RX ファミリ
ロングワード型キューバッファ（LONGQ）モジュール
Firmware Integration Technology

要旨

このモジュールはロングワード型（uint32_t）のリングバッファを構成し管理する関数を提供します。本説明では 32 ビットデータを 1 エントリとします。

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。

- RX ファミリ

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合わせて変更し、十分評価してください。

対象コンパイラ

- Renesas Electronics C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX
- 各コンパイラの動作確認内容については5.1動作確認環境を参照してください。

目次

1. 概要

ロングワード型キューバッファ（LONGQ）モジュールは、アプリケーションが提供するバッファ領域を基本的なリングバッファとして扱う方法を提供しています。

本モジュールでは R_LONGQ_Open()関数に渡された個々のバッファに対してキューコントロールブロック（QCB）を割り当てます。キューにデータを追加／削除するために、QCB はバッファへのデータの出し入れのインデックスを保持します。QCB はコンパイル時に静的に配置することも、実行時に malloc を使用して動的に配置することも可能です。r_longq_config.h ファイルにある設定オプションで、QCB を静的に割り当てるか、動的に割り当てるかを設定します。静的に割り当てる場合、サポートされるバッファ／キューの最大数も設定する必要があります。

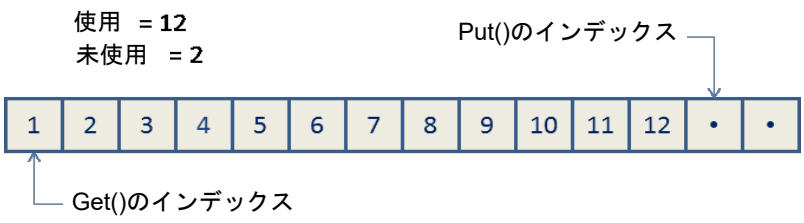
1 つのバッファに対して 1 つのコントロールブロックが用意されます。R_LONGQ_Open()が実行される際にアプリケーションのバッファ領域のポインタとサイズが渡され、QCB へのポインタが返されます。このポインタはハンドルと呼ばれ、以後、他のすべての API 関数に渡されます。API 関数はハンドルで示されるキューに対して操作を行います。複数のキュー間で、グローバルまたは静的なデータが共有されることはありませんので、API 関数は別のキューに対して再入可能です。

本モジュールは割り込みを使用しません。割り込みとアプリケーションの双方でキューが変更される可能性がある場合、必要に応じて、ユーザアプリケーションで、同じキューに同時にアクセスしないように一方に制限をかけてください。また、キューが優先レベルの異なる複数のタスクからアクセスされる場合、タスクの切り替えを禁止するか、ミューテックスやセマフォを使用してキューを確保するかをユーザアプリケーションで指定してください。

1.1 LONGQ モジュールを使用する

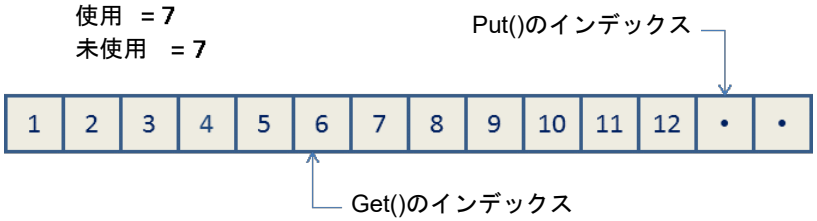
以下に API 呼び出し時のキューの動きを説明します。

```
#define BUFSIZE 14

uint32_t my_buf[BUFSIZE];
longq_hdl_t my_que;
longq_err_t err;
uint32_t data,i;

err = R_LONGQ_Open(my_buf, BUFSIZE, false, &my_que);

/* add 12 entries to queue */
for (i=0; i < 12; i++)
{
    R_LONGQ_Put(my_que, i+1);
}
```



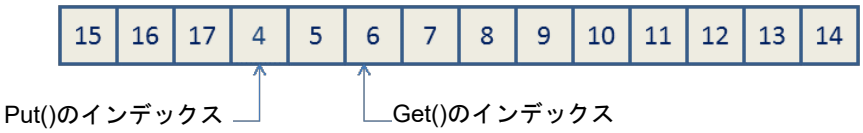

```
/* remove 5 entries from queue */
R_LONGQ_Get(my_que, &data);      // data = 1
R_LONGQ_Get(my_que, &data);      // data = 2
R_LONGQ_Get(my_que, &data);      // data = 3
R_LONGQ_Get(my_que, &data);      // data = 4
R_LONGQ_Get(my_que, &data);      // data = 5
```

使用 = 7
未使用 = 7



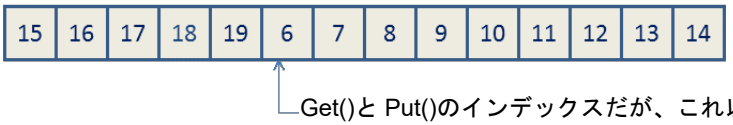

```
/* add 5 entries to queue */
R_LONGQ_Put(my_que, 13);
R_LONGQ_Put(my_que, 14);
R_LONGQ_Put(my_que, 15);
R_LONGQ_Put(my_que, 16);
R_LONGQ_Put(my_que, 17);
```

使用 = 12
未使用 = 2



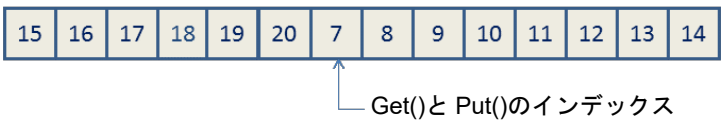

```
/* attempt to add 3 more entries to queue */
err = R_LONGQ_Put(my_que, 18);
err = R_LONGQ_Put(my_que, 19);
err = R_LONGQ_Put(my_que, 20);   // err=LONGQ_ERR_QUEUE_FULL
```

使用 = 14
未使用 = 0



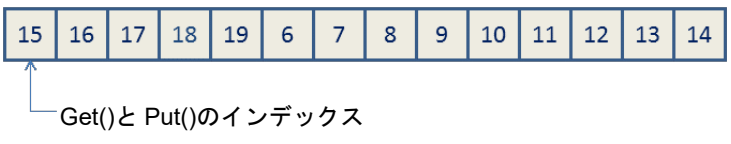

```
/* NOTE: If Open() was called with ignore_overflow = true, last Put() would
succeed */
```

使用 = 14
未使用 = 0




```
/* flush queue */
R_LONGQ_Flush(my_que);
```

使用 = 0
未使用 = 14



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

ハードウェアの要求はありません。

2.2 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。

- ルネサスボードサポートパッケージ（r_bsp）v.3.10 以降

2.3 制限事項

ソフトウェアに関する制限事項はありません。

2.4 サポートされているツールチェーン

本 FIT モジュールは「5.1 動作確認環境」に示すツールチェーンで動作確認を行っています。

2.5 ヘッダファイル

コンパイル時に設定可能なオプションは、ファイル r_longq¥ref¥r_longq_config_reference.h に含まれます。このファイルをプロジェクトのサブディレクトリ r_config にコピーして、r_longq_config.h というファイル名に変更してください。設定変更が必要な場合はコピーしたファイル r_longq_config.h を変更し、元のファイルは参照用として確保しておきます。

すべての API 呼び出しとサポートされるインタフェース定義はファイル r_longq¥r_longq_if.h に記載されています。ユーザアプリケーションではこのファイルと r_longq_config.h ファイルをインクルードする必要があります。

2.6 整数型

ご使用のツールチェーンが C99 をサポートしている場合、以下に示すように stdint.h で定義します。C99 をサポートしていない場合、ルネサスのコーディング規定で定義されているとおり、typedefs.h ファイルがプロジェクトに含まれています。

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。

2.7 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すべて r_longq_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_longq_config.h）	
#define LONGQ_CFG_PARAM_CHECKING_ENABLE	• 1：ビルド時にパラメータチェック処理をコードに含めます。 • 0：ビルド時にパラメータチェック処理をコードから省略します。 • BSP_CFG_PARAM_CHECKING_ENABLE（デフォルト）： システムのデフォルト設定を使用します。 注: ビルド時にパラメータチェックのコードを省略することで、コードサイズを小さくすることができます。
#define LONGQ_CFG_USE_HEAP_FOR_CTRL_BLKS ※デフォルト値は“0”	キューのデータの出し入れのインデックスを保持するために、コントロールブロックはキューごとに必要です。デフォルトではコントロールブロックはコンパイル時に配置されます。実行時に動的にメモリを割り当てるには、この値を“1”に設定します。
#define LONGQ_CFG_MAX_CTRL_BLKS ※デフォルト値は“32”	コンパイル時に配置されるコントロールブロック数を指定します。この定数は LONGQ_CFG_USE_HEAP_FOR_CTRL_BLKS が“1”のときには無視されます。
#define LONGQ_CFG_PROTECT_QUEUE ※デフォルト値は “0”	この定義は、アプリケーションと割り込みレベルの両方から競合することなくアクセスすることができるように、キューを保護するために使用されます。 = 1: キューを保護するために割り込み禁止を使用する = 0: キューを保護しない 注 1：LONGQ_CFG_PROTECT_QUEUE = 1 とする場合、BSP_CFG_RUN_IN_USER_MODE = 0 としてください。もし上記を設定しない場合、ビルドエラーが発生します。 注 2: もし LONGQ_CFG_PROTECT_QUEUE = 1 の場合キュー全体は保護されませんが、調歩同期式モードで使用されるときは、キューは保護されます。
#define LONGQ_CFG_CRITICAL_SECTION ※デフォルト値は “0”	この定義は関数 R_LONGQ_Put(), R_LONGQ_Get()のクリティカルセクションを保護するために使用され、アプリケーションと割り込みレベルの両方から競合無しにアクセスすることができます。 = 1: クリティカルセクションを保護するために割り込みを禁止します。 = 0: クリティカルセクションは保護されません。 注 1: LONGQ_CFG_CRITICAL_SECTION = 1 とする場合、BSP_CFG_RUN_IN_USER_MODE = 0 としてください。BSP_CFG_RUN_IN_USER_MODE = 0 のときに LONGQ_CFG_CRITICAL_SECTION = 1 とすると、ビルドエラーが発生します。

2.8 コードサイズ

本モジュールの ROM サイズ、RAM サイズを掲載しています。

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「2.7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。

下表の値は下記条件で確認しています。

モジュールリビジョン: r_longq rev1.90

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00
コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
GCC for Renesas RX 8.3.0.202102
コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、統合開発環境のデフォルト設定に以下のオプションを追加 -Wl,--no-gc-sections
これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
IAR C/C++ Compiler for Renesas RX version 4.20.3
コンパイルオプション：統合開発環境のデフォルト設定

コンフィグレーションオプション: デフォルト設定

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					
		Renesas Compiler		GCC		IAR Compiler	
		パラメータチェックあり	パラメータチェックなし	パラメータチェックあり	パラメータチェックなし	パラメータチェックあり	パラメータチェックなし
コントロールブロックにヒープを使用	ROM	277 バイト＋ 269 バイト （注 1）	211 バイト＋ 273 バイト （注 1）	1120 バイト＋ 208 バイト （注 1）	1016 バイト＋ 200 バイト （注 1）	728 バイト＋ 290 バイト （注 1）	656 バイト＋ 282 バイト （注 1）
	RAM	16 バイト(malloc()による領域確保) × コントロールブロック数		16 バイト(malloc()による領域確保) × コントロールブロック数		16 バイト(malloc()による領域確保) × コントロールブロック数	
コントロールブロックをコンパイル時に配置	ROM	324 バイト＋ 269 バイト （注 1）	257 バイト＋ 273 バイト （注 1）	696 バイト＋ 520 バイト （注 1）	584 バイト＋ 520 バイト （注 1）	424 バイト＋ 282 バイト （注 1）	352 バイト＋ 274 バイト （注 1）
	RAM	1 バイト+16 バイト× LONGQ_CFG_MAX_CTRL_BLKS		16 バイト× LONGQ_CFG_MAX_CTRL_BLKS		16 バイト× LONGQ_CFG_MAX_CTRL_BLKS	

注 1：LONGQ_CFG_PROTECT_QUEUE および LONGQ_CFG_CRITICAL_SECTION が 1 の場合

2.9 FIT モジュールの追加方法

本モジュールは、使用するプロジェクトごとに追加する必要があります。ルネサスでは、スマート・コンフィグレータを使用した(1)、(3)、(5)の追加方法を推奨しています。ただし、スマート・コンフィグレータは、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)、(4)の方法を使用してください。

(1) e^2 studio 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
e^2 studio のスマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: e^2 studio 編 (R20AN0451)」を参照してください。

(2) e^2 studio 上で FIT コンフィグレータを使用して FIT モジュールを追加する場合
e^2 studio の FIT コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加することができます。詳細は、アプリケーションノート「RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)」を参照してください。

(3) CS+上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
CS+上で、スタンドアロン版スマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: CS+編 (R20AN0470)」を参照してください。

(4) CS+上で FIT モジュールを追加する場合
CS+上で、手動でユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」を参照してください。

(5) IAREW 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
スタンドアロン版スマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: IAREW 編 (R20AN0535)」を参照してください。

2.10 for 文、while 文、do while 文について

本モジュールでは、レジスタの反映待ち処理などで for 文、while 文、do while 文（ループ処理）を使用しています。これらループ処理には、「WAIT_LOOP」をキーワードとしたコメントを記述しています。そのため、ループ処理にユーザがフェイルセーフの処理を組み込む場合、「WAIT_LOOP」で該当の処理を検索できます。

以下に記述例を示します。

while 文の例：

```
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}
```

for 文の例：

```
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}
```

do while 文の例：

```
/* Reset completion waiting */
do
{
    reg = phy read(ether channel, PHY REG CONTROL);
    count++;
 } while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

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_LONGQ_Open()	ユーザが用意したバッファ領域に対してキューコントロールブロック（QCB）を割り当て、これを初期化します。他の API 関数で使用するキューハンドルを返します。
R_LONGQ_Close()	ハンドルに対応するキューコントロールブロックを解放します。
R_LONGQ_Put()	キューにエントリを追加します。
R_LONGQ_Get()	キューから最も古いエントリを取り出します。
R_LONGQ_Flush()	キューを空の状態にリセットします。
R_LONGQ_Used()	キューで使用されているエントリ数を読み出します。
R_LONGQ_Unused()	キューで未使用のエントリ数を返します。
R_LONGQ_GetVersion()	実行時にモジュールのバージョンを返します。

3.2 戻り値

以下は API 関数が返すエラーコードです。enum は API 関数の宣言とともに r_longq_if.h に含まれます。

```
typedef enum LONGQ_err           // LONGQ API エラーコード
{
    LONGQ_SUCCESS = 0,
    LONGQ_ERR_NULL_PTR,          // 受け取った ptr が NULL です。要求される引数がありません。
    LONGQ_ERR_INVALID_ARG,       // パラメータに対して引数が無効です。
    LONGQ_ERR_MALLOC_FAIL,       // コントロールブロックを確保できません。ヒープサイズを
                                 // 増やしてください。
    LONGQ_ERR_NO_MORE_CTRL_BLKS, // コントロールブロックを割り当てられません。
                                 // LONGQ_MAX_CTRL_BLKS を増やしてください。
    LONGQ_ERR_QUEUE_FULL,        // キューがいっぱいです。これ以上エントリを追加できません。
    LONGQ_ERR_QUEUE_EMPTY        // キューが空です。エントリが取り出せません。
} longq_err_t;
```

3.3 R_LONGQ_Open()

この関数はユーザが用意したバッファ領域に対してキューコントロールブロック（QCB）を割り当て、これを初期化します。他の API 関数で使用するキューハンドルを返します。

Format

```
longq_err_t R_LONGQ_Open(uint32_t * const p_buf,
                         uint16_t const size,
                         bool const ignore_overflow,
                         longq_hdl_t * const p_hdl)
```

Parameters

p_buf

バッファのポインタ

size

バッファサイズ (単位：エレメント数)

ignore_overflow

true＝キューがフルの場合、最も古いエントリを上書きしてエントリの追加を継続します。
false＝Put()関数が呼び出されて、キューがフルだった場合、エラーを返します。

p_hdl

キューのハンドルへのポインタ (ここに値を設定)

Return Values

```
LONGQ_SUCCESS                /* キューが正常に初期化されました。 */

LONGQ_ERR_NULL_PTR           /* ptr が NULL です。要求された引数がありません。 */
LONGQ_ERR_INVALID_ARG        /* サイズが 1 以下です。 */
LONGQ_ERR_MALLOC_FAIL        /* コントロールブロックを確保できません。ヒープサイズを
                                増やしてください。*/
LONGQ_ERR_NO_MORE_CTRL_BLKS      /* コントロールブロックを割り当てられません。
                                    config.h のLONGQ_MAX_CTRL_BLKS を増やしてください。 */
```

Properties

ファイル r_longq_if.h にプロトタイプ宣言されています。

Description

この関数は p_buf で指定されたバッファ領域に対してキューコントロールブロック（QCB）を配置または割り当てます。キューを空の状態に初期化し、p_hdl でコントロール構造体を示すハンドルを提供します。ハンドルは他の API 関数でキューの ID として使用されます。

Example

```
#define BUFSIZE 80

uint32_t tx_buf[BUFSIZE];
longq_hdl_t tx_que;
longq_err_t longq_err;

longq_err = R_LONGQ_Open(tx_buf, BUFSIZE, false, &tx_que);
```

Special Notes:

なし

3.4 R_LONGQ_Close()

この関数はハンドルに対応するキューコントロールブロックを解放します。

Format

```
longq_err_t R_LONGQ_Close(longq_hdl_t const hdl)
```

Parameters

hdl

キューのハンドル

Return Values

LONGQ_SUCCESS */* コントロールブロックは正常に解放されました。 */*

LONGQ_ERR_NULL_PTR */* hdl が NULL です。 */*

Properties

ファイル r_longq_if.h にプロトタイプ宣言されています。

Description

ハンドルに対応するキューのコントロールブロックが、実行時に動的に配置された場合（config.h の LONGQ_USE_HEAP_FOR_CTRL_BLKS を“1”に設定）、メモリはこの関数によって解放されます。コントロールブロックがコンパイル時に静的に配置された場合 （LONGQ_USE_HEAP_FOR_CTRL_BLKS を“0”に設定）、他のバッファ領域に対してこのコントロールブロックが使用可能であることを示します。このハンドルに対応するバッファの内容に影響はありません。

Example

```
longq_hdl_t tx_que;
longq_err_t longq_err;

longq_err = R_LONGQ_Open(tx_buf, BUFSIZE, false, &tx_que);

R_LONGQ_Close(tx_que);
```

Special Notes:

なし

3.5 R_LONGQ_Put()

この関数はキューにエントリを追加します。

Format

```
longq_err_t R_LONGQ_Put(longq_hdl_t const hdl,
                        uint32_t const datum)
```

Parameters

hdl

キューのハンドル

datum

キューに追加するエントリ

Return Values

LONGQ_SUCCESS	*/* エントリがキューに正常に追加されました。 */*
LONGQ_ERR_NULL_PTR	*/* hdl が NULL です。 */*
LONGQ_ERR_QUEUE_FULL	*/* キューがいっぱいです。キューにエントリを追加できません。*/*

Properties

ファイル r_longq_if.h にプロトタイプ宣言されています。

Description

この関数は hdl に対応するキューに datum の内容を追加します。Open()関数実行時、キューがフルで、ignore_overflow が false の場合、“LONG_ERR_QUEUE_FULL”を返します。Open()関数実行時、キューがフルで、ignore_overflow が true の場合、最も古いエントリを datum のエントリで上書きし、“LONGQ_SUCCESS”を返します。

Example

```
longq_hdl_t tx_que;
longq_err_t longq_err;
uint32_t data = 0x0056;

longq_err = R_LONGQ_Open(tx_buf, BUFSIZE, false, &tx_que);
longq_err = R_LONGQ_Put(tx_que, data);
```

Special Notes:

なし

3.6 R_LONGQ_Get()

この関数はキューからエントリを取り出します。

Format

```
longq_err_t R_LONGQ_Get(longq_hdl_t const hdl,
                        uint32_t * const p_datum)
```

Parameters

hdl

キューのハンドル

p_datum

エントリの呼び出し先のポインタ

Return Values

LONGQ_SUCCESS	*/* キューからエントリが正常に取り出されました。 */*
LONGQ_ERR_NULL_PTR	*/* hdl または p_datum が NULL です。 */*
LONGQ_ERR_QUEUE_EMPTY	*/* キューは空です。読み出すデータがありません。 */*

Properties

ファイル r_longq_if.h にプロトタイプ宣言されています。

Description

この関数は（可能な場合）hdl に対応するキューから最も古いエントリを取り出し、そのデータを p_datum で示される場所に読み出します。

Example

```
longq_hdl_t rx_que;
longq_err_t longq_err;
uint32_t data;

longq_err = R_LONGQ_Open(rx_buf, BUFSIZE, false, &rx_que);

/* queue filled with data by R_LONGQ_Put()elsewhere */
longq_err = R_LONGQ_Get(rx_que, &data);
```

Special Notes:

なし

3.7 R_LONGQ_Flush()

この関数はキューを空の状態にリセットします。

Format

```
longq_err_t R_LONGQ_Flush(longq_hdl_t const hdl)
```

Parameters

hdl

キューのハンドル

Return Values

LONGQ_SUCCESS */* キューは正常にリセットされました。 */*

LONGQ_ERR_NULL_PTR */* hdl が NULL です。 */*

Properties

ファイル r_longq_if.h にプロトタイプ宣言されています。

Description

この関数は hdl で指定されたキューを空の状態にリセットします。

Example

```
longq_hdl_t rx_que;
longq_err_t longq_err;

long_err = R_LONGQ_Open(rx_buf, BUFSIZE, false, &rx_que);

/* queue filled with data by R_LONGQ_Put()elsewhere */
R_LONGQ_Flush(rx_que);
```

Special Notes:

なし

3.8 R_LONGQ_Used()

この関数はキューにあるエントリ数を読み出します。

Format

```
longq_err_t R_LONGQ_Used(longq_hdl_t const hdl,
                         uint16_t * const p_cnt)
```

Parameters

hdl

キューのハンドル

p_cnt

キューのエントリ数を格納する変数のポインタ

Return Values

LONGQ_SUCCESS */* キューにあるエントリ数は*p_cnt で正常に読み出されました。 */*

LONGQ_ERR_NULL_PTR */* hdl または p_cnt が NULL です。 */*

Properties

ファイル r_longq_if.h にプロトタイプ宣言されています。

Description

この関数は hdl に対応するキューにあるエントリ数を、p_cnt で示される場所に読み出します。

Example

```
longq_hdl_t rx_que;
longq_err_t longq_err;
uint16_t count;

longq_err = R_LONGQ_Open(rx_buf, BUFSIZE, false, &rx_que);

/* queue filled with data by R_LONGQ_Put()elsewhere */
R_LONGQ_Used(rx_que, &count);
```

Special Notes:

なし

3.9 R_LONGQ_Unused()

この関数はキュー内の未使用エントリ数を返します。

Format

```
longq_err_t R_LONGQ_Unused(longq_hdl_t const hdl,
                           uint16_t * const p_cnt)
```

Parameters

hdl

キューのハンドル

p_cnt

キューの未使用エントリ数が格納される変数のポインタ

Return Values

LONGQ_SUCCESS */* キューの未使用エントリ数は*p_cnt で正常に読み出されました。 */*

LONGQ_ERR_NULL_PTR */* hdl または p_cnt が NULL です。 */*

Properties

ファイル r_longq_if.h にプロトタイプ宣言されています。

Description

この関数は hdl に対応するキューの未使用エントリ数を、p_cnt で示される場所に読み出します。

Example

```
longq_hdl_t tx_que;
longq_err_t longq_err;
uint16_t count;

longq_err = R_LONGQ_Open(tx_buf, BUFSIZE, false, &tx_que);

/* queue filled with data by R_LONGQ_Put()elsewhere */
R_LONGQ_Unused(tx_que, &count);
```

Special Notes:

なし

3.10 R_LONGQ_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_LONGQ_GetVersion(void)
```

Parameters

なし

Return Values

バージョン番号

Properties

ファイル r_longq_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。

Example

```
uint32_t  version;

version = R_LONGQ_GetVersion();
```

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、本 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あります。

4.1 longq_demo_rskrx231, longq_demo_rskrx231_gcc

longq_demo_rskrx231, longq_demo_rskrx231_gcc は、LONGQ の API の使い方を示しています。このデモプロジェクトでは、キューの初期化、キューへの 32 ビットデータ格納、キューの要素数の取得、キューのデータを取得してバーチャルコンソールに出力します。また、LONGQ モジュールのバージョン番号をバーチャルコンソールに出力します。バーチャルコンソールは、「コンソールを開く」> 「Renesas デバッグ仮想コンソール」を選択すると使用できます。

4.2 longq_demo_rskrx71m, longq_demo_rskrx71m_gcc

longq_demo_rskrx71m, longq_demo_rskrx71m_gcc は、LONGQ の API の使い方を示しています。このデモプロジェクトでは、キューの初期化、キューへの 32 ビットデータ格納、キューの要素数の取得、キューのデータを取得してバーチャルコンソールに出力します。また、LONGQ モジュールのバージョン番号をバーチャルコンソールに出力します。バーチャルコンソールは、「コンソールを開く」> 「Renesas デバッグ仮想コンソール」を選択すると使用できます。

4.3 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「終了」をクリックします。

4.4 デモのダウンロード方法

デモプロジェクトは、RX Driver Package には同梱されていません。デモプロジェクトを使用する場合、個別に各 FIT モジュールをダウンロードする必要があります。「スマートブラウザ」の「アプリケーションノート」タブから、本アプリケーションノートを右クリックして「サンプル・コード（ダウンロード）」を選択することにより、ダウンロードできます。

5. 付録

5.1 動作確認環境

本 FIT モジュールの動作確認環境を以下に示します。

表 5.1 動作確認環境 (Rev.1.60)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V4.2.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V2.04.01
	コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev1.60
使用ボード	Renesas Starter Kit+ for RX65N (型名：RTK500565NSxxxxxBE)

表 5.2 動作確認環境 (Rev.1.70)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.0.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V2.08.00
	コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev1.70
使用ボード	Renesas Starter Kit for RX231 (型名：R0K505231SxxxBE) Renesas Starter Kit+ for RX71M (型名：R0K50571MSxxxBE)

表 5.3 動作確認環境 (Rev.1.71)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.1.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.00.00
	コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev1.71

表 5.4　動作確認環境 (Rev.1.80)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.3.0 IAR Embedded Workbench for Renesas RX 4.10.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.8.4.201801 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	IAR C/C++ Compiler for Renesas RX version 4.10.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev1.80
使用ボード	Renesas Starter Kit for RX231 (型名：R0K505231xxxxxx)

表 5.5　動作確認環境 (Rev.1.82)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio Version 2020-10 (20.10.0)
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.02.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev1.82
使用ボード	Renesas Starter Kit for RX231 (型名：R0K505231SxxxBE) Renesas Starter Kit+ for RX71M (型名：R0K50571MSxxxBE)

表 5.6　動作確認環境 (Rev.1.90)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio 2021-07 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.03.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 8.3.0.202102 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、統合開発環境のデフォルト設定に以下のオプションを追加 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン／リトルエンディアン
モジュールのリビジョン	Rev1.90
使用ボード	Renesas Starter Kit+ for RX671（型名：RTK55671xxxxxxxxxx）

表 5.7　動作確認環境 (Rev.2.00)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio 2022-10 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.04.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 8.3.0.202202 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、統合開発環境のデフォルト設定に以下のオプションを追加 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン／リトルエンディアン
モジュールのリビジョン	Rev2.00
使用ボード	Renesas Starter Kit for RX71M（型名：R0K50571MC000BE） Renesas Starter Kit for RX231 (型名：R0K505231S000BE)

5.2 トラブルシューティング

(1) Q : 本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「Could not open source file "platform.h"」エラーが発生します。

A : FIT モジュールがプロジェクトに正しく追加されていない可能性があります。プロジェクトへの追加方法をご確認ください。

- CS+を使用している場合
  アプリケーションノート RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」
- e^2 studio を使用している場合
  アプリケーションノート RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)」

また、本 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。

6. 参考ドキュメント

ユーザーズマニュアル：ハードウェア
（最新版をルネサス エレクトロニクスホームページから入手してください）

テクニカルアップデート／テクニカルニュース
（最新の情報をルネサス エレクトロニクスホームページから入手してください）

ユーザーズマニュアル：開発環境
RX ファミリ CC-RX コンパイラ ユーザーズマニュアル（R20UT3248）
（最新版をルネサス エレクトロニクスホームページから入手してください）

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。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.30	Mar.02.15	—	初版発行
1.40	Jun.30.15	—	FIT モジュールが RX231 グループに対応
1.50	Sep.30.15	—	FIT モジュールが RX23T グループに対応
		5	2.2「ソフトウェアの要求」に r_bsp を追加
		6	2.8「コードサイズ」のコードサイズを更新
		8	3.2「戻り値」の定義“LONGQ_ERR_SUCCESS”を削除。
1.60	Jan.29.16	6	2.8「コードサイズ」のコードサイズを更新
		17	4「デモプロジェクト」の章を追加
		プログラム	RX ファミリに対応 RX ファミリのシリーズ/グループ/ボードに依存しないで組み込めるよう XML ファイルを修正 R_LONGQ_Open 関数の初期設定処理を修正 コーディングルールに従ってプログラムを修正
1.70	Jun.01.18	—	Smart Configurator での GUI によるコンフィグオプション設定機能に対応
		—	デモプロジェクトを更新
		5	2.4「サポートされているツールチェーン」のツールチェーンを変更
		6	2.7「コンパイル時の設定」の LONGQ_CFG_MAX_CTRL_BLKS のデフォルト値を変更 2.8「コードサイズ」のコードサイズを更新
		7	2.9「FIT モジュールの追加方法」の章を追加
		8	2.10「for 文、while 文、do while 文について」の章を追加
		18	4.4「デモのダウンロード方法」の章を追加
		19	5.「付録」の章を追加 5.1「動作確認環境」の章を追加
		20	5.2「トラブルシューティング」の章を追加
		21	6.「参考ドキュメント」の章を追加
		プログラム	以下のマクロ定義のデフォルト値を変更。 ・LONGQ_CFG_MAX_CTRL_BLKS (変更) (2) ⇒ (32)
1.71	Dec.03.18	19	5.1 動作確認環境 表 5.3 動作確認環境 (Rev.1.71)を追加。
		プログラム	FIT モジュールのサンプルプログラムをダウンロードするためのアプリケーションノートのドキュメント番号を xml ファイルに追加。
1.80	Feb.07.19	—	以下のコンパイラに対応 ・GCC for Renesas RX ・IAR C/C++ Compiler for Renesas RX
		1	「対象コンパイラ」の章を追加 関連ドキュメントを削除
		7	2.8「コードサイズ」の章を更新
		18	3.10「R_LONGQ_GetVersion()」の章を更新
		21	5.1「動作確認環境」の章を更新
		23	「ホームページとサポート窓口」の章を削除

1.80	Feb.07.19	プログラム	R_LONGQ_GetVersion 関数のインライン展開を削除
1.81	Jun.10.20	—	API 関数のコメントを Doxygen スタイルに変更
		9	2.9「FIT モジュールの追加方法」の章を更新
		12..19	3. 「API 関数」の、各 API の Reentrant を削除
1.82	Nov.30.20	—	開発環境のバージョンアップに伴い、サンプルコードのプロジェクトを更新。
1.90	Oct.29.21	6	「2.7 コンパイル時の設定」を更新。 新しいマクロ定義 LONGQ_CFG_PROTECT_QUEUE および LONGQ_CFG_CRITICAL_SECTION を追加。
		7	2.8「コードサイズ」の章を更新
		13..17	3. API 関数 5 つの API の”Special Notes”を削除。 R_LONGQ_Put, R_LONGQ_Get, R_LONGQ_Flush, R_LONGQ_Used, R_LONGQ_Unused。
		21	5.1 動作確認環境 表 5.6 動作確認環境 (Rev.1.90)を追加。
		プログラム	関数 R_LONGQ_Put, R_LONGQ_Get, R_LONGQ_Flush, R_LONGQ_Used, R_LONGQ_Unused にキューの保護処理を追加。
2.00	Nov.30.22	19 22	デモプロジェクトの更新と追加 5.1 動作確認環境 表 5.7 動作確認環境 (Rev.2.00)を追加。
		プログラム	デモプロジェクトの更新と追加

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. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   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. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「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。
5. クロックについて
   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。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   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、将来の拡張機能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。
8. 製品間の相違について
   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、フラッシュメモリ、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、ノイズ幅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。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、お客様の責任において、お客様の機器・システムを設計ください。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. 当社製品または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. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング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。
6. 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、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、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（宇宙機器と、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想定していません。たとえ、当社が想定してい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。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 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。
9. 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、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、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。
10. 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。
11. 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。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。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 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。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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直接的、間接的に支配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.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。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。