

Quadra4 Windows Applications and DLL Description

Version 1.00 2/15/2010

General Description:

The Versalent Quadra-4 is a USB connected 4-axis incremental encoder feedback, and analog drive device which can be used to implement PC based servo loops. It does not contain the servo-control software and is merely a position reporting device combined with an analog output device which allows the PC to implement servo control. Quad4.DLL is provided to allow quick and easy servo software development, by providing access to the Quadra-4s position and control features. This is a dotNET dll (framework 2.0) which offers properties and methods for setting/reading encoder multiplication, analog output range, digital ins/outs and all other features of the Quadra-4. Authored in C# it can be used in any of the dotNET languages/environments as well as other environments (with appropriate interfaces).

Using Quad4.dll is simple, and a full C# multi-Quadra-4, multi-axis PID servo application based on this dll is included for your use and/or reference. Just add a reference to this DLL in your dotNET project, instantiate one or more Quadra4 objects, and you have full access to the properties and methods of the device(s). The application provided is multi-threaded (one thread for each Quadra-4 connected to your PC) so that the application GUI is responsive while one or several Quadra-4s are busy maintaining servo control.

For even quicker and simpler development, Visual Basic 6 support is also provided. A full VB6 multi-Quadra-4 servo control application w/source is also provided. This is a single-threaded application which can serve as the base code for your own multi-axis servo controller.

Summary of Software Suite Provided:

1. **Quad4.dll** .. dotNET dll for use with any of the dotNET languages
2. **Quadra4CSservo Application** .. full multi-axis multi-threaded application for simple PID servo control of up to 5 Quadra-4s (20 axes)
3. **Quadra4VB6servo Application** .. full multi-axis **VB6** single-threaded application for simple PID servo control of up to 5 Quadra-4s (20 axes)
4. **Quadra4Diagnostic Application** .. a **VB6** Windows tool that allows quick verification of Quadra-4 operation, manual setting of any Quadra-4 feature
5. **Quadra4VBNServo Application** .. full multi-axis **VB.NET** multi-threaded application for simple PID servo control of up to 5 Quadra-4s (20 axes) .. *future release date unspecified.*

Details and Usage of Software Suite Components:

Quad4.dll –

General Usage in the Quadra4CSservo Application:

Quad4.dll provides all the properties and methods needed to operate all of the features of a Quadra-4 device. The properties and methods that operate on any of the 4 encoder interfaces are indexed making it easy to implement a single software construct which can operate four independent servo loops ... typically with a simple

```
for (i = 1; i <= 4; i++) { RunServo(i) }      code structure. Typically your RunServo()
```

(as in the demo application) is operated in a separate thread from the display window so the GUI remains responsive as the servos are serviced repeatedly. The **Quadra4CSservo Application** referenced above does exactly that. You can download the dll, along with the 2 required support DLLs, at <http://www.versalent.biz/downloads/Quad4DLL.zip>

Download the installable version of Quadra4CSservo at <http://www.versalent.biz/downloads/Q4CSservo.zip>

Download the full Vis Studio 2005 source code for Quadra4CSservo at <http://www.versalent.biz/downloads/Q4CSservoSource.zip>

Multiple Quadra-4s can be connected to a single PC. This application creates a new display window for each Quadra-4, and each of these display windows instantiates a new Quad4 object. Quadra-4 devices are uniquely identified by a serial number on the bottom of the unit housing, and each window presents a list of the auto-discovered Quadra-4 serial numbers allowing a window (and its Quad4 object) to be connected to one selected Quadra-4 device. As structured above, each window creates a new thread for its RunServo() function so the Quadra4CSservo Application operates in several threads – one thread for all of the display windows, and one additional thread for each Quadra-4 device.

Operation is simple .. once the application starts, the operator can choose from 1-5 Quadra-4's to operate, that number of display windows and Quad4 objects are created, the operator can then choose which window is connect to which Quadra-4 (device) serial number by initializing a different serial number in each window. Each window creates a separate servo thread which is ready to go as soon as the Run Servos button is pressed.

Although the provided application is a demo application only, it can serve as a model for your own code, or can be the base-code which you modify to meet your own needs. Either way, your servos can be operational quickly and without writing any (or very little) code since full source code (MS Visual Studio 2005, framework 2.0) is provided. You can use this code license-free in any way you like. Versalent will in the future provide a VB.net equivalent code base which will use this same dll for its Quadra-4 device interface.

Quad4 object Properties

(note default values in effect after Quadra-4 powerup, and/or initialization)

(x in the following properties = 1-4 corresponding to the Quadra-4 Encoder Interface #)

DeviceInitialized .. access is read only, returns true if the object has been successfully initialized, or false if not initialized.

encoderMultiplier[x] access is read/write ,valid settings are 0-2. Any attempt to set to an invalid value is ignored.

0 = X1 encoder multiplication

1 = X2 encoder multiplication

2 = X4 encoder multiplication (default)

analogRange[x] .. access is read/write ,valid values are 0-2. Any attempt to set to an invalid value is ignored.

0 = 0 volts to +10 volts full scale

1 = -5v to +5v full scale

2 = -10v to +10v full scale (default)

encoderTerminator[x] .. access is read/write, valid values are 0-1. Any attempt to set an invalid value is ignored. Note that this single property controls the three 100 ohm terminators between differential encoder phases PHA and /PHA, PHB and /PHB and INDEX and /INDEX

0 = Terminators OFF (default)

1 = Terminators ON

indexStyle[x] .. access is read/write, valid values are 0-1. Any attempt to set to an invalid value is ignored.

0 = EncoderPosition[x] saved in IndexPosition[x] on index[x] pulse (default)

1 = Encoder Counter[x] cleared to 0 on index[x]

encoderPosition[x] .. access is read only .. returns the Unsigned 32-bit integer from the Quadra-4's 32-bit encoder position counter[x]

indexPosition[x] .. access is read only .. returns the Unsigned 32-bit integer (the last encoderPosition[x] saved on last index[x] pulse)

digitalOutputLevel ..access is read/write, valid values are 3 or 5. Any other value is ignored

3 = Digital Outputs 3 volt maximum high level (default)

5 = Digital Outputs 5 volt maximum high level

digitalOutputs .. access is write only, valid values are 0x00 thru 0xff. Each bit position 0-7 corresponds to Quadra-4 DIG0-7 (output). A '1' sets the associated bit HIGH, '0' sets it LOW. Bits not previously configured as OUTPUT are ignored.

digitalInputs .. access is read only, returns 0x00 thru 0xff. Each bit position 0-7 corresponds to Quadra-4 DIG0-7 (inputs). Bits not previously configured as INPUTS are read as '1'.

(Maximum Low Input Threshold = $0.2 * \text{max high level}$, 3 or 5v)

(Minimum High Input Threshold = $0.8 * \text{max high level}$, 3 or 5v)

digitalIODirections ..access is read/write, valid values are 0x00 thru 0xff. Each bit position 0-7 corresponds to the direction control for the associated bit. A '0' bit sets the corresponding I/O to INPUT, '1' sets it to OUTPUT. (default is 0x00 meaning all INPUTS).

FPGAversion .. access is read only .. returns a string of 2 to 4 characters identifying the current FPGA logic version installed in the Quadra-4 device. (i.e. "1.4")

firmwareVersion .. access is read only .. returns a string of 2 to 5 characters identifying the current microcontroller firmware version installed in the Quadra-4 device. (i.e. "1.3")

Methods

String ReadReport() .. executing this method transfers the most recently arrived USB 'report' data into the Quad4 object which can then be read by your application using methods and properties. So typically a software loop will be established which executes ReadReport() followed by reading the encoderPosition[x] properties. Note that the Quadra-4 device sends reports continuously every 1 millisecond and does **not** need to be interrogated to do so. This is annunciated by the USB OUT LED which appears lit continuously (although it actually flashes once briefly for each USB report which is sent). So data is delivered automatically to the host's USB buffers and is available for retrieval immediately using ReadReport() -- as long as you don't read at intervals of less than 1ms. If you do, the thread in which the ReadReport() occurs will wait until the next 1ms USB cycle completes. An empty return string indicates success, otherwise it will contain a description of the error.

String WriteReport() .. executing this method sends the current Quad4 object properties/data to the Quadra-4 device. Note that this report is NOT automatically sent from the host. It is only sent when you execute this method (or the special PreloadCounter() method). So the Quadra-4's USB INPUT LED remains off and only flashes briefly when your application executes this method – and when running a servo

loop, this occurs often enough to make the LED appear to be on continuously. There is a slight delay in writing data. When this method is executed, data is transferred from the Quad4 object into the USB interface buffers. On the next (asynchronous) 1ms USB cycle that data is sent to the Quadra-4 device, and on its microcontroller's asynchronous cycle the data is read from the Quadra-4's USB buffers and updated to the hardware. So a digital I/O bit change may be delayed as much as 1ms within the host, and an additional 250usec within the Quadra-4 , therefore appearing at the output of the Quadra-4 delayed as much as 1.25ms . And depending on the host's thread-slicing, the delay can be even greater if the host postpones your servo thread execution at an inopportune time. An empty return string indicates success, otherwise it will contain a description of the error.

Bool SetAnalogOutput(int AxisNumber, Uint16 Value)

AxisNumber = 1 to 4 corresponding the Quadra-4 Encoder Interface Number

The following table shows three sets of values and the resulting servo output voltages which will be generated. 0x3FF represents 10-bits of binary data and therefore the maximum value which can be accepted by the Quadra-4's 10-bit DACs.

Analog Range	Value	Output Voltage
0 (0 to +10V)	0x000 (min value)	0.0 volts
0 (0 to +10V)	0x1FF (half scale)	5.0 volts
0 (0 to +10V)	0x3FF (full scale)	10.0 volts
1 (-5V to +5V)	0x000 (min value)	-5.0 volts
1 (-5V to +5V)	0x1FF (half scale)	0.0 volts
1 (-5V to +5V)	0x3FF (full scale)	+5.0 volts
2 (-10V to +10V)	0x000 (min value)	-10.0 volts
2 (-10V to +10V)	0x1FF (half scale)	0.0 volts
2 (-10V to +10V)	0x3FF (full scale)	+ 10.0 volts

Returns true for successful, false for error

Bool PreloadCounter(int AxisNumber, UInt32 Value)

AxisNumber = 1 to 4 corresponding the Quadra-4 Interface Number

Value = 32-bit unsigned integer value to load into counter. Loads immediately because this method executes a WriteReport() before returning.

Returns true for successful, false for error

String GetDeviceSerialNos(Uint ProdID, Uint VendorID)

ProdID = 0x82CD for Quadra-4

VendorID = 0x10C4 for Versalent

Returns a semicolon separated string containing all the serial numbers of Quadra-4 devices found attached to the computer. User can then split() this string using the semicolon as the delimiter to extract all serial numbers.

Messages InitTheHID(string SerialNumber)

This command initializes the specified Quadra-4 getting it ready to start returning its position information, and accept Analog and digital I/O commands. Messages is an object containing multiple strings which are status strings describing the initialization progress.

Void Shutdown (void)

Breaks the connection between the Quad4 object and the Quadra-4 device which was created during initialization. The Quad4 object is then available to be connected to the same, or a different Quadra-4 device.

Simplified ServoLoop Algorithm Example:

```
Quad4 Q4 = new Quad4(); //create a Quadra4 object
Serials = Q4.GetDeviceSerialNos(0x82CD, 0x10C4) //ProductID, VendorID
Q4.InitTheHID(a_serialNumber) //initialize one of them

While (exit == false) //repeat continuously
{
    Q4.ReadReport(); //get USB position data
    For (i = 1; i < 5;i++)
    {
        Drive= RunServo(Q4.encoderPosition[i]); //calculate appropriate drive signal
        Q4.SetAnalogOutput( i, Drive); //sends drive signal to Q4 object
    }
    WriteReport(); //transfer Q4 data to Quadra-4 via USB
}
*****
```

The example servoloop code above shows the basic structure of a C# servo application. Once a Quad4 object is created and initialized (connected to a Quadra-4's serial number), a repeated loop of reading position, computing a new drive voltage, and sending that drive to the servo output... is executed. The characteristics of your loop are contained within your RunServo() function above. This could be a simple proportional position control, full PID control, velocity control or other. The Versalent applications provide a simple PID drive calculation, and the RunServo() is actually executed in its own thread .. one for each Quadra-4 connected.

Notice that ReadReport() and WriteReport() are executed only once for all 4 servo axes. This is a performance feature. As described above, if WriteReport() methods are executed more often than the USB messaging rate, the method may block the thread from further execution awaiting a USB message before returning. So all 4 servo outputs are computed, the SetAnalogOutput() method is executed 4 times to set the Q4 object values, and only one WriteReport() is executed per loop which sends all servo outputs to the Quadra-4 in one USB message.

Quadra4CSservo Application Description

This application demonstrates the use of the Quad4 dll. The Startup form asks the user how many Quadra-4's to operate and it creates that many GUI interface windows. Each GUI window creates its own Quad4 object and a thread to run its 4 servo axes. From the list of available Quadra-4 serial numbers found using the Quad4.GetDeviceSerialNos() method, each window can initialize (connect to) one Quadra-4. The operator cannot initialize the same Quadra-4 in multiple windows .. only one window can drive any particular Quadra-4.

When the Start Servos button is pressed, a simple PID servo loop is initiated on each axis of each initialized Quadra-4. The operator can select different PID gains to stabilize each loop. Although it is common practice to marshal the screen data (positions, drives etc) from the RunServo() thread which reads/calculates them to the GUI thread, the overhead of that marshaling is oppressive and slows processing considerably. Since all of those GUI fields are written only by the RunServo() thread, it is acceptable to disable thread protections so that the RunServo() thread can write directly to the GUI thread controls without the overhead of marshaling. Performance is improved considerably and it is safe to do so since there is no write-contention from the GUI thread. The Window property CheckForIllegalCrossThreadCalls property must be set to false to allow compiling with what Visual Studio considers 'dangerous' cross-thread writes.

Quadra4DiagnosticApplication is available at <http://www.versalent.biz/downloads/Q4diag.zip>... This application is provided as a quick system-checker – it allows you to manually set/exercise all Quadra-4 features. This ‘tester’ implements no servo loops, but allows you to manually control all of the Quadra-4 features, to monitor and validate proper operation. Closed loop servo systems are often difficult to debug and this application provides a way to break the loop and fully test a single Quadra-4 .. from encoder counting, to analog output voltage control, to digital IO, to index pulse operation. Since this is a diagnostic application, no source code is provided and it therefore cannot be modified. It provides your base-line reference testing when a hardware problem is suspected since it allows for a very quick validation of all Quadra-4 features.