SH7786 グループ
ルネサス EHCI 対応 USB2.0 ホストコントローラ IP
USB Basic Firmware uITRON 版

R01AN1452JJ0103
Rev.1.03
2012.12.07

要旨

本資料はルネサス EHCI 対応 USB2.0 ホストコントローラ IPを使用した USB インタフェース制御用サンプルプログラムである「ルネサス EHCI 対応 USB2.0 ホストコントローラ IP USB Basic Firmware uITRON 版」の取扱説明書です。

動作確認デバイス

SH7786

目次

1. 資料概要

1.1 概要

本資料は、ルネサス EHCI 対応 USB2.0 ホストコントローラ IPを使用した USB インタフェース制御用サンプルプログラムである「ルネサス EHCI 対応 USB2.0 ホストコントローラ IP USB Basic Firmware uITRON 版」の取扱説明書です。

ルネサス EHCI 対応 USB2.0 ホストコントローラ IP用 USB Basic Firmware は uITRON に対応しています。

本書は必ずデータシートと併用してご利用ください。

1.2 関連ドキュメント

1. Universal Serial Bus Revision 2.0 specification
   【http://www.usb.org/developers/docs/】
2. SH7786 グループユーザーズマニュアルハードウェア編
   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/prod/usb/】

1.3 用語一覧

本資料で使用される用語と略語は以下のとおりです。

USB	: Universal Serial Bus
EHCI	: Enhanced Host Controller Interface
OHCI	: Open Host Controller Interface
USB-BASIC-F/W	: USB basic firmware for ルネサス EHCI 対応 USB2.0 ホストコントローラ IP (uITRON)
uITRON	: USB basic firmware for uITRON system
HEW	: High-performance Embedded Workshop
HCD	: Host control driver of USB-BASIC-F/W
MGR	: Peripheral device state maneger of HCD
HDCD	: Host device class driver (device driver and USB class driver)
HUBCD	: Hub class sample driver
APL	: Application program

2. 概要

2.1 USB-BASIC-F/Wの特長

USB-BASIC-F/Wは以下のような特徴を所持しています。

ホスト機能が動作可能
コントロール転送（エニュメレーション処理）のサンプルプログラムを提供
デバイス接続／切断処理のサンプルプログラムを提供
サスペンド／レジューム処理のサンプルプログラムを提供
HUBCD のサンプルプログラムを提供
USB-BASIC-F/Wをカスタマイズしないまま複数のデバイスクラスドライバを実装することが可能
（接続可能なデバイスの数だけデバイスクラスドライバを登録します。）

以下の機能は、お客様のシステムに合わせて作成してください。

USB ケーブル接続時の過電流検出処理
ディスクリプタ解析処理
デバイスクラスドライバ

2.2 開発目的

USB-BASIC-F/Wは以下の目的で開発しております。

お客様におけるルネサス EHCI 対応 USB2.0 ホストコントローラ IPを使用した USB 通信プログラムの開発を容易にする。

2.3 機能概要

USB-BASIC-F/Wが提供する機能は以下のとおりです。

Low-Speed / Full-Speed / High-Speed デバイスとのエニュメレーション
USB コネクタの接続／切断、サスペンド／レジューム、USB バスリセット処理
パイプ 0 でコントロール転送
パイプ 1～30 でのデータ転送（Bulk 転送、Interrupt 転送、Isochronous 転送）
転送エラー判定

2.4 タスク構成

USB Basic Firm は、ホスト機能を行うホストドライバ、デバイス状態管理を行うホストマネージャ、USB デバイスとの通信を管理する HCD Transfer、USB ポートの管理を行う HCD System、USB ハブのダウンポートに接続したデバイス制御を行うハブクラスドライバおよび、アプリケーションから構成されています。

ホストドライバは、各タスクからのメッセージで H/W 制御を開始します。また、H/W 制御終了および処理結果と H/W 要求を各タスクに通知します。

ホストマネージャは、ルートポートに接続されたデバイスの状態管理とエニュメレーションを行うサンプルプログラムです。また、アプリケーションがデバイスステートを更新する場合は、デバイスアドレスによりホストマネージャがホストドライバもしくはハブクラスドライバにメッセージを発行します。ハブクラスドライバは、USB ハブのダウンポートに接続されたデバイスのエニュメレーションと状態管理を行うサンプルプログラムです。

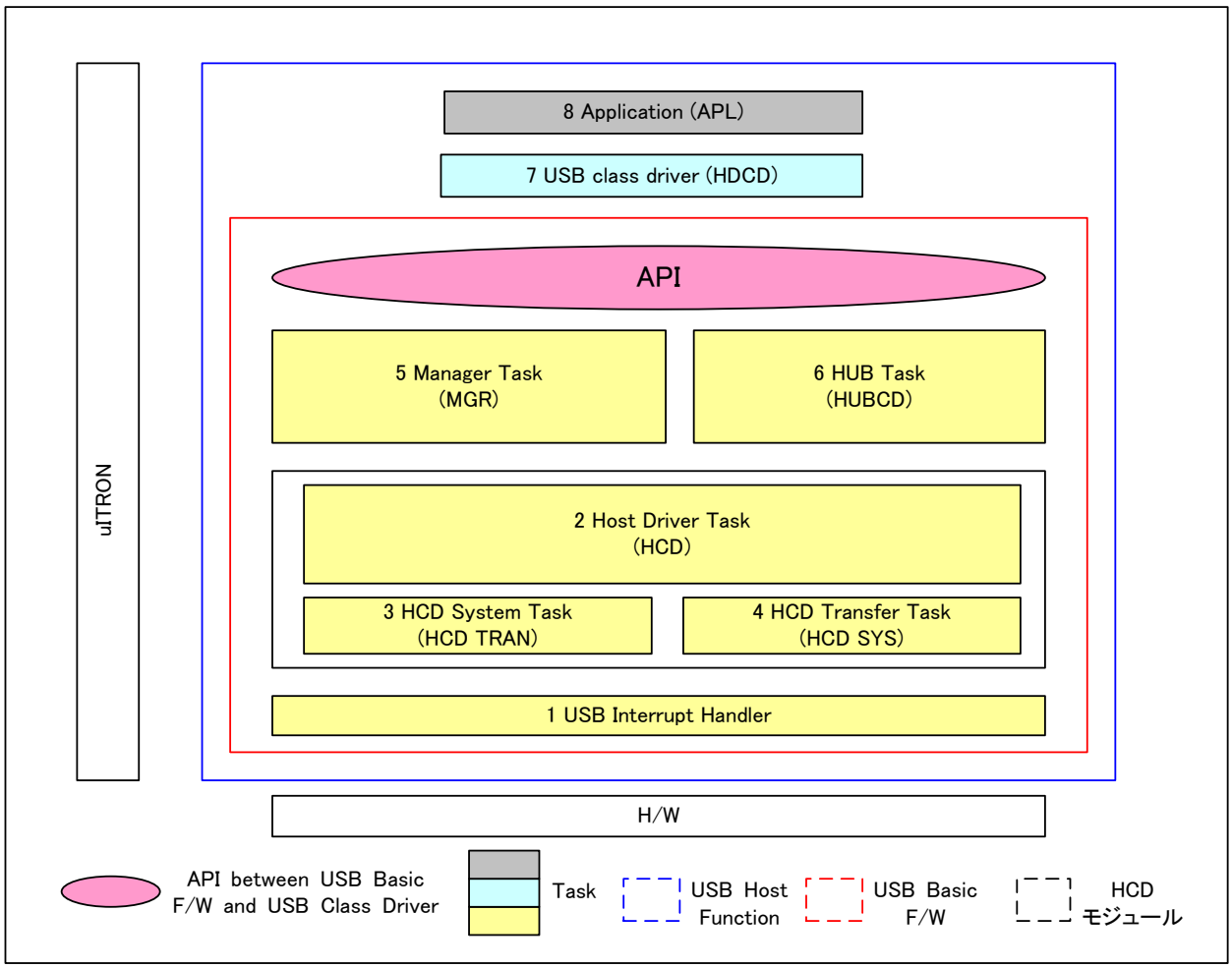


図2.1　USB-BASIC-F/Wのタスク構成

表2.1　タスク機能概要

No	モジュール名	機能
1	USB 割込みハンドラ	USB 割り込みハンドラ （USB パケット送受信終了／特殊信号検出）
2	ホストコントロールドライバ（HCD）	ホスト通信管理
3	HCD Transfer(HCD_TRN)	ホスト機能の H/W 制御 ホストトランザクション管理
4	HCD System (HCD_SYS)	ホスト機能の H/W 制御 USB ポート状態管理 Bus Reset 制御
5	マネージャ（MGR）	デバイス状態管理 エニュメレーション HCD/HUBCD 制御メッセージ判定
6	ハブクラスドライバ（HUBCD）	HUB ダウンポートデバイス状態管理 HUB ダウンポートエニュメレーション
7	ホストデバイスクラスドライバ(HDCD)	デバイスクラス依存の処理実行 (システムにあわせてご用意ください)
8	アプリケーション（APL）	アプリケーションの実行 (システムにあわせてご用意ください)

2.5 概略フロー

USB-BASIC-F/Wは、USBデータ送受信を行う制御関数をタスクで構成しています。

割り込みが発生した場合、HCD Transferへメッセージで通知します。HCD Transferタスクは USB割り込みハンドラからメッセージを受け取ると、割り込み要因を判別して該当する処理を実行します。（各タスクの概略フローは次章以降に記述します）

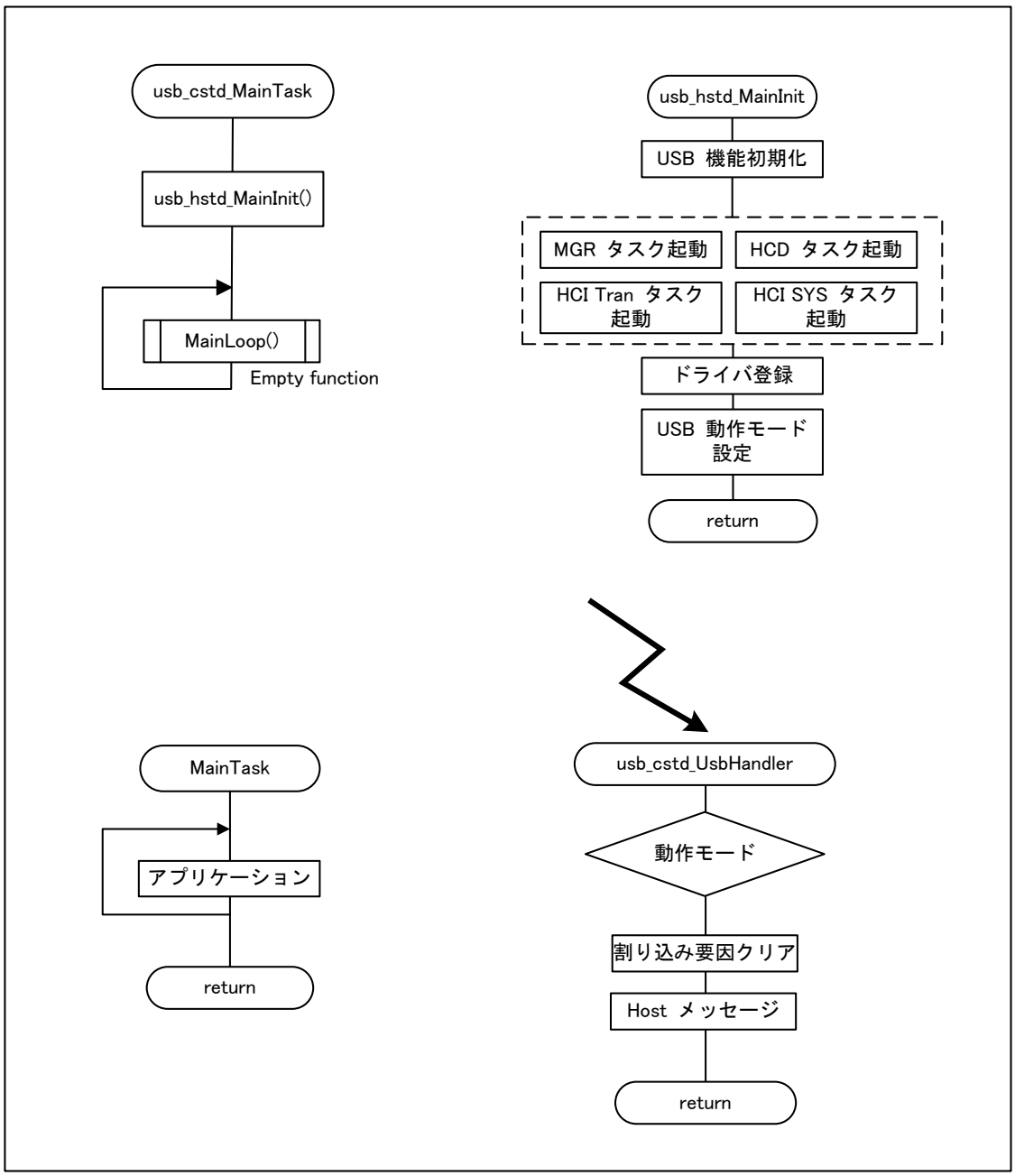


図2.2 概略フロー

2.6 uITRON タスク関連図

USB-BASIC-F/Wのタスク関連図を以下に示します。

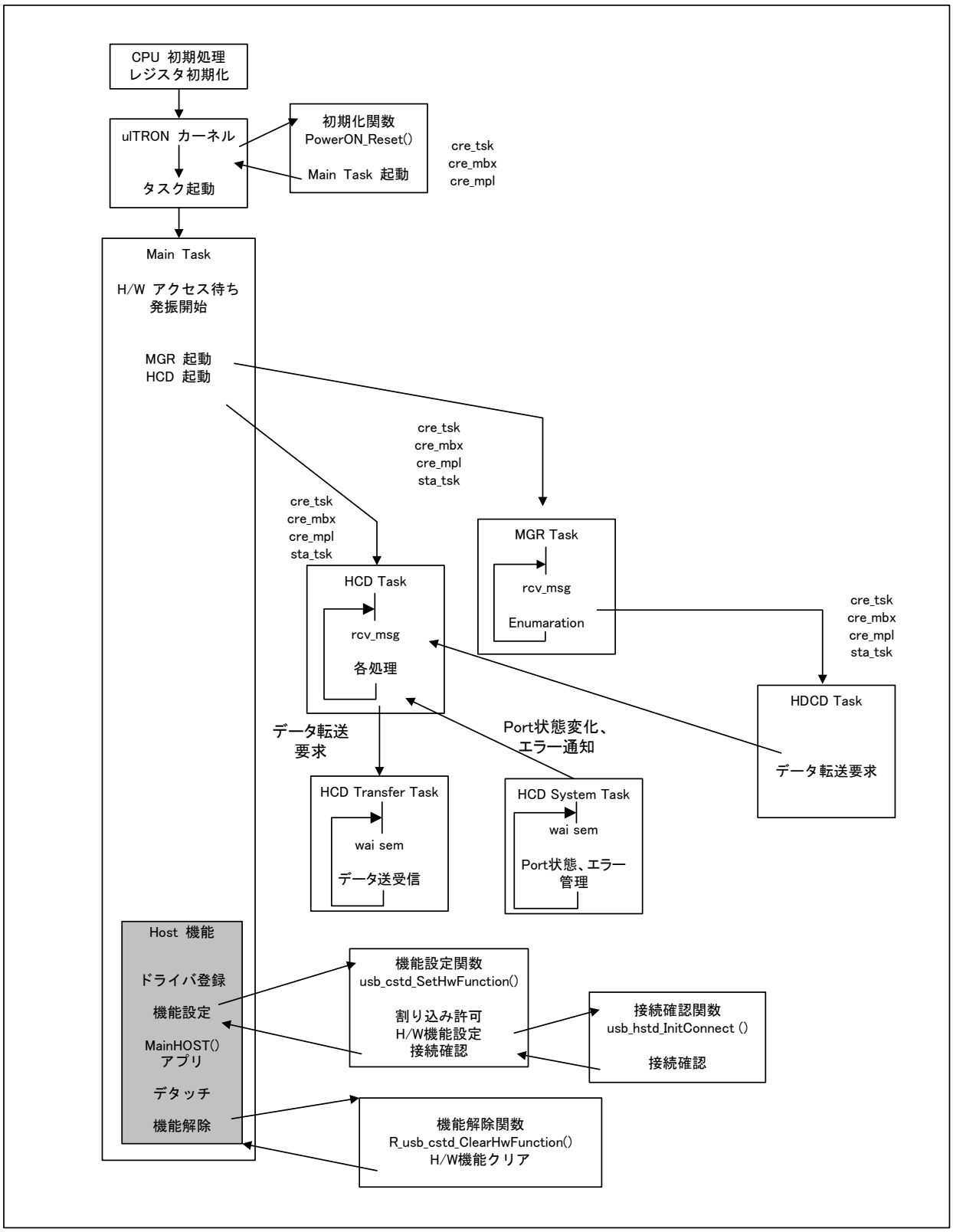


図2.3 タスク関連図

3. USB-BASIC-F/Wを動作させるには

3.1 概要

USB-BASIC-F/Wは、スケジューラマクロ（"r_usb_cItron.h" / "r_usb_cMacItron.h"）、ユーザ情報（"r_usb_cDefUsrPb.h/r_usb_cDefUsr.h"）および OS 設定（"r_usb_cKernelId.h" / "main.c" / "r_usb_HSMPL_apl.c"）を変更し HDCD を usb_hstd_MainLoop 関数に追加することにより、ユーザシステムに適合した USB ドライバとなります。

3.2 USB-BASIC-F/Wの変更

USB-BASIC-F/Wを動作させるには、以下のプログラムおよびヘッダファイルの変更が必要です。

1. main.c およびルネサス EHCI 対応 USB2.0 ホストコントローラ IP用のサンプル関数を提供しますがユーザシステムにあわせて変更してください。
   制御用 MCU の初期化、割り込みハンドラ、割り込み制御など（表 3.1 参照）
   指定時間待ち関数の時間調整（usb_cstd_DelayXms()関数、usb_cstd_Delay1us()関数）
   ループ処理等で指定時間のウェイトを行っています。ご使用のシステムに合わせてループ回数を変更するなど指定時間になるように調整してください。
   USB-BASIC-F/Wでは、USB 割り込み禁止関数（usb_cstd_IntDisable()関数）が USB 割り込みを禁止し、USB 割り込み許可関数（usb_cstd_IntEnable()関数）が USB 割り込みを許可しています。ご使用の MCU に合わせて設定を行ってください。
2. ユーザカスタマイズが必要なファイルがあります。
   「5.ユーザ定義情報」を参照して、各種ユーザ設定を変更してください。
3. デバッグ情報の出力機能
   r_usb_cMacPrint.h ファイルでデバッグ情報出力の許可禁止を設定してください。
   （シリアルドライバ等を作成することによりデバッグ情報を出力できます）

表3.1 関数一覧

型	関数名	機能概要
void	usb_cstd_TargetInit(void)	システム初期化
void	usb_cstd_UsbIntHand (void)	USB 割り込みハンドラ
void	usb_cstd_UsbintInit (void)	USB 割り込み許可
void	usb_cstd_IntEnable(void)	USB 割込み許可
void	usb_cstd_IntDisable(void)	USB 割込み禁止
void	usb_cstd_Delay1us (uint16_t time)	1us 待ち処理
void	usb_cstd_DelayXms (uint16_t time)	1ms 待ち処理
void	usb_cstd_VbusControl(uint16_t port, uint16_t command)	VBUS ON/OFF 制御

3.3 HDCD の作成

USB-BASIC-F/Wを動作させるには、お客様のシステムに合わせた HDCD を作成する必要があります。

HDCDを作成の際、必ず以下の処理を行うようにして下さい。

・HDCDの登録(R_usb_hstd_DriverRegistration()にてMGRにHDCDを登録)
・クラスチェック処理の作成(エニュメレーション中にクラスのチェックを行う必要があります)
・PIPE設定(HDCDにて使用するPIPEの設定をR_usb_hstd_SetPipeRegistration()にて行ってください)

上記以外の処理はシステムにあわせご作成下さい。

3.4 ご注意

クラス規定やベンダ固有リクエストの発行が必要な場合をはじめ、通信速度、プログラム容量等を考慮する場合、さらにユーザインタフェースを個別に設定する場合には、お客様にてカスタマイズしてください。

【注】 USB-BASIC-F/Wは、USB 通信動作を保証するものではありません。システムに適用される場合は、お客様における動作検証はもとより、多種多様なデバイスに対する接続確認を実施してください。

4. ユーザ定義マクロ

4.1 概要

USB-BASIC-F/Wは、uITRON システムコール、デバッグ出力をマクロ記述しているため、マクロヘッダファイルを書き換えることにより、お客様固有の実行ファイルを作成することができます。お客様のシステムに合わせて、マクロを変更してください。マクロは以下の 2 種で構成され、uITRON システムコールマクロは（r_usb_cMacItron.h）に、デバッグ出力マクロは（r_usb_cMacPrint.h）に定義しています。

1. uITRON システムコールマクロ
2. デバッグ出力マクロ

4.2 uITRON システムコールマクロ

uITRON のシステムコールマクロです。

ご使用になる ITRON にあわせてお客様で書き換えてください。

また、r_usb_cMacItron.h ファイルで ITRON マクロをUSB-BASIC-F/W用に再定義しており、ITRON を未使用の場合でも条件コンパイルによって ITRON マクロを定義していますので、こちらのファイルもご使用になる ITRON にあわせてお客様で書き換えてください。

```
#define   USB_CRE_TSK(ID,INFO)          cre_tsk( (USB_ID_t)ID, (USB_TSK_t*)INFO )
#define   USB_DEL_TSK(ID)               del_tsk( (USB_ID_t)ID )
#define   USB_STA_TSK(ID,CODE)          sta_tsk( (USB_ID_t)ID, (USB_VI_t)CODE )
#define   USB_ACT_TSK(ID)               act_tsk( (USB_ID_t)ID )
#define   USB_TER_TSK(ID)               ter_tsk( (USB_ID_t)ID )
#define   USB_EXT_TSK()                 ext_tsk( )
#define   USB_REF_TST(ID, STS)          ref_tst( (USB_ID_t)ID, (USB_RTST_t*)STS )

#define   USB_DLY_TSK(TIME)             dly_tsk( (USB_RT_t)TIME )

#define   USB_CRE_MBX(ID, INFO)         cre_mbx( (USB_ID_t)ID, (USB_MBX_t*)INFO )
#define   USB_DEL_MBX(ID)               del_mbx( (USB_ID_t)ID )
#define   USB_SND_MSG(ID, MESS)         snd_mbx( (USB_ID_t)ID, (USB_MSG_t*)MESS )
#define   USB_ISND_MSG(ID, MESS)        isnd_mbx( (USB_ID_t)ID, (USB_MSG_t*)MESS )
#define   USB_RCV_MSG(ID, MESS)         rcv_mbx( (USB_ID_t)ID, (USB_MSG_t**)MESS )
#define   USB_PRCV_MSG(ID, MESS)        prcv_mbx( (USB_ID_t)ID, (USB_MSG_t**)MESS )
#define   USB_TRCV_MSG(ID, MESS, TM)    trcv_mbx( (USB_ID_t)ID, (USB_MSG_t**)MESS, (USB_TM_t)TM )

#define   USB_CRE_MPL(ID, INFO)         cre_mpf( (USB_ID_t)ID, (USB_MPL_t*)INFO )
#define   USB_DEL_MPL(ID)               del_mpf( (USB_ID_t)ID )
#define   USB_PGET_BLK(ID, BLK)         pget_mpf( (USB_ID_t)ID, (USB_MH_t*)BLK )
#define   USB_IPGET_BLK(ID, BLK)        ipget_mpf( (USB_ID_t)ID, (USB_MH_t*)BLK )
#define   USB_REL_BLK(ID, BLK)          rel_mpf( (USB_ID_t)ID, (USB_MH_t)BLK )
```

```
#define    USB_CRE_SEM(ID, INFO)          cre_sem( (USB_ID_t)ID, (USB_SEM_t*)INFO )
#define    USB_WAI_SEM(ID)                wai_sem( (USB_ID_t)ID )
#define    USB_POL_SEM(ID)                pol_sem( (USB_ID_t)ID )
#define    USB_SIG_SEM(ID)                sig_sem( (USB_ID_t)ID )
#define    USB_DEL_SEM(ID)                del_sem( (USB_ID_t)ID )
#define    USB_ISIG_SEM(ID)               isig_sem( (USB_ID_t)ID )

#define    USB_CRE_ALM(ID, INFO)          cre_alm( (USB_ID_t)ID, (USB_ALM_t*)INFO )
#define    USB_STA_ALM(ID, TIME)          sta_alm( (USB_ID_t)ID, (USB_RT_t)TIME )
#define    USB_STP_ALM(ID)                stp_alm( (USB_ID_t)ID )
#define    USB_DEL_ALM(ID)                del_alm( (USB_ID_t)ID )
```

4.3 デバッグ情報出力マクロ

UART および表示デバイスなどにデバッグ情報を出力するマクロです。シリアルドライバ、表示デバイス用ドライバが必要になります。

```
#define    USB_SPRINTF0(FORM)                      fprintf(stderr,FORM)
#define    USB_SPRINTF1(FORM,x1)                   fprintf(stderr,FORM,x1)
#define    USB_SPRINTF2(FORM,x1,x2)                fprintf(stderr,FORM,x1,x2)
#define    USB_SPRINTF3(FORM,x1,x2,x3)             fprintf(stderr,FORM,x1,x2,x3)
#define    USB_SPRINTF4(FORM,x1,x2,x3,x4)          fprintf(stderr,FORM,x1,x2,x3,x4)
#define    USB_SPRINTF5(FORM,x1,x2,x3,x4,x5)       fprintf(stderr,FORM,x1,x2,x3,x4,x5)
#define    USB_SPRINTF6(FORM,x1,x2,x3,x4,x5,x6)    fprintf(stderr,FORM,x1,x2,x3,x4,x5,x6)
#define    USB_SPRINTF7(FORM,x1,x2,x3,x4,x5,x6,x7) fprintf(stderr,FORM,x1,x2,x3,x4,x5,x6,x7)
#define    USB_SPRINTF8(FORM,x1,x2,x3,x4,x5,x6,x7,x8)  fprintf(stderr,FORM,x1,x2,x3,x4,x5,x6,x7,x8)
#define    USB_PRINTF0(FORM)                       printf(FORM)
#define    USB_PRINTF1(FORM,x1)                    printf(FORM,x1)
#define    USB_PRINTF2(FORM,x1,x2)                 printf(FORM,x1,x2)
#define    USB_PRINTF3(FORM,x1,x2,x3)              printf(FORM,x1,x2,x3)
#define    USB_PRINTF4(FORM,x1,x2,x3,x4)           printf(FORM,x1,x2,x3,x4)
#define    USB_PRINTF5(FORM,x1,x2,x3,x4,x5)        printf(FORM,x1,x2,x3,x4,x5)
#define    USB_PRINTF6(FORM,x1,x2,x3,x4,x5,x6)     printf(FORM,x1,x2,x3,x4,x5,x6)
#define    USB_PRINTF7(FORM,x1,x2,x3,x4,x5,x6,x7)  printf(FORM,x1,x2,x3,x4,x5,x6,x7)
#define    USB_PRINTF8(FORM,x1,x2,x3,x4,x5,x6,x7,x8)  printf(FORM,x1,x2,x3,x4,x5,x6,x7,x8)
```

デバッグ情報を出力しない場合は下記のようにコメントアウトしてください。

```
//#define    USB_DEBUGSIO_PP          /* enable sirial out (printf) */
//#define    USB_DEBUGLCD_PP          /* enable display out (fprintf) */
```

5. ユーザ定義情報

5.1 概要

USB-BASIC-F/Wは、ユーザシステム定義情報ファイル（r_usb_cDefUsrPb.h）、ユーザ定義情報ファイル（r_usb_cDefUsr.h）、EHCI ユーザ定義情報ファイル（r_usb_hEhciDefUsr.h）、OHCI ユーザ定義情報ファイル（r_usb_hOhciDefUsr.h）を書き換えることにより、お客様固有の実行ファイルを作成することが可能です。システムに合わせて、下記項目を変更してください。

*下記項目以外は固定値となりますので変更しないで下さい

5.2 ユーザシステム定義情報ファイル（r_usb_cDefUsrPb.h）

1. USB ポート指定
2. OS、生成システムコール対応の指定
3. CPU のバイトエンディアン指定

5.2.1 USB ポート指定

USB ポート数を下記の 2 パターンより選択し指定してください。

1) USB_1PORT_PP　　：USB ポート 1 ポートを使用する。

2) USB_2PORT_PP　　：USB ポート 2 ポートを使用する。

例）USB ポート 1 ポートの場合

```
#define USB_PORTSEL_PP          USB_1PORT_PP
```

5.2.2 OS、生成システムコール対応の指定

OS の生成システムコール対応可否により選択し、指定してください。

1) USB_OS_CRE_USE_PP　　：生成システムコール対応の場合

2) USB_OS_CRE_NOTUSE_PP　　：生成システムコール非対応の場合

例）生成システムコール非対応の場合

```
#define    USB_OS_CRE_MODE_PP        USB_OS_CRE_NOTUSE_PP
```

5.2.3 CPU のバイトエンディアン指定

CPU のバイトエンディアンを下記の 2 パターンより選択し指定してください。

データ送受信時のエンディアンを設定します。（r_usb_cTypedef.h で定義しています）

1) USB_BYTE_LITTLE_PP　　：リトルエンディアン（上位バイトから順に定義）

2) USB_BYTE_BIG_PP　　：ビッグエンディアン（下位バイトから順に定義）

例）CPU のバイトエンディアンがリトルエンディアンの場合

```
#define   USB_CPUBYTE_PP          USB_BYTE_LITTLE_PP
```

5.3 ユーザ定義情報ファイル(r_usb_cDefUsr.h）

1. コントロールリードデータバッファサイズ
2. デバイスアドレスの初期値
3. HUB ダウンポート数

5.3.1 コントロールリードデータバッファサイズ

コントロールリード転送で受信するデータバッファサイズを指定してください。

例）デバイスディスクリプタ 20Byte、
コンフィグレーションディスクリプタ 256Byte の場合

```
#define  USB_DEVICESIZE  20u
#define  USB_CONFIGSIZE  256u
```

5.3.2 デバイスアドレス

PORT0 に接続されるデバイスのデバイスアドレスを指定してください。
例）デバイスアドレスを 1 から開始する場合

```
#define  USB_DEVICEADDR   1u
```

5.3.3 HUB ダウンポート数（ホスト機能選択時）

接続可能な HUB のダウンポート数を指定してください。
例）接続可能な HUB のダウンポートが 4 つまでの場合

```
#define  USB_HUBDOWNPORT 4u
```

5.4 EHCI ユーザ定義情報ファイル(r_usb_hEhciDefUsr.h)

1. EHCI H/W のアドレス指定
2. EHCI ピリオディックフレームリストのサイズ指定
3. EHCI キューヘッドデータ構造メモリの最大数指定
4. EHCI qTD データ構造メモリの最大数指定
5. EHCI iTD データ構造メモリの最大数指定
6. EHCI siTD データ構造メモリの最大数指定
7. EHCI iTD データ構造の最大データ転送サイズ指定
8. EHCI タイムアウト時間指定

5.4.1 EHCI H/W のアドレス指定

EHCI H/W にアクセスするための基準アドレスを指定してください。
各レジスタは、指定された基準アドレスからのオフセットでアドレス指定を行っています。
例）0x000A0000 番地に設定する場合

```
#define USB_EHCI_BASE              0x000A0000
```

5.4.2 EHCI ピリオディックフレームリストのサイズ指定

EHCI ピリオディックフレームリストのサイズを 256、512、1024 の値から指定してください。
このサイズ指定によりピリオディック転送のスケジューリング幅が変更されます。
例）256 サイズを指定する場合

```
#define USB_EHCI_PFL_SIZE          256
```

5.4.3 EHCI キューヘッドデータ構造メモリの最大数指定

EHCI キューヘッドデータ構造メモリの最大数を指定してください。
キューヘッドデータ構造は、コントロール転送、バルク転送、インターラプト転送のエンドポイントパイプ数分必要となります。お客様のシステムに合わせて設定してください。
例）16 を指定する場合

```
#define USB_EHCI_NUM_QH             16
```

5.4.4 EHCI qTD データ構造メモリの最大数指定

EHCI qTD(デバイスステータスレジスタキュー要素転送記述子)データ構造メモリの最大数を指定してください。

qTD は、コントロール転送、バルク転送、インターラプト転送において転送管理用の記述子としてキューヘッドとリンクして使用されます。一つの qTD で転送可能な最大データサイズは 20480byte です。また、コントロール転送の場合は、SETUP ステージ、DATA ステージ、STATUS ステージ毎に一つの qTD が必要となります。お客様のシステムに合わせて設定してください。

例）256 を指定する場合

```
#define USB_EHCI_NUM_QTD            256
```

5.4.5 EHCI iTD データ構造メモリの最大数指定

EHCI iTD（ハイスピード用アイソクロナス転送記述子）データ構造メモリの最大数を指定してください。

iTD データ構造は、ハイスピードアイソクロナス転送のエンドポイントパイプ数分必要となります。お客様のシステムに合わせて設定してください。

例）4 を指定する場合

```
#define USB_EHCI_NUM_ITD            4
```

5.4.6 EHCI siTD データ構造メモリの最大数指定

EHCI siTD（スプリットトランザクションアイソクロナス転送記述子）データ構造メモリの最大数を指定してください。

siTD データ構造は、スプリットトランザクションアイソクロナス転送のエンドポイントパイプ数分必要となります。お客様のシステムに合わせて設定してください。

例）4 を指定する場合

```
#define USB_EHCI_NUM_SITD           4
```

5.4.7 EHCI iTD データ構造の最大データ転送サイズ指定

EHCI iTD データ構造の最大データ転送サイズを指定してください。

この転送サイズは、ハイスピードアイソクロナス転送の 1 回分（1 トランザクション）の最大転送サイズを指定するものです。設定可能な最大データサイズは 1024 です。お客様のシステムに合わせて設定してください。

例）512 を指定する場合

```
#define USB_EHCI_ITD_DATA_SIZE      512
```

5.4.8 EHCI タイムアウト時間指定

EHCI タイムアウト時間を msec 単位で指定してください。

例）3 秒を指定する場合

```
#define USB_EHCI_TIMEOUT            3000
```

5.5 OHCI ユーザ定義情報ファイル(r_usb_hOhciDefUsr.h)

1. OHCI H/W のアドレス指定
2. OHCI エンドポイントデータ構造メモリの最大数指定
3. OHCI エンドポイントディスクリプタデータ構造メモリの最大数指定
4. OHCI トランスファディスクリプタデータ構造メモリの最大数指定
5. OHCI アイソクロナスデバイスの最大数指定
6. OHCI アイソクロナスの最大データ転送サイズ指定
7. OHCI アイソクロナスの最大フレーム数指定
8. OHCI タイムアウト時間指定

5.5.1 OHCI H/W のアドレス指定

OHCI H/W にアクセスするための基準アドレスを指定してください。

各レジスタは、指定された基準アドレスからのオフセットでアドレス指定を行っています。

例）0x000A0000 番地に設定する場合

```
#define USB_OHCI_BASE                0x000A0000
```

5.5.2 OHCI エンドポイントデータ構造メモリの最大数指定

OHCI エンドポイントデータ構造メモリの最大数を指定してください。

OHCI エンドポイントデータ構造は、コントロール転送、バルク転送、インターラプト転送、アイソクロナス転送のエンドポイントパイプ数分必要となります。お客様のシステムに合わせて設定してください。

例）16 を指定する場合

```
#define USB_OHCI_NUM_ENDPOINT        16
```

5.5.3 OHCI エンドポイントディスクリプタデータ構造メモリの最大数指定

OHCI エンドポイントディスクリプタデータ構造メモリの最大数を指定してください。

OHCI エンドポイントディスクリプタデータ構造は、コントロール転送、バルク転送、インターラプト転送、アイソクロナス転送のエンドポイントパイプ数＋インタラプト転送のスケジューリング用に 31 個必要となります。お客様のシステムに合わせて設定してください。

例）64 を指定する場合

```
#define USB_OHCI_NUM_ED              64
```

5.5.4 OHCI トランスファディスクリプタデータ構造メモリの最大数指定

OHCI トランスファディスクリプタデータ構造メモリの最大数を指定してください。

OHCI トランスファディスクリプタデータ構造は、コントロール転送、バルク転送、インターラプト転送、アイソクロナス転送において転送管理用の記述子として使用されます。一つのトランスファディスクリプタで転送可能な最大データサイズは 8192byte です。また、コントロール転送の場合は、SETUP ステージ、DATA ステージ、STATUS ステージ毎に一つの qTD が必要となります。お客様のシステムに合わせて設定してください。

例）256 を指定する場合

```
#define USB_OHCI_NUM_TD              256
```

5.5.5 OHCI アイソクロナスデバイスの最大数指定

OHCI アイソクロナスデバイスの最大数を指定してください。
お客様のシステムに合わせて設定してください。
例）4 を指定する場合

```
#define USB_OHCI_ISO_MAXDEVICE              4
```

5.5.6 OHCI アイソクロナスの最大データ転送サイズ指定

OHCI アイソクロナスデバイスの最大データ転送サイズを指定してください。
この転送サイズは、OHCI アイソクロナス転送の 1 回分（1 トランザクション）の最大転送サイズを指定するものです。設定可能な最大データサイズは 1023 です。お客様のシステムに合わせて設定してください。
例）256 を指定する場合

```
#define USB_OHCI_ISO_MAX_PACKET_SIZE        256
```

5.5.7 OHCI アイソクロナスの最大フレーム数指定

OHCI アイソクロナス転送の最大フレーム数を指定してください。
このフレーム数は、OHCI アイソクロナス転送時の最大転送フレーム数を指定するものです。設定値は 2 のべき乗で、設定可能な最大フレーム数は 8 です。お客様のシステムに合わせて設定してください。
例）8 を指定する場合

```
#define USB_OHCI_ISO_MAX_FRAME              8
```

5.5.8 OHCI タイムアウト時間指定

OHCI タイムアウト時間を msec 単位で指定してください。
例）3 秒を指定する場合

```
#define USB_OHCI_TIMEOUT            3000
```

6. サンプルプログラム

6.1 概要

USB-BASIC-F/Wで動作する HDCD の作成方法について説明します。

USB-BASIC-F/Wのサンプルプログラムは、以下のプロジェクトで構成されます。

・Itron_StdFw　：uITRON 版

6.2 ファイル一覧

6.2.1 フォルダ構成

以下にUSB-BASIC-F/Wが提供するファイルのフォルダ構成を示します。

<HEWワークスペース：USBSTDFW>

```
＋――― USBSTDFW
｜　　＋――― class
｜　　｜　　＋――― hubd　　サンプルHUBドライバ
｜　　｜　　＋――― SMPL　　USB標準リクエストサンプル
｜　　＋――― include　　共通ヘッダファイル
｜　　＋――― USB20
｜　　　　＋――― HCD　　ホストコントロールドライバ
｜　　　　｜　　＋――― HCI
｜　　　　｜　　　　＋――― EHCI　　EHCIドライバ
｜　　　　｜　　　　＋――― Include　　HCIヘッダファイル
｜　　　　｜　　　　＋――― OHCI　　OHCIドライバ
｜　　　　＋――― LIB　　共通ライブラリ
＋――― SmplMain
　　＋――― APL　　サンプルアプリケーション
```

6.2.2 ファイル一覧

以下にUSB-BASIC-F/Wが提供するファイル一覧を示します。

表6.1 ファイル一覧

Folder	ファイル名	概要
HCD	r_usb_hControlRW.c	コントロールリード／ライト処理
HCD	r_usb_hDriver.c	HCD タスク
HCD	r_usb_hDriverAPI.c	HCD/MGR API 関数
HCD	r_usb_hManager.c	MGR タスク
HCD	r_usb_hSignal.c	USB 信号制御、発振制御処理
HCD	r_usb_hStdFunction.c	USB 機能拡張ライブラリ関数
HCD	r_usb_hLibUSBIP.c	USB ライブラリ関数
HCI	r_usb_hHci.c	EHCI/OHCI 機能提供関数
HCI¥EHCI	r_usb_hEhciMain.c	EHCI 主要関数
HCI¥EHCI	r_usb_hEhciMemory.c	EHCI メモリリソース用関数
HCI¥EHCI	r_usb_hEhciTransfer.c	EHCI 転送処理
HCI¥include	r_usb_hEhciDefUsr.h	EHCI ユーザ設定定義
HCI¥include	r_usb_hEhciExtern.h	EHCI 外部参照定義
HCI¥include	r_usb_hEhciTypedef.h	EHCI 変数型定義
HCI¥include	r_usb_hHciLocal.h	HCI 共通参照ヘッダファイル
HCI¥include	r_usb_hOhciDefUsr.h	OHCI ユーザ設定定義
HCI¥include	r_usb_hOhciExtern.h	OHCI 外部参照定義
HCI¥include	r_usb_hOhciTypedef.h	OHCI 変数型定義
HCI¥OHCI	r_usb_hOhciMain.c	OHCI 主要関数
HCI¥OHCI	r_usb_hOhciMemory.c	OHCI メモリリソース用関数
HCI¥OHCI	r_usb_hOhciTransfer.c	OHCI 転送処理
LIB	r_usb_cDataIO.c	データリード／ライト、FIFO アクセス処理
LIB	r_usb_cIntHandler.c	USB 割り込みハンドラ
LIB	r_usb_cLibUSBIP.c	USB ライブラリ関数
include	r_usb_cDefUSBIP.h	USB ドライバ用各種定義
include	r_usb_cItron.h	システムヘッダファイル
include	r_usb_cMacItron.h	マクロ定義（スケジューラマクロ）
include	r_usb_cMacPrint.h	マクロ定義（デバッグ表示用マクロ）
include	r_usb_cTypedef.h	変数型定義
include	r_usb_cDefHCIIP.h	59xIP 互換定義
include	r_usb_cDefUsr.h	ユーザ設定（H/W 動作指定）定義
include	r_usb_cDefUsrPb.h	USB ドライバ用各種定義
include	r_usb_cExtern.h	USB-BASIC-F/W 外部参照定義
include	r_usb_cRevision.h	共通ライブラリレビジョン指定
include	r_usb_cKernelId.h	μitronOS 用 ID 定義
include	r_usb_hHci.h	EHCI/OHCI 機能提供用の外部参照定義
class¥hubd	r_usb_hHubsys.c	HUBCD 関数
class¥SMPL	r_usb_smp_cSub.c	共通ライブラリ関数
class¥SMPL	r_usb_smp_hSub.c	ホストスタンダードリクエスト
SmplMain	main.c	サンプルメインプログラム
SmplMain	SH7786.c	SH7786 用 処理
SmplMain	SH7786_Extern.h	SH7786 用 外部参照定義
APL	r_usb_HSMPL_apl.c	ホスト Port1 サンプルアプリ

Folder	ファイル名	概要
APL	r_usb_cdata.c	Port1 サンプルアプリ用データ転送用データ
APL	r_usb_cdata.h	Port1 サンプルアプリ用データ転送設定

6.3 uITRON システム資源

USB-BASIC-F/Wで使用している uITRON 資源は以下のとおりです。"r_usb_cKernelId.h"ヘッダファイルで定義します。

表6.2 uITRON 資源

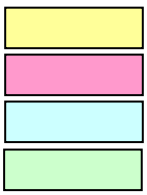
	名称	概要
タスク スタックサイズ： USB_TSK_STK（0x0800）	usb_hstd_HcdTask	HCD タスク Task_ID：USB_HCD_TSK タスク優先順位： USB_HCD_PRI
	usb_hstd_MgrTask	MGR タスク Task_ID：USB_MGR_TSK タスク優先順位： USB_MGR_PRI
	Main_Task	メインタスク Task_ID：USB_SMP_TSK タスク優先順位： USB_SMP_PRI
	usb_hHubTask	HUBCD タスク Task_ID：USB_HUB_TSK タスク優先順位： USB_HUB_PRI
	HCD System Task	HCD SYS タスク Task_ID：USB_HCI_SYS_TSK タスク優先順位： USB_HCI_SYS_PRI
	HCD Transfer Task	HCD TRN タスク Task_ID：USB_HCI_TRN_TSK タスク優先順位： USB_HCI_TRN_PRI
メールボックス 最大優先度：1 待ちタスクキュー：FIFO 順 メッセージキュー：FIFO 順	USB_HCD_MBX	HCD 用メール BOXID
	USB_MGR_MBX	MGR 用メール BOXID
	USB_CLS_MBX	HDCD 用メール BOXID
	USB_HUB_MBX	HUBCD 用メール BOXID
	USB_HUBC_MBX	HUBCD コントロール転送用メール BOXID
固定長メモリプール ブロック数： USB_BLK_CNT（0x20） ブロックサイズ： USB_BLK_SIZ（0x40） 待ちタスクキュー：FIFO 順	USB_HCD_MPL	HCD 用メモリプール ID
	USB_MGR_MPL	MGR 用メモリプール ID
	USB_CLS_MPL	HDCD 用メモリプール ID
	USB_HUB_MPL	HUBCD 用メモリプール ID
セマフォ 初期値：1 最大数：1 待ちタスクキュー：FIFO 順	ID：USB_TRN_SEM	転送管理用セマフォ

セマフォ 初期値：0 最大数：32 待ちタスクキュー：FIFO 順	ID：USB_HCI_SYS_SEM	HCD System 用セマフォ
	ID：USB_HCI_TRN_SEM	HCD Transfer 用セマフォ
セマフォ 初期値：1 最大数：1 待ちタスクキュー：FIFO 順	ID：USB_HCI_MEM_SEM	HCI メモリ管理用セマフォ
セマフォ 初期値：0 最大数：1 待ちタスクキュー：FIFO 順	ID：USB_HCI_DC_SEM	HCI デバイス切断通知セマフォ
セマフォ 初期値：0 最大数：1 待ちタスクキュー：FIFO 順	ID：USB_HCI_TRCC_S_SEM	HCD System Task 用転送キャンセル完了通知セマフォ
セマフォ 初期値：0 最大数：1 待ちタスクキュー：FIFO 順	ID：USB_HCI_TRCC_T_SEM	HCD Transfer Task 用転送キャンセル完了通知セマフォ
OS のベースタイマ	HW タイマ	1ms

6.4 セクション

USB-BASIC-F/Wで使用するセクションは以下のとおりです。

セクション名	概要
P_usblib	USB-F/W 共通処理プログラム領域
P_hcd	HCD、HCI プログラム領域
P_hub	HUB クラスプログラム領域
C_usblib	USB-F/W 共通処理 固定値データ領域
C_hcd	HCD、HCI 固定値データ領域
C_hub	HUB クラス固定値データ領域
D_usblib	USB-F/W 共通処理初期値ありメモリ領域
D_hcd	HCD、HCI 初期値ありメモリ領域
D_hub	HUB クラス初期値ありメモリ領域
B_usblib	USB-F/W 共通処理初期値無しメモリ領域
B_hcd	HCD、HCI 初期値無しメモリ領域
B_hub	HUB クラス初期値無しメモリ領域
B_ehci_non_cache	EHCI 転送用初期値無しメモリ領域＊１、＊４
B_ohci_non_cache	OHCI 転送用初期値無しメモリ領域＊２
B_hcd_non_cache	コントロール転送用初期値無しメモリ領域＊３



プログラム領域
固定データ領域
初期値有りデータ領域
初期値無しデータ領域

＊１　B_ehci_non_cache は非キャッシュ領域で 4KByte アライメントのアドレスに配置して下さい。

＊２　B_ohci_non_cache は非キャッシュ領域で 256Byte アライメントのアドレスに配置して下さい。

＊３　非キャッシュ領域に配置して下さい。

＊４　B_ehci_non_cache において、アイソクロナス転送用ディスクリプタの変数
ehci_Itd[USB_EHCI_NUM_ITD]は 64byte アライメントに配置してください。

6.5 uITRON コンフィギュレータ

USB-BASIC-F/Wは ITRON コンフィギュレータを使用して以下の設定を行っています。

1. カーネル動作条件：カーネル割り込みマスクレベル【14】
2. 時間管理機能：時間管理機能を使用する。
   タイマ割り込み番号【0x0400】
   タイマ割り込みレベル【13】
   タイマイベントハンドラスタックサイズ【0x0200】
   タイムティック周期【1ms】
3. サービスコールの選択：すべてのサービスコールを使用する。
4. 割り込み、CPU 例外ハンドラ
   【Power On Reset】
   【TRAPA】
   【SYSTEM TIMER】
   【UsbInt_Hand：C 言語】
   【_kernel_tmrint：C 言語】
5. 初期化ルーチン【PowerON_Reset_PC：スタックサイズ 0x0100：C 言語】

6.6 uITRON タスクの生成と起動

USB-BASIC-F/Wは初期化ルーチン（PowerON_Reset()関数）でメインタスクとメモリプールとメールボックスを生成しメインタスクを起動します。メインタスクでは HCD、MGR タスクを生成し起動します。HDCD は Set_Configuration リクエスト応答のタイミングでタスクを生成し起動します。

6.7 uITRON システムコール

USB-BASIC-F/Wは以下のシステムコールを使用しています

表6.3 システムコール

システムコール	概要
USB_CRE_TSK	タスクを生成します。
USB_DEL_TSK	タスクを削除します。
USB_STA_TSK	タスクを起動します。
USB_TER_TSK	タスクを中断します。
USB_DLY_TSK	指定した時間待ちます。
USB_CRE_MBX	メールボックスを生成します。
USB_DEL_MBX	メールボックスを削除します。
USB_SND_MSG	タスク（割り込み）からタスクにメッセージを送信します。
USB_ISND_MSG	タスク（割り込み）からタスクにメッセージを送信します。
USB_RCV_MSG	メッセージバッファからメッセージを受信します。
USB_PRCV_MSG	メッセージバッファからメッセージを受信します。
USB_TRCV_MSG	タイムアウト付きでメッセージを受信します。
USB_CRE_MPL	メモリプールを生成します。
USB_DEL_MPL	メモリプールを削除します。
USB_PGET_BLK/ USB_IPGET_BLK	タスク（割り込み）で可変長メモリブロックを取得します。
USB_REL_BLK	可変長メモリブロックを開放します。
USB_CRE_SEM	セマフォを生成します。
USB_WAI_SEM	セマフォを取得します。
USB_POL_SEM	セマフォを確認します。
USB_SIG_SEM/ USB_ISIG_SEM	タスク（割り込み）でセマフォを開放します。
USB_DEL_SEM	セマフォを削除します。

7. ホストドライバ（HCD）

7.1 基本機能

HCD は、MGR、HUBCD、および HDCD から発行される H/W に対する要求を解析し HCD TRN 及び、HCD SYS に対してデータ転送又は、H/W 制御を要求します。HCD の機能は以下のとおりです。

（1）Control 転送要求

（2）Data 転送（Bulk/Isochronous/Interrupt）要求

（3）USB バスリセット要求およびリセットハンドシェイク結果通知

（4）サスペンド信号／レジューム信号送出

7.2 HCD に対する要求発行

HCD に対して H/W 制御要求を発行する場合は、後述の API 関数を用います。なお、接続されているデバイスステートの更新要求は、HCD が直接制御する場合と USB ハブ経由で制御する場合があります。MGR がデバイスアドレスを判断し制御要求を HCD / HUBCD タスクに発行します。

7.3 HCD 使用時の注意事項

HCD を使用して USB 通信を行う場合は以下の注意事項があります

7.3.1 バス占有率とパイプの競合

HCD は複数のデバイスに対しての通信要求を出すことが可能です。ただし、HCD は USB バス占有率の計算、および HDCD が使用する通信パイプの競合チェックなどは行いません。複数の HDCD を実装する場合は、パイプの競合が発生しないよう HDCD を改変する必要があります。なお、HUBCD は PIPE6～PIPE10 を使用しています。

7.3.2 デバイスアドレス

MGR は接続されたデバイスをユーザ指定の"USB_DEVICEADDR"としてデバイスとエニュメレーションします。また、HUBCD は、ダウンポートに接続されたデバイスに対して”USB_DEVICEADDR+1”以降のデバイスアドレスを自動的に割り振ります。

7.3.3 複数 HDCD の実装

MGR は複数のデバイスに対して、同時進行でエニュメレーションを行うことができません。

7.3.4 HDCD 起動

MGR（HUBCD）はデバイスと構成が確立された（SET_CONFIGURATION リクエスト応答）場合に、登録されているコールバック関数で HDCD に構成を通知します。HDCD がデータ受信から始まる場合はデータ通信を許可します。

7.4 HCD タスク起動シーケンス

HCD/MGR タスクの起動シーケンスを以下に示します。

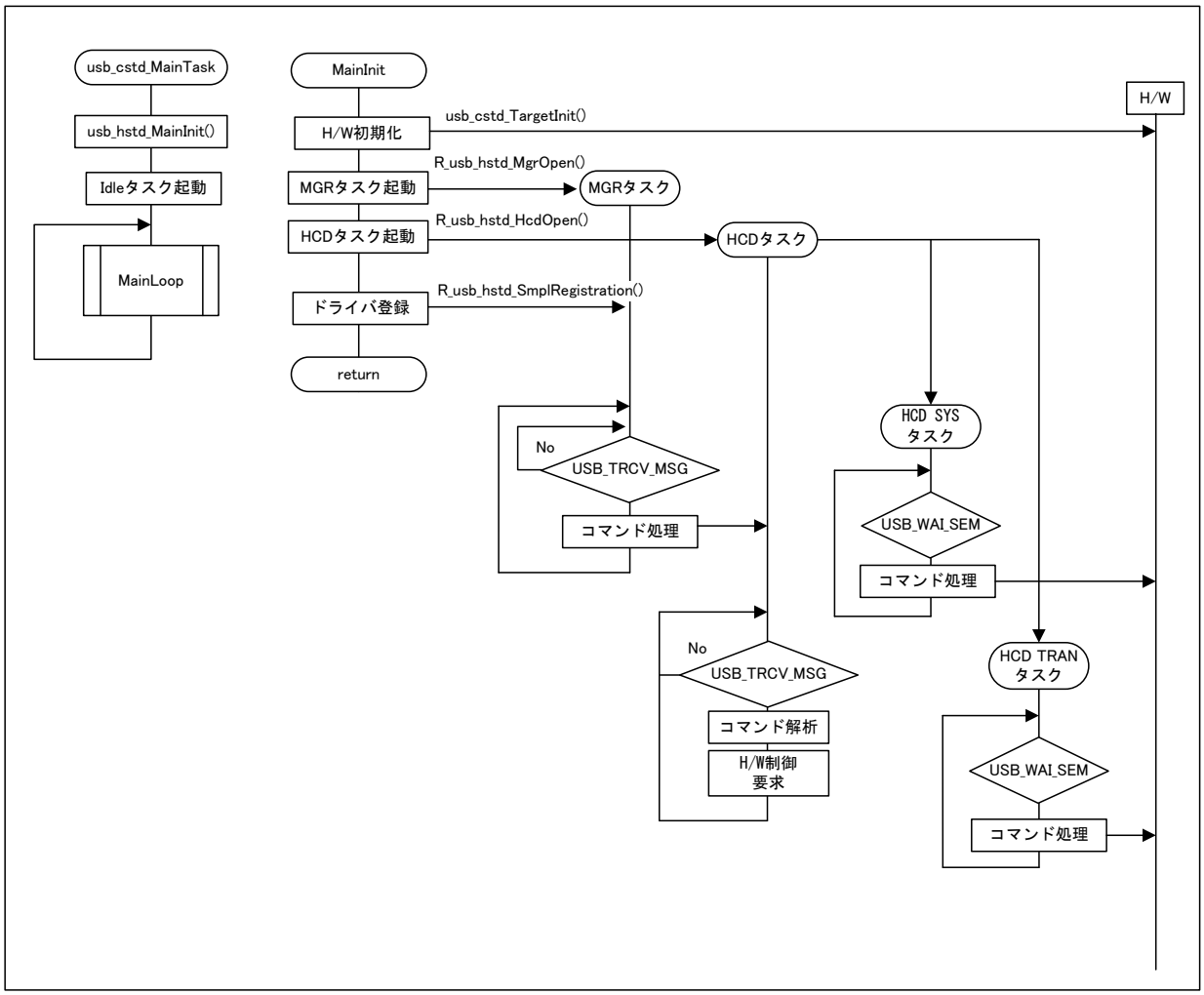


図7.1　HCD/MGR 起動シーケンス

7.5 HCD 概略フロー

HCD の概略フローを以下に示します。

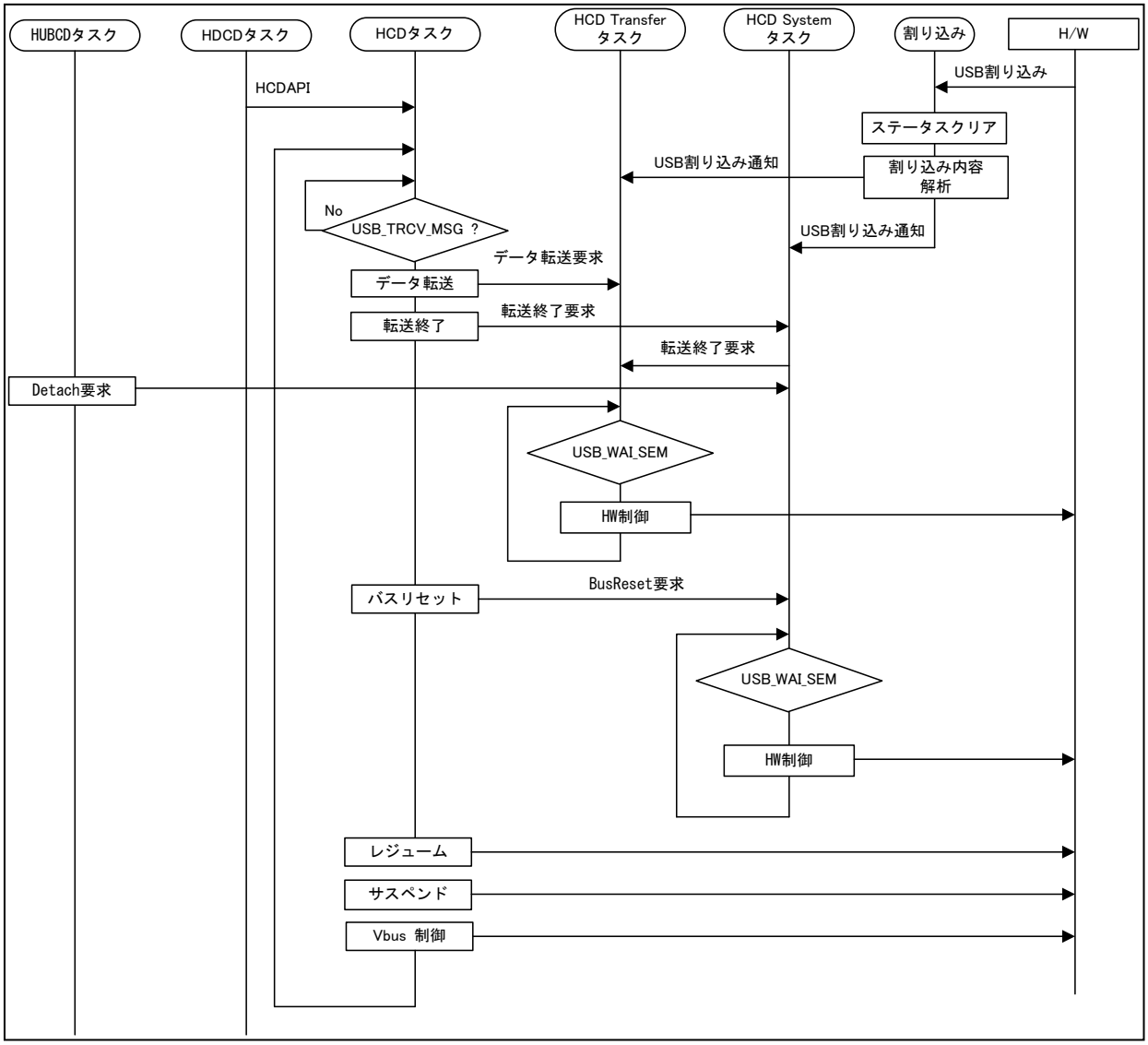


図7.2　HCD 概略フロー

7.6 HCDAPI 関数

MGR、HUBCDおよびHDCDからはAPI関数でH/Wの制御要求を行います。API関数はr_usb_hDriverAPI.cファイルにあり、各API関数の戻り値はスケジューラマクロのエラーコードになります。

表7.1 HCDAPI 関数一覧

	関数名	説明
1	R_usb_hstd_HcdOpen()	HCD タスク起動
2	R_usb_hstd_HcdClose()	HCD タスク停止
3	R_usb_hstd_TransferStart()	データ転送要求
4	R_usb_hstd_SetPipeRegistration()	パイプコンフィグレーション設定要求
5	R_usb_hstd_TransferEnd()	データ転送強制終了要求
6	R_usb_hstd_ChangeDeviceState()	USB デバイスステート変更要求

7.6.1 HCDAPI 関数詳細

表7.2 R_usb_hstd_HcdOpen()

名称	HCD タスク起動		
呼出形式	USB_ER_t R_usb_hstd_HcdOpen(void)		
引数	void	―	―
戻り値	USB_ER_t	―	エラーコード
説明	HCD タスクを起動する関数です。 HCD Transfer タスク、HCD System タスクを生成しグローバル変数の初期設定、タスク、メール BOX、メモリプールの生成を行い、タスクを起動します。 HCD タスクは H/W、MGR、HDCD からの要求待ちになります。		
注意事項	本関数をコールすることにより HCD タスクの起動を行います。起動後はコールしないでください。		

表7.3 R_usb_hstd_HcdClose()

名称	HCD タスク停止		
呼出形式	USB_ER_t R_usb_hstd_HcdClose(void)		
引数	void	―	―
戻り値	USB_ER_t	―	エラーコード
説明	HCD タスクを停止する関数です。 メール BOX、メモリプールを開放しタスクを終了します。		
注意事項	本関数をコールすることにより HCD タスクを停止します。停止後はコールしないでください。		

表7.4　R_usb_hstd_TransferStart()

名称	データ転送要求		
呼出形式	USB_ER_t R_usb_hstd_TransferStart(USB_UTR_t *utr_table)		
引数	USB_UTR_t	*utr_table	転送情報。構造体定義を参照。
戻り値	USB_ER_t		エラーコード
説明	各 PIPE のデータ転送を行う関数です。 データ転送終了（指定データサイズ、ショートパケット受信、エラー発生）時にコールバック関数の引数（utr_table）で送受信の残りデータ長、ステータス、エラーカウントおよび転送終了を通知します。utr_table で示されるデータ転送情報を HCD にメッセージ送信し HCD で処理を実行します。		
注意事項	引数が示す構造体のメンバは、PIPE 番号、転送データの先頭アドレス、転送データ長、Setup packet のアドレス、ステータス、終了時のコールバック関数、エラーカウントを含みます。		

表7.5　R_usb_hstd_SetPipeRegistration()

名称	PIPE コンフィグレーション		
呼出形式	USB_ER_t R_usb_hstd_SetPipeRegistration (uint16_t *table, uint16_t pipe)		
引数	uint16_t	*table	パイプ情報テーブル
	uint16_t	pipe	パイプ番号
戻り値	USB_ER_t		エラーコード
説明	パイプコンフィグレーション情報を設定する関数です。 各 PIPE の設定は HDCD 登録時にレジストされた情報テーブル（devicedriver.pipetable）の内容で設定します。 処理終了時に usb_cstd_ClassProcessResult コールバック関数で設定完了を通知します。 HCD にメッセージを送信し HCD で処理を実行します。		
注意事項	情報テーブル（devicedriver.pipetable）の内容は、Descriptor 内容を解析するなどしてユーザで設定してください。 本関数は本来 R8A6659x 系の ASSP/USB IP で設定を必要とする H/W のパイプコンフィグレーションを設定するための関数です。EHCI 対応の Basic Firm ではパイプコンフィグレーション情報の一部を使用するため、本関数による設定を必要とします。		

表7.6　R_usb_hstd_TransferEnd()

名称	データ転送強制終了要求		
呼出形式	void R_usb_hstd_TransferEnd(uint16_t pipe, uint16_t status)		
引数	uint16_t	pipe	パイプ番号
	uint16_t	status	
戻り値	USB_ER_t		エラーコード
説明	各 PIPE のデータ転送を強制終了する関数です。 データ転送強制終了時に R_usb_hstd_TransferStart()関数のコールバック関数の引数（utr_table）で送受信の残りデータ長、ステータス、エラーカウントおよび強制終了を通知します。 HCD にメッセージを送信し HCD で処理を実行します。		
注意事項	R_usb_hstd_TransferStart()実行時のコールバック関数を実行します。		

表7.7　R_usb_hstd_ChangeDeviceState()

名称	デバイス状態更新要求		
呼出形式	USB_ER_t R_usb_hstd_ChangeDeviceState(USB_CB_INFO_t complete, uint16_t msginfo, uint16_t rootport)		
引数	USB_CB_INFO_t	complete	コールバック関数
	uint16_t	msginfo	メッセージ情報
		connect_inf	接続状態
	uint16_t	rootport	ポート番号
戻り値	USB_ER_t		エラーコード
説明	デバイスの状態管理をする関数です。 処理終了時にコールバック関数で設定完了を通知します。 HCD にメッセージを送信し HCD で処理を実行します。		
注意事項	状態管理を行うメッセージ情報は以下のとおりです。 #define USB_MSG_HCD_ATTACH ：接続状態への遷移要求 #define USB_MSG_HCD_DETACH ：接続状態から切断状態への遷移要求 #define USB_MSG_HCD_USBRESET：USB リセット発行要求 #define USB_MSG_HCD_SUSPEND：サスペンドへの遷移要求 #define USB_MSG_HCD_RESUME ：レジューム信号発行要求 #define USB_MSG_HCD_REMOTE ：リモートウェイクアップ付きサスペンドへの遷移要求 #define USB_MSG_HCD_VBON ：VBUS 出力開始要求 #define USB_MSG_HCD_VBOFF ：VBUS 出力停止要求 #define USB_MSG_HCD_CLR_STALL：STALL クリア要求		

7.7 HCD コールバック関数

表7.8 R_usb_hstd_TransferStart コールバック

名称	データ転送要求のコールバック関数		
呼出形式	(*USB_CB_INFO_t)(USB_UTR_t*);		
引数	USB_UTR_t*		R_usb_hstd_TransferStart()関数の引数の USB_UTR_t ポインタ
戻り値	－	－	－
説明	データ転送終了（アプリケーションから指定されたサイズのデータ転送終了およびショートパケット受信等によるトランスファー終了）で実行されます。 送受信の残りデータ長およびエラーカウントが更新されます。		
注意事項			

表7.9 R_usb_hstd_ChangeDeviceState(USB_MSG_HCD_USBRESET)コールバック

名称	USB バスリセットのコールバック関数		
呼出形式	(*USB_CB_INFO_t)(uint16_t,uint16_t)		
引数	uint16_t		USB_NOCONNECT：未接続 USB_HSCONNECT：Hi-speed デバイス USB_FSCONNECT：Full-speed デバイス USB_LSCONNECT：Low-speed デバイス
	uint16_t		NOARGUMENT：未使用
戻り値	－	－	－
説明	USB バスリセット処理終了で実行されます。 接続されたデバイスの通信速度を引数として応答します。 USB バスリセット中に切断検出した場合、もしくは速度未定の場合は未接続を応答します。		
注意事項			

表7.10 その他コールバック

名称	その他のコールバック関数		
呼出形式	(*USB_CB_INFO_t)(uint16_t,uint16_t)		
引数	uint16_t		NOARGUMENT：未使用
	uint16_t		msginfo：コマンド種別
戻り値	－	－	－
説明	USB_MSG_HCD_ATTACH：アタッチ処理終了で実行されます。 USB_MSG_HCD_DETACH：デタッチ処理終了で実行されます。 USB_MSG_HCD_SUSPEND：サスペンド処理終了で実行されます。 USB_MSG_HCD_RESUME：レジューム処理終了で実行されます。 USB_MSG_HCD_REMOTE：リモートウェイクアップ処理終了で実行されます。 USB_MSG_HCD_VBON：VBUS オン処理終了で実行されます。 USB_MSG_HCD_VBOFF：VBUS オフ処理終了で実行されます。 USB_MSG_HCD_SETDEVICEINFO：パイプ設定処理終了で実行されます。 usb_hstd_ControlEnd： データ転送終了で実行されます。 USB_MSG_HCD_CLRSEQBIT：シーケンストグルビットクリアで実行されます。 USB_MSG_HCD_SETSEQBIT：シーケンストグルビットセットで実行されます。		
注意事項			

7.8 構造体定義

7.8.1 USB_HCDINFO_t 構造体

R_usb_hstd_ChangDeviceState()、R_usb_hstd_SetPipeRegistration()、R_usb_hstd_TransferEnd()、および R_usb_hhub_ChangeState()関数が HCD に送信するメッセージの構造体について説明します。

```
typedef struct {
    USB_MH_t          msghead;    // OS が使用するメッセージヘッダ
    uint16_t          msgInfo;    // USB-BASIC-F/Wが使用するメッセージ情報
    uint16_t          keyword;    //  サブ情報（ポート番号、パイプ番号等）
    void              *tranadr;   // R_usb_hstd_SetPipeRegistration()時のみパイプ情報テーブルを指定
    USB_CB_INFO_t     complete;   //  処理終了時のコールバック関数
}USB_HCDINFO_t;
```

表7.11　USC_HCDINFO_t 構造体のメンバ

変数名	機能
msghead	OS が使用するメッセージヘッダですので、クライアントは使用しないでください。。
msgInfo	USB-BASIC-F/Wが使用しますので、クライアントは使用しないでください。
keyword	以下のようにメッセージによって内容が異なります。 パイプ番号：R_usb_hstd_ChangDeviceState()、R_usb_hstd_SetPipeRegistration()、R_usb_hstd_TransferEnd() デバイスアドレス：R_usb_hhub_ChangeState()
*tranadr	R_usb_hstd_SetPipeRegistration()関数： PIPE 設定情報テーブルのアドレスを指定します。 他の API 関数： 指定されても無視します。
complete	HCD が処理を終了した場合に実行したい関数アドレスを指定します。 コールバック関数は void (*USB_CB_INFO_t)(uint16_t,uint16_t)の型宣言をしてください。 R_usb_hstd_ChangeDeviceState()関数の USB リセット信号出力制御はリセットハンドシェイクの結果をコールバックの第 1 引数として応答します。他の関数はコールバックされたことで処理が終了したことになります。

7.8.2 HCD に登録する情報（USB_HCDREG_t 構造体）

HDCD を登録するための構造体について以下説明します。

コールバック関数が実行されるタイミングは 11 章を参照ください。

```
typedef struct {
    uint16_t            rootport;       /* root port */
    uint16_t            devaddr;        /* Device address */
    uint16_t            devstate;       /* Device satate */
    uint16_t            ifclass;        /* Interface Class */
    uint16_t            *tpl;           /* Target Peripheral List */
    uint16_t            *pipetbl ;      /* Pipe Define Table address */
    USB_CB_INFO_t       classinit;      /* Driver init */
    USB_CB_CHECK_t      classcheck;     /* Driver check */
    USB_CB_INFO_t       devconfig;      /* Device configuered */
    USB_CB_INFO_t       devdetach;      /* Device detach */
    USB_CB_INFO_t       devsuspend;     /* Device suspend */
    USB_CB_INFO_t       devresume;      /* Device resume */
    USB_CB_INFO_t       overcurrent;    /* Device over current */
}USB_HCDREG_t;
```

表7.12　USB_HCDREG_t 構造体のメンバ

変数名	機能
rootport	HCD が使用します。接続されているポート番号が登録されます。
devaddr	HCD が使用します。デバイスアドレスが登録されます。
devstate	HCD が使用します。デバイスの接続状態が登録されます。
ifclass	HDCD が動作するインタフェースクラスコードを登録してください。
*tpl	HDCD が動作するターゲットペリフェラルリストを登録してください。
*pipetbl	パイプ情報テーブルのアドレスを登録してください。
classinit	ドライバ登録時に起動する関数を登録してください。
classcheck	HDCD チェック時に起動する関数を登録してください。 TPL が一致した場合に呼び出します。
devconfig	コンフィガード状態遷移時に起動する関数を登録してください。 SET_CONFIGURATION リクエストが終了した場合に呼び出します。
devdetach	デタッチ状態遷移時に起動する関数を登録してください。
devsuspend	サスペンド状態遷移時に起動する関数を登録してください。
devresume	レジューム状態遷移時に起動する関数を登録してください。
overcurrent	オーバーカレント検出時に起動する関数を登録してください。

デバイスクラスごとにＴＰＬを作成します。TPL は、ベンダ ID とプロダクト ID をセットで登録します。当該デバイスクラスでサポートする全てのセットを登録してください。

全てのデバイスをサポートするデバイスクラスとして登録する場合には、USB_NOVENDOR と USB_NOPRODUCT をセットにして登録してください。

```
const uint16_t tpl[] = {
        1,                              /* Number of list（リスト数） */
        0,                              /* Reserved */
        USB_NOVENDOR,                   /* Vendor ID */
        USB_NOPRODUCT,                  /* Product ID */
};
```

8. HCD Transfer(HCD TRN)

8.1 基本機能

HCD Transfer は、データ転送制御用のプログラムです。HCD から発行されるデータ転送要求に従いデータ転送を行います。コールバック関数で制御結果を HDCD 等に通知します。HCD Transfer の機能は以下のとおりです。

（1）Control 転送および結果通知

（2）Data 転送および結果通知

8.2 HCD Transfer に対する要求発行

HCD Transfer に対するデータ転送要求は HCD から行われます。HUBCD 又は、HDCD から API 経由で HCD に対して転送要求が出され転送情報を設定後、HCD から HCD TRN に対して転送要求を発行します。

転送結果についてはコールバック関数を用いて要求元に直接通知します。

8.3 HCD Transfer タスク起動シーケンス

HCD Transfer タスクの起動シーケンスは、図 7.4　HCD/MGR 起動シーケンスを参照ください。

8.4 HCD Transfer 概略フロー

HCD Transfer の概略フローは、図 7.5　HCD 概略フローを参照ください。

9. HCD System(HCD SYS)

9.1 基本機能

HCD System は、ポート状態の管理及び H/W 制御を行うプログラムです。HCD、HUBCD からの H/W 制御要求に従い H/W 制御を行い、ポート状態変化を検出した場合、MGR に通知します。

HCD System の機能は以下のとおりです。

（1）Port 状態変化(Attach/Detach/オーバカレント)通知

（2）Bus Reset H/W 制御

（3）切断処理 H/W 制御

9.2 Port 状態変化通知

Port 状態の変化は、USB 割り込みハンドラより HCD System に通知され、Port 状態変化通知を受け取った HCD System は Port の状態を確認し、Attach,Detach,オーバカレントの通知を MGR へコールバック関数を通じて通知する

9.3 Bus Reset H/W 制御

Bus Reset H/W制御は、HCDからメッセージでHCD Systemに対して要求が出されます。要求を受け取ったHCD Systemは、要求に従いBus Resetを制御します。

9.4 切断処理 H/W 制御

切断処理H/W制御は、HUBCDからメッセージでHCD Systemに対して要求が出されます。要求を受け取ったHCD Systemは、要求に従い切断処理を制御します。

9.5 HCD System タスク起動シーケンス

HCD System タスクの起動シーケンスは、図 7.4　HCD/MGR 起動シーケンスを参照ください。

9.6 HCD System 概略フロー

HCD System の概略フローは、図 7.5　HCD 概略フローを参照ください。

10. ホストコントロール転送

10.1 概要

セットアップパケット用の USB リクエストおよび、データ送受信用のユーザバッファを HCD に通知する（13.2 章 USB 通信用構造体参照）ことで、コントロール転送を行うことができます。

10.1.1 基本仕様

HCD はデータ転送要求を受け取ると要求内容をチェックし転送に必要な情報を設定後、HCD TRN に対して転送要求を出します。要求を受け取った HCD TRN は、コントロール転送のスケジューリングを行い、構造体で指定されたセットアップ情報，ユーザバッファを転送メモリとして使用し、コントロール転送を実現しています。

ユーザバッファはデータステージで送受信するデータサイズを確保してください。

10.1.2 転送結果の通知

HCD TRN は、コントロール転送が終了するとサブミット時に指定されたコールバック関数を実行し、HDCD に対してコントロール転送の終了を通知します。

HCD TRN は、通信結果を以下の 5 種類で HDCD に通知します。

USB_CTRL_END	：コントロール転送正常終了
USB_DATA_STALL	：データステージ、ステータスステージで STALL 応答 もしくはデータステージ、ステータスステージでMAXPエラー
USB_DATA_OVR	：データステージで受信データサイズオーバー
USB_DATA_ERR	：無応答検出
USB_DATA_STOP	：コントロール転送強制終了（デタッチ検出による終了を含む）

10.2　コントロール転送の動作

10.2.1　コントロールライト動作

HCD TRN は、データコントロール転送を以下の手順で行っています。

① HCD からの転送要求でコントロール転送が開始します。

② HCD から通知された転送要求を元に転送スケジュールを作成後、H/W を制御し通信を開始します。
(ステージ管理は転送スケジュールに従って H/W が実施)

③ 転送完了後、割り込みハンドラから HCD Transfer に転送完了割り込み発生を通知します。

④ 転送完了通知受信後、HCD から指定されたコールバック関数によって転送完了を通知します。

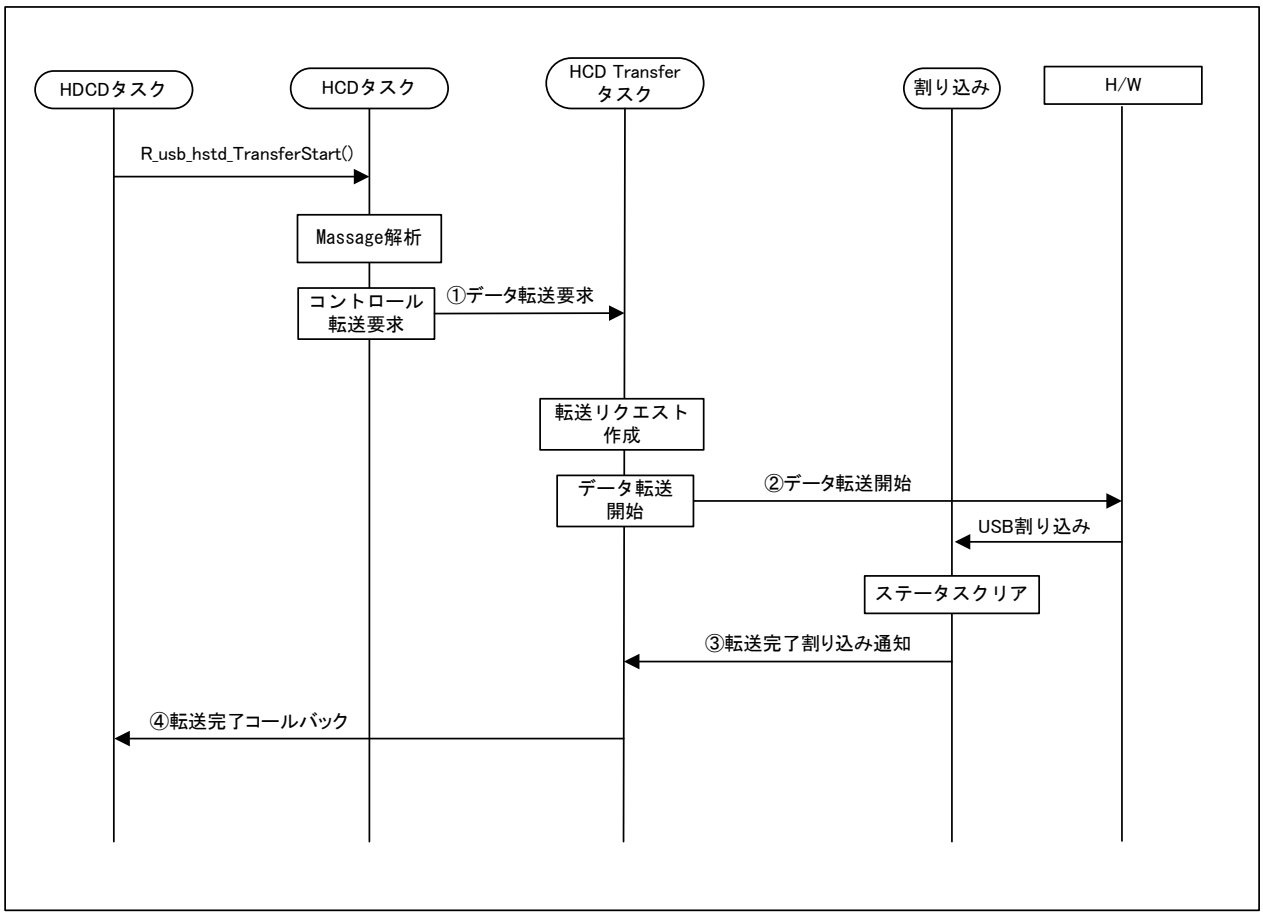


図10.1　コントロールライト概略フロー

10.2.2 コントロールリード動作

HCD TRN は、コントロールリード転送を以下の手順で行っています。

① HCD からの転送要求でコントロール転送が開始します。

② HCD から通知された転送要求を元に転送リクエストを作成しリクエストを H/W に設定し通信を開始します。(ステージ管理は転送スケジュールに従って H/W が実施)

③ 転送完了後、割り込みハンドラから HCD Transfer に割り込み発生を通知します。

④ 転送完了通知受信後、受信データをユーザーバッファにコピーします。

⑤ HCD から指定されたコールバック関数を通じ転送完了を通知します。

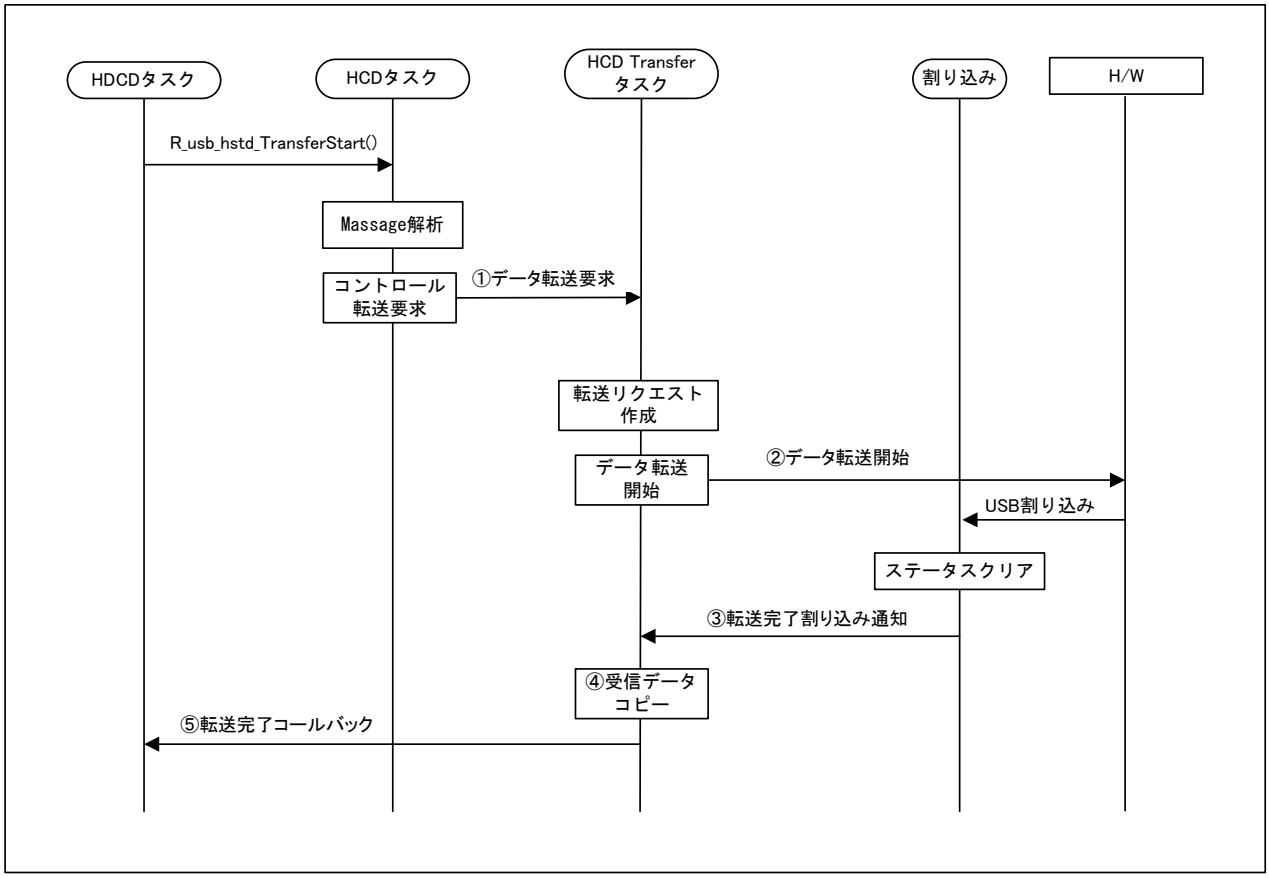
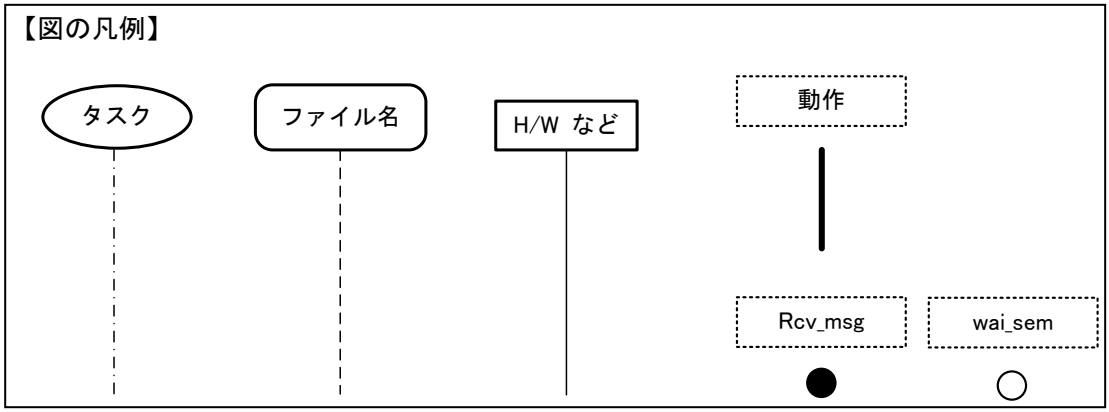


図10.2 コントロールリード概略フロー

10.3 コントロール転送シーケンス



10.3.1 ノーデータコントロールシーケンス

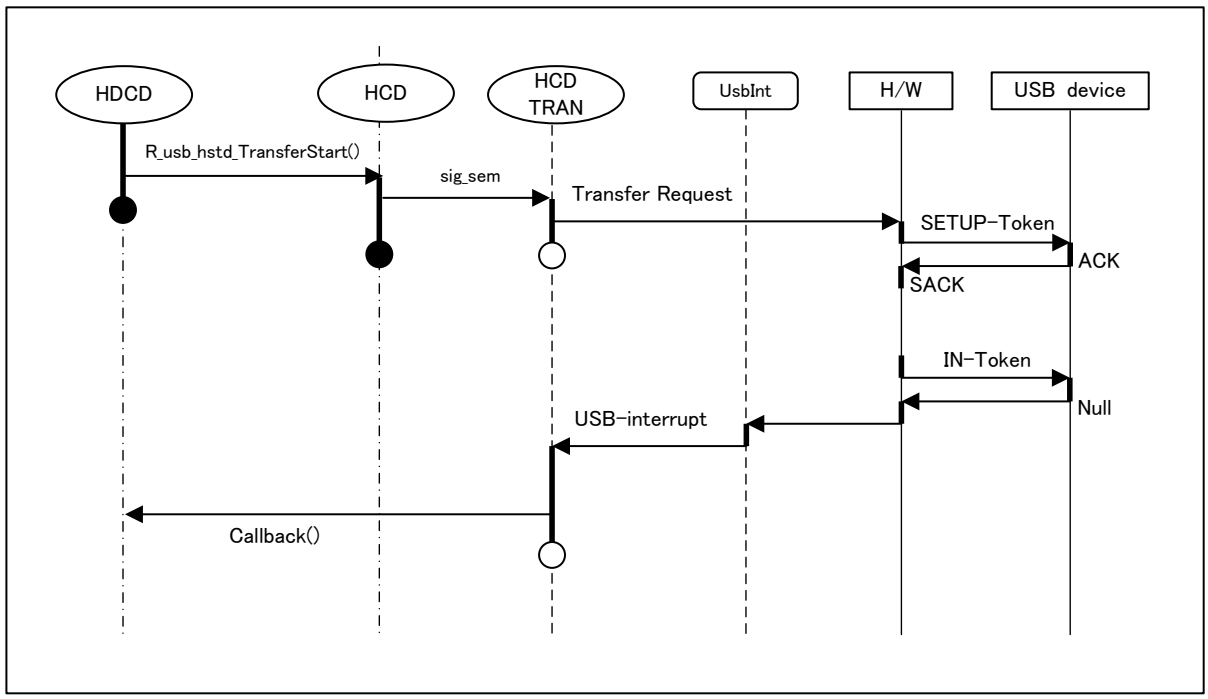


図10.3 ノーデータコントロールシーケンス

10.3.2 コントロールライトシーケンス

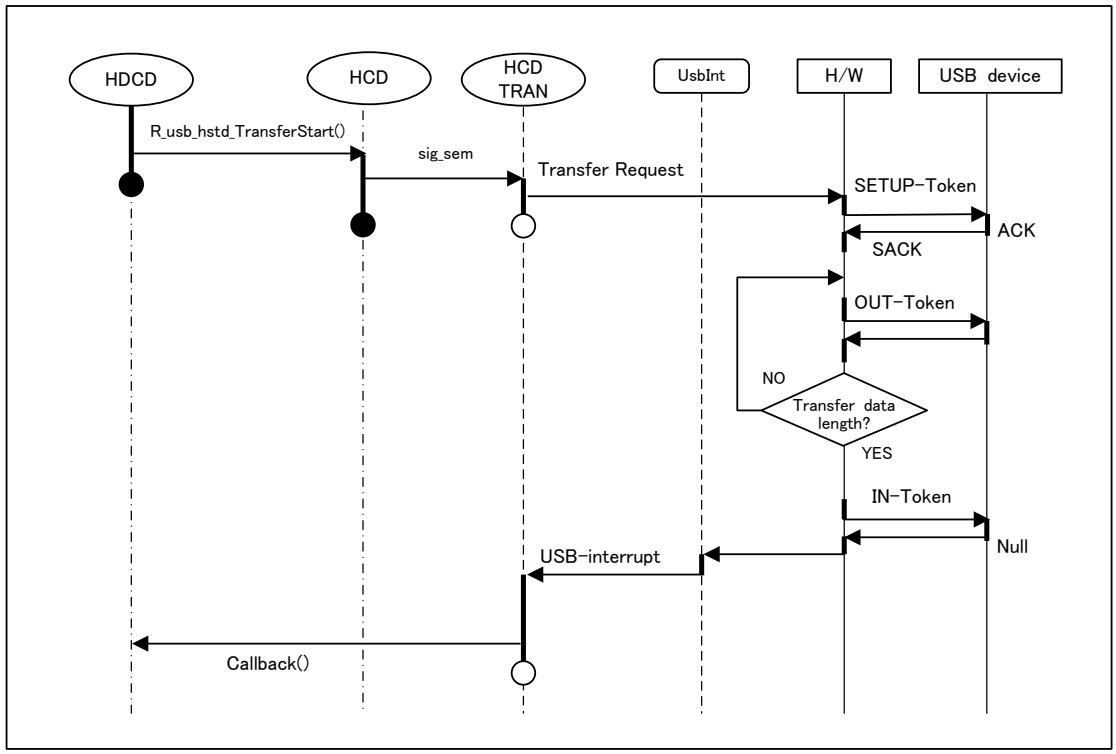


図10.4 コントロールライトシーケンス

10.3.3 コントロールリードシーケンス

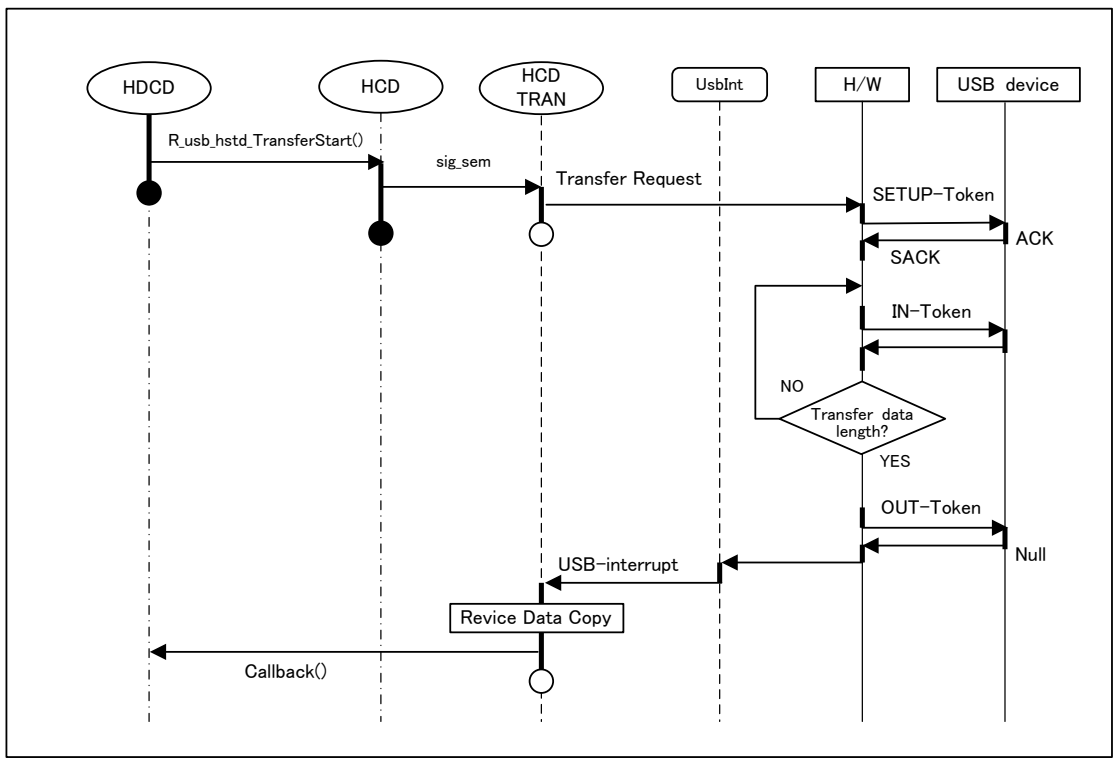


図10.5 コントロールリードシーケンス

11. ホストマネージャ（MGR）

11.1 基本機能

MGR は、HCD モジュールと HDCD 間の機能を補完するタスクです。MGR には、以下の機能があります。

（1）HDCD の登録

（2）接続デバイスの状態管理

（3）接続デバイスのエニュメレーション

（4）Descriptor から Endpoint 情報の検索

（5）HDCD の起動

（6）デタッチ検出

11.2 HDCD の登録

APL は MGRAPI 関数で HDCD を登録します。

11.3 デバイスの状態管理

MGR はデバイス状態変更要求を受けると、要求メッセージのデバイスアドレスが"USB_DEVICEADDR"未満の場合は HCD に対して、デバイスアドレスが"USB_DEVICEADDR"以上の場合は HUBCD に対してデバイス状態変更の要求を発行します。

11.4 USB 標準リクエスト

MGR は HCD System からデバイスアタッチの通知を受けると USB リセットを発行し、継続してエニュメレーションを行います。MGR はリセットハンドシェイクの結果を保存し、下記のコントロール転送を行います。このとき MGR はポート 0 に接続されたデバイスに"USB_DEVICEADDR"を割り振ります。

なお、デバイスから取得した Descriptor 情報は一時保存され、MGRAPI 関数にて取得できます。

（1）GET_DESCRIPTOR (DeviceDescriptor)

（2）SET_ADDRESS

（3）GET_DESCRIPTOR (ConfigurationDescriptor)

（4）SET_CONFIGURATION

11.5 HDCD の確認および起動

MGR はエニュメレーション時に GET_DESCRIPTOR リクエストで取得した情報を、コールバックを通じ HDCD に通知します。HDCD は、コールバック関数内でデバイス情報を確認し接続されたデバイスが動作可能であるかを判定し、結果を MGR に通知します。その結果が動作可能な場合は、エニュメレーション（SET_CONFIGURATION リクエスト実行）後に HDCD を起動します。

11.6 MGR 概略フロー

HCD SYS から callback された場合の MGR の概略フローを以下に示します。

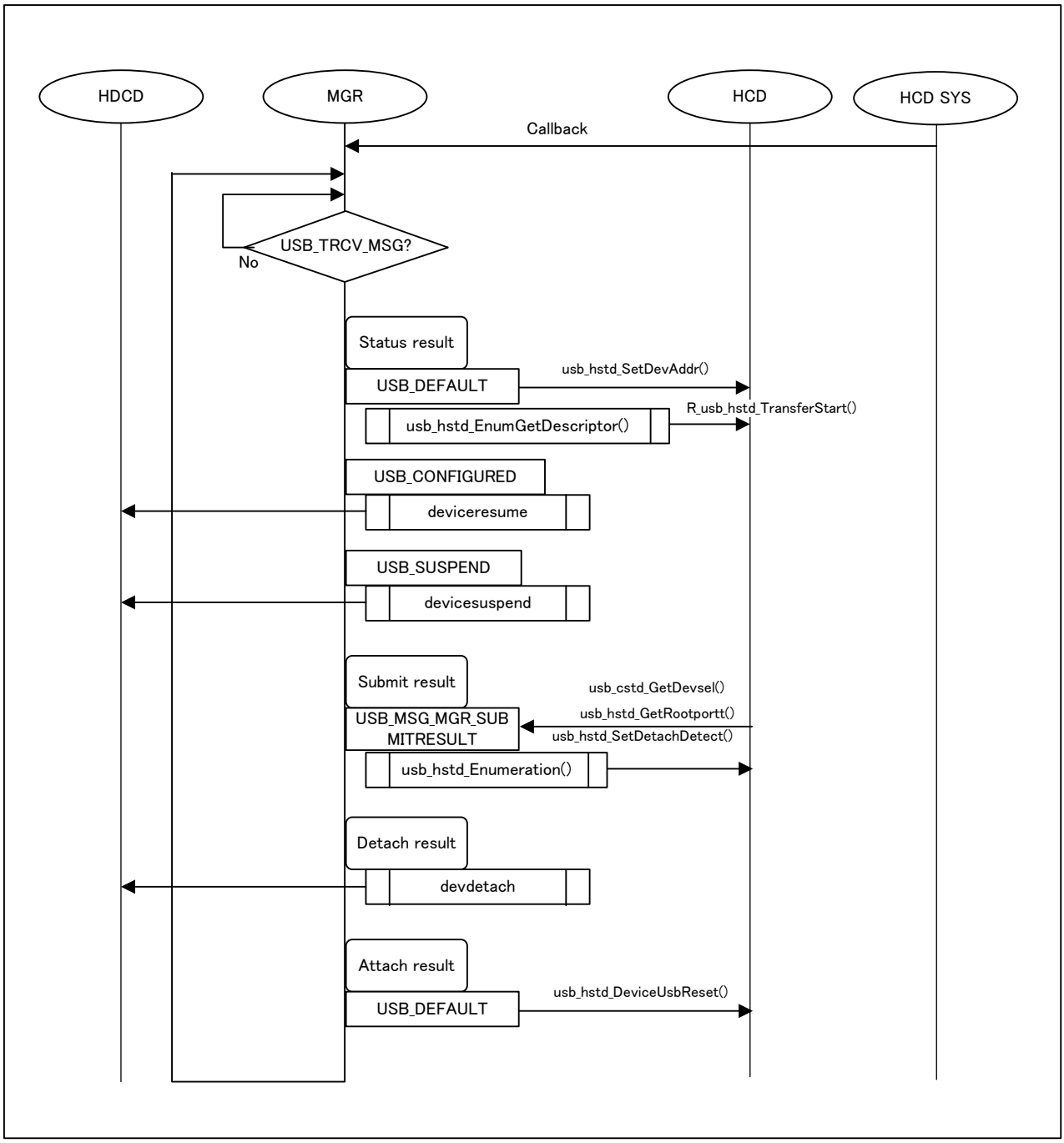


図11.1 MGR 概略フロー1

APL/HDCD から request された場合の MGR の概略フローを以下に示します。

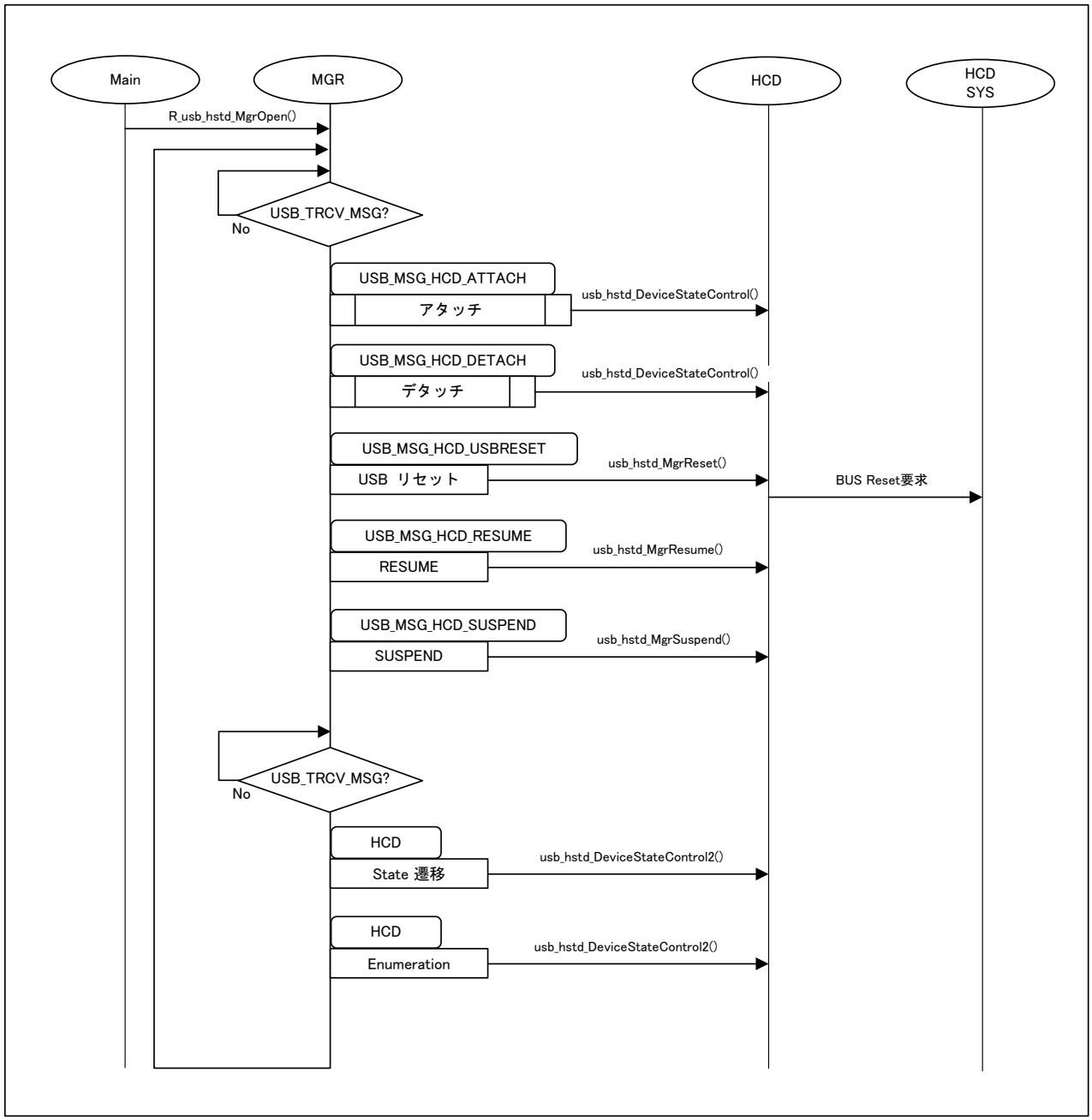


図11.2　MGR 概略フロー2

11.7 MGRAPI 関数

HDCDからはAPI関数でデバイスの状態遷移要求および情報確認を行います。API関数はr_usb_hDriverAPI.cファイルにあります。

表11.1 MGRAPI 関数一覧

	関数名	説明
1	R_usb_hstd_MgrOpen()	MGR タスク起動
2	R_usb_hstd_MgrClose()	MGR タスク停止
3	R_usb_hstd_DriverRegistration()	HDCD 登録
4	R_usb_hstd_DriverRelease()	HDCD 解放
5	R_usb_hstd_MgrChangeDeviceState()	接続されているデバイスの状態変更要求

11.7.1 MGRAPI 関数詳細

表11.2 R_usb_hstd_MgrOpen()

名称	MGR タスク起動		
呼出形式	USB_ER_t R_usb_hstd_MgrOpen(void)		
引数	void	―	―
戻り値	USB_ER_t	―	エラーコード
説明	MGR タスクを起動する関数です。 グローバル変数の初期設定を行い、タスク、メール BOX、メモリプールの生成を行い、タスクを起動します。 MGR タスクは HDCD もしくは HCD からのメッセージ待ち状態になります。		
注意事項	本関数をコールすることにより MGR タスクを起動します。起動後はコールしないでください。		

表11.3 R_usb_hstd_MgrClose()

名称	MGR タスク停止		
呼出形式	USB_ER_t R_usb_hstd_MgrClose(void)		
引数	void	―	―
戻り値	USB_ER_t	―	エラーコード
説明	MGR タスクを停止する関数です。 メール BOX、メモリプールを開放しタスクを終了します。		
注意事項	本関数をコールすることにより MGR タスクを停止します。停止後はコールしないでください。		

表11.4　R_usb_hstd_DriverRegistration()

名称	HDCD 登録		
呼出形式	void R_usb_hstd_DriverRegistration(USB_HCDREG_t *callback)		
引数	USB_HCDREG_t	*callback	
戻り値	void	－	－
説明	HDCD を登録する関数です。 HDCD 情報、パイプ情報テーブル、およびデバイスステート遷移時のコールバック関数を登録します。		
注意事項	登録ドライバ数を更新し、新しい領域に HDCD を登録ます。		

表11.5　R_usb_hstd_DriverRelease()

名称	HDCD 開放		
呼出形式	void R_usb_hstd_DriverRelease(uint8_t devclass)		
引数	uint8_t	devclass	デバイスアドレス
戻り値	void	－	－
説明	HDCD を開放する関数です。 開放されたドライバ領域は開き領域になります。		
注意事項	登録ドライバ数は変化しません。		

表11.6　R_usb_hstd_MgrChangeDeviceState()

名称	接続されているデバイスの状態変更要求		
呼出形式	USB_ER_t R_usb_hstd_MgrChangeDeviceState(USB_B_INFO_t complete, uint16_t msginfo, uint16_t devaddr)		
引数	USB_CB_INFO_t	complete	コールバック関数
	uint16_t	msginfo	メッセージ情報
	uint16_t	devaddr	デバイスアドレス
戻り値	USB_ER_t		エラーコード
説明	USB ポートに接続されているペリフェラルデバイスの状態変更を行う関数です。 本関数は MGR にデバイス状態変更要求をメッセージ送信で通知し、MGR が HCD または HUBCD に処理を実行させます。 処理終了時にコールバック関数で設定完了を上位レイヤーに通知します。 以下の要求を引数（msginfo）に指定し、本関数をコールしてください。 USB_GO_POWEREDSTATE: 接続状態から切断状態への遷移要求 USB_DO_RESET_AND_ENUMERATION: 接続状態から通信初期化後接続状態への遷移要求（USB reset 発行およびエニュメレーション実行要求） USB_PORT_ENABLE: VBUS 供給開始要求 USB_PORT_DISABLE: VBUS シャットダウン要求 USB_DO_GLOBAL_SUSPEND: リモートウェイクアップ許可付きサスペンドへの遷移要求 （Port 以下の全ての接続デバイスへのサスペンド） USB_DO_SELECTIVE_SUSPEND: リモートウェイクアップ許可付きセレクティブサスペンドへの遷移要求（Hub 以下のデバイスに対してデバイス指定サスペンド） USB_DO_GLOBAL_RESUME: グローバルレジューム実行要求 （サスペンドステートからサスペンドする前のステートへの遷移要求） （Port 以下の全ての接続デバイスへのレジューム） USB_DO_SELECTIVE_RESUME: セレクティブレジューム実行要求 （サスペンドステートからサスペンドする前のステートへの遷移要求） （Hub 以下のデバイスに対してデバイス指定レジューム）		
注意事項			

11.8 コールバック関数詳細

表11.7 classcheck

名称	classcheck コールバック関数		
呼出形式	(*USB_CB_CHECK_t)(uint16_t **);		
引数	uint16_t	**Table	Table[0]デバイスディスクリプタ Table[1]コンフィグレーションディスクリプタ Table[2]インタフェースディスクリプタ Table[3]ディスクリプタチェック結果 Table[4]HUB 種別 Table[5]ポート番号 Table[6]通信速度 Table[7]デバイスアドレス
戻り値	－	－	－
説明	ディスクリプタ情報などを HDCD に通知し HDCD が動作可能か否かを結果に応答してください。チェック結果は USB_DONE：HDCD 動作可能 USB_ERROR：HDCD 動作不能 のいずれかを Table[3]に入れ応答してください。		
注意事項			

表11.8 その他コールバック

名称	その他のコールバック関数		
呼出形式	(*USB_CB_INFO_t)(uint16_t ,uint16_t);		
引数	uint16_t		NOARGUMENT：未使用
	uint16_t		NOARGUMENT：未使用
戻り値	－	－	－
説明	classinit ：MGR 起動時に実行されます。 deviceconfig：Set_Configuration リクエスト発行時に実行されます。 devicedetach：デタッチ検出時に実行されます。 devicesuspend：サスペンド遷移時に実行されます。 deviceresume：レジューム遷移時に実行されます。		
注意事項			

11.9 構造体定義

11.9.1 USB_MGRINFO_t 構造体

usb_hstd_NotifAtorDetach()、usb_hstd_StatusResult()、usb_hstd_OvcrNotifiation()、R_usb_hstd_MgrChangeDeviceState()および usb_hstd_SubmitResult()関数が MGR に送信するメッセージの構造体について説明します。

```
typedef struct {
    USB_MH_t        msghead;    // OS が使用するメッセージヘッダ
    uint16_t        msginfo;    // USB-BASIC-F/Wが使用するメッセージ情報
    uint16_t        keyword;    // サブ情報（ポート番号、パイプ番号等）
    uint16_t        result      // 処理結果
} USB_MGRINFO_t;
```

表 11.9 usb_hstd_NotifAtorDetach()、usb_hstd_StatusResult()、usb_hstd_OvcrNotifiation()で使用する際の USB_MGRINFO_t構造体のメンバ

変数名	機能
msghead	OS が使用するメッセージヘッダですので、クライアントは使用しないでください。
msginfo	USB_MSG_MGR_AORDETACH(usb_hstd_NotifAtorDetach()関数で使用)、 USB_MSG_MGR_STATUSRESULT(usb_hstd_StatusResult()関数で使用)、 USB_MSG_MGR_OVERCURRENT(usb_hstd_OvcrNotifiation()関数で使用)
keyword	ポート番号
result	HCD が処理を終了した結果を返します

表 11.10　usb_hstd_SubmitResult()で使用する際のUSB_MGRINFO_t構造体のメンバ

変数名	機能
msghead	OS が使用するメッセージヘッダですので、クライアントは使用しないでください。
msginfo	USB_MSG_MGR_SUBMITRESULT(usb_hstd_SubmitResult()関数で使用)
keyword	パイプ番号
result	HCD が処理を終了した結果を返します

表 11.11　R_usb_hstd_MgrChangeDeviceState()で使用する際のUSB_MGRINFO_t構造体のメンバ

変数名	機能
msghead	OS が使用するメッセージヘッダですので、クライアントは使用しないでください。
msginfo	USB_GO_POWEREDSTATE、 USB_DO_RESET_AND_ENUMERATION、 USB_PORT_ENABLE、 USB_PORT_DISABLE、 USB_DO_GLOBAL_SUSPEND、 USB_DO_SELECTIVE_SUSPEND、 USB_DO_GLOBAL_RESUME、 USB_DO_SELECTIVE_RESUME (R_usb_hstd_MgrChangeDeviceState()関数で使用)
keyword	デバイスアドレス
result	HCD が処理を終了した結果を返します

11.10 デバイスステート管理

11.10.1 状態遷移概略シーケンス

デバイス状態遷移の概略シーケンスを以下に示します。

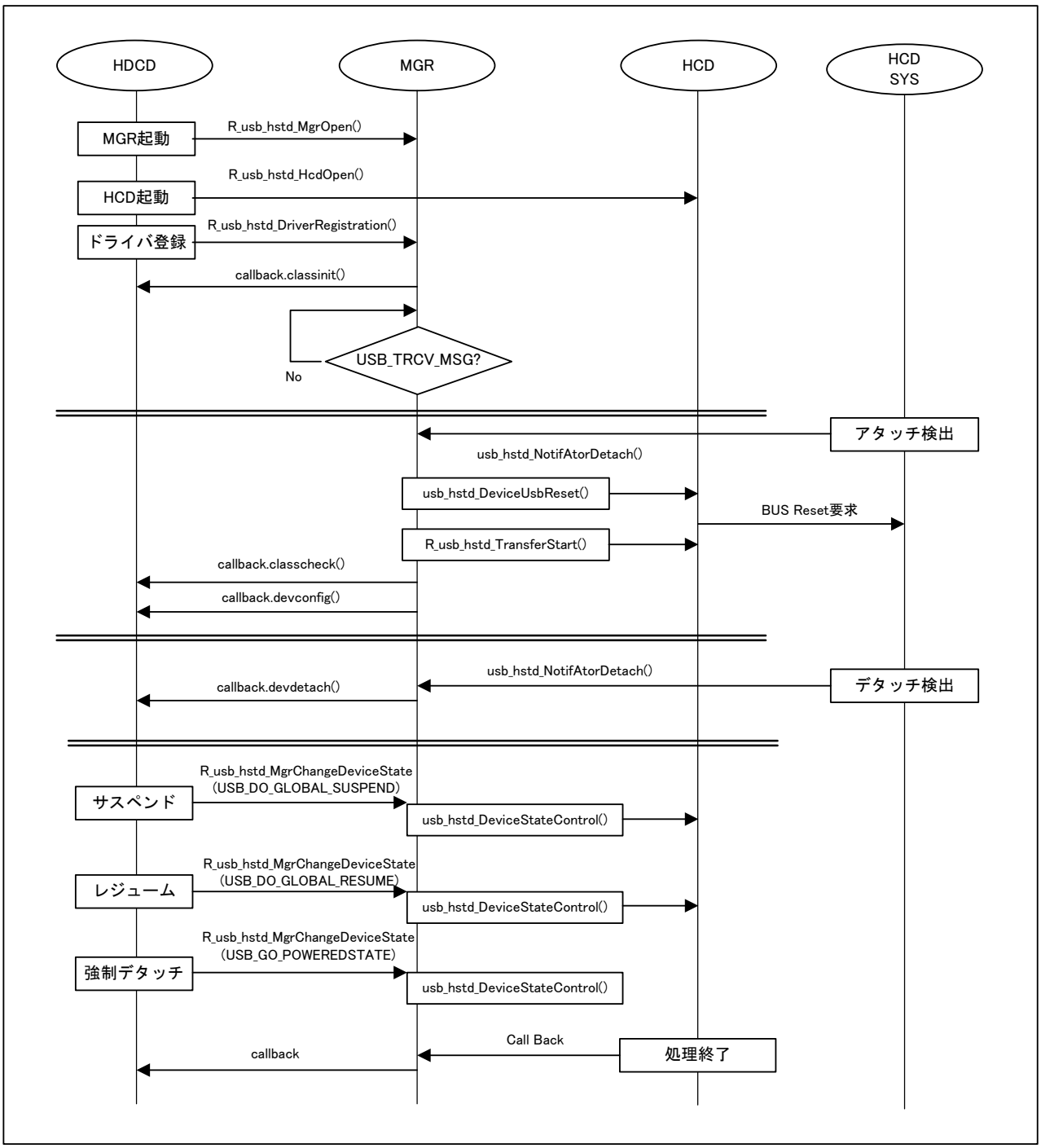


図11.3　状態遷移シーケンス

11.10.2　アタッチ／デタッチ

アタッチ／デタッチシーケンスを以下に示します。

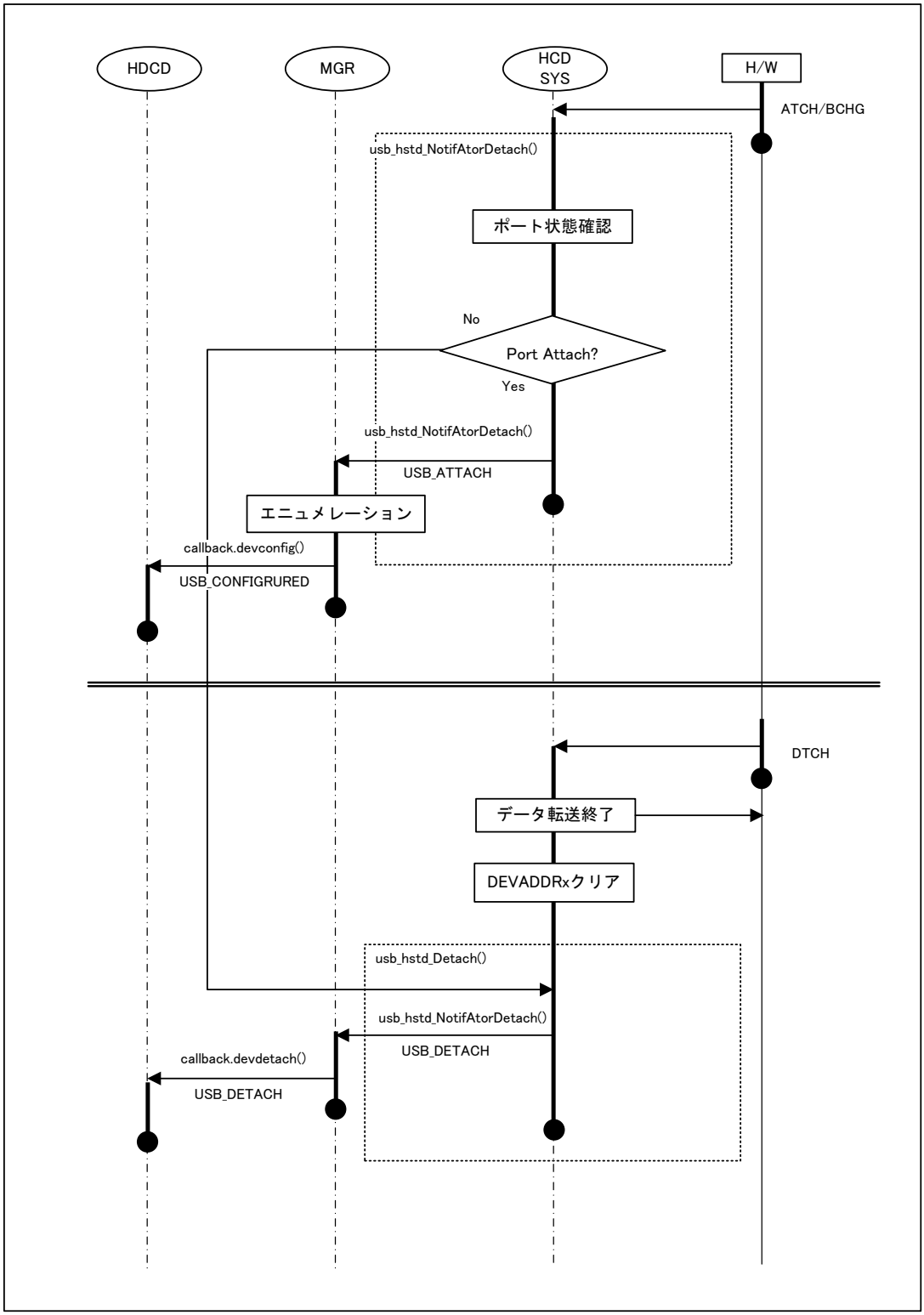


図11.4　アタッチ／デタッチ

11.10.3　サスペンド／レジューム

サスペンド／レジュームシーケンスを以下に示します。

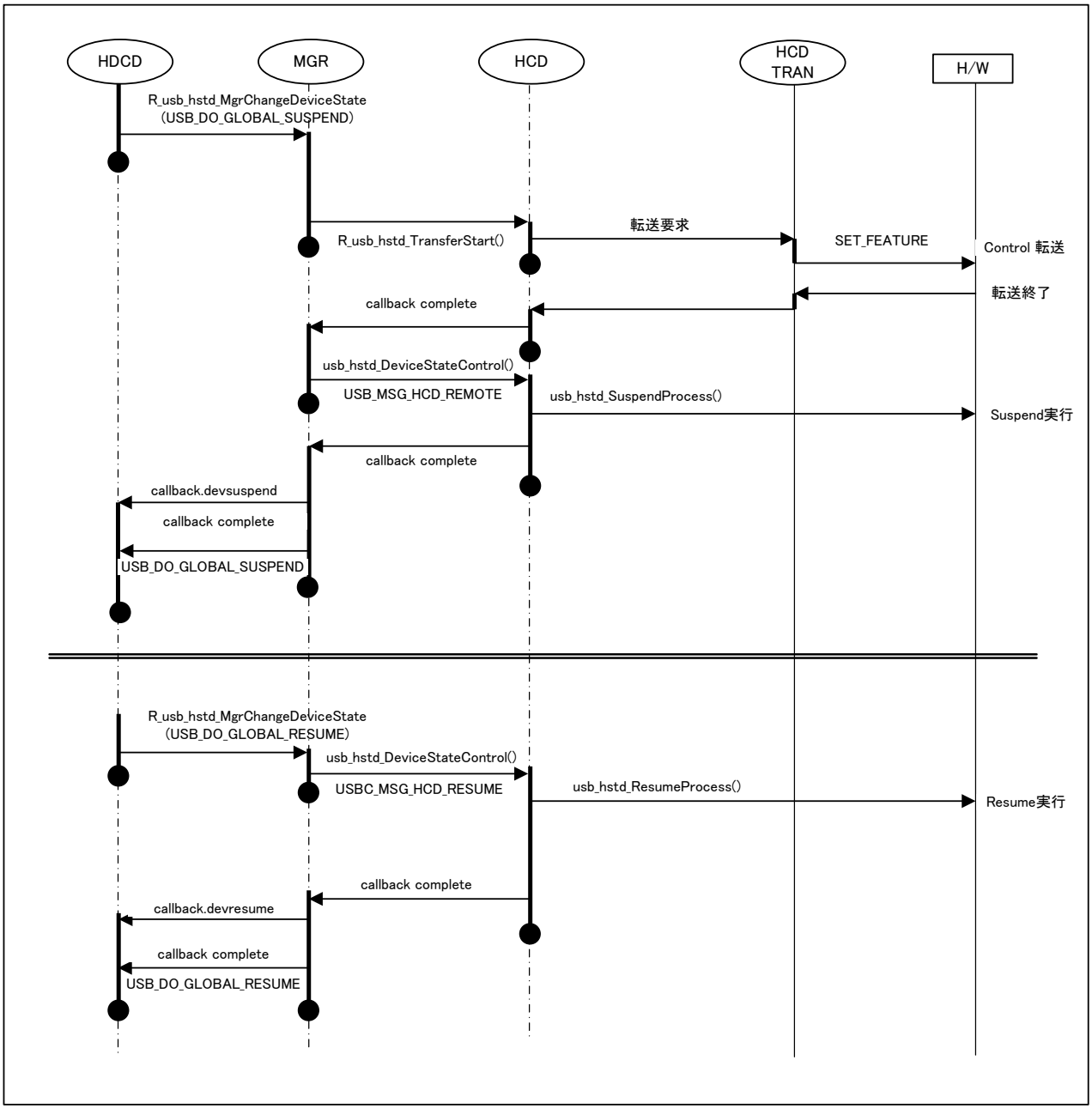


図11.5　サスペンド／レジューム

11.11 エニュメレーション

エニュメレーションシーケンスを以下に示します。

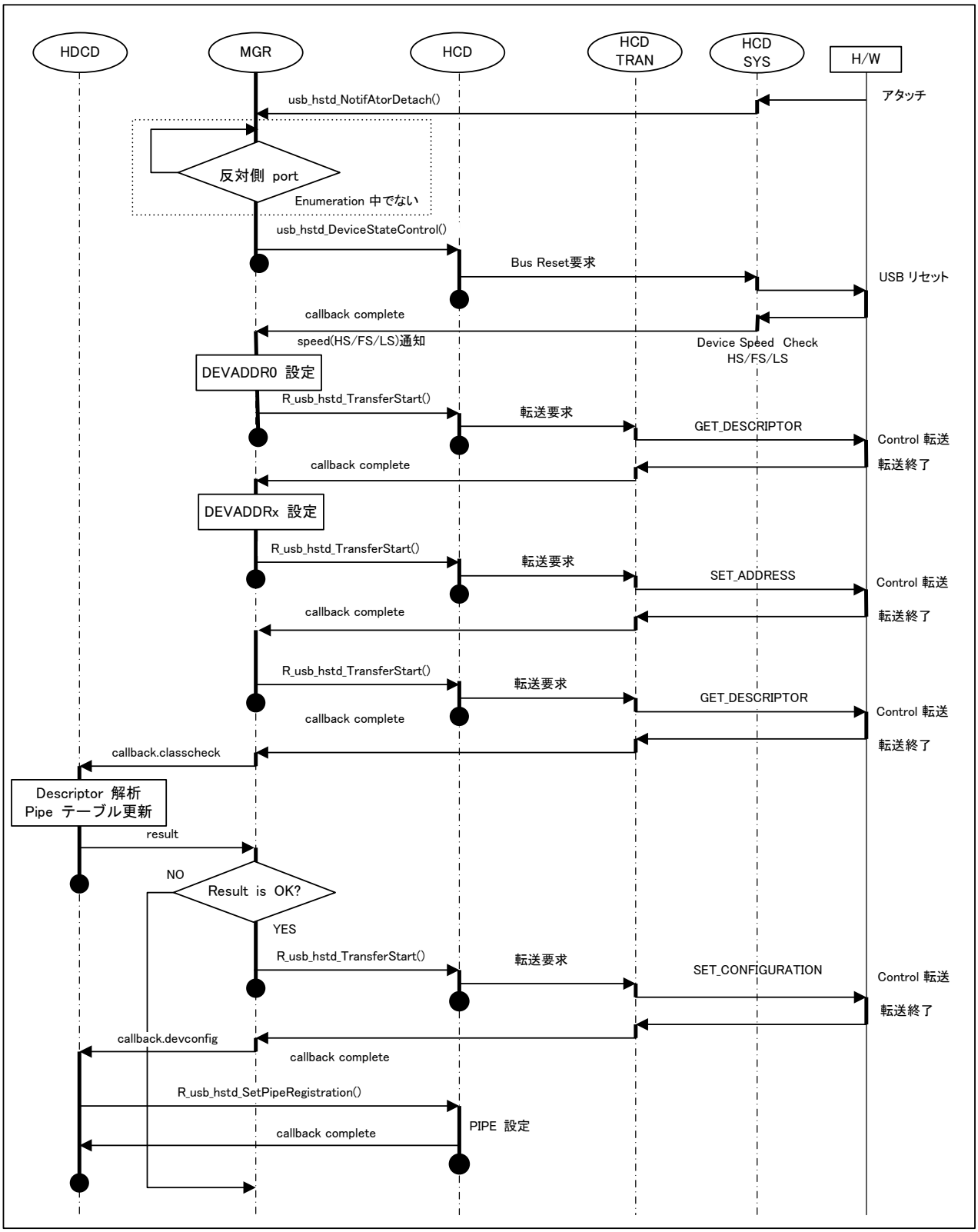


図11.6 エニュメレーション

12. HUB クラスドライバ（HUBCD）

12.1 基本機能

HUBCD は、接続された USB ハブのダウンポートの状態を管理し、HCD と HDCD 間の機能を補完するタスクです。HUBCD には、以下の機能があります。

（1）USB ハブのダウンポートに接続されたデバイスの状態管理

（2）USB ハブのダウンポートに接続されたデバイスのエニュメレーション

（3）HDCD の起動

12.2 HUBCD 用 API 関数

HUBCD用のAPI関数を表12.1に示します。

表12.1 HUBCD API 関数一覧

	関数名	説明
1	R_usb_hhub_Open()	HUBCD タスク起動
2	R_usb_hhub_Close()	HUBCD タスク開放 （通常は MGR が実行するため、上位層からの呼び出しは不要）
3	R_usb_hhub_Registration()	HUBCD 登録
4	R_usb_hhub_ChangeDeviceState()	HUB の下に接続されたデバイスの状態変更要求 （uITRON 対応版のみ）
5	R_usb_hhub_GetHubInformation()	HUB 情報取得
6	R_usb_hhub_GetPortInformation()	HUB　Port 情報取得

12.2.1 HUBCDAPI 関数詳細

表12.2 R_usb_hhub_Open()

名称	HUBCD タスク起動		
呼出形式	USB_ER_t R_usb_hhub_Open(uint16 devaddr, uint16 data2)		
引数	uint16_t	devaddr	USB ハブのデバイスアドレス
	uint16_t	data2	未使用
戻り値	USB_ER_t	－	エラーコード
説明	HUBCD タスクを起動する関数です。 グローバル変数の初期設定を行い、タスク、メール BOX、メモリプールの生成を行い、タスクを起動します。 HUBCD タスクは MGR からの要求待ちになります。 通常は MGR が実行するため、上位層からの呼び出しは不要です。		
注意事項	HUBCD タスク起動後は、本関数をコールしないでください。		

表12.3　R_usb_hhub_Close()

名称	HUBCD タスク停止		
呼出形式	USB_ER_t R_usb_hhub_Close(uint16 hubaddr, uint16 data2)		
引数	uint16_t	hubaddr	USB ハブのデバイスアドレス
	uint16_t	data2	未使用
戻り値	USB_ER_t	－	エラーコード
説明	HUBCD タスクを停止する関数です。 メール BOX、メモリプールを開放しタスクを終了します。 通常は MGR が実行するため、上位層からの呼び出しは不要です。		
注意事項	HUBCD タスク停止後は本関数をコールしないでください。		

表12.4　R_usb_hhub_Registration()

名称	HDCD 登録		
呼出形式	void R_usb_hhub_Registration(void)		
引数	void		
戻り値	void	－	－
説明	本関数は、HUBCD 情報を HCD が管理するテーブルに登録する関数です 本関数は、HCD が管理する登録ドライバ数を更新し、新しい領域に HUBCD を登録ます。		
注意事項			

表12.5　R_usb_hhub_ChangeDeviceState()

名称	HUB の下に接続されたデバイスの状態変更要求		
呼出形式	USB_ER_t R_usb_hhub_ChangeDeviceState(USB_CB_INFO_t complete, uint16_t msginfo, uint16_t devaddr)		
引数	USB_CB_INFO_t	complete	コールバック関数
	uint16_t	msginfo	メッセージ情報
	uint16_t	devaddr	デバイスアドレス
戻り値	USB_ER_t		エラーコード
説明	HUB の下に接続されたデバイスに対して状態変更を要求する関数です。 #define　USB_MSG_HCD_ATTACH：接続状態への遷移要求 #define　USB_MSG_HCD_DETACH：接続状態から切断状態への遷移要求 #define　USB_MSG_HCD_USBRESET：USB リセット発行要求 #define　USB_MSG_HCD_SUSPEND　：サスペンドへの遷移要求 #define　USB_MSG_HCD_RESUME　　：レジューム信号発行要求 #define　USB_MSG_HCD_REMOTE：リモートウェイクアップ許可付きサスペンドへの遷移要求 #define　USB_MSG_HCD_VBON：VBUS 出力開始要求 #define　USB_MSG_HCD_VBOFF：VBUS 出力停止要求 #define　USB_MSG_HCD_CLR_STALL：STALL クリア要求		
注意事項	uITRON 対応版のみ使用します。		

表12.6　R_usb_hhub_GetHubInformation()

名称	HUB 情報取得		
呼出形式	uint16_t R_usb_hhub_GetHubInformation(uint16_t hubaddr, USB_CB_t complete)		
引数	uint16_t	hubaddr	USB ハブのデバイスアドレス
	USB_CB_t	complete	コールバック関数
戻り値	uint16_t	－	エラーコード
説明	HUB の Descriptor 情報を取得する関数です。		
注意事項			

表12.7　R_usb_hhub_GetPortInformation()

名称	HUB　Port 情報取得		
呼出形式	uint16_t R_usb_hhub_GetPortInformation(uint16_t hubaddr, uint16_t port, USB_CB_t complete)		
引数	uint16_t	hubaddr	USB ハブのデバイスアドレス
	uint16_t	port	Port 番号
	USB_CB_t	complete	コールバック関数
戻り値	uint16_t	－	エラーコード
説明	HUB の Port 状態を取得する関数です。		
注意事項			

12.3　ダウンポートの状態管理

HUBCD はタスクが起動されるとすべてのダウンポートに対して

1) ポートパワー許可（HubPortSetFeature：USB_HUB_PORT_POWER）

2) ポートの初期化（HubPortClrFeature：USB_HUB_C_PORT_CONNECTION）

3) ポートステータスの取得（HubPortStatus：USB_HUB_PORT_CONNECTION）

を実施し、ダウンポートのデバイス接続状況を確認します。

12.4　ダウンポートへのデバイス接続

HUBCD は USB ハブからダウンポートのデバイスアタッチの通知を受けると USB ハブに対して USB リセット信号要求を発行（HubPortSetFeature：USB_HUB_PORT_RESET）します。その後、MGR タスクの資源を使って接続されたデバイスとエニュメレーションを行います。HUBCD はリセットハンドシェイクの結果を保存し、"USB_DEVICEADDR+1"以降のアドレスを順次割り振ります。

12.5 クラスリクエスト

HUBCD がサポートしているクラスリクエストは以下のとおりです。

表12.8 USB ハブクラスリクエスト

リクエスト	実装状況	関数名	機能
ClearHubFeature	×		
ClearPortFeature	○	usb_hhub_PortClrFeature	USB_HUB_PORT_ENABLE USB_HUB_PORT_SUSPEND USB_HUB_C_PORT_CONNECTION USB_HUB_C_PORT_ENABLE USB_HUB_C_PORT_SUSPEND USB_HUB_C_PORT_OVER_CURRENT USB_HUB_C_PORT_RESET
ClearTTBuffer	×		
GetHubDescriptor	○	R_usb_hhub_GetHubInformation	ディスクリプタ取得
GetHubStatus	×		
GetPortStatus	○	R_usb_hhub_GetPortInformation	ポートステータス取得
ResetTT	×		
SetHubDescriptor	×		
SetHubFeature	×		
SetPortFeature	○	usb_hhub_PortSetFeature	USB_HUB_PORT_POWER USB_HUB_PORT_RESET USB_HUB_PORT_SUSPEND USB_HUB_C_PORT_ENABLE
GetTTState	×		
StopTT	×		

12.6　ダウンポートのデバイスステート管理

12.6.1　HUB へのアタッチ／デタッチ

アタッチ／デタッチシーケンスを以下に示します。

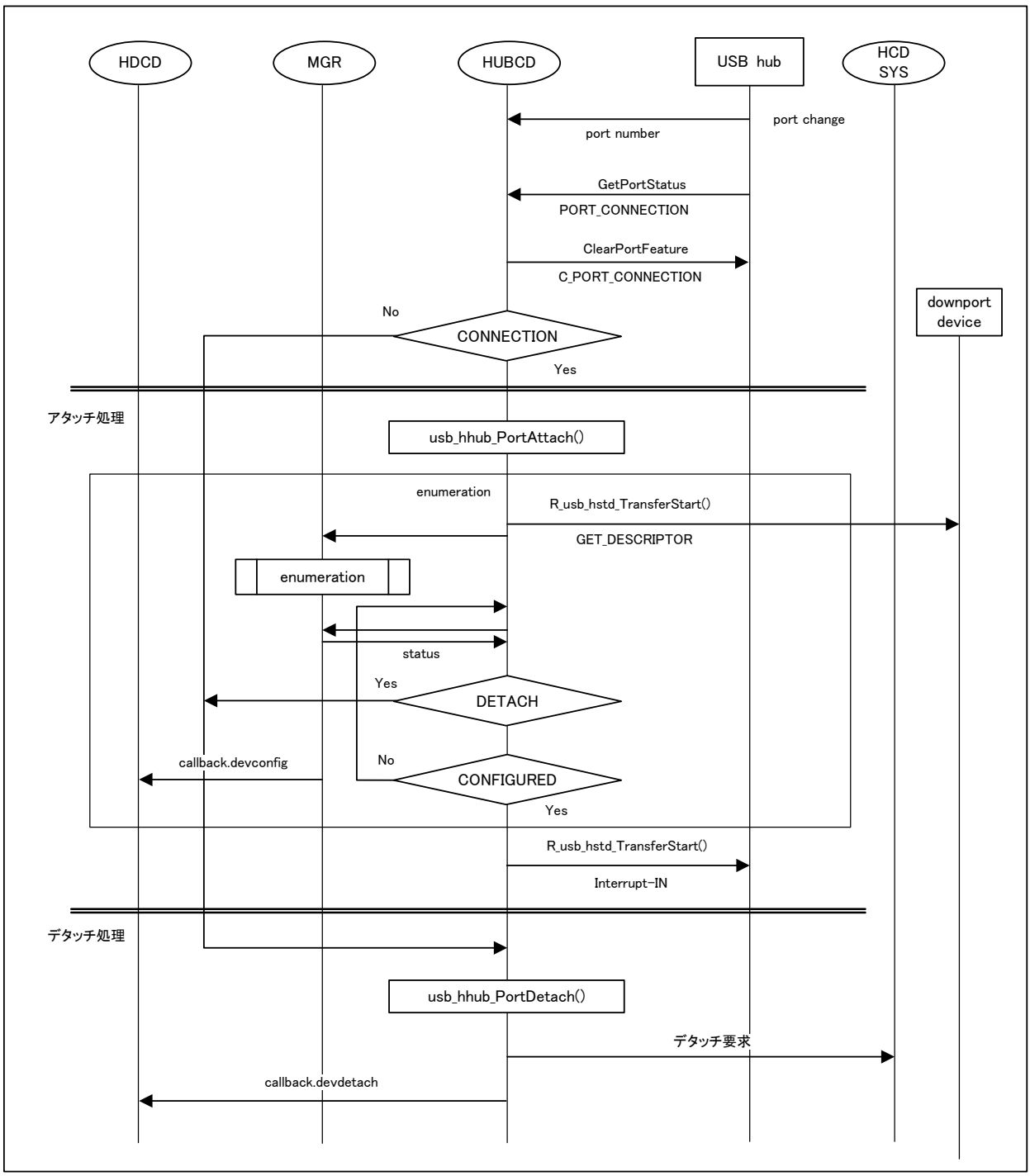


図12.1　HUB へのアタッチ／デタッチ

12.7　ダウンポート状態管理とエニュメレーション

エニュメレーションシーケンスを以下に示します。

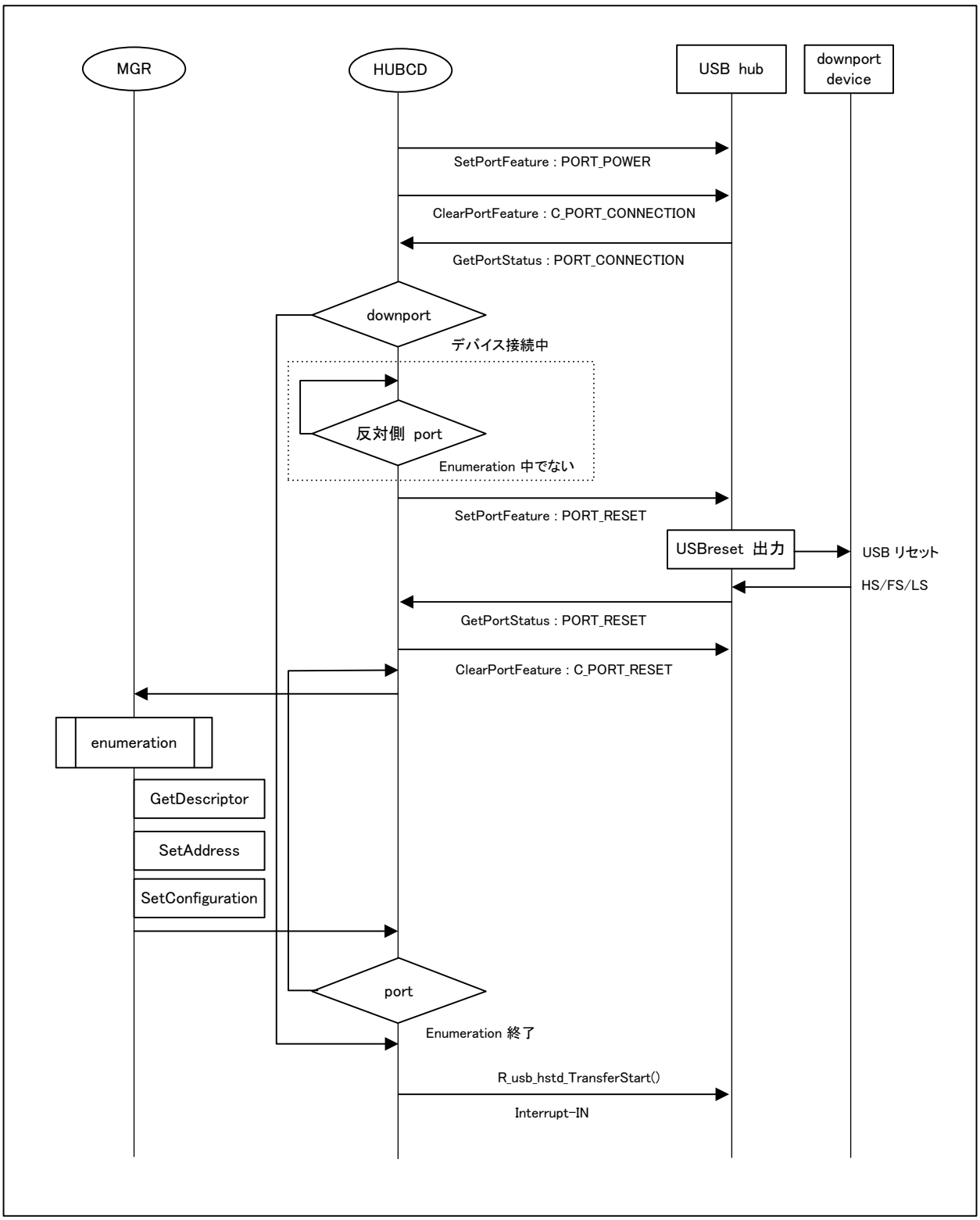


図12.2　ダウンポートデバイスのエニュメレーション

13. データ転送

13.1 概要

USB-BASIC-F/Wはデータ送受信用のユーザバッファを HCD に通知することで、データ転送を行うことができます。データ転送はお客様固有の機能仕様であり、転送方法、通信開始／終了タイミング、およびバッファ構成などは、システムにあわせて変更していただく必要があります。

13.1.1 基本仕様

USB-BASIC-F/Wは、構造体で指定されたユーザバッファと H/W 間のデータ転送を、パイプ情報テーブルで指定された情報に従って実現しています。

＊ EHCI対応USB-BASIC-F/WはR8A6659x向けUSB-BASIC-F/WとI/F互換を取るため、パイプ情報テーブルを使用しますが、パイプ情報テーブル内のR8A6659x H/W依存設定（例えばFIFOのバッファサイズ等）は使用しません。

13.1.2 転送結果のフィードバック

USB-BASIC-F/Wはデータ転送が終了すると登録されたコールバック関数で、HDCD に対してデータ転送の終了を通知します。

USB-BASIC-F/Wは、通信結果を以下の 9 種類で HDCD に通知します。

USB_CTRL_END：	Control 転送が正常に終了した場合
USB_DATA_NONE：	データ送信が正常終了した場合
USB_DATA_OK：	データ受信が正常終了した場合
USB_DATA_SHT：	データ受信は正常終了したが指定されたデータ長未満で終了した場合
USB_DATA_OVR：	受信データがサイズオーバーした場合
USB_DATA_ERR：	無応答もしくはオーバ／アンダーランエラーを検出した場合
USB_DATA_STALL：	STALL 応答もしくは MaxPacketSize エラーを検出した場合
USB_DATA_STOP：	データ転送を強制終了した場合
USB_DATA_TMO※：	タイムアウト発生による強制終了は CALLBACK しません

13.1.3 データ転送時の注意事項

等時性データ転送ではインターバル時間が守られるようにサブミット関数を実行してください。

13.1.4 データ受信時の注意事項

ショートパケットを受信した場合は、残り受信予定のデータ長を tranlen に格納してトランスファー終了します。

13.2 USB 通信用構造体（USB_UTR_t 構造体）

R_usb_hstd_TransferStart()関数の引数となる構造体について説明します。

USB 通信は、以下の構造体を HCD に通知することで接続されたペリフェラルデバイスと通信が可能です。

```
struct USB_SUTR   {
    USB_MH_t        msghead;  // OS が使用するメッセージヘッダ
    uint16_t        msginfo;  // USB-BASIC-F/Wが使用するメッセージ情報。
    uint16_t        keyword;  // サブ情報（ポート番号、パイプ番号等）
    void            *tranadr; // 転送データの先頭アドレス
    uint32_t        tranlen;  // 転送データ長
    uint16_t        *setup;   // Setup packet (only host control transfer)
    uint16_t        status;   // 転送終了ステータス
    uint16_t        pipectr;  // パイプ制御レジスタの状態
    USB_CB_t        complete; // 処理終了時のコールバック関数
    uint8_t         errcnt;   // エラーカウント
    uint8_t         segment;  // セグメントコード
}USB_UTR_t;
```

表13.1　USB_UTR_t 構造体のメンバ

変数名	Read/ Write	機能
msghead	---	OS が使用するメッセージヘッダです。使用しないでください。
msginfo	R	メッセージ種別 リクエスト内容を設定します。 API 関数を使用することでUSB-BASIC-F/Wが設定します。 USB 通信を行う場合は USB_MSG_HCD_SUBMITUTR を設定します。
keyword	R	サブコード USB 通信を行う場合はパイプ番号を指定します。
*tranadr	R	USB 通信バッファアドレス USB 通信バッファアドレスを通知します。
tranlen	R/W	USB 通信データ長 USB 通信データ長を通知します。 ユーザバッファサイズ未満のトランスファーサイズを指定します。
*setup	R	Setup パケットデータ HCD に Control 転送時のセットアップパケットのリクエストとデバイスアドレスを通知します。
status	W	USB 通信ステータス HCD が USB 通信結果を応答します。
pipectr	---	EHCI 対応USB-BASIC-F/Wでは使用しないため設定値は無視されます。
complete	R	Callback 関数 USB 通信を終了した場合に実行したい関数アドレスを指定します。 コールバック関数は下記で型宣言してください。 typedef void (*USB_CB_t)(USB_UTR_t*);
errcnt	---	EHCI 対応USB-BASIC-F/Wでは使用しないため設定値は無視されます。

変数名	Read/ Write	機能
segment	---	EHCI 対応USB-BASIC-F/Wでは使用しないため設定値は無視されます。

(1) USB 通信バッファアドレス

受信または ControlRead 転送： 受信データを格納するバッファのアドレスを指定します。

送信または ControlWrite 転送： 送信データが格納してあるバッファアドレスを指定します。

NoDataControl 転送： 指定されても無視します。

＊ USB Basic F/Wでは本アドレスで指定されたバッファを直接転送メモリとして使用しますので転送が完了するまで必要な領域を確保して、その領域へリード/ライトアクセスしないようにしてください。

(2) USB 通信データ長

受信または ControlRead 転送： 受信データ長を格納します。

送信または ControlWrite 転送： 送信データ長を格納します。

NoDataControl 転送： "0"を指定してください。

USB 通信終了後に送受信の残りデータ長が格納されます。ホスト機能でコントロール転送を行った場合はデータステージに対する残りデータ長が格納されます。

＊アイソクロナス転送を行う場合は必ずUSB通信データ長をマックスパケットサイズより小さくなるように設定してください。マックスパケットサイズより大きいサイズを設定した場合、動作の保障はできません。

(3) Setup パケットデータ

R_usb_hstd_TransferStart()関数でコントロール転送を行う場合の、構造体メンバ（*setup）は下記の USB リクエストのデータテーブルです。

*setup には、uint16_t[5]の配列のテーブルアドレスを指定してください。

表13.2 setup_packet の配列

値	
bRequest	bmRequestType
wValue	
wIndex	
wLength	
Device Address	

(4) USB 通信ステータス

下記のステータス情報を返します。

USB_CTRL_END：	Control 転送が正常に終了した場合
USB_DATA_NONE：	データ送信が正常終了した場合
USB_DATA_OK：	データ受信が正常終了した場合
USB_DATA_SHT：	データ受信は正常終了したが指定されたデータ長未満で終了した場合
USB_DATA_OVR：	受信データがサイズオーバーした場合
USB_DATA_ERR：	無応答もしくはオーバ／アンダーランエラーを検出した場合
USB_DATA_STALL：	STALL 応答もしくは MaxPacketSize エラーを検出した場合
USB_DATA_STOP：	データ転送を強制終了した場合
USB_DATA_TMO※：	タイムアウト発生による強制終了は CALLBACK しません

(5) PIPECTR レジスタ

EHCI対応USB-BASIC-F/Wでは使用しないため設定値は無視されます。

(6) セグメント情報

EHCI対応USB-BASIC-F/Wでは使用しないため設定値は無視されます。

13.3 パイプ定義

13.3.1 概要

HDCD は当該クラスドライバに適合したパイプ設定を、パイプ情報テーブルとして保持しなければなりません。HDCD はパイプ情報テーブルもドライバ登録時に登録する必要があります。

＊ EHCI対応USB-BASIC-F/WはR8A6659x向けUSB-BASIC-F/WとI/F互換を取るため、パイプ情報テーブルを使用しますが、パイプ情報テーブル内のR8A6659x H/W依存設定（例えばFIFOのバッファサイズ等）は使用しません。使用しない設定値は無視されます。ご注意下さい。

13.3.2 パイプ情報テーブル

パイプ情報テーブルは、以下 6 項目（uint16_t×6）で構成されます。

1. パイプウインドウ選択レジスタ
2. パイプコンフィグレーションレジスタ（EHCI 対応USB-BASIC-F/Wでは無視されまる設定が含まれます）
3. パイプバッファ指定レジスタ（EHCI 対応USB-BASIC-F/Wでは設定値を無視されます）
4. パイプマックスパケットサイズレジスタ
5. パイプ周期制御レジスタ（EHCI 対応USB-BASIC-F/Wでは設定値を無視されます）
6. FIFO ポート使用方法（EHCI 対応USB-BASIC-F/Wでは設定値を無視されます）

13.3.3 パイプ定義

USB-BASIC-F/Wでサンプルとして記述されているパイプ定義は以下の構成になっています。

情報テーブルに記載が可能な各定義項目は r_usb_cDefUSBIP.h でマクロ定義しています。

例）

```
uint16_t usb_gpstd_SmplEpTbl1 [] = {                                    ← 登録された情報テーブル
    USB_PIPE1,                                                          ←パイプ定義項目 1
    USB_BULK|USB_BFREOFF|USB_DBLBOFF|USB_CNTMDOFF|USB_SHTNAKON|USB_DIR_P_OUT|USB_EP1,
                                                                        ←パイプ定義項目 2
    (uint16_t)USB_BUF_SIZE(512u) | USB_BUF_NUMB(8u),                    ←パイプ定義項目 3
    USB_SOFT_CHANGE,                                                    ←パイプ定義項目 4
    USB_IFISOFF | USB_IITV_TIME(0u),                                    ←パイプ定義項目 5
    USB_CUSE,                                                           ←パイプ定義項目 6
                    :
                    :
    USB_PDTBLEND
};
```

(1) パイプ定義項目 1

パイプウインドウ選択レジスタに設定する値を指定します。

パイプ選択　　：選択パイプ（PIPE1～PIPE30）を指定してください。

例）パイプ 1 のとき

```
USB_PIPE1,
```

(2) パイプ定義項目 2

パイプコンフィグレーション設定を指定します。
転送タイプ : USB_BULK、USB_INT、USB_ISO のいずれか一つを指定してください。
BRDY 動作指定 : EHCI 対応USB-BASIC-F/Wでは使用しないため設定値は無視されます。
ダブルバッファモード : EHCI 対応USB-BASIC-F/Wでは使用しないため設定値は無視されます。
連続送受信モード : EHCI 対応USB-BASIC-F/Wでは使用しないため設定値は無視されます。
SHTNAK 動作指定 : EHCI 対応USB-BASIC-F/Wでは使用しないため設定値は無視されます。
転送方向 : USB_DIR_H(P)_OUT/IN のいずれかを指定してください。
エンドポイント番号 : エンドポイント番号(EP1～EP15)を指定してください。

例）バルク転送、BFRE オフ、ダブルバッファ、連続送受信、OUT 方向、EP2 のとき
ホストの場合：
USB_BULK | USB_BFREOFF | USB_DBLBON | USB_CNTMDOFF | USB_SHTNAKOFF | USB_DIR_H_OUT | USB_EP2,

(3) パイプ定義項目 3

EHCI対応USB-BASIC-F/Wでは使用しないため設定値は無視されます。

(4) パイプ定義項目 4

パイプマックスパケットサイズレジスタとデバイスアドレスの設定を指定します。
デバイスアドレス ：デバイスアドレスを指定してください。
サンプルアプリケーションではソフトで変更する為、初期値に USB_NONE を設定しています。
※マックスパケットサイズ ：パイプのマックスパケットサイズを指定してください。
サンプルアプリケーションではソフトで変更する為、初期値に USB_NONE を設定しています。

例 1）マックスパケットサイズ 64、デバイスアドレス 3 のとき
DEV_ADDR(3)｜MAX_PACKET(64)

(5) パイプ定義項目 5

EHCI対応USB-BASIC-F/Wでは使用しないため設定値は無視されます。

(6) パイプ定義項目 6

EHCI 対応USB-BASIC-F/Wでは使用しないため設定値は無視されます。

13.3.4 パイプ定義設定上の制限事項

デバイスクラスでトランスファー単位の通信同期を取ってください。
同一パイプ番号で異なるパイプの同時設定はできません。（パイプ定義項目 1）
（R_usb_hstd_SetPipeRegistration()関数でその都度パイプ設定を変更してください。）

【注】 テーブルの最後には、必ず USB_PDTBLEND を書いてください。

13.3.5 基本機能

HDCD は、R_usb_hstd_DriverRegistration()関数でパイプ情報テーブルを登録します。

パイプ情報テーブルは接続されたデバイスに合わせて更新する必要があります。

デバイスから取得したディスクリプタ（EndpointDescriptor）を解析し、パイプ情報テーブルを変更してください。

(1) パイプ情報更新関数

EndpointDescriptor からパイプ情報テーブルのデータを更新する usb_hstd_ChkPipeInfo()関数と、更新した情報をパイプ情報テーブルに設定する、usb_hstd_SetPipeInfo()関数を準備しました。

転送タイプ	: USB_BULK、USB_INT、USB_ISO のいずれか一つが設定されます。
転送方向	: USB_DIR_H_OUT、USB_DIR_H_IN のどちらかが設定されます。
エンドポイント番号	: エンドポイント番号（EP1～EP15）が設定されます。
マックスパケットサイズ	: パイプのマックスパケットサイズが設定されます。
インターバル間隔	: インターバル値が設定されます。

13.4　データ送信動作

USB-BASIC-F/Wは、データ送信を以下の手順で行っています。

① HDCD が API 関数を使用してデータ転送要求を発行します。

② PIPE 番号を判定してデータ転送を開始します。

③ 送信するデータを H/W に設定し転送を開始します。

④ 転送が完了すると割り込みハンドラから HCD TRAN に割り込み発生を通知します。

⑤ 転送完了割り込み受信後、コールバック関数を呼び出します。

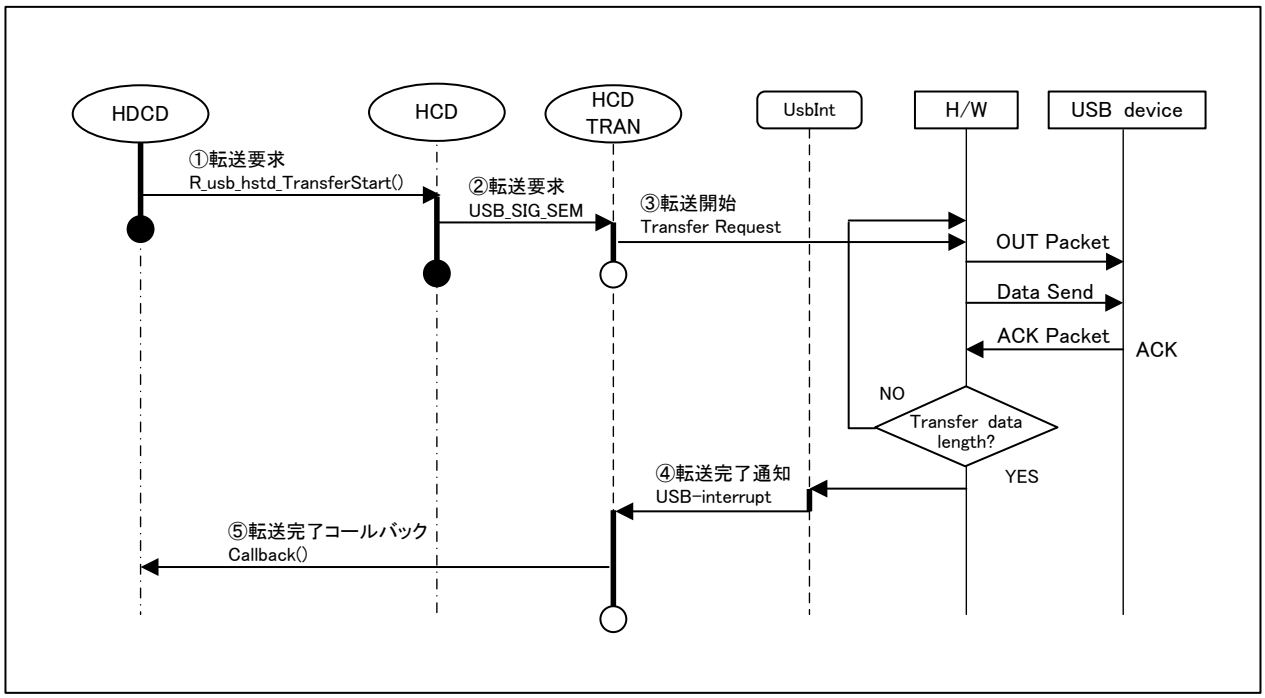


図13.1　データ送信概略フロー

13.5　データ受信動作

USB-BASIC-F/Wは、データ受信を以下の手順で行っています。

①-②データ送信と同様です。

③ 受信バッファを H/W に設定し受信開始

④ 転送が完了すると割り込みハンドラから HCDTRAN に割り込み発生を通知します。

⑤ 転送完了割り込み受信後、受信データをユーザーバッファにコピー

⑥ コールバック関数を呼び出します。

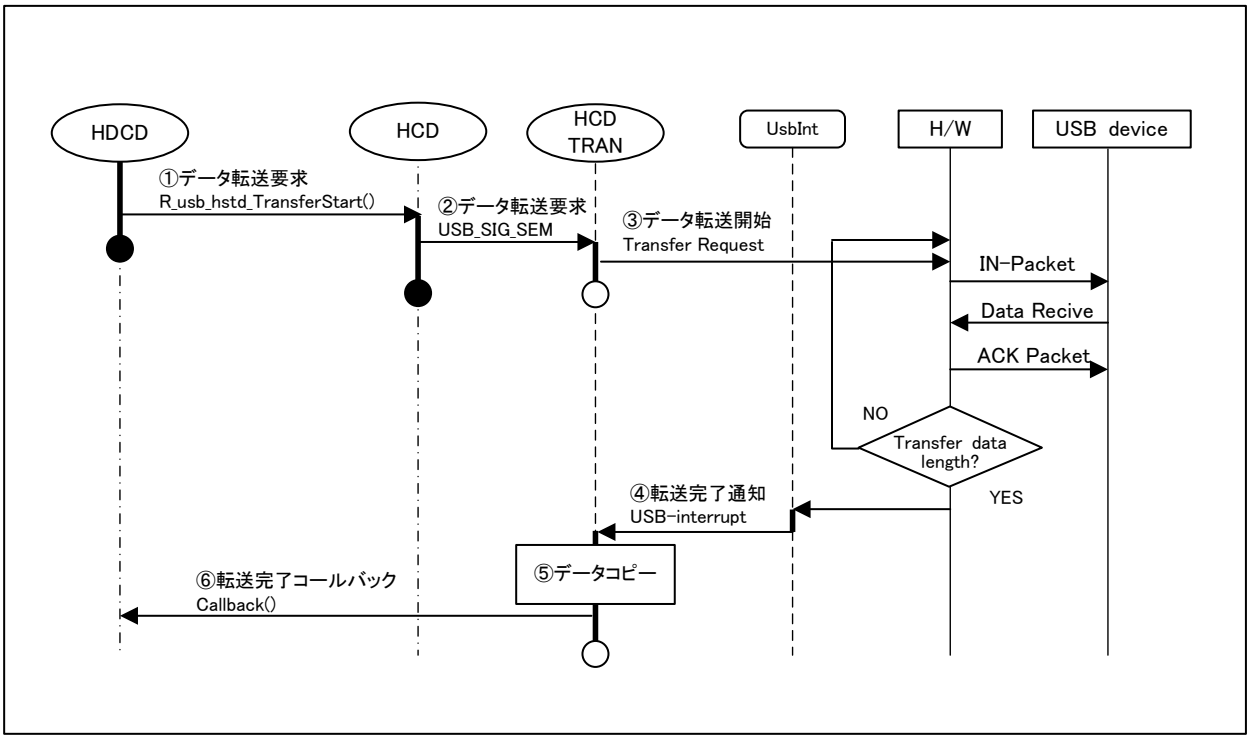


図13.2　データ受信概略フロー

14. 制限事項

EHCI 対応 USB2.0 ホストコントローラ IP 用USB-BASIC-F/Wには、以下の制限事項があります。

1. MCU が高負荷になるシステムでのアイソクロナス転送では、パケットを欠落する可能性があります。
   （アイソクロナス転送で送信するパケットを所定時間内に設定することが出来ない場合など）
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. HDCD はお客様にてご用意ください。
4. USB-BASIC-F/Wは接続した HUB および HUB ダウンポート接続デバイスのサスペンド／レジュームに対応しておりません。
5. HUB ダウンポートデバイスの最大数は 4 までです。それ以上のダウンポート対応はお客様にて、HUBCD をご変更ください。
6. EHCI 対応USB-BASIC-F/Wは R8A6659x 向けUSB-BASIC-F/Wと I/F 互換を取るため、パイプ情報テーブルを使用しますが、パイプ情報テーブル内の R8A6659x H/W 依存設定（例えば FIFO のバッファサイズ等）は使用しません。
7. 制限事項 6 の影響により、パイプ情報テーブルの仕様制限を受けるため、最大接続デバイス数と最大パイプ数が以下の制限を受けることになります。ご注意ください。
   ・最大接続デバイス数：　14 デバイス
   ・最大パイプ数：　30 パイプ
8. アイソクロナス転送は一回の転送要求（R_usb_hstd_TransferStart()）にマックスパケットサイズを超える転送データには対応しておりません。
9. High-Speed アイソクロナス転送は High-Band-Width 仕様には対応しておりません。
10. HDCD の AUDIO クラスドライバ実装にてアイソクロナス OUT 転送機能をご用意される場合、本 USB-BASIC-F/Wは対向オーディオデバイスのオーディオクロックと同期を取る機能は実装しておりません。そのため、オーディオデバイスとのアイソクロナス OUT 転送においてノイズ等の現象が発生する可能性があります。オーディオデバイスとのオーディオクロック同期につきましてはお客様のシステムにて実装してください。
11. EHCI 対応USB-BASIC-F/Wは SOF 期間を調整する機能は対応しておりません。
12. EHCI 対応USB-BASIC-F/Wは R_usb_hstd_TransferStart()関数で指定された USB 通信バッファのメモリを H/W が直接転送メモリとして使用します。一旦 H/W が転送を開始したら、上位 S/W は転送終了まで USB 通信バッファのメモリにアクセスしないでください。また、受信転送においては、USB 通信バッファの外側においても、その両端付近でライトアクセスすることにより USB 通信バッファの両端をキャッシュメモリにロードするようなアクセスは行わないでください。キャッシュメモリと実メモリとの整合性が取れなくなります。この問題を避けるため、以下の手法でご対応願います。
    ・転送中は USB 通信バッファ付近のメモリにライトアクセスしない
    ・USB 転送バッファの両端をキャッシュメモリ境界に揃える
    ・非キャッシュ領域の USB 通信バッファを使用する

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	2011.03.15	—	初版発行
1.01	2011.03.30	P19,20	メールボックス USB_HUBC_MBX を追加 セマフォ USB_HCI_DC_SEM、USB_HCI_TRCC_S_SEM、USB_HCI_TRCC_T_SEM を追加
		P24	図 7.2 修正
		P38	10.3.2 節のタイトル追加
		P48	図 11.4 修正
		P55	図 12.1 修正
1.02	2011.09.06	P2	関連ドキュメント 2. のドキュメント名を修正
		P13-P14	デバイスアドレス、ＥＨＣＩタイムアウト指定の例)をソースのデフォルト値に合せ修正
		P15	5.5 OHCI ユーザー定義ファイルの 7.の項目を修正
		P16	5.5.7 のタイトル、内容修正
		P21	「6.4 セクション」追加
		P39	「11.5 HDCD の確認及び起動」の内容変更
		P45-P46	表 11.9,表 11.10,表 11.11 のタイトル変更
		P59	「(1)通信バッファアドレス」にバッファに関する注意書きを追加
1.03	2012.12.07	P17	6.2.1 フォルダ構成 smpl→SMPL(名称変更) Project フォルダを削除
		P18	6.2.2 ファイル一覧 “r_usb_cKernelId.h”ファイルを追加 “SMPL”と”hubd”フォルダのパスを修正
		P61	13.2 USB 通信用構造体（USB_UTR_t 構造体） 「(1)通信バッファアドレス」にバッファに関する注意書きを修正
		P21	6.4 セクション アイソクロナス転送用ディスクリプタの配置に対する制限を追記
		P1-P69	アプリケーションノート番号変更

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下さい。

   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・電話番号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 エレクトロニクス販売株式会社 〒 100-0004 千代田区大手町 2-6-2（日本ビル） (03)5201-5307

■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/