updated: 7-17-17

LL_USB_big sample app for Linux – overview

Sample app “LL_USB_big” 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.

App was built and tested in ubuntu 14.04 LTS using ncurses version 5.9+20140118. and library lusb1.0. Code is well documented and LL_USB_big.h contains line to use to build application for easy copy and paste to command line:

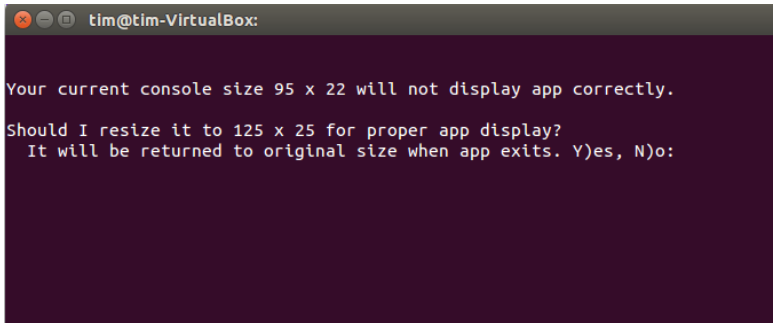
sudo gcc -o LL_USB_big LL_USB_big.c scanFuncs.c -lLL_USB30x -lm -pthread -lncurses &>makeout

Source code files are:

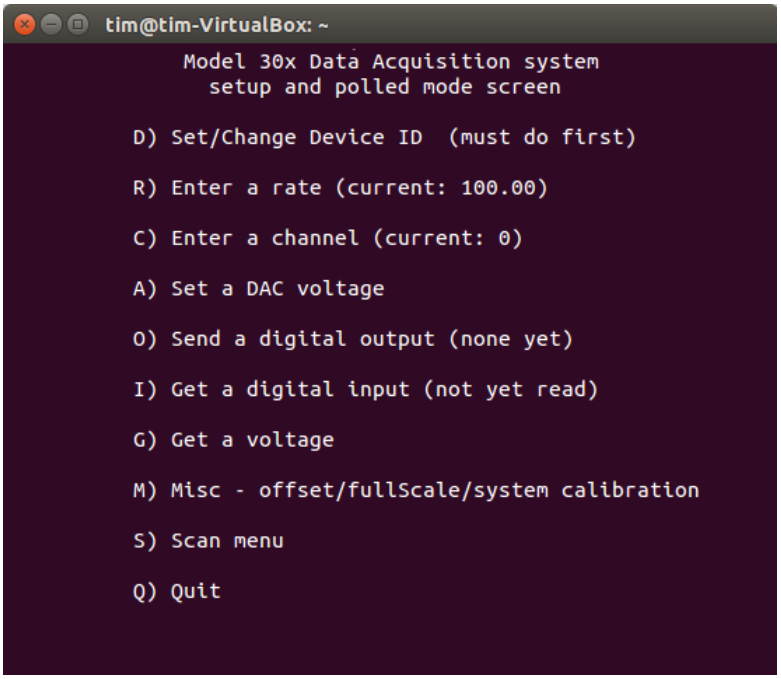
LL_USB_big.c: main source file which contains function main(...)
LL_USB_big.h: main header file
scanFuncs.c: mostly scan related functions including function that processes scan data.
helperFuncs.c: various functions used mostly by LL_USB_big.c, but also scan data-taking thread.

Operation Instructions

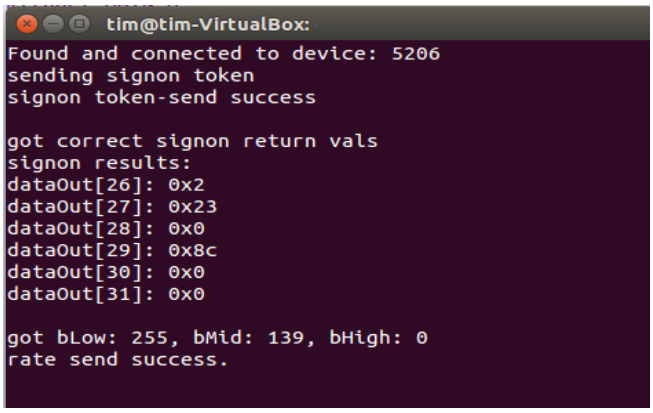
When app is started, which may require super user rights, “*sudo ./LL_USB_big*,” if console is smaller than 125x25 (required to show all 7 channels in "multi-chan calibration w/digital input" scan mode), you'll be asked if you'd like to have it resized. If the answer is yes, the console screen will be resized, if possible. If the answer is "no" and the console is less than the very minimum size for the app's menus to display (80x25), the app will not run until the console is somehow manually resized by you the user/developer. If app resizes your console screen it will be returned to it's original size when the app exits.



The first screen (cropped) after the console resizing (if that was necessary) is the main "setup and polled mode" screen which is where one makes the initial connection to one of our m30x devices. Pressing "D" on the keyboard will display a prompt at the bottom left of the screen to enter an ID, after which the app will attempt to connect to it.

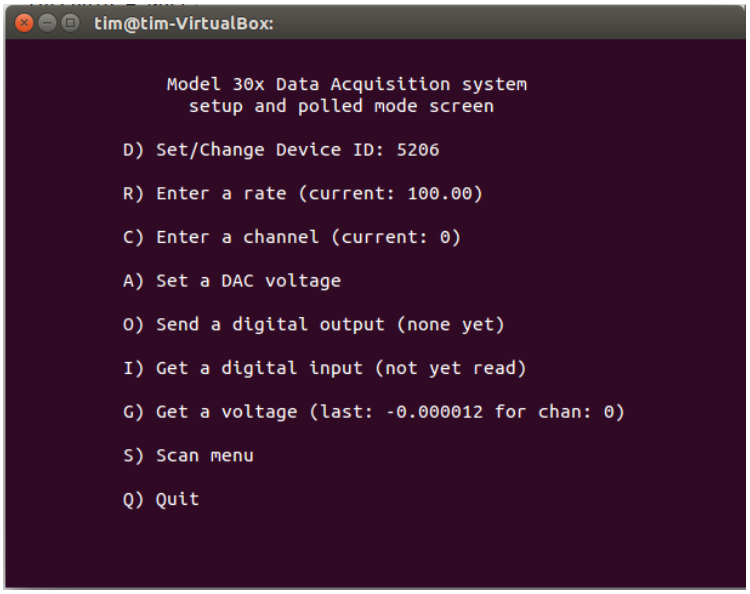


As it connects, a screen of various connect and sign-on related information will be displayed. It will also attempt to get a voltage conversion on channel 0, which is the default channel when initially signed on. The initial rate is 100Hz.



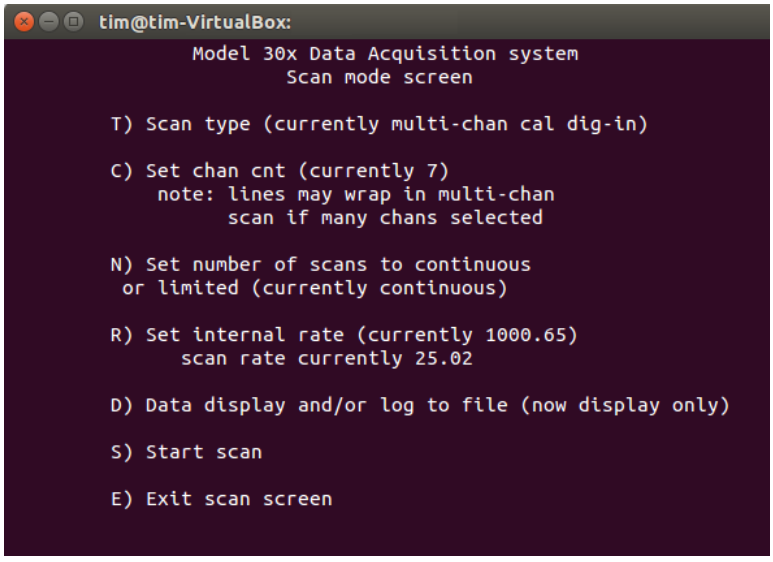
Once connected, the device ID will show on the panel, as well as the conversion result. One should verify that the “sign-on return vals” are the same as shown in the screenshot, and that the conversion result is correct for channel 0 before proceeding. If you miss seeing something, the process can be repeated any time.

Other menu items on this screen also present prompts at the bottom left of the screen when the appropriate key is pressed, except the "S" menu item which will open a "scan menu" screen.

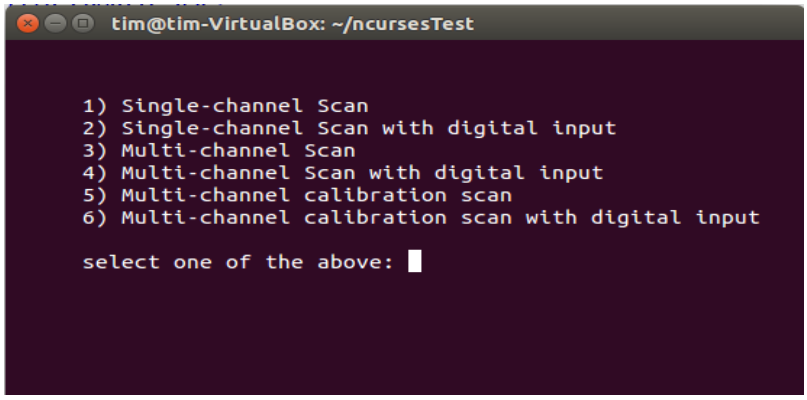


As with the polled menu screen, each menu item will display a prompt of options to be selected. The default is single channel scan, and it will continue to scan until the <ENTER> key is pressed.

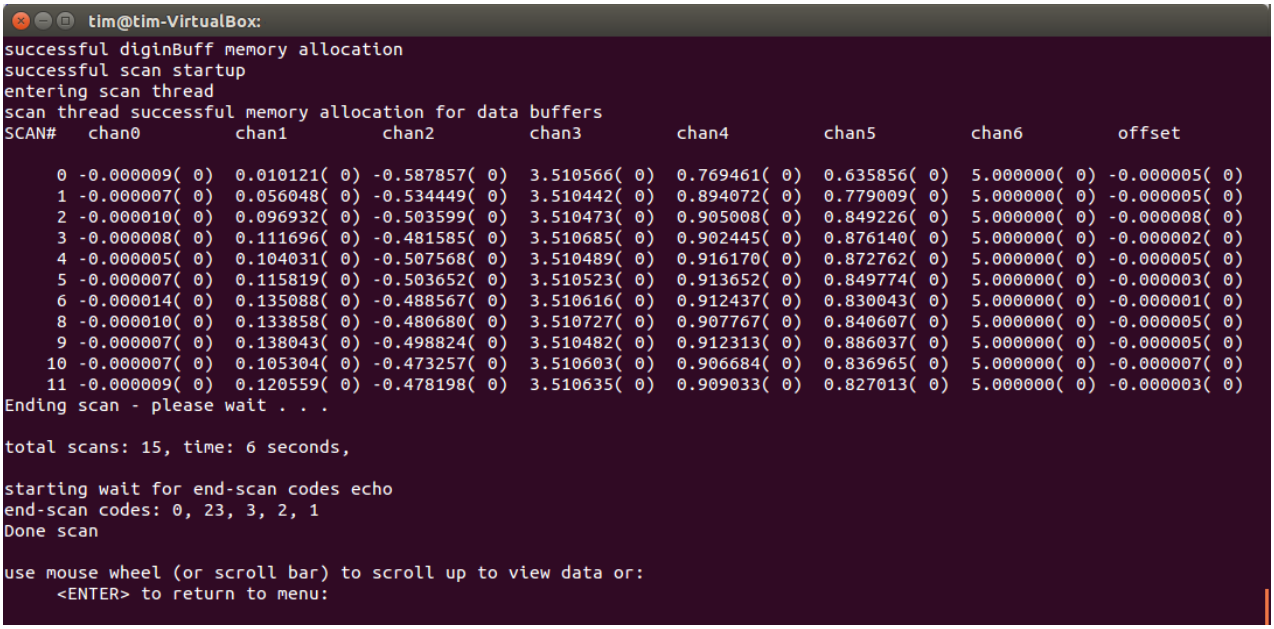
One item in the scan menu screen that may be confusing is the relationship between the "internal data rate" and "scan rate." The internal rate is the rate the board is set to run and can be set from the polled menu screen or scan menu screen. The "scan rate" is the rate the data for an entire scan will be completed. The scan rate becomes different from the internal rate when multiple channels are scanned as there is a settling time equal to "internal rate / (numchans * 5)." So if 7 channels (plus calibration channel 7) are scanned in a multi-channel-cal scan mode and the internal rate is 1000Hz the scan rate would be approximately 25Hz.



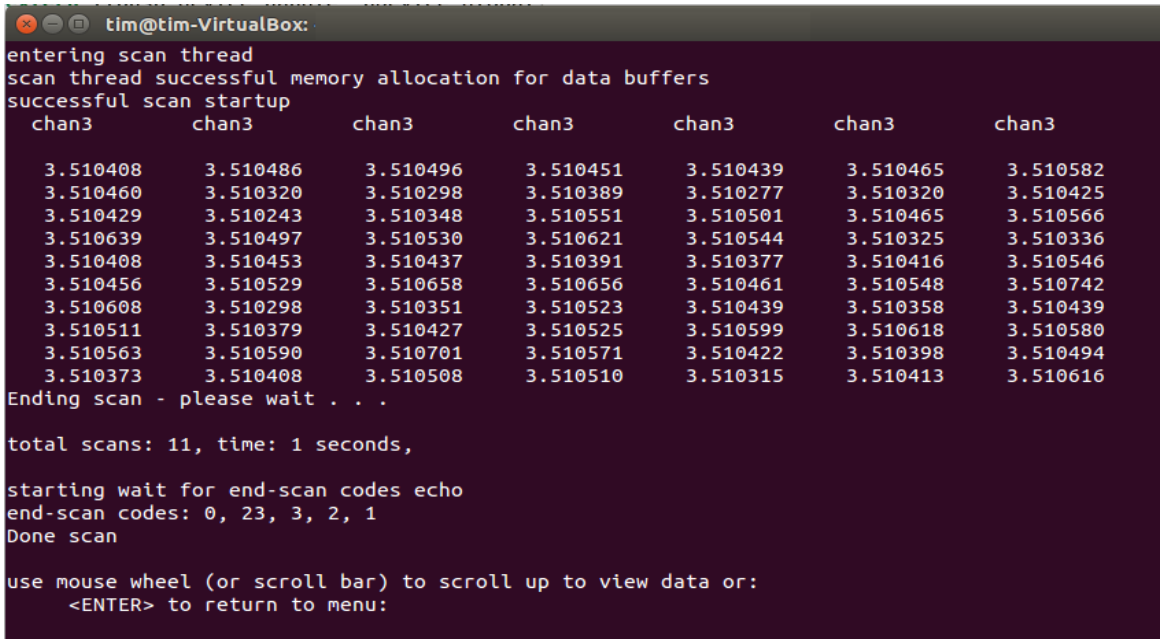
An example of setting up for scanning is to use “T” to set scan type:



Selecting 6, for example will return you to the main scan screen. You can then select the scan channels. There is a maximum of 7 channels in this multi-chan-calibration-digital-input scan since the 8th channel (the offset channel) is automatically included. The offset channel is then subtracted from all the other channels (in helperFuncs.c) before the scan data is displayed and/or written to the log file. A screenshot of the scan results is shown below. If the console screen size wasn't resized at startup, this screen may look kinda messy since the lines would automatically wrap.



If single-channel scan had been selected, the screen would show 7 columns with the voltage for the channel that was chosen as shown in the picture below:



At the end of a scan, the firmware will return the end-scan codes shown in the screenshot above. If for some reason, either the application's or firmware's buffers wrap, the scan will exit and display an indication that was the cause of an early exit. The way the code is written, using threads, it's unlikely that either of those buffers should wrap. As the internal data rate is increased, more data is handled at a lower level (driver) which is much faster, virtually eliminating any buffer wrap concerns. When that's being done, the display of data will be “displayed” in larger “blocks” relative to the size of the package requested of the lower level driver. The code shows the handling of all that and can of course be “tweaked” if there isn't any problem related to faster data rates. Note that the actual “scan rate” can influence the buffer wraps, making them less likely as the number of channels is increased since more processing of the scan data is done inside the firmware automatically. As mentioned on the “end of scan” screen the window can be scrolled up to display more data, up to the console buffer size capability.

If a log file is created, it's name is based on the current system time at the time it is created, making each one unique.

Additional Notes

This app was coded and built within the Oracle VM VirtualBox, a “virtual OS” running in Windows7. Some of the debug-to-screen content has been left active in the code, such as return values during initialization and sign-on instead of making a separate debug-to-file for such basic debug information. The screenshots above show some of those values such as the values returned after a sign-on, as well as an end-scan. Those values should always be the same as shown, otherwise a problem is indicated.