RX ファミリ

バイト型キューバッファ（BYTEQ）モジュール Firmware Integration Technology

R01AN1683JU0150
Rev. 1.50
2015.09.30

要旨

このモジュールはバイト型のリングバッファを構成し管理する関数を提供します。

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)
- e^{2}studio に組み込む方法 Firmware Integration Technology (R01AN1723JU)
- CS+に組み込む方法 Firmware Integration Technology (R01AN1826JJ)

目次

1. 概要

バイト型キューバッファ（BYTEQ）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_BYTEQ_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_byteq_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。

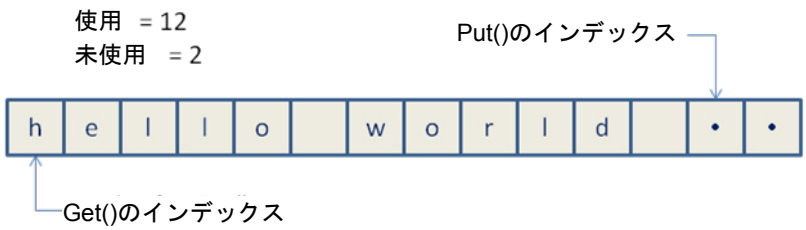
1 つのバッファに対して 1 つのコントロールブロックが用意されます。R_BYTEQ_Open()が実行される際にアプリケーションのバッファ領域のポインタとサイズが渡され、QCB へのポインタが返されます。このポインタはハンドルと呼ばれ、以後、他のすべての API 関数に渡されます。API 関数はハンドルで示されるキューに対して操作を行います。複数のキュー間で、グローバルまたは静的なデータが共有されることはありませんので、API 関数は別のキューに対して再入可能です。

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。

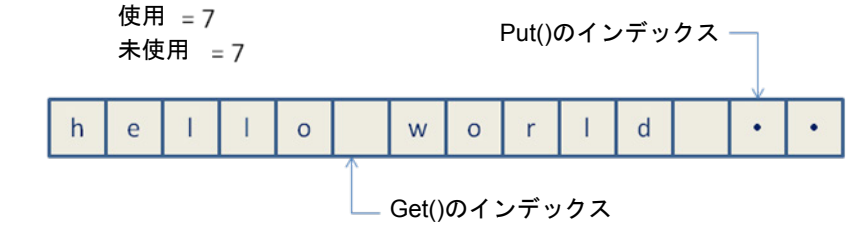
1.1 BYTEQ モジュールを使用する

以下に API 呼び出し時のキューの動きを説明します。

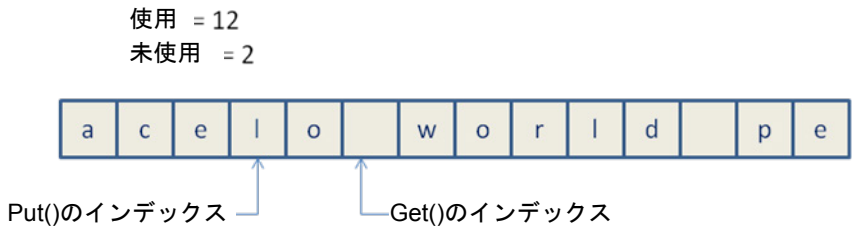
```
#define BUFSIZE 14

uint8_t      my_buf[BUFSIZE];
byteq_hdl_t   my_que;
byteq_err_t   err;
uint8_t      byte;

err = R_BYTEQ_Open(my_buf, BUFSIZE, &my_que);

// add 12 bytes to queue
R_BYTEQ_Put(my_que, 'h');
R_BYTEQ_Put(my_que, 'e');
R_BYTEQ_Put(my_que, 'l');
R_BYTEQ_Put(my_que, 'l');
R_BYTEQ_Put(my_que, 'o');
R_BYTEQ_Put(my_que, ' ');
R_BYTEQ_Put(my_que, 'w');
R_BYTEQ_Put(my_que, 'o');
R_BYTEQ_Put(my_que, 'r');
R_BYTEQ_Put(my_que, 'l');
R_BYTEQ_Put(my_que, 'd');
R_BYTEQ_Put(my_que, ' ');
```



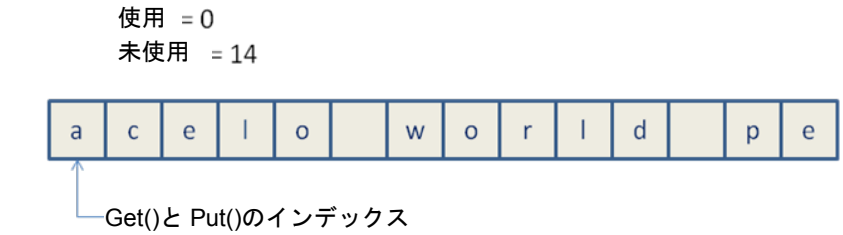
```
// remove 5 bytes from queue
R_BYTEQ_Get(my_que, &byte);     // byte = 'h'
R_BYTEQ_Get(my_que, &byte);     // byte = 'e'
R_BYTEQ_Get(my_que, &byte);     // byte = 'l'
R_BYTEQ_Get(my_que, &byte);     // byte = 'l'
R_BYTEQ_Get(my_que, &byte);     // byte = 'o'
```




```
// add 5 bytes to queue
R_BYTEQ_Put(my_que, 'p');
R_BYTEQ_Put(my_que, 'e');
R_BYTEQ_Put(my_que, 'a');
R_BYTEQ_Put(my_que, 'c');
R_BYTEQ_Put(my_que, 'e');
```




```
// flush queue
R_BYTEQ_Flush(my_que);
```



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.3.00

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 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（RX23T 以外）
- Renesas RX Toolchain v.2.03.00（RX23T）

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_byteq¥ref¥r_byteq_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_byteq_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_byteq_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_byteq¥r_byteq_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_byteq_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（固定幅整数型）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 r_byteq_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_byteq_config.h）	
#define BYTEQ_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 BYTEQ_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 BYTEQ_CFG_MAX_CTRL_BLKS ※デフォルト値は“4”	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 BYTEQ_CFG_USE_HEAP_FOR_CTRL_BLKS が“1”の時には無視されます。

2.8　コードサイズ

ツールチェーンでのコードサイズは、最適化レベル 2、およびコードサイズ重視の最適化を前提としたサイズです。ROM (コードおよび定数) と RAM (グローバルデータ) のサイズは、本モジュールのコンフィギュレーションヘッダファイルで設定される、ビルド時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。

ROM および RAM のコードサイズ		
	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: 262 + malloc()/free()に 313 = 575 バイト	ROM: 170 + malloc()/free()に 313 = 483 バイト
	RAM: 0 + malloc()/free()に 8 = 8 バイト	RAM: 0 + malloc()/free()に 8 = 8 バイト
コントロールブロックをコンパイル時に配置	ROM: 303 バイト	ROM: 208 バイト
	RAM: 1 + 12 x BYTEQ_CFG_MAX_CTRL_BLKS	RAM: 1 + 12 x BYTEQ_CFG_MAX_CTRL_BLKS

2.9　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(R01AN1723JU)」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(R01AN1685JU)」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_BYTEQ_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_BYTEQ_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_BYTEQ_Put()	キューにバイトデータを追加します。
R_BYTEQ_Get()	キューから最も古いバイトデータを取り出します。
R_BYTEQ_Flush()	キューを空の状態にリセットします。
R_BYTEQ_Used()	キューで使用されているバイト数を読み出します。
R_BYTEQ_Unused()	キューで未使用のバイト数を読み出します。
R_BYTEQ_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_byteq_if.h に含まれます。

```
typedef enum _byteq_err          // BYTEQ API エラーコード
{
    BYTEQ_SUCCESS = 0,
    BYTEQ_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。
    BYTEQ_ERR_INVALID_ARG,       // パラメータに対して引数が無効です。
    BYTEQ_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。
    BYTEQ_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。
                                 // BYTEQ_MAX_CTRL_BLKS を増やしてください。
    BYTEQ_ERR_QUEUE_FULL,        // キューがいっぱいです。これ以上バイトを追加できません。
    BYTEQ_ERR_QUEUE_EMPTY        // キューが空です。バイトが取り出せません。
} byteq_err_t;
```

3.3 R_BYTEQ_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

```
byteq_err_t R_BYTEQ_Open(uint8_t * const p_buf,
                         uint16_t const size,
                         byteq_hdl_t * const p_hdl)
```

Parameters

p_buf

バイト型バッファのポインタ

size

バッファサイズ (単位：バイト)

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

```
BYTEQ_SUCCESS           /* キューが正常に初期化されました。 */
BYTEQ_ERR_NULL_PTR      /* p_buf が NULL です。 */
BYTEQ_ERR_INVALID_ARG   /* サイズが 1 以下です。 */
BYTEQ_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。*/
BYTEQ_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
                               BYTEQ_MAX_CTRL_BLKS を増やしてください。 */
```

Properties

ファイル r_byteq_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。

Reentrant

この関数は、別のバッファ領域に対して再入可能（リエントラント）です。

Example

```
#define BUFSIZE 80

uint8_t tx_buf[BUFSIZE];
byteq_hdl_t tx_que;
byteq_err_t byteq_err;

byteq_err = R_BYTEQ_Open(tx_buf, BUFSIZE, &tx_que);
```

Special Notes:

なし

3.4　R_BYTEQ_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

```
byteq_err_t R_BYTEQ_Close(byteq_hdl_t const hdl)
```

Parameters

hdl

キューのハンドル

Return Values

BYTEQ_SUCCESS　　*/* コントロールブロックは正常に解放されました。 */*
BYTEQ_ERR_NULL_PTR　　*/* hdl が NULL です。 */*

Properties

ファイル r_byteq_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 BYTEQ_USE_HEAP_FOR_CTRL_BLKS を“1”に設定）、本関数は（free()関数によって）メモリを解放します。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（BYTEQ_USE_HEAP_FOR_CTRL_BLKS を“0”に設定）、本関数は、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。

Reentrant

この関数は、別のキューに対して再入可能（リエントラント）です。

Example

```
byteq_hdl_t tx_que;
byteq_err_t byteq_err;

byteq_err = R_BYTEQ_Open(tx_buf, BUFSIZE, &tx_que);

byteq_err = R_BYTEQ_Close(tx_que);
```

Special Notes:

なし

3.5　R_BYTEQ_Put()

この関数はキューにバイトデータを追加します。

Format

```
byteq_err_t R_BYTEQ_Put(byteq_hdl_t const hdl,
                        uint8_t const byte)
```

Parameters

hdl

キューのハンドル

byte

キューに追加するバイト

Return Values

```
BYTEQ_SUCCESS          /* byte の内容がキューに正常に追加されました。 */
BYTEQ_ERR_NULL_PTR     /* hdl が NULL です。 */
BYTEQ_ERR_QUEUE_FULL   /* キューがいっぱいです。キューに byte の内容を追加できません。*/
```

Properties

ファイル r_byteq_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 byte の内容を追加します。

Reentrant

この関数は、別のキューに対して再入可能（リエントラント）です。

Example

```
byteq_hdl_t tx_que;
byteq_err_t byteq_err;
uint8_t byte = 'A';

byteq_err = R_BYTEQ_Open(tx_buf, BUFSIZE, &tx_que);
byteq_err = R_BYTEQ_Put(tx_que, byte);
```

Special Notes:

割り込みとアプリケーションの双方がキューにアクセスする場合、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。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。

3.6　R_BYTEQ_Get()

この関数はキューからバイトデータを取り出します。

Format

```
byteq_err_t R_BYTEQ_Get(byteq_hdl_t const hdl,
                        uint8_t * const p_byte)
```

Parameters

hdl

キューのハンドル

p_byte

バイトデータの呼び出し先のポインタ

Return Values

```
BYTEQ_SUCCESS          /* キューからバイトデータが正常に取り出されました。 */
BYTEQ_ERR_NULL_PTR     /* hdl が NULL です。 */
BYTEQ_ERR_INVALID_ARG  /* p_byte が NULL です。 */
BYTEQ_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_byteq_if.h にプロトタイプ宣言されています。

Description

この関数は hdl に対応するキューから最も古いバイトデータを取り出し、そのデータを p_byte で示される場所に読み出します。

Reentrant

この関数は、別のキューに対して再入可能（リエントラント）です。

Example

```
byteq_hdl_t rx_que;
byteq_err_t byteq_err;
uint8_t byte;

byteq_err = R_BYTEQ_Open(rx_buf, BUFSIZE, &rx_que);

/* queue filled with data by R_BYTEQ_Put()elsewhere */
byteq_err = R_BYTEQ_Get(rx_que, &byte);
```

Special Notes:

割り込みとアプリケーションの双方がキューにアクセスする場合、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。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。

3.7　R_BYTEQ_Flush()

この関数はキューを空の状態にリセットします。

Format

```
byteq_err_t R_BYTEQ_Flush(byteq_hdl_t const hdl)
```

Parameters

hdl

キューのハンドル

Return Values

BYTEQ_SUCCESS　　*/* *キューは正常にリセットされました。* */*
BYTEQ_ERR_NULL_PTR　　*/* *hdl が NULL です。* */*

Properties

ファイル r_byteq_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。

Reentrant

この関数は、別のキューに対して再入可能（リエントラント）です。

Example

```
byteq_hdl_t rx_que;
byteq_err_t byteq_err;

byteq_err = R_BYTEQ_Open(rx_buf, BUFSIZE, &rx_que);

/* queue filled with data by R_BYTEQ_Put()elsewhere */
byteq_err = R_BYTEQ_Flush(rx_que);
```

Special Notes:

割り込みとアプリケーションの双方がキューにアクセスする場合、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。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。

3.8 R_BYTEQ_Used()

この関数はキューにあるデータバイト数を読み出します。

Format

```
byteq_err_t R_BYTEQ_Used(byteq_hdl_t const hdl,
                         uint16_t * const p_cnt)
```

Parameters

hdl

キューのハンドル

p_cnt

キューのデータバイト数を格納する変数のポインタ

Return Values

```
BYTEQ_SUCCESS       /* キューにあるバイト数は*p_cnt で正常に読み出されました。 */
BYTEQ_ERR_NULL_PTR  /* hdl が NULL です。 */
BYTEQ_ERR_INVALID_ARG  /* p_cnt が NULL です。 */
```

Properties

ファイル r_byteq_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バイト数を、p_cnt で示される場所に読み出します。

Reentrant

この関数は再入可能（リエントラント）です。

Example

```
byteq_hdl_t rx_que;
byteq_err_t byteq_err;
uint16_t count;

byteq_err = R_BYTEQ_Open(rx_buf, BUFSIZE, &rx_que);

/* queue filled with data by R_BYTEQ_Put()elsewhere */
byteq_err = R_BYTEQ_Used(rx_que, &count);
```

Special Notes:

割り込みとアプリケーションの双方がキューにアクセスする場合、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。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。

3.9 R_BYTEQ_Unused()

この関数はキュー内の未使用データバイト数を読み出します。

Format

```
byteq_err_t R_BYTEQ_Unused(byteq_hdl_t const hdl,
                           uint16_t * const p_cnt)
```

Parameters

hdl

キューのハンドル

p_cnt

キューの未使用バイト数が格納される変数のポインタ

Return Values

BYTEQ_SUCCESS	*/* キューの未使用バイト数は*p_cnt で正常に読み出されました。 */*
BYTEQ_ERR_NULL_PTR	*/* hdl が NULL です。 */*
BYTEQ_ERR_INVALID_ARG	*/* p_cnt が NULL です。 */*

Properties

ファイル r_byteq_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バイト数を、p_cnt で示される場所に読み出します。

Reentrant

この関数は再入可能（リエントラント）です。

Example

```
byteq_hdl_t tx_que;
byteq_err_t byteq_err;
uint16_t count;

byteq_err = R_BYTEQ_Open(tx_buf, BUFSIZE, &tx_que);

/* queue filled with data by R_BYTEQ_Put()elsewhere */
byteq_err = R_BYTEQ_Unused(tx_que, &count);
```

Special Notes:

割り込みとアプリケーションの双方がキューにアクセスする場合、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。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。

3.10　R_BYTEQ_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_BYTEQ_GetVersion(void)
```

Parameters

なし

Return Values

バージョン番号

Properties

ファイル r_byteq_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

```
uint32_t   version;

version = R_BYTEQ_GetVersion();
```

Special Notes:

この関数は#pragma inline ディレクティブを使用してインライン化されています。

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

Rev.	発行日	改訂内容	
		ページ	ポイント
1.30	2015.03.02	—	初版発行
1.40	2015.06.30	—	FIT モジュールの RX231 グループ対応
1.50	2015.09.30	— 5 6	FIT モジュールの RX23T グループ対応 2.2 「ソフトウェアの要求」に r_bsp を追加 2.8 「コードサイズ」のコードサイズを更新

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。

RENESAS

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　〒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。
総合お問合せ窓口：http://japan.renesas.com/contact/

Colophon 3.0