RX ファミリ
バッテリバックアップ機能モジュール Firmware Integration Technology

要旨

本アプリケーションノートは、Firmware Integration Technology（FIT）を使用したバッテリバックアップ機能モジュールについて説明します。

本モジュールは、バッテリバックアップ電源電圧や VBATT 端子電圧の低下の有無をユーザに通知します。その内容に応じて、リアルタイムクロックの値が保証できるか、VBATT 端子電圧が低下していないかを判断できます。

本書では以後、本モジュールのことをバッテリバックアップ機能 FIT モジュールと呼称します。

動作確認デバイス

- RX230 グループ
- RX231 グループ
- RX23W グループ
- RX671 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。

対象コンパイラ

- Renesas Electronics C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX

各コンパイラの動作確認内容については「4.1 動作確認環境」を参照してください。

関連ドキュメント

- 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）

目次

1. 概要

1.1 バッテリバックアップ機能 FIT モジュールとは

本モジュールは API として、プロジェクトに組み込んで使用します。本モジュールの組み込み方については、「2.12 FIT モジュールの追加方法」を参照してください。

1.2 バッテリバックアップ機能 FIT モジュールの概要

本モジュールを使用すると、VBATT 端子電圧低下検出機能の設定や、バッテリバックアップ機能の状態を読み出すことができます。また、コールバック関数の引数で次の 5 つの状況を判別できます。

< 共通 >
(1) バッテリバックアップ電源電圧低下の検出なし
(2) バッテリバックアップ電源電圧低下の検出あり
< RX230、RX231、RX23W >
(3) VBATT 端子電圧低下の検出によるノンマスカブル割り込みの発生中
(4) VBATT 端子電圧低下の検出によるマスカブル割り込みの発生中
< RX671 >
(5) 改ざん検出によるマスカブル割り込みの発生中

1.2.1　API 関数の仕様

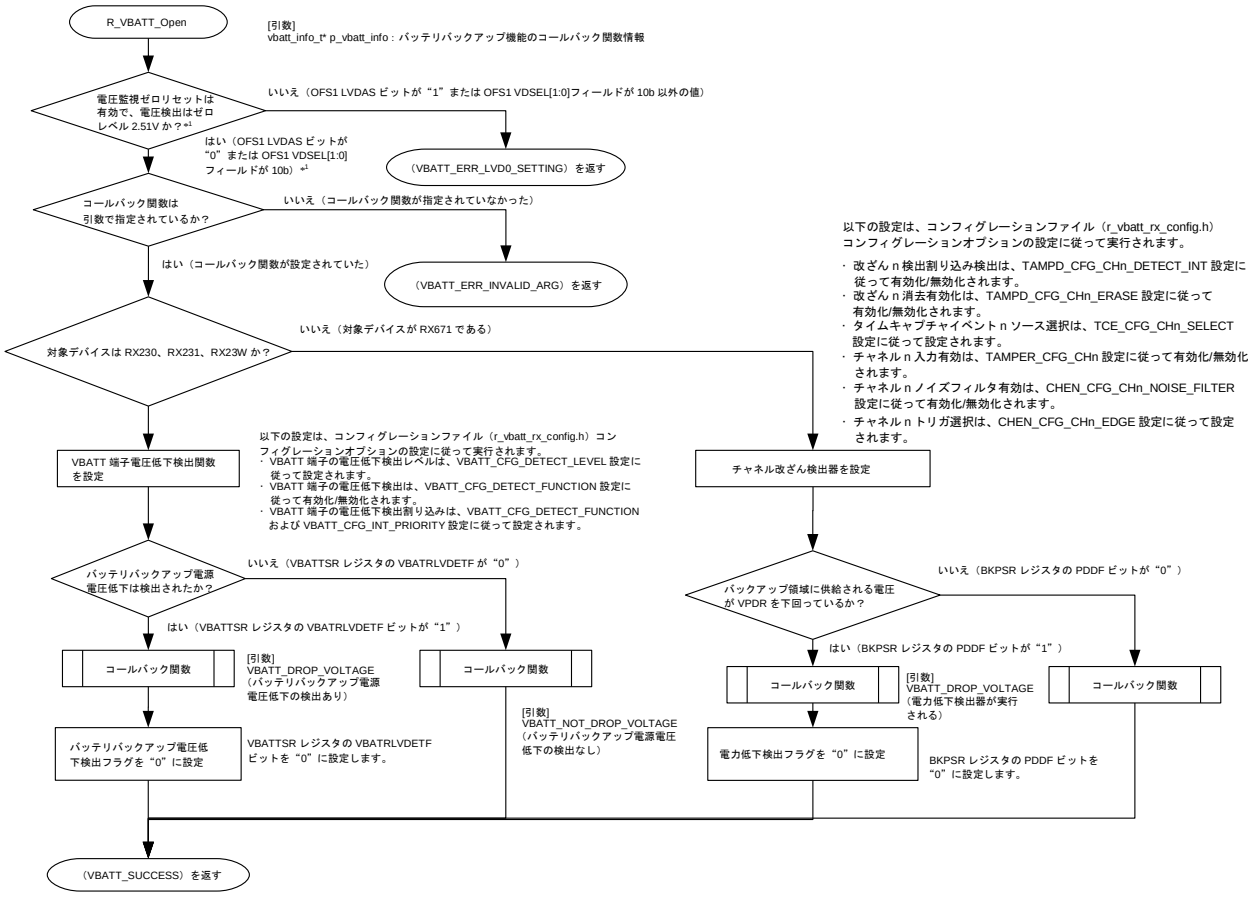
R_VBATT_Open()関数をコールすると、電圧監視 0 の設定が正しいか、引数にコールバック関数が設定されているかを判定し、不正である場合は、エラーを返し、関数の実行を終了します。正しく設定できている場合は、コンフィグレーションファイル（r_vbatt_rx_config.h）のコンフィグレーションオプションの設定に従って、VBATT 端子電圧低下検出機能（RX230、RX231、RX23W）または改ざん検出（RX671）の設定を行います。その後、バッテリバックアップ電源の電圧（VCC 端子および VBATT 端子）が低下したか（VBATTSR レジスタの VBATRLVDETF ビット（RX230、RX231、RX23W）または BKPSR レジスタの PDDF ビット（RX671））を判定し、その結果に応じて以下の処理を行います。

バッテリバックアップ電源電圧低下の検出ありの場合は、引数を VBATT_DROP_VOLTAGE にして、コールバック関数を呼び出します。コールバック関数呼び出し後は、バッテリバックアップ電源電圧低下の検出フラグを“0”（バッテリバックアップ電源電圧低下は未検出）にします。

バッテリバックアップ電源電圧低下の検出なしの場合は、引数を VBATT_NOT_DROP_VOLTAGE にして、コールバック関数を呼び出します。

図1.1 に R_VBATT_Open()関数のフローチャートを示します。

コンフィグレーションオプションおよびコールバック関数の引数で使用されるマクロ定義については、「2.7 コンパイル時の設定」、「2.11 コールバック関数」を参照してください。



【注】　*1 RX231 マイクロコントローラの場合。RX671 の場合、電圧検出レベル（OFS1.VDSEL ビット）は 2.94V（01b）、2.87V（10b）、2.80V（11b）のいずれかから選択できます。

図1.1　R_VBATT_Open()関数のフローチャート

R_VBATT_Control()関数をコールすると、引数の設定値が正しく設定されているかが判定され、不正である場合は、エラーが返されて関数の実行は終了します。引数の設定が有効な場合、引数に設定した値に従って、VBATT 端子電圧低下検出機能の有効/無効および検出レベル（RX230、RX231、RX23W）または改ざん検出（RX671）が設定され、割り込みも設定されます。

図1.2 に R_VBATT_Control()関数のフローチャートを示します。

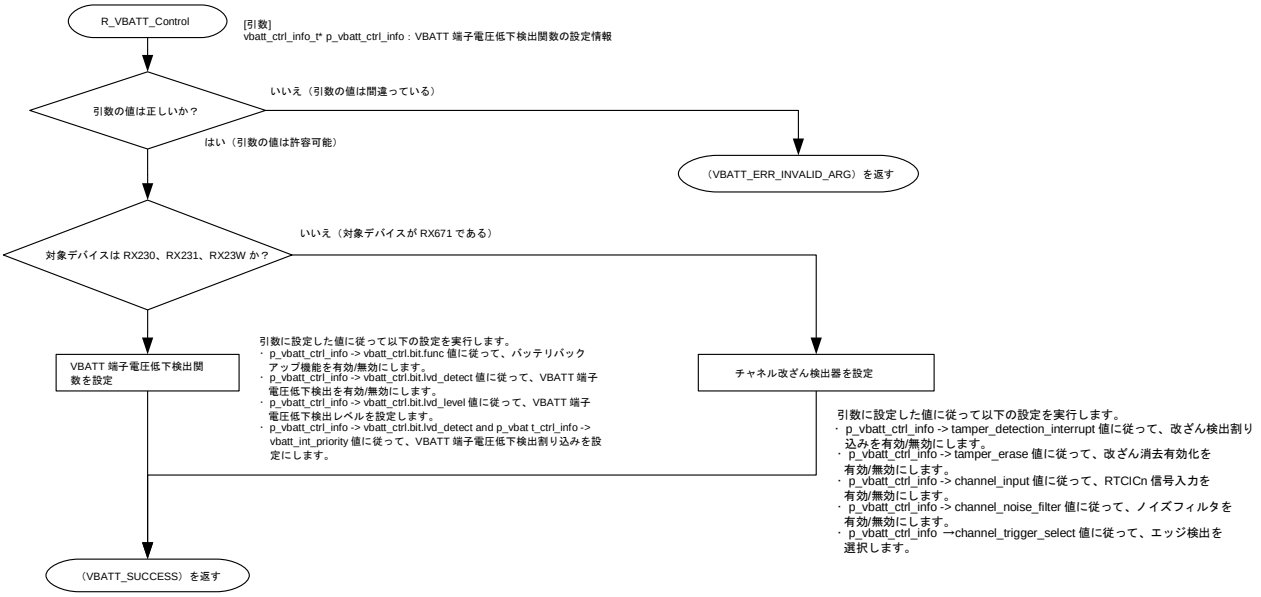


図1.2　R_VBATT_Control()関数のフローチャート

R_VBATT_GetStatus 関数をコールすると、VBATT 端子電圧低下検出は有効か、引数の設定値が正しく設定されているかが判定され、不正である場合は、エラーが返されて関数の実行は終了します。引数の設定が有効な場合、VBATT ステータスレジスタ（VBATTSR）（RX230、RX231、RX23W）または（TAMPIMR および TAMPSR）（RX671）の値が読み出されます。読み出された値は、引数として受領されたアドレスに保存されます。

図1.3 に R_VBATT_GetStatus()関数のフローチャートを示します。

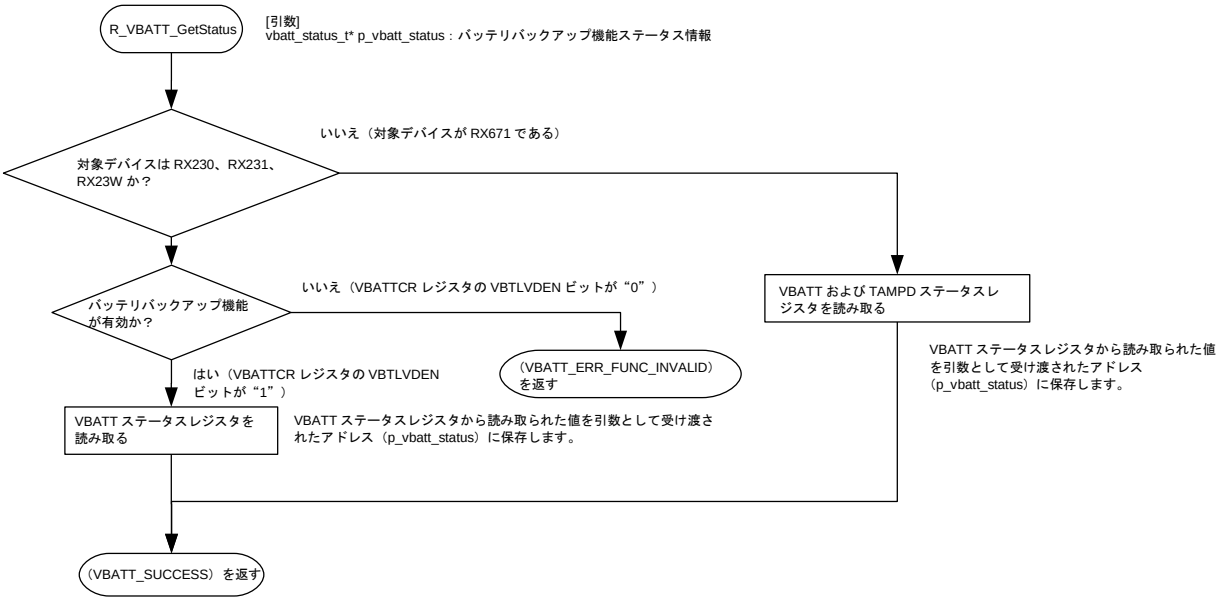


図1.3　R_VBATT_GetStatus()関数のフローチャート

1.2.2　割り込みの仕様

RX230、RX231、RX23W の場合、VBATT 端子電圧低下検出時の割り込みをマスカブル割り込みに設定にして、マスカブル割り込みが発生すると、引数を VBATT_MASKABLE_INTERRUPT にして、コールバック関数を呼び出します。

RX671 の場合、改ざん検出割り込みが発生すると、VBATT_TAMPER_CH0_INTERRUPT、VBATT_TAMPER_CH1_INTERRUPT、VBATT_TAMPER_CH2_INTERRUPT（RX671）を引数としてコールバック関数がコールされます。

図1.4 に r_vbatt_isr()関数のフローチャートを示します。

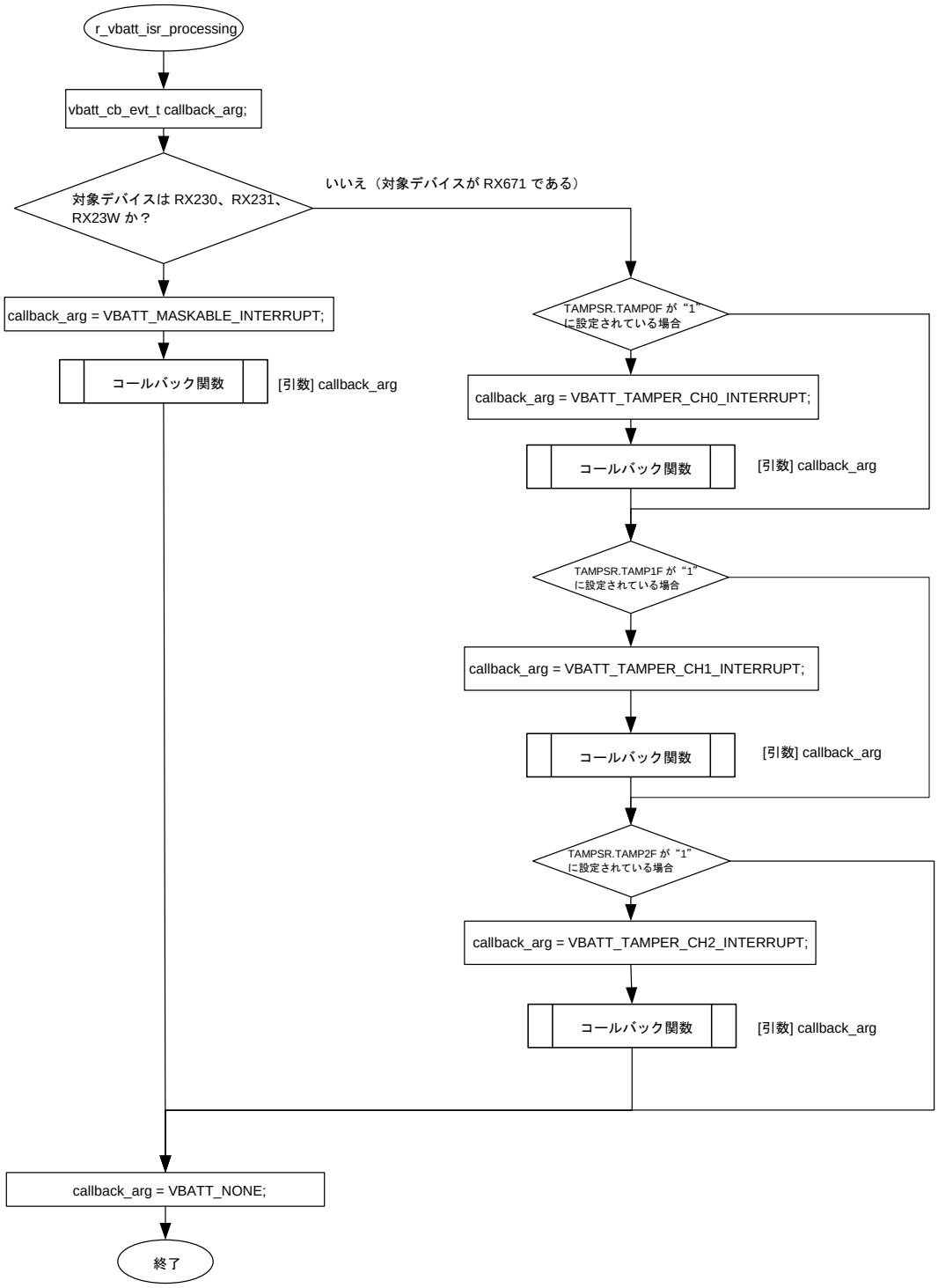


図1.4　r_vbatt_isr()関数のフローチャート

VBATT 端子電圧低下検出時の割り込みをノンマスカブル割り込みに設定にして、ノンマスカブル割り込みが発生すると、引数を VBATT_NON_MASKABLE_INTERRUPT（RX230、RX231、RX23W）にして、コールバック関数を呼び出します。

図1.5 に r_vbatt_nmi_isr()関数のフローチャートを示します。

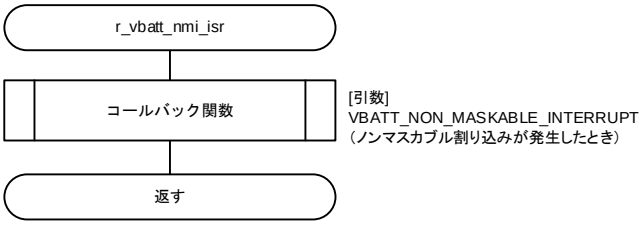


図1.5　r_vbatt_nmi_isr()関数のフローチャート

1.3　VBATT FIT モジュールを使用する

VBATT FIT モジュールを C++プロジェクト内で使用する

C++プロジェクトでは、FIT VBATT モジュールのインタフェースヘッダファイルを extern “C”の宣言に追加してください。

```
Extern “C”
{
#include “r_smc_entry.h”
#include “r_vbatt_rx_if.h”
}
```

1.4　API の概要

表1.1 に本モジュールに含まれる API 関数を示します。

表1.1　API 関数

関数	関数説明
R_VBATT_Open()	デバイス RX230、RX231、RX23W では、コンフィグレーションオプションの設定に従って、VBATT 端子電圧低下検出機能の有効/無効、検出レベル、割り込みを設定します。その後、バッテリバックアップ電源電圧低下の有無を判定し、コールバック関数を呼び出します。 デバイス RX671 では、コンフィグレーションオプションの設定に従って、TAMPD 関数の有効/無効、検出レベル、タイムキャプチャイベント、チャネル入力、チャネルノイズフィルタ、割り込みを設定します。その後、バックアップ領域の電圧が VPDR を下回っているかどうかを判定し、コールバック関数を呼び出します。
R_VBATT_Control()	デバイス RX230、RX231、RX23W では、引数の設定に従って、バッテリバックアップ機能の有効/無効、VBATT 端子電圧低下検出機能の有効/無効、検出レベル、割り込みを設定します。 デバイス RX671 では、引数の設定に従って、TAMPD 関数の有効/無効、検出レベル、タイムキャプチャイベント、チャネル入力、チャネルノイズフィルタ、割り込みを設定します。
R_VBATT_GetStatus()	デバイス RX230、RX231、RX23W では、バッテリバックアップ機能のステータスを取得するために、VBATT ステータスレジスタ（VBATTSR）を読み出します。次に、その情報を引数として受け渡されたアドレスに保存します。 デバイス RX671 では、改ざん検出器関数の状態を取得するために、TAMPD ステータスレジスタを読み出します。次に、その情報を引数として受け渡されたアドレスに保存します。
R_VBATT_ReadBackupData ()	バックアップレジスタを読み取ります。 この関数はデバイス RX671 でしか使用できません
R_VBATT_WriteBackupData ()	バックアップレジスタを書き込みます。 この関数はデバイス RX671 でしか使用できません
R_VBATT_GetVersion()	本モジュールのバージョン番号を返します。

1.5　使用例

リセットスタート直後に R_VBATT_Open()関数をコールしてください。コールバック関数では、引数を判断して状況に応じた処理を行ってください。

表1.2 にコールバック関数の引数と行うべき処理の一覧を示します。図1.6 は、バッテリバックアップ機能 FIT モジュールの使用例を示しています。

表1.2　コールバック関数の引数と行うべき処理

引数	VBATT_NOT_DROP_VOLTAGE（バッテリバックアップ電源電圧低下の検出なし）	VBATT_DROP_VOLTAGE（バッテリバックアップ電源電圧低下の検出あり）	VBATT_MASKABLE_INTERRUPT VBATT_NON_MASKABLE_INTERRUPT（VBATT 端子電圧の低下検出割り込み） TAMPER_INTERRUPT（改ざん検出割り込み）
行うべき処理	必要に応じて RAM、RTC の割り込みレジスタなどをリセットしてください。	リアルタイムクロックを初期化してください。	デバイス RX230、RX231、RX23W では、外部バッテリが残り少ないことの警告表示やデータのバックアップなどを行ってください。 デバイス RX671 では、システムへの外部侵入が検出されたことの警告表示などを行ってください。
理由	バッテリバックアップ電源電圧が低下しなかったときは、RTC のレジスタの値は保存されますが、RAM や RTC の割り込みレジスタはリセット後の値となります。このため、RAM や RTC の割り込みレジスタは再設定が必要となります。	バッテリバックアップ電源電圧が低下したときは、リアルタイムクロックのレジスタや RAM はリセット後の値となります。このため、初期化が必要となります。	デバイス RX230、RX231、RX23W では、VBATT 端子の電源電圧が低下したときにこの割り込みが発生します。VBATT 端子の電源電圧低下に対応して処理を行うことが可能です。 デバイス RX671 では、改ざん検出器がシステムへの侵入を検出したときにこの割り込みが発生します。改ざん検出器が侵入を検出した状況に対応して処理を行うことが可能です。

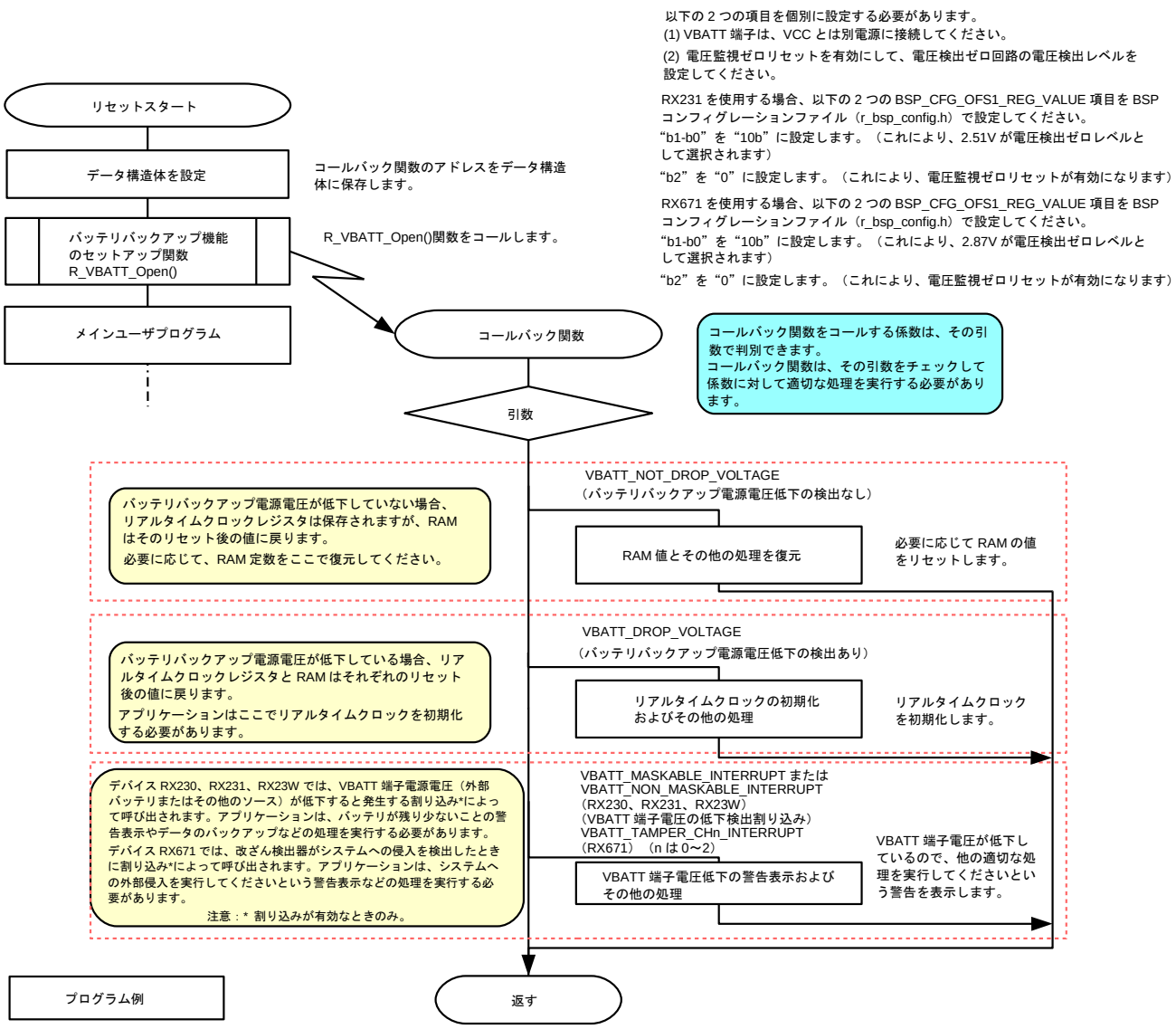


図1.6　バッテリバックアップ機能 FIT モジュールの使用例

1.6　制限事項

- VBATT 端子は、VCC とは別電源に接続してください。
- 電圧監視機能ゼロリセットを有効に設定し、電圧検出ゼロ回路の電圧検出レベルを以下のように設定してください。

RX231、RX230、RX23W グループ：
　電圧検出レベルを 2.51V に設定します。

RX671 グループ：
　電圧検出レベルを 2.94V、2.87V、2.80V のいずれかに設定できます。

- RX671 グループに関する制限事項
  改ざん検出器と RTC のタイムキャプチャ関数を同時に使用する場合、RTC のタイムキャプチャ関数を次のように設定してください。ノイズフィルタ設定（capture.filter）を OFF（RTC_FILTER_OFF）に設定し、エッジ検出（capture.edge）を RTC_EDGE_RISING に設定します。
  VBATT_TAMPER_TCE_TAMPER_EVENT を選択する場合、タイムキャプチャ検出に必要な時間を待機してから検出フラグをクリアする必要があります。詳細は、「4.3.1.2 デバイス RX671 の例」を参照してください。

```
rtc_capture_cfg_t capture;

capture.pin = RTC_PIN_0;
capture.edge = RTC_EDGE_RISING;
capture.filter = RTC_FILTER_OFF;
err = R_RTC_Control(RTC_CMD_CONFIG_CAPTURE, &capture);
```

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 MCU が以下の機能をサポートしている必要があります。

- バッテリバックアップ機能

2.2 ソフトウェアの要求

このドライバは以下の FIT モジュールに依存しています。

- ボードサポートパッケージ（r_bsp）v5.00 以降

2.3 サポートされているツールチェーン

本ドライバは、「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

マクロ定義 VBATT_CFG_DETECT_FUNCTION が VBATT_DTCT_ENABLE_NMI_ENABLE または VBATT_DTCT_ENABLE_INT_ENABLE のときに R_VBATT_Open 関数を実行すると、VBATT 端子電圧低下検出割り込みが有効になります。

TAMP 検出割り込みを有効にするには、R_VBATT_Open 関数を実行します（マクロ定義 VBATT_CFG_TAMPER_CHn_DETECT_INT は、VBATT_TAMPER_DETECT_INT_ENABLE（n は 0～2））。

表 2.1 に本 FIT モジュールが使用する割り込みベクタを示します。

表2.1　使用する割り込みベクター覧

デバイス	割り込みベクタ
RX230 RX231 RX23W	VBATT 端子電圧低下検出割り込み（ベクタ番号：91）*1
RX671	改ざん検出割り込み（ベクタ番号：91）

【注】 *1 マクロ定義 VBATT_CFG_DETECT_FUNCTION が VBATT_DTCT_ENABLE_NMI_ENABLE の場合はノンマスカブル割り込み、VBATT_DTCT_ENABLE_INT_ENABLE の場合はマスカブル割り込みとなります。

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_vbatt_rx_if.h に記述されています。

2.6 整数型

このドライバは 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_vbatt_rx_config.h で行います。
オプション名および設定値に関する説明を、下表に示します。

r_vbatt_rx_config.h のコンフィグレーションオプション	
以下の定義は、RX231、RX230、RX23W グループで使用されます	
#define VBATT_CFG_DETECT_FUNCTION 【注】 デフォルト値は“VBATT_DTCT_DISABLE”	VBATT 端子の電圧低下検出機能を使用するか選択できます。また、電圧の低下を検出したときに発生させる割り込みを選択できます。 “VBATT_DTCT_DISABLE”の場合、 VBATT 端子の電圧低下検出機能は「無効」、割り込みは「禁止」にします。 “VBATT_DTCT_ENABLE_INT_DISABLE”の場合、 VBATT 端子の電圧低下検出機能は「有効」、割り込みは「禁止」にします。 “VBATT_DTCT_ENABLE_NMI_ENABLE”の場合、 VBATT 端子の電圧低下検出機能は「有効」、割り込みは「ノンマスカブル割り込み」を「有効」にします。 “VBATT_DTCT_ENABLE_INT_ENABLE”の場合、 VBATT 端子の電圧低下検出機能は「有効」、割り込みは「マスカブル割り込み」を「有効」にします。
#define VBATT_CFG_DETECT_LEVEL 【注】 デフォルト値は “VBATT_DTCT_LEVEL_2_20_V”	VBATT 端子の電圧低下検出レベルを選択できます。 “VBATT_DTCT_LEVEL_2_20_V”の場合、「2.20V」にします。 “VBATT_DTCT_LEVEL_2_00_V”の場合、「2.00V」にします。
以下の定義は、RX671 グループで使用されます	
#define VBATT_CFG_TAMPER_CH0 【注】 デフォルト値は“VBATT_TAMPER_DISABLE”	チャネル 0 入力に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_DISABLE = TAMPI0/RTCIC0 信号入力は無効 VBATT_TAMPER_ENABLE = TAMPI0/RTCIC0 信号入力は有効
#define VBATT_CFG_TAMPER_CH1 【注】 デフォルト値は“VBATT_TAMPER_DISABLE”	チャネル 1 入力に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_DISABLE = TAMPI1/RTCIC1 信号入力は無効 VBATT_TAMPER_ENABLE = TAMPI1/RTCIC1 信号入力は有効

r_vbatt_rx_config.h のコンフィグレーションオプション	
#define VBATT_CFG_TAMPER_CH2 【注】 デフォルト値は“VBATT_TAMPER_DISABLE”	チャネル 2 入力に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_DISABLE = TAMPI2/RTCIC2 信号入力は無効 VBATT_TAMPER_ENABLE = TAMPI2/RTCIC2 信号入力は有効
#define VBATT_CFG_TAMPER_CH0_DETECT_INT 【注】 デフォルト値は“VBATT_TAMPER_DETECT_INT_DISABLE”	改ざん 0 検出割り込み有効化に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_DETECT_INT_DISABLE = 改ざん 0 検出割り込みは無効 VBATT_TAMPER_DETECT_INT_ENABLE = 改ざん 0 検出割り込みは有効
#define VBATT_CFG_TAMPER_CH0_ERASE 【注】 デフォルト値は“VBATT_TAMPER_ERASE_DISABLE”	改ざん 0 消去有効化に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_ERASE_DISABLE = バックアップレジスタは、改ざん 0 イベントに応答して消去されない VBATT_TAMPER_ERASE_ENABLE = バックアップレジスタは、改ざん 0 イベントに応答して消去される
#define VBATT_CFG_TAMPER_TCE_CH0_SELECT 【注】 デフォルト値は“VBATT_TAMPER_TCE_RTCIC_PIN”	タイムキャプチャイベント 0 ソース選択に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_TCE_RTCIC_PIN = RTCIC0 端子からの入力信号 VBATT_TAMPER_TCE_TAMPER_EVENT = 改ざん 0 イベント
#define VBATT_CFG_TAMPER_CHEN_CH0_NOISE_FILTER 【注】 デフォルト値は“VBATT_TAMPER_CHEN_NOISE_FILTER_DISABLE”	チャネル 0 ノイズフィルタ有効化に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_CHEN_NOISE_FILTER_DISABLE = TAMPI0/RTCIC0 端子のノイズフィルタは無効 VBATT_TAMPER_CHEN_NOISE_FILTER_ENABLE = TAMPI0/RTCIC0 端子のノイズフィルタは有効
#define VBATT_CFG_TAMPER_CHEN_CH0_EDGE 【注】 デフォルト値は“VBATT_TAMPER_CHEN_RISING_EDGE”	チャネル 0 トリガ選択に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_CHEN_FALLING_EDGE = TAMPI0 端子における入力の立ち下がりエッジ VBATT_TAMPER_CHEN_RISING_EDGE = TAMPI0 端子における入力の立ち上がりエッジ
#define VBATT_CFG_TAMPER_CH1_DETECT_INT 【注】 デフォルト値は“VBATT_TAMPER_DETECT_INT_DISABLE”	改ざん 1 検出割り込み有効化に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_DETECT_INT_DISABLE = 改ざん 1 検出割り込みは無効 VBATT_TAMPER_DETECT_INT_ENABLE = 改ざん 1 検出割り込みは有効

r_vbatt_rx_config.h のコンフィグレーションオプション	
#define VBATT_CFG_TAMPER_CH1_ERASE 【注】 デフォルト値は “VBATT_TAMPER_ERASE_DISABLE”	改ざん 1 消去有効化に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_ERASE_DISABLE = バックアップレジスタは、改ざん 1 イベントに応答して消去されない VBATT_TAMPER_ERASE_ENABLE = バックアップレジスタは、改ざん 1 イベントに応答して消去される
#define VBATT_CFG_TAMPER_TCE_CH1_SELECT 【注】 デフォルト値は “VBATT_TAMPER_TCE_RTCIC_PIN”	タイムキャプチャイベント 1 ソース選択に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_TCE_RTCIC_PIN = RTCIC1 端子からの入力信号 VBATT_TAMPER_TCE_TAMPER_EVENT = 改ざん 1 イベント
#define VBATT_CFG_TAMPER_CHEN_CH1_NOISE_FILTER 【注】 デフォルト値は “VBATT_TAMPER_CHEN_NOISE_FILTER_DISABLE”	チャネル 1 ノイズフィルタ有効化に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_CHEN_NOISE_FILTER_DISABLE = TAMPI1/RTCIC1 端子のノイズフィルタは無効 VBATT_TAMPER_CHEN_NOISE_FILTER_ENABLE = TAMPI1/RTCIC1 端子のノイズフィルタは有効
#define VBATT_CFG_TAMPER_CHEN_CH1_EDGE 【注】 デフォルト値は “VBATT_TAMPER_CHEN_RISING_EDGE”	チャネル 1 トリガ選択に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_CHEN_FALLING_EDGE = TAMPI1 端子における入力の立ち下がりエッジ VBATT_TAMPER_CHEN_RISING_EDGE = TAMPI1 端子における入力の立ち上がりエッジ
#define VBATT_CFG_TAMPER_CH2_DETECT_INT 【注】 デフォルト値は “VBATT_TAMPER_DETECT_INT_DISABLE”	改ざん 2 検出割り込み有効化に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_DETECT_INT_DISABLE = 改ざん 2 検出割り込みは無効 VBATT_TAMPER_DETECT_INT_ENABLE = 改ざん 2 検出割り込みは有効
#define VBATT_CFG_TAMPER_CH2_ERASE 【注】 デフォルト値は “VBATT_TAMPER_ERASE_DISABLE”	改ざん 2 消去有効化に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_ERASE_DISABLE = バックアップレジスタは、改ざん 2 イベントに応答して消去されない VBATT_TAMPER_ERASE_ENABLE = バックアップレジスタは、改ざん 2 イベントに応答して消去される
#define VBATT_CFG_TAMPER_TCE_CH2_SELECT 【注】 デフォルト値は “VBATT_TAMPER_TCE_RTCIC_PIN”	タイムキャプチャイベント 2 ソース選択に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_TCE_RTCIC_PIN = RTCIC2 端子からの入力信号 VBATT_TAMPER_TCE_TAMPER_EVENT = 改ざん 2 イベント

r_vbatt_rx_config.h のコンフィグレーションオプション	
#define VBATT_CFG_TAMPER_CHEN_CH2_NOISE_FILTER 【注】 デフォルト値は“VBATT_TAMPER_CHEN_NOISE_FILTER_DISABLE”	チャネル 2 ノイズフィルタ有効化に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_CHEN_NOISE_FILTER_DISABLE = TAMPI2/RTCIC2 端子のノイズフィルタは無効 VBATT_TAMPER_CHEN_NOISE_FILTER_ENABLE = TAMPI2/RTCIC2 端子のノイズフィルタは有効
#define VBATT_CFG_TAMPER_CHEN_CH2_EDGE 【注】 デフォルト値は“VBATT_TAMPER_CHEN_RISING_EDGE”	チャネル 2 トリガ選択に設定 以下のマクロ定義から選択してください。 VBATT_TAMPER_CHEN_FALLING_EDGE = TAMPI2 端子における入力の立ち下がりエッジ VBATT_TAMPER_CHEN_RISING_EDGE = TAMPI2 端子における入力の立ち上がりエッジ
以下の定義は、すべての対象デバイスで使用されます	
#define VBATT_CFG_INT_PRIORITY 【注】 デフォルト値は“5”	RX231、RX230、RX23W では、VBATT 端子電圧低下検出割り込みをマスカブル割り込みで使用する場合の割り込み優先レベルを選択できます。 RX671 では、改ざん割り込みの割り込み優先レベルを選択できます。 “1”～“15”で選択した値を割り込み優先レベルに設定します。 【注】 本設定は VBATT_CFG_DETECT_FUNCTION で“VBATT_DTCT_ENABLE_INT_ENABLE”を選択したときのみ有効です。（RX231、RX230、RX23W）

2.8　コードサイズ

本モジュールの ROM サイズ、RAM サイズ、最大使用スタックサイズを下表に示します。

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

下表の値は下記条件で確認しています。

モジュールリビジョン：r_vbatt_rx rev2.00

コンパイラバージョン：Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00
（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）
GCC for Renesas RX 8.3.0.202004
（統合開発環境のデフォルト設定に“-std=gnu99”オプションを追加）
IAR C/C++ Compiler for Renesas RX version 4.20.1
（統合開発環境のデフォルト設定）

コンフィグレーションオプション：デフォルト設定

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					
		Renesas Compiler		GCC		IAR Compiler	
		パラメータチェックあり	パラメータチェックなし	パラメータチェックあり	パラメータチェックなし	パラメータチェックあり	パラメータチェックなし
RX230	ROM	550 バイト	493 バイト	1192 バイト	1128 バイト	995 バイト	827 バイト
	RAM	4 バイト		4 バイト		4 バイト	
	スタック	44 バイト		-		124 バイト	
RX231	ROM	550 バイト	493 バイト	1192 バイト	1128 バイト	995 バイト	827 バイト
	RAM	4 バイト		4 バイト		4 バイト	
	スタック	44 バイト		-		124 バイト	
RX671	ROM	789 バイト	692 バイト	2064 バイト	1994 バイト	1549 バイト	1185 バイト
	RAM	4 バイト		4 バイト		4 バイト	
	スタック	44 バイト		-		116 バイト	

【注】 1. BSP を含んだサイズです。

2.9 引数

API 関数の引数として使用される構造体を示します。この構造体は、API 関数のプロトタイプ宣言とともに r_vbatt_rx_if.h に記述されています。

```
/* バッテリバックアップ機能のデータ構造体 */
typedef volatile struct
{
   vbatt_callback_t    callbackfunc;     /* コールバック関数 */
} vbatt_info_t;


/* VBATT 端子電圧低下検出機能を再設定するための構造体 */
typedef volatile struct
{
#if defined(BSP_MCU_RX230) || defined(BSP_MCU_RX231) || defined(BSP_MCU_RX23W)
    uint8_t    rsv2;           /* 予約                                 */
    uint8_t    rsv1;           /* 予約                                 */
    uint8_t    vbatt_int_priority;     /* VBATT 端子電圧低下検出マスカブル割り込みの
                                          割り込み優先レベル */
    union
    {
     uint8_t    byte;
     R_BSP_ATTRIB_STRUCT_BIT_ORDER_LEFT_4
     (
       uint8_t rsv:3,              /* 予約  */
           uint8_t lvd_level:2,    /* VBATT 端子電圧低下検出レベル */
           uint8_t lvd_detect:2,   /* VBATT 端子電圧低下検出機能   */
           uint8_t func:1          /* バッテリバックアップ機能  */
      ) bit;
    } vbatt_ctrl;
#endif

#if defined(BSP_MCU_RX671)

    uint8_t tamper_channel;        /* 改ざんチャネルが有効 */

    /* VBATT_TAMPER_INT_ENABLE
    VBATT_TAMPER_INT_DISABLE */
    uint8_t tamper_detection_interrupt;    /* TAMPCR
                              0：改ざん n 検出割り込みは無効
                              1：改ざん n 検出割り込みは有効*/
    /* VBATT_TAMPER_ERASE_ENABLE
       VBATT_TAMPER_ERASE_DISABLE */
    uint8_t tamper_erase;          /* TAMPCR
                         0：バックアップレジスタは、改ざん n イベントに応答して消去されない
                         1：バックアップレジスタは、改ざん n イベントに応答して消去される*/
    /* VBATT_TAMPER_TCE_RTCIC_PIN
       VBATT_TAMPER_TCE_TEMPER_EVENT */
    uint8_t time_capture_source;  /* TCECR
                         0：RTCICn 端子からの入力信号
                         1：改ざん n イベント */
    /* VBATT_TAMPER_CHEN_INPUT_DISABLE
       VBATT_TAMPER_CHEN_INPUT_ENABLE */
    uint8_t channel_input;         /* TAMPICR1
                         0：RTCICn 信号入力は無効
```

```
                        1：RTCICn 信号入力は有効*/
    /* VBATT_TAMPER_CHEN_NOISE_FILTER_DISABLE
       VBATT_TAMPER_CHEN_NOISE_FILTER_ENABLE */
    uint8_t channel_noise_filter;   /* TAMPICR2
                        0：RTCICn 端子のノイズフィルタは無効
                        1：RTCICn 端子のノイズフィルタは有効*/
    /* VBATT_TAMPER_CHEN_FALLING_EDGE
       VBATT_TAMPER_CHEN_RISING_EDGE */
    uint8_t channel_trigger_select;  /* TAMPICR2
                        0：RTCICn 端子における入力の立ち下がりエッジ
                        1：RTCICn 端子における入力の立ち上がりエッジ*/

    uint8_t tamper_int_priority;    /* 割り込み優先度。1=低、15=高 */
#endif /* defined(BSP_MCU_RX671) */
} vbatt_ctrl_info_t;

/* バッテリバックアップ機能のステータス情報構造体 */
typedef volatile struct
{
#if defined(BSP_MCU_RX230) || defined(BSP_MCU_RX231) || defined(BSP_MCU_RX23W)
    union
    {
      uint8_t    byte;
      R_BSP_ATTRIB_STRUCT_BIT_ORDER_LEFT_3
      (
        uint8_t rsv:6,          /* 予約 */
        uint8_t vbatt_mon:1,    /* VBATT 端子電圧モニタフラグ */
        uint8_t rsv1:1          /* 予約 */
       ) bit;
    } vbatt_status;
#endif
#if defined(BSP_MCU_RX671)
    uint8_t tamper_channel;           /* 改ざんチャネルがステータスを取得する必要あり*/
    uint8_t tamper_detection_flag;    /* 改ざん検出フラグ */
    uint8_t tamper_level_monitoring_flag;  /* チャネルレベル監視フラグ */
    bool  action_clear;               /* アクションクリアステータスレジスタ */
#endif
} vbatt_status_t;
```

2.10 戻り値

API 関数の戻り値を示します。この列挙型は、API 関数のプロトタイプ宣言とともに r_vbatt_rx_if.h で定義されています。

```
typedef enum                    /* バッテリバックアップ API 関数に対するコールに使用される
                                   戻り値 */
{
   VBATT_SUCCESS,               /* 問題なく処理が完了した */
   VBATT_ERR_LOCK_FUNC,         /* 関数が複数回呼び出された */
                                   （未実装）
   VBATT_ERR_LVD0_SETTING,      /* オプション機能選択レジスタ 1（OFS1）の
                                   電圧監視 0 設定が不正 */
   VBATT_ERR_INVALID_ARG,       /* 引数が不正である場合 */
   VBATT_ERR_FUNC_INVALID,      /* VBATT 端子電圧低下検出が無効なときに
                                   R_VBATT_GetStatus()がコールされた */
   VBATT_ERR_OTHER              /* その他のエラー */
} vbatt_return_t;
```

2.11 コールバック関数

本モジュールでは、R_VBATT_Open()関数を呼び出したタイミング、または、割り込みが発生したタイミングで、コールバック関数を呼び出します。コールバック関数は、引数を持ち、バッテリバックアップ電源電圧の低下を検出したかや VBATT 端子電圧低下時の割り込み処理からの呼び出しかによって、引数に設定される値が決まります。

表2.2 に、コールバック関数に受け渡される引数の定数定義（enum vbatt_cb_evt_t）を示します。

コールバック関数の設定では、コールバック関数として登録する関数のアドレスを、「2.9 引数」に記載されている構造体メンバ“callbackfunc”に保存してください。

表2.2　コールバック関数の引数に渡される定数定義一覧（enum vbatt_cb_evt_t）

定数定義	引数に渡される条件
VBATT_NOT_DROP_VOLTAGE	バッテリバックアップ電源電圧の低下を検出していない状態（VBATTSR レジスタの VBATRLVDETF ビットが“0”）で、R_VBATT_Open()関数を呼び出したとき
VBATT_DROP_VOLTAGE	バッテリバックアップ電源電圧の低下を検出した状態（VBATTSR レジスタの VBATRLVDETF ビットが“1”）で、R_VBATT_Open()関数を呼び出したとき
VBATT_MASKABLE_INTERRUPT	VBATT 端子電圧の低下によって、マスカブル割り込みが発生したとき
VBATT_NON_MASKABLE_INTERRUPT	VBATT 端子電圧の低下によって、ノンマスカブル割り込みが発生したとき
VBATT_TAMPER_CH0_INTERRUPT VBATT_TAMPER_CH1_INTERRUPT VBATT_TAMPER_CH2_INTERRUPT	改ざん検出器がシステムへの侵入を検出したことによるマスカブル割り込みが発生したとき

2.12 FIT モジュールの追加方法

本モジュールは、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)のいずれかの追加方法を推奨しています。ただし、Smart Configurator は、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)、(4)の方法を使用してください。

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

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

(3) CS+上で Smart Configurator を使用して FIT モジュールを追加する場合
CS+上で、スタンドアロン版 Smart Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、「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 Smart Configurator 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 Smart Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、「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。

2.13 for 文、while 文、do while 文について

本モジュールでは、レジスタの反映待ち処理等で 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。

以下に記述例を示します。

```
while 文の例：
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* PLL が安定したことを確認するために遅延時間が必要。 */
}

for 文の例：
/* 基準カウンタを 0 に初期化。 */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}

do while 文の例：
/* リセット完了待機中 */
do
{
    reg = phy_read(ether_channel, PHY_REG_CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /*
WAIT_LOOP */
```

3. API 関数

R_VBATT_Open()

デバイス RX230、RX231、RX23W では、VBATT 端子電圧低下検出機能の設定およびバッテリバックアップ電源電圧低下を判定する関数です。

この関数は他の API 関数を使用する前に実行される必要があります。バッテリバックアップ電源電圧の低下を検出したときおよび検出しなかったときの処理は、この関数をコールしたときに呼び出されるコールバック関数で行います。

デバイス RX671 では、TAMPD 検出関数を設定し、バックアップ領域の電圧が V_{PDR} を下回っているかどうかを判定します。

この関数は他の API 関数を使用する前に実行される必要があります。バックアップ領域の電圧が V_{PDR} を下回っていることが検出される場合および電圧低下が検出されない場合の処理は、この関数をコールしたときに呼び出されるコールバック関数によって実行されます。

Format

```
vbatt_return_t R_VBATT_Open (
  vbatt_info_t *      p_vbatt_info
)
```

Parameters

p_vbatt_info

バッテリバックアップ情報のデータ構造体へのポインタ

この関数で使用するメンバを以下に示します。この構造体の詳細については、「2.9 引数」を参照してください。

```
vbatt_callback_t        callbackfunc;        /* コールバック関数のアドレス */
```

Return Values

```
VBATT_SUCCESS                   /* 問題なく処理が完了した */
VBATT_ERR_LVD0_SETTING          /* オプション機能選択レジスタ 1（OFS1）の
                                   電圧監視 0 設定が不正 */
VBATT_ERR_INVALID_ARG           /* 引数が不正である場合 */
```

Properties

この関数のプロトタイプ宣言は、r_vbatt_rx_if.h に記述されています。

Description

デバイス RX230、RX231、RX23W では、コンフィグレーションオプションの設定に従って、VBATT 端子電圧低下検出機能の有効/無効、検出レベル、割り込みを設定します。その後、バッテリバックアップ電源電圧低下の有無を判定し、コールバック関数を呼び出します。

デバイス RX671 では、コンフィグレーションオプションの設定に従って、TAMPD 関数の有効/無効、検出レベル、タイムキャプチャイベント、チャネル入力、チャネルノイズフィルタ、割り込みを設定します。その後、バックアップ領域の電圧が VPDR を下回っているかどうかを判定し、コールバック関数を呼び出します。

コールバック関数を呼び出すとき：

- バッテリバックアップ電源電圧の低下が検出されなかった場合：
  VBATT_NOT_DROP_VOLTAGE に設定された変数へのポインタが引数として受け渡されます。
- バッテリバックアップ電源電圧の低下が検出された場合：
  VBATT_DROP_VOLTAGE に設定された変数へのポインタが引数として受け渡されます。

Example

```
#include "r_vbatt_rx_if.h"

void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event);

void main(void)
{
  vbatt_return_t      ret;
  vbatt_info_t     vbatt_info;

  vbatt_info.callbackfunc = vbatt_callback;

  ret = R_VBATT_Open(&vbatt_info);
  if (VBATT_SUCCESS != ret)
  {
    /* エラー発生時に処理を実行してください */
  }

  while(1);

}

void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event)
{
  switch(*vbatt_cb_event)
  {
    /* バッテリバックアップ電源電圧の低下は未検出 */
    case VBATT_NOT_DROP_VOLTAGE:

       /* 必要に応じて RAM を再設定してください */

    break;

    /* バッテリバックアップ電源電圧の低下を検出 */
    case VBATT_DROP_VOLTAGE:

       /* リアルタイムクロックを初期化してください */

    break;

    /* VBATT 電圧の低下が検出され割り込み発生 */
    case VBATT_MASKABLE_INTERRUPT:
    case VBATT_NON_MASKABLE_INTERRUPT:

       /* 警告表示、バックアップなどを処理してください */

    break;

    /* 改ざん検出割り込み */
    case VBATT_TAMPER_CH0_INTERRUPT:
```

```
        /* 警告表示などを処理してください */

    break;

    /* 改ざん検出割り込み */
    case VBATT_TAMPER_CH1_INTERRUPT:

        /* 警告表示などを処理してください */

    break;


    /* 改ざん検出割り込み */
    case VBATT_TAMPER_CH2_INTERRUPT:

        /* 警告表示などを処理してください */

    break;

    default:

        /* 何もしないこと */

    break;
  }
}
```

Special Notes

バッテリバックアップモード時に改ざんが検出された場合、この関数を実行すると改ざん検出ステータスがクリアされます。

したがって、バッテリバックアップモード時に改ざん検出を確認する場合、R_VBATT_GetStatus 関数を使用して確認できます。

R_VBATT_Control()

デバイス RX230、RX231、RX23W では、バッテリバックアップ機能の有効/無効の設定、および VBATT 端子電圧低下検出機能を設定する関数です。R_VBATT_Open()関数で設定した内容から変更するときに使用します。

デバイス RX671 では、TAMPD 関数の有効/無効を設定し、TAMPD 関数を設定します。R_VBATT_Open()関数で設定した内容から変更するときに使用します。

Format

```
vbatt_return_t R_VBATT_Control (
  vbatt_ctrl_info_t *    p_vbatt_ctrl_info
)
```

Parameters

p_vbatt_ctrl_info

VBATT 端子電圧低下検出機能によって使用されるデータ構造体へのポインタ
この関数で使用するメンバを以下に示します。この構造体の詳細については、「2.9 引数」を参照してください。

```
typedef volatile struct
{
#if defined(BSP_MCU_RX230) || defined(BSP_MCU_RX231) || defined(BSP_MCU_RX23W)
uint8_t rsv2; /* 予約領域 */
uint8_t rsv1; /* 予約領域 */
uint8_t vbatt_int_priority; /* VBATT 端子電圧低下検出
割り込み（マスカブル割り込み）の割り込み優先レベル */
union
{
uint8_t byte;
struct
{
uint8_t rsv:3;          /* 予約領域 */
uint8_t lvd_level:2;    /* VBATT 端子電圧低下検出レベル */
uint8_t lvd_detect:2;   /* VBATT 端子電圧低下検出機能 */
uint8_t func:1;         /* バッテリバックアップ機能の有効/無効 */
} bit;
} vbatt_ctrl;
#endif
#if defined(BSP_MCU_RX671)

    uint8_t tamper_channel;                /* 改ざんチャネルが有効 */

    /* VBATT_TAMPER_INT_ENABLE
    VBATT_TAMPER_INT_DISABLE */
    uint8_t tamper_detection_interrupt;     /* TAMPCR
                                    0：改ざん n 検出割り込みは無効
                                    1：改ざん n 検出割り込みは有効*/
    /* VBATT_TAMPER_ERASE_ENABLE
       VBATT_TAMPER_ERASE_DISABLE */
    uint8_t tamper_erase;                  /* TAMPCR
                        0：バックアップレジスタは、改ざん n イベントに応答して消去されない
                        1：バックアップレジスタは、改ざん n イベントに応答して消去される*/
```

```
    /* VBATT_TAMPER_TCE_RTCIC_PIN
       VBATT_TAMPER_TCE_TEMPER_EVENT */
    uint8_t time_capture_source;          /* TCECR
                                       0：RTCICn 端子からの入力信号
                                       1：改ざん n イベント */
    /* VBATT_TAMPER_CHEN_INPUT_DISABLE
       VBATT_TAMPER_CHEN_INPUT_ENABLE */
    uint8_t channel_input;              /* TAMPICR1
                                       0：RTCICn 信号入力は無効
                                       1：RTCICn 信号入力は有効*/
    /* VBATT_TAMPER_CHEN_NOISE_FILTER_DISABLE
       VBATT_TAMPER_CHEN_NOISE_FILTER_ENABLE */
    uint8_t channel_noise_filter;         /* TAMPICR2
                                       0：RTCICn 端子のノイズフィルタは無効
                                       1：RTCICn 端子のノイズフィルタは有効*/
    /* VBATT_TAMPER_CHEN_FALLING_EDGE
       VBATT_TAMPER_CHEN_RISING_EDGE */
    uint8_t channel_trigger_select;       /* TAMPICR2
                                       0：RTCICn 端子における入力の立ち下がりエッジ
                                       1：RTCICn 端子における入力の立ち上がりエッジ*/
    uint8_t tamper_int_priority;          /* 割り込み優先度。1=低、15=高 */
#endif /* defined(BSP_MCU_RX671) */
} vbatt_ctrl_info_t;
```

Return Values

VBATT_SUCCESS	/* 問題なく処理が完了した */
VBATT_ERR_INVALID_ARG	/* 引数が不正である場合 */

Properties

この関数のプロトタイプ宣言は、r_vbatt_rx_if.h に記述されています。

Description

デバイス RX230、RX231、RX23W では、引数の設定に従って、バッテリバックアップ機能の有効/無効、VBATT 端子電圧低下検出機能の有効/無効、検出レベル、割り込みを設定します。

デバイス RX671 では、引数の設定に従って、TAMPER 検出関数および割り込みを設定します。

Example

デバイス RX230、RX231、RX23W の例

```
#include "r_vbatt_rx_if.h"

void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event);

void main(void)
{
  vbatt_return_t      ret;
  vbatt_info_t     vbatt_info;
  vbatt_ctrl_info_t      vbatt_ctrl_info;

  vbatt_info.callbackfunc = vbatt_callback;

  ret = R_VBATT_Open(&vbatt_info);
  if (VBATT_SUCCESS != ret)
```

```
  {
    /* エラー発生時に処理を実行してください */
  }

  /* バッテリバックアップ機能の有効化 */
  vbatt_ctrl_info.vbatt_ctrl.bit.func = 1;
  /* VBATT 低下検出機能の有効化とマスカブル割り込みの有効化 */
  vbatt_ctrl_info.vbatt_ctrl.bit.lvd_detect = VBATT_DTCT_ENABLE_INT_ENABLE;
  /* VBATT 低下検出レベルは 2.00V */
  vbatt_ctrl_info.vbatt_ctrl.bit.lvd_level = VBATT_DTCT_LEVEL_2_00_V;
  /* 割り込み優先レベルは 7*/
  vbatt_ctrl_info.vbatt_int_priority = 7;

  ret = R_VBATT_Control(&vbatt_ctrl_info);
  if (VBATT_SUCCESS != ret)
  {
       /* エラー発生時に処理を実行してください */
  }

  while(1);
}
void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event)
{
  /* vbatt_cb_evt_t に基づいて処理してください */
}
```

デバイス RX671 の例

```
#include "r_vbatt_rx_if.h"

void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event);

void main(void)
{
  vbatt_return_t   ret;
  vbatt_info_t     vbatt_info;
  vbatt_ctrl_info_t      vbatt_ctrl_info;

    vbatt_info.callbackfunc = vbatt_callback;

  ret = R_VBATT_Open(&vbatt_info);
  if (VBATT_SUCCESS != ret)
  {
    /* エラー発生時に処理を実行してください */
  }

  vbatt_ctrl_info_t   vbatt_ctrl_info;
  /* 改ざん n チャネルは有効 */
  vbatt_ctrl_info.tamper_channel = VBATT_TAMPER_CH0;
  /* 改ざん検出割り込みは有効*/
  vbatt_ctrl_info.tamper_detection_interrupt = VBATT_TAMPER_DETECT_INT_ENABLE;
  /* バックアップレジスタは、改ざん n イベントに応答して消去される */
  vbatt_ctrl_info.tamper_erase = VBATT_TAMPER_ERASE_ENABLE;
  /* RTCICn 端子からの入力信号 */
  vbatt_ctrl_info.time_capture_source = VBATT_TAMPER_TCE_TAMPER_EVENT;
  /* RTCICn 信号入力は有効 */
  vbatt_ctrl_info.channel_input = VBATT_TAMPER_CHEN_INPUT_ENABLE;
  /* RTCICn 端子のノイズフィルタは有効 */
  vbatt_ctrl_info.channel_noise_filter = VBATT_TAMPER_CHEN_NOISE_FILTER_ENABLE;
  /* RTCICn 端子における入力の立ち下がりエッジ*/
  vbatt_ctrl_info.channel_trigger_select = VBATT_TAMPER_CHEN_FALLING_EDGE;
  /* 割り込み優先度。1=低、15=高 */
  vbatt_ctrl_info.tamper_int_priority = 5;
  ret = R_VBATT_Control(&vbatt_ctrl_info);
  if (VBATT_SUCCESS != ret)
  {
    /* エラー発生時に処理を実行してください */
  }
  while(1);
}
void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event)
{
  /* vbatt_cb_evt_t に基づいて処理してください */
}
```

Special Notes

1. 引数の設定可能範囲および設定内容は、下表を参照してください。

デバイス RX230、RX231、RX23W の構造体（vbatt_ctrl_info_t）			
メンバ	ビット	設定可能範囲	設定内容
vbatt_ctrl	func	0～1	バッテリバックアップ機能の有効/無効を変更できます。 0：バッテリバックアップ機能は無効 1：バッテリバックアップ機能は有効
	lvd_detect	設定内容のマクロ定義で選択	VBATT 端子の電圧低下検出機能を使用するか選択できます。また、電圧の低下を検出したときに発生させる割り込みを選択できます。 • “VBATT_DTCT_DISABLE”の場合、VBATT 端子の電圧低下検出機能は「無効」、割り込みは「禁止」にします。 • “VBATT_DTCT_ENABLE_INT_DISABLE”の場合、VBATT 端子の電圧低下検出機能は「有効」、割り込みは「禁止」にします。 • “VBATT_DTCT_ENABLE_NMI_ENABLE”の場合、VBATT 端子の電圧低下検出機能は「有効」、割り込みは「ノンマスカブル割り込み」を「有効」にします。 • “VBATT_DTCT_ENABLE_INT_ENABLE”の場合、VBATT 端子の電圧低下検出機能は「有効」、割り込みは「マスカブル割り込み」を「有効」にします。
	lvd_level	設定内容のマクロ定義で選択	VBATT 端子の電圧低下検出レベルを選択できます。 • “VBATT_DTCT_LEVEL_2_20_V”の場合、「2.20V」にします。 • “VBATT_DTCT_LEVEL_2_00_V”の場合、「2.00V」にします。
vbatt_int_priority	–	1～15	VBATT 端子電圧低下検出割り込みをマスカブル割り込みで使用する場合の割り込み優先レベルを選択できます。“1”～“15”で選択した値を割り込み優先レベルに設定します。 【注】 本設定が有効なのは、lvd_detect によって VBATT_DTCT_ENABLE_INT_ENABLE が選択されたときのみです。

デバイス RX671 の構造体（vbatt_ctrl_info_t）			
メンバ	ビット	設定可能範囲	設定内容
tamper_channel	–	設定内容のマクロ定義で選択	制御用チャネルが選択されます。 • VBATT_TAMPER_CH0 の場合、チャネル 0 が選択されます。 • VBATT_TAMPER_CH1 の場合、チャネル 1 が選択されます。 • VBATT_TAMPER_CH2 の場合、チャネル 2 が選択されます。
tamper_detection_interrupt	–	設定内容のマクロ定義で選択	tamper_channel 検出割り込みの有効/無効を選択します。 • VBATT_TAMPER_DETECT_INT_ENABLE の場合、tamper_channel 検出割り込みは有効です。 • VBATT_TAMPER_DETECT_INT_DISABLE の場合、tamper_channel 検出割り込みは無効です。
tamper_erase	–	設定内容のマクロ定義で選択	tamper_channel イベントに応答してバックアップレジスタを消去するかどうかを選択します。 • VBATT_TAMPER_ERASE_ENABLE の場合、バックアップレジスタは、tamper_channel イベントに応答して消去されます。 • VBATT_TAMPER_ERASE_DISABLE の場合、バックアップレジスタは、tamper_channel イベントに応答して消去されません。
time_capture_source	–	設定内容のマクロ定義で選択	RTC のタイムキャプチャイベントのソースを選択します。 • VBATT_TAMPER_TCE_RTCIC_PIN の場合、RTCICn 端子からの入力信号です。 • VBATT_TAMPER_TCE_TAMPER_EVENT の場合、tamper_channel イベントです。
channel_input	–	設定内容のマクロ定義で選択	RTCICn 端子からの入力の有効/無効を選択します。 • VBATT_TAMPER_CHEN_INPUT_DISABLE の場合、tamper_channel 信号入力は無効です。 • VBATT_TAMPER_CHEN_INPUT_ENABLE の場合、tamper_channel 信号入力は有効です。
channel_noise_filter	–	設定内容のマクロ定義で選択	RTCICn 端子のノイズフィルタの有効/無効を選択します。 • VBATT_TAMPER_CHEN_NOISE_FILTER_DISABLE の場合、tamper_channel 端子のノイズフィルタは無効です。 • VBATT_TAMPER_CHEN_NOISE_FILTER_ENABLE の場合、tamper_channel 端子のノイズフィルタは有効です。
channel_trigger_select	–	設定内容のマクロ定義で選択	改ざんイベント検出のトリガとして使用する RTCICn 端子からの入力信号に効果的なエッジを選択します。 • VBATT_TAMPER_CHEN_FALLING_EDGE の場合、tamper_channel 端子における入力の立ち下がりエッジです。 • VBATT_TAMPER_CHEN_RISING_EDGE の場合、tamper_channel 端子における入力の立ち上がりエッジです。

デバイス RX671 の構造体（vbatt_ctrl_info_t）			
メンバ	ビット	設定可能範囲	設定内容
tamper_int_priority	–	1～15	割り込み優先レベルは、tamper_channel 検出割り込みが有効な場合に選択できます。 “1”～“15”で選択した値を割り込み優先レベルに設定します。 【注】 本設定が有効なのは、tamper_detection_interrupt によって VBATT_TAMPER_DETECT_INT_ENABLE が選択されたときのみです。

2. デバイス RX671 では、この関数を実行すると、改ざん検出ステータスがクリアされます。

R_VBATT_GetStatus()

バッテリバックアップ機能のステータスを取得する関数です。バッテリバックアップ機能のステータスを確認したいときに使用します。

Format

```
vbatt_return_t R_VBATT_GetStatus (
  vbatt_status_t *    p_vbatt_status
)
```

Parameters

p_vbatt_status

バッテリバックアップ機能ステータスを格納する変数へのポインタ

この関数で使用するメンバを以下に示します。この構造体の詳細については、「2.9 引数」を参照してください。

```
typedef volatile struct
{
#if defined(BSP_MCU_RX230) || defined(BSP_MCU_RX231) || defined(BSP_MCU_RX23W)
    union
    {
       uint8_t    byte;
       R_BSP_ATTRIB_STRUCT_BIT_ORDER_LEFT_3
       (
          uint8_t rsv:6,        /* 予約                                */
          uint8_t vbatt_mon:1,  /* VBATT 端子電圧モニタフラグ          */
          uint8_t rsv1:1        /* 予約     */
       ) bit;
    } vbatt_status;
#endif
#if defined(BSP_MCU_RX671)
    uint8_t tamper_channel;           /* 改ざんチャネルがステータスを取得する必要あり*/
    uint8_t tamper_detection_flag;    /* 改ざん検出フラグ */
    uint8_t tamper_level_monitoring_flag;  /* チャネルレベル監視フラグ */
    bool  action_clear;  /* アクションクリアステータスレジスタ */
#endif
} vbatt_status_t;
```

Return Values

VBATT_SUCCESS	/* 問題なく処理が完了した */
VBATT_ERR_INVALID_ARG	/* 引数が不正である場合 */
VBATT_ERR_FUNC_INVALID	/* VBATT 端子電圧低下検出が無効なときに R_VBATT_GetStatus()がコールされた */

Properties

この関数のプロトタイプ宣言は、r_vbatt_rx_if.h に記述されています。

Description

デバイス RX230、RX231、RX23W では、バッテリバックアップ機能のステータスを取得するために、VBATT ステータスレジスタ（VBATTSR）を読み出します。次に、その情報を引数として受け渡されたアドレスに保存します。

デバイス RX671 では、改ざん検出器関数の状態を取得するために、TAMPD ステータスレジスタを読み出します。次に、その情報を引数として受け渡されたアドレスに保存します。

Example

デバイス RX230、RX231、RX23W の例

```
#include "r_vbatt_rx_if.h"

void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event);

void main(void)
{
  vbatt_return_t         ret;
  vbatt_info_t          vbatt_info;
  vbatt_status_t         vbatt_status;
  vbatt_ctrl_info_t      vbatt_ctrl_info;

  vbatt_info.callbackfunc = vbatt_callback;

  ret = R_VBATT_Open(&vbatt_info);
  if (VBATT_SUCCESS != ret)
  {
    /* エラー発生時に処理を実行してください */
  }

  /* VBATT 低下検出機能の有効化 */
  vbatt_ctrl_info.vbatt_ctrl.bit.func = 1;
  vbatt_ctrl_info.vbatt_ctrl.bit.lvd_detect = VBATT_DTCT_ENABLE_INT_DISABLE;
  vbatt_ctrl_info.vbatt_ctrl.bit.lvd_level = VBATT_DTCT_LEVEL_2_20_V;
  vbatt_ctrl_info.vbatt_int_priority = 5;
  ret = R_VBATT_Control(&vbatt_ctrl_info);
  if (VBATT_SUCCESS != ret)
  {
      /* エラー発生時に処理を実行してください */
  }

  /* バッテリバックアップ機能のステータスを取得 */
  ret = R_VBATT_GetStatus(&vbatt_status);
  if (VBATT_SUCCESS != ret)
  {
      /* エラー発生時に処理を実行してください */
  }

  while(1);
}
void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event)
{
  /* vbatt_cb_evt_t に基づいて処理してください */
}
```

デバイス RX671 の例

```
#include "r_vbatt_rx_if.h"

void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event);

void main(void)
{
  vbatt_return_t   ret;
  vbatt_info_t     vbatt_info;
  vbatt_status_t   vbatt_status;
  vbatt_ctrl_info_t vbatt_ctrl_info;

  vbatt_info.callbackfunc = vbatt_callback;

  ret = R_VBATT_Open(&vbatt_info);
  if (VBATT_SUCCESS != ret)
  {
    /* エラー発生時に処理を実行してください */
  }

  /* 改ざん n チャネルは有効*/
  vbatt_ctrl_info.tamper_channel = VBATT_TAMPER_CH0;
  /* 改ざん検出割り込みは有効 */
  vbatt_ctrl_info.tamper_detection_interrupt = VBATT_TAMPER_DETECT_INT_ENABLE;
  /* バックアップレジスタは、改ざん n イベントに応答して消去される */
  vbatt_ctrl_info.tamper_erase = VBATT_TAMPER_ERASE_ENABLE;
  /* RTCICn 端子からの入力信号 */
  vbatt_ctrl_info.time_capture_source = VBATT_TAMPER_TCE_RTCIC_PIN;
  /* RTCICn 信号入力は有効 */
  vbatt_ctrl_info.channel_input = VBATT_TAMPER_CHEN_INPUT_ENABLE;
  /* RTCICn 端子のノイズフィルタは有効 */
  vbatt_ctrl_info.channel_noise_filter = VBATT_TAMPER_CHEN_NOISE_FILTER_ENABLE;
  /* RTCICn 端子における入力の立ち下がりエッジ */
  vbatt_ctrl_info.channel_trigger_select = VBATT_TAMPER_CHEN_FALLING_EDGE;
  /* 割り込み優先度。1=低、15=高*/
  vbatt_ctrl_info.tamper_int_priority = 5;
  ret = R_VBATT_Control(&vbatt_ctrl_info);
  if (VBATT_SUCCESS != ret)
  {
    /* エラー発生時に処理を実行してください */
  }
  /* バッテリバックアップ機能のステータスを取得 */
  vbatt_status.tamper_channel = VBATT_TAMPER_CH0;
  vbatt_status.action_clear = true;
  ret = R_VBATT_GetStatus(&vbatt_status);
  if (VBATT_SUCCESS != ret)
  {
    /* エラー発生時に処理を実行してください */
  }
   while(1);
}
void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event)
{
  /* vbatt_cb_evt_t に基づいて処理してください */
}
```

Special Notes

1. 以下にステータスフラグの配置を示します。

デバイス RX230、RX231、RX23W の構造体（vbatt_ctrl_info_t）			
ビット	b7～b2	b1	b0
ビット名	予約領域	VBATT 端子電圧モニタフラグ	予約領域
シンボル	rsv	vbatt_mon	rsv1
関数	不定	0：VBATT < Vdetvbt 1：VBATT ≥ Vdetvbt または VBATT 検出機能が無効	不定
デバイス RX671 の構造体（vbatt_ctrl_info_t）			
メンバ	設定可能範囲		設定内容
tamper_channel	VBATT_TAMPER_CH0 VBATT_TAMPER_CH1 VBATT_TAMPER_CH2		改ざんチャネルがステータスを取得する必要あり
tamper_detection_flag	0～1		0：改ざんイベントが検出されていない 1：改ざんイベントが検出された
tamper_level_monitoring_flag	0～1		0：RTCIC 端子の入力信号レベルが低い 1：RTCIC 端子の入力信号レベルが高い
action_clear	0～1		0：クリアステータスレジスタは無効 1：クリアステータスレジスタは有効

2. デバイス RX671 では、改ざん検出を有効にしてバッテリバックアップモードへ移行するとき、R_VBATT_Open 関数を実行する前にこの関数を使用すると、バッテリバックアップモード時に改ざん検出が検出されたかどうかを確認できます。なお、R_VBATT_Open 関数を実行すると、改ざん検出ステータスはクリアされるので注意してください。

R_VBATT_ReadBackupData()

この関数はデバイス RX671 でしか使用できません。

この関数はバックアップレジスタからデータを読み取ります。

Format

```
vbatt_return_t R_VBATT_ReadBackupData(
      uint8_t index,
      uint8_t * p_data
)
```

Parameters

uint8_t index,

読み取る必要があるバックアップレジスタのインデックス

範囲：0～127

*uint8_t * p_data,*

バックアップレジスタから読み取ったデータを保存する変数へのポインタ

Return Values

VBATT_SUCCESS	/* 問題なく処理が完了した */
VBATT_ERR_INVALID_ARG	/* 引数が不正である場合 */

Properties

この関数のプロトタイプ宣言は、r_vbatt_rx_if.h に記述されています。

Description

この関数はバックアップレジスタからデータを読み取ります。

Example

```
#include "r_vbatt_rx_if.h"
void main(void)
{
   vbatt_return_t   ret;
   uint8_t index = 0;
   /* バックアップレジスタから読み取った変数の保存内容
   uint8_t backup_register;
   ret = R_VBATT_ReadBackupData(index, &backup_register);
   while(ret != VBATT_SUCCESS);

}
```

Special Notes

なし

R_VBATT_WriteBackupData()

この関数はデバイス RX671 でしか使用できません。

この関数はバックアップレジスタにデータを書き込みます。

Format

```
vbatt_return_t R_VBATT_WriteBackupData(
      uint8_t index,
      uint8_t * p_data
)
```

Parameters

uint8_t index,

書き込む必要があるバックアップレジスタのインデックス

範囲：0～127

*uint8_t * p_data,*

バックアップレジスタに書き込むデータを保存する変数へのポインタ

Return Values

VBATT_SUCCESS	/* 問題なく処理が完了した */
VBATT_ERR_INVALID_ARG	/* 引数が不正である場合 */

Properties

この関数のプロトタイプ宣言は、r_vbatt_rx_if.h に記述されています。

Description

この関数はバックアップレジスタにデータを書き込みます。

Example

```
#include "r_vbatt_rx_if.h"
void main(void)
{
   vbatt_return_t   ret;
   uint8_t index = 0;
   /* バックアップレジスタに書き込む変数の保存内容
   uint8_t backup_register = 0x5;
   ret = R_VBATT_WriteBackupData(index, &backup_register);
   while(ret != VBATT_SUCCESS);

}
```

Special Notes

なし

R_VBATT_GetVersion()

API のバージョンを返す関数です。

Format

```
uint32_t R_VBATT_GetVersion(void)
```

Parameters

なし

Return Values

バージョン番号

Properties

この関数のプロトタイプ宣言は、r_vbatt_rx_if.h に記述されています。

Description

この関数は本 API のバージョン番号を返します。

Example

```
uint32_t    version;

version = R_VBATT_GetVersion();
```

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以下に示します。

表 4.1　動作確認環境（Rev. 2.20）

項目	内容
統合開発環境	Renesas Electronics e^2 studio Version 2022-07 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	Renesas Electronics C/C++ Compiler Package 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 for Renesas RX 8.3.0.202202 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 -std=gnu99 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 コンパイラオプション：統合開発環境のデフォルト設定。
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev. 2.20
使用ボード	Renesas Starter Kit+ for RX671 (product No.: RTK55671xxxxxxxxx)

表 4.2　動作確認環境（Rev. 2.10）

項目	内容
統合開発環境	Renesas Electronics e^2 studio Version 2021-07 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00 コンパイラオプション：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 for Renesas RX 8.3.0.202004 コンパイラオプション：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。 -std=gnu99 リンカオプション：“Optimize size (-Os)”を使用する場合は、以下のユーザ定義オプションを統合開発環境のデフォルト設定に追加してください。 -WI,--no-gc-sections リンカが誤って FIT 周辺機能モジュールで宣言された割り込み関数を破棄することによる GCC リンカ問題を解決します。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 コンパイラオプション：統合開発環境のデフォルト設定。
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev. 2.10
使用ボード	Renesas Starter Kit+ for RX671 (product No.: RTK55671xxxxxxxxx)

表 4.3　動作確認環境（Rev. 2.00）

項目	内容
統合開発環境	Renesas Electronics e^2 studio Version 2021-07 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00 コンパイラオプション：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 for Renesas RX 8.3.0.202004 コンパイラオプション：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。 -std=gnu99 リンカオプション：“Optimize size (-Os)”を使用する場合は、以下のユーザ定義オプションを統合開発環境のデフォルト設定に追加してください。 -WI,--no-gc-sections リンカが誤って FIT 周辺機能モジュールで宣言された割り込み関数を破棄することによる GCC リンカ問題を解決します。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 コンパイラオプション：統合開発環境のデフォルト設定。
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev. 2.00
使用ボード	Renesas Starter Kit+ for RX671 (product No.: RTK55671xxxxxxxxxx)

表 4.4　動作確認環境（Rev. 1.04）

項目	内容
統合開発環境	Renesas Electronics e^2 studio V7.1.0
C コンパイラ	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 コンパイラオプション：統合開発環境のデフォルト設定に以下のオプションを追加。 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev. 1.04
使用ボード	Renesas Solution Starter Kit for RX23W（型名：RTK5523Wxxxxxxxxxx）

表 4.5　動作確認環境（Rev. 1.03）

項目	内容
統合開発環境	Renesas Electronics e^2 studio V7.3.0 IAR Embedded Workbench for Renesas RX 4.10.1
C コンパイラ	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 コンパイラオプション：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 for Renesas RX 4.8.4.2018.01 コンパイラオプション：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。 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.10.1 コンパイラオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev. 1.03
使用ボード	Renesas Starter Kit for RX231（型名：R0K505231S900BE）

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”」エラーが発生しました。
A：FIT モジュールがプロジェクトに正しく追加されていない可能性があります。プロジェクトへの追加方法をご確認ください。
— CS+を使用している場合：
アプリケーションノート「RX ファミリ CS+に組み込む方法 Firmware Integration Technology（R01AN1826）」
— e^2 studio を使用している場合：
アプリケーションノート「RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology（R01AN1723）」
また、本 FIT モジュールを使用する場合、ボードサポートパッケージ FIT モジュール（BSP モジュール）もプロジェクトに追加する必要があります。アプリケーションノート「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。

(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、「This MCU is not supported by the current r_vbatt_rx module.」エラーが発生しました。
A：追加した FIT モジュールがユーザプロジェクトのターゲットデバイスに対応していない可能性があります。追加した FIT モジュールの対象デバイスを確認してください。

(3) 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、「Parameter error in configures file」エラーが発生しました。
A：“r_vbatt_rx_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_vbatt_rx_config.h”ファイルを確認してください。設定が間違っている場合は、その設定に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　サンプルコード

4.3.1　RTC FIT モジュールと組み合わせて使用する場合の例

4.3.1.1　デバイス RX230、RX231、RX23W の例

リアルタイムクロックの設定に RTC FIT モジュールを使用し、バッテリバックアップ機能 FIT モジュールと組み合わせて使用する場合のサンプルコードを示します。

コンフィグレーションオプションは次のとおりに設定します。

- BSP FIT モジュールの BSP_CFG_OFS1_REG_VALUE は“0xFFFFFFFA”
- BSP FIT モジュールの BSP_CFG_RTC_ENABLE は“1”
- BSP FIT モジュールの BSP_CFG_SOSC_DRV_CAP は“0”または“1”
- RTC FIT モジュールと VBATT FIT モジュールの設定は、デフォルト値。

次の(1)～(3)の順に動作します。

(1) R_VBATT_Open()関数をコールする。
（R_VBATT_Open()関数をコールするとコールバック関数が呼び出される。）

(2) バッテリバックアップ機能のコールバック関数で、バッテリバックアップ電源電圧の低下の有無に応じて、RTC を設定する。バッテリバックアップ電源電圧の低下の有無は、コールバック関数の引数で判断する。
(2-A) コールバック関数の引数が VBATT_NOT_DROP_VOLTAGE の場合、R_RTC_Open()関数と R_RTC_Read()関数を使用して RTC の FIT モジュールを再設定する。
(2-B) コールバック関数の引数が VBATT_DROP_VOLTAGE の場合、R_RTC_Open()関数を使用して RTC を初期化する。

(3) RTC 周期割り込みのコールバック関数で、R_RTC_Read()関数を使用して現在時刻を読み出す。その読み出した時刻をデバッグコンソールに表示する。

```
#include <stdio.h>                          /* デバッグ表示用として printf()使用のため必要 */
#include "r_rtc_rx_if.h"
#include "r_vbatt_rx_if.h"

static void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event);
static void rtc_callback(void *event);

static tm_t rtc_curr_time;                  /* RTC から現在時刻を保存するために使用されるデータ構造体 */

void main(void)
{
    vbatt_return_t      ret;                /* API 関数からの戻り値確認用 */
    vbatt_info_t        vbatt_info;         /* バッテリバックアップ機能のデータ構造体 */

   SYSTEM.RSTSR1.BIT.CWSF = 1;              /* コールド/ウォームスタート判定フラグ */

    vbatt_info.callbackfunc = vbatt_callback;   /* コールバック関数の設定 */

    /* VBATT 端子電圧検出機能の設定およびバッテリバックアップ電源電圧低下の判定 */
    ret = R_VBATT_Open(&vbatt_info);
    if (VBATT_SUCCESS != ret)
    {
        while(1);
    }

    while(1);
}


/* バッテリバックアップ機能のコールバック関数 */
static void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event)
{
    rtc_err_t    ret;                       /* API 関数からの戻り値確認用 */
    rtc_init_t   rtc_info                   /* RTC データ構造体 */
```

この関数実行により、設定したコールバック関数（vbatt_callback()）が呼び出されます。

図4.1　RTC FIT モジュールと組み合わせて使用する例（1/3）

```
    /* 引数を判定 */
    switch(*vbatt_cb_event)
    {
        /* バッテリバックアップ電源電圧の低下が未検出の場合 */
        case VBATT_NOT_DROP_VOLTAGE:
            /* RTC データ構造体を再設定 */
            rtc_info.output_freq = RTC_OUTPUT_OFF;          /* RTCOUT 出力禁止 */
            rtc_info.periodic_freq = RTC_PERIODIC_1_HZ;     /* RTC の周期割り込み発生の周期： 1 秒ごと */
            rtc_info.periodic_priority = 8;                 /* 割り込み優先レベル */
            rtc_info.set_time = false;                      /* RTC クロックカウンタの I/O レジスタを更新しない */
            rtc_info.p_callback = rtc_callback;             /* コールバック関数の設定 */

            /* RTC の再設定 */
            ret = R_RTC_Open(&rtc_info, &rtc_curr_time);
            if (RTC_SUCCESS != ret)
            {
                while(1);
            }

            /* RTC クロックカウンタの I/O レジスタから現在時刻情報を読み出し
             データ構造体に保存 */
            ret = R_RTC_Read(&rtc_curr_time, NULL);
            if (RTC_SUCCESS != ret)
            {
                while(1);
            }

        break;

        /* バッテリバックアップ電源電圧の低下を検出した場合 */
        case VBATT_DROP_VOLTAGE:
            /* RTC データ構造体の設定 */
            rtc_info.output_freq = RTC_OUTPUT_OFF;          /* RTCOUT 出力禁止 */
            rtc_info.periodic_freq = RTC_PERIODIC_1_HZ;     /* RTC の周期割り込み発生の周期： 1 秒ごと */
            rtc_info.periodic_priority = 8;                 /* 割り込み優先レベル */
            rtc_info.set_time = true;                       /* RTC クロックカウンタの I/O レジスタを更新 */
            rtc_info.p_callback = rtc_callback;             /* コールバック関数の設定 */

            /* 現在時刻の情報構造体の時刻設定を"2015-06-30 12:34:56"に設定 */
            rtc_curr_time.tm_sec  = 56;                     /* 秒（0～59） */
            rtc_curr_time.tm_min  = 34;                     /* 分（0～59） */
            rtc_curr_time.tm_hour = 12;                     /* 時（0～23） */
            rtc_curr_time.tm_mday = 30;                     /* 日（1～31） */
            rtc_curr_time.tm_mon  = 6;                      /* 月 （0～11、0=1 月） */
            rtc_curr_time.tm_year = 115;                    /* 年（基準は 1900 年） */
            rtc_curr_time.tm_wday = 0;                      /* 曜日（0～6、0=日曜日） */
            rtc_curr_time.tm_yday = 0;                      /* 年間の日数（0～365） */
            rtc_curr_time.tm_isdst = 0;                     /* サマータイム有効（>0）、
                                                            無効（=0） */

            /* RTC の初期化 */
            ret = R_RTC_Open(&rtc_info, &rtc_curr_time);
            if (RTC_SUCCESS != ret)
            {
                while(1);
            }

        break;

        /* VBATT 端子電圧の低下による割り込み */
        case VBATT_MASKABLE_INTERRUPT:
        case VBATT_NON_MASKABLE_INTERRUPT:
            /* 本サンプルコードでは未使用 */
        break;

        default:
            /* 処理なし */
        break;
    }
}
```

RTC の I/O レジスタは保存されますが、RAM やその他のレジスタはリセットされます。このため、RTC FIT モジュールを再設定しています。

RTC の I/O レジスタは保存されているので、現在時刻の情報は、R_RTC_Read()関数によって RTC の I/O レジスタから読み出され、データ構造体へ再保存されます。

バッテリバックアップ電源電圧の低下が検出された場合、マイクロコントローラは RTC 動作が保証されない状態となっているため、RTC は再初期化されます。

図4.2　RTC FIT モジュールと組み合わせて使用する例（2/3）

```
/* RTC のコールバック関数 */
static void rtc_callback(void *event)
{
    rtc_err_t ret;                                  /* API 関数からの戻り値確認用 */

    /* 周期割り込みの場合 */
    if (*(rtc_cb_evt_t *)event == RTC_EVT_PERIODIC)
    {
        /* 周期割り込みの場合 */
        /* RTC クロックカウンタの I/O レジスタから現在時刻情報を読み出し
           データ構造体に保存 */
        ret = R_RTC_Read(&rtc_curr_time, NULL);
        if (RTC_SUCCESS != ret)
        {
            while(1);
        }

        /* デバッグコンソールへの表示 */
        printf("%d/%d/%d %02d:%02d:%02d¥n",     rtc_curr_time.tm_year + 1900,
                                                rtc_curr_time.tm_mon,
                                                rtc_curr_time.tm_mday,
                                                rtc_curr_time.tm_hour,
                                                rtc_curr_time.tm_min,
                                                rtc_curr_time.tm_sec);
    }
}
```

周期割り込み発生ごと（1 秒ごと）に現在時刻を読み出しています。

図4.3　RTC FIT モジュールと組み合わせて使用する例（3/3）

4.3.1.2　デバイス RX671 の例

リアルタイムクロックの設定に RTC FIT モジュールを使用し、バッテリバックアップ機能 FIT モジュールと組み合わせて使用する場合のサンプルコードを示します。また、タイムキャプチャおよび改ざん機能を同時に使用する方法も示します。

コンフィグレーションオプションは次のとおりに設定します。

- BSP FIT モジュールの BSP_CFG_OFS1_REG_VALUE は“0xFFFFFFFA”
- BSP FIT モジュールの BSP_CFG_RTC_ENABLE は“1”
- BSP FIT モジュールの BSP_CFG_SOSC_DRV_CAP は“0”または“1”
- VBATT FIT モジュールの VBATT_CFG_TAMPER_CH0 は“VBATT_TAMPER_ENABLE”
- VBATT FIT モジュールの VBATT_CFG_TAMPER_CH0_DETECT_INT は“VBATT_TAMPER_DETECT_INT_ENABLE”
- VBATT FIT モジュールの VBATT_CFG_TAMPER_CH0_ERASE は“VBATT_TAMPER_ERASE_DISABLE”
- VBATT FIT モジュールの VBATT_CFG_TAMPER_TCE_CH0_SELECT は“VBATT_TAMPER_TCE_TAMPER_EVENT”
- VBATT FIT モジュールの VBATT_CFG_TAMPER_CHEN_CH0_NOISE_FILTER は“VBATT_TAMPER_CHEN_NOISE_FILTER_ENABLE”
- VBATT FIT モジュールの VBATT_CFG_TAMPER_CHEN_CH0_EDGE は“VBATT_TAMPER_CHEN_RISING_EDGE”
- VBATT FIT モジュール設定のその他には、デフォルト値が使用されます。
- RTC FIT モジュール設定には、デフォルト値が使用されます。

次の(1)～(3)の順に動作します。

(1) R_VBATT_Open()関数をコールする。
（R_VBATT_Open()関数をコールするとコールバック関数が呼び出される。）

(2) バッテリバックアップ機能のコールバック関数で、バッテリバックアップ電源電圧の低下の有無に応じて、RTC を設定する。バッテリバックアップ電源電圧の低下の有無は、コールバック関数の引数で判断する。
(2-A) コールバック関数の引数が VBATT_NOT_DROP_VOLTAGE の場合、R_RTC_Open()、R_RTC_Read()、R_RTC_Control()関数を使用して RTC の FIT モジュールを再設定する。
(2-B) コールバック関数の引数が VBATT_DROP_VOLTAGE の場合、R_RTC_Open()および R_RTC_Control()関数を使用して RTC を初期化する。
(2-C) コールバック関数の引数が VBATT_TAMPER_CH0_INTERRUPT の場合、必要な処理を実行する。

(3) RTC 周期割り込みのコールバック関数で、R_RTC_Read()関数を使用して現在時刻を読み出す。その読み出した時刻をデバッグコンソールに表示する。

```
#include <stdio.h>                          /* デバッグ表示用として printf()使用のため必要 */
#include "r_rtc_rx_if.h"
#include "r_vbatt_rx_if.h"

static void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event);
static void rtc_callback(void *event);

static tm_t rtc_curr_time;                  /* RTC から現在時刻を保存するために使用されるデータ構造体 */
static tm_t rtc_capture_time;               /* RTC タイムキャプチャ */

void main(void)
{
    vbatt_return_t      vbatt_ret;          /* API 関数からの戻り値確認用 */
    vbatt_info_t        vbatt_info;         /* バッテリバックアップ機能のデータ構造体 */
     rtc_err_t           rtc_ret;           /* API 関数からの戻り値確認用 */

   SYSTEM.RSTSR1.BIT.CWSF = 1;              /* コールド/ウォームスタート判定フラグ */

    vbatt_info.callbackfunc = vbatt_callback;   /* コールバック関数の設定 */

    /* VBATT 端子電圧検出機能の設定およびバッテリバックアップ電源電圧低下の判定 */
    vbatt_ret = R_VBATT_Open(&vbatt_info);
    if (VBATT_SUCCESS != vbatt_ret)
    {
        while(1);
    }

    while(1)
        {
         /* タイムキャプチャを取得 */
         rtc_ret = R_RTC_Control(RTC_CMD_CHECK_PIN0_CAPTURE, &rtc_capture_time);
         if (RTC_SUCCESS == rtc_ret)
         {
              printf("tamper 0 detect time: %d/%d %02d:%02d:%02d¥n", rtc_capture_time.tm_mon,
                                                                     rtc_capture_time.tm_mday,
                                                                     rtc_capture_time.tm_hour,
                                                                     rtc_capture_time.tm_min,
                                                                     rtc_capture_time.tm_sec);
         }
    }
}

/* バッテリバックアップ機能のコールバック関数 */
static void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event)
{
    rtc_err_t     ret;                      /* API 関数からの戻り値確認用 */
    rtc_init_t    rtc_info;                 /* RTC データ構造体 */
    rtc_capture_cfg_t capture;              /* RTC キャプチャ構成の構造体 */
    vbatt_status_t      vbatt_status;
    vbatt_return_t    vbatt_ret;            /* API 関数からの戻り値確認用 */
```

この関数実行により、設定したコールバック関数（vbatt_callback()）が呼び出されます。

図4.4　RTC FIT モジュールと組み合わせて使用する例（1/3）

```
/* 引数を判定 */
switch(*vbatt_cb_event)
{
    /* バッテリバックアップ電源電圧の低下が未検出の場合 */
    case VBATT_NOT_DROP_VOLTAGE:
        /* RTC データ構造体を再設定 */
        rtc_info.output_freq = RTC_OUTPUT_OFF;          /* RTCOUT 出力禁止 */
        rtc_info.periodic_freq = RTC_PERIODIC_1_HZ;     /* RTC の周期割り込み発生の周期： 1 秒ごと */
        rtc_info.periodic_priority = 8;                 /* 割り込み優先レベル */
        rtc_info.set_time = false;                      /* RTC クロックカウンタの I/O レジスタを更新しない */
        rtc_info.p_callback = rtc_callback;             /* コールバック関数の設定 */

        /* RTC キャプチャ構成の構造体を再設定 */
        capture.pin = RTC_PIN_0;
        capture.edge = RTC_EDGE_RISING;                 /* RX671 を使用する場合は常に立ち上がりエッジ */
        capture.filter = RTC_FILTER_OFF;                /* VBATT_CFG_TAMPER_CHEN_CH0_NOISE_FILTER が
                                                           VBATT_TAMPER_CHEN_NOISE_FILTER_ENABLE の場合はフィルタを OFF */

        /* RTC の再設定 */
        ret = R_RTC_Open(&rtc_info, &rtc_curr_time);
        if (RTC_SUCCESS != ret)
        {
            while(1);
        }

        /* RTC クロックカウンタの I/O レジスタから現在時刻情報を読み出し
           データ構造体に保存 */
        ret = R_RTC_Read(&rtc_curr_time, NULL);
        if (RTC_SUCCESS != ret)
        {
            while(1);
        }

        /* RTC 構成タイムキャプチャ */
        ret = R_RTC_Control(RTC_CMD_CONFIG_CAPTURE, &capture);
        if (RTC_SUCCESS != ret)
        {
            while(1);
        }

    break;

    /* バッテリバックアップ電源電圧の低下を検出した場合 */
    case VBATT_DROP_VOLTAGE:
        /* RTC データ構造体の設定 */
        rtc_info.output_freq = RTC_OUTPUT_OFF;          /* RTCOUT 出力禁止 */
        rtc_info.periodic_freq = RTC_PERIODIC_1_HZ;     /* RTC の周期割り込み発生の周期： 1 秒ごと */
        rtc_info.periodic_priority = 8;                 /* 割り込み優先レベル */
        rtc_info.set_time = true;                       /* RTC クロックカウンタの I/O レジスタを更新 */
        rtc_info.p_callback = rtc_callback;             /* コールバック関数の設定 */

        /* 現在時刻の情報構造体の時刻設定を"2015-06-30 12:34:56"に設定 */
        rtc_curr_time.tm_sec  = 56;                     /* 秒（0～59） */
        rtc_curr_time.tm_min  = 34;                     /* 分（0～59） */
        rtc_curr_time.tm_hour = 12;                     /* 時（0～23） */
        rtc_curr_time.tm_mday = 30;                     /* 日（1～31） */
        rtc_curr_time.tm_mon  = 6;                      /* 月 （0～11、0=1 月） */
        rtc_curr_time.tm_year = 115;                    /* 年（基準は 1900 年） */
        rtc_curr_time.tm_wday = 0;                      /* 曜日（0～6、0=日曜日） */
        rtc_curr_time.tm_yday = 0;                      /* 年間の日数（0～365） */
        rtc_curr_time.tm_isdst = 0;                     /* サマータイム有効（>0）、
                                                           無効（=0）*/

        /* RTC キャプチャ構成の構造体を再設定 */
        capture.pin = RTC_PIN_0;
        capture.edge = RTC_EDGE_RISING;
        capture.filter = RTC_FILTER_OFF;

        /* RTC の初期化 */
        ret = R_RTC_Open(&rtc_info, &rtc_curr_time);
        if (RTC_SUCCESS != ret)
        {
            while(1);
        }

        /* RTC 構成タイムキャプチャ */
        ret = R_RTC_Control(RTC_CMD_CONFIG_CAPTURE, &capture);
```

RTC の I/O レジスタは保存されますが、RAM やその他のレジスタはリセットされます。このため、RTC FIT モジュールを再設定しています。

RTC の I/O レジスタは保存されているので、現在時刻の情報は、R_RTC_Read()関数によって RTC の I/O レジスタから読み出され、データ構造体へ再保存されます。

バッテリバックアップ電源電圧の低下が検出された場合、マイクロコントローラは RTC 動作が保証されない状態となっているため、RTC は再初期化されます。

```
                if (RTC_SUCCESS != ret)
                {
                    while(1);
                }

            break;

            /* VBATT 端子電圧の低下による割り込み */
            case VBATT_MASKABLE_INTERRUPT:
            case VBATT_NON_MASKABLE_INTERRUPT:
                /* 本サンプルコードでは未使用 */
            break;

           /* 改ざん検出割り込み */
            case VBATT_TAMPER_CH0_INTERRUPT:
                /* Tamper detection notification */
                printf("tamper 0 detect!!¥n");

                /* サブクロックの 1 周期を待機 */
                R_BSP_SoftwareDelay(33, BSP_DELAY_MICROSECS);

                /* 改ざん検出クリア */
                vbatt_status.tamper_channel = VBATT_TAMPER_CH0;
                vbatt_status.action_clear = true;
                vbatt_ret = R_VBATT_GetStatus(&vbatt_status);
                if (VBATT_SUCCESS != vbatt_ret)
                {
                    /* エラー発生時に処理を実行してください */
                }
            break;
           /* 改ざん検出割り込み */
            case VBATT_TAMPER_CH1_INTERRUPT:
                /* 本サンプルコードでは未使用 */
            break;

           /* 改ざん検出割り込み */
            case VBATT_TAMPER_CH2_INTERRUPT:
                /* 本サンプルコードでは未使用 */
            break;

            default:
                /* 処理なし */
            break;
    }
}
```

図4.5　RTC FIT モジュールと組み合わせて使用する例（2/3）

```
/* RTC のコールバック関数 */
static void rtc_callback(void *event)
{
    rtc_err_t ret;                                  /* API 関数からの戻り値確認用 */

    /* 周期割り込みの場合 */
    if (*(rtc_cb_evt_t *)event == RTC_EVT_PERIODIC)
    {
        /* 周期割り込みの場合 */
        /* RTC クロックカウンタの I/O レジスタから現在時刻情報を読み出し
           データ構造体に保存 */
        ret = R_RTC_Read(&rtc_curr_time, NULL);
        if (RTC_SUCCESS != ret)
        {
            while(1);
        }

        /* デバッグコンソールへの表示 */
        printf("%d/%d/%d %02d:%02d:%02d¥n",     rtc_curr_time.tm_year + 1900,
                                                rtc_curr_time.tm_mon,
                                                rtc_curr_time.tm_mday,
                                                rtc_curr_time.tm_hour,
                                                rtc_curr_time.tm_min,
                                                rtc_curr_time.tm_sec);
    }
}
```

周期割り込み発生ごと（1 秒ごと）に現在時刻を読み出しています。

図4.6　RTC FIT モジュールと組み合わせて使用する例（3/3）

5. デモプロジェクト

デモプロジェクトには、FIT モジュールとそのモジュールが依存するモジュール（例：r_bsp）を使用する main()関数が含まれます。本 FIT モジュールには以下のデモプロジェクトが含まれます。

5.1 vbatt_demo_rskrx671, vbatt_demo_rskrx671_gcc

これは RSK RX671 用のバッテリバックアップ機能(VBATTB)のデモです(FIT モジュール"r_vbatt_rx")。このサンプルデモはリアルタイムクロック設定に使用される RTC FIT モジュールと組み合わせたバッテリバックアップ機能 FIT モジュールの使用方法を紹介します。また、タイマキャプチャとタンパ機能の使用方法についても紹介します。

5.2 ワークスペースにデモを追加する

デモプロジェクトは、本アプリケーションノートで提供されるファイルの FITDemos サブディレクトリにあります。ワークスペースにデモプロジェクトを追加するには、「ファイル」 >> 「インポート」を選択し、「インポート」ダイアログから「一般」の「既存プロジェクトをワークスペースへ」を選択して「次へ」ボタンをクリックします。「インポート」ダイアログで「アーカイブ・ファイルの選択」ラジオボタンを選択し、「参照」ボタンをクリックして FITDemos サブディレクトリを開き、使用するデモの zip ファイルを選択して「終了」をクリックします。

5.3 デモのダウンロード方法

デモプロジェクトは、RX Driver Package には同梱されていません。デモプロジェクトを使用する場合は、個別に各 FIT モジュールをダウンロードする必要があります。「スマートブラウザ」の「アプリケーションノート」タブから、本アプリケーションノートを右クリックして「サンプル・コード（ダウンロード）」を選択することにより、ダウンロードできます。

6. 参考ドキュメント

ユーザーズマニュアル：ハードウェア
　最新版をルネサス エレクトロニクスホームページから入手してください。
テクニカルアップデート／テクニカルニュース
　最新の情報をルネサス エレクトロニクスホームページから入手してください。
ユーザーズマニュアル：開発環境
　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。）

テクニカルアップデートの対応について

本モジュールは以下のテクニカルアップデートの内容を反映しています。

なし

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Aug. 24.15	−	初版発行
1.01	Aug. 31.15	プログラム	最新の iodefine.h（V1.0C）に対応するため、バッテリバックアップ機能 FIT モジュールを改修 ■内容 iodefine.h（V0.9E 以降）を使用した場合、コンパイルエラーが発生する。 ■対策 バッテリバックアップ機能 FIT モジュール Rev.1.01 以降をご使用ください。
1.02	Feb. 01.19	プログラム	機能関連 Smart Configurator での GUI によるコンフィグレーションオプション設定機能に対応 ■内容 GUI によるコンフィグレーションオプション設定機能に対応するため、設定ファイルを追加。
1.03	May 20.19	−	以下のコンパイラに対応 - GCC for Renesas RX - IAR C/C++ Compiler for Renesas RX
		1	対象コンパイラを追加
		3	「1.2 バッテリバックアップ機能 FIT モジュールの概要」章を更新
		8	「1.5 制限事項」章を追加
		9	「2.4 使用する割り込みベクタ」章を追加
		11	「2.8 コードサイズ」章を追加
		15	「2.13 for 文、while 文、do while 文について」章を追加
		23	「R_VBATT_GetVersion()」章を更新
		24	「4.1 動作確認環境」章を追加
		プログラム	R_VBATT_GetVersion 関数のインライン展開を削除
1.04	Jun.30.19	1	対象デバイス：RX23W サポートを追加
		24	「表 4.1 動作確認環境（Rev. 1.04）」を「4.1 動作確認環境」章に追加
		26	「4.3 サンプルコード」章を更新
1.05	Jun.10.20	−	API 関数のコメントを Doxygen スタイルに変更
		1	関連ドキュメント R01AN1833 を削除
		14	「2.12 FIT モジュールの追加方法」章を変更
		16～21	API 説明ページの「リエントラント」項目を削除

Rev.	発行日	改訂内容	
		ページ	ポイント
2.00	Mar.31.21	1	RX671 のサポートを追加
		3-11	1.2 バッテリバックアップ機能 FIT モジュールの概要 1.3 API の概要 1.4 使用例 1.5 制限事項 RX671 の説明を更新
		12	RX671 の割り込みベクタ番号を追加
		13-16	2.7 コンパイル時の設定 RX671 の説明を更新
		17	「2.8 コードサイズ」章を追加
		18-19	「2.9 引数」章の RX671 の構造体を更新
		23-36	第 3 章 API 関数 デバイス RX671 の例を更新
		37	「R_VBATT_ReadBackupData()」章を更新
		38	「R_VBATT_WriteBackupData()」章を更新
		40	4.1 動作確認環境： Rev.2.00 の表を追加
		プログラム	RX671 のサポートを追加
2.10	Sep.13.21	41	4.1 動作確認環境： Rev.2.10 の表を追加
		52	「5.デモプロジェクト」の記載内容見直し
2.20	Jul.29.22	41	4.1 動作確認環境： Rev.2.20 の表を追加
		プログラム	デモプロジェクトを更新。

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。