HID Class DLL packages up the calls and links to unmanaged code into one simple to use DLL. The result is just a few simple calls. The DLL even handles unplug/replug events seamlessly.
Microchip licenses to you the right to use, modify, copy and distribute Software only when embedded on a Microchip microcontroller or digital signal controller that is integrated into your product or third party product (pursuant to the sublicense terms in the accompanying license agreement).

You should refer to the license agreement accompanying this Software for additional information regarding your rights and obligations.

SOFTWARE AND DOCUMENTATION ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
(INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
Release Notes

HID Class DLL

Version 1.10, 15 April 2010

This version simplifies the naming scheme, though the old names, though deprecated, will continue to be supported. Also it adds an overloaded HIDClassInit that allows specifying the timeout duration. The original defaults to 1000mS.
Running the Demos

See "How to run the Custom HID demos" for running the Demos. This DLL is specifically used in the "Generic HID - HID DLL - PC Software" application.
HIDClass Namespace

This is namespace HIDClass.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCHPHIDClass</td>
<td>This is class HIDClass::MCHPHIDClass.</td>
</tr>
</tbody>
</table>

Symbol Reference > [HIDClass Namespace](#)
MCHPHIDClass Class

```c
ref class MCHPHIDClass;
```

Description

This is class HIDClass::MCHPHIDClass.

Class Hierarchy

Symbol Reference > HIDClass Namespace > MCHPHIDClass Class
MCHPHIDClass Methods

Symbol Reference > HIDClass Namespace > MCHPHIDClass Class > MCHPHIDClass Methods

Microchip HID Class DLL 1.10 - [15 April 2010]
Copyright © 2009, 2010 Microchip Technology, Inc. All rights reserved.
**Files**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HID class.h</td>
<td>The HID Class DLL provides a simple interface to HID Class Devices.</td>
</tr>
</tbody>
</table>

Symbol Reference > [Files](#)
HID class.h

HID Class DLL

The firmware on the corresponding device must have a generic report descriptor. See USB Device HID Custom Demos Firmware project. This hides all the low level details of bridging to unmanaged functions leaving four simple calls.

Namespaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIDClass</td>
<td>This is namespace HIDClass.</td>
</tr>
</tbody>
</table>

Symbol Reference > Files > HID class.h
Using the Library

### Topics

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Library Architecture</strong></td>
<td>HID Class DLL provides a interface to the HID class device.</td>
</tr>
</tbody>
</table>
| How the Library Works    | This DLL assumes the device we are attaching to enumerates as a generic HID device. That is the data is passed as an array of bytes (usually 64). See the firmware in the Microchip Applications Library, "USB Device - HID - Custom Demos". The DLL encapsulates the windows OS calls necessary

The DLL encapsulates the bridge to unmanaged code and interfacing to the operating system from your program.
to search though the list of connected devices and find the device with the specified Vendor and Product ID. Then once it's found it opens a pipeline to the device. It also automatically tries to reopen the device in case it has been disconnected and reconnected. This makes recovering...

**Getting Started**

The DLL is simple and easy to use. Associate the DLL with the project, declare the name space, initialize the DLL and then read and write as necessary: First Associate the DLL with the application. In Visual C++ 2008 Express Edition, select Project | Properties (Alt-F7). "Framework and References" tab. Add New Reference button, Browse Tab. Then navigate to the DLL location and add it to the project. Next, Add "using namespace **HIDClass**;" to the namespace section at the top of your code. Then you only need three calls to use the library. HIDClassInit, HIDWriteReport and HIDReadReport. Other calls...

**Using the Library**

Microchip HID Class DLL 1.10 - [15 April 2010]
Copyright © 2009, 2010 Microchip Technology, Inc. All rights reserved.

Contents | Index | Home
Library Architecture

HID Class DLL provides an interface to the HID class device.

The DLL encapsulates the bridge to unmanaged code and interfacing to the operating system from your program.

Using the Library > Library Architecture
How the Library Works

This DLL assumes the device we are attaching to enumerates as a generic HID device. That is the data is passed as an array of bytes (usually 64). See the firmware in the Microchip Applications Library, "USB Device - HID - Custom Demos".

The DLL encapsulates the windows OS calls necessary to search though the list of connected devices and find the device with the specified Vendor and Product ID. Then once it's found it opens a pipeline to the device. It also automatically tries to reopen the device in case it has been disconnected and reconnected. This makes recovering from a disconnect/reconnect event seamless to the application.

The compilation setting must be /clr, /clr:pure or /clr:safe. This dll is a .NET assembly, so it can't be used with a compilation setting that produces pure unmanaged code.
Getting Started

The DLL is simple and easy to use. Associate the DLL with the project, declare the name space, initialize the DLL and then read and write as necessary:

First Associate the DLL with the application. In Visual C++ 2008 Express Edition, select Project | Properties (Alt-F7). "Framework and References" tab. Add New Reference button, Browse Tab. Then navigate to the DLL location and add it to the project.

Next, Add "using namespace HIDClass;" to the namespace section at the top of your code.

Then you only need three calls to use the library. HIDClassInit, HIDWriteReport and HIDReadReport. Other calls are provided but are not required to communicate with device.

HID Class Init configures the DLL with the needed information that does not change: Vendor ID, Product ID, Buffer Size and Timeout (optional). This can be called from your application's constructor.

Both Read and Write functions first check to see if the device has been opened before attempting to communication with the device. If it not it opens a channel and then completes the communication. This allows seamless recovery if a device has been disconnected then reconnected.

The read and write functions both return a boolean indicating if the transfer was successful or not. Suggest testing these to verify the transfer and recover gracefully if not.

HIDClassInit (0x04D8, 0x0032, 64);
OutBuffer[0] = SOME_COMMAND;
OutBuffer[1] = SOME_COMMAND_PARAMETER_1;
OutBuffer[2] = SOME_COMMAND_PARAMETER_2;
OutBuffer[3] = SOME_COMMAND_PARAMETER_3;
if (MCHPHIDCLASS::HIDWriteReport (OutBuffer,
{
    if (MCHPHIDCLASS::HIDReadReport (InBuffer
    {
        // process InBuffer
    }
    else
    {
        // appropriate reaction to not receiv
    }
} else
{
    MessageBox ("Is the device connected?");
}
This section describes the API of the library.

**Topics**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HID Class</strong></td>
<td>HID Class DLL simplifies the interface to a HID class USB device. It assumes the firmware is configured to send an array of bytes. See the project in Microchip Solutions\USB Device - Custom Demos\Generic HID - Firmware. While there are quite a number of functions, only three are necessary: HIDClassInit, HIDWriteReport, HIDReadReport. The rest are optional or aliases for backwards compatibility.</td>
</tr>
</tbody>
</table>
HID Class

HID Class DLL simplifies the interface to a HID class USB device. It assumes the firmware is configured to send an array of bytes. See the project in Microchip Solutions\USB Device - Custom Demos\Generic HID - Firmware.

While there are quite a number of functions, only three are necessary: HIDClassInit, HIDWriteReport, HIDReadReport. The rest are optional or aliases for backwards compatibility.

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIDClassInit</td>
<td>This is the overview for the HIDClassInit method overload.</td>
</tr>
<tr>
<td>HIDWriteReport</td>
<td>Transfers data to the end device.</td>
</tr>
<tr>
<td>HIDReadReport</td>
<td>Transfers a data from the end device.</td>
</tr>
<tr>
<td>HIDClassVersion</td>
<td>Returns the version of the DLL</td>
</tr>
<tr>
<td>HIDIsConnected</td>
<td>OS inquiry to find out of the previously specified VID &amp; PID is currently attached.</td>
</tr>
<tr>
<td>HIDCloseReport</td>
<td>Closes the communication channel.</td>
</tr>
<tr>
<td>USBHIDClassInit</td>
<td>Initializes the DLL with the VID, PID and buffersize of the device.</td>
</tr>
<tr>
<td>USBHIDIsConnected</td>
<td>OS inquiry to find out of the previously specified VID &amp; PID is currently attached.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>USBHIDReadReport</td>
<td>Transfers data from the end device.</td>
</tr>
<tr>
<td>USBHIDWriteReport</td>
<td>Transfers data to the end device.</td>
</tr>
</tbody>
</table>

**Library API > HID Class**

Microchip HID Class DLL 1.10 - [15 April 2010]
Copyright © 2009, 2010 Microchip Technology, Inc. All rights reserved.

[Contents] [Index] [Home]
## HIDClassInit Method

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCHPHIDClass::HIDClassInit (unsigned int, unsigned int, unsigned int)</td>
<td>Initializes the DLL with the VID, PID and buffersize of the device.</td>
</tr>
<tr>
<td>MCHPHIDClass::HIDClassInit (unsigned int, unsigned int, unsigned int, unsigned int)</td>
<td>Initializes the DLL with the VID, PID, buffersize and timeout of the device.</td>
</tr>
</tbody>
</table>
MCHPHIDClass::HIDClassInit Method (unsigned int, unsigned int, unsigned int)

C

```c
static void HIDClassInit(
    unsigned int VendorID,
    unsigned int ProductID,
    unsigned int BuffSize
);
```

**Description**

Initializes the DLL with the VID, PID, and Buffersize of the device. This function should be called before attempting to communicate with the device.

**Preconditions**

None

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VendorID</td>
<td>16 bit unsigned integer.</td>
</tr>
<tr>
<td>ProductID</td>
<td>16 bit unsigned integer</td>
</tr>
<tr>
<td>BufferSize</td>
<td>16 bit unsigned integer. Usually 64</td>
</tr>
</tbody>
</table>

**Returns**

Nothing

**Remarks**
This function, or an overloaded variant, should be called prior to attempting to communicate with the device.

Example

```c
MCHPHIDCLASS::HIDClassInit(0x4D8, 0x0F00, 64);
```

Library API > HID Class > HIDClassInit Method > MCHPHIDClass::HIDClassInit Method (unsigned int, unsigned int, unsigned int)
MCHPHIDClass::HIDClassInit Method (unsigned int, unsigned int, unsigned int, unsigned int, unsigned int)

C

```c
static void HIDClassInit(
    unsigned int VendorID,
    unsigned int ProductID,
    unsigned int BuffSize,
    unsigned int Timeout
);
```

**Description**

Initializes the DLL with the VID, PID, and Buffersize of the device. This function should be called before attempting to communicate with the device. The reads and write use overlapped transfers. This version allows specifying how long to wait before timing out. The default is 1000mS.

**Preconditions**

None

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VendorID</td>
<td>16 bit unsigned integer.</td>
</tr>
<tr>
<td>ProductID</td>
<td>16 bit unsigned integer</td>
</tr>
<tr>
<td>BufferSize</td>
<td>16 bit unsigned integer. Usually 64</td>
</tr>
<tr>
<td>TimeOut</td>
<td>16 bit unsigned integer. Time in mS to wait for a read or write to timeout before</td>
</tr>
</tbody>
</table>
Returns

Nothing

Remarks

This function, or an overloaded variant, should be called prior to attempting to communicate with the device.

Example

```cpp
MCHPHIDCLASS::HIDClassInit(0x4D8, 0x0F00, 64, 250);
```
MCHPHIDClass::HIDWriteReport Method

C

```c
static bool HIDWriteReport(
    unsigned char buffer[],
    unsigned int nBytes
);
```

Description

Transfers a buffer to the end device. Returns true if successful, false if the transfer fails.

Preconditions

The DLL should be initialized via a call to `HIDClassInit` or variant.

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer</td>
<td>up to 64 byte array</td>
</tr>
<tr>
<td>nBytes</td>
<td>unsigned int. Number of valid bytes in the buffer.</td>
</tr>
</tbody>
</table>

Returns

true if the transfer is successful, false if not. A failure to transfer usually means the device has been disconnected.

Remarks

The full buffer size is always transferred, regardless of how many are valid.
Example

```c
ToSendBuffer[0] = 'H';
ToSendBuffer[1] = 'i';
ToSendBuffer[2] = ' ';
ToSendBuffer[3] = 'M';
ToSendBuffer[4] = 'o';
ToSendBuffer[5] = 'm';
ToSendBuffer[6] = '!';
MCHPHIDCLASS::HIDWriteReport(ToSendBuffer, 7);
```
MCHPHIDClass::HIDReadReport Method

```c
static bool HIDReadReport(
    unsigned char buffer[])
```

**Description**

Transfers a buffer from the end device. Returns true if successful, false if the transfer fails.

**Preconditions**

The DLL should be initialized via a call to `HIDClassInit`.

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer</td>
<td>A byte array up to 64 bytes long.</td>
</tr>
</tbody>
</table>

**Returns**

true if the transfer is successful, false if not. A transfer failure may mean the device is no longer connected or it could just mean the end device didn't send any data before timeout.

**Remarks**

**Example**

```c
if (MCHPHIDCLASS::HIDReadReport (ReadBuffer)) {
```
// Process buffer if arrived
switch (ReadBuffer[0])
{
}
else
{
    // do other stuff if no data
}
MCHPHIDClass::HIDClassVersion Method

C

```c
static unsigned int HIDClassVersion();
```

Description

Reports back the DLL, Major, Minor and Dot release numbers of the bill.

Preconditions

None

Returns

unsigned int representing the version of the DLL. Formatted as three 8 bit unsigned integers, major release.minor release.dot release. For example: Version 1.23.45 would be encoded as 0x00011739

Example

```c
Version = MCHPHIDCLASS::HIDClassVersion ()
Major = (Version & 0x00FF0000) >> 16;
Minor = (Version & 0x0000FF00) >> 8;
Dot = Version & 0x000000FF;
```

Library API > HID Class > MCHPHIDClass::HIDClassVersion Method
MCHPHIDClass::HIDIsConnected Method

C

```c
static bool HIDIsConnected();
```

Description

Polls the OS to find out if the previously specified VID & PID is currently attached. This is an OS inquiry only. No bus traffic is generated.

Preconditions

None

Returns

Nothing

Example

```c
MCHPHIDCLASS::HIDIsConnected()
```

Library API > HID Class > MCHPHIDClass::HIDIsConnected Method
MCHPHIDClass::HIDCloseReport Method

C

\[
\text{static void } \text{HIDCloseReport}();
\]

Description

Closes the communication channel. The next attempt to read from or write to the device will re-establish communications.

Preconditions

The DLL should be initialized via a call to \text{HIDClassInit} or variant.

Returns

True if the device is connected, False if not.

Example

\[
\text{if (MCHPHIDCLASS::HIDIsConnected ())}
\{
    \text{MCHPHIDClass::HIDCloseReport ();}
\}
\]

Library API > HID Class > MCHPHIDClass::HIDCloseReport Method
MCHPHIDClass::USBHIDClassInit Method

C

    static void USBHIDClassInit(
        unsigned int VendorID,
        unsigned int ProductID,
        unsigned int BuffSize
    );

Description

Initializes the DLL with the VID, PID, and Buffersize of the device. This function should be called before attempting to communicate with the device.

Preconditions

None

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VendorID</td>
<td>16 bit unsigned integer.</td>
</tr>
<tr>
<td>ProductID</td>
<td>16 bit unsigned integer</td>
</tr>
<tr>
<td>BufferSize</td>
<td>16 bit unsigned integer. Usually 64</td>
</tr>
</tbody>
</table>

Returns

Nothing

Remarks
This function, or an overloaded variant, should be called prior to attempting to communicate with the device.

Example

```
MCHPHIDCLASS::USBHIDClassInit(0x4D8, 0x0F00, 64);
```
MCHPHIDClass::USBHIDIsConnected Method

C

```c
static bool USBHIDIsConnected();
```

**Description**

Polls the OS to find out if the previously specified VID & PID is currently attached. This is an OS inquiry only. No bus traffic is generated.

**Preconditions**

None

**Returns**

Nothing

**Example**

```c
MCHPHIDCLASS::USBHIDIsConnected();
```

Library API > HID Class > MCHPHIDClass::USBHIDIsConnected Method
MCHPHIDClass::USBHIDReadReport Method

C

\[
\text{static bool USBHIDReadReport(}
\text{    unsigned char buffer[]}
\text{);}\]

Description

Transfers a buffer from the end device. Returns true if successful, false if the transfer fails.

Preconditions

The DLL should be initialized via a call to `HIDClassInit` or variant.

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer</td>
<td>A byte array up to 64 bytes long.</td>
</tr>
</tbody>
</table>

Returns

true if the transfer is successful, false if not. A transfer failure may mean the device is no longer connected or it could just mean the end device didn't send any data before timeout.

Remarks

Example

```c
if (MCHPHIDCLASS::USBHIDReadReport (ReadBuffer))
{
```
// Process buffer if arrived
switch (ReadBuffer[0])
{
}
else
{
    // do other stuff if no data
}
MCHPHIDClass::USBHIDWriteReport Method

```c
static bool USBHIDWriteReport(
    unsigned char buffer[],
    unsigned int nBytes
);
```

Description

Transfers the buffer to the end device. Returns true if successful, false if the transfer fails.

Preconditions

The DLL should be initialized via a call to `HIDClassInit` or variant.

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer</td>
<td>up to 64 byte array</td>
</tr>
<tr>
<td>nBytes</td>
<td>unsigned int. Number of valid bytes in the buffer.</td>
</tr>
</tbody>
</table>

Returns

true if the transfer is successful, false if not. A failure to transfer usually means the device has been disconnected.

Remarks

The full buffer size is always transferred, regardless of how many are valid.
Example

```
ToDoSendBuffer[0] = 'H';
ToDoSendBuffer[1] = 'i';
ToDoSendBuffer[2] = ' ';
ToDoSendBuffer[3] = 'M';
ToDoSendBuffer[4] = 'o';
ToDoSendBuffer[5] = 'm';
ToDoSendBuffer[6] = '!';
MCHPHIDCLASS::USBHIDWriteReport(ToDoSendBuffer, 7);
```

Library API > HID Class > MCHPHIDClass::USBHIDWriteReport Method
Library Migration

This release is 100% backwards compatible.. no migration necessary
### Contents

- Introduction
- SW License Agreement
- Release Notes
- Running the Demos

### Symbol Reference

**HIDClass Namespace**

**MCHPHIDClass Class**

**MCHPHIDClass Methods**

**Files**

- HID class.h

### Using the Library

- Library Architecture
- How the Library Works
- Getting Started

### Library API

**HID Class**

**HIDClassInit Method**

- MCHPHIDClass::HIDClassInit Method (unsigned int, unsigned int)
- MCHPHIDClass::HIDClassInit Method (unsigned int, unsigned int)
- MCHPHIDClass::HIDWriteReport Method
- MCHPHIDClass::HIDReadReport Method
- MCHPHIDClass::HIDClassVersion Method
- MCHPHIDClass::HIDIsConnected Method
- MCHPHIDClass::HIDCloseReport Method
- MCHPHIDClass::USBHIDClassInit Method
- MCHPHIDClass::USBHIDIsConnected Method
- MCHPHIDClass::USBHIDReadReport Method
- MCHPHIDClass::USBHIDWriteReport Method

### Library Migration
Index

F | G | H | I | L | M | R | S | U

F
Files

G
Getting Started

H
HID Class
HID class.h
HIDClass namespace
HIDClassInit method
How the Library Works

I
Introduction

L
Library API
Library Architecture
Library Migration

M
MCHPHIDClass class
about MCHPHIDClass class
MCHPHIDClass methods
MCHPHIDClass::HIDClassInit method

R
Release Notes
Running the Demos

S
SW License Agreement

U
Using the Library