

RX ファミリ

RYZ014A 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 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 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)

以降、本モジュールを“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

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

関連ドキュメント

- [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] e² 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 e² 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)
- [9] RYZ014 Modules User's Manual: AT Command (R11UZ0093)

目次

1. 概要	5
1.1 RYZ014A Cellular FIT モジュールとは	5
1.2 RYZ014A Cellular FIT モジュールの概要	5
1.2.1 RYZ014A Cellular モジュールとの接続	5
1.2.2 ソフトウェア構成	6
1.2.3 API の概要	7
1.2.4 状態遷移図	8
2. API 情報	9
2.1 ハードウェアの要求	9
2.2 ソフトウェアの要求	9
2.3 サポートされているツールチェーン	9
2.4 使用する割り込みベクタ	9
2.5 ヘッダファイル	9
2.6 整数型	9
2.7 コンパイル時の設定	10
2.8 コードサイズ	13
2.9 引数	14
2.10 戻り値	16
2.11 FIT モジュールの追加方法	16
2.12 RTOS の使用要件	16
3. API 関数	17
3.1 R_CELLULAR_Open()	17
3.2 R_CELLULAR_Close()	19
3.3 R_CELLULAR_APConnect()	20
3.4 R_CELLULAR_IsConnected()	22
3.5 R_CELLULAR_Disconnect()	23
3.6 R_CELLULAR_CreateSocket()	25
3.7 R_CELLULAR_ConnectSocket()	27
3.8 R_CELLULAR_ShutdownSocket()	29
3.9 R_CELLULAR_CloseSocket()	31
3.10 R_CELLULAR_SendSocket()	33
3.11 R_CELLULAR_ReceiveSocket()	35
3.12 R_CELLULAR_DnsQuery()	37
3.13 R_CELLULAR_GetTime()	39
3.14 R_CELLULAR_SetTime()	41
3.15 R_CELLULAR_SetEDRX()	43
3.16 R_CELLULAR_SetPSM()	46
3.17 R_CELLULAR_GetICCID()	49
3.18 R_CELLULAR_GetIMEI()	51
3.19 R_CELLULAR_GetIMSI()	53
3.20 R_CELLULAR_GetPhonenum()	55
3.21 R_CELLULAR_GetRSSI()	57
3.22 R_CELLULAR_GetSVN()	59

3.23	R_CELLULAR_Ping()	61
3.24	R_CELLULAR_GetAPConnectState()	63
3.25	R_CELLULAR_GetCellInfo()	65
3.26	R_CELLULAR_AutoConnectConfig()	67
3.27	R_CELLULAR_SetBand()	69
3.28	R_CELLULAR_GetPDPAddress()	71
3.29	R_CELLULAR_SoftwareReset()	73
3.30	R_CELLULAR_HardwareReset()	75
3.31	R_CELLULAR_FactoryReset()	77
3.32	R_CELLULAR_RTS_Ctrl()	79
4.	付録	81
4.1	動作確認環境	81
4.2	トラブルシューティング	81
4.3	復帰処理	82
4.3.1	RYZ014A Cellular モジュールがEXIT URC を通知した場合	82
4.3.2	API でタイムアウトが発生した場合	82
5.	参考ドキュメント	83
	改訂記録	84

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 2 種類の実装タイプがありますが、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 :
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。
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モジュールが担当する。

1.2.1 RYZ014A Cellular モジュールとの接続

RX65N Cloud Kit と RYZ014A Cellular モジュールの接続例を、図 1.1 に示します。本 FIT モジュールは、PMOD_9、PMOD_10 端子を使用しません。

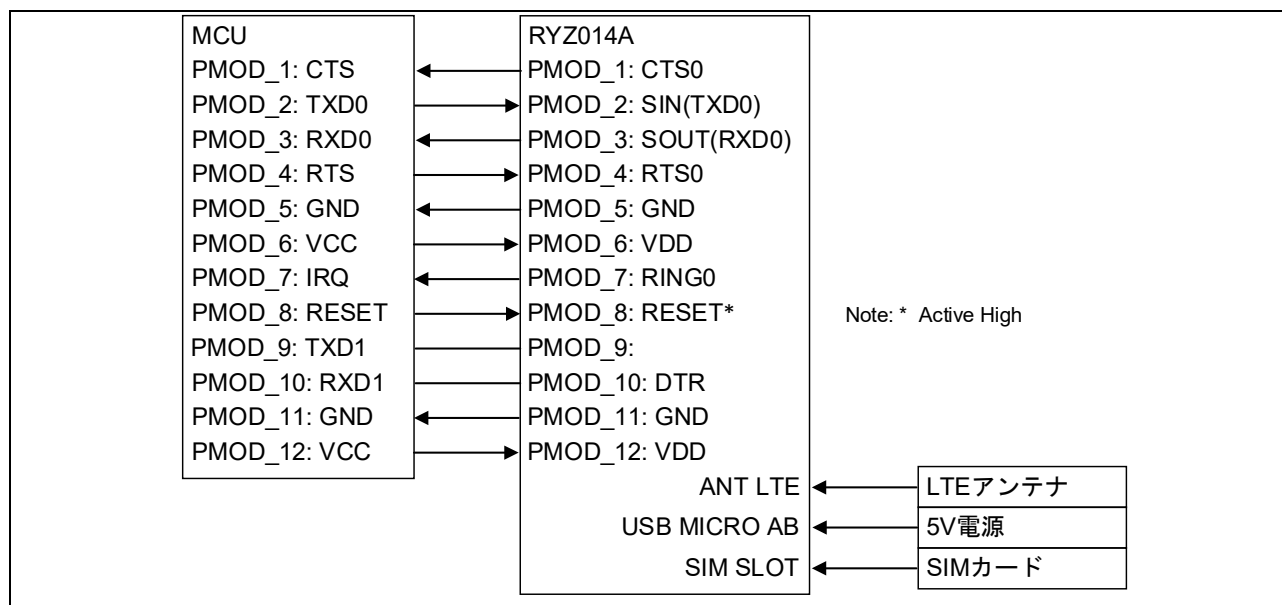


図 1.1 RX65N Cloud Kit と RYZ014A Cellular モジュールの接続例

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。

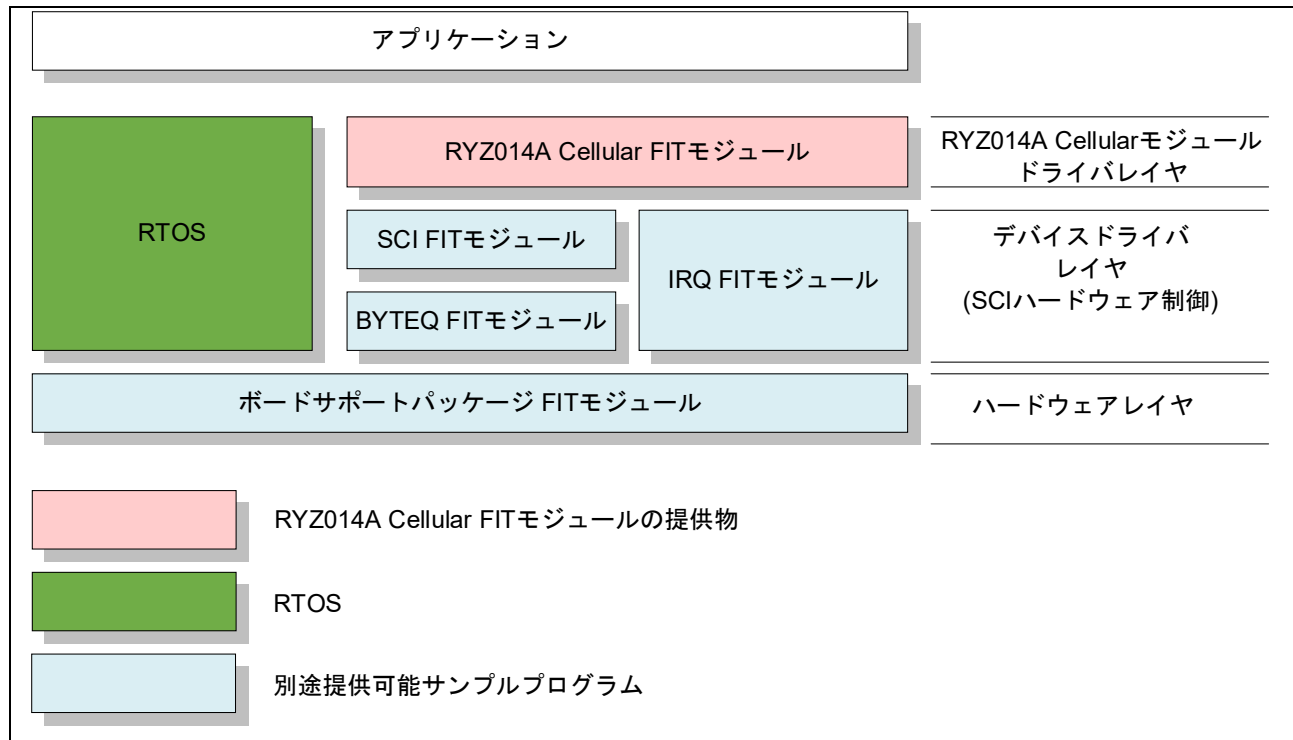


図 1.2 ソフトウェア構成図

- (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り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 と AzureRTOS 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_SetPSM	PSM (Power Saving Mode)の設定を行います。
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	SVN (Software Version Number)と、リビジョンを取得します。
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_SetBand	通信で使用する周波数帯を設定します。
R_CELLULAR_GetPDPAddress	PDP アドレスを取得します。
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。

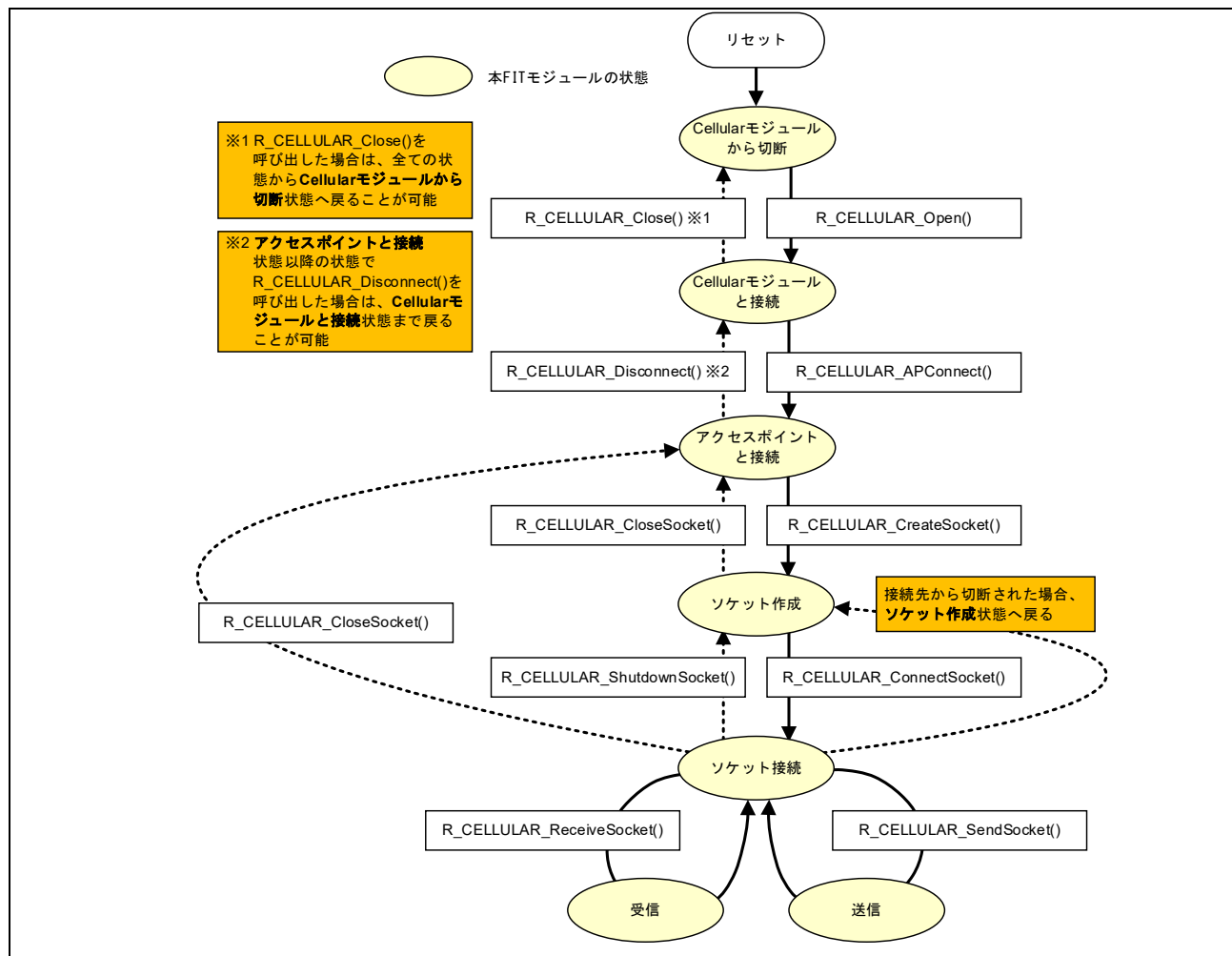


図 1.3 RYZ014A Cellular FIT モジュールの状態遷移図

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 “globaldata.iot”	接続するアクセスポイント名を指定します。 使用する SIM カード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 カード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 カード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_AP_PIN_CODE ※デフォルトは “”	使用する SIM カードの 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 “1”	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 カード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_GATT ※デフォルトは “100”	アクセスポイントへの接続に失敗した場合にリトライする回数です。 “0~100”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_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 で使用することができます。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 “0”	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 0 を使用する場合があります。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 出力とする場合があります。
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 “2”	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 22 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 “2”	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 22 を使 用する場合があります。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 “2”	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 22 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 22 を使 用する場合があります。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 “D”	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 D0 を使用する場合があります。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 “0”	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 D0 を使用する場合があります。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 “4”	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、IRQ4 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。オプションの詳細については、「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 <code>r_sci_rx_config.h</code>	
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	チャン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ンネルに 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	チャン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 で指定したチャン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	チャン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 で指定したチャン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。オプションの詳細については、「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 <code>r_irq_rx_config.h</code>	
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ンネルを指定します。 Cellular モジュールの RING 端子と接続するチャンネルに“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。オプションの詳細については、「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 <code>r_bsp_config.h</code>	
BSP_CFG_RTOS_USED ※デフォルト値は“0”	リアルタイム OS の種類を選択します。 本 FIT モジュールを使用する場合は以下を設定してください。 FreeRTOS の場合 : “1” AzureRTOS の場合 : “5”

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 (FreeRTOS の場合)または libs/threadx/tx_user.h (AzureRTOS の場合)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、AzureRTOS に対するオプション名および設定値に関する説明を表 2.6 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。

表 2.6 Configuration options (tx_user.h)

Configuration options in tx_user.h	
USE_TX_TIMER_TICKS_PER_SECOND ※デフォルト値は"0"	RTOS tick 割り込み周期のユーザ設定を有効にします。 本 FIT モジュールを使用する場合は、"1"に設定してください。
TX_TIMER_TICKS_PER_SECOND ※デフォルト値は 100	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、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 2.7 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 2.7 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 rev1.09

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.04.00

(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)

コンフィギュレーションオプション: デフォルト設定

表 2.7 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 35k 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 (全ての 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`

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`

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

- `st_cellular_datetime_t`

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

- `st_cellular_edrx_config_t`

PSM 設定・取得構造体 (`R_CELLULAR_SetPSM()`で使用)

- `st_cellular_psm_config_t`

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

- `st_cellular_iccid_t`

IMEI 取得構造体 (`R_CELLULAR_GetIMEI()`で使用)

- `st_cellular_imei_t`

IMSI 取得構造体 (`R_CELLULAR_GetIMSI()`で使用)

- `st_cellular_imsi_t`

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

- `st_cellular_phonenum_t`

RSSI 取得構造体 (`R_CELLULAR_GetRSSI()`で使用)

- `st_cellular_rssi_t`

SVN 取得構造体 (`R_CELLULAR_GetSVN()`で使用)

- `st_cellular_svn_t`

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

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

- st_cellular_ipaddr_t

PING 設定構造体 (R_CELLULAR_Ping()で使用)

- st_cellular_ping_cfg_t

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) e² 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
e² 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: e² studio 編 (R20AN0451)」を参照してください。
- (2) e² 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
e² 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 e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)」を参照してください。
- (3) CS+上でスマート・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 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

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

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

RYZ014A Cellular 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。

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。

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)

p_cfg へ NULL を設定した場合に使用される初期値 :

```
<baud_rate = 921600 / ap_gatt_retry_count = 100 / sci_timeout = 10000 / tx_process_size = 1500
/ rx_process_size = 1500 / packet_data_size = 0 / exchange_timeout = 60
```

```
/ connect_timeout = 200 / send_timeout = 10 / creatable_socket = 6 >
```

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 カードの PIN コード  
    921600,           //モジュールとの通信ボーレート (921600 を推奨)  
    0,                //AP への接続リトライ上限回数  
    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()

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

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

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。

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);
```

Special Notes

なし

3.3 R_CELLULAR_APConnect()

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*」へのポインタ

Return values

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

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 3.1 に示すマクロへ適切な値を設定する。(詳細は表 2.1 参照)
 ※*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 3.2 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 3.3 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。

接続のリトライは、*CELLULAR_CFG_AT_COMMAND_RETRY_GATT* マクロの値または *st_cellular_cfg_t* 構造体のメンバ *ap_gatt_retry_count* の値だけ 1 秒間隔で実行されます。

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、RYZ014A 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 RYZ014A Cellular 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 3.1 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 カードの PIN コード
CELLULAR_CFG_AUTH_TYPE	認証プロトコル (0: None、1: PAP、2: CHAP)
CELLULAR_CFG_AT_COMMAND_RETRY_GATT	アクセスポイントへの接続リトライ上限回数

表 3.2 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 カードの PIN コード
uint8_t	ap_gatt_retry_count	アクセスポイントへの接続リトライ上限回数

表 3.3 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

なし

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

<i>CELLULAR_SUCCESS</i>	<i>/* AP へ接続中 */</i>
<i>CELLULAR_ERR_PARAMETER</i>	<i>/* 引数が無効な値 */</i>
<i>CELLULAR_ERR_NOT_OPEN</i>	<i>/* Open 関数を実行していない */</i>
<i>CELLULAR_ERR_NOT_CONNECT</i>	<i>/* AP に未接続 */</i>
<i>CELLULAR_ERR_OTHER_API_RUNNING</i>	<i>/* 他の API が実行中 */</i>

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

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

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

なし

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

<i>p_ctrl</i> (IN/OUT)	ユーザが宣言した <i>st_cellular_ctrl_t</i> 構造体へのポインタ
<i>protocol_type</i> (IN)	プロトコルタイプ (TCP = 6 を指定)
<i>ip_version</i> (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	<i>/* 正常終了 */</i>
CELLULAR_ERR_PARAMETER	<i>/* 引数が無効な値 */</i>
CELLULAR_ERR_NOT_OPEN	<i>/* Open 関数を実行していない */</i>
CELLULAR_ERR_MODULE_COM	<i>/* Cellular モジュールとの通信に失敗 */</i>
CELLULAR_ERR_MODULE_TIMEOUT	<i>/* Cellular モジュールからの応答がない */</i>
CELLULAR_ERR_NOT_CONNECT	<i>/* AP に接続していない */</i>
CELLULAR_ERR_SOCKET_CREATE_LIMIT	<i>/* ソケット作成数が限界を超えた */</i>
CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING	<i>/* 他の AT コマンドが実行中 */</i>
CELLULAR_ERR_OTHER_API_RUNNING	<i>/* 他の API が実行中 */</i>
CELLULAR_ERR_SIM_NOT_SUPPORT_IPV6	<i>/* SIM が IPv6 非対応 */</i>

※*ip_version* で IPv6 が指定された場合、かつ SIM カードが IPv6 非対応の場合のみ出力

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。*protocol_type* に対しては CELLULAR_PROTOCOL_TCP (6)を、*ip_version* に対しては CELLULAR_PROTOCOL_IPV4 (4)をまたは CELLULAR_PROTOCOL_IPV6 (6)指定してください。

設定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。

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。

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_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

<i>p_ctrl</i> (IN/OUT)	ユーザが宣言した <i>st_cellular_ctrl_t</i> 構造体へのポインタ
<i>socket_no</i> (IN)	ソケット番号
<i>p_ip_addr</i> (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")
<i>port</i> (IN)	接続先ポート番号

Return values

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

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

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、*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_addr[] = "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(&p_ctrl, socket_no, ip_addr, 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

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

Return values

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

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

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(&p_ctrl, socket_no, ip_adder, port_no);
    ret = R_CELLULAR_ShutdownSocket(&p_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

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

Return values

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

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

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(&p_ctrl, socket_no, ip_adder, port_no);
    ret = R_CELLULAR_CloseSocket(&p_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

<i>p_ctrl</i> (IN/OUT)	ユーザが宣言した <i>st_cellular_ctrl_t</i> 構造体へのポインタ
<i>socket_no</i> (IN)	ソケット番号
<i>data</i> (IN)	送信データへのポインタ
<i>length</i> (IN)	送信データサイズ (1 以上)
<i>timeout_ms</i> (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

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

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

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

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

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(&p_ctrl, socket_no, ip_adder, port_no);
    ret = R_CELLULAR_SendSocket(&p_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

```
e_cellular_err_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

<i>p_ctrl</i> (IN/OUT)	ユーザが宣言した <i>st_cellular_ctrl_t</i> 構造体へのポインタ
<i>socket_no</i> (IN)	ソケット番号
<i>data</i> (IN)	受信データへのポインタ
<i>length</i> (IN)	受信データサイズ (1 以上)
<i>timeout_ms</i> (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

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

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

socket_no で指定したソケットから、受信データ格納領域 *data* へ、*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。

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(&p_ctrl, socket_no, ip_adder, port_no);
    ret = R_CELLULAR_ReceiveSocket(&p_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

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

Return values

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

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、*domain_name* で指定したドメインの、*ip_version* で指定した IP バージョンの IP アドレスを取得し、*p_addr* へ格納します。

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。*ip_version* で IPv6 が指定され、かつ SIM カードが IPv6 非対応の場合は、*CELLULAR_ERR_SIM_NOT_SUPPORT_IPV6* を返します。

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

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

Return values

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

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

RYZ014A 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。

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）となっているため、日時情報を使用する場合は *R_CELLULAR_SetTime()* を実行して現在日時を設定してください。

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

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

Return values

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

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

p_time に格納されている日時情報を RYZ014A Cellular モジュールへ設定します。

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()

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

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

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_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。

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。

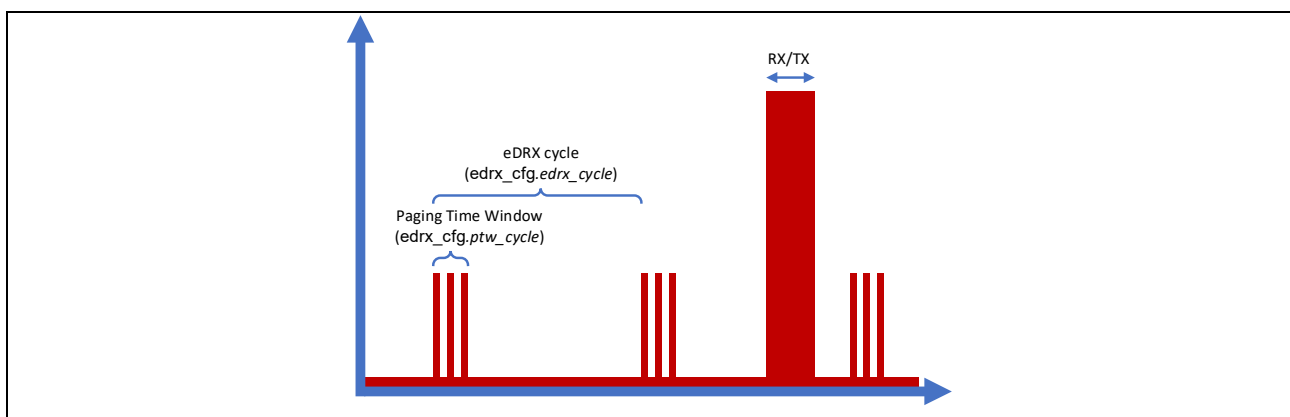


図 3.1 eDRX パラメータ

3.16 R_CELLULAR_SetPSM()

Cellular モジュールへ、PSM (Power Saving Mode)の設定を行います。

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

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

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_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。

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 cycle)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_EDRX_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。

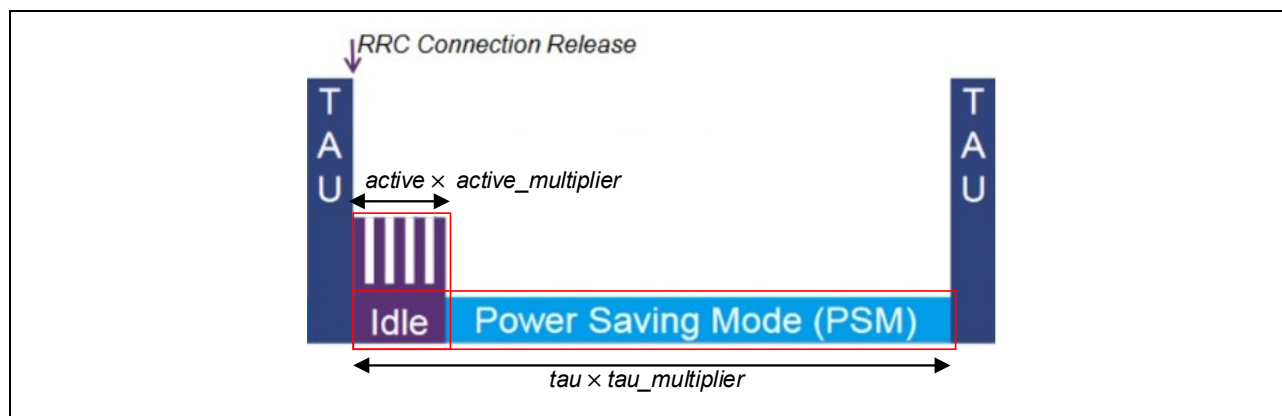


図 3.2 PSM パラメータ

3.17 R_CELLULAR_GetICCID()

SIM カードが保持する ICCID (IC Card Identifier、SIM カード識別番号)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

<i>p_ctrl</i> (IN/OUT)	ユーザが宣言した <i>st_cellular_ctrl_t</i> 構造体へのポインタ
<i>p_iccid</i> (IN/OUT)	取得した ICCID を格納する構造体へのポインタ

Return values

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

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

SIM カードに設定されている ICCID を取得し、引数 *p_iccid* へ設定した *st_cellular_iccid_t* 構造体変数に格納します。取得可能な ICCID は、最大 22 桁です。

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

3.18 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

<i>p_ctrl</i> (IN/OUT)	ユーザが宣言した <i>st_cellular_ctrl_t</i> 構造体へのポインタ
<i>p_imei</i> (IN/OUT)	取得した IMEI を格納する構造体へのポインタ

Return values

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

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* 構造体変数に格納します。取得可能な IMEI は、最大 15 桁です。

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

3.19 R_CELLULAR_GetIMSI()

SIM カードが保持する 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

<i>p_ctrl</i> (IN/OUT)	ユーザが宣言した <i>st_cellular_ctrl_t</i> 構造体へのポインタ
<i>p_imsi</i> (IN/OUT)	取得した IMSI を格納する構造体へのポインタ

Return values

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

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

SIM カードに設定されている IMSI を取得し、引数 *p_imsi* へ設定した *st_cellular_imsi_t* 構造体変数に格納します。取得可能な IMSI は、最大 15 桁です。

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

3.20 R_CELLULAR_GetPhonenum()

SIM カード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

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

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

Properties

r_cellular_if.h にプロトタイプ宣言されています。

Description

SIM カード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* 構造体変数に格納します。取得可能な電話番号は、最大 15 桁です。

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

3.21 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

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

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

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* 構造体変数に格納します。

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。

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

3.22 R_CELLULAR_GetSVN()

RYZ014A Cellular モジュールが保持する SVN・リビジョン情報を取得します。

Format

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

Parameters

<i>p_ctrl</i> (IN/OUT)	ユーザが宣言した <i>st_cellular_ctrl_t</i> 構造体へのポインタ
<i>p_svn</i> (IN/OUT)	取得した SVN・リビジョンを格納する構造体へのポインタ

Return values

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

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)・リビジョン情報を取得し、引数 *p_svn* へ設定した *st_cellular_svn_t* 構造体変数に格納します。

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

3.23 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

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

Return values

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

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 3.4 参照)

- (1) *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 *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 3.4 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

3.24 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

<i>p_ctrl</i> (IN/OUT)	ユーザが宣言した <i>st_cellular_ctrl_t</i> 構造体へのポインタ
<i>level</i> (IN)	自動通知レベルの設定
<i>p_result</i> (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

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

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。

また、*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;
```

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

3.25 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

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

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 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。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

3.26 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

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

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_AutoConnctConfig(&cellular_ctrl,  
                                   CELLULAR_ENABLE_AUTO_CONNECT);
```

Special Notes

なし

3.27 R_CELLULAR_SetBand()

使用する周波数帯を設定します。

Format

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

Parameters

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

Return values

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

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周波数帯を設定します。引数 *p_band* へ設定するバンドリストは、文字列で入力してください。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周波数帯は“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。

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(&g_cellular_ctrl, "1,2,3");
```

Special Notes

なし

3.28 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

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

Return values

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

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 カードが IPv6 非対応などにより 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。

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

3.29 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

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

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 モジュールと接続」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、*R_CELLULAR_APConnect()* が正常に完了している必要があります。

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

Special Notes

なし

3.30 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

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

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。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 モジュールと接続」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、*R_CELLULAR_APConnect()*が正常に完了している必要があります。

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

Special Notes

なし

3.31 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

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

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 が正常に完了した場合、*R_CELLULAR_Open()* 実行完了後の状態へと推移します。

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。OEM 復元ポイントが未作成の Cellular モジュールを使用する場合、本 API は *CELLULAR_ERR_MODULE_COM* を返しますが、RYZ014A の初期状態へ初期化されます。

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_FactoryReset(&cellular_ctrl);
```

Special Notes

なし

3.32 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

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

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 e ² studio Ver.2022-04
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 V3.04.00 コンパイルオプション: 統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	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		Rev1.09
使用ボード		Renesas RX65N Cloud Kit (型名: RTK5RX65N0SxxxxxBE)
		Renesas RX72N Envision Kit (型名: RTK5RX72N0C00000BJ)
RTOS	FreeRTOS	10.4.3-rx-1.0.1 (kernel only)
	Azure RTOS	v6.1.12_rel-rx-1.0.0
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。

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 モジュールと接続」状態）へ復帰します。

4.3.2 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、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。

以上で、アクセスポイントへ接続可能な状態（図 1.3 の「Cellular モジュールと接続」状態）へ復帰します。

5. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート／テクニカルニュース

（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：開発環境

RX ファミリ CC-RX コンパイラ ユーザーズマニュアル (R20UT3248)

（最新版をルネサス エレクトロニクスホームページから入手してください。）

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

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、入力信号や入出力ブルアップ電源を入れないでください。入力信号や入出力ブルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後、に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

6. 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、 V_{IL} (Max.) から V_{IH} (Min.) までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、 V_{IL} (Max.) から V_{IH} (Min.) までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違くと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

高品質水準： 輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。

7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限られません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因したまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものいたします。
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。