RZ/T1 グループ

R01AN2631JJ0130
Rev.1.30
Dec 07, 2017

USB Peripheral Communications Device Class Driver (PCDC)

要旨

本アプリケーションノートでは、USB Peripheral コミュニケーションデバイスクラスドライバについて説明します。本ドライバは USB Peripheral Basic firmware (USB-BASIC-FW)と組み合わせることで動作します。以降、本サンプルソフトウェアを PCDC と称します。

本アプリケーションノートのサンプルプログラムは「RZ/T1 グループ初期設定 Rev.1.30」をベースに作成しています。

動作環境については「RZ/T1 グループ初期設定アプリケーションノート(R01AN2554JJ0130)」を参照してください。

対象デバイス

RZ/T1 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
2. USB Class Definitions for Communications Devices Revision 1.2
3. USB Communications Class Subclass Specification for PSTN Devices Revision 1.2
   【http://www.usb.org/developers/docs/】

4. RZ/T1 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. R01UH0483JJ0130)
5. RZ/T1 グループ初期設定 (ドキュメント No. R01AN2554 JJ0130)
6. USB Peripheral Basic firmware アプリケーションノート (ドキュメント No. R01AN2630 JJ0130)

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

PCDC は、USB-BASIC-FW と組み合わせることで、USB Peripheral コミュニケーションデバイスクラスドライバとして動作します。PCDC は、USB コミュニケーションデバイスクラス仕様の Abstract Control Model に準拠し、USB ホストとの通信を行うことができます。

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。

- USB ホストとのデータ転送
- CDC クラスリクエストに応答
- コミュニケーションデバイスクラスノーティフィケーション送信サービスの提供

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。

ACM	:	Abstract Control Model. This is the USB interface subclass used for virtual COM ports, based in the old V.250 (AT) command standard. See PSTN below.
APL	:	Application program
CDC	:	Communications Devices Class
CDCC	:	Communications Devices Class Communications Interface Class
CDCD	:	Communications Devices Class Data Class Interface
H/W	:	Renesas USB device
PCD	:	Peripheral control driver of USB-BASIC-FW
PCDC	:	Periperal Communications Devices Class
PSTN	:	Public Switched Telephone Network, contains the ACM (above) standard.
USB	:	Universal Serial Bus
USB-BASIC-FW	:	USB Peripheral Basic firmware for RZ/T1 グループ

2. ソフトウェア構成

Figure 2.1 に PCDC のソフトウェア構成、Table 2.1 にモジュール機能概要を示します。

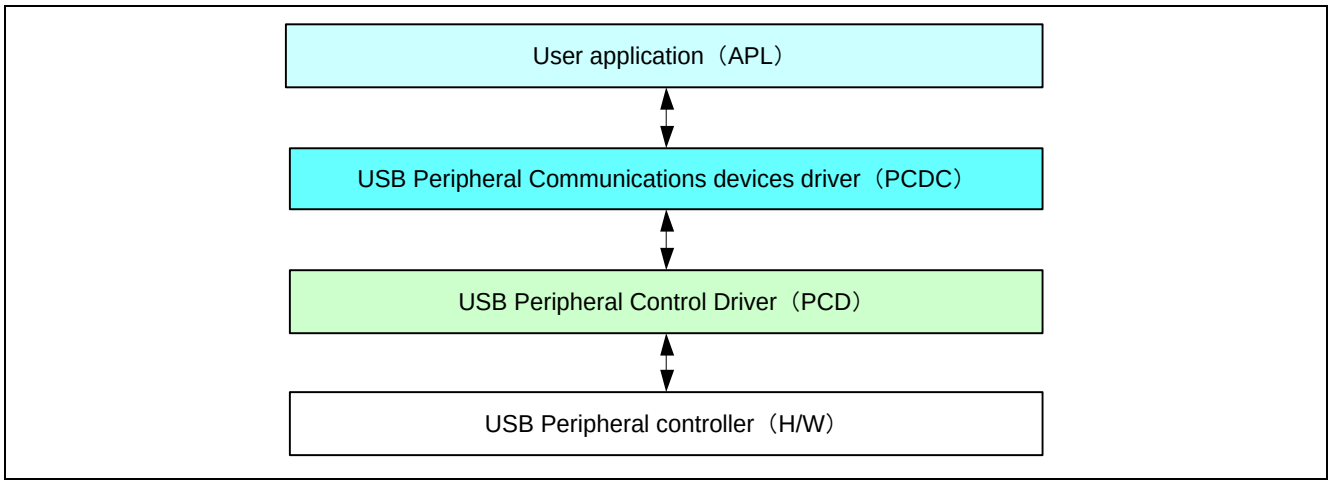


Figure 2.1　ソフトウェア構成図

Table 2.1　各モジュール機能概要

モジュール名	機能概要
APL	ユーザーアプリケーションプログラム（システムに合わせて用意してください）
PCDC	ペリフェラル CDC クラスドライバ ・　APL からの CDC に関するリクエストおよび、データ通信を PCD へ要求
PCD	USB Peripheral H/W 制御（USB-BASIC-FW）

3. ペリフェラルコミュニケーションデバイスクラス (PCDC)

3.1 基本機能

PCDC は、コミュニケーションデバイスクラス仕様 Abstract Control Model サブクラスに準拠しています。
PCDC の主な機能を以下に示します。

1. USB ホストからの機能照会に対する応答
2. USB ホストからのクラスリクエストに対する応答
3. USB ホストとのデータ通信
4. USB ホストへのシリアル通信エラー報告

3.2 Abstract Control Model 概要

Abstract Control Model サブクラスは、USB 機器と従来のモデム（RS-232C 接続）との間を埋める技術で、従来のモデムを使用するアプリケーションプログラムが使用可能です。
以下に PCDC がサポートするクラスリクエスト・クラスノーティフィケーションを記します。

3.2.1 クラスリクエスト（ホスト→デバイスへの通知）

PCDC がサポートするクラスリクエストを Table 3.1 に示します。

Table 3.1　CDC クラスリクエスト

リクエスト	コード	説明	対応
SendEncapsulatedCommand	0x00	プロトコルで定義された AT コマンド等を送信する。	×
GetEncapsulatedResponse	0x01	SendEncapsulatedCommand で送信したコマンドに対するレスポンスを要求する。	×
SetCommFeature	0x02	機器固有の 2 バイトコードや、カントリー設定の禁止／許可を設定する。	×
GetCommFeature	0x03	機器固有の 2 バイトコードや、カントリー設定の禁止／許可状態を取得する。	×
ClearCommFeature	0x04	機器固有の 2 バイトコードや、カントリー設定の禁止／許可設定をデフォルト状態に戻す。	×
SetLineCoding	0x20	通信回線設定を行う。（通信速度、データ長、パリティビット、ストップビット長）	○
GetLineCoding	0x21	通信回線設定状態を取得する。	○
SetControlLineState	0x22	通信回線制御信号 RTS、DTR の設定を行う。	○
SendBreak	0x23	ブレーク信号の送信を行う。	×

○：実装　×：未実装(Stall 応答)

Abstract Control Model リクエストについては、USB Communications Class Subclass Specification for PSTN Devices Revision 1.2 の Table11：Requests-Abstract Control Model を参照して下さい。

3.2.2　クラスリクエストのデータフォーマット

PCDC がサポート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。

(1). SetLineCoding

UART 回線設定を行う為にホストがデバイスに対して送信するクラスリクエストです。
SetLineCoding データフォーマットを以下に示します。

Table 3.2　SetLineCoding フォーマット

bmRequestType t	bReques	wValue	wIndex	wLength	Data
0x21	SET_LINE_CODING (0x20)	0x00	0x00	0x07	Line Coding Structure Table 3.3　Line Coding Structure フォーマット参照

Table 3.3　Line Coding Structure フォーマット

Offset	Field	Size	Value	Description
0	DwDTERate	4	Number	データ端末の速度（bps）
4	BcharFormat	1	Number	ストップビット　0 - 1 Stop bit 1 - 1.5 Stop bit 2 - 2 Stop bit
5	BparityType	1	Number	パリティ　0 - None 1 - Odd 2 - Even
6	BdataBits	1	Number	データビット　（5、6、7、8）

(2). GetLineCoding

UART 回線設定状態を要求する為にホストがデバイスに対して送信するクラスリクエストです。
GetLineCoding データフォーマットを以下に示します。

Table 3.4　GetLineCoding フォーマット

bmRequestType	bReques	wValue	wIndex	wLength	Data
0xA1	GET_LINE_CODING (0x21)	0x00	0x00	0x07	Line Coding Structure Table 3.3　Line Coding Structure フォーマット参照

(3). SetControlLineState

UART のフロー制御用信号を設定する為にホストがデバイスに対して送信するクラスリクエストです。
PCDC では RTS/DTR の制御をサポートしていません。
SET_CONTROL_LINE_STATE データフォーマットを以下に示します。

Table 3.5　SET_CONTROL_LINE_STATE フォーマット

bmRequestType	bReques	wValue	wIndex	wLength	Data
0x21	SET_CONTROL_ LINE_STATE (0x22)	Control Signal Bitmap Table 3.6　Control Signal Bitmap フォーマット参照	0x00	0x00	None

Table 3.6　Control Signal Bitmap フォーマット

Bit Position	Description
D15～D2	予約（0 にリセット）
D1	DCE の送信機能を制御　0 - RTS OFF 1 - RTS ON
D0	DTE がレディ状態かの通知　0 - DTR not present 1 - DTR present

3.2.3　クラスノーティフィケーション（デバイス→ホストへの通知）

PCDC のクラスノーティフィケーション対応/非対応を Table 3.7 に示します。

Table 3.7　CDC クラスノーティフィケーション

ノーティフィケーション	コード	説明	対応
NETWORK_CONNECTION	0x00	ネットワーク接続状況を通知する	×
RESPONSE_AVAILABLE	0x01	GET_ENCAPSLATED_RESPONSE への応答	×
SERIAL_STATE	0x20	シリアル回線状態を通知する	○

○：実装　×：未実装

(1). SerialState

UART の状態変化を検出した場合、ホストへ状態通知を行います。

PCDC ではオーバーランエラー、パリティエラー、フレーミングエラー検出をサポートしています。状態通知は正常状態からエラー検出した場合に行います。エラーを連続検出しても状態通知を連続送信しません。

SerialState データフォーマットを以下に示します。

Table 3.8　SerialState フォーマット

bmRequestType	bReques	wValue	wIndex	wLength	Data
0xA1	SERIAL_STATE (0x20)	0x00	0x00	0x02	UART State bitmap Table 3.9　UART State bitmap フォーマット参照

Table 3.9　UART State bitmap フォーマット

Bits	Field	Description
D15～D7		予約
D6	bOverRun	オーバーランエラー検出
D5	bParity	パリティエラー検出
D4	bFraming	フレーミングエラー検出
D3	bRingSignal	着信（Ring signal）を感知した
D2	bBreak	ブレーク信号検出
D1	bTxCarrier	Data Set Ready：回線が接続されて通信可能
D0	bRxCarrier	Data Carrier Detect：回線にキャリア検出

3.3 PC の仮想 COM ポートについて

Windows OS 搭載 PC は CDC デバイスを仮想 COM ポートとして利用することが可能です。

Windows OS 搭載 PC に PCDC を実装した評価ボードを接続すると、エニュメレーション設定に続き、CDC クラスリクエストの GetLineCoding 及び SetControlLineState を行った後、仮想 COM デバイスとしてデバイスマネージャに登録されます。Windows デバイスマネージャに仮想 COM ポートとして登録された後は、Windows OS 標準搭載のハイパーターミナル等のターミナルアプリで評価ボードとデータ通信が可能です。

ターミナルアプリのシリアルポート設定を行うことで、クラスリクエスト SetLineCoding による通信回線設定が可能です。ターミナルアプリのウインドウから入力したデータ（又はファイル送信）は評価ボードへ転送され、評価ボードから転送したデータはターミナルアプリのウインドウに出力されます。

ターミナルアプリによっては最後に受信したデータがマックスパケットサイズの場合、継続するデータがあると判断して受信データをターミナルに表示しないことがあります。この場合、マックスパケットサイズ未満のデータを受信することで、それまでに受信したデータがターミナルに表示されます。

3.4 API

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_pcdc_if.h に記載しています。
本モジュールのコンフィグレーションオプションの設定は、r_usb_pcdc_config.h で行います。
Table 3.10 にオプション名および設定値に関する説明を示します。

Table 3.10　PCDC コンフィグレーションオプション

定義名	デフォルト値	説明
USB_PMSC_USE_PIPE_IN	USB_PIPE1	Bulk IN 転送パイプ番号
USB_PMSC_USE_PIPE_OUT	USB_PIPE2	Bulk OUT 転送パイプ番号
USB_PCDC_USE_PIPE_STATUS	USB_PIPE6	Interrupt IN 転送パイプ番号

Table 3.11 に PCDC API 一覧を示します。

Table 3.11　PCDC API 一覧

関数名	機能概要
R_usb_pcdc_SendData	USB 送信処理
R_usb_pcdc_ReceiveData	USB 受信処理
R_usb_pcdc_SerialStateNotification	クラスノーティフィケーション”SerialState”送信
R_usb_pcdc_ctrltrans	CDC 用コントロール転送処理

3.4.1　R_usb_pcdc_SendData

USB 送信処理

形式

void	R_usb_pcdc_SendData(uint8_t *table, uint32_t size, USB_UTR_CB_t complete)

引数

*table	転送データアドレス
size	転送サイズ
complete	処理完了通知コールバック関数

戻り値

―	―

解説

指定されたアドレスから転送サイズ分のデータを送信します。
送信完了後、コールバック関数 complete が呼出されます。

補足

USB 送信処理結果はコールバック関数の引数で得られます。
USB-BASIC-FW アプリケーションノートの USB 通信用構造体（USB_UTR_t 構造体）を参照して下さい。

使用例

```
void usb_apl_task( void )
{
  uint8_t    send_data[] = {0x01,0x02,0x03,0x04,0x05};  /* USB 送信データ */
  uint32_t   size = 5;                                  /* USB 送信データ数 */

  R_usb_pcdc_SendData(send_data, size, &usb_complete);
}

/* USB 送信完了通知用コールバック関数 */
void usb_complete(USB_UTR_t *mess)
{
  /* USB 送信完了時の処理を記述して下さい。 */
}
```

3.4.2 R_usb_pcdc_ReceiveData

USB 受信処理

形式

void	R_usb_pcdc_ReceiveData (uint8_t *table, uint32_t size, USB_UTR_CB_t complete)

引数

*table	転送データアドレス
size	転送サイズ
complete	処理完了通知コールバック関数

戻り値

―	―

解説

HCD に USB 受信要求を行います。
指定した転送サイズ分のデータ受信完了、またはマックスパケットサイズ未満のデータを受信した場合、コールバック関数 complete が呼出されます。
USB 受信データは、転送データアドレスで指定された領域に格納されます。

補足

USB 受信処理結果はコールバック関数の引数で得られます。
USB-BASIC-FW アプリケーションノートの USB 通信用構造体（USB_UTR_t 構造体）を参照して下さい。

使用例

```
void usb_apl_task( void )
{
  uint8_t    receive_data[64];                    /* USB 受信データ格納領域 */
  uint32_t   size = 64;                           /* USB 受信要求サイズ */

  R_usb_pcdc_ReceiveData(receive_data, size, &usb_complete);
}

/* USB 受信完了通知用コールバック関数 */
void usb_complete(USB_UTR_t *mess)
{
  /* USB 受信完了時の処理を記述して下さい。 */
}
```

3.4.3　R_usb_pcdc_SerialStateNotification

クラスノーティフィケーション”SerialState”を送信

形式

void	R_usb_pcdc_SerialStateNotification(USB_SCI_SerialState_t serial_state, USB_UTR_CB_t complete)

引数

serial_state	シリアルステータス
complete	処理完了通知コールバック関数

戻り値

―	―

解説

CDC クラスノーティフィケーション”SerialState”を USB ホストに送信します。
この送信はインタラプト IN 転送を使用します。
送信完了後、コールバック関数 complete が呼出されます。

補足

USB_SCI_SerialState_t は"Table 3.9　UART State bitmap フォーマット"に従って定義されています。
USB 送信処理結果はコールバック関数の引数で得られます。
USB-BASIC-FW アプリケーションノートの USB 通信用構造体（USB_UTR_t 構造体）を参照して下さい。

使用例

```
void usb_apl_task( void )
{
  USB_SCI_serialState_t  serial_state;  /* シリアルステータス */

  serial_state.WORD = 0x0000;
  serial_state.bParity = 0x0020;        /* D5:パリティエラー */
  R_usb_pcdc_SerialStateNotification(serial_state, &usb_complete);
}

/* シリアルステート送信完了通知用コールバック関数 */
void usb_complete(USB_UTR_t *mess)
{
  /* シリアルステート送信完了時の処理を記述して下さい。 */
}
```

3.4.4 R_usb_pcdc_ctrltrans

CDC 用コントロール転送処理

形式

void R_usb_pcdc_ctrltrans (USB_REQUEST_t *preq, uint16_t ctsq)

引数

*preq	クラスリクエストメッセージへのポインタ
ctsq	コントロール転送ステージ情報 USB_CS_IDST：Idle or setup stage USB_CS_RDDS：Control read data stage USB_CS_WRDS：Control write data stage USB_CS_WRND：Control write nodata status stage USB_CS_RDSS：Control read status stage USB_CS_WRSS：Control write status stage USB_CS_SQER：Sequence error

戻り値

—	—

解説

本 API は USB_PCDREG_t 構造体のメンバ ctrltrans にコールバック関数として登録してください。
リクエストタイプが CDC クラスリクエストの場合、コントロール転送ステージに対応した処理を呼び出します。

補足

—

使用例

```
void pcdc_init(void)
{
  USB_PCDREG_t  driver;

  driver.ctrltrans = &R_usb_pcdc_ctrltrans;
  R_usb_pstd_DriverRegistration(&driver);
}
```

4. サンプルアプリケーション

本章では、PCDC と USB-BASIC-FW を組み合わせ、USB ドライバとして使用するために必要な初期設定の方法と、メインルーチン処理方法及び API 関数を使用したデータ転送例を示します。

4.1 動作環境

PCDC の動作環境例を Figure 4.1 に示します。

また、動作確認を行った OS 環境を Table 4.1 に示します。

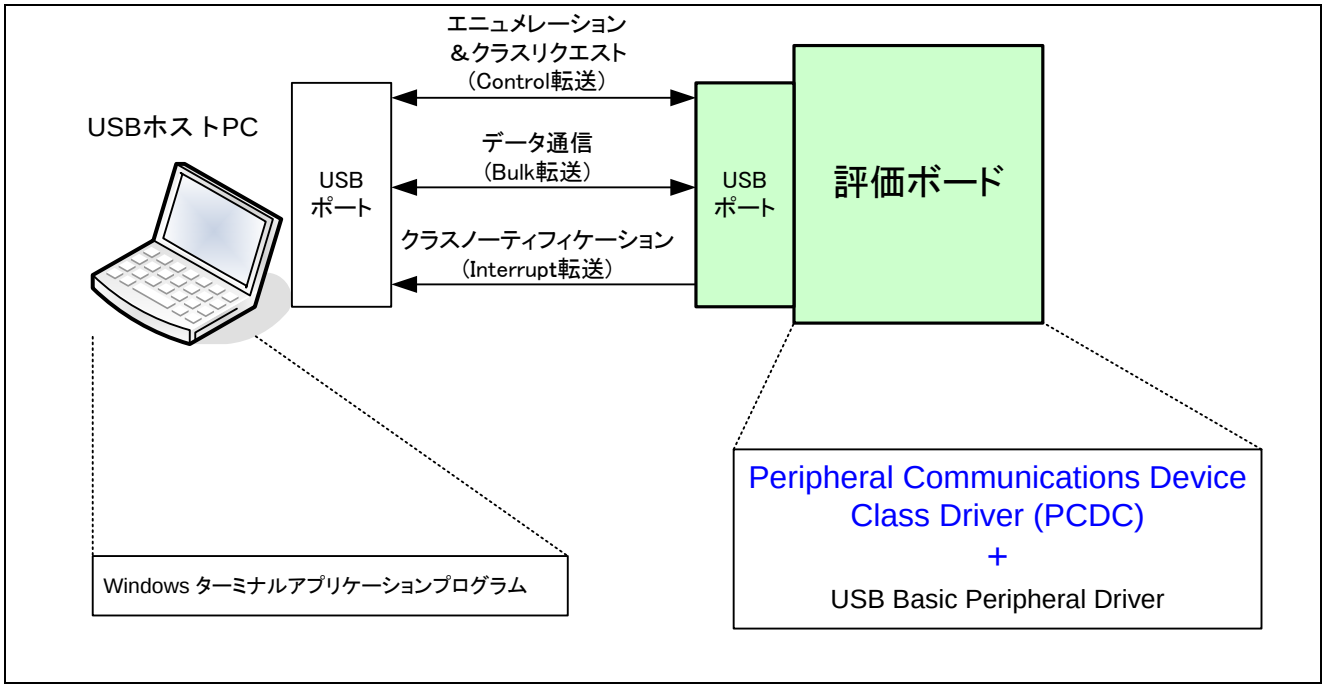


Figure 4.1　動作環境例

Table 4.1 動作確認済み OS

OS 名	備考
Windows 7 32bit	—
Windows 7 64bit	—
Windows 8.1 32bit	—
Windows 8.1 64bit	—
Windows 10 32bit	—
Windows 10 64bit	—

4.2 仕様概要

サンプルアプリケーションの主な機能を以下に示します。

1. USB の初期設定をする。
2. CDC クラスリクエストに応答する。
3. USB ホストからデータを受信する。
4. その受信データを USB ホストへ送信する。

4.3 初期設定

初期設定例を以下に示します。

```
void usbf_main(void)
{
    /* USB ドライバの初期設定「4.3.1」参照 */
    pcdc_registration();
    /* USB モジュールの起動「4.3.2」参照 */
    R_USB_Open();

    apl_init();
    /* メインルーチン「4.4」参照 */
    pcdc_main();
}
```

4.3.1 USB ドライバの初期設定

USB ドライバの設定では、USB-BASIC-FW に対するクラスドライバの情報登録を行います。
詳細は、USB-BASIC-FW アプリケーションノートの「ペリフェラルクラスドライバの登録方法」を参照してください。

4.3.2 USB モジュールの起動

R_USB_Open()を呼び出すことで、USB モジュールをハードウェアマニュアルの初期設定シーケンスに従って設定し、USB 割り込みハンドラの登録と USB 割り込み許可設定を行います。

4.4 メインルーチン

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割り込み処理関数（R_usb_pstd_poll）を呼びだすことで動作します。メインルーチン内で R_usb_pstd_poll()を呼び出すことで割り込みの有無を確認し、割り込みが発生していた場合割り込みに応じた処理を行います。

```
void pcdc_main(void)
{
    while(1)
    {
        R_usb_pstd_poll();

        switch( cdc_dev_info.state )
        {
            case STATE_DATA_TRANSFER:
                cdc_data_transfer();
                break;
            case STATE_ATTACH:
                cdc_connect_wait();
                break;
            case STATE_DETACH:
                cdc_detach_device();
                break;

            default:
                 break;
        }
    }
}
```

4.4.1 APL

APL は、以下の処理を行います。

1. APL は、ステートとそのステートに関連するイベントにより管理を行っています。 APL では、まず、ステート (Table 4.2 参照)の確認を行います。なお、このステートは、APL が管理する構造体(「4.4.2」参照) のメンバに格納されています。
2. 次に APL は、そのステートに関連するイベント(Table 4.3 参照)の確認を行い、そのイベントに応じた処理を行います。そのイベント処理後、APL は、必要に応じてステートを変化させます。なお、このイベントは、APL が管理する構造体(「4.4.2」参照) のメンバに格納されています。

以下に、APL の処理概要を示します。

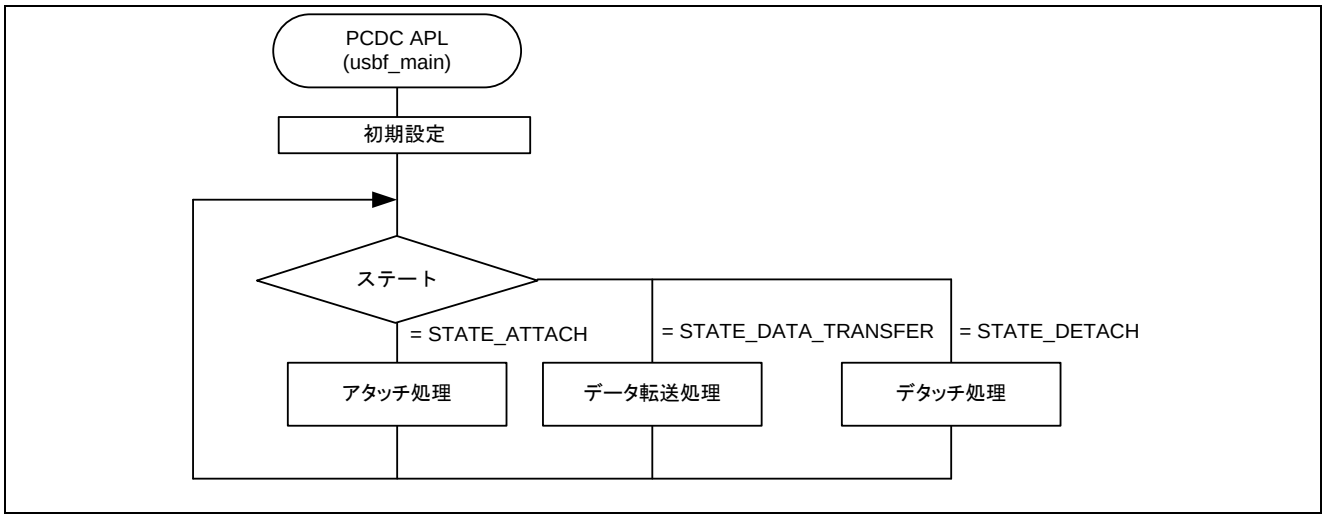


Figure 4.2 メインループ処理

Table 4.2 ステート一覧

ステート	ステート処理概要	関連イベント
STATE_ATTACH	アタッチ処理	EVENT_CONFIGURD
STATE_DATA_TRANSFER	データ転送処理	EVENT_USB_READ_START
		EVENT_USB_READ_COMPLETE
		EVENT_USB_WRITE_START
		EVENT_USB_WRITE_COMPLETE
		EVENT_COM_NOTIFY_START
		EVENT_COM_NOTIFY_COMPLETE
STATE_DETACH	デタッチ処理	--

Table 4.3 イベント一覧

イベント	概要
EVENT_CONFIGURD	USB デバイス接続完了
EVENT_USB_READ_START	USB データ受信要求
EVENT_USB_READ_COMPLETE	USB データ受信完了
EVENT_USB_WRITE_START	USB データ送信要求
EVENT_USB_WRITE_COMPLETE	USB データ送信完了
EVENT_COM_NOTIFY_START	Class Notification“SerialState” 送信要求
EVENT_COM_NOTIFY_COMPLETE	Class notification”SerialState” 送信完了
EVENT_NONE	イベントなし

4.4.2 ステートとイベントの管理

ステートとイベントは、以下の構造体によって管理されています。この構造体は、APLが用意している構造体です。

イベント取得処理で、取り出すイベントが存在しない場合、”EVENT_NONE”がセットされます。

```
typedef struct DEV_INFO           /* Structure for CDC device control */
{
    uint16_t    state;            /* State for application */
    uint16_t    event_cnt;        /* Event count */
    uint16_t    event[EVENT_MAX]; /* Event. */
}
DEV_INFO_t;
```

以下にステートごとの処理概要を示します。

1)　　アタッチ処理 (STATE_ATTACH)

== 概要 ==

このステートでは、USB ホストに接続され、Enumeration が完了したことを APL に通知する処理を行い、ステートを STATE_DATA_TRANSFER に変更します。

== 内容 ==

① APL では、はじめに初期化関数がステートを STATE_ATTACH にセットし、イベントを EVENT_NONE にセットします。

② USB ホストに接続するまで STATE_ATTACH 状態が継続され、cdc_connect_wait()がコールされます。USB ホストに接続した後、USB ホストとの Enumeration が完了するとコールバック関数 cdc_configured()が USB ドライバよりコールされ、このコールバック関数がイベント EVENT_CONFIGURD を発行します。なお、このコールバック関数は、USB_PCDREG_t 構造体のメンバ devconfig に設定した関数です。

③ EVENT_CONFIGURD では、ステートを STATE_DATA_TRANSFER に変更し、イベント EVENT_USB_READ_START を発行します。

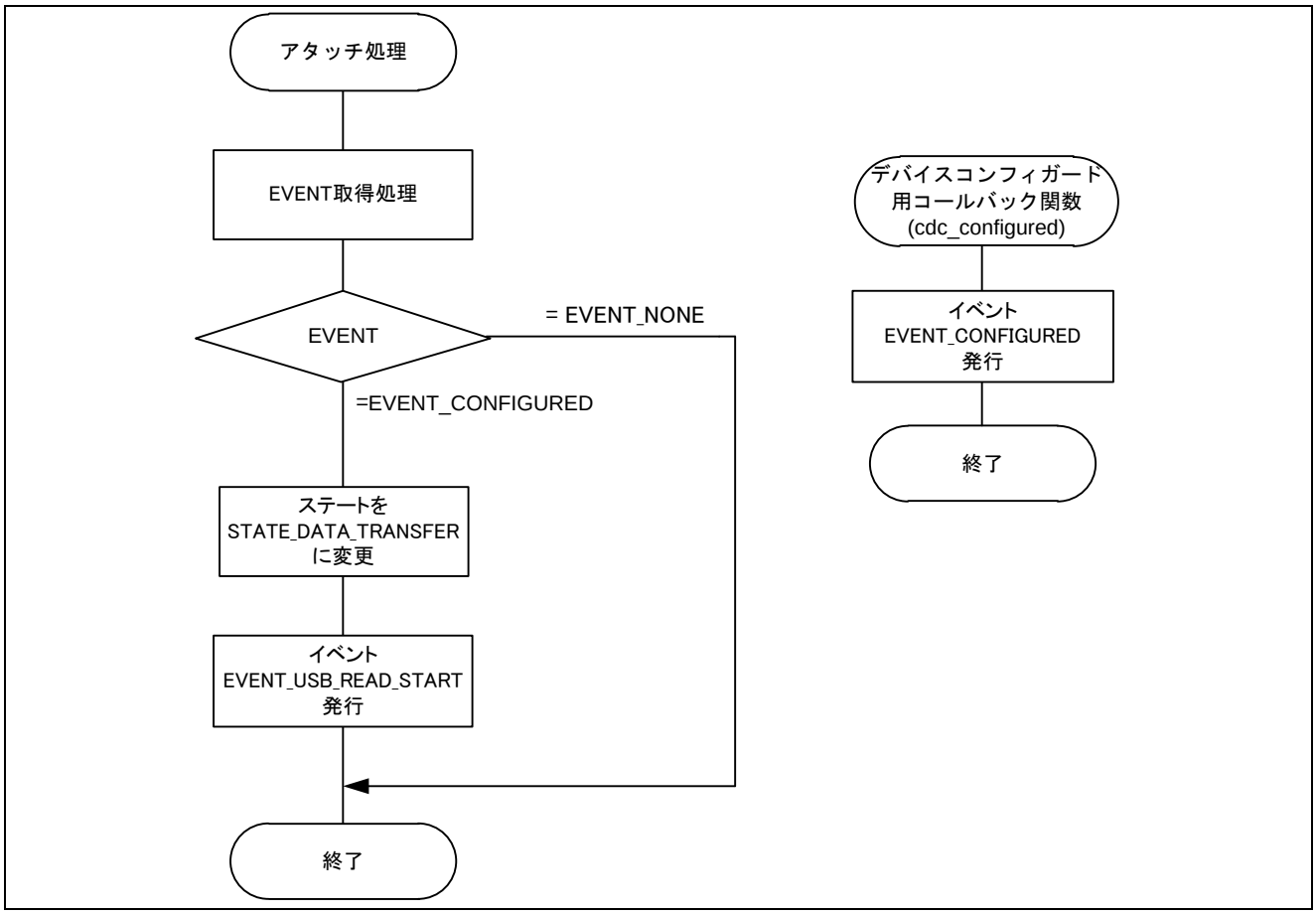


Figure 4.3 アタッチ処理概略フロー

2)　データ転送処理(STATE_DATA_TRANSFER)

== 概要 ==

このステートでは、USB ホストから最初のデータを受信するまでの間、定期間隔で USB ホストに初期接続メッセージを送信します。最初のデータ受信後は、USB ホストからのデータを受信し、その受信したデータをそのまま USB ホストに送信するループバック処理を行います。

== 内容 ==

[初期接続メッセージ]

1. USB ホストに接続すると USB ホストに対し以下のメッセージが一定周期で送信されます。

   PCDC.Virtual serial COM port. Type characters into terminal.
   The target will receive the characters over USB CDC, then copy them to a USB transmit buffer to be echoed back over USB."

2. PC 上の Terminal ソフトからデータが転送されると”Echo Mode.”が Terminal ソフト上に表示されます。本表示以降は、ループバック処理が行われます。

［データ受信処理］

① EVENT_USB_READ_START では、USB ホストから送信されるデータを受信するため USB ドライバに対しデータ受信要求を行います。

② データ受信処理が完了するとコールバック関数 cdc_read_complete()がコールされます。このコールバック関数では、EVENT_USB_READ_COMPLETE を発行します。

③ EVENT_USB_READ_COMLETE では、イベント EVENT_USB_WRITE_START を発行します。

［データ送信処理］

④ EVENT_USB_WRITE_START では、上記受信したデータを USB ホストに送信するため、USB ドライバに対しデータ送信要求を行います。

⑤ データ送信転送処理が完了するとコールバック関数 cdc_write_complete()がコールされます。このコールバック関数では、EVENT_USB_WRITE_COMPLETE を発行します。

⑥ EVENT_USB_WRITE_COMLETE では、イベント EVENT_USB_READ_START を発行します。これにより、上記①が再度処理されます。

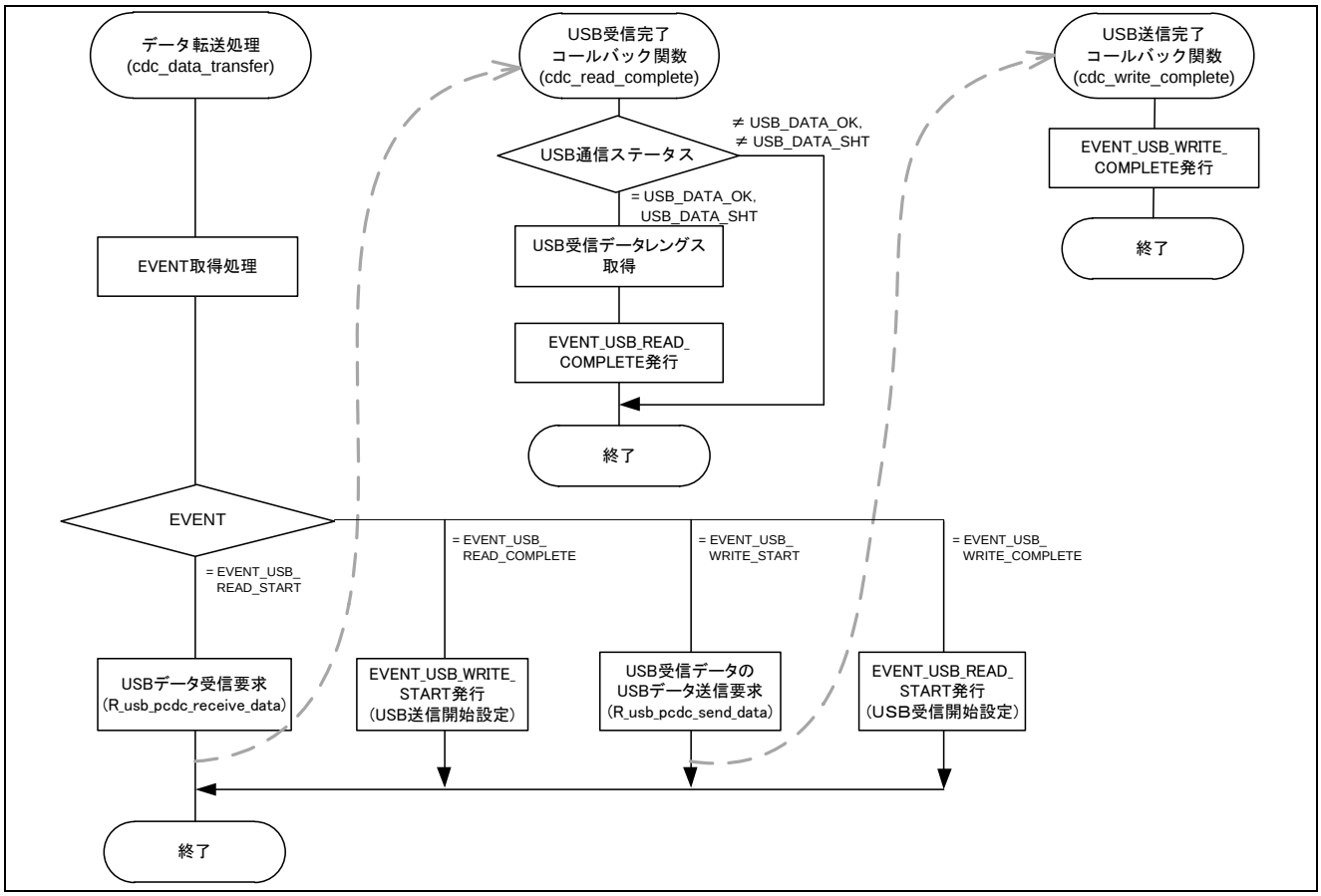


Figure 4.4 データ転送処理概略フロー

3)　　デタッチ処理 (STATE_DETACH)

接続された USB ホストから切断するとコールバック関数 cdc_detach()が USB ドライバよりコールされます。このコールバック関数では、ステートを STATE_DETACH に変更します。STATE_DETACH では、変数のクリア処理、ステートを STATE_ATTACH に変更するなどの処理を行います。

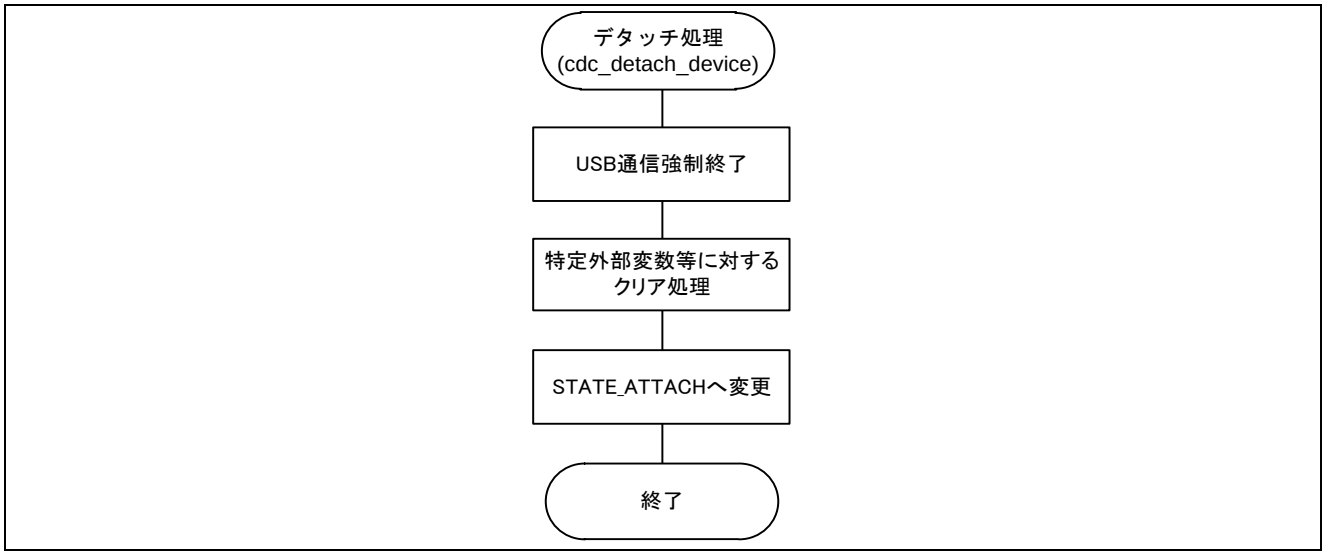


Figure 4.5 デタッチ処理概略フロー

Appendix A. 初期設定の変更点

USB-BASIC-FW を動作させるために「RZ/T1 グループ初期設定 Rev.1.30」を変更しています。
サンプルプログラムは、IAR embedded workbench for ARM(以下、EWARM)と DS-5 と e^2 studio の環境をサポートしています。
本章では、変更点について記述します。

フォルダとファイル

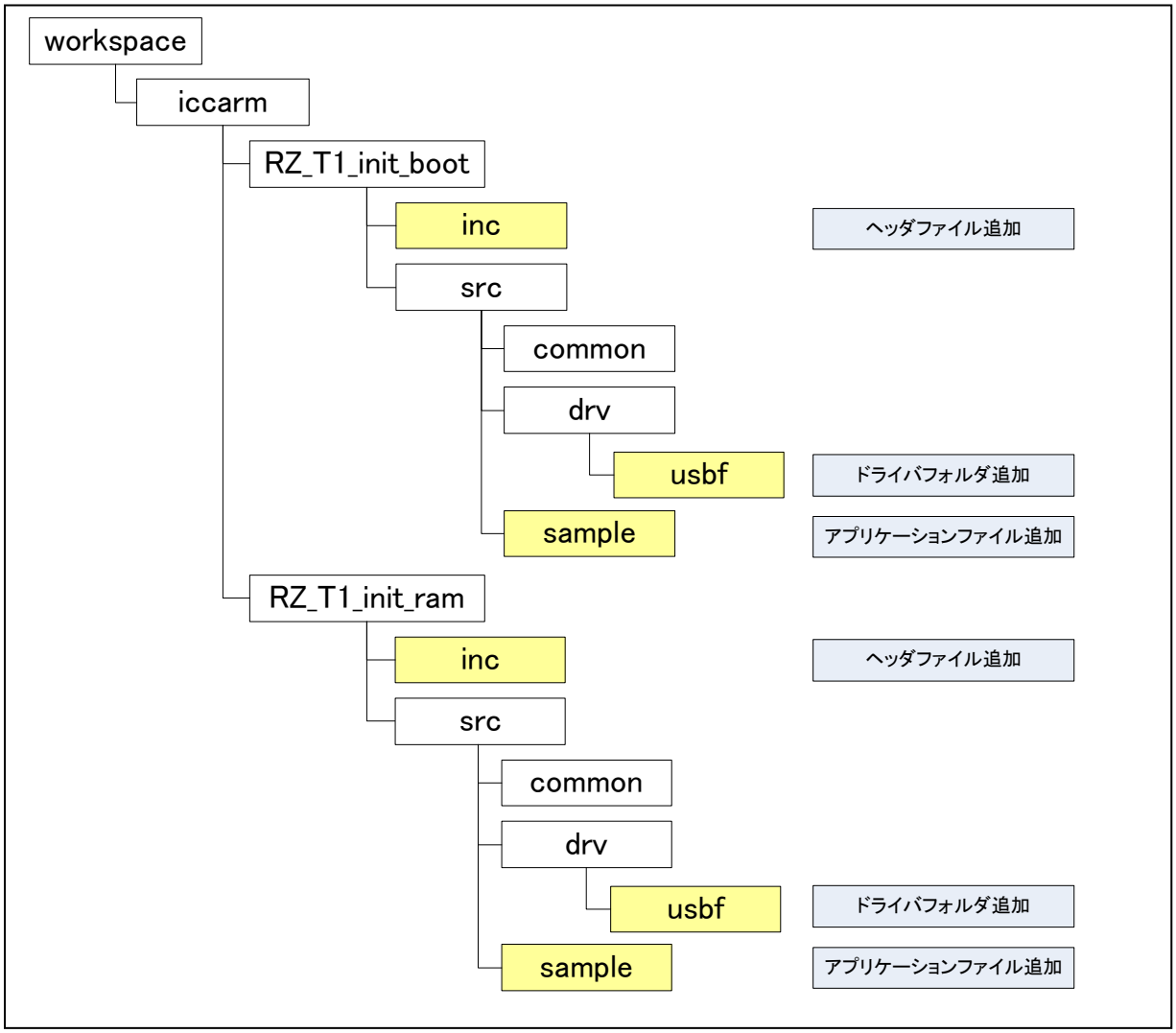
「RZ/T1 グループ初期設定 Rev.1.30」では、開発環境とブート方法によってフォルダ構成が異なります。
全ての開発環境とブート方法の各フォルダに対して、下記の変更をしています。

・”inc”フォルダに下記のファイルを追加
r_usb_basic_config.h
r_usb_basic_if.h
r_usb_cdefusbip.h
r_usb_pcdc_config.h
r_usb_pcdc_if.h

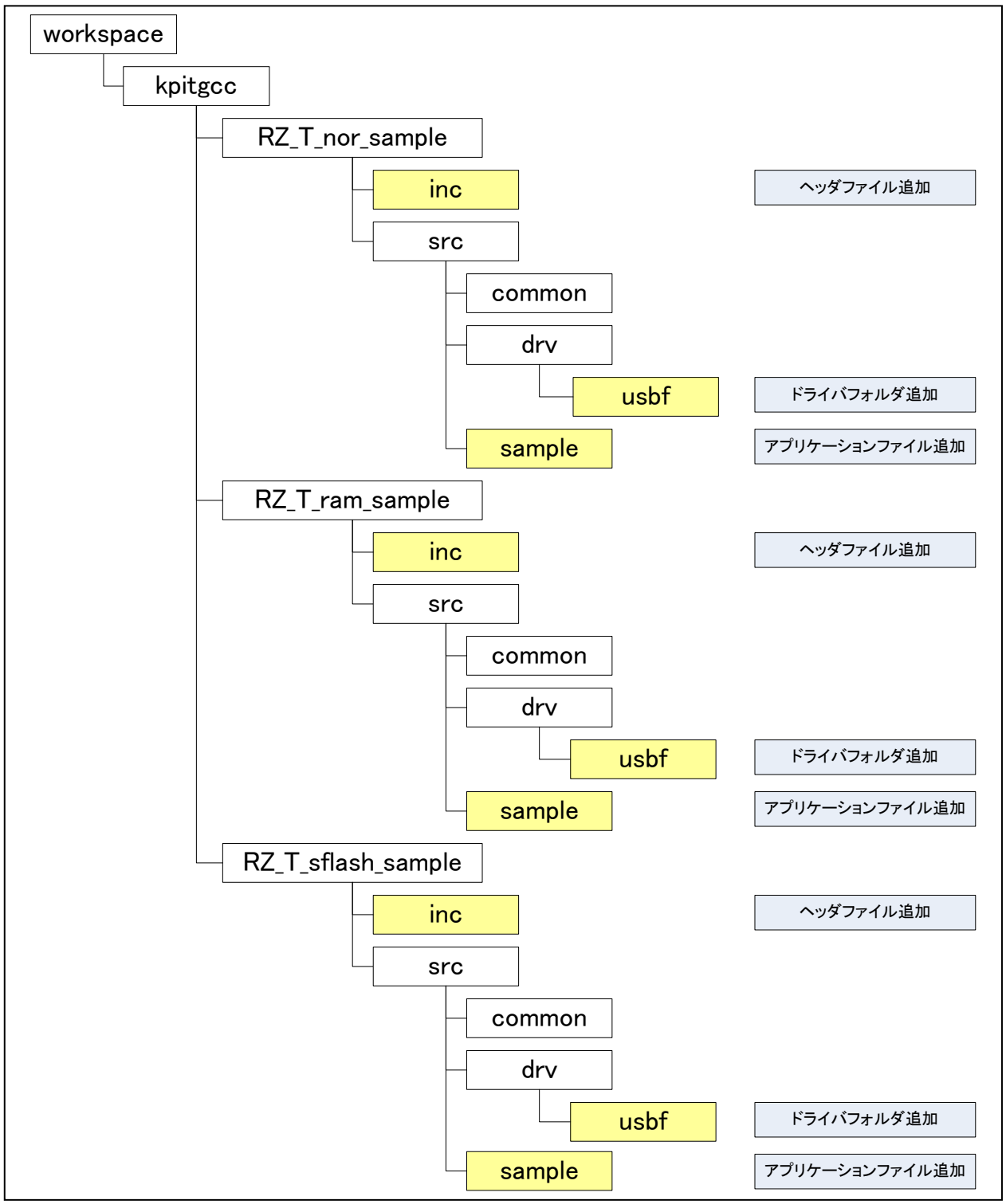
・”sample”フォルダに下記のファイルを追加
r_usb_pcdc_apl.c
r_usb_pcdc_descriptor.c

・”drv”フォルダに usbf フォルダと usbf フォルダ以下のファイルを追加

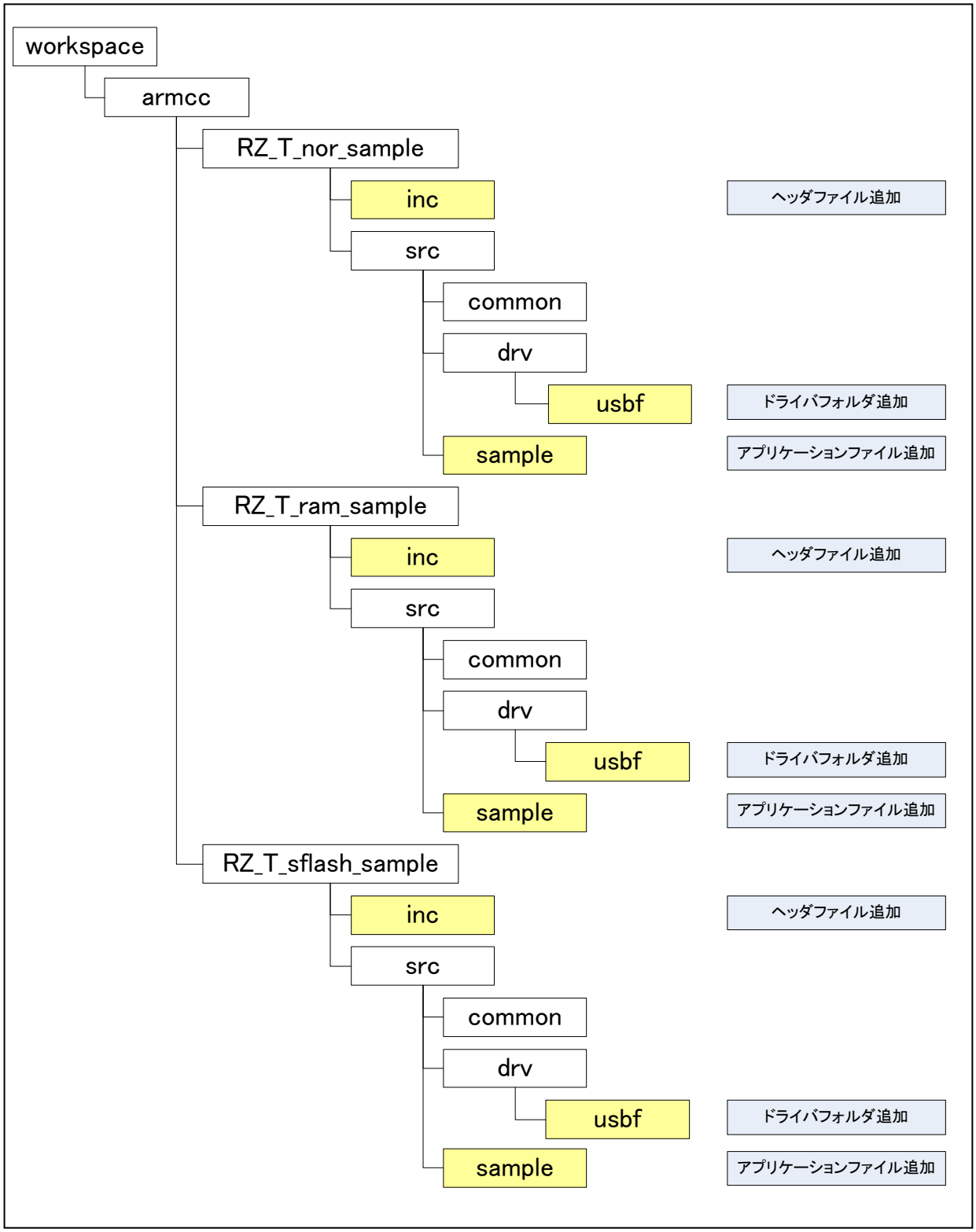
以下に EWARM のフォルダ構成を示します。



以下に e² studio のフォルダ構成を示します。



以下に DS-5 のフォルダ構成を示します。



USB-BASIC-FW 関数の呼び出し

¥src¥sample¥int_main.c の main()に USB-BASIC-FW の usbf_main()の呼び出しを追加しています。

```
extern void usbf_main(void);

int main (void)
{
    /* Initialize the port function */
    port_init();

    /* Initialize the ECM function */
    ecm_init();

    /* Initialize the ICU settings */
    icu_init();

    /* USBf main */
    usbf_main();

    while (1)
    {
        /* Toggle the PF7 output level(LED0) */
        PORTF.PODR.BIT.B7 ^= 1;

        soft_wait();  // Soft wait for blinking LED0

    }

}
```

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Aug 21, 2015	—	初版発行
1.10	Dec 25, 2015	19	Appendix A 追加
1.20	Feb 29, 2016	21	Armcc 関連情報追加
1.30	Dec 07, 2017	—	RZ/T1 初期設定 Ver 1.30 に対応

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子
   （または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（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、その他の不適切に使用しないでください。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、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、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。また、当社製品および技術を、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル（無人航空機を含みます。）の開発、設計、製造、使用もしくは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、かつ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。
   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. お客様の転売、貸与等により、本書（本ご注意書きを含みます。）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、当社は一切その責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
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. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。

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。

(Rev.3.0-1　2016.11)

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：https://www.renesas.com/contact/