Renesas USB MCU

USB Host Communication Device Class Driver (HCDC) using Basic Mini Firmware

R01AN0643JJ0215
Rev.2.15
Mar 28, 2016

要旨

本資料は、Renesas USB MCU の USB Basic Mini Firmware を使用した USB Host Communication Device Class Driver (HCDC)のアプリケーションノートです。

動作確認デバイス

RL78/G1C, 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

本アプリケーションノートは、USB-BASIC-F/W (1.2 章を参照) を使用した USB Host Communication Device Class Driver (HCDC) および、サンプルアプリケーションに関して記述しています。

1.1 機能と特長

USB Host Communication Device Class Driver (HCDC) 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（詳細は PSTN デバイス・サブクラスを参照）に準拠し、CDC ペリフェラルデバイスとの通信を行うことができます。

本クラスドライバは弊社の提供する USB Basic Mini Firmware と組み合わせて使用することを前提にしています。

1.2 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. 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
   ルネサス エレクトロニクスホームページ　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 用語と略語

本書では使用される用語と略語は以下のとおりです。

API	:	Application Program Interface
APL	:	Application program
ACM	:	Abstract Control Model. This is the USB interface subclass used for virtual COM ports, based in the old V.250 (AT) command standard. See PSTN below
CDC	:	Communications devices class
CDCC	:	Communications Devices Class － Communications Class Interface
CDCD	:	Communications Devices Class － Data Class Interface
cstd	:	USB-BASIC-F/Wの Peripheral & Host共通関数のprefix
Data Transfer	:	Generic name of Control transfer, Bulk transfer and Interrupt transfer
HCD	:	Host control driver of USB-BASIC-F/W
HCDC	:	USB Host Communication Device Class Driver (HCDC)
hcdc	:	HCDCの関数及びファイルのprefix
HDCD	:	Host device class driver (device driver and USB class driver)
HEW	:	High-performance Embedded Workshop
HM	:	Hardware Manual
hstd	:	USB-BASIC-F/WのHost関数のprefix
MGR	:	Peripheral device state manager of HCD
PP	:	プリプロセス定義
PSTN	:	Public Switched Telephone Network. Contains the ACM (above) standard. See also Chapter 1.2
RSK	:	Renesas Starter Kit
SW1/SW2/SW3	:	RSKに実装された3つのスイッチ
USB	:	Universal Serial Bus
USB-BASIC-FW	:	USB Basic Mini Firmware (Peripheral & Host USB Basic Mini Firmware(USB low level) for Renesas USB MCU)
タスク	:	処理の単位
スケジューラ	:	タスク動作を簡易的にスケジューリングするもの
スケジューラマクロ	:	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

本書は章の順番通りに読み進める必要はありません。はじめにサンプルプログラムの内容を確認し、ユーザ個別のソリューションに必要な関数およびインタフェースの情報をお読みください。

4.3 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。

ユーザ独自のソリューションを作成するためにはアプリケーションの変更が必要です。5 章はホスト CDC アプリケーションの動作を説明しています。

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こ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"USB-BASIC-F/W Application Note"で説明しています。HCDC のタスクについては 4.4 章を参照してください。

2. デバイスクラスドライバの登録

ユーザが作成したクラスドライバは、USB-BASIC-F/W に登録することで USB デバイスクラスドライバとして機能します。*r_usb_hcdc_apl.c* ファイル内の *usb_hapl_registration()* 関数を参考に USB-BASIC-F/W にクラスドライバを登録してください。詳細は、USB-BASIC-F/W のアプリケーションノートを参照してください。

3. 動作確認済環境

3.1 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

3.2 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. R8C/34K Group USB Host 評価ボード(型名: R0K5R8C34DK2HBR)

4. ソフトウェア構成

4.1 モジュール構成

Figure 4.1 に HCDC のソフトウェアモジュール構成図を、Table 4-1 にモジュール説明を示します。

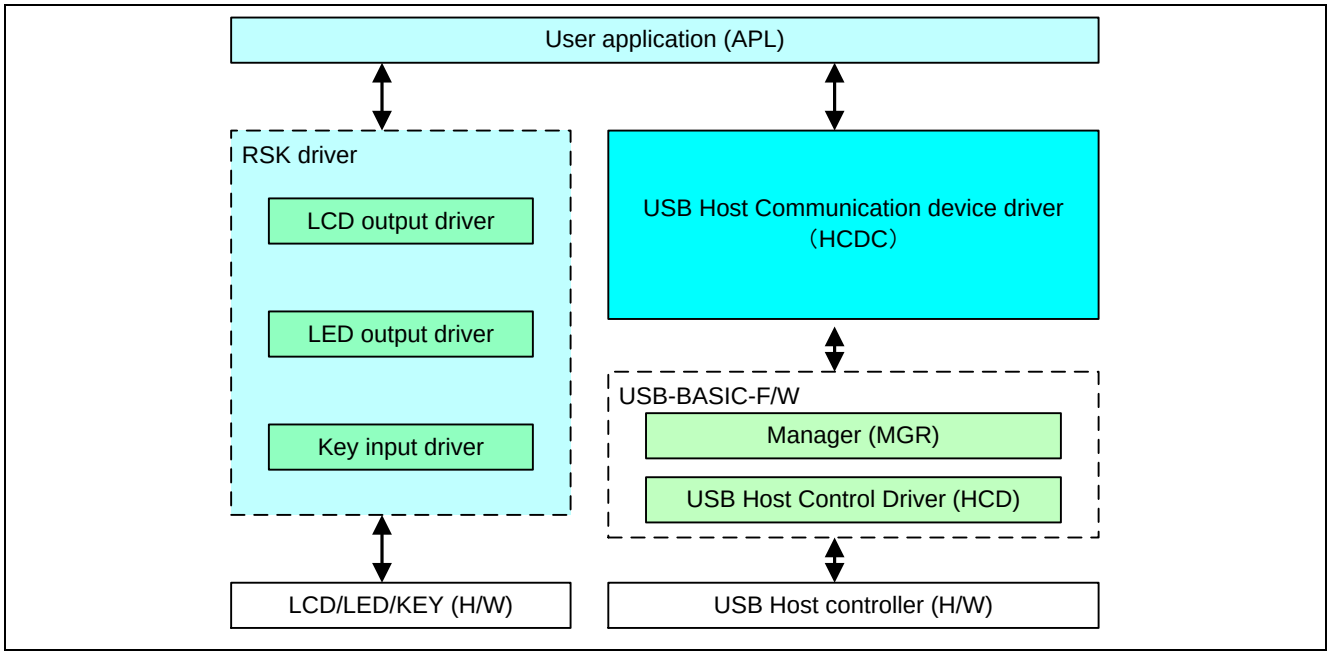


Figure 4.1 モジュール構成図

Table 4-1 モジュール機能説明

モジュール名	機能概要	備考
APL	ユーザアプリケーションのサンプルプログラムです。 スイッチ操作により CDC デバイスの CUART 設定を変更します。 CDC デバイスから受信した UART ステートを LCD に表示します。	ユーザ作成
HCDC	登録されたデバイスクラスドライバは接続デバイスの操作をチェックします。 USB-BASIC-F/W は APL を経由して接続デバイスと HCDC の対応を確認します。 APL の要求に従って、USB-BASIC-F/W に以下のデータ転送要求を行います。 1) CDC リクエストによる接続デバイスの制御 2) CDC ノティフィケーションによる接続デバイスの状態確認 3) 接続デバイスとのデータ通信 転送結果はコールバック関数によって APL に通知します。	
USB-BASIC-F/W	USB ホストコントロールドライバです。 (ハードウェア制御とデバイスステート管理を行います。)	

4.2 アプリケーションプログラム機能概要

ホストデモ APL の主な機能は以下のとおりです。

1. 接続した USB ペリフェラルデバイスに通信速度などのシリアルステートを設定する。
2. 接続した USB ペリフェラルからデータを受け取り、受信したデータを USB ペリフェラルに送信するループバック動作を行う。
3. USB 評価ボード(RSK)に実装されている SW2、SW3 を使用することで通信速度の変更を行う。データ通信中に通信速度を変更しないでください。

Table 4-2 にスイッチ入力の仕様を示します。

Table 4-2 ユーザスイッチ入力動作

スイッチ名称	動作内容	スイッチ番号
ボーレート選択	通信速度を選択する (1200→2400→4800→….)	SW2
ボーレート設定	SW で選択した通信速度を確定する	SW3

4.3 ファイル構成

4.3.1 フォルダ構成

以下に本デバイスクラスで提供するファイルのフォルダ構成を示します。

各 MCU と評価基板に依存するソースコードはそれぞれのハードウェアリソースフォルダ (***devicename**\src\HwResource)にあります。

```
workspace
  ＋[ RL78 / R8C ]
    ＋[ CCRL / CS+ / IAR / e² studio / HEW ]
      ＋[ RL78G1C / R8C3MK / R8C34K ]
          ＋ HOST                                   ホストビルド結果
          ＋ src
            ＋――― CDCFW [ Communication Device  Class driver ]   Table 4-3 参照
            │       ＋――― inc                     CDC ドライバ共通ヘッダファイル
            │       ＋――― src                     CDC ドライバ
            ＋―――SmplMain [サンプルアプリケーション]
            │       ＋――― APL                     ループバックアプリケーション
            ＋―――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、「10 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。

4.3.2 ファイル一覧

Table 4-3 に HCDC が提供するファイル、フォルダ名を示します。

Table 4-3 ファイル構成

フォルダ名	ファイル名	説明	備考
CDCFW/inc	r_usb_class_usrcfg.h	USB ホスト CDC ユーザ定義マクロ	
CDCFW/inc	r_usb_hcdc_define.h	HCDC 型定義、マクロ定義	
CDCFW/inc	r_usb_hcdc_api.h	HCDC API 関数プロトタイプ宣言	
CDCFW/src	r_usb_hcdc_api.c	HCDC API 関数	
CDCFW/src	r_usb_hcdc_driver.c	HCDC ドライバ関数	
SmplMain	main.c	メインループ処理	
SmplMain/APL	r_usb_hcdc_apl.c	サンプルアプリケーションプログラム	

4.4 システムリソース

4.4.1 システムリソース定義

HCDC をスケジューラに登録して使用するためのタスク ID とタスク優先度定義を Table 4-4 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 4-4 スケジューラ登録 ID 一覧

スケジューラ登録タスク	説明	備考
USB_HCDC_TSK	**HCDC** (R_usb_hcdc_task) Task ID: USB_HCDC_TSK Task priority: 2	
USB_HCDCSMP_TSK	**APL** (usb_hcdc_main_task) Task ID: USB_HCDCSMP_TSK Task priority: 3	
USB_HCD_TSK	**HCD** (R_usb_hstd_HcdTask) Task ID: USB_HCD_TSK Task priority: 0	
USB_MGR_TSK	**MGR** (R_usb_hstd_MgrTask) Task ID: USB_MGR_TSK Task priority: 1	
メールボックス ID / デフォルト受信タスク	**メッセージ名称**	**備考**
USB_HCDC_MBX / USB_HCDC_TSK	HCDC -> HCDC / APL -> HCDC mailbox ID	
USB_HCDCSMP_MBX / USB_HCDCSMP_TSK	HCDC -> APL mailbox ID	
USB_HCD_MBX / USB_HCD_TSK	HCD task mailbox ID	
USB_MGR_MBX / USB_MGR_TSK	MGR task mailbox ID	

5. ホスト CDC サンプルアプリケーションプログラム (APL)

ホストデモアプリケーションは CDC ペリフェラルデバイスが接続されたとき、USB データの”ループバック”通信を行います。HCDC アプリケーションは、USB コミュニケーションデバイスクラス規格の Abstract Control Model サブクラス（USB PSTN デバイスモデル）に従います。1.2 章の 2 項、3 項を参照してください。

5.1 動作環境について

Figure 5.1 に本ソフトウェアの動作環境を示します。PC(図の一番右の箱)にシリアルポートがない場合は 2 台の USB シリアルコンバータを組み合わせて接続することで代用できます。多くの USB シリアルコンバータが CDC デバイスではなく、ベンダクラス(CDC ACM)のタイプであることに注意してください。そのようなコンバータを使用する場合は、5.4 章に示したように HCDC を修正する必要があります。

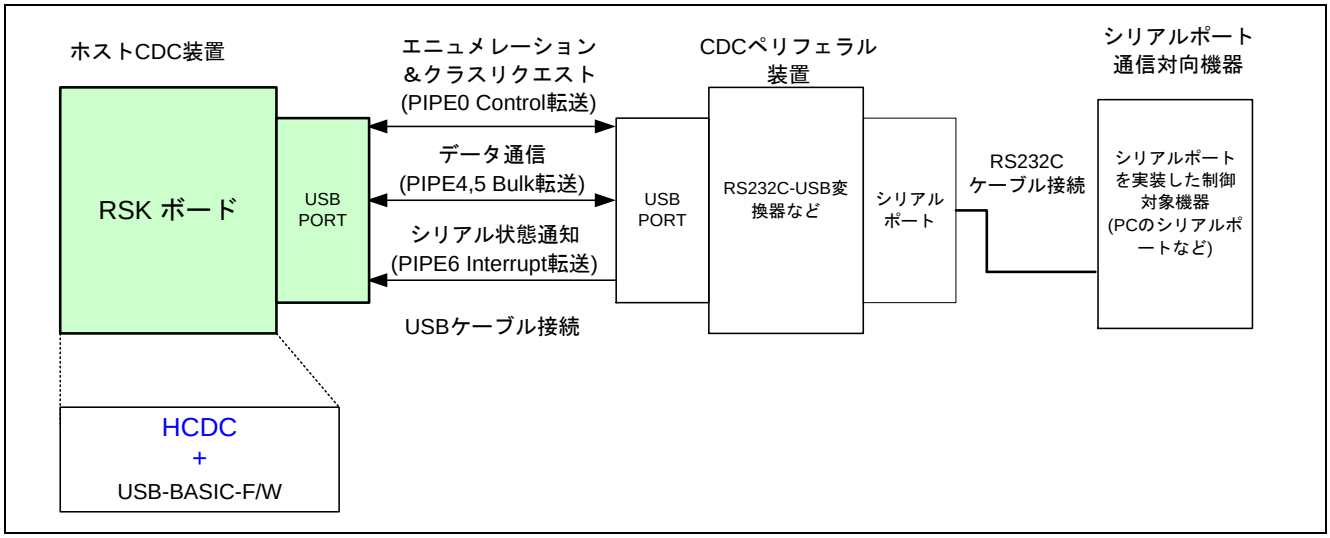


Figure 5.1 動作環境例

5.2 アプリケーションプログラム説明

Figure 5.2 と Figure 5.5 に関するアプリケーション操作は以下のとおりです。

- CDC ペリフェラルデバイスの接続 (No.0-1 処理に対応)
- 接続した CDC ペリフェラルデバイスの初期化(処理 2-1)
  クラスリクエスト SET_CONTROL_LINE_STATE で RTS、DTR の設定を行います。

  クラスリクエスト SET_LINE_CODING で、通信速度、データビット数、ストップビット長、パリティビット設定します。
  クラスリクエスト GET_LINE_CODING で、通信速度、データビット数、ストップビット長、パリティビット設定値を取得します。
- データ転送開始(処理 2-2)
  UART 状態を受け取るコールバック関数の登録(インタラプト IN 転送開始)
  データ受信開始、受信完了(バルク IN 転送開始、コールバック発生)
- 接続デバイスのボーレート変更(処理 4-1)
  スイッチ 2 の操作でボーレートを仮設定(処理 5-1)
  スイッチ 3 の操作でボーレートの決定(処理 5-2)
  　ボーレートが決定されると SET_LINE_CODING リクエストを接続デバイスに通知します
- データ受信完了(処理 4-2)
  HCDC からコールバック関数が発生すると USB デバイスとの通信が終了します

- データ送信開始(処理 6-1)
  受信データを CDC デバイス(ループバック)に伝え、データ転送は完了します。 (バルク OUT 転送開始とコールバック発生)
- データ転送完了(処理 7-2)
  データ受信再開(バルク IN 転送開始)
- シリアルステータス受信完了(処理 4-3)
  HCDC からコールバック関数(*usb_hcdc_smp_SerialStateReceive*)が発生すると USB デバイスとの通信が終了します
- 注意: データ受信、データ送信が失敗されたとき、データ受信を再開します。

5.3 エンドポイント仕様

HCDC が使用するエンドポイント仕様を Table 5-1 に示します。

Table 5-1 エンドポイント仕様

EP 番号	Pipe 番号	転送方法	説明
0	0	Control In/Out	標準リクエスト、クラスリクエスト
受信した ディスクリプタに従う *1	4 or 5	Bulk In	デバイスからホストへのデータ転送
	4 or 5	Bulk Out	ホストからデバイスへのデータ転送
	6	Interrupt In*2	デバイスからホストへの状態通知

Note)
*1 エンドポイント番号はデバイスのエンドポイントディスクリプタに従います。
*2 ベンダクラスの CDC デバイスが接続された場合はインタラプト転送を行いません。

5.4 接続する CDC ペリフェラルデバイスについて

ご使用前に CDC ペリフェラルデバイスの仕様をご確認下さい。ペリフェラル CDC デバイスに市販されている USB－シリアル変換デバイスを使用される場合、インタフェースディスクリプタのインタフェースクラスコードが"communication interface class" ではなく、ベンダクラスになっている場合があります。そのような場合は、CDC 変換機は動作しません。

ベンダクラスの USB－シリアル変換デバイスを接続する場合、以下の変更が必要になります。

File: *r_usb_class_usrcfg.h*

```
/*****************************************
Macro definitions (USER DEFINE)
*****************************************/
/* Select CDC Interface class */
//#define USB_HCDC_IF_CLASS  USB_IFCLS_VEN       /* CDC Device Vendor class */
#define USB_HCDC_IF_CLASS    USB_IFCLS_CDCC      /* CDC Device CDC class */
```

5.5 APL 関数一覧

サンプルアプリケーションの関数一覧を Table 5-2 に示します。

Table 5-2 サンプルアプリケーション関数一覧

関数名	説明
main	メインループ処理
usb_hcdc_main_init	システム初期化 ホスト USB 用の各種タスクスタートアップ処理
usb_hcdc_main_task	HCDC サンプルアプリケーションタスク
usb_hcdc_registration	HCDC ドライバ登録
usb_hsmpl_class_check	接続デバイスチェック
usb_hsmpl_device_state	デバイスステート変化検出コールバック
usb_hcdc_smp_SendEncapsulatedCommand	クラスリクエスト送信: *SendEncapsulatedCommand*
usb_hcdc_smp_GetEncapsulatedResponse	クラスリクエスト送信: *GetEncapsulatedResponse*
usb_hcdc_smp_SetCommFeature	クラスリクエスト送信: *SetCommFeature*
usb_hcdc_smp_GetCommFeature	クラスリクエスト送信: *GetCommFeature*
usb_hcdc_smp_ClrCommFeature	クラスリクエスト送信: *ClearCommFeature*
usb_hcdc_smp_SetLineCoding	クラスリクエスト送信: *SetLineCoding*
usb_hcdc_smp_GetLineCoding	クラスリクエスト送信: *GetLineCoding*
usb_hcdc_smp_SendBreak	クラスリクエスト送信: *SendBreak*
usb_hcdc_smp_SetControlLineState	クラスリクエスト送信: *SetControlLineState*
usb_hcdc_smp_SerialStateReceive	Interrupt-IN 通知を受けるコールバック関数
usb_hcdc_smp_InTransResult	Bulk-IN 転送完了を受けるコールバック関数
usb_hcdc_smp_OutTransResult	Bulk-OUT 転送完了を受けるコールバック関数
usb_hcdc_smp_crass_request_result	クラスリクエスト送信完了を受けるコールバック関数
usb_hcdc_smp_init	サンプルアプリケーションタスクに初期化要求メッセージを送信する関数
usb_hcdc_sw_request	スイッチ入力確認コマンド発行
usb_hcdc_sw_process	スイッチ入力操作
usb_hcdc_get_line_coding_rcv_process	*GetLineCoding* リクエスト受信処理
usb_hcdc_smpl_message_send	サンプルアプリへメッセージ通知
usb_hsmpl_transfer_result	サンプルアプリへ転送結果通知
usb_hsmpl_dummy_fnc	空関数

5.6 ホストアプリケーションタスクシーケンス

LCD 表示、APL 状態遷移、操作説明を以下に示します。

5.6.1 APL 状態遷移

Figure 5.2 にアプリケーション状態遷移を示しています。各ブロックは状態遷移プログラムです。

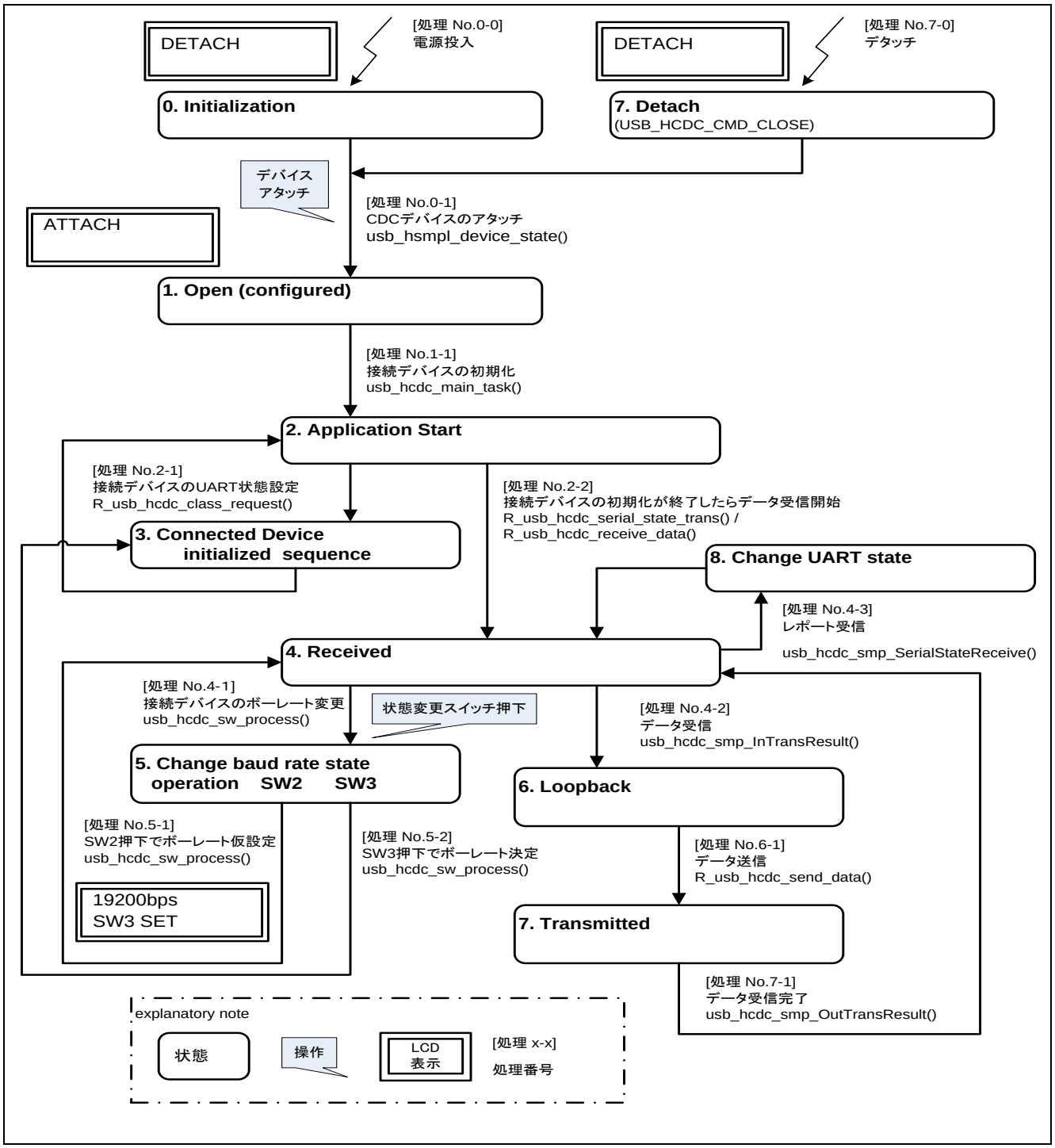


Figure 5.2 Application State Transitions

5.7 処理フロー図

Figure 5.3 に、APL タスク処理概要フローを示します。コマンド処理の詳細について Table 5-3 を参照してください。

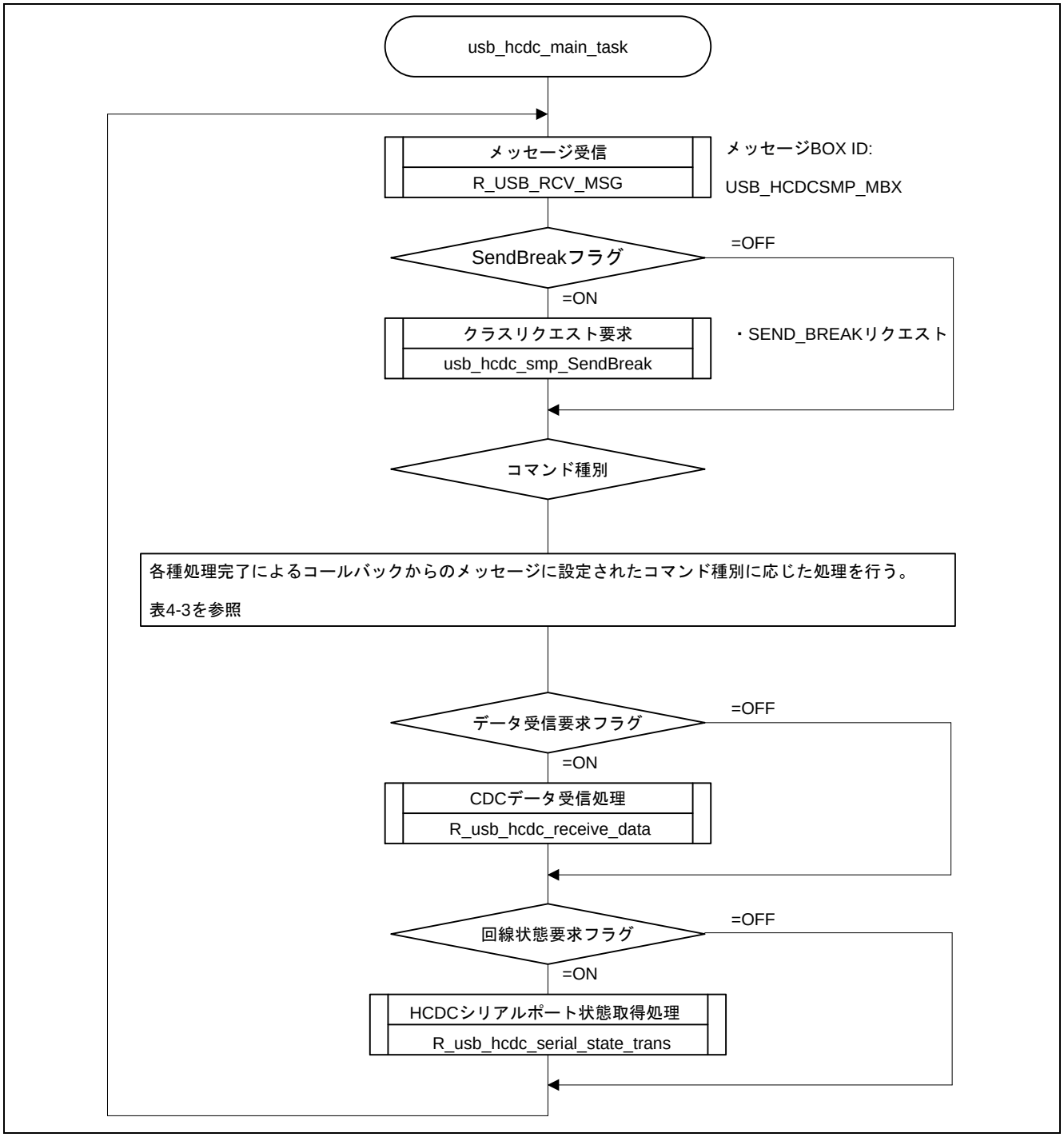


Figure 5.3 APL タスク処理概要フロー

Table 5-3 APL タスク詳細処理

メッセージ	定義	処理内容
USB_HCDC_CMD_INIT	アプリケーション初期化	アプリケーションタイトル表示 *SetControlLineState* クラスリクエストを HCDC タスクに要求 *usb_hcdc_smp_SetControlLineState* ().
USB_HCDC_CMD_ SET_CONTROL_LINE_STATE	クラスリクエスト *SetControlLineState* 転送完了	*SetLineCoding* クラスリクエストを HCDC タスクに要求 *usb_hcdc_smp_SetLineCoding ()*.
USB_HCDC_CMD_ SET_LINE_CODING	クラスリクエスト *SetLineCoding* 転送完了	***G****etLineCoding* クラスリクエストを HCDC タスクに要求 *usb_hcdc_smp_GetLineCoding* ()
USB_HCDC_CMD_ GET_LINE_CODING	クラスリクエスト *GetLineCoding* 転送完了	受信した LineCoding 表示 データ受信要求フラグ ON 回線状態要求フラグ ON
USB_HCDC_CMD_RX_OK	受信データ取得(受信レングス > 0)	ループバック送信 *usb_hcdc_smp_OutTransResult ()*.
USB_HCDC_CMD_RX_NG	受信データ非取得	データ受信要求フラグ ON (デモはデータ受信を継続する)
USB_HCDC_CMD_TX_OK **USB_HCDC_CMD_TX_NG**	データ送信完了 データ送信失敗	データ受信要求フラグ ON (デモはデータ受信を継続する)
USB_HCDC_CMD_ RCV_SERIAL_STATE	クラス通知 *SerialState* 受信完了	回線状態表示. *SetLineState* 要求フラグ ON.
USB_HCDC_CMD_ RCV_SERIAL_STATE_NG	クラス通知 *SerialState* 受信失敗	*SetLineState* 要求フラグ ON.
USB_HCDC_CMD_ SEND_BREAK	クラスリクエスト *SendBreak* 転送完了	*SendBreak* 完了表示
USB_HCDC_CMD_ SET_COMM_FEATURE	クラスリクエスト *SetCommFeature* 転送完了	*SetCommFeature* 完了表示
USB_HCDC_CMD_ GET_COMM_FEATURE	クラスリクエスト *SetCommFeature* 転送完了	*GetCommFeature* 完了表示
USB_HCDC_CMD_ CLR_COMM_FEATURE	クラスリクエスト *ClearCommFeature* 転送完了	*ClearCommFeature* 完了表示
USB_HCDC_CMD_ SEND_ENCAPSULATED_COMMAND	クラスリクエスト *SendEncapsualtedCommand* 転送完了	*SendEncapsulatedCommand* 完了表示
USB_HCDC_CMD_ GET_ENCAPSULATED_RESPONSE	クラスリクエスト *GetEncapsulatedResponse* 転送完了	*GetEncapsulatedResponse* 完了表示
USB_HCDC_CMD_SW_CHECK	周期処理 スイッチ入力による通信速度設定	SW2: 通信速度を仮更新して **USB_HCDC_CMD_SW_CHECK** メッセージを APL タスクに送信 SW3: 通信速度を確定して **USB_HCDC_CMD_SET_CONTROL_LINE_STATE** メッセージを APL タスクに送信
Undefined signal		Display received command.

5.8 シーケンスチャート APL-HCDC-HCD

サンプルアプリケーションプログラムのシーケンスを以下に示します。

5.8.1 起動、CDC デバイスアタッチ

サンプルアプリケーションプログラム起動から、エニュメレーション完了、アプリタスク起動、パイプコントロールレジスタ設定完了までのシーケンスを Figure 5.4 と Figure 5.5 に示します。

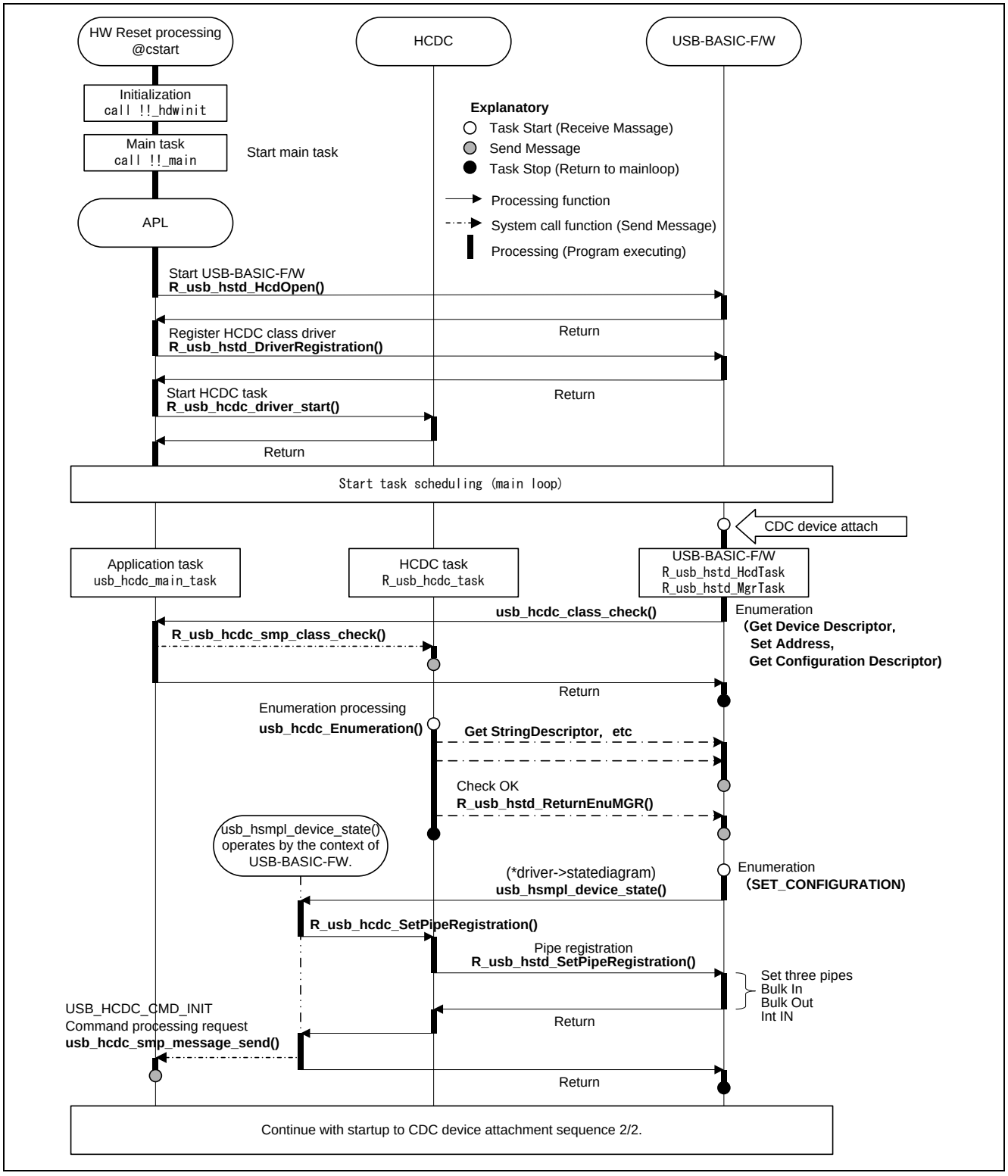


Figure 5.4 起動から CDC デバイスアタッチのシーケンス (1/2)

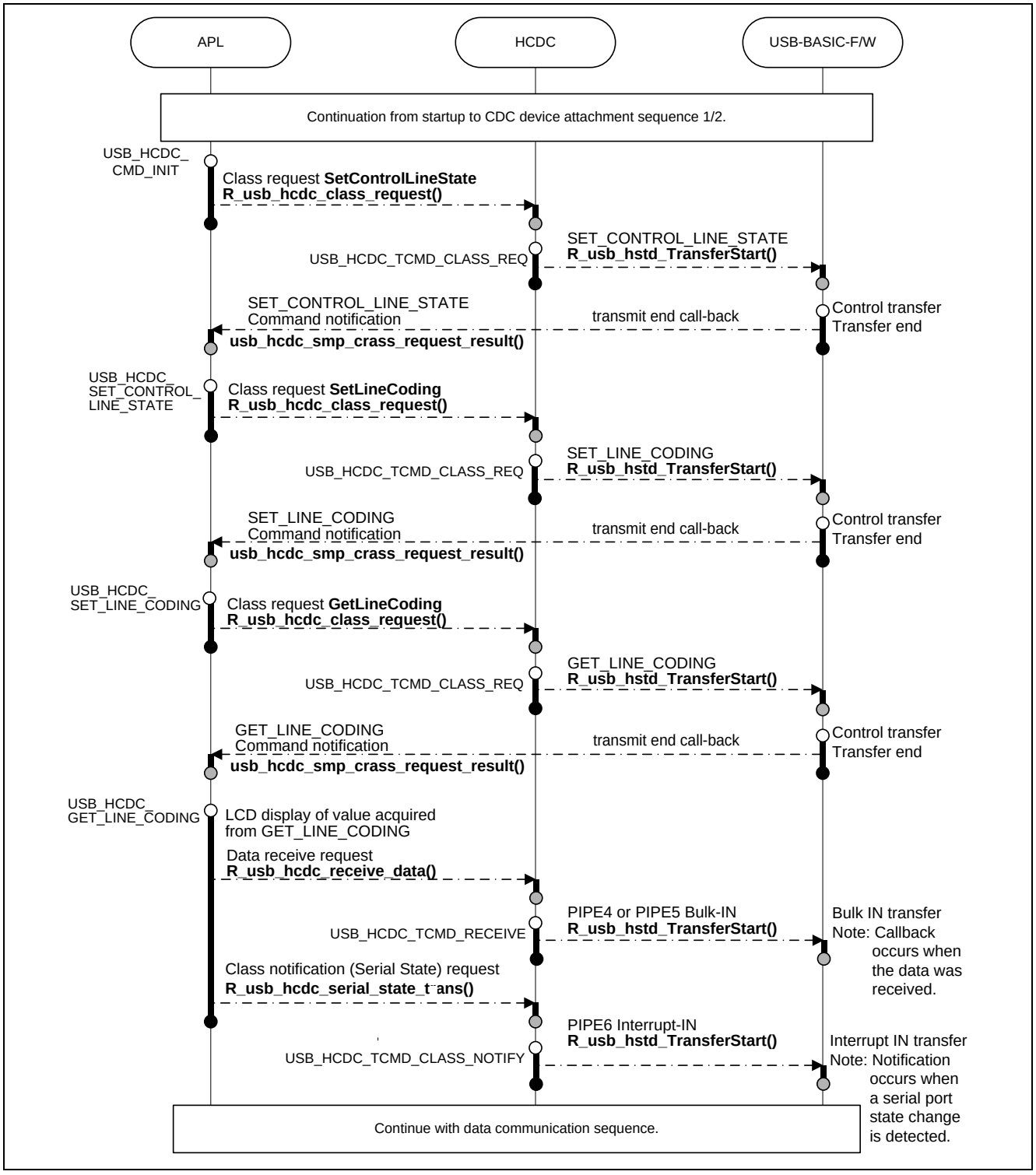


Figure 5.5 起動から CDC デバイスアタッチのシーケンス(2/2)

5.8.2 データ通信

データ転送シーケンスを Figure 5.6 に示します。

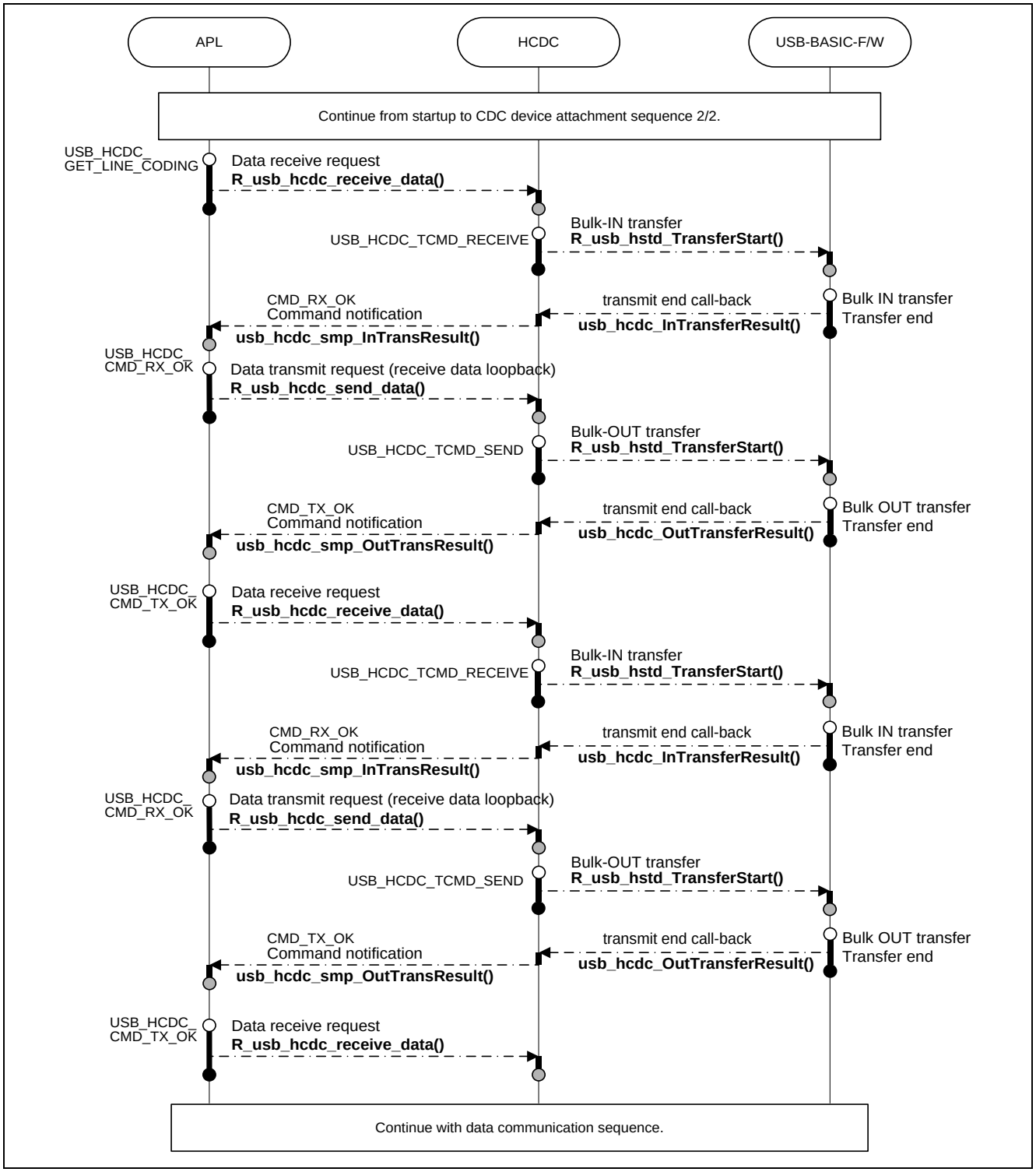


Figure 5.6 データ転送シーケンス

5.8.3 シリアルポート状態変化通知

シリアルポート状態変化通知時のシーケンスを Figure 5.7 に示します。

シリアルポート状態変化を受けるには事前に、CDC ノティフィケーション *SerialState* 受信処理を行ってください。

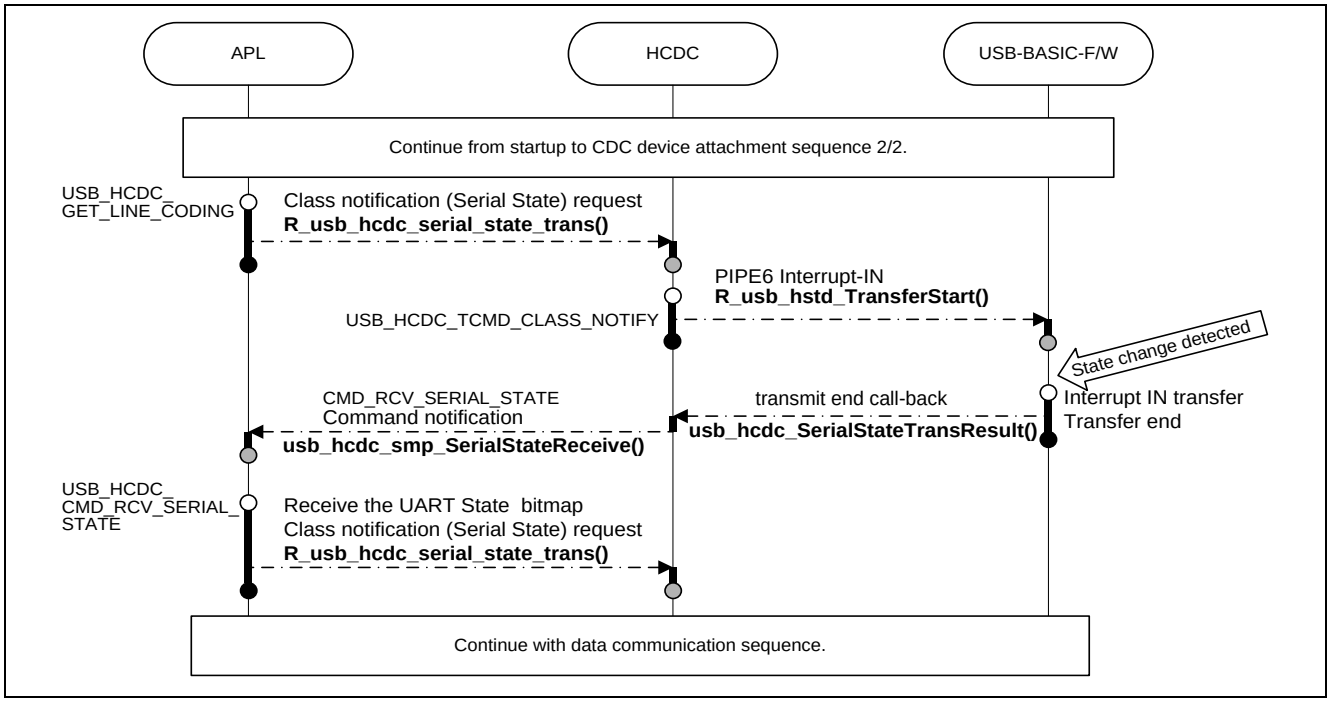


Figure 5.7 シリアルポート状態変化通知シーケンス

5.8.4 BREAK 信号出力

Figure 5.8 に BREAK 信号出力のシーケンスを示します。

BREAK 信号は、グローバル変数 *usb_shcdc_test_send_break* に USB_ON を設定することによって、接続された CDC ペリフェラルデバイスに *SEND_BREAK* リクエストで要求されます。

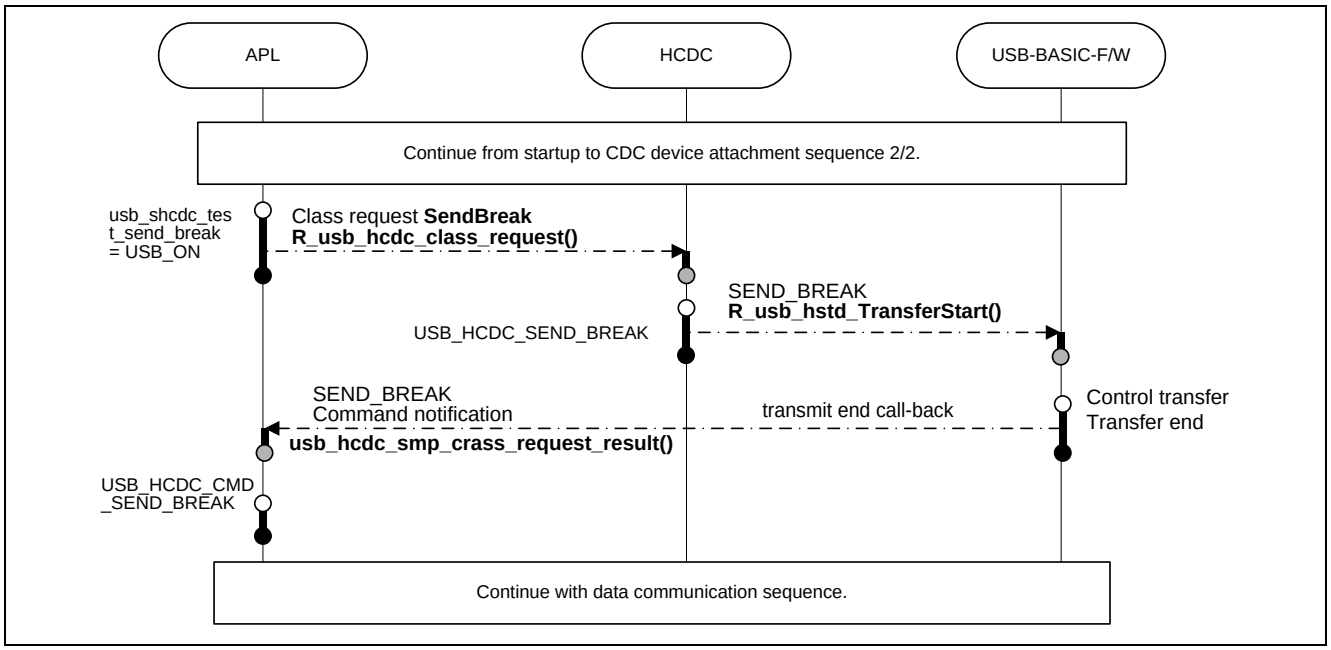


Figure 5.8 BREAK 信号出力シーケンス

5.8.5 CDC デバイスデタッチ

CDC デバイスデタッチ時のシーケンスを Figure 5.9 に示します。

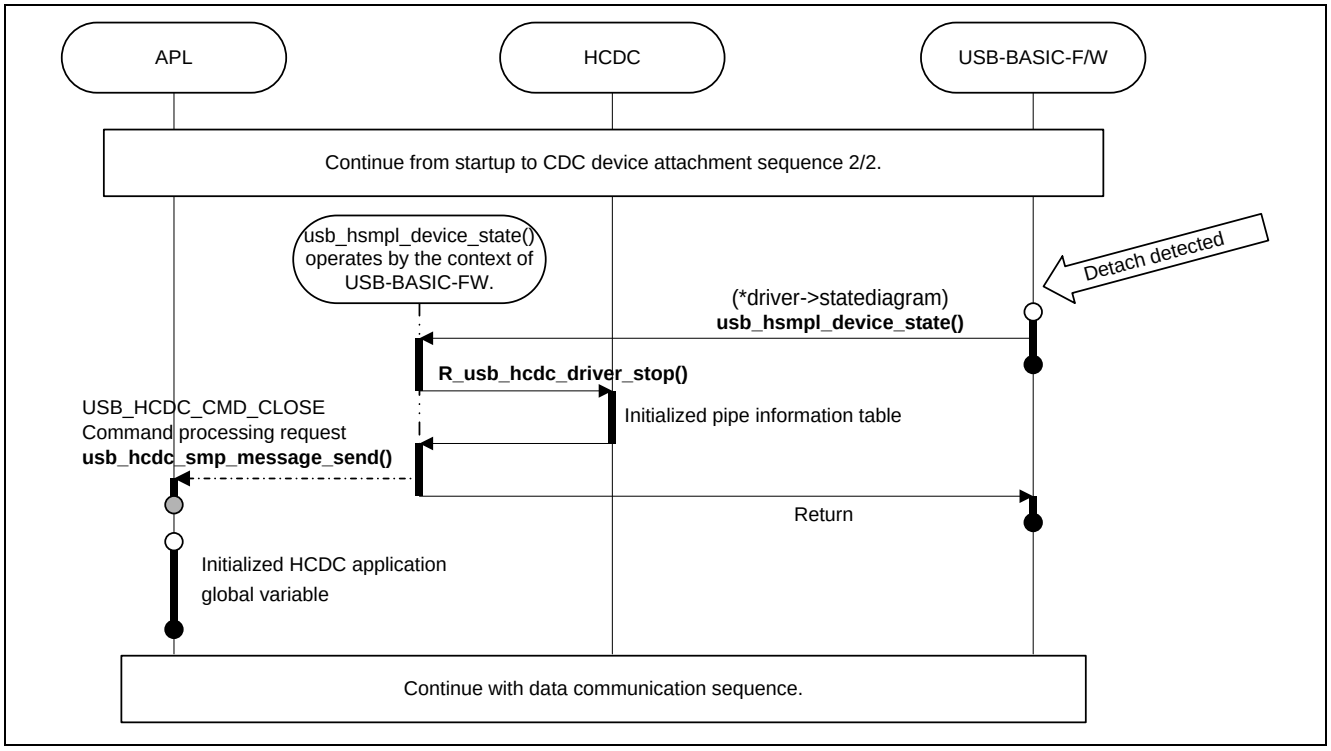


Figure 5.9 CDC デバイスデタッチシーケンス

6. コミュニケーションデバイスクラス (CDC), PSTN, and ACM

本ソフトウェアは Communication Device Class 仕様の Abstract Control Model(ACM)サブクラスに準拠します。詳細は 1.2 章の PSTN サブクラスドキュメントを参照してください。

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。

6.1 基本機能

HCDC の主な機能は、以下のとおりです。

1. 接続デバイスの照合
2. 通信回線設定の実施
3. 通信回線設定状態の取得
4. CDC ペリフェラル機器とのデータ通信

6.2 ACM クラスリクエスト(Host to Device)

本ソフトウェアで対応している ACM クラスリクエストを Table 6-1 に示します。

Table 6-1 CDC 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。	Yes
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。	Yes
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。	Yes
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。	Yes
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。	Yes
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）	Yes
GetLineCoding	0x21	通信回線設定状態を取得する。	Yes
SetControlLineState	0x22	通信回線制御信号 RTS、DTR の設定を行う。	Yes
SendBreak	0x23	ブレイク信号の送信を行う。	Yes

詳細は “USB Class Definitions for Communications Devices , Revision 1.2”の Management Element Requests の Table 19 “Class-Specific Request Codes”と“USB Communications Class Subclass Specification for PSTN Devices, Revision 1.2”の Abstract Control Model requests,の Table 11, “Requests - Abstract Control Model”を参照ください。

6.3 CDC ノティフィケーション (Notifications from Device to Host)

本ソフトウェアで対応している CDC ノティフィケーションを Table 6-2 に示します。

Table 6-2 CDC ノティフィケーション

ノティフィケーション	コード	説明	対応
NetworkConnection	0x00	ネットワーク接続状況を通知する	No
ResponseAvailable	0x01	GET_ENCAPSLATED_RESPONSE の応答	No
SerialState	0x20	シリアル回線状態を通知する	Yes

詳細は, “USB Class Definitions for Communications Devices , Revision 1.2”の Management Element Requests の Table 20 “Class-Specific Request Codes”と“USB Communications Class Subclass Specification for PSTN Devices, Revision 1.2”の Abstract Control Model requests,の Table 28, “Requests - Abstract Control Model”を参照ください。

7. USB Host コミュニケーションデバイスクラスドライバ(HCDC)

7.1 基本機能

本ソフトウェアは、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。1.2 章を参照してください。

HCDC の主な機能は、以下のとおりです。

1. CDC ペリフェラルデバイスに対してクラスリクエスト送信
2. CDC ペリフェラルデバイスとのデータ通信
3. CDC ペリフェラルデバイスからのシリアル通信エラー情報受信

7.2 HCDC タスク説明

本タスクは、USB_HCDC_MBX にメッセージを受け取り、メッセージ種別に従って処理を実行します。メッセージ種別による処理概要を Table 7-1 に示します。

Table 7-1 HCDC メッセージ種別

メッセージタイプ	処理概要	メッセージ送信元
USB_HCDC_ TCMD_OPEN	エニュメレーションシーケンスに従い、ストリングディスクリプタの取得やパイプ設定を行います。	usb_hsmp l _class_check() usb_hcdc_CheckResult() USB-BASIC-F/W がエニュメレーション処理で接続デバイスの動作可否をコールバック関数で確認します。
USB_HCDC_ TCMD_RECEIVE	バルク IN 転送開始 データ転送完了時に APN にコールバック関数で通知する	R_usb_hcdc_receive_data() バルク IN 転送終了時に API 関数が実行される
USB_HCDC_ TCMD_SEND	バルク OUT 転送開始 データ転送完了時に APL にコールバック関数で通知する	R_usb_hcdc_send_data() バルク IN 転送終了時に API 関数が実行される
USB_HCDC_TCMD_ CLASS_NOTIFY	インタラプト IN 転送開始 データ転送完了時に APN にコールバック関数で通知する	R_usb_hcdc_serial_state_trans() インタラプト IN 転送終了時に API 関数が実行される
USB_HCDC_TCMD_ CLASS_REQ	CDC クラスリクエスト送信 リクエスト種別は APL が引数を使用して通知する。コントロール転送完了時に APN にコールバック関数で通知する	R_usb_hcdc_class_request() クラスリクエスト発行のサンプル関数から API 関数が実行される

7.3 ターゲットペリフェラルリスト(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。ホストデバイスがどのような周辺機器をサポートするのかは、それぞれのホストデバイスごとに異なります。これをデバイスの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 を判定する処理は *r_usb_hcdc_driver.c* ファイルの *usb_hcdc_Enumaration()* 関数を参照ください。

7.4 構造体

7.4.1 HCDC リクエスト構造体

CDC リクエスト *SetLineCoding* および *GetLineCoding*.で使用する UART 設定パラメータ用の構造体を Table 7-2 に示します。

Table 7-2 USB_HCDC_LineCoding_t Structure

型	メンバ名	説明	備考
uint32_t	dwDTERate	回線速度	単位：bps
uint8_t	bCharFormat	ストップビット設定	
uint8_t	bParityType	パリティ設定	
uint8_t	bDataBits	データビット長	

CDC リクエスト *SetControlLineState*.で使用する UART 設定パラメータ用の構造体を Table 7-3 に示します。

Table 7-3 USB_HCDC_ControlLineState_t Structure

型	メンバ名	説明	備考
uint16_t (D15-D8)	rsv1:8	予約領域 1	
uint16_t (D7-D2)	rsv2:6	予約領域 2	
uint16_t (D1)	bRTS:1	Carrier control for half duplex modems 0 - Deactivate carrier, 1 - Activate carrier	
uint16_t (D0)	bDTR:1	Indicates to DCE if DTE is present or not 0 - Not Present, 1 - Present	

CDC リクエスト *SendEncapsulatedCommand* および *GetEncapsulatedResponse* で使用する AT コマンドパラメータ用の構造体を Table 7-4 に示します。

Table 7-4 USB_HCDC_Encapsulated_t Structure

型	メンバ名	説明	備考
uint8_t	*p_data	AT コマンドデータが格納されている領域	
uint16_t	wLength	AT コマンドデータのサイズ	単位：byte

CDC リクエスト *SendBreak* で使用する Break 信号送出パラメータ用の構造体を Table 7-5 に示します。

Table 7-5 USB_HCDC_BreakDuration_t Structure

型	メンバ名	説明	備考
uint16_t	wTime_ms	Duration of Break	単位：ms

7.4.2 CommFeature 機能選択用共用体

CDC リクエスト *SetCommFeature* および *GetCommFeature* で使用する“Feature Selector”パラメータ用の構造体を Table 7-6 と Table 7-7 に、共用体を Table 7-8 に示します。

Table 7-6 USB_HCDC_AbstractState_t Structure

型	メンバ名	説明	備考
uint16_t	rsv1:8	予約領域 1	
uint16_t	rsv2:6	予約領域 2	
uint16_t	bDMS:1	Data Multiplexed State	
iomt16_t	bIS:1	Idle Setting	

Table 7-7 USB_HCDC_CountrySetting_t Structure

型	メンバ名	説明	備考
uint16_t	country_code	Country code in hexadecimal format as defined in [ISO3166],	

Table 7-8 USB_HCDC_CommFeature_t Structure

型		メンバ名	説明	備考
union	uint16_t	data	Status data	
	USB_HCDC_AbstractState_t	AbstractState	Abstract Control Model selection parameters. Refer to Table 7-6	
	USB_HCDC_CountrySetting_t	CountrySetting	Country Setting selection parameters. Refer to Table 7-7	
uint16_t		wValue	Feature selector code	

7.4.3 CDC リクエスト入力パラメータ共用体

CDC リクエスト毎のパラメータ構造体を共用体として定義したものを Table 7-9 に示します。.

Table 7-9 USB_HCDC_ClassRequestParm_t 構造体

リクエスト	リクエストコード/構造体の型	メンバー名	説明
SetLineCoding	USB_HCDC_SET_LINE_CODING / USB_HCDC_LineCoding_t	*LineCoding	データステージで送受信するデータアドレス Table 7-2 参照
GetLineCoding	USB_HCDC_GET_LINE_CODING / USB_HCDC_LineCoding_t		
SetControlState	USB_HCDC_SET_ CONTROL_LINE_STATE / USB_HCDC_ControlLineState_t	ControlLineState	wValue に設定する値 Table 7-3 参照
SendEncapsulated Command	USB_HCDC_SEND_ ENCAPSULATED_COMMAND / USB_HCDC_Encapsulated_t	Encapsulated	データステージで送受信するデータアドレスと wValue に設定する値 Table 7-4 参照
GetEncapsulated Response	USB_HCDC_GET_ ENACAPSULATED_RESPONSE / USB_HCDC_Encapsulated_t		
SendBreak	USB_HCDC_SEND_BREAK / USB_HCDC_BreakDuration_t	BreakDuration	wValue に設定する値 Table 7-5 参照
SetCommFeature	USB_HCDC_SET_COMM_FEATURE / USB_HCDC_CommFeature_t	*CommFeature	データステージで送受信するデータアドレス Table 7-8 参照
GetCommFeature	USB_HCDC_GET_COMM_FEATURE / USB_HCDC_CommFeature_t		
ClearCommFeature	USB_HCDC_CLR_COMM_FEATURE No structure		

7.4.4 CDC リクエスト API 関数の構造体

CDC リクエスト用の構造体を Table 7-10 に示します。

Table 7-10 USB_HCDC_ClassRequest_UTR_t 構造体

型	メンバ名	説明
usb_addr_t	devadr	デバイスアドレス
uint8_t	bRequestCode	クラスリクエスト種別（Table 7-9 参照）
USB_CDC_ClassRequestParm_t	parm	パラメータ設定値（Table 7-9 参照）
usb_cb_t	complete	クラスリクエスト処理完了コールバック関数

7.4.5 CDC ノティフィケーションフォーマット

Table 7-11 と Table 7-12 に CDC ノティフィケーションのデータフォーマットを示します。

Table 7-11 Response_Available notification format

Type	Member	Description	Remarks
uint8_t	bmRequestType	0xA1	
uint8_t	bRequest	RESPONSE_AVAILABLE(0x01)	
uint16_t	wValue	0x0000	
uint16_t	wIndex	Interface	
uint16_t	wLength	0x0000	
uint8_t	Data	--	

Table 7-12 Serial_State notification format

Type	Member	Description	Remarks
uint8_t	bmRequestType	0xA1	
uint8_t	bRequest	SERIAL_STATE(0x20)	
uint16_t	wValue	0x0000	
uint16_t	wIndex	Interface	
uint16_t	wLength	0x0002	
uint16_t	Data	UART State bitmap	Refer to Table 7-13

UART ポート状態変化の検出によりホストに通知されるクラスノティフィケーション“*SerialState*”の UART State Bitmap 構造体を Table 7-13 に示します。

Table 7-13 USB_HCDC_SerialState_t 構造体

型	メンバ名	説明	備考
uint16_t (D15-D8)	rsv1:8	予約領域 1	
uint16_t (D7)	rsv2:1	予約領域 2	
uint16_t (D6)	bOverRun:1	オーバーランエラー検出	
uint16_t (D5)	bParity:1	パリティエラー検出	
uint16_t (D4)	bFraming:1	フレミングエラー検出	
uint16_t (D3)	bRingSignal:1	着信（Ring signal）を感知	
uint16_t (D2)	bBreak:1	ブレイク信号検出	
uint16_t (D1)	bTxCarrier:1	回線が接続されて通信可能	Data Set Ready
uint16_t (D0)	bRxCarrier:1	回線にキャリア検出	Data Carrier Detect

7.5 HCDC API 一覧

HCDC API 一覧を Table 7-14 に示します。

Table 7-14 HCDC API 関数一覧

関数名	機能概要	備考
R_usb_hcdc_task	HCDC タスク処理	
R_usb_hcdc_smp_class_check	接続デバイス動作確認用メッセージ通知	
R_usb_hcdc_driver_start	HCDC ドライバ開始	
R_usb_hcdc_driver_stop	HCDC ドライバ停止	
R_usb_hcdc_SetPipeRegistration	パイプコントロールレジスタ設定処理	
R_usb_hcdc_receive_data	USB 受信処理	
R_usb_hcdc_receive_data_end	USB 受信停止処理	
R_usb_hcdc_send_data	USB 送信処理	
R_usb_hcdc_send_data_end	USB 送信停止処理	
R_usb_hcdc_serial_state_trans	クラスノティフィケーション処理	
R_usb_hcdc_serial_state_trans_stop	クラスノティフィケーション停止処理	
R_usb_hcdc_device_information	接続デバイスの状態確認	
R_usb_hcdc_change_device_state	接続デバイスの状態変更	
R_usb_hcdc_class_request	クラスリクエスト処理	

R_usb_hcdc_task

HCDC タスク処理

形式

void R_usb_hcdc_task(void)

引数

― ―

戻り値

― ―

解説

*usb_hcdc_task()*関数を呼び出します。

HCDC タスクは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

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_apl_task_switch(void)
{
  while( 1 )
  {

    if( USB_FLGSET == R_usb_cstd_Scheduler() )    /* Scheduler */
    {
       R_usb_hstd_HcdTask();    /* HCD Task */
       R_usb_hstd_MgrTask();    /* MGR Task */
       usb_hcdc_main_task();    /* HCDC Application Task */
       R_usb_hcdc_task();       /* HCDC Task */

    }
    else
    {
    }
  }
}
```

R_usb_hcdc_smp_class_check

ディスクリプタチェック処理

形式

void	R_usb_hcdc_smp_class_check (uint8_t **table)

引数

**table	Address array of the device information
	[0] : Address of Device Descriptor
	[1] : Address of Configuration Descriptor
	[2] : Address of global variable that mean the Device Address

戻り値

―	―

解説

本関数は接続デバイスの動作可否を判断する処理の実行を HCDC タスクに要求します。USB-BASIC-F/W が *classcheck* コールバックを実行した場合に本関数を呼び出してください。

HCDC タスクはペリフェラルデバイスのコンフィグレーションディスクリプタからエンドポイントディスクリプタを参照し、パイプ情報テーブル編集及び、使用するパイプ情報のチェックを行います。

補足

使用例

```
void  usb_hcdc_registration(void)
{
  usb_hcdreg_t  driver;

  driver.ifclass      = USB_IFCLS_CDCC;
  driver.classcheck   = &R_usb_hcdc_smp_class_check;
  driver.statediagram = &usb_hcdc_device_state;
  R_usb_hstd_DriverRegistration(&driver);
}
```

R_usb_hcdc_driver_start

HCDC ドライバ起動

形式

void	R_usb_hcdc_driver_start(void)

引数

—	—

戻り値

—	—

解説

HCDC ドライバタスクを起動します。

補足

使用例

```
void usb_hcdc_task_start( void )
{
  /* Target board initialize */
  usb_cpu_target_init();

  /* USB-IP initialized */
  R_usb_hstd_ChangeDeviceState(USB_DO_INITHWFUNCTION);

  /* HCD driver open & registration */
  R_usb_hstd_HcdOpen();              /* HCD task, MGR task open */
  usb_hcdc_registration();           /* HCDC driver registration */
  R_usb_hcdc_driver_start();         /* HCDC Task start */

  /* Scheduler initialized */
  R_usb_hstd_ChangeDeviceState(USB_DO_SETHWFUNCTION);

}
```

R_usb_hcdc_driver_stop

HCDC ドライバ停止

形式

void R_usb_hcdc_driver_stop(void)

引数

― ―

戻り値

― ―

解説

本関数はパイプ情報テーブルを初期化します。

補足

使用例

```
USB_STATIC void usb_hcdc_device_state(uint16_t data, uint16_t state)
{
  switch( state )
  {
   case USB_STS_DETACH:
      usb_smpl_set_suspend_flag(USB_NO);
      usb_shcdc_line_speed_control_seq = USB_HCDC_SMP_LINE_SPEED_INIT;
      R_usb_hcdc_driver_stop();
      usb_hcdc_smp_message_send(USB_HCDC_CMD_CLOSE);
   break;
            .
            .
            .
}
```

R_usb_hcdc_SetPipeRegistration

パイプ設定処理

形式

void	R_usb_hcdc_SetPipeRegistration (usb_addr_t devaddr)

引数

devaddr	デバイスアドレス

戻り値

—	—

解説

本関数はパイプ情報テーブルのアドレスフィールドを更新します。CDC 通信で使用されるパイプをハードウェアに設定します。

HCDC では合計 3 本のパイプをセットアップします。データ通信用の Bulk IN、Bulk OUT パイプとシリアル状態を受け取る Interrupt IN パイプです。

補足

1. USB-BASIC-F/W のアプリケーションノート（パイプ情報）を参照してください。
2. エンドポイントディスクリプタを参照できないパイプ情報テーブルのフィールドはあらかじめ設定しておいてください。

使用例

```
void usb_hcdc_smp_open(usb_adrr_t devadr)
{
  usb_er_t      err;

  if (devadr != 0)
  {
    usb_shcdc_devadr = devadr;     /* Device Address store */

    /* Host CDC Pipe Registration */
    err = R_usb_hcdc_SetPipeRegistration(usb_shcdc_devadr);
  }
}
```

R_usb_hcdc_receive_data

データ受信要求

形式

usb_er_t R_usb_hcdc_receive_data (uint8_t *table, usb_leng_t size, usb_cb_t complete)

引数

*table	データ受信バッファ領域へのポインタ
size	転送サイズ
complete	処理完了通知コールバック関数

戻り値

USB_E_OK	正常終了
USB_E_ERROR	終了失敗

解説

USB-BASIC-F/W に対し、データ受信要求を行ないます。

引数“***table***”が示すアドレスに引数“*size*” バイトのデータを受信します。

データ受信処理（"*size*"バイトのデータ受信、もしくはショートパケット受信）が完了するとコールバック関数を呼び出します。

補足

1. データ転送処理結果はコールバック関数の引数 “*usb_utr_t* *” で通知します。
2. *usb_utr_t* に関しては USB-BASIC-F/W のアプリケーションノートを参照してください。

使用例

```
/* Example of usage. */
  uint8_t   data[64];                          /* Data buff */
  usb_leng_t size = 64;                        /* Data size */

    R_usb_hcdc_receive_data((uint8_t *)data, size, (usb_cb_t)&usb_complete)

/* Callback function */
void  usb_complete( usb_utr_t *mess );
{
  /* Describe the processing performed when the USB receive is completed. */
}
```

R_usb_hcdc_receive_data_end

データ受信強制終了

形式

usb_er_t R_usb_hcdc_receive_data_end (void)

引数

― ―

戻り値

USB_E_OK	正常終了
USB_E_ERROR	終了失敗
USB_E_QOVR	オーバラップ（転送終了中のパイプに対する転送終了要求）

解説

USB-BASIC-F/W に対してデータ転送の強制終了を要求します。

データ転送要求時（*R_usb_hcdc_receive_data*）に設定したコールバック関数で転送終了を通知します。コールバック関数の引数（usb_utr_t）で、送受信の残りデータ長、パイプコントロールレジスタの値、転送ステータス=USB_DATA_STOP を設定します。

補足

1. データ転送処理結果はコールバック関数の引数 “*usb_utr_t **” で通知します。
2. *usb_utr_t* に関しては USB-BASIC-F/W の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_task(void)
{

  /* reception end request */
  err = R_usb_hcdc_receive_data_end();

  return err;
    :
}
```

R_usb_hcdc_send_data

データ送信要求

形式

usb_er_t R_usb_hcdc_send_data(uint8_t *table, usb_leng_t size, usb_cb_t complete)

引数

*table	送信データバッファ領域へのポインタ
size	転送サイズ
complete	処理完了通知コールバック関数

戻り値

USB_E_OK	正常終了
USB_E_ERROR	終了失敗

解説

USB-BASIC-F/W に対し、データ転送要求を行ないます。

引数“*table*”が示すアドレスから引数“*size*” バイトのデータを送信します。

データ送信処理が完了するとコールバック関数を呼び出します。

補足

1. データ転送処理結果はコールバック関数の引数 “*usb_utr_t **” で通知します。
2. *usb_utr_t* に関しては USB-BASIC-F/W のアプリケーションノートを参照してください。

使用例

```
/* Example of usage. */
  uint8_t   data[] = {0x01,0x02,0x03,0x04,0x05};/* USB send data */
  usb_leng_t size = 5;                          /* Data size */

  R_usb_hcdc_send_data((uint8_t *)data, size,(usb_cb_t)&usb_complete)

/* Callback function */
void  usb_complete( usb_utr_t *mess );
{
  /*  Describe the processing performed when the USB transmit is completed.  */
}
```

R_usb_hcdc_send_data_end

データ送信強制終了

形式

usb_er_t	R_usb_hcdc_send_data_end (void)

引数

―	―

戻り値

USB_E_OK	正常終了
USB_E_ERROR	終了失敗
USB_E_QOVR	オーバラップ（転送終了中のパイプに対する転送終了要求）

解説

Description

USB-BASIC-F/W に対してデータ転送の強制終了を要求します。

データ転送要求時（*R_usb_hcdc_send_data*）に設定したコールバック関数で転送終了を通知します。コールバック関数の引数（usb_utr_t）で、送受信の残りデータ長、パイプコントロールレジスタの値、転送ステータス=USB_DATA_STOP を設定します。

補足

1. データ転送処理結果はコールバック関数の引数 “*usb_utr_t* *” で通知します。
2. *usb_utr_t* に関しては USB-BASIC-F/W の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_task(void)
{

  /* Transfer end request */
  err = R_usb_hcdc_send_data_end();

  return err;
    :
}
```

R_usb_hcdc_serial_state_trans

CDC ノティフィケーション要求

形式

usb_er_t R_usb_hcdc_serial_state_trans(usb_cb_t *complete)

引数

complete	処理完了を通知するコールバック関数

戻り値

USB_E_OK	正常終了
USB_E_ERROR	終了失敗

解説

CDC ノティフィケーション転送要求を行ないます。(7.4.5 章参照).

ノティフィケーション受信後 *complete* コールバック関数を呼び出します。

コールバック関数の引数は *usb_utr_t*型で HCDC のグローバル変数です。HCDC は *usb_utr_t*構造の *tranadr* メンバに受信データを格納します。 CDC ノティフィケーションが正しく通知されると、構造体の *result* メンバに USB_YES を設定します。 コールバック関数の引数のメンバ(tranadr)に CDC クラスノティフィケーションフォーマット領域(Table 7-12 参照)の先頭アドレスが設定されます。この領域の” UART State bitmap”からシリアルステートを取り出してください。

補足

1. シリアルステータスのビットパターンは“Table 7-13”を参照してください。
2. データ転送処理結果はコールバック関数の引数 “*usb_utr_t* *” で通知します。
3. *usb_utr_t* に関しては USB-BASIC-F/W のアプリケーションノートを参照してください。
4. 接続デバイスがベンダクラスデバイスの場合は、CDC ノティフィケーションは要求できません。
5. *ResponseAvailable* ノティフィケーションには対応していません。

使用例

```
void usb_hcdc_main_task(void)
{
  USB_MGRINFO_t          *mess;
  usb_er_t               err;

  while (1)
  {
  err  = R_USB_RCV_MSG(USB_HCDCSMP_MBX,(USB_MSG_t**)&mess);
  if (err == USB_E_OK)
  {
   err = R_usb_hcdc_serial_state_trans( mess,
         (usb_cb_t *)&usb_hcdc_smp_SerialStateReceive );
   if( err != USB_E_OK )
   {
      USB_PRINTF0("### usb_pcdc_MainTask function bulk read error\n");
   }
  }
}


/* Example of a callback function of R_usb_hcdc_serial_state_trans */
void usb_hcdc_smp_SerialStateReceive(usb_utr_t *mess)
{
  uint16_t *status;
  uint16_t msginfo;

  if (mess->result == USB_YES)
  {
   /* Command set */
   msginfo = USB_HCDC_CMD_RCV_SERIAL_STATE;
  }
  else
  {
   /* Command set */
   msginfo = USB_HCDC_CMD_RCV_SERIAL_STATE_NG;
  }
  status = (uint16_t *)mess->tranadr;      /* Status set */
  /* [0] bmRequestType/bRequest */
  /* [1] wValue */
  /* [2] wIndex */
  /* [3] wLength :2 */
  /* [4] data : Serial State(UART State bitmap) */
  usb_hcdc_smp_message_send( mess, status[4]);
}
```

R_usb_hcdc_serial_state_trans_end

CDC ノティフィケーション終了

形式

usb_er_t R_usb_hcdc_serial_state_trans_end (void)

引数

― ―

戻り値

USB_E_OK	正常終了
USB_E_ERROR	終了失敗
USB_E_QOVR	オーバラップ（転送終了中のパイプに対する転送終了要求）

解説

USB-BASIC-F/W に対してデータ転送の強制終了を要求します。

データ転送要求時（*R_usb_hcdc_serial_satte_trans*）に設定したコールバック関数で転送終了を通知します。コールバック関数の引数（usb_utr_t）で、送受信の残りデータ長、パイプコントロールレジスタの値、転送ステータス=USB_DATA_STOP を設定します。

補足

1. データ転送処理結果はコールバック関数の引数 “*usb_utr_t **” で通知します。
2. *usb_utr_t* に関しては USB-BASIC-F/W の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_task(void)
{

  /* notification end request */
  err = R_usb_hcdc_serial_state_trans_end();

  return err;
    :
}
```

R_usb_hcdc_class_request

CDC リクエスト要求

形式

usb_er_t R_usb_hcdc_class_request (USB_HCDC_ClassRequest_UTR_t *pram)

引数

*pram CDC クラスリクエストパラメータ

戻り値

－ エラーコード (USB_E_OK/USB_E_ERROR)

解説

HCDC ドライバに対し、CDC クラスリクエスト発行要求を行います。

引数*parm* の構造体メンバ *bRequestCode* でリクエスト種別を判断します。

1. SendEncapsulatedCommand
2. GetEncapsulatedResponse
3. SetCommFeature
4. GetCommFeature
5. ClearCommFeature
6. SetLineCoding
7. GetLineCoding
8. SetControlLineState
9. SendBreak

どのように使用するかの詳細はサンプルアプリケーション *r_usb_hcdc_apl.c* を参照してください。

引数 *USB_HCDC_ClassRequest_UTR_t* 構造体の種別は 7.4.4 章を参照してください。

補足

1. データ転送処理結果はコールバック関数の引数"*usb_utr_t* *"で通知します。
2. *usb_utr_t.*に関しては USB-BASIC-F/W のアプリケーションノートを参照してください。

使用例

```
USB_HCDC_LineCoding_t          usb_shcdc_line_coding;
USB_HCDC_ClassRequest_UTR_t    utr_req;
/* Example of usage. */
  usb_shcdc_line_coding.dwDTERate    = USB_HCDC_SPEED_9600;
  usb_shcdc_line_coding.bDataBits    = USB_HCDC_DATA_BIT_8;
  usb_shcdc_line_coding.bCharFormat  = USB_HCDC_STOP_BIT_1;
  usb_shcdc_line_coding.bParityType  = USB_HCDC_PARITY_BIT_NONE;
  utr_req.bRequestCode          = USB_HCDC_SET_LINE_CODING;
  utr_req.complete              = (usb_cb_t)&usb_hcdc_smp_SetLine_CODING_Result;
  utr_req.parm.LineCoding       = &usb_shcdc_line_coding;
  utr_req.devadr                = devadr;
  return = R_usb_hcdc_class_request( utr_req );
/* Example of callback function. */
void usb_hcdc_smp_SetLine_CODING_Result(usb_utr_t *mess)
{
  /* Describe the processing performed when the class request is completed. */
}
```

R_usb_hcdc_device_information

デバイスステータス取得

形式

void	R_usb_hcdc_device_information(uint16_t *deviceinfo)

引数

*deviceinfo	デバイス情報格納用バッファへのポインタ

戻り値

―	―

解説

USB デバイス情報を取得します。 引数"**deviceinfo*"で指定されたアドレスに以下の情報を保存します。
[0]: 接続されているルートポート番号 (port 0: USB_0, port 1: USB_1)
[1]: デバイスステート
(未接続:USB_STS_DETACH, エニュメレーション中:USB_STS_DEFAULT/USB_STS_ADDRESS, コンフィガード:USB_STS_CONFIGURED, サスペンド中:USB_STS_SUSPEND)
[2]: 構成番号 (*g_usb_HcdDevInfo[g_usb_MgrDevAddr].config*)
[3]: 接続速度 (FS: USB_FSCONNECT, LS: USB_LSCONNECT, 未接続: USB_NOCONNECT)

補足

1. 引数**deviceinfo* に 4 ワードの領域を確保してください。
2. デバイスアドレスが 0 の場合にこの関数が呼ばれると以下の情報を応答します。
   (1) デバイスがエニュメレーション中でない（デバイス未接続）
   table[0] = USB_NOPORT, table[1] = USB_STS_DETACH
   (2) デバイスがエニュメレーション中
   table[0] = Port number, table[1] = USB_STS_DEFAULT

使用例

```
void  usb_smp_task(void)
{
  uint16_t  tbl[4];
    :
  /* Device information check */
  R_usb_hcdc_device_information(tbl);
    :
}
```

R_usb_hcdc_change_device_state

デバイス状態変更処理

形式

usb_er_t R_usb_hcdc_change_device_state (usb_strct_t msginfo,
usb_strct_t keyword,
usb_cb_info_t complete)

引数

msginfo	更新するデバイス状態
keyword	ポートナンバーなど *msginfo* に従い内容は異なります
complete	状態変更が終了した場合に実行されるコールバック関数

戻り値

USB_E_OK	正常終了
USB_E_ERROR	終了失敗

解説

引数 *msginfo* に以下の値を設定しデバイス状態変更を USB-BASIC-F/W に要求してください。

- USB_DO_PORT_ENABLE / USB_DO_PORT_DISABLE
  keyword で指定されたポートの許可/禁止（VBUS 出力の on/off 制御）を行います。
- USB_DO_GLOBAL_SUSPEND
  keyword で指定されたポートをサスペンドにします。
- USB_DO_GLOBAL_RESUME
  keyword で指定されたポートをレジュームします。
- USB_DO_CLEAR_STALL
  keyword で指定されたパイプの STALL 状態を解除します。

補足

1. USB-BASIC-F/W が接続もしくは切断を検出した場合は、USB-BASIC-F/W は自動的にエニュメレーションシーケンス処理、もしくはデタッチシーケンス処理を行います。
2. 本関数を使用して USB 状態を変更した場合は、API 関数 *R_usb_hstd_DriverRegistration()*を使用して登録したドライバ構造体の USB 状態遷移コールバックは呼ばれません。

使用例

```
void  usb_smp_task(void)
{
  R_usb_hcdc_change_device_state
   (USB_DO_GLOBAL_SUSPEND, USB_PORT0, usb_hsmpl_status_result);
}
```

8. 制限事項

HCDC 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構造体メンバのアドレスアライメントずれが発生することがあります)
2. HCDC ドライバに接続可能なデバイスは 1 つだけです。2 つ以上のデバイスを同時に接続しないでください。

9. 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。

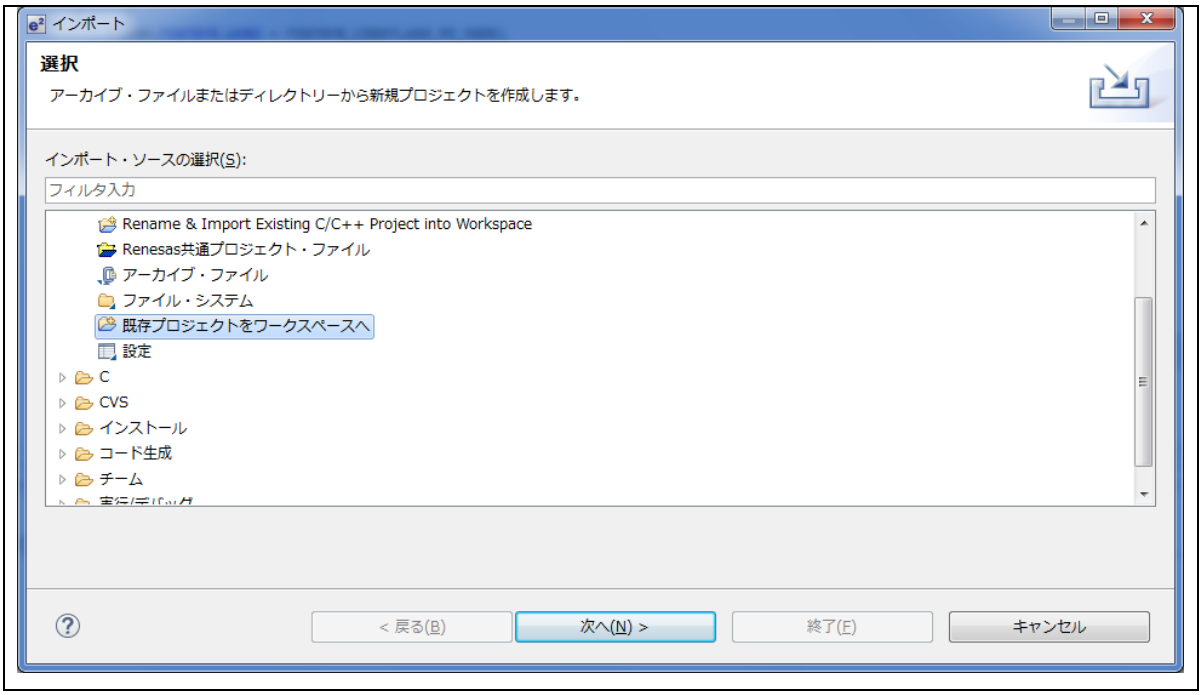


Figure 9-1 インポートの選択

(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。

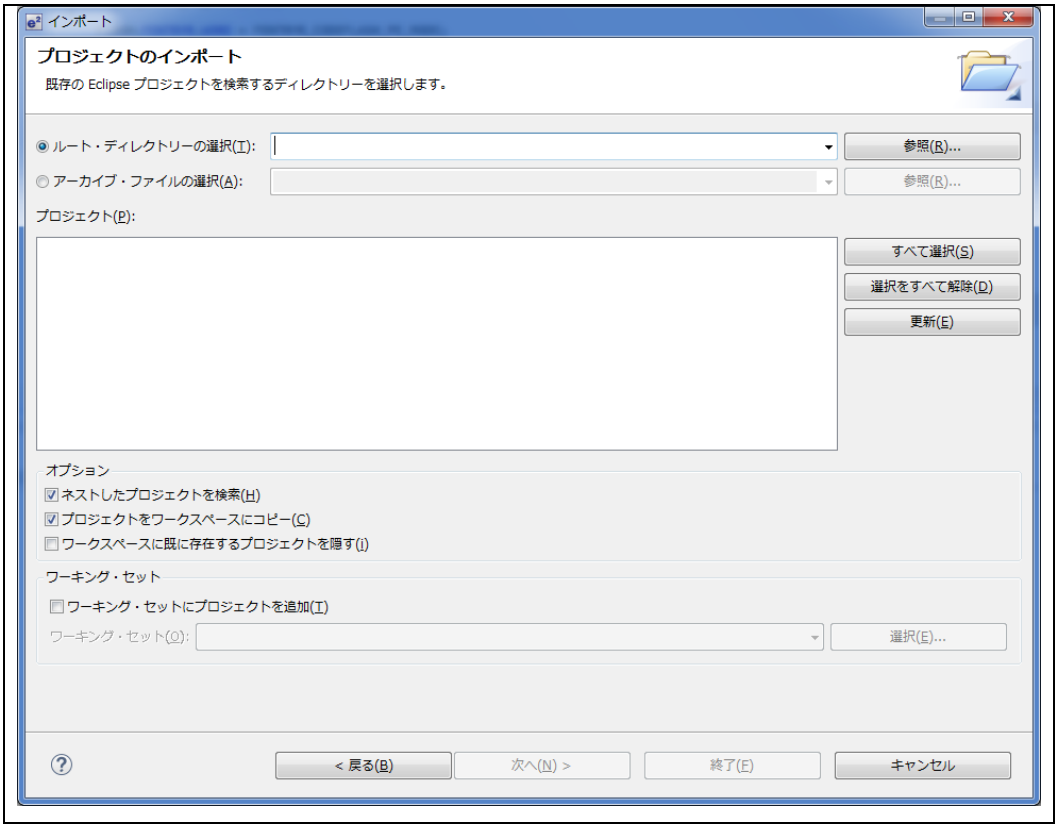


Figure 9-2　プロジェクトのインポート画面

(5). [終了]をクリック して下さい。

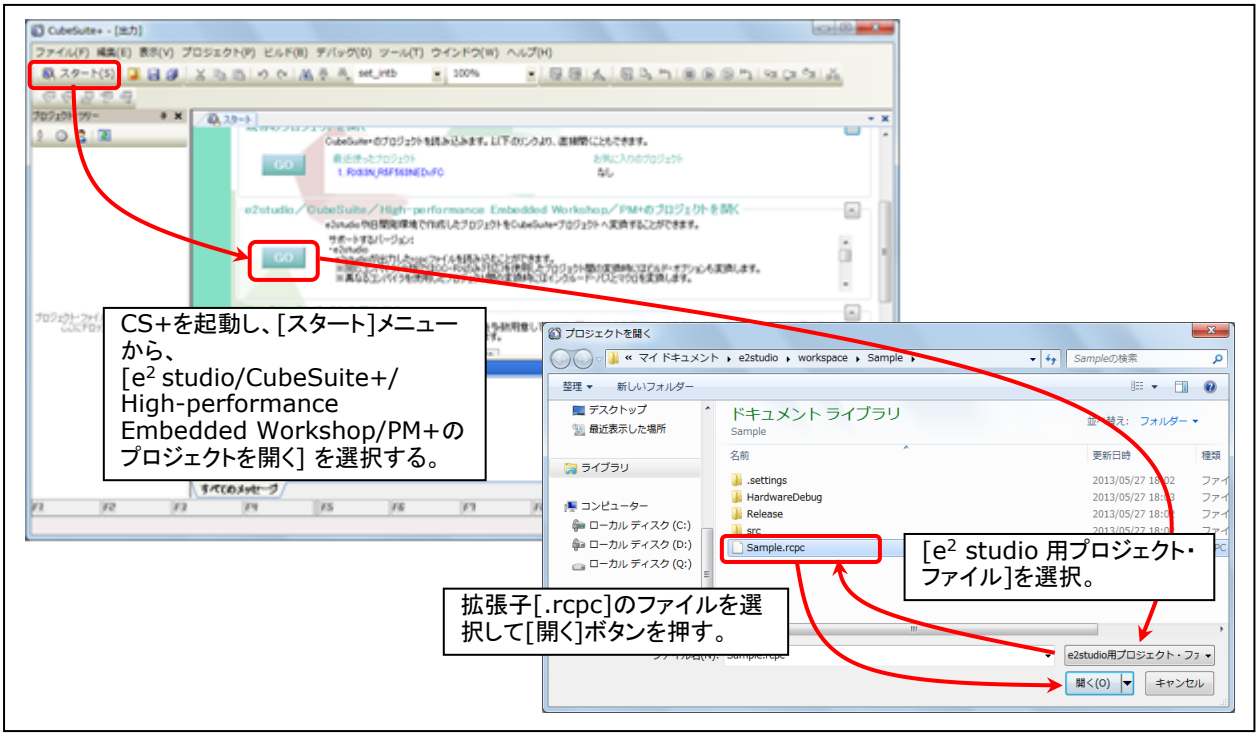
プロジェクトのワークスペースへのインポートが完了します。

10. 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、workspce\RL78\CCRL\(MCU 名)フォルダ内に用意されています。



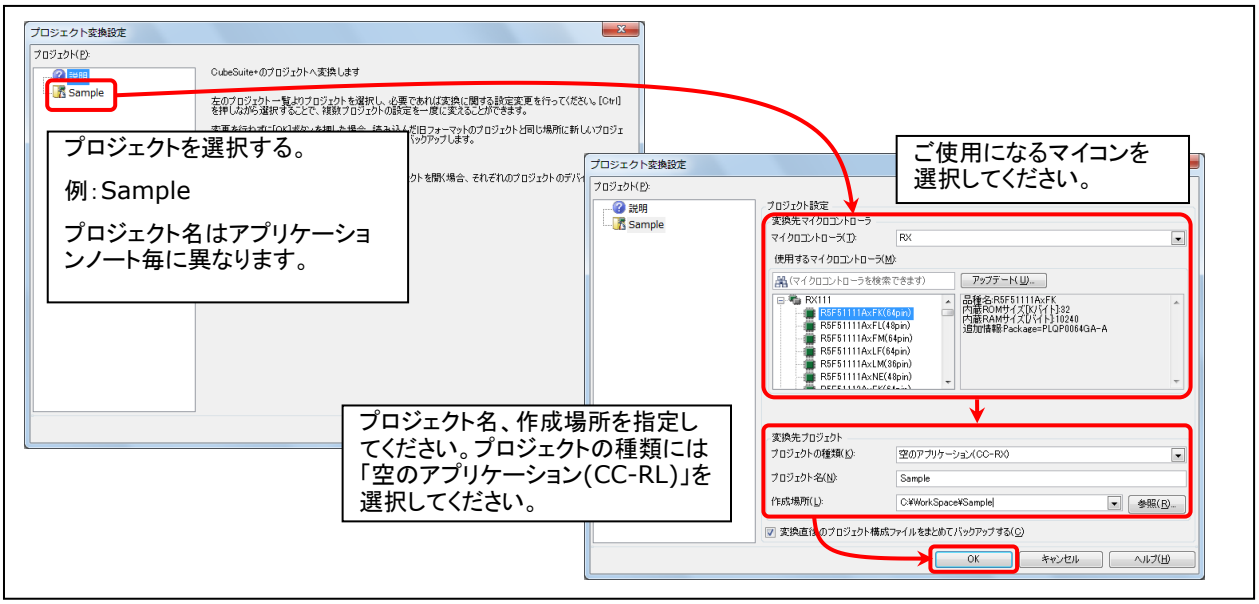


Figure 10-1 e² studio 用プロジェクトの CS+読み込み方法

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	2011.04.28	—	初版発行
2.00	2012.11.30	—	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
2.10	2013.08.01	—	RX111 に対応、誤記訂正
2.11	2013.10.31	—	3.3.1 フォルダ構成を変更。これに伴い、1.4 のパス表記を修正。誤記訂正
2.12	2014.03.31	—	R8C に対応、誤記訂正
2.13	2015.03.16	—	動作確認デバイスから RX111 を削除。
2.14	2016.01.18	—	Technical Update(発行番号: TN-RL*-A055A/J)に対応しました。
2.15	2016.03.28	—	CC-RL コンパイラをサポート

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/