# RZ/A1H グループ

R01AN3466JJ0100
Rev.1.00
Sep 30, 2016

## USB Host Human Interface Device Class Driver (HHID)

### 要旨

本アプリケーションノートでは 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（HHID）について説明します。本ドライバは USB Basic Firmware（USB-BASIC-FW）と組み合わせることで動作します。以降、本ドライバを HHID と称します。

### 対象デバイス

RZ/A1H グループ

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

### 関連ドキュメント

1. Universal Serial Bus Revision 2.0 specification
   http://www.usb.org/developers/docs/
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. RZ/A1H グループ、RZ/A1M グループ　ユーザーズマニュアル ハードウェア編
   (ドキュメント No.R01UH0403JJ)
5. RZ/A1H グループ USB Host and Peripheral Interface Driver（ドキュメント No.R01AN3291JJ）
6. RZ/A1Hグループ ARM® Development Studio 5（DS-5™）のセミホスティング機能を使用したNOR型フラッシュメモリへのダウンロード例（ドキュメントNo.R01AN1957JJ）
7. RZ/A1H グループレジスタ定義ヘッダ・ファイル iodefine.h（R01AN1860JJ）
8. RZ/A1H グループ初期設定例（R01AN1864JJ）

— ルネサス エレクトロニクスホームページ
【http://japan.renesas.com/】
— USB デバイスページ
【http://japan.renesas.com/prod/usb/】

# 目次

# 1. 概要

HHID は、USB-BASIC-FW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
- 一つのUSBチャネルに対しUSB Hubを使って最大3つのHIDデバイスの接続が可能。

## 1.1 必ずお読みください

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

[Note]

1. ドキュメント(Document No:R11AN3291JJ)に記載された API を使ったアプリケーションプログラムの作成方法は当該ドキュメントに記載されています。
2. ドキュメント(Document No:R11AN0022JJ)に記載された API を使用した場合、本書の「7.2 HHID API 一覧」に記載された API を使用する必要はありません。

## 1.2 動作確認済環境

HHID の動作確認済環境をFigure 1-1に示します。

**Figure 1-1 動作確認条件**

| 項目 | 内容 |
|---|---|
| 使用マイコン | RZ/A1H |
| 動作周波数（注） | CPU クロック（Iφ）: 400MHz |
| | 画像処理クロック（Gφ）: 266.37MHz |
| | 内部バスクロック（Bφ）: 133.33MHz |
| | 周辺クロック 1（P1φ）: 66.67MHz |
| | 周辺クロック 0（P0φ）: 33.33MHz |
| 動作電圧 | 電源電圧（I/O）: 3.3V |
| | 電源電圧（内部）: 1.8V |
| 統合開発環境 | ARM®t統合開発環境 |
| | ・ARM Development Studio（DS-5™） Version 5.16 |
| | IARt 統合開発環境 |
| | ・IAR Embedded Workbench for ARM Version 7.40 |
| コンパイラ | ARM C/C++ Compiler/Linker/Assembler Ver.5.03 [Build 102] |
| | KPIT GNUARM-RZ v14.01 |
| | IAR C/C++ Compiler for ARM 7.40 |
| 動作モード | ブートモード 0 |
| | （CS0 空間 16 ビットブート） |
| ターミナルソフトの通信設定 | ・通信速度 ：115200bps |
| | ・データ長 ：8 ビット |
| | ・パリティ ： なし |
| | ・ストップビット長 ：1 ビット |
| | ・フロー制御 ： なし |
| 使用ボード | GENMAI ボード |
| | ・R7S72100 CPU ボード RTK772100BC00000BR |
| 使用デバイス<br>（ボード上で使用する機能） | ・シリアルインターフェース（Dsub-9 コネクタ） |
| | ・USB1 コネクタ、USB2 コネクタ |

## 1.3 制限事項

本ドライバには以下の制限事項があります。

- 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。

## 用語一覧

| | | |
|---|---|---|
| APL | : | Application program |
| cstd | : | Peripheral & Host USB-BASIC-FW用の関数&ファイルのプレフィックス |
| HCD | : | Host control driver of USB-BASIC-FW |
| HDCD | : | Host device class driver (device driver and USB class driver) |
| HHID | : | USB Host Human Interface Device Class Driver |
| hstd | : | Host USB-BASIC-FW用の関数&ファイルのプレフィックス |
| HUBCD | : | Hub class sample driver |
| MGR | : | Peripheral device state maneger of HCD |
| non-OS | : | USB basic firmware for Renesas USB MCU and USB ASSP for OS less system |
| PP | : | プリプロセス定義 |
| USB | : | Universal Serial Bus |
| USB-BASIC-FW | : | USB Basic Host and Peripheral firmware for RZ/A1H グループ (non-OS) |
| タスク | : | 処理の単位 |
| スケジューラ | : | non-OSでタスク動作を簡易的にスケジューリングするもの |
| スケジューラマクロ | : | non-OSでスケジューラ機能を呼び出すために使用されるもの |

## 2. ソフトウェア構成

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。 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。

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

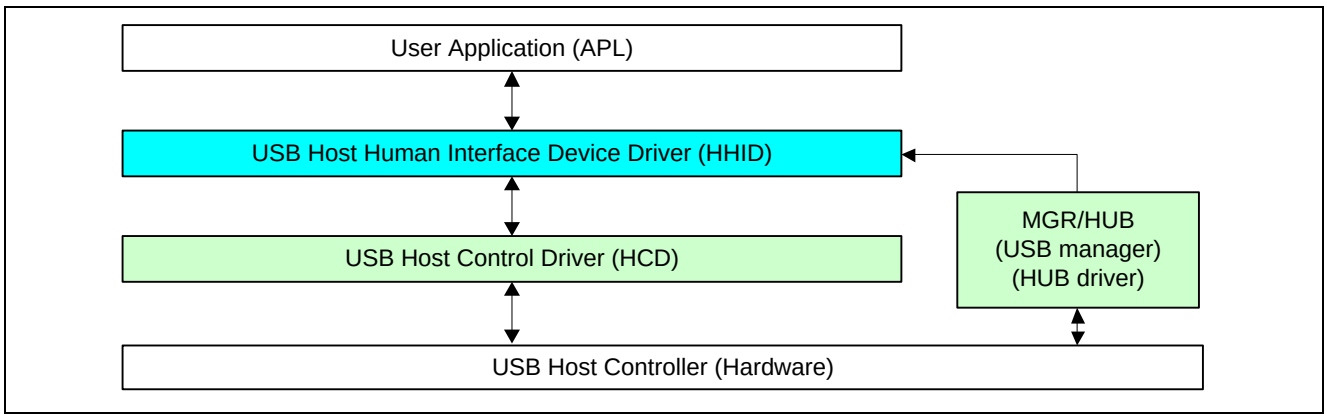


**Figure 2-1 モジュール構成図**

**Table 2.1 モジュール機能概要**

| モジュール名 | 説明 |
|---|---|
| HHID | APL からの HID デバイスに関するリクエストおよびデータ通信を HCD へ要求します。 |
| HCD/MGR | 接続されたデバイスとエニュメレーションをして HCDC を起動します。またデバイスの状態管理も行います。 |
| HCD | USB Host H/W 制御ドライバです。 |

## 3. システム資源

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

**Table 3.1 タスク情報**

| 関数名 | タスク ID | 優先度 | 概要 |
|---|---|---|---|
| usb_hhid_task | USB_HHID_TSK | USB_PRI_4 | HHID タスク |

**Table 3.2 メールボックス情報**

| メールボックス名 | 使用タスク ID | 待ちタスクキュー | 概要 |
|---|---|---|---|
| usb_hhid_MBX | USB_HHID_TSK | FIFO 順 | HHID 用メールボックス |

**Table 3.3 メモリプール情報**

| メモリプール名 | 待ちタスクキュー | メモリブロック（注） | 概要 |
|---|---|---|---|
| usb_hcdc_Task | FIFO 順 | 20byte | HHID 用固定長メモリプール |

（注）全システムのメモリブロックの最大数は、USB_BLKMAX で定義されます。初期値は 20 です。

## 4. ターゲットペリフェラルリスト（TPL）

USB ホストドライバ（USB-BASIC-F/W）とデバイスクラスドライバを組み合わせて使用する際、デバイスドライバごとにターゲットペリフェラルリスト（TPL）を作成する必要があります。

TPL の詳細は USB Host and Peripheral Interfece Driver アプリケーションノート(Document No.R01AN3291JJ)の「ターゲットペリフェラルリストの設定方法」を参照してください。

## 5. コンパイル時の設定

本プロジェクトを使用する場合、USB-BASIC-FWをホストに設定する必要があります。USB-BASIC-FWの設定は、ドキュメント(Document No:R01AN3291JJ)を参照してください。

HHID のコンフィギュレーションオプションの設定は、r_usb_hhid_config.h で行います。

オプション名および設定値に関する説明を、下表に示します。

| Configuration options in r_usb_hhid_config.h | |
|---|---|
| USB_CFG_HHID_INT_IN<br>USB_CFG_HHID_INT_IN2<br>USB_CFG_HHID_INT_IN3 | データ転送に使用されるパイプ番号を指定してください。<br>(PIPE6 から PIPE8 のうちいずれかを指定してください。同じパイプ番号は指定しないでください。) |

# 6. ヒューマンインターフェースデバイスクラス（HID）

## 6.1 基本機能

HID 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。
HID 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

## 6.2 クラスリクエスト

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**

| 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 |

HID 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。

### a). GetReport リクエストフォーマット

Table 6.2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。

**Table 6.2 GetReport フォーマット**

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

### b). SetReport リクエストフォーマット

Table 6.3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。

**Table 6.3 SetReport フォーマット**

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

### c). GetIdle リクエストフォーマット

Table 6.4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。

**Table 6.4 GetIdle フォーマット**

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

### d). SetIdle リクエストフォーマット

Table 6.5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。

**Table 6.5 SetIdle フォーマット**

| bmRequestType | bRequest | wValue | wIndex | wLength | Data |
|---|---|---|---|---|---|
| 0x21 | SET_IDLE<br>（0x0A） | Duration<br>& ReportID | Interface | 0 | Not applicable |

### e). GetProtocol リクエストフォーマット

**Table 6.6**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。

**Table 6.6 GetProtocol フォーマット**

| bmRequestType | bRequest | wValue | wIndex | wLength | Data |
|---|---|---|---|---|---|
| 0xA1 | GET_PROTOCOL<br>（0x03） | 0 | Interface | 1 | 0(BootProtocol) /<br>1(ReportProtocol) |

### f). SetProtocol リクエストフォーマット

Table 6.7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。

**Table 6.7 SetProtocol フォーマット**

| bmRequestType | bRequest | wValue | wIndex | wLength | Data |
|---|---|---|---|---|---|
| 0x21 | SET_PROTOCOL<br>（0x0B） | 0(BootProtocol) /<br>1(ReportProtocol) | Interface | 0 | Not applicable |

## 6.2.1 クラスリクエストパラメータ構造体（USB_HHID_CLASS_REQUEST_PARM_t）

HID クラスリクエストのパラメータ構造体を**Table 6.8**に示します。

**Table 6.8 USB_HHID_CLASS_REQUEST_PARM_t 構造体**

| 型 | メンバ名 | 説明 |
|---|---|---|
| uint16_t | devadr | デバイスアドレス |
| USB_REGADR_t | ipp | USB IP ベースアドレス |
| uint16_t | ip | USB IP 番号 |
| uint16_t | bRequestCode | クラスリクエストコード |
| void* | tranadr | 転送データ格納バッファ |
| uint32_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 |

### 6.2.2 レポートフォーマット

HID で扱うレポートフォーマットを以下に記します。

(1). 受信レポートフォーマット

Table 6.9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 6.9 受信レポートフォーマット**

| offset<br>(データ長) | Keyboard モード<br>(8 バイト) | Mouse モード<br>(3 バイト) |
|---|---|---|
| 0(Top Byte) | Modifier keys | b0：Button 1<br>b1：Button 2<br>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 6.10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 6.10 送信レポートフォーマット**

| offset<br>(データ長) | Keyboard モード<br>(1 バイト) | Mouse モード<br>(非サポート) |
|---|---|---|
| 0(Top Byte) | b0：LED 0(NumLock)<br>b1：LED 1(CapsLock)<br>b2：LED 2(ScrollLock)<br>b3：LED 3(Compose)<br>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、インター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変更してください。

# 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

HHID の基本機能を以下に示します。

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

## 7.2 HHID API 一覧

Table 7.1に HHID API 一覧を示します。

[Note]

1. ドキュメント(Document No: R01AN3291JJ)に記載された API を使用する場合、アプリケーションプログラムでは、以下の API を使用する必要はありません。

**Table 7.1 HHID API 関数一覧**

| 関数名 | 機能概要 |
|---|---|
| R_usb_hhid_task | HHID タスク |
| R_usb_hhid_DriverRelease | ドライバ開放 |
| R_usb_hhid_TransferEnd | USB データ転送強制終了 |
| R_usb_hhid_DeviceInformation | HID デバイスステータス取得 |
| R_usb_hhid_ChangeDeviceState | デバイス状態変更処理 |
| R_usb_hhid_GetReportLength | レポートレングス取得処理 |
| R_usb_hhid_SetPipeRegistration | パイプ設定処理 |
| R_usb_hhid_get_hid_protcol | プロトコルコード取得 |
| R_usb_hhid_driver_start | HHID ドライバ起動 |
| R_usb_hhid_class_request | クラスリクエスト送信 |
| R_usb_hhid_PipeTransfer | USB データ転送要求 |
| R_usb_hhid_class_check | ディスクリプタチェック処理 |
| R_usb_hhid_get_pipetbl | パイプ情報テーブルのアドレス取得 |

### 7.2.1 R_usb_hhid_task

**HHID タスク**

**形式**

void R_usb_hhid_task(USB_VP_INT_t stacd)

**引数**

stacd タスクスタートコード（未使用）

**戻り値**

― ―

**解説**

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

**補足**

1. 本 API はユーザプログラムで呼び出してください。

**使用例**

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

    if( USB_FLGSET == R_usb_cstd_CheckSchedule() )
    {
       R_usb_hstd_HcdTask((USB_VP_INT)0);         /* HCD Task */
       R_usb_hstd_MgrTask((USB_VP_INT)0);         /* MGR Task */
       R_usb_hhub_Task((USB_VP_INT)0);            /* HUB Task */
       R_usb_hhid_task((USB_VP_INT)0);            /* HHID Task */
       usb_hhid_main_task((USB_VP_INT)0);         /* HHID Application Task */
    }
    else
    {
       /* Idle Task (sleep sample) */
       R_usb_cstd_IdleTask(0);
    }
  }
}
```

### 7.2.2 R_usb_hhid_DriverRelease

ドライバ開放

**形式**

| | |
|---|---|
| void | R_usb_hhid_DriverRelease(USB_UTR_t *ptr) |

**引数**

| | |
|---|---|
| *ptr | USB 通信構造体 |

**戻り値**

| | |
|---|---|
| ― | ― |

**解説**

登録した HHID クラスを解放します。

**補足**

1. 登録した HHID が不要になった場合、ユーザアプリケーションで本 API を呼び出してください。
2. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。

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

**使用例**

```
void usb_smp_task( void )
{
  USB_UTR_t *ptr;
    :
  R_usb_hhid_DriverRelease(ptr);
    :
}
```

### 7.2.3 R_usb_hhid_TransferEnd

**USB データ転送強制終了**

**形式**

USB_ER_t R_usb_hhid_TransferEnd(USB_UTR_t *ptr, uint16_t pipe)

**引数**

| | |
|---|---|
| ptr | USB 通信構造体 |
| pipe | パイプ番号 |

**戻り値**

| | |
|---|---|
| USB_E_OK | 正常終了 |
| USB_E_ERROR | 終了失敗 |

**解説**

各パイプのデータ転送を強制終了します。
本 API は、HCD にデータ転送の強制終了要求を行ない、要求を受けた HCD は、データ転送強制終了処理を行います。
データ転送強制終了時に、データ転送要求時(R_usb_hhid_ PipeTransfer)に設定したコールバック関数が呼び出されます。このコールバック関数の引数には、送受信の残りデータ長、ステータス、エラーカウント及び強制終了の情報が設定されます。

**補足**

1. ユーザアプリケーションまたはクラスドライバで本 API を呼び出してください。
2. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。
   USB_REGADR_t ipp : USB IP のアドレス
   uint16_t ip : USB IP 番号
3. 第 1 引数には自動変数(スタック)領域以外の領域を指定してください。

**使用例**

```
void  usb_smp_task(USB_UTR_t *ptr)
{
  uint16_t status;
  uint16_t pipe;
    :
  pipe   = USB_PIPE1;
  status = USB_DATA_STOP;

  /* 転送終了要求 */
  err = R_usb_hhid_TransferEnd(ptr, pipe, status);

  return err;
    :
}
```

### 7.2.4 R_usb_hhid_DeviceInformation

**HID デバイスステータス取得**

**形式**

| | |
|---|---|
| void | R_usb_hhid_DeviceInformation(USB_UTR_t *ptr, uint16_t *deviceinfo, uint16_t devadr) |

**引数**

| | |
|---|---|
| *ptr | USB 通信構造体 |
| *deviceinfo | デバイス情報格納用バッファへのポインタ |
| devaddr | デバイスアドレス |

**戻り値**

| | |
|---|---|
| ― | ― |

**解説**

USB ポートに接続されているデバイスの情報を取得します。
デバイス情報テーブルに格納される情報を以下に示します。

［0］接続されているルートポートの番号
［1］デバイスステート
［2］構成番号
［3］インターフェイスクラスコード 1
［4］接続速度
［5］--
［6］--
［7］--
［8］ルートポート 0 のステータス
［9］ルートポート 1 のステータス

**補足**

1. ユーザアプリケーションまたはクラスドライバで本 API を呼び出してください。
2. 本 F/W はマルチインターフェイス未対応のため、[5]、[6]、[7]は使用しません。
3. 引数*tbl には 20byte の領域を用意してください。
4. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。

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

**使用例**

```
void  usb_smp_task(void)
{
  uint16_t      tbl[10];
    :
  ptr->ip = USB_HOST_USBIP_NUM;       /* USB IP 番号を USB_UTR_t 構造体に設定 */
  /* デバイス情報確認 */
  R_usb_hhid_DeviceInformation(ptr, devaddr, &tbl);
    :
}
```

### 7.2.5 R_usb_hhid_ChangeDeviceState

デバイス状態変更処理

**形式**

void R_usb_hhid_ChangeDeviceState(USB_UTR_t *ptr, uint16_t msginfo, uint16_t devadr)

**引数**

| | |
|---|---|
| *ptr | USB 通信構造体 |
| msginfo | 通信ステータス |
| devadr | デバイスアドレス |

**戻り値**

| | |
|---|---|
| ― | ― |

**解説**

デバイスの状態変更を行います。
msginfo に以下の値を設定し、本関数を呼び出すことで USB デバイスステートの変更を HCD に要求します。

| msginfo | 概要 |
|---|---|
| USB_DO_GLOBAL_SUSPEND | サスペンドへの遷移要求 |
| USB_DO_GLOBAL_RESUME | レジューム信号発行要求 |

**補足**

1. ユーザアプリケーションまたはクラスドライバで本 API を呼び出してください。
2. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。
   USB_REGADR_t ipp : USB IP のアドレス
   uint16_t ip : USB IP 番号

**使用例**

```
void  usb_smp_task( void )
{
  USB_UTR_t *ptr;

  ptr = &usbip;
  ptr->ip = USB_HOST_USBIP_NUM;
  ptr->ipp = R_usb_cstd_GetUsbIpAdr( USB_HOST_USBIP_NUM );
    :
  /* 状態遷移要求 */
  R_usb_hhid_ChangeDeviceState(ptr, USB_DO_GLOBAL_SUSPEND);
    :
}
```

### 7.2.6 R_usb_hhid_GetReportLength

レポートレングス取得処理

**形式**

uint16_t R_usb_hhid_GetReportLength(uint16_t ipno, uint16_t devadr)

**引数**

| | |
|---|---|
| ipno | USB モジュール番号 |
| devadr | デバイスアドレス |

**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**

1. ユーザアプリケーションまたはクラスドライバで本 API を呼び出してください。
2. 引数 ipno には、HID デバイスを接続した USB モジュールの番号(下記参照)を指定してください。

| USB モジュール | USB モジュール番号 |
|---|---|
| USBb | USB_IP0 |
| USBA/USBAa | USB_IP1 |

**使用例**

```
void usb_smp_task( void )
{
  uint16_t   usb_smp_report_length;
    :
  usb_smp_report_length = R_usb_hhid_GetReportLength(USB_IP0, devadr);
    :
}
```

### 7.2.7 R_usb_hhid_SetPipeRegistration

パイプ設定処理

**形式**

void R_usb_hhid_SetPipeRegistration(USB_UTR_t *ptr, uint16_t devadr)

**引数**

| | |
|---|---|
| *ptr | USB 通信構造体 |
| devadr | デバイスアドレス |

**戻り値**

| | |
|---|---|
| — | — |

**解説**

H/W のパイプコンフィグレーションを行います。HDCD 登録時に登録されたパイプ情報テーブルの内容を各パイプに設定します。

**補足**

1. 本 API は初期化処理中にユーザアプリケーションから呼び出してください。
2. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。

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

**使用例**

```
void  usb_smp_task( void )
{
    :
  R_usb_hhid_SetPipeRegistration(ptr, devadr);
    :
}
```

### 7.2.8 R_usb_hhid_get_hid_protocol

プロトコルコード取得

**形式**

uint8_t R_usb_hhid_get_hid_protocol(uint16_t ipno, uint16_t devadr)

**引数**

| | |
|---|---|
| ipno | USB モジュール番号 |
| devadr | デバイスアドレス |

**戻り値**

| | |
|---|---|
| － | USB デバイスのプロトコルコード（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. ユーザアプリケーションまたはクラスドライバで本 API を呼び出してください。
2. bInterfaceProtocol は、Interface Descriptor に含まれます。
3. 引数 ipno には、HID デバイスを接続した USB モジュールの番号(下記参照)を指定してください。

| USB モジュール | USB モジュール番号 |
|---|---|
| USB0 | USB_IP0 |
| USB1 | USB_IP1 |

**使用例**

```
void  usb_smp_task( void )
{
  uint8_t    protocol;
    :
  /* プロトコルコード取得 */
  protocol = R_usb_hhid_get_hid_protocol(USB_IP0, devadr);
    :
}
```

### 7.2.9 R_usb_hhid_driver_start

**HHID ドライバ起動**

**形式**

void R_usb_hhid_driver_start(USB_UTR_t *ptr)

**引数**

*ptr USB 通信構造体

**戻り値**

― ―

**解説**

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

**補足**

1. 初期設定時にユーザアプリケーションで本 API を呼び出してください。
2. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。
   USB_REGADR_t ipp : USB IP のアドレス
   uint16_t ip : USB IP 番号

**使用例**

```
void usb_hstd_task_start( void )
{
  USB_UTR_t  *ptr;
    :
  ptr->ip = USB_HOST_USBIP_NUM;   /* USB IP 番号設定 */
  ptr->ipp = R_usb_cstd_GetUsbIpAdr( ptr->ip );  /* USB IP アドレス設定 */
    :
  R_usb_hhid_driver_start( ptr );    /* Host Class Driver Task Start Setting */
  usb_hstd_usbdriver_start( ptr );   /* Host USB Driver Start Setting */
  usb_hapl_registration( ptr );      /* Host Application Registration */
  usb_hapl_task_start( ptr );        /* Host Application Task Start Setting */
    :
}
```

### 7.2.10 R_usb_hhid_class_request

クラスリクエスト送信

**形式**

USB_ER_t R_usb_hhid_class_request(void *pram)

**引数**

*pram HID クラスリクエスト構造体へのポインタ (6.2.1章を参照ください)

**戻り値**

USB_E_OK 正常終了
USB_E_ERROR 送信失敗

**解説**

HID 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。

**補足**

1. ユーザアプリケーションで本 API を呼び出してください。
2. USB_HHID_CLASS_REQUEST_PARM_t 構造体(6.2.1章を参照)の bRequestCode メンバに発行するクラスリクエストの定義値を設定してください。使用例を参照してください。

| クラスリクエスト | 定義値 |
|---|---|
| Get_Descriptor(HID) | USB_HID_GET_HID_DESCRIPTOR |
| Get_Descriptor(Report) | USB_HID_GET_REPORT_DESCRIPTOR |
| Get_Descriptor(Physical) | USB_HID_GET_PHYSICAL_DESCRIPTOR |
| Set_Report | USB_HID_SET_REPORT |
| Get_Report | USB_HID_GET_REPORT |
| Set_Idle | USB_HID_SET_IDLE |
| Get_Idle | USB_HID_GET_IDLE |
| Set_Protocol | USB_HID_SET_PROTOCOL |
| Get_Protocol | USB_HID_GET_PROTOCOL |

使用例

```
void usb_hhid_smpl_set_report(USB_UTR_t *ptr, 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.ip = ptr->ip;
  class_req.ipp = ptr->ipp;
  class_req.tranadr = p_data;
  class_req.tranlen = length;
  class_req.complete = complete;

  R_usb_hhid_class_request((void*)&class_req);
}
```

### 7.2.11 R_usb_hhid_PipeTransfer

**USB データ転送要求**

**形式**

| | |
|---|---|
| USB_ER_t | R_usb_hhid_PipeTransfer(USB_UTR_t *ptr, uint8_t *buf, uint32_t size, USB_CB_t complete, uint16_t pipe) |

**引数**

| | |
|---|---|
| *ptr | USB 通信構造体 |
| *buf | データ格納バッファ領域へのポインタ |
| size | 読み出しデータサイズ |
| complete | コールバック関数 |
| pipe | パイプ番号 |

**戻り値**

| | |
|---|---|
| USB_E_OK | 正常終了 |

**解説**

USB デバイスに対し、データ転送要求を行ないます。
USB デバイスより送信されたデータ受信が完了すると、第 3 引数で指定したコールバック関数がコールされ、データ受信の完了を通知します。
このコールバック関数の引数（ptr）には、送受信の残りデータ長、ステータス、エラーカウント及び転送終了の情報が設定されています。

**補足**

1. ユーザアプリケーションまたはクラスドライバで本 API を呼び出してください。
2. 第 2 引数には自動変数(スタック)領域以外の領域を指定してください。
3. 受信したデータが MaxPacketSize の n 倍、かつ引数 size に指定したサイズに満たない場合は、データ転送の途中であると判断しコールバック関数 complete は発生しません。
4. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。

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

**使用例**

```
USB_ER_t usb_smp_task(void)
{
  uint16_t  pipe;

  usbip.ip = USB_HOST_USBIP_NUM;
  usbip.ipp = R_usb_cstd_GetUsbIpAdr( usbip.ip );
  pipe = PIPE6;
                    :
  R_usb_hhid_PipeTransfer(&usbip, buf, size,(USB_CB_t)usb_data_received, pipe);
}

/* データ受信完了通知コールバック関数 */
void usb_data_received(USB_UTR_t *mess, uint16_t data2, uint16_t data3)
{
        :
}
```

### 7.2.12 R_usb_hhid_class_check

ディスクリプタ情報取得

**形式**

USB_ER_t R_usb_hhid_class_check(USB_UTR_t *ptr, uint16_t **table)

**引数**

| | |
|---|---|
| *ptr | USB 通信構造体 |
| **table | デバイス情報テーブル<br>[0]：デバイスディスクリプタ<br>[1]：コンフィグレーションディスクリプタ<br>[2]：インタフェースディスクリプタ<br>[3]：ディスクリプタチェック結果<br>[4]：HUB 種別<br>[5]：ポート番号<br>[6]：通信速度<br>[7]：デバイスアドレス<br>[8]：パイプ情報テーブル |

**戻り値**

― ―

**解説**

本 API はクラスドライバレジストレーション関数です。この関数は、スタートアップ時の HHID 登録時にドライバレジストレーション構造体メンバ classcheck にコールバック関数として登録され、エニュメレーション動作のコンフィグレーションディスクリプタ受信時に呼出されます。
チェック結果が OK の場合、ディスクリプタチェック結果（table[3]）に USB_DONE を、チェック結果が NG の場合は USB_ERROR を設定して終了します。
ペリフェラルデバイスのディスクリプタ情報を取得します。

**補足**

―

**使用例**

```
void  usb_hhid_registration(USB_UTR_t *ptr)
{
  USB_HCDREG_t  driver;

  driver.ifclass      = (uint16_t)USB_IFCLS_HID;
  (省略)
  driver.classcheck   = (USB_CB_CHECK_t)&R_usb_hhid_class_check;
  (省略)
  driver.devresume = (USB_CB_INFO_t)&usb_hhid_dummy_function;
  R_usb_hstd_DriverRegistration(ptr, (USB_HCDREG_t*)&driver);
}
```

### 7.2.13 R_usb_hhid_get_pipetbl

パイプ情報テーブルのアドレス取得

**形式**

uint16_t* R_usb_hhid_get_pipetbl (USB_UTR_t *ptr, uint16_t devadr)

**引数**

| | |
|---|---|
| *ptr | USB 通信構造体 |
| devadr | デバイスアドレス |

**戻り値**

| | |
|---|---|
| — | — |

**解説**

第二引数で渡されたアドレスのデバイスとのデータ通信に使用するパイプ情報テーブルのアドレスを取得します。

**補足**

1. ユーザアプリケーションまたはクラスドライバで本 API を呼び出してください。
2. USB 通信用構造体 USB_UTR_t の以下のメンバ設定が必要です。

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

**使用例**

```
void R_usb_hhid_SetPipeRegistration(USB_UTR_t *ptr, uint16_t devadr)
{
  uint16_t    *pipetbl;

  pipetbl = R_usb_hhid_get_pipetbl( ptr,  devadr);
  pipetbl[3] |= (uint16_t)(devadr << USB_DEVADDRBIT);
  R_usb_hstd_SetPipeRegistration(ptr, pipetbl, pipetbl[0]);

}
```

# 8. サンプルアプリケーション

## 8.1 アプリケーション仕様

評価基板に接続された 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、ターミナルソフト上に表示されます。

[Note]

1. 1 つの USB モジュールにつき、USB Hub を使用した場合で最大 3 つの HID デバイスを接続することができます。

## 8.2 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。

### 8.2.1 初期設定

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

### 8.2.2 メインループ

このメインループでは、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(RSK)に HID デバイスが ATTACH され、Enumeration 完了後に R_USB_GetEvent 関数をコールすると戻り値に USB_STS_CONFIGURED がセットされます。APL では、USB_STS_CONFIGURED を確認するとクラスリクエスト 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、R_USB_GetEvent 関数をコールすると戻り値に USB_STS_REQUEST_COMPLETE がセットされます。APL は、USB_STS_REQUEST_COMPLETE を確認すると、R_USB_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、R_USB_GetEvent 関数をコールすると戻り値に USB_STS_READ_COMPLETE がセットされます。APL は、USB_STS_READ_COMPLETE を確認すると、ターミナルソフト上に HID デバイスから受信したデータを表示します。その後、R_USB_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。

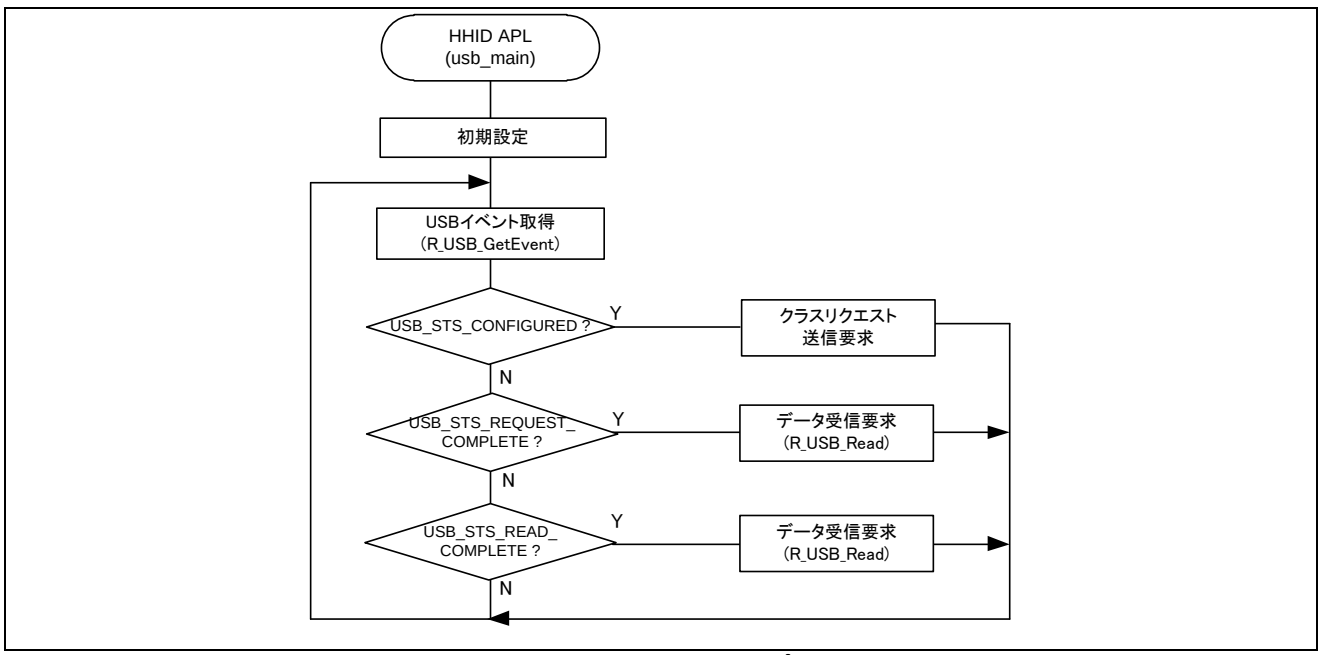


**Figure 8-1 メインループ処理**

## 8.3 表示情報

APL は HID デバイスの接続状態、及び接続された HID デバイスから受信データ内容をターミナルソフトに表示します。

マウス接続時 : X/Y 軸の移動量をターミナルソフトに表示、クリック時 LED 点灯
キーボード接続時 : 最後に入力されたキーデータをターミナルソフトに表示

HID デバイスから受信データの内容が NULL（キーボード非押下、マウスの X/Y 軸移動なし）の場合、表示を更新しません。

# 9. セットアップ

## 9.1 ハードウェア

HHID の動作環境例をFigure 9-1に示します。評価ボードのセットアップ、エミュレータなどの使用方法については各取扱説明書を参照ください。

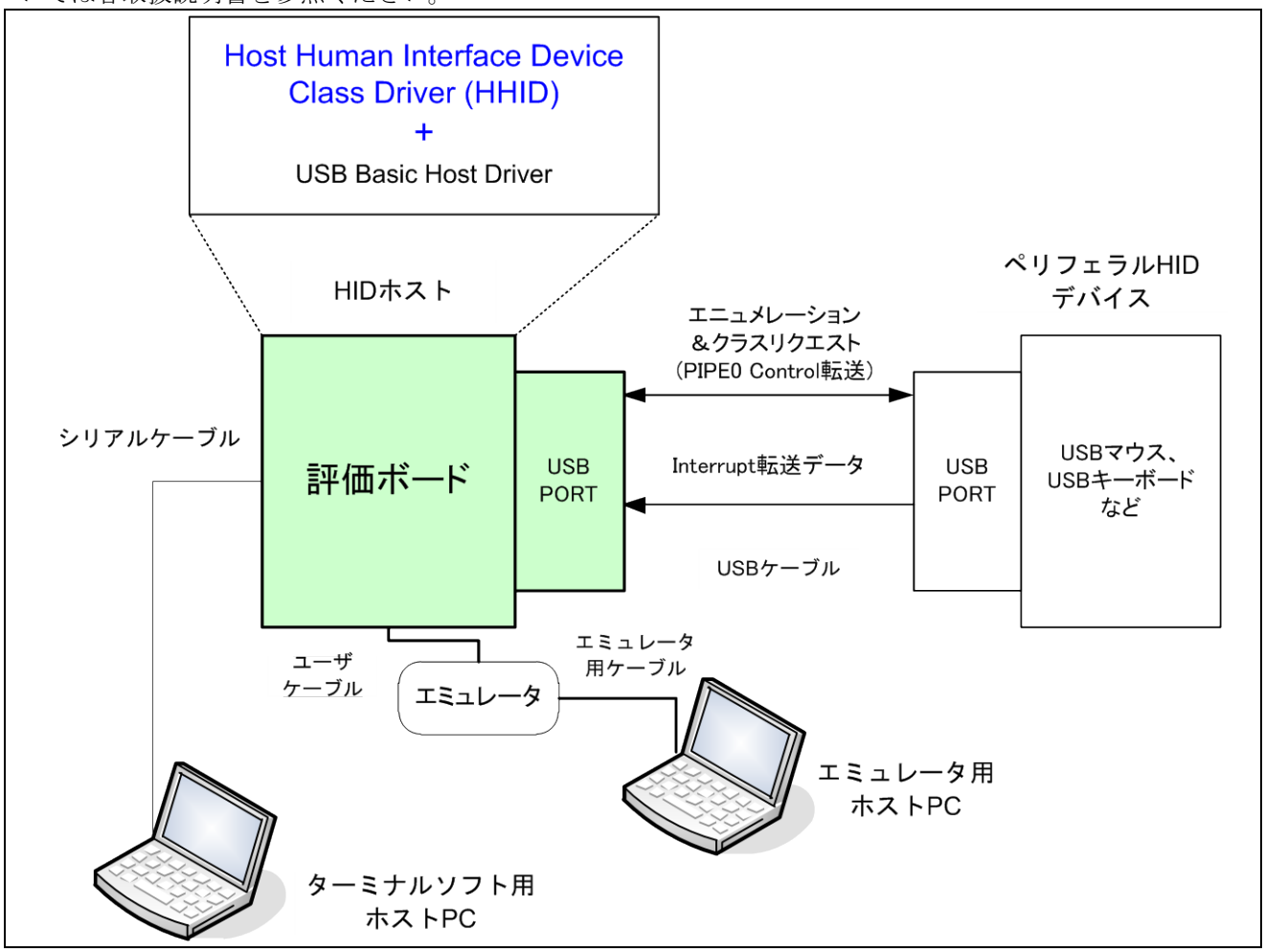


**Figure 9-1 動作環境例**

## ホームページとサポート窓口

- ルネサス エレクトロニクスホームページ
  http://japan.renesas.com/
- お問合せ先
  http://japan.renesas.com/inquiry

## 改訂記録

| Rev. | 発行日 | 改訂内容 | |
|---|---|---|---|
| | | ページ | ポイント |
| 1.00 | Sep 30, 2016 | － | 初版発行 |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

## 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/