API User's Manual

For the

Chopper USB Libraries

Version 1.0.0

Prepared by:
New Focus
3635 Peterson Way
Santa Clara, CA 95054

1 Introduction

The Chopper USB Libraries allow software running on a PC to communicate with a New Focus Chopper over a Universal Serial Bus. There is a native interface (NpChopperLib.dll) and a .NET interface (NpChopperLibWrap.dll). The HID driver is used by both of these libraries and is already installed on a Windows PC.

2 The Native Interface

The library that provides the native interface is NpChopperLib.dll. The public interface functions are defined in the C++ header file NpChopperLib.h:

```
////////////////////////////////////////////////////////////////////////////////
// This method sets the logging flag to the passed in value.
//
// value:      The boolean value used to set the logging flag.
////////////////////////////////////////////////////////////////////////////////
void HidSetLogging (bool value)

////////////////////////////////////////////////////////////////////////////////
// This method sets the read timeout value in milliseconds.
//
// nMilliseconds:  The read timeout value in milliseconds (-1 = wait forever).
////////////////////////////////////////////////////////////////////////////////
void HidSetReadTimeout (int nMilliseconds)

////////////////////////////////////////////////////////////////////////////////
// This method gets the number of discovered devices.
//
// Returns:    The number of discovered devices.
////////////////////////////////////////////////////////////////////////////////
int HidGetDeviceCount ()

////////////////////////////////////////////////////////////////////////////////
// This method gets the array of device keys (discovered devices).
//
// DeviceKeys: The array of device keys (discovered devices).
// Returns:    The number of device keys in the list.
////////////////////////////////////////////////////////////////////////////////
int HidGetDeviceKeys (char* DeviceKeys)

////////////////////////////////////////////////////////////////////////////////
// This method discovers all chopper devices connected via USB.
////////////////////////////////////////////////////////////////////////////////
void HidDiscover ()

////////////////////////////////////////////////////////////////////////////////
// This method reads binary data from the specified chopper device.
//
// pDeviceKey: The device key of the chopper.
// pBuffer:            The read buffer.
// nBytesRead: The number of bytes read.
// Returns:            True for success, false for failure.
////////////////////////////////////////////////////////////////////////////////
bool HidReadBinary (char* pDeviceKey, unsigned char* byteArr, int* nBytesRead)

////////////////////////////////////////////////////////////////////////////////
// This method reads ascii data from the specified chopper device.
//
// pDeviceKey: The device key of the chopper.
// pBuffer:            The read buffer.
// Returns:            True for success, false for failure.
////////////////////////////////////////////////////////////////////////////////
bool HidRead (char* pDeviceKey, char* pBuffer)

////////////////////////////////////////////////////////////////////////////////
// This method writes binary data to the specified chopper device.
//
```

```
// pDeviceKey: The device key of the chopper.
// pBuffer:           The write buffer.
// Returns:           True for success, false for failure.
////////////////////////////////////////////////////////////////////////////////
bool HidWriteBinary (char* pDeviceKey, unsigned char* byteArr)

////////////////////////////////////////////////////////////////////////////////
// This method writes ascii data to the specified chopper device.
//
// pDeviceKey: The device key of the chopper.
// pBuffer:           The write buffer.
// Returns:           True for success, false for failure.
////////////////////////////////////////////////////////////////////////////////
bool HidWrite (char* pDeviceKey, char* pBuffer)

////////////////////////////////////////////////////////////////////////////////
// This method sends a command to the specified chopper device and returns
// the response data.
//
// pDeviceKey: The device key of the chopper.
// pCommand:   The command buffer.
// pBuffer:           The response buffer.
// Returns:           True for success, false for failure.
////////////////////////////////////////////////////////////////////////////////
bool HidQuery (char* pDeviceKey, char* pCommand, char* pBuffer)

////////////////////////////////////////////////////////////////////////////////
// This method shuts down USB communication.
////////////////////////////////////////////////////////////////////////////////
void HidShutdown ()
```

3 The .NET Interface

The library that provides the .NET interface is NpChopperLibWrap.dll. It can be used by development languages such as C# and LabVIEW, or any other language that allows methods in a .NET assembly to be called. NpChopperLibWrap.dll uses the native interface functions to communicate with the instrument, but its interface methods wrap around the native functions and provide some enhanced functionality (e.g. logging and overloaded method calls that allow for different argument types to be passed into the I/O calls). The public interface is defined as follows:

```
/// <summary>
/// Default Constructor.
/// </summary>
public USB ()

/// <summary>
/// Constructor.
/// </summary>
/// <param name="bLogging">True to turn on logging, false to turn off logging.</param>
public USB (bool bLogging)

/// <summary>
/// This property gets / sets the logging flag.
/// </summary>
public bool Logging

/// <summary>
/// This property gets the log file path.
/// </summary>
public string LogFilePath

/// <summary>
/// This property sets the read timeout value in milliseconds.
/// </summary>
public int ReadTimeout

/// <summary>
/// This property gets the discovered device count.
/// </summary>
public int DeviceCount

/// <summary>
/// This method gets the device keys from the device table.
/// </summary>
/// <returns>The device keys.</returns>
public string[] GetDeviceKeys ()

/// <summary>
/// This method discovers all chopper devices connected via USB.
/// </summary>
public void Discover ()

/// <summary>
/// This method reads binary data from the specified chopper device.
/// </summary>
/// <param name="deviceKey">The device key of the chopper.</param>
/// <param name="buffer">The read buffer.</param>
/// <param name="bytesRead">The number of bytes read.</param>
/// <returns>True for success, false for failure.</returns>
public bool Read (string deviceKey, byte[] buffer, ref int bytesRead)

/// <summary>
/// This method reads ascii data from the specified chopper device.
/// </summary>
/// <param name="deviceKey">The device key of the chopper.</param>
/// <param name="buffer">The read buffer.</param>
/// <returns>True for success, false for failure.</returns>
public bool Read (string deviceKey, StringBuilder buffer)
```

```
/// <summary>
/// This method writes binary data to the specified chopper device.
/// </summary>
/// <param name="deviceKey">The device key of the chopper.</param>
/// <param name="buffer">The write buffer.</param>
/// <returns>True for success, false for failure.</returns>
public bool Write (string deviceKey, byte[] buffer)

/// <summary>
/// This method writes ascii data to the specified chopper device.
/// </summary>
/// <param name="deviceKey">The device key of the chopper.</param>
/// <param name="cmd">The command to send to the device.</param>
/// <returns>True for success, false for failure.</returns>
public bool Write (string deviceKey, string cmd)

/// <summary>
/// This method sends a command to the specified chopper device and returns
/// the response data.
/// </summary>
/// <param name="deviceKey">The device key of the chopper.</param>
/// <param name="cmd">The command to send to the device.</param>
/// <param name="buffer">The response buffer.</param>
/// <returns>True for success, false for failure.</returns>
public bool Query (string deviceKey, string cmd, StringBuilder buffer)

/// <summary>
/// This method shuts down USB communication.
/// </summary>
public void Shutdown ()
```

3.1 Logging

Logging can be turned on by passing a Boolean true value into the constructor or by setting the Logging property to true. When logging is turned on the following information is written to the log file: general information about the operating system, device discovery data, data being written to the device, response data being read from the device, and general debugging information. If a C# exception is thrown then this information is logged even if logging is not turned on. Log files are named Log<YYYY-MM-DD>.txt and are created in one of the following folders (depending upon system settings and permissions): (1) the \Log folder located in the directory containing the current executable (.exe), (2) the same folder path as in #1, except the highest level folder is replaced with “My Documents” (e.g. C:\Program Files\Newport\Install Folder\Bin\Log would be replaced with C:\My Documents\Newport\Install Folder\Bin\Log), and (3) the My Documents\Log folder.