_version
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
protocol_type (IN)	*プロトコルタイプ（TCP = 6 を指定）*
ip_version (IN)	*IP バージョン（IPv4 = 4、IPv6 = 6 を指定）*

Return values

1 ～ 6 のいずれかの値	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_SOCKET_CREATE_LIMIT	*/* ソケット作成数が限界を超えた */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*
CELLULAR_ERR_SIM_NOT_SUPPORT_IPV6	*/* SIM が IPv6 非対応 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

利用可能なソケットに対して、プロトコルタイプおよび IP バージョンの設定を行います。引数 *protocol_type* へ 6 (TCP)を、引数 *ip_version* へ 4 (IPv4)または 6 (IPv6)を指定してください。IPv6 非対応の SIM を使用する場合、引数 *ip_version* へ 6 (IPv6)を指定すると戻り値で CELLULAR_ERR_SIM_NOT_SUPPORT_IPV6 エラーを返します。

処理が正常終了した場合はソケットが生成され、番号を戻り値で返します。生成されるソケットの番号は 1 から 6 までの整数値です。

引数 *protocol_type* へ設定可能な値は、以下の通りです。

```
#define CELLULAR_PROTOCOL_TCP          (6)      // TCP
```

引数 *ip_version* へ設定可能な値は、以下の通りです。

```
#define CELLULAR_PROTOCOL_IPV4         (4)      // IPv4
#define CELLULAR_PROTOCOL_IPV6         (6)      // IPv6
```

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
int32_t socket_no;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
socket_no = R_CELLULAR_CreateSocket (&cellular_ctrl, CELLULAR_PROTOCOL_TCP,
                                     CELLULAR_PROTOCOL_IPV4);
```

Special Notes

なし

3.7　R_CELLULAR_ConnectSocket()

指定した IP アドレスおよびポートへ接続します。

Format

```
e_cellular_err_t R_CELLULAR_ConnectSocket (
    st_cellular_ctrl_t * const p_ctrl,
    const uint8_t socket_no,
    const uint8_t * const p_ip_addr,
    const uint16_t port
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
socket_no (IN)	*ソケット番号*
p_ip_addr (IN)	*接続先 IP アドレスへのポインタ（文字列）*
	IPv4 の場合はピリオドで区切った 10 進数形式で指定
	（例：”104.120.11.132”）
	IPv6 の場合はコロンで区切った 16 進数形式で指定
	（例：”1234:5678:9ABC:DEF0:1234:5678:9ABC:DEF0”）
	ホスト名で指定することも可能
	（例：”www.renesas.com”）
port *(IN)*	*接続先ポート番号*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_NOT_CONNECT	*/* アクセスポイントに接続していない */*
CELLULAR_ERR_ALREADY_SOCKET_CONNECT	*/* ソケット接続済み */*
CELLULAR_ERR_SOCKET_NOT_READY	*/* ソケットのステータスが異常 */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

引数 *socket_no* へ指定したソケットを使用して、引数 *p_ip_addr* へ設定した IP アドレスまたはホスト名、引数 *port* へ指定したポート番号に接続します。IP アドレスまたはホスト名は文字列で指定してください。

本 API を使用する前に、R_CELLULAR_CreateSocket()を実行して利用するソケットを作成してください。実行されていない場合は、戻り値で CELLULAR_ERR_SOCKET_NOT_READY を返します。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
int8_t socket_no;
uint16_t port_no = 33333;
uint8_t ip_adder[] = "104.120.11.132";

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
socket_no = R_CELLULAR_CreateSocket (&cellular_ctrl, CELLULAR_PROTOCOL_TCP,
                                     CELLULAR_PROTOCOL_IPV4);
if (0 < socket_no)
{
   ret = R_CELLULAR_ConnectSocket(&cellular_ctrl, socket_no, ip_adder,
                                  port_no);
}
```

Special Notes

なし

3.8　R_CELLULAR_ShutdownSocket()

ソケット通信を切断します。

Format

```
e_cellular_err_t R_CELLULAR_ShutdownSocket (
    st_cellular_ctrl_t * const p_ctrl,
    const uint8_t socket_no
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
socket_no (IN)	*ソケット番号*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_SOCKET_NOT_READY	*/* ソケットのステータスが異常 */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

引数 *socket_no* で指定したソケットの通信を切断します。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
int8_t socket_no;
uint16_t port_no = 33333;
uint8_t ip_adder[] = “104.120.11.132”;

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
socket_no = R_CELLULAR_CreateSocket (&cellular_ctrl, CELLULAR_PROTOCOL_TCP,
                                     CELLULAR_PROTOCOL_IPV4);
if (0 < socket_no)
{
   ret = R_CELLULAR_ConnectSocket(&cellular_ctrl, socket_no, ip_adder,
                                  port_no);
   ret = R_CELLULAR_ShutdownSocket(&cellular_ctrl, socket_no);
}
```

Special Notes

なし

3.9　R_CELLULAR_CloseSocket()

ソケットをクローズします。

Format

```
e_cellular_err_t R_CELLULAR_CloseSocket (
    st_cellular_ctrl_t * const p_ctrl,
    const uint8_t socket_no
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
socket_no (IN)	*ソケット番号*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_SOCKET_NOT_READY	*/* ソケットのステータスが異常 */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

引数 *socket_no* で指定したソケットをクローズします。ソケットが通信中だった場合は、ソケット接続を切断します。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
int8_t socket_no;
uint16_t port_no = 33333;
uint8_t ip_adder[] = “104.120.11.132”;

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
socket_no = R_CELLULAR_CreateSocket (&cellular_ctrl, CELLULAR_PROTOCOL_TCP,
                                     CELLULAR_PROTOCOL_IPV4);
if (0 < socket_no)
{
   ret = R_CELLULAR_ConnectSocket(&cellular_ctrl, socket_no, ip_adder,
                                  port_no);
   ret = R_CELLULAR_CloseSocket(&cellular_ctrl, socket_no);
}
```

Special Notes

なし

3.10 R_CELLULAR_SendSocket()

指定したソケットでデータ送信を実行します。

Format

```
e_cellular_err_t R_CELLULAR_SendSocket (
    st_cellular_ctrl_t * const p_ctrl,
    const uint8_t socket_no,
    uint8_t * const data,
    const int32_t length,
    const uint32_t timeout_ms
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
socket_no (IN)	*ソケット番号*
data (IN)	*送信データへのポインタ*
length (IN)	*送信データサイズ (1 以上)*
timeout_ms (IN)	*タイムアウト設定 (ms 単位、設定値=1~0xffffffff)*

Return values

送信バイト数	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_NOT_CONNECT	*/* アクセスポイントに接続していない */*
CELLULAR_ERR_SOCKET_NOT_READY	*/* ソケットのステータスが異常 */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

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

本 API を使用する前に R_CELLULAR_ConnectSocket()を実行してください。実行されていない場合は、戻り値で CELLULAR_ERR_SOCKET_NOT_READY を返します。

本 API が戻り値で CELLULAR_ERR_MODULE_TIMEOUT を返した場合、Cellular モジュールはそれ以降の AT コマンドに応答しない場合があります。その場合は、復帰処理として R_CELLULAR_HardwareReset()を実行してください。R_CELLULAR_HardwareReset()実行後の本 FIT モジュールの状態については、3.41 節を参照してください。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
int32_t socket_no;
int32_t recv_ret;
uint16_t port_no = 33333;
uint8_t ip_adder[] = “104.120.11.132”;

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
socket_no = R_CELLULAR_CreateSocket (&cellular_ctrl, CELLULAR_PROTOCOL_TCP,
                                     CELLULAR_PROTOCOL_IPV4);
if (0 < socket_no)
{
   ret = R_CELLULAR_ConnectSocket(&cellular_ctrl, socket_no, ip_adder,
                                  port_no);
   recv_ret = R_CELLULAR_SendSocket(&cellular_ctrl, socket_no, data, length,
                                    timeout);
}
```

Special Notes

なし

3.11 R_CELLULAR_ReceiveSocket()

指定したソケットでデータを受信します。

Format

```
int32_t R_CELLULAR_ReceiveSocket (
    st_cellular_ctrl_t * const p_ctrl,
    const uint8_t socket_no,
    uint8_t * const data,
    const int32_t length,
    const uint32_t timeout_ms
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
socket_no (IN)	*ソケット番号*
data (IN)	*受信データへのポインタ*
length (IN)	*受信データサイズ (1 以上)*
timeout_ms (IN)	*タイムアウト設定 (ms 単位、設定値=0~0xffffffff、0=タイムアウトなし)*

Return values

受信バイト数	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_SOCKET_NOT_READY	*/* ソケットのステータスが異常 */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

引数 *socket_no* で指定したソケットから、受信データ格納領域 *data* へ、引数 *length* で指定した受信バイト数だけデータを取得および格納します。受信データがない場合は、戻り値で 0 を返します。

本 API を使用する前に、R_CELLULAR_ConnectSocket()を実行してください。実行されていない場合は、戻り値で CELLULAR_ERR_SOCKET_NOT_READY を返します。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
int8_t socket_no;
uint16_t port_no = 33333;
uint8_t ip_adder[] = "104.120.11.132";

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
socket_no = R_CELLULAR_CreateSocket (&cellular_ctrl, CELLULAR_PROTOCOL_TCP,
                                     CELLULAR_PROTOCOL_IPV4);
if (0 < socket_no)
{
   ret = R_CELLULAR_ConnectSocket(&cellular_ctrl, socket_no, ip_adder,
                                  port_no);
   ret = R_CELLULAR_ReceiveSocket(&cellular_ctrl, socket_no, data, length,
                                  timeout);
}
```

Special Notes

なし

3.12 R_CELLULAR_DnsQuery()

DNS クエリを実行します。

Format

```
e_cellular_err_t R_CELLULAR_DnsQuery (
    st_cellular_ctrl_t * const p_ctrl,
    uint8_t *const domain_name,
    const uint8_t ip_version,
    st_cellular_ipaddr_t * const p_addr
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
domain_name (IN)	*ドメイン名の格納領域へのポインタ*
ip_version (IN)	*取得する IP アドレスの IP バージョン（IPv4 = 4、IPv6 = 6 を指定）*
p_addr (IN/OUT)	*ユーザが宣言した st_cellular_ipaddr_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_NOT_CONNECT	*/* アクセスポイントに接続していない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*
CELLULAR_ERR_SIM_NOT_SUPPORT_IPV6	*/* IPv6 非対応 */*

Properties

r_cellular_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、引数 *domain_name* で指定したドメインの、引数 *ip_version* で指定した IP バージョンの IP アドレスを取得し、引数 *p_addr* へ設定した st_cellular_ipaddr_t 構造体へ格納します。st_cellular_ipaddr_t 構造体のメンバを、表 3.5 に示します。

本 API を使用する前に、R_CELLULAR_APConnect()を実行してアクセスポイントへ接続してください。実行されていない場合は、戻り値で CELLULAR_ERR_NOT_CONNECT を返します。

IPv6 非対応の SIM を使用する場合、引数 *ip_version* で IPv6 を指定すると戻り値で CELLULAR_ERR_SIM_NOT_SUPPORT_IPV6 エラーを返します。

表 3.5 IP アドレス取得構造体のメンバ

Members in st_cellular_ipaddr_t structure		
uint8_t	ipv4[]	IPv4 アドレス
uint8_t	ipv6[]	IPv6 アドレス

Reentrant

可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_ipaddr_t addr = {0};
uint8_t domain_name[] = “www.renesas.com”;

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
ret = R_CELLULAR_DnsQuery (&cellular_ctrl, domain_name, CELLULAR_PROTOCOL_IPV4,
                           &addr);
```

Special Notes

なし

3.13 R_CELLULAR_GetTime()

RYZ014A Cellular モジュールが保持する日時情報を取得します。

Format

```
e_cellular_err_t R_CELLULAR_GetTime (
    st_cellular_ctrl_t * const p_ctrl,
    st_cellular_datetime_t * const p_time
)
```

Parameters

p_ctrl (IN/OUT)　　*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

p_time (IN/OUT)　　*取得した日時を格納する構造体へのポインタ*

Return values

CELLULAR_SUCCESS　　*/* 正常終了 */*

CELLULAR_ERR_PARAMETER　　*/* 引数が無効な値 */*

CELLULAR_ERR_NOT_OPEN　　*/* Open 関数を実行していない */*

CELLULAR_ERR_MODULE_COM　　*/* Cellular モジュールとの通信に失敗 */*

CELLULAR_ERR_MODULE_TIMEOUT　　*/* Cellular モジュールからの応答がない */*

CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING　　*/* 他の AT コマンドが実行中 */*

CELLULAR_ERR_OTHER_API_RUNNING　　*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールに設定されている日時情報を取得し、引数 *p_time* へ設定した st_cellular_datetime_t 構造体へ格納します。st_cellular_datetime_t 構造体のメンバを、表 3.6 に示します。

Cellular モジュールの起動時の日時情報は“70/01/01/00:00:00+00”(年/月/日/時刻:分:秒:タイムゾーン)となっています。R_CELLULAR_APConnect()を実行してネットワークへ接続すると、自動的にネットワーク時刻が取得されます。

表 3.6 時刻設定・取得構造体のメンバ

Members in st_cellular_datetime_t structure		
uint8_t	year	年 (2 桁)
uint8_t	month	月
uint8_t	day	日
uint8_t	hour	時間
uint8_t	min	分
uint8_t	sec	秒
int8_t	timezone	タイムゾーン

Reentrant

可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_datetime_t cellular_time = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_GetTime(&cellular_ctrl, &cellular_time);
```

Special Notes

なし

3.14 R_CELLULAR_SetTime()

RYZ014A Cellular モジュールへ、日時情報を設定する関数です。

Format

```
e_cellular_err_t R_CELLULAR_SetTime (
    st_cellular_ctrl_t * const p_ctrl,
    const st_cellular_datetime_t * const p_time
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
p_time (IN/OUT)	*取得した日時を格納する構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

引数 *p_time* へ設定した st_cellular_datetime_t 構造体に格納されている日時情報を Cellular モジュールへ設定します。st_cellular_datetime_t 構造体のメンバについては、表 3.6 を参照してください。

本関数をアクセスポイント接続前に実行した場合、本関数で設定した日時情報はアクセスポイント接続時に自動的に取得されたネットワーク時刻で上書きされます。

Reentrant

可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_datetime_t cellular_time = {21,8,1,12,34,56,36};
                                       //2021 年 8 月 1 日 12 時 34 分 56 秒+36
                                       //タイムゾーンは 15 分単位

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_SetTime(&cellular_ctrl, &cellular_time);
```

Special Notes

なし

3.15 R_CELLULAR_SetEDRX()

RYZ014A Cellular モジュールへ、eDRX (extended Discontinuous Reception)のパラメータ設定を行います。

Format

```
e_cellular_err_t R_CELLULAR_SetEDRX (
    st_cellular_ctrl_t * const p_ctrl,
    const st_cellular_edrx_config_t * const p_config,
    st_cellular_edrx_config_t * const p_result
)
```

Parameters

p_ctrl (IN/OUT)　　*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

p_config (IN)　　*ユーザが宣言した st_cellular_edrx_config_t 構造体へのポインタ (設定)*

p_result (IN/OUT)　　*ユーザが宣言した st_cellular_edrx_config_t 構造体へのポインタ (設定値確認)*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールへ、eDRX のパラメータ設定を行います。引数 *p_config* へ設定した st_cellular_edrx_config_t 構造体に格納されている eDRX パラメータが設定されます。st_cellular_edrx_config_t 構造体のメンバを、表 3.7 に示します。

引数 *p_result* へ設定した st_cellular_edrx_config_t 構造体で、適用される eDRX 周期および Paging Time Window (PTW)を確認できます。

表 3.7 eDRX 設定・取得構造体のメンバ

Members in st_cellular_edrx_config_t structure		
e_cellular_edrx_mode_t	edrx_mode	eDRX モード
e_cellular_edrx_cycle_t	edrx_cycle	eDRX 周期
e_cellular_ptw_cycle_t	ptw_cycle	PTW 周期

st_cellular_edrx_config_t 構造体のメンバ edrx_mode へ設定可能な値は、下記の通りです。

```
typedef enum
{
    CELLULAR_EDRX_MODE_INVALID = 0,              // Disable the edrx function
    CELLULAR_EDRX_MODE_ACTIVE = 1,               // Enable the edrx function
    CELLULAR_EDRX_MODE_ACTIVE_RESULT = 2,  // Activate the edrx function and return the results
    CELLULAR_EDRX_MODE_INIT = 3                  // Initialize and disable the edrx function
} e_cellular_edrx_mode_t;
```

st_cellular_edrx_config_t 構造体のメンバ edrx_cycle へ設定可能な値は、下記の通りです。

```
typedef enum
{
    CELLULAR_EDRX_CYCLE_5_SEC = 0,     // edrx cycle (5.12sec)
    CELLULAR_EDRX_CYCLE_10_SEC,        // edrx cycle (10.24sec)
    CELLULAR_EDRX_CYCLE_20_SEC,        // edrx cycle (20.48sec)
    CELLULAR_EDRX_CYCLE_40_SEC,        // edrx cycle (40.96sec)
    CELLULAR_EDRX_CYCLE_81_SEC,        // edrx cycle (81.92sec)
    CELLULAR_EDRX_CYCLE_163_SEC,       // edrx cycle (163.84sec)
    CELLULAR_EDRX_CYCLE_327_SEC,       // edrx cycle (327.68sec)
    CELLULAR_EDRX_CYCLE_655_SEC,       // edrx cycle (655.36sec)
    CELLULAR_EDRX_CYCLE_1310_SEC,      // edrx cycle (1,310.72sec)
    CELLULAR_EDRX_CYCLE_2621_SEC       // edrx cycle (2,621.44sec)
} e_cellular_edrx_cycle_t;
```

st_cellular_edrx_config_t 構造体のメンバ ptw_cycle へ設定可能な値は、下記の通りです。

```
typedef enum
{
    CELLULAR_PTW_CYCLE_1_SEC,          // PTW (1.28sec)
    CELLULAR_PTW_CYCLE_2_SEC,          // PTW (2.56sec)
    CELLULAR_PTW_CYCLE_4_SEC,          // PTW (3.84sec)
    CELLULAR_PTW_CYCLE_5_SEC,          // PTW (5.12sec)
    CELLULAR_PTW_CYCLE_6_SEC,          // PTW (6.40sec)
    CELLULAR_PTW_CYCLE_7_SEC,          // PTW (7.68sec)
    CELLULAR_PTW_CYCLE_9_SEC,          // PTW (8.96sec)
    CELLULAR_PTW_CYCLE_10_SEC,         // PTW (10.24sec)
    CELLULAR_PTW_CYCLE_11_SEC,         // PTW (11.52sec)
    CELLULAR_PTW_CYCLE_13_SEC,         // PTW (12.80sec)
    CELLULAR_PTW_CYCLE_14_SEC,         // PTW (14.08sec)
    CELLULAR_PTW_CYCLE_15_SEC,         // PTW (15.36sec)
    CELLULAR_PTW_CYCLE_16_SEC,         // PTW (16.64sec)
    CELLULAR_PTW_CYCLE_18_SEC,         // PTW (17.92sec)
    CELLULAR_PTW_CYCLE_19_SEC,         // PTW (19.20sec)
    CELLULAR_PTW_CYCLE_20_SEC          // PTW (20.48sec)
} e_cellular_ptw_cycle_t;
```

Reentrant

不可

Thread Safety

対応

Examples

eDRX 周期を 20 秒、PTW を 2 秒に設定する場合 (設定値の詳細は、r_cellular_if.h のコメントを参照)

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_edrx_config_t edrx_cfg = {CELLULAR_EDRX_MODE_ACTIVE,
                                      CELLULAR_EDRX_CYCLE_20_SEC,
                                      CELLULAR_PTW_CYCLE_2_SEC};
st_cellular_edrx_config_t edrx_ret = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);

/* 適用されたパラメータを確認しない場合 */
ret = R_CELLULAR_SetEDRX(&cellular_ctrl, &edrx_cfg, NULL);

/* 適用されたパラメータを確認する場合 */
ret = R_CELLULAR_SetEDRX(&cellular_ctrl, &edrx_cfg, &edrx_ret);
```

Special Notes

本関数の各設定値に対応する eDRX パラメータを、図 3.1 に示します。

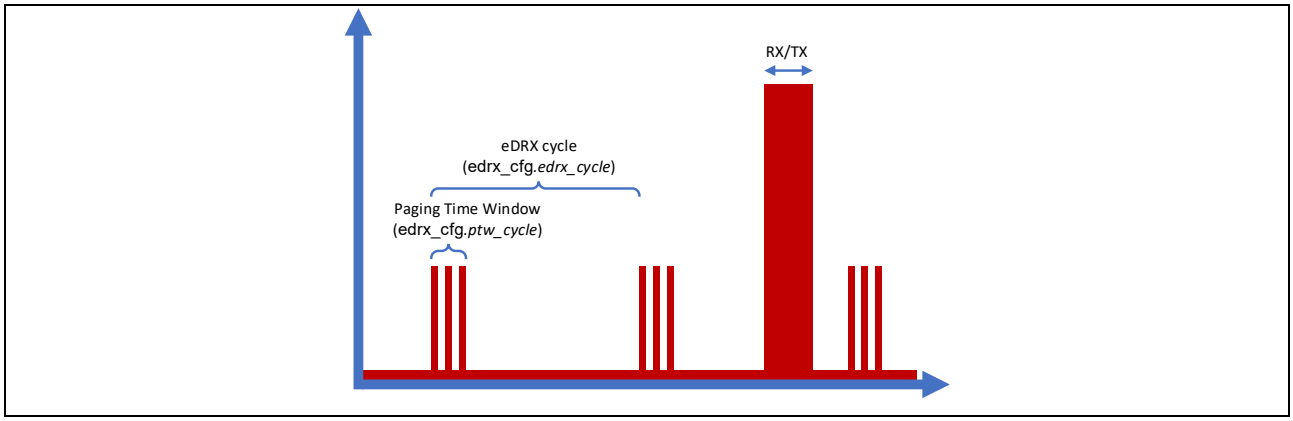

図 3.1 eDRX パラメータ

3.16 R_CELLULAR_GetEDRX()

eDRX (extended Discontinuous Reception)のパラメータを取得します。

Format

```
e_cellular_err_t R_CELLULAR_GetEDRX (
    st_cellular_ctrl_t * const p_ctrl,
    st_cellular_edrx_config_t * const p_result
)
```

Parameters

p_ctrl (IN/OUT)　　*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

p_result (IN/OUT)　　*ユーザが宣言した st_cellular_edrx_config_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

適用されている eDRX パラメータを取得し、引数 *p_result* へ設定した st_cellular_edrx_config_t 構造体へ格納します。st_cellular_edrx_config_t 構造体のメンバについては、表 3.7 を参照してください。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_edrx_config_t edrx_ret = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_GetEDRX(&cellular_ctrl, &edrx_ret);
```

Special Notes

なし

3.17 R_CELLULAR_SetPSM()

RYZ014A Cellular モジュールへ、PSM (Power Saving Mode)のパラメータ設定を行います。

Format

```
e_cellular_err_t R_CELLULAR_SetPSM (
    st_cellular_ctrl_t * const p_ctrl,
    const st_cellular_psm_config_t * const p_config,
    st_cellular_psm_config_t * const p_result
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
p_config (IN)	*ユーザが宣言した st_cellular_psm_config_t 構造体へのポインタ (設定)*
p_result (IN/OUT)	*ユーザが宣言した st_cellular_psm_config_t 構造体へのポインタ (設定値確認)*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*
CELLULAR_ERR_IRQ_OPEN	*/* IRQ 端子の初期化に失敗 */*
CELLULAR_ERR_RECV_TASK	*/* シリアル受信に失敗 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールへ、PSM のパラメータ設定を行います。引数 *p_config* へ設定した st_cellular_psm_config_t 構造体に格納されている PSM パラメータが設定されます。st_cellular_psm_config_t 構造体のメンバを、表 3.8 に示します。

引数 *p_result* へ設定した st_cellular_psm_config_t 構造体で、適用される PSM パラメータを確認できます。

PSM を有効に設定している状態で R_CELLULAR_Close()が実行された場合、本 FIT モジュールによる PSM の制御は無効になります。PSM の制御を再度有効にする場合は、R_CELLULAR_Open()実行後に本 API を実行してください。

表 3.8 PSM 設定・取得構造体のメンバ

Members in st_cellular_psm_config_t structure		
e_cellular_psm_mode_t	psm_mode	PSM モード
e_cellular_tau_cycle_t	tau_cycle	TAU 周期
e_cellular_cycle_multiplier_t	tau_multiplier	TAU 周期乗数
e_cellular_active_cycle_t	active_cycle	Active 時間
e_cellular_cycle_multiplier_t	active_multiplier	Active 時間乗数

Cellular モジュールへ設定される値は、それぞれ以下の式で計算されます。

$$TAU = tau \times tau\_multiplier$$
$$ActiveTime = active \times active\_multiplier$$

st_cellular_psm_config_t 構造体のメンバ psm_mode へ設定可能な値は下記の通りです。

```
typedef enum
{
    CELLULAR_PSM_MODE_INVALID       = 0,    // Disable the PSM function
    CELLULAR_PSM_MODE_ACTIVE        = 1,    // Enable the PSM function
    CELLULAR_PSM_MODE_INIT          = 2     // Initialize and disable the PSM function
} e_cellular_psm_mode_t;
```

st_cellular_psm_config_t 構造体のメンバ tau_cycle (Tracking Area Update cycle)へ設定可能な値は下記の通りです。

```
typedef enum
{
    CELLULAR_TAU_CYCLE_10_MIN = 0,          // TAU cycle (10min)
    CELLULAR_TAU_CYCLE_1_HOUR,              // TAU cycle (1hour)
    CELLULAR_TAU_CYCLE_10_HOUR,             // TAU cycle (10hour)
    CELLULAR_TAU_CYCLE_2_SEC,               // TAU cycle (2sec)
    CELLULAR_TAU_CYCLE_30_SEC,              // TAU cycle (30sec)
    CELLULAR_TAU_CYCLE_1_MIN,               // TAU cycle (1min)
    CELLULAR_TAU_CYCLE_320_HOUR,            // TAU cycle (320hour)
    CELLULAR_TAU_CYCLE_NONE                 // TAU cycle (Timer is deactivated)
} e_cellular_tau_cycle_t;
```

st_cellular_psm_config_t 構造体のメンバ active_cycle (Active Time)へ設定可能な値は下記の通りです。

```
typedef enum
{
    CELLULAR_ACTIVE_CYCLE_2_SEC = 0,        // Active time (2sec)
    CELLULAR_ACTIVE_CYCLE_1_MIN,            // Active time (1min)
    CELLULAR_ACTIVE_CYCLE_6_MIN,            // Active time (6min)
    CELLULAR_ACTIVE_CYCLE_NONE              // Active time (Timer is deactivated)
} e_cellular_active_cycle_t;
```

st_cellular_psm_config_t 構造体のメンバ tau_multiplier、active_multiplier へ設定可能な値は下記の通りです。

```
typedef enum
{
    CELLULAR_CYCLE_MULTIPLIER_0 = 0,          // Multiplier 0
    CELLULAR_CYCLE_MULTIPLIER_1 = 1,          // Multiplier 1
    CELLULAR_CYCLE_MULTIPLIER_2 = 2,          // Multiplier 2
                    ⋮
    CELLULAR_CYCLE_MULTIPLIER_30 = 30,        // Multiplier 30
    CELLULAR_CYCLE_MULTIPLIER_31 = 31         // Multiplier 31
} e_cellular_cycle_multiplier_t;
```

Reentrant

不可

Thread Safety

対応

Examples

TAU を 10 分、ActiveTime を 1 分に設定する場合 (設定値の詳細は、r_cellular_if.h のコメントを参照)

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_psm_config_t psm_cfg = {CELLULAR_PSM_MODE_ACTIVE,
                                    CELLULAR_TAU_CYCLE_10_MIN,
                                    CELLULAR_CYCLE_MULTIPLIER_1,
                                    CELLULAR_ACTIVE_CYCLE_1_MIN,
                                    CELLULAR_CYCLE_MULTIPLIER_1};
st_cellular_psm_config_t psm_ret = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);

/* 適用されたパラメータを確認しない場合 */
ret = R_CELLULAR_SetPSM(&cellular_ctrl, &psm_cfg, NULL);

/* 適用されたパラメータを確認する場合 */
ret = R_CELLULAR_SetPSM(&cellular_ctrl, &psm_cfg, &psm_ret);
```

※psm_cfg.tau_cycle へ 10min、psm_cfg.tau_multiplier へ乗数 1 を指定＝10min × 1 = 10min
※psm_cfg.active_cycle へ 1min、psm_cfg.active_multiplier へ乗数 1 を指定＝1min × 1 = 1min

Special Notes

本関数の各設定値に対応する PSM パラメータを図 3.2 に示します。

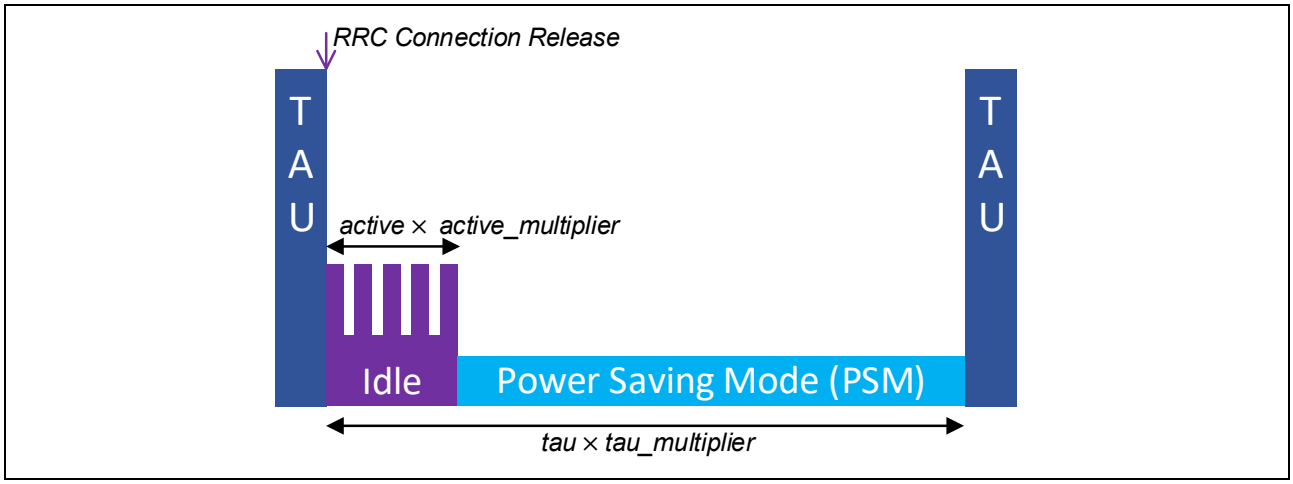

図 3.2 PSM パラメータ

3.18 R_CELLULAR_GetPSM()

PSM (Power Saving Mode)のパラメータを取得します。

Format

```
e_cellular_err_t R_CELLULAR_GetPSM (
    st_cellular_ctrl_t * const p_ctrl,
    st_cellular_psm_config_t * const p_result
)
```

Parameters

p_ctrl (IN/OUT)　　*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

p_result (IN/OUT)　　*ユーザが宣言した st_cellular_psm_config_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

適用されている PSM パラメータを取得し、引数 *p_result* へ設定した st_cellular_psm_config_t 構造体へ格納します。st_cellular_psm_config_t 構造体のメンバについては、表 3.8 を参照してください。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_psm_config_t psm_ret = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_GetPSM(&cellular_ctrl, &psm_ret);
```

Special Notes

なし

3.19 R_CELLULAR_GetICCID()

SIM が保持する ICCID (IC Card Identifier、SIM 識別番号)を取得します。

Format

```
e_cellular_err_t R_CELLULAR_GetICCID (
    st_cellular_ctrl_t * const p_ctrl,
    st_cellular_iccid_t * const p_iccid
)
```

Parameters

p_ctrl (IN/OUT)　　*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

p_iccid (IN/OUT)　　*取得した ICCID を格納する構造体へのポインタ*

Return values

CELLULAR_SUCCESS　　*/* 正常終了 */*

CELLULAR_ERR_PARAMETER　　*/* 引数が無効な値 */*

CELLULAR_ERR_NOT_OPEN　　*/* Open 関数を実行していない */*

CELLULAR_ERR_MODULE_COM　　*/* Cellular モジュールとの通信に失敗 */*

CELLULAR_ERR_MODULE_TIMEOUT　　*/* Cellular モジュールからの応答がない */*

CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING　　*/* 他の AT コマンドが実行中 */*

CELLULAR_ERR_OTHER_API_RUNNING　　*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

SIM に設定されている ICCID を取得し、引数 *p_iccid* へ設定した st_cellular_iccid_t 構造体へ格納します。st_cellular_iccid_t 構造体のメンバを、表 3.9 に示します。取得可能な ICCID は、最大 22 桁です。

表 3.9 ICCID 取得構造体のメンバ

Members in st_cellular_iccid_t structure	
uint8_t　iccid[]	ICCID

Reentrant

可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_iccid_t cellular_iccid = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_GetICCID(&cellular_ctrl, &cellular_iccid);
```

Special Notes

なし

3.20 R_CELLULAR_GetIMEI()

RYZ014A Cellular モジュールが保持する IMEI (International Mobile Equipment Identifier、端末識別番号)を取得します。

Format

```
e_cellular_err_t R_CELLULAR_GetIMEI (
    st_cellular_ctrl_t * const p_ctrl,
    st_cellular_imei_t * const p_imei
)
```

Parameters

p_ctrl (IN/OUT)　　*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

p_imei (IN/OUT)　　*取得した IMEI を格納する構造体へのポインタ*

Return values

CELLULAR_SUCCESS　　*/* 正常終了 */*

CELLULAR_ERR_PARAMETER　　*/* 引数が無効な値 */*

CELLULAR_ERR_NOT_OPEN　　*/* Open 関数を実行していない */*

CELLULAR_ERR_MODULE_COM　　*/* Cellular モジュールとの通信に失敗 */*

CELLULAR_ERR_MODULE_TIMEOUT　　*/* Cellular モジュールからの応答がない */*

CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING　　*/* 他の AT コマンドが実行中 */*

CELLULAR_ERR_OTHER_API_RUNNING　　*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールに設定されている IMEI を取得し、引数 *p_imei* へ設定した st_cellular_imei_t 構造体へ格納します。st_cellular_imei_t 構造体のメンバを、表 3.10 に示します。取得可能な IMEI は、最大 15 桁です。

表 3.10 IMEI 取得構造体のメンバ

Members in st_cellular_imei_t structure	
uint8_t　imei[]	IMEI

Reentrant

可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_imei_t cellular_imei = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_GetIMEI(&cellular_ctrl, &cellular_imei);
```

Special Notes

なし

3.21 R_CELLULAR_GetIMSI()

SIM が保持する IMSI (International Mobile Subscriber Identity、加入者識別番号)を取得します。

Format

```
e_cellular_err_t R_CELLULAR_GetIMSI (
    st_cellular_ctrl_t * const p_ctrl,
    st_cellular_imsi_t * const p_imsi
)
```

Parameters

p_ctrl (IN/OUT)　　*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

p_imsi (IN/OUT)　　*取得した IMSI を格納する構造体へのポインタ*

Return values

CELLULAR_SUCCESS　　*/* 正常終了 */*

CELLULAR_ERR_PARAMETER　　*/* 引数が無効な値 */*

CELLULAR_ERR_NOT_OPEN　　*/* Open 関数を実行していない */*

CELLULAR_ERR_MODULE_COM　　*/* Cellular モジュールとの通信に失敗 */*

CELLULAR_ERR_MODULE_TIMEOUT　　*/* Cellular モジュールからの応答がない */*

CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING　　*/* 他の AT コマンドが実行中 */*

CELLULAR_ERR_OTHER_API_RUNNING　　*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

SIM に設定されている IMSI を取得し、引数 *p_imsi* へ設定した st_cellular_imsi_t 構造体へ格納します。st_cellular_imsi_t 構造体のメンバを、表 3.11 に示します。取得可能な IMSI は、最大 15 桁です。

表 3.11 IMSI 取得構造体のメンバ

Members in st_cellular_imsi_t structure	
uint8_t　imsi[]	IMSI

Reentrant

可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_imsi_t cellular_imsi = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_GetIMSI(&cellular_ctrl, &cellular_imsi);
```

Special Notes

なし

3.22 R_CELLULAR_GetPhonenum()

SIM が保持する電話番号を取得します。

Format

```
e_cellular_err_t R_CELLULAR_GetPhonenum (
    st_cellular_ctrl_t * const p_ctrl,
    st_cellular_phonenum_t * const p_phonenum
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
p_phonenum (IN/OUT)	*取得した電話番号を格納する構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

SIM に設定されている電話番号を取得し、引数 *p_phonenum* へ設定した st_cellular_phonenum_t 構造体へ格納します。st_cellular_phonenum_t 構造体のメンバを、表 3.12 に示します。取得可能な電話番号は、最大 15 桁です。

表 3.12 電話番号取得構造体のメンバ

Members in st_cellular_phonenum_t structure		
uint8_t	phonenum[]	電話番号

Reentrant

可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_phonenum_t cellular_phonenum = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_GetPhonenum(&cellular_ctrl, &cellular_phonenum);
```

Special Notes

なし

3.23 R_CELLULAR_GetRSSI()

RYZ014A Cellular モジュールから RSSI (Received Signal Strength Indicator、受信電波強度)および BER (Bit Error Rate、符号誤り率)を取得します。

Format

```
e_cellular_err_t R_CELLULAR_GetRSSI (
    st_cellular_ctrl_t * const p_ctrl,
    st_cellular_rssi_t * const p_rssi
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
p_rssi (IN/OUT)	*取得した RSSI および BER を格納する構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールから RSSI および BER を取得し、引数 *p_rssi* へ設定した st_cellular_rssi_t 構造体へ格納します。st_cellular_rssi_t 構造体のメンバを、表 3.13 に示します。RSSI および BER は、値が不明な場合(アクセスポイント未接続状態等)は“99”が取得されます。

表 3.13 RSSI 取得構造体のメンバ

Members in st_cellular_rssi_t structure		
uint8_t	rssi[]	RSSI
uint8_t	ber[]	BER

Reentrant

可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_rssi_t cellular_rssi = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
ret = R_CELLULAR_GetRSSI(&cellular_ctrl, &cellular_rssi);
```

Special Notes

なし

3.24 R_CELLULAR_GetSVN()

RYZ014A Cellular モジュールが保持する SVN およびリビジョンを取得します。

Format

```
e_cellular_err_t R_CELLULAR_GetSVN (
    st_cellular_ctrl_t * const p_ctrl,
    st_cellular_svn_t * const p_svn
)
```

Parameters

p_ctrl (IN/OUT)　　*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

p_svn (IN/OUT)　　*取得した SVN およびリビジョンを格納する構造体へのポインタ*

Return values

CELLULAR_SUCCESS　　*/* 正常終了 */*

CELLULAR_ERR_PARAMETER　　*/* 引数が無効な値 */*

CELLULAR_ERR_NOT_OPEN　　*/* Open 関数を実行していない */*

CELLULAR_ERR_MODULE_COM　　*/* Cellular モジュールとの通信に失敗 */*

CELLULAR_ERR_MODULE_TIMEOUT　　*/* Cellular モジュールからの応答がない */*

CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING　　*/* 他の AT コマンドが実行中 */*

CELLULAR_ERR_OTHER_API_RUNNING　　*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールに設定されている SVN (ソフトウェアバージョン番号)およびリビジョンを取得し、引数 *p_svn* へ設定した st_cellular_svn_t 構造体へ格納します。st_cellular_svn_t 構造体のメンバを、表 3.14 に示します。

表 3.14 SVN 取得構造体のメンバ

Members in st_cellular_svn_t structure		
uint8_t	svn[]	ソフトウェアバージョン番号
uint8_t	revision[]	リビジョン
uint8_t	lr_svn[]	LR バージョン番号

Reentrant

可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_svn_t cellular_svn = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_GetSVN(&cellular_ctrl, &cellular_svn);
```

Special Notes

なし

3.25 R_CELLULAR_Ping()

PING を実行します。

Format

```
e_cellular_err_t R_CELLULAR_Ping (
    st_cellular_ctrl_t * const p_ctrl,
    const uint8_t * const p_host,
    const st_cellular_ping_cfg_t * const p_cfg
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
p_host (IN)	*接続先ホスト名または IP アドレスが格納された配列へのポインタ*
p_cfg (IN)	*ユーザが宣言した st_cellular_ping_cfg_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_NOT_CONNECT	*/* アクセスポイントに接続していない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

PING を実行します。引数 *p_host* へ設定する接続先 IP アドレスおよびホスト名は、文字列で入力してください。

引数 *p_cfg* は、下記の 2 通りの設定が可能です。(設定値の詳細は、表 3.15 を参照)

(1) 引数 *p_cfg* へ NULL を設定し、デフォルト値で PING を実行する。
(2) 各メンバへ適切な値を設定した st_cellular_ping_cfg_t 構造体へのポインタを引数 *p_cfg* へ設定する。

Smart Configurator で CELLULAR_CFG_URC_CHARGET_ENABLED を 1 (コールバック関数を使用する)に設定している場合、登録したコールバック関数で PING の詳細結果を受け取ることができます。(表 2.1 参照)

表 3.15 設定可能な PING パラメータ (R_CELLULAR_Ping())

Members in st_cellular_ping_cfg_t structure		
uint8_t	count	PING のエコーバック回数(デフォルト 4 回，最小 1 回，最大 64 回)
uint16_t	len	パケットサイズ(デフォルト 32byte，最小 32byte，最大 1400byte).
uint16_t	interval	パケット送信間隔(デフォルト 1 秒，最小 1 秒，最大 600 秒)
uint8_t	timeout	タイムアウト(デフォルト 10 秒，最小 1 秒，　最大 60 秒).

Reentrant

不可

Thread Safety

対応

Examples

```
/* コールバック関数例 (関数名がデフォルト設定の場合) */
void my_sw_urc_charget_function(void * p_arg)
{
    uint8_t * p_urc = p_arg;
    printf("URC = %s¥n", p_urc);
}

e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_ping_cfg_t ping_cfg = {64, 1400, 1, 10};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);

/* ホスト名を入力する場合 */
ret = R_CELLULAR_Ping(&cellular_ctrl, "www.renesas.com", NULL);

/* IP アドレスを入力する場合 */
ret = R_CELLULAR_Ping(&cellular_ctrl, "104.120.11.132", NULL);

/* コンフィグ構造体を使用する場合 */
ret = R_CELLULAR_Ping(&cellular_ctrl, "www.renesas.com", &ping_cfg);
```

Special Notes

なし

3.26 R_CELLULAR_GetAPConnectState()

RYZ014A Cellular モジュールのアクセスポイントへの接続状態を確認します。

Format

```
e_cellular_err_t R_CELLULAR_GetAPConnectState (
    st_cellular_ctrl_t * const p_ctrl,
    const e_cellular_network_result_t level,
    st_cellular_notice_t * const p_result
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
level (IN)	*自動通知レベルの設定*
p_result (IN/OUT)	*アクセスポイントへの接続ステータスを格納する構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールのアクセスポイントへの接続状態を取得し、引数 *p_result* へ設定した st_cellular_notice_t 構造体へ格納します。st_cellular_notice_t 構造体のメンバを、表 3.16 に示します。

引数 *level* で自動通知設定を有効化している場合、かつ Smart Configurator で CELLULAR_CFG_URC_CHARGET_ENABLED を 1 (コールバック関数を使用する)に設定している場合、登録したコールバック関数で Cellular モジュールから通知されるアクセスポイント接続情報を受け取ることができます。(表 2.1 参照)

引数 *level* へ設定可能な値は下記の通りです。

```
typedef enum
{
    CELLULAR_DISABLE_NETWORK_RESULT_CODE = 0,
            // Disable notification of network registration status change
    CELLULAR_ENABLE_NETWORK_RESULT_CODE_LEVEL1,
            // Enable notification of network registration status change
    CELLULAR_ENABLE_NETWORK_RESULT_CODE_LEVEL2,
            // Enable detailed notification of network registration status change and location information
    CELLULAR_ENABLE_NETWORK_RESULT_CODE_LEVEL3,
            // Enable network registration, location information and EMM cause value information
    CELLULAR_ENABLE_NETWORK_RESULT_CODE_LEVEL4,
            // For a UE that wants to apply PSM, enable network registration and location information
    CELLULAR_ENABLE_NETWORK_RESULT_CODE_LEVEL5
            // For a UE that wants to apply PSM, enable network registration, location information and
               EMM cause value information
} e_cellular_network_result_t;
```

表 3.16 アクセスポイント接続状態取得構造体のメンバ

Members in st_cellular_notice_t structure		
e_cellular_network_result_t	level	Network status notification level
e_cellular_reg_stat_t	stat	EPS registration status
uint8_t	ta_code[]	Tracking area code in hexadecimal format, string.
uint8_t	cell_id[]	E-UTRAN cell ID in hexadecimal format, string.
e_cellular_access_tec_t	access_tec	Access technology of the serving cell
uint8_t	active_time[]	Active time value (T3324), string.
uint8_t	tau[]	Periodic TAU value (T3412), string.

st_cellular_notice_t 構造体のメンバ stat へ設定される値は、下記の通りです。

```
typedef enum
{
    CELLULAR_REG_STATS_VALUE0 = 0,     // Not registered
    CELLULAR_REG_STATS_VALUE1,         // Registered, home network
    CELLULAR_REG_STATS_VALUE2,
                    // Not registered, but MT is currently trying to attach or searching an operator to register to
    CELLULAR_REG_STATS_VALUE3,         // Registration denied
    CELLULAR_REG_STATS_VALUE4,         // Unknown (for example, out of E-UTRAN coverage)
    CELLULAR_REG_STATS_VALUE5,         // Registered, roaming
    CELLULAR_REG_STATS_VALUE6,         // Registered for "SMS only", home network (not applicable)
    CELLULAR_REG_STATS_VALUE7,         // Registered for "SMS only", roaming (not applicable)
    CELLULAR_REG_STATS_VALUE8,         // Attached for emergency bearer services only
    CELLULAR_REG_STATS_VALUE9,         // Registered for "CSFB not preferred", home network (not applicable)
    CELLULAR_REG_STATS_VALUE10,        // Registered for "CSFB not preferred", roaming (not applicable)
    CELLULAR_REG_STATS_VALUE80         // Registered, temporary connection lost
} e_cellular_reg_stat_t;
```

st_cellular_notice_t 構造体のメンバ access_tec へ設定される値は、下記の通りです。

```
typedef enum
{
    CELLULAR_ACCESS_TEC0 = 0,   // GSM
    CELLULAR_ACCESS_TEC1,       // GSM Compact
    CELLULAR_ACCESS_TEC2,       // UTRAN
    CELLULAR_ACCESS_TEC3,       // GSM w/EGPRS
    CELLULAR_ACCESS_TEC4,       // UTRAN w/HSDPA
    CELLULAR_ACCESS_TEC5,       // UTRAN w/HSUPA
    CELLULAR_ACCESS_TEC6,       // UTRAN w/HSDPA and HSUPA
    CELLULAR_ACCESS_TEC7        // E-UTRAN
} e_cellular_access_tec_t;
```

Reentrant

不可

Thread Safety

対応

Examples

```
/* コールバック関数例 (関数名がデフォルト設定の場合) */
void my_sw_urc_charget_function(void * p_arg)
{
   uint8_t * p_urc = p_arg;
   printf("URC = %s¥n", p_urc);
}

e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_notice_t cellular_notice = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
ret = R_CELLULAR_GetAPConnectState(&cellular_ctrl,
             CELLULAR_ENABLE_NETWORK_RESULT_CODE_LEVEL2, &cellular_notice);
```

Special Notes

なし

3.27 R_CELLULAR_GetCellInfo()

セル情報を取得します。

Format

```
e_cellular_err_t R_CELLULAR_GetCellInfo (
    st_cellular_ctrl_t * const p_ctrl,
    const e_cellular_info_type_t type
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
type (IN)	*取得するセル情報の詳細設定*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールが属しているセル情報を取得し、コールバック関数で取得したセル情報の通知を行います。本 API 実行時に Cellular モジュールの機能レベルが 1 (Full functionality)ではない場合、機能レベルが 1 に設定されます。機能レベルの詳細は、RYZ014 Modules User's Manual: AT Command の「3.11 Set Phone Functionality: AT+CFUN」を参照してください。

本 API を使用する場合、Smart Configurator で CELLULAR_CFG_URC_CHARGET_ENABLED を 1 (コールバック関数を使用する)に設定してください。(表 2.1 参照) コールバック関数を使用する設定が行われていない場合、戻り値で CELLULAR_ERR_PARAMETER を返します。

引数 *type* へ設定可能な値は下記の通りです。

```
typedef enum
{
    CELLULAR_INFO_TYPE0 = 0,    // Report information for the serving cell only
    CELLULAR_INFO_TYPE1 = 1,    // Report information for the intra-frequency cells only
    CELLULAR_INFO_TYPE2 = 2,    // Report information for the intra-frequency cells only
    CELLULAR_INFO_TYPE7 = 7,    // Report information for all cells
    CELLULAR_INFO_TYPE9 = 9
                        // Report information for the serving cell only with RSRP/CINR on main antenna.
} e_cellular_info_type_t;
```

Reentrant

不可

Thread Safety

対応

Examples

```
/* コールバック関数例 (関数名がデフォルト設定の場合) */
void my_sw_urc_charget_function(void * p_arg)
{
    uint8_t * p_urc = p_arg;
    printf("URC = %s¥n”, p_urc);
}

e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
ret = R_CELLULAR_GetCellInfo(&cellular_ctrl, CELLULAR_INFO_TYPE9);
```

Special Notes

なし

3.28 R_CELLULAR_AutoConnectConfig()

アクセスポイントへの自動接続設定を行います。

Format

```
e_cellular_err_t R_CELLULAR_AutoConnectConfig (
    st_cellular_ctrl_t * const p_ctrl,
    e_cellular_auto_connect_t const type
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
type (IN)	*自動接続設定*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュール起動時またはリセット後に、アクセスポイントへ自動接続を行うか否かを設定します。設定値は Cellular モジュールの不揮発メモリで保持されます。

引数 *type* へ設定可能な値は下記の通りです。

```
typedef enum
{
    CELLULAR_DISABLE_AUTO_CONNECT = 0,      // Disable automatic connection to an access point
    CELLULAR_ENABLE_AUTO_CONNECT
                                // Enable automatic connection to an access point (next start-up or reboot)
} e_cellular_auto_connect_t;
```

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_AutoConnectConfig(&cellular_ctrl,
                                   CELLULAR_ENABLE_AUTO_CONNECT);
```

Special Notes

なし

3.29 R_CELLULAR_SetOperator()

オペレータモードを設定します。

Format

```
e_cellular_err_t R_CELLULAR_SetOperator (
    st_cellular_ctrl_t * const p_ctrl,
    const uint8_t * const p_operator
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
p_operator (IN)	*オペレータモード設定*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールが通信で使用するオペレータモードを設定します。引数 *p_operator* へ、オペレータモードを文字列で設定してください。RYZ014A は、オペレータモード”standard”のみサポートします。オペレータモードの詳細は、RYZ014 Module System Integration Guide の「2.2.2 Operator Modes」を参照してください。

PIN コードが設定されている SIM を使用する場合、本 API を実行後に R_CELLULAR_UnlockSIM()を実行して PIN コードを入力してください。詳細は、3.35 節を参照してください。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_SetOperator(&cellular_ctrl, “standard”);
```

Special Notes

なし

3.30 R_CELLULAR_SetBand()

使用する LTE バンドを設定します。

Format

```
e_cellular_err_t R_CELLULAR_SetBand(
    st_cellular_ctrl_t * const p_ctrl,
    const uint8_t * const p_band
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
p_band (IN)	*バンドリスト設定*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールが通信で使用する LTE バンドを設定します。設定値は、R_CELLULAR_SetOperator()によって設定されているオペレータモードに対して設定されます。引数 *p_band* へ設定するバンドリストは、文字列で設定してください。複数のバンドを設定する場合は、カンマ「,」で区切ってください。例えば、バンド 1、2、3 を設定する場合、”1,2,3”を入力してください。

Cellular モジュールへ設定可能なバンドは”1,2,3,4,5,8,12,13,14,17,18,19,20,25,26,28,66"です。利用するキャリアや仕向け先などに合わせて、適切な値を設定してください。各地域で、主に使用されるバンドを表 3.17 に示します。日本では、バンド 3 は使用しないでください。

PIN コードが設定されている SIM を使用する場合、本 API を実行後に R_CELLULAR_UnlockSIM()を実行して PIN コードを入力してください。詳細は、3.35 節を参照してください。

表 3.17 地域毎の主に使用されるバンド

Region	Bands
North America	2, 4, 5, 12, 13, 25
EMEA	1, 3, 8, 20, 28
Japan	1, 8, 18, 19, 26
Australia	1, 3, 8, 28

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_SetBand(&cellular_ctrl, "1,2,3");
```

Special Notes

なし

3.31 R_CELLULAR_GetPDPAddress()

RYZ014A Cellular モジュールに割り当てられている PDP アドレスを取得します。

Format

```
e_cellular_err_t R_CELLULAR_GetPDPAddress (
    st_cellular_ctrl_t * const p_ctrl,
    st_cellular_ipaddr_t * const p_addr
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
p_addr(IN/OUT)	*ユーザが宣言した st_cellular_ipaddr_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_NOT_CONNECT	*/* アクセスポイントに接続していない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールに割り当てられている PDP アドレスを取得します。IPv6 が利用可能な場合は、IPv4 アドレスおよび IPv6 アドレスが、引数 *p_addr* へ設定した st_cellular_ipaddr_t 構造体の ipv4 および ipv6 メンバへそれぞれ格納されます。SIM が IPv6 非対応などの要因で IPv6 が利用できない場合は、IPv4 アドレスのみ ipv4 メンバへ格納され、ipv6 メンバへ 0 が格納されます。st_cellular_ipaddr_t 構造体のメンバについては、表 3.5 を参照してください。

本 API を使用する前に R_CELLULAR_APConnect()を実行してアクセスポイントへ接続してください。本 API 実行時に Cellular モジュールがアクセスポイントへ接続していない場合は、戻り値で CELLULAR_ERR_NOT_CONNECT を返します。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_ipaddr_t addr = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
ret = R_CELLULAR_GetPDPAddress (&cellular_ctrl, &addr);
```

Special Notes

なし

3.32 R_CELLULAR_FirmUpgrade()

RYZ014A Cellular モジュールのファームウェアをネットワーク経由で更新する FOTA (Firmware upgrade Over-The-Air)を実行します。

Format

```
e_cellular_err_t R_CELLULAR_FirmUpgrade (
    st_cellular_ctrl_t * const p_ctrl,
    const uint8_t * const p_url,
    const e_cellular_firmupgrade_t command,
    const uint8_t spid
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
p_url (IN)	*接続先 URL*
command (IN)	*実行コマンド (実行 = 1、キャンセル = 2)*
spid (IN)	*セキュリティプロファイル ID (不使用 = 0、使用 = 1 ~ 6)*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_NOT_CONNECT	*/* アクセスポイントに接続していない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールのファームウェアを FOTA で更新します。ファームウェア更新の進行状況は R_CELLULAR_GetUpgradeState()を実行することで確認できます。

FOTA を実行するために SSL/TLS サーバへ接続する場合は、引数 *spid* に R_CELLULAR_ConfigSSLProfile()で設定したセキュリティプロファイル ID を指定してください。

本 API を使用する前に R_CELLULAR_APConnect()を実行してアクセスポイントへ接続してください。本 API 実行時に Cellular モジュールがアクセスポイントへ接続していない場合は、戻り値で CELLULAR_ERR_NOT_CONNECT を返します。

FOTA の進行状況は、R_CELLULAR_GetUpgradeState()を実行して取得できます。また、コールバック関数で+SQNSUPGRADE URC の upgrade_state から取得することもできます。Smart Configurator で

CELLULAR_CFG_URC_CHARGET_ENABLED を 1 (コールバック関数を使用する)に設定すると、登録したコールバック関数で URC を受け取ることができます。(表 2.1 参照)

本 API は、FOTA を Asynchronous upgrade モードで実行するためノンブロッキングです。本 API は、FOTA 開始後に CELLULAR_SUCCESS を返して終了します。ユーザは、FOTA の進行状況が“”downloading”, 100”になった後、リセットを実行してファームウェア更新を完了させてください。任意のリセットが使用可能です。詳細は、RYZ014 Modules User's Manual: AT Command の「6.1 Device Upgrade: AT+SQNSUPGRADE」を参照してください。リセット後、更新されたファームウェアでの起動が完了すると+SYSSTART URC が通知されます。その後、R_CELLULAR_Close()を実行後に R_CELLULAR_Open()を実行してください。

本 API を実行して FOTA を開始した後は、FOTA の進行状況が“”downloading”, 100”になるまで Cellular モジュールのリセットを実行しないでください。FOTA を中断する場合は、引数 *commnad* に 2 を設定して本 API を実行してください。

本 FIT モジュールは、FOTA の完了までのタイムアウトの検出および処理を実施しません。ユーザは、必要に応じて FOTA の完了までのタイムアウト処理を実装してください。

引数 *command* へ設定可能な値は、以下の通りです。

```
typedef enum
{
    CELLULAR_FIRM_UPGRADE_NONBLOCKING = 1, // Performs firmware upgrade in non-blocking mode
    CELLULAR_FIRM_UPGRADE_CANCEL = 2          // Cancel firmware upgrade
} e_cellular_firmupgrade_t;
```

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
ret = R_CELLULAR_FirmUpgrade (&cellular_ctrl, “https://www.renesas.com”,
                               CELLULAR_FIRM_UPGRADE_NONBLOCKING, 0);
```

Special Notes

本 API で実行する Asynchronous upgrade モードの FOTA は、Cellular モジュールファームウェアの一部のバージョンで非対応です。そのため、使用する Cellular モジュールのファームウェアバージョンによって本 API が使用できない場合があります。

3.33 R_CELLULAR_FirmUpgradeBlocking()

RYZ014A Cellular モジュールのファームウェアをネットワーク経由で更新する FOTA (Firmware upgrade Over-The-Air)を、ブロッキングモードで実行します。

Format

```
e_cellular_err_t R_CELLULAR_FirmUpgradeBlocking (
    st_cellular_ctrl_t * const p_ctrl,
    const uint8_t * const p_url,
    const uint8_t spid
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
p_url (IN)	*接続先 URL*
spid (IN)	*セキュリティプロファイル ID (不使用 = 0、使用 = 1 ~ 6)*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_NOT_CONNECT	*/* アクセスポイントに接続していない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

FOTA をブロッキングモードで実行します。本 API を実行して開始した FOTA は中断できません。また、本 API の実行中は他の API を実行できません。

FOTA を実行するために SSL/TLS サーバへ接続する場合は、引数 *spid* に R_CELLULAR_ConfigSSLProfile()で設定したセキュリティプロファイル ID を指定してください。

本 API を使用する前に R_CELLULAR_APConnect()を実行してアクセスポイントへ接続してください。本 API 実行時に Cellular モジュールがアクセスポイントへ接続していない場合は、戻り値で CELLULAR_ERR_NOT_CONNECT を返します。

Smart Configurator で CELLULAR_CFG_URC_CHARGET_ENABLED を 1 (コールバック関数を使用する)に設定している場合、登録したコールバック関数で更新状況を受け取ることができます。(表 2.1 参照)

FOTA が正常に完了すると、本 API は CELLULAR_SUCCESS を返して終了します。FOTA 完了後は、R_CELLULAR_Close()を実行後に R_CELLULAR_Open()を実行してください。

本 API の実行中は、Cellular モジュールのリセットを実行しないでください。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
ret = R_CELLULAR_FirmUpgradeBlocking(&cellular_ctrl,
                                     “https://www.renesas.com”, 0);
```

Special Notes

なし

3.34 R_CELLULAR_GetUpgradeState()

RYZ014A Cellular モジュールの FOTA によるファームウェア更新の進行状況を取得します。

Format

```
e_cellular_err_t R_CELLULAR_GetUpgradeState (
    st_cellular_ctrl_t * const p_ctrl,
    st_cellular_updatestate_t * const p_state
)
```

Parameters

p_ctrl (IN/OUT)　　*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

p_state (IN/OUT)　　*ユーザが宣言した st_cellular_updatestate_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS　　*/* 正常終了 */*

CELLULAR_ERR_PARAMETER　　*/* 引数が無効な値 */*

CELLULAR_ERR_NOT_OPEN　　*/* Open 関数を実行していない */*

CELLULAR_ERR_MODULE_COM　　*/* Cellular モジュールとの通信に失敗 */*

CELLULAR_ERR_MODULE_TIMEOUT　　*/* Cellular モジュールからの応答がない */*

CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING　　*/* 他の AT コマンドが実行中 */*

CELLULAR_ERR_OTHER_API_RUNNING　　*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールの FOTA の進行状況を取得し、引数 *p_state* へ設定した st_cellular_updatestate_t 構造体へ格納します。st_cellular_updatestate_t 構造体のメンバを、表 3.18 に示します。

R_CELLULAR_FirmUpgrade()の実行で開始された FOTA の進行状況を取得できます。取得した値の詳細は、RYZ014 Modules User's Manual: AT Command の「6.1 Device Upgrade: AT+SQNSUPGRADE」を参照してください。

表 3.18 FOTA 状態取得構造体のメンバ

Members in st_cellular_updatestate_t structure	
uint8_t　state[]	FOTA の進行状況

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_updatestate_t state = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_GetUpgradeState (&cellular_ctrl, &state);
```

Special Notes

なし

3.35 R_CELLULAR_UnlockSIM()

RYZ014A Cellular モジュールへ PIN コードを入力します。

Format

```
e_cellular_err_t R_CELLULAR_UnlockSIM (
    st_cellular_ctrl_t * const p_ctrl,
    const uint8_t * const p_pass
  )
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
p_pass (IN)	*PIN コード (文字列)*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールへ PIN コードを入力します。本 API 実行時に Cellular モジュールの機能レベルが 1 (Full functionality)または 4 (Disable RF)ではなかった場合、機能レベルが 4 に設定されます。機能レベルの詳細は、RYZ014 Modules User's Manual: AT Command の「3.11 Set Phone Functionality: AT+CFUN」を参照してください。本 API を PIN コードが設定されていない SIM を使用中に実行した場合は、戻り値で CELLULAR_SUCCESS を返します。

PIN コードが設定されている SIM を使用する場合は、処理中に再起動が発生する一部の API 実行後に PIN コードを入力する必要があります。PIN コードを入力するまで、Cellular モジュールは SIM へアクセスできずアクセスポイントへ接続できません。処理中に再起動が発生する API については、4.3.2 節を参照してください。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_UnlockSIM (&cellular_ctrl, “0000”);
```

Special Notes

なし

3.36 R_CELLULAR_WriteCertificate()

RYZ014A Cellular モジュールの不揮発メモリへ、証明書または秘密鍵を書き込みます。

Format

```
e_cellular_err_t R_CELLULAR_WriteCertificate (
    st_cellular_ctrl_t * const p_ctrl,
    const e_cellular_nvm_type_t data_type,
    const uint8_t index,
    const uint8_t * p_data,
    const uint32_t size
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
data_type (IN)	*書き込む情報 (0 = 証明書、1 = 秘密鍵)*
index (IN)	*書き込み先インデックス (0 ~ 19)*
p_data (IN)	*書き込むデータ*
size (IN)	*書き込むデータのサイズ (バイト単位、設定値=1 ~16384)*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールの不揮発メモリへ、証明書または秘密鍵を書き込みます。証明書および秘密鍵を、それぞれ 20 個まで書き込むことが可能です。

引数 *data_type* へ設定可能な値は、以下の通りです。

```
typedef enum
{
    CELLULAR_NVM_TYPE_CERTIFICATE = 0,      //Certificate
    CELLULAR_NVM_TYPE_PRIVATEKEY            //Private key
} e_cellular_nvm_type_t;
```

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
const uint8_t CERTIFICATE[] = "-----BEGIN CERTIFICATE-----"¥
                              "AAAAAAAAAAAAAAAAAAAAAAAA¥n"¥
                              "-----END CERTIFICATE-----¥n";
const uint32_t CERTIFICATE_LENGTH = sizeof( CERTIFICATE );
const uint8_t PRIVATEKEY[] = "-----BEGIN RSA PRIVATE KEY-----¥n"¥
                             "AAAAAAAAAAAAAAAAAAAAAAAA¥n"¥
                             "-----END RSA PRIVATE KEY-----¥n";
const uint32_t PRIVATEKEY_LENGTH = sizeof( PRIVATEKEY );

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);

/* 証明書を書き込む */
ret = R_CELLULAR_WriteCertificate (&cellular_ctrl,
                                   CELLULAR_NVM_TYPE_CERTIFICATE, 0,
                                   CERTIFICATE, CERTIFICATE_LENGTH);

/* 秘密鍵を書き込む */
ret = R_CELLULAR_WriteCertificate (&cellular_ctrl,
                                   CELLULAR_NVM_TYPE_PRIVATEKEY, 0,
                                   PRIVATEKEY, PRIVATEKEY_LENGTH);
```

Special Notes

なし

3.37 R_CELLULAR_EraseCertificate()

RYZ014A Cellular モジュールの不揮発メモリに書き込まれている、証明書または秘密鍵を削除します。

Format

```
e_cellular_err_t R_CELLULAR_EraseCertificate (
    st_cellular_ctrl_t * const p_ctrl,
    const e_cellular_nvm_type_t data_type,
    const uint8_t index
  )
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
data_teype (IN)	*削除する情報 (0 = 証明書、1 = 秘密鍵)*
index (IN)	*削除先インデックス (0 ~ 19)*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールの不揮発メモリに書き込まれている、証明書または秘密鍵を削除します。

引数 *data_type* へ設定可能な値は、3.36 節を参照してください。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_EraseCertificate (&cellular_ctrl,
                                   CELLULAR_NVM_TYPE_CERTIFICATE, 0);
```

Special Notes

なし

3.38 R_CELLULAR_GetCertificate()

RYZ014A Cellular モジュールの不揮発メモリに書き込まれている証明書または秘密鍵を取得します。

Format

```
e_cellular_err_t R_CELLULAR_GetCertificate (
    st_cellular_ctrl_t * const p_ctrl,
    const e_cellular_nvm_type_t data_type,
    const uint8_t index,
    st_cellular_certificate_t * const p_cert
)
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
data_teype (IN)	*取得する情報 (0 = 証明書、1 = 秘密鍵)*
index (IN)	*取得先インデックス (0 ~ 19)*
p_cert (IN/OUT)	*ユーザが宣言した st_cellular_certificate_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールの不揮発メモリに書き込まれている証明書または秘密鍵を取得し、引数 *p_cert* へ設定した st_cellular_certificate_t 構造体へ格納します。st_cellular_certificate_t 構造体のメンバを、表 3.19 に示します。取得可能なデータサイズは、最大 2048byte です。

引数 *data_type* へ設定可能な値は、3.36 節を参照してください。

表 3.19 証明書・秘密鍵取得構造体のメンバ

Members in st_cellular_certificate_t structure		
uint8_t	certificate[]	証明書または秘密鍵

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_certificate_t cert = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_GetCertificate (&cellular_ctrl,
                                 CELLULAR_NVM_TYPE_CERTIFICATE, 0, &cert);
```

Special Notes

なし

3.39 R_CELLULAR_ConfigSSLProfile()

RYZ014A Cellular モジュールのセキュリティプロファイル設定を行います。

Format

```
e_cellular_err_t R_CELLULAR_ConfigSSLProfile (
    st_cellular_ctrl_t * const p_ctrl,
    const uint8_t security_profile_id,
    const e_cellular_cert_validate_level_t cert_valid_level,
    const uint8_t ca_certificate_id,
    const uint8_t client_certificate_id,
    const uint8_t client_privatekey_id
  )
```

Parameters

p_ctrl (IN/OUT)	*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*
security_profile_id (IN)	*セキュリティプロファイル ID (1~6)*
cert_valid_level (IN)	*証明書検証レベル*
ca_certificate_id (IN)	*ルート証明書参照インデックス*
	(0~19)：インデックス指定
	上記以外：ルート証明書を指定しない
client_certificate_id (IN)	*クライアント証明書参照インデックス*
	(0~19)：インデックス指定
	上記以外：クライアント証明書を指定しない
client_privatekey_id (IN)	*秘密鍵参照インデックス*
	(0~19)：インデックス指定
	上記以外：秘密鍵を指定しない

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

セキュリティプロファイルの設定を行います。R_CELLULAR_WriteCertificate()で書き込んだ証明書と秘密鍵情報を紐づけ、ID で管理します。

本 API で設定したセキュリティプロファイル ID は、R_CELLULAR_FirmUpgrade()および R_CELLULAR_FirmUpgradeBlocking()で使用します。

引数 *cert_valid_level* へ設定可能な値は、以下の通りです。

```
typedef enum
{
    CELLULAR_NO_CERT_VALIDATE                   = 0,    //certificate not validated
    CELLULAR_VALIDATE_CERT_EXPDATE              = 1,    //Validate certificate chain, validity period
    CELLULAR_VALIDATE_CERT_EXPDATE_CN           = 5
                                        //Validate certificate chain, validity period, common name
} e_cellular_cert_validate_level_t;
```

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};
st_cellular_certificate_t cert = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);

/* ホスト名指定(www.renesas.com など)で SSL サーバに接続する場合 */
ret = R_CELLULAR_ConfigSSLProfile(&cellular_ctrl,
                                  1,
                                  CELLULAR_VALIDATE_CERT_EXPDATE_CN,
                                  0,  // サーバに対応する CA 証明書インデックス=指定必須
                                  255,
                                  255);

/* クライアント証明書を要求するサーバに接続する場合 */
ret = R_CELLULAR_ConfigSSLProfile(&cellular_ctrl,
                                  1,
                                  CELLULAR_VALIDATE_CERT_EXPDATE,
                                  0,  //サーバに対応する CA 証明書インデックス
                                  1,  //サーバに対応するクライアント証明書インデックス=指定必須
                                  1); //サーバに対応する秘密鍵インデックス=指定必須

/* サーバに関係なく SSL 通信をしたい場合(クライアント証明書を要求しないサーバ) */
ret = R_CELLULAR_ConfigSSLProfile(&cellular_ctrl,
                                  1,
                                  CELLULAR_NO_CERT_VALIDATE,
                                  255,  //インデックス指定不要
                                  255,  //インデックス指定不要
                                  255); //インデックス指定不要
```

Special Notes

なし

3.40 R_CELLULAR_SoftwareReset()

RYZ014A Cellular モジュールを AT コマンドによりリセットします。

Format

```
e_cellular_err_t R_CELLULAR_SoftwareReset (
    st_cellular_ctrl_t * const p_ctrl
)
```

Parameters

p_ctrl (IN/OUT)　　*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールを AT コマンドによりリセットします。

再起動後の動作は R_CELLULAR_AutoConnectConfig()によって設定された値によって異なります。自動接続を無効に設定している場合は R_CELLULAR_Open()実行完了後の状態（図 1.3 の「Cellular モジュール・FIT モジュールの初期化完了」状態)となります。自動接続を有効に設定している場合は、本 API 正常終了後に自動的にアクセスポイントへ接続され、R_CELLULAR_APConnect()実行完了後の状態（図 1.3 の「アクセスポイントと接続」状態)となります。本 FIT モジュールの状態遷移図は、図 1.3 を参照してください。アクセスポイントへの自動接続は、一度 R_CELLULAR_APConnect()が正常に完了している必要があります。

PIN コードが設定されている SIM を使用する場合、本 API を実行後に R_CELLULAR_UnlockSIM()を実行して PIN コードを入力してください。詳細は、3.35 節を参照してください。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_AutoConnetConfig(&cellular_ctrl, CELLULAR_ENABLE_AUTO_CONNECT);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
ret = R_CELLULAR_SoftwareReset(&cellular_ctrl);
```

Special Notes

なし

3.41 R_CELLULAR_HardwareReset()

RYZ014A Cellular モジュールを RESET 端子でリセットします。

Format

```
e_cellular_err_t R_CELLULAR_HardwareReset (
    st_cellular_ctrl_t * const p_ctrl
)
```

Parameters

p_ctrl (IN/OUT)　　*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*
CELLULAR_ERR_RECV_TASK	*/* シリアル受信に失敗 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールを RESET 端子でリセットします。

再起動後の動作は R_CELLULAR_AutoConnectConfig()によって設定された値によって異なります。本 FIT モジュールの状態遷移図は、図 1.3 を参照してください。

自動接続を無効に設定している場合は、アクセスポイントから切断後の状態 (図 1.3 の「アクセスポイントから切断」状態)となります。本 API を実行後は、R_CELLULAR_Close()を実行後に R_CELLULAR_Open()を実行してください。

自動接続を有効に設定している場合は、本 API 正常終了後に自動的にアクセスポイントへ接続され、R_CELLULAR_APConnect()実行完了後の状態 (図 1.3 の「アクセスポイントと接続」状態)となります。アクセスポイントへの自動接続は、一度 R_CELLULAR_APConnect()が正常に完了している必要があります。

PIN コードが設定されている SIM を使用する場合、本 API を実行後に R_CELLULAR_UnlockSIM()を実行して PIN コードを入力してください。詳細は、3.35 節を参照してください。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_AutoConnetConfig(&cellular_ctrl, CELLULAR_ENABLE_AUTO_CONNECT);
ret = R_CELLULAR_APConnect(&cellular_ctrl, NULL);
ret = R_CELLULAR_HardwareReset(&cellular_ctrl);
```

Special Notes

なし

3.42 R_CELLULAR_FactoryReset()

RYZ014A Cellular モジュールを工場出荷時の設定に戻します。

Format

```
e_cellular_err_t R_CELLULAR_ FactoryReset (
    st_cellular_ctrl_t * const p_ctrl
)
```

Parameters

p_ctrl (IN/OUT)　　*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_MODULE_COM	*/* Cellular モジュールとの通信に失敗 */*
CELLULAR_ERR_MODULE_TIMEOUT	*/* Cellular モジュールからの応答がない */*
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	*/* 他の AT コマンドが実行中 */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*
CELLULAR_ERR_IRQ_OPEN	*/* IRQ 端子の初期化に失敗 */*
CELLULAR_ERR_RECV_TASK	*/* シリアル受信に失敗 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールを、工場出荷時の設定に戻します。本 API は、正常に処理が完了したことを確認するため、未使用の CID を使用します。そのため、1 つ以上の CID が未使用の状態で本関数を実行してください。すべての CID が使用されていた場合は、CID8 を使用します。CID の詳細は、RYZ014 Modules User's Manual: AT Command の「8.8 Define PDP Context: AT+CGDCONT」を参照してください。

本 API を実行後は、R_CELLULAR_Close()を実行後に R_CELLULAR_Open()を実行してください。

本 FIT モジュールは、OEM 復元ポイント作成に対応していません。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_FactoryReset(&cellular_ctrl);
```

Special Notes

なし

3.43 R_CELLULAR_RTS_Ctrl()

RYZ014A Cellular モジュールの RTS0 端子出力の制御を行います。

Format

```
e_cellular_err_t R_CELLULAR_RTS_Ctrl (
    st_cellular_ctrl_t * const p_ctrl,
    const uint8_t lowhigh
)
```

Parameters

p_ctrl (IN/OUT)　　*ユーザが宣言した st_cellular_ctrl_t 構造体へのポインタ*

lowhigh (IN)　　*RTS0 端子の出力状態*

Return values

CELLULAR_SUCCESS	*/* 正常終了 */*
CELLULAR_ERR_PARAMETER	*/* 引数が無効な値 */*
CELLULAR_ERR_NOT_OPEN	*/* Open 関数を実行していない */*
CELLULAR_ERR_OTHER_API_RUNNING	*/* 他の API が実行中 */*

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

Cellular モジュールの RTS0 端子出力の制御を行います。

RTS0 端子へ High 出力を設定する場合は *lowhigh* に 1 を、Low 出力を設定する場合は *lowhigh* に 0 を設定してください。

Reentrant

不可

Thread Safety

対応

Examples

```
e_cellular_err_t ret;
st_cellular_ctrl_t cellular_ctrl = {0};

ret = R_CELLULAR_Open(&cellular_ctrl, NULL);
ret = R_CELLULAR_RTS_Ctrl (&cellular_ctrl, 1);  //High 出力
ret = R_CELLULAR_RTS_Ctrl (&cellular_ctrl, 0);  //Low 出力
```

Special Notes

なし

4. 付録

4.1 動作確認環境

本 FIT モジュールの動作確認環境を表 4.1 に示します。

表 4.1 動作確認環境

項目		内容
統合開発環境		ルネサスエレクトロニクス製 e² studio Ver.2023-01
コンパイラ	CC-RX	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.05.00 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
	GCC	GCC for Renesas 8.3.0.202202-GNURX Toolchain
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		Rev1.12
使用ボード		Renesas CK-RX65N (型名：RTK5CK65N0SxxxxxBE)
使用コネクティビティモジュール (Cellular モジュール)		PMOD Expansion Board for RYZ014A (型名：RTKYZ014A0B00000BE) <Firmware revision> MR: 1.7 UE: 5.4.1.2 LR: 5.4.1.2-58697
RTOS	FreeRTOS	10.4.3-rx-1.0.1 (kernel only)
FIT	BSP FIT	Ver 7.10
	SCI FIT	Ver 4.30
	IRQ FIT	Ver 4.00

4.2 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)」
- e² 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 e² 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_cellular_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_cellular_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：アクセスポイントへ接続するため R_CELLULAR_APConnect()を実行すると、返り値で CELLULAR_ERR_AP_CONNECT_FAILED が返されアクセスポイントへの接続に失敗します。コールバック関数で R_CELLULAR_APConnect()実行中の URC を確認すると、”+CEREG: 80” URC が通知されていました。

A：通信に使用する LTE バンド設定が適切ではない可能性があります。R_CELLULAR_SetBand()を実行して、Cellular モジュールを使用する地域に対して適切なバンドを設定してください。詳細は、3.30 節を参照してください。

4.3　復帰処理

RYZ014A Cellular モジュールおよび本 FIT モジュールを使用時に本節に示す事象が発生した場合、ユーザは復帰処理を実行してください。

4.3.1　RYZ014A Cellular モジュールが^EXIT URC を通知した場合

RYZ014A Cellular モジュールは、^EXIT URC を通知して自己リセットする場合があります。ユーザは、^EXIT URC をコールバック関数で検出することを推奨します。^EXIT URC の通知を検出した場合、ユーザは以下の処理を実行してください。

(1) +SYSSTART URC を検出。
　　^EXIT URC 後の+SYSSTART URC をコールバック関数で検出します。
(2) R_CELLULAR_Close()を実行。
(3) R_CELLULAR_Open()を実行。

以上で、アクセスポイントへ接続可能な状態（図 1.3 の「Cellular モジュール・FIT モジュールの初期化完了」状態）へ復帰します。

4.3.2　RYZ014A Cellular モジュールが+SYSSTART URC を通知した場合

RYZ014A Cellular モジュールは、起動完了後に+SYSSTART URC を通知します。+SYSSTART URC は、R_CELLULAR_Open()を含む以下の API 実行中の再起動に伴って通知されます。

- R_CELLULAR_Open()
- R_CELLULAR_SetPSM()
- R_CELLULAR_SetOperator()
- R_CELLULAR_SetBand()
- R_CELLULAR_FirmUpgradeBlocking()
- R_CELLULAR_SoftwareReset()
- R_CELLULAR_HardwareReset()
- R_CELLULAR_FactoryReset()

Cellular モジュールが予期せず再起動した場合は、上記 API の実行中以外に+SYSSTART URC が通知されます。ユーザは、+SYSSTART URC をコールバック関数で検出することを推奨します。Cellular モジュールで予期しない再起動が発生した場合は、ユーザは以下の処理を実行してください。

(1) R_CELLULAR_Close()を実行。
(2) R_CELLULAR_Open()を実行。

以上で、アクセスポイントへ接続可能な状態（図 1.3 の「Cellular モジュール・FIT モジュールの初期化完了」状態）へ復帰します。

4.3.3　API でタイムアウトが発生した場合

API でタイムアウトが発生した場合、戻り値で CELLULAR_ERR_MODULE_TIMEOUT を返してユーザへ通知します。その場合は、ユーザは以下の処理を実行してください。

(1) R_CELLULAR_HardwareReset()を実行。
　　RYZ014A Cellular モジュールの RESETN 端子でリセットを実行します。
(2) R_CELLULAR_Close()を実行。
(3) R_CELLULAR_Open()を実行。

以上で、アクセスポイントへ接続可能な状態（図 1.3 の「Cellular モジュール・FIT モジュールの初期化完了」状態）へ復帰します。

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。）

RYZ014 Module System Integration Guide (R19AN0074)
（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。）

RYZ014 Modules User's Manual: AT Command (R11UZ0093)
（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。）

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。）

GNU-RX Compiler マニュアル
（最新版を下記のホームページから入手してください。）
https://llvm-gcc-renesas.com/ja/gnu-tools-manuals/gnu-compiler/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.04	2022/3/18	-	新規作成
1.05	2022/4/28	P.9	2.7 コンパイル時の設定 以下のマクロ定義を追加。 ・ CELLULAR_CFG_EX_TIMEOUT
		P.11	2.7 コンパイル時の設定 表 2.5 Configuration options (FreeRTOSConfig.h)を追加。 表 2.6 Configuration options (tx_user.h)を追加。
		P.18	3.3 R_CELLULAR_APConnect() アクセスポイント接続情報の設定方法を変更。
		P.19	3.3 R_CELLULAR_APConnect() 認証プロトコル設定を追加。
1.06	2022/06/01	P.6	1.2.3 API の概要 表 1.1 に以下の API を追加 ・R_CELLULAR_GetICCID() ・R_CELLULAR_GetIMEI() ・R_CELLULAR_GetIMSI() ・R_CELLULAR_GetPhonenum() ・R_CELLULAR_GetRSSI() ・R_CELLULAR_GetSVN() ・R_CELLULAR_Ping() ・R_CELLULAR_GetAPConnectState() ・R_CELLULAR_GetCellInfo() ・R_CELLULAR_AutoConnectConfig() ・R_CELLULAR_SoftwareReset()
		P.9	2.7 コンパイル時の設定 以下のマクロ定義を追加。 ・CELLULAR_CFG_AUTH_TYPE
		P.12	2.8 コードサイズ 各サイズを修正
		P.13	2.9 引数 以下の構造体を追加 ・st_cellular_iccid_t ・st_cellular_imei_t ・st_cellular_imsi_t ・st_cellular_phonenum_t ・st_cellular_rssi_t ・st_cellular_svn_t ・st_cellular_notice_t
		P.45~59	下記の新規 API ページを追加 ・3.17 R_CELLULAR_GetICCID() ・3.18 R_CELLULAR_GetIMEI() ・3.19 R_CELLULAR_GetIMSI() ・3.20 R_CELLULAR_GetPhonenum() ・3.21 R_CELLULAR_GetRSSI() ・3.22 R_CELLULAR_GetSVN() ・3.23 R_CELLULAR_Ping() ・3.24 R_CELLULAR_GetAPConnectState() ・3.25 R_CELLULAR_GetCellInfo() ・3.26 R_CELLULAR_AutoConnectConfig() ・3.27 R_CELLULAR_SoftwareReset()

1.07	2022/07/01	P.6	1.2.3 API の概要 表 1.1 に以下の API を追加 ・R_CELLULAR_SetBand() ・R_CELLULAR_GetPDPAddress()
		P.9	2.7 コンパイル時の設定 以下のマクロ定義を追加。 ・CELLULAR_CFG_NETWORK_NOTIFY_LEVEL ・CELLULAR_CFG_PSM_PREPARATION_TIME ・CELLULAR_CFG_RING_LINE_ACTIVE_TIME
		P.60~61	下記の新規 API ページを追加 ・3.28 R_CELLULAR_SetBand() ・3.29 R_CELLULAR_GetPDPAddress()
		プログラム	R_CELLULAR_GetPhonenum()で取得できる電話番号の最大桁数を 15 桁に修正
1.08	2022/09/30	P.7	1.2.3 API の概要 表 1.1 に以下の API を追加 ・R_CELLULAR_HardwareReset() ・R_CELLULAR_FactoryReset() ・R_CELLULAR_RTS_Ctrl()
		P.9~10	2.7 コンパイル時の設定 以下のマクロ定義を追加。 ・CELLULAR_CFG_PSM_WAKEUP_LATENCY ・CELLULAR_CFG_URC_CHARGET_ENABLED ・CELLULAR_CFG_URC_CHARGET_FUNCTION ・CELLULAR_CFG_IRQ_NUM
		P.14	2.9 引数 以下の構造体を追記。 ・st_cellular_notice_t ・st_cellular_edrx_config_t ・st_cellular_psm_config_t
		P.16~P.17	3.1 R_CELLULAR_Open() コールバック関数の仕様変更に伴い、内容を修正。 表 2.1 Configuration options (r_cellular_config.h)にてコールバック関数を設定する仕様に変更。
		P.18	3.2 R_CELLULAR_Close() R_CELLULAR_Close()の仕様変更に伴い、Description を修正。
		P.41~P.43	3.15 R_CELLULAR_SetEDRX() 引数の変更に伴い内容を修正。
		P.44~P.46	3.16 R_CELLULAR_SetPSM() 引数の変更に伴い内容を修正。
		P.53~P.54	3.23 R_CELLULAR_Ping() コールバック関数の仕様変更に伴い、内容を修正。
		P.55~P.56	3.24 R_CELLULAR_GetAPConnectState() コールバック関数の仕様変更に伴い、内容を修正。
		P.57~P.58	3.25 R_CELLULAR_GetCellInfo() コールバック関数の仕様変更に伴い、内容を修正。
		P.65	3.29 R_CELLULAR_SoftwareReset() 新規 API 実装に伴い 3.29 節へ移動。
		P.67~P.70	下記の API を追加 ・3.30 R_CELLULAR_HardwareReset() ・3.31 R_CELLULAR_FactoryReset() ・3.32 R_CELLULAR_RTS_Ctrl()
		P.72	下記の節を追加 ・4.3 復帰処理

1.09	2022/11/30	P.5	1.2.1 RYZ014A Cellular モジュールとの接続 図 1.1 を修正
		P.10~P.11	2.7 コンパイル時の設定 以下のマクロ定義のデフォルト値を修正 ・CELLULAR_CFG_AP_NAME ・CELLULAR_CFG_AUTH_TYPE 以下のマクロ定義を修正 ・CELLULAR_CFG_SCI_PRIORITY 以下のマクロ定義を追加 ・CELLULAR_CFG_CTS_SW_CTRL ・CELLULAR_CFG_CTS_PORT ・CELLULAR_CFG_CTS_PIN ・CELLULAR_CFG_PFS_SET_VALUE
		P.15	2.9 引数 以下の構造体を追記 ・st_cellular_ipaddr_t ・st_cellular_ping_cfg_t
		P.17~P.80	3. API 関数 R_CELLULAR_Open()を除く API 関数をスレッドセーフ化 API 関数のスレッドセーフ化に伴い、R_CELLULAR_Open()を除く各 API 関数の返り値（Return values）に下記を追加 ・CELLULAR_ERR_OTHER_API_RUNNING
		P.20	3.3 R_CELLULAR_APConnect() IPv6 対応に伴い内容を修正
		P.25	3.6 R_CELLULAR_CreateSocket() IPv6 対応に伴い内容を修正 返り値に下記を追加 ・CELLULAR_ERR_SIM_NOT_SUPPORT_IPV6
		P.27	3.7 R_CELLULAR_ConnectSocket() IPv6 対応に伴い内容を修正
		P.37	3.12 R_CELLULAR_DnsQuery() IPv6 対応に伴い内容を修正 返り値に下記を追加 ・CELLULAR_ERR_SIM_NOT_SUPPORT_IPV6
		P.49	3.17 R_CELLULAR_GetICCID() Description へ最大取得可能桁数を追記
		P.51	3.18 R_CELLULAR_IMEI() Description へ最大取得可能桁数を追記
		P.53	3.19 R_CELLULAR_IMSI() Description へ最大取得可能桁数を追記
		P.55	3.20 R_CELLULAR_GetPhonenum() Description へ最大取得可能桁数を追記
		P.61	3.23 R_CELLULAR_Ping() パラメータ設定可能化に伴い内容を修正
		P.71	3.28 R_CELLULAR_GetPDPAddress() IPv6 対応に伴い内容を修正 返り値に下記を追加 ・CELLULAR_ERR_SIM_NOT_SUPPORT_IPV6
1.10	2023/4/7	全体	AzureRTOS に関する記述を削除
		P.5~P.10	1.2 RYZ014A Cellular FIT モジュールの概要 実装タイプ C に関する記述を追加 WDT に関する記述を追加 1.2.1 の内容を一部修正 1.2.4 の図 1.3 を修正

		P.12	2.7 コンパイル時の設定 以下のマクロ定義のデフォルト値を修正 ・CELLULAR_CFG_ATC_RETRY_CGATT 以下のマクロ定義を修正 ・CELLULAR_CFG_ATC_RETRY_CGATT
		P.19~P.21	3.1 R_CELLULAR_Open() +SYSSTART URC を受信した場合の対処方法を追記 引数 p_cfg が NULL だった場合の初期値の説明を修正
		P.22~P.23	3.2 R_CELLULAR_Close() Return values に戻り値を追加 Description を修正 Examples を修正
		P.38	3.10 R_CELLULAR_SendSocket() 戻り値が CELLULAR_ERR_MODULE_TIMEOUT だった場合の対処方法を追記
		P.51~P.52	下記の API を追加 ・3.16 R_CELLULAR_GetEDRX()
		P.53~P.55	3.17 R_CELLULAR_SetPSM() Return values に戻り値を追加 Description を修正
		P.57~P.58	下記の API を追加 ・3.18 R_CELLULAR_GetPSM()
		P.80~P.81	下記の API を追加 ・3.29 R_CELLULAR_SetOperator()
		P.82	3.30 R_CELLULAR_SetBand() Description を一部修正
		P.86~P.102	下記の API を追加 ・3.32 R_CELLULAR_FirmUpgrade() ・3.33 R_CELLULAR_FirmUpgradeBlocking() ・3.34 R_CELLULAR_GetUpgradeState() ・3.35 R_CELLULAR_UnlockSIM() ・3.36 R_CELLULAR_WriteCertificate() ・3.37 R_CELLULAR_EraseCertificate() ・3.38 R_CELLULAR_GetCertificate() ・3.39 R_CELLULAR_ConfigSSLProfile()
		P.105	3.41 R_CELLULAR_HardwareReset() Return values に戻り値を追加
		P.107	3.42 R_CELLULAR_FactoryReset() Return values に戻り値を追加
		P.112	4.2 トラブルシューティング (3)を追加
		P.113	4.3 復帰処理 4.3.2 節を新規作成
1.11	2023/5/31	P.28	3.5 R_CELLULAR_Disconnect() ・Return values を修正
		P.30	3.6 R_CELLULAR_CreateSocket() ・Return values を修正 ・Description を修正
		P.34	3.8 R_CELLULAR_ShutdownSocket() ・Return values を修正
		P.36	3.9 R_CELLULAR_CloseSocket() ・Return values を修正
		P.38	3.10 R_CELLULAR_SendSocket() ・第 5 引数のタイムアウト設定 timeout_ms の最小値を 1 [ms]へ変更

		P.40	3.11 R_CELLULAR_ReceiveSocket() ・Return values を修正 ・Description を修正
		プログラム	・R_CELLULAR_SendSocket() - 内部処理でデータ送信エラーが発生した場合、タイムアウトを待たずに処理を抜け、返り値で送信済みバイト数を返すように修正 ・R_CELLULAR_ReceiveSocket() - 第 5 引数のタイムアウト設定 timeout_ms へ 0 (タイムアウトなし)を設定して本関数を実行中にアクセスポイントから切断された状態が一定時間続いた場合は、処理を終了し返り値で受信済みデータサイズを返すように修正
1.12	2024/7/22	P.12~P.13	2.7 コンパイル時の設定 以下のマクロ定義のデフォルト値を修正 ・CELLULAR_CFG_AUTH_TYPE ・CELLULAR_CFG_UART_SCI_CH ・CELLULAR_CFG_CTS_PORT ・CELLULAR_CFG_CTS_PIN ・CELLULAR_CFG_RTS_PORT ・CELLULAR_CFG_RTS_PIN ・CELLULAR_CFG_RESET_PORT ・CELLULAR_CFG_RESET_PIN ・CELLULAR_CFG_IRQ_NUM
		P.19	2 API 情報 ・2.12 節を新規作成
		P.27	3.3 R_CELLULAR_APConnect() ・アクセスポイント接続確認のリトライ中断方法を追記
		P.112	4.1 動作確認環境 ・表 4.1 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。