RZ/T1 グループ

R01AN2632JJ0130
Rev.1.30
Dec 07, 2017

USB Peripheral Mass Storage Class Driver (PMSC)

要旨

本アプリケーションノートでは、Peripheral Mass Storage Class Driver について説明します。本ドライバは USB Peripheral Basic Firmware（USB-BASIC-FW）と組み合わせることで動作します。以降、本サンプルソフトウェアを PMSC と称します。
本アプリケーションノートのサンプルプログラムは「RZ/T1 グループ初期設定 Rev.1.30」をベースに作成しています。
動作環境については「RZ/T1 グループ初期設定アプリケーションノート(R01AN2554JJ0130)」を参照してください。

対象デバイス

RZ/T1 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. RZ/T1 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. R01UH0483JJ0130)
5. RZ/T1 グループ初期設定 (ドキュメント No. R01AN2554 JJ0130)
6. USB Peripheral Basic Firmware アプリケーションノート (ドキュメント No. R01AN2630 JJ0130)

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

PMSC は、USB マスストレージクラスの BOT プロトコルで構築されています。USB-BASIC-FW 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。
- 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にカプセル化されたストレージコマンドに対する応答

制限事項

本モジュールには、以下の制限事項があります。
- 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。）

用語一覧

APL	:	Application program
BOT	:	Mass storage class Bulk Only Transport
CBW	:	Command Block Wrapper
CSW	:	Command Status Wrapper
PCD	:	Peripheral Control Driver of USB-BASIC-FW
PMSC	:	Peripheral Mass Storage USB Class Driver
USB-BASIC-FW	:	USB Peripheral basic firmware for RZ/T1 グループ

2. ソフトウェア構成

Figure 2-1 にソフトウェア構成、Table 2-1 にモジュール機能概要を示します。

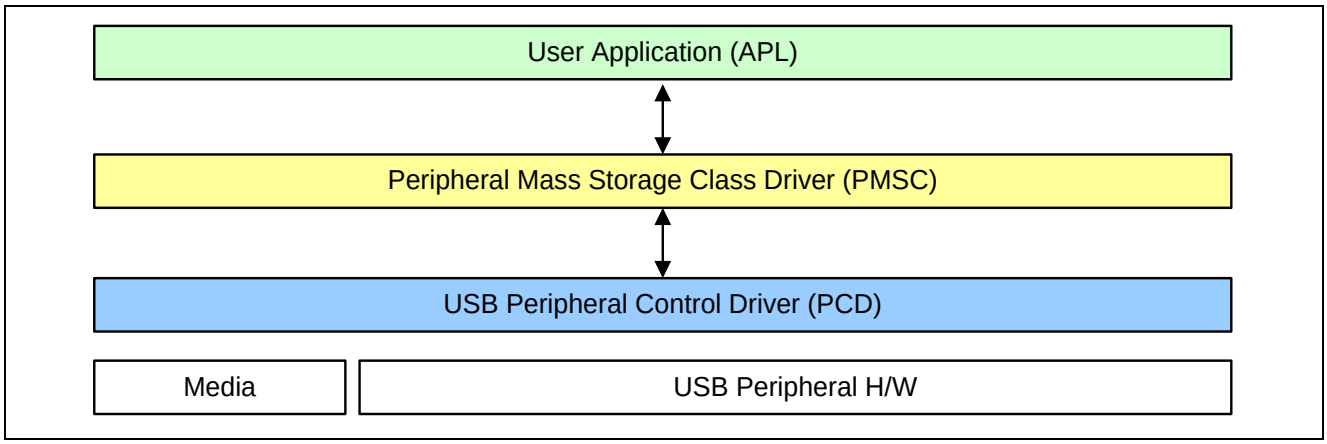


Figure 2-1 ソフトウェア構成図

Table 2-1 モジュール機能概要

モジュール名	機能概要
APL	ユーザーアプリケーションプログラム（システムにあわせてご用意ください）
PMSC	ペリフェラルマスストレージクラスドライバ ・ クラスリクエスト応答 ・ BOT プロトコル制御 ・ CBW の受信、解析 ・ ストレージコマンド処理 ・ CSW の生成、送信 ・ Media アクセス
PCD	USB Peripheral H/W 制御ドライバ（USB-BASIC-FW）

3. ペリフェラルマスストレージクラスドライバ (PMSC)

3.1 基本機能

PMSC の機能を以下に示します。
(1). 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
(2). BOT にカプセル化されたストレージコマンドに対する応答

3.2 クラスリクエスト

PMSC がサポートするクラスリクエストを Table 3-1 に示します。

Table 3-1 クラスリクエスト

リクエスト	bRequest	説明	対応
Mass Storage Reset	0xFF	マスストレージデバイスと接続インタ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	○

○：実装　×：未実装(Stall 応答)

3.3 BOT プロトコル

BOT 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 CBW、CSW の形式で転送します。
BOT プロトコル概要を Figure 3-1 に示します。

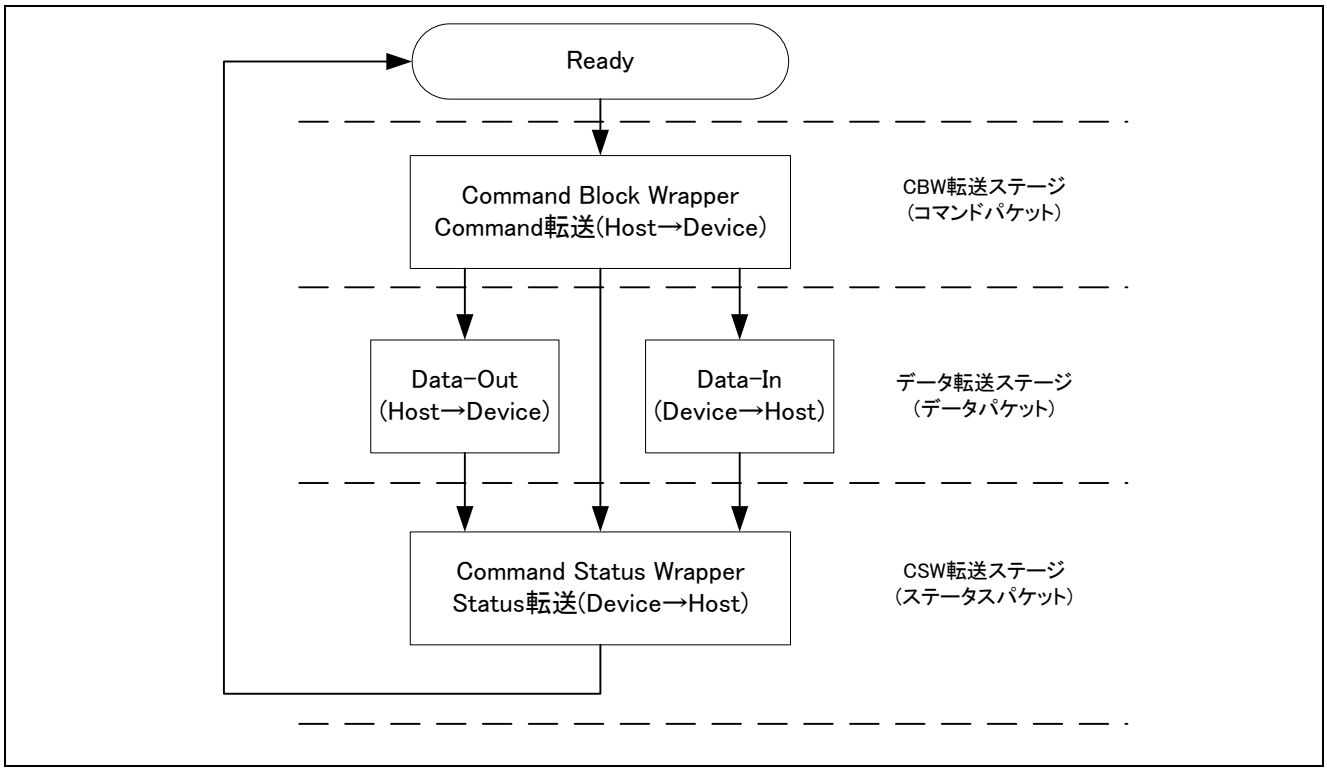


Figure 3-1 BOT プロトコル概要

3.3.1 CBW 処理

Host から CBW を受け取った PMSC は、まず CBW の有効性を確認します。
CBW が有効であった場合は、CBW 内のストレージコマンド（CBWCB）を解析し、解析した情報（コマンドの有効性、データ転送方向とサイズ）を基に処理します。

転送データサイズが USB_ATAPI_BLOCK_UNIT を超えた場合はデータを分割して転送を行います。
READ10 以外のデータ送信コマンドは PMSC が用意している返答用データテーブルから送信データを作成します。返答用データテーブルはストレージコマンドセットで定められた値によって構成しています。

PMSC がサポートするストレージコマンドを Table 3-2 に示します。

Table 3-2 ストレージコマンド

コマンド名	コード	説明	種別	対応
TEST_UNIT_READY	0x00	ペリフェラル機器の状態確認	No Data	○
REQUEST_SENSE	0x03	ペリフェラル機器の状態取得	IN	○
FORMAT_UNIT	0x04	論理ユニットのフォーマット	OUT	×
INQUIRY	0x12	論理ユニットのパラメータ情報取得	IN	○
MODE_SELECT6	0x15	パラメータ指定	OUT	×
MODE_SENSE6	0x1A	論理ユニットのパラメータ取得	IN	×
START_STOP_UNIT	0x1B	論理ユニットのアクセス許可／禁止	No Data	×
PREVENT_ALLOW	0x1E	メディアの取り出し許可／禁止	No Data	○
READ_FORMAT_CAPACITY	0x23	フォーマット可能な容量取得	IN	○
READ_CAPACITY	0x25	論理ユニットの容量情報取得	IN	○
READ10	0x28	データ読み出し	IN	○
WRITE10	0x2A	データ書き込み	OUT	○
SEEK	0x2B	論理ブロックアドレスに移動	No Data	×
WRITE_AND_VERIFY	0x2E	確認付きデータ書き込み	OUT	×
VERIFY10	0x2F	データ確認	No Data	×
MODE_SELECT10	0x55	パラメータ指定	OUT	○
MODE_SENSE10	0x5A	論理ユニットのパラメータ取得	IN	○

○：実装　×：未実装(Stall 応答)

3.3.2 送受信データのないストレージコマンドのシーケンス

(a). CBW 転送ステージ

PMSC は PCD へ CBW 受信要求を行います。PCD は CBW を受信すると、CBW 受信要求時に設定されたコールバック関数を呼出します。コールバックを受けた PMSC が CBW の有効性を確認し、CBWCB を解析します。

PMSC は送受信データがないコマンドであることを確認し、ストレージコマンド解析結果と CBW 内情報を比較し、ストレージコマンドを実行します。

(b). CSW 転送ステージ

PMSC は実行結果から CSW を作成し、PCD を介して Host に CSW を送信します。

Figure 3-2 に、シーケンスを示します。

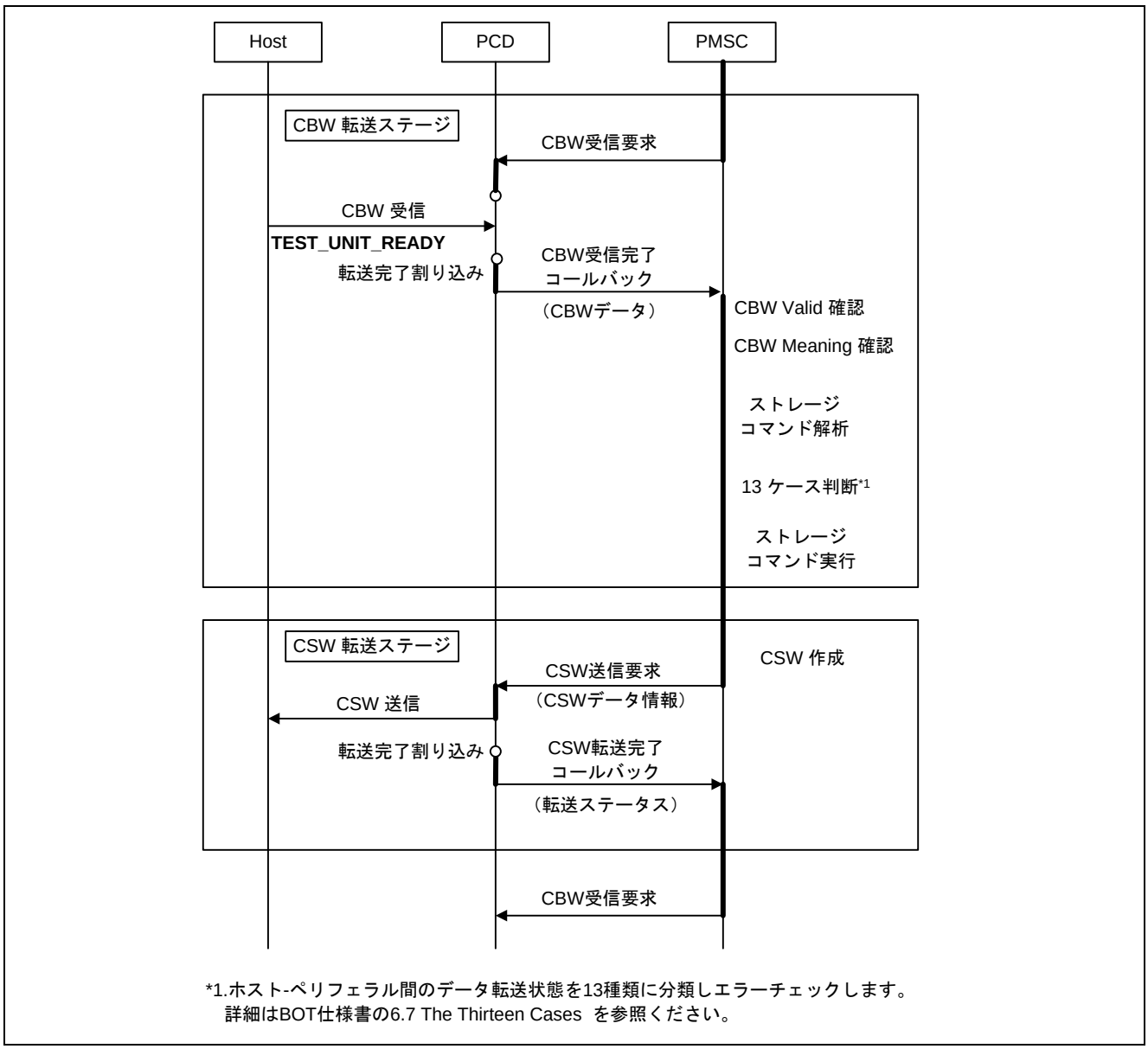


Figure 3-2 送受信データのないストレージコマンドのシーケンス

3.3.3 送信（IN）データがあるストレージコマンドのシーケンス

(a). CBW 転送ステージ

3.3.2 (a) と同様。

(b). DATA 転送ステージ

PMSC は実行結果からデータ格納領域とデータサイズを PCD に通知し Host とデータ通信を行います。

PMSC は PCD より送信完了が通知されると、要求したサイズの送信が完了しているか確認します。完了していない場合は、再度 DATA 送信要求をして、DATA 転送ステージを継続します。完了していた場合は、CSW 転送ステージに移行します。

(c). CSW 転送ステージ

3.3.2 (b) と同様。

Figure 3-3 に、シーケンスを示します。

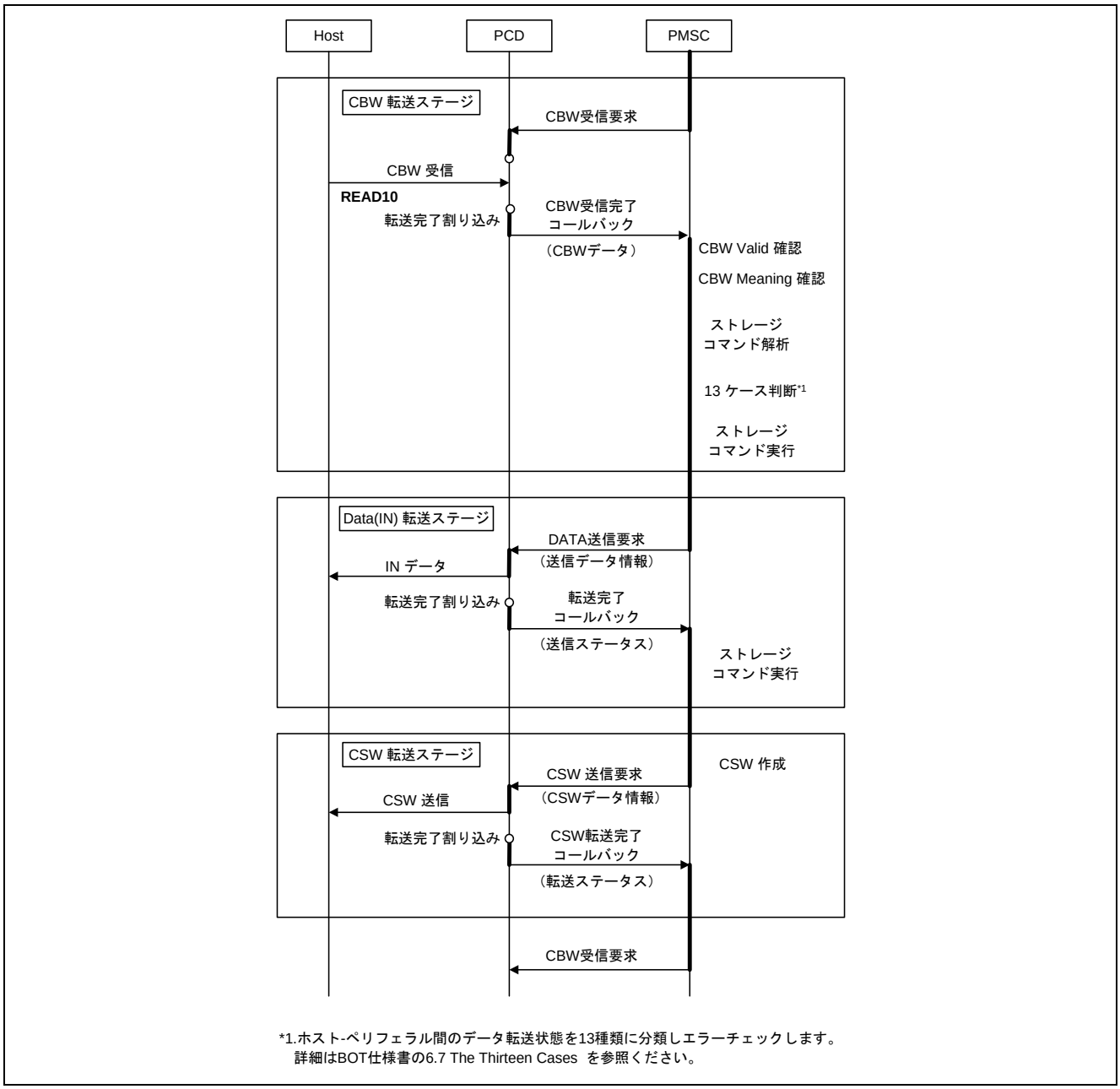


Figure 3-3 送信（IN）データがあるストレージコマンドのシーケンス

3.3.4 受信（OUT）データがあるストレージコマンドのシーケンス

(a). CBW 転送ステージ

3.3.2 (a) と同様。

(b). DATA 転送ステージ

PMSC は実行結果からデータ格納領域とデータサイズを PCD に通知し Host とデータ通信を行います。

PMSC は PCD より受信完了が通知されると、要求したサイズの受信が完了しているか確認します。完了していない場合は、再度 DATA 受信要求をして、DATA 転送ステージを継続します。完了していた場合は、CSW 転送ステージに移行します。

(c). CSW 転送ステージ

3.3.2 (b) と同様。

Figure 3-4 に、シーケンスを示します。

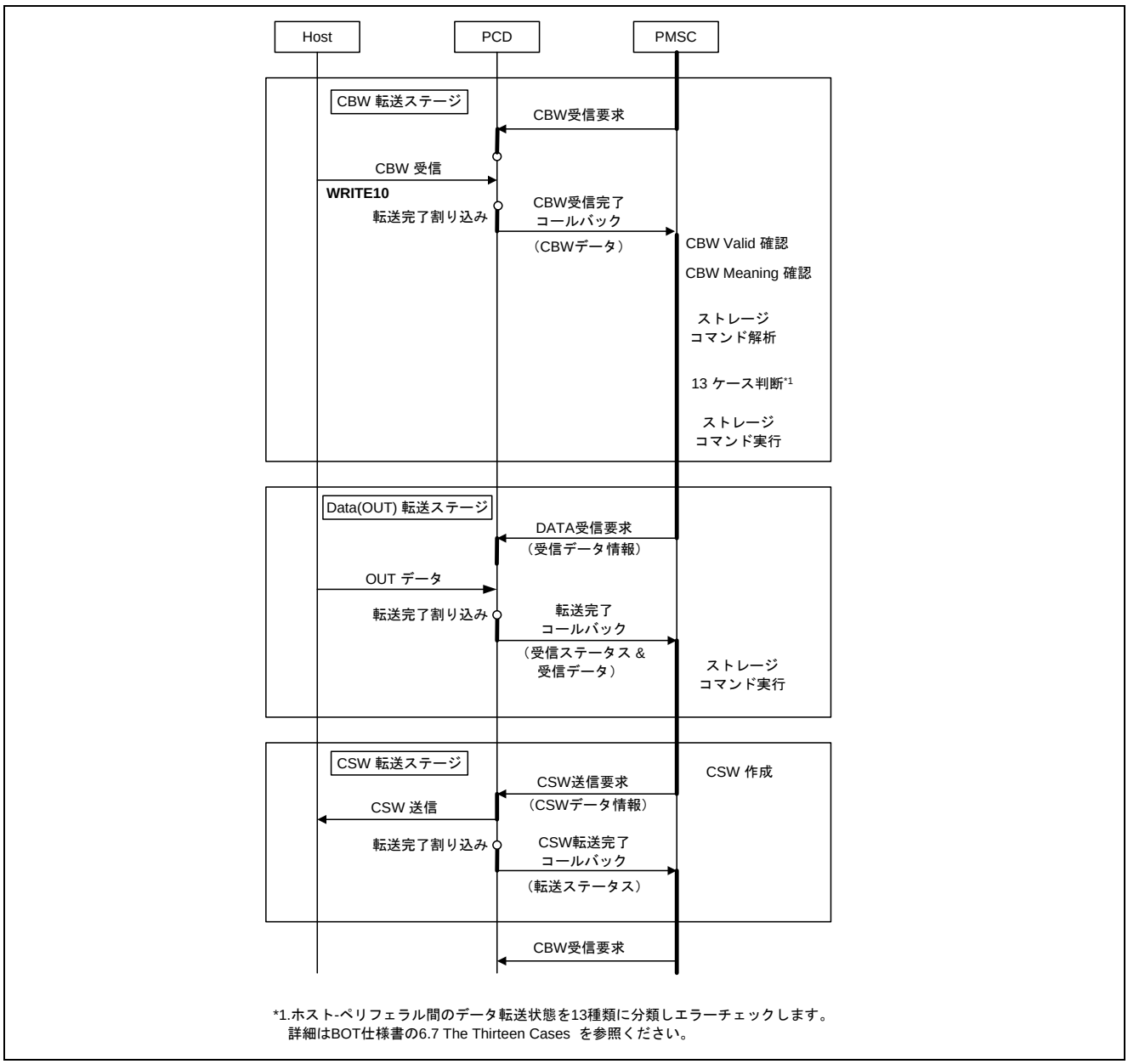


Figure 3-4 受信（OUT）データがあるストレージコマンドのシーケンス

3.3.5 クラスリクエストのシーケンス

(a). Setup ステージ

PCD は SETUP を受信すると、SETUP ステージになり、PMSC に受信を通知します。
PMSC は SETUP に応じて、応答データ作成処理をします。

(b). データステージ

PMSC は PCD に対して送信要求をします。PCD はコントロール転送のデータステージを実行してデータステージを終了します。

(c). ステータスステージ

PCD はステータスステージを実行してコントロール転送を終了します。

Figure 3-5 に、シーケンスを示します。

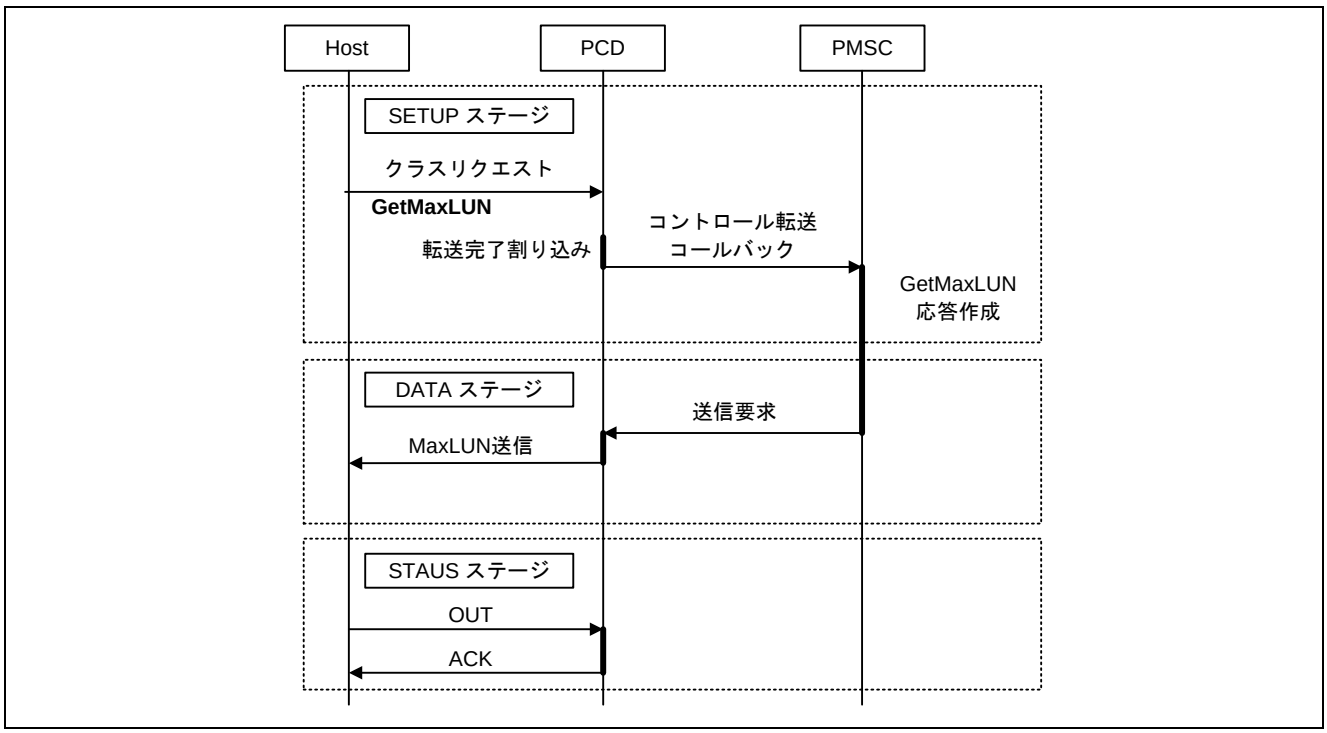


Figure 3-5 クラスリクエストのシーケンス

3.4 API

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_pmsc_if.h に記載しています。
本モジュールのコンフィグレーションオプションの設定は、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、Table 3-3 に示します。

Table 3-3 PMSC コンフィグレーションオプション

定義名	デフォルト値	説明
USB_PMSC_USE_PIPE_IN	USB_PIPE1	IN 転送パイプ番号
USB_PMSC_USE_PIPE_OUT	USB_PIPE2	OUT 転送パイプ番号
USB_ATAPI_BLOCK_UNIT	0x200ul	ATAPI ブロックサイズ（バイト単位）
USB_RAM_PP	0	RAM ディスクタイプ定義
USB_SDRAM_PP	1	RAM ディスクタイプ定義
USB_MEDIA_TYPE_PP	USB_SDRAM_PP	メディアタイプ設定
RAMDISK_MEDIA_SIZE	(64ul * 1024ul * 1024ul)	RAM ディスクサイズ（バイト単位）
RAMDISK_SECT_SIZE	0x200ul	RAM ディスクセクタサイズ（バイト単位）
RAMDISK_TOTALSECT	(RAMDISK_MEDIASIZE / RAMDISK_SECTSIZE)	RAM ディスクセクタ数
MEDIA_ADDRESS	0x68000000	メディアの先頭アドレス

Table 3-4 に API 関数を示します。

Table 3-4 PMSC API 関数一覧

関数名	説明
R_usb_pmsc_Open	PMSC オープン関数
R_usb_pmsc_SetInterface	PMSC SET_INTERFACE 処理関数
R_usb_pmsc_CtrlTrans	PMSC コントロール転送処理関数
R_usb_pmsc_poll	PMSC ポーリング処理関数

3.4.1 R_usb_pmsc_Open

PMSC オープン関数

形式

void R_usb_pmsc_Open(void)

引数

― ―

戻り値

― ―

解説

本 API は USB_PCDREG_t 構造体のメンバ devconfig にコールバック関数として登録してください。
本 API は CBW 受信設定を行います。

補足

―

使用例

```
void usb_pmsc_task_start( void )
{
 USB_PCDREG_t   driver;

 driver.devconfig  = &R_usb_pmsc_Open;
 R_usb_pstd_DriverRegistration(&driver);
}
```

3.4.2 R_usb_pmsc_SetInterface

PMSC SET_INTERFACE 処理関数

形式

void	R_usb_pmsc_SetInterface(uin16_t data1)

引数

data1	Alternate 番号

戻り値

－	－

解説

本 API は USB_PCDREG_t 構造体のメンバ interface にコールバック関数として登録してください。
本 API は CBW 受信設定を行います。

補足

－

使用例

```
void usb_pmsc_task_start( void )
{
 USB_PCDREG_t   driver;

 driver.interface  = &R_usb_pmsc_SetInterface;
 R_usb_pstd_DriverRegistration(&driver);
}
```

3.4.3 R_usb_pmsc_CtrlTrans

PMSC コントロール転送処理

形式

void R_usb_pmsc_CtrlTrans (USB_REQUEST_t *preq, uint16_t ctsq)

引数

*preq	クラスリクエストメッセージのポインタ
ctsq	コントロール転送ステージ情報 USB_CS_IDST : Idle or setup stage USB_CS_RDDS : Control read data stage USB_CS_WRDS : Control write data stage USB_CS_WRND : Control write nodata status stage USB_CS_RDSS : Control read status stage USB_CS_WRSS : Control write status stage USB_CS_SQER : Sequence error

戻り値

—	—

解説

本 API は USB_PCDREG_t 構造体のメンバ ctrltrans にコールバック関数として登録してください。
リクエストタイプがマスストレージクラスリクエストの場合、コントロール転送ステージに対応した処理を呼び出します。

補足

—

使用例

```
void usb_pmsc_task_start( void )
{
 USB_PCDREG_t   driver;

 driver.ctrltrans  = &R_usb_pmsc_CtrlTrans;
 R_usb_pstd_DriverRegistration(&driver);
}
```

3.4.4 R_usb_pmsc_poll

PMSC ポーリング処理関数

形式

void	R_usb_pmsc_poll(void)

引数

―	―

戻り値

―	―

解説

本 API はメインループ内で呼び出してください。
転送完了の有無を判別し、転送完了の場合、BOT プロトコル処理シーケンスを進めます。

補足

―

使用例

```
void usb_apl(void)
{
    while( 1 )
    {
        R_usb_pstd_poll();
        R_usb_pmsc_poll();
    }
}
```

4. サンプルアプリケーション

本章では、PMSC と USB-BASIC-FW を組み合わせ、USB ドライバとして使用するために必要な初期設定の方法と、メインルーチン処理方法及び API 関数を使用したデータ転送例を示します。

4.1 動作環境

PMSC の動作環境例を Figure 4-1 に示します。
また、動作確認を行った OS 環境を Table 4-1 に示します。

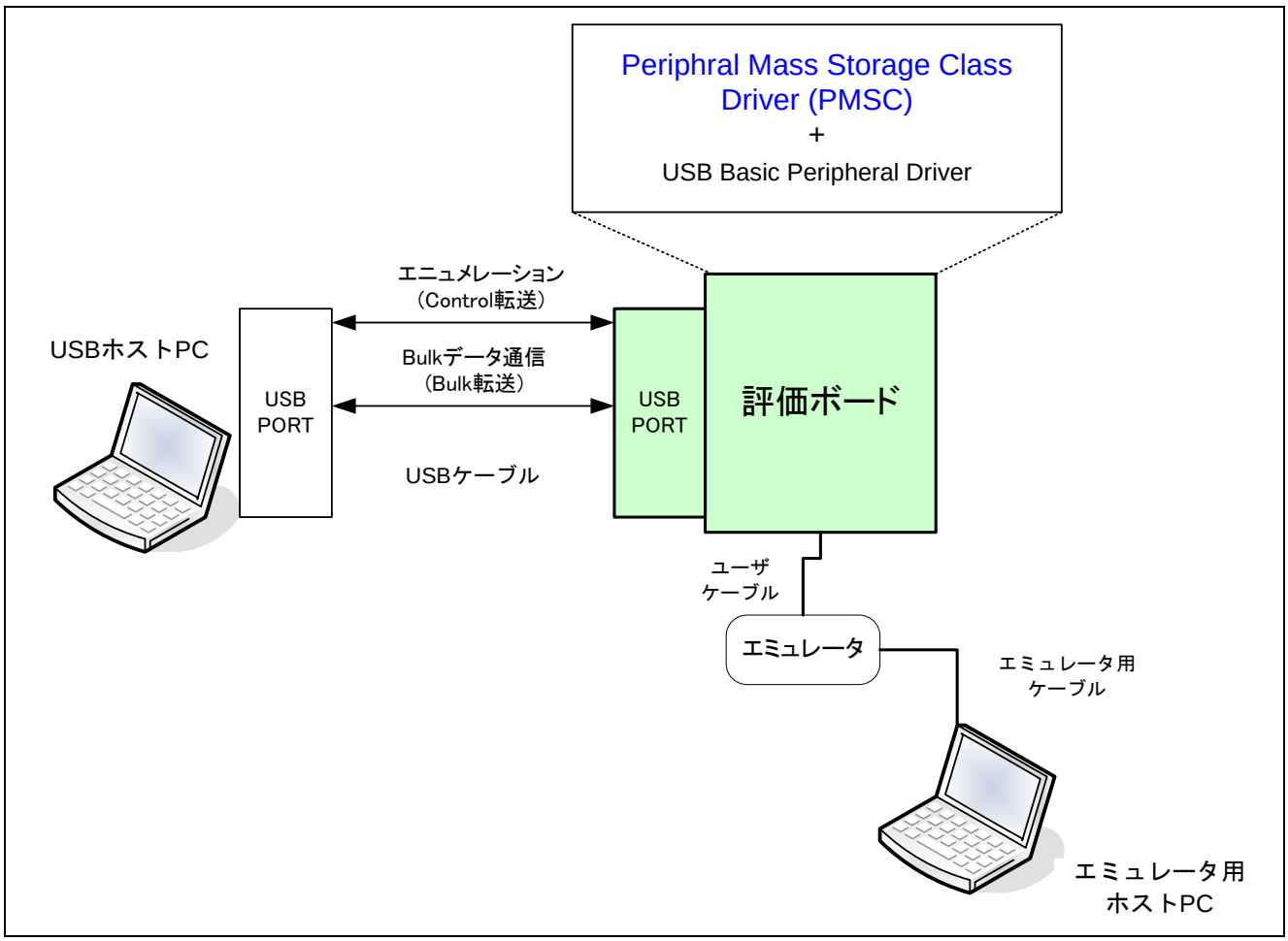


Figure 4-1 動作環境例

Table 4-1 動作確認済み OS

OS 名	備考
Windows 7 32bit	—
Windows 7 64bit	—
Windows 8.1 32bit	—
Windows 8.1 64bit	—
Windows 10 32bit	—
Windows 10 64bit	—

4.2 仕様概要

サンプルアプリケーションは、初期設定とメインループの 2 つの部分で構成されています。

PMSC が USB ホストからの要求に対して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をするので、サンプルアプリケーションは、ホストからの転送されたデータに対して処理をせず、USB ドライバを定期的にコールするのみです。

サンプルアプリケーションの処理フローを Figure 4-2 に示します。

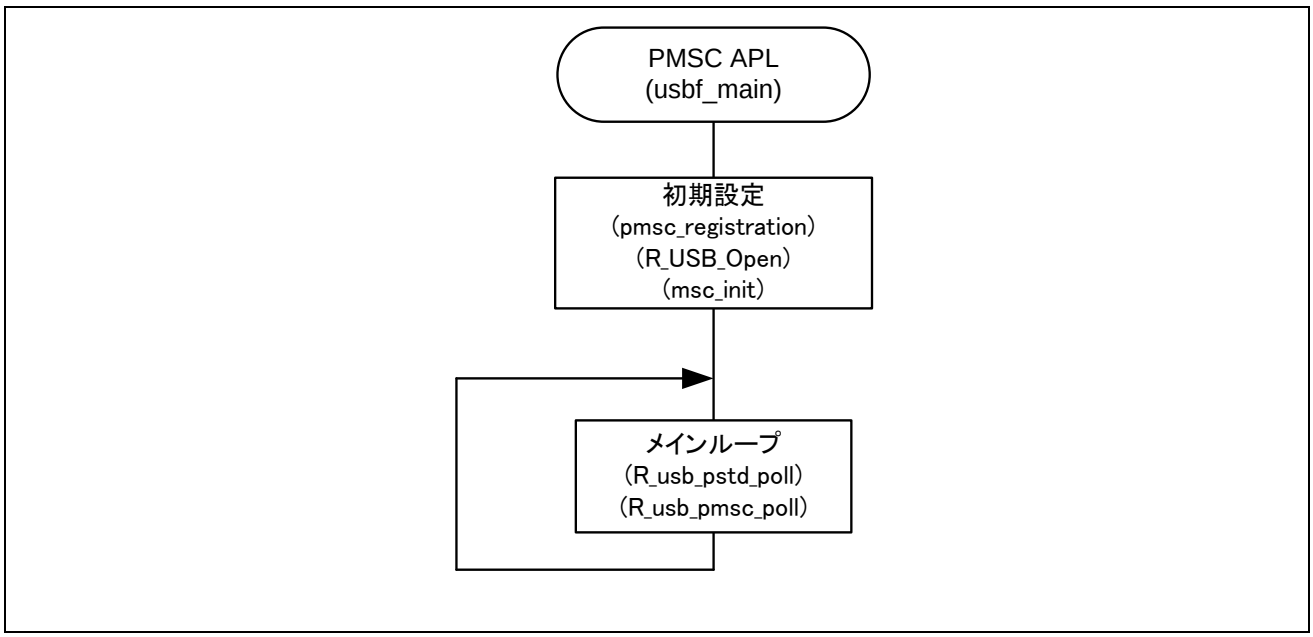


Figure 4-2 フローチャート

USB ホスト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 4-3 にパソコン接続での動作画面を示します。

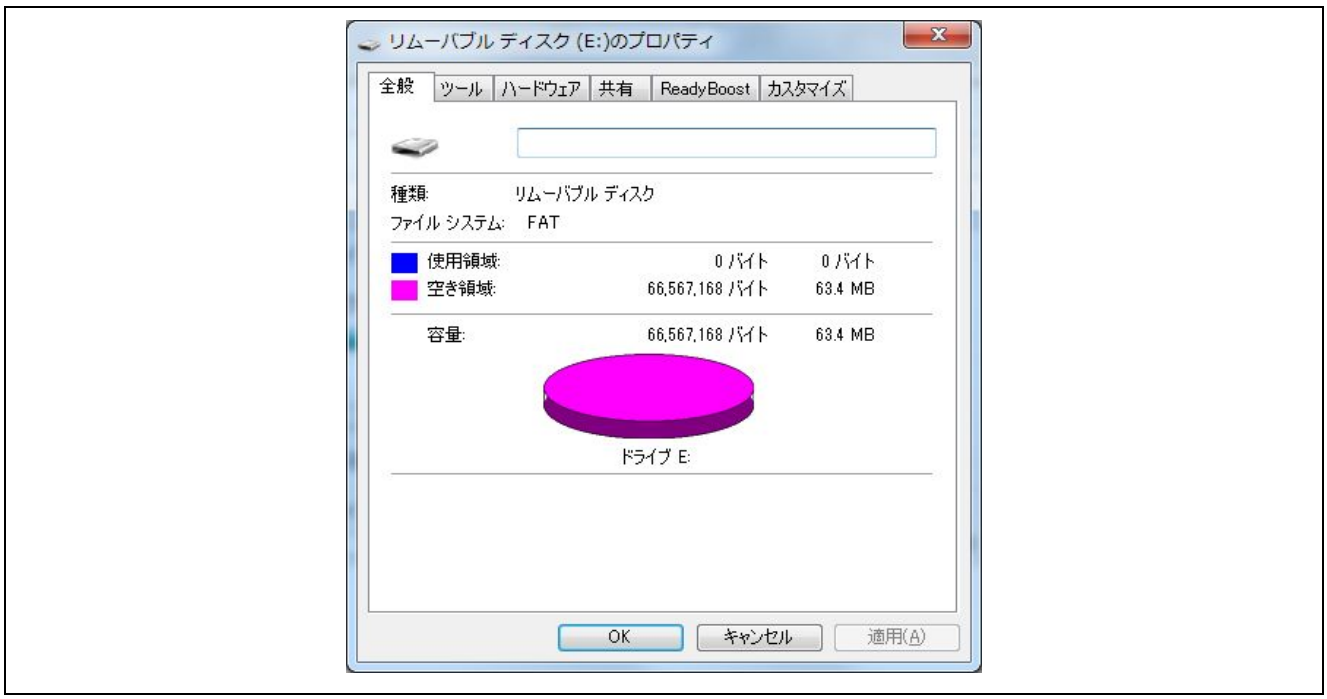


Figure 4-3 動作画面

4.3 初期設定

初期設定例を示します。

```
void usbf_main(void)
{
    /* USB ドライバの初期設定（「4.3.1」参照） */
    pmsc_registration();

    /* USB モジュールの起動（「4.3.2」参照） */
    R_USB_Open();

    /* アプリケーションの初期設定（「4.3.3」参照） */
    msc_init();

    /* メインループ */
    while(1)
    {
        R_usb_pstd_poll();
        R_usb_pmsc_poll();
    }
}
```

4.3.1 USB ドライバの初期設定

クラスドライバ登録用構造体（USB_PCDREG_t）の各メンバに情報を設定後、R_usb_pstd_DriverRegistration()を呼び出すことで USB-BASIC-FW に対するクラスドライバの情報を登録します。

パイプ情報テーブルおよびディスクリプタ情報は r_usb_pmsc_descriptor.c に記述しています。

ディスクリプタ情報は USB 規格書をもとに作成してください。

USB_PCDREG_t で宣言された構造体に設定する情報例を以下に示します。

```
void pmsc_registration(void)
```

```
{
  USB_PCDREG_t driver;      // クラスドライバ登録用構造体

  /* パイプ情報テーブルを設定 */
  driver.pipetbl     = &usb_gpmsc_EpTbl[0];
  /* Device Descriptor テーブルを設定 */
  driver.devicetbl   = (uint8_t*)&usb_gpmsc_DeviceDescriptor;
  /* Qualifier Descriptor テーブルを設定 */
  driver.qualitbl    = (uint8_t*)&usb_gpmsc_QualifierDescriptor;
  /* Configuration Descriptor テーブルを設定 */
  driver.configtbl   = (uint8_t**)&usb_gpmsc_ConPtr;           // (注 1)
  /* Other Configuration Descriptor テーブルを設定 */
  driver.othertbl    = (uint8_t**)&usb_gpmsc_ConPtrOther;      // (注 1)
  /* String Descriptor テーブルを設定 */
  driver.stringtbl   = (uint8_t**)&usb_gpmsc_StrPtr;           // (注 1)

  /* デフォルトステート遷移時に呼び出される関数を設定 */
  driver.devdefault   = &msc_default;
  /* エニュメレーション完了時に呼び出される関数を設定 */
  driver.devconfig    = &msc_configured;
  /* USB デバイス切断時に呼ばれる関数を設定 */
  driver.devdetach    = &msc_detach;
  /* デバイスをサスペンド状態に移行時に呼ばれる関数を設定 */
  driver.devsuspend   = &msc_suspended;
  /* デバイスのサスペンド状態解除時に呼ばれる関数を設定 */
  driver.devresume    = &msc_resume;
  /* インタフェース変更時に呼ばれる関数を設定 */
  driver.interface    = &R_usb_pmsc_SetInterface;
  /* 標準リクエスト以外のコントロール転送処理時に呼ばれる関数を設定 */
  driver.ctrltrans    = &R_usb_pmsc_CtrlTrans;

  /* PCD へクラスドライバ情報を登録 */
  R_usb_pstd_DriverRegistration(&driver);
}
```

（注1）ディスクリプタテーブルを設定した配列の先頭アドレスを設定してください。

```
[例]
uint8_t *usb_gpmsc_StrPtr[] =
{
    usb_gpmsc_StringDescriptor0,
    usb_gpmsc_StringDescriptor1,
    usb_gpmsc_StringDescriptor2,
}
```

4.3.2 USB モジュールの起動

USB-BASIC-FW の API 関数 R_USB_Open()を呼び出すことで、USB モジュールをハードウェアマニュアルの初期設定シーケンスに従って設定し、USB 割り込みハンドラの登録と USB 割り込み許可設定をします。

4.3.3 アプリケーションの初期設定

サンプルアプリケーションは、リムーバブルディスクのメディア領域に SDRAM 領域を使用します。

r_ram_disk_format_data.c ファイルのグローバル変数 g_ramdisk_mem[RAMDISK_MEDIASIZE]を SDRAM 領域に割り付けて実装しています。メディア領域を変更する場合は、動作環境に応じたメモリ配置をした上で、定義値 MEDIA_ADDRESS(Table 3-3 参照)を変更してください。

ソフトウェア起動時に、領域を全てゼロクリアした後、r_ram_disk_format_data.c ファイルのグローバル変数 ram_disk_boot_sector[RAMDISK_SECTSIZE]を領域の先頭に書き込んで、FAT16 ファイルシステムフォーマットします。

Appendix A. 初期設定の変更点

USB-BASIC-FW を動作させるために「RZ/T1 グループ初期設定 Rev.1.30」を変更しています。
サンプルプログラムは、IAR embedded workbench for ARM(以下、EWARM)と DS-5 と e^2 studio の環境をサポートしています。ただし、RAM ブートでは SDRAM を使用することができないため、非対応となります。
本章では、変更点について記述します。

フォルダとファイル

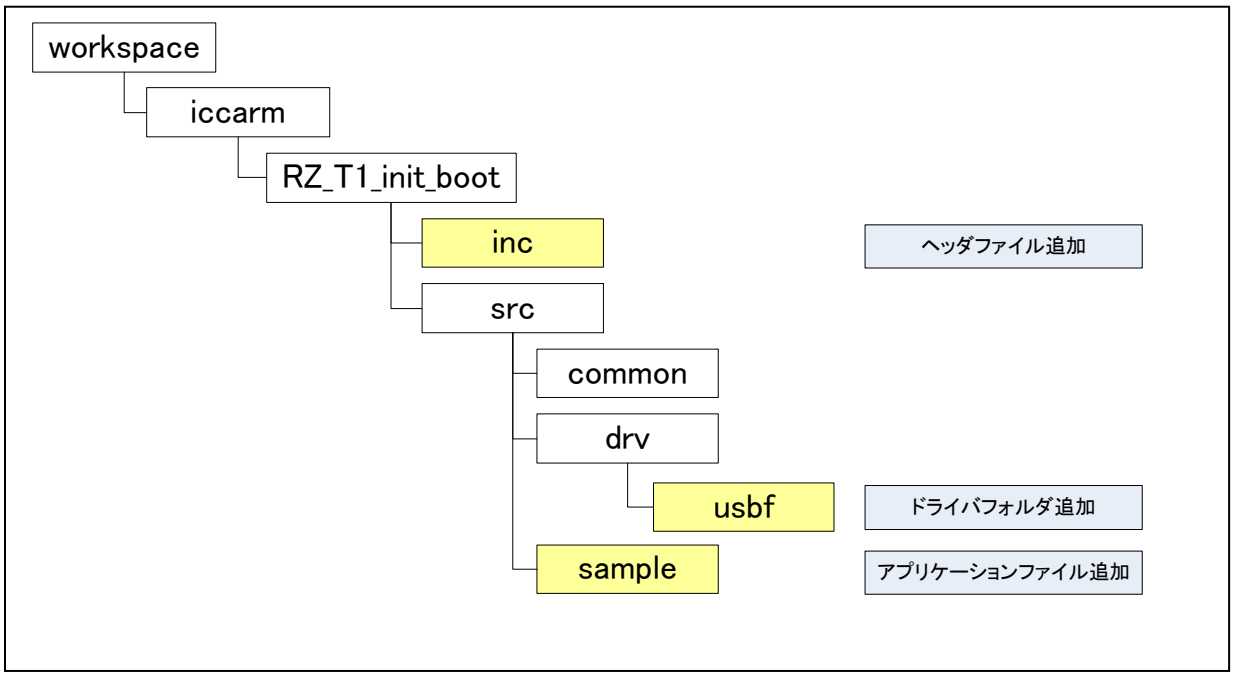
「RZ/T1 グループ初期設定 Rev.1.30」では、開発環境とブート方法によってフォルダ構成が異なります。
全ての開発環境とブート方法の各フォルダに対して、下記の変更をしています。

・”inc”フォルダに下記のファイルを追加
r_usb_basic_config.h
r_usb_basic_if.h
r_usb_cdefusbip.h
r_usb_pmsc_config.h
r_usb_pmsc_if.h

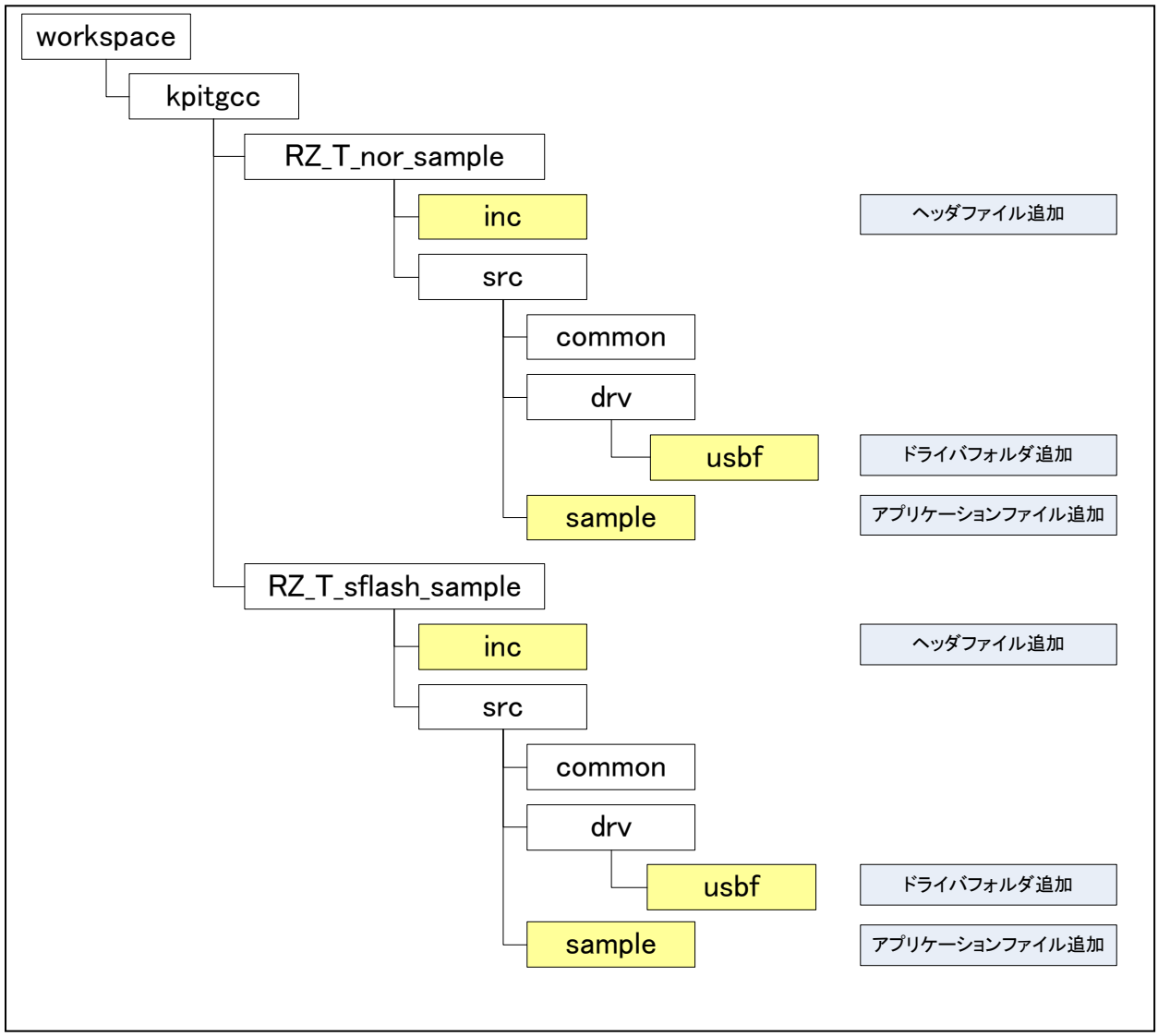
・”sample”フォルダに下記のファイルを追加
r_ram_disk_format_data.c
r_usb_pmsc_apl.c
r_usb_pmsc_descriptor.c

・”drv”フォルダに usbf フォルダと usbf フォルダ以下のファイルを追加

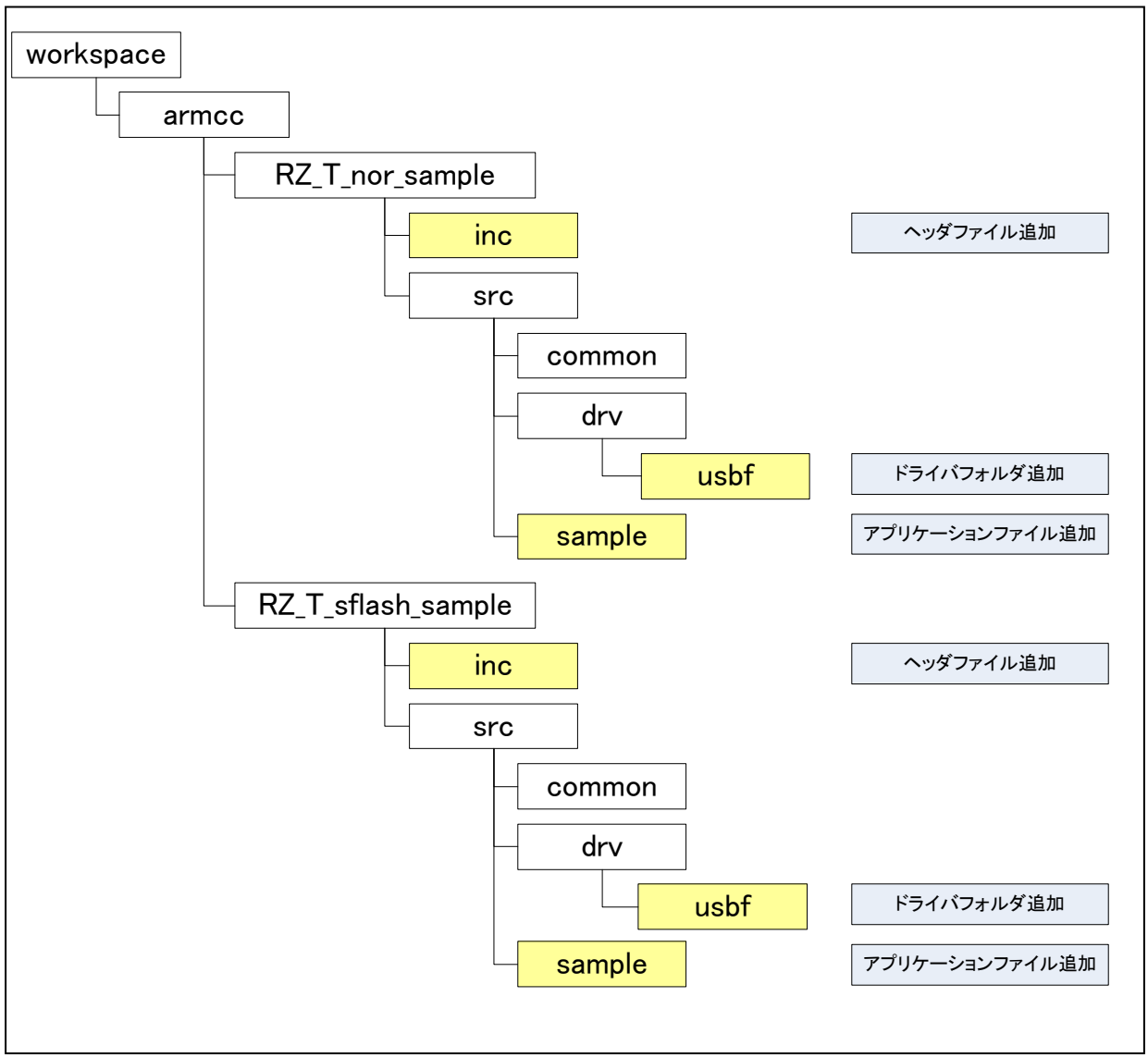
以下に EWARM のフォルダ構成を示します。



以下に e² studio のフォルダ構成を示します。



以下に DS-5 のフォルダ構成を示します。



セクション

コード領域とデータ領域のセクションサイズを変更して、以下のセクションを追加しています。

セクション名	アドレス	割り当てる変数	ファイル
USB_MEDIA	0x48000000	g_ramdisk_mem	r_ram_disk_format_data.c

e² studio

e² studio の設定画面でセクションを設定しています。
変更内容は下記の通りです。
・.data セクションの固定アドレスを 0x0007F000 から 0x00040000 に変更
・USB_MEDIA のセクション設定を追加

[プロジェクト] → [プロパティ] → [C/C++ ビルド] → [設定] → [セクション] で参照できます。

コード内の変数定義は下記の通りです。

r_ram_disk_format_data.c

```
#ifdef __GNUC__
uint8_t g_ramdisk_mem[RAMDISK_MEDIASIZE] __attribute__ ((section ("USB_MEDIA")));
#endif
```

EWARM

EWARM では、リンカ設定ファイル(.icf ファイル)でセクションを設定しています。

USB_MEDIA セクションを CS2_region に設定しています。

```
place in CS2_region { section USB_MEDIA };
```

コード内の変数定義は下記の通りです。

r_ram_disk_format_data.c

```
#ifdef __ICCARM__
#pragma location="USB_MEDIA"
uint8_t g_ramdisk_mem[RAMDISK_MEDIASIZE];
#endif
```

DS-5

DS-5 では、scatter 設定ファイルでセクションを設定しています。

USB_MEDIA セクションを設定しています。

```
LOAD_MODULE5 0x48000000    (0x48000000 - 0x48ffffff)
{
        USB_MEDIA 0x48000000  0x8000000
        {
                r_ram_disk_format_data.o(USB_MEDIA)
        }
}
```

コード内の変数定義は下記の通りです。

r_ram_disk_format_data.c

```
#ifdef __CC_ARM
#pragma arm section zidata = "USB_MEDIA"
uint8_t g_ramdisk_mem[RAMDISK_MEDIASIZE];
#endif
```

USB-BASIC-FW 関数の呼び出し

¥src¥sample¥int_main.c の main()に USB-BASIC-FW の usbf_main()の呼び出しを追加しています。

```
extern void usbf_main(void);

int main (void)
{
    /* Initialize the port function */
    port_init();

    /* Initialize the ECM function */
    ecm_init();

    /* Initialize the ICU settings */
    icu_init();

    /* USBf main */
    usbf_main();

    while (1)
    {
        /* Toggle the PF7 output level(LED0) */
        PORTF.PODR.BIT.B7 ^= 1;

        soft_wait();  // Soft wait for blinking LED0

    }

}
```

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
1.00	Aug 21, 2015	—	初版発行
1.10	Dec 25, 2015	20	Appendix A 追加
1.20	Feb 29, 2016	22,24	DS-5 関連情報追加
1.30	Dec 07, 2017	—	RZ/T1 初期設定 Ver 1.30 に対応

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/