Command Library API

AG-UC2-UC8 3.0.0

1791 Deere Ave.
Irvine, CA 92606
(877) 835-9620

E-MAIL: TECH@NEWPORT.COM
WWW.NEWPORT.COM

Table of Contents

1 Libraries 4

2 Sample Code 4

3 CmdLibAgilis 3.0.0 API 5

4 VCPIOLib 1.0.0 API 13

1 Libraries

The Command Library API is an Application Programmer's Interface (API) for communicating with an Agilis UC2 or an Agilis UC8 controller. This API is contained within the Agils Command Library (CmdLibAgilis.dll). CmdLibAgilis.dll is dependent upon two other libraries: VCPIOLib.dll and VCPWrap.dll.

VCPIOLib.dll is a library which performs device I/O functions such as initialization, discovery, open, close, read, write, and shutdown. Some of these functions are wrapped in methods which are contained in CmdLibAgilis.dll. Therefore, VCPIOLib is primarily used for device discovery and proper shutdown of device communication.

VCPWrap.dll is a low-level library that performs device communication using the Virtual COM Port. The user does not need to directly access the methods in this library, but it must be present because it is a support library that is used by both CmdLibAgils.dll and VCPIOLib.dll for USB communication.

2 Sample Code

The sample code that is installed with the AG-UC2-UC8 application demonstrates how to use the provided libraries. There is sample source code in C#, Python and LabVIEW.

The OpenMultipleDevices sample shows how to communicate with multiple devices in one program. It discovers all available devices and then loops through the list of discovered devices. Each time through the loop it opens the device it wants to communicate with and then displays the firmware version of the controller and writes this same information to a log file. Before exiting the application, Shutdown is called to properly shutdown device communication.

The StepAmplitude sample shows how to get and set the step amplitude of the controller. First it discovers the available devices and then opens the first one that it can communicate with. Then it puts the controller into remote mode and sets the current channel. Then it gets the negative and positive step amplitude values and displays them. Then it sets the negative and positive step amplitude values to their maximum value. Then it gets the new negative and positive step amplitude values and displays them again. Finally, the device is closed and Shutdown is called to properly shutdown device communication.

3 CmdLibAgilis 3.0.0 API

```
/// <summary>
/// Constructor.
/// </summary>
/// <param name="vcpIOLib">The Virtual COM Port I/O library object.</param>
public CmdLibAgilis (VCPIOLib.VCPIOLib vcpIOLib)
    : base (vcpIOLib)

/// <summary>
/// This property gets / sets number of minutes to wait for a Measure command to complete
before reporting a timeout.
/// </summary>
public float MeasureCmdTimeoutMinutes

/// <summary>
/// This method determines if the passed in command is valid or not.
/// </summary>
/// <param name="cmd">The command.</param>
/// <returns>True if the command is valid, otherwise false.</returns>
public bool IsValidCmd (string cmd)

/// <summary>
/// This method determines if the passed in command is a query or not.
/// </summary>
/// <param name="cmd">The command.</param>
/// <returns>True if the command is a query, otherwise false.</returns>
public bool IsQueryCmd (string cmd)

/// <summary>
/// This method gets the channel number (one-relative).
/// </summary>
/// <param name="channel">The current channel number.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "CC"</remarks>
public bool GetChannel (ref int channel)

/// <summary>
/// This method sets the current channel number (one-relative).
/// </summary>
/// <param name="channel">The channel number to become the current selection.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "CC"</remarks>
public bool SetChannel (int channel)

/// <summary>
/// This method gets the jog mode.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <param name="mode">The jog mode.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "JA"</remarks>
public bool GetJogMode (int axis, ref int mode)

/// <summary>
/// This method sets the jog mode and starts jogging.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <param name="mode">The jog mode.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "JA"</remarks>
public bool StartJogging (int axis, int mode)
```

```
/// <summary>
/// This method measures the current position.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <param name="distance">The distance of the current position to the limit in 1/1000th of the
total travel.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "MA"</remarks>
public bool MeasurePosition (int axis, ref int distance)

/// <summary>
/// This method sets the controller to local mode.
/// </summary>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "ML"</remarks>
public bool SetLocalMode ()

/// <summary>
/// This method sets the controller to remote mode.
/// </summary>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "MR"</remarks>
public bool SetRemoteMode ()

/// <summary>
/// This method sets the jog mode, starts jogging and stops automatically when the limit is
activated.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <param name="mode">The jog mode.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "MV"</remarks>
public bool MoveToLimit (int axis, int mode)

/// <summary>
/// This method gets the target position for an absolute move or a relative move.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <param name="targetPos">The target position.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "PA"</remarks>
public bool GetTargetPosition (int axis, ref int targetPos)

/// <summary>
/// This method performs an absolute move to the specified target position.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <param name="targetPos">The target position.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "PA"</remarks>
public bool AbsoluteMove (int axis, int targetPos)

/// <summary>
/// This method gets the limit switch status.
/// </summary>
/// <param name="switchStatus">The limit switch status.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "PH"</remarks>
public bool GetLimitSwitchStatus (ref int switchStatus)

/// <summary>
/// This method performs a relative move by the specified number of steps.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <param name="steps">The number of steps.</param>
```

```
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "PR"</remarks>
public bool RelativeMove (int axis, int steps)

/// <summary>
/// This method resets the controller.
/// </summary>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "RS"</remarks>
public bool Reset ()

/// <summary>
/// This method stops motion on the specified axis.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "ST"</remarks>
public bool StopMotion (int axis)

/// <summary>
/// This method gets the step amplitude setting in the positive direction.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <param name="stepAmplitude">The step amplitude.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "SU"</remarks>
public bool GetStepAmplitudePositive (int axis, ref int stepAmplitude)

/// <summary>
/// This method gets the step amplitude setting in the negative direction.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <param name="stepAmplitude">The step amplitude.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "SU"</remarks>
public bool GetStepAmplitudeNegative (int axis, ref int stepAmplitude)

/// <summary>
/// This method sets the step amplitude setting in the positive direction.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <param name="stepAmplitude">The step amplitude.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "SU"</remarks>
public bool SetStepAmplitudePositive (int axis, int stepAmplitude)

/// <summary>
/// This method sets the step amplitude setting in the negative direction.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <param name="stepAmplitude">The step amplitude.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "SU"</remarks>
public bool SetStepAmplitudeNegative (int axis, int stepAmplitude)

/// <summary>
/// This method gets the error code of the previous command.
/// </summary>
/// <param name="errorCode">The error code of the previous command.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "TE"</remarks>
public bool GetErrorPreviousCommand (ref int errorCode)

/// <summary>
```

```
/// This method gets the number of accumulated steps in the positive direction minus the number
of steps
/// in the negative direction since powering the controller or since the last ZP (zero
position) command.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <param name="numberSteps">The number of accumulated steps.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "TP"</remarks>
public bool GetStepsAccumulated (int axis, ref int numberSteps)

/// <summary>
/// This method gets the status of the specified axis.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <param name="axisStatus">The axis status.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "TS"</remarks>
public bool GetAxisStatus (int axis, ref int axisStatus)

/// <summary>
/// This method gets the firmware version of the controller.
/// </summary>
/// <param name="firmwareVersion">The firmware version of the controller.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "VE"</remarks>
public bool GetFirmwareVersion (ref string firmwareVersion)

/// <summary>
/// This method resets the step counter to zero for the specified axis.
/// </summary>
/// <param name="axis">The axis number.</param>
/// <returns>True for success, false for failure.</returns>
/// <remarks>For more information see documentation on the firmware command "ZP"</remarks>
public bool ResetStepCounter (int axis)


/// <summary>
/// This property gets the baud rate used by this device.
/// </summary>
public int BaudRate

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

/// <summary>
/// This method discovers the devices that are available for communication.
/// </summary>
public void DiscoverDevices ()

/// <summary>
/// This method gets the number of discovered devices.
/// </summary>
/// <returns>The number of discovered devices.</returns>
public int GetDeviceCount ()

/// <summary>
/// This method gets all the device keys from the list of discovered devices.
/// </summary>
/// <returns>All the device keys that have been discovered.</returns>
public string[] GetDeviceKeys ()

/// <summary>
```

```
/// This method determines if the passed in device key is valid or not.
/// </summary>
/// <param name="deviceKey">The device key.</param>
/// <returns>True if the device key is in the list of discovered devices, otherwise
false.</returns>
public bool DeviceKeyIsValid (string deviceKey)

/// <summary>
/// This method opens the specified device.
/// </summary>
/// <param name="deviceKey">The device key.</param>
/// <returns>The error code.</returns>
public int Open (string deviceKey)

/// <summary>
/// This method closes the specified device.
/// </summary>
/// <returns>The error code.</returns>
public int Close ()

/// <summary>
/// This method reads a string response from the device.
/// </summary>
/// <param name="value">The string response.</param>
/// <returns>The error code.</returns>
public bool Read (ref string value)

/// <summary>
/// This method reads data from the specified device.
/// </summary>
/// <param name="buffer">The buffer to hold the response data.</param>
/// <returns>The error code.</returns>
public int Read (StringBuilder buffer)

/// <summary>
/// This method writes data to the specified device.
/// </summary>
/// <param name="buffer">The buffer to hold the data.</param>
/// <returns>The error code.</returns>
public int Write (string buffer)

/// <summary>
/// This method sends the passed in query to the device and returns the string response.
/// </summary>
/// <param name="cmd">The query string.</param>
/// <param name="value">The string response.</param>
/// <returns>The error code.</returns>
public bool Query (string cmd, ref string value)

/// <summary>
/// This method sends the passed in command string to the specified device
/// and reads the response data.
/// </summary>
/// <param name="command">The command string to send.</param>
/// <param name="buffer">The buffer to hold the response data.</param>
/// <returns>The error code.</returns>
public int Query (string command, StringBuilder buffer)

/// <summary>
/// This method writes the passed in string to the log file if logging is turned on.
/// </summary>
/// <param name="outputText">The text to output.</param>
public void WriteLog (string outputText)

/// <summary>
```

```
/// This method writes the passed in arguments to the log file according
/// to the specified format.
/// </summary>
/// <param name="logging">True if logging, otherwise false.</param>
/// <param name="format">The format specifier.</param>
/// <param name="args">The data values to output.</param>
public void WriteLog (bool logging, string format, params object[] args)

/// <summary>
/// This method writes the passed in arguments to the log file according
/// to the specified format.
/// </summary>
/// <param name="format">The format specifier.</param>
/// <param name="args">The data values to output.</param>
public void WriteLog (string format, params object[] args)

/// <summary>
/// This method reads a string response from the device.
/// </summary>
/// <param name="deviceKey">The device key.</param>
/// <param name="value">The string response.</param>
/// <returns>True for success, false for failure.</returns>
public bool ReadString (ref string value)

/// <summary>
/// This method sends the passed in command to the device.
/// </summary>
/// <param name="deviceKey">The device key.</param>
/// <param name="cmd">The command string.</param>
/// <returns>True for success, false for failure.</returns>
public bool WriteString (string cmd)

/// <summary>
/// This method sends the passed in query to the device and returns the string response.
/// </summary>
/// <param name="cmd">The query string.</param>
/// <param name="lengthOfCmdInResponse">The length of the command sent back in the
response.</param>
/// <param name="value">The string response.</param>
/// <returns>True for success, false for failure.</returns>
public bool QueryString (string cmd, int lengthOfCmdInResponse, ref string value)

/// <summary>
/// This method sends the passed in command to the device along with an integer parameter.
/// </summary>
/// <param name="deviceKey">The device key.</param>
/// <param name="cmd">The command string.</param>
/// <param name="value">The integer parameter.</param>
/// <returns>True for success, false for failure.</returns>
public bool WriteInt (string cmd, int value)

/// <summary>
/// This method sends the passed in query to the device and returns the integer response.
/// </summary>
/// <param name="cmd">The query string.</param>
/// <param name="lengthOfCmdInResponse">The length of the command sent back in the
response.</param>
/// <param name="value">The integer response.</param>
/// <param name="shouldConvert">True if the response string should be converted to a number,
otherwise false.</param>
/// <returns>True for success, false for failure.</returns>
public bool QueryInt (string cmd, int lengthOfCmdInResponse, ref int value, bool shouldConvert)

/// <summary>
```

```
/// This method sends the passed in command to the device along with an unsigned integer
parameter.
/// </summary>
/// <param name="deviceKey">The device key.</param>
/// <param name="cmd">The command string.</param>
/// <param name="value">The unsigned integer parameter.</param>
/// <returns>True for success, false for failure.</returns>
public bool WriteUInt (string cmd, uint value)

/// <summary>
/// This method sends the passed in query to the device and returns the unsigned integer
response.
/// </summary>
/// <param name="cmd">The query string.</param>
/// <param name="lengthOfCmdInResponse">The length of the command sent back in the
response.</param>
/// <param name="value">The unsigned integer response.</param>
/// <param name="shouldConvert">True if the response string should be converted to a number,
otherwise false.</param>
/// <returns>True for success, false for failure.</returns>
public bool QueryUInt (string cmd, int lengthOfCmdInResponse, ref uint value, bool
shouldConvert)

/// <summary>
/// This method sends the passed in command to the device along with a floating point
parameter.
/// </summary>
/// <param name="deviceKey">The device key.</param>
/// <param name="cmd">The command string.</param>
/// <param name="value">The floating point parameter.</param>
/// <returns>True for success, false for failure.</returns>
public bool WriteFloat (string cmd, float value)

/// <summary>
/// This method sends the passed in query to the device and returns the floating point
response.
/// </summary>
/// <param name="cmd">The query string.</param>
/// <param name="lengthOfCmdInResponse">The length of the command sent back in the
response.</param>
/// <param name="value">The floating point response.</param>
/// <param name="shouldConvert">True if the response string should be converted to a number,
otherwise false.</param>
/// <returns>True for success, false for failure.</returns>
public bool QueryFloat (string cmd, int lengthOfCmdInResponse, ref float value, bool
shouldConvert)

/// <summary>
/// This method sends the passed in command to the device along with a decimal parameter.
/// </summary>
/// <param name="deviceKey">The device key.</param>
/// <param name="cmd">The command string.</param>
/// <param name="value">The decimal parameter.</param>
/// <returns>True for success, false for failure.</returns>
public bool WriteDecimal (string cmd, decimal value)

/// <summary>
/// This method sends the passed in query to the device and returns the decimal response.
/// </summary>
/// <param name="cmd">The query string.</param>
/// <param name="lengthOfCmdInResponse">The length of the command sent back in the
response.</param>
/// <param name="value">The decimal response.</param>
/// <param name="shouldConvert">True if the response string should be converted to a number,
otherwise false.</param>
```

```
/// <returns>True for success, false for failure.</returns>
public bool QueryDecimal (string cmd, int lengthOfCmdInResponse, ref decimal value, bool
shouldConvert)

/// <summary>
/// This method sends the passed in command to the device along with a double-precision
parameter.
/// </summary>
/// <param name="deviceKey">The device key.</param>
/// <param name="cmd">The command string.</param>
/// <param name="value">The double-precision parameter.</param>
/// <returns>True for success, false for failure.</returns>
public bool WriteDouble (string cmd, double value)

/// <summary>
/// This method sends the passed in query to the device and returns the double-precision
response.
/// </summary>
/// <param name="cmd">The query string.</param>
/// <param name="lengthOfCmdInResponse">The length of the command sent back in the
response.</param>
/// <param name="value">The double-precision response.</param>
/// <param name="shouldConvert">True if the response string should be converted to a number,
otherwise false.</param>
/// <returns>True for success, false for failure.</returns>
public bool QueryDouble (string cmd, int lengthOfCmdInResponse, ref double value, bool
shouldConvert)
```

4 VCPIOLib 1.0.0 API

```
/// <summary>
/// Constructor.
/// </summary>
public VCPIOLib ()

/// <summary>
/// Constructor.
/// </summary>
/// <param name="isLogging">True to turn on logging, otherwise false.</param>
public VCPIOLib (bool isLogging)

/// <summary>
/// This method performs initialization for this class.
/// </summary>
/// <param name="isLogging">True to turn on logging, otherwise false.</param>
private void DeviceIOInit (bool isLogging)

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

/// <summary>
/// This property gets / sets the device key of the last device that was successfully opened.
/// </summary>
public virtual string LastOpenedDeviceKey

/// <summary>
/// This method discovers the devices that are available for communication.
/// </summary>
public virtual void DiscoverDevices ()

/// <summary>
/// This method gets the number of discovered devices.
/// </summary>
/// <returns>The number of discovered devices.</returns>
public virtual int GetDeviceCount ()

/// <summary>
/// This method gets all the device keys from the list of discovered devices.
/// </summary>
/// <returns>All the device keys that have been discovered.</returns>
public virtual string[] GetDeviceKeys ()

/// <summary>
/// This method determines if the passed in device key is valid or not.
/// </summary>
/// <param name="deviceKey">The device key.</param>
/// <returns>True if the device key is in the list of discovered devices, otherwise
false.</returns>
public virtual bool DeviceKeyIsValid (string deviceKey)

/// <summary>
/// This method returns true if the current device is open, otherwise false is returned.
/// </summary>
/// <returns>True if the current device is open, otherwise false.</returns>
public virtual bool DeviceIsOpen ()

/// <summary>
/// This method opens the first valid device in the list of discovered devices.
/// </summary>
```

```
/// <param name="baudrate">The VCP baud rate.</param>
public void OpenFirstValidDevice (int baudrate)

/// <summary>
/// This method handles the FirstDevOpened event.
/// </summary>
/// <param name="deviceKey">The device key.</param>
private void OnFirstDevOpened (string deviceKey)

/// <summary>
/// This method opens the specified device.
/// </summary>
/// <param name="deviceKey">The device key.</param>
/// <param name="baudrate">The VCP baud rate.</param>
/// <returns>The error code.</returns>
public virtual int Open (string deviceKey, int baudrate)

/// <summary>
/// This method closes the specified device.
/// </summary>
/// <returns>The error code.</returns>
public virtual int Close ()

/// <summary>
/// This method reads a string response from the device.
/// </summary>
/// <param name="value">The string response.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool Read (ref string value)

/// <summary>
/// This method reads data from the specified device.
/// </summary>
/// <param name="buffer">The buffer to hold the response data.</param>
/// <returns>The error code.</returns>
public virtual int Read (ref StringBuilder buffer)

/// <summary>
/// This method writes data to the specified device.
/// </summary>
/// <param name="buffer">The buffer to hold the data.</param>
/// <returns>The error code.</returns>
public virtual int Write (string buffer)

/// <summary>
/// This method sends the passed in query to the device and returns the string response.
/// </summary>
/// <param name="cmd">The query string.</param>
/// <param name="value">The string response.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool Query (string cmd, ref string value)

/// <summary>
/// This method sends the passed in command string to the specified device
/// and reads the response data.
/// </summary>
/// <param name="command">The command string to send.</param>
/// <param name="buffer">The buffer to hold the response data.</param>
/// <returns>The error code.</returns>
public virtual int Query (string command, ref StringBuilder buffer)

/// <summary>
/// This method writes the passed in string to the log file if logging is turned on.
/// </summary>
/// <param name="outputText">The text to output.</param>
```

```
public virtual void WriteLog (string outputText)

/// <summary>
/// This method writes the passed in arguments to the log file according
/// to the specified format.
/// </summary>
/// <param name="logging">True if logging, otherwise false.</param>
/// <param name="format">The format specifier.</param>
/// <param name="args">The data values to output.</param>
public virtual void WriteLog (bool logging, string format, params object[] args)

/// <summary>
/// This method writes the passed in arguments to the log file according
/// to the specified format.
/// </summary>
/// <param name="format">The format specifier.</param>
/// <param name="args">The data values to output.</param>
public virtual void WriteLog (string format, params object[] args)

/// <summary>
/// This method reads a string response from the device.
/// </summary>
/// <param name="value">The string response.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool ReadString (ref string value)

/// <summary>
/// This method sends the passed in command to the device.
/// </summary>
/// <param name="cmd">The command string.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool WriteString (string cmd)

/// <summary>
/// This method sends the passed in query to the device and returns the string response.
/// </summary>
/// <param name="cmd">The query string.</param>
/// <param name="lengthOfCmdInResponse">The length of the command sent back in the
response.</param>
/// <param name="value">The string response.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool QueryString (string cmd, int lengthOfCmdInResponse, ref string value)

/// <summary>
/// This method sends the passed in command to the device along with an integer parameter.
/// </summary>
/// <param name="cmd">The command string.</param>
/// <param name="value">The integer parameter.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool WriteInt (string cmd, int value)

/// <summary>
/// This method sends the passed in query to the device and returns the integer response.
/// </summary>
/// <param name="cmd">The query string.</param>
/// <param name="lengthOfCmdInResponse">The length of the command sent back in the
response.</param>
/// <param name="value">The integer response.</param>
/// <param name="shouldConvert">True if the response string should be converted to a number,
otherwise false.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool QueryInt (string cmd, int lengthOfCmdInResponse, ref int value, bool
shouldConvert)

/// <summary>
```

```
/// This method sends the passed in command to the device along with an unsigned integer
parameter.
/// </summary>
/// <param name="cmd">The command string.</param>
/// <param name="value">The unsigned integer parameter.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool WriteUInt (string cmd, uint value)

/// <summary>
/// This method sends the passed in query to the device and returns the unsigned integer
response.
/// </summary>
/// <param name="cmd">The query string.</param>
/// <param name="lengthOfCmdInResponse">The length of the command sent back in the
response.</param>
/// <param name="value">The unsigned integer response.</param>
/// <param name="shouldConvert">True if the response string should be converted to a number,
otherwise false.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool QueryUInt (string cmd, int lengthOfCmdInResponse, ref uint value, bool
shouldConvert)

/// <summary>
/// This method sends the passed in command to the device along with a floating point
parameter.
/// </summary>
/// <param name="cmd">The command string.</param>
/// <param name="value">The floating point parameter.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool WriteFloat (string cmd, float value)

/// <summary>
/// This method sends the passed in query to the device and returns the floating point
response.
/// </summary>
/// <param name="cmd">The query string.</param>
/// <param name="lengthOfCmdInResponse">The length of the command sent back in the
response.</param>
/// <param name="value">The floating point response.</param>
/// <param name="shouldConvert">True if the response string should be converted to a number,
otherwise false.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool QueryFloat (string cmd, int lengthOfCmdInResponse, ref float value, bool
shouldConvert)

/// <summary>
/// This method sends the passed in command to the device along with a decimal parameter.
/// </summary>
/// <param name="cmd">The command string.</param>
/// <param name="value">The decimal parameter.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool WriteDecimal (string cmd, decimal value)

/// <summary>
/// This method sends the passed in query to the device and returns the decimal response.
/// </summary>
/// <param name="cmd">The query string.</param>
/// <param name="lengthOfCmdInResponse">The length of the command sent back in the
response.</param>
/// <param name="value">The decimal response.</param>
/// <param name="shouldConvert">True if the response string should be converted to a number,
otherwise false.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool QueryDecimal (string cmd, int lengthOfCmdInResponse, ref decimal value,
bool shouldConvert)
```

```
/// <summary>
/// This method sends the passed in command to the device along with a double-precision
parameter.
/// </summary>
/// <param name="cmd">The command string.</param>
/// <param name="value">The double-precision parameter.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool WriteDouble (string cmd, double value)

/// <summary>
/// This method sends the passed in query to the device and returns the double-precision
response.
/// </summary>
/// <param name="cmd">The query string.</param>
/// <param name="lengthOfCmdInResponse">The length of the command sent back in the
response.</param>
/// <param name="value">The double-precision response.</param>
/// <param name="shouldConvert">True if the response string should be converted to a number,
otherwise false.</param>
/// <returns>True for success, false for failure.</returns>
public virtual bool QueryDouble (string cmd, int lengthOfCmdInResponse, ref double value, bool
shouldConvert)

/// <summary>
/// This method performs the required steps to cleanly stop communication.
/// </summary>
public virtual void Shutdown ()
```

Visit Newport Online at:

www.newport.com

North America
Newport Corporation
1791 Deere Ave.
Irvine, CA 92606, USA
Tel.: (877) 835-9620

Sales
e-mail: sales@newport.com

Technical Support
e-mail: tech@newport.com

Service, RMAs & Returns
e-mail: service@newport.com

Asia
Lot J3-8, Wuxi Export Processing Zone
New District
Wuxi, Jiangsu, 214028, China
Tel.: 400 799 8000

Sales
e-mail: china@newport.com

Technical Support
e-mail: china@newport.com

Service, RMAs & Returns
e-mail:
service-newport-asia@mksinst.com

Europe
MICRO-CONTROLE Spectra-Physics
S.A.
7 rue des Plantes
Zone Industrielle 45340
Beaune la Rolande, France
Tel.: +33 (0)1.60.91.68.68
e-mail: france@newport.com