LCP-USB Inclinometer sensor

DLL Interface library description

Description

The LCP-USB sensor is connected to a USB host device (typically a PC) with a standard 4 pin USB-A connector, the LCP-USB interface is USB 2.0 compatible, the sensor has a maximum USB message rate of 100 messages per second.

In operation the sensor samples the two inclination axes, applies digital filtering to the sampled data and outputs the results as X and Y readings calibrated in degrees.

Customers can develop their own applications to interface to the LCP-USB device by including references to Level Developments provided Windows DLL files, the functions contained in the DLL files are detailed in this document together with basic examples of usage.

USB driver

The LCP-USB has a HID USB interface, and is supported by two Level Developments supplied DLL files HidLibrary.dll and LCPLibrary.dll, when developing applications for the LCP-USB device the DLL files must be included in the application directory (e.g. same directory as the EXE file), the DLL files should also be included as 'References' in the user application.

The DLL files are compatible with 32-bit or 64-bit applications.

LCPLibrary.dll functions

The library has a single class called LCPDevice. In the users c# code for example, this is declared like this:

```
        private static LCPDevice sensor = new LCPDevice();
```

And thereafter the functions are called using sensor.*Function()*;

For example, to get the X axis value:-

```
double XAngle;
int XVal;

XVal = sensor.Get_X();                    // get scaled integer value of X axis angle

if (XVal == -1000000)
       // there is an error, do something..

else   // value is good, rescale to floating point degrees
       XAngle = (double)( XVal / 1000.0);
```

All functions which return an int, return the value -1000000 in the case of error. This indicates no response or timeout to the request sent to the sensor. Usually this is caused by the sensor not being connected.

List of Functions

`Bool Initialise(int VendorId, int ProductId)`
Sets up the library and connects to the sensor using the Vendor ID and Product ID. For the LCP_USB, the Vendor ID is 0x461 and the Product ID is 0x21. If the operation is successful, true is returned, otherwise false.

`Void Dispose()`
Finishes operation with the sensor and closes the connection to the sensor tidily. This must be called when closing the user app.

`int Get_Product`
Reads the product code from the sensor. The LCP-USB sensor is code 2. This is not adjustable

`int Get_Serial()`
`Reads the serial number from the sensor.` Set at the factory, not adjustable.

`int Get_X()`
`Reads the X Axis angle from the sensor` in degrees, scaled by 1000. i.e. a value of 11234 equals 11.234 degrees

`int Get_Y()`
Reads the Y Axis angle from the sensor in degrees, scaled by 1000. i.e. a value of 11234 equals 11.234 degrees.

`int Get_360()`
Reads the 360 degree angle from the sensor in degrees, scaled by 1000. i.e. a value of 11234 equals 11.234 degrees.

`int Get_Filter()`
`Reads` the filter setting from the sensor.

Index	Frequency
0	0.125Hz
1	0.25Hz
2	0.5Hz
3	1.0Hz
4	2Hz
5	4Hz
6	8Hz
7	16Hz
8	32Hz

int Get_X_Direction()
Reads the X direction setting.

0 = normal
1 = reversed

int Get_Y_Direction()
Reads the Y direction setting.

0 = normal
1 = reversed

int Get_360_Direction()
Reads the 360 direction setting.

0 = normal
1 = reversed

int Get_Axes()
Reads the setting number of axes from the sensor.

1 = single 360 degree axis
2 = separate X and Y axes

`void Set_Filter(byte index)`
Sets the filter in the sensor.

Index	Frequency
0	0.125Hz
1	0.25Hz
2	0.5Hz
3	1.0Hz
4	2Hz
5	4Hz
6	8Hz
7	16Hz
8	32Hz

`void Set_X_Direction(byte direction)`
Sets the X axis direction

0 = normal
1 = reversed

`void Set_Y_Direction(byte direction)`
Sets the Y axis direction

0 = normal
1 = reversed

`void Set_360_Direction(byte direction)`
Sets the 360 axis direction

0 = normal
1 = reversed

```
void Set_Axes(byte n)
```
Sets the number of axes in the sensor

1 = single 360 degree axis

2 = separate X and Y axes

```
void Set_Zero(byte n)
```
Sets the zero point in the sensor

0 = absolute

1 = relative