Picard Industries PiUsb.dll for the USB Twister

Version 1.0 – July 29, 2010

Library Overview

The PiUsb library is the software required to write your own programs for use with any Picard Industries USB product.

The PiUsb library can be used from any programming language or application that can call functions in a DLL (Dynamic Link Library). This includes:

- Visual C++
- Visual C#
- Visual Basic
- Delphi

The PiUsb library supports Windows 2000, Windows XP, Windows Vista, and Windows 7. It supports development of 32-bit applications, which can be used on 32-bit or 64-bit versions of Windows.

At the time this documentation was written, the sample application was version 1.0, and PiUsb.dll was version 1.9.

How to use the Library

C++:

Copy these files to your project source code folder:

PiUsb.h
PiUsb.def
PiUsb.lib

PiUsb.h contains the function prototypes for all the functions available in the dll. It also contains #define statements to define useful constants for your program. You should "#include" this file in any source code file that references the DLL.

PiUsb.def and PiUsb.lib define all the functions in the dll. One or both of these files may be used by the linker to resolve the function references.

Copy this file to your executable folder:

PiUsb.dll

USB Twister Position

The USB Twister uses a stepper motor to rotate a shaft. You command the motor to move to a new position with the `piRunTwisterToPosition()` function, specifying the destination position in steps (or counts). There are 200 steps for each revolution of the shaft, so each step corresponds to 1.8 degrees. You can specify any position between -32767 and +32767. If you specify a position less than -32767, the motor will move to position -32767. Similarly, if you specify a position greater than +32767, the motor will move to position +32767.

You can read the current position at any time (while moving or not) with the `piGetTwisterPosition()`

function. You can reset the position to zero at any time with the `piSetTwisterPositionZero()` function.

You can command the motor to move continuously in either the + or – direction with the `piRunTwisterContinuous()` function. When you issue this command, the position is set to zero and remains there during the continuous move. You can leave the motor running for as long as you wish.

The positive direction (increasing counts) is counter-clockwise rotation if you are looking at the USB Twister from the shaft end.

USB Twister Velocity

The `piRunTwisterToPosition()` and `piRunTwisterContinuous()` functions let you specify the move velocity. The velocity is a number between 1 and 13, where 1 is the slowest speed and 13 is the highest speed.

We recommend that you limit your velocity to be 10 or less. These velocities should work well with most devices you connect to the Twister. If your load is small and light, you may be able to use some or all of the faster velocities between 11 and 13. If you attempt to move too large a load at too high a speed, the motor may stall, or may miss steps and not move the full distance.

The following table shows the velocity settings and the approximate speed they corresponds to.

Approximate Speeds

Velocity	Counts/Sec	Degrees/Sec	RPM
1	133	240	40
2	143	257	43
3	154	277	46
4	167	300	50
5	182	328	55
6	200	360	60
7	222	400	67
8	250	450	75
9	286	514	86
10	333	600	100
11	400	720	120
12	500	900	150
13	667	1200	200

piConnectTwister

Connect the program to the USB Twister and return a device pointer that must be used in all other function calls that refer to this device.

Function prototype:

C++:
```
void * __stdcall piConnectTwister(int * ErrorNumber, int SerialNumber);
```

Visual Basic:
```
Declare Function piConnectTwister(ErrorNumber as Long, ByVal SerialNumber as Long) as Long
```

Parameters:

ErrorNumber Returns the error number upon completion of the function. Valid values are:

```
PI_NO_ERROR
PI_DEVICE_NOT_FOUND
PI_CANNOT_CREATE_OBJECT
```

SerialNumber The serial number of the USB Twister (printed on the side of the USB Twister).

Return value:

Returns a pointer to the device. You must pass this pointer into all other PiUsb Library functions that use this device.

If *ErrorNumber* is anything other than `PI_NO_ERROR`, then `NULL` is returned.

Example (C++):

Connect to USB Twister with serial number 12.

```
int ErrorNumber;
void * pUsb1 = piConnectTwister(&ErrorNumber, 12);
```

piDisconnectTwister

Disconnect the program from the USB Twister. After completion of this function, the pointer returned by `piConnectTwister()` is no longer valid, and should be set to `NULL`.

Function prototype:

C++:
```
void __stdcall piDisconnectTwister(void * DevicePtr);
```

Visual Basic:
```
Declare Sub piDisconnectTwister(ByVal DevicePtr as Long)
```

Parameters:

DevicePtr The device pointer that was returned by the `piConnectTwister()` function.

Return value:

None.

piGetTwisterMovingStatus

Return a value indicating whether the USB Twister is moving.

Function prototype:

C++:
```
int __stdcall piGetTwisterMovingStatus(BOOL * Moving, void * DevicePtr);
```

Visual Basic:
```
Declare Function piGetTwisterMovingStatus(Moving as Long, ByVal DevicePtr as Long) as Long
```

Parameters:

Moving Returns the value 1 (TRUE) if the USB Twister motor is moving, and 0 (FALSE) if the USB Twister motor is stopped.

DevicePtr The device pointer that was returned by the `piConnectTwister()` function.

Return value:

Returns the error number. Valid values are:

```
PI_NO_ERROR
PI_DEVICE_NOT_FOUND
```

Example (C++):

```
int Moving;
int ErrorNumber = piGetTwisterMovingStatus(&Moving, pUsb1);
```

piGetTwisterPosition

Return the position of the USB Twister.

Function prototype:

C++:
```
int __stdcall piGetTwisterPosition(int * Position, void *
DevicePtr);
```

Visual Basic:
```
Declare Function piGetTwisterPosition(Position as Long, ByVal
DevicePtr as Long) as Long
```

Parameters:

Position Returns the current position of the USB Twister motor in steps.

DevicePtr The device pointer that was returned by the `piConnectTwister()` function.

Return value:

Returns the error number. Valid values are:

```
PI_NO_ERROR
PI_DEVICE_NOT_FOUND
```

Example (C++):

```
int Position;
int ErrorNumber = piGetTwisterPosition(&Position, pUsb1);
```

piGetTwisterVelocity

Return the velocity of the current or most recent Run command.

Function prototype:

C++:
```
int __stdcall piGetTwisterVelocity(int * Velocity, void *
DevicePtr);
```

Visual Basic:
```
Declare Function piGetTwisterVelocity(Velocity as Long, ByVal
DevicePtr as Long) as Long
```

Parameters:

Velocity Returns the current velocity of the USB Twister motor, a number between 1 and 13.

DevicePtr The device pointer that was returned by the `piConnectTwister()` function.

Return value:

Returns the error number. Valid values are:

```
PI_NO_ERROR
PI_DEVICE_NOT_FOUND
```

Example (C++):

```
int Velocity;
int ErrorNumber = piGetTwisterVelocity(&Velocity, pUsb1);
```

piHaltTwister

Halt motion of the USB Twister.

Function prototype:

C++:
```
int __stdcall piHaltTwister(void * DevicePtr);
```

Visual Basic:
```
Declare Function piHaltTwister(ByVal DevicePtr as Long) as Long
```

Parameters:

DevicePtr The device pointer that was returned by the `piConnectTwister()` function.

Return value:

Returns the error number. Valid values are:

```
PI_NO_ERROR
PI_DEVICE_NOT_FOUND
```

Example (C++):

```
int ErrorNumber = piHaltTwister(pUsb1);
```

piRunTwisterContinuous

Start the USB Twister moving continuously. The motor position will be set to zero and will remain at zero during continuous motion.

Function prototype:

C++:
```
int __stdcall piRunTwisterContinuous(int Direction, int Velocity,
void * DevicePtr);
```

Visual Basic:
```
Declare Function piRunTwisterContinuous(ByVal Direction as Long,
ByVal Velocity as Long, ByVal DevicePtr as Long) as Long
```

Parameters:

Direction — A value of 1 (or larger) means to move in the positive direction. A value of 0 (or negative) means to move in the negative direction.

Velocity — The velocity to use for the motion, a number between 1 and 13.

DevicePtr — The device pointer that was returned by the `piConnectTwister()` function.

Return value:

Returns the error number. Valid values are:

```
PI_NO_ERROR
PI_DEVICE_NOT_FOUND
```

Example (C++):

```
int ErrorNumber = piRunTwisterContinuous(1, 5, pUsb1);
```

piRunTwisterToPosition

Move the USB Twister to a specified position.

Function prototype:

C++:
```
int __stdcall piRunTwisterToPosition(int Position, int Velocity,
void * DevicePtr);
```

Visual Basic:
```
Declare Function piRunTwisterToPosition(ByVal Position as Long,
ByVal Velocity as Long, ByVal DevicePtr as Long) as Long
```

Parameters:

Position The destination position, in counts. This should be a value between -32767 and +32767.

Velocity The velocity to use for the motion, a number between 1 and 13.

DevicePtr The device pointer that was returned by the `piConnectTwister()` function.

Return value:

Returns the error number. Valid values are:

```
PI_NO_ERROR
PI_DEVICE_NOT_FOUND
```

Example (C++):

```
int ErrorNumber = piRunTwisterToPosition(2000, 5, pUsb1);
```

piSetTwisterPositionToZero

Set the current position to zero. If the motor is moving, it will be halted before setting the position to zero.

Function prototype:

C++:
```
int __stdcall piSetTwisterPositionToZero(* DevicePtr);
```

Visual Basic:
```
Declare Function piSetTwisterPositionToZero(DevicePtr as Long) as
Long
```

Parameters:

DevicePtr The device pointer that was returned by the `piConnectTwister()` function.

Return value:

Returns the error number. Valid values are:

```
PI_NO_ERROR
PI_DEVICE_NOT_FOUND
```

Example (C++):

```
int ErrorNumber = piSetTwisterPositionToZero(pUsb1);
```

C++ Example

```
#include “PiUsb.h”

void *      pUsb1;
int         ErrorNumber;
int         TwisterSerialNumber = 12;
int         Position;

pUsb1 = piConnectTwister(&ErrorNumber, TwisterSerialNum);
if (ErrorNumber != PI_NO_ERROR)
      AfxMessageBox("Unable to find USB Twister.");
else
      AfxMessageBox("USB Twister Connected.");

// Move to position 2000 at velocity 5
ErrorNumber = piMoveTwisterToPosition(2000, 5, pUsb1);
if (ErrorNumber != PI_NO_ERROR)
      AfxMessageBox( "USB Twister was disconnected." );

ErrorNumber = piGetTwisterPosition(&Position, pUsb1);
if (ErrorNumber != PI_NO_ERROR)
      AfxMessageBox( " USB Twister was disconnected." );
else
{
      Cstring s;
      s.Format(“Position is %d”, Position);
      AfxMessageBox(s);

piDisconnectTwister(pUsb1);
pUsb1 = NULL;
```