RX63x グループ

RX630 用割り込みモード UART ドライバ

R01AN0820JU0101
Rev.1.01
2013.03.11

はじめに

RX630 グループは 13 チャネルのシリアルコミュニケーションインタフェース（SCI）を持っています。この設定モードには、調歩同期式シリアルインタフェースの Universal Asynchronous Receiver/Transmitter（UART）として使用できる機能が含まれます。これは RS232 入出力ラインドライバ／レシーバ（トランシーバ）と共に使われると RS232 シリアルポートとして広く知られています。RSK RX630 開発ボードは、RS232 トランシーバに接続された 1 つの SCI チャネルを持ち、DB9 シリアルケーブルコネクタを搭載しています。このため、このボードは UART 通信ソフトウェアドライバの開発とデモンストレーションのプラットフォームとして適しています。

本アプリケーションノートでは、UART に設定された SCI 周辺機能を使用した調歩同期式通信を行うためのソフトウェアドライバについて説明します。ソフトウェアはバッファを使用した割り込み駆動の入出力を行うもので、リアルタイムアプリケーションに適したドライバの実例となっています。本文書と共に、RX630 SCI を UART モードで使用する際のレジスタレベルの設定と制御が、作業用の C コード HEW プロジェクトとして提供されています。

本アプリケーションノートは、ルネサス RX MCU を使用した調歩同期式シリアル通信を必要とするアプリケーションの開発を行う開発者およびレジスタレベルでの SCI 周辺機能の設定と操作の実例を必要としている開発者を対象としています。

対象デバイス

RSK RX630 スタータキットボードに搭載された RX630

関連資料

RX630 グループユーザーズマニュアルハードウェア編

High-performance Embedded Workshop ユーザーズマニュアル

目次

1. アプリケーションの概要

本アプリケーションノートは、UART に設定した SCI 周辺機能を使用して調歩同期式シリアル通信を行うために用意されたソフトウェアドライバの使用方法を示しています。ソフトウェアはバッファを使用した割り込み駆動の入出力を行うもので、リアルタイムアプリケーションに適したドライバの実例となっています。このコード例の主な利点は、SCI ハードウェアの初期化と効率的でコントロールを占有しないシリアル入出力を実行する SCI 割り込みの使用を示していることです。

ソフトウェアは特に RSK RX630 開発ボードを対象としていますが、ルネサス RX MCU を使用する他のボードにも容易に適合させることができます。UART ドライバコードはサンプルアプリケーションコードとは分かれているため、移植性が高くなります。プラットフォームに依存しない方法でドライバにアクセスできるようにいくつかの便宜的な関数が用意されています。

2. UART の概要

UART は Universal Asynchronous Receiver/Transmitter の略称です。UART の役割は、並列バイトデータを形式化されたシリアルデータビットストリームに変換することです。シリアルバスに送り出されるデータの各バイトはセルフクロック方式です。このデータは、送信されるデータフィールドの識別と検査のために様々なビットがオプションで含まれるパケットの形式になっています。通常 UART は RS232 標準ラインドライバ/レシーバ IC を使用して RS232 シリアルバスに接続されます。RX630 シリアルコミュニケーションインタフェースは一般的な UART 以上の機能を持っていますが、通常の UART の機能を行うよう設定することもできます。

3. ツールとリソース

以下のものが必要となります。

- HEW（High-performance Embedded Workshop）ツール
- RSK RX630_UART プロジェクト HEW ワークスペースパッケージ
- RSK RX630 MCU 開発ボード
- E1 もしくは E20 エミュレータ
- DB9 コネクタがついた RS232 ケーブル
- PC ターミナルアプリケーション（Microsoft®ハイパーターミナルなど）

4. プロジェクトのビルド

RSK RX630 UART プログラムをビルドするには、使用するルネサス RSK RX630 開発ボードの HEW ワークスペースプロジェクト RSK RX630_UART.hws を開きます。ボードと E1 もしくは E20 エミュレータの接続に関する詳細は、個々のボードに対応するインストールガイドを参照してください。HEW 開発ツールソフトウェアに関しては、High-performance Embedded Workshop ユーザーズマニュアルを参照するか、HEW で Help →Help Topics をクリックしてください。

表 1 は RSK RX630UART プロジェクトのワークスペースに含まれているファイルを示しています。

表 1

C ソースファイル	C ヘッダファイル
Dbsct.c hwsetup.c lcd.c led.c resetprg.c sbrk.c uart.c vecttbl.c	iodefine.h lcd.h rskRX6xxdef.h sbrk.h stacksct.h uart.h

5.　接続と設定

5.1　　RSK ボードの接続

- E1 もしくは E20 デバッガ/エミュレータを RSK ボードに接続し、アプリケーションをダウンロードします。
- PC の RS232COM ポートのシリアルケーブルを RSK RX630UART プログラムを実行している MCU に接続します。
  注:　ご使用の RSK ボードに DSUB-9 コネクタが搭載されていない場合は、ボードの回路図を参考にして UART の RX と TX ピンを見つけ直接接続してください。

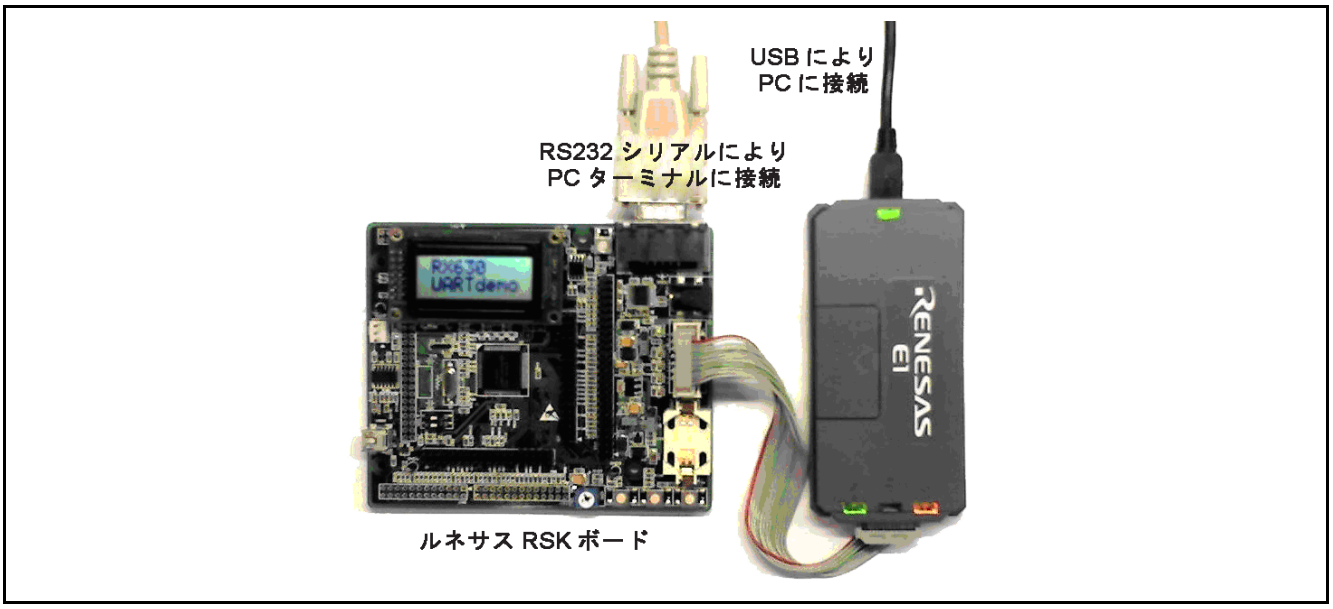

図 1　UART デモンストレーションプロジェクトのための接続

5.2　　シリアルポートの設定

- RX MCU における UART の設定:8 データビット、1 ストップビット、パリティなし、フローコントロールなし。
- 実際の速度を知るために BAUDRATE を確認してください。これは uart.c ファイルで定義されています。標準的な値は 115200 です。
- ご使用のターミナルアプリケーションのシリアル通信ポートの設定を RSK RX630UART プログラムの設定に一致させます。
- シリアル接続の設定が正しければ、ボードのスタートアップ時に PC ターミナル上に初期メッセージが表示されます。
- PC ターミナルのキーボードから任意の文字を入力すると、ターミナルの画面上にその文字が表示されます。

6. アプリケーション例の操作

プログラムは RX630MCU の UART を介して文字データを受け取り、同じデータをシリアルインタフェース経由で送り返します。これは PC のターミナルプログラム（Microsoft®ハイパーターミナルなど）上で確認することができます。

アプリケーション例を実行する手順は次のようにまとめられます。各ステップの詳細な説明については、各セクションを参照してください。

1. セクション **4:プロジェクトのビルド**に説明されているようにサンプルコードをコンパイルしダウンロードします。
2. RS232 ポートを PC もしくはシリアルデータターミナルに接続します。**5.1:RSK ボードの接続**を参照してください。
3. 端末ポートの設定が 115200 ボー、8 ビット、パリティなし、1 ストップビット、ハンドシェイクなし、となっていることを確認します。
4. また、ターミナルでシリアル接続を有効にします。
5. Reset Go をクリックして、ソフトウェアをスタートします。
6. LED の点滅を確認します。
7. 初期メッセージがターミナルディスプレイ上に表示されます。
8. ターミナルに文字を入力し、文字がエコーバックされることを確認します。

7. ソフトウェア

このドライバは、SCI 周辺機能からの割り込みイベントを使用して RAM ベースのデータキューとの間のシリアル通信データ転送を呼び出します。これによりアプリケーションはシリアル接続を介したデータの書き込みと読み出しを最小のレイテンシで行うことができます。目的は送信および受信手順の関数呼び出しでコントロールを占有する動作を避けることです。

アプリケーションレベルでは、送信データは直ちに RAM 上のキューに書き込まれ、その後アプリケーションは別の処理を実行することができます。送信キューにデータが存在するときには、送信割り込みサービスルーチンがバックグランドでシリアルバスへのデータ送信を開始します。つまり、トランスミットデータレジスタが次の文字を受け入れられる状態で、送信データがある場合にのみ送信割り込みハンドラが短時間動作します。

非同期のデータ受信では、アプリケーションがいつ送られてくるかわからないデータを待ちつづけることは処理能力の大きな浪費となります。受信データバッファフル割り込みハンドラが新たなデータが送られてきたときにのみ自動的に実行されることで待ち時間をなくすることができます。アプリケーションが後ほどデータにアクセスできるように割り込みハンドラが受信データをいったん RAM 上のキューに格納するため、アプリケーションはこのイベントに直ちに対応する必要はありません。

表 2　uart.c 関数の一覧

ドライバ関数名	目的
sci_uart_init	SCI チャネルを調歩同期式 UART として動作するよう初期化します。
sci_put_char	1 バイトのデータをシリアルポートに書き出します。
sci_get_char	シリアルポート受信割り込みハンドラにより書き込まれた受信バッファから 1 バイトを読み出します。
sci_write_str	シリアルポートから NUL で終端される文字列を出力します。
sci_read_count_get	受信バッファ内の未処理データの現在の数を取得します。
sci_tx_interrupt_enable	送信に関連する割り込みを許可します。
sci_rx_interrupt_enable	受信データレディ割り込みと受信エラー割り込みを許可します。
sci_tx_interrupt_disable	送信に関連する割り込みを禁止します。
sci_rx_interrupt_disable	受信データレディ割り込みと受信エラー割り込みを禁止します。
SCI0_TXI0_isr	SCI TXI（トランスミットデータレジスタエンプティ）割り込みの ISR。呼び出し不可。
SCI0_RXI0_isr	SCI RXI（レシーブデータレジスタフル）割り込みの ISR。呼び出し不可。
SCI0_TEI0_isr	SCI TEI（送信完了）割り込みの ISR。呼び出し不可。

8. SCI UART ドライバ関数

8.1 sci_uart_init

SCI チャネルが調歩同期式 UART として動作するための準備を行います。

フォーマット

```
voidsci_uart_init(void);
```

パラメータ

なし

戻り値

なし

プロパティ

“uart.h”ファイルにプロトタイプ化されます。

説明

この関数では入出力端子の設定、動作モードとシリアルポート通信パラメータの設定、割り込みのセットアップを行います。この関数は SCI シリアルチャネル上のデータ転送の前に呼び出す必要があります。

リエントラント

- リエントラントではありません。

使用例

```
/* Initialize the SCI channel 0 as UART.
   Do this before calling any other functions of the UART driver */

    sci_uart_init();

/* Now ready to begin using the UART */
```

注意事項:

シリアル通信設定:

- 調歩同期式
- 8 データビット
- パリティなし
- 1 ストップビット
- ハードウェアフロー制御: オフ

UART 入出力端子

- P21 が RxD0 端子として使用されます。
- P20 が TxD0 端子として使用されます。

クロック入力:PCLK を分周比 1 で使用します。
ボーレート:#define によって 115200 ボーに設定されます。

制約

このサンプルコードでは SCI チャネル番号は SCI チャネル 0 にハードコーディングされており、入出力は RX630 RSK ボードで RS232 シリアルポートに接続された端子に割り当てられます。他の設定をサポートするには sci_uart_init 関数内のこれらの値を変更する必要があります。または、コードを再度入力し、より動的な設定方法を使用することもできます。

8.2 sci_put_char

1 バイトのデータをシリアルポートに書き出します。

フォーマット

```
bool sci_put_char(uint8_t write_data);
```

パラメータ

write_data

書き出されるバイトデータ

戻り値

Boolean 結果コード:

True: データが受け付けられました。

False: キューが一杯です。データはキューに書き込まれません。

プロパティ

“uart.h”ファイルにプロトタイプ化されます。

説明

この関数は 1 バイトのデータをシリアルポートに書き出します。この関数はレイテンシを小さくするためにバッファ付き出力を使用しています。SCI チャネルのトランスミットデータレジスタが空であれば write_data の値は直接データレジスタにコピーされ、この関数はデータが受け付けられたことを示す true を結果として直ちに返します。レジスタが空でない場合は、データは送信データキューに格納され、関数はデータが受け付けられたことを示す true を結果として直ちに返します。グローバル変数 g_uart_tx_ready は sci_put_char 関数と送信割り込みハンドラ（ハードウェア割り込みハンドラのセクションを参照）との間で 2 つの処理間の動作を調整するフラグとして共有されます。

この関数は、sci_uart_init の実行時にリセットされる固有のキュー位置カウンタを持っています。送信キューに 1 バイトが追加されるたびに、キューの位置が進められ、カウンタの値は 1 だけ増加します。カウンタがキューバッファの終わりに達すると、位置ポインタはバッファの先頭アドレスに戻されます。

キューに蓄えられたデータは送信割り込みハンドラによって書き出され、データカウントの値が減じられます。キューが一杯の時にはこの関数は直ちに false を結果として返し、データのキューへの書き込みは行われません。

リエントラント

- リエントラントではありません。

使用例

```
/* Transmit a character to the serial port */
result = sci_put_char('A');
```

注意事項:

- この関数を呼び出す前に sci_uart_init 呼び出しが完了していなければなりません。
- キューが一杯という状況を最小限とするために、アプリケーションの必要性に応じてボーレートやバッファサイズを調整します。

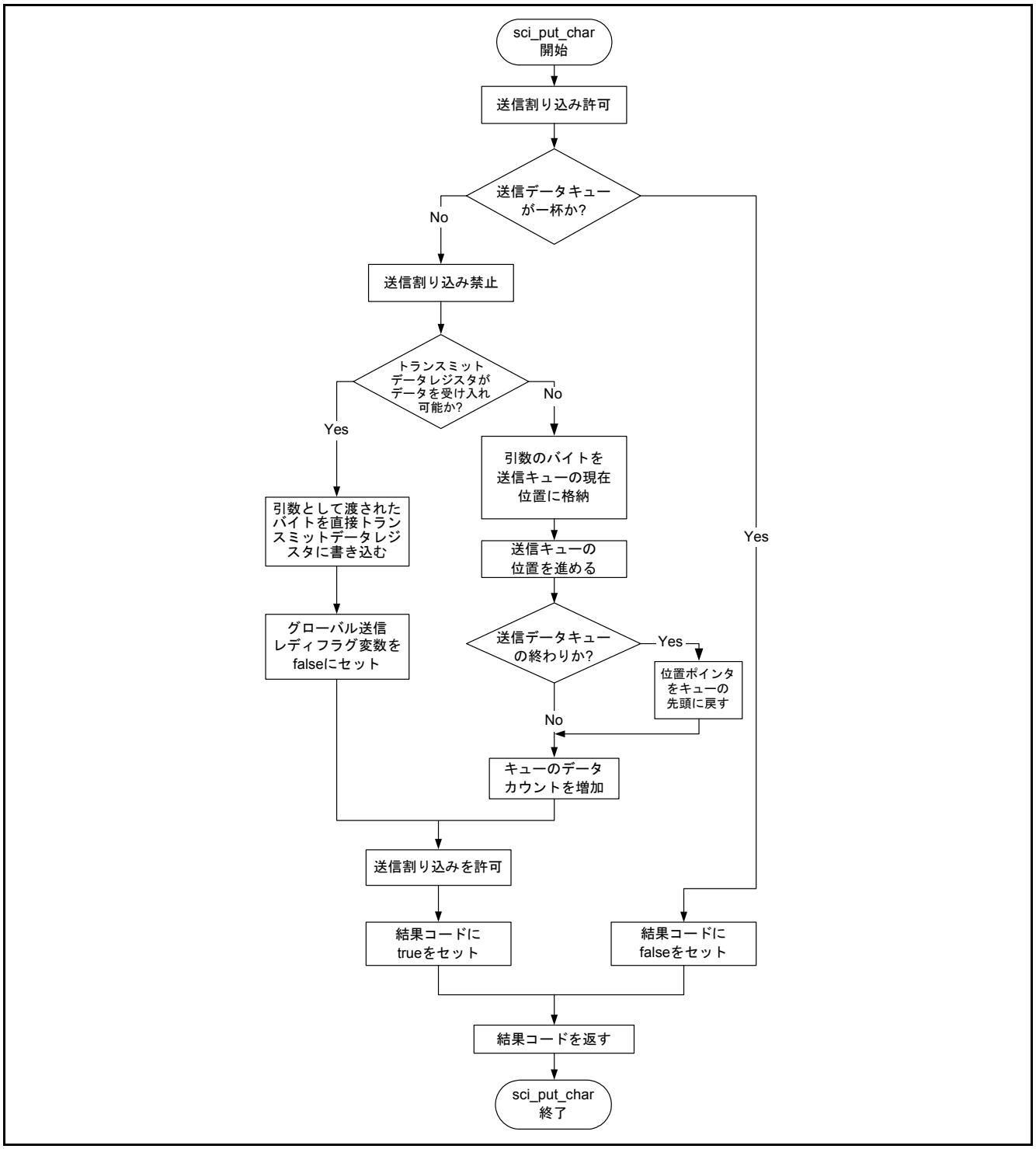

図 2　sci_put_char 関数のフローチャート

8.3　sci_get_char

受信バッファから 1 バイトを読み出します。

フォーマット

```
bool sci_get_char(uint8_t * read_data);
```

パラメータ

read_data

読み出されたデータがコピーされる場所を示すポインタ

戻り値

Boolean 結果コード

True:　　データが存在し、文字が read_data 引数に戻されました。

False:　　データが存在せず、read_data 引数には値 0 が書き込まれました。

プロパティ

“uart.h”ファイルにプロトタイプ化されます。

説明

この関数は、シリアルポート受信割り込みハンドラがデータを格納したバッファから 1 バイトのデータを読み出します。SCI ハードウェアがシリアルバスから新たなデータを受信したときには、受信 ISR はデータを RAM 上の循環キューに転送します（ハードウェア割り込みハンドラのセクションを参照）。キューバッファの終わりに達したときには、キューの位置はバッファの先頭に戻され、データの格納が続けられます。まだ読み出されていないバイト数のカウントは受信データ割り込みハンドラによって加算されます。

sci_get_char 関数は呼び出されるたびにカウント数を 1 ずつ減じます。

最終的にはカウントが 0 となり受信キュー内に処理すべきデータがないことが示されます。sci_get_char が呼び出されたときに受信キューにまだ読み込まれていないデータが残っているときには、現在のキューポインタ位置が示す場所から引数 read_data により示された場所にデータバイトがコピーされ、関数は true を結果コードとして返します。キューが空のときに sci_get_char が呼び出されると、引数 read_data に 0 が書き込まれ、関数は false を結果コードとして直ちに返します。

リエントラント

- リエントラントではありません。

使用例

次の例はこの関数の使われ方を示しています。

```
static uint8_t old_char;
uint8_t new_char;

/* Read a received character. */
if(sci_get_char(&new_char)) /* Data is returned in new_char */
{
   old_char = new_char;    /* Got a new character. Do something with it. */
}
else
{
```

```
        /* No data was available. Try again later or report an error */
    }
```

注意事項:

この関数を呼び出す前に sci_uart_init 呼び出しが完了していなければなりません。呼び出しが行われていないと新たなデータは読み込まれず受信キューは空のままです。

まず関数 sci_read_count_get を呼び出してカウントが 0 でないことを確認すれば、受信キューが空の時の sci_get_char 呼び出しを避けることができます。

制約

アプリケーションがデータを読み出す前に受信キューが一杯となったとき、受信された新たなデータの受信キューへの格納は、その場所に残っていた未処理データを上書きする形で続けられます。通常、この状態が生じる前にアプリケーションがキューからデータを読み出すことが期待されます。簡単な修正で、オプションによりこの状態でエラーを発生してフラグをセットし、アプリケーションにデータが失われたことを知らせるようにすることができます。バッファサイズは 1 個の#define 文により容易に変更することができます。現状の版ではオーバラン状態の発生は読み込みカウント値をバッファサイズ値と比べることで知ることができます。読み込みカウントがバッファサイズより大きければ、オーバランが生じていることを意味します。

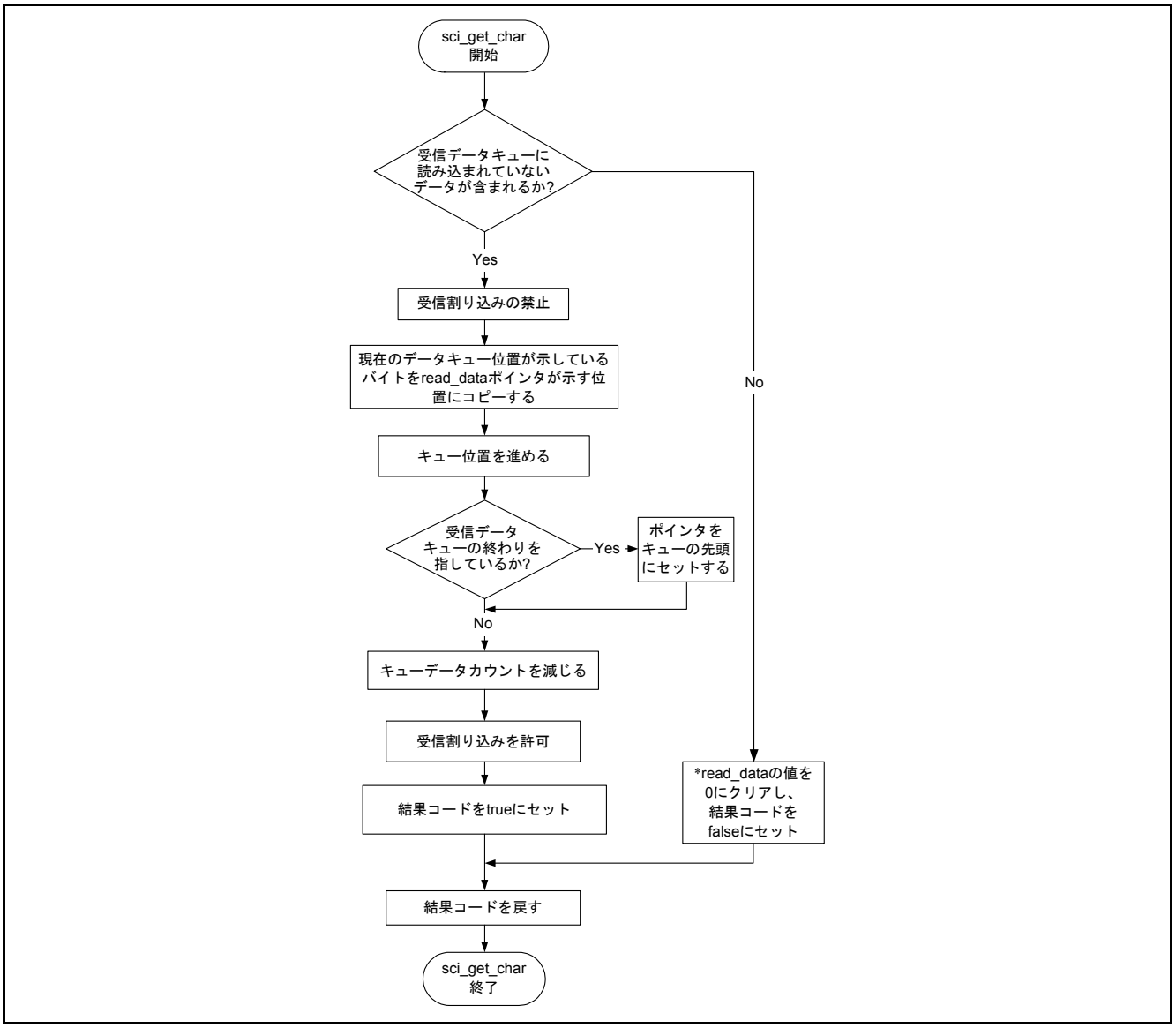

図 3　sci_get_char 関数のフローチャート

8.4 sci_write_str

シリアルポートから NUL で終端される文字列を出力します。

フォーマット

```
uint32_t sci_write_str(uint8_t *source_string);
```

パラメータ

**source_string*

バイトデータの NUL で終端される文字列へのポインタ

戻り値

書き出されたバイト数

プロパティ

“uart.h”ファイルにプロトタイプ化されます。

説明

この関数は低レベルの sci_put_char 関数を使用して上位レベルの手順を構成する際の、簡単で有用な実例です。sci_write_str はソース文字列データを処理しながら 0（NUL）の終端が現れるまで、もしくは送信キューが一杯となるまで(内部的な sci_put_char 呼び出しが false を戻すまで)sci_put_char を繰り返し呼び出します。終端文字として使用されている NUL は出力されません。実際に書き込まれた合計のバイト数がこの関数の呼び出し側に戻されます。

リエントラント

- リエントラントではありません。この関数はデータをシリアル UART 送信キューに書き込みます。

使用例

```
uint32_t num_written;
uint8_t title_str[] = "RX630 UART Demonstration¥r¥n";

num_written = sci_write_str(title_str);

if (num_written < strlen(title_str))
{
   ... /* Transmit queue must have been full. Handle error. */
}
```

注意事項:

この関数がソース文字列の出力中に送信キューが一杯になると、文字列全体を書き出す前に処理が中断されます。この場合には、実際に書き出されたバイト数のみが戻されます。

8.5　sci_read_count_get

受信データキュー内の未処理データの現在の数を取得します。

フォーマット

```
uint32_t sci_read_count_get(void);
```

パラメータ

なし

戻り値

受信データキュー内に読み出されずに残っているバイト数

プロパティ

“uart.h”ファイルにプロトタイプ化されます。

説明

sci_read_count_get はアプリケーションに読み込まれていないデータが受信バッファ内に何バイトあるかを確認するために有効です。この関数を事前に呼び出しバッファが空の状態で sci_get_char 関数を呼び出すことを避けるためにも利用できます。

リエントラント

- リエントラントです。

使用例

```
/* Find out how much unprocessed data the receive queue is holding. */
num_in_rx_queue = sci_read_count_get()

if(num_in_rx_queue  > (RX_QUEUE_SIZE / 2))
{
   ... /* Receive queue is over half full. Better start reading from it! */
}
```

9. ハードウェア割り込みハンドラ

SCI 割り込みハンドラは送信キューにデータを格納し受信キューのデータを処理する上位レベルの関数と関連して動作します。このとき送信キューと受信キューのそれぞれでバッファ領域は循環バッファとして扱われます。バッファの終わりに達したときに先頭に折り返すことでバッファの境界をはみ出すという問題を避けることができます。

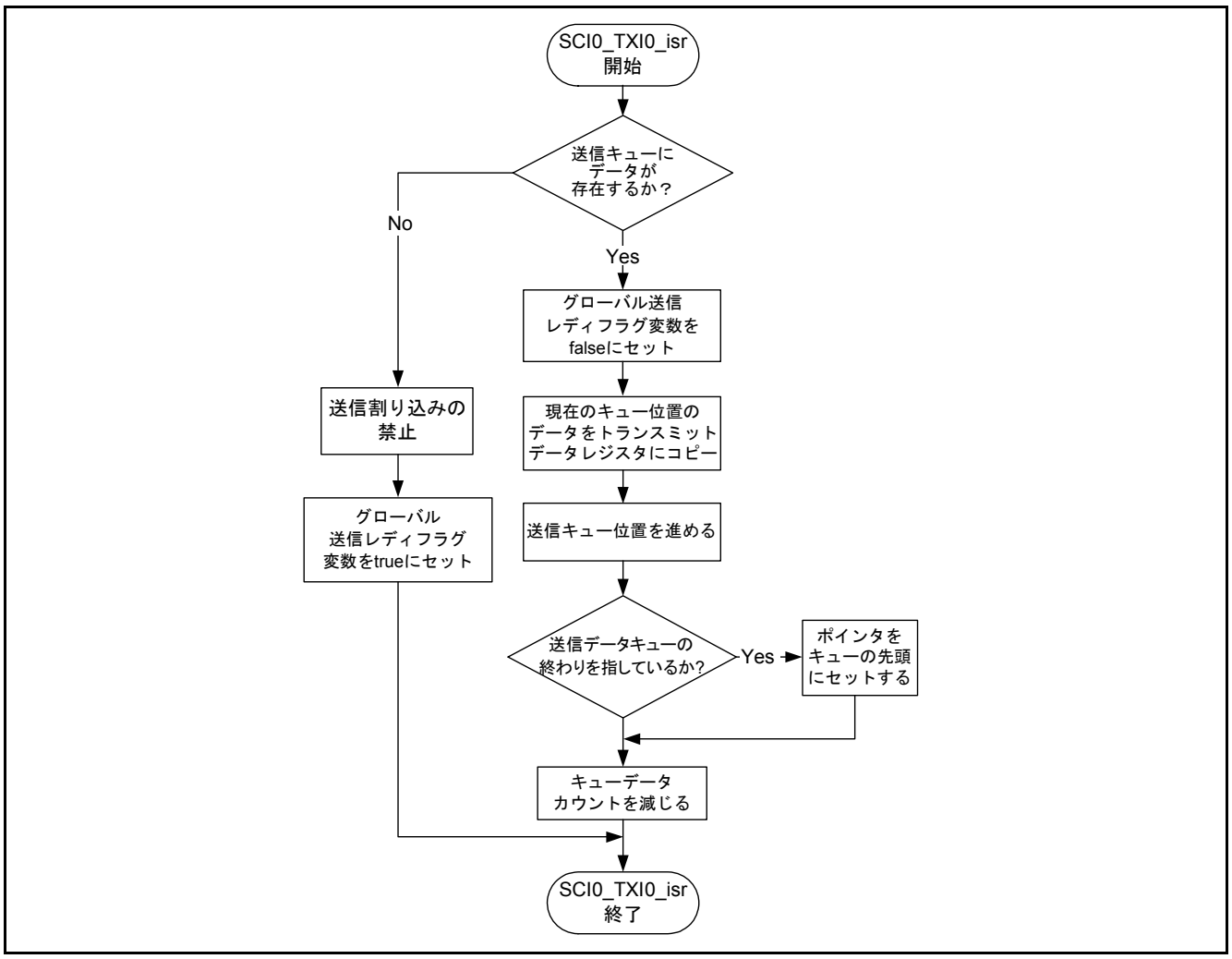

図 4　送信データバッファエンプティ割り込みサービスルーチンのフローチャート

この ISR は SCI チャネルのトランスミットデータレジスタエンプティ割り込み要求が発生したときに呼び出されます。アプリケーションに送信するデータがあるときには、sci_put_char 関数を呼び出し、この関数がデータを直接 SCI トランスミットデータレジスタに格納し（その時点の送信動作がアイドル状態の場合）、もしくは SCI が送信動作中であれば、送信データキューに格納します。トランスミットデータレジスタのバイトがシリアルシフトレジスタに転送されると、データレジスタは次のデータを受け入れることが可能となり割り込み要求が発生し、その結果この送信 ISR が再び呼び出されます。グローバル変数 g_uart_tx_ready は sci_put_char 関数とこの割り込みとの間で両関数の相互動作を調整するフラグとして共有されます。

送信キュー内に送信するデータがある場合、この ISR はキューから 1 バイトを読み出し、トランスミットデータレジスタにコピーします。この関数は、sci_uart_init の実行時にリセットされる固有のキュー位置カウンタを持っています。キューからデータが取り出されると、この位置が進められます。キューバッファの終わりに達すると、キュー位置はバッファの先頭に折り返され、この位置から順にデータが取り出されるようになります。送信キュー内に保持されているバイト数のカウントはグローバル変数 g_tx_count で行われます。カウント値はこの関数で 0 になるまで減じられます。カウンタの値が 0 となると、この ISR はその後の割り込みを禁止し、新たなデータを受け入れ可能であることを示すグローバルステータスフラグをセットします。

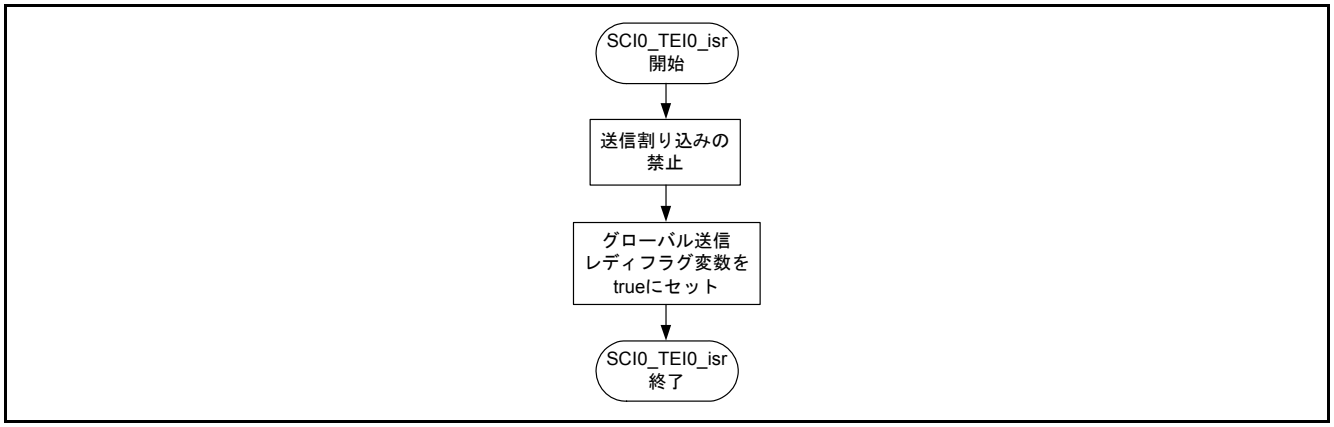

図 5　送信完了割り込みサービスルーチンのフローチャート

送信完了（TEI）：割り込みハンドラは送信動作が完了したときに呼び出されます。この ISR は単にハードウェアがアイドル状態であり、別の送信動作を行うことができることを示すグローバルステータスフラグをセットするのみです。

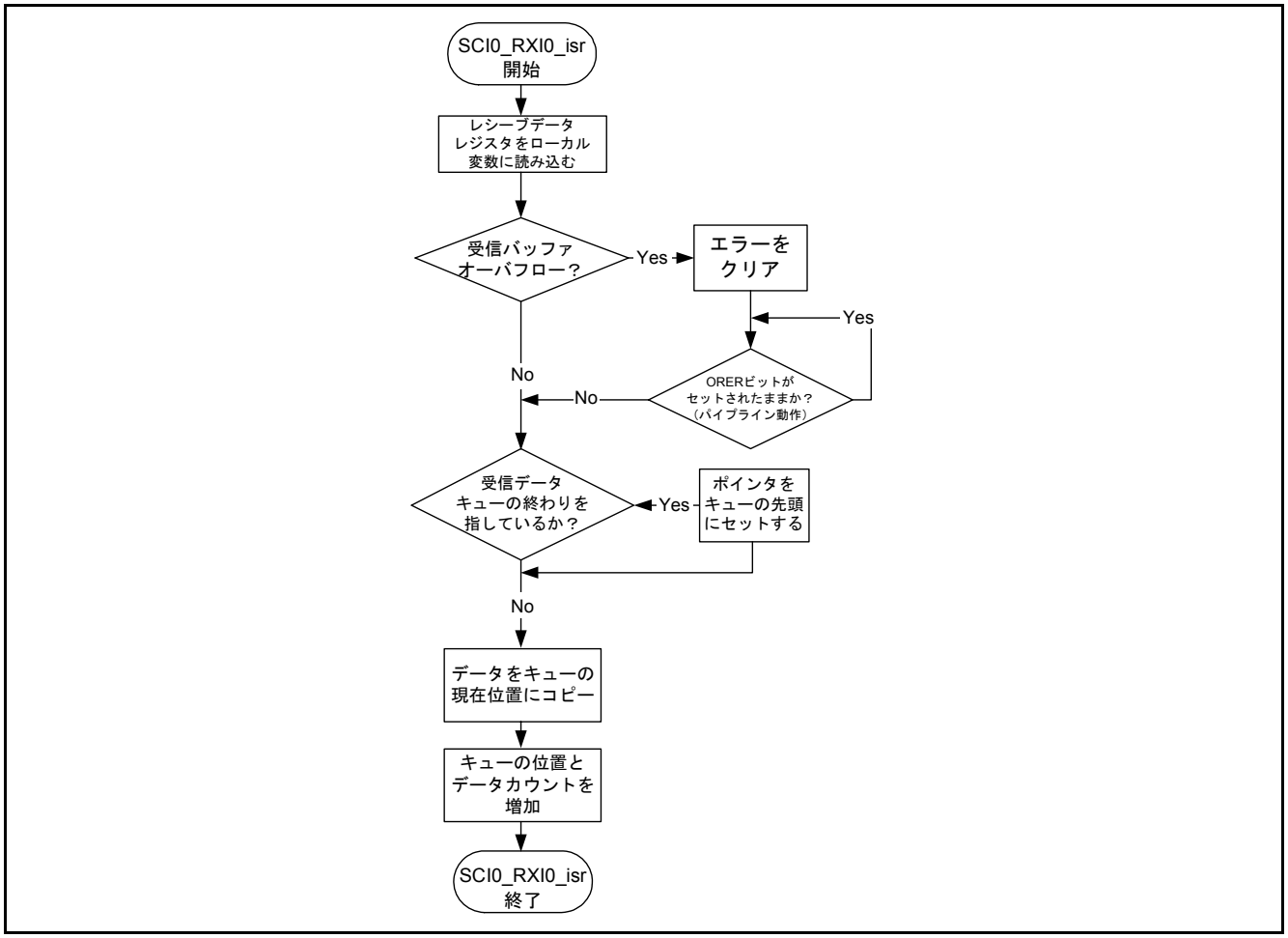

図 6　受信データバッファフル割り込みサービスルーチンのフローチャート

SCI ハードウェアがシリアルバスから新たなデータを受信すると、受信 ISR がそのデータを RAM 上の循環キューに転送します。キューバッファの終わりに達すると、キュー位置はバッファの先頭に折り返されデータの格納が続けられます。キュー内にコピーされているバイト数のカウントはグローバル変数 g_rx_count で行われます。カウント値はアプリケーションがキュー内のデータを処理するにしたがって sci_get_char 関数によって減じられます。この関数は、sci_uart_init の実行時にリセットされる固有のキュー位置カウンタを持っています。

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	RX63x グループ　アプリケーションノート RX630 用割り込みモード UART ドライバ

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.10.4	—	初版発行
1.01	2013.3.11	—	新しい文書テンプレートへの変更、細部改訂

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/