AVR280 : USBホストCDC実演

1. 序説

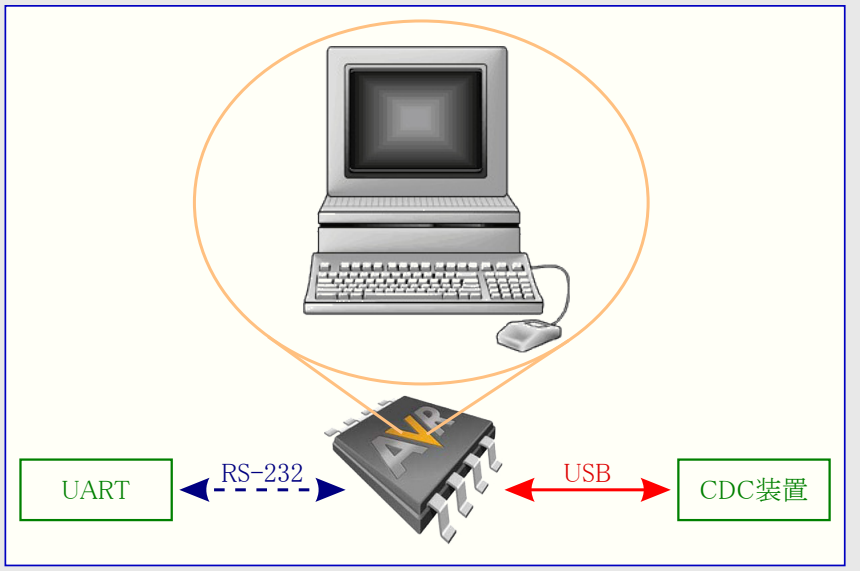
RS232インターフェースはUSBインターフェースに置き換えられ、新しい世代のPCから無くなっています。この変更に従うために、UARTインターフェースに基く応用はUSBへ移転しなければなりません。USBへの移転はPCと装置の両方の側での大変な開発を意味し得ます。この開発を避けるために、Atmelは以下の利点を持つCDCクラス(Communication Device Class:通信装置クラス)に基く解決法を提供します。

- CDC装置応用(AVR272応用記述をご覧ください) : PCに接続することができるUSB装置のようなもので仮想シリアル ポートとして表れます。
- CDCホスト応用 : この応用はPCを置き換えて、CDC装置を迎え入れ、簡単で強力な通信を許します。

本資料の狙いはSTK525またはUSBKEYスタータ キットを使ってホストCDC応用を開始して実装する方法を記述することで、そして最後に2つのPC間で双USB-UASRブリッジの簡単な例を紹介します。

AVR USBソフトウェア構成(http://www.atmel.com)とCDC仕様(http://www.usb.org)の熟知が仮定されています。

図1-1. ホストCDC応用



2. 動作の理屈

2.1. CDC構成設定

2.1.0.1. USB

CDCクラス構成設定は以下のような2つのインターフェースを含みます。

■ データ インターフェース : データ交換のために、転送方向当たり1つで、2つのパイプ(代表的に大量(バルク))から成ります。

■ 通信インターフェース : 装置の動作状態を管理するための要求送出、それと事象通知に使われます。これは2つの要素間で共用されます。

- 管理要素はエンドポイント0から成り、標準と特定の要求を通して装置の構成設定と制御を行います。
- 任意選択通知要素は割り込み(また大量(バルク)かもしれません)エンドポイントから成り、ホストに事象を通知するために装置によって使われます。転送されるメッセージは可変長領域が後続する、8バイトの先頭部として構成されます。

結論として、代表的なCDC応用は既定制御エンドポイントに加えて2つのパイプが必要です。しかし、(既定ドライバを使って)任意選択の通知要素を含む場合にだけ、CDC装置が標準PCによって受け入れられることに注意してください。

8ビット AVR® マイクロ コントローラ

応用記述

Rev. 7727A-09/07, 7727AJ2-05/21

2.1.0.2. 模式

CDCｸﾗｽ仕様は多数の通信模式を導入します。各模式は実装されるｲﾝﾀｰﾌｪｰｽの形式によって特性付けられ、命令や規約が支援されます。これは特別な規約層(例えばV42bis)を使うかもしれないUSB変復調器やUSB電話などさえとも通信するために、特に使われます。

しかし、このような模式ついて関心を持たず、只2つの応用間でﾃﾞｰﾀを交換することを欲する使用者に対して、これらは無関係です。

この応用記述はこれらの模式と関連する命令や要求を読者に短く紹介します。けれども、それの基本的な到着点はﾎｽﾄCDCﾌｧｰﾑｳｪｱ低位基本構造を説明することと、評価例とで最も簡単な構成設定を実演することです。

2.2. 転送

2.2.1. ﾃﾞｰﾀ転送

2.2.1.1. 生ﾃﾞｰﾀ

一旦ﾎｽﾄによって装置が列挙(接続認識)されてしまうと、応用間でﾃﾞｰﾀを簡単に送ることができます。

- 装置は何れかの量のﾃﾞｰﾀでそのINｴﾝﾄﾞﾎﾟｲﾝﾄを満たすことができます。ﾎｽﾄは可能な限り度々INｴﾝﾄﾞﾎﾟｲﾝﾄをﾎﾟｰﾘﾝｸﾞし、有り得るどんなﾃﾞｰﾀもそのINﾊﾟｲﾌﾟで取得することを引き受けます。
- ﾎｽﾄは何れかの量のﾃﾞｰﾀでそのOUTﾊﾟｲﾌﾟを満たすことができます。装置はﾎｽﾄが送信を完了すると直ぐにそのOUTｴﾝﾄﾞﾎﾟｲﾝﾄでﾊﾟｹｯﾄを受信します。

それはUARTの容易さを除き、USBの能力でﾃﾞｰﾀを交換する2つの基本的な応用に対して最も簡単な方法です。

- 大量(ﾊﾞﾙｸ)転送法の使用は原理的に全速(Full-speed)に於いて最大1.2Mﾊﾞｲﾄ/秒に達せます。
- USBは低減された容量(と低費用)の接続性

これはUART-USBﾌﾞﾘｯｼﾞの最も簡単な使用の場合です。このような構成設定では、一旦装置が列挙(接続認識)されて構成設定されると、管理ｲﾝﾀｰﾌｪｰｽは脇に置かれて、ﾃﾞｰﾀ ｲﾝﾀｰﾌｪｰｽを通して全てのﾃﾞｰﾀが交換されます。

2.2.1.2. ｶﾌﾟｾﾙ化ﾃﾞｰﾀ

より高い水準の応用に関し、CDCｸﾗｽは規約ﾃﾞｰﾀ包みでﾊﾟｹｯﾄを扱うためにﾃﾞｰﾀ ｶﾌﾟｾﾙ化の形式を定義します。

この形態は簡単な実演の目標の向こうになるため、ここでは検討されません。より多くの情報については「CDCｸﾗｽ仕様1.1版」の11頁で表1.をご覧ください。

2.2.2. 通信管理

ｸﾗｽのこの部分は簡単なﾃﾞｰﾀ交換応用に対して必須ではありません。

2.2.2.1. 管理要素

既定制御ｴﾝﾄﾞﾎﾟｲﾝﾄを通して、ﾎｽﾄは装置に双方向の要求を送ることができます。これらの要求の形式はUSB仕様2.0で定義される配置に従います。

図2-1. 管理要求ﾊﾟｹｯﾄ形式

bmRequestType	bRequest	wValue	wIndex	wLength	ﾃﾞｰﾀ

次の2つ要求形式が考慮され得ます。

■ 装置を特定の状態または構成設定に設定する、または情報を送るための要求
- SET_LINE_CODING：この要求はﾊﾟﾘﾃｨ、ﾎﾞｰﾚｰﾄと他のいくつかのﾊﾟﾗﾒｰﾀで装置を構成設定します。
- SET_HOOK_STATE：この要求は装置の信号線を特定の状態(ｵﾝﾌｯｸ、ｵﾌﾌｯｸ、詮索)に置きます。
- SEND_ENCAPSULATED_COMMAND：この要求は特定のｶﾌﾟｾﾙ化規約内でﾊﾟｹｯﾄを送ります。
- その他(CDCｸﾗｽ仕様の6.1.項をご覧ください。)

■ 装置の状態または構成設定、他の何かの情報を得るための要求
- GET_LINE_CODING：この要求は装置構成設定を取得します。
- GET_ENCAPSULATED_RESPONSE：この要求は特定のｶﾌﾟｾﾙ化規約内でﾊﾟｹｯﾄを取得します。
- その他(CDCｸﾗｽ仕様の6.1.項をご覧ください。)

2.2.2.2. 通知要素

この任意選択ﾊﾟｲﾌﾟ(割り込みIN転送)を通して、装置はﾎｽﾄへ特別な事象通知を転送することができます。

図2-2. 通知ﾊﾟｹｯﾄ形式

bmRequestType	bNotification	wValue	wIndex	wLength	任意ﾃﾞｰﾀ

いくつかの例:

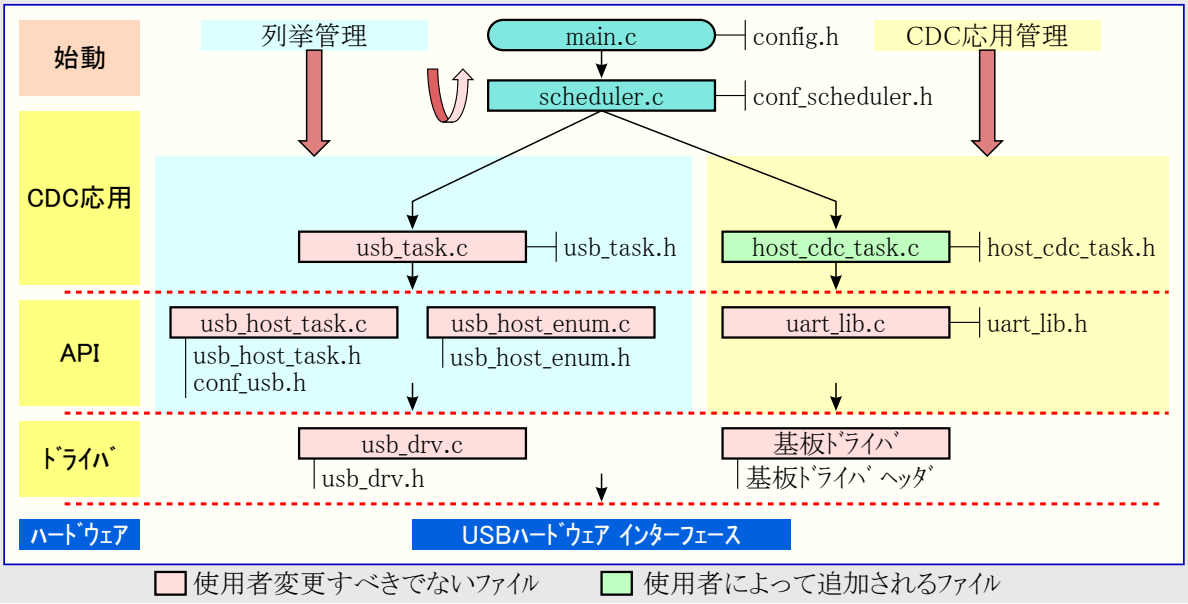
- NETWORK_CONNECTION：装置は(例えば、変更後の)接続状態についてﾎｽﾄに通知します。
- RESPONSE_AVAILABLE：装置は応用が可能なことをﾎｽﾄに通知します。ﾎｽﾄはこの応答を得るために管理要素を通してGET_ENCAPSULATED_RESPONSE要求を送ることが必要です。
- その他(CDCｸﾗｽ仕様の6.3.項をご覧ください。)

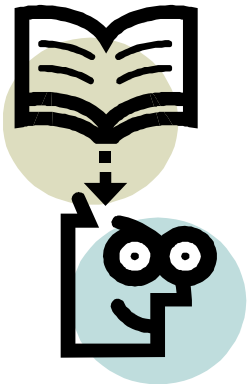
通信管理についてのより多くの詳細と情報に関しては「CDCｸﾗｽ仕様」の6章を読んでください。

3. Atmelｿﾌﾄｳｪｱ基本設計

下は動作に関して必要とされる全てのﾌｧｲﾙが表れるﾎｽﾄCDCﾌｧｰﾑｳｪｱの基本設計の概要です。

図3-1. ﾎｽﾄCDCﾌｧｰﾑｳｪｱ基本設計





CDCの管理は"host_cdc_task.c"ﾌｧｲﾙで行われます。計画機構によって周期的に呼び出される主関数は以下のような3つの主な操作を扱うhost_cdc_task(void)です。

- ■ 接続(と切断の)検出
  - ・それのｲﾝﾀｰﾌｪｰｽ記述子に従ったどれかの新しい装置の受け入れまたは排除
  - ・CDCｿﾌﾄｳｪｱ ﾁｬﾈﾙとの物理的なﾊﾟｲﾌﾟの連携
- ■ ﾃﾞｰﾀ転送
  - ・やって来るﾃﾞｰﾀの流れ検査
  - ・使用者応用によって要求された場合のﾃﾞｰﾀ送出
- ■ 管理転送
  - ・事情通知が受信されたかどうかの検査
  - ・使用者は既定制御ｴﾝﾄﾞﾎﾟｲﾝﾄを通して要求を送ることもできます。

3.1. 列挙(接続認識)

装置がﾎｽﾄに繋がる時に列挙(接続認識)が始まります。USBﾎｽﾄ ﾌｧｰﾑｳｪｱの低位ﾀｽｸが("conf_usb_h"ﾌｧｲﾙで定義された)支援ｲﾝﾀｰﾌｪｰｽの一覧と装置記述子を比較することによって装置ｲﾝﾀｰﾌｪｰｽが受け入れられたなら、その後にhost_cdc_task()は接続通知(Is_new_device_connection_event()ﾏｸﾛがTRUEを返すか)を見ます。

現在受け入れられたｲﾝﾀｰﾌｪｰｽ数がGet_nb_supported_interface()関数によって返されます。各ｲﾝﾀｰﾌｪｰｽ番号に対して、ﾌﾟﾛｸﾞﾗﾑはGet_class(i)、Get_subclass(i)、Get_protocol(i)のﾏｸﾛによって装置のｸﾗｽ、補助ｸﾗｽ、規約符号にｱｸｾｽします。

最初に、ﾌﾟﾛｸﾞﾗﾑは接続した装置がCDCﾃﾞｰﾀ ｲﾝﾀｰﾌｪｰｽを持つかどうかを調べます。装置ｲﾝﾀｰﾌｪｰｽが受け入れられたなら、接続関数はより共通的な名前のpipe_cdc_data_bulkinとpipe_cdc_data_bulkoutとそれらを連携するために、どのﾊﾟｲﾌﾟがINでどのﾊﾟｲﾌﾟがOUTかを検索します。その後、ﾊﾟｲﾌﾟでのﾃﾞｰﾀ受信を許可するためにINﾊﾟｲﾌﾟは凍結を解除されます。

その後、ﾌﾟﾛｸﾞﾗﾑは接続した装置がCDC通信ｲﾝﾀｰﾌｪｰｽ(通知要素)を持つかどうかを調べます。もし持つなら、ﾊﾟｲﾌﾟ ｱﾄﾞﾚｽも共通名のpipe_cdc_comm_intで格納されます。使用者はﾗﾍﾞﾙのCDC_USE_MANAGEMENT_INTERFACEを定義することによって管理ｲﾝﾀｰﾌｪｰｽ検出を許可します。管理要求が使われるかどうかにそれが必要とされるので、ｲﾝﾀｰﾌｪｰｽ番号も変数cdc_interface_commに格納されます。

一旦接続が受け入れられると、cdc_connectedﾌﾗｸﾞが"1"に設定され、使用者応用によってﾊﾟｲﾌﾟを使うことができます。

下は対応する関数のｺｰﾄﾞです。

ｺｰﾄﾞ3-1. CDC装置接続検出

```
if(Is_new_device_connection_event())        // 装置接続
   {
      for(i=0;i<Get_nb_supported_interface();i++)
      {
         // ﾃﾞｰﾀ ｲﾝﾀｰﾌｪｰｽ
         if((Get_class(i)==CDC_DATA_CLASS) && (Get_protocol(i)==CDC_DATA_PROTOCOL))
         {
            cdc_connected=1;
            Host_enable_sof_interrupt();
            LOG_STR_CODE(log_cdc_connect);
            if(Is_ep_addr_in(Get_ep_addr(i,0)))
            {  // そうなら、それをCDCﾃﾞｰﾀINﾊﾟｲﾌﾟと連携
               pipe_cdc_data_bulkin = host_get_hwd_pipe_nb(Get_ep_addr(i,0));
               pipe_cdc_data_bulkout = host_get_hwd_pipe_nb(Get_ep_addr(i,1));
            }
            else
            {  // そうでなければ、逆で連携
               pipe_cdc_data_bulkin = host_get_hwd_pipe_nb(Get_ep_addr(i,1));
               pipe_cdc_data_bulkout = host_get_hwd_pipe_nb(Get_ep_addr(i,0));
            }
            Host_select_pipe(PIPE_CDC_DATA_IN);
            Host_continuous_in_mode();
            Host_unfreeze_pipe();
            break;
         }
         // 管理ｲﾝﾀｰﾌｪｰｽ
#ifdef CDC_USE_MANAGEMENT_INTERFACE
         if(Get_class(i)==CDC_COMM_CLASS && Get_protocol(i)==CDC_COMM_PROTOCOL)
         {
            cdc_interface_comm = i;          // ｲﾝﾀｰﾌｪｰｽ番号格納
            pipe_cdc_comm_int = host_get_hwd_pipe_nb(Get_ep_addr(i,0));
            Host_select_pipe(PIPE_CDC_COMM);
            Host_continuous_in_mode();
            Host_unfreeze_pipe();
         }
#endif
      }
   }
```

3.2. ﾃﾞｰﾀ転送

ﾃﾞｰﾀ転送は実装が非常に容易です。

3.2.1. ﾃﾞｰﾀ受信

CDC装置が接続された場合、ﾌﾟﾛｸﾞﾗﾑはﾊﾟｲﾌﾟが新しいﾃﾞｰﾀを受信したかどうかを可能な限り度々(関数が移行される度毎に)調べます。

現在のﾌｧｰﾑｳｪｱはﾊﾟｲﾌﾟ ﾃﾞｰﾀで以下のような2つの操作を許します。

■ ﾃﾞｰﾀは配列に格納されます。

- cdc_stream_in_array[CDC_STREAM_IN_SIZE]配列はﾌｧｰﾑｳｪｱによって到着ﾃﾞｰﾀで満たされます。
- rx_counter変数は配列内で書かれるべ次ﾊﾞｲﾄの位置を示します(0に初期化され、この緩衝部はrx_counterがCDC_STREAM_IN_SIZEと等しい時が満杯です)。
- 使用者ﾌｧｰﾑｳｪｱが配列からﾃﾞｰﾀを読む時に、ﾃﾞｰﾀを引き継ぐのに別の指標変数を使うか、または一度に配列全てを読んでrx_counterを解除するかのどちらかでなければなりません。配列からの各ﾊﾞｲﾄ読み込みに対して使用者がrx_counterを減ら(-1)して、配列全体を読む前にその関数を抜けた場合、問題が発生するでしょう(例えば次のﾊﾟｲﾌﾟ受信での不正ﾃﾞｰﾀ)。

■ ﾃﾞｰﾀはUARTに送られます(USB-UARTﾌﾞﾘｯｼﾞ接続)。

- 全てのﾊﾞｲﾄがUARTに送られます。
- これは次のﾃﾞｰﾀへの引継ぎを試みる前に送信されるべき各ﾊﾞｲﾄを待つ閉塞(完了復帰型)関数です(この限度は容易に上げることができます)。
- この動作形態はCDC_USE_UARTﾗﾍﾞﾙを定義することによって許可されます。

ｺｰﾄﾞ3-2. 装置からのﾃﾞｰﾀ読み込み

```
Host_select_pipe(PIPE_CDC_DATA_IN);
   if (Is_host_in_received() && (Is_host_stall()==FALSE))
   {
#ifdef CDC_USE_UART
      while (Host_data_length_U8() != 0)
      {
         uart_putchar(Host_read_byte());
      }
      Host_ack_in_received();        // ﾊﾟｲﾌﾟは空
      Host_send_in();                // 更なるﾃﾞｰﾀ受信の準備可
#else
      while ((rx_counter != CDC_STREAM_IN_SIZE) && (Host_data_length_U8() != 0))
      {
         cdc_stream_in_array[rx_counter] = Host_read_byte();
         rx_counter++;
      }
      if (Host_data_length_U8() == 0)
      {
         Host_ack_in_received();     // ﾊﾟｲﾌﾟは空
         Host_send_in();             // 更なるﾃﾞｰﾀ受信の準備可
      }
#endif
   }
```

これらの操作は評価目的用に実装されています。使用者は現在の関数を"現状そのまま"として使うことができますが、ﾊﾟｲﾌﾟへ直接引き継ぐ、自身のﾃﾞｰﾀ処理部(または例えばﾊﾟｹｯﾄ包み)を実装することも自由です。

3.2.2. データ送信

動作の理屈はデータ受信段よりも非常に簡単です。送信されるべきデータが最初にcdc_stream_out_array[CDC_STREAM_OUT_SIZE]配列へ格納されます。tx_counter全体変数が現在格納されたバイト数、その結果として格納されるべき次バイトの位置を示します。

配列に対して2つの有り得る供給元があります。

- 使用者プログラム：使用者特定ファームウェアはデータを配列に格納し、各バイト書き込みに対してtx_counterを増やすことができます(データが全く格納されていない時に0へ初期化されます)。
- UART：ラベルCDC_USE_UARTが定義されている場合、配列はUARTで受信された新しいバイトの各々で満たされます。

更に、OUTパイプに転送されるべき配列データに関して2つの有り得る状況があります。

- 緩衝部満杯：緩衝部が満ちる(tx_counter=CDC_STREAM_OUT_SIZE)と直ぐに、全ての配列データがパイプに転送されます。
- 時間超過：cdc_cpt_sof時間計数変数はsof_action()関数に於いて各USBフレーム開始(1ms毎の)割り込みで増加されます。この計数器が使用者定義のCDC_NB_MS_BEFORE_FLUSHに達するか、または越える時に配列が空でなければ、全ての緩衝部データがパイプに転送されます。

USBのパイプ送信は cdc_pipe_out_usb_flush()関数によって保証されます。

コード3-3. 装置へのデータ送出

```
#ifdef CDC_USE_UART
   // USBに格納されるべきUARTの新規データ有無検査
   if (uart_test_hit() && (tx_counter != CDC_STREAM_OUT_SIZE))
   {
      cdc_stream_out_array[tx_counter] = uart_getchar();
      tx_counter++;
   }
#endif

   // パイプ破棄の必要(緩衝部満杯または時間超過)の有無検査必要
   if(((cdc_cpt_sof>=CDC_NB_MS_BEFORE_FLUSH) && (tx_counter!=0)) || (tx_counter == CDC_STREAM_OUT_SIZE))
   {
      cdc_cpt_sof=0;
      cdc_pipe_out_usb_flush();
   }
```

コード3-4. 付加関数

```
void cdc_pipe_out_usb_flush (void)
{
   Host_select_pipe(PIPE_CDC_DATA_IN);      // 大量(バルク)INは凍結されなければならず、
   Host_freeze_pipe();                      // 大量(バルク)OUTは送られないかもしれません。
   if (PIPE_GOOD == host_send_data(PIPE_CDC_DATA_OUT, tx_counter, cdc_stream_out_array))
   {
      tx_counter = 0;                // フレームが送られない場合、次の時に再び試みます(データは失われません)。
   }
   Host_select_pipe(PIPE_CDC_DATA_IN);
   Host_unfreeze_pipe();
}

void sof_action(void)
{
   cdc_cpt_sof++;
}
```

3.3. 通信管理

3.3.1. 管理要素

CDCｸﾗｽ仕様で定義されたCDC特定要求はｴﾝﾄﾞﾎﾟｲﾝﾄ0を通して転送されます。これらの要求は以下の形式で定義することができます。

ｺｰﾄﾞ3-5. 管理要求配置

```
#define host_cdc_get_line_coding()   (usb_request.bmRequestType    = BM_REQUEST_TYPE,¥
                                      usb_request.bRequest         = B_REQUEST,¥
                                      usb_request.wValue           = W_VALUE,¥
                                      usb_request.wIndex           = W_INDEX,¥
                                      usb_request.wLength          = W_LENGTH,¥
                                      usb_request.uncomplete_read  = FALSE,¥
                                      host_send_control(data_stage))
```

ここで赤文字(**訳注**:原文は太斜体)のﾊﾟﾗﾒｰﾀは特定の要求に合うﾊﾟﾗﾒｰﾀによって置き換えられなければなりません。

W_INDEX領域は代表的に装置の(許可されていれば)管理ｲﾝﾀｰﾌｪｰｽに割り当てられた"cdc_interface_comm"と等価です。

W_LENGTH領域は要求で送信されるべきﾃﾞｰﾀ ﾊﾞｲﾄ数を含みます。

(送信または受信されるべき)ﾃﾞｰﾀ ﾊﾞｲﾄは"data_stage"配列に格納されます。

使用者は"host_cdc_task.h"ﾌｧｲﾙを編集することによって容易に他の要求を追加することができます。けれども、それらのいくつかは既にAtmelﾎｽﾄCDCﾌｧｰﾑｳｪｱに統合されています。

3.3.1.1. ｶﾌﾟｾﾙ化要求

これらの要求は特定規約に従ってｶﾌﾟｾﾙ化された特定要求を送るのに使われます。

表3-1. SEND_ENCAPSULATED_COMMAND要求

bmRequestType	bRequest	wValue	wIndex	wLength	ﾃﾞｰﾀ
0 01 00001	$00	$00	ｲﾝﾀｰﾌｪｰｽ	この要求に関連するﾃﾞｰﾀの量	制御規約に基く命令

表3-2. GET_ENCAPSULATED_RESPONSE要求

bmRequestType	bRequest	wValue	wIndex	wLength	ﾃﾞｰﾀ
1 01 00001	$01	$00	ｲﾝﾀｰﾌｪｰｽ	この受信に関連するﾃﾞｰﾀの量	ﾃﾞｰﾀに依存する規約

3.3.1.2. 通信ﾊﾟﾗﾒｰﾀ要求

これらの要求はそれのUART通信のためにCDC装置へ(またはから)形態を設定(または取得)するのに使われます。

表3-3. SET_LINE_CODING要求

bmRequestType	bRequest	wValue	wIndex	wLength	ﾃﾞｰﾀ
0 01 00001	$20	$00	ｲﾝﾀｰﾌｪｰｽ	$07	信号線符号化構造体

表3-4. GET_LINE_CODING要求

bmRequestType	bRequest	wValue	wIndex	wLength	ﾃﾞｰﾀ
1 01 00001	$21	$00	ｲﾝﾀｰﾌｪｰｽ	$07	信号線符号化構造体

表3-5. 信号線符号化構造体

変位(ｵﾌｾｯﾄ)	領域	ﾊﾞｲﾄ数	値種別	説明
0	dwDTERate	4	数値	秒当たりのﾋﾞｯﾄ数でのﾃﾞｰﾀ端末速度
4	bCharFormat	1	数値	停止ﾋﾞｯﾄ: 0=1停止ﾋﾞｯﾄ、1=1.5停止ﾋﾞｯﾄ、2=2停止ﾋﾞｯﾄ
5	bParityType	1	数値	ﾊﾟﾘﾃｨ: 0=なし、1=奇数、2=偶数、3=ﾏｰｸ1(1)、4=ｽﾍﾟｰｽ(0)
6	bDataBits	1	数値	ﾃﾞｰﾀ ﾋﾞｯﾄ数(5,6,7,8,または16)

3.3.2. 通知要素

追加のINｴﾝﾄﾞﾎﾟｲﾝﾄから成る通知要素が装置からの事象通知を受け取ります。

現在のAtmelのﾎｽﾄCDCｿﾌﾄｳｪｱはこのﾊﾟｲﾌﾟに対するﾊﾟｹｯﾄ包みを含みません。しかし使用者の通知処理部を迎えるための場所は準備されています。

このｲﾝﾀｰﾌｪｰｽによって提供される機能についてのより多くの情報に関してはCDCｸﾗｽ仕様を読んでください。

ｺｰﾄﾞ3-5. 通知ﾊﾟｲﾌﾟ処理部雛形

```
Host_select_pipe(PIPE_CDC_COMM);
   if (Is_host_in_received())
   {
      // 装置によって送られた通知ﾒｯｾｰｼﾞをここで扱います。
      // 通知ﾒｯｾｰｼﾞは以下の構造体を持ちます。:
      //  bmRequestType, bNotification, wValue, wIndex, wLength, Data (wLengthはData領域のﾊﾞｲﾄ数)

      //   - NETWORK_CONNECTION：装置がﾈｯﾄﾜｰｸに接続されたことを示します。
      //   - RESPONSE_AVAILABLE：装置がｶﾌﾟｾﾙ化された応答を持つことを示します(ﾎｽﾄ要求待機)。
      //   - SERIAL_STATE：装置のUARTの状態(異常、搬送とその他の信号)を示します。
      //   - その他 ～

      // ～ さて ～ 只ｺｰﾃﾞｨﾝｸﾞしてください。～
      Host_ack_in_received();
      Host_send_in();
   }
```

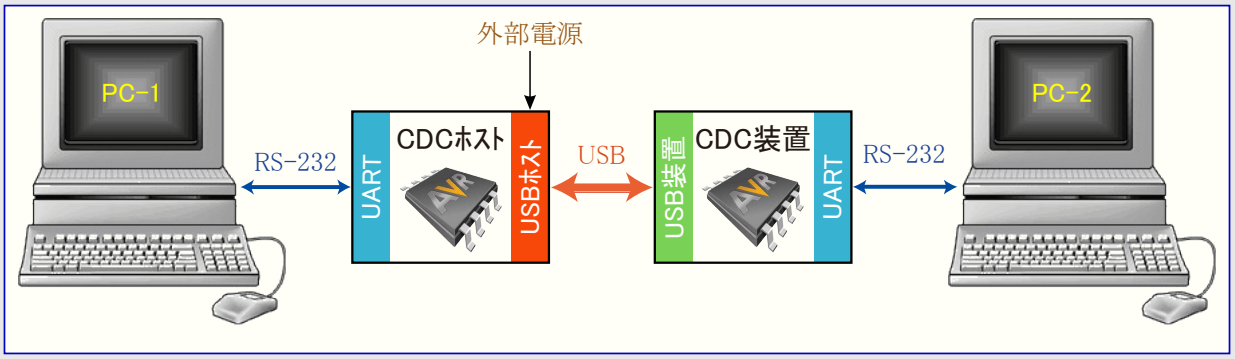
4. 例

4.1. 概要

この理屈の全てが複雑と思われるかもしれないので、ここでは評価目的(双USB-UARTﾌﾞﾘｯｼﾞ)でCDCｸﾗｽ実装について素早い評価と使用者の応援を許す簡単な例です。

この構成設定では、CDCﾎｽﾄ応用がPC-1のｼﾘｱﾙ ﾎﾟｰﾄに接続されたUARTを持ちます。CDC装置も同じPC-1の別のｼﾘｱﾙ ﾎﾟｰﾄか、または別のPC-2のｼﾘｱﾙ ﾎﾟｰﾄに接続されたUARTを持ちます。両方のCDC応用がUSB接続を通して共に接続されます。工業的または消費者の応用として総合的に役立たないこの応用は、単にCDCﾃﾞｰﾀ転送の構造を示します。

図4-1. 双USB-UARTﾌﾞﾘｯｼﾞ評価例



注: それらがUSBﾎﾟｰﾄだけ利用可能なら、PCでRS232を提供するために追加のCDC装置基板を使うことができます。

CDC装置はAtmelのｳｪﾌﾞｻｲﾄ(http://www.atmel.com)で入手可能なｿﾌﾄｳｪｱ1式を用いてどのAVR USBにも実装することができます。CDCﾎｽﾄはこれもｲﾝﾀｰﾈｯﾄで入手可能な対応するｿﾌﾄｳｪｱ1式を使います。

4.2. ﾊｰﾄﾞｳｪｱ

両ｿﾌﾄｳｪｱ1式は入手可能なｽﾀｰﾀ ｷｯﾄで走行することができます。著述時点に於いて、ﾎｽﾄCDC1式は(AT90USB647/1287を特徴とする)STK525またはUSBKEY製品で走行することができ、装置CDC1式はSTK525、USBKEY(AT90USB64x/128x)またはSTK526(AT90USB82/162)で走行することができます。

USB装置基板は最も簡単な動作のためにﾊﾞｽ給電形態に構成設定されるべきです。USBﾎｽﾄ基板はUSB装置基板へ電力を供給するために、自己給電(外部電源)で構成設定されなければなりません。

4.3. ソフトウェア

4.3.1. マイクロ コントローラ

4.3.1.1. 動作説明

この例は管理インターフェースを通してデータを交換しません。けれども、他のCDC応用との共通性を保証するために、このインターフェースが実装されるかもしれません。

一旦列挙(接続認識)されると、PC-1から受信した全てのバイトがUARTからCDCホストのOUTパイプに転送され、その後にCDC装置のOUTエンドポイントで受信され、そして最後にUARTを通してPC-2に転送されます。PC-2によって送られたバイトは逆方向で進んでPC-1のシリアル ポート緩衝部に到着します。

4.3.1.2. 構成設定

正しい動作を保証するために各マイクロ コントローラに於いていくつかのパラメータが定義されなければなりません。ソフトウェア1式は変更される必要がなく、それは"現状そのまま"として動作し、下で与えられる値で構成設定されます。

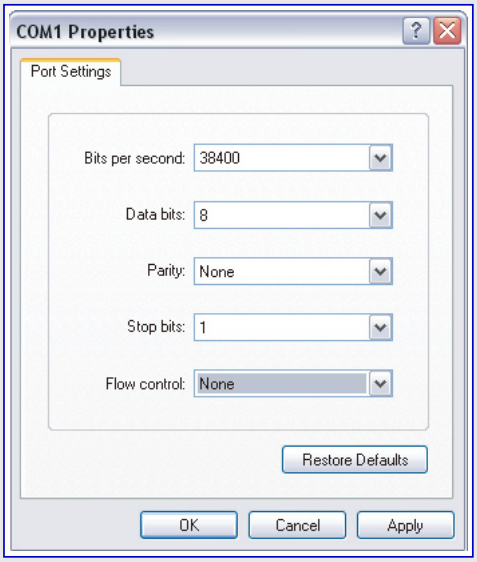
- UARTボーレート："config.h"ファイルで、望むボーレートにラベルBAUDRATEを定義してください。1式は38.4kbps(BAUDRATE=38400)の既定値でやって来ます。
- "conf_usb.h"ファイルでのUSB構成設定：
  - CLASS_SUBCLASS_PROTOCL配列はCDCデータ インターフェースとCDC通信インターフェースの値を含まなければなりません。
  - Host_sof_action()はsof_action()関数(これの原型も宣言されなければなりません)へリンクされなければなりません。
- ホストCDC構成設定："config.h"ファイルで、または"host_cdc_task.c"ファイルによってアクセルし易いどれかのヘッダ ファイルは以下の値またはラベルを定義しなければなりません。
  - CDC_USE_MANAGEMENT_INTERFACEは(例えこのインターフェースを通してデータが全く交換されなくても)定義されなければなりません。
  - CDC_USE_UARTは(USB-UARTブリッジ機能を許可するために)定義されなければなりません。
  - CDC_NB_MS_BEFORE_FLUSHは人間制御応用に関してあまり重要ではなく、故にこれは既定による$05に設定することができます(この値の影響は応用に依存するかもしれません)。
  - CDC_STREAM_OUT_SIZEとCDC_STREAM_IN_SIZEは予期するデータ速度に依存します。より高い予期されるデータ速度(実際の秒当たりで送られるかもしれない文字数)はより高いそれら2つの値になります。しかし、(データ損失を防ぐために)データ パイプまたはエンドポイントの容量を超えないでください。既定値は16($10)です。

4.3.2. PCのシリアル ポート

PCのどれかのシリアル ポートを手軽に利用するため、あなたは端末を動かすかもしれません。Windows下でハイパーターミナルを開始することができます(プログラム⇒アクセサリ⇒通信)。最初に応用(ホストまたは装置)が接続されるCOMポートを選択しなければなりません。そしてポートの詳細な構成設定が正しく入力されなければなりません。無変更のAtmelの1式を使うなら、右の形態を設定しなければなりません。

その後にシリアル ポート開始で送る文字の入力や、ポートで受信された文字を見ることができる端末ダイアログ画面が現れます。

図4-2. ハイパーターミナル構成設定



5. 結び

結論として、変復調器、移動電話、LANインターフェースのような装置を支援するために、CDCクラスはかなり多くの異なる構成設定を網羅し、多数の通信規格に合う広範囲な仕様です。

このような装置の実装に必要とされる作業負担が非常に重要で有り得るとは言え、データ送信の基本的な実装は容易に得やすくなります。この応用記述は今日の技術を用いて、使用が簡単で信頼に足る強力な通信方法を構成するのに、AtmelのAVR USB製品のホスト能力を使うことを望む人々を手助けするために作成されました。USB-UARTブリッジ(または同様)のような応用を容易に作成することができます。

6. 関連資料

・AVR USB製品ﾃﾞｰﾀｼｰﾄ
・USBｿﾌﾄｳｪｱ ﾌﾚｰﾑﾜｰｸ
・CDC装置応用記述(AVR272)
・USBCDCｸﾗｽ仕様

以下で入手できます。

・http://www.atmel.com
・http://www.usb.com

本社

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131, USA
TEL 1(408) 441-0311
FAX 1(408) 487-2600

国外営業拠点

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
TEL (852) 2245-6100
FAX (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-Yvelines
Cedex
France
TEL (33) 1-30-60-70-00
FAX (33) 1-30-60-71-11

Atmel Japan
104-0033 東京都中央区
新川1-24-8
東熱新川ﾋﾞﾙ 9F
ｱﾄﾒﾙ ｼﾞｬﾊﾟﾝ株式会社
TEL (81) 03-3523-3551
FAX (81) 03-3523-7581

製造拠点

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
TEL 1(408) 441-0311
FAX 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
TEL 1(408) 441-0311
FAX 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3
France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR
Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn
Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Biometrics
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex
France
TEL (33) 4-76-58-47-50
FAX (33) 4-76-58-47-60

文献請求
www.atmel.com/literature

本応用記述はAtmelのAVR280応用記述(doc7727.pdf Rev.7727A-09/07)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意訳されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。

青字の部分はﾘﾝｸとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。