Renesas USB MCU

R01AN0664JJ0215
Rev.2.15
Mar 28, 2016

USB Host Human Interface Device Class Driver (HHID) using Basic Mini Firmware

要旨

本資料は、Renesas USB MCU の USB Basic Mini Firmware を使用した USB Host Human Interface Device Class Driver (HHID) 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 Human Interface Device Class Driver (HHID) 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 Human Interface Device Class Driver (HHID)は USB ヒューマンインターフェースデバイスクラス仕様（以降 HID と記述）に準拠し、HID ペリフェラルデバイスとの通信を行うことが出来ます。

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 Human Interface Devices Version 1.1
3. HID Usage Tables Version 1.1
   [http://www.usb.org/developers/docs/]
4. 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
cstd	:	USB-BASIC-F/Wの Peripheral & Host共通関数のprefix
CS+	:	ルネサス統合開発環境
Data Transfer	:	Generic name of Control transfer, Bulk transfer and Interrupt transfer
HCD	:	Host control driver of USB-BASIC-F/W
HDCD	:	Host device class driver (device driver and USB class driver)
HEW	:	High-performance Embedded Workshop
HHID	:	Host human interface device
HID	:	Human interface device class
HM	:	Hardware Manual
hstd	:	Prefix for host function of USB-BASIC-F/W
KBD	:	Keyboard device
MGR	:	Peripheral device state manager of HCD
MSE	:	Mouse device
PP	:	プリプロセス定義
RSK	:	Renesas Starter Kit
SW1/SW2/SW3	:	User switches on RSK
USB	:	Universal Serial Bus
USB-BASIC-FW	:	USB-BASIC-F/W (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 HID 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。HHID 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_hhid_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

HHID は HID クラスドライバと、マウス、キーボードのデバイスドライバから構成されます。

Figure 4.1 に HHID 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。

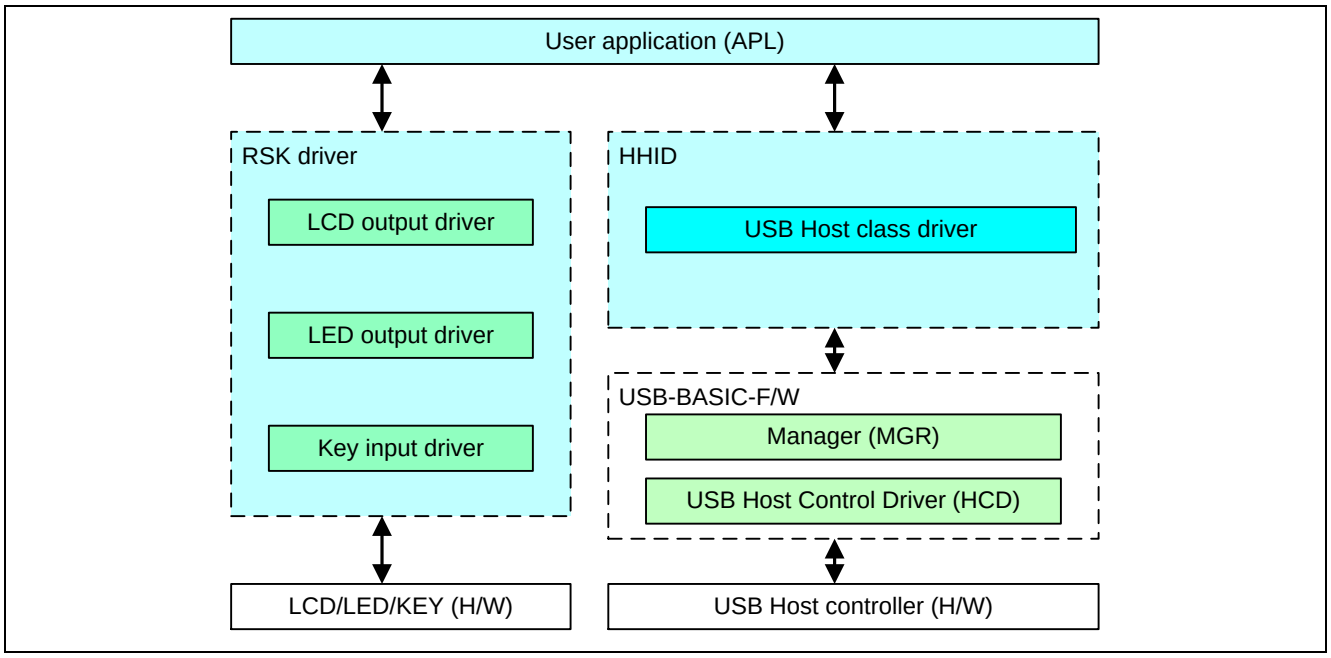


Figure 4.1 モジュール構成図

Table 4-1 モジュール機能概要

モジュール名	機能概要	備考
APL	ユーザアプリケーションプログラムです。 スイッチ 操作によりHIDデバイスとの通信開始、サスペンド、レジューム制御を行います。 HID デバイスから受信したレポート情報を LCD に表示します。	ユーザ作成
HHID	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 HHID 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) HID リクエストによる接続デバイスの制御 2) 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 ペリフェラルからデータを受け取り、受信したデータを LCD 表示する。
   a) USB マウス接続時(Mouse モードと呼ぶ)は、X 軸、Y 軸の移動量を LCD に数値表示する。また、クリックボタン押下で LED 点灯を行う。
   b) USB キーボード接続時(Keyboard モードと呼ぶ)は、LCD に入力されたキーデータを 1 文字表示する。またデバイスが通常動作時に *NumLock LED* を点灯しサスペンド時は *NumLock LED* を消灯する。
2. USB デバイスをサスペンド／レジューム操作する。
   a) RSK 上の SW3 押下により USB デバイスを交互にサスペンド、レジュームする。
   b) USB デバイスからリモートウェイクアップ信号を受け取るとレジュームする。

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
データ転送開始	レポート受信要求を出します。	SW2
デバイス状態変更	USB デバイスの状態を以下のように変更します。 データ受信待ち状態：サスペンド状態に移行 サスペンド状態 ：データ受信待ち状態に移行	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
              ＋――― HIDFW [Human Interface Device Class driver ]    Table 4-3 参照
              │        ＋――― inc                    HID ドライバ共通ヘッダファイル
              │        ＋――― src                    HID ドライバ
              ＋―――SmplMain [サンプルアプリケーション]
              │        ＋――― APL                    レポート表示アプリケーション
              ＋―――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 HHID 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
HIDFW/inc	r_usb_class_usrcfg.h	USB ホスト HID ユーザ定義マクロ	
HIDFW/inc	r_usb_hhid_define.h	HHID 型定義、マクロ定義	
HIDFW/inc	r_usb_hhid_api.h	HHID API 関数プロトタイプ宣言	
HIDFW/src	r_usb_hhid_api.c	HHID API 関数	
HIDFW/src	r_usb_hhid_driver.c	HHID ドライバ関数	
SmplMain	main.c	メインループ処理	
SmplMain/APL	r_usb_hhid_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

HHID 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_HHID_TSK	**HHID** (R_usb_hhid_task) Task ID: USB_HHID_TSK Task priority: 2	
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_HHID_MBX / USB_HHID_TSK	HHID -> HHID / APL -> HHID 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 HID 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 HID ペリフェラルデバイスが接続されたとき、受信した USB データの表示を行います。 HHID アプリケーションは USB ヒューマンインタフェースクラス規格に従います。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 および Figure 5.2 に本ソフトウェアの動作環境を示します。

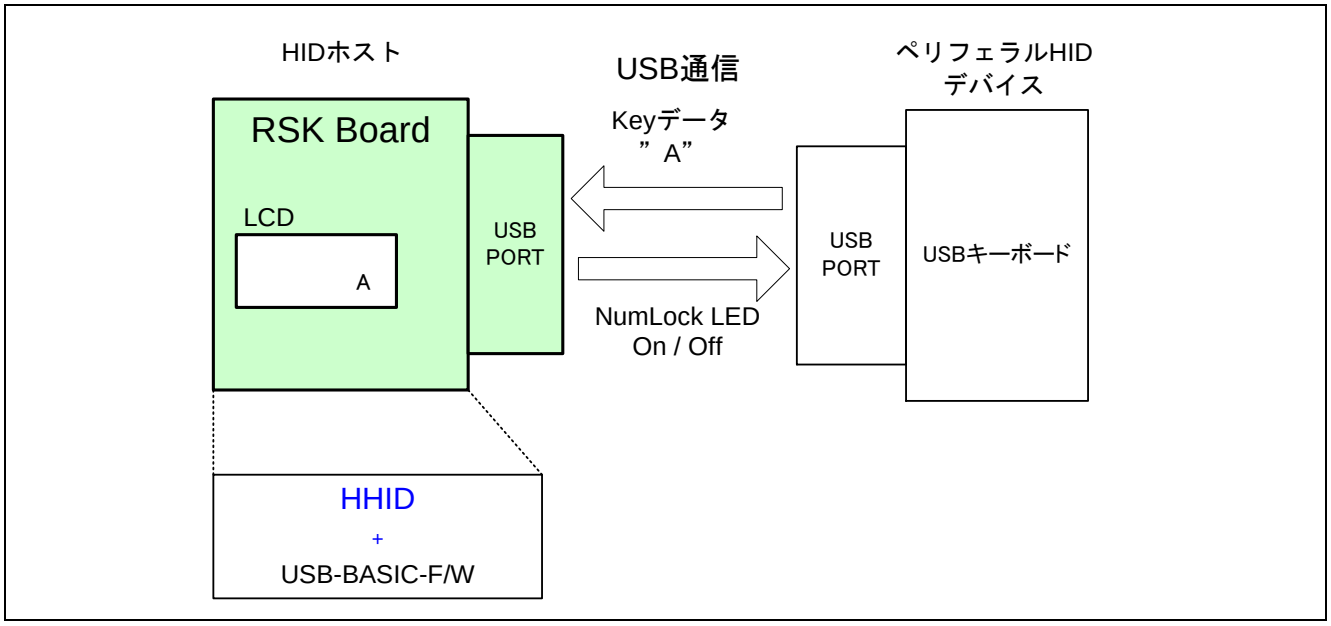


Figure 5.1 動作環境例 (Keyboard 接続時)

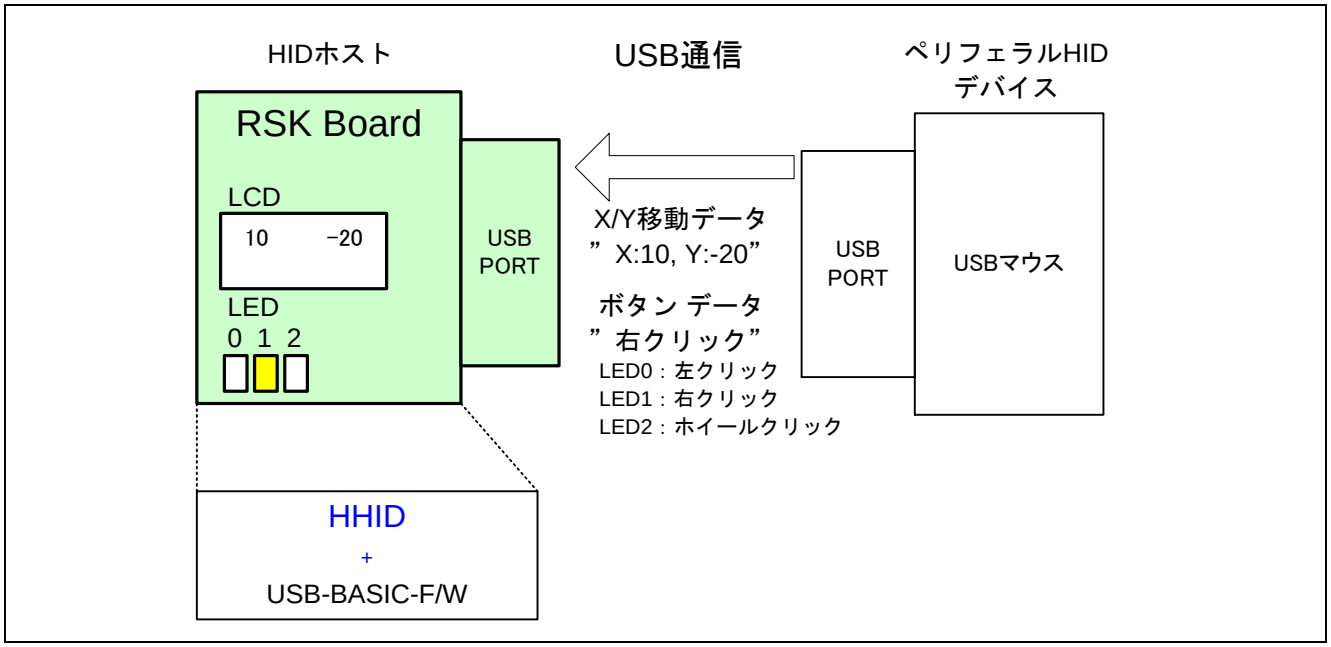


Figure 5.2 動作環境例 (Mouse 接続時)

5.1.1 レポート受信

r_usb_class_usrcfg.h ファイルの USB_HHID_GET_REPORT_PIPE0 マクロを有効にするとレポート受信をコントロール転送（GET_REPORT リクエスト）で行います。

5.2 アプリケーションプログラム説明

Figure 5.4 に関するアプリケーション操作は以下のとおりです。

- HID ペリフェラルデバイスの接続 (No.0-1 処理に対応)

  接続された USB デバイスがマウスであるかキーボードであるかを自動判別します。マウスもしくはキーボードはコンフィグレーションディスクリプタの *bInterfaceProtocol* で判別します（Table 5-1 参照）。本アプリケーションプログラムはレポートディスクリプタを解析しません。

- データ通信開始（処理 No.1-1）
  SW2 の押下により、USB デバイスと通信を開始します（Table 4-2 参照）。

- データ通信完了（処理 No.2-1）
  データ通信が完了すると HHID からコールバック関数が呼び出されます。

- データ通信中の操作（処理 No.2-2）
  SW3 の押下によりデータ通信を停止し、USB デバイスをサスペンドに移行させます（Table 4-2 参照）。

- サスペンド中の操作（処理 No.4-1）
  SW3 の押下により USB デバイスをレジュームし、データ通信を再開します。（Table 4-2 参照）。

- データ通信中の動作（処理 No.3-1）
  USB デバイスから受信したレポートを解析し、LCD 表示を行います（Table 5-2 参照）。

Table 5-1 モード判定

bInterfaceProtocol	Mode	説明
0x01	Keyboard モード	キーボードデバイス接続と認識
0x02	Mouse モード	マウスデバイス接続と認識
その他	動作不可	動作可能な HID デバイスと認識されません

Table 5-2 D データ通信中の動作

モード	説明
Keyboard モード	受信したキーデータの表示（キーコード ASCII 変換）
Mouse モード	受信した座標データの表示

5.3 エンドポイント仕様

HHID が使用するエンドポイント仕様を Table 5-3. に示します。

Table 5-3 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	6	Interrupt In	デバイスからホストへのデータ転送

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。

5.4 接続する HID ペリフェラルデバイスについて

FullSpeed/LowSpeed のキーボードは接続可能です。

FullSpeed/LowSpeed の 3 ボタンマウスは接続可能です。

HUB 内蔵デバイスや複合デバイスは接続できません。

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 5-4 に示します。

Table 5-4 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_hsmpl_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_hhid_MainTask	HHID サンプルアプリケーションタスク
usb_hapl_registration	HHID ドライバ登録
usb_hhid_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_hhid_smpl_data_trans_result	データ転送完了処理
usb_hhid_smpl_mse_data	マウスデータ受信処理
usb_hhid_smpl_val_to_str	数値を文字列に変換
usb_hhid_smpl_kbd_data	キーボードデータ受信処理
usb_hhid_smp_status_set	サンプルアプリケーションモード設定処理
usb_hhid_smpl_get_hid_descriptor	HID ディスクリプタ取得処理(未使用)
usb_hhid_smpl_get_report_descriptor	リポートディスクリプタ取得処理(未使用)
usb_hhid_smpl_get_physical_descriptor	フィジカルディスクリプタ取得処理(未使用)
usb_hhid_smpl_kbd_led_ctl	キーボード LED 点灯/非点灯設定処理
usb_hhid_smpl_set_report	SET_REPORT リクエスト送信処理(未使用)
usb_hhid_smpl_get_report	GET_REPORT リクエスト送信処理(未使用)
usb_hhid_smpl_set_idle	SET_IDLE リクエスト送信処理(未使用)
usb_hhid_smpl_get_idle	GET_IDLE リクエスト送信処理(未使用)
usb_hhid_smpl_set_protocol	SET_PROTOCOL リクエスト送信処理(未使用)
usb_hhid_smpl_get_protocol	GET_PROTOCOL リクエスト送信処理(未使用)
usb_hsmpl_class_result	HID クラスリクエストコールバック
usb_hhid_smpl_get_report_result	GET REPORT リクエストコールバック処理
usb_hhid_smpl_kbd_led_ctl_result	SET REPORT リクエストコールバック処理

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 LCD 表示

本 APL は、USB デバイスの接続状態、及び接続された USB デバイスから受信したレポート内容を LCD に表示します。

- マウス接続時　：X/Y 軸の移動量（-128～127）を LCD に表示する。
- キーボード接続時：最後に入力されたキーデータを LCD に表示する。

USB デバイスから受信レポートの内容が NULL（キーボード非押下時、マウスの X/Y 軸移動なし）の場合、LCD 表示を更新しません。LCD 表示状態遷移を Figure 5.3 に示します。

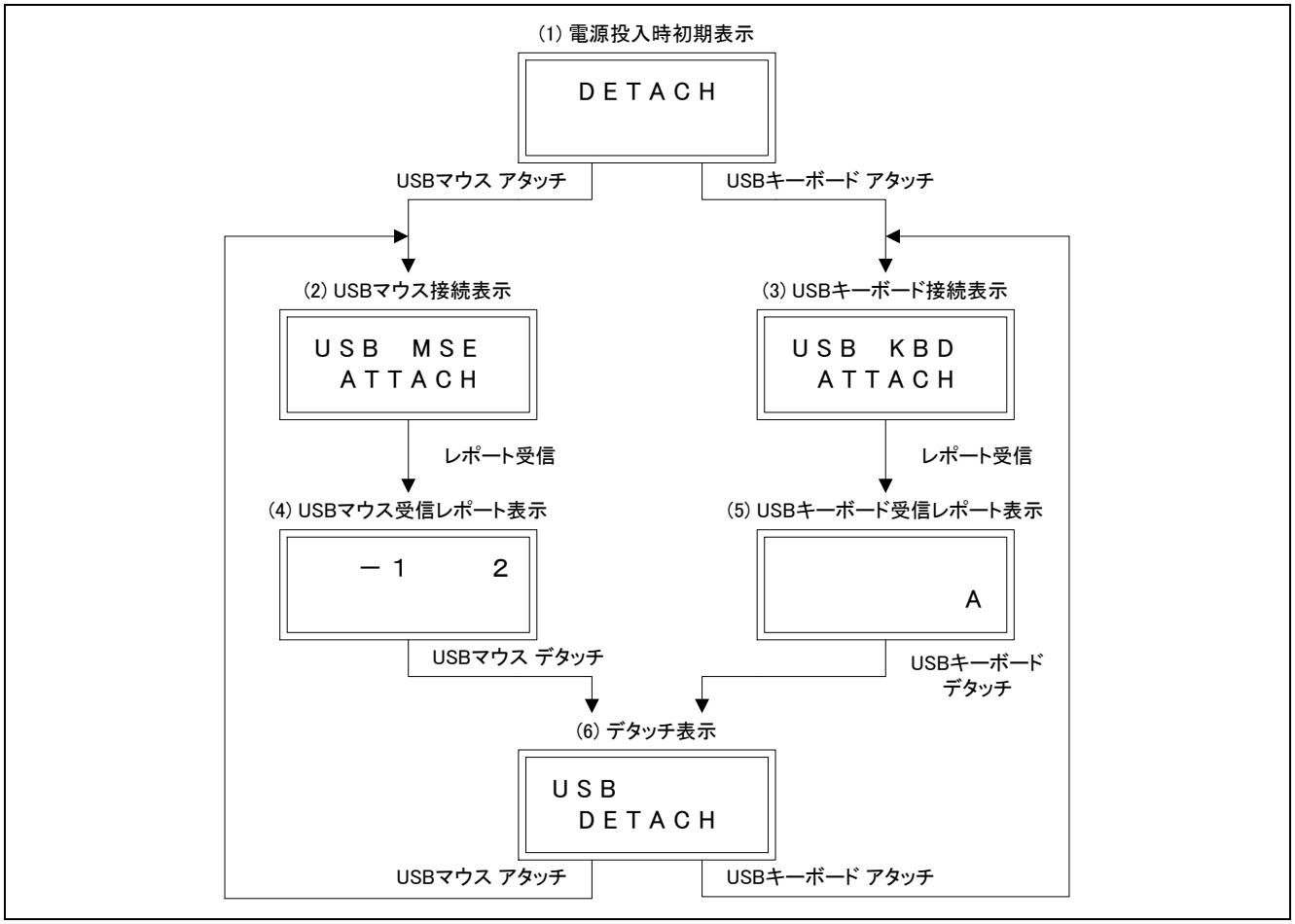


Figure 5.3 LCD 表示状態遷移図

5.6.2 APL 状態遷移

Figure 5.4 に APL 状態遷移図を示します。 各ブロックは状態遷移プログラムです。

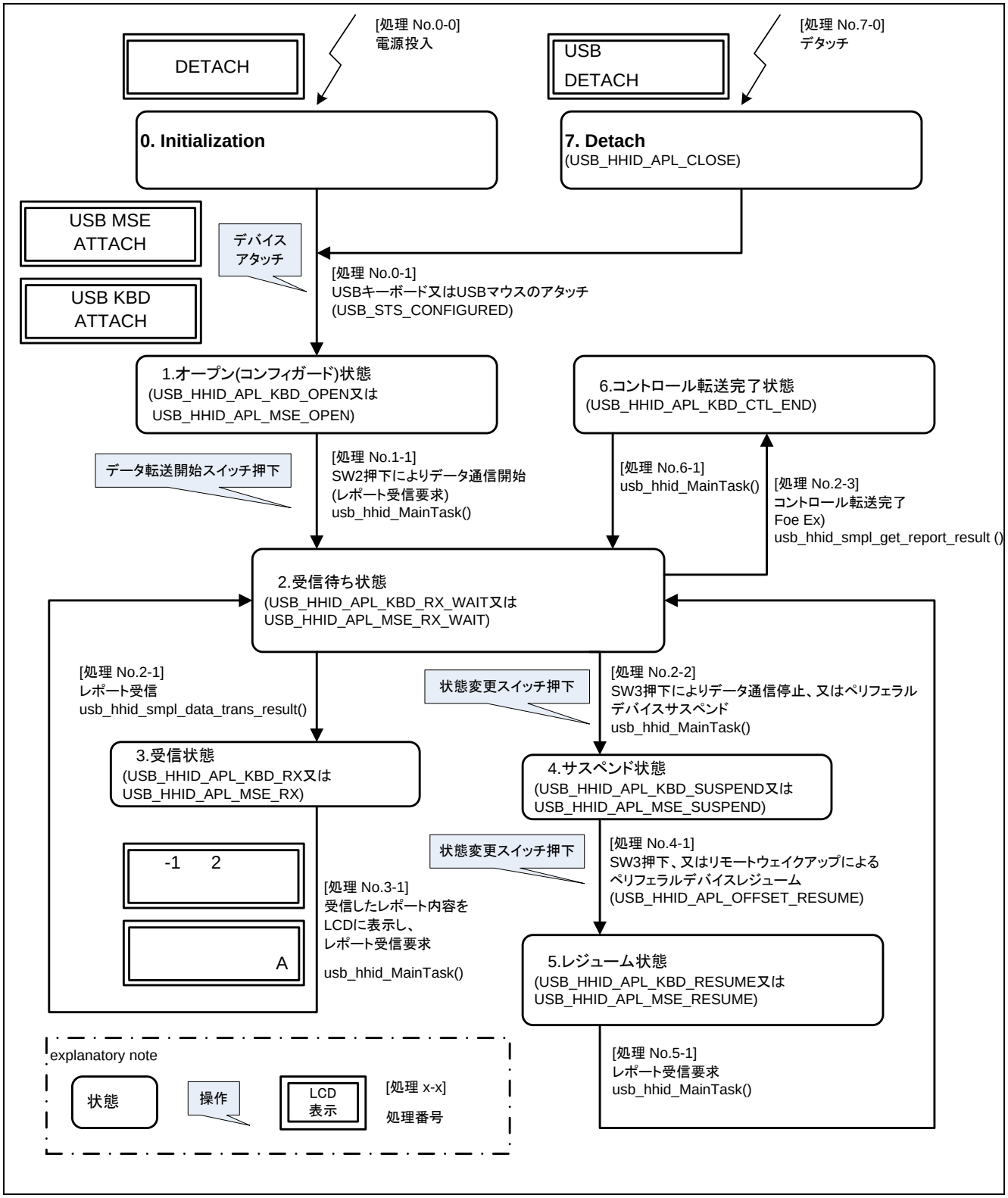


Figure 5.4 APL 状態遷移図

5.7 処理フロー図

Figure 5.5 に、APL タスク処理概要フローを示します。

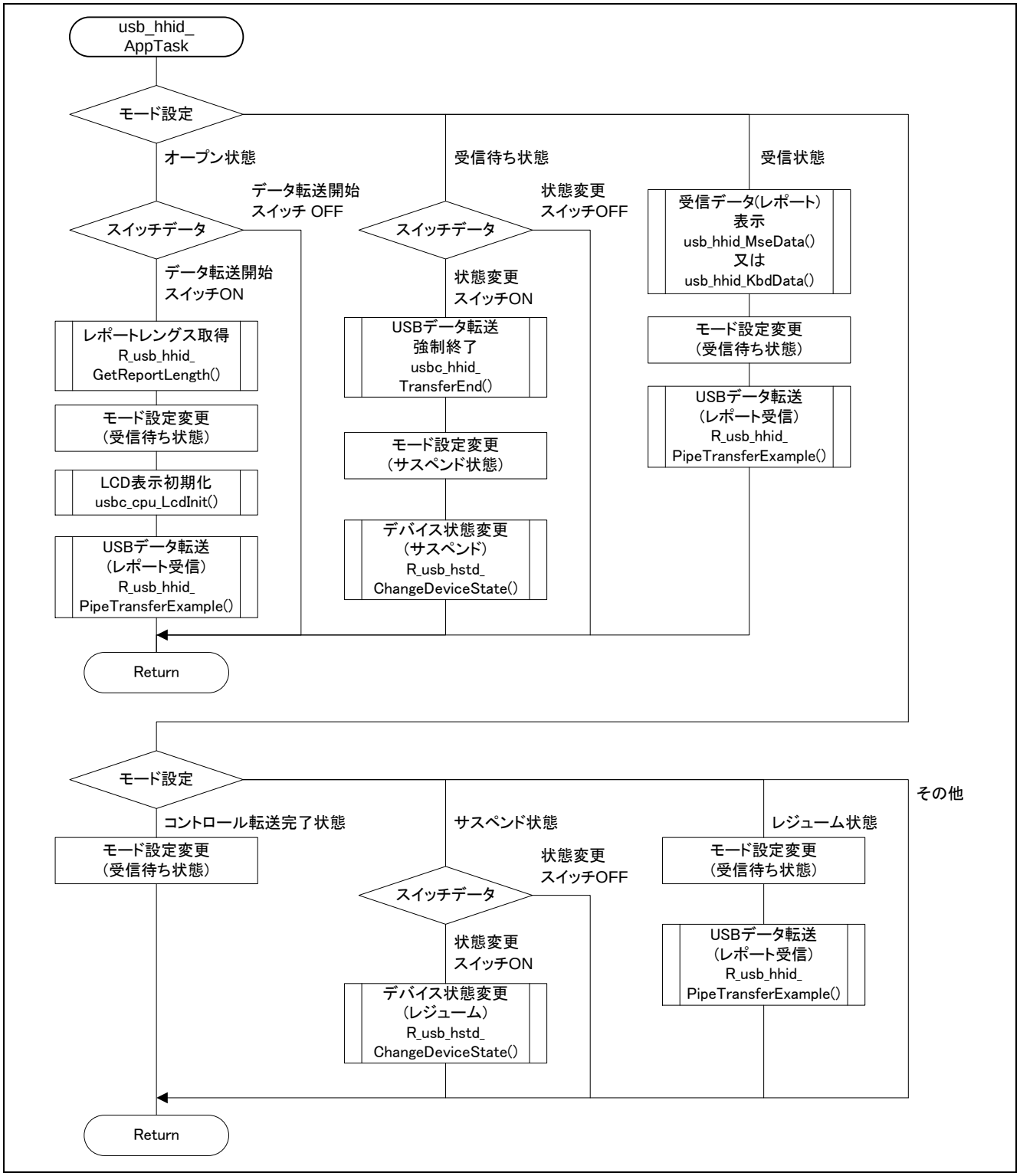


Figure 5.5 APL タスク処理概要フロー

5.8 シーケンスチャート APL-HHID-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、HID 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までのシーケンスを Figure 5.6 に示します。

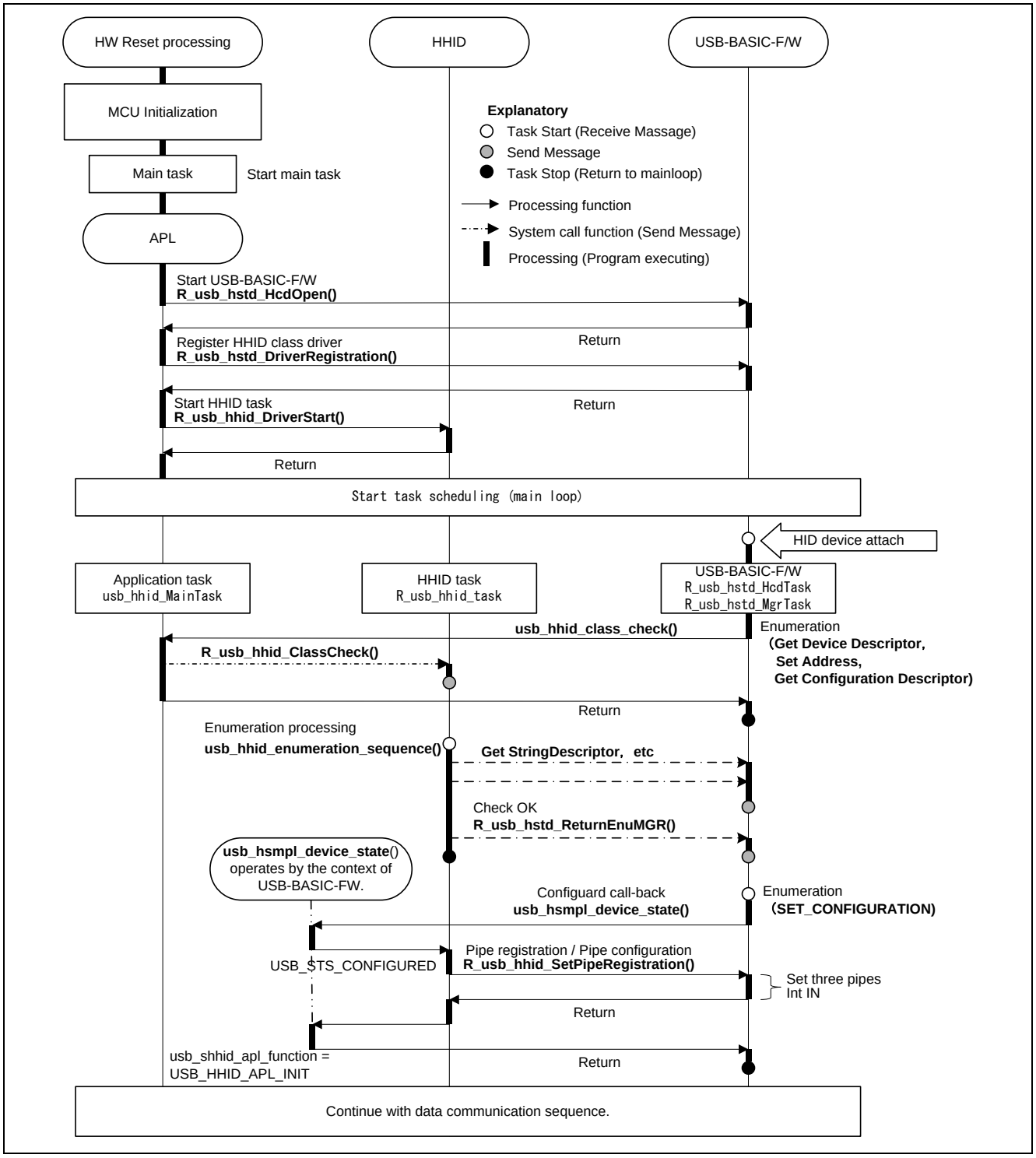


Figure 5.6 起動から HID デバイスアタッチのシーケンス

5.8.2 データ通信

Figure 5.7 と Figure 5.8 にキーボードデバイスが接続された場合のデータ転送シーケンスを示しています。レポートをインタラプト転送で受け取るケースは Figure 5.7 です。レポートをコントロール転送で受け取るケースは Figure 5.8 です。

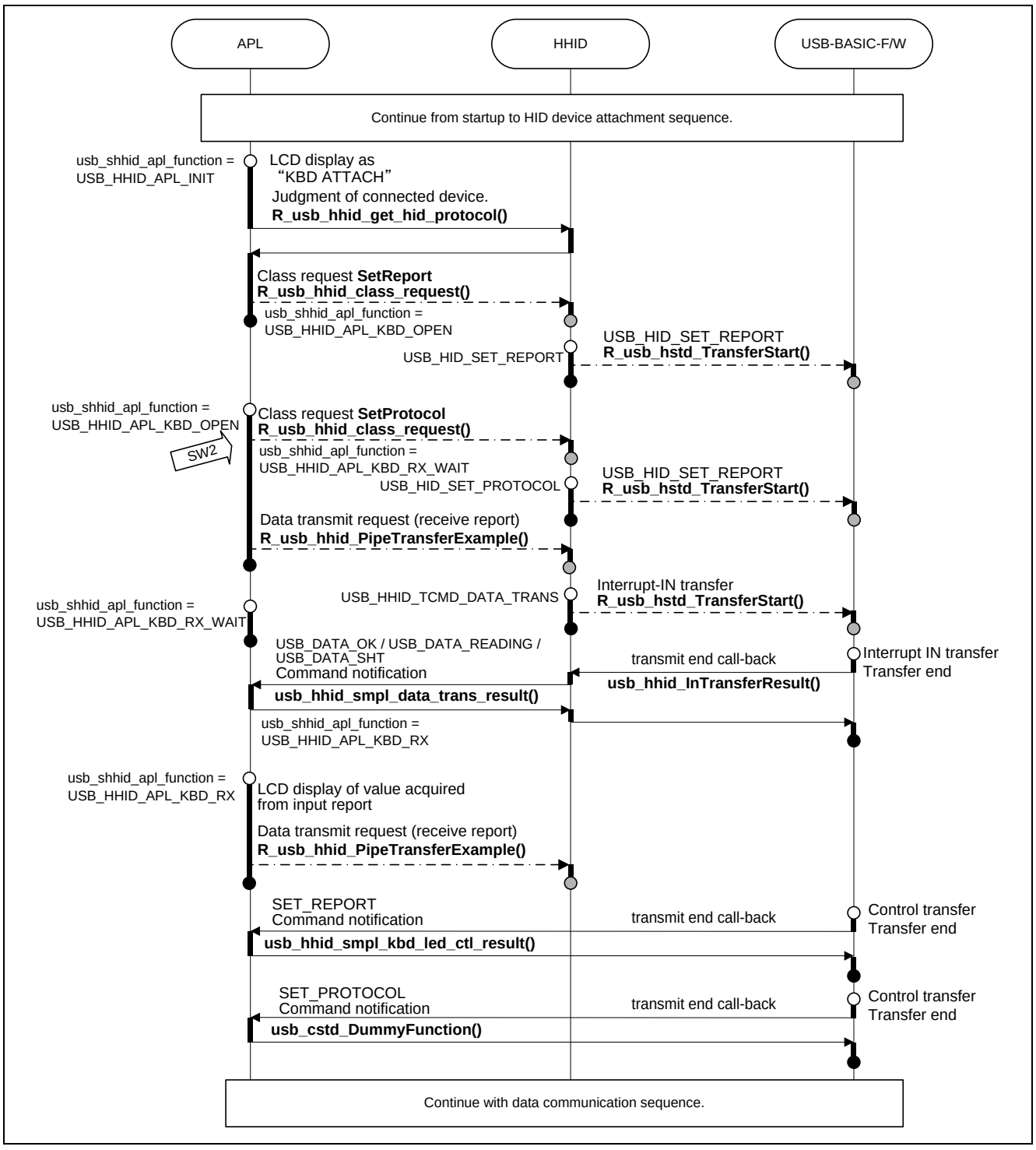


Figure 5.7 データ転送シーケンス（キーボード Interrupt-IN）

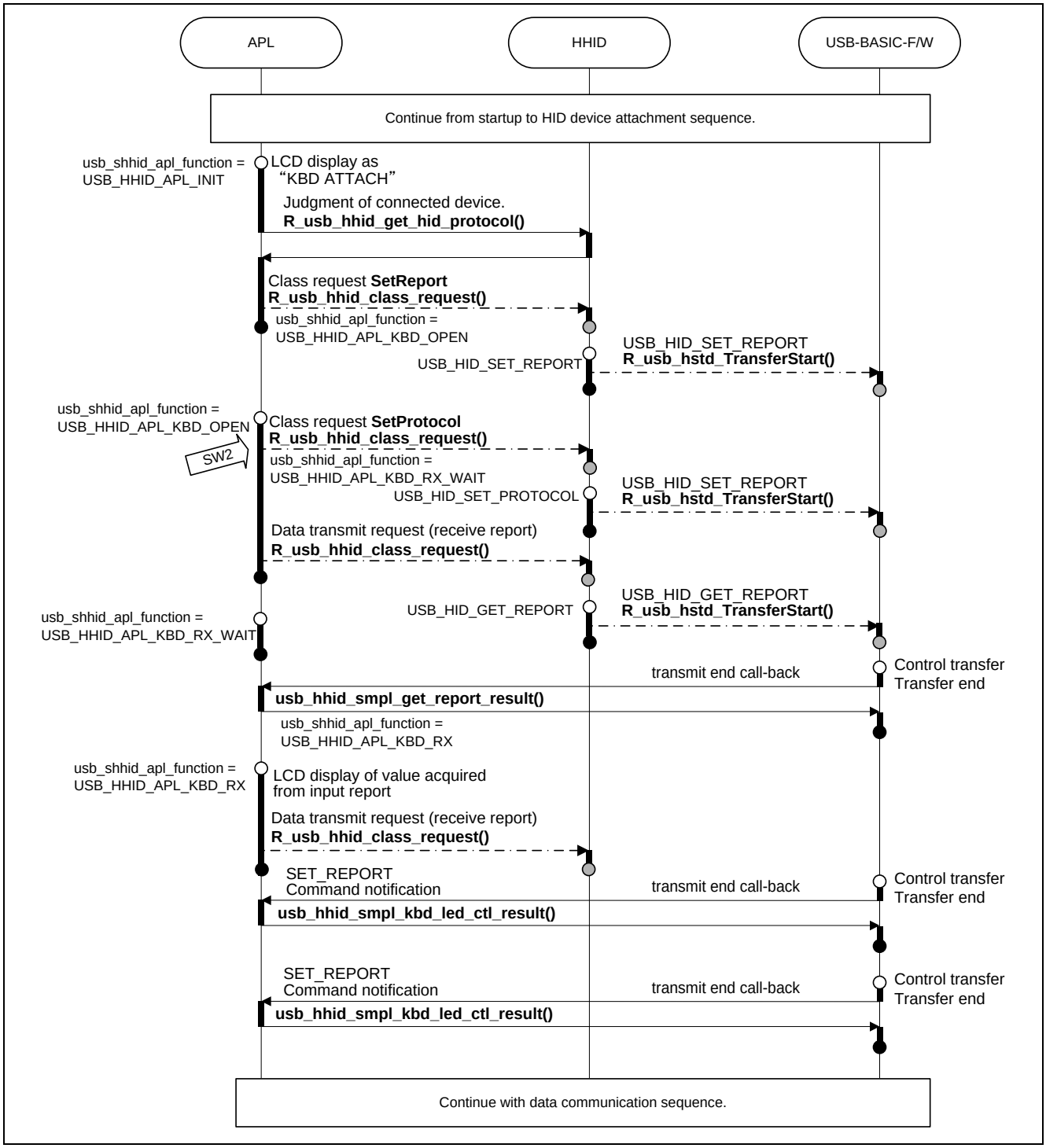


Figure 5.8 データ転送シーケンス（キーボード Control transfer）

Figure 5.9 と Figure 5.10 にマウスデバイスが接続された場合のデータ転送シーケンスを示しています。レポートをインタラプト転送で受け取るケースは Figure 5.9 です。レポートをコントロール転送で受け取るケースは Figure 5.10 です。

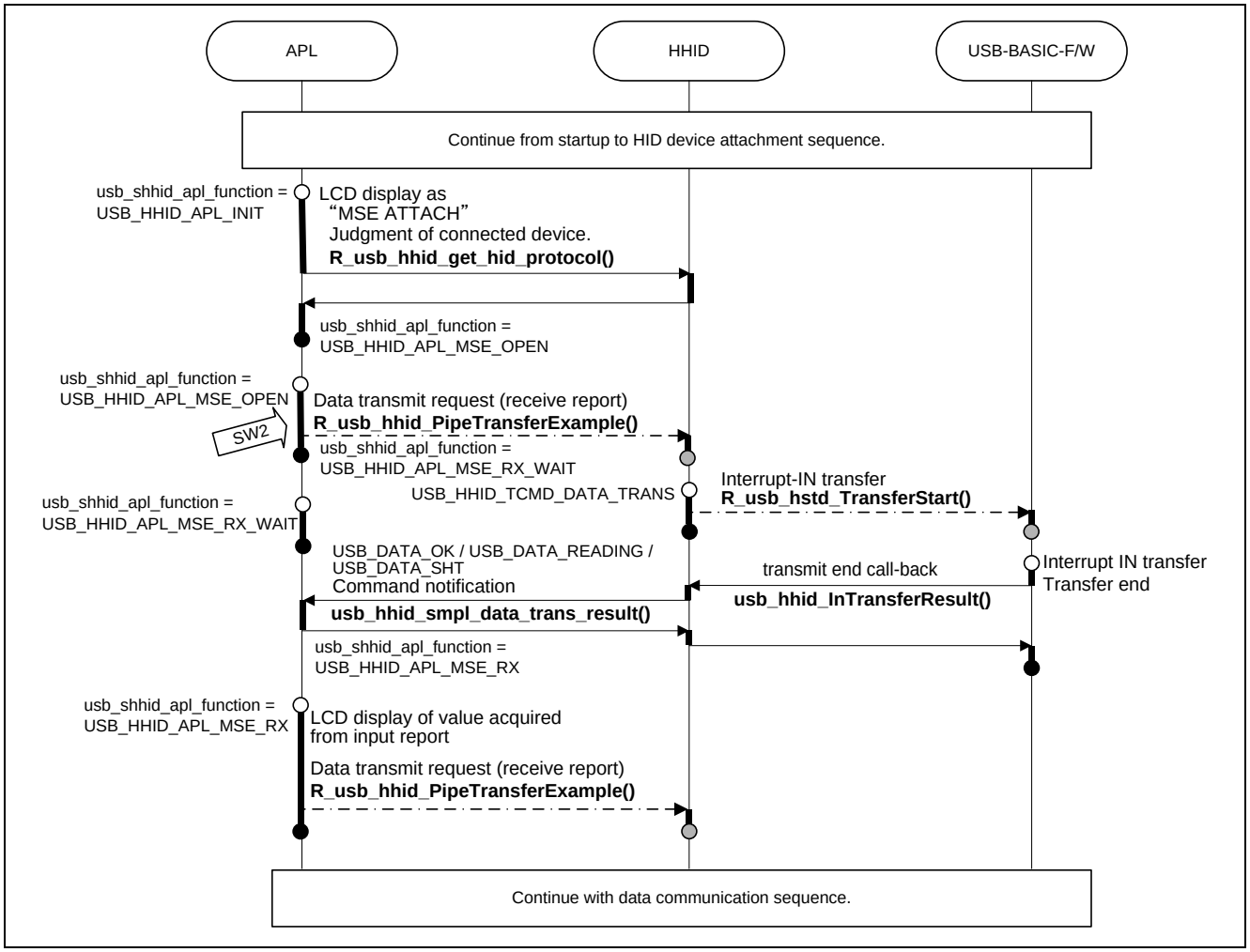


Figure 5.9 データ転送シーケンス（マウス Interrupt-IN）

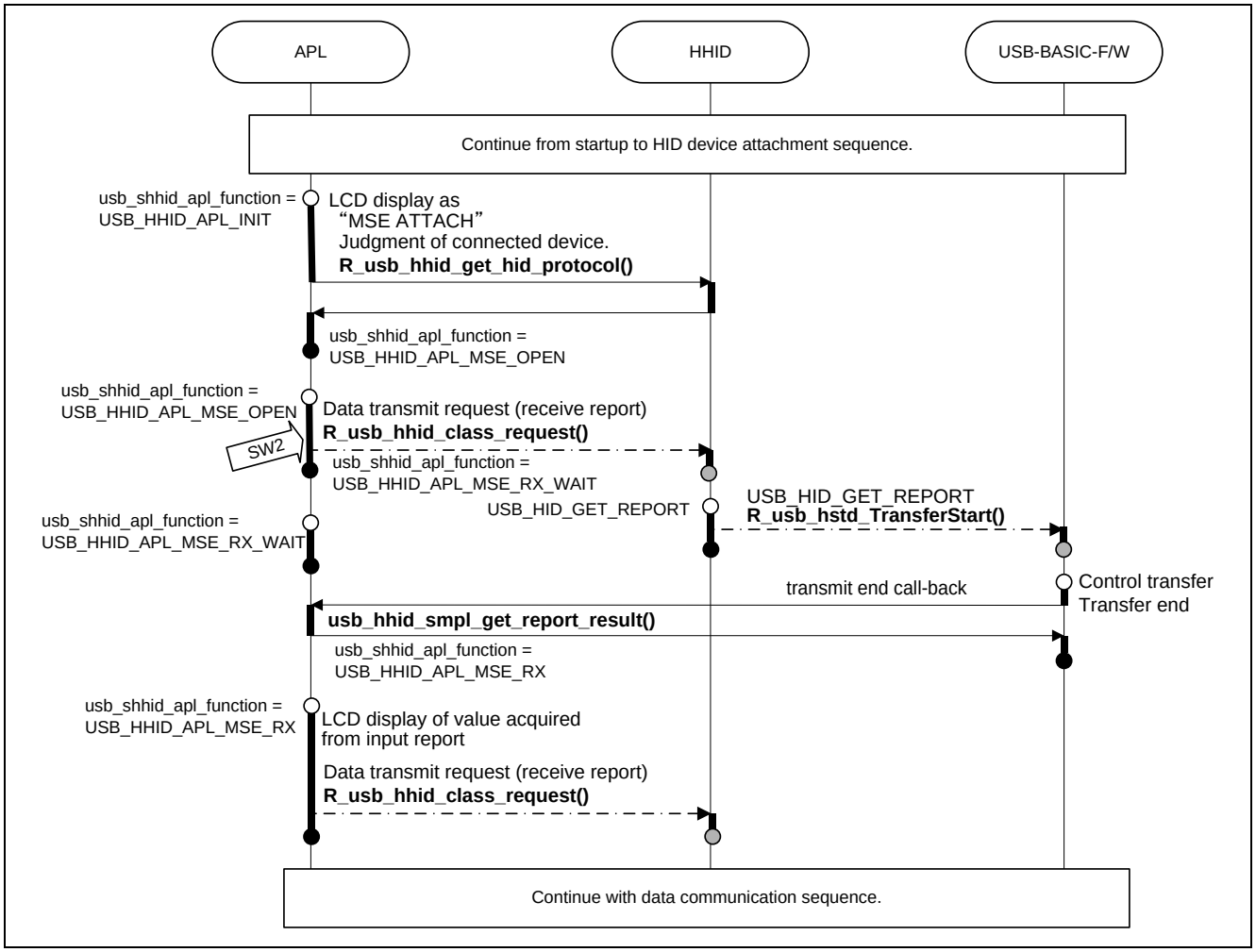


Figure 5.10 データ転送シーケンス（マウス Control transfer）

5.8.3 HID 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 5.11 に示します。

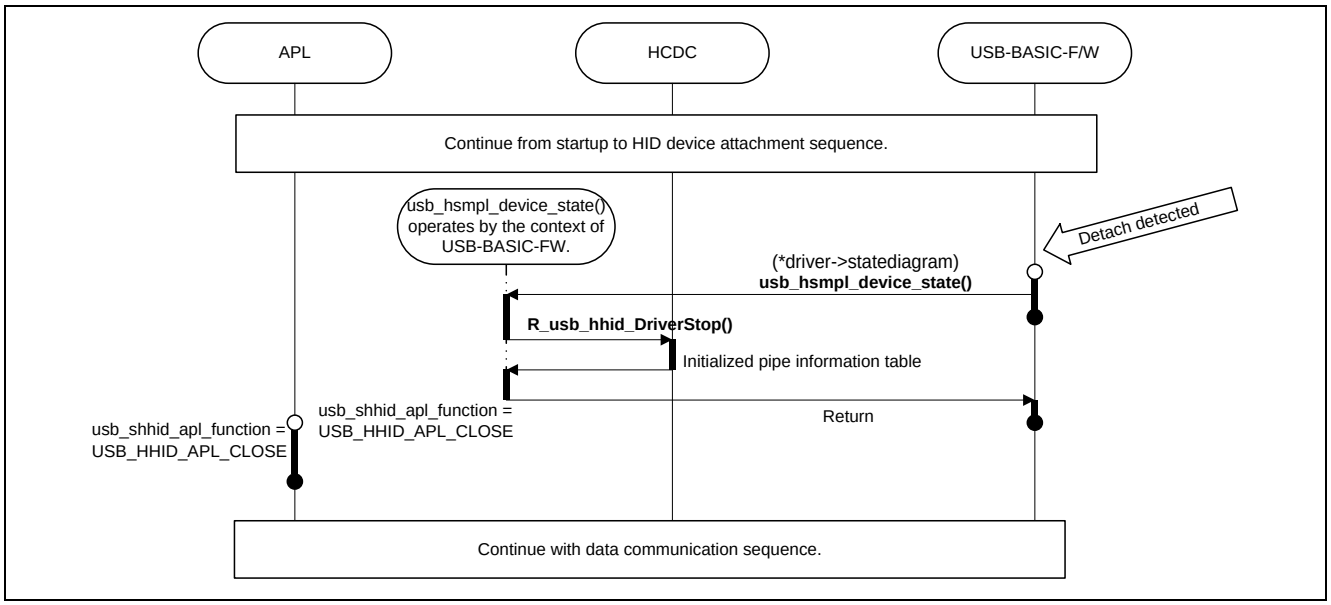


Figure 5.11 HID デバイスデタッチシーケンス

5.8.4 HID デバイスサスペンド、レジューム

HID デバイスサスペンド、レジューム時のシーケンスを Figure 5.12 と Figure 5.13 に示します。.

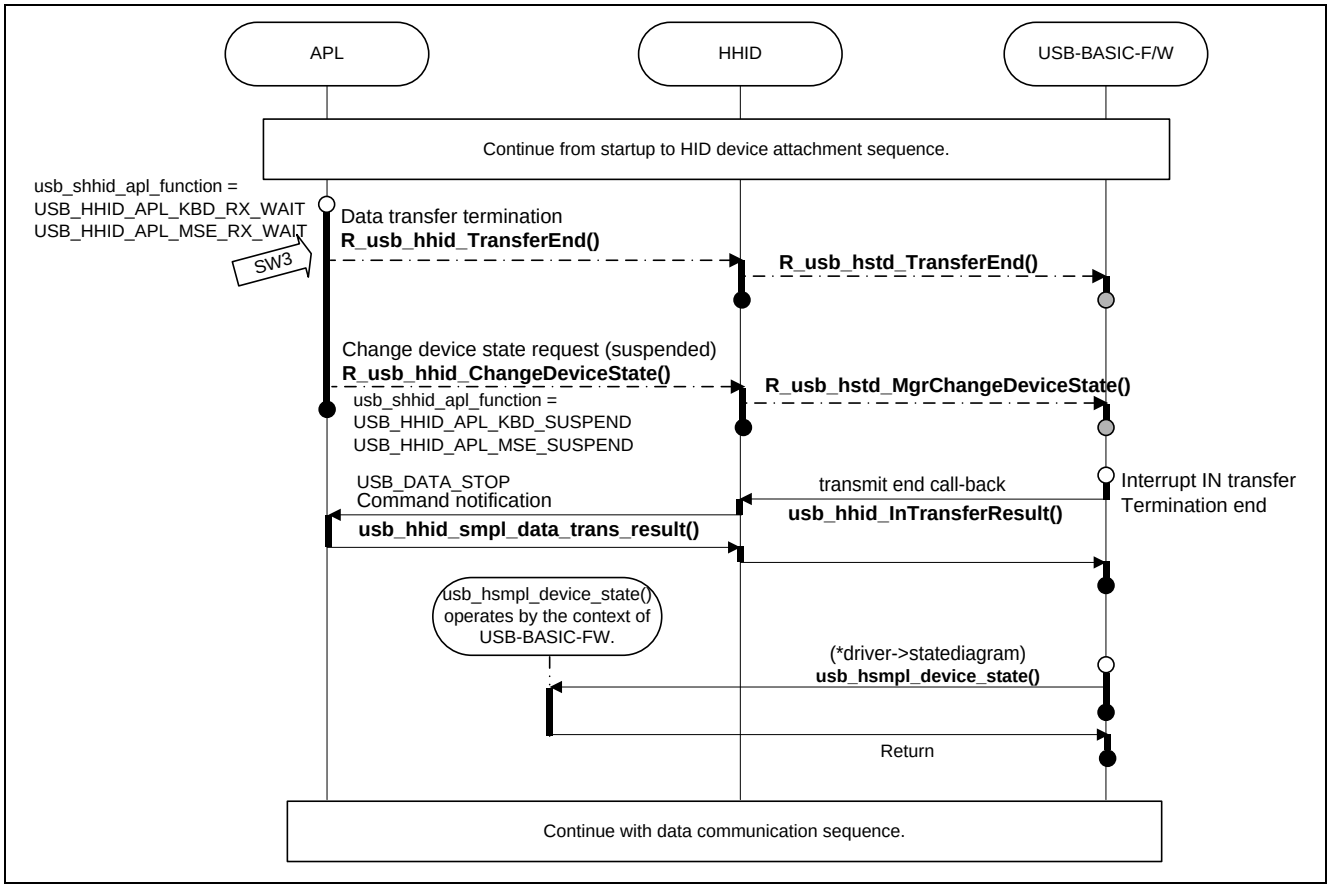


Figure 5.12 HID デバイスサスペンドシーケンス

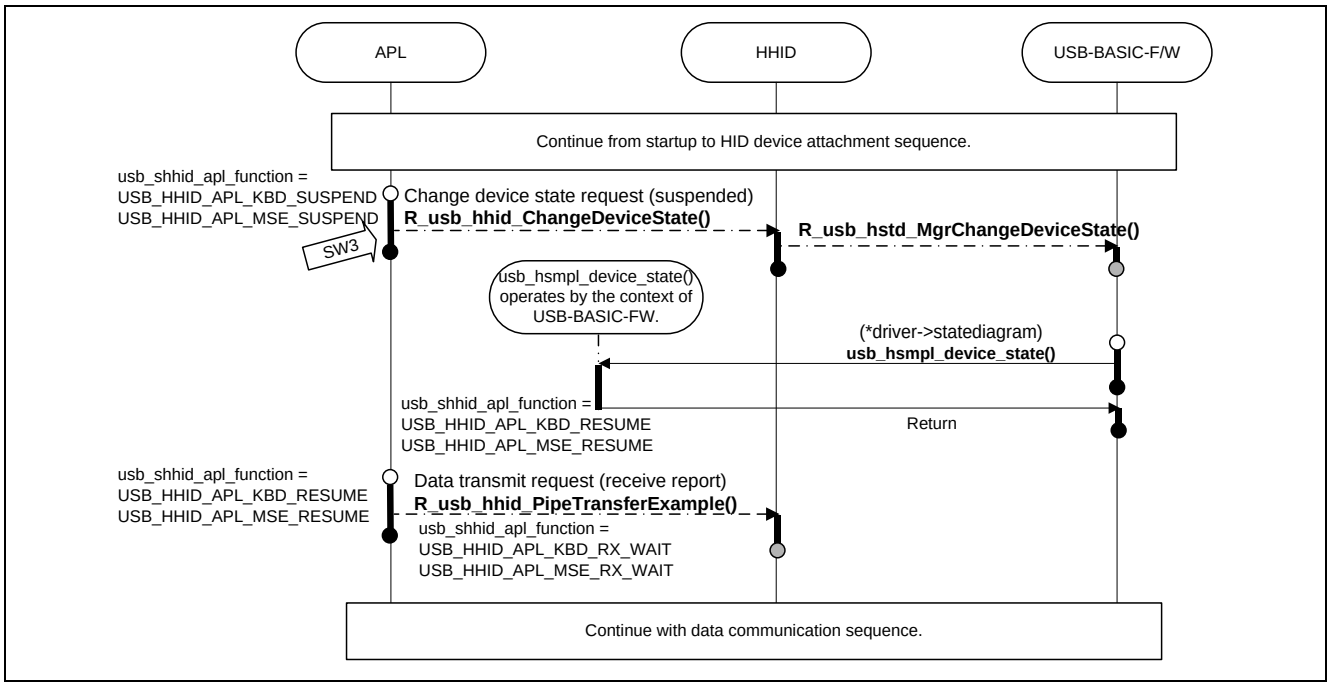


Figure 5.13 HID デバイスレジュームシーケンス

6. Human Interface Device Class (HID)

本ソフトウェアは Human Interface Device Class 仕様に準拠します。詳細は 1.2 章を参照してください。

HID クラスはコンピュータシステムを人間が操作、制御する主要なデバイスです。

HID クラスデバイスの例を以下に示します。
キーボードやポインティング： 標準マウス、トラックボール、およびジョイスティック。
フロントパネル制御： ノブ、スイッチ、ボタン、およびスライダー。
電話、VCR のリモートコントロール、ゲームまたはシミュレーション装置などの制御
データグローブ、スロットル、ハンドル、およびラダーペダル。

6.1 基本機能

HHID 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. デバイスの能力と状態について問い合わせ.
3. 出力状態と特徴項目の設定
4. HID ペリフェラルデバイスからデータ受信

6.2 HID クラスリクエスト（ホスト→デバイスへの要求）

Table 6-1 に HID で対応しているクラスリクエストを示します。

Table 6-1 HID クラスリクエスト

リクエスト	コード	説明	対応
Get_Report	0x01	HID デバイスにレポートを要求する	Yes
Set_Report	0x09	HID デバイスにレポートを通知する	Yes
Get_Idle	0x02	HID デバイスに Duration 時間を要求する	No
Set_Idle	0x0A	HID デバイスに Duration 時間を通知する	No
Get_Protocol	0x03	HID デバイスにプロトコルを要求する	No
Set_Protocol	0x0B	HID デバイスにプロトコルを通知する	No
Ger_Report_Descriptor	Standard	レポートディスクリプタを要求する	Yes
Get_Hid_Descriptor	Standard	HID ディスクリプタを要求する	Yes

詳細は“USB Device Class Definitions for Human Interface Devices, Revision 1.1”の 7 章を参照ください。

7. USB ホストヒューマンインタフェースデバイスクラスドライバ（HHID）

7.1 基本機能

本ソフトウェアは、ヒューマンインタフェースデバイスクラス仕様に準拠しています。1.2 章を参照してください。

HHID の基本機能を以下に示します。

1. HID デバイスに対し、HID クラスリクエストを発行する。
2. HID デバイスとデータ送受信を行う。

7.2 HHID タスク説明

本タスクは、USB_HHID_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 HHID 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_HHID_ 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。	R_usb_hhid_ClassCheck() usb_cstd_AnsiCallBack() 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_HHID_ TCMD_DATA_TRANS	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_hhid_PipeTransferExample() インタラプト IN 転送終了時に API 関数が実行される
USB_HHID_ TCMD_CLASS_REQ	HID 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_hhid_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により異なります。サポートする周辺機器を記載したリストをターゲットペリフェラルリスト(TPL)と呼びます。

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

7.4 構造体

7.4.1 HHID リクエスト構造体

HID クラスリクエストのパラメータ構造体を Table 7-2 に示します。

Table 7-2 USB_HHID_CLASS_REQUEST_PARM_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	クラスリクエストコード。Table 7-3 を参照
void*	tranadr	転送データ格納バッファ
usb_leng_t	tranlen	転送サイズ
uint16_t	duration	インタラプト転送に対する応答間隔時間レート(4ms 単位)
uint8_t	set_protocol	プロトコル値(Boot Protocol(=0)/Report Protocol(=1))
uint8_t*	get_protocol	プロトコル値格納アドレス
usb_cb_t	complete	クラスリクエスト処理完了コールバック関数

7.4.2 HHID クラスリクエストコード

HID クラスリクエストコードを Table 7-3 に示します。

Table 7-3 HHID クラスリクエストコード

クラスリクエスト	定義値	対応
Get_Descriptor(HID)	USB_HID_GET_HID_DESCRIPTOR	Yes
Get_Descriptor(Report)	USB_HID_GET_REPORT_DESCRIPTOR	Yes
Get_Descriptor(Physical)	USB_HID_GET_PHYSICAL_DESCRIPTOR	Yes
Set_Report	USB_HID_SET_REPORT	Yes
Get_Report	USB_HID_GET_REPORT	Yes
Set_Idle	USB_HID_SET_IDLE	No
Get_Idle	USB_HID_GET_IDLE	No
Set_Protocol	USB_HID_SET_PROTOCOL	No
Get_Protocol	USB_HID_GET_PROTOCOL	No

7.4.3 HID レポートフォーマット

(1). 受信レポートフォーマット

Table 7-4 に、HID デバイスから通知される受信レポートフォーマットを示します。

インタラプト IN 転送及び、クラスリクエスト *GetReport.*により受信します。

Table 7-4 受信レポートフォーマット

Offset / Application	Keyboard モード	Mouse モード
データ長	8 バイト	3 バイト
0 (Top Byte)	Modifier keys	b0: Button 1 b1: Button 2 b2-7: Reserved
+1	Reserved	X displacement
+2	Keycode 1	Y displacement
+3	Keycode 2	-
+4	Keycode 3	-
+5	Keycode 4	-
+6	Keycode 5	-
+7	Keycode 6	-

(2). 送信レポートフォーマット

Table 7-5 に、HID デバイスに通知する送信レポートフォーマットを示します。
クラスリクエスト *SetReport.*で送信を行います。

Table 7-5 送信レポートフォーマット

Offset / Application	Keyboard モード	Mouse モード
データ長	1 バイト	非サポート
0 (Top Byte)	b0: LED 0 (NumLock) b1: LED 1(CapsLock) b2: LED 2(ScrollLock) b3: LED 3(Compose) b4: LED 4(Kana)	-
+1 ~ +16	-	-

(3). 注意事項

データ通信で用いるレポートフォーマットはレポートディスクリプタに従う必要があります。本ドライバではレポートディスクリプタの取得と解析は行わず、インタフェースプロトコルコードに従ってレポートフォーマットを決定しています。HID クラス仕様にあわせてユーザモディファイしてください。

7.5 HHID API 一覧

HHID API 一覧を Table 7-6 に示します。

Table 7-6 List of HHID API Functions

関数名	説明	備考
R_usb_hhid_task	HHID タスク処理	
R_usb_hhid_ClassCheck	接続デバイス動作確認用メッセージ通知	
R_usb_hhid_DriverStart	HHID ドライバ開始	
R_usb_hhid_DriverStop	HHID ドライバ停止	
R_usb_hhid_SetPipeRegistration	パイプコントロールレジスタ設定処理	
R_usb_hhid_PipeTransferExample	USB データ転送処理	
R_usb_hhid_TransferEnd	USB データ転送停止処理	
R_usb_hhid_class_request	クラスリクエスト処理	
R_usb_hhid_DeviceInformation	接続デバイスの状態確認	
R_usb_hhid_ChangeDeviceState	接続デバイスの状態変更	
R_usb_hhid_GetReportLength	レポート長(MaxPacketSize)確認	
R_usb_hhid_get_hid_protocol	インタフェースプロトコル確認	

R_usb_hhid_task

HHID タスク処理

形式

void	R_usb_hhid_task(void)

引数

―	―

戻り値

―	―

解説

*usb_hhid_task()*関数を呼び出します。

HHID 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_hhid_main_task();    /* HHID Application Task */
       R_usb_hhid_task();       /* HHID Task */

    }
    else
    {
    }
  }
}
```

R_usb_hhid_ClassCheck

ディスクリプタチェック処理

Format

void	R_usb_hhid_ClassCheck (uint8_t **table)

Argument

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

Return Value

—	—

Description

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 HHID 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。

HHID 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。

Note

Example

```
USB_STATIC void usb_hhid_class_check(uint8_t **table)
{
       R_usb_hhid_ClassCheck(table);
       usb_shhid_smpl_devaddr = (usb_addr_t)(*table[2]);
}
```

R_usb_hhid_DriverStart

HHID ドライバ起動

形式

void R_usb_hhid_DriverStart(void)

引数

― ―

戻り値

― ―

解説

HHID ドライバタスクを起動します。

補足

使用例

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

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

  /* HCD driver open & registratuion */
  R_usb_hstd_HcdOpen();              /* HCD task, MGR task open */
  usb_hhid_registration();           /* HHID driver registration */
  R_usb_hhid_DriverStart();          /* HHID Task Start */

  /* Scheduler initialized */
  R_usb_hstd_ChangeDeviceState(USB_DO_SETHWFUNCTION);

}
```

R_usb_hhid_DriverStop

HHID ドライバ停止

形式

void R_usb_hhid_DriverStop (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_hsmpl_device_state(uint16_t data, uint16_t state)
{
  switch( state )
  {
   case USB_STS_DETACH:
       usb_smpl_set_suspend_flag(USB_NO);
       usb_shhid_active = USB_NO;
       usb_shhid_apl_function = USB_HHID_APL_CLOSE;
       R_usb_hhid_DriverStop();
   break;
            .
            .
            .
}
```

R_usb_hhid_SetPipeRegistration

パイプ設定処理

形式

void R_usb_hhid_SetPipeRegistration(usb_addr_t devadr)

t 引数

devadr 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。HID 通信で使用されるパイプをハードウェアに設定します。

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_smp_task( void )
{
    :
  R_usb_hhid_SetPipeRegistration (devadr);
    :
}
```

R_usb_hhid_PipeTransferExample

USB データ転送要求

形式

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

引数

*table	データ格納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

```
usb_er_t usb_smp_task(void)
{
  uint8_t    data[64];                          /* Data buff */
  usb_lenguint16_t size = 64;                   /* Data size */
                       :
                       :
  R_usb_hhid_TransferExample(data, size,(usb_cb_t)usb_data_received);

}

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

R_usb_hhid_TransferEnd

USB データ転送強制終了

形式

usb_er_t R_usb_hhid_TransferEnd(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_hhid_PipeTransferExample*, *R_usb_hhid_class_request*）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。

r_usb_class_usrcfg.h ファイルの *USB_HHID_GET_REPORT_PIPE0* マクロの設定に従って、コントロール転送かインタラプト転送を停止します。

- *USB_HHID_GET_REPORT_PIPE0* マクロ有効: コントロール転送停止
- *USB_HHID_GET_REPORT_PIPE0* マクロ無効: インタラプト転送停止

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_hhid_TransferEnd();

  return err;
    :
}
```

R_usb_hhid_class_request

HID クラスリクエスト送信

形式

usb_er_t R_usb_hhid_class_request(USB_HHID_CLASS_REQUEST_PARM_t *pram)

引数

*pram	HID 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

HHID ドライバに対し、HID 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. Get_Descriptor(HID)
2. Get_Descriptor(Report)
3. Get_Descriptor(Physical)
4. Set_Report
5. Get_Report
6. Set_Idle
7. Get_Idle
8. Set_Protocol
9. Get_Protocol

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_hhid_apl.c* を参照してください。

引数 *USB_HHID_CLASS_REQUEST_PARM_t* 構造体の種別は 7.4 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

```
void usb_hhid_smpl_set_report(uint16_t devadr, uint8_t *p_data, uint16_t
length, usb_cb_t complete)
{
  USB_HHID_CLASS_REQUEST_PARM_t   class_req;
  /* SET_REPORT */
  class_req.bRequestCode = USB_HID_SET_REPORT;
  class_req.devadr    = devadr;
  class_req.tranadr   = p_data;
  class_req.tranlen   = length;
  class_req.complete  = complete;
    R_usb_hhid_class_request(class_req);
}
```

R_usb_hhid_DeviceInformation

デバイスステータス取得

形式

void	R_usb_hhid_DeviceInformation(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_hhid_DeviceInformation(tbl);
    :
}
```

R_usb_hhid_ChangeDeviceState

デバイス状態変更処理

形式

usb_er_t R_usb_hhid_ChangeDeviceState (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_hhid_ChangeDeviceState
   (USB_DO_GLOBAL_SUSPEND, USB_PORT0, usb_hsmpl_status_result);
}
```

R_usb_hhid_GetReportLength

レポートレングス取得処理

形式

uint16_t R_usb_hhid_GetReportLength(void)

引数

― ―

戻り値

― Max パケットサイズ

解説

接続された USB デバイスの Max パケットサイズを取得します。

補足

使用例

```
void usb_smp_task( void )
{
  uint16_t  usb_smp_report_length;
    :
  usb_smp_report_length = R_usb_hhid_GetReportLength();
    :
}
```

R_usb_hhid_get_interfaceprotocol

プロトコルコード取得

形式

uint8_t R_usb_hhid_get_interfaceprotocol(void)

引数

― ―

戻り値

― Protocol code of USB device（*bInterfaceProtocol*）

解説

接続された USB デバイスのプロトコルコード（bInterfaceProtocol）を取得します。

補足

1. *bInterfaceProtocol* は Interface Descriptor に含まれます。
2. マルチインタフェースデバイスの場合は、最初の HID クラスのプロトコルコードを応答します。

使用例

```
void  usb_smp_task( void )
{
  uint8_t    protocol;
    :
  /* Gets the interface protocol value */
  protocol = R_usb_hhid_get_interfaceprotocol();
    :
}
```

8. 制限事項

HHID には、以下の制限事項があります。

1. HID ドライバは、レポートディスクリプタを解析してレポートフォーマットを決める必要があります。
   (本 HHID ドライバはインターフェースプロトコルからレポート形式を決定します。)
2. 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)
3. HHID 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。

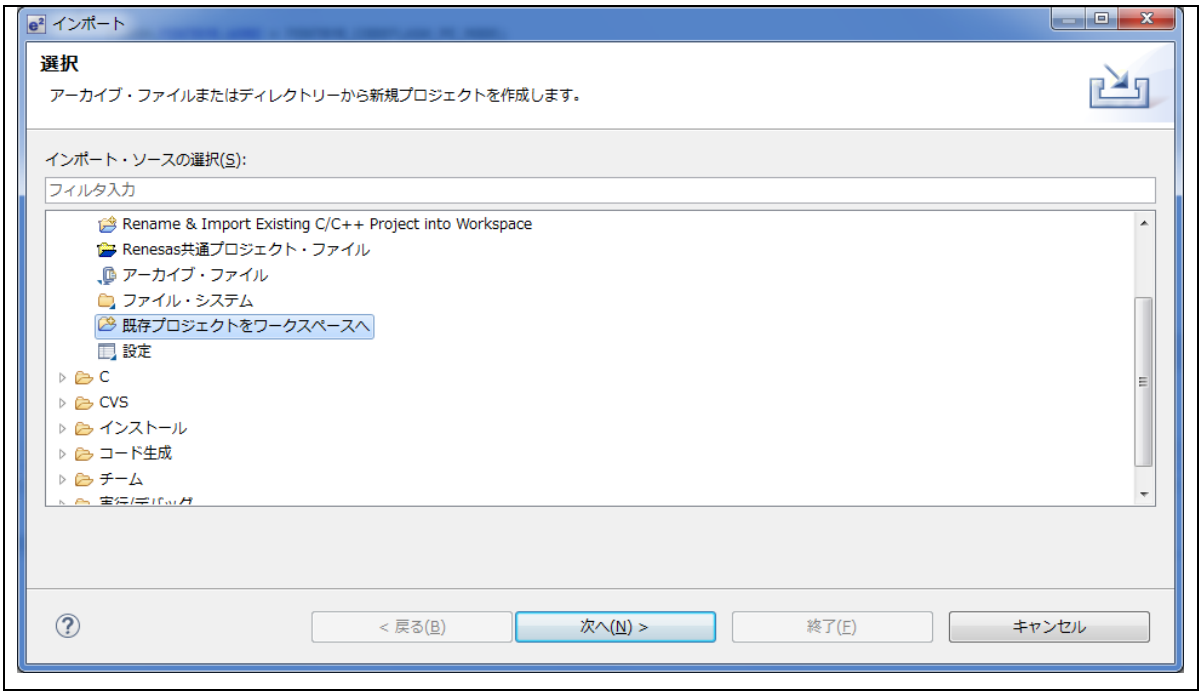


Figure 9-1 インポートの選択

(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。

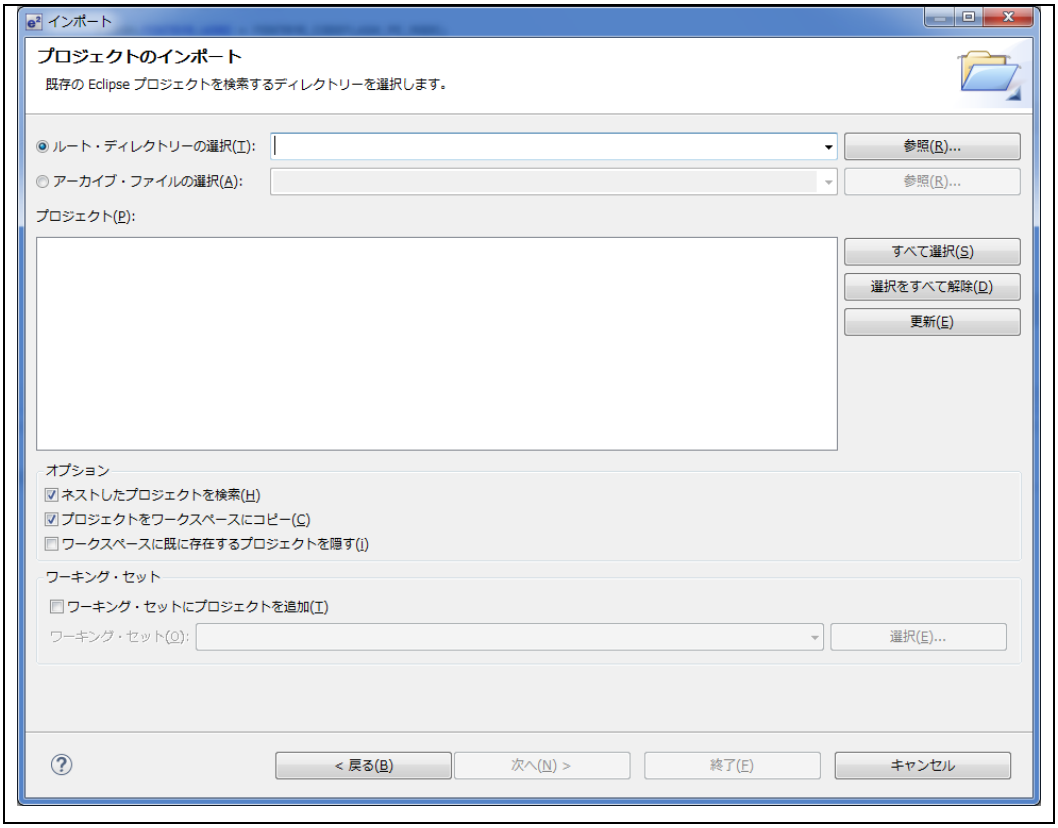


Figure 9-2　プロジェクトのインポート画面

(5). [終了]をクリック して下さい。

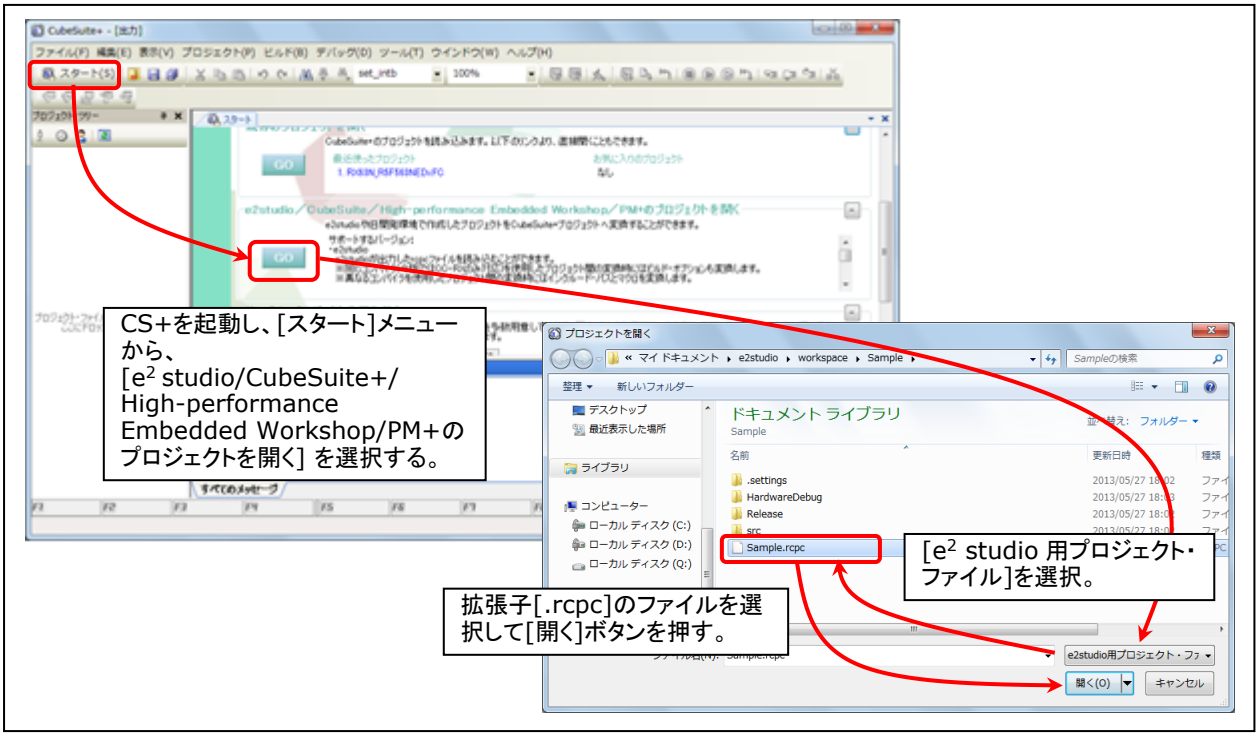
プロジェクトのワークスペースへのインポートが完了します。

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



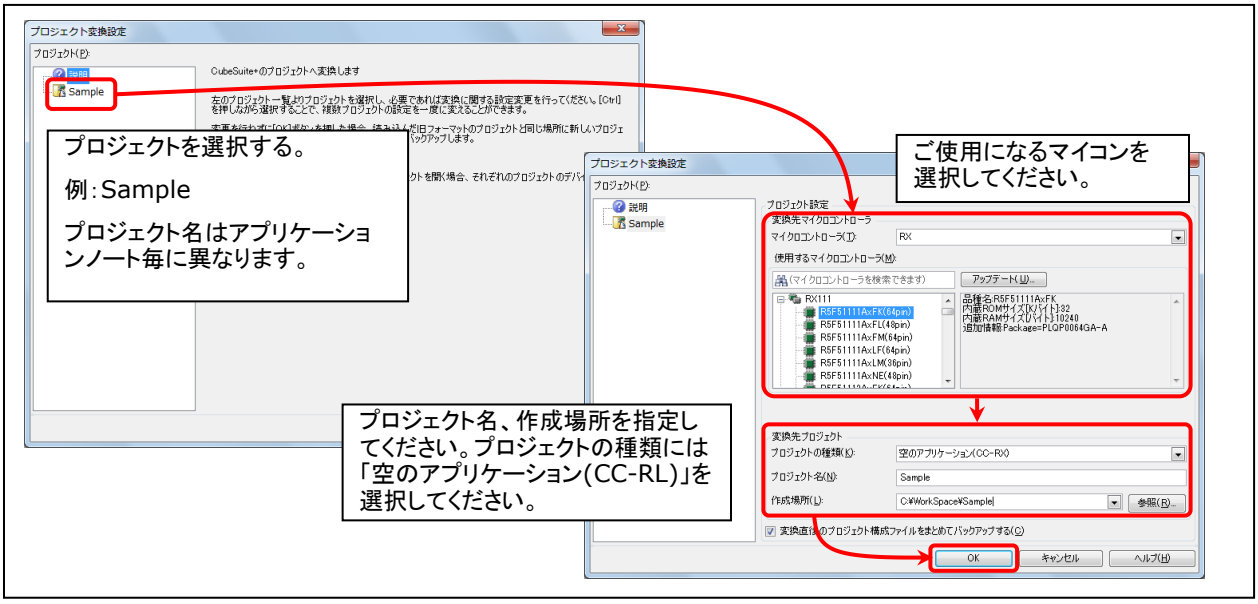


Figure 10-1　e² studio 用プロジェクトの CS+読み込み方法

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	2011.03.12	—	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	2013.03.31	—	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/