last update 6-28-17

LL_USB_min sample app for Linux – overview

Sample app "LL_USB_min" makes use of "libLL_USB30x.so" built and tested in Oracle VM VirtualBox, within Win7, running ubuntu 14.04 LTS. Built with installed library **lusb1.0** and library, **pthread 1.1**. "libLL_USB30x.so" will need to be built and installed as per the instructions accompanying it in order for this app to make use of it.

The library makes use of many of the functions mentioned in the documentation for our Windows DLL API which can be found at our website. The link below provides the API as well as much more information that further explains using the functions, and some information concerning interaction with the hardware. Below is the link:

lawsonlabs.com web link

The functions currently available can be found in the "main.h" header file distributed with the library source code, and recognized by the "EX_" preceding the rest of the function name. There are also many other functions within the library which wrap, extend, and are called by those functions, but can also be called by themselves. All are shown within the "main.h" header file distributed with the library source code.

Sample app LL_USB_min makes use of only a handful of the functions available, in order to demonstrate their usage in connecting to the device and performing some basic tasks. The app connects to the device, reads/displays the voltage from the default channel 0, sets channel 6, and then reads/displays the voltage again. Next the app sets DAC 0 to 3.5 volts and changes to channel 3. Please connect DAC 0 to channel 3 (on board: analog out 1 to 4+ and GD to 4-). The app reads the voltage from that channel (should read approx. 3.5 volts). It then demonstrates how to get the "actual rate" the board will run at based on a requested rate, since the board is not capable of always running at the precise rate one might request. It sets a temporary rate variable to 1000(Hz) and then calls into the library to get what the "actual board rate" would be at that requested rate. Note that trying to set the rate, for example with a call to **EX_SendRate(...)** with the rate that was returned by that call, could then set it to a different rate. That returned rate should only be used within an app for timing within the app. Finally, the app does a single-channel 100Hz scan on channel 3, displaying the results for until a key is pressed, and then exits.

It makes use of the following function calls within the library:

EX_ConnectOneDevice(...)
EX_GetOneConversion(...)
EX_SendChan(...)
EX_SendDAC(...)
EX_GetCalculatedRate(...)

the following is also used by this app but is not shown in the Windows DLL API at our website as it is exclusive to our Linux library and is the function passed to the scan thread:

doScan(...)

The device ID is hard-coded to 5206 at the top of the "main()" function call so that will need to be changed to "your" device ID before recompiling for your Linux system. Running may require super user rights, "*sudo ./LL_USB_min*."