Renesas USB MCU

USB Peripheral Mass Storage Class Driver (PMSC) using Basic Mini Firmware

R01AN0710JJ0215
Rev.2.15
Mar 28, 2016

要旨

本資料は、Renesas USB MCUを使用したUSB Peripheral Mass Storage Class Driver (PMSC)のアプリケーションノートです。

動作確認デバイス

RL78/G1C, RL78/L1C, R8C/3MU, R8C/34U, 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。

なお、本プログラムは Renesas Starter Kit 上で動作確認を行っています。

目次

1. はじめに 2
2. 動作確認済環境 4
3. ソフトウェア構成 4
4. ペリフェラル用 MSC サンプルアプリケーション(APL) 9
5. デバイスクラスドライバ（PDCD） 13
6. USB マスストレージクラスドライバ（PMSCD） 20
7. ペリフェラルマスストレージデバイスドライバ（PMSDD） 26
8. メディアドライバインタフェース 30
9. EEPROM メディアドライバ 32
10. スケジューラへの組み込みについて 35
11. 制限事項 36
12. e² studio 用プロジェクトのセットアップ 37
13. e² studio 用プロジェクトを CS+で使用する場合 39

1. はじめに

本資料は、Renesas USB MCUを使用したUSB Peripheral Mass Storage Class Driver (PMSC)および、ストレージデバイスのサンプルドライバに対する取扱説明書です。

1.1 機能と特長

USB Peripheral マスストレージクラスドライバ（以降 PMSC と記述）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、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。

1.2 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. Renesas USB MCU ユーザーズマニュアル ハードウェア編
5. Renesas USB MCU USB Basic Mini Firmware アプリケーションノート
6. メディアドライバ API マニュアル
   ルネサスエレクトロニクスのホームページより入手できます。

— 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

API	:	Application Program Interface
APL	:	Application program
BOT	:	USB mass storage class Bulk Only Transport 詳細は、USB Implementers Forum 発行の [Universal Serial Bus Mass Storage Class Bulk-Only Transport Revision 1.0]を参照下さい。
cstd	:	USB-BASIC-FWの Peripheral & Host共通関数のprefix
CS+	:	ルネサス統合開発環境
DDI	:	Device driver interfaceまたはPMSDD API.
H/W	:	Renesas USB MCU
PCD	:	Peripheral control driver of USB-BASIC-FW
PDCD	:	Peripheral device class driver (device driver and USB class 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 (sample ATAPI driver)
PP	:	Pre-processed definition
pstd	:	USB-BASIC-FWの Peripheral関数のprefix
RSK	:	Renesas Starter Kits
Scheduler Macro	:	Used to call a scheduler function
SW1/SW2/SW3	:	User switches on theRSK Borad
USB	:	Universal Serial Bus
USB-BASIC-FW	:	USB Basic Firmware mini for Renesas USB device
タスク	:	処理の単位
スケジューラ	:	スケジューラ機能を呼び出すために使用されるもの
データ転送	:	Control転送、Bulk転送、Interrupt転送の総称

1.4 本書の読み方

3.2章にソース一覧を掲載しています。MCU 固有ソースは、"/*devicename*/src/HwResource"にあります。アプリケーションに必要なファイルを確認してください。

4章はペリフェラル用 PMSC サンプルアプリケーションの説明をしています。

すべてのコードモジュールはタスクに分割されます。タスク間でメッセージの受け渡しが行われているとを予めご理解ください。関数(タスク)の実行順序はスケジューラが決定します。このため重要なタスクに優先権を持たせることができます。また、タスクに登録されたコールバックメカニズムを使用することで、各タスクは並列処理（ノンブロッキング）で動作します。タスクのメカニズムは1.2章の"Renesas USB MCU USB Basic Mini Firmware アプリケーションノート"で説明しています。PMSC のタスクについては3.3章を参照してください

2. 動作確認済環境

2.1 コンパイラ

動作確認を行ったコンパイラは以下の通りです。

a. CA78K0R コンパイラ V.1.71
b. CC-RL コンパイラ V.1.01
c. IAR C/C++ Compiler for RL78 version 2.10.4
d. KPIT GNURL78-ELF v15.02
e. C/C++ Compiler Package for M16C Series and R8C Family V.6.00 Release 00

2.2 評価ボード

動作確認を行った評価ボードは以下の通りです。

a. Renesas Starter Kit for RL78/G1C (型名: R0K5010JGC001BR)
b. Renesas Starter Kit for RL78/L1C (型名: R0K50110PC010BR)
c. R8C/34K Group USB Peripheral 評価ボード(型名: R0K5R8C34DK2PBR)

3. ソフトウェア構成

3.1 モジュール構成

PDCD は、Figure 3-1で示すように PMSCD と PMSDD の 2 層構造になっています。

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）、および PCD に対するインタフェース関数群（PCI）の 3 層構造から成り立っています。
RTOS の場合、PMSCD と PMSDD は、それぞれ uITRON 上でタスクとして動作します。
PMSCD 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。

また、PMSDD は Media Driver を介して Media のデータにアクセスします。

モジュール構成をFigure 3-1に示します。また、各モジュールの概要をTable 3-1に示します。

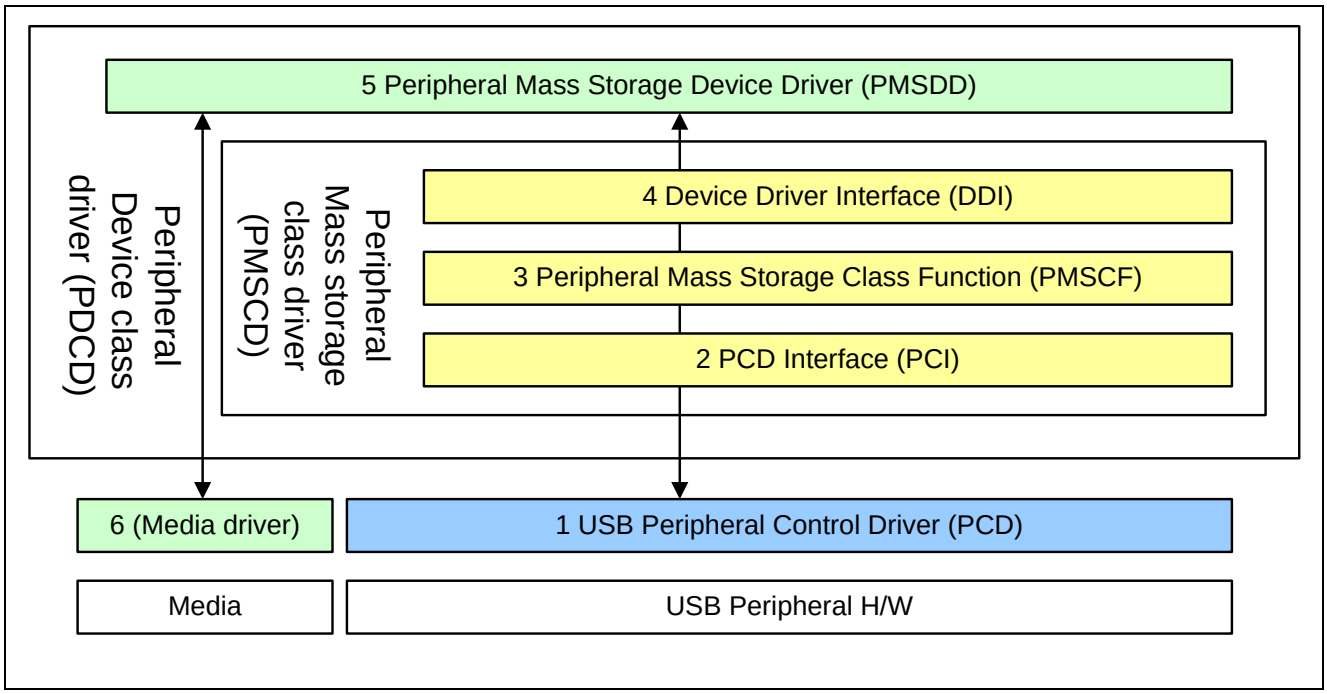

Figure 3-1 ソフトウェア構成図

3.1.1 PDCD

PDCD は、PMSDD と PMSCD で構成され、ホストからのクラスリクエストの処理やホストからのストレージコマンドの応答を行います。

Table 3-1には、PDCD、PCD および Media driver の概要が記載されています。Media driver は、置き換え可能なストレージドライバを意味します。

3.1.2 PMSCD

PMSCD は、BOT プロトコルやデータ送受信を処理する PMSCF、PMSDD とインタフェースをとる DDI、PCD とインタフェースをとるための PCI の 3 層で構成されます。これら 3 層の主な機能を以下に示します。

1. PMSCF
   (a). BOT プロトコル制御
   (b). CBW の解析、データ送受信、PMSDD/PCD と連携による CSW の生成
   (c). クラスリクエスト(MassStorageReset, GetMaxLUN)に対する応答
2. PCI
   (a). クラスリクエスト受信処理
   (b). STALL 状態のクリア処理や関連コールバック関数設定
   (c). PCD に対するデータ送受信要求および関連コールバック関数設定
3. DDI
   (a). マスストレージデバイスドライバ登録処理
   (b). ATAPI コマンド処理完了通知

Table 3-1　各モジュール機能概要

章番号	モジュール名	機能概要	格納フォルダ/関連ファイル	参照する章、ドキュメント
USB Basic FW	PCD	USB Peripheral H/W 制御ドライバです。	src/USBSTDFW	「ルネサス USB デバイス USB Basic Firmware アプリケーションノート」を参照してください。
5	PCI	PMSCF－PCD 間のインタフェース関数です。	src/MSCFW/PMSC/r_usb_pmsc_pci.c	
	PMSCF	PMSCD の本体部分です。BOT プロトコルデータの制御や、クラスリクエストの対応をします。また、PMSDD とのストレージコマンドやデータの受け渡しも行います。	src/MSCFW/PMSC/ r_usb_pmsc_request.c r_usb_pmsc_driver.c	
	DDI	PMSDD－PMSCF 間のインタフェース関数です。 ドライバ登録および ATAPI コマンド完了コールバック処理	src/MSCFW/PMSC/r_usb_pmsc_ddi.c	
6	PMSDD	マスストレージデバイスドライバです。PMSCD からのストレージコマンド処理や、Media driver を介して Media をアクセスします。 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。	src/MSCFW/MEDIA/r_usb_atapi_driver.c	
7	Block Media Driver	Block media ストレージ機器の制御ドライバです（サンプルでは PMSDD に含まれています）。	src/MSCFW/MEDIA/r_usb_atapi_memory.c	アプリケーションノート(R01AN1443JU_RX)を参照してください。

3.2 ファイル構成一覧

3.2.1 フォルダ構成

以下に、PMSC で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。

```
workspace
    ＋[ RL78 / R8C ]
        ＋ [ CCRL / CS+ / IAR / e2 studio / HEW ]
            ＋ [ RL78G1C / RL78L1C / R8C3MK / R8C3MU / R8C34K / R8C34U ]
                ＋ PERI                                  ビルド結果
                ＋ src
                ＋――― media_driver [Mass Storage Class 用メディアドライバ]
                │          ＋――― eeprom                 EEPROM ドライバ
                ＋――― MSCFW [Mass Storage Class driver ]
                │          ＋――― inc                    MSC ドライバ共通ヘッダファイル
                │          ＋――― MEDIA                  メディアドライバ
                │          ＋――― PMSC                   MSC ドライバ
                ＋―――SmplMain [サンプルアプリケーション ]
                │          ＋――― APL                    サンプルアプリケーション
                ＋―――USBSTDFW [全ての USB ドライバに共通な基本ファームウェア ]
                │          ＋――― inc                    USB ドライバ共通ヘッダファイル
                │          ＋――― src                    USB ドライバ
                ＋――― HwResource [ MCU 初期化等のハードウェアアクセス層 ]
                           ＋――― inc                    H/W リソースヘッダファイル
                           ＋―――src                     H/W リソース
```

[Note]

a. CS+フォルダ下には、CA78K0R コンパイラ用のプロジェクトが格納されています。

b. e2 studio フォルダ下には、KPIT GNU コンパイラ用のプロジェクトが格納されています。

c. CS+上で CC-RL コンパイラをご使用になる場合は、「13 **e2 studio 用プロジェクトを CS+で使用する場合**」を参照してください。

Table 3-2に PDCD 関連で提供するファイル構成を示します。

Table 3-2　ファイル構成

ファイル名	説明	備考
src/MSCFW/inc/r_usb_catapi_define.h	デバイスドライバ用ヘッダファイル	
src/MSCFW/inc/r_usb_cmsc_define.h	PDCD(PMSCD+PMSDD) 共通ヘッダファイル	
src/MSCFW/inc/r_usb_pmsc_api.h	PMSC API 関数ヘッダファイル	
src/MSCFW/inc/r_usb_pmsc_define.h	PMSCD 用ヘッダファイル	
src/MSCFW/inc/r_usb_pmsc_extern.h	外部参照用ヘッダファイル	
src/MSCFW/MEDIA/r_usb_atapi_driver_config.h	デバイスドライバ設定用ヘッダファイル	
src/MSCFW/MEDIA/r_usb_atapi_driver.c	デバイスドライバ(PMSDD/media driver)	ATAPI 用サンプルコード
src/MSCFW/PMSC/r_usb_pmsc_ddi.c	PMSDD 用インタフェース関数(DDI) ドライバ登録､ATAPI コマンド完了コールバック	
src/MSCFW/PMSC/r_usb_pmsc_driver.c	USB クラスドライバ (PMSCF)	
src/MSCFW/PMSC/r_usb_pmsc_pci.c	PCD 用インタフェース関数 (PCI)	
src/MSCFW/PMSC/r_usb_pmsc_request.c	PCD 用インタフェース関数(クラスリクエスト)	
src/MSCFW/APL/r_usb_pmsc_descriptor.c	マスストレージクラス用ディスクリプタ	ユーザシステムに応じて要変更

3.3　システム資源

スケジューラは、メールボックスのメッセージの有無およびタスク優先度に従いタスクスケジューリングを行います。

PMSC をスケジューラに登録して使用する為に定義した ID、優先度を**Table 3-3**に示します。

これらについては、“r_usb_ckernelid.h”ヘッダファイルで定義します。

Table 3-3　スケジューラ登録ID一覧

オブジェクトタイプ	タスク ID/メールボックス ID	概要
Task	USB_PCD_TSK / USB_TID_0	usb_pstd_pcd_task (r_usb_pdriver.c) Priority: USB_TID_0 (default=0)
	USB_PMSC_TSK / USB_TID_2	PMSCD, or **usb_pmsc_Task** *(r_usb_pmsc_driver.c)* Priority: USB_TID_2(default=2)
	USB_PFLSH_TSK / USB_TID_1	PMSDD, or **usb _pmsc_SmpAtapiTask** *(r_usb_atapi_driver.c)* Priority: USB_TID_1 (default=1)
Mailbox ID	USB_PMSC_MBX / USB_PMSC_TSK	PDCD => PMSCD / PMSDD => PMSCD *(r_usb_pmsc_pci.c, r_usb_pmsc_driver.c, r_usb_pmsc_ddi.c)*
	USB_PFLSH_MBX / USB_PFLSH_TSK	PMSCD => PMSDD mailbox ID *(r_usb_atapi_driver.c)*

4. ペリフェラル用 MSC サンプルアプリケーション(APL)

ペリフェラル用 MSC サンプルアプリケーションプログラム (以降、APL)は、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、ファイルの読み書きなど、通信を行うことができます。

マスストレージ規格で定義された転送プロトコル(BOT)のコマンドセットを使用することによりストレージデバイスを制御することが出来ます。

以下に USB でサポート可能なコマンドを示します。

- SFF-8070i, (ATAPI) - 本サンプルコードがサポートするコマンドセットです。.
- SFF-8020i, MMC-2 (ATAPI)
- QIC-157
- UFI
- SCSI transparent command set

本サンプルマスストレージデバイスドライバは、SFF-8070i(ATAPI)ストレージコマンドセットをサポートします。.

4.1 動作環境について

この PMSC サンプルファームウェアでは、ストレージメディアは、512K EEPROM を使用しています。この EEPROM は、SPI/CSI で制御する仕様になっています。

なお、EEPROM は、RSK ボード上には実装されていません。この PMSC サンプルファームウェアを動作させるためには、EEPROM をご用意いただき、SPI 接続のボードの改造が必要になります。

[Note]

CSI(Communication Serial Interface)は、RL78 シリーズで実装されているインタフェース機能です。

APL の動作環境例をFigure 4-1、APL 動作例をFigure 4-2に、EEPROM 接続仕様をTable 4-1に示します。

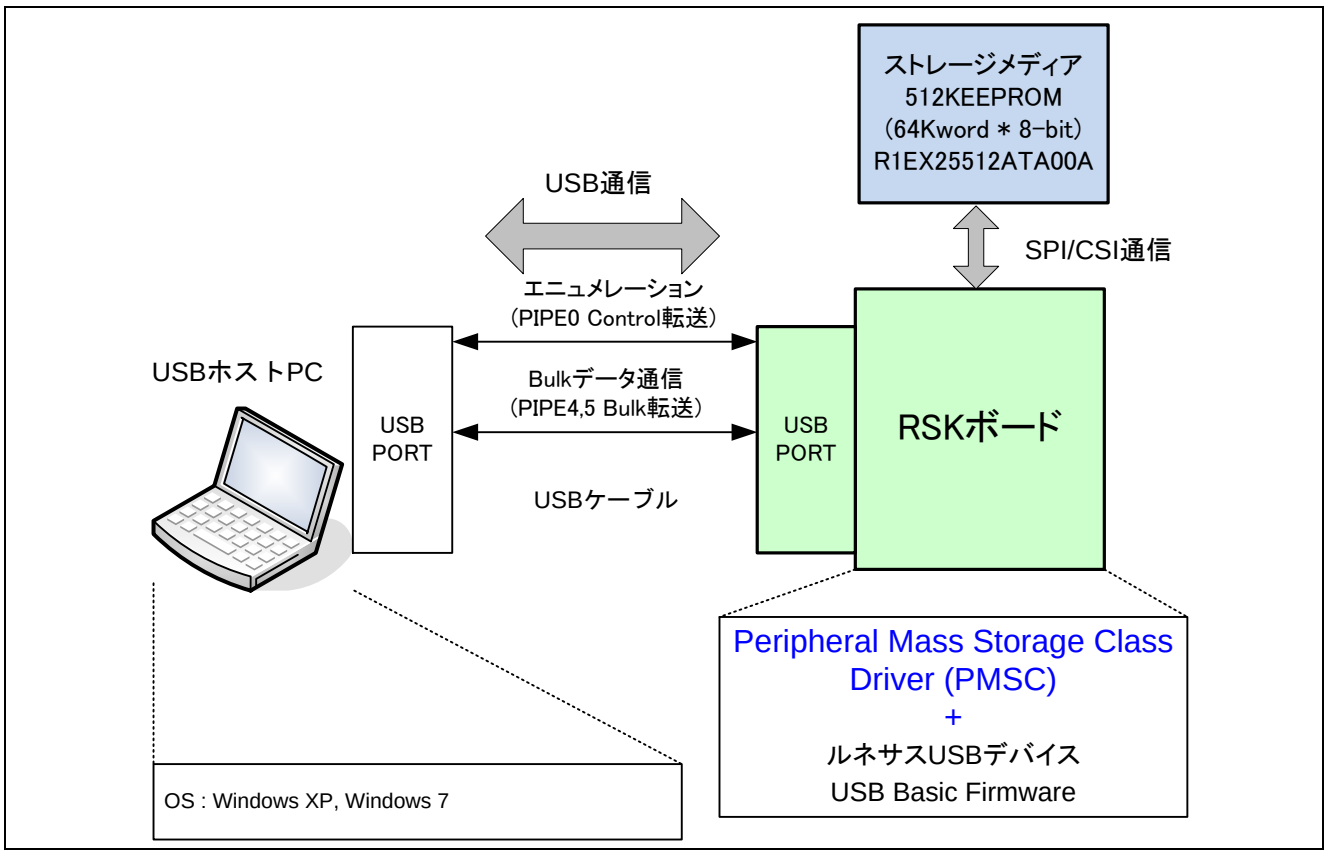

Figure 4-1 APL 動作環境例

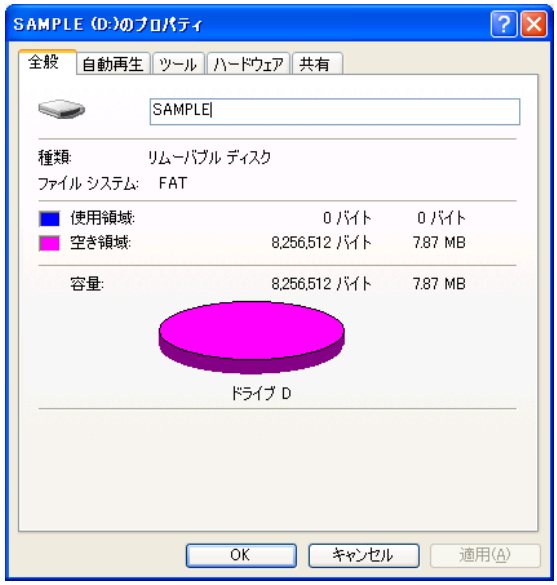

Figure 4-2 APL 動作例

Table 4-1 EEPROM 接続仕様

RL78/G1C			
接続信号種別	**CSI01 信号名**	**RSK ポート/接続端子**	**EEPROM 端子/ピン番号**
クロック	SCK01	P75/J1-6	C/6
データ転送(RL78/G1C->EEPROM)	SI01	P74/J1-7	D/5
データ転送(RL78/G1C<-EEPROM)	SO01	P73/J1-8	Q/2
チップセレクト	--	P30/J1-12	S/1
RL78/L1C			
接続信号種別	**CSI20 信号名**	**RSK ポート/接続端子**	**EEPROM 端子/ピン番号**
クロック	SCK20	P10/J4-2	C/6
データ転送(RL78/L1C->EEPROM)	SI20	P11/J4-1	D/5
データ転送(RL78/L1C<-EEPROM)	SO20	P12/J3-25	Q/2
チップセレクト	--	P30/J2-12	S/1

4.2 アプリケーションプログラムフロー

ホストからの要求に応じ、マスストレージクラスドライバおよびマスストレージデバイスドライバが処理を行いますので、アプリケーションプログラムとしての処理はなにも行いません。

サンプルのマスストレージデバイスドライバは、SFF-8070i(ATAPI)ストレージコマンドに対応しています。

APL の処理概要フローを**Figure 4-3**に示します。

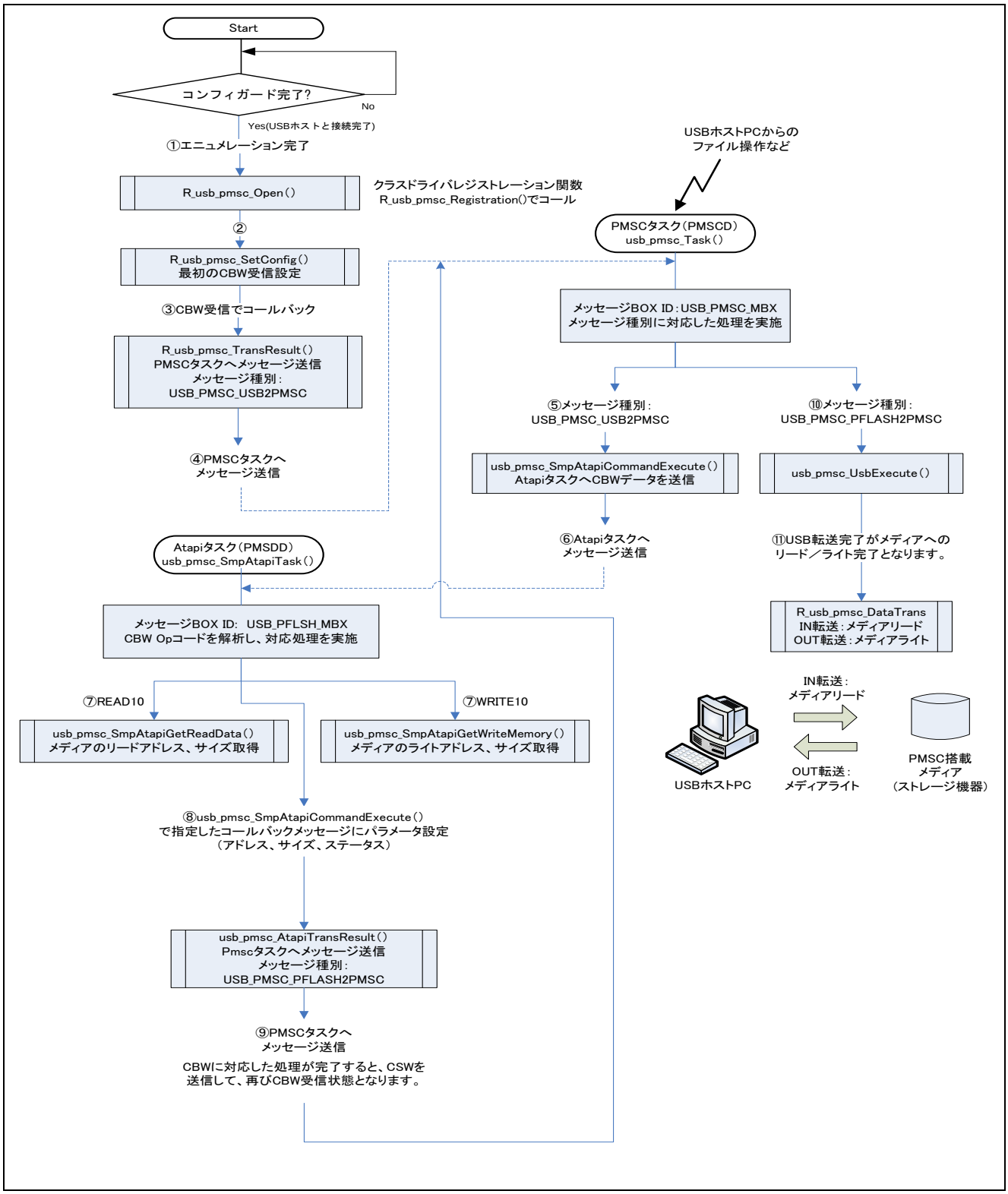

Figure 4-3 APL 処理概要フロー

4.3 APL 関数一覧

Table 4-2に APL の関数を示します。

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_pmsc_task_start	ペリフェラル USB 用の各種タスクスタートアップ処理
usb_papl_task_start	アプリタスク・スタートアップ処理
usb_apl_task_switch	タスクスイッチングループ

5. デバイスクラスドライバ（PDCD）

5.1 基本機能

PDCD の機能は以下のとおりです。

(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(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

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**Figure 5-1**に示します。

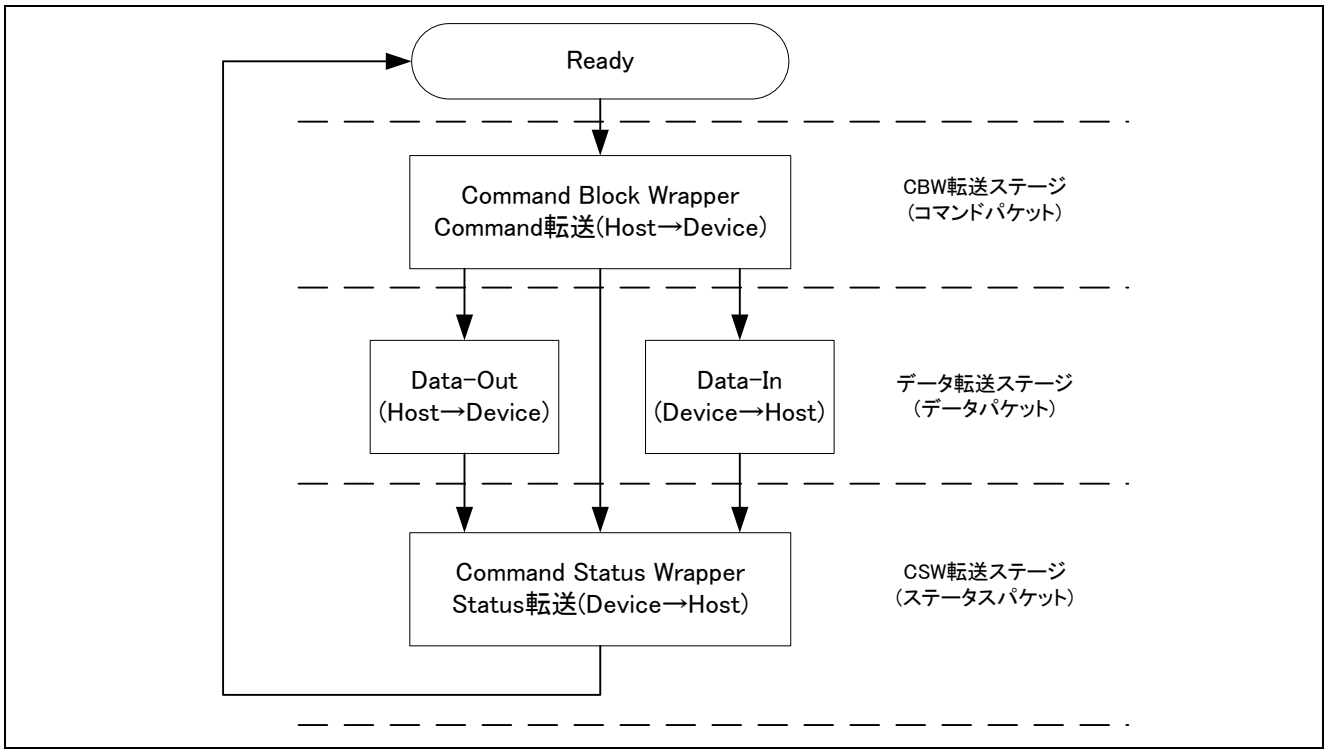

Figure 5-1 BOT プロトコル概要

5.2.1 CBW 処理

Host から CBW（Command Block Wrapper）を受け取った PMSCD 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を PMSDD に通知し、コマンド解析を要求します。PMSCD は PMSDD が解析した情報（コマンドの有効性、データ転送方向とサイズ）とラッパ内情報（データ通信方向とサイズ）を基に処理を行います。

5.2.2 送受信データのないストレージコマンドのシーケンス

Figure 5-2にデータ転送を伴わないストレージコマンドのシーケンスを示します。

(1). CBW 転送ステージ

CBW 転送ステージでは PMSCD から PCD へ CBW 受信要求を行い、PCD で 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 PMSCD が CBW の有効性を確認し、PMSDD にストレージコマンド（CBWCB）を渡します。PMSDD は送受信データがないコマンドであることを確認し、PMSCD にその結果を返します。PMSCD では PMSDD のストレージコマンド解析結果と CBW 内情報を比較し、PMSDD にストレージコマンド実行を要求します。PMSDD はストレージコマンド処理を実行し、その結果を PMSCD にコールバックで戻します。

(2). CSW 転送ステージ

PMSCD は実行結果から CSW（Command Status Wrapper）を作成し、PCD を介して Host に CSW を送信します。

PCD 動作の詳細は「Renesas USB MCU USB Basic Mini Firmware アプリケーションノート」を参照下さい。

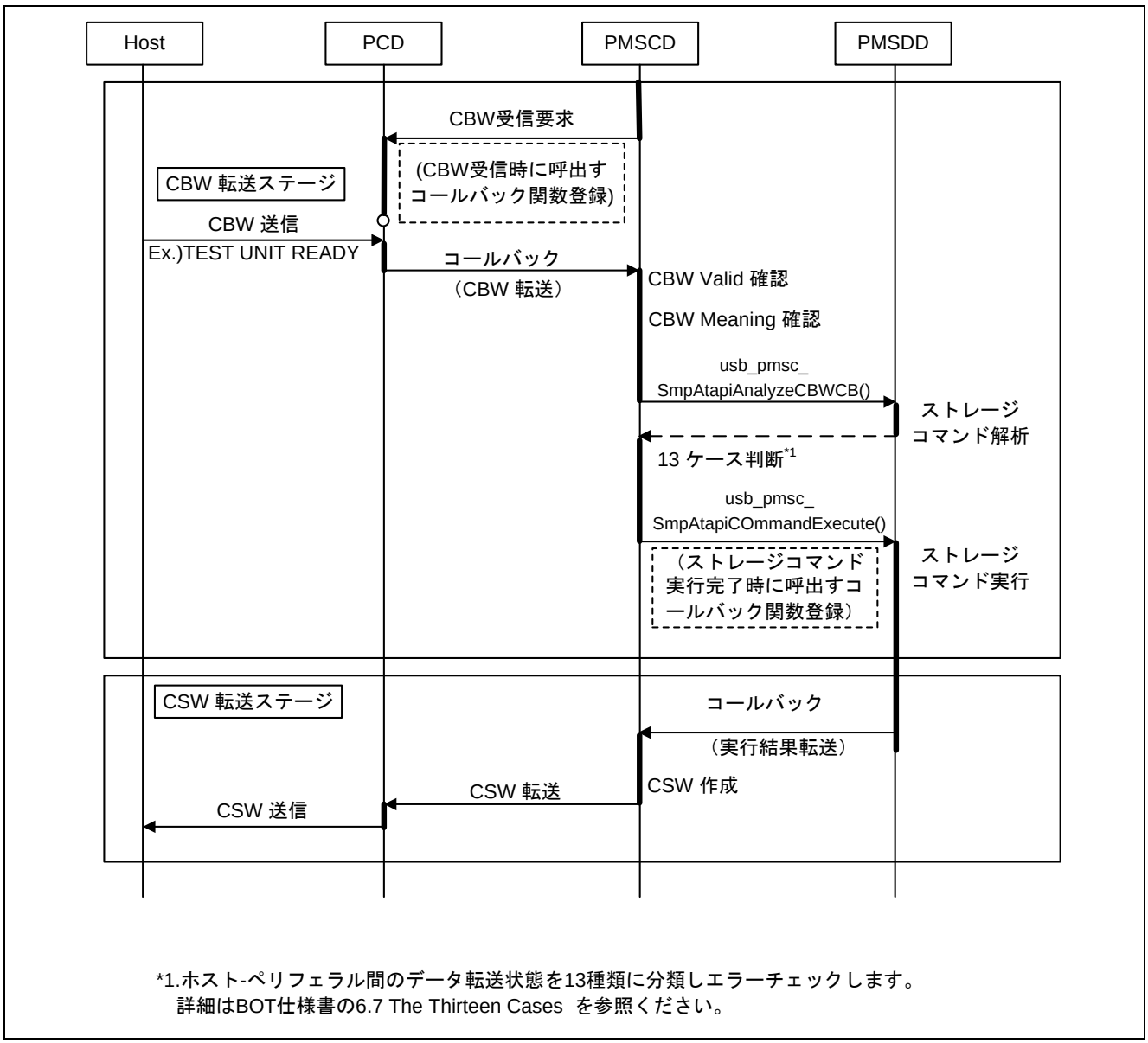

Figure 5-2 送受信データのないストレージコマンドのシーケンス

5.2.3 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

Figure 5-3に Peripheral 側からの送信（IN）データがあるストレージコマンドのシーケンスを示します。

(1). CBW 転送ステージ

CBW 転送ステージでは PMSCD から PCD へ CBW 受信要求を行い、PCD で 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 PMSCD が CBW の有効性を確認し、PMSDD にストレージコマンド（CBWCB）を渡します。PMSDD はデータ送信コマンドであることを確認し、PMSCD にその結果を返します。PMSCD では PMSDD のストレージコマンド解析結果と CBW 内情報を比較し、PMSDD にストレージコマンド実行を要求します。PMSDD はストレージコマンド処理を実行し、その結果を PMSCD にコールバックで戻します。

(2). データ転送ステージ

PMSCD 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。PMSCD は PCD より送信完了が通知されると再度 PMSDD にコマンド継続を要求し、データ送信を繰り返します。

(3). CSW 転送ステージ

PMSDD からコマンド処理終了の結果を受け取ると CSW（Command Status Wrapper）を作成し、PCD を介して Host に CSW を送信します。

PCD 動作の詳細は「Renesas USB MCU USB Basic Firmware アプリケーションノート」を参照下さい。

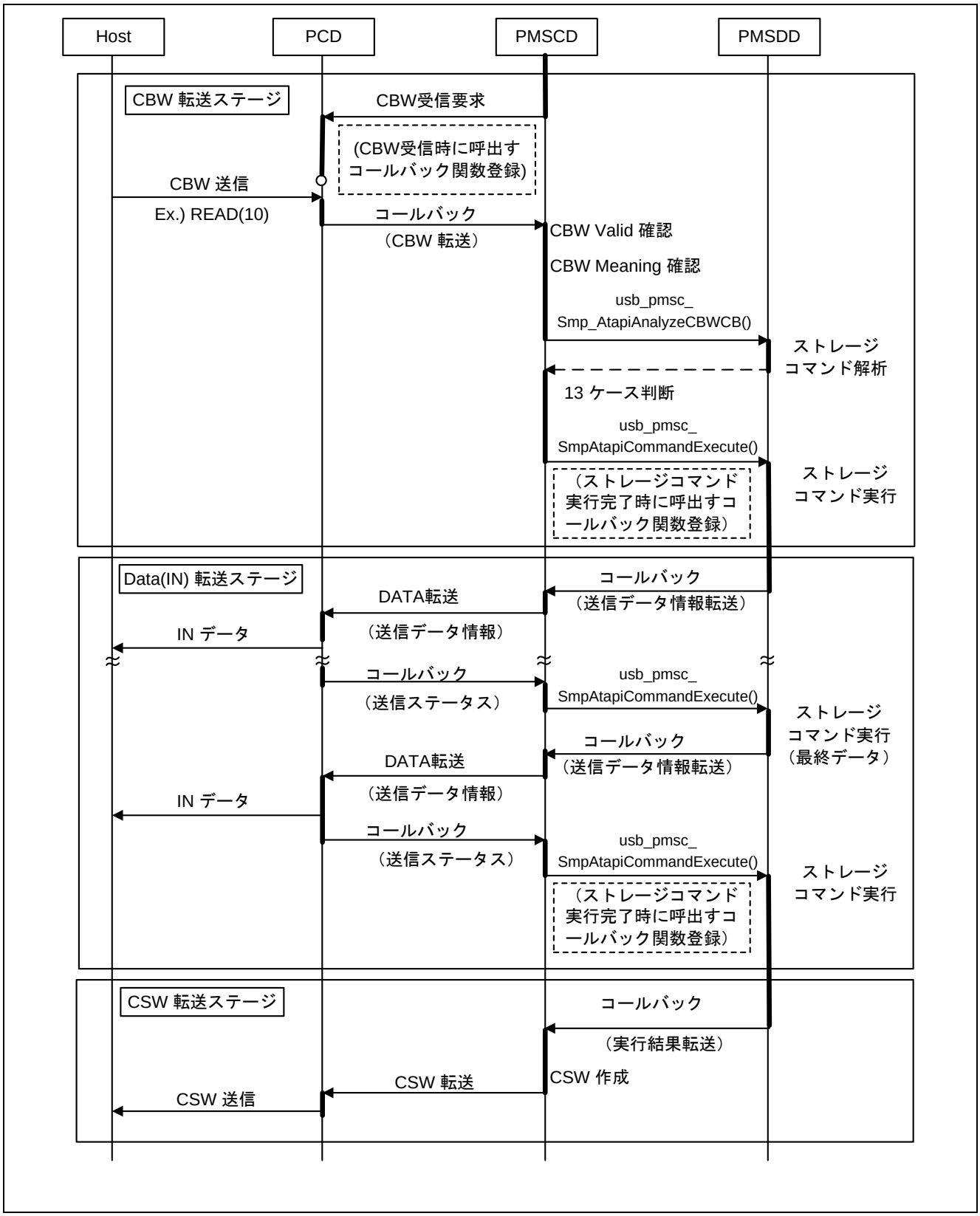

Figure 5-3　送信（IN）データがあるストレージコマンドのシーケンス

5.2.4 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

Figure 5-4に Peripheral 側への受信（OUT）データがあるストレージコマンドのシーケンスを示します。

(1). CBW 転送ステージ

CBW 転送ステージでは PMSCD から PCD へ CBW 受信要求を行い、PCD で 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 PMSCD が CBW の有効性を確認し、PMSDD にストレージコマンド（CBWCB）を渡します。PMSDD はデータ受信コマンドであることを確認し、PMSCD にその結果を返します。PMSCD では PMSDD のストレージコマンド解析結果と CBW 内情報を比較し、PMSDD にストレージコマンド実行を要求します。PMSDD はストレージコマンド処理を実行し、その結果を PMSCD にコールバックで戻します。

(2). データ転送ステージ

PMSCD 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。PMSCD は PCD より受信完了が通知されると再度 PMSDD にコマンド継続を要求し、データ受信を繰り返します。

(3). CSW 転送ステージ

PMSDD からコマンド処理終了の結果を受け取ると CSW（Command Status Wrapper）を作成し、PCD を介して Host に CSW を送信します。

PCD 動作の詳細は「Renesas USB MCU USB Basic Mini Firmware アプリケーションノート」を参照下さい。

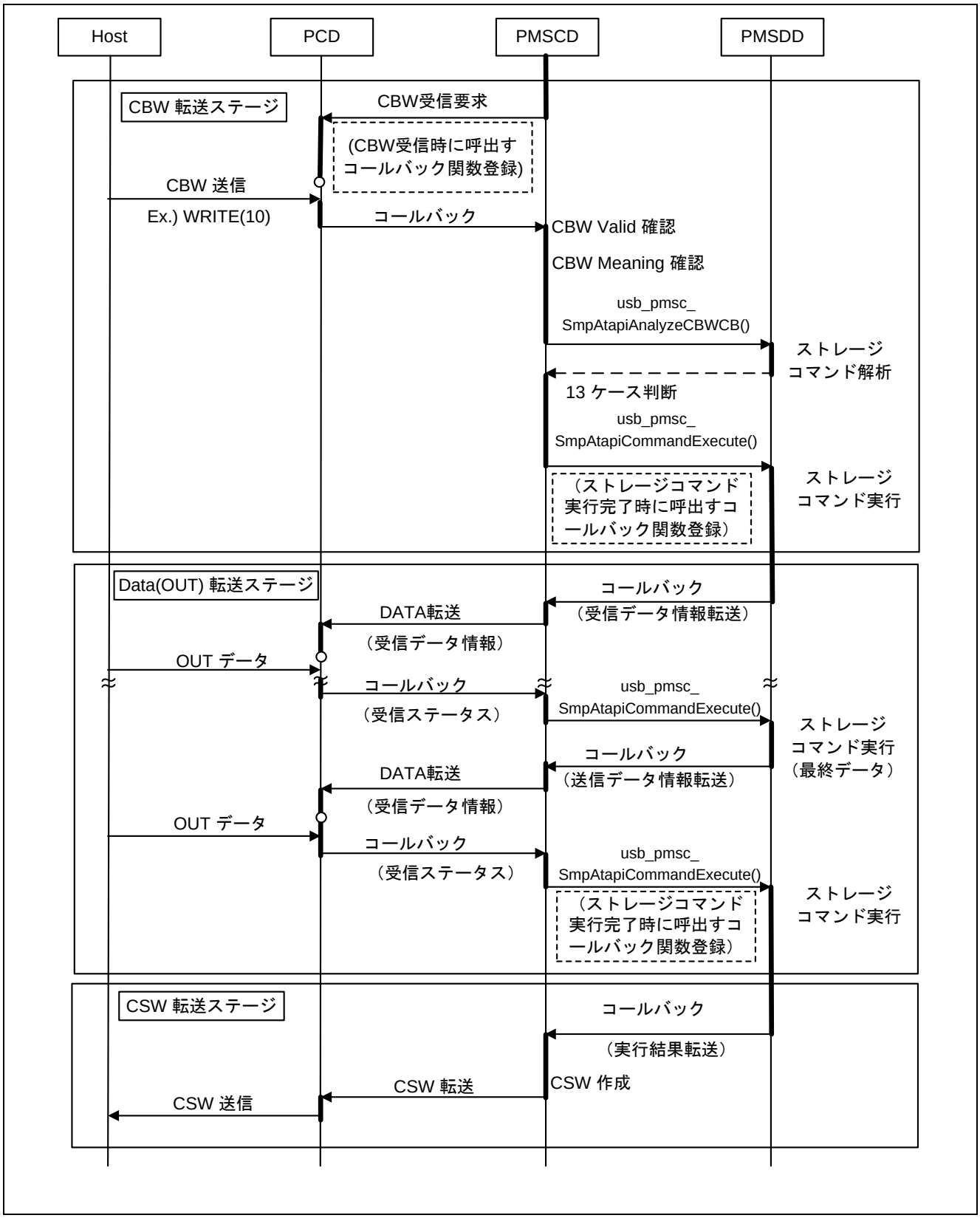

Figure 5-4　受信（OUT）データがあるストレージコマンドのシーケンス

5.2.5 クラスリクエストのアクセスシーケンス

Figure 5-5にマスストレージクラスリクエスト要求がきた場合のシーケンスを示します。

(1). Setup ステージ

PCD はコントロール転送の SETUP ステージでクラスリクエストを受信した場合に、PMSCD にリクエスト受信を通知します。

(2). Data ステージ

PMSCD はコントロール転送のデータステージを実行し、PCD に対してコールバック関数でデータステージ終了を通知します。

(3). ステータスステージ

PCD はステータスステージを実行しコントロール転送を終了します。

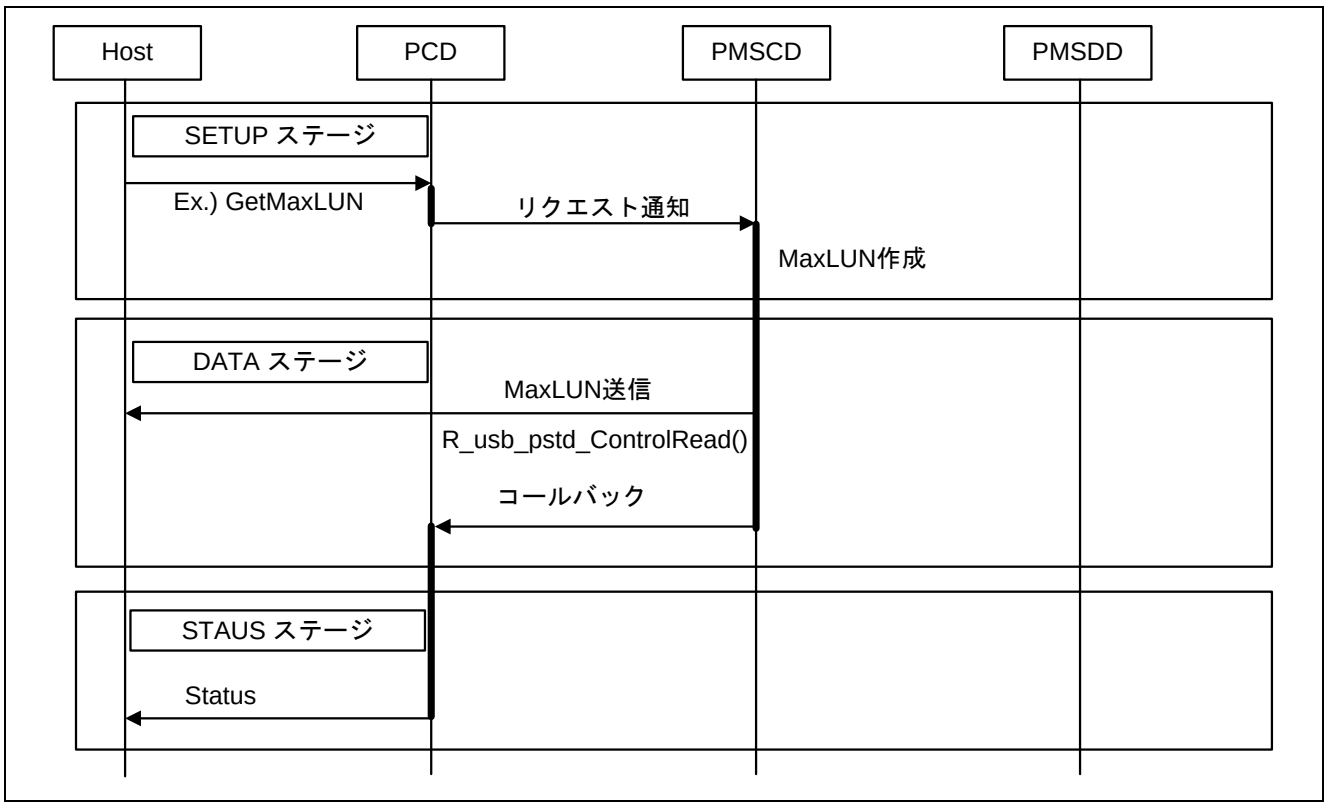

Figure 5-5 クラスリクエストのシーケンス

6. USB マスストレージクラスドライバ（PMSCD）

6.1 基本機能

PMSCD の基本機能はレジストレーション、ペリフェラルマスストレージクラスドライバのオープン処理、クローズ処理のみです。他のペリフェラルマスストレージのシーケンス処理等については、”5. **デバイスクラスドライバ（PDCD）**”章に記載されていますので、この章をご覧ください。

6.2 API 関数

Table 6-1に API 関数を示します。

Table 6-1 API 関数一覧

関数名	説明
R_usb_pmsc_Registration	PMSC ドライバ登録
R_usb_pmsc_Open	PMSC ドライバオープン関数
R_usb_pmsc_Close	PMSC ドライバクローズ関数

R_usb_pmsc_Registration

PMSC ドライバ登録

形式

void R_usb_pmsc_Registration(void)

引数

― ―

戻り値

― ―

解説

ペリフェラル用マスストレージクラスの登録を行います。

アプリケーションプログラムに合わせて登録関数の変更を行います。

usb_pcdreg_t 型構造体を使用して、PCD へ登録して下さい。

（型については「Renesas USB MCU USB Basic Mini Firmware アプリケーションノート」を参照してください）

usb_pcdreg_t 型構造体メンバに登録する内容は次の通りです。

pipetbl	パイプ情報テーブルのアドレス
devicetbl	Device Descriptor アドレス
configtbl	Configuration Descriptor アドレス
stringtbl	String Descriptor アドレステーブルのアドレス
statediagram	USB ステート遷移時に起動するコールバック関数
ctrltrans	ユーザ用コントロール転送時に起動するコールバック関数

補足

- コールバック処理が不要の場合はダミー関数を用意して登録して下さい。
- USB の状態遷移については”Universal Serial Bus Specification Revision 2.0” の”Figure 9-1. Device State Diagram”を参照下さい。
- String Descriptor アドレステーブルとは複数の String Descriptor アドレスをテーブルとして並べた物です。以下に参考例を記します。

```
uint8_t  *usb_gpmsc_StrPtr[USB_STRINGNUM] =
{
  usb_gpmsc_StringDescriptor0,    /* 言語 ID String Descriptor アドレス */
  usb_gpmsc_StringDescriptor1,    /* iManufacturer String Descriptor アドレス */
  usb_gpmsc_StringDescriptor2,    /* iProduct String Descriptor アドレス */
  usb_gpmsc_StringDescriptor3,    /* iInterface String Descriptor アドレス */
  usb_gpmsc_StringDescriptor4,    /* iConfiguration String Descriptor アドレス */
  usb_gpmsc_StringDescriptor5,    /* iConfiguration String Descriptor アドレス */
  usb_gpmsc_StringDescriptor6     /* iSerialNumber String Descriptor アドレス */
};
```

使用例

```
void usb_pmsc_task_start( void )
{
   R_usb_pmsc_Registration();     /* Peripheral Application Registration */
   R_usb_pstd_PcdChangeDeviceState(USB_DO_SETHWFUNCTION); /* Initialize USB HW */
   R_usb_pmsc_driver_start();   /* Peripheral Class Driver Task Start Setting */
   usb_pstd_usbdriver_start(); /* Peripheral USB Driver Start Setting */
   usb_papl_task_start();      /* Peripheral Application Task Start Setting */
}
```

R_usb_pmsc_Open

PMSC オープン関数

形式

usb_er_t R_usb_pmsc_Open(uint16_t data1, uint16_t data2)

引数

uint16_t	data1	：非使用
uint16_t	data2	：非使用

戻り値

uint16_t	－	処理結果 0：USB_E_OK

解説

USB ホスト装置と接続され、USB 通信が可能となった時に呼び出される関数で、CBW 受信設定を行います。

補足

usb_pcdreg_t 構造体のメンバ statediagram に登録したコールバック関数から本関数をコールしてください。

使用例

```
void usb_pmsc_change_device_state( uint16_t data1, uint16_t device_state )
{
    switch ( device_state )
    {
    case USB_STS_CONFIGURED:
        R_usb_pmsc_Open(data1, device_state);
    break;
    }
}
```

R_usb_pmsc_Close

PMSC クローズ関数

形式

usb_er_t R_usb_pmsc_Close(uint16_t data1, uint16_t data2)

引数

uint16_t	data1	：非使用
uint16_t	data2	：非使用

戻り値

uint16_t	－	処理結果 0：USB_E_OK

解説

デタッチ状態遷移時に呼び出される関数で処理は行っていません。必要に応じて処理を追加してください。

補足

本関数は usb_pcdreg_t 構造体のメンバ statediagram にコールバック関数として登録してください。

使用例

```
void usb_pmsc_change_device_state( uint16_t data1, uint16_t device_state )
{
    switch ( device_state )
    {
    case USB_STS_DETACH:
        R_usb_pmsc_Close(data1, device_state);
    break;
    }
}
```

6.3 PMSCD の登録

PMSCD は PCD 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。

サンプルの関数を参考に、PeripheralRegistration()関数で登録してください。

詳細は「Renesas USB MCU USB Basic Mini Firmware アプリケーションノート」をご参照ください。

6.4 ユーザ定義テーブル

PCD が使用するディスクリプタテーブル及びパイプ情報テーブルを作成する必要があります。

サンプルの r_usb_PMSCdescriptor.c ファイル、r_usb_PMSCdefEp.h ファイルを参考に作成してください。

詳細は「Renesas USB MCU USB Basic Mini Firmware アプリケーションノート」をご参照ください。

7. ペリフェラルマスストレージデバイスドライバ（PMSDD）

PMSDD の主な機能は以下のとおりです。

1) PMSCD から受け取ったストレージコマンドの解析
2) PMSCD から受け取ったストレージコマンドの実行
3) ストレージコマンド実行による通信データ情報及び実行結果の PMSCD への転送

PMSDD は PMSCD から通知されるストレージコマンドの解析と、コマンド処理を行います。

PMSDD の特長は以下のとおりです。

1. SFF-8070i(ATAPI)に対応しています。
2. ストレージコマンドのうち、以下のコマンドに応答します。
   READ10
   INQUIRY
   REQUEST_SENSE
   MODE_SENSE6
   MODE_SENSE10
   READ_FORMAT_CAPACITY
   READ_CAPACITY
   WRITE10
   WRITE_AND_VERIFY
   MODE_SELECT6
   MODE_SELECT10
   FORMAT_UNIT
   TEST_UNIT_READY
   START_STOP_UNIT
   SEEK
   VERIFY10
   PREVENT_ALLOW
3. 転送データ長がブロック数（ユーザ指定）より大きい場合にはデータ転送を分割します。
4. マスタブートレコーダ（FAT12）のサンプルテーブルを用意しています。

7.1 PMSDD 構造体

USB_PMSC_CDB_t はストレージコマンドの構造体です。ストレージコマンド(SFF-8070i)のフォーマットはコマンド種別によって異なるため、共用体を用いています。10 種類のコマンド種別を 4 つのパターンに分類した共用体をTable 7-1に詳細を示します。

Table 7-1 USB_PMSC_CDB_t 構造体

共用体メンバ	型	構造体メンバ	bit 数	コマンド種別
s_usb_ptn0	uint8_t	uc_OpCode		コマンド判定（共通）
	uint8_t s_LUN	b_LUN	3	
		b_reserved	5	
	uint8_t	uc_data		
s_usb_ptn12	uint8_t	uc_OpCode		INQUIRY / REQUEST_SENSE
	uint8_t s_LUN	b_LUN	3	
		b_reserved4	4	
		b_immed	1	
	uint8_t	uc_rsv2[2]		
	uint8_t	uc_Allocation		
	uint8_t	uc_rsv1[1]		
	uint8_t	uc_rsv6[6]		
s_usb_ptn378	uint8_t	uc_OpCode		未使用（FORMAT UNIT）
	uint8_t s_LUN	b_LUN	3	
		b_FmtData	1	
		b_CmpList	1	
		b_Defect	3	
	uint8_t	ul_LBA0		
	uint8_t	ul_LBA1		
	uint8_t	ul_LBA2		
	uint8_t	ul_LBA3		
	uint8_t	uc_rsv6[6]		
s_usb_ptn4569	uint8_t	uc_OpCode		READ10 / WRITE10 / WRITE _AND_VERIFY / MODE_SENSE / FORMAT CAPACITY / MODE SELECT
	uint8_t s_LUN	b_LUN	3	
		b_1	1	
		b_reserved2	2	
		b_ByteChk	1	
		b_SP	1	
	uint8_t	ul_LogicalBlock0		
	uint8_t	ul_LogicalBlock1		
	uint8_t	ul_LogicalBlock2		
	uint8_t	ul_LogicalBlock3		
	uint8_t	uc_rsv1[1]		
	uint8_t	us_Length_Hi		
	uint8_t	us_Length_Lo		
	uint8_t	uc_rsv3[3]		

USB_PMSC_CBM_t はストレージコマンド解析結果の構造体です。Table 7-2に詳細を示します。

Table 7-2　USB_PMSC_CBM_t 構造体

型	メンバ名	説明	備考
uint32_t	ar_rst	PMSDD でのストレージコマンド解析結果（データ方向）	usb_pmsc_SmpAtapi AnalyzeCbwCb 関数の解析結果
uint32_t	ul_size	PMSDD でのストレージコマンド解析結果（データサイズ）	usb_pmsc_SmpAtapi AnalyzeCbwCb 関数の解析結果

7.2　PMSDD 関数一覧

Table 7-3に PMSDD の関数を示します。

Table 7-3　PMSDD 関数一覧

関数名	説明
usb_pmsc_SmpAtapiAnalyzeCbwCb	ストレージコマンド(SFF-8070i)の解析
usb_pmsc_SmpAtapiTask	PMSDD のメインタスク
usb_pmsc_SmpAtapiInitMedia	PMSDD 起動時の処理
usb_pmsc_SmpAtapiCloseMedia	PMSDD 終了時の処理
usb_pmsc_SmpAtapiCommandExecute	PMSDD から PMSDD タスクにメッセージを送信

7.3 PMSDD タスク説明

PMSDD は PMSCD からストレージコマンドとデータ転送結果を受け取り、ストレージコマンドを実行します。Table 7-4に PMSDD のコマンド処理を示します。転送データサイズが USB_ATAPI_TRANSFER_UNIT を超えた場合はデータを分割して転送を行います。また、メモリアクセス以外のデータ送信コマンドは返答用データテーブル（*1 usb_gpmsc_AtapiDataSize[]、usb_gpmsc_AtapiDataIndx[]、usb_gpmsc_AtapiReqIndx[]、usb_gpmsc_AtapiRdDataTbl[]）から送信データを作成します。

*1 返答用データテーブルはストレージコマンドセット SFF-8070i で定められた値によって構成され、テーブル参照の為のインデックスはサブクラスで定められるコマンド(uc_OpCode：Table 7-1 USB_PMSC_CDB_t 構造体参照)によって決定します。

Table 7-4 ストレージコマンド対応関数

ストレージコマンド	対応関数	備考
READ10	pmsc_atapi_get_read_memory()	先頭アドレスとサイズを取得
INQUIRY	pmsc_atapi_get_read_data()	配列 g_pmsc_atapi_rd_dat_tbl からコマンド応答データ選択
REQUEST_SENSE	pmsc_atapi_get_read_data ()	配列 g_pmsc_atapi_rd_dat_tbl からコマンド応答データ選択
MODE_SENSE10	pmsc_atapi_get_read_data ()	配列 g_pmsc_atapi_rd_dat_tbl からコマンド応答データ選択
READ_FORMAT_CAPACITY	pmsc_atapi_get_read_data ()	配列 g_pmsc_atapi_rd_dat_tbl からコマンド応答データ選択
READ_CAPACITY	pmsc_atapi_get_read_data ()	配列 g_pmsc_atapi_rd_dat_tbl からコマンド応答データ選択
WRITE10	pmsc_atapi_get_write_memory()	先頭アドレスとサイズを取得
WRITE_AND_VERIFY	pmsc_atapi_get_write_memory()	先頭アドレスとサイズを取得
MODE_SELECT10	pmsc_atapi_get_write_memory()	先頭アドレスとサイズを取得
FORMAT_UNIT	pmsc_atapi_get_write_memory()	先頭アドレスとサイズを取得
TEST_UNIT_READY	usb_pmsc_SmpAtapiTask()	Status = USB_PMSC_CMD_COMPLETE
START_STOP_UNIT	usb_pmsc_SmpAtapiTask()	Status = USB_PMSC_CMD_COMPLETE
SEEK	usb_pmsc_SmpAtapiTask()	Status = USB_PMSC_CMD_COMPLETE
VERIFY10	usb_pmsc_SmpAtapiTask()	Status = USB_PMSC_CMD_COMPLETE
PREVENT_ALLOW	usb_pmsc_SmpAtapiTask()	Status = USB_PMSC_CMD_FAILED
その他	usb_pmsc_SmpAtapiTask()	Status = USB_PMSC_CMD_ERROR

8. メディアドライバインタフェース

本章は、PMSC 用に使用されるメディアドライバインタフェースの概要について記載しています。

メディアドライバインタフェース用 API の詳細説明や新しいメディアドライバの作成方法について別冊アプリケーションノート(R01AN1443JU_RX)を参照してください。

PMSC は、ストレージメディアとして様々なメディアデバイスで対応することができます。ストレージメディアドライバインターフェース関数は、下層のメディアドライバに関わらずです。共通 API (R_MEDIA_Read, R_MEDIA_Write など)として提供されます。したがって、PMSC は、この API をサポートするメディアドライバならどれでも使用できます。

デフォルトでは、RAM ディスクメディアドライバとして動作します。特定の RAM 領域をメディア領域として使用しています。

8.1 メディアドライバ API 関数概要

メディアドライバ API は、PMSC から呼び出され固有のメディアデバイスドライバにアクセスします。メディアドライバの選択は、ユーザがコンフィグレーションファイルを変更し、設定を行う必要があります。コンフィグレーションファイルは、*r_media_driver_api_config.h* と *r_usb_atapi_driver_config.h* があります。*r_media_driver_api_config.h* は、メディアドライバ API 用のコンフィグレーションファイルで、メディアデバイスのリストが記述されています。*r_usb_atapi_driver_config.h* は、PMSC 用のコンフィグレーションファイルで、ユーザが選択したメディアデバイスの設定を行います。

ストレージコマンドは SFF-8070i(ATAPI)に対応し、このストレージコマンドは、*r_usb_atapi_driver.c* で処理され、ストレージメディアへのリードやライト処理等は、以下に示すメディアドライバ API を経由してストレージドライバに渡されます。

Table 8-1 Block Access Media Driver API 関数

関数名	説明
R_MEDIA_Initialize	Registers the media driver
R_MEDIA_Open	Open media driver
R_MEDIA_Close	Close media driver
R_MEDIA_Read	Read from a media device
R_MEDIA_Write	Write to a media device
R_MEDIA_Ioctl	Perform control and query operations on a media device

8.2 メディアドライバの選択

各メディアドライバは、メディアデバイスへ読み書き等を行う関数へのポインタが設定されたドライバ実装構造体が用意されています。このドライバ実装構造体の実体は、ATAPI_MEDIA_DEVICE_DRIVER という定義名で *r_usb_atapi_driver_config.h* ファイル内にマクロ定義されています。

セクタまたはブロックサイズの値は、Windows などの USB ホスト(FAT)とメディアドライバの両者で対応できる一致した値(512 や 4096 など)を設定します。

8.2.1 メディアドライバ関数の初期設定

ドライバ実装構造体は、以下の API によって初期設定処理を行う必要があります。

```
R_MEDIA_Initialize(&ATAPI_MEDIA_DEVICE_DRIVER);
```

上記 API はドライバ実装構造体(ATAPI_MEDIA_DEVICE_DRIVER)へのポインタを g_MediaDriverList へコピーします。他の API(R_MEDIA_Open, R_MEDIA_Read など)は、該当のメディアドライバ関数をコールします。

なお、デフォルトでは、この初期設定処理は *r_media_driver_api.c* で行っています。

8.3 ストレージメディアの変更(追加)

デフォルトの EEPROM からフラッシュメモリなどの異なる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用意し、メディア API として登録する必要があります。

8.3.1 メディアドライバ関数のメディア API への登録手順

(1). r_media_driver_api.h 内で定義された戻り値と引数をもつメディアドライバ関数のソースコードを作成し、media_driver_s 構造体の各メンバに該当するメディアドライバ関数へのポインタを設定します。

(2). r_usb_apapi_driver_config.h に media_driver_s 構造体のインスタンス名をマクロ名 ATAPI_MEDIA_DEVICE_DRIVER に定義します。このマクロ名を R_MEDIA_Initialize 関数(API)の引数に指定し、初期化処理を行います。

9. EEPROM メディアドライバ

EEPROM メディアドライバは、ストレージメディアデバイスとして EEPROM メモリを使用しています。シンプルなメディアドライバとして提供することができ、USB ホストと MCU 間の PMSC データ通信を容易に実現することができます。

(1). メディアデバイス

64KByte の EEPROM メディアとして動作します。(EEPROM: Renesas R1EX25512ATA00A)

(2). メディアは、リムーバブル FAT ファイルストレージデバイスとしてプリフォーマットされています。

(3). メディアドライバは、WindowsOS(XP, 7 etc)への接続をサポートしています。

(4). デフォルトフォーマットは、USB Host からのフォーマットコマンドによる上書きが可能です。

(5). ホストは RAM ディスクに対してリード/ライトが可能であり、FAT フォーマット情報のアップデートが可能です。

9.1 EEPROM メディアドライバデフォルトフォーマット

EEPROM メディアデバイスは、リムーバブル FAT ファイルストレージデバイスとしてプリフォーマットされています。

このフォーマットは、ハードコードによって定義されたブートセクタ、FAT テーブル、ディレクトリ領域をもつ初期化データセクションとして実装されます。ブートセクタ情報等の初期化データは *r_eeprom_disk_format_data.c* ファイルに定義され、システム起動時にシステムに登録されます。

EEPROM の最初の領域は、ブートセクタ領域です。これは FAT フォーマットストレージデバイスのブロック 0 をデフォルトブートセクタ領域とする仕様に基づきます。したがって、すべてのストレージブロックはこの場所に関連します。USB ホストがメディアデバイスにアクセスするとき、論理ブロック番号(LBN)とブロックカウントに関する通知を行います。ホストは、フォーマットされたストレージデバイスの操作方法を知っているため、ホストは EEPROM 上の固有フォーマットに関する追加情報を取得するため、第一セクタをリードし、追加ファイル情報を探します。この情報からホストはアクセスすべきブロック数を取得します。

ホストは、デフォルトブートセクタ、FAT テーブル等でフォーマットを置き換えるため EEPROM を再フォーマットすることができます。この場合、ホストは、データブロックが存在する場所を EEPROM イメージから知ることができます。

[Note]

この PMSC ファームウェアから FAT 経由でストレージにアクセスすることはできません。11章を参照してください。

9.2 グローバル領域

Table 9-1 メディアドライバグローバル領域

型	変数名	説明	
uint8_t	eeprom_boot_sector	プライマリブートレコード	(sector 0)
uint8_t	usb_gpmsc_Table1	ダミー	(sector 1)
uint8_t	usb_gpmsc_TableFat	FAT テーブル	(sector 2 and 3)
uint8_t	usb_gpmsc_RootDir	ディレクトリエントリ	(sector 4)

にメディアドライバで使用しているグローバル領域を示します。

Table 9-1 メディアドライバグローバル領域

型	変数名	説明	
uint8_t	eeprom_boot_sector	プライマリブートレコード	(sector 0)
uint8_t	usb_gpmsc_Table1	ダミー	(sector 1)
uint8_t	usb_gpmsc_TableFat	FAT テーブル	(sector 2 and 3)
uint8_t	usb_gpmsc_RootDir	ディレクトリエントリ	(sector 4)

9.3 定数定義

Table 9-2にメディアドライバの定数定義一覧を示します。

Table 9-2 PMSDD 定数定義

説明	定義名	値	補足
Media type	EEPROM_MEDIATYPE	0xF8u	変更可
Signature	EEPROM_SIGNATURE	0xAA55u	変更不可
Sector size	EEPROM_SECTSIZE	512ul	変更可
Cluster size	EEPROM_CLSTSIZE	0x01u	変更可
FAT number	EEPROM_FATNUM	0x02u	変更可
Media size *1	EEPROM_MEDIASIZE	64*1024（=64Kbyte）	変更可
Total number of sectors*2	EEPROM_TOTALSECT	EEPROM_MEDIA_SIZE / EEPROM_SECTSIZE	変更不可
FAT Table Length*2	EEPROM_FATLENGTH	341ul（FAT12）	変更不可
FAT table length*2	EEPROM_FATSIZE	(((EEPROM_TOTALSECT-8) / EEPROM_FATLENGTH)+1)	変更不可
Root directory	EEPROM_ROOTTOP	(((EEPROM_FATSIZE * EEPROM_FATNUM+1)/8+1)*8)	変更不可 (未使用)
FAT start	EEPROM_FATTOP	(EEPROM_ROOTTOP - (EEPROM_FATSIZE * EEPROM_FATNUM))	変更不可 (未使用)
Root Directory size	EEPROM_ROOTSIZE	1ul	変更不可 (未使用)

*1. WindowsXP の PC に接続するには最低でも 20KByte 以上の容量が必要です。
メディアサイズを 2MByte 未満に指定すると FAT12 が選択されます。
*2.総セクタ数、FAT データ長、FAT テーブル長はメディアサイズにより自動計算します。

9.4 動作説明

Table 9-3にメディア用変数一覧、Figure 9-1にメディアブロック図を示します。

Table 9-3 メディア用変数

カテゴリ	セクタ番号	物理アドレス	サイズ
PBR	Sector 0	0x0000	512Byte
Dummy area	Sector 1	0x0200	512Byte
FAT1	Sector 2	0x0400	512Byte× EEPROM_FATSIZE
FAT2	Sector 3	0x0600	512Byte× EEPROM_FATSIZE
ROOT DIR	Sector 4	0x0800	512Byte×16

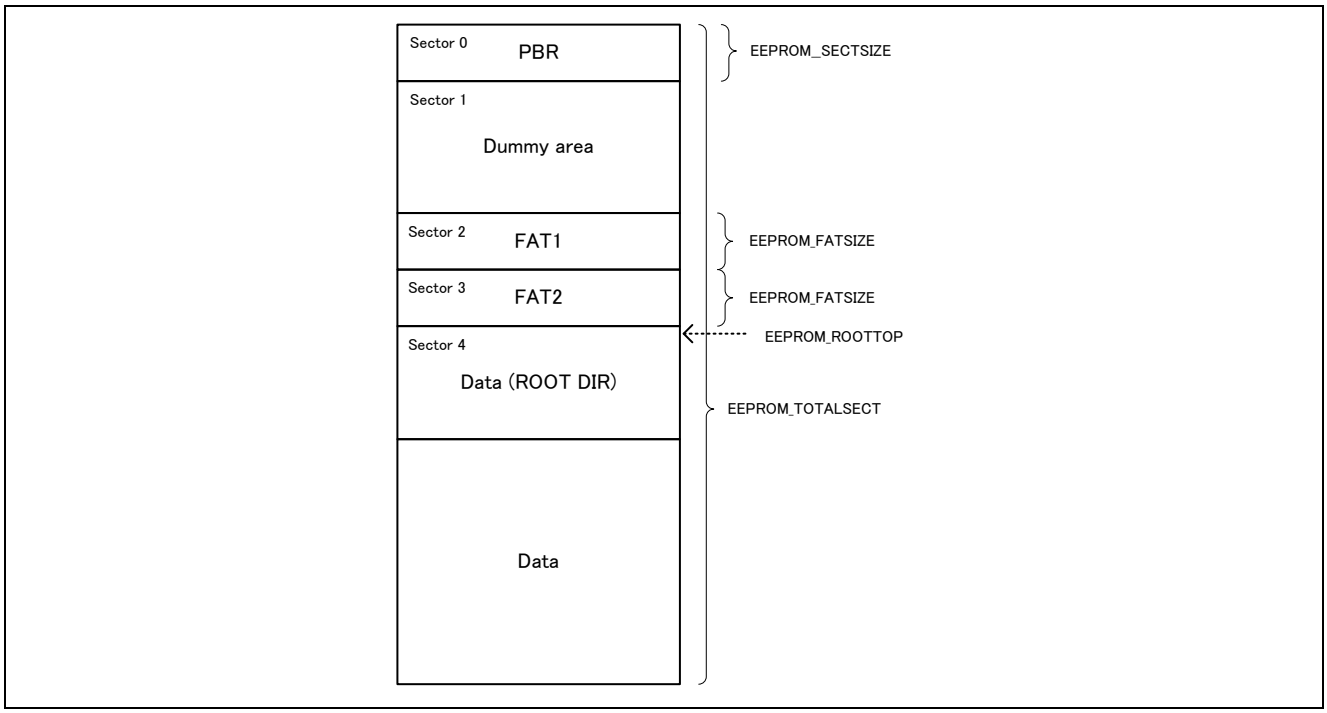

Figure 9-1 メディアブロック

10. スケジューラへの組み込みについて

スケジューラを使用する場合、タスク ID、メール BOXID、メモリプール ID 等のリソース登録は "r_usb_ckernelid.h"ファイルに対して行います。

サンプルファイルでは以下のように登録しています。

```
/* Peripheral MSC Sample Task */
#define  USB_PFLSH_TSK     USB_TID_1      /* Task ID */
#define  USB_PFLSH_MBX     USB_PFLSH_TSK  /* Mailbox ID */

/* Peripheral MSC Driver Task */
#define  USB_PMSC_TSK   USB_TID_2         /* Task ID */
#define  USB_PMSC_MBX   USB_PMSC_TSK      /* Mailbox ID */
```

11. 制限事項

このPMSCファームウェアは、FATをサポートしていませんので、ファイルシステムコール経由でストレージにアクセスすることはできません。PMSCデバイスからストレージにアクセスするためには、このPMSCファームウェアにFATドライバを追加する必要があります。

12. e² studio 用プロジェクトのセットアップ

(1). e² studio を起動してください。

※ はじめてe² studio を起動する場合、Workspace 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). [ファイル] → [インポート]を選択してください。インポートの選択ダイアログが表示されます。

(3). インポートの選択画面で、 [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。

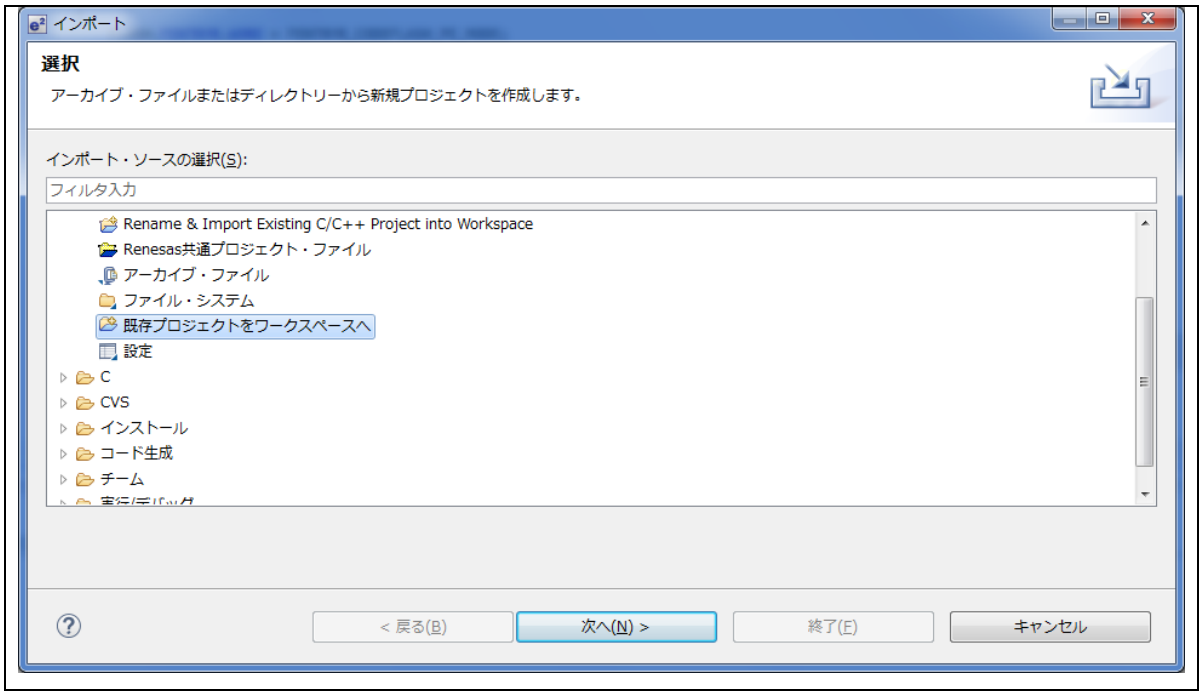

Figure 12-1 インポートの選択

(4). [ルートディレクトリの選択] の [参照] ボタンを押下して、「.cproject 」(プロジェクトファイル)が格納されたフォルダを選択して下さい。

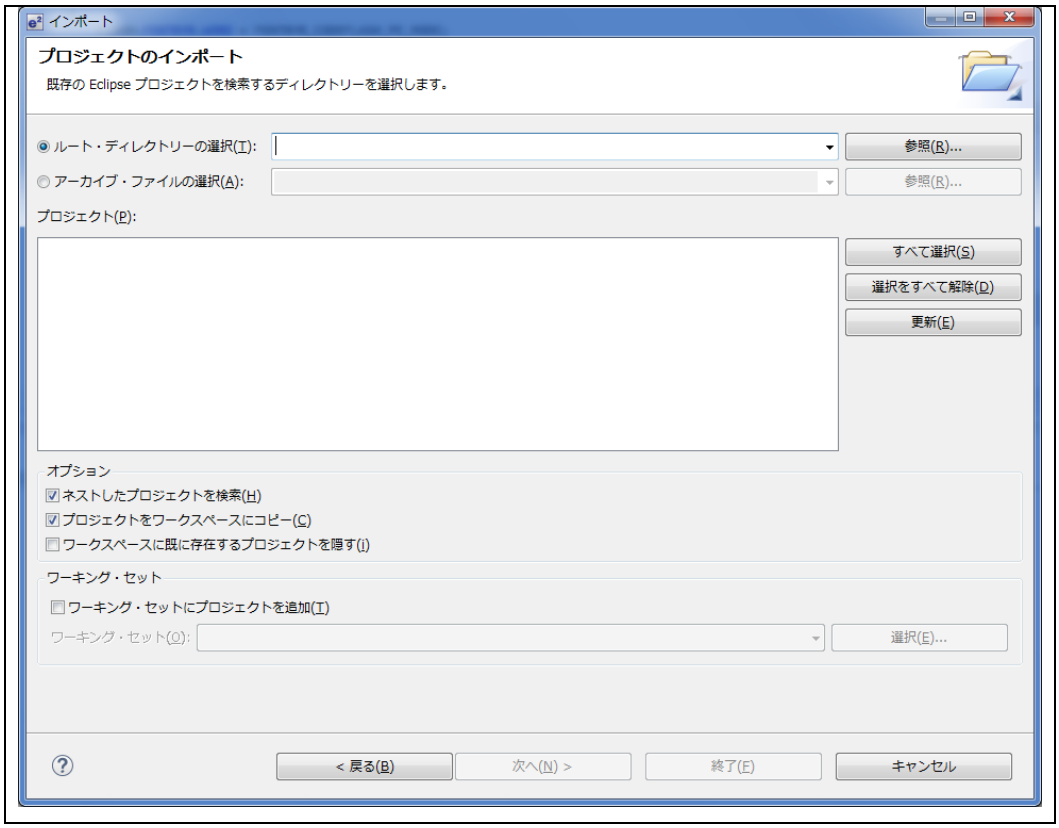

Figure 12-2　プロジェクトのインポート画面

(5). [終了]をクリック して下さい。

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す。

13. e² studio 用プロジェクトを CS+で使用する場合

本プロジェクトは、統合環境 e² studio で作成されています。本プロジェクトを CS+で動作させる場合は、下記の手順を行ってください。

[Note]

rcpc ファイルは、workspace¥RL78¥CCRL¥(MCU 名)フォルダ内に用意されています。

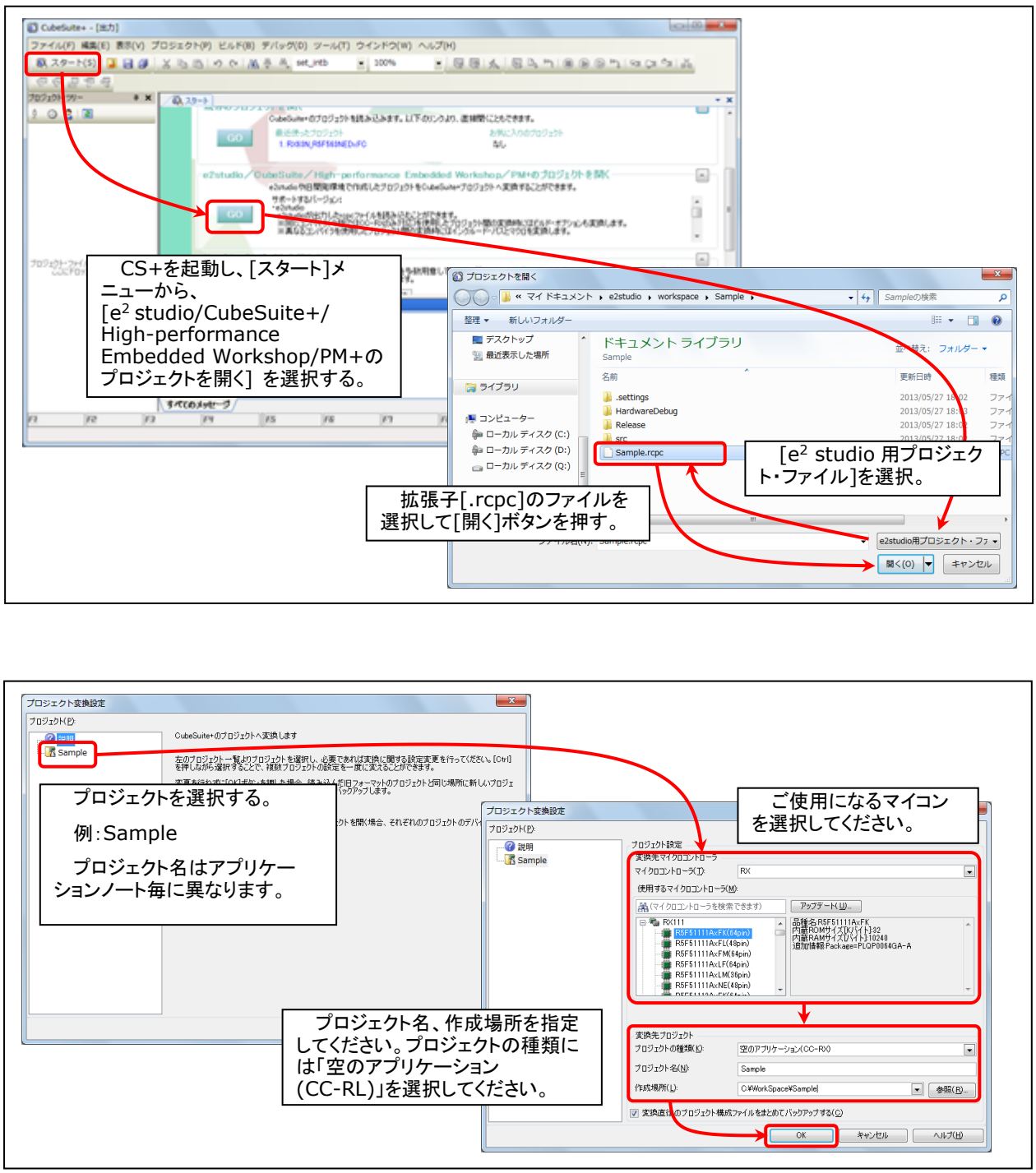

Figure 13-1 e² studio 用プロジェクトの CS+読み込み方法

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/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011/5/31	—	初版発行
2.00	2013/4/5	—	ファームウェアアップデートによるドキュメントの改訂 メディアドライバの章を追加
2.10	2013/8/1	—	サポートデバイスに RX111,RL78/L1C を追加
2.11	2013/10/31	—	2.2.1 フォルダ構成を変更。これに伴い、1.4 のパス表記を修正。
2.12	2014/3/31	—	R8C に対応、誤記訂正
2.13	2015/3/16	—	RX111 を動作確認デバイスから削除。
2.14	2016/1/18	—	Technical Update(発行番号: TN-RL*-A055A/J, TN-RL*-A033B/J)に対応しました。
2.15	2016/3/28	—	CC-RL コンパイラをサポートしました。

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/