Renesas USB MCU

USB Host Mass Storage Class Driver (HMSC) using Basic Mini Firmware

R01AN0624JJ0214
Rev.2.14
Mar 28, 2016

要旨

本資料は、Renesas USB MCUを使用した USB Host マスストレージクラスドライバ(HMSC)のアプリケーションノートです。

動作確認デバイス

R8C/3MK, R8C/34K

動作確認デバイスと同様の USB モジュールを持つ他の MCU でも本プログラムを使用することができます。このアプリケーションノートのご使用に際しては十分な評価を行ってください。

目次

1. 資料概要

本資料は、Renesas USB MCUを使用した USB Host マスストレージクラスドライバ(HMSC)及びサンプルドライバに対するアプリケーションノートです。

本 S/W では、M3S-TFAT-Tiny（以下 TFAT と略します）を使用しています。そのため、RX ファミリ M3S-TFAT-Tiny : FAT ファイルシステムソフトウェア(ドキュメント No. R20AN0038JJ)も併せてご使用ください。

1.1 機能と特長

USB Host マスストレージクラスドライバは、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。ファイルシステム、ストレージデバイスドライバと組み合わせることで BOT 対応の USB ストレージ機器と通信を行うことが可能です。

1.2 関連ドキュメント

1. Universal Serial Bus 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. Usr’s Manual: Hardware
5. Renesas USB MCU USB Basic Mini Firmware アプリケーションノート(ドキュメント No. R01AN0326JJ)
6. RX ファミリ M3S-TFAT-Tiny : FAT ファイルシステムソフトウェア(ドキュメント No. R20AN0038JJ)

— 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/usb/】

1.3 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
BOT	:	Mass storage class Bulk Only Transport
cstd	:	Prefix of function and file for Peripheral & Host Common Basic (USB low level) F/W
FSL	:	FAT File System Library
HCD	:	Host Control Driver of USB-BASIC-F/W
HDCD	:	Host Device Class Driver (device driver and USB class driver)
HEW	:	High-performance Embedded Workshop
hstd	:	Function & File for Host Basic (USB low level) F/W
HMSCD	:	Host Mass Storage Class Driver
HMSDD	:	Host Mass Storage Device Driver
MGR	:	Peripheral device state manager of HCD
PP	:	プリプロセス定義
Scheduler	:	タスク動作を簡易的にスケジューリングするもの
Scheduler Macro	:	スケジューラ機能を呼び出すために使用されるもの
SW1/SW2/SW3	:	評価ボードに実装されたユーザースイッチ
Task	:	処理の単位
TFAT	:	Tiny FAT file system software for microcontrollers (M3S-TFAT-Tiny-RX)
USB-BASIC-F/W	:	USB Basic Mini Firmware for Renesas USB MCU
USB	:	Universal Serial Bus

2. デバイスクラスドライバの登録

作成したデバイスクラスドライバは、USB ドライバに登録することでデバイスクラスドライバとして機能します。

r_usb_hmsc_apl.c 内の関数 usb_hapl_registration()を参考に、USB ドライバに登録してください。

詳細は、Renesas USB MCU USB Basic Mini Firmware アプリケーションノートの「ホストクラスドライバの登録方法」の章を参照してください。

3. ソフトウェア構成

3.1 モジュール構成

HDCD(ホストデバイスクラスドライバ)は HMSDD（ホストマスストレージデバイスドライバ）と HMSCD（USB ホストマスストレージクラスドライバ）の総称です。

Figure 3-1に HDCD のモジュール構成、Table 3-1にモジュール機能概要を示します。

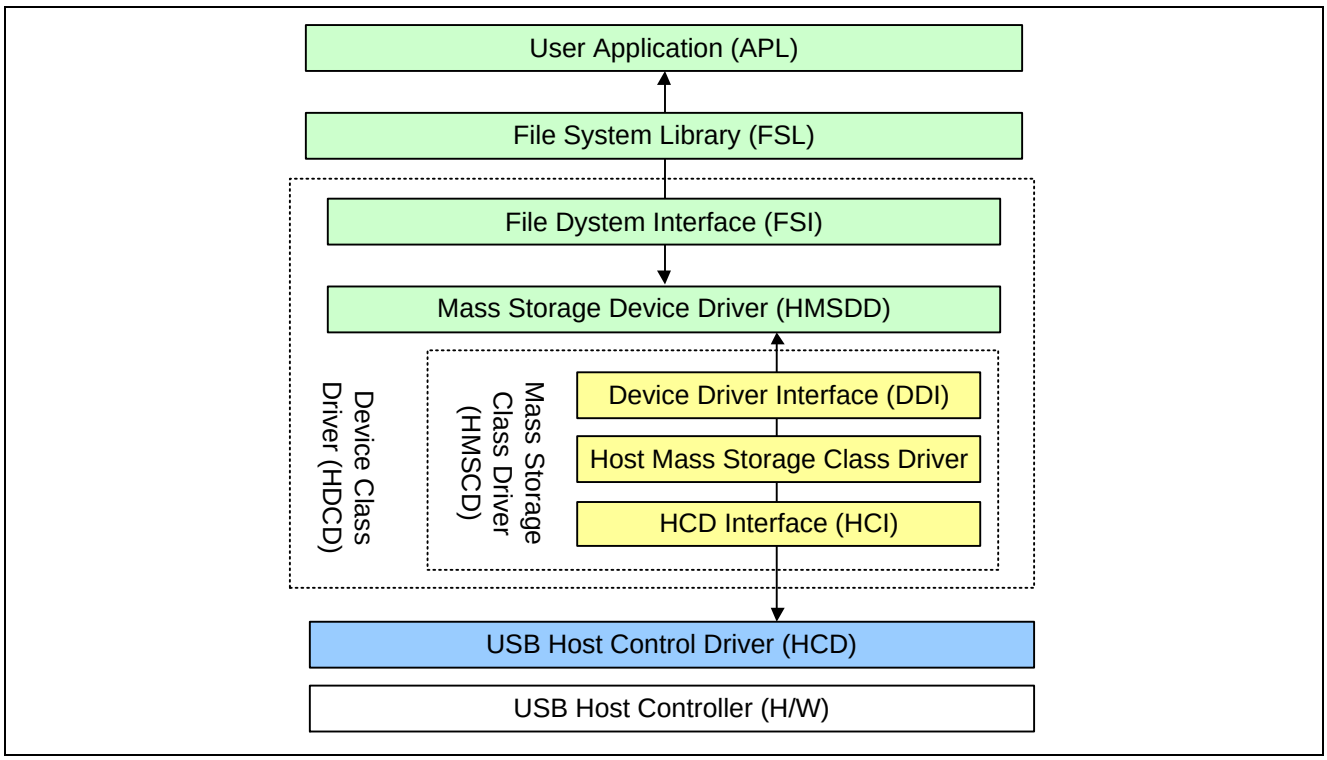

Figure 3-1 ソフトウェア構成図

Table 3-1 モジュール説明

モジュール名	説明
APL	FSL 関数を呼び出してストレージ機能を実現します。 システム仕様にあわせてユーザが作成します。
FSL	ユーザ仕様の FAT ファイルシステムです。
FSI	FSL－HMSDD 間のインタフェース関数です。 FSL にあわせて改変してください。
HMSDD	HMSDD です。 システム仕様にあわせてユーザが作成（改変）してください。
DDI	HMSDD－HMSCD 間のインタフェース関数です。 HMSDD に合わせて改変してください。
HMSCD	HMSCD です。ストレージコマンドに BOT プロトコルを付加して HCD へ要求します。また、BOT のシーケンスを管理します。 システム仕様にあわせてストレージコマンドを追加（改変）してください。
HCI	HMSCD－HCD 間のインタフェース関数です。
HCD	USB Host H/W 制御ドライバです。

3.2 ファイル構成

以下に HDCD が提供するファイルのフォルダ構成を示します。

各デバイス、評価ボードに依存するソースコードは各 H/W リソースフォルダ（HwResource）に格納しています。

[R8C 共通]
＋(Project file etc)
＋ src
(HCDC Sample Code)
＋―――HMSC **[*Mass Storage Class driver and FAT*]**
｜ ＋―――src ホスト MSC ドライバ
｜ ＋―――inc ホスト MSC ヘッダファイル
｜ ＋―――TFAT FAT File System ライブラリ
(Sample Code) **[*Class driver and sample application*]**
＋―――SmplMain
｜ ＋――― APL サンプルアプリケーション
(ハードウェア設定) **[*MCU 初期化等のハードウェアアクセス層*]**
＋――― HwResource H/W リソース
(USB-BASIC-F/W code) [*全ての **USB** ドライバに共通な基本ファームウェア*]
＋―――USBSTDFW
＋――― inc USB ドライバ共通ヘッダファイル
＋――― src USB ドライバ

3.2.1 ファイル一覧

Table 3-2にHDCDが提供するファイル構成を示します。

Table 3-2 ファイル構成

フォルダ名	ファイル名	概要
APL	r_usb_hmsc_apl.c	サンプルアプリケーションプログラム
HMSC	r_usb_hmsc_api.c	HMSC 用 API 関数
HMSC	r_usb_hmsc_ddi.c	HMSDD 用インタフェース関数（DDI）
HMSC	r_usb_hmsc_driver.c	USB クラスドライバ（Host Mass Storage Class）
HMSC	r_usb_hmsc_fsi.c	FSL インタフェース関数
HMSC	r_usb_hmsc_hci.c	HCD 用インタフェース関数（HCI）
HMSC	r_usb_hstorage_driver.c	デバイスドライバ（HMSDD）
TFAT	r_tfat_drv_if.c	TFAT 用インタフェース関数
SmplMain	r_usb_hmsc_apl.c	サンプルアプリケーション

3.3 システム資源

3.3.1 システムリソース定義

Table 3-3～Table 3-5に、HMSC が使用している資源を示します。

Table 3-3 タスク情報

関数名	タスク ID	優先度	概要
usb_hstd_HcdTask	USB_HCD_TSK	USB_PRI_1	HCD タスク
usb_hstd_MgrTask	USB_MGR_TSK	USB_PRI_2	MGR タスク
usb_hhub_Task	USB_HUB_TSK	USB_PRI_3	HUBCD タスク
usb_hmsc_Task	USB_HMSC_TSK	USB_PRI_3	マスストレージクラスタスク
usb_hmsc_StrgDriveTask	USB_HSTRG_TSK	USB_PRI_3	HMSDD タスク
usb_hmsc_SampleAplTask	USB_HMSCSMP_TSK	USB_PRI_4	APL タスク

Table 3-4 メールボックス情報

メールボックス名	使用タスク ID	待ちタスクキュー	概要
USB_HCD_MBX	USB_HCD_TSK	FIFO 順	HCD 用メールボックス
USB_MGR_MBX	USB_MGR_TSK	FIFO 順	MGR 用メールボックス
USB_HUB_MBX	USB_HUB_TSK	FIFO 順	HUBCD 用メールボックス
USB_ANSI_MBX	－	FIFO 順	ANSI 用メールボックス
USB_HMSC_MBX	USB_HMSC_TSK	FIFO 順	HMSCD 用メールボックス
USB_HSTRG_MBX	USB_HSTRG_TSK	FIFO 順	HMSDD 用メールボックス
USB_HMSCSMP_MBX	USB_HMSCSMP_TSK	FIFO 順	APL 用メールボックス

Table 3-5 メモリプール情報

メモリプール名	待ちタスクキュー	メモリブロック（注）	概要
USB_HCD_MPL	FIFO 順	12byte	HCD 用固定長メモリプール
USB_MGR_MPL	FIFO 順	12byte	MGR 用固定長メモリプール
USB_HUB_MPL	FIFO 順	12byte	HUBCD 用固定長メモリプール
USB_HMSC_MPL	FIFO 順	12byte	HMSC 用固定長メモリプール
USB_HSTRG_MBX	FIFO 順	12byte	HDCD 用固定長メモリプール
USB_HMSCSMP_MPL	FIFO 順	12byte	APL 用固定長メモリプール

（注）全システムのメモリブロックの最大数は、USB_BLKMAX で定義されます。初期値は 5 です。

3.4 USB ストレージ機器へのアクセス

FSI はセクタ番号を論理ブロックアドレスにセクタ数を転送データサイズに変換します。HMSDD はドライブ番号から USB ストレージ機器へのアクセスであるかを判断します。HMSCD はドライブ番号をユニット番号に変換し、BOT プロトコルに従い、HCD を経由して USB ストレージ機器にアクセスします。

Figure3-2に UBS ストレージ機器へのアクセスシーケンスを示します。

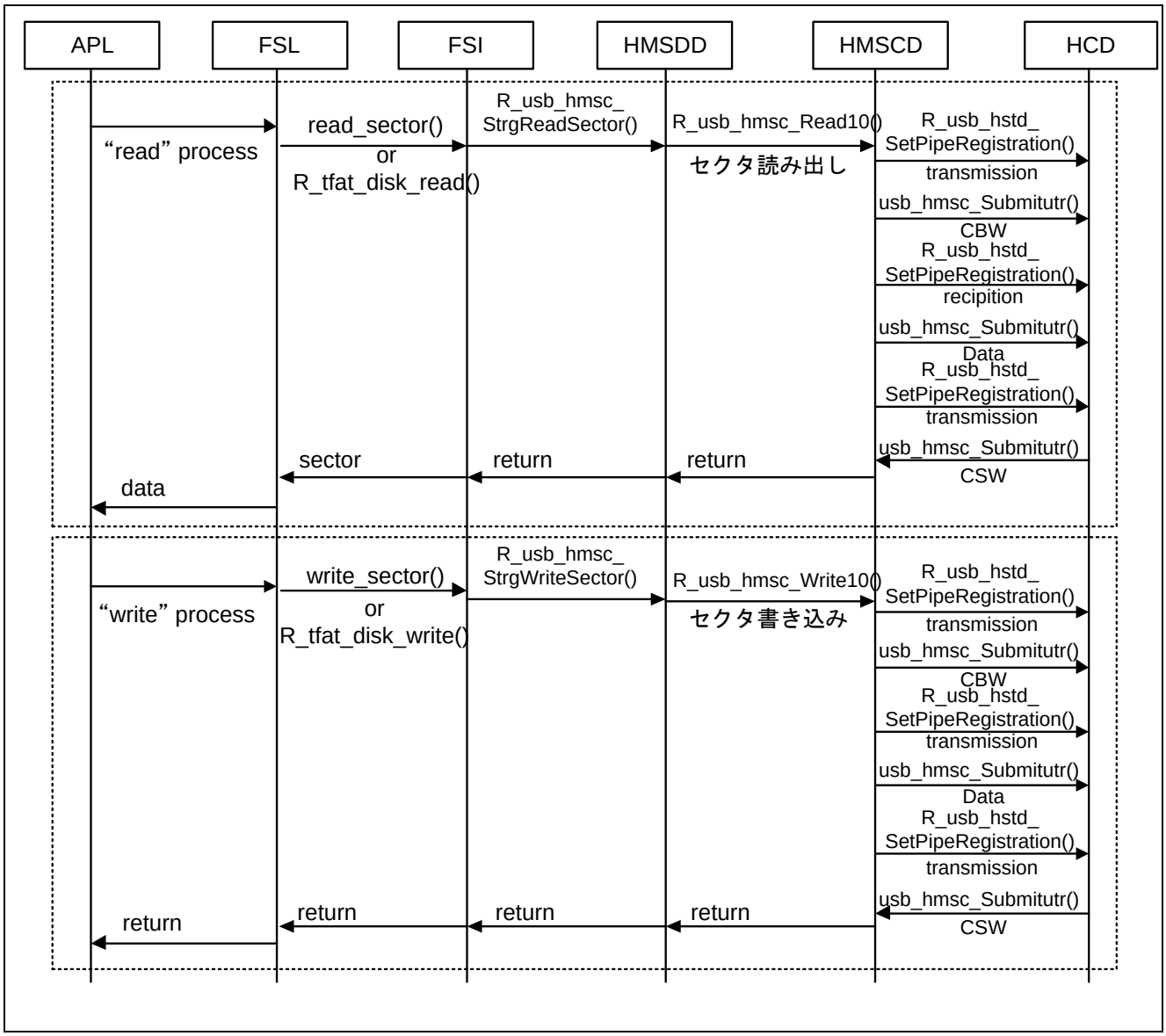

Figure3-2 USB ストレージ機器へのアクセスシーケンス

4. サンプルアプリケーションプログラム（APL)

HMSC サンプルアプリケーションプログラム(APL)の説明を以下に示します。

4.1 アプリケーションプログラム機能概要

APL は、評価ボードに実装されたスイッチ入力をトリガに動作します。

APL の主な機能を以下に示します。

- USB デバイスとエニュメレーション完了後、スイッチ(SW")押下により 512byte の'a'データを'hmscdemo.txt' ファイルへ一回書き込む。
- 再度スイッチが押下されるまで、"hmscdemo.txt"ファイルの読み出しを行う。
- 接続デバイスに対してアクセスエラーが発生した場合は、デバイスをサスペンド状態にする。

スイッチ入力の仕様をTable 4-1に示します。

Table 4-1 スイッチ入力による APL 動作内容

スイッチ名称	動作内容	スイッチ(SW)番号
ファイル操作スイッチ	スイッチが押されると、Write 処理を 1 回実施。 その後、Read 処理ループ。 以降、スイッチが押されるたび Read 処理ループの開始/停止を繰り返す。	SW2

4.2 動作環境について

本 APL の動作環境例をFigure4-1に示します。

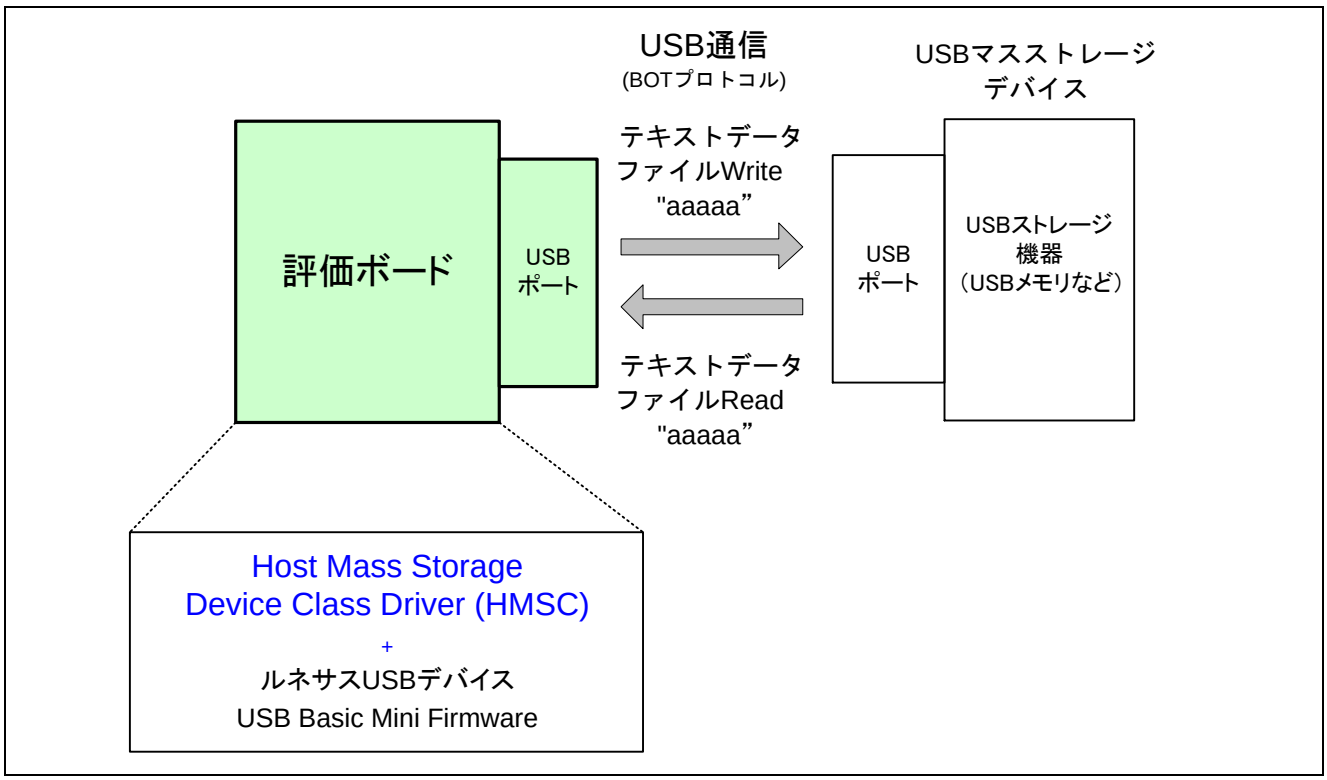

Figure4-1 動作環境例

4.3 APL 関数

Table 4-2に、APL 関数一覧を示します。

Table 4-2 APL 関数一覧

関数名	説明
usb_cstd_task_start()	タスクスタート処理
usb_apl_task_switch()	タスク切り替え処理
usb_hapl_task_start()	APL タスクスタート処理
usb_hapl_registration()	HMSC ドライバ登録
usb_hmsc_PrAplTitle	S/W タイトル表示処理
usb_hmsc_AplClear()	APL クリア処理
usb_shmsc_DemoStateChange()	デモ状態変更処理
usb_hmsc_SampleAplTask()	APL メインタスク
usb_hmsc_SmplMessageSend()	APL タスクへのメッセージ送信処理
usb_hmsc_SampleAplSpecifiedPath()	APL プロセス選択

4.4 APL 処理概略

4.4.1 APL 処理概略フロー

Figure4-2に、APL 処理概略フローを示します。

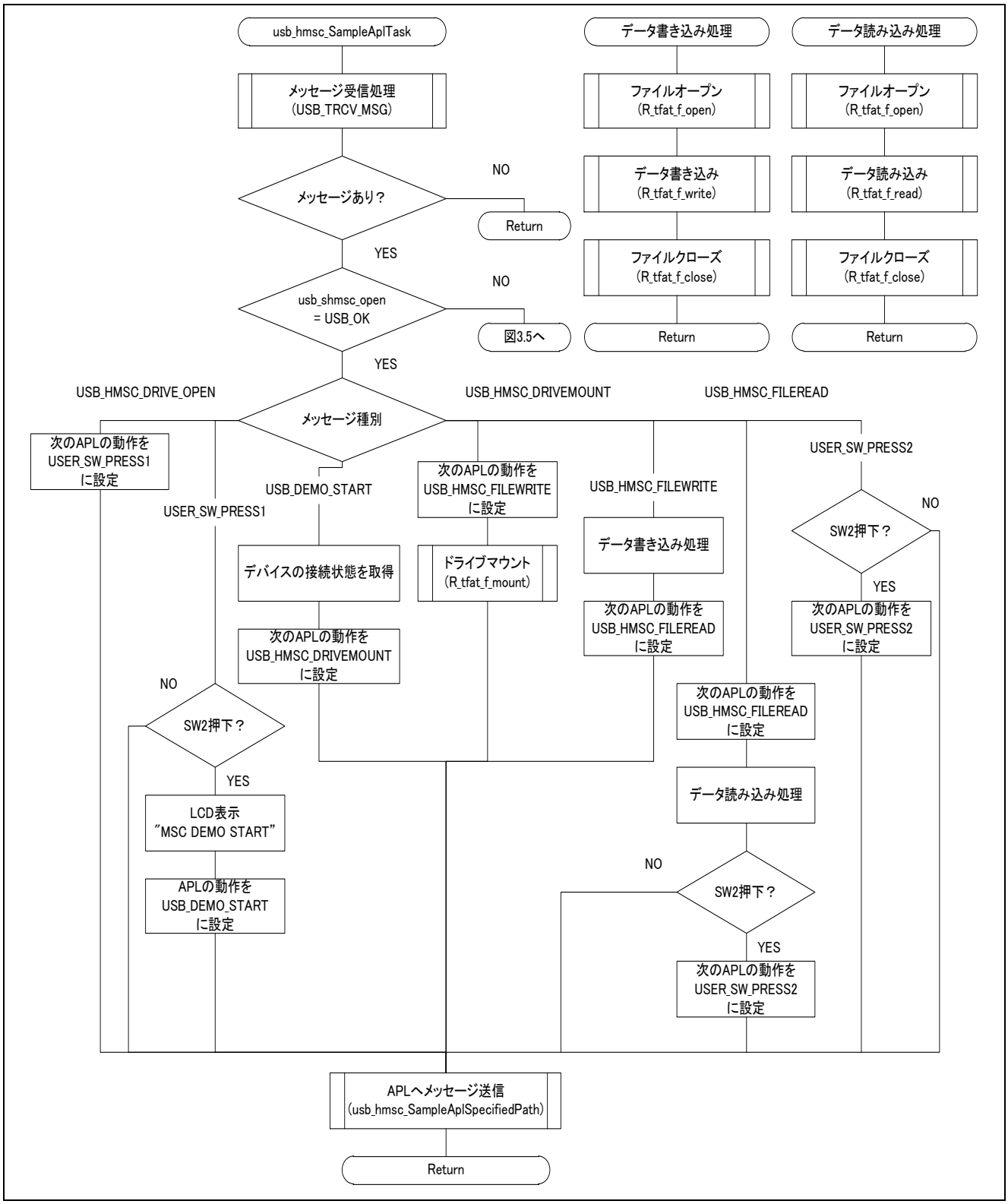

Figure4-2 APL 処理概略フロー

5. ファイルシステムインタフェース（FSI）

5.1 機能と特長

FSI (File System Interface)では、セクタ番号を論理ブロックアドレスへ変換し、セクタ数を転送データサイズへ変換を行いますので、FSL のインタフェース仕様にあわせて FSI を作成してください。

5.2 FSI API 関数

Table 5-1にサンプルの FSI API 関数を示します。

Table 5-1 FSI 関数

関数名		説明
TFAT	その他ファイルシステム	
R_tfat_disk_read	read_sector	データを読み出す
R_tfat_disk_write	write_sector	データを書き込む

[Note]

R_tfat_disk_read/R_tfat_disk_write 関数では、データのリード/ライト完了待ちを処理するため R_usb_hmsc_WaitLoop 関数をコールしています。

6. ホストマスストレージデバイスドライバ（HMSDD）

HMSDD は HMSCD を使用する場合にアプリケーションが起動します。HMSDD はドライブ番号からストレージ機器を選択します。

6.1 機能と特長

USB ストレージ機器のドライブ開始番号は r_usb_hmsc_define.h ファイルの USB_DRIVE でユーザ指定してください。ストレージコマンドを発行するために、以下の共通チェックを実施します。

a) ドライブ番号がトータルドライブ数より少ない

b) ドライブ番号が USB_DRIVE 以上

c) ストレージ機器が USB 接続されている

d) ドライブ番号からユニット番号を検索できる

HMSDD には以下の制限があります。

- 最大 8 台(TFAT 使用の場合は最大 1 台)の USB ストレージ機器が接続可能（HMSCD の登録回数まで）です。
- 最大 4 ユニット(TFAT 使用の場合は最大 1 ユニット)の USB ストレージ機器が接続可能です。
- 最大 10 パーティション(TFAT 使用の場合は最大 1 パーティション)の USB ストレージ機器が接続可能です。（最大ユニット数、最大パーティションはヘッダファイルで変更可能です）
- GetMaxUnit リクエストに応答しないデバイスは 1 ユニットデバイスとして動作します。
- デバイスタイプがダイレクトアクセスデバイス（INQUIRY コマンドで確認）のみ対応しています。
- セクタサイズが 512 バイトの USB ストレージ機器が接続可能です。
- READ_CAPACITY コマンドに応答しないデバイスはセクタサイズを 512 バイトとして動作します。
- ブートレコーダのパーティションタイプは以下の値をもとに判定しています。
  0x05/0x0F: 拡張パーティション
  0x01: マスタブートレコーダで FAT12
  0x04/0x06/0x0E: マスタブートレコーダで FAT16
  0x0B/0x0C: マスタブートレコーダで FAT32
  上記以外: パーティションブートレコーダとみなし JMP コード等を判定
- パーティション情報は、以下の検索方法により取得します。
  PBR: PBR を見つけるまでのパーティションエントリ上の開始セクタ番号の合計＋MBR のパーティションエントリ上の開始セクタ番号
  MBR: MBR を見つけるまでのパーティションエントリ上の開始セクタ番号の合計
- 接続できない（ストレージ機器として認識できない）場合があります。

6.2 論理ユニット番号 (LUN)

デバイスが GetMaxUnit リクエストに応答せず、ユニット数が未定の場合はユニット数=0 として動作します。HMSCD は、BOT 仕様で USB パケットを作成します。データパケットの CBWCB フィールド（ストレージコマンド）は SCSI 仕様に従い作成します。このとき、CBW パケットの bCBWLUN フィールドとストレージコマンド内の LUN フィールドの設定をTable 6-1に示します。

Table 6-1 LUN フィールド設定

	bCBWLUN フィールド	コマンド内の LUN フィールド
usb_ghmsc_MaxLUN=0 の場合	0	0
usb_ghmsc_MaxLUN=0 でない場合	ユニット番号	0

6.3 論理ブロックアドレス変換

BOT 仕様では論理ブロックアドレスにしたがって情報の読み出し、書き込みを行います。また情報の読み出し、書き込みはバイト数でデータサイズを指定します。

論理ブロックアドレスはセクタ番号とオフセットセクタ番号から計算します。また、転送サイズはセクタ数とセクタサイズから計算します。

論理ブロックアドレス = 論理セクタ番号 + オフセットセクタ番号

転送サイズ = セクタ数 * セクタサイズ

6.4 HMSDD API 関数

Table 6-2に HMSDD API の関数を示します。

Table 6-2 HMSDD 関数

関数名	説明
R_usb_hmsc_StrgDriveOpen()	ドライブをマウントする
R_usb_hmsc_StrgDriveClose()	ドライブを開放する
R_usb_hmsc_StrgReadSector()	データを読み出す
R_usb_hmsc_StrgWriteSector()	データを書き込む
R_usb_hmsc_StrgDriveSearch()	パーティション情報を読み込み、ドライブを検索する
R_usb_hmsc_StrgUserCommand()	ストレージコマンドを発行する

R_usb_hmsc_StrgDriveOpen

ドライブのマウントを行います

形式

usb_er_t R_usb_hmsc_StrgDriveOpen(uint16_t side)

引数

side ドライブ番号

戻り値

USB_DONE 正常終了

USB_ERROR エラー終了

解説

引数で指定されたドライブをマウントします。（セクタ情報を初期化します。）

この関数は以下の場合にエラーを返します。

1. パーティション番号を検索できない場合
2. ストレージ機器からドライブ情報が正しく読み込めなかった場合

補足

1. 本関数はクラスドライバまたは FAT ライブラリ I/F 関数から呼び出してください。

使用例

```
int dev_open(USB_UTR_t *ptr, int side)
{
            :
    /* Mounts Drive */
    error = R_usb_hmsc_StrgDriveOpen((uint16_t)side);

    if( error == USB_ERROR )
    {
            :
       return (-1);
    }
    return (0);
}
```

R_usb_hmsc_StrgDriveClose

ドライブを解放します

形式

usb_er_t R_usb_hmsc_StrgDriveClose(uint16_t side)

引数

side	ドライブ番号

戻り値

USB_DONE	正常終了
USB_ERROR	エラー終了

解説

引数で指定されたドライブを解放します。（セクタ情報をクリアします。）

この関数は以下の場合にエラーを返します。

1. ストレージ機器からドライブ情報が正しく読み込めなかった場合

補足

1. 本関数はクラスドライバまたは FAT ライブラリ I/F 関数から呼び出してください。

使用例

```
int dev_close(int side)
{
            :
    /* close function */
    error = R_usb_hmsc_StrgDriveClose((uint16_t)side);

    /* Condition compilation by the difference of the operating system */
    if( error == USB_ERROR )
    {
        :
        return (-1);
    }
    return (0);
}
```

R_usb_hmsc_StrgReadSector

引数で指定されたドライブのセクタ情報を読み出します

形式

usb_er_t R_usb_hmsc_StrgReadSector(uint16_t side, uint8_t *buff, uint32_t secno, uint16_t seccnt, uint32_t trans_byte)

引数

side	ドライブ番号
*buff	読み出しデータ格納領域
secno	セクタ番号
seccnt	セクタ数
trans_byte	転送データ長

戻り値

USB_DONE	正常終了
USB_ERROR	エラー終了

解説

引数で指定されたドライブのセクタ情報を読み出します

この関数は以下の場合にエラーを返します。

1. ストレージ機器からドライブ情報が正しく読み込めなかった場合

補足

1. 本関数は FAT ライブラリ I/F 関数から呼び出してください。

使用例

```
int read_sector(USB_UTR_t *ptr, int side, unsigned char *buff, unsigned long secno,
              long seccnt)
{
              :
    error = R_usb_hmsc_StrgReadSector((uint16_t)side,buff,secno,(uint16_t)seccnt,
              trans_byte);

    if( error == USB_ERROR )
    {
              :
        return (-1);
    }
    return (0);
}
```

R_usb_hmsc_StrgWriteSector

引数で指定されたドライブへセクタ情報を書き込みます

形式

USB_ER_t R_usb_hmsc_StrgWriteSector(uint16_t side, uint8_t *buff, uint32_t secno,
uint16_t seccnt, uint32_t trans_byte)

引数

side	ドライブ番号
*buff	書き込みデータ格納領域
secno	セクタ番号
seccnt	セクタ数
trans_byte	転送データ長

戻り値

USB_DONE	正常終了
USB_ERROR	エラー終了

解説

引数で指定されたドライブのセクタ情報を書き込みます

この関数は以下の場合にエラーを返します。

1. ストレージ機器からドライブ情報が正しく読み込めなかった場合

補足

1. 本関数は FAT ライブラリ I/F 関数からから呼び出してください。

使用例

```
int write_sector(USB_UTR_t *ptr, int side, unsigned char *buff, unsigned long secno,
                 long seccnt)
{
                :
    error = R_usb_hmsc_StrgWriteSector((uint16_t)side,buff,secno,(uint16_t)seccnt,
                 trans_byte);

    if( error == USB_ERROR )
    {
                :
        return (-1);
    }
    return (0);
}
```

R_usb_hmsc_StrgDriveSeach

パーティション情報を読み込み、ドライブを検索します

形式

uint16_t R_usb_hmsc_StrgDriveSearch(uint16_t addr)

引数

addr デバイスアドレス

戻り値

USB_DONE ドライブ検出

USB_ERROR ドライブ未検出

解説

ストレージ機器のユニット数を確認し、ストレージ機器の各ユニットにINQUIRYコマンドの発行を行い、機器がアクセス可能か確認します。

アクセス可能な全ユニットに対して論理ブロックアドレス=0x00 の領域を読み込み、ドライブ数をカウントします。

補足

1. 本関数はアクセス可能なドライブ数を探すだけでドライブのマウントは行いません。

使用例

```
void usb_hmsc_DriveOpen( uint16_t addr, uint16_t data2)
{
    uin16_t err;
                :
    err = R_usb_hmsc_StrgDriveSearch( addr );
                :
}
```

R_usb_hmsc_StrgUserCommand

ストレージコマンドを発行

形式

uint16_t R_usb_hmsc_StrgUserCommand(uint16_t side, uint16_t command , uint8_t *buff)

引数

*ptr	USB 通信構造体
side	ドライブ番号
command	発行するコマンド
*buff	データポインタ

戻り値

USB_DONE	正常終了
USB_ERROR	エラー終了

解説

引数で指定されたドライブへストレージコマンドを発行します。引数にドライブ番号、コマンド、データポインタを指定することで、対応した API 関数が処理を行います。

以下に、R_usb_hmsc_StrgUserCommand が対応するストレージコマンドを示します。

ストレージコマンド	対応関数	概要
TEST_UNIT_READY	R_usb_hmsc_TestUnit	ペリフェラル機器の状態確認
REQUEST_SENCE	R_usb_hmsc_RequestSense	ペリフェラル機器の状態取得
INQUIRY	R_usb_hmsc_Inquiry	論理ユニットのパラメータ情報取得
MODE_SELECT6	R_usb_hmsc_ModeSelect6	パラメータ指定
PREVENT_ALLOW	R_usb_hmsc_PreventAllow	メディアの取り出し許可/禁止
READ_FORMAT_CAPACITY	R_usb_hmsc_ReadFormatCapacity	フォーマット可能な容量取得
READ_CAPACITY	R_usb_hmsc_ReadCapacity	論理ユニットの容量情報取得
MODE_SENSE10	R_usb_hmsc_ModeSense10	論理ユニットのパラメータ取得

補足

1. ユーザアプリケーションプログラムまたはクラスドライバで呼び出してください。

使用例

```
void  usb_smp_task(void)
{
    :
  /* TEST_UNIT_READY 発行 */
  R_usb_hmsc_StrgUserCommand(ptr, side, USB_ATAPI_TEST_UNIT_READY, buf);
    :
}
```

6.5 ドライブ管理

HMSDD は、接続されたストレージ機器の初期ドライブ番号として USB_DRIVE を設定します。HMSDD はマルチユニット、複数ドライブのストレージ機器も接続が可能です。アプリケーションからドライブ番号を指定するだけで、任意のユニットの該当パーティションブロックに対するアクセスができます。ストレージ機器は最大 4 ユニット、各ユニットで最大 10 パーティションブロックまで対応しています。ユニット数もパーティションブロック数もヘッダファイルによる指定で変更が可能です。

例えば、USB_DRIVE=1 とした場合に 3 ユニットの USB ストレージ機器で、ユニット 0 が 3 パーティション（拡張含めてパーティションが合計 3 つ）、ユニット 1 がメディアなし、ユニット 2 が 2 パーティション構成だった場合はTable 6-3、Figure6-1のようになります。

Table 6-3 ドライブ番号

ユニット	パーティション	ドライブ	備考
0	0	1（B）	FAT32
0	1		種別なし（ドライブなし）
0	2		拡張パーティション（継続情報読み出し）
0	2-0	2（C）	FAT32（パーティション 2 の継続情報）
0	2-0		種別なし（ドライブなし）
0	3	3（D）	FAT32
1			メディアなし
2	0	4（E）	FAT16
2	1	5（F）	FAT16

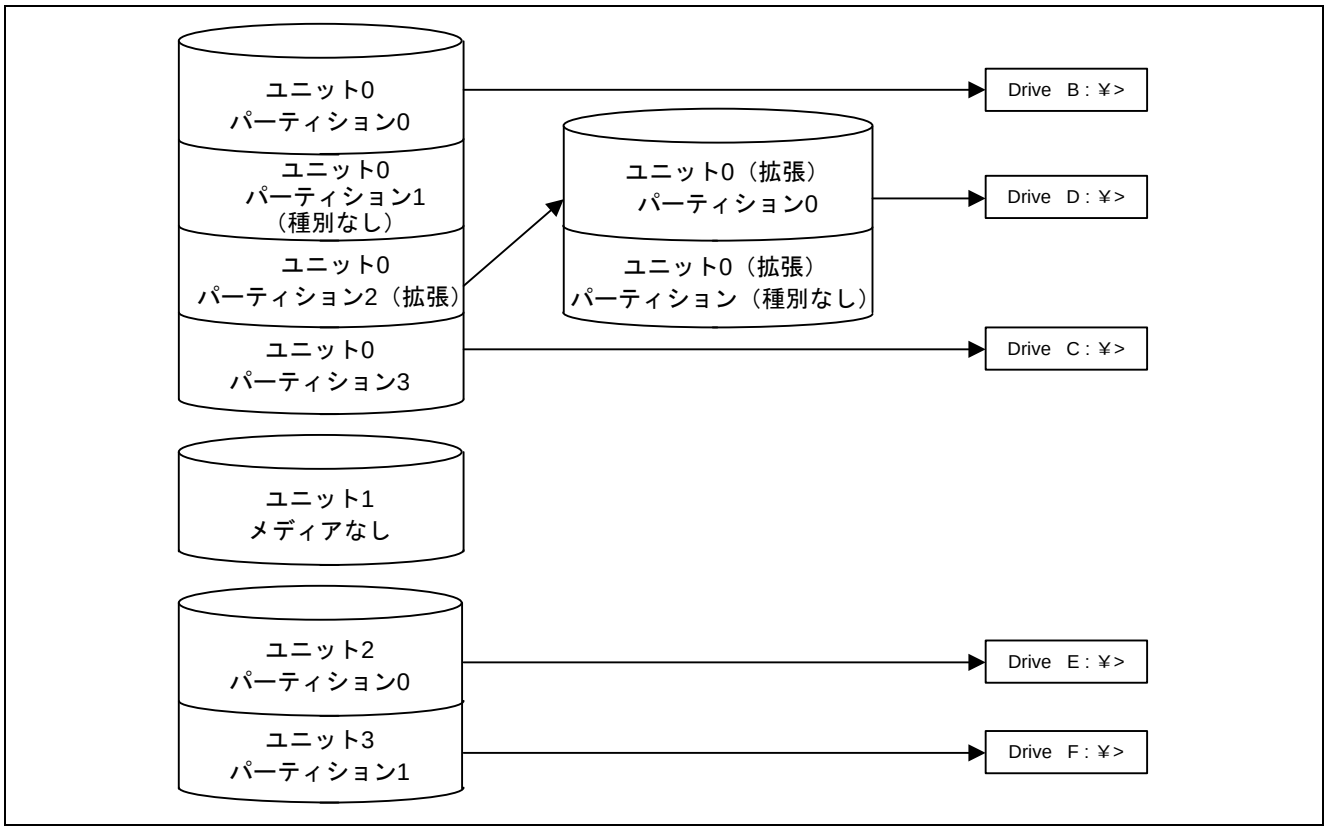

Figure6-1 ドライブ管理

7. ホストマスストレージクラスドライバ（HMSCD）

7.1 機能と特徴

HMSCD は、接続された USB ストレージ機器の動作可否判定（デバイス照合）及び BOT プロトコルによるストレージコマンド通信を行います。

HMSCD は、HMSDD に対するインターフェース(DDI 関数)、HCD に対するインターフェース（HCI 関数）、HMSCD 本体の 3 層構造です。HMSCD は、USB ストレージ機器のアクセスに必須なストレージコマンドとサンプルのストレージコマンドに対応しています。

HMSCD の特徴を以下に示します。

- USB マスストレージクラスの BOT に対応しています。
- USB マスストレージサブクラスの SFF-8070i（ATAPI）に対応しています。
- データ転送に使用するパイプを、IN/OUT 転送で共有しています。また、複数のデバイスを接続した場合も 1 本のパイプを共有します。

7.2 HMSCD に対する要求発行

USB ストレージ機器へアクセスする場合は、後述のインタフェース関数を用います。

HMSCD は、上位層からの要求に対してコールバック関数の戻り値で結果を通知します。

7.3 HMSCD 構造体

Table 7-1、Table 7-2に HMSCD が使用する構造体構成を示します。

Table 7-1 USB_MSC_CBW_t 構造体

型	メンバ名	説明	備考
uint32_t	dCBWSignature	CBW Signature	0x55534243: USBC
uint32_t	dCBWTag	CBW Tag	CSW に対応する Tag
uint8_t	dCBWDTL_Lo	CBW DataTransfer Length	送受信するデータのデータ長
uint8_t	dCBWDTL_ML		
uint8_t	dCBWDTL_MH		
uint8_t	dCBWDTL_Hi		
uint8_t	bmCBWFlags	CBW Direction	データ送受信方向
uint8_t	bCBWLUN	Logical Unit Number	ユニット番号
uint8_t	bCBWCBLength	CBWCB Length	コマンド長
uint8_t	CBWCB[16+33]	CBWCB	コマンドブロック

Table 7-2 USB_MSC_CSW_t 構造体

型	メンバ名	説明	備考
uint32_t	dCSWSignature	CSW Signature	0x55534253: USBS
uint32_t	dCSWTag	CSW Tag	CBW に対応する Tag
uint8_t	dCSWDataResidue_Lo	CSW DataResidue	使用されたデータ長
uint8_t	dCSWDataResidue_ML		
uint8_t	dCSWDataResidue_MH		
uint8_t	dCSWDataResidue_Hi		
uint8_t	bCSWStatus	CSW Status	コマンドステータス
uint8_t	dummy[51]	dummy	偶数調整

7.4 USB ホストコントロールドライバインタフェース(HCI)

HCI は、HMSCD、HCD 間のインターフェース関数です。

RTOS の場合、HCI はドライブがマウントされるまで HCD のクラス領域を使用してメッセージを送受信します。ドライブのマウントが終了している場合は HMSCD 専用領域を使用します。

non-OS の場合、HMSCD の領域を使用してメッセージを送受信します。

なお、複数のデバイスに対し、同時にエニュメレーション、アクセスを行うことはできません。

7.5 デバイスドライバインタフェース(DDI)

DDI は、HMSDD、HMSCD 間のインターフェース関数です。

DDI 関数は HMSCD 起動関数、終了関数、接続デバイスのチェック関数及び、サンプルのストレージコマンド関数で構成されます。

7.6 HMSC API 関数

Table 7-3～Table 7-5に HMSCD の API 関数を示します。

Table 7-3 HMSCD 関数

関数名	説明
R_usb_hmsc_SetDevSts()	HMSCD 動作状態の設定を行う
R_usb_hmsc_GetDevSts()	HMSCD 動作状態の応答を行う
R_usb_hmsc_Information()	データ送受信で使用しているパイプ番号を取得する
R_usb_hmsc_Task()	ホストマスストレージクラスタスク

Table 7-4 HCI 関数

関数名	説明
R_usb_hmsc_GetMaxUnit()	Get_MaxLUN リクエストを発行する
R_usb_hmsc_MassStorageReset()	MassStorageReset リクエストを発行する
R_usb_hmsc_ClearStall()	USB ドライバに STALL 解除の要求を行う

Table 7-5 DDI 関数

関数名	説明
R_usb_hmsc_Initialized()	Host Mass Storage の初期化を行う
R_usb_hmsc_ClassCheck()	接続デバイスの確認を行う
R_usb_hmsc_Read10()	READ10 コマンドを発行する
R_usb_hmsc_Write10()	WRITE10 コマンドを発行する
R_usb_hmsc_DriveSpeed()	ドライブの通信速度を取得します
R_usb_hmsc_Release()	HMSCD 解放

R_usb_hmsc_SetDevSts

HMSCD 動作状態の設定を行う

形式

uint16_t R_usb_hmsc_SetDevSts(uint16_t data)

引数

data 動作状態

戻り値

－ USB_DONE

解説

HMSCD の動作状態を引数で指定された値に変更します。
動作状態を以下に示します。

動作状態	概要
USB_HMSC_DEV_DET	デタッチ状態
USB_HMSC_DEV_ATT	アタッチ状態（エニュメレーション完了）
USB_HMSC_DEV_ENU	エニュメレーション中

補足

ユーザアプリケーションプログラムまたはクラスドライバで本関数を呼び出してください。

戻り値は常に USB_DONE になります。

使用例

```
void  usb_smp_task(USB_UTR_t *ptr)
{
    :
  /* デバイス状態をアタッチ状態に変更 */
  R_usb_hmsc_SetDevSts((uint16_t)USB_HMSC_DEV_ATT);
    :
}
```

R_usb_hmsc_GetDevSts

HMSCD 動作状態の応答を行う

形式

uint16_t R_usb_hmsc_GetDevSts(void)

引数

—	—

戻り値

usb_ghmsc_AttSts	USB_HMSC_DEV_ATT（接続）
	USB_HMSC_DEV_DET（切断）

解説

HMSCD の動作状態を取得します。

補足

ユーザアプリケーションプログラムまたはクラスドライバで本関数を呼び出してください。

使用例

```
void  usb_smp_task(USB_UTR_t *ptr)
{
    :
  /* デバイス状態確認*/
  if(R_usb_hmsc_GetDevSts() == USB_HMSC_DEV_DET)
  {
    /* 切断処理 */
  }
    :
}
```

R_usb_hmsc_Information

データ送受信で使用しているパイプ番号を取得する

形式

uint16_t R_usb_hmsc_Information(uint16_t pipe_offset)

引数

pipe_offset	パイプ情報テーブル

戻り値

－	パイプ番号

解説

パイプ情報テーブルから、データ送信（受信）に使用しているパイプ番号を取得します。

補足

ユーザアプリケーションプログラムまたはクラスドライバで本関数を呼び出してください。

使用例

```
void  usb_smp_task(USB_UTR_t *ptr)
{
  uint16_t  pipeno;
    :
  /* パイプ番号取得*/
  pipeno = R_usb_hmsc_Information(usb_ghmsc_InPipe[msgnum][0]);
    :
}
```

R_usb_hmsc_Task

ホストマスストレージクラスタスク

形式

void	R_usb_hmsc_Task(void)

引数

―	―

戻り値

―	―

解説

HMSCD のタスクです。

BOT の制御を行います。

補足

スケジューラ処理を行うループ内で本関数を呼び出してください。

当該ループについては、USB-BASIC-F/W アプリケーションノートを参照してください。

使用例

```
void  usb_smp_mainloop(void)
{
  while(1)
  {
  /* スケジューラ起動 */
  R_usb_cstd_Scheduler();
  /* フラグチェック */
  if(USB_FLGSET == R_usb_cstd_CheckSchedule())
  {
       :
   usb_hmsc_Task();
       :
  }
    :
}
```

R_usb_hmsc_GetMaxUnit

Get_MaxLUN リクエストを発行する

形式

usb_er_t R_usb_hmsc_GetMaxUnit(uint16_t addr, USB_CB_t complete)

引数

addr	デバイスアドレス
complete	コールバック関数

戻り値

USB_E_OK	GET_MAX_LUN 発行
USB_E_ERROR	GET_MAX_LUN 未発行
USB_E_QOVR	GET_MAX_LUN 未発行

解説

GET_MAX_LUN リクエストを発行して、最大ユニット数を取得します。

補足

ユーザアプリケーションプログラムまたはクラスドライバで本関数を呼び出してください。

使用例

```
void  usb_smp_task(USB_UTR_t *ptr)
{
  USB_ER_t  err;
    :
  /* 最大ユニット数取得*/
  err = R_usb_hmsc_GetMaxUnit(mess, addr, (USB_CB_t)usb_hmsc_StrgCheckResult);
  if(err == USB_E_QOVR)
  {
    /* エラー処理 */
  }
    :
}
```

R_usb_hmsc_MassStorageReset

MassStorageReset リクエストを発行する

形式

usb_er_t R_usb_hmsc_MassStorageReset(uint16_t drvnum, USB_CB_t complete)

引数

drvnum	ドライブ番号
complete	コールバック関数

戻り値

USB_E_OK	MASS_STORAGE_RESET 発行
USB_E_ERROR	MASS_STORAGE_RESET 未発行
USB_E_QOVR	MASS_STORAGE_RESET 未発行

解説

MASS_STORAGE_RESET リクエストを発行して、プロトコルエラーを解除します。

補足

ユーザアプリケーションプログラムまたはクラスドライバで本関数を呼び出してください。

使用例

```
void  usb_smp_task(USB_UTR_t *ptr)
{
  USB_ER_t  err;
    :
  /* プロトコルエラー解除*/
  err = R_usb_hmsc_MassStorageReset(drvnum, (USB_CB_t)usb_hmsc_CheckResult);
  if(err == USB_E_QOVR)
  {
    /* エラー処理 */
  }
    :
}
```

R_usb_hmsc_ClearStall

STALL 解除の要求を行う

形式

void R_usb_hmsc_ClearStall(uint16_t type, uint16_t msgnum , USB_CB_t complete)

引数

type	通信方向
msgnum	ドライバ番号
complete	コールバック関数

戻り値

―	―

解説

パイプの STALL 状態を解除します。

補足

ユーザアプリケーションプログラムまたはクラスドライバで本関数を呼び出してください。

使用例

```
void  usb_smp_task(USB_UTR_t *ptr)
{
    :
/* デバイスの STALL 解除*/
R_usb_hmsc_ClearStall((uint16_t)USB_DATA_NONE, msgnum,
  (USB_CB_t)usb_hmsc_ClearStallCheck2);
    :
}
```

R_usb_hmsc_Initialized

Host Mass Storage の初期化を行う

形式

void	R_usb_hmsc_Initialized(uint16_t data1, uint16_t data2)

引数

data1	未使用
data2	未使用

戻り値

―	―

解説

Host Mass Storage の初期化処理を行います。

補足

本 S/W では、usb_hapl_registration()で本 API をコールバック関数として登録されています。

使用例

```
void  usb_hapl_registration(USB_UTR_t *ptr)
{
    :
  /* Driver init */
  driver.classinit = &R_usb_hmsc_Initialized;
    :
}
```

R_usb_hmsc_ClassCheck

接続デバイスの確認を行う

形式

void	R_usb_hmsc_ClassCheck(uint16_t **table)

引数

**table	未使用

戻り値

―	―

解説

デバイス数、ドライブ数のチェックと、インタフェースディスクリプタテーブルの解析を行います。
下記項目にて HMSCD とのマッチングを確認し、動作可能の場合はシリアル番号を読み出します。
Bulk エンドポイントディスクリプタテーブルでパイプ情報テーブルを更新（Endpoint アドレス、Max パケットサイズ等）します。
インタフェースディスクリプタの情報確認

bInterfaceSubClass = USBC_ATAPI / USBC_SCSI

bInterfaceProtocol = USBC_BOTP

bNumEndpoint > USBC_TOTALEP

ストリング Descriptor の情報確認

12 文字以上のシリアル番号（エラー時は警告表示）

エンドポイント Descriptor の情報確認

bmAtributes = 0x02（Bulk エンドポイントが必要）

bEndpointAdress（IN/OUT 双方のエンドポイントが必要）

補足

本 S/W では、usb_hapl_registration()で本 API をコールバック関数として登録されています。

接続可能なストレージデバイス数は USB_MAXSTRAGE で定義されています。(r_usb_hmsc_define.h 参照)

動作可能なドライブ数は USB_MAXDRIVE で定義されています。(r_usb_hmsc_define.h 参照)

使用例

```
void  usb_hapl_registration(USB_UTR_t *ptr)
{
    :
  /* Driver check */
  driver.classcheck   = &R_usb_hmsc_ClassCheck;
    :
}
```

R_usb_hmsc_Read10

READ10 コマンドを発行

形式

uint16_t R_usb_hmsc_Read10(uint16_t side, uint8_t *buff, uint32_t secno, uint16_t seccnt, uint32_t trans_byte)

引数

side	ドライブ番号
*buff	読み出しデータエリア
secno	セクタ番号
seccnt	セクタ数
trans_byte	転送データ長

戻り値

－	エラーコード。

解説

USB デバイスに READ10 コマンドを発行します。

コマンドエラーの場合は REQUEST_SENSE コマンドを発行しエラー情報を取得します。

補足

ユーザアプリケーションプログラムまたはクラスドライバで本関数を呼び出してください。

使用例

```
void  usb_smp_task(USB_UTR_t *ptr)
{
  uint32_t  result;
    :
  /* READ10 発行 */
  result = R_usb_hmsc_read10(side, buff, secno, seccnt, trans_byte);
  if(result != USB_HMSC_OK)
  {
    /* エラー処理 */
  }
    :
}
```

R_usb_hmsc_Write10

WRITE10 コマンドを発行する

形式

uint16_t	R_usb_hmsc_Write10(uint16_t side, uint8_t *buff, uint32_t secno, uint16_t seccnt, uint32_t trans_byte)

引数

side	ドライブ番号
*buff	書き込みデータエリア
secno	セクタ番号
seccnt	セクタ数
trans_byte	転送データ長

戻り値

－	エラーコード。

解説

USB デバイスに WRITE10 コマンドを発行します。

コマンドエラーの場合は REQUEST_SENSE コマンドを発行しエラー情報を取得します。

補足

ユーザアプリケーションプログラムで本関数を呼び出してください。

使用例

```
void  usb_smp_task(USB_UTR_t *ptr)
{
  uint32_t  result;
    :
  /* WRITE10 発行 */
  result = R_usb_hmsc_Write10(side, buff, secno, seccnt, trans_byte);
  if(result != USB_HMSC_OK)
  {
    /* エラー処理 */
  }
    :
}
```

R_usb_hmsc_DriveSpeed

ドライブの通信速度を取得

形式

uint16_t R_usb_hmsc_DriveSpeed(uint16_t side)

引数

side ドライブ番号

戻り値

－ 通信速度

解説

ドライブの通信速度を取得します。

補足

ユーザアプリケーションプログラムまたはクラスドライバで本関数を呼び出してください。

使用例

```
void  usb_smp_task(USB_UTR_t *ptr)
{
  uint16_t  result;
    :
  /* 通信速度取得 */
  result = R_usb_hmsc_DriveSpeed(side);
    :
}
```

R_usb_hmsc_Release

ホストマスストレージクラスドライバを解放

形式

void	R_usb_hmsc_Release(void)

引数

—	—

戻り値

—	—

解説

HCD から HMSCD 登録を解除します。

補足

ユーザアプリケーションプログラムまたはクラスドライバで本関数を呼び出してください。

使用例

```
void  usb_smp_task(USB_UTR_t *ptr)
{
  uint16_t  result;
    :
  /* HMSCD 登録解除*/
  R_usb_hmsc_Release();
    :
}
```

8. Target Peripheral List (TPL)

ホスト動作する場合に、デバイスクラス仕様は、すべての種類の USB ペリフェラルデバイスに対応する必要はありません。デバイスがどの周辺機器をサポートするのかは、それぞれのデバイスで異なります。サポートする周辺機器を記載したリストをターゲットペリフェラルリスト(TPL)と呼びます。

TPL は VID と PID で構成されます。VID(/PID)によるチェックを無効とするには、USB_NOVENDOR (/USB_NOPRODUCT) を指定してください。 TPL は *r_usb_hmsc_apl.c* ファイル内の配列 *usb_gapl_devicetpl[]* に記載してください。

9. 制限事項

HMSCD 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. GetMaxUnit リクエストに 3 秒以上応答がない（Control 転送が終了しない）場合は 1 ユニットデバイスとして処理を行います。
3. 複数のストレージ機器を接続するには HMSCD を接続可能な機器の数だけ登録する必要があります。
4. HMSDD にも制限事項があります。詳細は 7 章を参照ください。
5. 7 章の機能を満たすストレージ機器でも接続を保証するものではありません。
6. TFAT は 1 デバイスしか対応していません。詳細は TFAT のマニュアルを参照ください。

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
2.10	2013.8.1	—	First Release
2.12	2014.3.31	—	R8C に対応、誤記訂正
2.13	2015.3.16	—	動作確認デバイスから RX111 を削除。
2.14	2016.3.28	—	"USB Basic Mini Firmwarer(R01AN0326)"をリビジョンアップしたことによる当該ドライバのリビジョンアップ

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。

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：http://japan.renesas.com/contact/