RZ/A1H グループ

R01AN3429JJ0110
Rev.1.10
Sep 30, 2016

USB Peripheral Communications Device Class Driver (PCDC)

要旨

本アプリケーションノートでは、USB Peripheral コミュニケーションデバイスクラスドライバ（PCDC）について説明します。本モジュールは USB Basic Firmware(USB-BASIC-FW)と組み合わせることで動作します。以降、本モジュールを USB PCDC と称します。

対象デバイス

RZ/A1H 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. Universal Serial Bus Revision 2.0 specification
2. USB Class Definitions for Communications Devices Revision 1.2
3. USB Communications Class Subclass Specification for PSTN Devices Revision 1.2
   【http://www.usb.org/developers/docs/】
4. 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）
5. RZ/A1H グループ USB Host and Peripheral Interface Driver（ドキュメント No.R01AN3291JJ）
6. 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）
7. RZ/A1H グループレジスタ定義ヘッダ・ファイル iodefine.h（R01AN1860JJ）
8. RZ/A1H グループ初期設定例（R01AN1864JJ）

— 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

PCDC は、USB-BASIC-FWと組み合わせることで、USB Peripheral コミュニケーションデバイスクラスドライバ（以降 PCDC と記述）として動作します。PCDC は、USB コミュニケーションデバイスクラス仕様（以降 CDC と記述）の Abstract Control Model に準拠し、USB ホストとの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 ホストとのデータ転送
- CDC クラスリクエストに応答
- コミュニケーションデバイスクラスノーティフィケーション送信サービスの提供

1.1 必ずお読みください

お客様が、このドライバを使ってアプリケーションプログラムを作成する場合は、ドキュメント(Document No:R11AN3291JJ)に記載された API の使用をお勧めします。当該ドキュメントは、パッケージ内の "reference_documents"フォルダにあります。

[Note]

1. ドキュメント(Document No:R11AN3291JJ)に記載された API を使ったアプリケーションプログラムの作成方法は当該ドキュメントに記載されています。
2. ドキュメント(Document No:R11AN0022JJ)に記載された API を使用した場合、本書の「6.2 PCDC API 一覧」に記載された API を使用する必要はありません。

1.2 動作確認済環境

PCDC の動作確認済環境をTable 1.1に示します。

Table 1.1 動作確認条件

項目	内容
使用マイコン	RZ/A1H
動作周波数（注）	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®t統合開発環境
	・ARM Development Studio（DS-5™） Version 5.16
	IARt 統合開発環境
	・IAR Embedded Workbench for ARM Version 7.40
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）
ターミナルソフトの通信設定	・通信速度 ：115200bps
	・データ長 ：8 ビット
	・パリティ ： なし
	・ストップビット長 ：1 ビット
s	・フロー制御 ： なし
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）	・シリアルインターフェース（Dsub-9 コネクタ）
	・USB1 コネクタ、USB2 コネクタ

1.3 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

ACM	:	Abstract Control Model.
APL	:	Application program
CDC	:	Communications Devices Class
CDCC	:	Communications Devices Class Communications Interface Class
CDCD	:	Communications Devices Class Data Class Interface
CPD	:	Serial Communication Port Driver
cstd	:	Peripheral & Host USB-BASIC-FW用の関数およびファイルのプレフィックス
non-OS	:	USB basic firmware for OS less system
PCD	:	Peripheral control driver of USB-BASIC-FW
PCDC	:	Periperal用 Communications Devices Class
PCDCD	:	Periperal Communications Devices Class Driver
PDCD	:	Peripheral device class driver (device driver and USB class driver)
PP	:	プリプロセス定義
PSTN	:	Public Switched Telephone Network, contains the ACM (above) standard.
Scheduler	:	non-OSでタスク動作を簡易的にスケジューリングするもの
Scheduler Macro	:	non-OSでスケジューラ機能を呼び出すために使用されるもの
Task	:	処理の単位
USB	:	Universal Serial Bus
USB-BASIC-FW	:	USB basic firmware for RZ/A1H グループ (non-OS/ RTOS)

2. ソフトウェア構成

Figure 2-1に PCDC のモジュール構成、Table 2.1にモジュール機能概要を示します。

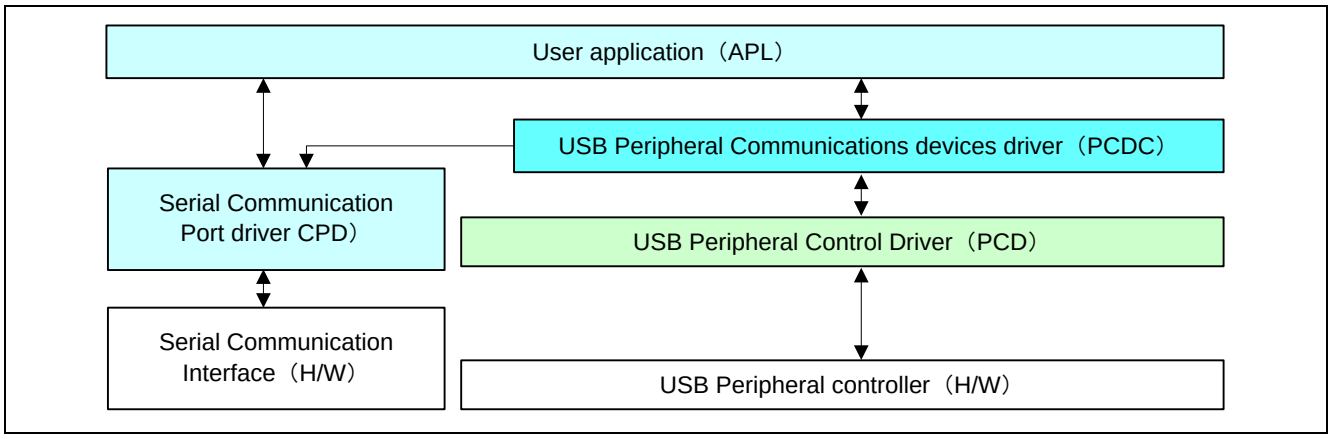

Figure 2-1 モジュール構成図

Table 2.1 各モジュール機能概要

モジュール名	機能概要
PCDC	APL からの CDC に関するリクエストおよび、データ通信を PCD へ要求します。
PCD	USB Peripheral H/W 制御ドライバです。
CPD	シリアルポートの制御ドライバです。

3. システム資源

Table 3.1～Table 3.3に、PCDC が使用しているシステム資源を示します。

Table 3.1 タスク情報

関数名	タスク ID	優先度	概要
usb_pcdc_Task	USB_PCDC_TSK	USB_PRI_3	PCDC タスク

Table 3.2 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_PCDC_MBX	USB_PCDC_TSK	FIFO 順	PCDC 用メールボックス

Table 3.3 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_PCDC_MPL	FIFO 順	40byte	PCDC 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 20 です。

4. コンパイル時の設定

本プロジェクトを使用する場合、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、ドキュメント(Document No:R01AN3291JJ)を参照してください。
本モジュールのコンフィギュレーションオプションの設定は、r_usb_pcdc_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_pcdc_config.h	
USB_PCDC_USE_PIPE_IN USB_PCDC_USE_PIPE_OUT	データ転送で使用されるパイプ番号を指定してください。(USB_PIPE1 から USB_PIPE5 のうちいずれかを指定してください。)
USB_PCDC_USE_PIPE_STATUS	クラスノーティフィケーション通知で使用されるパイプ番号を指定してください。(USB_PIPE6 から USB_PIPE9 のうちいずれかを指定してください。)
USB_UART_ENABLE	UART を使用する場合に、本定義を有効にしてください。

5. コミュニケーションデバイスクラス（CDC）、PSTN and ACM

5.1 基本機能

CDC は、コミュニケーションデバイスクラス仕様 Abstract Control Model サブクラスに準拠しています。CDC の主な機能を以下に示します。

1. USB ホストからの機能照会に対する応答
2. USB ホストからのクラスリクエストに対する応答
3. USB ホストとのデータ通信
4. USB ホストへのシリアル通信エラー報告

5.2 Abstract Control Model 概要

Abstract Control Model サブクラスは、USB 機器と従来のモデム（RS-232C 接続）との間を埋める技術で、従来のモデムを使用するアプリケーションプログラムが使用可能です。
以下に本 S/W でサポートするクラスリクエスト・クラスノーティフィケーションを記します。

5.2.1 クラスリクエスト（ホスト→デバイスへの通知）

PCDC で対応しているクラスリクエストをTable 5.1に示します。

Table 5.1 CDC クラスリクエスト

リクエスト	コード	説明	対応
SendEncapsulatedCommand	0x00	プロトコルで定義された AT コマンド等を送信する。	×
GetEncapsulatedResponse	0x01	SendEncapsulatedCommand で送信したコマンドに対するレスポンスを要求する。	×
SetCommFeature	0x02	機器固有の 2 バイトコードや、カントリー設定の禁止／許可を設定する。	×
GetCommFeature	0x03	機器固有の 2 バイトコードや、カントリー設定の禁止／許可状態を取得する。	×
ClearCommFeature	0x04	機器固有の 2 バイトコードや、カントリー設定の禁止／許可設定をデフォルト状態に戻す。	×
SetLineCoding	0x20	通信回線設定を行う。（通信速度、データ長、パリティビット、ストップビット長）	○
GetLineCoding	0x21	通信回線設定状態を取得する。	○
SetControlLineState	0x22	通信回線制御信号 RTS、DTR の設定を行う。	○
SendBreak	0x23	ブレイク信号の送信を行う。	×

Abstract Control Model リクエストについては、USB Communications Class Subclass Specification for PSTN Devices Revision 1.2 の Table11：Requests-Abstract Control Model を参照して下さい。

5.2.2 クラスリクエストのデータフォーマット

CDC が対応するクラスリクエストのデータフォーマットを以下に記します。

(1). SetLineCoding

UART 回線設定を行う為にホストがデバイスに対して送信するクラスリクエストです。
SetLineCoding データフォーマットを以下に示します。

Table 5.2 SetLineCoding フォーマット

bmRequestType t	bReques	wValue	wIndex	wLength	Data
0x21	SET_LINE_CODING (0x20)	0x00	0x00	0x07	Line Coding Structure Table 5.3 Line Coding Structure フォーマット参照

Table 5.3 Line Coding Structure フォーマット

Offset	Field	Size	Value	Description
0	DwDTERate	4	Number	データ端末の速度（bps）
4	BcharFormat	1	Number	ストップビット 0 - 1 Stop bit 1 - 1.5 Stop bit 2 - 2 Stop bit
5	BparityType	1	Number	パリティ 0 - None 1 - Odd 2 - Even
6	BdataBits	1	Number	データビット （5、6、7、8）

(2). GetLineCoding

UART 回線設定状態を要求する為にホストがデバイスに対して送信するクラスリクエストです。
GetLineCoding データフォーマットを以下に示します。

Table 5.4 GetLineCoding フォーマット

bmRequestType t	bReques	wValue	wIndex	wLength	Data
0xA1	GET_LINE_CODING (0x21)	0x00	0x00	0x07	Line Coding Structure Table 5.3 Line Coding Structure フォーマット参照

(3). SetControlLineState

UART のフロー制御用信号を設定する為にホストがデバイスに対して送信するクラスリクエストです。
本 S/W では RTS/DTR の制御をサポートしていません。
SET_CONTROL_LINE_STATE データフォーマットを以下に示します。

Table 5.5 SET_CONTROL_LINE_STATE フォーマット

bmRequestType t	bReques	wValue	wIndex	wLength	Data
0x21	SET_CONTROL_ LINE_STATE (0x22)	Control Signal Bitmap Table 5.6 Control Signal Bitmap フォーマット参照	0x00	0x00	None

Table 5.6 Control Signal Bitmap フォーマット

Bit Position	Description
D15～D2	予約（0 にリセット）
D1	DCE の送信機能を制御 0 - RTS OFF 1 - RTS ON
D0	DTE がレディ状態かの通知 0 - DTR OFF 1 - DTR ON

5.2.3 クラスノーティフィケーション（デバイス→ホストへの通知）

本 S/W のクラスノーティフィケーション対応/非対応をTable 5.7に示します。

Table 5.7 CDC クラスノーティフィケーション

ノーティフィケーション	コード	説明	対応
NETWORK_CONNECTION	0x00	ネットワーク接続状況を通知する	×
RESPONSE_AVAILABLE	0x01	GET_ENCAPSLATED_RESPONSE への応答	×
SERIAL_STATE	0x20	シリアル回線状態を通知する	○

(1). SerialState

UART ポートに状態変化を検出した場合、ホストへ状態通知を行います。

本 S/W ではオーバーランエラー、パリティエラー、フレーミングエラー検出をサポートしています。状態通知は正常状態からエラー検出した場合に行います。エラーを連続検出しても状態通知を連続送信しません。

SerialState データフォーマットを以下に示します。

Table 5.8 SerialState フォーマット

bmRequestType t	bReques	wValue	wIndex	wLength	Data
0xA1	SERIAL_STATE (0x20)	0x00	0x00	0x02	UART State bitmap Table 5.9 UART State bitmap フォーマット参照

Table 5.9 UART State bitmap フォーマット

Bits	Field	Description	対応
D15～D7		予約	―
D6	bOverRun	オーバーランエラー検出	○
D5	bParity	パリティエラー検出	○
D4	bFraming	フレーミングエラー検出	○
D3	bRingSignal	着信（Ring signal）を感知した	×
D2	bBreak	ブレーク信号検出	×
D1	bTxCarrier	Data Set Ready：回線が接続されて通信可能	×
D0	bRxCarrier	Data Carrier Detect：回線にキャリア検出	×

5.3 PC の仮想 COM ポートについて（参考）

Windows OS 搭載 PC は CDC デバイスを仮想 COM ポートとして利用することが可能です。

Windows OS 搭載 PC に本 S/W を実装した RSK ボードを接続すると、エニュメレーション設定に続き、CDC クラスリクエストの GetLineCoding 及び SetControlLineState を行った後、仮想 COM デバイスとしてデバイスマネージャに登録されます。Windows デバイスマネージャに仮想 COM ポートとして登録された後は、WindowsOS 標準搭載のハイパーターミナル等のターミナルアプリで CDC デバイスとデータ通信が可能です。

ターミナルアプリのシリアルポート設定を行うことで、クラスリクエスト SetLineCoding による UART 設定が可能です。ターミナルアプリのウインドウから入力したデータ（又はファイル送信）は EP2 を使用して評価ボードへ転送され、評価ボード側から PC へのデータ転送は EP1 を使用して行われます。

ターミナルアプリによっては最後に受信したデータが MAX パケットサイズ（Full-Speed:64Byte）の場合、継続するデータがあると判断して受信データをターミナルに表示しないことがあります。この場合、MAX パケットサイズ未満のデータを受信することで、それまでに受信したデータがターミナルに表示されます。

6. USB ペリフェラルコミュニケーションデバイスクラスドライバ（PCDC）

6.1 基本機能

PCDC の基本機能を以下に示します。

1. USB ホストとのデータ転送
2. CDC クラスリクエストに応答
3. コミュニケーションデバイスクラスノーティフィケーション送信サービスの提供

6.2 PCDC API 一覧

Table 6.1に PCDC API 一覧を示します。

[Note]

1. ドキュメント(Document No: R01AN329JJ)に記載された API を使用する場合、アプリケーションプログラムでは、以下の API を使用する必要はありません。

Table 6.1 PCDC API 一覧

関数名	機能概要
R_usb_pcdc_SendData	USB 送信処理
R_usb_pcdc_ReceiveData	USB 受信処理
R_usb_pcdc_SerialStateNotification	USB ホスト装置へクラスノーティフィケーション SerialState を送信
R_usb_pcdc_usr_ctrl_trans_function	CDC 用コントロール転送処理
R_usb_pcdc_task	PCDC タスク
R_usb_pcdc_driver_start	PCDC ドライバ起動

6.2.1 R_usb_pcdc_SendData

USB 送信処理

形式

void	R_usb_pcdc_SendData(USB_UTR_t *ptr, uint8_t *buf, uint32_t size, USB_CB_t complete)

引数

*ptr	USB 通信用構造体
*buf	転送データアドレス
size	転送サイズ
complete	処理完了通知コールバック関数

戻り値

―	―

解説

転送データアドレス buf で指定されたアドレスから、転送サイズ size 分のデータを USB 送信します。
送信完了後、コールバック関数 complete が呼出されます。

補足

1. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。
   USB_REGADR_t ipp : USB IP のアドレス
   uint16_t ip : USB IP 番号
2. USB 送信処理結果はコールバック関数の引数”USB_UTR_t *mess”で得られます。
3. 第 2 引数には自動変数(スタック)領域以外の領域を指定してください。
4. USB Basic Firmware アプリケーションノートの USB 通信用構造体（USB_UTR_t 構造体）を参照して下さい。

使用例

```
  #define USB_PERI_USBIP_NUMUSB_USBIP_0
// #define  USB_PERI_USBIP_NUMUSB_USBIP_1

{
  USB_UTR_t  utr;
  USB_UTR_t  *ptr;
  uint16_t   size = 5;                              /* USB 送信データ数 */

  ptr = (USB_UTR_t *)&utr;
  ptr->ip = USB_PERI_USBIP_NUM;                     /* USB IP 番号設定 */
  ptr->ipp = R_usb_cstd_GetUsbIpAdr( ptr->ip );  /* USB IP ベースアドレス取得 */

  R_usb_pcdc_SendData(ptr, (uint8_t *)send_data, size, (USB_CB_t)&usb_complete)
}

/* USB 送信完了通知用コールバック関数 */
void  usb_complete( USB_UTR_t *mess, uint16_t data1, uint16_t data2 );
{
  /* USB 送信完了時の処理を記述して下さい。 */
}
```

6.2.2 R_usb_pcdc_ReceiveData

USB 受信処理

形式

void R_usb_pcdc_ReceiveData (USB_UTR_t *ptr, uint8_t *buf, uint32_t size, USB_CB_t complete)

引数

*ptr	USB 通信用構造体
*buf	転送データアドレス
size	転送サイズ
complete	処理完了通知コールバック関数

戻り値

—	—

解説

USB 受信要求を行います。
USB から転送サイズ size 分のデータ受信完了、又は MAX パケットサイズ未満のデータを受信した場合、コールバック関数 complete が呼出されます。
USB 受信データは、転送データアドレス buf で指定されたアドレスで指定された領域に格納されます。

補足

1. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。
   USB_REGADR_t ipp : USB IP のアドレス
   uint16_t ip : USB IP 番号
2. USB 受信処理結果はコールバック関数の引数”USB_UTR_t *mess”で得られます。
3. 第 2 引数には自動変数(スタック)領域以外の領域を指定してください。
4. 受信したデータが MaxPacketSize の n 倍、かつ引数 size に指定したサイズに満たない場合は、データ転送の途中であると判断しコールバック関数(complete)は発生しません。
5. USB Basic Firmware アプリケーションノートの USB 通信用構造体（USB_UTR_t 構造体）を参照して下さい。

使用例

```
  #define USB_PERI_USBIP_NUM USB_USBIP_0
// #define  USB_PERI_USBIP_NUM USB_USBIP_1

{
  USB_UTR_t  utr, *ptr;

  ptr = (USB_UTR_t *)&utr;
  ptr->ip = USB_PERI_USBIP_NUM;                   /* USB IP 番号設定 */
  ptr->ipp = R_usb_cstd_GetUsbIpAdr( ptr->ip );  /* USB IP ベースアドレス取得 */

  R_usb_pcdc_ReceiveData(ptr, (uint8_t *)receive_data, size,
(USB_CB_t)&usb_complete)
}

/* USB 受信完了通知用コールバック関数 */
void  usb_complete( USB_UTR_t *mess, uint16_t data1, uint16_t data2 );
{
  /* USB 受信完了時の処理を記述して下さい。 */
}
```

6.2.3 R_usb_pcdc_SerialStateNotification

USB ホスト装置へクラスノーティフィケーション SerialState を送信する

形式

void	R_usb_pcdc_SerialStateNotification(USB_UTR_t *ptr, uint16_t serial_state, USB_CB_t complete)

引数

*ptr	USB 通信用構造体
serial_state	シリアルステータス
complete	処理完了通知コールバック関数

戻り値

―	―

解説

CDC クラスノーティフィケーション・シリアルステートを USB ホストに送信します。
シリアルステート送信はインタラプトパイプ EP3 を使用します。
送信完了後、コールバック関数 complete が呼出されます。

補足

1. シリアルステータスのビットパターンは"Table 5.9 UART State bitmap フォーマット"を参照してください。
2. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。
   USB_REGADR_t ipp : USB IP のアドレス
   uint16_t ip : USB IP 番号
3. USB 送信処理結果はコールバック関数の引数”USB_UTR_t *mess”で得られます。
4. USB Basic Firmware アプリケーションノートの USB 通信用構造体（USB_UTR_t 構造体）を参照して下さい。

使用例

```
  #define USB_PERI_USBIP_NUMUSB_USBIP_0
// #define  USB_PERI_USBIP_NUMUSB_USBIP_1

{
  USB_UTR_t  utr;
  USB_UTR_t  *ptr;
  uint16_t   state;       /* シリアルステータス */

  ptr = (USB_UTR_t *)&utr;
  ptr->ip = USB_PERI_USBIP_NUM;                   /* USB IP 番号設定 */
  ptr->ipp = R_usb_cstd_GetUsbIpAdr( ptr->ip );  /* USB IP ベースアドレス取得 */

  state = 0x0020;         /* D5:パリティエラー */
  R_usb_pcdc_SerialStateNotification(ptr, state, (USB_CB_t)&usb_complete);
}

/* シリアルステート送信完了通知用コールバック関数 */
void  usb_complete( USB_UTR_t *mess, uint16_t data1, uint16_t data2 )
{
  /* シリアルステート送信完了時の処理を記述して下さい。 */
}
```

6.2.4 R_usb_pcdc_usr_ctrl_trans_function

CDC 用コントロール転送処理

形式

void R_usb_pcdc_usr_ctrl_trans_function(USB_UTR_t *ptr, USB_REQUEST_t *preq, uint16_t ctsq)

引数

*ptr	USB 通信用構造体
*preq	クラスリクエストメッセージへのポインタ
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

リクエストタイプが CDC 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。
本 API をデバイスクラスドライバ・レジストレーションでコントロール転送時に呼出されるコールバック関数として USB_PCDREG_t 構造体メンバ *ctrltrans* に登録してください。

補足

―

使用例

```
USB_PCDREG_t  driver;

/* Control Transfer */
driver.ctrltrans = (USB_CB_TRN_t)&R_usb_pcdc_usr_ctrl_trans_function;
R_usb_pstd_DriverRegistration(ptr, &driver);
```

6.2.5 R_usb_pcdc_Task

PCDC タスク

形式

void R_usb_pcdc_Task(USB_VP_INT_t stacd)

引数

stacd タスクスタートコード（非使用）

戻り値

― ―

解説

PCDC 処理タスク。
アプリから要求された処理を行い、アプリに処理結果を通知します。

補足

1. 本 API はユーザプログラムで呼び出してください。
2. non-OS で動作させる場合、タスクスイッチング処理から本 API がスケジュールされるように登録します。

使用例

```
void usb_apl_task_switch(void)
{
  while( 1 )
  {
    /* Scheduler */
    R_usb_cstd_Scheduler();

    if( USB_FLGSET == R_usb_cstd_CheckSchedule() )
    {
       R_usb_pstd_PcdTask((USB_VP_INT)0);        /* PCD Task */
       /* Peripheral Communications Devices Class Task */
       R_usb_pcdc_Task(0);
       /* Peripheral Communications Class Application Task */
       usb_pcdc_main_task(0);
    }
  }
}
```

6.2.6 R_usb_pcdc_driver_start

PCDC ドライバ起動

形式

void R_usb_pcdc_driver_start(USB_UTR_t *ptr)

引数

*ptr USB 通信構造体

戻り値

― ―

解説

PCDC ドライバタスクの優先度を設定します。
優先度が設定されることで、メッセージの送受信が可能になります。

補足

1. 初期設定時にユーザアプリケーションで呼び出してください。
2. 本関数コール前に USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。

USB_REGADR_t	ipp	: USB IP のアドレス
uint16_t	ip	: USB IP 番号

使用例

```
void usb_apl( void )
{
  USB_UTR_t  *ptr;
    :
  ptr->ip = USB_PERI_USBIP_NUM;      /* USB IP 番号設定 */
  ptr->ipp = R_usb_cstd_GetUsbIpAdr( ptr->ip );  /* USB IP アドレス設定 */
    :
  R_usb_pcdc_driver_start( ptr );/* Peripheral Class Driver Task Start Setting */
   :
}
```

7. サンプルアプリケーション

7.1 アプリケーション仕様

PCDC のサンプルアプリケーション(以降、APL) の主な機能を以下に示します。

1. ループバックモード（Echo モード）
   USB ホストから受信したデータを、USB ホストへ送信します。

7.2 アプリケーション処理概要

APL は、初期設定、メインループの 2 つの部分から構成されます。以下にそれぞれの処理概要を示します。

7.2.1 初期設定

初期設定では、USB コントローラの初期設定およびアプリケーションプログラムの初期化処理を行います。

7.2.2 メインループ

このメインループでは、CDC デバイスから受信したデータをそのまま CDC デバイスへ送信するループバック処理をメインに行います。以下にメインループの処理概要を示します。

1. USB Host との Enumeration 完了後に R_USB_GetEvent 関数をコールすると戻り値に USB_STS_CONFIGURED がセットされます。APL では、USB_STS_CONFIGURED を確認すると R_USB_Read 関数をコールし、USB Host から送信されるデータのデータ受信要求を行います。
2. USB Host からのデータ受信が完了し、R_USB_GetEvent 関数をコールすると戻り値に USB_STS_READ_COMPLETE がセットされます。APL では、USB_STS_READ_COMPLETE を確認すると R_USB_Write 関数をコールし、受信データを USB Host に送信するためのデータ送信要求を行います。
3. USB Host へのデータ送信が完了し、R_USB_GetEvent 関数をコールすると戻り値に USB_STS_WRITE_COMPLETE がセットされます。APL では、USB_STS_WRITE_COMPLETE を確認すると R_USB_Read 関数をコールし、USB Host から送信されるデータのデータ受信要求を行います。
4. 上記2と3の処理が繰り返し行われます。

Figure 7-1に、APL の処理概要を示します。

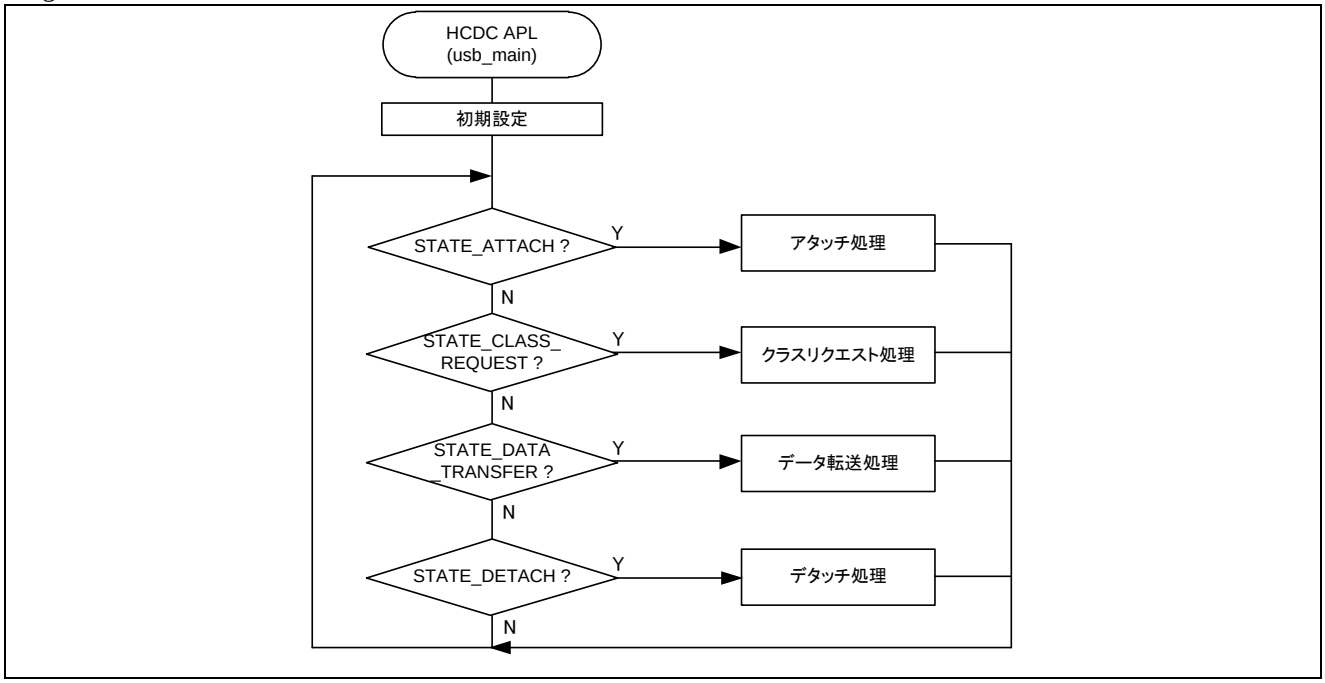

Figure 7-1 メインループ処理

8. セットアップ

8.1 ハードウェア

PCDC の動作環境例をFigure 8-1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。

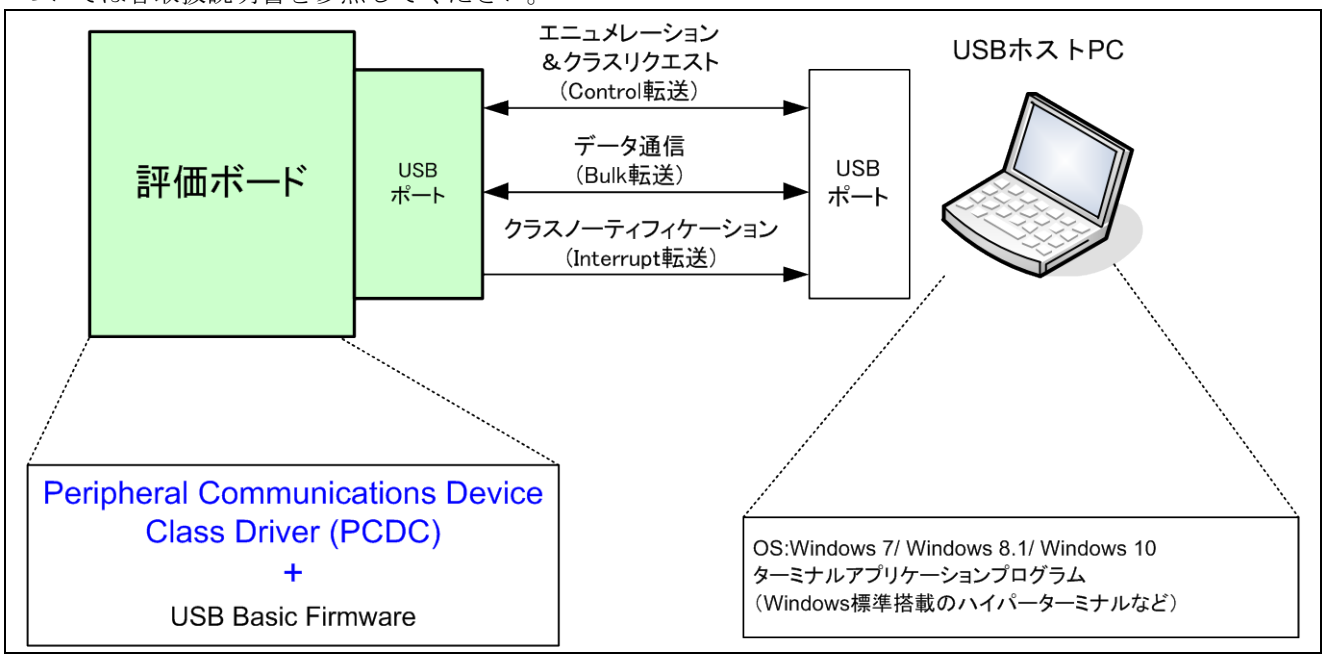

Figure 8-1 動作環境例

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2016.09.01	—	初版発行
1.10	2016.09.30	—	USB-BASIC-F/W が改定されたため、バージョンアップ
		20	「8. セットアップ」追加

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

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. リザーブアドレスのアクセス禁止

   【注意】リザーブアドレス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、事前に問題ないことをご確認ください。

   同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。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。

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/