

RX ファミリ

コンペアマッチタイマ (CMT)モジュール Firmware Integration Technology

R01AN1856JU0230
Rev.2.30
2014.07.10

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。

このモジュールは、RX MCU 上でコンペアマッチタイマ (CMT) を使用するための基本関数を提供します。

動作確認デバイス

以下は、この API によって現時点でサポートできるデバイスの一覧です。

- RX110 グループ
- RX111 グループ
- RX210 グループ
- RX63N 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 (R01AN1833JU)
- ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685JU)
- e² studio に組み込む方法 Firmware Integration Technology (R01AN1723JU)
- CubeSuite+に組み込む方法 Firmware Integration Technology (R01AN1826JJ)

目次

1. 概要	2
2. API 情報.....	3
3. API 関数.....	5

1. 概要

このソフトウェアモジュールは、RX コンペアマッチタイマ (CMT) 周辺回路へ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、プリスケアラと 16 ビットのコンペアレジスタを備えたフリーランニングカウンタが搭載されています。フリーランニングカウンタがコンペアレジスタに一致したときに割り込みを発生させることができます。コンペアイベントの発生時にカウンタは自動的にリセットされて再開されるため、RTOS スケジューラのように何度も繰り返すソフトウェアイベントを整調するのに理想的なタイマです。CMT ユニットには、2 つのチャンネルが内蔵されています。RX MCU には 1 つまたは 2 つの CMT ユニットが搭載されているため、合計で 2 つまたは 4 つの独立した CMT チャンネルが含まれることになります。

このドライバには、CMT チャンネルの作成と開始、チャンネルの一時停止と再開、およびチャンネルの終了の各関数が用意されています。ユーザアプリケーションコードは、コールバック関数を介して呼び出すことができます。

1.1 FIT CMT モジュールの使用

CMT モジュールの主な使用目的は、繰り返しイベントと固定の時間間隔を簡単に生成できるようにすることです。

CMT モジュールをプロジェクトに追加した後、`r_cmt_rx_config.h` ファイルを変更し、インストール内容に合わせてソフトウェアを設定する必要があります。

関数 `R_CMT_CreatePeriodic` および `R_CMT_CreateOneShot` を使用してタイマを開始します。引数としてコールバック関数へのポインタを提供すると、タイマの期限が切れたときにコールバックが呼び出されます。コールバックの実行中は、割り込みはデフォルトで無効にされることに注意してください。コールバックは、ISR という枠の中で実行されているからです。このため、コールバック関数は、迅速に完了できるようにサイズを小さくすることを推奨します。

理論的には、CMT タイマの最大クロック速度は、 $PCLK/8$ に限定されます。周期タイマ関数を使用してクロックを生成するときには、割り込みとコールバックの処理にある程度の時間がかかることに留意してください。このため、生成可能な最大周波数を限定しています。ここには極めて多くの変数があります。

1.2 コールバック関数

コールバックの定義は、以下のように FIT 1.0 仕様のルールに従っています。

- コールバック関数は 1 つの引数を取ります。この引数は、「`void *pdata`」です。
- 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チェックします。少なくとも、ポインタが `null` でないこと、および `FIT_NO_FUNC` マクロに等しくないことをチェックします。

1.2.1 コールバック関数のプロトタイプ宣言の例

ユーザ専用のコールバック関数を提供する必要があります。コールバック関数は、値 (`void`) を返さない標準 C 関数であり、以下の宣言のように、`void` へのポインタである 1 つのパラメータを備えています。

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

1.2.2 pdata 引数の参照解除

ISR がコールバック関数を呼び出すと、割り込みを引き起こした CMT チャンネル番号を含んだ値にポインタを渡します。FIT コールバックは `void` ポインタを取得するため、参照解除できるようにポインタを型キャストする必要があります。CMT チャンネル番号は、0~3 の範囲の整数定数として渡されるので、ポインタを参照解除するときに任意の整数型にキャストすることができます。

使用例

```
void my_cmt_callback(void * pdata)
{
    uint8_t cmt_event_channel_number;

    cmt_event_channel_number = *((uint8_t *)pdata); //cast pointer to uint8_t
    ...
}
```

2. API 情報

このドライバの API は、ルネサスの API 命名基準に従っています。

2.1 ハードウェア要件

このドライバでは、CMT 周辺回路を搭載した RX MCU が必要です。

2.2 ハードウェアリソース要件

このドライバは、CMT 以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 の周辺クロック設定値によって決まります。

2.3 ソフトウェア要件

このドライバは、以下のソフトウェアのサポートに依存しています。

- このソフトウェアは、使用中の MCU モデルをサポートする、FIT 対応の BSP モジュールがあるかどうか
に依存しています。
- CMT を開始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

なし

2.5 サポートされているツールチェイン

このドライバは、以下のツールチェインでテストと動作確認を行っています。

- ルネサス RX ツールチェイン v1.00 以降

2.6 ヘッダファイル

すべての API 呼び出しはこのソフトウェアのプロジェクトコードとして提供されている 1 個のファイル「r_cmt_rx_if.h」をインクルードすることによって行われます。

ビルドタイムのコンフィグレーションオプションは、ファイル「r_cmt_rx_config.h」で選択または定義されます。

2.7 整数型

このプロジェクトでは、コードをわかりやすく、移植性をより大きくするために、ANSI C99 の固定長整数型 (exact width integer type) 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 コンフィグレーションの概要

ソフトウェアの一部の機能や動作は、コンフィグレーションオプションによってビルド時に決定されます。このオプションは、ユーザが選択する必要があります。

表 1 CMT モジュールのコンフィグレーションオプションのリスト

r_cmt_rx_config.h にあるコンフィグレーションオプション		
CMT_RX_CFG_IPR	(5)	CMT 割り込みで使用される割り込み優先レベル

2.9 API のデータ構造

このセクションでは、ドライバの API 関数で使用するデータ構造について詳しく説明します。

2.9.1 特殊データ型

強力な型チェックを行ってエラーを低減するため、API 関数で使用する多くのパラメータは、提供された型定義を使用して引数を渡す必要があります。許容可能な値はパブリックインタフェースファイル `r_cmt_if.h` に定義されています。

2.10 戻り値

すべての CMT 関数は、呼び出しが成功か失敗かを示すブール値を返します。

2.11 プロジェクトへのモジュールの追加

既存の e2Studio プロジェクトにドライバを追加する必要があります。インクルードファイルパスが自動的に更新されるため、e2Studio FIT プラグインを使用してドライバをプロジェクトに追加するのが最良の方法です。または、このアプリケーションノートに付随するアーカイブからドライバをインポートして、以下の手順に従って手動で追加することもできます。

1. このアプリケーションノートは zip ファイルパッケージで配布されており、この中のフォルダ `r_cmt_rx` に FIT CMT サポートモジュールが含まれています。
2. パッケージを任意の場所に解凍します。
3. ファイルブラウザウィンドウで、配布パッケージを解凍したディレクトリに移動して `r_cmt_rx` フォルダを見つけます。
4. e2Studio ワークスペースを開きます。
5. e2Studio プロジェクトのエクスプローラウィンドウで、CMT モジュールを追加するプロジェクトを選択します。
6. ブラウザウィンドウからプロジェクトの最上位レベルの e2Studio プロジェクトに `r_cmt_rx` フォルダをドラッグアンドドロップします（またはコピーして貼り付けます）。
7. 以下に示すように、パスをモジュールファイルに追加することにより、プロジェクトのソースサーチ/インクルードパスを更新します。
 - a. 「Add directory path」コントロールに移動します。
 - i. プロジェクト名 -> Properties -> C/C++ Build -> Setting -> Compiler -> Source - Add（緑色の+アイコン）
 - b. 以下のパスを追加します。
 - i. "\${workspace_loc}/\${ProjName}/r_cmt_rx}"
 - ii. "\${workspace_loc}/\${ProjName}/r_cmt_rx/src}"

プラグインを使用する場合でも、あるいはプロジェクトにパッケージを手動で追加する場合でも、アプリケーション用のドライバを設定する必要があります。

8. プロジェクトの `r_cmt_rx/ref/` ソースフォルダで `r_cmt_rx_config_reference.h` ファイルを見つけて、これをプロジェクトの `r_config` フォルダにコピーします。
9. `r_config` フォルダにコピーしたファイルの名前を `r_cmt_rx_config.h` に変更します。
10. コピーした `r_cmt_rx_config.h` ファイルを編集して必要なコンフィグレーション設定を作成します。「コンフィグレーションの概要」を参照してください。

CMT サポートモジュールは、特定の MCU 情報とサポート関数について `r_bsp` パッケージを使用します。`r_bsp` パッケージは、`r_bsp` フォルダにある `platform.h` ヘッダファイルにより簡単に設定されます。`r_bsp` パッケージを設定するには、`platform.h` を開いて、使用しているボードの `#include` コメントを外します。例えば、RSKR111 ボード上でデモを実行するには、「`./board/rskrx111/r_bsp.h`」マクロの `#include` コメントを外し、他のすべてのボードの `#include` は確実にコメントアウトします。

3. API 関数

3.1 まとめ

この設計には、次の関数が含まれています。

関数	説明
R_CMT_CreatePeriodic()	使用されていない CMT チャンネルを見つけて、適切なプリスケアラを設定することで任意の周期にタイマを設定し、ユーザのコールバック関数をタイマの割り込みに関連付けて、タイマを開始します。ユーザが停止するまでタイマは動作を継続し、割り込みを生成し、任意の周波数でコールバックを呼び出します。
R_CMT_CreateOneShot()	R_CMT_CreatePeriodic と似ていますが、最初の割り込みとコールバックの後、タイマは停止します。
R_CMT_Control()	一時停止、再開、ステータスの報告をタイマに命令します。
R_CMT_Stop()	CMT チャンネルをオフにして割り込みを無効にし、使用中でなければ CMT ユニットの電源を切ります。

3.2 R_CMT_CreatePeriodic

この関数は、使用されていない CMT チャンネルを見つけ、要求された周波数にこのチャンネルを設定し、ユーザのコールバック関数をタイマの割り込みに関連付けて、電源を入れてタイマを開始します。

フォーマット

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

パラメータ

frequency_hz

任意の周波数 (Hz) ^{【注】 1.}。タイマの範囲と分解能は、周辺クロックの設定値によって決まります。CMT チャンネルに最適なプリスケーラがドライバによって選択されます。

callback

ユーザのコールバック関数へのポインタ。これは、単一の void *引数でなければなりません。

channel

CMT FIT モジュールは、使用されていない最初の CMT チャンネルを見つけてこれを呼び出し側に割り当てます。これにより、複数のドライバが CMT ドライバを使用できるようになり、すべてのタイマチャンネルをあらかじめ割り当てておく必要がなくなります。この引数を使えば、ドライバは、割り当てられているチャンネルを呼び出し側に示すことができるようになります。

戻り値

true: 成功。CMT は初期化されました。

false: 空いている CMT チャンネルがないか、あるいは無効な設定値です。

プロパティ

ファイル「r_cmt_rx_if.h」にプロトタイプ宣言されています。

説明

R_CMT_CreatePeriodic 関数は、未使用の CMT チャンネルを見つけてこれを呼び出し側に割り当て、コンペアマッチイベントの発生時に呼び出すユーザの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でコンペアマッチを生成するよう設定されています。

リエントラント

この関数は、同じ CMT チャンネルに対して再入動作が可能ないように設計されていません。ただし、異なるチャンネルに対しては再入動作が可能です。

【注】 1. 最大周期周波数

ハードウェアでは CMT タイマの最大クロック速度は PCLK/8 に限定されています。ただし、周期タイマ関数を使用してクロックを生成するときには、割り込みとコールバックの処理にある程度の時間がかかることに留意してください。要求周波数が増大するにつれて、割り込みとコールバックの処理に必要なプロセッサの時間の割合が増大します。ある点に達すると、多くの時間が費やされて他の有用な作業を行うための時間がなくなります。このため、これによって生成可能な最大周波数が限定されます。最大実用周波数は、ユーザのシステム設計に依存しますが、通常、数キロヘルツ以下の周波数が適度な周波数になります。

3.3 R_CMT_CreateOneShot

この関数は、未使用の CMT チャネルを検出し、要求された期間についてこれを設定し、ユーザのコールバック関数をタイマの割り込みに関連付けた後、電源を入れてタイマを開始します。タイマは、最初の割り込みとコールバックの後に停止されます。

フォーマット

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

パラメータ

period_us

任意の周期（マイクロ秒単位）。タイマの範囲と分解能は、周辺クロックの設定値によって決まります。CMT チャネルに最適なプリスケラがドライバによって選択されます。

callback

ユーザのコールバック関数へのポインタ。これは、単一の void *引数でなければなりません。

channel

CMT FIT モジュールは、使用されていない最初の CMT チャネルを見つけてこれを呼び出し側に割り当てます。これにより、複数のドライバが CMT ドライバを使用できるようになり、すべてのタイマチャネルをあらかじめ割り当てておく必要がなくなります。この引数を使えば、ドライバは、割り当てられているチャネルを呼び出し側に示すことができるようになります。

戻り値

true: 成功。CMT は初期化されました。

false: 空いている CMT チャネルがないか、あるいは無効な設定値です。

プロパティ

ファイル「r_cmt_rx_if.h」にプロトタイプ宣言されています。

説明

R_CMT_CreateOneShot 関数は、未使用の CMT チャネルを見つけてこれを呼び出し側に割り当て、コンペアマッチイベントの発生時に呼び出すユーザの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期の後にコンペアマッチを生成するよう設定されています。タイマは、コンペアマッチイベントが 1 回発生すると停止します。

リエントラント

この関数は、同じ CMT チャネルに対して再入動作が可能ないように設計されていません。ただし、異なるチャネルに対しては再入動作が可能です。

3.4 R_CMT_Stop()

CMT チャンネルを停止し、可能なら CMT ユニットの電源を切ります。

フォーマット

```
bool R_CMT_Stop(uint32_t channel);
```

パラメータ

channel 停止する CMT タイマのチャンネル

戻り値

true: 成功。CMT は終了しました。

false: CMT チャンネルは終了できませんでした。あるいは開かれていませんでした。

プロパティ

ファイル「r_cmt_rx_if.h」にプロトタイプ宣言されています。

説明

この関数は、チャンネルへの割り当てを解除して割り当てられた割り込みを無効にすることで CMT チャンネルを開放します。CMT チャンネルは、R_CMT_CreatePeriodic または R_CMT_CreateOneShot のいずれかの関数で再び開かれていない限り、再度使用することはできません。開かれていない状態の CMT チャンネルに対してこの関数が呼び出された場合、エラーコードが返されます。

リエントラント

再入可能 (リエントラント)

3.5 R_CMT_Control

この関数は、CMT チャンネルを制御して監視するさまざまな方法を提供します。

フォーマット

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

パラメータ

channel	制御する CMT チャンネル番号
command	以下を実行するコマンド CMT_RX_CMD_IS_CHANNEL_COUNTING CMT_RX_CMD_PAUSE CMT_RX_CMD_RESUME CMT_RX_CMD_RESTART CMT_RX_CMD_GET_NUM_CHANNELS

戻り値

true:	コマンドは正常に完了しました。pdata をチェックしてください。
false:	コマンドは正常に完了しませんでした。

プロパティ

ファイル「r_cmt_rx_if.h」にプロトタイプ宣言されています。

説明

この関数は、以下に示す多くのコマンドを提供します。

CMT_RX_CMD_IS_CHANNEL_COUNTING は、CMT チャンネルが現在動作しているかどうかを通知します。
*pdata をチェックしてください。

CMT_RX_CMD_PAUSE は、タイマを終了せずに（電源をオフにせずに）一時停止します。

CMT_RX_CMD_RESUME は、カウンタをゼロにリセットすることなく、一時停止したタイマを再開します。

CMT_RX_CMD_RESTART は、カウンタをゼロにリセットした後、一時停止したタイマを再開します。

CMT_RX_CMD_GET_NUM_CHANNELS は、使用可能なチャンネルの総数を返します。

リエントラント

再入可能（リエントラント）

3.6 R_CMT_GetVersion()

この関数は、ランタイム時にドライバのバージョン番号を返します。

フォーマット

```
uint32_t R_CMT_GetVersion(void);
```

パラメータ

なし

戻り値

バージョン番号。メジャーバージョンとマイナーバージョンの数値が単一の 32 ビット値に格納されています。

プロパティ

ファイル「r_cmt_rx_if.h」にプロトタイプ宣言されています。

説明

この関数は、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、先頭の 2 バイトがメジャーバージョン番号で末尾の 2 バイトがマイナーバージョン番号です。

リエントラント

使用例

この関数の使用例を示します。

```
/* Retrieve the version number and convert it to a string. */

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);
```

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/>

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
2.00	2013.11.06	—	初版 GSCE 発行
2.10	2013.12.20	—	CMCOR 値の式を修正。
2.30	2014.07.10	1	追加の MCU のサポートを表示するように更新
		2	コールバック関数に関するセクション 1.2 を追加
		6	最大周期周波数に関する注記を追加

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. 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、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い合わせください。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用途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

■営業お問い合わせ窓口

<http://www.renesas.com>

※営業お問い合わ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 〒100-0004 千代田区大手町2-6-2（日本ビル）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問い合わせ窓口：<http://japan.renesas.com/contact/>