RZ/A1H グループ

USB Peripheral Mass Storage Class Driver (PMSC)

R01AN3507JJ0120
Rev.1.20
2017.08.31

要旨

本アプリケーションノートでは、USB Peripheral Mass Storage Class Driver（PMSC）について説明します。本ドライバは USB Host and Peripheral Driver（USB-BASIC-FW）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。

動作確認デバイス

RZ/A1H グループ

関連ドキュメント

1. Universal Serial Bus Revision 2.0 specification
2. RZ/A1H グループ、RZ/A1M 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.R01UH0403JJ）
3. RZ/A1H グループ USB Host and Peripheral Interface Driver（ドキュメント No.R01AN3291JJ）
4. RZ/A1H グループ ARM® Development Studio 5（DS-5™）のセミホスティング機能を使用した
   NOR 型フラッシュメモリへのダウンロード例（ドキュメント No.R01AN1957JJ）
5. RZ/A1H グループレジスタ定義ヘッダ・ファイル iodefine.h（ドキュメント No.R01AN1860JJ）
6. RZ/A1H グループ初期設定例（ドキュメント No.R01AN1864JJ）

— ルネサス エレクトロニクスホームページ
【https://www.renesas.com/】
— USB デバイスページ
【http://japan.renesas.com/prod/usb/】

目次

1. 概要 3
1.1 動作確認済環境 3
1.2 制限事項 3
1.3 注意事項 4
1.4 用語一覧 4

2. 動作環境 4

3. ファイル構成とディレクトリ 5

4. コンパイル時の設定 6

5. クラスドライバ概要 7
5.1 クラスリクエスト 7
5.2 ストレージコマンド 7

6. デバイスクラスドライバ（PMSCD） 8
6.1 基本機能 8
6.2 BOT プロトコル概要 8

7. メディアドライバインタフェース 9
7.1 メディアドライバ API 関数 9
7.2 構造体/列挙型定義 16
7.2.1 usb_media_driver_t（構造体） 16
7.2.2 usb_media_ret_t（列挙型） 16
7.2.3 ioctl_cmd_t（列挙型） 16
7.3 ストレージメディアドライバの登録 17
7.4 ストレージメディアドライバの実装 17

8. サンプルアプリケーション（APL） 18
8.1 アプリケーション仕様 18
8.2 アプリケーション用コンフィグレーションファイル（r_usb_pmsc_apl_config.h） 18
8.3 アプリケーション処理概要 19
8.4 ディスクリプタ 19
8.4.1 g_msc_device 20
8.4.2 g_msc_qualifier_descriptor 21
8.4.3 g_msc_fs_configuration 21
8.4.4 g_msc_hs_configuration 22
8.4.5 g_msc_stringX（X = 0～6） 24

1. 概要

PMSC は USB-BASIC-FW 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ルドライバ、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、PMSC がサポートしている機能を示します。

1. BOT プロトコルによるストレージコマンド制御
2. 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

1.1 動作確認済環境

本ドライバの動作確認済環境を Table 1.1 に示します。

Table 1.1　動作確認済環境

項目	内容
使用マイコン	RZ/A1H
動作周波数	CPU クロック（Iφ）: 400MHz 画像処理クロック（Gφ）: 266.37MHz 内部バスクロック（Bφ）: 133.33MHz 周辺クロック 1（P1φ）: 66.67MHz 周辺クロック 0（P0φ）: 33.33MHz
動作電圧	電源電圧（I/O）：3.3V 電源電圧（内部）：1.8V
統合開発環境	ARM®統合開発環境 ARM Development Studio（DS-5™）　Version 5.26
	IAR 統合開発環境 IAR Embedded Workbench for ARM Version 7.70
コンパイラ	ARM C/C++ Compiler/Linker/Assembler Ver.5.03 [Build 102]
	KPIT GNUARM-RZ v14.01
	IAR C/C++ Compiler for ARM 7.40
動作モード	ブートモード 0（CS0 空間 16 ビットブート）
使用ボード	GENMAI ボード R7S72100 CPU ボード RTK772100BC00000BR
使用デバイス （ボード上で使用する機能）	USB1 コネクタ、USB2 コネクタ

1.2 制限事項

本ドライバには以下の制限事項があります。

1. 型の異なるメンバで構造体を構成しています。
   コンパイラによっては構造体のメンバにアドレスアライメントずれを発生することがあります。
2. 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。
3. 本ドライバがサポートするセクタサイズは 512 のみです。

1.3　注意事項

1. 本ドライバは、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. 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。

1.4　用語一覧

本資料で使用される用語と略語は以下のとおりです。

APL	:	Application program
API	:	Application programing Interface
BOT	:	Mass storage class Bulk Only Transport
DDI	:	Device Driver Interface
PCD	:	Peripheral Control Driver
PCI	:	PCD Interface
PMSCD	:	Peripheral Mass Storage USB Class Driver (PMSCF + PCI + DDI)
PMSCF	:	Peripheral Mass Storage Class Function
PMSDD	:	Peripheral Mass Storage Device Driver (ATAPI driver)
USB	:	Universal Serial Bus
USB-BASIC-FW	:	USB Basic Peripheral Driver
USB Host	:	PMC USB Host
データ転送	:	Control 転送、Bulk 転送、Interrupt 転送の総称

2.　動作環境

PMSC の動作環境例を Figure 2.1 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してください。

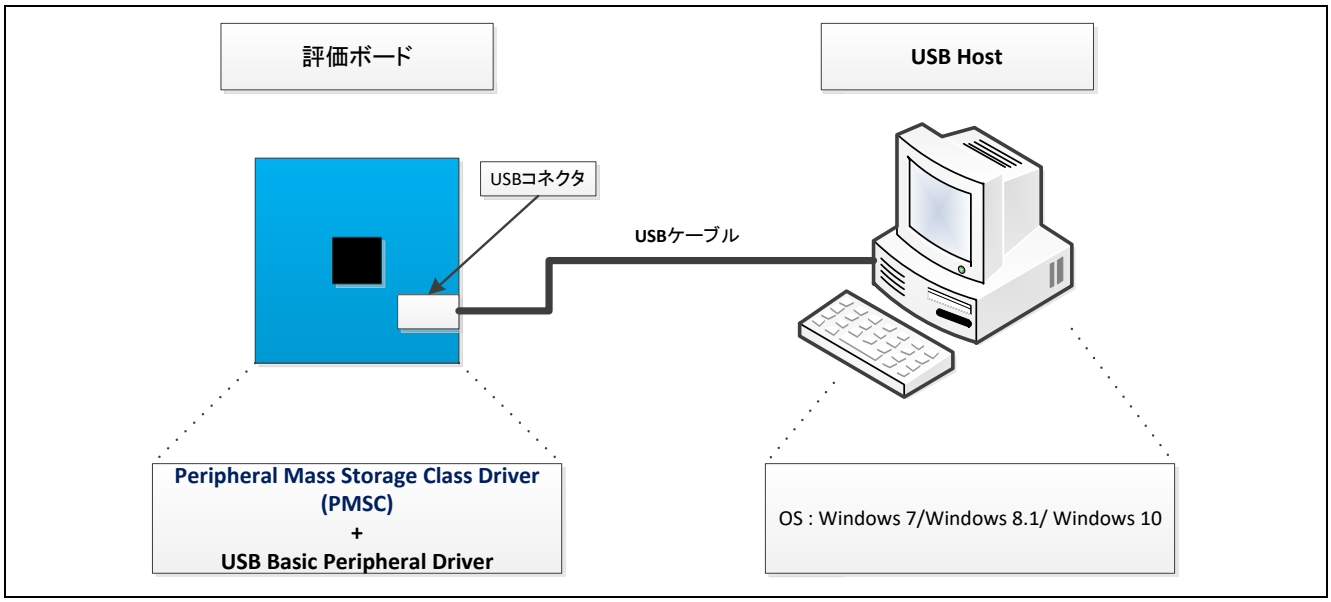

Figure 2.1　動作環境例

3.　ファイル構成とディレクトリ

Figure 3.1 に PMSC のソフトウェア構成、Table 3.1 に各階層の機能概要を示します。

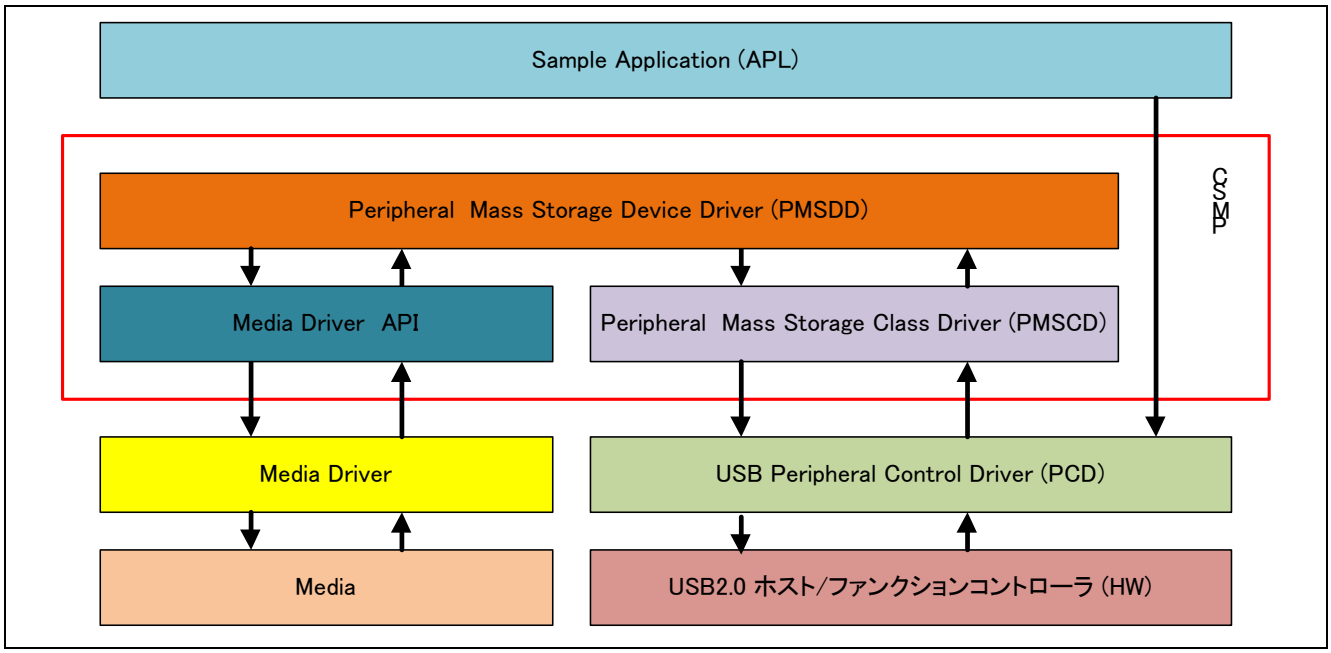

Figure 3.1　ソフトウェア構成図

Table 3.1　各階層の機能概要

モジュール名	機能概要
APL	USB 初期設定
PMSDD	PMSCD からのストレージコマンド処理を行う Media Driver を介して Media へアクセスを行う
PMSCD	BOT プロトコルデータの制御、クラスリクエスト対応を行う CBW の解析、データ送受信を行う PMSDD/PCD と連携し、CSW を生成する
Media Driver API	PMSDD – Media Driver 間のインターフェイス関数
Media Driver	メディアコントロール
PCD	割り込み処理 USB Host からの要求解析 デバイスステート管理 ハードウェア制御
H/W	ハードウェア

4. コンパイル時の設定

PMSC を動作させる場合、USB-BASIC-FW をペリフェラルとして設定する必要があります。

USB-BASIC-FW の設定は、「RZ/A1H グループ USB Host and Peripheral Interface Driver（ドキュメント No.R01AN3291JJ）」を参照してください。

併せて、PMSC のコンフィギュレーションオプション設定を r_usb_pmsc_config.h で行います。オプション名および設定値に関する説明を下表に示します。

Configuration options in r_usb_pmsc_config.h		
使用パイプ設定		
	USB_CFG_PMSC_BULK_IN	データ転送（Bulk In）で使用するパイプ番号を指定してください。 USB_PIPE1~ USB_PIPE5 のいずれかを指定してください。（注 1）
	USB_CFG_PMSC_BULK_OUT	データ転送（Bulk Out）で使用するパイプ番号を指定してください。 USB_PIPE1~ USB_PIPE5 のいずれかを指定してください。（注 1）
Inquiry コマンド応答データ設定		
	USB_CFG_PMSC_VENDOR	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。 （例）"Renesas "
	USB_CFG_PMSC_PRODUCT	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。 （例）"Mass Storage　　"
	USB_CFG_PMSC_REVISION	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。 （例）"1.00"
転送セクタ数設定		
	USB_CFG_PMSC_TRANS_COUNT	PCD 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_PMSC_TRANS_COUNT "バイトの値を転送サイズとして指定します。（1～127） 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が、"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。

注1. USB_CFG_PMSC_BULK_IN と USB_CFG_PMSC_BULK_OUT に対し同じパイプ番号は指定しないでください。

5. クラスドライバ概要

5.1 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 Table 5.1 に示します。

Table 5.1 クラスリクエスト

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

5.2 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 Table 5.2 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 を返します。

Table 5.2 ストレージコマンド

コマンド	コード	説明
TEST_UNIT_READY	0x00	ペリフェラル機器の状態確認
REQUEST_SENSE	0x03	直前のストレージコマンド実行結果のエラー情報取得
INQUIRY	0x12	論理ユニットのパラメータ情報取得
READ_FORMAT_CAPACITY	0x23	フォーマット可能な容量取得
READ_CAPACITY	0x25	論理ユニットの容量情報取得
READ10	0x28	データ読み出し
WRITE10	0x2A	データ書き込み
MODE_SENSE10	0x5A	論理ユニットのパラメータ取得

6. デバイスクラスドライバ（PMSCD）

6.1 基本機能

PMSCD 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
3. 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

6.2 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 Figure 6.1 に示します。

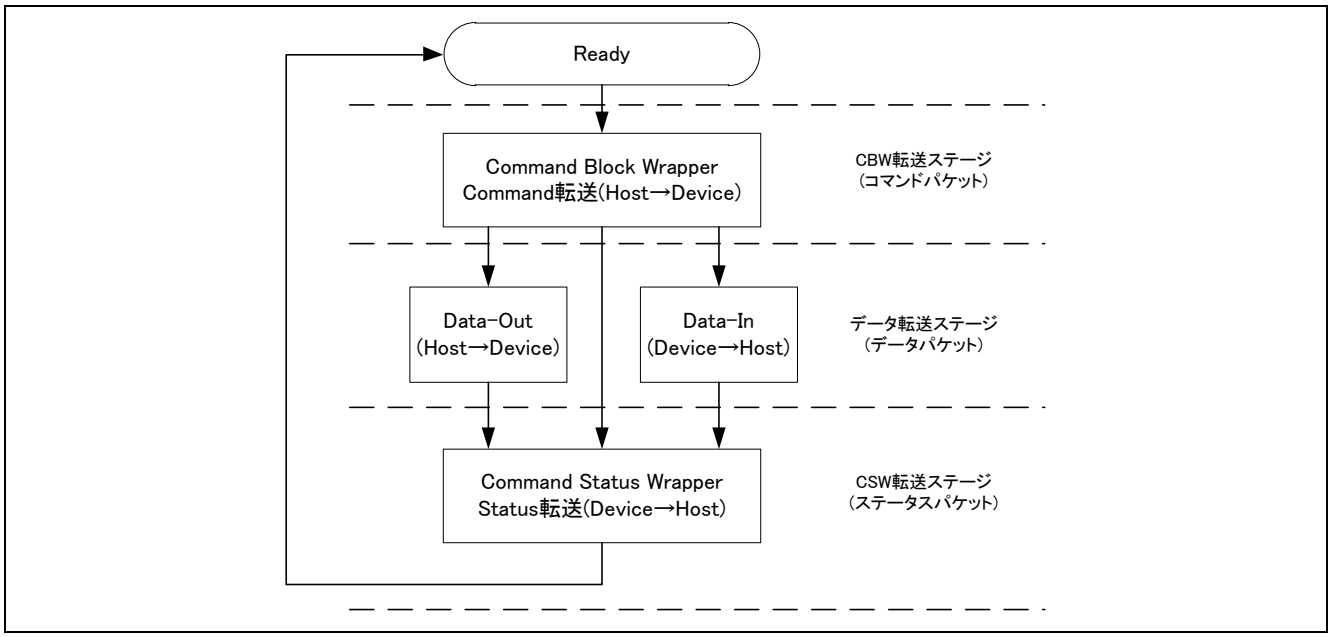

Figure 6.1 BOT プロトコル概要

7. 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。

7.1 メディアドライバ API 関数

メディアドライバ API 関数は、PMSC から呼び出されます。メディアドライバ API 関数は、ユーザーによって実装されたメディアドライバ関数を呼び出します。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。

Table 7.1 にメディアドライバ API 関数一覧を示します。

Table 7.1 メディアドライバ API 関数

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

7.1.1 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

ユーザー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、ユーザー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ーにおいて実装する必要があります。
2. ユーザー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、「7.3 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 */
}
```

7.1.2 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み
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、ユーザー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ーにおいて実装する必要があります。
2. R_USB_media_open 関数をコールした時、上記 1 のメディアドライバ関数がコールされるための登録処理が必要です。メディアドライバ関数の登録方法については、「7.3 ストレージメディアドライバの登録」を参照してください。
3. 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。
4. R_USB_media_close 関数が呼び出されない限り、R_USB_media_open 関数の呼び出しは 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。

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 */
}
```

7.1.3 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. 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ーにおいて実装する必要があります。
2. R_USB_media_close 関数をコールした時、上記 1 のメディアドライバ関数がコールされるための登録処理が必要です。メディアドライバ関数の登録方法については、「7.3 ストレージメディアドライバの登録」を参照してください。

使用例

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

7.1.4 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. 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ーにおいて実装する必要があります。
2. R_USB_media_read 関数をコールした時、上記 1 のメディアドライバ関数がコールされるための登録処理が必要です。メディアドライバ関数の登録方法については、「7.3 ストレージメディアドライバの登録」を参照してください。

使用例

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

7.1.5 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. 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ーにおいて実装する必要があります。
2. R_USB_media_write 関数をコールした時、上記 1 のメディアドライバ関数がコールされるための登録処理が必要です。メディアドライバ関数の登録方法については、「7.3 ストレージメディアドライバの登録」を参照してください。

使用例

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

7.1.6 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。

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ーにおいて実装する必要があります。
2. R_USB_media_ioctl 関数をコールした時、上記 2 のメディアドライバ関数がコールされるための登録処理が必要です。メディアドライバ関数の登録方法については、「7.3 ストレージメディアドライバの登録」を参照してください。
3. 引数(command)にしているコマンドコードはユーザーによって定義してください。

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 = (uin64_t)block_size * (uint64_t)num_blocks;
```

7.2 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。

7.2.1 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
```

7.2.2 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。

以下に、usb_media_ret_t 列挙型を示します。

```
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
```

7.2.3 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。

以下に、ioctl_cmd_t 列挙型を示します。

```
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]

1. メディアドライバ実装にあたりユーザー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。

7.3 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、ユーザーは、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を例として、その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

ユーザーがシリアル 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。

7.4 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「7.1 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]

1. メディアドライバ関数の実装に必要な処理については、「7.1 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  */
```

8. サンプルアプリケーション（APL）

8.1 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、GENMAI ボード上で動作します。GENMAI ボードを USB ホスト(PC)に接続すると、リムーバブルディスクとして認識されます。リムーバブルディスクとして認識後、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。

動作環境例については、「2 動作環境」を参照してください。

[Note]

1. Windows 7 / Windows 8.1 / Windows 10 をサポートしている PC との USB 通信が可能です。

8.1.1 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、GENMAI 上の SDRAM 領域を使用しています。FAT32 でメディア領域をフォーマットして、動作確認を行っています。

8.2 アプリケーション用コンフィグレーションファイル（r_usb_pmsc_apl_config.h）

アプリケーションの動作設定は、r_usb_pmsc_apl_config.h で行います。以下に、r_usb_pmsc_apl_config.h で行う設定項目を示します。

1. USE_USBIP

   使用する USB モジュールの選択を行います。USB_IP0 / USB_IP1 のいずれかを指定してください。
   本設定は、r_usb_basic_config.h で行う、USB_CFG_USE_USBIP の設定と合わせてください。

```
#define    USE_USBIP    USE_IP0    // USB0 モジュールを使用する場合
#define    USE_USBIP    USE_IP1    // USB1 モジュールを使用する場合
```

2. USB_SPEED

   PMSC の動作スピードの選択を行います。USB_HS / USB_FS のいずれかを指定してください。

```
#define    USB_SPEED    USB_HS     // High speed で動作させる場合
#define    USB_SPEED    USB_FS     // Full speed で動作させる場合
```

3. 注意事項

   r_usb_pmsc_apl_config.h 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「4 コンパイル時の設定」を参照してください。

8.3 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	:	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	:	メインループ内で R_USB_GetEvent 関数をコールし、USB ドライバを動作させます。

PMSC は、USB ホスト(PC)からの要求をマスストレージクラスドライバ（MSCD）及びマス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、メインループ内で R_USB_GetEvent 関数のコールのみ行います。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、PMSC の 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、「5.2 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メインループ処理内で必ずコールしてください。

Figure 8.1 に、APL の処理概要を示します。

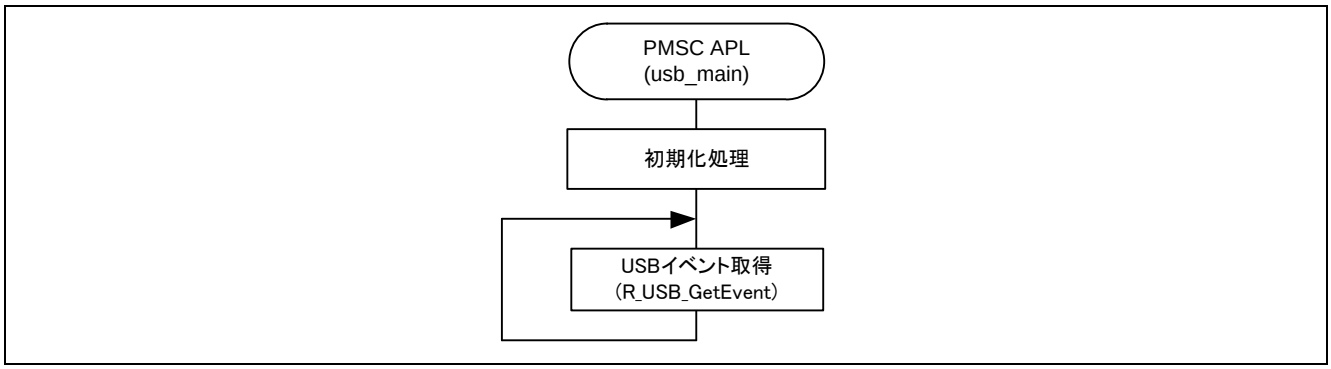

Figure 8.1 APL 処理概要

8.4 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。

[Note]

1. Vender ID、Product ID は、必ずお客様用の Vender ID、Product ID をご使用いただきますようお願いします。

PMSC のディスクリプタ一覧を Table 8.1 に示します。

Table 8.1 ディスクリプター覧

ディスクリプタ	変数名
Device Descriptor	g_msc_device[]
Device Qualifier Descriptor	g_msc_qualifier_descriptor[]
FS 用 Configuration Descriptor	g_msc_fs_configuration[]（注 1）
FS 用 Interface Descriptor	g_msc_fs_configuration[]（注 1）
FS 用 Endpoint Descriptor	g_msc_fs_configuration[]（注 1）
HS 用 Configuration Descriptor	g_msc_hs_configuration[]（注 2）
HS 用 Interface Descriptor	g_msc_hs_configuration[]（注 2）
HS 用 Endpoint Descriptor	g_msc_hs_configuration[]（注 2）
String Descriptor	g_msc_string0[]
	g_msc_string1[]
	g_msc_string2[]
	g_msc_string3[]
	g_msc_string4[]
	g_msc_string5[]
	g_msc_string6[]

注 1 g_msc_fs_configuration は、FS 用 Configuration Descriptor、FS 用 Interface Descriptor、FS 用 Endpoint Descriptor を含みます。

注 2 g_msc_hs_configuration は、HS 用 Configuration Descriptor、HS 用 Interface Descriptor、HS 用 Endpoint Descriptor を含みます。

8.4.1 g_msc_device

Device Descriptor の設定値を Table 8.2 に示します。

Table 8.2 Device Descriptor

Offset	Field	値	備考
0	bLength	USB_DD_BLENGTH	
1	bDescriptorType	USB_DT_DEVICE	
2	bcdUSB	USB_BCDNUM(下位 1Byte)	
3		USB_BCDNUM(上位 1Byte)	
4	bDeviceClass	0x00	
5	bDeviceSubClass	0x00	
6	bDeviceProtocol	0x00	
7	bMAXPacketSize0	USB_DCPMAXP	
8	idVendor	USB_VENDORID(下位 1Byte)	
9		USB_VENDORID(上位 1Byte)	
10	idProduct	USB_PRODUCTID(下位 1Byte)	
11		USB_PRODUCTID(上位 1Byte)	
12	bcdDevice	USB_RELEASE(下位 1Byte)	
13		USB_RELEASE(上位 1Byte)	
14	iManufacturer	0x01	g_msc_string1[]
15	iProduct	0x02	g_msc_string2[]
16	iSerialNumber	0x06	g_msc_string3[]
17	bNumConfigurations	USB_CONFIGNUM	

8.4.2 g_msc_qualifier_descriptor

Device Qualifier Descriptor の設定値を Table 8.3 に示します。

Table 8.3 Device Qualifier Descriptor

Offset	Field	値	備考
0	bLength	USB_QD_BLENGTH	
1	bDescriptorType	USB_DT_DEVICE_QUALIFIER	
2	bcdUSB	USB_BCDNUM(下位 1Byte)	
3		USB_BCDNUM(上位 1Byte)	
4	bDeviceClass	0x00	
5	bDeviceSubClass	0x00	
6	bDeviceProtocol	0x00	
7	bMAXPacketSize0	USB_DCPMAXP	
8	bNumConfigurations	USB_CONFIGNUM	
9	bReserved	0x00	

8.4.3 g_msc_fs_configuration

g_msc_fs_configuration は、Full Speed 動作時の Configuration Descriptor、Interface Descriptor、Endpoint Descriptor の値を含みます。Table 8.4 に各ディスクリプタの記述箇所を示します。

Table 8.4 g_msc_fs_configuration

Offset	Descriptor
0	Configuration Descriptor
9	Interface Descriptor
18～	Endpoint Descriptor

Table 8.5 Configuration Descriptor

Offset	Field	値	備考
0	bLength	USB_CD_BLENGTH	
1	bDescriptorType	USB_SOFT_CHANGE	
2	wTotalLength	USB_PMSC_CD_WTOTALLENGTH % 256	
3		USB_PMSC_CD_WTOTALLENGTH / 256	
4	bNumInterfaces	0x01	
5	bConfigurationValue	0x01	
6	iConfiguration	0x04	g_msc_string4[]
7	bmAttributes	USB_CF_RESERVED | USB_CF_SELFP	Self-powered
8	bMaxPower	10 / 2	10mA

Table 8.6 Interface Descriptor

Offset	Field	値	備考
0	bLength	USB_ID_BLENGTH	
1	bDescriptorType	USB_DT_INTERFACE	
2	bInterfaceNumber	0x00	
3	bAlternateSetting	0x00	
4	bNumEndpoints	0x02	
5	bInterfaceClass	USB_IFCLS_MAS	MSC
6	bInterfaceSubClass	USB_INTERFACE_SUBCLASS	
7	bInterfaceProtocol	USB_BOTP	
8	iInterface	0x03	g_msc_string3[]

Table 8.7 Endpoint Descriptor 0

Offset	Field	値	備考
0	bLength	USB_ED_BLENGTH	
1	bDescriptorType	USB_DT_ENDPOINT	
2	bEndpointAddress	USB_EP_IN | USB_EP1	b'7 : Direction b'6 – b'4: Reserved b'3 – b'0: Endpoint number
3	bmAttribute	USB_EP_BULK	b'5 – b'4: Usage type b'3 – b'2: Synchronization Type b'1 – b'0: Transfer type
4	wMaxPacketSize	64	
5		0	
6	bInterval	0x00	

Table 8.8 Endpoint Descriptor 1

Offset	Field	値	備考
0	bLength	USB_ED_BLENGTH	
1	bDescriptorType	USB_DT_ENDPOINT	
2	bEndpointAddress	USB_EP_OUT | USB_EP2	b'7 : Direction b'6 – b'4: Reserved b'3 – b'0: Endpoint number
3	bmAttribute	USB_EP_BULK	b'5 – b'4: Usage type b'3 – b'2: Synchronization Type b'1 – b'0: Transfer type
4	wMaxPacketSize	64	
5		0	
6	bInterval	0x00	

8.4.4 g_msc_hs_configuration

g_msc_fs_configuration は、High Speed 動作時の Configuration Descriptor、Interface Descriptor、Endpoint Descriptor の値を含みます。Table 8.9 に各ディスクリプタの記述箇所を示します。

Table 8.9 g_msc_hs_configuration

Offset	Descriptor
0	Configuration Descriptor
9	Interface Descriptor
18～	Endpoint Descriptor

Table 8.10 Configuration Descriptor

Offset	Field	値	備考
0	bLength	USB_CD_BLENGTH	
1	bDescriptorType	USB_SOFT_CHANGE	
2	wTotalLength	USB_PMSC_CD_WTOTALLENGTH % 256	
3		USB_PMSC_CD_WTOTALLENGTH / 256	
4	bNumInterfaces	0x01	
5	bConfigurationValue	0x01	
6	iConfiguration	0x05	g_msc_string5[]
7	bmAttributes	USB_CF_RESERVED | USB_CF_SELFP	Self-powered
8	bMaxPower	10 / 2	10mA

Table 8.11 Interface Descriptor

Offset	Field	値	備考
0	bLength	USB_ID_BLENGTH	
1	bDescriptorType	USB_DT_INTERFACE	
2	bInterfaceNumber	0x00	
3	bAlternateSetting	0x00	
4	bNumEndpoints	0x02	
5	bInterfaceClass	USB_IFCLS_MAS	MSC
6	bInterfaceSubClass	USB_INTERFACE_SUBCLASS	
7	bInterfaceProtocol	USB_BOTP	
8	iInterface	0x03	g_msc_string3[]

Table 8.12 Endpoint Descriptor 0

Offset	Field	値	備考
0	bLength	USB_ED_BLENGTH	
1	bDescriptorType	USB_DT_ENDPOINT	
2	bEndpointAddress	USB_EP_IN | USB_EP1	b’7 : Direction b’6 – b’4: Reserved b’3 – b’0: Endpoint number
3	bmAttribute	USB_EP_BULK	b’5 – b’4: Usage type b’3 – b’2: Synchronization Type b’1 – b’0: Transfer type
4	wMaxPacketSize	0	
5		2	512byte
6	bInterval	0x00	

Table 8.13 Endpoint Descriptor 1

Offset	Field	値	備考
0	bLength	USB_ED_BLENGTH	
1	bDescriptorType	USB_DT_ENDPOINT	
2	bEndpointAddress	USB_EP_OUT | USB_EP2	b’7 : Direction b’6 – b’4: Reserved b’3 – b’0: Endpoint number
3	bmAttribute	USB_EP_BULK	b’5 – b’4: Usage type b’3 – b’2: Synchronization Type b’1 – b’0: Transfer type
4	wMaxPacketSize	0	
5		2	512byte
6	bInterval	0x00	

8.4.5 g_msc_stringX（X = 0～6）

String Descriptor の設定値を Table 8.14～Table 8.20 に示します。

Table 8.14 g_msc_string0

Offset	Field	値	備考
0	bLength	USB_PMSC_SD0_BLENGTH	
1	bDescriptorType	USB_DT_STRING	
2	wLANGID[0]	0x09(下位 1Byte)	
3		0x04(上位 1Byte)	

Table 8.15 g_msc_string1

Offset	Field	値	備考
0	bLength	USB_PMSC_SD1_BLENGTH	
1	bDescriptorType	USB_DT_STRING	
2	bString	'R'	
3		0x00	
4		'E'	
5		0x00	
6		'N'	
7		0x00	
8		'E'	
9		0x00	
10		'S'	
11		0x00	
12		'A'	
13		0x00	
14		'S'	
15		0x00	

Table 8.16 g_msc_string2

Offset	Field	値	備考
0	bLength	USB_PMSC_SD2_BLENGTH	
1	bDescriptorType	USB_DT_STRING	
2	bString	'U'	
3		0x00	
4		'S'	
5		0x00	
6		'B'	
7		0x00	
8		' '	
9		0x00	
10		'M'	
11		0x00	
12		'a'	
13		0x00	
14		's'	
15		0x00	
16		's'	
17		0x00	
18		' '	
19		0x00	

20		'S'	
21		0x00	
22		't'	
23		0x00	
24		'o'	
25		0x00	
26		'r'	
27		0x00	
28		'a'	
29		0x00	
30		'g'	
31		0x00	
32		'e'	
33		0x00	

Table 8.17 g_msc_string3

Offset	Field	値	備考
0	bLength	USB_PMSC_SD3_BLENGTH	
1	bDescriptorType	USB_DT_STRING	
2	bString	'R'	
3		0x00	
4		'e'	
5		0x00	
6		'm’	
7		0x00	
8		'o'	
9		0x00	
10		'v'	
11		0x00	
12		'a'	
13		0x00	
14		'b'	
15		0x00	
16		'l'	
17		0x00	
18		'e'	
19		0x00	
20		' '	
21		0x00	
22		'D'	
23		0x00	
24		'r'	
25		0x00	
26		'i'	
27		0x00	
28		'v'	
29		0x00	
30		'e'	
31		0x00	

Table 8.18 g_msc_string4

Offset	Field	値	備考
0	bLength	USB_PMSC_SD4_BLENGTH	
1	bDescriptorType	USB_DT_STRING	
2	bString	'F'	
3		0x00	
4		'u'	
5		0x00	
6		'l'	
7		0x00	
8		'l'	
9		0x00	
10		'-'	
11		0x00	
12		'S'	
13		0x00	
14		'p'	
15		0x00	
16		'e'	
17		0x00	
18		'e'	
19		0x00	
20		'd'	
21		0x00	

Table 8.19 g_msc_string5

Offset	Field	値	備考
0	bLength	USB_PMSC_SD4_BLENGTH	
1	bDescriptorType	USB_DT_STRING	
2	bString	'H'	
3		0x00	
4		'I'	
5		0x00	
6		'-'	
7		0x00	
8		'S'	
9		0x00	
10		'p'	
11		0x00	
12		'e'	
13		0x00	
14		'e'	
15		0x00	
16		'd'	
17		0x00	

Table 8.20 g_msc_string6

Offset	Field	値	備考
0	bLength	USB_PMSC_SD6_BLENGTH	
1	bDescriptorType	USB_DT_STRING	
2	bString	'0'	
3		0x00	
4		'0'	
5		0x00	
6		'0'	
7		0x00	
8		'0'	
9		0x00	
10		'0'	
11		0x00	
12		'0'	
13		0x00	
14		'0'	
15		0x00	
16		'0'	
17		0x00	
18		'0'	
19		0x00	
20		'0'	
21		0x00	
22		'0'	
23		0x00	
24		'1'	
25		0x00	

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
Rev.1.20	2017.08.31		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、その他の不適切に使用しないでください。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、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、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損害が生じても、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。また、当社製品および技術を、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル（無人航空機を含みます。）の開発、設計、製造、使用もしくは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、かつ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。
   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. お客様の転売、貸与等により、本書（本ご注意書きを含みます。）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、当社は一切その責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
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. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。

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。

(Rev.3.0-1　2016.11)

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/