RX ファミリ

IWDT 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)を使用した独立ウォッチドッグタイマ（IWDT）モジュールについて説明します。本モジュールは、IWDT を使用して IWDT 周辺ドライバの計数動作を制御します。以降、本モジュールを IWDT FIT モジュールと称します。

対象デバイス

- RX110、RX111、RX113、RX130 グループ
- RX13T グループ
- RX230、RX231、RX23W、RX23T、RX24T、RX24U グループ
- RX64M グループ
- RX651、RX65N グループ
- RX66T グループ
- RX66N 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 IWDT 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 IWDT FIT モジュールの概要

この IWDT ドライバは、RX110、RX111、RX113、RX130、RX13T、RX230、RX231、RX23T、RX23W, RX24T、RX24U、RX64M、RX651、RX65N、RX66T、RX66N、RX71M、 RX72T、RX72M および RX72N で IWDT 周辺装置をサポートします。

このドライバはオートスタートとレジスタスタートの両モードをサポートしています。コンパイル時の設定オプションを用いてオートスタートモードを選択することにより、ビルドから R_IWDT_Open()コードを取り除いています。オートスタートモードでは、リセット後に自動的に IWDT のダウンカウンタが開始されます。レジスタスタートモードでは、R_IWDT_Open()を呼び出して R_IWDT_Control()のリフレッシュ動作後に IWDT のダウンカウンタが開始されます。

R_IWDT_Control()のリフレッシュコマンドは、IWDT カウンタを更新するために定期的に行う必要があります。呼び出しが行われない場合、IWDT カウンタでアンダフローが発生し、リセット信号またはノンマスカブル割り込み（NMI）信号が出力されます。

NMI 信号を選択した場合、割り込みハンドラを作成し、この割り込みを処理するように登録する必要があります。オートスタートモードを使用するときには、アプリケーションは、割り込みコントローラユニット（ICU）でアンダフロー/リフレッシュエラー割り込みも有効にする必要があります。レジスタスタートモードでは、これは R_IWDT_Open()関数によって処理されます。

IWDT モジュールは、周辺クロック B（PCLKB）と IWDT 専用クロック（IWDTCLK）の両方を使用して動作します。PCLKB は、IWDTCLK よりも少なくとも 4 倍の速さで動作させる必要があります。どちらのクロックもこのモジュールを呼び出す前に動作させて安定させておく必要があります。

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 1.1API 関数一覧

関数	関数説明
R_IWDT_Open()	IWDT の関連レジスタを初期化し、IWDT カウンタのオプションを設定します。この Open 関数は、IWDT が r_bsp_config.h の OFS0 レジスタの設定で、オートスタートモードに設定されている場合には使用できません。
R_IWDT_Control()	IWDT ステータス（アンダフローステータス、リフレッシュエラーステータス、IWDT カウンタ値）を取得し、IWDT のダウンカウンタをリフレッシュします。
R_IWDT_GetVersion()	実行時にドライバのバージョン番号を返します。

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。

- IWDT

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、NULL に相当する値が API 関数のポインター引数として設定されている場合、パラメーターチェックのためにエラーが返されることがあります。 したがって、API 関数のポインター引数として NULL 相当値を渡さないでください。

ライブラリ関数の仕様により、NULL 値は 0 として定義されています。したがって、API 関数ポインター引数に渡された変数または関数が RAM の開始アドレス（アドレス 0x0）にある場合、上記の現象が発生します。この場合、API 関数ポインター引数に渡される変数または関数がアドレス 0x0 にないように、セクション設定を変更するか、RAM の上部にダミー変数を準備します。

CCRX プロジェクト（e2 studio V7.5.0）の場合、RAM 開始アドレスは 0x4 に設定され、変数がアドレス 0x0 に配置されないようにします。 GCC プロジェクト（e2 studio V7.5.0）および IAR プロジェクト（EWRX V4.12.1）の場合、RAM の開始アドレスは 0x0 であるため、上記の対策が必要です。

IDE バージョンのアップグレードにより、セクションのデフォルト設定が変更される場合があります。 最新の IDE を使用する場合は、セクションの設定を確認してください。

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

IWDT 割り込みは、R_IWDT_Open 関数を実行すると有効化されます。

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タ

デバイス	割り込みベクタ
RX110 RX111 RX113 RX130 RX13T RX210 RX231 RX23W RX23T RX24T RX24U RX63N RX631	ノンマスカブル割り込み（WUNI）
RX64M RX651 RX65N RX66T RX66N RX71M RX72T RX72M RX72N	ノンマスカブル割り込み IWUNI*1 割り込み（ベクタ番号: 95）

注 1：対応するノンマスカブル割り込み要求許可ビットが゛0゛（無効）になっている場合です。

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_ iwdt_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_iwdt_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。

コンフィギュレーションオプション（*r_iwdt_rx_config.h*）	
IWDT_CFG_PARAM_CHECKING_ENABLE 1	0 に設定した場合、パラメータチェックの処理はビルドから除外されます。1 に設定した場合、パラメータチェックの処理はビルドに含まれます。BSP_CFG_PARAM_CHECKING_ENABLE を使用するとシステムのデフォルトに設定されます。
BSP_CFG_OFS0_REG_VALUE 0xFFFFFFFF	この定義が 0xFFFFFFFF に設定されている場合、電源投入時のウォッチドッグタイマは停止状態になり、R_IWDT_Open()関数を使用して初期化する必要があります。電源投入時のウォッチドッグタイマをオートスタートモードに設定すると、R_IWDT_Open()関数の処理がビルドから除外され、ウォッチドッグタイマは電源投入時に自動的に開始されます。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制限事項」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
		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	
RX110	ROM	325 バイト	179 バイト	レジスタスタートモード時
	RAM	1 バイト	1 バイト	レジスタスタートモード時
	最大スタックサイズ	28 バイト		R_IWDT_Control 関数使用時
RX13T	ROM	343 バイト	180 バイト	レジスタスタートモード時
	RAM	1 バイト	1 バイト	レジスタスタートモード時
	最大スタックサイズ	28 バイト		R_IWDT_Control 関数使用時
RX231	ROM	325 バイト	179 バイト	レジスタスタートモード時
	RAM	1 バイト	1 バイト	レジスタスタートモード時
	最大スタックサイズ	28 バイト		R_IWDT_Control 関数使用時
RX23W	ROM	343 バイト	180 バイト	レジスタスタートモード時
	RAM	1 バイト	1 バイト	レジスタスタートモード時
	最大のスタック使用量	28 バイト		R_IWDT_Control 関数使用時
RX65N	ROM	327 バイト	179 バイト	レジスタスタートモード時
	RAM	1 バイト	1 バイト	レジスタスタートモード時
	最大スタックサイズ	28 バイト		R_IWDT_Control 関数使用時
RX66T	ROM	342 バイト	179 バイト	レジスタスタートモード時
	RAM	1 バイト	1 バイト	レジスタスタートモード時
	最大スタックサイズ	28 バイト		R_IWDT_Control 関数使用時
RX66N	ROM	343 バイト	167 バイト	レジスタスタートモード時
	RAM	1 バイト	1 バイト	レジスタスタートモード時
	最大スタックサイズ	12 バイト		R_IWDT_Control 関数使用時
RX72T	ROM	343 バイト	180 バイト	レジスタスタートモード時
	RAM	1 バイト	1 バイト	レジスタスタートモード時
	最大スタックサイズ	28 バイト		R_IWDT_Control 関数使用時
RX72M	ROM	343 バイト	180 バイト	レジスタスタートモード時
	RAM	1 バイト	1 バイト	レジスタスタートモード時
	最大スタックサイズ	28 バイト		R_IWDT_Control 関数使用時
RX72N	ROM	343 バイト	180 バイト	レジスタスタートモード時
	RAM	1 バイト	1 バイト	レジスタスタートモード時
	最大スタックサイズ	12 バイト		R_IWDT_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	GCC		備考
		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	
RX110	ROM	640 バイト	344 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX13T	ROM	680 バイト	352 バイト	
	RAM	4 バイト	4 バイト	
	最大のスタック使用量	-		
RX231	ROM	640 バイト	344 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX65N	ROM	640 バイト	344 バイト	
	RAM	4 バイト	4 バイト	
	最大のスタック使用量	-		
RX66T	ROM	640 バイト	344 バイト	
	RAM	4 バイト	4 バイト	
	最大のスタック使用量	-		
RX66N	ROM	688 バイト	351 バイト	
	RAM	4 バイト	4 バイト	
	最大のスタック使用量	-		
RX72T	ROM	640 バイト	344 バイト	
	RAM	4 バイト	4 バイト	
	最大のスタック使用量	-		
RX72M	ROM	688 バイト	352 バイト	
	RAM	4 バイト	4 バイト	
	最大のスタック使用量	-		
RX72N	ROM	688 バイト	351 バイト	
	RAM	4 バイト	4 バイト	
	最大のスタック使用量	-		

ROM、RAM およびスタックのコードサイズ				
デバイス	カテゴリ	IAR コンパイラ		備考
		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	
RX110	ROM	568 バイト	336 バイト	
	RAM	6 バイト	6 バイト	
	最大のスタック使用量	140 バイト		
RX13T	ROM	532 バイト	300 バイト	
	RAM	1 バイト	1 バイト	
	最大のスタック使用量	28 バイト		
RX231	ROM	568 バイト	336 バイト	
	RAM	6 バイト	6 バイト	
	最大のスタック使用量	144 バイト		
RX65N	ROM	591 バイト	336 バイト	
	RAM	5 バイト	5 バイト	
	最大のスタック使用量	148 バイト		
RX66T	ROM	568 バイト	336 バイト	
	RAM	5 バイト	5 バイト	
	最大のスタック使用量	148 バイト		
RX66N	ROM	489 バイト	253 バイト	
	RAM	1 バイト	1 バイト	
	最大のスタック使用量	28 バイト		
RX72T	ROM	568 バイト	336 バイト	
	RAM	5 バイト	5 バイト	
	最大のスタック使用量	148 バイト		
RX72M	ROM	532 バイト	336 バイト	
	RAM	1 バイト	1 バイト	
	最大のスタック使用量	160 バイト		
RX72N	ROM	489 バイト	253 バイト	
	RAM	1 バイト	1 バイト	
	最大のスタック使用量	28 バイト		

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_iwdt_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_iwdt_rx_if.h で記載されています。

以下に本モジュールの API 関数で使用する戻り値（エラーコード）を示します。戻り値の列挙型は、API 関数の宣言と共に r_iwdt_rx_if.h に記載されています。

```
typedef enum e_iwdt_err        // IWDT API エラーコード
{
    IWDT_SUCCESS=0,
    IWDT_ERR_OPEN_IGNORED,     // モジュールはすでにオープンされています。
    IWDT_ERR_INVALID_ARG,      // 無効な引数です。
    IWDT_ERR_NULL_PTR,         // 引数のポインタが NULL です。
    IWDT_ERR_NOT_OPENED,       // Open 関数が呼び出されていません。
    IWDT_ERR_BUSY,             // IWDT リソースがロックされています。
} iwdt_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+で開発中プロジェクトに FIT モジュールを追加
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_IWDT_Open()

この関数は、IWDT の関連レジスタを初期化し、IWDT カウンタのオプションを設定します。この関数は、IWDT が r_bsp_config.h の OFS0 レジスタによってオートスタートモードに設定されている場合には使用できません。

この関数は他の API 関数を呼び出す前に呼び出す必要があります。

Format

```
iwdt_err_t      R_IWDT_Open (
    void * const      p_cfg
)
```

Parameters

*void * const p_cfg*

iwdt_ config_t 型（下記を参照）の設定構造体へのポインタです。

レジスタスタートモードの実行時の設定可能オプションのすべてが以下で宣言されています。今後別のモード設定の使用を可能とするため、この構造体は Open()呼び出しで void ポインタにキャストされています。

```
typedef enum e_iwdt_timeout          // IWDT タイムアウト時間
{
#if defined (BSP_MCU_RX11_ALL) || defined(BSP_MCU_RX130) ||
defined(BSP_MCU_RX23_ALL) || defined(BSP_MCU_RX24U)
    IWDT_TIMEOUT_128 =0x0000u,       // 128（サイクル）
    IWDT_TIMEOUT_512 =0x0001u,       // 512（サイクル）
    IWDT_TIMEOUT_1024=0x0002u,       // 1024（サイクル）
    IWDT_TIMEOUT_2048=0x0003u,       // 2048（サイクル）
#else                    //* RX210, RX63N, RX64M, RX71M, RX65N, RX66T, RX72T */
    IWDT_TIMEOUT_1024 =0x0000u,      // 1024（サイクル）
    IWDT_TIMEOUT_4096 =0x0001u,      // 4096（サイクル）
    IWDT_TIMEOUT_8192 =0x0002u,      // 8192（サイクル）
    IWDT_TIMEOUT_16384=0x0003u,      // 16,384（サイクル）
#endif
    IWDT_NUM_TIMEOUTS
} iwdt_timeout_t;

typedef enum e_iwdt_clock_div        // IWDT クロック分周比
{
      IWDT_CLOCK_DIV_1  =0x0000u,    // IWDTCLK
      IWDT_CLOCK_DIV_16 =0x0020u,    // IWDTCLK/16
      IWDT_CLOCK_DIV_32 =0x0030u,    // IWDTCLK/32
      IWDT_CLOCK_DIV_64 =0x0040u,    // IWDTCLK/64
      IWDT_CLOCK_DIV_128=0x00F0u,    // IWDTCLK/128
      IWDT_CLOCK_DIV_256=0x0050u     // IWDTCLK/256
} iwdt_clock_div_t;

typedef enum e_iwdt_window_end       // ウィンドウ終了位置
{
      IWDT_WINDOW_END_75=0x0000u,    // 75%
      IWDT_WINDOW_END_50=0x0100u,    // 50%
      IWDT_WINDOW_END_25=0x0200u,    // 25%
```

```
      IWDT_WINDOW_END_0 =0x0300u      // 0% （ウィンドウ終了位置は未指定）
} iwdt_window_end_t;

typedef enum e_iwdt_window_start      // ウィンドウ開始位置
{
      IWDT_WINDOW_START_25 =0x0000u, // 25%
      IWDT_WINDOW_START_50 =0x1000u, // 50%
      IWDT_WINDOW_START_75 =0x2000u, // 75%
      IWDT_WINDOW_START_100=0x3000u  // 100%（ウィンドウ開始位置は未
                                     // 指定）
} iwdt_window_start_t;
typedef enum e_iwdt_timeout_control  // タイムアウトと
                                     // リフレッシュエラー時の信号制御
{
      IWDT_TIMEOUT_NMI  =0x00u       // NMI 要求出力が有効
      IWDT_TIMEOUT_RESET=0x80u,      // リセット出力が有効
} iwdt_timeout_control_t;

typedef enum e_iwdt_count_stop        // スリープモードのカウント停止
{
      IWDT_COUNT_STOP_DISABLE=0x00u, // カウント停止が無効
      IWDT_COUNT_STOP_ENABLE =0x80u  // 低消費電力モードへの遷移時に
                                     //     カウント停止が有効
} iwdt_count_stop_t;


typedef struct st_iwdt_config
{
    iwdt_timeout_t          timeout;          /* タイムアウト時間 */
    iwdt_clock_div_t        iwdtclk_div;      /* IWDT クロック分周比 */
    iwdt_window_start_t     window_start;     /* ウィンドウ開始位置 */
    iwdt_window_end_t       window_end;       /* ウィンドウ終了位置 */
    iwdt_timeout_control_t  timeout_control;  /* タイムアウト時のリセットまたは NMI
*/
    iwdt_count_stop_t       count_stop_enable;/* スリープモードのカウント停止フラグ
*/
 } iwdt_config_t;
```

Return Values

[IWDT_SUCCESS]	*/* IWDT が初期化されました。 */*
[IWDT_ERR_OPEN_IGNORED]	*/* モジュールはすでにオープンされています。 */*
[IWDT_ERR_INVALID_ARG]	*/* p_cfg 構造体の要素に無効な値が含まれています。 */*
[IWDT_ERR_NULL_PTR]	*/* “p_cfg”ポインタが NULL です。 */*
[IWDT_ERR_BUSY]	/* IWDT リソースがロックされています。 */

Properties

ファイル r_iwdt_rx_if.h にプロトタイプ宣言されています。

Description

独立ウォッチドッグタイマの設定可能オプションをすべて設定します。

Example

```
    iwdt_config_t  config;
    iwdt_err_t     err;
```

```
    /*
     * Configure the IWDT for:
     * - A 8.738 S timer period:
     *      15 kHz clock = 66.67 uS/tick; So 66.67 uS/tick * 128 * 1024 = 8.738
S
     * - A 100% refresh-permitted window (start 100% end 0%)
     * - Reset on counter underflow
     * - Count stop  enabled
     */
    config.timeout = IWDT_TIMEOUT_1024;
    config.iwdtclk_div = IWDT_CLOCK_DIV_128;
    config.window_start = IWDT_WINDOW_START_100;
    config.window_end = IWDT_WINDOW_END_0;
    config.timeout_control = IWDT_TIMEOUT_RESET;
    config.count_stop_enable = IWDT_COUNT_STOP_ENABLE;

    err = R_IWDT_Open(&config);
```

Special Notes:

Open 関数はレジスタスタートモード（BSP_CFG_OFS0_REG_VALUE = 0xFFFFFFFF）でのみ使用可能です。この関数は IWDT モジュールの設定を行いますが IWDT カウンタは開始しません。IWDT カウンタを開始するには R_IWDT_Control()関数でリフレッシュコマンドを使用する必要があります。

R_IWDT_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 IWDT_ERR_OPEN_IGNORED が返されます。

R_IWDT_Open()関数を呼び出す前に、ユーザアプリケーションで周辺クロック B（PCLKB）と内部クロックの IWDT 専用クロック（IWDTCLK）を初期化する必要があります。PCLKB クロックは、BSP でコンパイル時の構成設定により、設定することができます。IWDTCLK はレジスタに直接に設定することで初期化できます。IWDT 専用クロックの動作を開始する設定例を以下に示します。

```
R_BSP_RegisterProtectDisable(BSP_REG_PROTECT_CGC); // レジスタ保護オフ
SYSTEM.ILOCOCR.BIT.ILCSTP = 0;                     // IWDT 発振器の有効化
R_BSP_RegisterProtectEnable(BSP_REG_PROTECT_CGC);  // レジスタ保護オン
```

ユーザは PCLKB クロック周波数が分周後の IWDTCLK クロック周波数の 4 倍以上になるようにする必要があります。

NMI 信号を選択した場合、NMI ハンドラを構成するための設定が別に必要となります。レジスタスタートモードでは、割り込みコントローラモジュール（ICU）で IWDT 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設定は、R_IWDT_Open()で行います。オートスタートモードでは、これはユーザアプリケーションで行う必要があります。設定例を以下に示します。

```
    ICU.NMIER.BIT.IWDTEN = 1;   // IWDT アンダフロー/リフレッシュエラー割り込みの有効化
```

オートスタートモードとレジスタスタートモードのいずれにおいても、ユーザアプリケーションでこの割り込みを処理する関数を用意する必要があります。NMI ハンドラの実装例を以下に示します。

```
void iwdt_nmi_func(void *p_args)
{
    /* ここで何らかの処理を行う */
    while(IWDT.IWDTSR.BIT.REFEF == 1)
    {
       IWDT.IWDTSR.BIT.REFEF = 0;  // リフレッシュエラーフラグのクリア
    }
    while(IWDT.IWDTSR.BIT.UNDFF == 1)
    {
       IWDT.IWDTSR.BIT.UNDFF = 0;  // アンダフローフラグのクリア
```

```
    }
}
```

関数を登録するには、R_BSP_InterruptWrite()を呼び出します。例を以下に示します。

```
    err = R_IWDT_Open(&config);

    /* IWDT アンダフローが発生したら必ず呼び出されるよう iwdt_nmi_func()を登録します。*/
    err = R_BSP_InterruptWrite(BSP_INT_SRC_IWDT_ERROR, iwdt_nmi_func);
```

レジスタスタートモードでは、上記の例に示すように、R_IWDT_Open()の呼び出しの直後に関数を登録する必要があります。オートスタートモードでは、IWDT 割り込みを有効にした直後に R_BSP_InterruptWrite()関数を呼び出す必要があります。

R_IWDT_Control()

この関数は IWDT のステータスを取得し、IWDT のダウンカウンタをリフレッシュします。この関数は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使用されます。

Format

```
iwdt_err_t      R_IWDT_Control (
    iwdt_cmd_t const        cmd,
    uint16_t                * p_status
)
```

Parameters

iwdt_cmd_t const cmd
実行コマンド（下記の列挙子を参照）

uint16_t p_status
カウンタとステータスフラグの格納位置へのポインタ

cmd 引数には以下の列挙子を使用します。

```
IWDT_CMD_GET_STATUS,          // IWDT ステータスの取得
IWDT_CMD_REFRESH_COUNTING,    // カウンタのリフレッシュ
```

以下のマスクは*p_status パラメータ内の各フィールドを表しています。

```
#define IWDT_STAT_REFRESH_ERR_MASK      (0x8000)
#define IWDT_STAT_UNDERFLOW_ERR_MASK    (0x4000)
#define IWDT_STAT_ERROR_MASK            (0xC000)
#define IWDT_STAT_COUNTER_MASK          (0x3FFF)
```

Return Values

```
[IWDT_SUCCESS]          /* コマンドが正常に完了しました。          */
[IWDT_ERR_INVALID_ARG]  /* 引数の値が無効です。                    */
[IWDT_ERR_NULL_PTR]     /* p_status が NULL です。                 */
[IWDT_ERR_NOT_OPENED]   /* Open 関数が呼び出されていません。       */
[IWDT_ERR_BUSY]         /* IWDT リソースがロックされています。     */
```

Properties

ファイル r_iwdt_rx_if.h にプロトタイプ宣言されています。

Description

IWDT_CMD_REFRESH_COUNTING コマンドが選択された場合、ウォッチドッグカウンタが開始値に初期化され、カウントは続行されます。

IWDT_CMD_GET_STATUS コマンドが選択された場合、IWDT ステータスレジスタの値が*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が発生したか（正当なウィンドウ外からリフレッシュを呼び出した）、あるいはアンダフローが発生したか（カウンタの期限切れ）を示します。残りのビットは現在のカウンタ値を示します。

Example

```
iwdt_config_t  config;
iwdt_err_t     err;
uint16_t       status;
  :
err = R_IWDT_Open(&config);                              /* レジスタスタートモード */
  :
err = R_IWDT_Control(IWDT_CMD_REFRESH_COUNTING, NULL); /* カウントの開始 */
  :
err = R_IWDT_Control(IWDT_CMD_GET_STATUS, &status);    /* IWDT ステータスの取得 */
```

Special Notes:

IWDT_CMD_REFRESH_COUNTING コマンドでは、2 番目の引数は無視されます。

IWDT_CMD_REFRESH_COUNTING コマンドでウォッチドッグカウントを初期化する際、サイクル数で最大 4 サイクル必要となります（1 サイクル間の IWDT 専用クロック（IWDTCLK）数は、オープン時に設定したクロック分周比により異なります）。そのため、リフレッシュ許可期間終了位置から 4 カウント前、もしくはカウンタがアンダフローする 4 カウント前までに、IWDT_CMD_REFRESH_COUNTING コマンドの実行を完了してください。

R_IWDT_GetVersion()

この関数は実行時にドライバのバージョン番号を返します。

Format
uint32_t R_IWDT_GetVersion (void)

Parameters
なし

Return Values
バージョン番号

Properties
ファイル r_iwdt_rx_if.h にプロトタイプ宣言されています。

Description
このモジュールのバージョンを返します。バージョン番号は符号化され、最上位の 2 バイトがメジャーバージョン番号を、最下位の 2 バイトがマイナーバージョン番号を示しています。

Example
```
uint32_t   version;
  :
version = R_IWDT_GetVersion();
```

Special Notes:
なし

4. 端子設定

IWDT FIT モジュールは端子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 iwdt_demo_rskrx113

これは、RSKRX113 スタータキット（FIT モジュール"r_iwdt_rx"）用の RX113 独立ウォッチドッグタイマ（IWDT）の簡単なデモです。デモは、68ms の期間に対して IWDT を設定し、アンダフロー/リフレッシュエラーで割り込みを生成するよう設定しています。次に IWDT が開始され、50ms ごとに合計で 10 秒の間、リフレッシュを行い、その間 LED0 が点滅します。その後、IWDT のリフレッシュが停止し、ウォッチドッグタイマの期限が切れて割り込みが生成されるようになります。割り込みハンドラは、グローバルフラグを設定し、これにより、main()は LED0 の点滅を停止し、IWDT のアンダフローの発生を示す表示インジケータとして LED2 の点滅を開始します。

設定と実行

1. サンプルコードをコンパイルしてダウンロードします。
2. ソフトウェアを実行します。PC が Main で停止した場合、F8 を押して再開します。
3. ブレークポイントを設定してグローバル変数を確認します。

対応ボード

RSKRX113

5.2 iwdt_demo_rskrx231

これは、RSKRX231 スタータキット（FIT モジュール"r_iwdt_rx"）用の RX231 独立ウォッチドッグタイマ（IWDT）の簡単なデモです。デモは、68ms の期間に対して IWDT を設定し、アンダフロー/リフレッシュエラーで割り込みを生成するよう設定しています。次に IWDT が開始され、50ms ごとに合計で 10 秒の間、リフレッシュを行い、その間 LED0 が点滅します。その後、IWDT のリフレッシュが停止し、ウォッチドッグタイマの期限が切れて割り込みが生成されるようになります。割り込みハンドラは、グローバルフラグを設定し、これにより、main()は LED0 の点滅を停止し、IWDT のアンダフローの発生を示す表示インジケータとして LED2 の点滅を開始します。

設定と実行

1. サンプルコードをコンパイルしてダウンロードします。
2. ソフトウェアを実行します。PC が Main で停止した場合、F8 を押して再開します。
3. ブレークポイントを設定してグローバル変数を確認します。

対応ボード

RSKRX231

5.3 iwdt_demo_rskrx64m

これは、RSK RX64M スタータキット（FIT モジュール"r_iwdt_rx"）用の RX64M 独立ウォッチドッグタイマ（IWDT）の簡単なデモです。デモは、68ms の期間に対して IWDT を設定し、アンダフロー/リフレッシュエラーで割り込みを生成するよう設定しています。次に IWDT が開始され、50ms ごとに合計で 10 秒の間、リフレッシュを行い、その間 LED0 が点滅します。その後、IWDT のリフレッシュが停止し、ウォッチドッグタイマの期限が切れて割り込みが生成されるようになります。割り込みハンドラは、グローバルフラグを設定し、これにより、main()は LED0 の点滅を停止し、IWDT のアンダフローの発生を示す表示インジケータとして LED2 の点滅を開始します。

設定と実行

1. サンプルコードをコンパイルしてダウンロードします。

2. ソフトウェアを実行します。PC が Main で停止した場合、F8 を押して再開します。
3. ブレークポイントを設定してグローバル変数を確認します。

対応ボード

RSKRX64M

5.4 iwdt_demo_rskrx71m

これは、RSK RX71M スタータキット（FIT モジュール"r_iwdt_rx"）用の RX71M 独立ウォッチドッグタイマ（IWDT）の簡単なデモです。デモは、68ms の期間に対して IWDT を設定し、アンダフロー/リフレッシュエラーで割り込みを生成するよう設定しています。次に IWDT が開始され、50ms ごとに合計で 10 秒の間、リフレッシュを行い、その間 LED0 が点滅します。その後、IWDT のリフレッシュが停止し、ウォッチドッグタイマの期限が切れて割り込みが生成されるようになります。割り込みハンドラは、グローバルフラグを設定し、これにより、main()は LED0 の点滅を停止し、IWDT のアンダフローの発生を示す表示インジケータとして LED2 の点滅を開始します。

設定と実行

1. サンプルコードをコンパイルしてダウンロードします。
2. ソフトウェアを実行します。PC が Main で停止した場合、F8 を押して再開します。
3. ブレークポイントを設定してグローバル変数を確認します。

対応ボード

RSKRX71M

5.5 iwdt_demo_rskrx65n

これは、RSK RX65N スタータキット（FIT モジュール"r_iwdt_rx"）用の RX65N 独立ウォッチドッグタイマ（IWDT）の簡単なデモです。デモは、68ms の期間に対して IWDT を設定し、アンダフロー/リフレッシュエラーで割り込みを生成するよう設定しています。次に IWDT が開始され、50ms ごとに合計で 10 秒の間、リフレッシュを行い、その間 LED0 が点滅します。その後、IWDT のリフレッシュが停止し、ウォッチドッグタイマの期限が切れて割り込みが生成されるようになります。割り込みハンドラは、グローバルフラグを設定し、これにより、main()は LED0 の点滅を停止し、IWDT のアンダフローの発生を示す表示インジケータとして LED2 の点滅を開始します。

設定と実行

1. サンプルコードをコンパイルしてダウンロードします。
2. ソフトウェアを実行します。PC が Main で停止した場合、F8 を押して再開します。
3. ブレークポイントを設定してグローバル変数を確認します。

対応ボード

RSKRX65N

5.6 iwdt_demo_rskrx65n_2m

これは、RSK RX65N-2MB スタータキット（FIT モジュール"r_iwdt_rx"）用の RX65N-2MB 独立ウォッチドッグタイマ（IWDT）の簡単なデモです。デモは、68ms の期間に対して IWDT を設定し、アンダフロー/リフレッシュエラーで割り込みを生成するよう設定しています。次に IWDT が開始され、50ms ごとに合計で 10 秒の間、リフレッシュを行い、その間 LED0 が点滅します。その後、IWDT のリフレッシュが停止し、ウォッチドッグタイマの期限が切れて割り込みが生成されるようになります。割り込みハンドラは、グローバルフラグを設定し、これにより、main()は LED0 の点滅を停止し、IWDT のアンダフローの発生を示す表示インジケータとして LED2 の点滅を開始します。

設定と実行

1. サンプルコードをコンパイルしてダウンロードします。
2. ソフトウェアを実行します。PC が Main で停止した場合、F8 を押して再開します。
3. ブレークポイントを設定してグローバル変数を確認します。

対応ボード

RSKRX65N-2MB

5.7 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。

5.8 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 動作確認環境 (Rev.3.40)

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.7.7.0 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.3.40
使用ボード	Renesas Starter Kit+ for RX72N（型名：RTK5572Nxxxxxxxxxx）

表 6.2 動作確認環境 (Rev.3.30)

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。
エンディアン	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.3.30
使用ボード	RX13T CPU Card (型名：RTK0EMXA10C00000BJ)

表 6.3 動作確認環境 (Rev.3.20)

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。
エンディアン	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.3.20
使用ボード	Renesas Starter Kit+ for RX72M（型名：RTK5572Mxxxxxxxxxx）

表 6.4 動作確認環境 (Rev.3.10)

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 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.3.10
使用ボード	Renesas Solution Starter Kit for RX23W（型名：RTK5523Wxxxxxxxxx）

表 6.5 動作確認環境 (Rev.3.00)

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 -WI,--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.3.00
使用ボード	Renesas Starter Kit+ for RX65N-2MB（型名：RTK50565Nxxxxxxxxx）

表 6.6 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.7.3.0
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. 2.00
使用ボード	Renesas Starter Kit for RX72T（型名：RTK5572Txxxxxxxxxx）

表 6.7 動作確認環境 (Rev.1.91)

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.1.91
使用ボード	Renesas Starter Kit for RX66T（型名：RTK50566T0SxxxxxBE） Renesas Starter Kit+ for RX 65N-2MB（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

表 6.8 動作確認環境 (Rev.1.90)

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.1.90
使用ボード	Renesas Starter Kit for RX66T（型名：RTK50566T0SxxxxxBE） Renesas Starter Kit+ for RX 65N-2MB（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

表 6.9 動作確認環境 (Rev.1.81)

項目	内容
統合開発環境（IDE）	ルネサスエレクトロニクス製 e² studio V.6.0.0
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 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	Rev.1.81
使用ボード	Renesas Starter Kit+ for RX 65N-2MB（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

表 6.10 動作確認環境 (Rev.1.80)

項目	内容
統合開発環境（IDE）	ルネサスエレクトロニクス製 e² studio V.6.0.0
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 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	Rev.1.80
使用ボード	Renesas Starter Kit+ for RX 65N-2MB（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

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_iwdt_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_iwdt_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_iwdt_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「2.8 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
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	ポイント
1.00	2013.11.15	—	初版発行
1.20	2014.03.25	1,2 4 4 6	RX110、RX210、RX63N のサポートについての記述を追加。 r_iwdt_config.h テーブルから OFS0 を削除。 r_bsp_config.h 設定テーブルを追加。 e_iwdt_timeout に#ifdef を追加。
1.30	2014.12.30	1,3,5 15	RX113 のサポートについての記述を追加。 デモプロジェクトのセクションを追加。
1.40	2015.03.06	1,3,5,14	RX64M/RX71M のサポートについての記述を追加。 デモプロジェクトのセクションを追加。
1.50	2015.06.02	1,3,5,14	RX231 のサポートについての記述を追加。 デモプロジェクトのセクションを追加。
1.51	2016.03.01	1,3,4,6	RX130、RX230、RX23T のサポートについての記述を追加。
1.60	2016.10.01	1,4,6,9	RX65N のサポートについての記述を追加。 「2.9 コードサイズ」を変更。
1.70	2017.02.28	— — 4 12 プログラム	FIT モジュールの RX24T グループ（ROM 512KB 版を含む）、RX24U グループ対応。 誤記修正。 「2.5 対応ツールチェーン」に RXC v2.06.00 を追加。 3.4 R_IWDT_Control(): Special Notes に IWDT_CMD_REFRESH_COUNTING コマンドに関する記載を追加。 引数のチェックを、NULL と FIT_NO_PTR の両方でチェックするように修正。
1.80	2017.07.21	— 4 5 8	FIT モジュールの RX130 グループ（ROM 512KB 版）と RX65N グループ（ROM 2MB 版）対応。 「2.5 対応ツールチェーン」に RXC v2.07.00 を追加。 「2.6 割り込みベクタ」を追加。 「2.12 プロジェクトへの FIT モジュールの追加」を追加。
1.81	2017.10.31	17 17 18 19	「4.5 iwdt_demo_rskr65n」を追加。 「4.6 iwdt_demo_rskr65n_2m」を追加。 「4.8 デモのダウンロード方法」を追加。 「5. 付録」を追加。
1.90	2018.09.28	1, 3, 4 6 20	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.1.90 に対応する表を追加
1.91	2018.11.16	— 20	XML 内にドキュメント番号を追加。 Renesas Starter Kit+ for RX66T の型名を変更。 Rev.1.91 に対応する表を追加
2.00	2019.02.01	— 1、3、4、6 9-15 20	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.2.00 に対応する表を追加。

3.00	2019.05.20	—	以下のコンパイラをサポート。 - GCC for Renesas RX - IAR C/C++ Compiler for Renesas RX
		1	RX210、RX631、RX63N の更新終了につき、「対象デバイス」からこれらのデバイスを削除。 「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。
		3	「1.2 IWDT FIT モジュールの概要」 RX210、RX631、RX63N の説明を削除。
		4	「2.2 ソフトウェアの要求」r_bsp v5.20 以上が必要
		6	「2.8 コードサイズ」セクションを更新。
		22	表 6.1「動作確認環境」： Rev.3.00 に対応する表を追加。
		26	「Web サイトおよびサポート」のセクションを削除。
		プログラム	GCC と IAR コンパイラに関して、以下を変更。 R_IWDT_GetVersion 関数のインライン展開を削除。
3.10	2019.06.28	1、4	RX23W のサポートを追加。
		6	RX23W に対応するコードサイズを追加。
		11	R_IWDT_Open 関数内で、defined(BSP_MCU_RX24U) を追加。
		22	R_IWDT_Open 関数内で、RX71M, RX65N, RX66T, RX72T に関するコメントを追加。 「6.1 動作確認環境」： Rev.3.10 に対応する表を追加。 RX23W のサポートを追加。
		プログラム	
3.20	2019.08.15	1, 3, 4	RX72M のサポートを追加。
		6-8	RX72M に対応するコードサイズを追加。
		22	「6.1 動作確認環境」： Rev.3.20 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。
		プログラム	
3.30	2019.11.25	1, 3, 5	RX13T のサポートを追加。
		4	2.3 制限事項 制限事項を追加。
		7-9	RX13T に対応するコードサイズを追加。
		23	「6.1 動作確認環境」： Rev.3.30 に対応する表を追加。 RX13T のサポートを追加。
		プログラム	API 関数のコメントを Doxygen スタイルに変更。
3.40	2019.12.30	1, 3, 5	RX66N、 RX72N のサポートを追加。
		7-9	RX66N、 RX72N に対応するコードサイズを追加。
		23	「6.1 動作確認環境」： Rev.3.40 に対応する表を追加。 RX66N、 RX72N のサポートを追加。
		プログラム	

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。