libLL_USB30x.so and python sample app overview

library coded in C (gcc compiled) and app in python for Linux – overview

Library "libLL_USB30x.so" built and tested in Ubuntu 18.04.6 LTS.

Built with installed library " libusb-1.0-0:amd64 2:1.0.21-2, " library, "libpthread-2.27.so," “Python 2.7.17,” and compiled with “gcc version 7.5.0.”

The library makes use of some of the function “names” mentioned in the documentation for our Windows DLL API which can be found at our website at:
lawsonlabs.com
web link just to make it a bit easier for those who may have interacted with that DLL and now working in Linux. Below are the function calls accessible with calls into our library which replicate those in our Windows DLL.

```
BOOLEAN EX_ConnectOneDevice(USHORT usDevID, double dblRate, BYTE bChan)
BOOLEAN EX_SendChan(USHORT usDevID, BYTE bChan, BYTE* pbLastDigin)
BOOLEAN EX_SendDAC(USHORT usDevID, DOUBLE dblVoltage, BYTE bDAC)
BOOLEAN EX_GetOneConversion (USHORT usDevID, double* pdblMilliVolts)
BOOLEAN EX_GetCalculatedRate (USHORT usDevID, double* pdblRate)
BOOLEAN EX_SendDigout(USHORT usDevID, BYTE bDigOut)
BOOLEAN EX_GetDigin(USHORT usDevID, BYTE* bDigIn)
BOOLEAN EX_SendRate(USHORT usDevID, double* dblRate)
```

All functions currently available can be found in the "main.h" header file distributed with the library code. There are many other functions within the library which wrap, extend, and are called by the EX_ functions, but can also be called by themselves. All are shown within the "main.h" header file and can be included, as can any of the variables, by referencing them in the usual way for python type access as shown in the sample python code.

The example python app makes use of most of the functionality within the library, coded in C. The python app performs some basic calls into our 30x USB library and prints the results directly to the console screen with user interaction.

The python app is intended to be used with out model 302 board, and, as part of the user interaction, a device ID is entered which is used to connect to the device.

The “main.c” file, included with the library code, contains build commands shown at the top in case one may desire to modify the library to fit a specific need. The command line compilation strings for the library are self-explanatory as to where the library is to be placed. The python sample, has no header and is executed in the typical way at the command line.

The python app and library are intended to be used as an example for a developer to develop their own code. Specific places of interest are marked with the text:
LOOK HERE
in order to simplify the understanding of the most specifically the library, and to a lesser degree the python interaction with the library.

Special notes about device disconnect

There are various ways to determine if a device has been disconnected, something that may be of most concern while in scanning mode. If a device is disconnected or loses power while scanning, and the "*fAllowPrintf*" flag is set, then the library will automatically notify the app. There are various flags and other variables that can also be read to determine what happened. For example "*fBadScanEnd*" and "*fPossibleScanUSB_Err*" should be checked if a scan seems to have ended prematurely. If either of those are set there was a scan related problem. When a scan ends normally, the library checks the end-scan return tokens to make sure they are correct and if they are, it will set "*fGotCorrectEndScanCodes*" true. The end-scan codes are also placed into a global array pointed to by "*paEndScanCodes*" which the app can read. If there are no indications that the scan's thread terminated prematurely due to an error, the app should wait until "*fScanThreadWait*" is set to false before proceeding on to other tasks.

At any time, the app can call "*checkConnectLLabs(...)*" and on return, check "*bIsBrdConnected*" and "*bIsDevConnected*" to see if the board is connected and the device ID requested is available. That call can also be used, and the flags tested at any time to see if a board is connected and the desired device ID is available.