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。

対象デバイス

・RX230 グループ

・RX231 グループ

・RX23W 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

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、「2.12FIT 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 4 つの状況を判別できます。

(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
(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

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、コンフィギュレーションファイル（r_vbatt_rx_config.h）のコンフィギュ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）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。

コンフィギュ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。

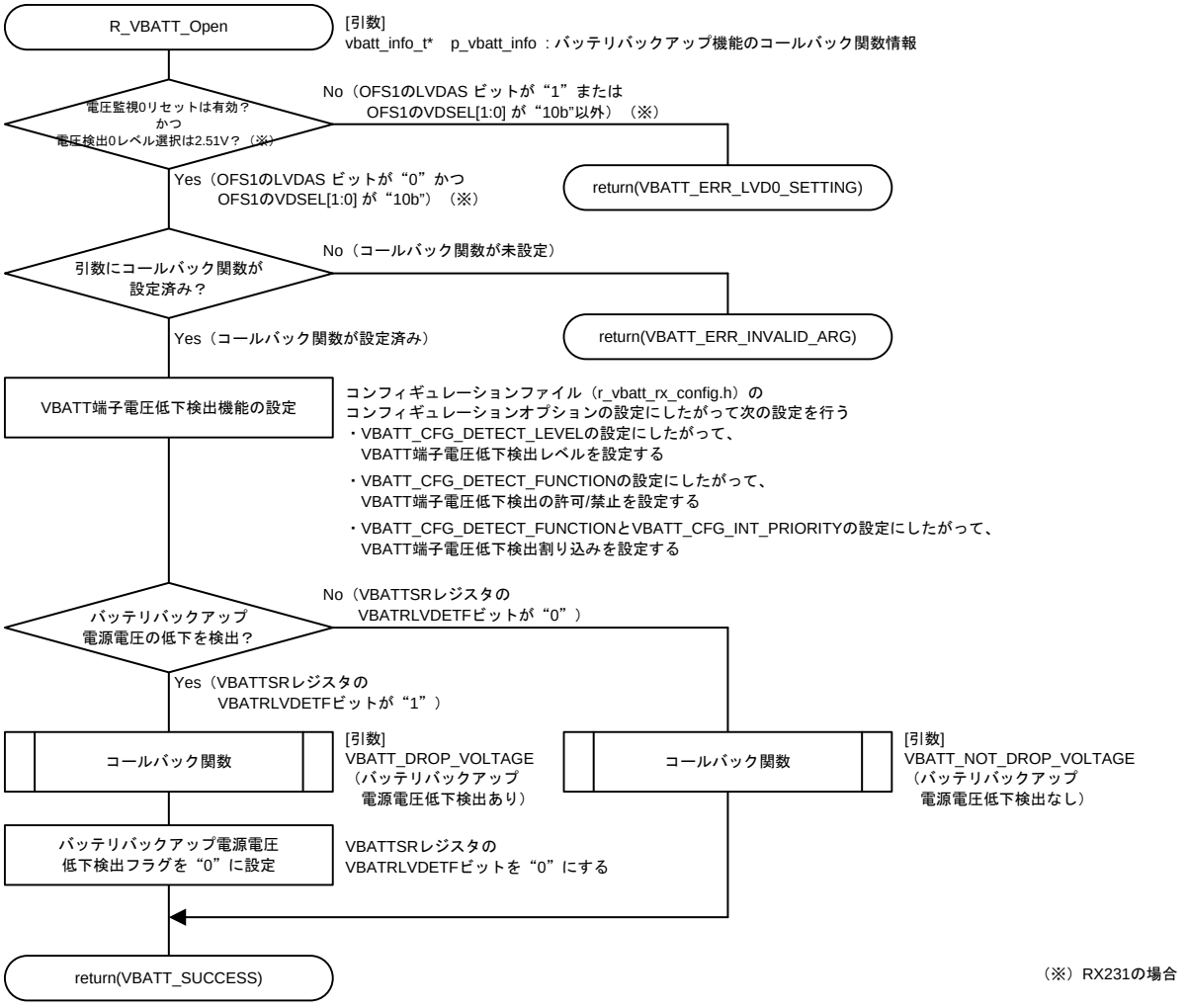

図 1.1　R_VBATT_Open 関数のフローチャート

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、引数の設定にしたがって、VBATT 端子電圧低下検出機能の有効／無効や検出レベル、割り込みを設定します。

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

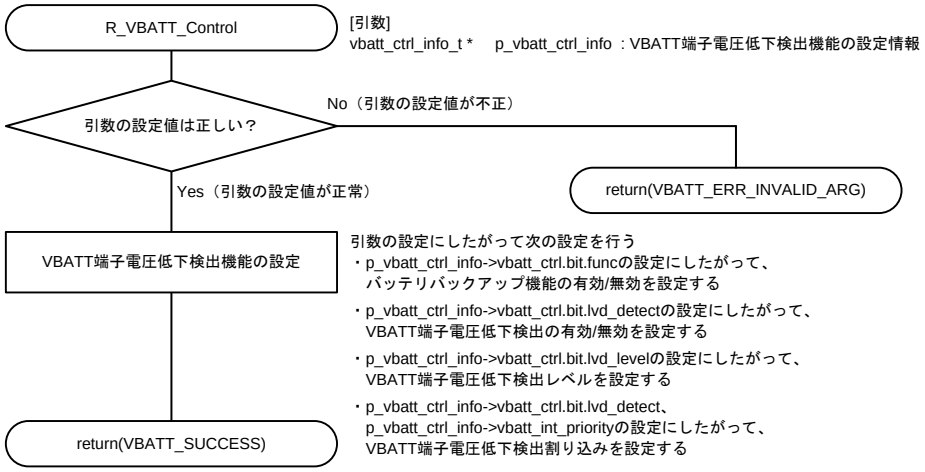

図 1.2　R_VBATT_Control 関数のフローチャート

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、VBATT ステータスレジスタ（VBATTSR）の値を読み出します。読み出した値は、引数で受け取ったアドレスに設定します。

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

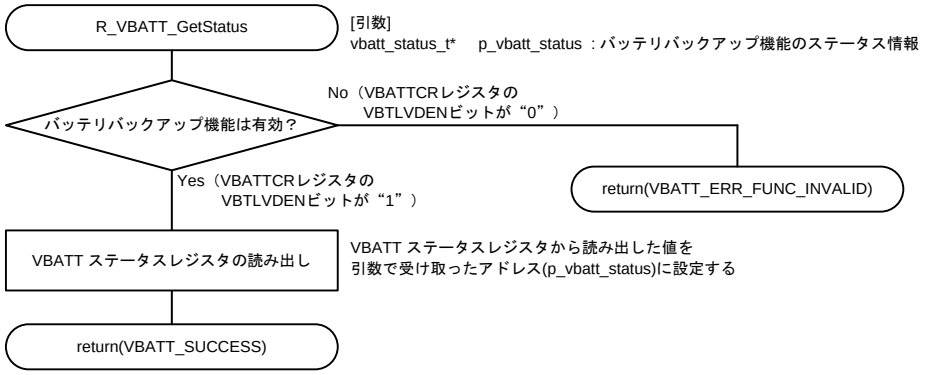

図 1.3　R_VBATT_GetStatus 関数のフローチャート

1.2.2　割り込みの仕様

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。

図 1.4 に r_vbatt_isr 関数のフローチャートを示します。

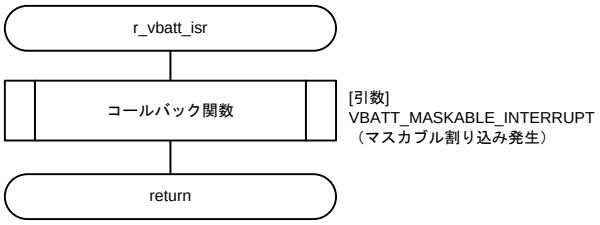

図 1.4　r_vbatt_isr 関数のフローチャート

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 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。

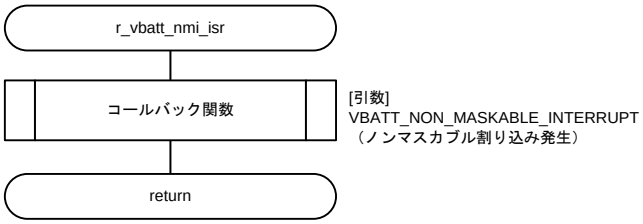

図 1.5　r_vbatt_nmi_isr 関数のフローチャート

1.3　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 関数一覧

関数	関数説明
R_VBATT_Open()	コンフィギュレーションオプションの設定にしたがって、VBATT 端子電圧低下検出機能の有効／無効や検出レベル、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。
R_VBATT_Control()	引数の設定にしたがって、VBATT 端子電圧低下検出機能の有効／無効や検出レベル、割り込みを設定します。
R_VBATT_GetStatus()	バッテリバックアップ機能の状態を取得します。
R_VBATT_GetVersion()	本モジュールのバージョン番号を返します。

1.4　処理例

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を、図 1.6 に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）
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再設定してください。	リアルタイムクロックを初期設定してください。	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。
理由	バッテリバックアップ電源が低下しなかったときは、RTC のレジスタの値は保持されますが、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、初期設定が必要となります。	VBATT 端子の電源電圧が低下したときに発生する割り込みです。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。

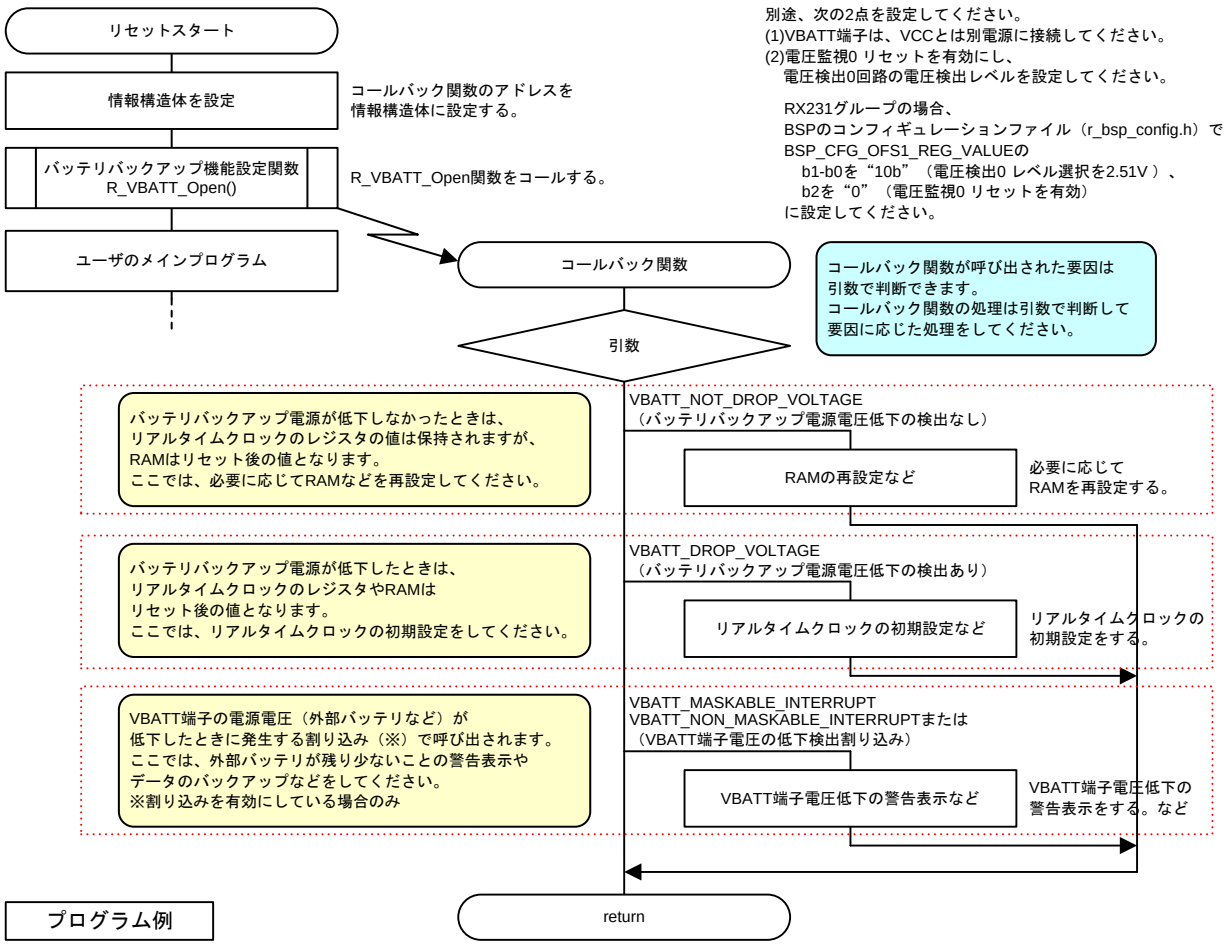

図 1.6　バッテリバックアップ機能 FIT モジュールの使用例

1.5　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 0 リセットを有効に設定し、電圧検出 0 回路の電圧検出レベルを選択してください。（RX231 グループの場合 2.51V）

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 以上

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

マクロ定義 VBATT_CFG_DETECT_FUNCTION が VBATT_DTCT_ENABLE_NMI_ENABLE または VBATT_DTCT_ENABLE_INT_ENABLE の時に 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。

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

【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 に記載しています。

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ギュ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。

Configuration options in r_vbatt_rx_config.h	
#define VBATT_CFG_DETECT_FUNCTION ※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 ※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。
#define VBATT_CFG_INT_PRIORITY ※デフォルト値は“5”	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。 ※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。

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「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。

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 rev1.03

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00
(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 4.8.4.201801
(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 4.10.1
(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	**6117 バイト**	**6060 バイト**	**8844 バイト**	**8788 バイト**	**5126 バイト**	**4946 バイト**
	RAM	**6958 バイト**		**6764 バイト**		**1640 バイト**	
	スタック	**128 バイト**		-		**184 バイト**	
RX231	ROM	**6177 バイト**	**6120 バイト**	**8972 バイト**	**8916 バイト**	**5206 バイト**	**5026 バイト**
	RAM	**6974 バイト**		**6780 バイト**		**1656 バイト**	
	スタック	**128 バイト**		-		**184 バイト**	

注 1　BSP を含んだサイズです。

2.9　引数

API 関数の引数であ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 に記載されています。

```
/* バッテリバックアップ機能の情報構造体 */
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
{
    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;
} vbatt_ctrl_info_t;



/* バッテリバックアップ機能の状態を格納するための構造体 */
typedef volatile struct
{
    union
    {
        uint8_t     byte;
        struct
        {
            uint8_t  rsv:6;             /* 予約領域 */
            uint8_t  vbatt_mon:1;       /* VBATT 端子電圧モニタフラグ */
            uint8_t  pwr_drp_dtct:1;    /* バッテリバックアップ電源電圧低下検出フラグ */
        } bit;
    } vbatt_status;
} 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 で記載されています。

```
typedef enum                        /* バッテリバックアップ機能 API 関数コール時の戻り値 */
{
    VBATT_SUCCESS,                  /* 問題なく処理が終了した場合 */
    VBATT_ERR_LOCK_FUNC,            /* 関数が多重コールされた場合 */ （未実装）
    VBATT_ERR_LVD0_SETTING,         /* オプション機能選択レジスタ 1（OFS1）による
                                       電圧監視 0 の設定が不正である場合 */
    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の状態で
                                       R_VBATT_GetStatus 関数を実行した場合 */
    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の引数に渡される定数定義一覧(enum vbatt_cb_evt_t)を示します。

コールバック関数の設定は、「2.9 引数」に記載の構造体メンバ“callbackfunc”に、コールバック関数として登録したい関数のアドレスを格納してください。

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

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、スマート・コンフィグレータを使用した(1)、(3)、(5)の追加方法を推奨しています。ただし、スマート・コンフィグレータは、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)、(4)の方法を使用してください。

(1) e^2 studio 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
e^2 studio のスマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: e^2 studio 編 (R20AN0451)」を参照してください。

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

(3) CS+上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
CS+上で、スタンドアロン版スマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: CS+編 (R20AN0470)」を参照してください。

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

(5) IAREW 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
スタンドアロン版スマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: IAREW 編 (R20AN0535)」を参照してください。

2.13　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)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}
```

for 文の例：

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

do while 文の例：

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

3.　API 関数

R_VBATT_Open ()

この関数は、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、本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。

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機能の情報構造体のポインタ。

本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	*/* 問題なく処理が終了した場合 */*
VBATT_ERR_LVD0_SETTING	*/* オプション機能選択レジスタ 1（OFS1）による電圧監視 0 の設定が不正である場合 */*
VBATT_ERR_INVALID_ARG	*/* 引数が不正である場合 */*

Properties

r_vbatt_rx_if.h にプロトタイプ宣言されています。

Description

コンフィギュレーションオプションの設定にしたがって、VBATT 端子電圧低下検出機能の有効／無効や検出レベル、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。

コールバック関数を呼び出し時、

・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場合、
VBATT_NOT_DROP_VOLTAGE を設定した変数のポインタを引数に渡します。

・バッテリバックアップ電源電圧の低下を検出している場合、
VBATT_DROP_VOLTAGE を設定した変数のポインタを引数に渡します。

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)
 {
       /* Please do the processing at the time of the error */
 }

 while(1);

}

void vbatt_callback(vbatt_cb_evt_t * vbatt_cb_event)
{
 switch(*vbatt_cb_event)
 {
       /* Battery backup power voltage drop not detected */
       case VBATT_NOT_DROP_VOLTAGE:

             /* Please set RAM again as needed */

       break;

       /* Battery backup power voltage drop detected */
       case VBATT_DROP_VOLTAGE:

             /* Please initialize the Realtime Clock */

       break;

       /* VBATT voltage drop detected interrupt */
       case VBATT_MASKABLE_INTERRUPT:
       case VBATT_NON_MASKABLE_INTERRUPT:

             /* Please process warning indication, backup, etc. */

       break;

       default:

             /* Do nothing */

       break;
 }
}
```

Special Notes:

なし

R_VBATT_Control ()

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の設定をする関数です。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。

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の情報構造体のポインタ。

本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
{
        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;
} vbatt_ctrl_info_t;
```

Return Values

```
VBATT_SUCCESS                   /*  問題なく処理が終了した場合 */
VBATT_ERR_INVALID_ARG           /*  引数が不正である場合 */
```

Properties

r_vbatt_rx_if.h にプロトタイプ宣言されています。

Description

引数の設定にしたがって、バッテリバックアップ機能の有効／無効や 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。

Example

```
#include "r_vbatt_rx_if.h"

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

 /* Battery backup function enable */
 vbatt_ctrl_info.vbatt_ctrl.bit.func = 1;
 /* VBATT drop detect function enable and maskable interrupt enable */
 vbatt_ctrl_info.vbatt_ctrl.bit.lvd_detect = VBATT_DTCT_ENABLE_INT_ENABLE;
 /* VBATT drop detect level is 2.00V */
 vbatt_ctrl_info.vbatt_ctrl.bit.lvd_level = VBATT_DTCT_LEVEL_2_00_V;
 /* interrupt priority level is 7*/
 vbatt_ctrl_info.vbatt_int_priority = 7;

 ret = R_VBATT_Control(&vbatt_ctrl_info);
 if (VBATT_SUCCESS != ret)
 {
       /* Please do the processing at the time of the error */
 }

 while(1);
}
```

Special Notes:

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。

構造体(vbatt_ctrl_info_t)		設定可能範囲	設定内容
メンバ	ビット		
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”の場合、バッテリバックアップ機能を「無効」にします。 “1”の場合、バッテリバックアップ機能を「有効」にします。
	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。 ※本設定は lvd_detect で“VBATT_DTCT_ENABLE_INT_ENABLE”を選択したときのみ有効です。

R_VBATT_GetStatus()

バッテリバックアップ機能の状態を取得する関数です。バッテリバックアップ機能の状態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の状態を格納する変数のポインタ。

本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
{
        union
        {
                uint8_t         byte;
                struct
                {
                        uint8_t  rsv:6;             /*  予約領域 */
                        uint8_t  vbatt_mon:1;     /* VBATT 端子電圧モニタフラグ */
                        uint8_t  pwr_drp_dtct:1; /*バッテリバックアップ電源電圧低下検出フラグ */
                } bit;
        } vbatt_status;
} vbatt_status_t;
```

Return Values

VBATT_SUCCESS	/* 問題なく処理が終了した場合 */
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状態で R_VBATT_GetStatus 関数を実行した場合 */

Properties

r_vbatt_rx_if.h にプロトタイプ宣言されています。

Description

VBATT ステータスレジスタ（VBATTSR）を読み出して、バッテリバックアップ機能の状態を取得し、引数で受け取ったアドレスに情報を格納します。

Example

```
#include "r_vbatt_rx_if.h"

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

 /* VBATT drop detect function enable */
 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)
 {
       /* Please do the processing at the time of the error */
 }

 /* gets the state of the battery backup function */
 ret = R_VBATT_GetStatus(&vbatt_status);
 if (VBATT_SUCCESS != ret)
 {
       /* Please do the processing at the time of the error */
 }

 while(1);
}
```

Special Notes:

以下にステータスフラグの配置を示します。

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	バッテリバックアップ 電源電圧低下検出フラグ
シンボル	rsv	vbatt_mon	pwr_drp_dtct
機能	不定	0:VBATT < Vdetvbt 1:VBATT ≧ Vdetvbt または VBATT 検出機能無効	0:バッテリバックアップ電源 電圧の低下は未検出 1:バッテリバックアップ電源 電圧の低下を検出

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

r_vbatt_rx_if.h にプロトタイプ宣言されています。

Description

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.1.04)

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 V7.1.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev1.04
使用ボード	Renesas Solution Starter Kit for RX23W（型名：RTK5523Wxxxxxxxxx）

表 4.2 動作確認環境 (Rev.1.03)

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 V7.3.0 IAR Embedded Workbench for Renesas RX 4.10.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.01.00 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 4.8.4.2018.01 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
	IAR C/C++ Compiler for Renesas RX version 4.10.1 コンパイルオプション：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	Rev1.03
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"」エラーが発生します。

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

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

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

(2) Q : 本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「This MCU is not supported by the current r_vbatt_rx module.」エラーが発生します。

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」エラーが発生します。

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

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。

コンフィギュレーションオプションは次のとおり設定します。

- BSP FIT モジュールの BSP_CFG_OFS1_REG_VALUE は“0xFFFFFFFA”
- 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 を初期設定する。

(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 */
#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 の現在時刻を格納するための情報構造体 */

void main(void)
{
    vbatt_return_t      ret;                                /* API 関数の戻り値確認用 */
    vbatt_info_t        vbatt_info;                          /* バッテリバックアップ機能の情報構造体 */

    SYSTEM.RSTSR1.BIT.CWSF = 1;                              /* コールドスタート/ウォームスタート判別フラグを設定 */

    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を判定 */
    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 関数の戻り値確認用 */
    rtc_init_t  rtc_info;                                    /* RTC の情報構造体 */
```

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 の情報構造体を再設定 */
            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 時計カウンタの 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 時計カウンタの I/O レジスタから現在時刻情報を読み出し情報構造体に設定 */
            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 情報構造体の設定 */
            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 時計カウンタの I/O レジスタを更新する */
            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 の初期設定 */
            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 レジスタは保持されていますが、RAM やその他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 レジスタは保持されているので、現在時刻の情報は、R_RTC_Read 関数で RTC の I/O レジスタを読み出して、情報構造体へ再設定しています。

バッテリバックアップ電源電圧の低下を検出した場合は、RTC の動作が保証されない状態となっていますので、RTC を初期設定します。

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 関数の戻り値確認用 */

    /* 引数を判定 */
    if (*(rtc_cb_evt_t *)event == RTC_EVT_PERIODIC)
    {
        /* 周期割り込みの場合 */
        /* RTC 時計カウンタの I/O レジスタから現在時刻情報を読み出し情報構造体に設定 */
        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)

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。）

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。

対応しているテクニカルアップデートはありません。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Jun.30.15	—	初版発行
1.01	Aug.31.15	ﾌﾟﾛｸﾞﾗﾑ	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モジュール Rev.1.01 以降をご使用ください。
1.02	Feb.01.19	ﾌﾟﾛｸﾞﾗﾑ	機能関連 Smart Configurator での GUI によるコンフィグオプション設定機能に対応 ■内容 GUI によるコンフィグ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
		22	「R_VBATT_GetVersion()」章を更新
		23	「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	—	対象製品に RX23W を追加しました。
		23	「4.1 動作確認環境」に Rev1.04 の評価環境を追加しました。
		25	「4.3 サンプルコード」の内容を更新しました。
1.05	Jun.10.20	—	API 関数のコメントを Doxygen スタイルに変更
		1	関連ドキュメントの、R01AN1833 を削除
		14	2.12「FIT モジュールの追加方法」を更新
		16～21	API 説明ページの、「Reentrant」項目を削除

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。お客様の機器・システムの設計において、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には、お客様の責任において行ってください。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、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、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。
5. 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。
6. 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。
7. 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。
8. 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。
9. 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。
10. 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。
11. 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。
12. 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.4.0-1　2017.11)

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。