USB HID Host Library

2015-12-04

Content

Introduction

This is a library for making an **USB host** capable of reading and writing **64 byte packets** from/to generic[1] **USB HID devices**. At this moment only a version for PIC24 is available.

Usage in a program

Initialisation and test for readiness

The initialisation of the library and test for readiness of the USB HID device is done as follows:

```
uses USB_HOST_HID_Library, UartDebug;
...

procedure USB1Interrupt(); iv IVT_ADDR_USB1INTERRUPT;
begin
  USB_Interrupt;
end;

begin
  { Main program }

  InitMain;

  InitUsb;                              // <--- initialisation of the library
  repeat
  until USB_HID_Device_Ready or         // <--- test for readyness of the USB device
        (USB_Error > 0);

  if USB_HID_Device_Ready then          // the USB device is ready
  begin
```

[1] devices with a 64 byte read/write HID report

```
    uart_write_line('');
    uart_write_line('HID device Ready');
    uart_write_line('');

    LatA.0 := 1;                        // signal readyness (example)
    uart_write_line('');
  end
  else
  begin                                   // the USB device gives an error
    uart_write_line_word_hex(USB_Error);
    while true do; // stop all processing
  end;
```

The above example shows the error code if the device does produce an error.

As you can see the initialisation is done with the “InitUsb” routine, the readiness of the device is tested with “USB_HID_Device_Ready”. In case you do not want to block the software if “USB_HID_Device_Ready” stays false, you should also test “USB_Error”. If it becomes > 0 then you can stop waiting for “USB_HID_Device_Ready”, see the example above.

Reading and Writing packets

With this library a packet is **always 64 bytes** long (= “generic HID device”)

Reading and writing a packet is done as follows:

```
var Buff      : array[64] of byte;
      Success : boolean;
...
...
Success := USB_HID_Read(Buff); // a packet from the HID device is read into Buff

if Success then // a packet was received from the HID device
begin
  // process the buffer content here
end;
...
...
// define the buffer content here
Success := USB_HID_Write(Buff); // the content of Buff is written to the device

if not success then // the transmission failed,
                    // the device was not ready to accept a packet
begin
  // perhaps retry at some later time
end;
...
...
```

Both routines above return “true” if success, “false” if failiure.

Interface of the unit

```
procedure USB_Interrupt;
// To be called from the main interrupt routine (iv IVT_ADDR_USB1INTERRUPT)

procedure InitUsb;
// To be called once in the initialisation phase of the software (or when re-trying
initialisation after device detach)

function USB_HID_Device_Ready: boolean;
// Returns TRUE if an USB HID device is attached

function USB_HID_Read(var Buff: array[64] of byte): boolean;
// Returns true if some data has arrived, false means: no data arrived.
// The arguments are: //
//   Buff:  the user defined receive buffer (always 64 bytes in size).

function USB_HID_Write(var Buff: array[64] of byte): boolean;
// Returns success as true, false means: try later again (USB sendbuffer was still
busy).
// The arguments are: //
//   Buff:  the user defined send buffer (always 64 bytes in size).

var USB_Error: word;
// Returns the USB Error
// The USB_Error signals one error per bit (see below for the possible values)


const // Error constants

     // USB_Error constants
      USB_DEVICE_DESCRIPTOR_ERROR              = $001; // bit 0
      USB_CONFIG_DESCRIPTOR_ERROR              = $002; // bit 1
      USB_INTERFACE_DESCRIPTOR_ERROR           = $004; // bit 2
      USB_INTERFACE_DESCRIPTOR_CLASS_ERROR     = $008; // bit 3
      USB_INTERFACE_DESCRIPTOR_SUBCLASS_ERROR  = $010; // bit 4
      USB_INTERFACE_DESCRIPTOR_PROTOCOL_ERROR  = $020; // bit 5
      USB_ENDPOINT_DESCRIPTOR_ERROR            = $040; // bit 6
      USB_ENDPOINT_DESCRIPTOR_ATTRIBUTES_ERROR = $080; // bit 7
      USB_ENDPOINT_DESCRIPTOR_PACKETSIZE_ERROR = $100; // bit 8
```

[end of document]