1 Introduction

The present document gives information to use the ATMEL USB HID DLL functions.
There is also some simple code that show to kind of implementation :

1st : UsbHidSmallDemoCode

This is a simple console application which connect to a device using VID=0x03EB and PID=0x2013.
Once connected to a device, LEDs are blinking until button is pressed on the board.
When information is read from the device, the application close the device and exit.

2nd : UsbHidDemoCode :

This is a Dialog Box application which can be used to set LEDs present on the board, capture button action, set board to firmware upgrade mode, change default PID and VID, detect if device as been connected or discerned.
Those demo applications has been written in Visual C++ 6.0 and Visual C++ service pack 6

The USB HID DLL can be found with its include file in directory : AtUsbHid

2 DLL function

2.1 findHidDevice

Scans the connected USB devices for a HID one matching the specified Vendor ID and Product ID.

Input

const UINT VendorID : this is the vendor ID
const UINT ProductID: this is the product ID

Output

0 if fail, more information can be found using GetLastError().
GetLastError will return :
ERROR_USB_DEVICE_NOT_FOUND if Device is cannot be found.
ERROR_USB_DEVICE_NO_CAPABILITIES if device found but capabilities cannot be retrieved.
1 if connected successfully

2.2 closeDevice

Closes the USB device and all handles.

2.3 *writeData*

Sends data to the device. Call this function when data length is smaller than or equal to the report size.
The buffer mustn't exceed the write buffer capabilities of the connected device.

Input

UCHAR* buffer : pointer to the message to be written.

Output

0 if fail , GetLastError() will return ERROR_WRITE_FAULT code
1 if message send successfully

2.4 *readData*

Reads data from the device.
Call this function when data length is smaller than or equal to the report size.

Input

UCHAR* buffer - Pointer to the buffer that will contain the received packet

Output

0 if no data are available.
1 if data as been written in UCHAR* buffer.

2.5 *startBootLoader*

Set the device in Boot loader mode to perform a firmware upgrade using Flip.

Output

0 if fail to set Boot loader mode
1 if Boot loader mode as been device as been successfully sent to the device mode.

2.6 *hidRegisterDeviceNotification*

Registers the device for which a window will receive notifications.

Input

HWND hWnd - Handle to a window.

Output

0 if the function fails, To get extended error information, call GetLastError
1 if the function succeeds

2.7 hidUnregisterDeviceNotification

this function closes the specified device notification handle.

Input

HWND hWnd - Handle to a window.

Output

0 if the function fails. To get extended error information, call GetLastError.
1 if the function succeeds.

2.8 isMyDeviceNotification

This function must be call to check if this is a device our device which a trigger the call of OnDeviceChange.

Input

DWORD dwData, the value given by OnDeviceChange 2nd paramerters

Output

1 if this is our device that change it connected status
0 if this is another device

3 Load the DLL in Visual C++ Application

The file AtUsbHid.h provide some macros that help to load and used function present in the ATMEL USB HID DLL.
When designing a application using the DLL you needs 1st to do the following :

- create a handler to the DLL : **HINSTANCE hLib = NULL;**
- Load the DLL using the function **hLib =LoadLibrary(AT_USB_HID_DLL);**
- Load each DLL function using **loadFuncPointers(hLib)**

Once those step has been performed without error, DLL and function as bee loaded in your application and now can been called using the macro **DYNCALL(**DllFunction()**);**

When program is ended is nice to freed DLL from memory using function **FreeLibrary(hLib);**
You must be sure that USB device as been closed before freed the DLL from memory.

4 Using automatic device connection/disconnection feature

The DLL provide function that help user to detect if device status change.
To do so you have to do the following in your Application.

- Register you application to get device change notification using : **DYNCALL(hidRegisterDeviceNotification)((m_hWnd));**
- Add the function **ON_WM_DEVICECHANGE()** in your Message Map application
- Create a Function called **OnDeviceChange(UINT nEventType, DWORD dwData)** that will be called each time a device status change.
- In the function **OnDeviceChange**, call function **DYNCALL(isMyDeviceNotification(dwData));** that will tell you if this is your device status that change. (See code demo code in UsbHidDemoCodeDlg.cpp)

When exist the application is nice to unregistered your application using function :
DYNCALL(hidUnregisterDeviceNotification(m_hWnd));

5 Using readData

As data can be sent continuously by the device. It's interesting to read data using a timer base function. This avoid to poll continuously the readData function.
To do so you have to do the following in your application.

- Add the function **ON_WM_TIMER()** in your Message Map application
- Create a function **OnTimer(UINT nIDEvent)** that will call function **DYNCALL(readData(sbuffer)**
- Then you add to set the Timer to a read period using **SetTimer(1,50,0)**; when you device is connected. The function SetTimer set the read period.
- Kill the timer using **KillTimer(1)** when your device is disconnected.