RZ/A2M グループ

USB Host Human Interface Device Class Driver (HHID)

要旨

本アプリケーションノートでは、Host 用ヒューマンインターフェースデバイス（HHID）クラスドライバについて説明します。本モジュールは USB Basic Firmware（USB-BASIC-FW）と組み合わせることで動作します。以降、本モジュールを USB HHID モジュールと称します。

動作確認デバイス

RZ/A2M グループ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

目次

1.　概要

USB HHID モジュールは、USB-BASIC-FW モジュールと組み合わせることで、USB Host ヒューマンインターフェースデバイスクラスドライバ（以降 HHID と記述）として動作します。

以下に、本モジュールがサポートしている機能を示します。

- 接続されたHIDデバイス（USBマウス、USBキーボード）とデータ通信が可能
- 接続されたHIDデバイスに対し、HIDクラスリクエストを発行する
- Interrupt OUT転送をサポート
- 一つの USB モジュールに対し USB Hub を使って最大 4 つの HID デバイスの接続が可能。

API などの名称は、ポート 0 とポート 1 で異なります。

本書では、API 名称などはポート 0 のものを例として記載します。

表 1.1　使用する周辺機能と用途

周辺機能	用途
ホスト PC	サンプルコードのメッセージ出力

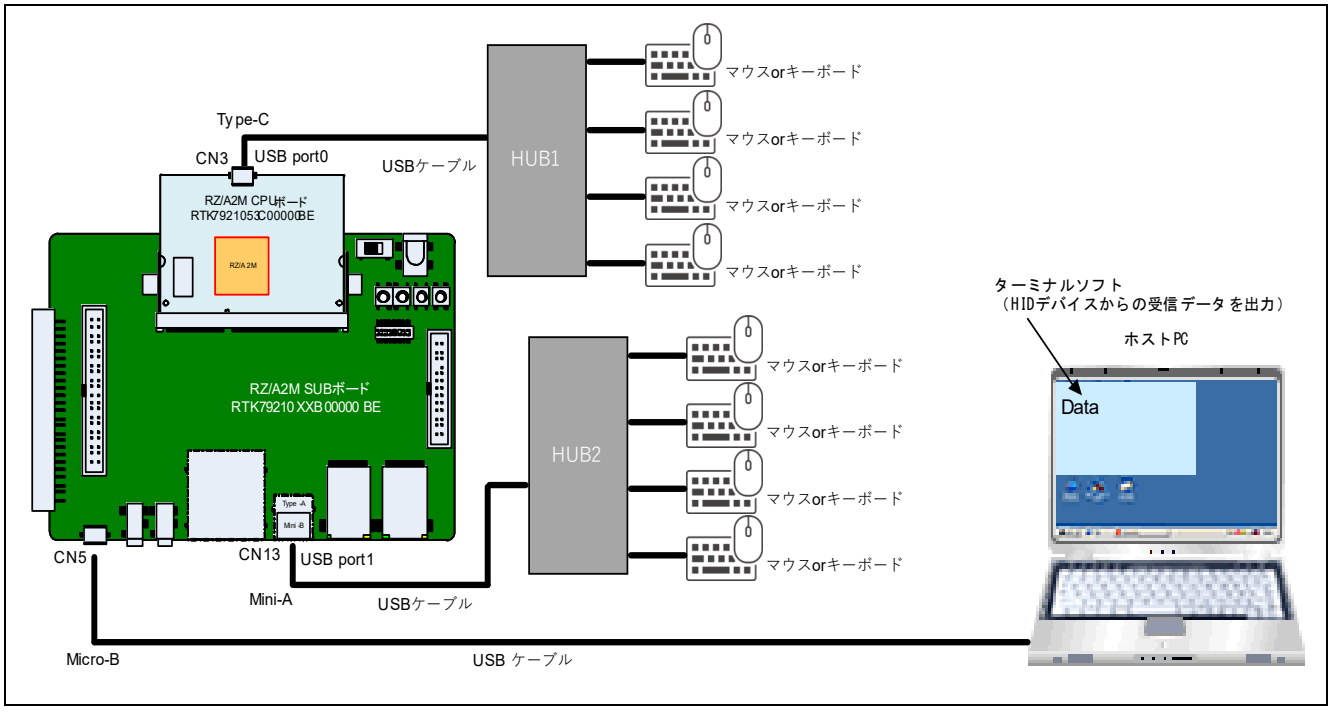


図 1.1　動作環境

1.1 制限事項

本モジュールには以下の制限事項があります。

1. HHID はレポートディスクリプタ解析を行っておりません。デバイスから取得したインターフェースプロトコル（Keyboard/Mouse）からレポートフォーマットを決定し、処理を行っています。
2. 本ドライバは、一つの USB モジュールに対し USB Hub を使って最大 4 つの HID デバイスの接続が可能です。

1.2 注意事項

本ドライバは、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。

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。

略称	詳細
APL	Application program
HCD	Host Control Driver for USB-BASIC-FW
HDCD	Host Device Class Driver (Device Driver and USB Class driver)
HHID	USB Host Human Interface Device Class Driver
HID	Human Interface Device Class
HUBCD	Hub Class Driver
MGR	Peripheral Device State Manager for HCD
Non-OS	USB Driver for OS-less
RTOS	USB Driver for the real-time OS
USB	Universal Serial Bus
USB-BASIC-FW	USB Basic Host and Peripheral Driver

2. 動作確認条件

本アプリケーションノートのサンプルコードは、下記の条件で動作を確認しています。

表 2.1　動作確認条件

項目	内容
使用マイコン	RZ/A2M
動作周波数（注）	CPU クロック（Iφ）：528MHz 画像処理クロック（Gφ）：264MHz 内部バスクロック（Bφ）：132MHz 周辺クロック 1（P1φ）：66MHz 周辺クロック 0（P0φ）：33MHz QSPI0_SPCLK：66MHz CKIO：132MHz
動作電圧	電源電圧（I/O）：3.3V 電源電圧（1.8/3.3V 切替 I/O（PVcc_SPI））：3.3V 電源電圧（内部）：1.2V
統合開発環境	e2 studio V7.8.0
C コンパイラ	GNU Arm Embedded Toolchain 6.3.1
	コンパイラオプション（ディレクトリパスの追加は除く） Release: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Os -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -Wstack-usage=100 -fabi-version=0 Hardware Debug: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Og -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -g3 -Wstack-usage=100 -fabi-version=0
動作モード	ブートモード 3 （シリアルフラッシュブート 3.3V 品）
ターミナルソフトの通信設定	• 通信速度：115200bps • データ長：8 ビット • パリティ：なし • ストップビット長：1 ビット • フロー制御：なし
使用ボード	RZ/A2M CPU ボード　RTK7921053C00000BE RZ/A2M SUB ボード　RTK79210XXB00000BE
使用デバイス （ボード上で使用する機能）	• シリアルフラッシュメモリ（SPI マルチ I/O バス空間に接続） メーカ名：Macronix 社、型名：MX25L51245GXD • RL78/G1C（USB 通信とシリアル通信を変換し、ホスト PC との通信に使用） • LED1

【注】 クロックモード 1（EXTAL 端子からの 24MHz のクロック入力）で使用時の動作周波数です。

3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

・USB Basic Host Driver アプリケーションノート(Document No. R01AN4715JJ0140)

4. ソフトウェア構成

USB HHID モジュールは 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。 接続された USB デバイスからデータを受け取ると、HCD を介して APL に通知します。又、APL から要求があった場合、HCD を介して USB デバイスに通知します。

図 4.1 に、HHID のモジュール構成、表 4.1 にモジュール機能概要を示します。

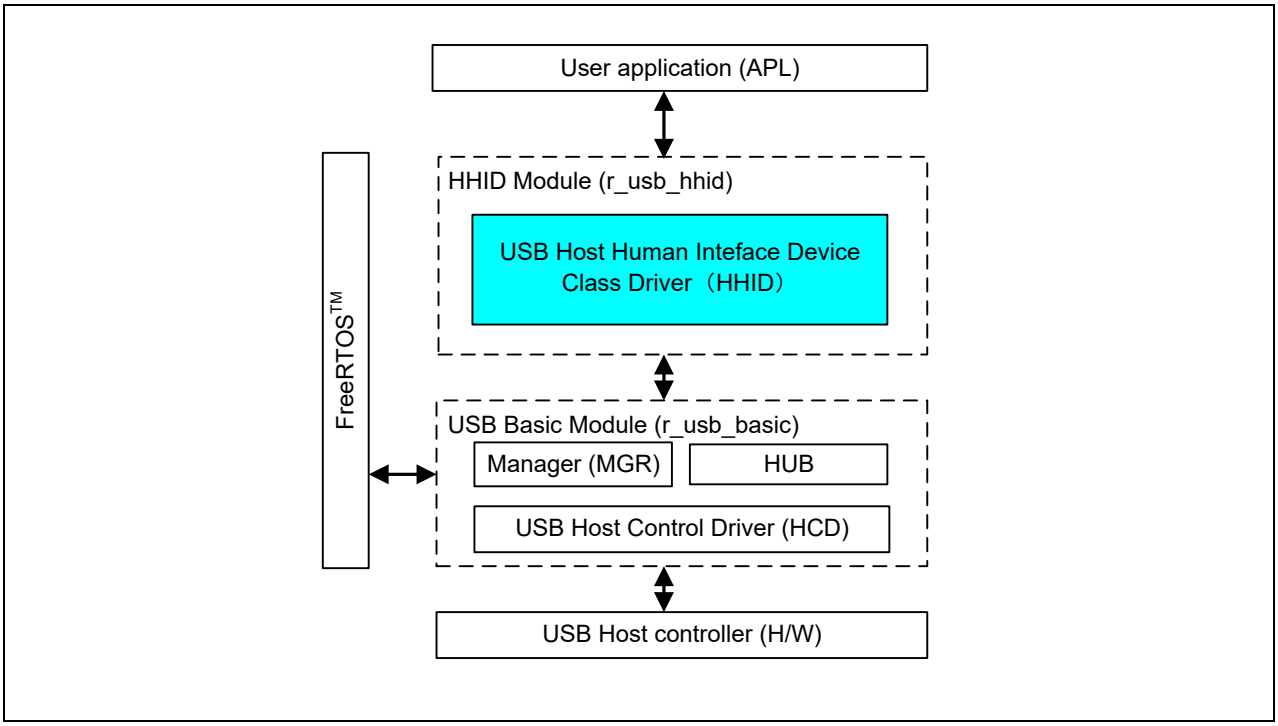


図 4.1 モジュール構成図

表 4.1 モジュール機能概要

モジュール名	機能概要
APL	サンプルアプリケーションプログラム ・ HID デバイスとの通信、データ読み捨て ・ HID デバイスから受信したレポート情報を PC（ターミナル）に表示する
HHID	USB デバイスからの要求を解析します。 HCD を介して、APL の SW 操作情報を USB デバイスに通知します。
HCD/MGR	USB Host H/W 制御ドライバです。
FreeRTOS™	FreeRTOS™

5. API 情報

本ドライバの API はルネサスの API の命名基準に従っています。

5.1 ハードウェアの要求

ご使用になる MCU が以下の機能をサポートしている必要があります。

- USB

5.2 ソフトウェアの要求

このドライバは以下のパッケージに依存しています。

- r_usb_basic

5.3 使用する割り込みベクタ

このドライバが使用する割り込みベクタを以下に示します。

表 5.1 使用する割り込みベクター覧

ポート	割り込みベクタ
0	USBHI0 割り込み(ベクタ番号: 63)
1	USBHI1 割り込み(ベクタ番号: 68)

5.4 ヘッダファイル

すべての API 呼び出しとそれをサポートするインタフェース定義は r_usb_basic_if.h と r_usb_hhid_if.h に記載されています。

5.5 整数型

このプロジェクトは ANSI C99 を使用しています。これらの型は stdint.h で定義されています。

5.6 コンパイル時の設定

コンパイル時の設定については、「9. コンフィグレーション (r_usb_hhid_config.h)」章および USB Basic Host Driver アプリケーションノート(ドキュメント No. R01AN4715JJ0140)の「コンフィグレーション」章を参照してください。

5.7 引数

API 関数の引数に使用される構造体については、USB Basic Host Driver アプリケーションノート(ドキュメント No. R01AN4715JJ0140)内の「構造体定義」の章を参照してください。

6. ターゲットペリフェラルリスト（TPL）

TPL については、USB Basic Host Driver アプリケーションノート(ドキュメント No. R01AN4715JJ0140)内の「ターゲットペリフェラルリスト(TPL)」の章を参照してください。

7. ヒューマンインターフェースデバイスクラス（HID）

7.1 基本機能

本ドライバは、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。

本ドライバの主な機能を以下に示します。

1. HID デバイスの照合
2. HID デバイスへのクラスリクエスト通知
3. HID デバイスとのデータ通信

7.2 クラスリクエスト

本ドライバがサポートしているクラスリクエストを表 5.1 に示します。

アプリケーションプログラムでのクラスリクエスト処理については、USB Basic Host Driver アプリケーションノート(ドキュメント No. R01AN4715JJ0140)内の「コントロール転送」の章を参照してください。

表 7.1 HID クラスリクエスト

対応記号表	リクエスト	コード	説明
a	USB_GET_REPORT	0x01	USB デバイスにレポートを要求する
b	USB_SET_REPORT	0x09	USB デバイスにレポートを通知する
c	USB_GET_IDLE	0x02	USB デバイスに Duration 時間を要求する
d	USB_SET_IDLE	0x0A	USB デバイスに Duration 時間を通知する
e	USB_GET_PROTOCOL	0x03	USB デバイスにプロトコルを要求する
f	USB_SET_PROTOCOL	0x0B	USB デバイスにプロトコルを通知する
	USB_GET_REPORT_DESCRIPTOR	Standard	レポートディスクリプタを要求する
	USB_GET_HID_DESCRIPTOR	Standard	HID ディスクリプタを要求する

本ドライバが対応するクラスリクエストのデータフォーマットを以下に記します。

7.2.1 GetReport リクエストフォーマット

表 7.2 に、GetReport リクエストのフォーマットを以下に示します。

コントロール転送によりデバイスからレポートデータを受信します。

表 7.2 GetReport フォーマット

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0xA1	GET_REPORT（0x01）	ReportType & ReportID	Interface	ReportLength	Report

7.2.2 SetReport リクエストフォーマット

表 7.3 に、SetReport リクエストのフォーマットを示します。

コントロール転送によりレポートデータをデバイスに送信します。

表 7.3 SetReport フォーマット

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	SET_REPORT (0x09)	ReportType & ReportID	Interface	ReportLength	Report

7.2.3 GetIdle リクエストフォーマット

表 7.4 に、GetIdle リクエストのフォーマットを示します。

レポート通知（インタラプト転送）の間隔時間を取得します。Idle rate は 4msec 単位です。

表 7.4 GetIdle フォーマット

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0xA1	GET_IDLE (0x02)	0 & ReportID	Interface	1	Idle rate

7.2.4 SetIdle リクエストフォーマット

表 7.5 に、SetIdle リクエストのフォーマットを示します。

レポート通知（インタラプト転送）の間隔時間を設定します。Duration は 4msec 単位です。

表 7.5 SetIdle フォーマット

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	SET_IDLE (0x0A)	Duration & ReportID	Interface	0	Not applicable

7.2.5 GetProtocol リクエストフォーマット

表 7.6 に、GetProtocol リクエストのフォーマットを示します。

現在設定されているプロトコル（ブートプロトコル又はレポートプロトコル）を取得します。

表 7.6 GetProtocol フォーマット

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0xA1	GET_PROTOCOL (0x03)	0	Interface	1	0(BootProtocol) / 1(ReportProtocol)

7.2.6 SetProtocol リクエストフォーマット

表 7.7 に、SetProtocol リクエストのフォーマットを示します。

プロトコル（ブートプロトコル又はレポートプロトコル）の設定を行います。

表 7.7 SetProtocol フォーマット

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0x21	SET_PROTOCOL（0x0B）	0(BootProtocol) / 1(ReportProtocol)	Interface	0	Not applicable

7.3 レポートフォーマット

HID で扱うレポートフォーマット例を以下に記します。

7.3.1 受信レポートフォーマット

表 7.8 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例を示します。

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。

表 7.8 受信レポートフォーマット例

offset (データ長)	Keyboard モード (8 バイト)	Mouse モード (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	―

7.3.2 送信レポートフォーマット

表 7.9 に、HID デバイスに通知する送信レポートフォーマット例を示します。

クラスリクエスト SetReport で送信を行います。

表 7.9 送信レポートフォーマット例

offset (データ長)	Keyboard モード (1 バイト)	Mouse モード (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	―	―

7.3.3 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。

8. API

Host Human Interface Device Class 固有の API を以下に示します。

API	ポート	説明
R_USBH0_HhidGetType() R_USBH1_HhidGetType()	0 1	接続された HID デバイスのタイプ情報取得
R_USBH0_HhidGetMxps() R_USBH1_HhidGetMxps()	0 1	接続された HID デバイスのマックスパケットサイズを取得
R_USBH0_HhidTask() R_USBH1_HhidTask()	0 1	接続された HID デバイスのストリングディスクリプタを取得(String lang IDs / String iProduct)

[Note]

その他の API については、USB Basic Host Driver アプリケーションノート(ドキュメント No.R01AN4715JJ0140)内の「API 情報」の章を参照してください。

8.1 R_USBH0_HhidGetType

接続された HID デバイスのタイプ情報を取得する

形式

usbh0_err_t R_USBH0_HhidGetType(uint8_t address, uint8_t *p_type)

引数

address	デバイスアドレス
p_type	タイプ情報を格納する領域へのポインタ

戻り値

USBH0_SUCCESS	正常終了 (タイプ情報取得完了)
USBH0_ERR_PARA	パラメータエラー
USBH0_ERR_NG	その他のエラー

解説

address に指定された情報をもとに、接続された HID デバイスのタイプ情報(マウス、キーボード、その他)を取得します。タイプ情報は、第 2 引数(p_type)が示す領域にセットされます。セットされるタイプ情報については、表 8.1 を参照してください。

表 8.1 タイプ情報

タイプ情報	内容
USBH0_HID_KEYBOARD	キーボード
USBH0_HID_MOUSE	マウス
USBH0_HID_OTHER	キーボード、マウス以外の HID デバイス

リエントラント

本 API は再入可能（リエントラント）です。

補足

1. 本 API をコールする前に address に対し、タイプ情報を取得したい HID デバイスのデバイスアドレスを指定してください。なお、address に対する指定に問題がある場合は、戻り値に USBH0_ERR_PARA が返されます。
2. 引数 p_type に対し USBH0_NULL を指定した場合は、戻り値に USBH0_ERR_PARA が返されます。
3. USB デバイスが CONFIGURED 状態の場合に、本 API をコールすることができます。CONFIGURED 以外の状態で本 API をコールすると戻り値に USBH0_ERR_NG が返されます。

使用例

```
void usb_application( void )
{
  switch ( usbh0_hid_event_get(g_usbh0_hhid_devaddr) )
  {
        :
    case USBH0_STS_CONFIGURED:
        :
        R_USBH0_HhidGetType( g_usbh0_hhid_devaddr, &type );
        if( USBH0_HID_KEYBOARD == type )
        {
            :
        }
            :
    break;
        :
  }
}
```

8.2 R_USBH0_HhidGetMxps

接続された HID デバイスのマックスパケットサイズを取得する

形式

uint16_t R_USBH0_HhidGetMxps(uint16_t pipe_id)

引数

pipe_id PIPE 番号

戻り値

Max Packet Size

解説

pipe_id をもとに、接続された HID デバイスのマックスパケットサイズを取得します。

リエントラント

本 API は再入可能（リエントラント）です。

補足

1. 本 API をコールする前に pipe_id に対し、PIPE 番号を指定してください。未使用 PIPE の番号を指定した場合は、戻り値に 0 が返されます。

使用例

```
void usb_application( void )
{
  switch ( usbh0_hid_event_get(g_usbh0_hhid_devaddr) )
  {
       :
    case USBH0_STS_CONFIGURED:
       :
       pipe_id = R_USBH0_HstdGetPipeID(
                    g_usbh0_hhid_devaddr, USBH0_EP_INT, USBH0_EP_IN, 0xFF);
       R_USBH0_HhidGetMxps( pipe_id );
       if( USBH0_HID_KEYBOARD == type )
       {
           :
       }
           :
    break;
       :
  }
}
```

8.3 R_USBH0_HhidTask

接続された HID デバイスのストリングディスクリプタ(String lang IDs / String iProduct)を取得します。

形式

void	R_USBH0_HhidTask(void)

引数

―	―

戻り値

―	―

解説

接続された HID デバイスの String lang IDs を取得します。
続けて、String iProduct を取得します。

補足

1. スケジューラ処理を行うループ内で呼び出してください。

使用例

```
void usbh_main(void)
{
    while(1)
    {
        /* Scheduler */
        R_USBH0_CstdScheduler();
        if( USBH0_FLGSET == R_USBH0_CstdCheckSchedule() )
        {
            R_USBH0_HstdMgrTask();          /* MGR task */
            R_USBH0_HhubTask();             /* HUB task */
            R_USBH0_HhidTask();             /* HHID Task */
        }
        usbh0_hid_main();
    }
} /* End of function usbh_main() */
```

9. サンプルアプリケーション

9.1 アプリケーション仕様

以下のアプリケーションプログラムが用意されています。

9.1.1 Normal mode アプリケーション (r_usbh0_hhid_main_normal.c)

ボードに接続された HID デバイス(マウス/キーボード)とのデータ転送を行います。HID デバイスから受信したデータは読み捨てられます。

[Note]

一つの USB モジュールにつき、USB Hub を使用した場合で最大 4 つの HID デバイスを接続することができます。

9.1.2 Demo mode アプリケーション (r_usbh0_hhid_main_demo.c)

ボードに接続された HID デバイス(マウス/キーボード)とのデータ転送を行います。HID デバイスからの受信データは、ホスト PC のターミナルソフト上に表示されます。

[Note]

一つの USB モジュールにつき、USB Hub を使用した場合で最大 4 つの HID デバイスを接続することができます。

9.2 アプリケーション処理概要

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

9.2.1 初期設定

初期設定では、MCU の端子設定、USB ドライバの設定、USB コントローラの初期設定を行います。

9.2.2 ユーザ定義関数（r_usbh0/1_pa_to_va）設定

本ドライバでは下記の二つの関数をアプリケーションに実装する必要があります。

```
uint32_t r_usbh0_pa_to_va(uint32_t paddr);
uint32_t r_usbh1_pa_to_va(uint32_t paddr);
```

サンプルアプリケーションでは下記のファイルに定義されています。

rza2m_usbh_hid_sample_freertos_gcc¥src¥renesas¥application¥r_usb_mmu_pa_to_va.c

r_usb_mmu_pa_to_va.c に定義されている下記の 4 つのマクロの値を、使用環境に合わせて設定してください。設定値は Smart Configurator の MMU タブから確認できます。

属性：Normal(Non-cacheable)の Internal RAM Area の持つ計 4MB の領域を 1MB ずつ下記のマクロに割り当ててください。

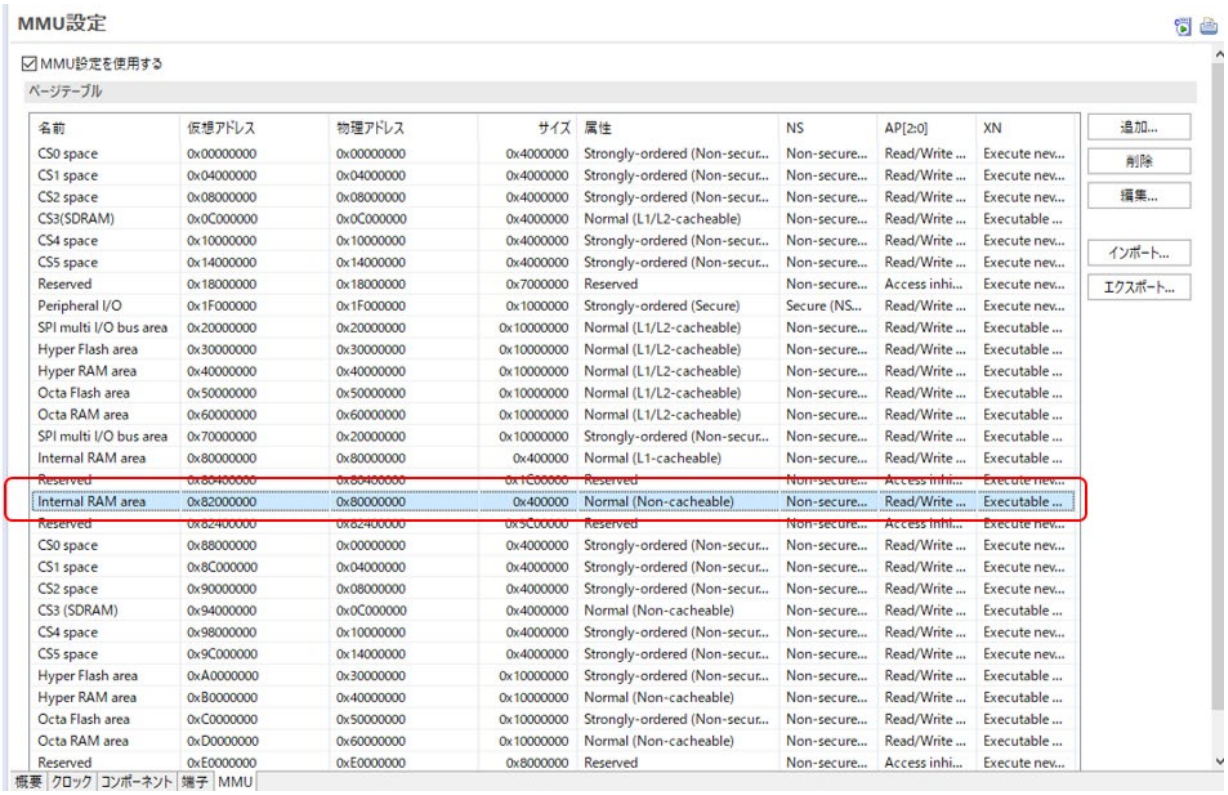
※MMU の初期設定が変更されていなければ、マクロの設定もデフォルトで使用可能です。

●設定例（初期設定）

下記のマクロにはそれぞれ、物理アドレス 0x80000000～0x80300000 に対応する仮想アドレスを 1MB ずつ設定してください。（下図　赤枠内参照）

連続している場合でも、必ず 1MB ずつ 4 つのマクロを定義する必要があります。

```
#define  USB_MMU_VIRTUAL_PAGE_0  (0x82000000)  //Virtual address corresponding to
                                                 0x80000000
#define  USB_MMU_VIRTUAL_PAGE_1  (0x82100000)  //Virtual address corresponding to
                                                 0x80100000
#define  USB_MMU_VIRTUAL_PAGE_2  (0x82200000)  //Virtual address corresponding to
                                                 0x80200000
#define  USB_MMU_VIRTUAL_PAGE_3  (0x82300000)  //Virtual address corresponding to
                                                 0x80300000
```



9.2.3 メインループ (r_usb_hhid_apl.c)

USB ドライバは初期設定後アプリケーションのメインルーチン内でスケジューラ（R_USBH0_CstdScheduler()）を呼び出すことで動作します。

メインルーチン内で R_USBH0_CstdScheduler()を呼ぶことでイベントの有無を確認し、イベントがある場合、スケジューラにイベントが発生していることを通知するためのフラグをセットします。

R_USBH0_CstdScheduler()呼び出し後、R_USBH0_CstdCheckSchedule()を呼び出しイベントの有無を確認してください。また、イベントの取得とそのイベントに対する処理は定期的に行う必要があります。（注 1）

Non OS メインループで呼び出している 3 つの Non OS 関数を OS タスクとして使用するため、BSP_CFG_RTOS_USED = 1 として、R_USBH0_Init()を呼び出してください。

そのため、BSP_CFG_RTOS_USED = 1 のときは、メインループで 3 つの Non OS 関数の呼び出しは不要です。

スケジューラ R_USBH0_CstdScheduler()は、Non OS、OS に関わらず各タスク関数にメッセージ送信し、タスク処理を制御します。

```
void usbh_main(void)
{
    while(1)
    {
        /* Scheduler */
        R_USBH0_CstdScheduler();
  #if(BSP_CFG_RTOS_USED == 0)
        if( USBH0_FLGSET == R_USBH0_CstdCheckSchedule() )
        {
            R_USBH0_HstdMgrTask();          /* MGR task */
            R_USBH0_HhubTask();             /* HUB task  (注 3) */
            R_USBH0_HhidTask();             /* HHID Task */
        }
  #endif /* (BSP_CFG_RTOS_USED == 0) */
        usbh0_hid_main();
    }
} /* End of function usbh_main() */
```

(注 2)

（注1） R_USBH0_CstdScheduler()でイベントを取得後、処理を行う前に再度 R_USBH0_CstdScheduler()で他のイベントを取得すると、最初のイベントは破棄されます。イベント取得後は必ず各タスクを呼び出し、処理を行ってください。

（注2） これらの処理は、アプリケーションプログラムのメインループ内に必ず記述してください。

（注3） HUB を使用する場合のみ、本関数を呼び出す必要があります。

9.2.4 アプリケーション処理

アプリケーションでは、HID デバイスからのデータ受信処理をメインに行います。以下にメインループの処理概要を示します。

1. USB Host に HID デバイスが ATTACH され、Enumeration 完了後に usbh0_hid_event_get 関数をコールすると戻り値に USBH0_STS_CONFIGURED がセットされます。APL では、USBH0_STS_CONFIGURED を確認するとクラスリクエスト SET_PROTOCOL を HID デバイスに送信します。
2. HID デバイスに対するクラスリクエスト SET_PROTOCOL の送信が完了し、usbh0_hid_event_get 関数をコールすると戻り値に USBH0_STS_REQUEST_COMPLETE がセットされます。APL は、USBH0_STS_REQUEST_COMPLETE を確認すると、usbh0_data_read 関数をコールし、HID デバイスからの送信されるデータのデータ受信要求を行います。
3. HID デバイスからのデータ受信が完了し、usbh0_hid_event_get 関数をコールすると戻り値に USBH0_STS_READ_COMPLETE がセットされます。APL では、USBH0_STS_READ_COMPLETE を確認すると usbh0_data_read 関数をコールし、HID デバイスからの送信されるデータのデータ受信要求を行います。
4. 上記 3 の処理が繰り返し行われます。

表 9.1 にステート一覧を示します。

表 9.1 ステート一覧

ステート	概要
USBH0_STS_CONFIGURED	コンフィギュアード
USBH0_STS_REQUEST_COMPLETE	クラスリクエスト完了
USBH0_STS_READ_COMPLETE	データ受信完了

図 9.1 に APL の処理フローチャートを示します。

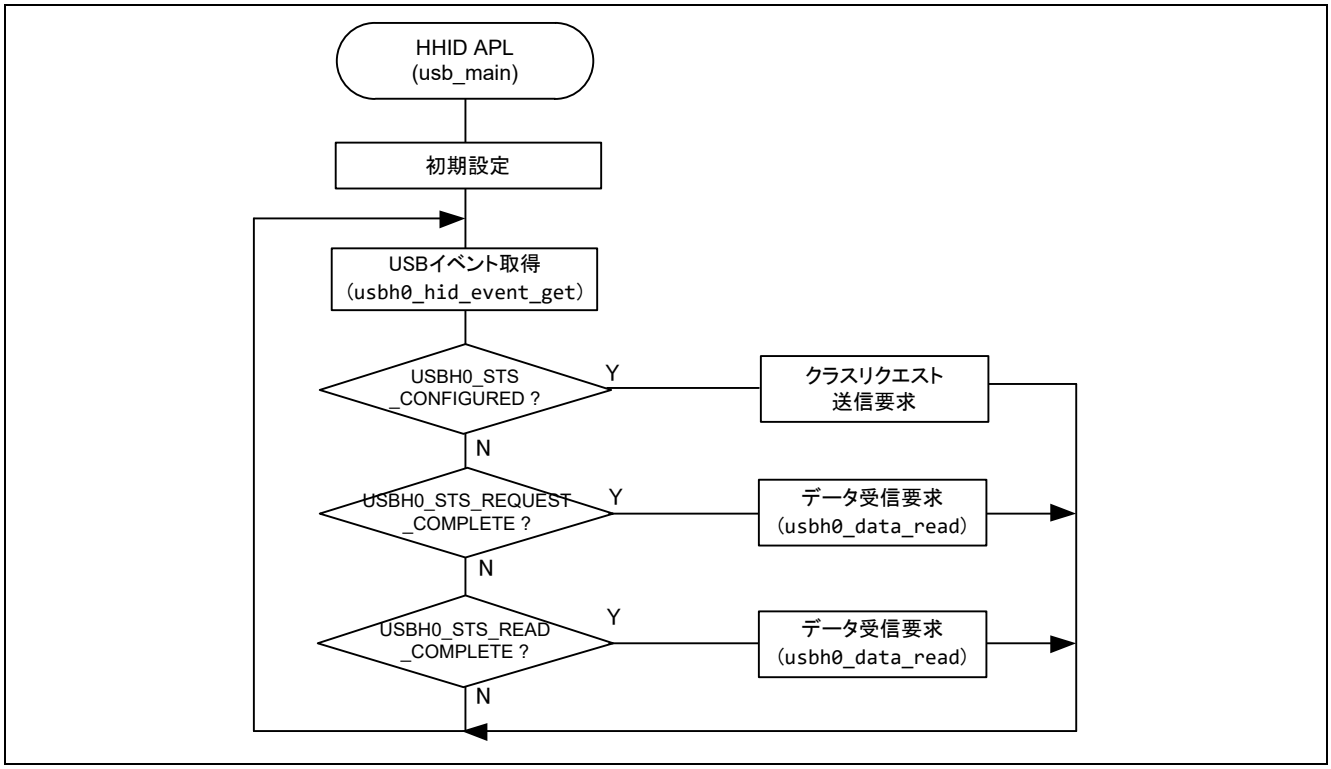


図 9.1 メインループ処理フローチャート

9.2.5 ステートの管理

以下にステートごとの処理概要を示します。

1) デバイスコンフィギュアード (USBH0_STS_CONFIGURED)

== 概略 ==

このステートは、HID デバイスが接続され、デバイスコンフィギュアードになったことを示します。

このステートでは、Set Protocol リクエストをデバイスに送信します。

== 内容 ==

① Set Protocol リクエストをデバイスに送信します。

② Set Protocol リクエストが完了したら、ステートを USBH0_STS_REQUEST_COMPLETE に変更します。

2) クラスリクエスト完了 (USBH0_STS_REQUEST_COMPLETE)

== 概要 ==

このステートは、HID デバイスへのクラスリクエストが完了したことを示します。

HID デバイスに対して、データ受信要求を行います。

== 内容 ==

① データ受信要求を行います。

② データ受信が完了後、ステートを USBH0_STS_READ_COMPLETE に変更します。

3) データ受信完了 (USBH0_STS_READ_COMPLETE)

== 概要 ==

このステートは、HID デバイスへのデータ受信が完了したことを示します。

続けて、HID デバイスに対して、データ受信要求を行います。

== 内容 ==

① データ受信要求を行います。

② データ受信が完了後、ステートを USBH0_STS_READ_COMPLETE に変更します。

10. コンフィグレーション

以下の設定を行ってください。

r_usbh0_basic_config.h、r_usbh1_basic_config.h ファイルに対する設定を行ってください。
r_usbh0_basic_config.h、r_usbh1_basic_config.h については、USB Basic Host Driver アプリケーションノート (ドキュメント No. R01AN4715JJ0120)内の「コンフィグレーション」の章を参照してください。

10.1 r_usb_hhid_apl.h

以下の書く定義に対する設定を行ってください。

1. USBHx_APL_MODE 定義
   USBHx_APL_MODE 定義に対し、以下のどちらかを指定してください。

```
#define   USBHx_APL_MODE     HID_NORMAL  // NORMAL モード
#define   USBHx_APL_MODE     HID_DEMO    // DEMO モード
```

11. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

RZ/A2M グループ ユーザーズマニュアル ハードウェア編
（最新版をルネサス エレクトロニクスホームページから入手してください。）

RTK7921053C00000BE（RZ/A2M CPU ボード）ユーザーズマニュアル
（最新版をルネサス エレクトロニクスホームページから入手してください。）

RTK79210XXB00000BE（RZ/A2M SUB ボード）ユーザーズマニュアル
（最新版をルネサス エレクトロニクスホームページから入手してください。）

Arm Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C
（最新版を Arm ホームページから入手してください。）

Arm Cortex™-A9 Technical Reference Manual Revision: r4p1
（最新版を Arm ホームページから入手してください。）

Arm Generic Interrupt Controller Architecture Specification - Architecture version2.0
（最新版を Arm ホームページから入手してください。）

Arm CoreLink™ Level 2 Cache Controller L2C-310 Technical Reference Manual Revision: r3p3
（最新版を Arm ホームページから入手してください。）

テクニカルアップデート／テクニカルニュース
（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：統合開発

統合開発環境 e2 studio のユーザーズマニュアルは、ルネサス エレクトロニクスホームページから入手してください。
（最新版をルネサス エレクトロニクスホームページから入手してください。）

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2019 年 4 月 15 日	26	初版発行
1.10	2019 年 5 月 17 日	26	表 2.1　動作確認条件 コンパイラオプション"-mthumb-interwork"を削除
1.20	2020 年 3 月 31 日	20	ユーザ定義関数の追加
1.30	2020 年 6 月 30 日		Basic VerUP

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. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   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. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「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。
5. クロックについて
   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。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   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、将来の拡張機能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。
8. 製品間の相違について
   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、フラッシュメモリ、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、ノイズ幅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生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、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またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 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、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング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）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、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（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合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、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他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。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. 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。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
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。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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直接的、間接的に支配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。

(Rev.4.0-1　2017.11)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。