RX ファミリ

WDT 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 WDT モジュールについて説明します。本モジュールは WDT を使用して、WDT 周辺機能のカウント操作の制御を行います。以降、本モジュールを WDT FIT モジュールと称します。

対象デバイス

- RX230、RX231 グループ
- RX23W グループ
- RX64M グループ
- RX65N、RX651 グループ
- RX66T グループ
- RX66N グループ
- RX671 グループ
- RX71M グループ
- RX72T グループ
- RX72M グループ
- RX72N グループ

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

ターゲットコンパイラ

- ルネサスエレクトロニクス製 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環境に関する詳細な内容は、セクション「6.1 動作確認環境」を参照してください。

目次

1.　概要

1.1　WDT 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、「2.13 FIT モジュールの追加方法」を参照してください。

1.2　WDT FIT モジュールの概要

この WDT FIT モジュールは、WDT の周辺機能をサポートしています。WDT の詳細は、ユーザーズマニュアル　ハードウェア編で詳述されています。

このドライバは、オートスタートとレジスタスタート両方のモードをサポートしています。コンパイル時にてオートスタートモードを選択することにより R_WDT_Open()関数のコードは、ビルドから除去されます。オートスタートモードでは、リセット後に自動的に WDT のダウンカウントが開始されます。レジスタスタートモードでは、R_WDT_Open()関数を呼び出して R_WDT_Control()関数のリフレッシュ動作後に WDT のダウンカウントが開始されます。

R_WDT_Control()関数のリフレッシュコマンドは、WDT カウンタを更新するために定期的に行わなければなりません。この関数の呼び出しが行われていない場合は、WDT カウンタがアンダフローし、リセットまたはノンマスカブル割り込み（NMI）が発生します。

ノンマスカブル割り込みを選択している場合は、この割り込み処理をユーザ側で設定する必要があります。オートスタートモードを使用する場合、アプリケーションでは、割り込みコントローラ（ICU）でアンダフロー/リフレッシュエラー割り込みを有効にする必要があります。

これは、レジスタスタートモードでは R_WDT_Open()関数によって設定されます。

1.3　WDT FIT モジュールを使用する

1.3.1　WDT 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 WDT 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_wdt_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 関数一覧

関数	関数説明
R_WDT_Open()	この関数は WDT FIT モジュールを初期化する関数です。 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。ただし r_bsp_config.h の OFS0 レジスタの設定で WDT のオートスタートモードが有効にされている場合は使用しません。
R_WDT_Control()	コマンドを引数に指定することにより、WDT のステータス（アンダフローフラグ、リフレッシュエラーフラグ、WDT カウンタ値）を取得する。また、WDT のダウンカウンタのリフレッシュを行います。
R_WDT_GetVersion()	本モジュールのバージョン番号を返します。

1.5 制限事項

WDT FIT モジュールは、マスカブル割り込みには対応していません。

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。

- WDTA

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) v5.20 以上

2.3 制限事項

2.3.1 RAM の配置に関する制限事項

FIT では、API 関数のポインタ引数に NULL と同じ値を設定すると、パラメータチェックにより戻り値がエラーとなる場合があります。そのため、API 関数に渡すポインタ引数の値は NULL と同じ値にしないでください。

ライブラリ関数の仕様で NULL の値は 0 と定義されています。そのため、API 関数のポインタ引数に渡す変数や関数が RAM の先頭番地(0x0 番地)に配置されていると上記現象が発生します。この場合、セクションの設定変更をするか、API 関数のポインタ引数に渡す変数や関数が 0x0 番地に配置されないように RAM の先頭にダミーの変数を用意してください。

なお、CCRX プロジェクト(e2 studio V7.5.0)の場合、変数が 0x0 番地に配置されることを防ぐために RAM の先頭番地が 0x4 になっています。GCC プロジェクト(e2 studio V7.5.0)、IAR プロジェクト(EWRX V4.12.1)の場合は RAM の先頭番地が 0x0 になっていますので、上記対策が必要となります。

IDE のバージョンアップによりセクションのデフォルト設定が変更されることがあります。最新の IDE を使用される際は、セクション設定をご確認の上、ご対応ください。

2.4 サポートされているツールチェーン

本 FIT モジュールは「6.1 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。

2.5 使用する割り込みベクタ

引数を指定して R_WDT_Open 関数を実行すると、ノンマスカブル割り込み（NMI）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 RX64M RX65N RX66T RX66N RX671 RX71M RX72T RX72M RX72N	ノンマスカブル割り込み

2.6 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_wdt_rx_if.h に記載しています。

2.7 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。

2.8 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_wdt_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。

内のコンフィギュレーションオプション r_wdt_rx_config.h	
WDT_CFG_PARAM_CHECKING_ENABLE 1 ※デフォルト値は “1”	1：ビルド時にパラメータチェック処理をコードに含めます。 0：ビルド時にパラメータチェック処理をコードから省略します。 BSP_CFG_PARAM_CHECKING_ENABLE（デフォルト）：システムのデフォルト設定を使用します。 注: ビルド時にパラメータチェックのコードを省略することで、コードサイズを小さくすることができます。

内のコンフィギュレーションオプション r_bsp_config.h	
BSP_CFG_OFS0_REG_VALUE 0xFFFFFFFF ※デフォルト値は “1”	この定義が 0xFFFFFFFF に設定されている場合、WDT は電源投入時に無効にされており、R_WDT_Open 関数を使用して初期化する必要があります。WDT のオートスタートを有効に設定した場合は、R_WDT_Open 関数のコードは、ビルドから削除され、リセット後に自動でカウントが開始されます。設定オプションについては、r_bsp_config.h を参照してください

2.9　コードサイズ

本モジュールのコード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「2.8 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。掲載した値は、「2.4 サポートされているツールチェーン」の C コンパイラでコンパイルオプションがデフォルト時の参考値です。コンパイルオプションのデフォルトは最適化レベル：2、最適化のタイプ：サイズ優先、データ・エンディアン：リトルエンディアンです。コードサイズは C コンパイラのバージョンやコンパイルオプションにより異なります。

ROM、RAM およびスタックのコードサイズ

デバイス	分類	使用メモリ		備考
		ルネサス製コンパイラ		
		パラメータチェック処理あり	パラメータチェック処理なし	
RX230	ROM	316 バイト	177 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	already_opened のみ
	最大使用スタックサイズ	28 バイト		R_WDT_Control 関数使用時
RX231	ROM	316 バイト	177 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	already_opened のみ
	最大使用スタックサイズ	28 バイト		R_WDT_Control 関数使用時
RX23W	ROM	316 バイト	178 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	already_opened のみ
	最大のスタック使用量	28 バイト		R_WDT_Control 関数使用時
RX64M	ROM	316 バイト	177 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	already_opened のみ
	最大使用スタックサイズ	28 バイト		R_WDT_Control 関数使用時
RX65N	ROM	316 バイト	177 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	already_opened のみ
	最大使用スタックサイズ	28 バイト		R_WDT_Control 関数使用時
RX66T	ROM	316 バイト	177 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	既に開いている場合のみ
	最大のスタック使用量	28 バイト		R_WDT_Control 関数使用時
RX66N	ROM	316 バイト	178 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	already_opened のみ
	最大使用スタックサイズ	28 バイト		R_WDT_Control 関数使用時

ROM、RAM およびスタックのコードサイズ				
デバイス	分類	使用メモリ		備考
		ルネサス製コンパイラ		
		パラメータチェック処理あり	パラメータチェック処理なし	
RX72T	ROM	316 バイト	177 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	already_opened のみ
	最大使用スタックサイズ	28 バイト		R_WDT_Control 関数使用時
RX71M	ROM	317 バイト	177 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	already_opened のみ
	最大使用スタックサイズ	28 バイト		R_WDT_Control 関数使用時
RX72M	ROM	316 バイト	178 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	already_opened のみ
	最大使用スタックサイズ	28 バイト		R_WDT_Control 関数使用時
RX72N	ROM	316 バイト	178 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	already_opened のみ
	最大使用スタックサイズ	28 バイト		R_WDT_Control 関数使用時
RX671	ROM	307 バイト	175 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	already_opened のみ
	最大使用スタックサイズ	12 バイト		R_WDT_Control 関数使用時

ROM、RAM およびスタックのコードサイズ				
デバイス	カテゴリ	使用メモリ		備考
		GCC		
		パラメータチェック処理あり	パラメータチェック処理なし	
RX230	ROM	600 バイト	328 バイト	レジスタスタートモード
	RAM	4 バイト	4 バイト	既に開いている場合のみ
	最大のスタック使用量	-		R_WDT_Control 関数を使用する場合
RX231	ROM	600 バイト	328 バイト	レジスタスタートモード
	RAM	4 バイト	4 バイト	既に開いている場合のみ
	最大のスタック使用量	-		R_WDT_Control 関数を使用する場合
RX64M	ROM	600 バイト	328 バイト	レジスタスタートモード
	RAM	4 バイト	4 バイト	既に開いている場合のみ
	最大のスタック使用量	-		R_WDT_Control 関数を使用する場合
RX65N	ROM	600 バイト	328 バイト	レジスタスタートモード
	RAM	4 バイト	4 バイト	既に開いている場合のみ
	最大のスタック使用量	-		R_WDT_Control 関数を使用する場合
RX66T	ROM	600 バイト	328 バイト	レジスタスタートモード
	RAM	4 バイト	4 バイト	既に開いている場合のみ
	最大のスタック使用量	-		R_WDT_Control 関数を使用する場合
RX66N	ROM	632 バイト	336 バイト	レジスタスタートモード
	RAM	4 バイト	4 バイト	既に開いている場合のみ
	最大のスタック使用量	-		R_WDT_Control 関数を使用する場合
RX71M	ROM	600 バイト	328 バイト	レジスタスタートモード
	RAM	4 バイト	4 バイト	既に開いている場合のみ
	最大のスタック使用量	-		R_WDT_Control 関数を使用する場合
RX72T	ROM	600 バイト	328 バイト	レジスタスタートモード
	RAM	4 バイト	4 バイト	既に開いている場合のみ
	最大のスタック使用量	-		R_WDT_Control 関数を使用する場合

ROM、RAM およびスタックのコードサイズ				
デバイス	カテゴリ	使用メモリ		備考
		GCC		
		パラメータチェック処理あり	パラメータチェック処理なし	
RX72M	ROM	632 バイト	336 バイト	レジスタスタートモード
	RAM	4 バイト	4 バイト	既に開いている場合のみ
	最大のスタック使用量	-		R_WDT_Control 関数を使用する場合
RX72N	ROM	632 バイト	336 バイト	レジスタスタートモード
	RAM	4 バイト	4 バイト	既に開いている場合のみ
	最大のスタック使用量	-		R_WDT_Control 関数を使用する場合
RX671	ROM	648 バイト	352 バイト	レジスタスタートモード
	RAM	4 バイト	4 バイト	既に開いている場合のみ
	最大のスタック使用量	-		R_WDT_Control 関数を使用する場合

ROM、RAM およびスタックのコードサイズ				
デバイス	カテゴリ	使用メモリ		備考
		IAR コンパイラ		
		パラメータチェック処理あり	パラメータチェック処理なし	
RX230	ROM	496 バイト	292 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	既に開いている場合のみ
	最大のスタック使用量	156 バイト		R_WDT_Control 関数を使用する場合
RX231	ROM	496 バイト	292 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	既に開いている場合のみ
	最大のスタック使用量	156 バイト		R_WDT_Control 関数を使用する場合
RX64M	ROM	496 バイト	292 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	既に開いている場合のみ
	最大のスタック使用量	156 バイト		R_WDT_Control 関数を使用する場合
RX65N	ROM	496 バイト	292 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	既に開いている場合のみ
	最大のスタック使用量	156 バイト		R_WDT_Control 関数を使用する場合
RX66T	ROM	496 バイト	292 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	既に開いている場合のみ
	最大のスタック使用量	156 バイト		R_WDT_Control 関数を使用する場合
RX66N	ROM	496 バイト	292 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	既に開いている場合のみ
	最大のスタック使用量	160 バイト		R_WDT_Control 関数を使用する場合
RX71M	ROM	496 バイト	292 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	既に開いている場合のみ
	最大のスタック使用量	156 バイト		R_WDT_Control 関数を使用する場合
RX72T	ROM	496 バイト	292 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	既に開いている場合のみ
	最大のスタック使用量	156 バイト		R_WDT_Control 関数を使用する場合

ROM、RAM およびスタックのコードサイズ				
デバイス	カテゴリ	使用メモリ		備考
		IAR コンパイラ		
		パラメータチェック処理あり	パラメータチェック処理なし	
RX72M	ROM	496 バイト	292 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	既に開いている場合のみ
	最大のスタック使用量	160 バイト		R_WDT_Control 関数を使用する場合
RX72N	ROM	496 バイト	292 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	既に開いている場合のみ
	最大のスタック使用量	160 バイト		R_WDT_Control 関数を使用する場合
RX671	ROM	465 バイト	245 バイト	レジスタスタートモード
	RAM	1 バイト	1 バイト	既に開いている場合のみ
	最大のスタック使用量	172 バイト		R_WDT_Control 関数を使用する場合

2.10 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_wdt_rx_if.h に記載されています。

2.11 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_wdt_rx_if.h で記載されています。

```
typedef enum e_wdt_err      // WDT API error codes
{
    WDT_SUCCESS=0,
    WDT_ERR_OPEN_IGNORED,   // The module has already been opened.
    WDT_ERR_INVALID_ARG,    // Argument is not valid.
    WDT_ERR_NULL_PTR,       // Received null pointer.
    WDT_ERR_NOT_OPENED,     // Open function has not been called.
    WDT_ERR_BUSY,           // WDT resource is locked.
} wdt_err_t;
```

2.12 コールバック関数

なし

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

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

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

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

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

2.14 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_WDT_Open()

この関数は WDT FIT モジュールを初期化する関数です。 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。ただし r_bsp_config.h の OFS0 レジスタの設定で WDT のオートスタートモードが有効にされている場合は使用しません。

Format

```
wdt_err_t           R_WDT_Open (
void * const        p_cfg
)
```

Parameters

*void *p_cfg*

設定構造体へのポインタ *wdt_ config_t* (下記参照)

レジスタスタートモードに対する設定可能なオプションは下記です。

構造体は、void 型ポインタにキャストして Open()関数を呼び出します。

```
typedef enum e_wdt_timeout          // WDT Time-Out Period
{
    WDT_TIMEOUT_1024 =0x0000u,      // 1024 (cycles)
    WDT_TIMEOUT_4096 =0x0001u,      // 4096 (cycles)
    WDT_TIMEOUT_8192 =0x0002u,      // 8192 (cycles)
    WDT_TIMEOUT_16384=0x0003u,      // 16,384 (cycles)
WDT_NUM_TIMEOUTS
} wdt_timeout_t;

typedef enum e_wdt_clock_div        // WDT Clock Division Ratio
{
    WDT_CLOCK_DIV_4  =0x0010u,      // PCLK/4
    WDT_CLOCK_DIV_64 =0x0040u,      // PCLK/64
    WDT_CLOCK_DIV_128 =0x00F0u,     // PCLK/128
    WDT_CLOCK_DIV_512 =0x0060u,     // PCLK/512
    WDT_CLOCK_DIV_2048=0x0070u,     // PCLK/2048
    WDT_CLOCK_DIV_8192=0x0080u      // PCLK/8192
} wdt_clock_div_t;

typedef enum e_wdt_window_end       // Window End Position
{
    WDT_WINDOW_END_75=0x0000u,      // 75%
    WDT_WINDOW_END_50=0x0100u,      // 50%
    WDT_WINDOW_END_25=0x0200u,      // 25%
    WDT_WINDOW_END_0 =0x0300u       // 0%  (window end position is not
specified)
} wdt_window_end_t;

typedef enum e_wdt_window_start     // Window Start Position
{
    WDT_WINDOW_START_25 =0x0000u,   // 25%
    WDT_WINDOW_START_50 =0x1000u,   // 50%
    WDT_WINDOW_START_75 =0x2000u,   // 75%
    WDT_WINDOW_START_100=0x3000u    // 100%(window start position is not
specified)
} wdt_window_start_t;
```

```
typedef enum e_wdt_timeout_control  // Signal control when Time-out and
Refresh error
{
    WDT_TIMEOUT_NMI  =0x00u         // NMI request output is enabled
    WDT_TIMEOUT_RESET=0x80u,        // Reset output is enabled
} wdt_timeout_control_t;



typedef struct st_wdt_config
{
    wdt_timeout_t          timeout;           /* Timeout period */
    wdt_clock_div_t        wdtcks_div;        /* WDT clock division ratio */
    wdt_window_start_t     window_start;      /* Window start position */
    wdt_window_end_t       window_end;        /* Window end position */
    wdt_timeout_control_t  timeout_control;   /* Reset or NMI at timeout */
} wdt_config_t;
```

Return Values

[WDT_SUCCESS]	*/* WDT 初期化の成功*	*/*
[WDT_ERR_OPEN_IGNORED]	*/* エラー：モジュールは既にオープンされています*	*/*
[WDT_ERR_INVALID_ARG]	*/* エラー：パラメータに対して無効な引数です。*	*/*
[WDT_ERR_NULL_PTR]	*/* エラー：要求された引数がありません。*	*/*
[WDT_ERR_BUSY]	*/* エラー：WDT リソースがロックされています*	*/*

Properties

r_wdt_rx_if.h にプロトタイプ宣言されています。

Description

本関数は、WDT に関連するレジスタを初期化して、MCU ごとで選択可能なオプションを設定します。

Example

```
    wdt_config_t  config;
    wdt_err_t     err;

    /*
     * Configure the WDT for:
     * - A 2.6 ms timer period:
     *      PCLKB is 50MHz clock = 20 ns/tick; So 20 ns/tick / 128 * 1024 =
2.6 ms
     * - A 100% refresh-permitted window (start 100% end 0%)
     * - Reset on counter underflow
     * - Count stop disabled
     */
    config.timeout = WDT_TIMEOUT_1024;
    config.wdtcks_div = WDT_CLOCK_DIV_128;
    config.window_start = WDT_WINDOW_START_100;
    config.window_end = WDT_WINDOW_END_0;
    config.timeout_control = WDT_TIMEOUT_RESET;

    err = R_WDT_Open(&config);
```

Special Notes:

本関数は、レジスタスタートモードでのみ使用可能です。(r_bsp_config.h の BSP_CFG_OFS0_REG_VALUE が” 0xFFFFFFFF”のとき)

呼び出しても WDT カウントは開始しません。WDT カウンタを開始するには、R_WDT_Control 関数の引数に WDT_CMD_REFRESH_COUNTING を指定して実行する必要があります。

R_WDT_Open()関数は、リセット後に一度だけ呼び出してください。

2 回目以降の呼び出しは WDT_ERR_OPEN_IGNORED が返されます。

オートスタートモードで有効かつ、ノンマスカブル割り込みを選択している場合は、下記のように割り込みコントローラ（ICU）でアンダフロー/リフレッシュエラー割り込みを有効にしてください。

```
ICU.NMIER.BIT.WDTEN = 1;   // Enable WDT underflow/refresh error interrupt
```

オートスタートモードとレジスタスタートモードの両方でユーザアプリケーションにてこの割り込みを処理する関数が必要です。NMI ハンドラの実装例を次に示します。

```
void wdt_nmi_func(void *p_args)
{
    /* Do some processing here */

    while(WDT.WDTSR.BIT.REFEF == 1)
    {
       WDT.WDTSR.BIT.REFEF = 0;  // clear Refresh Error Flag
    }
    while(WDT.WDTSR.BIT.UNDFF == 1)
    {
       WDT.WDTSR.BIT.UNDFF = 0;  // clear Underflow Flag
    }
}
```

レジスタスタートモードでは、下記の例で示すように、R_WDT_Open ()関数を呼び出した直後に、R_BSP_InterruptWrite 関数を呼び出してください。オートスタートモードでは WDT 割り込みを有効にした後に R_BSP_InterruptWrite 関数を呼び出してください。

例：

```
err = R_WDT_Open(&config);

/* Register wdt_nmi_func() to be called whenever WDT underflow occurs.*/
bsp_err = R_BSP_InterruptWrite(BSP_INT_SRC_WDT_ERROR, wdt_nmi_func);
```

R_WDT_Control()

この関数は、WDT ステータスを取得し、WDT カウンタのリフレッシュを実行します。この機能はオートスタートモードとレジスタスタートモードで使用します。

Format

```
wdt_err_t               R_WDT_Control (
                wdt_cmd_t const         cmd,
                uint16_t  *             p_status
)
```

Parameters

wdt_cmd_t cmd

実行コマンド (下記参照)

*uint16_t *p_status*

取得したカウンタとステータスフラグを格納する変数へのポインタ

有効な cmd 値を以下に示します。

```
WDT_CMD_GET_STATUS,          // Get WDT status
WDT_CMD_REFRESH_COUNTING,  // Refresh the counter
```

Return Values

```
[WDT_SUCCESS]               /* コマンドが正常に完了しました                          */
[WDT_ERR_INVALID_ARG]       /* エラー : パラメータに対して無効な引数です。           */
[WDT_ERR_NULL_PTR]          /* エラー : 要求された引数がありません。                 */
[WDT_ERR_NOT_OPENED]        /* エラー : Open 関数が呼び出されていません              */
[WDT_ERR_BUSY]              /* エラー : WDT リソースがロックされています。           */
```

Properties

r_wdt_rx_if.h にプロトタイプ宣言されています。

Description

コマンドに WDT_CMD_REFRESH_COUNTING を選択している場合は、WDT カウントをリフレッシュします。

コマンドに WDT_CMD_GET_STATUS を選択している場合は、WDT のステータス（アンダフローフラグ、リフレッシュエラーフラグ、WDT カウンタ値）を取得して、*p_status に格納されます。上位 2 ビットはリフレッシュエラー(b15)と、アンダフロー(b14)が発生したかを示します。残りのビット(b13-b0)は、ダウンカウンタのカウンタ値を示しています。

Example

```
wdt_config_t  config;
wdt_err_t     err;
uint16_t       status;
  :
err = R_WDT_Open(&config);                              /* Register-Start mode
*/
  :
err = R_WDT_Control(WDT_CMD_REFRESH_COUNTING, NULL); /* Start counting */
  :
err = R_WDT_Control(WDT_CMD_GET_STATUS, &status);    /* Get WDT status */
```

Special Notes:

コマンドに WDT_CMD_REFRESH_COUNTING を選択した場合は、第 2 引数は無視されます。

R_WDT_GetVersion()

この関数は実行時に本モジュールのバージョンを返します。

Format

uint32_t R_WDT_GetVersion (void)

Parameters

なし

Return Values

本モジュールのバージョン.

Properties

r_wdt_rx_if.h にプロトタイプ宣言されています。

Description

この関数は本モジュールのバージョン番号を返します。

Example

```
uint32_t version;
:
version = R_WDT_GetVersion();
```

Special Notes:

なし

4. 端子設定

WDT FIT モジュールはピン設定を使用しません。

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 wdt_demo_rskrx230、wdt_demo_rskrx231、wdt_demo_rskrx64m、wdt_demo_rskrx71m、wdt_demo_rskrx72m、wdt_demo_rskrx671、wdt_demo_rskrx230_gcc、wdt_demo_rskrx231_gcc、wdt_demo_rskrx64m_gcc、wdt_demo_rskrx71m_gcc、wdt_demo_rskrx72m_gcc、wdt_demo_rskrx671_gcc

デモの一部として、プログラムはユーザに対し、WDT（ウォッチドッグタイマ）を構成するオプション（Timeout Cycles（タイムアウトサイクル数）Clock Division Ratio（クロック分周比）、Window Start（ウィンドウ期間開始時期）、Window End（ウィンドウ期間終了時期）、Timeout Control（タイムアウト制御））の選択を求めます。WDT を構成した後、このタイマはカウントダウンを開始し、アンダーフローが発生するまで LED0 は点滅します。カウンタの値が、更新許可期間、または更新禁止期間のどちらかに属している場合、Renesas Debug Virtual Console（ルネサスデバッグ仮想コンソール）でユーザへの通知が出力されます。カウンタの値が更新許可期間に属しているときに SW1 が押下された場合、WDT は更新されます。それ以外に、カウンタの値が更新禁止期間に属しているときに SW1 が押下された場合、更新エラーが発生します。

さらに、WDT アンダーフローエラーまたは更新エラーが発生した後、RESET ボタンを押して、WDT レジスタの保護機能を解除する必要があります。その結果、ユーザは新しい WDT 構成を入力することができます。

セットアップと実行

1. サンプルコードをコンパイルし、ダウンロードします。
2. 「Reset Go」（リセットして開始）をクリックし、ソフトウェアを起動します。Main 内で PC の動作が停止した場合、F8 キーを押して動作を再開します。
3. Renesas Debug Virtual Console（ルネサスデバッグ仮想コンソール）のメッセージに従って、WDT 入力オプションを選択します。その結果、WDT を構成し、カウントを開始することができます。
   入力の例:

```
Please select timeout cycles: (タイムアウトのサイクル数を選択してください: )
[0] WDT_TIMEOUT_1024  (1024 サイクル)
[1] WDT_TIMEOUT_4096  (4096 サイクル)
[2] WDT_TIMEOUT_8192  (8192 サイクル)
[3] WDT_TIMEOUT_16384 (16,384 サイクル)
Your input: (入力:) 3
Please select Clock Division Ratio: (クロックの分周比を選択してください: )
[0] WDT_CLOCK_DIV_4     (PCLK/4)
[1] WDT_CLOCK_DIV_64    (PCLK/64)
[2] WDT_CLOCK_DIV_128   (PCLK/128)
[3] WDT_CLOCK_DIV_512   (PCLK/512)
[4] WDT_CLOCK_DIV_2048  (PCLK/2048)
[5] WDT_CLOCK_DIV_8192  (PCLK/8192)
Your input: (入力:) 5
Please select Window Start: (ウィンドウ期間開始時期を選択してください: )
[0] WDT_WINDOW_START_25   (25%)
[1] WDT_WINDOW_START_50   (50%)
[2] WDT_WINDOW_START_75   (75%)
[3] WDT_WINDOW_START_100  (100%)
Your input: (入力:) 3
Please select Window End: (ウィンドウ期間終了時期を選択してください: )
[0] WDT_WINDOW_END_75   (75%)
[1] WDT_WINDOW_END_50   (50%)
[2] WDT_WINDOW_END_25   (25%)
[3] WDT_WINDOW_END_0    (0%)
Your input: (入力:) 3
Please select Timeout Control: (タイムアウト制御を選択してください: )
[0] WDT_TIMEOUT_RESET   (リセット出力)
[1] WDT_TIMEOUT_NMI     (NMI 出力)
Your input: (入力:) 1
```

上記の入力例を使用した場合、WDT 構成は次のように設定されます。

- WDT Time-Out Period (WDT タイムアウト期間): 16384 サイクル
- WDT Clock Division Ratio (WDT クロック分周比): PCLK/8192
- Window Start Position (ウィンドウ期間開始位置): 100%
- Window End Position (ウィンドウ期間終了位置): 0%
- WDT Time-Out Control (WDT タイムアウト制御): 1 (NMI、ノンマスカブル割り込み要求の出力が有効)

サポート対象ボード

RSKRX230
RSKRX231
RSKRX64M
RSKRX71M
RSKRX72M
RSKRX671

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「完了」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. 付録

6.1 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 6.1 動作環境の確認（Rev.2.60）

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio V.2021-07 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.03.00 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 8.3.0.202004 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、FCC 周辺モジュールで宣言された割り込み関数をリンカーが誤って破棄する GCC リンカーの問題を回避するためです。
エンディアン	IAR C/C++ Compiler for Renesas RX version 4.20.3 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	Rev.2.60
使用ボード	Renesas Starter Kit+ for RX671（型名：RTK55671xxxxxxxxxx）

表 6.2 動作環境の確認（Rev.2.50）

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio V.2021-07 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.03.00 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 8.3.0.202004 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、FCC 周辺モジュールで宣言された割り込み関数をリンカーが誤って破棄する GCC リンカーの問題を回避するためです。
エンディアン	IAR C/C++ Compiler for Renesas RX version 4.20.3 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	Rev.2.50
使用ボード	Renesas Starter Kit+ for RX671（型名：RTK55671xxxxxxxxxx）

表 6.3 動作確認環境 (Rev.2.40)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V.7.8.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.02.00 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 8.3.0.201904 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。
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.2.40
使用ボード	Renesas Starter Kit+ for RX72M （型名：RTK5572Mxxxxxxxxx）

表 6.4 動作確認環境 (Rev.2.30)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V.7.7.0 IAR Embedded Workbench for Renesas RX 4.12.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package 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 4.8.4.201902 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 4.12.1 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	Rev.2.30
使用ボード	Renesas Starter Kit+ for RX72N（型名：RTK5572Nxxxxxxxxx）

表 6.5 動作確認環境 (Rev.2.20)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio V.7.5.0 IAR Embedded Workbench for Renesas RX 4.12.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package 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 4.8.4.201902 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 4.12.1 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	Rev.2.20
使用ボード	Renesas Starter Kit+ for RX72M（型名：RTK5572Mxxxxxxxxxx）

表 6.6 動作確認環境 (Rev.2.10)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio V.7.5.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package 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	Rev.2.10
使用ボード	Renesas Solution Starter Kit for RX23W（型名：RTK5523Wxxxxxxxxx）

表 6.7 動作確認環境 (Rev.2.00)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio V.7.4.0 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 Package 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 4.8.4.201803 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 4.10.1 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	Rev.2.00
使用ボード	Renesas Starter Kit+ for RX65N-2MB（型名：RTK50565Nxxxxxxxxx）

表 6.8 動作確認環境 (Rev.1.40)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio V.7.3.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package 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	Rev.1.40
使用ボード	Renesas Starter Kit for RX72T（型名：RTK5572Txxxxxxxxxx）

表 6.9 動作確認環境 (Rev.1.31)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio V.7.3.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package 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	Rev.1.31
使用ボード	Renesas Starter Kit for RX66T（型名：RTK50566T0SxxxxxBE）

表 6.10 動作確認環境 (Rev.1.30)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V.7.0.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.00.00
	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	Rev.1.30
使用ボード	Renesas Starter Kit for RX66T（型名：RTK50566T0SxxxxxBE）

表 6.11 動作確認環境 (Rev.1.20)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V6.0.0
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 V2.07.00
	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	Rev1.20
使用ボード	Renesas Starter Kit+ for RX231（型名：R0K505231SxxxBE） Renesas Starter Kit+ for RX64M（型名：R0K50564MSxxxBE） Renesas Starter Kit+ for RX71M（型名：R0K50571MSxxxBE）

表 6.12 動作確認環境 (Rev.1.10)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V6.0.0
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 V2.07.00
	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	Rev1.10
使用ボード	Renesas Starter Kit+ for RX65N-2MB（型名：RTK50565N2SxxxxxBE）

表 6.13 動作確認環境 (Rev.1.00)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V5.0.1.005
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 V2.05.00
	コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
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	Rev1.00
使用ボード	Renesas Starter Kit+ for RX65N（型名：RTK500565NSxxxxxBE）

6.2 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_wdt_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「コンフィグ設定が間違っている場合のエラーメッセージ」エラーが発生します。

A：“r_wdt_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_wdt_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。

7. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

RX230 グループ、RX231 グループ　ユーザーズマニュアル　ハードウェア編(R01UH0496)
RX64M グループ　ユーザーズマニュアル　ハードウェア編(R01UH0377)
RX65N グループ、RX651 グループ　ユーザーズマニュアル　ハードウェア編(R01UH0590)
RX71M グループ　ユーザーズマニュアル　ハードウェア編(R01UH0493)
(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。

- なし

改訂記録

Rev.	発行日	改訂内容	
		ページ	Summary
1.00	2016.10.1	—	初版発行
1.10	2017.10.1	—	RX65N-2MB に対応
		3	「1.1 WDT FIT モジュールとは」の追加
			「1.2 API の概要」の追加
			「1.3 制限事項」の追加
		4	「2.2 ハードウェアリソースの要求」の削除
			「2.3 ソフトウェアの要求」
			章番号の変更
			文章の変更
			「2.4 制限事項」の削除
			「2.5 サポートされているツールチェーン」
			章番号の変更
			章タイトルの変更
			文章の変更
			「2.6 ヘッダファイル」
			章番号の変更
			文章の変更
			「2.7 整数型」
			章番号の変更
			文章の変更
			「2.4 使用する割り込み/例外ベクタ」の追加
		5	「2.8 コンパイル時の設定」
			文章の変更
			表タイトルの変更
		6	「2.9 コードサイズ」
			章番号の変更
			文章の変更
			「2.9 引数」の追加
			「2.10 戻り値」の追加
		7	「2.10 API データ構造体」の削除
			「2.11 FIT モジュールの追加方法」
			文章の変更
		8	「3.1 概要」の削除
			「3.2 戻り値」の削除
			「3.3 R_WDT_Open()」
			章番号の変更
			内容の修正
		11	「3.4 R_WDT_Control()」
			章番号の変更
		12	「3.5 R_WDT_GetVersion()」
			章番号の変更
		13	「4. 付録」の追加
		15	「5. 参考ドキュメント」の追加
		プログラム	RX65N-2MB 対応のため、R_WDT_GetVersion 関数のバージョン番号を変更

1.20	2017.10.31	1 4 6 14 16 17 18 プログラム	RX230、RX231、RX64M、RX71M のサポートに関する説明を追加 「2.4 使用する割り込み/例外ベクタ」内容を変更。 「2.8 コードサイズ」: ROM サイズを変更。 「4. デモプロジェクト」を追加 「5.1 動作確認環境」: Rev.1.20 に対応する表を追加 「5.2 トラブルシューティング」: 質問を 2 つ追加 「6 参考ドキュメント」: 内容を更新。 グローバル変数名を変更
1.30	2018.09.28	1、4 6 18	RX66T のサポートを追加。 RX66T に対応するコードサイズを追加。 「6.1 動作確認環境」：Rev.1.30 に対応する表を追加。
1.31	2018.11.16	— 1 18	XML 内にドキュメント番号を追加。 RX651 のサポートを追加。 Renesas Starter Kit+ for RX66T の型名を変更。 Rev.1.31 に対応する表を追加。
1.40	2019.02.01	— 1、4、7 10-15 19	RX72T グループのサポートを追加。 RX72T グループのサポートを追加。 各 API 関数で「Reentrant」の説明を削除。 「6.1 動作確認環境」Rev.1.40 に対応する表を追加。
2.00	2019.05.20	— 1 4 6-8 19 23 プログラム	以下のコンパイラをサポート。 - GCC for Renesas RX - IAR C/C++ Compiler for Renesas RX 「ターゲットコンパイラ」のセクションを追加。 関連ドキュメントを削除。 「2.2 ソフトウェアの要求」r_bsp v5.20 以上が必要 「2.8 コードサイズ」セクションを更新。 表 6.1「動作確認環境」： Rev.2.00 に対応する表を追加。 「Web サイトおよびサポート」のセクションを削除。 GCC コンパイラと IAR コンパイラに関して、以下を変更。 1. R_WDT_GetVersion 関数のインライン展開を削除。 2. NOP を BSP の固有関数で置き換えた。
2.10	2019.06.28	1、4 7 20 プログラム	RX23W のサポートを追加。 RX23W に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.2.10 に対応する表を追加。 RX23W のサポートを追加。
2.20	2019.08.15	1、4 7-9 20 プログラム	RX72M のサポートを追加。 RX72M に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.2.20 に対応する表を追加。 表 6.2：RX23W ボード名変更。 RX72M のサポートを追加。
2.30	2019.12.30	1、5 4 8-13 24	RX66N, RX72N のサポートを追加。 2.3 制限事項 制限事項を追加。 RX66N, RX72N に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.2.30 に対応する表を追加。

		プログラム	RX66N, RX72N のサポートを追加。

2.40	2020.6.30	22,23 24 プログラム	「5. デモプロジェクト」に RSKRX72M を追加。 「6.1 動作確認環境」： Rev.2.30 に対応する表を追加。 デモプロジェクトの更新と追加
2.50	2021.03.31	1 3 9,11,13 25 プログラム	RX671 のサポートを追加。 「1.3 WDT FIT モジュールを使用する」のセクションを追加。 「1.3.1 WDT FIT モジュールを C++プロジェクト内で使用する」のセクションを追加。 RX671 に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.2.50 に対応する表を追加。 RX671 のサポートを追加。
2.60	2021.09.13	24,25 27 プログラム	「5. デモプロジェクト」に RSKRX671 を追加。 「6.1 動作確認環境」： Rev.2.60 に対応する表を追加。 デモプロジェクトの更新と追加。

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。