RX63N グループ
シンプルタイマ･フレームワーク

R01AN0952JU0100
Rev.1.00
2013.02.08

はじめに

このアプリケーションノートはシンプルタイマ･フレームワークの特徴と機能を紹介するものです。本アプリケーションノートに付属するサンプルデモプロジェクトは RSK+RX63N 上で実行されるものですが、このシンプルタイマ･フレームワークはどのハードウェアプラットフォーム上でも実行することができます。

対象デバイス

RX63N グループ

目次

1. 概要

シンプルタイマ･フレームワークはコールバック機能を備えた連結リストのタイマです。その目的は、将来のイベントをスケジュールし、イベントが発生する際に必要な動作を行うことです。フレームワークは非常に柔軟性があり、大半のハードウェアプラットフォームで使用することができます。ユーザアプリケーションのすべてのモジュールまたはソフトウェア要素でシンプルタイマ･フレームワークのサービスを使用することができます。フレームワークに設定できるタイマ数に制限はありません。

2. シンプルタイマ･フレームワーク

この章ではシンプルタイマ･フレームワークの詳細を説明します。フレームワークの特徴と制約、およびフレームワークをユーザアプリケーションに導入するための要件が記されています。

2.1 要件

シンプルタイマ･フレームワークはハードウェアタイマによる周期的なタイムティックを必要とします。任意のシンプルタイマ周辺モジュールをこの目的のために使用できます。ルネサス MCU は TMR や CMT のようにタイムティックを生成できる複数のタイマ周辺モジュールを備えています。ティックの周期はユーザアプリケーションの要件により異なります。

シンプルタイマ･フレームワークをユーザアプリケーションに組み込む際に必要とされる 2 個のファイルは simple_timer.c および simple_timer.h です。ソースファイル.C にはシンプルタイマ･フレームワーク API のコードの本体が、ヘッダファイル.H にはフレームワークで使用する API と型定義が含まれています。シンプルタイマ･フレームワークで使用するメモリは 200 バイト未満の ROM と 10 バイト未満の RAM です。

ユーザアプリケーションで定義され使用されるタイマオブジェクトの数は利用可能な RAM の容量によってのみ制限されます。フレームワークのサービスを使用するモジュールおよびソフトウェア要素では正しく API を呼び出すためにヘッダファイル.H をインクルードしなければなりません。個々のシンプルタイマ･フレームワークのオブジェクトは 36 バイトの RAM を必要とします。

2.2 内部構造

シンプルタイマ･フレームワークのオブジェクト型の構造体は図 1 に示されています。name フィールドはタイマの名前を ASCII 文字列で格納しています。is_active フィールドはタイマが開始したか否かを示しています。start_time と stop_time フィールドは、それぞれの名前のとおり開始時刻と停止時刻を格納します。callback_func と param はタイマが完了したときに登録された関数を呼び出すために使用されます。このコールバック関数には param が引数として渡されます。コールバック関数はユーザアプリケーションの要件に応じてリスト内のタイマごとに異なる関数を指定できます。フィールド*prev と*next はタイマを連結するために使われます。

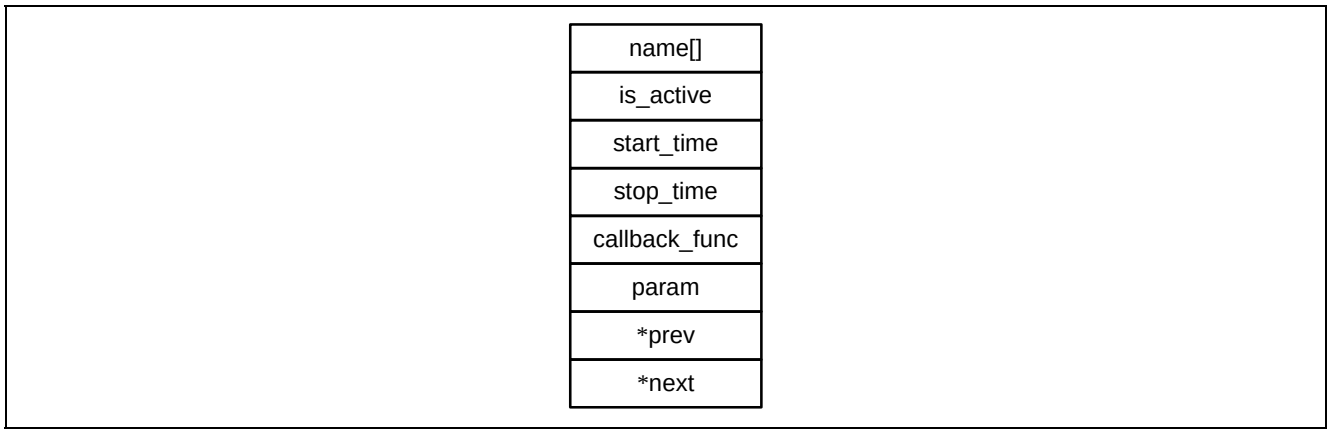

図 1　シンプルタイマ型

有効なタイマは stop_time の値によってソートされ、stop_time がもっとも小さなタイマが list_head ポインタによって指し示されています。図 2 はこの状態を示しており、図示のために stop_time と連結ポインタのみが示されています。

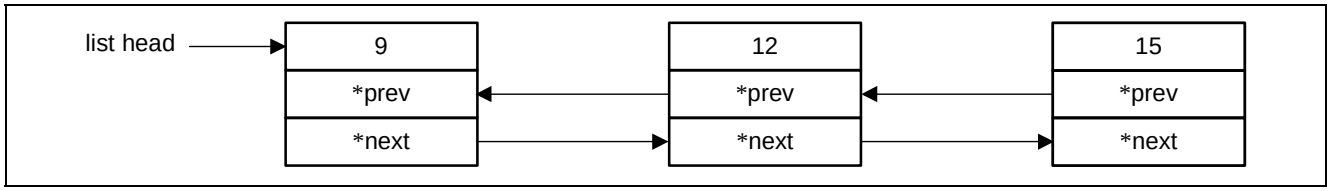

図 2　タイマの連結リスト

新しいタイマは stop_time の値をリスト内のタイマの stop_time の値との比較で検索し、連結ポインタの値を変更することで連結リストに組み込まれます。図 3 は stop_time の値が 13 の新しいタイマが連結リストに挿入される場合を示しています。

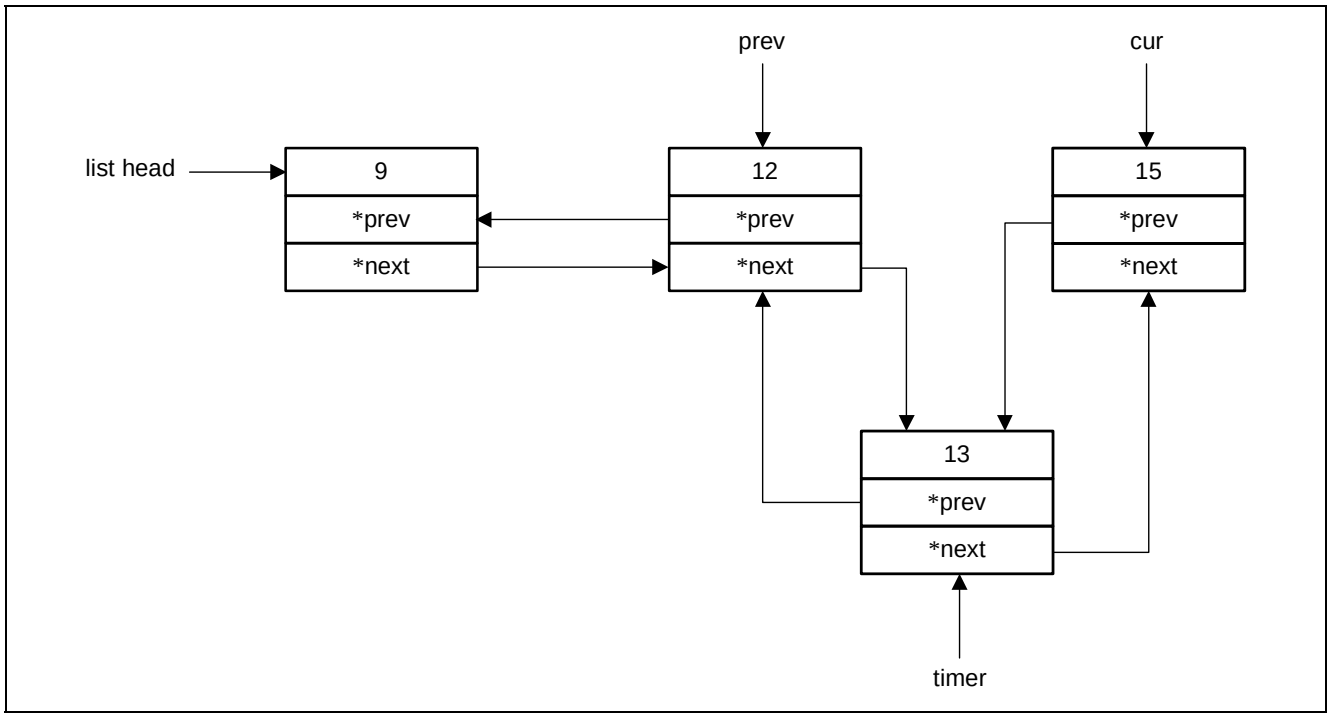

図 3　新しいタイマの挿入

シンプルタイマ･フレームワーク内のタイムティック値がタイマの stop_time の値以上になると、そのタイマは完了します。このとき、そのタイマはリストから外され、登録されていたコールバック関数が param 引数を伴って呼び出されます。シンプルタイマ･フレームワークは有効なタイマをキャンセルする機能を提供します。キャンセルされたタイマは必要に応じて連結ポインタを変更し連結リストから外されます。

3. サンプルデモプロジェクト

本アプリケーションノートでは RSK+RX63N 上で動作するサンプルデモプロジェクトを提供しています。しかし、シンプルタイマ･フレームワークはどのようなハードウェアプラットフォーム上でも実行することができます。この章ではサンプルデモプロジェクトに関して説明します。

3.1 デモプロジェクトのディレクトリ構成

図 4 はシンプルタイマ･フレームワークのデモプロジェクトのディレクトリ構成を示しています。bsp フォルダにはルネサスボードに固有のソースコードが格納されています。シンプルタイマ･フレームワークのソースファイルはフレームワークのテストを行うデバッグコンソールおよびタイムティックを生成する TMR モジュールと共に code フォルダに入っています。

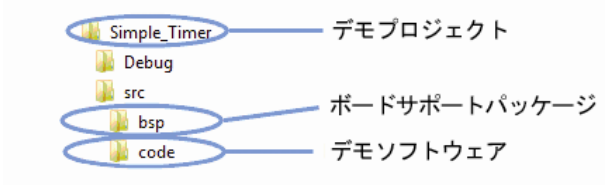

図 4　シンプルタイマフレームワークのデモのディレクトリ構成

3.2 デモプロジェクトのセットアップ

RSK+RX63N ボードの初期化は bsp フォルダ内のファイルで行われます。ソースコード main.c は LED、LCD、および押しボタンスイッチの初期化を行います。サンプルデモプロジェクトでは TMR モジュールによって生成される 1 秒間隔のタイムティックを使用しており、この初期化は main.c ファイル内で行われています。シンプルタイマ･フレームワークも main.c で初期化されます。

サンプルデモプロジェクトは、有効なタイマのリストをスタート、キャンセルおよびダンプする簡単なコマンドの送受信のために HEW デバッグコンソールを使用しています。HEW デバッグコンソールは RX63N のシリアルポートリソースをまったく使用しないため、非常に便利です。HEW デバッグコンソールはルネサス E1 デバッグ接続を使用しており、これ以外のケーブルや USB-シリアル変換器を必要としません。

以下がシリアルテストコマンドのメッセージフォーマットです。

タイマの開始:　　s<space><timerid><space><duration><CR>

タイマのキャンセル:　　c<space><timerid><CR>

タイマリストのダンプ:　d<CR>

サンプルデモプロジェクトではシンプルタイマ･フレームワークがどのように動作するかを示すために 3 個のタイマを使用しています。以下はデバッグコンソールからのサンプル出力の表示の一例です。

```
d
dumping timer list -- current time(4)
name         start time   stop time    callbackFn   param
==========   ==========   ==========   ==========   ==========

s 1 30
start timer: timer1 for 30 seconds -- Success

s 0 10
start timer: timer0 for 10 seconds -- Success
```

```
d
dumping timer list -- current time(26)
name        start time  stop time   callbackFn  param
==========  ==========  ==========  ==========  ==========
    timer0          23          33  0xfff8109b  0x00000000
    timer1          13          43  0xfff8109b  0x00000001

*** simple timer test callback: timer0 expired -- current time(33) ***

*** simple timer test callback: timer1 expired -- current time(43) ***

d
dumping timer list -- current time(47)
name        start time  stop time   callbackFn  param
==========  ==========  ==========  ==========  ==========
```

最初はリスト内に有効なタイマは存在しません。先ずタイマ 1 が 30 秒間として開始され、次にタイマ 0 が 10 秒間で開始されています。有効なタイマのリストでは動作中のタイマが停止時刻の順にソートされて表示されます。このリストにはコールバック関数とそのパラメータも併せて表示されます。タイマが完了すると、コールバック関数が呼び出され、そのタイマはリストから外されます。

```
s 0 100
start timer: timer0 for 100 seconds -- Success

s 1 200
start timer: timer1 for 200 seconds -- Success

s 2 300
start timer: timer2 for 300 seconds -- Success

d
dumping timer list -- current time(595)
name        start time  stop time   callbackFn  param
==========  ==========  ==========  ==========  ==========
    timer0         582         682  0xfff8109b  0x00000000
    timer1         588         788  0xfff8109b  0x00000001
    timer2         594         894  0xfff8109b  0x00000002

c 1
cancel timer: timer1 -- Success

d
dumping timer list -- current time(605)
name        start time  stop time   callbackFn  param
==========  ==========  ==========  ==========  ==========
    timer0         582         682  0xfff8109b  0x00000000
    timer2         594         894  0xfff8109b  0x00000002
```

この例ではタイマ 0、1、および 2 がそれぞれ 100 秒、200 秒、300 秒の時間として開始されています。リストには 3 個のタイマすべてが有効なことが示され、停止時刻でソートされています。次のコマンドではタイマ 1 をキャンセルし、2 個目のリストでその後の状態を表示しています。

4. シンプルタイマ･フレームワーク API

以下はシンプルタイマ･フレームワークの API となる関数です。この章では個々の API 関数について説明します。

```
st_status_code_t    simple_timer_init (void);
st_status_code_t    simple_timer_init_timer (uint8_t const *name, st_timer_t *timer);
st_status_code_t    simple_timer_start_timer (st_timer_t *timer,
                                              uint32_t duration,
                                              st_callback_func_t callback_func,
                                              uint32_t param);
st_status_code_t    simple_timer_cancel_timer (st_timer_t *timer);
uint32_t            simple_timer_get_time (void);
st_status_code_t    simple_timer_set_time (uint32_t time);
st_timer_t          *simple_timer_get_head (void);
void                simple_timer_increment_time (void);
void                simple_timer_check_timer_expiry (void);
```

4.1 simple_timer_init

この API はシンプルタイマ･フレームワークの初期化を行います。

形式

```
st_status_code_tsimple_timer_init(void);
```

パラメータ

なし

戻り値

ST_SUCEESS=正常終了

ST_SUCCESS 以外の値はエラーが生じたことを示しています。

所在

プロトタイプはファイル simple_timer.h に、

関数の本体はファイル simple_timer.c に存在します。

説明

この API はシンプルタイマ･フレームワークの変数を初期化します。フレームワークを使用する前にこの関数を呼び出さなければなりません。

4.2　simple_timer_init_timer

この API はシンプルタイマ･フレームワークのタイマを初期化します。

形式

```
st_status_code_t simple_timer_init_timer (uint8_t const *name,
                                          st_timer_t *timer)
```

パラメータ

`name`

タイマの名前を示す ASCII 文字列。

`timer`

初期化するタイマオブジェクトのポインタ。

戻り値

ST_SUCEESS=正常終了

ST_SUCCESS 以外の値はエラーが生じたことを示しています。

所在

プロトタイプはファイル simple_timer.h に、

関数の本体はファイル simple_timer.c に存在します。

説明

この API はシンプルタイマ･フレームワークのインスタンス化されたタイマオブジェクトを初期化します。API はこのタイマへのポインタとタイマの名前となる ASCII 文字列を使用します。この関数はタイマオブジェクトを作成した後に呼び出さなければなりません。

4.3　simple_timer_start_timer

この API はシンプルタイマ･フレームワークのタイマを開始します。

形式

```
status_code_t simple_timer_start_timer (st_timer_t *timer,
                                        uint32_t duration,
                                        st_callback_func_t callback_func,
                                        uint32_t param)
```

パラメータ

`timer`

開始されるタイマオブジェクトへのポインタ。

`duration`

このタイマのタイムアウト時間。

`callback_func`

タイマが完了した際に呼び出される関数のポインタ。

`param`

コールバック関数の呼び出し時に渡されるパラメータ。

戻り値

ST_SUCEESS=正常終了

ST_SUCCESS 以外の値はエラーが生じたことを示しています。

所在

プロトタイプはファイル simple_timer.h に、

関数の本体はファイル simple_timer.c に存在します。

説明

この API はシンプルタイマ･フレームワークのタイマを開始します。開始させるタイマはタイマポインタで示され、そのタイムアウト値が時間の引数で指定されます。シンプルタイマ･フレームワークではタイマの完了をユーザアプリケーションに伝えるためにコールバック関数が使用されます。param はコールバック関数に渡される引数で、何を示すものかは限定されていません。ユーザアプリケーションで uint32_t 以外の型を定義しているときには型のキャストを行うことを推奨します。このフィールドは 4 バイト長ですので、ポインタを渡すこともできます。

タイマがすでに起動されている（有効である）ときには、ST_FAILURE が戻されます。タイマが正常に開始されると、このタイマは停止時刻の順にソートされた状態でタイマの連結リストに格納されます。

4.4　simple_timer_cancel_timer

この API はシンプルタイマ･フレームワークのタイマをキャンセルします。

形式

```
st_status_code_t simple_timer_cancel_timer (st_timer_t *timer)
```

パラメータ

`timer`

キャンセルされるタイマオブジェクトをポインタで指定します。

戻り値

ST_SUCEESS=正常終了

ST_SUCCESS 以外の値はエラーが生じたことを示しています。

所在

プロトタイプはファイル simple_timer.h に、

関数の本体はファイル simple_timer.c に存在します。

説明

この API は有効なシンプルタイマ･フレームワークのタイマをキャンセルします。起動されていないタイマをキャンセルしようとすると、ST_FAILURE が戻されます。キャンセルされたタイマは有効なタイマの連結リストから外されます。

4.5 simple_timer_get_time

この API はシンプルタイマ･フレームワークにおけるタイムティックを返します。

形式

```
uint32_tsimple_timer_get_time (void)
```

パラメータ

なし

戻り値

現在のタイムティック値が uint32_t 型で戻されます。

所在

プロトタイプはファイル simple_timer.h に、

関数の本体はファイル simple_timer.c に存在します。

説明

この API はシンプルタイマ･フレームワークにおけるその時点のタイムティックの値を知るために使用されます。

4.6 simple_timer_set_time

この API はシンプルタイマ･フレームワークのタイムティックを設定します。

形式

```
st_status_code_tsimple_timer_set_time(uint32_ttime)
```

パラメータ

`time`

タイムティックの値

戻り値

ST_SUCEESS=正常終了

ST_SUCCESS 以外の値はエラーが生じたことを示しています。

所在

プロトタイプはファイル simple_timer.h に、

関数の本体はファイル simple_timer.c に存在します。

説明

この API はシンプルタイマ･フレームワークのタイムティックの値を設定するために使用されます。
この API によって明確に設定されていないとき、タイムティックのデフォルト値は 0 です。

4.7 simple_timer_get_head

この API はシンプルタイマ･フレームワークにおける先頭ポインタを返します。

形式

```
st_timer_t*simple_timer_get_head (void)
```

パラメータ

なし

戻り値

有効なタイマの連結リストの先頭を示す timer_list_head 変数の値が戻されます。

所在

プロトタイプはファイル simple_timer.h に、

関数の本体はファイル simple_timer.c に存在します。

説明

この API はシンプルタイマ･フレームワークの timer_list_head 変数の値を得るために使用されます。
NULL が戻されたときには、リスト内に有効なタイマが存在していないことを意味しています。

4.8 simple_timer_increment_time

この API はシンプルタイマ･フレームワークのタイムティックをインクリメントします。

形式

```
voidsimple_timer_increment_time (void)
```

パラメータ

なし

戻り値

なし

所在

プロトタイプはファイル simple_timer.h に、

関数の本体はファイル simple_timer.c に存在します。

説明

この API はシンプルタイマ･フレームワークのタイムティックをインクリメントします。通常、この API 呼び出しは周期タイマの割り込みサービスルーチンの中に追加されます。

4.9 simple_timer_check_timer_expiry

この API はシンプルタイマ･フレームワークのタイマが完了したかをチェックします。

形式

```
voidsimple_timer_check_timer_expiry (void)
```

パラメータ

なし

戻り値

なし

所在

プロトタイプはファイル simple_timer.h に、

関数の本体はファイル simple_timer.c に存在します。

説明

この API は、シンプルタイマ･フレームワークの有効なタイマのリストにあるタイマが完了したかをチェックするために使用されます。有効なタイマはリスト内で停止時刻に関してソートされており、タイマはリストの先頭から順にチェックされます。先頭ポインタが示すタイマが完了時刻に達しているときには、タイマはリストから外され、このタイマのコールバック関数が呼び出されます。API はリスト内の次のタイマに先頭ポインタを更新し、新たに先頭となったタイマの停止時刻に達していないかを調べます。リスト内にタイマがなくなるか、先頭ポインタが示すタイマの停止時刻に達するまで、この操作が繰り返されます。

この API の呼び出しは、通常、メインループに追加されます。

5. References

1. Group Hardware Manuals for the Renesas device on the RSK board.

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/inquiry

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2013.02.08	—	初版発行

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ものですが、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も、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。

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・電話番号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 エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■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/