

RX ファミリ

CMT モジュール

Firmware Integration Technology

R01AN1856JJ0320
Rev. 3.20
2016.12.16

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
- 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)
- e² 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	3
1.1	CMT FIT モジュールを使用する	3
1.2	コールバック関数.....	3
1.2.1	コールバック関数のプロトタイプ宣言例	3
1.2.2	pdata 引数を逆参照する.....	4
2.	API 情報.....	5
2.1	ハードウェアの要求	5
2.2	ハードウェアリソースの要求	5
2.3	ソフトウェアの要求	5
2.4	制限事項	5
2.5	対応ツールチェーン	5
2.6	ヘッダファイル	5
2.7	整数型.....	5
2.8	コンパイル時の設定	6
2.9	コードサイズ	6
2.10	API データ構造体	6
2.10.1	特殊なデータ型	6
2.11	戻り値.....	6
2.12	FIT モジュールの追加方法	7
3.	API 関数.....	8
3.1	概要	8
3.2	R_CMT_CreatePeriodic().....	9
3.3	R_CMT_CreateOneShot()	11
3.4	R_CMT_Stop()	13
3.5	R_CMT_Control()	14
3.6	R_CMT_GetVersion().....	17
4.	デモプロジェクト.....	18
4.1	cmt_demo_rskrx113.....	18
4.2	cmt_demo_rskrx231.....	18
4.3	cmt_demo_rskrx64M.....	18
4.4	cmt_demo_rskrx71m.....	18
4.5	ワークスペースにデモを追加する	18

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 チャンネルの 16 ビットタイマです。

各チャンネルには、プリスケアラ、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 チャンネルですが、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 チャンネルを 2、または 4 チャンネル持つことになります。

本モジュールは、CMT チャンネルの作成および開始、チャンネルの一時停止および再開、チャン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。タイマのコンペア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 チャンネル番号は、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 (RX65N)

2.6 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。

2.7 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。

2.8 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

2.9 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、「2.52.5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 チャンネルの場合 (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 チャンネルの場合 (RX113、RX210、RX230、RX231、RX23T、RX24T、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	

2.10 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。

2.10.1 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 関数で使用す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。

2.11 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。

2.12 FIT モジュールの追加方法

本モジュールは、e² studio で、使用するプロジェクトごとに追加する必要があります。

プロジェクトへの追加方法は、FIT プラグインを使用する方法と、手動で追加する方法があります。

FIT プラグインを使用すると、簡単にプロジェクトに FIT モジュールを追加でき、またインクルードファイルパスも自動的に更新できます。このため、プロジェクトへ FIT モジュールを追加する際は、FIT プラグインの使用を推奨します。

FIT プラグインを使用して FIT モジュールを追加する方法は、アプリケーションノート「e² studio に組み込む方法(R01AN1723)」の「3. FIT プラグインを使用して FIT モジュールをプロジェクトに追加する方法」を参照してください。

FIT プラグインを使用せず手動で FIT モジュールを追加する方法は、「4. 手作業で FIT モジュールをプロジェクトに追加する方法」を参照してください。

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。

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 チャンネルを検索し、適切なプリスケア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 チャンネルを終了し、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 チャンネルを検出し、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 チャンネルに最も適したプリスケア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 チャンネルを検出し、呼び出し元にそのチャン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ンネルを事前に割り当てずとも、複数の CMT チャン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ン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 チャンネルがない、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 チャンネルを検出し、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

本関数は同一 CMT チャンネルに対して再入動作を行いません。ただし、別の CMT チャンネルに対しては再入可能 (リエントラント) です。

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:

- 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。そうすると、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 チャンネルを検出し、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。最初の割り込みとコールバックの処理後、タイマは停止されます。

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 チャンネルに最も適したプリスケ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 チャンネルを検出し、呼び出し元にそのチャン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ンネルを事前に割り当てずとも、複数の CMT チャン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ン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 チャンネルがない、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 チャンネルを検出し、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

本関数は同一 CMT チャンネルに対して再入動作を行いません。ただし、別の CMT チャンネルに対しては再入可能（リエントラント）です。

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

3.4 R_CMT_Stop()

CMT チャンネルを停止し、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 タイマのチャンネル

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 チャンネルを開放します。

開放された CMT チャンネルは `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 タイマのチャンネル停止の例です。

```
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 チャンネルを制御し、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 チャンネル番号

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 チャン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:
使用可能な総チャンネル数を返します。

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;  
  
sprintf(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 チャンネルを使った 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ン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/>

お問い合わせ先

<http://japan.renesas.com/contact/>

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。

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
3.20	2016.12.16	—	RX65N-2M に対応

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、電氣的特性の範囲で、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、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したのですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 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に関し、当社は、何らの責任を負うものではありません。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. 当社製品を改造、改変、複製等しないでください。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）、交通用信号機器、
防災・防犯装置、各種安全装置等
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（原子力制御システム、軍事機器等）に使用されることを意図しておらず、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お客様または第三者に損害が生じて、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、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。お客様が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. 本資料に記載されている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大量破壊兵器の開発等の目的、軍事利用の目的その他軍事事務に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き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、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
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。

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と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。



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：<http://japan.renesas.com/contact/>