

RXファミリ

R01AN0514JJ0233

Rev.2.33

Jul 31, 2019

USB Peripheral Mass Storage Class Driver (PMSC)

要旨

本アプリケーションノートは、USB Peripheral Mass Storage Class Driver(PMSC)について説明します。本ドライバは USB Basic Peripheral Driver () と組み合わせることで動作します。以降、本ドライバを PMSC と称します。

対象デバイス

RX62N/RX621 グループ

RX63N/RX631 グループ

RX630 グループ

RX63T 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

1. USB Revision 2.0 Specification
2. USB Mass Storage Class Specification Overview Revision 1.1
3. USB Mass Storage Class Bulk-Only Transport Revision 1.0
【<http://www.usb.org/developers/docs/>】
4. RX62N/RX621 グループユーザーズマニュアル ハードウェア編 (ドキュメント No.R01UH0033)
5. RX63N/RX631 グループユーザーズマニュアル ハードウェア編 (ドキュメント No.R01UH0041)
6. RX630 グループユーザーズマニュアル ハードウェア編 (ドキュメント No.R01UH0040)
7. RX63T グループユーザーズマニュアル ハードウェア編 (ドキュメント No.R01UH0238)
8. USB Basic Host and Peripheral Driver アプリケーションノート(ドキュメント No. R01AN0512)

— ルネサス エレクトロニクスホームページ

【<http://japan.renesas.com/>】

— USB デバイスページ

【<http://japan.renesas.com/prod/usb/>】

目次

1. 概要	3
2. ソフトウェア構成.....	5
3. API情報	6
4. クラスドライバ概要.....	8
5. デバイスクラスドライバ (PDCD)	9
6. API.....	10
7. コンフィグレーション (r_usb_pmsc_config.h).....	11
8. メディアドライバインタフェース	13
9. サンプルアプリケーション	23
10. セットアップ.....	28
11. アプリケーションの作成方法	32
12. e ² studio用プロジェクトをCS+で使用する場合	33

1. 概要

PMSC は、と組み合わせることで、Peripheral Mass Storage Class Driver(PMSC)として動作します。PMSC は、USB マスストレージクラスの Bulk-Only Transport(BOT)プロトコルで構築されています。USB ペリフェラルコントローラドライバ、メディアドライバと組み合わせることで、BOT 対応のストレージ機器として USB ホストと通信を行うことができます。

以下に、本モジュールがサポートしている機能を示します。

- ・ BOTプロトコルによるストレージコマンド制御
- ・ USBホストからのマスストレージデバイスクラスリクエストに対する応答
- ・ BOT (Bulk Only Transport) にカプセル化されたストレージコマンドに対する応答

1.1 必ずお読みください

このドライバを使ってアプリケーションプログラムを作成する場合は、USB Basic Host and Peripheral Driver アプリケーションノート(ドキュメント No.R01AN0512)を参照いただきますようお願いいたします。このアプリケーションノートは、パッケージ内の"reference_documents"フォルダにあります。

1.2 制限事項

1. USB Host から送信されるマスストレージクラスコマンド(GetMaxLun)に対し、本ドライバは値 0 を返します。
2. 本ドライバがサポートするセクタサイズは 512 のみです。
3. DMA/DTC をサポートしていません。

1.3 注意事項

1. このドライバは、USB 通信動作を保証するものではありません。システムに適用される場合は、お客様における動作検証はもとより、多種多様なデバイスに対する接続確認を実施してください。
2. RX62N/RX621/RX63T/RX630 をご使用の場合、必ず"reference_documents"フォルダ下のドキュメント([r01an0514jj0232_usb.pdf](#), [r01an0512jj0232_usb.pdf](#))をご使用いただきますようお願いいたします。
3. ストレージ領域として使用するメディアを制御するメディアドライバ関数はお客様において実装いただく必要があります。
4. 本ドライバはストレージメディア領域として、以下の領域を使用しています。

MCU	ストレージメディア領域	ファイルシステム
RX62N	RSK 上の SDRAM	FAT16
RX63N	RSK 上の SDRAM	FAT16
RX63T	内蔵 RAM	FAT12
RX630	内蔵 RAM	FAT12

[Note]

以下の条件をすべて満たす場合、必ず、Windows®のタスクバーにある「ハードウェアの安全な取り出し」画面から Eject 処理を行った後で MSC デバイスをデタッチしてください。

- a. ストレージメディアを FAT12 でフォーマットしている。
- b. MSC Host に Windows® 8.1 または Windows 10 を使用している。

1.4 用語一覧

APL : Application program

BOT	:	Mass storage class Bulk Only Transport
DDI	:	Device Driver Interface
PCD	:	Peripheral Control Driver of
PCI	:	PCD Interface
PDCD	:	Periphera, Device Class Driver
PMSCD	:	Peripheral Mass Storage USB Class Driver (PMSCF + PCI + DDI)
PMSCF	:	Peripheral Mass Storage Class Function
PMSDD	:	Peripheral Mass Storage Device Driver (sample ATAPI driver)
RSK	:	Renesas Starter Kits
	:	USB Basic Peripheral Driver for
USB	:	Universal Serial Bus

2. ソフトウェア構成

PMSC は、PMSCD と PMSDD で構成されています。

PMSCD は、BOT プロトコル制御及びデータ送受信を行う PMSCF、PMSDD に対するインタフェース関数群 (DDI)、PCD に対するインタフェース関数群 (PCI) で構成されています。

PMSDD は、PCD を介してホストとの BOT プロトコル通信を行います。PMSDD では、PMSCD から受け取ったストレージコマンドを解析し実行します。

Figure 2-1にモジュール構成、Table 2-1にモジュール機能概要を示します。

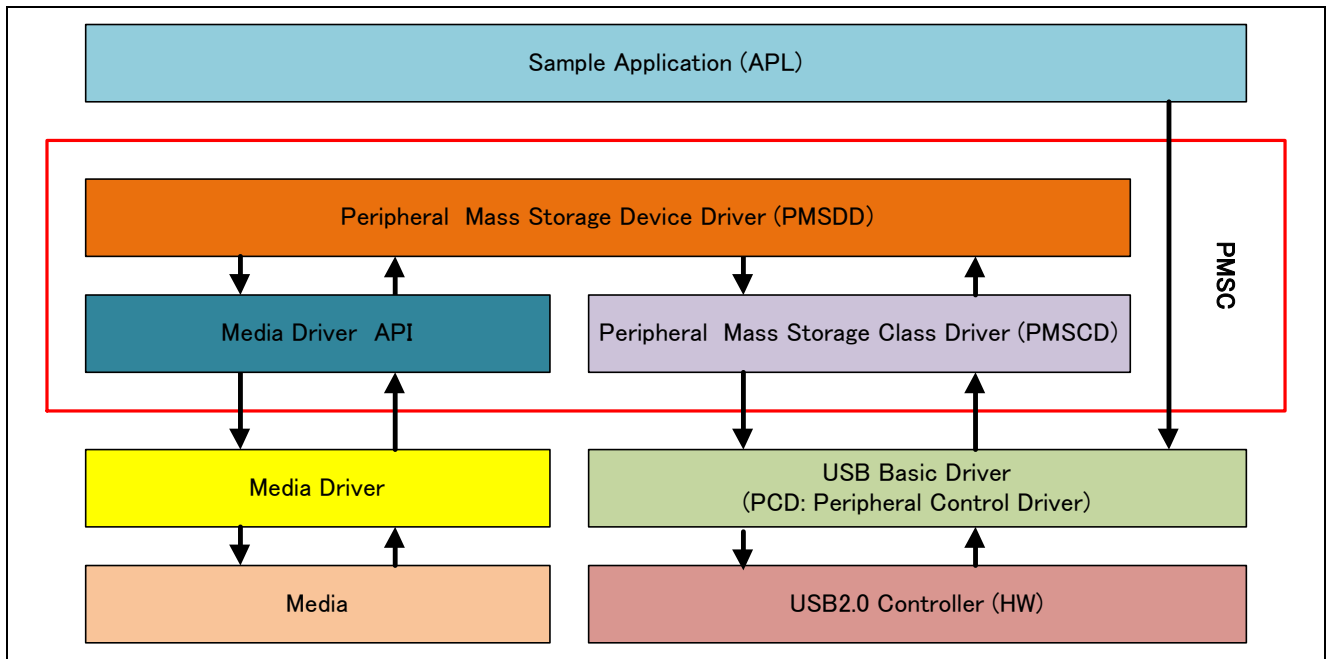


Figure 2-1 ソフトウェア構成図

Table 2-1 各モジュール機能概要

モジュール名	機能概要
PMSDD	マスタストレージデバイスドライバ <ul style="list-style-type: none"> PMSCD からのストレージコマンドの処理を行う Media driver を介して Media へのアクセスを行う
DDI	PMSDD-PMSCD 間のインタフェース関数
PMSCF	マスタストレージクラスドライバ <ul style="list-style-type: none"> BOT プロトコルデータの制御、クラスリクエストの対応を行う CBW の解析、データ送受信を行う PMSDD/PCD との連携し CSW を生成する
PCI	PMSCD - PCD 間のインタフェース関数
PCD	USB Peripheral H/W 制御ドライバ

3. API 情報

本ドライバの API はルネサスの API の命名基準に従っています。

3.1 ハードウェアの要求

ご使用になる MCU が以下の機能をサポートしている必要があります。

- USB

3.2 動作確認環境

このドライバの動作確認環境を以下に示します。

Table 3-1 動作確認環境

項目	内容
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 V.3.01.00 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
エンディアン	リトルエンディアン / ビッグエンディアン
使用ボード	Renesas Starter Kits for RX63N
ホスト環境	下記の OS に接続し動作確認を行っています。 1. Windows® 7 2. Windows® 8.1 3. Windows® 10

3.3 使用する割り込みベクタ

このドライバが使用する割り込みベクタを以下に示します。

Table 3-2 使用する割り込みベクター一覧

デバイス	割り込みベクタ
RX63N/RX631	USBIO 割り込み(ベクタ番号: 35) / USBR0 割り込み(ベクタ番号: 90) USB11 割り込み(ベクタ番号: 38) / USBR1 割り込み(ベクタ番号: 91)

3.4 ヘッダファイル

すべての API 呼び出しとそれをサポートするインタフェース定義は `r_usb_basic_if.h` と `r_usb_pmesc_if.h` に記載されています。

3.5 整数型

このプロジェクトは ANSI C99 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。

3.6 コンパイル時の設定

コンパイル時の設定については、本書の「7. コンフィグレーション (r_usb_pmesc_config.h)」と USB Basic Host and Peripheral Driver アプリケーションノート(ドキュメント No.R01AN0512)内の「コンフィグレーション」の章を参照してください。

3.7 ROM/RAM サイズ

本ドライバの ROM/RAM サイズを以下に示します。

	引数チェック実施時	引数チェック非実施時
ROM サイズ	20.1K バイト (Note 3)	19.6K バイト (Note 4)
RAM サイズ	13.4K バイト	13.4K バイト

[Note]

1. 上記のサイズには、USB Basic Driver のコードサイズが含まれています。
2. コンパイラの最適化オプションには、Default オプションが指定されています。
3. 「引数チェック実施時」の ROM サイズは、r_usb_basic_config.h ファイル内の USB_CFG_PARAM_CHECKING 定義に対し USB_CFG_ENABLE を指定した時の値です。
4. 「引数チェック非実施時」の ROM サイズは、r_usb_basic_config.h ファイル内の USB_CFG_PARAM_CHECKING 定義に対し USB_CFG_DISABLE を指定した時の値です。
5. RAM サイズは、r_usb_pmesc_config.h ファイル内の USB_CFG_PMESC_TRANS_COUNT 定義に対し 8 (数値)を指定した時の値です。

3.8 引数

API 関数の引数に使用される構造体については、USB Basic Host and Peripheral Driver アプリケーションノート(ドキュメント No.R01AN0512)内の「構造体」の章を参照してください。

4. クラスドライバ概要

4.1 クラスリクエスト

本ドライバがサポートするクラスリクエストを以下に示します。

Table 4-1 クラスリクエスト

リクエスト	コード	説明
Bulk-Only Mass Storage Reset	0xFF	マスタストレージデバイスと接続インターフェースのリセット
Get Max Lun	0xFE	デバイスがサポートする論理番号を通知

4.2 ストレージコマンド

本ドライバがサポートするストレージコマンドを以下に示します。下記以外のコマンドに対しては **STALL** 応答あるいは **CSW** による **FAIL** エラーを返します。

1. Test Unit Ready
2. Request Sense
3. Inquiry
4. Read Format Capacity
5. Read10
6. Write10
7. ModeSense10

5. デバイスクラスドライバ (PDCD)

5.1 基本機能

PDCD の機能を以下に示します。

1. SFF-8070i (ATAPI) に対応
2. USB ホストからのマスタストレージデバイスクラスリクエストに対する応答

5.2 BOT プロトコル概要

BOT (Bulk-Only Transport) とは、バルクイン/バルクアウトの 2 つの Endpoint のみを使用し、コマンド、データ、ステータス (コマンド処理の結果) を管理する転送プロトコルです。

USB 上で転送されるデータのうち、コマンドとステータスについては Command Block Wrapper (CBW)、Command Status Wrapper (CSW) の形式で転送を行います。BOT プロトコル概要を以下に示します。

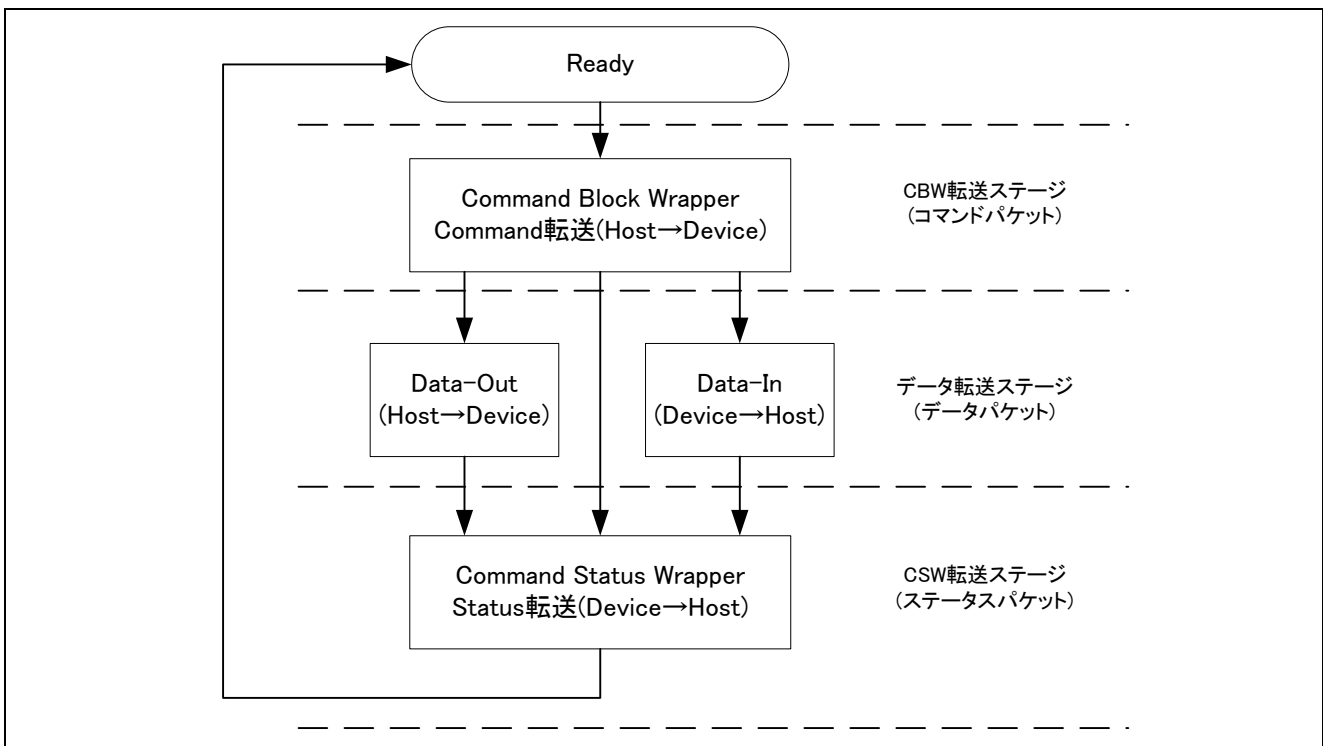


Figure 5-1 BOT プロトコル概要

6. API

アプリケーションプログラム内で使用する API については、USB Basic Host and Peripheral Driver アプリケーションノート(ドキュメント No.R01AN0512)内の「API」の章を参照してください。

7. コンフィグレーション (r_usb_pmesc_config.h)

お客様のシステムにあわせて以下の設定をお願いします。

[Note]

必ず r_usb_basic_config.h ファイルに対する設定もお願いします。r_usb_basic_config.h については、USB Basic Host and Peripheral Driver アプリケーションノート(ドキュメント No.R01AN0512)内の「コンフィグレーション」の章を参照してください。

1. 使用パイプ設定

Bulk IN, Bulk OUT 転送で使用するパイプ番号(PIPE1 から PIPE5)を指定してください。なお、USB_CFG_PMESC_BULK_IN と USB_CFG_PMESC_BULK_OUT に対し同じパイプ番号は指定しないでください。

```
#define    USB_CFG_PMESC_BULK_IN        パイプ番号 (USB_PIPE1 から USB_PIPE5)
#define    USB_CFG_PMESC_BULK_OUT      パイプ番号 (USB_PIPE1 から USB_PIPE5)
```

2. Inquiry コマンド応答データ設定

本ドライバは以下の各定義に指定したデータを Inquiry コマンドの応答データとして USB ホストに送信します。

(1). Vendor Information 設定

Inquiry コマンドの応答データである Vendor Information を指定してください。必ず 8 バイトのデータをダブルクォーテーションで括って指定してください。

```
#define    USB_CFG_PMESC_VENDOR        Vendor Information
例)
#define    USB_CFG_PMESC_VENDOR        "Renesas "
```

(2). Product Information 設定

Inquiry コマンドの応答データである Product Information を指定してください。必ず 16 バイトのデータをダブルクォーテーションで括って指定してください。

```
#define    USB_CFG_PMESC_PRODUCT      Product Information
例)
#define    USB_CFG_PMESC_PRODUCT      "Mass Storage  "
```

(3). Product Revision Level 設定

Inquiry コマンドの応答データである Product Revision Level を指定してください。必ず 4 バイトのデータをダブルクォーテーションで括って指定してください。

```
#define    USB_CFG_PMESC_REVISION     Product Information
例)
#define    USB_CFG_PMESC_REVISION     "1.00"
```

3. 転送セクタ数設定

PCD(USB Peripheral Control Driver)に要求する 1 回のデータ転送の最大セクタサイズを指定してください。本ドライバは、PCD に対し "1 セクタ(512)×USB_CFG_PMESC_TRANS_COUNT "バイトの値を転送サイズとして指定します。

この値を大きくすることにより PCD に対するデータ転送要求回数が減るため転送速度性能が向上する可能性があります。"1 セクタ(512)×USB_CFG_PMSC_TRANS_COUNT "バイト分の RAM が消費されますのでご注意ください。

```
#define    USB_CFG_PMSC_TRANS_COUNT    転送セクタ数 (1 から 255)
例)
#define    USB_CFG_PMSC_TRANS_COUNT    8
```

8. メディアドライバインタフェース

PMSC では、仕様の異なるメディアドライバへのアクセスを容易にするために共通のメディアドライバ API 関数を使用しています。

8.1 メディアドライバ API 関数

メディアドライバ API は、PMSC から呼び出され、ユーザによって実装されたメディアドライバ関数をコールします。本章では、メディアドライバ API 関数のプロトタイプと各関数の実装に必要な処理について説明します。

メディアドライバ API 関数一覧を示します。

Table 8-1 メディアドライバ API

メディアドライバ API	処理概要
R_USB_media_initialize	メディアドライバの初期化
R_USB_media_open	メディアドライバオープン
R_USB_media_close	メディアドライバクローズ
R_USB_media_read	メディアリード
R_USB_media_write	メディアライト
R_USB_media_ioctl	メディアデバイス特有のコントロール命令を処理

8.1.1 R_USB_media_initialize

メディアドライバ関数をメディアドライバに登録します。

形式

```
bool R_USB_media_initialize(media_driver_t * p_media_driver);
```

引数

p_meida_driver メディアドライバ構造体領域へのポインタ

戻り値

TRUE	正常終了
FALSE	エラー発生

解説

ユーザによって実装されたメディアドライバ関数をメディアドライバに登録します。なお、ユーザアプリケーションプログラム内の初期化処理等において必ず本APIをコールしてください。

補足

1. 上記「引数」、「戻り値」および「解説」等に記載した内容をサポートするメディアドライバ関数をユーザにおいて実装する必要があります。
2. ユーザによって実装されたメディアドライバ関数の登録方法については、「8.3 ストレージメディアドライバの登録」を参照ください。
3. 本APIはデバイスのレジスタ初期化処理やデバイス上の動作を開始するものではありません。それらの処理はR_USB_media_open()関数で行います。
4. 本APIで登録可能なメディアデバイス数は1つのみです。複数のメディアデバイス登録は非サポートです。

使用例

```
if (!R_USB_media_initialize(&g_ram_mediadriver))
{
    /* Handle the error */
}
result = R_USB_media_open();
if (USB_MEDIA_RET_OK != result)
{
    /* Process the error */
}
```

8.1.2 R_USB_media_open

メディアドライバおよびデバイスを動作可能な状態にします。

形式

```
usb_media_ret_t    R_USB_media_open(void);
```

引数

--

戻り値

USB_MEDIA_RET_OK	正常終了
USB_MEDIA_RET_PARAERR	パラメータエラー
USB_MEDIA_RET_DEV_OPEN	デバイスがすでにオープン済
USB_MEDIA_RET_NOTRDY	デバイスが応答しないまたは存在しない
USB_MEDIA_RET_OP_FAIL	その他のエラー

解説

メディアドライバで使用される周辺回路のハードウェアレジスタを初期化し、そのデバイスを動作可能な状態にします。なお、ユーザアプリケーションプログラム内の初期化处理等において必ず本APIをコールしてください。

補足

1. 本関数を呼び出す前に R_USB_media_initialize()関数を呼び出す必要があります。
2. R_USB_media_close()関数が呼び出されない限り、R_USB_media_open()の呼び出しは1回のみに限られます。なお、R_USB_media_close()関数を呼び出した後であれば、デバイス設定を初期状態に戻すために本関数を再び呼び出すことができます。
3. 上記「引数」、「戻り値」および「解説」等に記載した内容をサポートするメディアドライバ関数をユーザにおいて実装する必要があります。

使用例

```
if (!R_USB_media_initialize(&g_ram_mediadriver))
{
    /* Handle the error */
}

result = R_USB_media_open();
if (USB_MEDIA_RET_OK != result)
{
    /* Process the error */
}
```

8.1.3 R_USB_media_close

メディアドライバ用のリソースを解放し、ハードウェアを非アクティブな状態に戻します。

形式

```
usb_media_ret_t    R_USB_media_close(void);
```

引数

--

戻り値

USB_MEDIA_RET_OK	正常終了
USB_MEDIA_RET_PARAERR	パラメータエラー
USB_MEDIA_RET_OP_FAIL	その他のエラー

解説

メディアドライバ用のリソースを解放し、ハードウェアを非アクティブな状態に戻します。

補足

1. 本関数を呼び出す前に R_USB_media_initialize()関数を呼び出す必要があります。
2. 上記「引数」、「戻り値」および「解説」等に記載した内容をサポートするメディアドライバ関数をユーザにおいて実装する必要があります。

使用例

```
result = R_USB_media_close();
if (USB_MEDIA_RET_OK != result)
{
    /* Process the error */
}
```


8.1.4 R_USB_media_read

メディアデバイスからデータブロックを読み出します。

形式

```
usb_media_ret_t R_USB_media_read(uint8_t *p_buf, uint32_t lba, uint8_t count);
```

引数

p_buf	メディアデバイスから読みだされたデータを格納する領域へのポインタ
lba	読み出し開始論理ブロックアドレス
count	読み出しブロック数 (セクタ数)

戻り値

USB_MEDIA_RET_OK	正常終了
USB_MEDIA_RET_PARAERR	パラメータエラー
USB_MEDIA_RET_NOTRDY	デバイスがレディ状態ではありません
USB_MEDIA_RET_OP_FAIL	その他のエラー

解説

メディアデバイスからデータブロックのリード処理を行います。(第 2 引数で指定された LBA(Logical Block Address)から第 3 引数(count)に指定されたブロック数分のデータブロックをリードします。)

リードデータは第 1 引数(p_buf)で指定した領域に格納されます。

補足

1. 本関数を呼び出す前に R_USB_media_initialize()関数を呼び出す必要があります。
2. 上記「引数」、「戻り値」および「解説」等に記載した内容をサポートするメディアドライバ関数をユーザにおいて実装する必要があります。

使用例

```
result = R_USB_media_read(&buffer, lba, 1);
if (USB_MEDIA_RET_OK != result)
{
  /* Process the error */
}
```

8.1.5 R_USB_media_write

メディアデバイスにデータブロックを書き込みます。

形式

```
usb_media_ret_t R_USB_media_write(uint8_t *p_buf, uint32_t lba, uint8_t count);
```

引数

p_buf	メディアデバイスに書き込むデータを格納する領域へのポインタ
lba	書き込み開始論理ブロックアドレス
count	書き込みブロック数 (セクタ数)

戻り値

USB_MEDIA_RET_OK	正常終了
USB_MEDIA_RET_PARAERR	パラメータエラー
USB_MEDIA_RET_NOTRDY	デバイスがレディ状態ではありません
USB_MEDIA_RET_OP_FAIL	その他のエラー

解説

メディアデバイスへデータブロックのライト処理を行います。(第2引数で指定されたLBA(Logical Block Address)から第3引数(count)に指定されたブロック数分のデータブロックをライトします。)

ライトデータは第1引数(p_buf)が示す領域に設定してください。

補足

1. 本関数を呼び出す前に R_USB_media_initialize()関数を呼び出す必要があります。
2. 上記「引数」、「戻り値」および「解説」等に記載した内容をサポートするメディアドライバ関数をユーザにおいて実装する必要があります。

使用例

```
result = R_USB_media_write(&buffer, lba, 1);
if (MEDIA_RET_OK != result)
{
  /* Process the error */
}
```

8.1.6 R_USB_media_ioctl

メディアドライバの情報を取得します。

形式

```
usb_media_ret_t    R_USB_media_ioctl(ioctl_cmd_t command, void *p_data);
```

引数

command	メディアドライバに対するコマンドコード
p_data	メディア情報を格納する領域へのポインタ

戻り値

USB_MEDIA_RET_OK	正常終了
USB_MEDIA_RET_PARAERR	パラメータエラー
USB_MEDIA_RET_NOTRDY	デバイスがレディ状態ではありません
USB_MEDIA_RET_OP_FAIL	その他のエラー

解説

本関数はメディアドライバ固有なコマンドを引数(**command**)に指定し、メディアドライバからの戻り情報を取得する処理を行います。

PMSCではメディアドライバに対するコマンドコードとして、以下のコマンドを使用しています。

MEDIA_IOCTL_GET_NUM_BLOCKS	メディア領域のブロック数
MEDIA_IOCTL_GET_BLOCK_SIZE	1ブロックサイズ

補足

1. 本関数を呼び出す前に **R_USB_media_initialize()**関数を呼び出す必要があります。
2. ユーザは引数(**command**)に指定するコマンドコードを新たに定義することができます。
3. 上記「引数」、「戻り値」および「解説」等に記載した内容をサポートするメディアドライバ関数をユーザにおいて実装する必要があります。

使用例

```
uint32_t num_blocks;  
uint32_t block_size;  
uint64_t capacity;  
  
result = R_USB_media_ioctl(MEDIA_IOCTL_GET_NUM_BLOCKS, (void *)&num_blocks);  
result = R_USB_media_ioctl(MEDIA_IOCTL_GET_BLOCK_SIZE, (void *)&block_size);  
  
capacity = (uint64_t)block_size * (uint64_t)num_blocks;
```

8.2 構造体/列挙型定義

メディアドライバ API 関数で使用する構造体/列挙型について説明します。
これらは `r_usb_media_driver_if.h` ファイルで定義されています。

8.2.1 usb_media_driver_t (構造体)

`usb_media_driver_t` は、メディアデバイスの論理ユニット番号とメディアドライバで実装する必要のある関数へのポインタを保持する構造体です。

以下に、`usb_media_driver_t` 構造体を示します。

```
typedef struct media_driver_s
{
    usb_media_open_t    pf_media_open;    /* オープン関数のポインタ */
    usb_media_close_t   pf_media_close;   /* クローズ関数のポインタ */
    usb_media_read_t    pf_media_read;    /* リード関数のポインタ */
    usb_media_write_t   pf_media_write;   /* ライト関数のポインタ */
    usb_media_ioctl_t   pf_media_ctrl;    /* コントロール関数のポインタ */
} usb_media_driver_t
```

8.2.2 usb_media_ret_t (列挙型)

`usb_media_ret_t` には、メディアドライバ API が返す戻り値が定義されています。

```
typedef enum
{
    USB_MEDIA_RET_OK = 0,           /* 正常終了 */
    USB_MEDIA_RET_NOTRDY,          /* デバイスがレディ状態でない */
    USB_MEDIA_RET_PARERR,          /* パラメータエラー */
    USB_MEDIA_RET_OP_FAIL,         /* その他のエラー */
    USB_MEDIA_RET_DEV_OPEN,        /* デバイスは既にオープンしている */
    USB_MEDIA_RET_DEV_NO_INIT,     /* デバイスが初期化されていない */
} usb_media_ret_t
```

8.2.3 ioctl_cmd_t (列挙型)

`ioctl_cmd_t` には、`R_USB_media_ioctl` 関数に指定するコマンドコードが定義されています。

```
typedef enum
{
    USB_MEDIA_IOCTL_GET_NUM_BLOCKS, /* 論理ブロック数取得 */
    USB_MEDIA_IOCTL_GET_BLOCK_SIZE, /* 論理ブロックサイズ取得 */
} ioctl_cmd_t
```

[Note]

メディアドライバ実装にあたりユーザ独自のコマンドコードを追加する場合、上記 `ioctl_cmd_t` に追加してください。

8.3 ストレージメディアドライバの登録

PMSC のストレージメディアをフラッシュメモリなどの他のストレージメディアへ変更する場合、ユーザは、そのストレージメディアに対する読み出しまたは書き込みを行うためのメディアドライバ関数を実装し、メディアドライバ API に登録する必要があります。

以下に、シリアル SPI フラッシュのメディアドライバ関数登録手順を示します。

1. 登録するメディアドライバ関数について

ユーザがシリアル SPI フラッシュ用メディアドライバ関数として以下の関数を実装したものとします。

1. usb_media_ret_t spi_flash_open (void)
2. usb_media_ret_t spi_flash_close (void)
3. usb_media_ret_t spi_flash_read(uint8_t *p_buf,uint32_t lba, uint8_t count)
4. usb_media_ret_t spi_flash_write(uint8_t *p_buf,uint32_t lba, uint8_t count)
5. usb_media_ret_t spi_flash_ioctl(ioctl_cmd_t ioctl_cmd,void * ioctl_data)

2. メディアドライバ関数のメディア API への登録

- (1). シリアル SPI フラッシュ用の media_driver_s 構造体を定義してください。

この構造体の各メンバには、該当するメディアドライバ関数へのポインタを設定してください。

```
struct media_driver_t g_spi_flash_mediadriver =
{
    &spi_flash_open,
    &spi_flash_close,
    &spi_flash_read,
    &spi_flash_write,
    &spi_flash_ioctl
};
```

- (2). アプリケーションプログラムで、上記の media_driver_s 構造体を R_USB_media_initialize 関数 (API)の引数に指定し、初期化処理を行ってください。

```
== アプリケーションプログラム ==
R_USB_media_initialize(& g_spi_flash_mediadriver);
```

上記手順をおこなうことにより、メディアドライバが呼び出すメディアドライバ関数としてシリアル SPI フラッシュ関数が登録されます。

8.4 ストレージメディアドライバの実装

ご使用になるストレージメディアに対応するメディアドライバ関数をお客様において実装いただく必要があります。実装したメディアドライバ関数は、PMSC から「8.1 メディアドライバAPI関数」に記載された API を経由してコールされます。

[Note]

メディアドライバ関数の実装に必要な処理については、「8.1 メディアドライバAPI関数」に記載された各 API の処理内容を参考してください。

作成するメディアドライバ関数は、Open,Close,Read,Write,Control の 5 関数で、関数プロトタイプを以下に示します。

1. `usb_media_ret_t (*media_open_t)(uint8_t);` /* Open 関数型 */
2. `usb_media_ret_t (*media_close_t)(uint8_t);` /* Close 関数型 */
3. `usb_media_ret_t (*media_read_t)(uint8_t, uint8_t*, uint32_t, uint8_t);` /* Read 関数型 */
4. `usb_media_ret_t (*media_write_t)(uint8_t, uint8_t*, uint32_t, uint8_t);` /* Write 関数型 */
5. `usb_media_ret_t (*media_ioctl_t)(uint8_t, ioctl_cmd_t, void *);` /* Control 型関数 */

9. サンプルアプリケーション

9.1 アプリケーション仕様

PMSC のサンプルアプリケーション (以降、APL) は、RSK 上で動作します。RSK をホスト PC と接続するとリムーバブルディスクとして認識され、ファイルの読み書きなどデータ転送を行うことが可能です。Figure 9-1に PMSC 動作環境例、Figure 9-2に PMSC 動作例を示します。

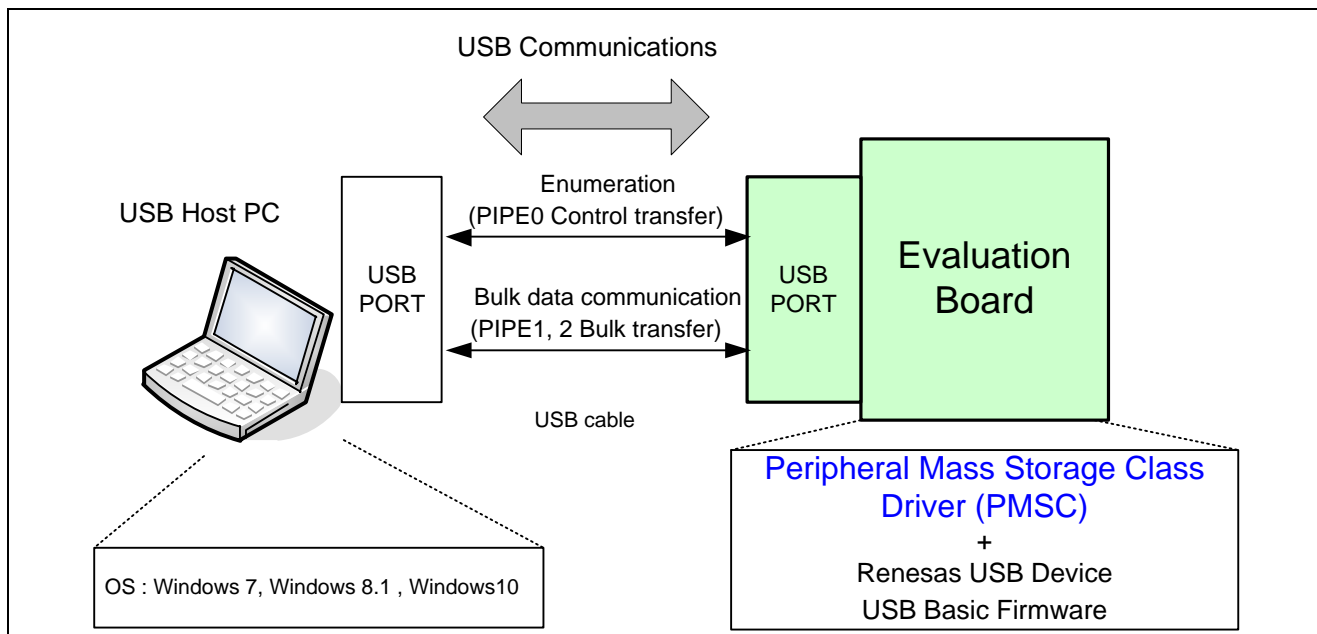


Figure 9-1 PMSC 動作環境例

9.1.1 リムーバブルディスクのメディア領域

APLでは、リムーバブルディスクのメディア領域は、RSK上のSDRAM領域またはMCUの内蔵RAMを使用しています。



Figure 9-2 PMSC 動作例

9.2 アプリケーション処理概要

APL は、初期設定およびメインループの 2 つの部分から成ります。

- 初期設定 : MCU の端子設定、USB コントローラの初期化および USB ドライバの初期化を行います。
- メインループ : ループ内で R_USB_GetEvent 関数をコールします。ループ処理中に USB ホストからのサスペンド要求やデタッチがあった場合、APL は MSC デバイス(RSK) を消費電力低減モードに移行します。消費電力低減モードについては、「9.3 MCU消費電力低減処理」を参照してください。

PMSC は、USB ホスト(PC)からの要求をマスタストレージクラスドライバ (MSCD) 及びマスタストレージデバイスドライバ (MSDD) が処理します。従って、PMSC の APL では、ホストから転送されたデータに対してなにも処理を行いません。初期化処理以外は、ループ内で R_USB_GetEvent 関数をコールするのみです。この PMSC のストレージ領域へのファイル書き込みやファイル読み出し等の処理は、APL は何も行わず、PMSC USB ドライバがこれらの処理を行います。

[Note]

1. PMSC がサポートしているストレージコマンドについては、「4.2 ストレージコマンド」を参照してください。
2. R_USB_GetEvent 関数はアプリケーションプログラムのループ処理内で必ずコールしてください。

以下に、APL の処理概要を示します。

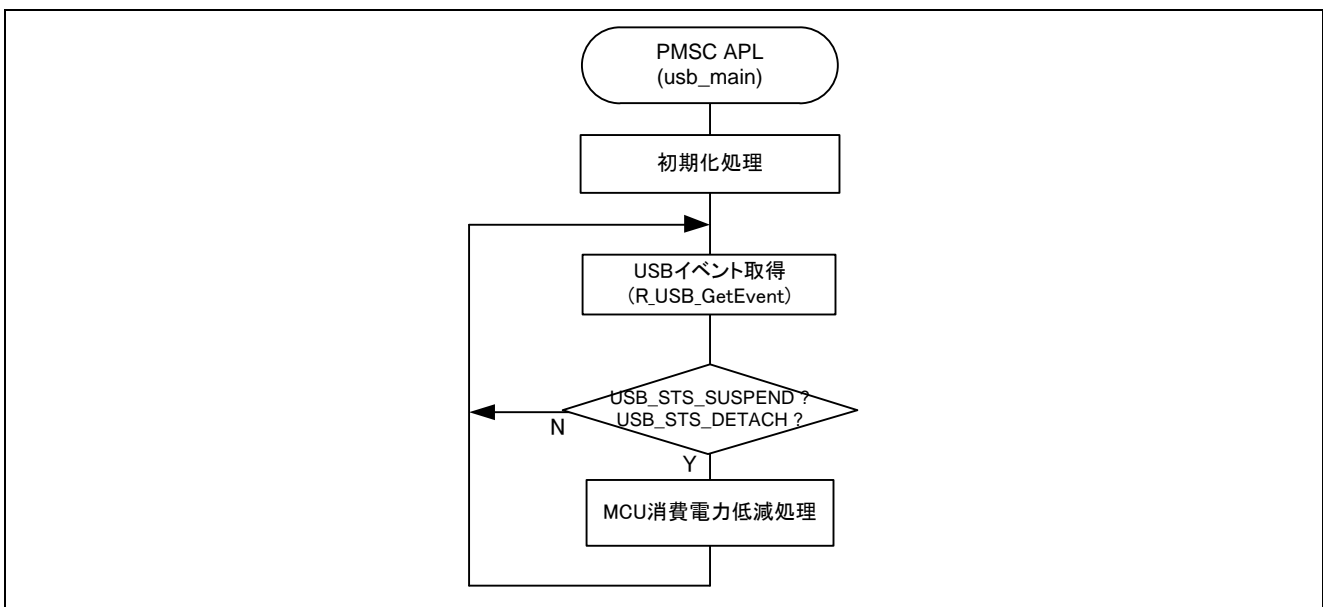


Figure 9-3 APL 処理概要

9.3 MCU 消費電力低減処理

MCU 消費電力低減処理は、Table 9-1の条件が成立すると消費電力低減モードに移行する処理を行います。なお、この処理を有効にするには、r_usb_pmesc_apl_config.h” ファイル内に“USE_LPW” 定義を定義してください。

Table 9-1 消費電力低減機能状態遷移条件

遷移条件		遷移状態
VBUS	USB ステート	
OFF	—	ソフトウェアスタンバイモード
ON	Suspend Configured	スリープモード
ON	Suspend Configured 以外	通常モード (プログラム実行状態)

- MSC デバイス(RSK)が USB Host からデタッチ (VBUS OFF) されると、APL は MCU をソフトウェアスタンバイモードに遷移するための処理を行います。ソフトウェアスタンバイモードからの復帰は、MSC デバイス(RSK)を USB Host にアタッチすることにより行われます。
- MSC デバイス(RSK)を USB Host に接続した状態で、USB Host から送信されるサスペンド信号を受信すると APL は、MCU をスリープモードに遷移するための処理を行います。なお、スリープモードからの復帰は、USB Host から送信されるレジューム信号の受信により行われます。

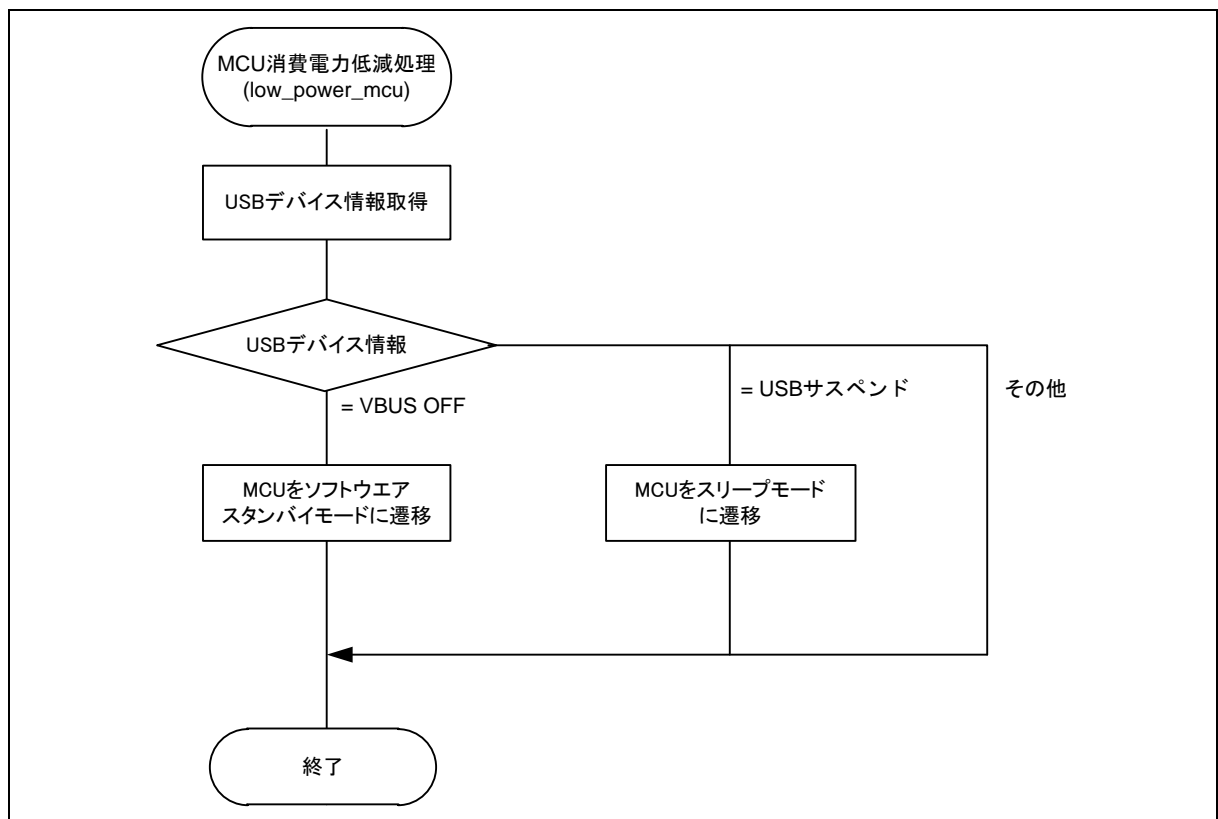


Figure 9-4 MCU 消費電力低減処理概略フロー

9.4 アプリケーションプログラム用コンフィグレーションファイル (r_usb_pmsc_apl_config.h)

以下の各定義に対する設定を行ってください。

1. USE_USBIP 定義

使用する USB モジュールのモジュール番号を指定してください。USE_USBIP0/USE_USBIP1 のいずれかを指定してください。

```
#define USE_USBIP USE_USBIP0 // USB0 モジュールを使用する場合
#define USE_USBIP USE_USBIP1 // USB1 モジュールを使用する場合
```

[Note]

RX63T または RX630 をご使用の場合、USE_USBIP0 を指定してください。

2. USB_SUPPORT_SPEED 定義

MSC デバイス(RSK)がサポートする USB 動作スピード(Full-speed)を指定してください。

```
#define USB_SUPPORT_SPEED USB_FS // Full-Speed 設定
```

1. 消費電力低減機能定義

消費電力低減機能の使用/非使用を指定してください。消費電力低減機能を使用する場合は、USB_SUPPORT_LPW 定義に対し USB_APL_ENABLE を指定し、消費電力低減機能を使用しない場合は、USB_SUPPORT_LPW 定義に対し USB_APL_DISABLE を指定してください。

```
#define USB_SUPPORT_LPW USB_APL_DISABLE //消費電力低減機能を非使用
#define USB_SUPPORT_LPW USB_APL_ENABLE //消費電力低減機能を使用
```

2. 注意事項

上記はアプリケーションプログラム用のコンフィグレーション設定です。上記の設定の他に USB ドライバのコンフィグレーション設定が必要です。USB ドライバのコンフィグレーション設定については、「USB Basic Host and Peripheral Driver アプリケーションノート」(Document No. R01AN0512JJ)を参照してください。

9.5 ディスクリプタ

PMSC のディスクリプタ情報は r_usb_pmsc_descriptor.c に記述しています。なお、Vendor ID は、必ずお客様用の Vendor ID をご使用いただきますようお願いいたします。

10. セットアップ

10.1 ハードウェア

10.1.1 動作環境例

PMSC の動作環境例をFigure 10-1に示します。評価ボードのセットアップ、エミュレータなどの使用方法については各取扱説明書を参照ください。

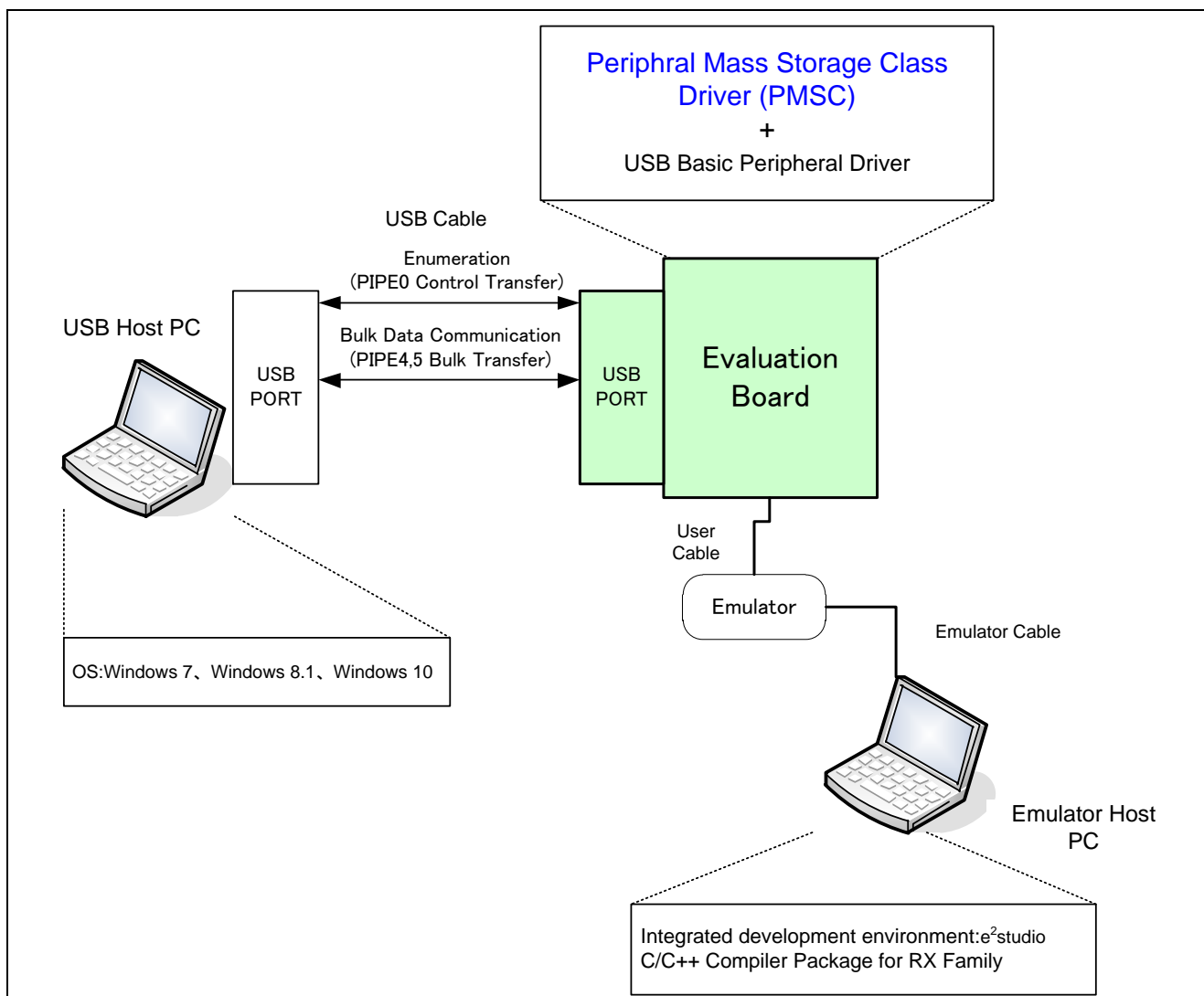


Figure 10-1 動作環境例

< PC 環境設定 >

Windows 7 以降の USB ホスト PC をご使用の場合、USB ホスト PC に対する設定等の必要はありません。

10.1.2 RSK 設定

RSK を USB Host モードに設定する必要があります。設定内容は以下を参照してください。

Table 10-1 RSK 設定

RSK	ジャンパ設定
RSK+RX63N	J3: Shorted Pin1-2 J4: Shorted Pin1-2 J18:Shorted Pin2-3

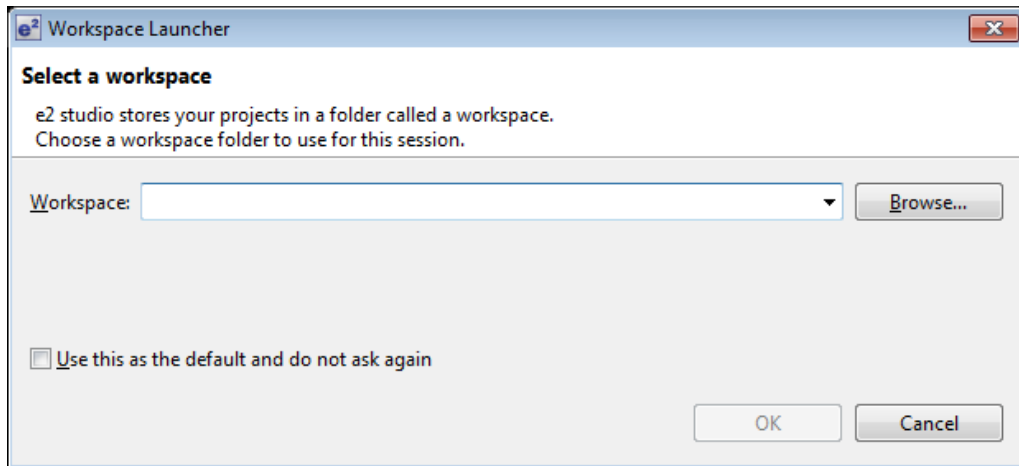
[Note]

RSK 設定の詳細については、RSK のユーザーズマニュアルを参照してください。

10.2 ソフトウェア

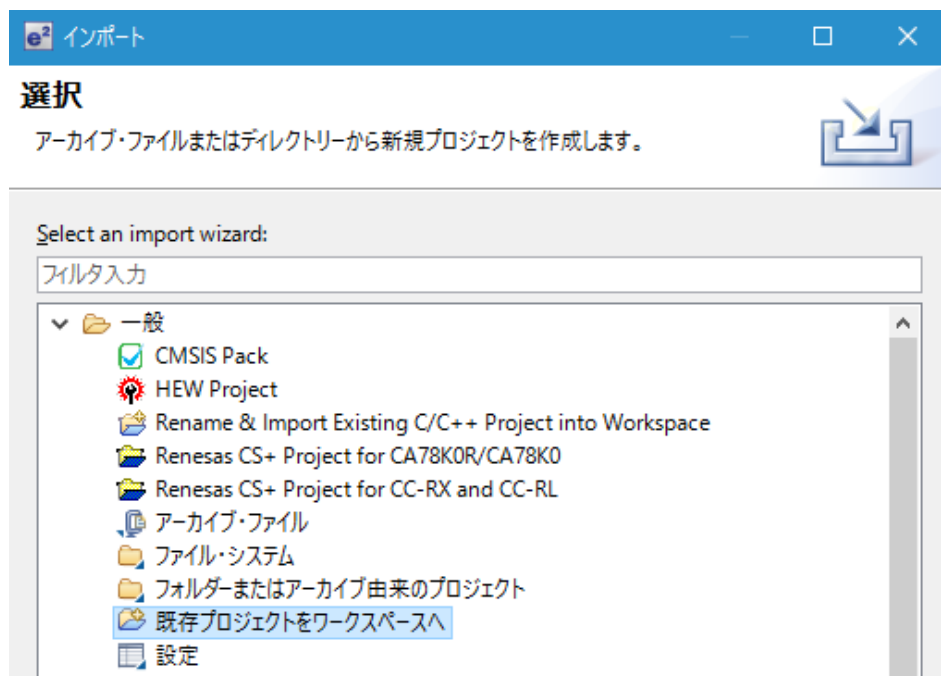
(1). e² studio を起動

- a) e² studio を起動してください。
- b) はじめて e² studio を起動する場合、Eclipse Launcher ダイアログが表示されますので、プロジェクトを格納するためのフォルダを指定してください。

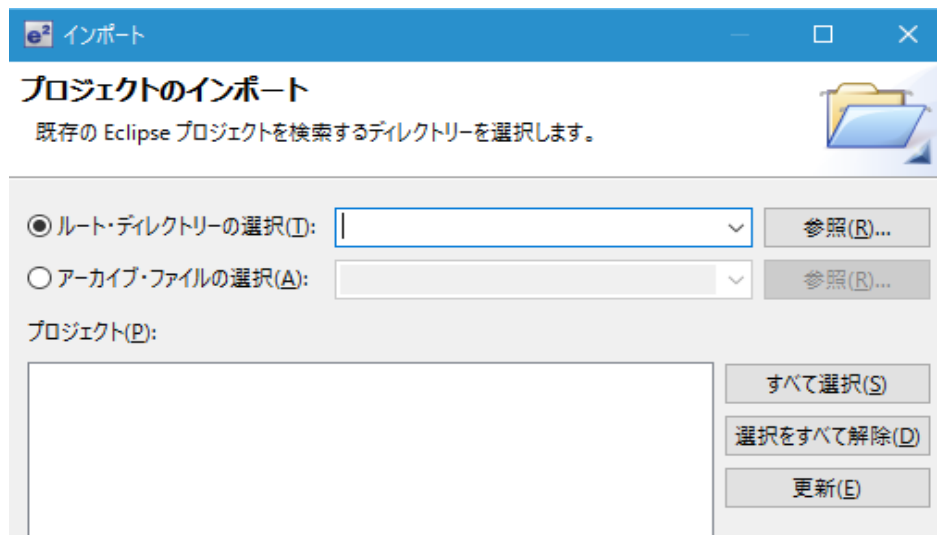


(2). ワークスペースへのプロジェクトの登録

- a) [ファイル] --> [インポート]を選択してください。
- b) [一般] => [既存プロジェクトをワークスペースへ]を選択してください。



- c) プロジェクトファイル".cproject"が格納されたフォルダを"ルート・ディレクトリの選択"に入力してください。



- d) “終了”をクリック

プロジェクトのワークスペースへのインポートが完了しました。同様の方法で他のプロジェクトを同一のワークスペースへインポートすることができます。

- (3). “Build”ボタンをクリックし、実行プログラムを生成してください。
- (4). デバッガへの接続を行い、実行プログラムをダウンロードしてください。“Run”ボタンをクリックすると、プログラムが実行されます。

11. アプリケーションの作成方法

USB Basic Host and Peripheral Driver アプリケーションノート(ドキュメント No.R01AN0512)内の「アプリケーションプログラムの作成方法」の章を参照してください。

[Note]

ユーザアプリケーションプログラム内の初期化処理等において必ず R_USB_media_initialize 関数(API) および R_USB_media_open 関数(API)をコールしてください。

12. e² studio 用プロジェクトを CS+で使用する場合

PMSC のプロジェクトは、統合開発環境 e² studio で作成されています。PMSC を CS+で動作させる場合は、下記の手順にて読み込んでください。

[Note]

「プロジェクト変換設定」ウィンドウ内の「変換直前のプロジェクト構成ファイルをまとめてバックアップする」のチェックを外してください。

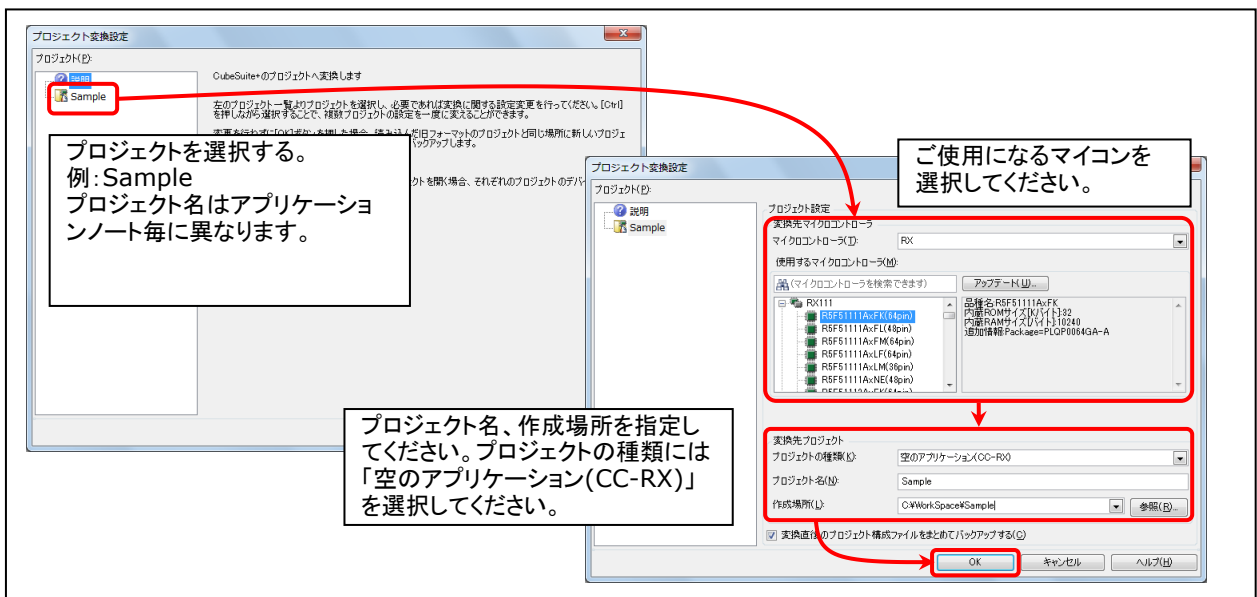
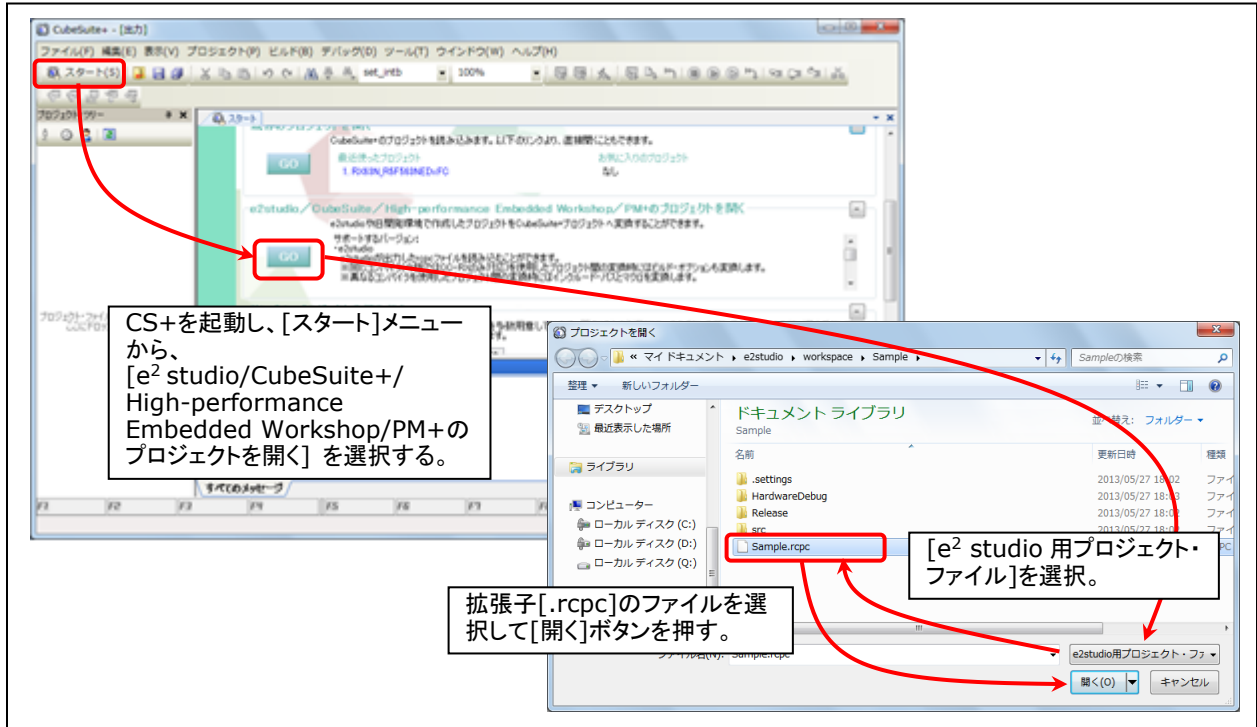


Figure 12-1 e² studio 用プロジェクトの CS+読み込み方法

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>

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
1.00	2010/11/24	—	初版発行
1.10	2011/9/1	—	動作確認デバイスに R8A66597, RX630 を追加
		2	関連ドキュメントと用語一覧の順番を変更 (1.3,1.4)
		3	モジュール構成で PMSCD の説明追記 (2.1)
		5	システム資源に nonOS 動作時のシステムリソース定義、USB 割込みハンドラ定義を追加
		6	BOT プロトコル概要 説明追記&図(図 3.1 BOT プロトコル概要)の追加
		7	送受信データのないストレージコマンドのシーケンス 説明追記(3.2.1)、図 3.2 シーケンス図への説明・注釈追記
		8	送信 (IN)データがあるストレージコマンドのアクセスシーケンス 説明追記(3.2.2)、図 3.3 シーケンス図への説明・注釈追記
		9	受信 (OUT)データがあるストレージコマンドのアクセスシーケンス 説明追記(3.2.3)、図 3.4 シーケンス図への説明・注釈追記
		27	送受信のないストレージコマンドの章を移動 (4.8.3→4.8.1)、図 4.1 シーケンス図への説明・注釈追記
		28	送信(IN)データがあるストレージコマンドの章を移動 (4.8.1→4.8.2)、図 4.2 シーケンス図への説明・注釈追記
		29	受信(OUT)があるストレージコマンドの章を移動 (4.8.2→4.8.3)、図 4.3 シーケンス図への説明・注釈追記
		33	PMSDD 構造体の説明修正(5.4)
		38	PMSDD タスク説明へ文章追記(5.8)
2.00	2012/3/1	—	動作確認デバイスに RX63N を追加
		—	章立て変更、ソースコード変更にともない、関数名等更新
		2	関連ドキュメントと用語一覧の順番を変更 (1.3,1.4)
		5	表 2.1 に関連ファイル、参照ドキュメントを追記
		6	2.2 ファイル構成一覧更新
		10	図 3.3 APL 処理概略フロー追加
		11	表 3.1 APL 関数一覧追加
		15	表 4.3 PMSDD 関数一覧更新
		23-29	API 関数、関数説明更新
31-36	7 章、8 章追加		
2.01	2013/2/1	—	“1.5 本書の読み方”の追加およびケアレミス修正
2.10	2013/4/1	—	V.2.10 用 First Release 1. 動作確認デバイスに RX63T, R8A66593 を追加。この追加に伴いこれらのデバイスに関する内容を追加 2. メディアドライバの章を変更 3. “RAM メディアドライバ”の章を追加
2.20	2015/9/30	—	アプリケーションプログラムを変更 フォルダ構成を変更 対象デバイスに RX63N と RX631 を追加。 対象デバイスから RX63N, RX631, R8A66597 および R8A66593 を削除
2.30		—	1. DMA 転送をサポート 2. USB Host and Peripheral Interface Driver アプリケーションノート(ドキュメント No.R01AN3293JJ)に対応

2.31	Sep 30, 2017	—	DMA/DTC 転送を非サポートに変更 2. USB Host and Peripheral Interface Driver アプリケーションノート(ドキュメント No.R01AN3293JJ)の記載内容を本ドキュメントに移行し、USB Host and Peripheral Interface Driver アプリケーションノートを削除した。
2.32	Mar 31, 2018	—	USB Basic Driver のリビジョンを Up しました。
2.33	Jul 31, 2019	—	対象デバイスに RX63N/RX631 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っていると、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、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、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。
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、当社は一切その責任を負いません。

6. 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。
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、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。
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。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. 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。
10. 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。
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。
12. 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.4.0-1 2017.11)



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 〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

■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：<https://www.renesas.com/contact/>