RX ファミリ
CMT モジュール
Firmware Integration Technology

R01AN1856JJ0320
Rev. 3.20
2017.07.21

要旨

本モジュールでは、RX MCU のコンペアマッチタイマを使用するための基本的な機能が提供されます。
本ドキュメントは、Firmware Integration Technology (FIT)を使用した CMT モジュールについて説明します。
以降、本モジュールを CMT FIT モジュールと称します。

対象デバイス

本モジュールは以下のデバイスで使用できます。

- RX110 グループ
- RX111 グループ
- RX113 グループ
- RX130 グループ
- RX210 グループ
- RX230 グループ
- RX231 グループ
- RX23T グループ
- RX24T グループ
- RX24U グループ
- RX63N グループ、RX631 グループ
- RX64M グループ
- RX65N グループ、RX651 グループ
- RX71M 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。

関連ドキュメント

- Firmware Integration Technology ユーザーズマニュアル (R01AN1833)
- ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)
- e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)
- CS+に組み込む方法 Firmware Integration Technology (R01AN1826)

目次

1. 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。

1.1 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。

1.2 コールバック関数

コールバック関数の定義は FIT 1.0 の仕様に準じています。

a. コールバック関数では 1 つの引数（void * pdata）を使用する。

b. コールバック関数を呼び出す前に、関数ポインタが有効であることを確認する。最低でも下記は確認すること。

   i. null でない。

   ii. FIT_NO_FUNC マクロと同等のマクロでない。

1.2.1 コールバック関数のプロトタイプ宣言例

コールバック関数はユーザによって提供されます。コールバック関数は、値(void)を返さない通常の C 関数です。また、void へのポインタを示す引数を 1 つ持ちます。以下に宣言を示します。

```
void my_cmt_callback(void * pdata);
```

1.2.2 pdata 引数を逆参照する

ISR がコールバック関数を呼び出すと、割り込みをトリガした CMT 番号を含む値へのポインタを渡します。FIT のコールバックは void 型のポインタを取るため、逆参照ができるようにポインタを型変換する必要があります。CMT チャネル番号は、0～3 の範囲で uint32_t として渡されます。

Example

```
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. API 情報

本アプリケーションノートのサンプルコードは、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

本モジュールを使用するには、ご使用の MCU が CMT 機能をサポートしていることが要求されます。

2.2 ハードウェアリソースの要求

本モジュールは、CMT 外のリソースを必要としません。CMT タイマの範囲と分解能は MCU を設定する周辺クロックによって決定されます。

2.3 ソフトウェアの要求

本モジュールは以下のソフトウェアに依存します。

- このソフトウェアは使用する MCU モデルをサポートするインストール済みの FIT BSP モジュールに依存します。
- 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。

2.4 制限事項

特になし。

2.5 対応ツールチェーン

本モジュールは下記ツールチェーンで動作確認を行っています。

- Renesas RX Toolchain v.2.02.00（RX110、RX111、RX113、RX210、RX231、RX63N、RX64M、RX71M）
- Renesas RX Toolchain v.2.03.00（RX130、RX230、RX23T、RX24T）
- Renesas RX Toolchain v.2.05.00（RX24U、RX651、RX65N）
- Renesas RX Toolchain v.2.06.00（RX24U）
- Renesas RX Toolchain v.2.07.00（RX65N-2MB、RX130-512KB）

2.6 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 CMT FIT モジュールで使用する割り込みベクタには、CMT FIT モジュールで使用する割り込みベクタの一覧を記載しています。

表 2.1 CMT FIT モジュールで使用する割り込みベクタ

デバイス	割り込みベクタ
RX110[*1] RX111[*1] RX113 RX130[*1] RX210 RX230 RX231 RX23T RX24T RX24U RX63N RX631	● 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。	

2.7 ヘッダファイル

すべての API 呼び出しは、本モジュールのプロジェクトで提供される r_cmt_rx_if.h ファイルを取り込むことによってアクセスできます。

r_cmt_rx_config.h ファイルで、ビルド時に設定可能なコンフィギュレーションオプションを選択あるいは定義できます。

2.8 整数型

コードをわかりやすく、また移植が容易に行えるように、本プロジェクトでは ANSI C99 (Exact width integer types (固定幅の整数型)) 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。

2.9 コンパイル時の設定

ビルド時に設定可能なコンフィギュレーションオプションは r_cmt_rx_config.h ファイルに含まれます。下表に各設定の概要を示します。

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

2.10 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「2.9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、「2.52.5対応ツールチェーン」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
MCU x 2 チャネルの場合（ RX110 、 RX111 、 RX130）	ROM	837 バイト	
	RAM	16 バイト	
	最大使用スタックサイズ	64 バイト	
MCU x 4 チャネルの場合（ RX113 、 RX210 、 RX230 、 RX231 、 RX23T 、 RX24T 、 RX24U 、 RX631 、 RX63N 、 RX64M 、 RX651 、 RX65N 、 RX71M）	ROM	1200 バイト	
	RAM	32 バイト	
	最大使用スタックサイズ	64 バイト	

2.11 API データ構造体

本モジュールの API で使用されるデータ構造体について説明します。

2.11.1 特殊なデータ型

強力な型チェックを行い、エラーを減少させるため、API 関数で使用されるパラメータの多くが、提供された型定義での引数を要求します。使用可能な値は、r_cmt_rx_if.h ファイルに定義されます。

2.12 戻り値

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。

2.13 FIT モジュールをプロジェクトに追加する方法

FIT モジュールは、本モジュールを使用するプロジェクトごとに追加する必要があります。ルネサスは、(1)または(3)で説明された「スマート・コンフィグレータ」の使用を推奨します。ただし、「スマート・コンフィグレータ」はすべての RX デバイスをサポートしているわけではありません。サポートしていない RX デバイスについては、(2)または(4)の方法で対応してください。

(1) e² studio の「スマート・コンフィグレータ」を使用して FIT モジュールをプロジェクトに追加する方法
e² studio の「スマート・コンフィグレータ」を使うと、FIT モジュールは自動的にプロジェクトに追加されます。詳しくは、「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(2) e² studio の「FIT コンフィグレータ」を使用して FIT モジュールをプロジェクトに追加する方法
e² studio の FIT コンフィグレータ」を使うと、FIT モジュールは自動的にプロジェクトに追加されます。詳しくは、「FIT モジュールをプロジェクトに追加する方法(R01AN1723)」を参照してください。

(3) CS+の「スマート・コンフィグレータ」を使用して FIT モジュールをプロジェクトに追加する方法
CS+の「スタンドアロン版スマート・コンフィグレータ」を使うと、FIT モジュールは自動的にプロジェクトに追加されます。詳しくは、「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(4) FIT モジュールを CS+のプロジェクトに追加する方法
CS+では、FIT モジュールを手動でプロジェクトに追加してください。詳しくは、「FIT モジュールを CS+のプロジェクトに追加する方法(R01AN1826)」を参照してください。

3. API 関数

3.1 概要

本モジュールには以下の関数が含まれます。

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。

3.2 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）(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

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_CreatePeriodic 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、呼び出しで指定された周波数でコンペアマッチが生成されるように設定されます。

Reentrant

不可

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:

- 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。

3.3 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）。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。

Reentrant

不可

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

3.4 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。

Reentrant

この関数は再入可能（リエントラント）です。

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

3.5 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

handle

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

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。

Reentrant

この関数は再入可能（リエントラント）です。

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

3.6 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。

Reentrant

この関数は再入可能（リエントラント）です。

Example

```
/* 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. デモプロジェクト

デモプロジェクトはスタンドアロンプログラムです。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。

4.1 cmt_demo_rskrx113

cmt_demo_rskrx113 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。

4.2 cmt_demo_rskrx231

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

4.3 cmt_demo_rskrx64M

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

4.4 cmt_demo_rskrx71m

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

4.5 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。

テクニカルアップデートの対応について

本モジュールは以下のテクニカルアップデートの内容を反映しています。

- 対応しているテクニカルアップデートはありません。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

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	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
		ﾌﾟﾛｸﾞﾗﾑ	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

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. 未使用端子の処理
   【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】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. リザーブアドレス（予約領域）のアクセス禁止
   【注意】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、将来の機能拡張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。
4. クロックについて
   【注意】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。
5. 製品間の相違について
   【注意】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、内部ROM、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、ノイズ輻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、その他の不適切に使用しないでください。かかる改造、改変、複製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、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、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できません。たとえ、意図し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。また、当社製品および技術を、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル（無人航空機を含みます。）の開発、設計、製造、使用もしくは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、かつ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。

   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. お客様の転売、貸与等により、本書（本ご注意書きを含みます。）記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は一切その責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
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. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。

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その総株主の議決権の過半数を直接または間接に保有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.3.0-1 2016.11)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/