RX ファミリ

CMT 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 CMT モジュールについて説明します。本モジュールは CMT を使用して、繰り返しのイベントを生成し、その間隔を固定します。以降、本モジュールを CMT FIT モジュールと称します。

CMT FITモジュールはRTOS (FreeRTOS, RI600V4, RI600PX) のプロジェクト に組み込むこ とが可能です。

このとき、CMT FITモジュールは、RTOSのタイムマネジメント機能によって使用される
CMTのチャネルとは異なるチャネルを使用します。

対象デバイス

- RX110 グループ
- RX111 グループ
- RX113 グループ
- RX130 グループ
- RX13T グループ
- RX140 グループ
- RX230 グループ
- RX231 グループ
- RX23T グループ
- RX23W グループ（チャネル 2, 3 は使用できません）
- RX23E-A グループ
- RX24T グループ
- RX24U グループ
- RX64M グループ (RTOS システムタイマをサポート)
- RX65N グループ、RX651 グループ (RTOS システムタイマをサポート)
- RX66T グループ
- RX66N グループ
- RX671 グループ
- RX71M グループ (RTOS システムタイマをサポート)
- 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 4
1.1 CMT FIT モジュールとは 4
1.2 CMT FIT モジュールの概要 4
1.3 CMT FIT モジュールを使用する 4
1.3.1 CMT FIT モジュールを C++プロジェクト内で使用する 4
1.4 API の概要 5
1.5 制限事項 5

2. API 情報 6
2.1 ハードウェアの要求 6
2.2 ソフトウェアの要求 6
2.3 制限事項 6
2.3.1 RAM の配置に関する制限事項 6
2.3.2 RX23W ではチャネル 2, 3 を使用できません。 6
2.4 サポートされているツールチェーン 6
2.5 使用する割り込みベクタ 6
2.6 ヘッダファイル 7
2.7 整数型 7
2.8 コンパイル時の設定 7
2.9 コードサイズ 8
2.10 引数 11
2.11 戻り値 11
2.12 コールバック関数 11
2.13 FIT モジュールの追加方法 12
2.14 for 文、while 文、do while 文について 13

3. API 関数 14
R_CMT_CreatePeriodic() 14
R_CMT_CreatePeriodicAssignChannelPriority() 16
R_CMT_CreateOneShot() 18
R_CMT_CreateOneShotAssignChannelPriority() 19
R_CMT_Stop() 21
R_CMT_Control() 22
R_CMT_GetVersion() 26

4. 端子設定 27

5. デモプロジェクト 28
5.1 cmt_demo_rskrx113, cmt_demo_rskrx113_gcc 28
5.2 cmt_demo_rskrx231, cmt_demo_rskrx231_gcc 28
5.3 cmt_demo_rskrx64m, cmt_demo_rskrx64m_gcc 28

5.4 cmt_demo_rskrx71m, cmt_demo_rskrx71m_gcc 28
5.5 cmt_demo_rskrx65n, cmt_demo_rskrx65n_gcc 28
5.6 cmt_demo_rskrx65n_2m, cmt_demo_rskrx65n_2m_gcc 28
5.7 cmt_demo_rskrx72m, cmt_demo_rskrx72m_gcc 28
5.8 cmt_demo_rskrx671, cmt_demo_rskrx671_gcc 28
5.9 ワークスペースへのデモ追加 29
5.10 デモのダウンロード方法 29

6. 付録 30
6.1 動作確認環境 30
6.2 トラブルシューティング 40

7. 参考ドキュメント 41

テクニカルアップデートの対応について 41

改訂記録 42

1. 概要

1.1 CMT 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 CMT FIT モジュールの概要

本モジュールは、RX の周辺機能であるコンペアマッチタイマ（CMT）を使用するためのシンプルなインタフェースを提供します。CMT は 2 チャネルの 16 ビットタイマです。

各チャネルには、プリスケーラ、16 ビットの比較レジスタと共にフリーランニングカウンタが含まれます。フリーランニングカウンタが比較レジスタと一致すると、割り込みの生成が可能になります。コンペアマッチイベントで、カウンタは自動的にリセット、再開されますので、RTOS スケジューラのように、繰り返しのソフトウェアイベントを調整するのに理想的なタイマとなります。CMT は 2 チャネルですが、RX MCU は製品によって、CMT を 1、または 2 ユニット装備していますので、それぞれ独立した CMT チャネルを 2、または 4 チャネル持つことになります。

本モジュールは、CMT チャネルの作成および開始、チャネルの一時停止および再開、チャネルの終了処理を行う関数を提供します。ユーザアプリケーションコードはコールバック関数を使って呼び出されます。

1.3 CMT FIT モジュールを使用する

CMT モジュールの本来の使用目的は繰り返しのイベントが簡単に生成できるようにし、その間隔を固定することです。

CMT FIT モジュールをプロジェクトに追加後、インストールに合わせてソフトウェアを設定するために、r_cmt_rx_config.h ファイルを変更する必要があります。

R_CMT_CreatePeriodic 関数と R_CMT_CreateOneShot 関数を使って、タイマを開始します。コールバック関数へのポインタを引数として提供します。タイマのコンペマッチイベントが発生するとコールバック関数が呼び出されます。コールバック関数は ISR に関連して実行されるため、コールバック関数の実行中は、割り込みが禁止されるようにデフォルトで設定されています。そのため、コールバック関数はできるだけ小さくして、処理が早く完了できるようにしてください。

理論上は、CMT タイマのクロックの最大速度は PCLK/8 に制限されています。クロックの生成に R_CMT_CreatePeriodic 関数を使用する場合、割り込みとコールバック関数の処理に少し時間を要することがありますので注意が必要です。そのため、生成し得る最大周波数を制限しています。

CMT チャネルを RTOS システムタイマとして使用するには、#define BSP_CFG_RTOS_USED と #define BSP_CFG_RTOS_SYSTEM_TIMER を r_bsp_config.h 内で設定します。これら 2 個のマクロは、CMT チャネルに対する保護を提供します。その結果、RTOS システムタイマのみが CMT チャネルを排他的に使用します。

1.3.1 CMT 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 GPIO 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_gpio_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_CMT_CreatePeriodic()	未使用の CMT チャネルを検索し、適切なプリスケーラを設定することによって、要求される周期の周波数のタイマを設定します。また、ユーザのコールバック関数をそのタイマの割り込みと関連付けし、タイマを開始します。ユーザがタイマを停止するまで、設定した周期で割り込み生成やコールバック関数の呼び出しが行われ、タイマは動き続けます。
R_CMT_CreateOneShot()	R_CMT_CreatePeriodic 関数と類似していますが、最初の割り込みに起因したコールバック関数で、タイマは停止されます。
R_CMT_Control()	タイマを一時的に停止、再開、あるいはステータスをレポートするコマンドです。
R_CMT_Stop()	CMT チャネルを終了し、割り込みを禁止します。使用中でなければ CMT 周辺機能を終了します。
R_CMT_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

特になし。

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。

- CMT

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
- CMT を起動する前に、周辺クロックを初期化しておく必要があります。

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。

2.3.2 RX23W ではチャネル 2, 3 を使用できません。

RX23W のチャネル 2，3 は、BLE FIT モジュールで使用されるため使用できません。

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

CMT の割り込みは、R_CMT_CreatePeriodic 関数と R_CMT_CreateOneShot 関数を実行することで有効化されます。

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

デバイス	割り込みベクタ
RX110*1 RX111*1 RX113 RX130*1 RX13T*1 RX140*1 RX230 RX231 RX23T RX23W RX23E-A*1 RX24T RX24U RX66T RX66N RX72T RX72M*1 RX72N	CMI0 割り込み[チャネル 0] (ベクタ番号：28) CMI1 割り込み[チャネル 1] (ベクタ番号：29) CMI2 割り込み[チャネル 2] (ベクタ番号：30) CMI3 割り込み[チャネル 3] (ベクタ番号：31)
RX64M RX651 RX65N RX71M	CMI0 割り込み[チャネル 0] (ベクタ番号：28) CMI1 割り込み[チャネル 1] (ベクタ番号：29) CMI2 割り込み[チャネル 2] (ベクタ番号：128)*2 CMI3 割り込み[チャネル 3] (ベクタ番号：129)*2

注 1：2 チャネル（CMT0、CMT1）のみ。
注 2：選択型割り込みに割り当てられている割り込みの割り込みベクタ番号は、ボードサポートパッケー
　　FIT モジュール（BSP モジュール）で指定したデフォルト値を示しています。

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_cmt_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_cmt_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_cmt_rx_config.h）	
CMT_RX_CFG_IPR ※デフォルト値は“5”	CMT 割り込みで使用される割り込み優先レベル

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」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」の C コンパイラでの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		GCC		IAR コンパイラ		
		パラメータチェック処理なし	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	パラメータチェック処理なし	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	
MCU x 2 チャネルの場合（RX110、RX111、RX130, RX13T）	ROM	1302 バイト	1302 バイト	2416 バイト	2416 バイト	1968 バイト	1968 バイト	
	RAM	16 バイト	16 バイト	16 バイト	16 バイト	10 バイト	10 バイト	
	最大使用スタックサイズ	188 バイト	188 バイト	-	-	120 バイト	120 バイト	
MCU x 4 チャネルの場合（RX113、RX230、RX231、RX23T、RX24T、RX24U、RX64M、RX651、RX65N、RX66T、RX66N, RX71M, RX72N）	ROM	1834 バイト	1834 バイト	3288 バイト	3288 バイト	2860 バイト	2802 バイト	
	RAM	32 バイト	32 バイト	32 バイト	32 バイト	20 バイト	20 バイト	
	最大使用スタックサイズ	60 バイト	60 バイト	-	-	108 バイト	108 バイト	
MCU x 4 チャネルの場合（RX23W）	ROM	1800 バイト	1800 バイト					
	RAM	32 バイト	32 バイト					
	最大使用スタックサイズ	112 バイト	112 バイト					
MCU x 4 チャネルの場合（RX72T）	ROM	1815 バイト	1815 バイト	3152 バイト	3152 バイト	2918 バイト	2979 バイト	
	RAM	32 バイト	32 バイト	32 バイト	32 バイト	20 バイト	20 バイト	
	最大使用スタックサイズ	68 バイト	68 バイト	-	-	200 バイト	200 バイト	最大割り込みスタック
MCU x 4 チャネルの場合（RX72M）	ROM	1834 バイト	1834 バイト	3288 バイト	3288 バイト	3018 バイト	3010 バイト	
	RAM	32 バイト	32 バイト	32 バイト	32 バイト	10 バイト	10 バイト	

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		GCC		IAR コンパイラ		
		パラメータチェック処理なし	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	パラメータチェック処理なし	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	
	最大使用スタックサイズ	156 バイト	156 バイト	-	-	204 バイト	204 バイト	
MCU x 2 チャネルの場合（RX23E-A）	ROM	1348 バイト	1348 バイト	2424 バイト	2424 バイト	2016 バイト	2016 バイト	
	RAM	16 バイト	16 バイト	16 バイト	16 バイト	10 バイト	10 バイト	
	最大使用スタックサイズ	196 バイト	196 バイト	-	-	204 バイト	204 バイト	
MCU x 4 チャネルの場合（RX671)	ROM	1851 バイト	1851 バイト	3416 バイト	3416 バイト	2863 バイト	2859 バイト	
	RAM	32 バイト	32 バイト	32 バイト	32 バイト	20 バイト	20 バイト	
	最大使用スタックサイズ	60 バイト	60 バイト	-	-	144 バイト	144 バイト	
MCU x 2 チャネルの場合（RX140)	ROM	1326 バイト	1326 バイト	2424 バイト	2424 バイト	2023 バイト	2023 バイト	
	RAM	16 バイト	16 バイト	16 バイト	16 bytes	10 バイト	10 バイト	
	最大使用スタックサイズ	60 バイト	60 バイト	-	-	128 バイト	128 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_cmt_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_cmt_rx_if.h で記載されています。

CMT モジュールで提供される関数はすべて、呼び出しの成功または失敗を示すブール型で値を返します。

2.12 コールバック関数

本モジュールでは、CMT 割り込みが発生したタイミングで、ユーザが設定したコールバック関数を呼び出します。

コールバック関数は、「2.9 引数」に記載された構造体メンバ "void (* callback)(void * pdata)" に、ユーザの関数のアドレスを格納することで設定されます。コールバック関数が呼び出されるとき、チャネル番号を格納する変数が引数として渡されます。

引数の型は void ポインタ型で渡されるため、コールバック関数の引数は下記の例を参考に void 型のポインタ変数としてください。

コールバック関数内部で値を使うときはキャストして値を使用してください。

```
void my_cmt_callback(void * pdata)
{
    uint32_t  cmt_event_channel_number;

    cmt_event_channel_number = *((uint32_t *)pdata); //cast pointer to
uint32_t
    ...
}
```

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_CMT_CreatePeriodic()

この関数は未使用の CMT チャネルを検出し、要求される周期の周波数に合わせてタイマを設定します。また、ユーザのコールバック関数をそのタイマの割り込みと関連付けし、タイマを起動して、カウントを開始します。

Format

```
bool R_CMT_CreatePeriodic( uint32_t frequency_hz
                           void (* callback) (void *pdata),
                           uint32_t *channel)
```

Parameters

frequency_hz

要求される周波数（Hz）(1~PCLK/8 Hz) (注 1)。周辺クロックの設定によって、タイマの範囲と分解能が決定されます。使用する CMT チャネルに最も適したプリスケーラが本モジュールによって選択されます。

callback

ユーザ設定のコールバック関数へのポインタ。引数 には void *を指定してください。

channel

CMT FIT モジュールは 1 つ目の使用されていない CMT チャネルを検出し、呼び出し元にそのチャネルを割り当てます。こうすることで、すべてのタイマのチャネルを事前に割り当てずとも、複数の CMT チャネルで本モジュールを使用することが可能になります。本引数は、割り当てられたチャネルを呼び出し元に返します。

Return Values

true: */*成功; CMT が初期化されました。 */*

false: */* 空いている CMT チャネルがない、または無効な設定です。 */*

Properties

"r_cmt_rx_if.h" ファイル内でプロトタイプ宣言されています。

Description

R_CMT_CreatePeriodic 関数は未使用の CMT チャネルを見つけ、そのチャネルを呼び出し側に割り当て、コンペアマッチイベントの際に呼び出すユーザコールバック関数を登録します。呼び出しの際に指定した周波数でコンペアマッチを生成するように、CMT が設定されます。

Example

この例では、コンペアマッチで実行されるコールバック関数を 10Hz(100ms)に設定しています。

cb はコンペアマッチイベントが発生したことを通知するユーザが提供するコールバック関数です。

```
uint32_t   ch;
bool       ret;

ret = R_CMT_CreatePeriodic(10, &cb, &ch);

if (true != ret)
{
    /* Handle the error */
}
```

Special Notes:

1. 最大周期周波数

ハードウェアでは、CMT タイマのクロックの最大速度は PCLK/8 に制限されています。クロックの生成に R_CMT_CreatePeriodic 関数を使用する場合、割り込みとコールバック関数の処理に少し時間を要することがあるので注意が必要です。要求された周波数が高くなると、割り込みとコールバックの処理に要するプロセッサ時間の割合が増します。そうなると、ある時点で時間を消費しすぎて、その他の有用な作業を処理するための時間がなくなってしまいます。そのため、生成し得る最大周波数を制限しています。実質的な最大周波数はご使用のシステム設計によりますが、一般に数キロヘルツ以下の周波数が妥当と言えます。

R_CMT_CreatePeriodicAssignChannelPriority()

この関数は、要求された周波数と希望の割り込み優先レベルに合わせて希望の CMT チャネルを構成し、ユーザコールバック関数をタイマの割り込みに関連付け、タイマの電源を投入してタイマを開始します。

Format

```
bool    R_CMT_CreatePeriodicAssignChannelPriority (
            uint32_t            frequency_hz
            void (* callback) (void *pdata),
            uint32_t            channel,
            cmt_priority_t      priority
)
```

Parameters

uint32_t frequency_hz

希望の周波数は Hz 単位です (1 ～ PCLK/8 Hz) 注記 1.タイマの範囲と分解能は、周辺回路クロックの設定によって決定されます。ドライバが、CMT チャネルにとって最善のプリスケーラを選択します。

callback

ユーザコールバック関数へのポインタ。単一の void * 引数を指定する必要があります。

uint32_t channel

構成に使用する、希望の CMT チャネル。

cmt_priority_t priority

タイマの割り込みに関する希望の優先レベル：

CMT_PRIORITY_0：割り込みは無効です
CMT_PRIORITY_1：最小の割り込み優先順位
CMT_PRIORITY_2
CMT_PRIORITY_3
CMT_PRIORITY_4
CMT_PRIORITY_5
CMT_PRIORITY_6
CMT_PRIORITY_7
CMT_PRIORITY_8
CMT_PRIORITY_9
CMT_PRIORITY_10
CMT_PRIORITY_11
CMT_PRIORITY_12
CMT_PRIORITY_13
CMT_PRIORITY_14
CMT_PRIORITY_15：最大の割り込み優先順位

Return Values

[true] */* 成功; CMT が初期化されました。 */*
[false] */* 空き CMT チャネルが利用できないか、無効な設定です 注記 2 */*

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

Description
R_CMT_CreatePeriodicAssignChannelPriority は、希望の割り込み優先順位レベルを使用して、希望の CMT チャネルを呼び出し側に割り当てます。また、比較の一致イベントが発生した場合に呼び出されるユーザコールバック関数を登録します。CMT は、呼び出しの中で指定した周波数 (頻度) で、一致の比較を生成するように構成されます。

Example
このサンプルは、CMT チャネル 0 と割り込み優先順位レベル 7 と指定のコールバックを使用して、一致の比較動作を 10 Hz (100 ms) の周波数に設定します。
このサンプルは、比較の一致イベントが発生するたびにユーザへの通知を行う目的で呼び出される、ユーザ指定のコールバック関数 cb を示しています。

```
uint32_t   ch = 0;
bool       ret;
cmt_priority_t priority = CMT_PRIORITY_7;

ret = R_CMT_CreatePeriodicAssignChannelPriority(10, &cb, ch, priority);

if (true != ret)
{
    /* Handle the error */
}
```

Special Notes:
1. 最大周期周波数
   ハードウェア内で、CMT タイマの最大クロック速度は PCLK/8 に制限されています。ただし、周期タイマ関数を使用してクロックを生成する場合、割り込みとコールバック処理がある程度の時間を要することに注意してください。要求する周波数を引き上げると、割り込みとコールバック処理がプロセッサ時間の中に占める比率が増加します。特定の時点で、消費する時間が過度に多くなり、他の有用な作業に残すことができる時間がほとんど存在しなくなります。したがって、この上限を設けて、生成に使用できる最大周波数を制限しています。この最大周期周波数はお客様のシステム設計によって異なりますが、一般的には、数キロヘルツ (kHz) が適切です。
2. 無効な設定
   以下の無効な設定のいずれかが見つかった場合、この関数は FALSE を返します。**無効なチャネル、無効な優先順位、チャネルが使用中だった、指定の周波数が使用できなかった。**

R_CMT_CreateOneShot()

この関数は未使用の CMT チャネルを検出し、要求される周期に合わせてタイマを設定します。また、ユーザのコールバック関数をそのタイマの割り込みと関連付けし、タイマを起動、カウントを開始します。

Format

```
bool R_CMT_CreateOneShot( uint32_t period_us,
                void (* callback)(void *pdata),
                uint32_t *channel)
```

Parameters

period_us

要求される周期（μs）(1~1,000,000us)。タイマの範囲と解像度が周辺クロックの設定によって決定されます。使用する CMT チャネルに最も適したプリスケーラが本モジュールによって選択されます。

callback

ユーザ設定のコールバック関数へのポインタ。引数 には void *を指定してください。

channel

CMT FIT モジュールは 1 つ目の使用されていない CMT チャネルを検出し、呼び出し元にそのチャネルを割り当てます。こうすることで、すべてのタイマのチャネルを事前に割り当てずとも、複数の CMT チャネルで本モジュールを使用することが可能になります。本引数は、割り当てられたチャネルを呼び出し元に返します。

Return Values

true: */*成功; CMT が初期化されました。 */*

false: */* 空いている CMT チャネルがない、または無効な設定です。 */*

Properties

ファイル r_cmt_rx_if.h にプロトタイプ宣言されています。

Description

R_CMT_CreateOneShot 関数が未使用の CMT チャネルを検出し、それを呼び出し元に割り当てます。また、コンペアマッチイベントで呼び出されるユーザ設定のコールバック関数を登録します。CMT は、指定された周期後にコンペアマッチが生成されるように設定されます。コンペアマッチイベントが 1 度発生すると、タイマは停止されます。

Example

この例では、コンペアマッチで実行されるコールバック関数を 10Hz(100ms)に設定しています。

```
uint32_t   ch;
bool       ret;

ret = R_CMT_CreateOneShot(100000, &cb, &ch);

if (true != ret)
{
    /* Handle the error */
}
```

Special Notes:

なし

R_CMT_CreateOneShotAssignChannelPriority()

この関数は、要求された周波数と希望の割り込み優先レベルに合わせて希望の CMT チャネルを構成し、ユーザコールバック関数をタイマの割り込みに関連付け、タイマの電源を投入してタイマを開始します。

Format

```
bool    R_CMT_CreateOneShotAssignChannelPriority (
            uint32_t        frequency_hz
            void (* callback) (void *pdata),
            uint32_t        channel,
            cmt_priority_t  priority
)
```

Parameters

uint32_t frequency_hz

希望の周波数は Hz 単位です (1 ~ 1,000,000us) 注記 1.タイマの範囲と分解能は、周辺回路クロックの設定によって決定されます。ドライバが、CMT チャネルにとって最善のプリスケーラを選択します。

callback

ユーザ設定のコールバック関数へのポインタ。引数 には void *を指定してください。

uint32_t channel

構成に使用する、希望の CMT チャネル。

cmt_priority_t priority

タイマの割り込みに関する希望の優先レベル：

CMT_PRIORITY_0：割り込みは無効です
CMT_PRIORITY_1：最小の割り込み優先順位
CMT_PRIORITY_2
CMT_PRIORITY_3
CMT_PRIORITY_4
CMT_PRIORITY_5
CMT_PRIORITY_6
CMT_PRIORITY_7
CMT_PRIORITY_8
CMT_PRIORITY_9
CMT_PRIORITY_10
CMT_PRIORITY_11
CMT_PRIORITY_12
CMT_PRIORITY_13
CMT_PRIORITY_14
CMT_PRIORITY_15：最大の割り込み優先順位

Return Values

[true] */* 成功; CMT が初期化されました。 */*
[false] */* 空き CMT チャネルが利用できないか、無効な設定です 注記 2 */*

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

Description
R_CMT_CreateOneShotAssignChannelPriority は、希望の割り込み優先順位レベルを使用して、希望の CMT チャネルを呼び出し側に割り当てます。また、比較の一致イベントが発生した場合に呼び出されるユーザコールバック関数を登録します。CMT は、呼び出しの中で指定した周波数 (頻度) で、一致の比較を生成するように構成されます。

Example
この例では、コンペアマッチで実行されるコールバック関数を 10Hz(100ms)に設定しています。

```
uint32_t   ch = 0;
bool       ret;
cmt_priority_t priority = CMT_PRIORITY_7;

ret = R_CMT_CreateOneShotAssignChannelPriority(100000, &cb, ch, priority);

if (true != ret)
{
    /* Handle the error */
}
```

Special Notes:
1. 無効な設定

   以下の無効な設定のいずれかが見つかった場合、この関数は FALSE を返します。**無効なチャネル、無効な優先順位、チャネルが使用中だった、指定の周波数が使用できなかった。**

R_CMT_Stop()

CMT チャネルを停止し、可能な状態であれば CMT 周辺機能を終了します。

Format

bool R_CMT_Stop(uint32_t channel);

Parameters

channel

停止する CMT タイマのチャネル

Return Values

true: */* 成功; CMT を終了しました。 */*

false: */* 無効な設定です。*/*

Properties

ファイル r_cmt_rx_if.h にプロトタイプ宣言されています。

Description

本関数は割り当てをクリアし、関連する割り込みを禁止することによって、CMT チャネルを開放します。開放された CMT チャネルは R_CMT_CreatePeriodic 関数、または R_CMT_CreateOneShot 関数で再起動されるまで使用できません。

CMT チャネルを既に RTOS システムタイマとして使用している場合、その CMT チャネルを channel で指定してこの関数を呼び出すと、FALSE が返される結果になります。

Example

CMT タイマのチャネル停止の例です。

```
uint32_t   ch;
bool       ret;

/* Open and start the timer */
ret = R_CMT_CreatePeriodic(10, &cb, &ch);

/* Stop the timer */
ret = R_CMT_Stop(ch);

if (true != ret)
{
    /* Handle the error */
}
```

Special Notes:

なし

R_CMT_Control()

この関数は CMT チャネルを制御し、監視する様々な方法を提供します。

Format

```
bool R_CMT_Control ( uint32_t channel,
                     cmt_commands_t command,
                     void *pdata);
```

Parameters

channel
制御対象の CMT チャネル番号

command
実行されるコマンド：

CMT_RX_CMD_IS_CHANNEL_COUNTING

CMT_RX_CMD_PAUSE

CMT_RX_CMD_RESUME

CMT_RX_CMD_RESTART

CMT_RX_CMD_GET_NUM_CHANNELS

CMT_RX_CMD_SET_PRIORITY

CMT_RX_CMD_GET_PRIORITY

Return Values

true: */* コマンドを正しく完了しました。pdata を確認してください。 */*
false: */* コマンドを正しく完了できませんでした。*/*

Properties

ファイル r_cmt_rx_if.h にプロトタイプ宣言されています。

Description

本関数では様々なコマンドが提供されます。

- CMT_RX_CMD_IS_CHANNEL_COUNTING:
  CMT チャネルが現在動作中かどうかを示します。*pdata を確認します。
- CMT_RX_CMD_PAUSE:
  タイマを一時停止します（終了はしません）。
- CMT_RX_CMD_RESUME:
  カウンタを 0 にリセットせずに、一時停止していたタイマを再開します。
- CMT_RX_CMD_RESTART:
  カウンタを 0 にリセットした後、一時停止していたタイマを再開します。
- CMT_RX_CMD_GET_NUM_CHANNELS:
  使用可能な総チャネル数を返します。

CMT チャネルを既に RTOS システムタイマとして使用している場合、その CMT チャネルを channel で指定し、CMT_RX_CMD_IS_CHANNEL_COUNTING、CMT_RX_CMD_PAUSE、CMT_RX_CMD_RESUME、

CMT_RX_CMD_RESTART のいずれかを command で指定してこの関数を呼び出すと、FALSE が返される結果になります。

Example 1

CMT タイマの一時停止と一時停止していたタイマの再開の例です。

```
uint32_t   ch;
bool       ret;

/* Open and Start the timer */
ret = R_CMT_CreatePeriodic(10, &cb, &ch);

if (true != ret)
{
    /* Handle the error */
}

/* Pause the timer */
ret = R_CMT_Control(ch, CMT_RX_CMD_PAUSE, NULL);

if (true != ret)
{
    /* Handle the error */
}

/* Restart the timer after resetting the counter to zero */
ret = R_CMT_Control(ch_info, CMT_RX_CMD_RESTART, NULL);

if (true != ret)
{
    /* Handle the error */
}
```

Example 2

CMT の状態確認と使用可能なチャネル数の取得方法の例です。

```
uint32_t   ch;
uint32_t   ch_num;
bool       ret;
bool       data;

/* Open and Start the timer */
ret = R_CMT_CreatePeriodic(10, &cb, &ch);

if (true != ret)
{
    /* Handle the error */
}

/* Check state of channel */
ret = R_CMT_Control(ch, CMT_RX_CMD_IS_CHANNEL_COUNTING, (void*)&data);

if (true != ret)
{
    /* Handle the error */
}
```

```
/* Get available of channel */
ret = R_CMT_Control(ch, CMT_RX_CMD_GET_NUM_CHANNELS, (void*)&ch_num);

if (true != ret)
{
    /* Handle the error */
}
```

Special Notes:

なし

R_CMT_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_CMT_GetVersion(void);

Parameters

なし

Return Values

メジャーバージョンとマイナーバージョンからなる 32 ビット値で示されるバージョン番号

Properties

ファイル r_cmt_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

この関数の使用方法を示す例。

```
/* バージョン番号を取り出し、取り出した番号を文字列に変換する*/

uint32_t   version, version_high, version_low;
char       version_str[9];

version = R_CMT_GetVersion();

version_high = (version >> 16)&0xf;
version_low  =  version & 0xff;

sprint(version_str, “CMT v%1.1hu.%2.2hu”, version_high, version_low);
```

Special Notes:

なし

4. 端子設定

CMT 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 cmt_demo_rskrx113, cmt_demo_rskrx113_gcc

cmt_demo_rskrx113, cmt_demo_rskrx113_gcc プロジェクトは、CMT チャネルを使った Timer Tick の作成方法、CMT 割り込みを扱うコールバック関数の設定方法、またコールバックの引数のチャネル情報を逆引きする方法をデモするものです。プログラム実行時、CMT のコールバック関数は 2 Hz 間隔で LED0 をトグルします。

5.2 cmt_demo_rskrx231, cmt_demo_rskrx231_gcc

cmt_demo_rskrx231, プロジェクトは、cmt_demo_rskrx113 と同じものです。

cmt_demo_rskrx231_gcc, プロジェクトは、cmt_demo_rskrx113_gcc と同じものです。

5.3 cmt_demo_rskrx64m, cmt_demo_rskrx64m_gcc

cmt_demo_rskrx64m プロジェクトは、cmt_demo_rskrx113 と同じものです。

cmt_demo_rskrx64m_gcc プロジェクトは、cmt_demo_rskrx113_gcc と同じものです。

5.4 cmt_demo_rskrx71m, cmt_demo_rskrx71m_gcc

cmt_demo_rskrx71m プロジェクトは、cmt_demo_rskrx113 と同じものです。

cmt_demo_rskrx71m_gcc プロジェクトは、cmt_demo_rskrx113_gcc と同じものです。

5.5 cmt_demo_rskrx65n, cmt_demo_rskrx65n_gcc

cmt_demo_rskrx65n, cmt_demo_rskrx65n_gcc プロジェクトは、CMT チャネルを使った Timer.Tick の作成方法、CMT 割り込みを扱うコールバック関数の設定方法、またコールバックの引数のチャネル情報を逆引きする方法をデモするものです。プログラム実行時、CMT のコールバック関数は 2 Hz 間隔で LED0 と LED1 をトグル（交互に点灯）します。

5.6 cmt_demo_rskrx65n_2m, cmt_demo_rskrx65n_2m_gcc

cmt_demo_rskrx65n_2m プロジェクトは、cmt_demo_rskrx65n と同じものです。

cmt_demo_rskrx65n_2m_gcc プロジェクトは、cmt_demo_rskrx65n_gcc と同じものです。

5.7 cmt_demo_rskrx72m, cmt_demo_rskrx72m_gcc

cmt_demo_rskrx72m プロジェクトは、cmt_demo_rskrx65n_2m と同じものです。

cmt_demo_rskrx72m_gcc プロジェクトは、cmt_demo_rskrx65n_2m_gcc と同じものです。

5.8 cmt_demo_rskrx671, cmt_demo_rskrx671_gcc

cmt_demo_rskrx671 プロジェクトは、cmt_demo_rskrx72m と同じものです。

cmt_demo_rskrx671_gcc プロジェクトは、cmt_demo_rskrx72m_gcc と同じものです。

5.9 ワークスペースへのデモ追加

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。

5.10 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.5.00)

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 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、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 4.20.3 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.5.00
使用ボード	Renesas Starter Kit+ for RX140（型名：RTK55671xxxxxxxxxx）

表 6.2 動作確認環境 (Rev.4.90)

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 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、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 4.20.3 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.4.90
使用ボード	Renesas Starter Kit+ for RX140（型名：RTK5RX140xxxxxxxxxx）

表 6.3 動作確認環境 (Rev.4.80)

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 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、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 4.20.3 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.4.80
使用ボード	Renesas Starter Kit+ for RX671（型名：RTK55671xxxxxxxxxx）

表 6.4 動作確認環境 (Rev.4.70)

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.4.70
使用ボード	Renesas Starter Kit+ for RX72M（型名：RTK5572Mxxxxxxxxxx）

表 6.5 動作確認環境 (Rev.4.60)

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.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 -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。
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.4.60
使用ボード	Renesas Starter Kit+ for RX72M（型名：RTK5572Mxxxxxxxxxx）

表 6.6 動作確認環境 (Rev.4.50)

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.7.8.0 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 V3.02.00 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 -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。
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.4.50
使用ボード	Renesas Solution Starter Kit for RX23W（型名：RTK5523Wxxxxxxxxx）

表 6.7 動作確認環境 (Rev.4.40)

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 V3.02.00 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	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.4.40
使用ボード	Renesas Solution Starter Kit+ for RX23E-A (product No.: RTK0ESXBxxxxxxxxxx)

表 6.8 動作確認環境 (Rev.4.31)

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.7.7.0 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.4.31
使用ボード	RX13T CPU Card (product No.: RTK0EMXA10C00000BJ) RX72N (product No.: RTK5572Nxxxxxxxxxx)

表 6.9 動作確認環境 (Rev.4.30)

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.4.30
使用ボード	RX13T CPU Card (product No.: RTK0EMXA10C00000BJ) RX72N (product No.: RTK5572Nxxxxxxxxxx)

表 6.10 動作確認環境 (Rev.4.20)

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.4.20
使用ボード	Renesas Starter Kit+ for RX72M（型名：RTK5572Mxxxxxxxxxx）

表 6.11 動作確認環境 (Rev.4.10)

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.4.10
使用ボード	Renesas Solution Starter Kit for RX23W（型名：RTK5523Wxxxxxxxxx）

表 6.12 動作確認環境 (Rev.4.00)

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.4.00
使用ボード	Renesas Starter Kit+ for RX65N-2MB（型名：RTK50565Nxxxxxxxxx）

表 6.13 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 V7.3.0
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	Rev3.40
使用ボード	Renesas Starter Kit for RX72T（型名：RTK5572Txxxxxxxxx）

表 6.14 動作確認環境（Rev.3.31）

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 V7.3.0
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	Rev3.31
使用ボード	Renesas Starter Kit for RX66T（型名：RTK50566T0SxxxxxBE） Renesas Starter Kit+ for RX65N-2M（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

表 6.15 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 V7.0.0
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 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	Rev3.30
使用ボード	Renesas Starter Kit for RX66T（型名：RTK50566T0SxxxxxBE） Renesas Starter Kit+ for RX65N-2M（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

表 6.16 動作確認環境（Rev.3.21）

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	Rev3.21
使用ボード	Renesas Starter Kit+ for RX65N-2M（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

表 6.17 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 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	Rev3.20
使用ボード	Renesas Starter Kit+ for RX65N-2M（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)」
- e² studio を使用している場合
  アプリケーションノート RX ファミリ e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)」

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

(2) Q：本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「This MCU is not supported by the current r_cmt_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_cmt_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_cmt _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

（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
2.90	2015.12.1	—	初版発行
2.91	2016.06.15	14 15	「4. デモプロジェクト」に RSKRX64M を追加 「テクニカルアップデートの対応について」追加
3.00	2016.10.1	— 6 10, 12, 13, 15	FIT モジュールの RX65N グループ対応 コードサイズ表のフォーマットを変更 コードサイズ表に RX65N グループのコードサイズを追加 API 関数のサンプルプログラムの記載を追加
3.10	2017.02.28	— — 5 ﾌﾟﾛｸﾞﾗﾑ	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 ワンショットタイマ動作中、意図しないタイミングで割り込みが発生した場合、CMT 停止中にもかかわらずコンペアマッチカウンタのみ動作中となる場合があったため、タイマを停止させるタイミングを修正 R_CMT_Stop 関数でタイマを停止する時のレジスタの設定順序を見直し R_CMT_Stop 関数で割り込みを禁止する時のレジスタの設定順序を見直し 全チャネルを使用している場合、ワンショットタイマのコールバック関数内で再度ワンショットタイマを設定しようとすると、エラーとなり設定出来ない問題を修正
3.20	2017.07.21	— 5 6 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 「2.13 FIT モジュールをプロジェクトに追加する方法」を更新
3.21	2017.10.31	19 20	RSKRX65N と RSKRX65N-2M を、「4.デモプロジェクト」に追加。 「4.8 デモのダウンロード方法」を追加 「5. 付録」を追加
3.30	2018.09.28	1, 3 1, 5 7 14 15 21	RTOS のサポートを追加 RX66T のサポートを追加 RX66T に対応するコードサイズを追加 R_CMT_Stop(): 説明を変更 R_CMT_Control(): 説明を変更 「6.1 動作確認環境」Rev. 3.30 に対応する表を追加
3.31	2018.11.16	— 21	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. 3.31 に対応する表を追加

3.40	2019.02.01	— 1, 6 7 11-18 21	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。 RX72T グループに対応するコードサイズを追加。 各 API 関数で「Reentrant」の説明を削除。 「6.1 動作確認環境」Rev 3.40 に対応する表を追加。。
3.50	2019.05.01	— 16,17, 19,20 22,23	**新規 API を追加。** R_CMT_CreatePeriodicAssignChannelPriority(); R_CMT_CreateOneShotAssignChannelPriority() R_CMT_Control(): **CMT チャネルの割り込み優先度を取得/設定するコマンドを追加。**
4.00	2019.05.20	— 1 5 6 7 22 25 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 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。 「2.2 ソフトウェアの要求」r_bsp v5.20 以上が必要 「2.4 使用する割り込みベクタ」：RX210、RX631、RX63N の説明を削除。 「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.4.00 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 と IAR コンパイラに関して、以下を変更。 1. R_CMT_GetVersion 関数のインライン展開を削除。 2. 「evenaccess」を、BSP のマクロ定義で置き換えた。 3. 割り込み関数の宣言を、BSP のマクロ定義で置き換えた。
4.10	2019.06.28	1、6 7 21 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.4.10 に対応する表を追加。 RX23W のサポートを追加。
4.20	2019.08.15	1, 6 8 22 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.4.20 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。
4.30	2019.11.25	1, 7 6 9 23 プログラム	RX13T, RX66N, RX72N のサポートを追加。 2.3 制限事項 制限事項を追加。 RX13T, 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.4.30 に対応する表を追加。 RX13T, RX66N, RX72N のサポートを追加。 API 関数のコメントを Doxygen スタイルに変更。 ulTRON のサポートを追加。
4.31	2019.11.29	プログラム	power_on（）および power_off（）の問題を修正。

4.40	2020.03.31	1, 7 10 23 25 プログラム	RX23E-A のサポートを追加。 RX23E-A に対応するコードサイズを追加。 デモプロジェクトの更新と追加 「6.1 動作確認環境」： Rev.4.40 に対応する表を追加。 RX23E-A のサポートを追加。 デモプロジェクトの更新と追加
4.50	2020.05.29	1 25 プログラム	RX23W のチャネル 2, 3 は、BLE FIT モジュールで使用されるため使用できません。 「6.1 動作確認環境」： Rev.4.50 に対応する表を追加。 RX23W のサポート BLE を追加しました。
4.60	2020.06.30	23 25 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.4.60 に対応する表を追加。 デモプロジェクトの更新と追加
4.70	2020.08.31	1 14,16 18,19 29 プログラム	“要旨”を更新 引数“frequency_hz”の範囲値を追加 引数“period_us”の範囲値を追加 「6.1 動作確認環境」： Rev.4.70 に対応する表を追加。 r_cmt_rx_config.h 内の_RI_TRACE_TIMER マクロにかかわる条件を追加 CMT を 2 チャネル持っているデバイスで RI600V4 を使用しているときのワーニングを修正
4.80	2021.03.31	1 4 10 29 プログラム	RX671 のサポートを追加。 「1.3.1CMT 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.4.80 に対応する表を追加。 RX671 のサポートを追加。
4.90	2021.04.15	1,7 10 29 プログラム	RX140 のサポートを追加。 RX140 に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.4.90 に対応する表を追加。 RX140 のサポートを追加。 デモプロジェクトに CS+ のサポートを追加。
5.00	2021.09.13	28 30 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.5.00 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。