LabVIEW HW #5 (a longer assignment!)

Each problem begins with a suggested descriptive name (including the *.vi* extension) for the solution VI that you will write. Suggested icons for use in the VI are given at the end of some problem statements. The palette locations of the cited icons are not give explicitly: these icons can be found with the aid of **Quick Drop**.

1. The 4th Edition of the Essick text contains a special Do-It-Yourself/"USE IT!" section ending the portion of the text that describes communication with Data Acquisition Devices using the Measurement & Automation Explorer (MAX). Copies will be distributed to those using earlier editions, so you should also complete these tasks.

2. **Derivation of Aliasing Condition** Given that we can write the aliasing condition as $f = \pm f_{\text{alias}} + n f_s$, use this to show that $\sin(2\pi f t_i) = \pm \sin(2\pi f_{\text{alias}} t_i)$, at all $t_i = i\,\Delta t$.

3. **Observation of Voltage Resolution** Find the size of the smallest observable voltage steps, ΔV, when you make a measurement using an analog input on an *n-bit* DAQ device, and verify that your observation is consistent with the prediction of Equation (1) from your *LabVIEW* workbook. Attach a small DC voltage (*e.g.*, the output of a thermopile) to the AI channel *ai0* of your DAQ, and then use MAX (under the **Analog Input** tab, with **Mode>>On Demand**) to acquire 100 (or so) digitized samples of this input voltage. After stopping the acquisition, carefully inspect the data samples of this "constant" input signal, which in reality will vary slightly due to electronic **noise** in the experimental setup [and, possibly, with the slow time variation of the source (*e.g.*, drift or "$1/f$ noise")]. You should find that the samples you have captured are distributed in *discrete* voltage levels. Determine the smallest spacing, ΔV, between two of these adjacent levels. Given the range and resolution of your DAQ device, does your value for ΔV agree with the prediction of Equation (1)? If your DAQ device can operate with a different input range (or "span"), repeat this procedure to verify that ΔV decreases (or increases) as expected.

4. **Pulse Train Generation Using MAX**
Some of the less expensive DAQ cards do not allow "hardware-based" timing, and you won't want to use one of those for this problem. Examples where you are stuck with (far less accurate and precise) "software-based" timing are the USB-6001 and myDAQ. While the USB-6008 does have a 32-bit counter, it lacks a gate, and so that device can only count edges, meaning that it cannot directly perform period or frequency measurement *in hardware* (*i.e.*, with high accuracy and precision).
…On the other hand, other USB devices, such as the USB-6211 and the ELVIS II do have internal clocks, and *all* of the PCI devices, such as the PCI-6251 and the ELVIS I, allow hardware timing. For this problem (an much of your lab work), you will need to

connect to a DAQ with hardware timing, and make *sure that you make use of it* (rather than using imprecise software timing).
…For this exercise, you will generate a "digital pulse train," that is, a waveform that alternates, *at a precise frequency*, between the HIGH and LOW voltages that we associate with digital "states." Note (Well!): there is no completely universal standard as to what those voltages are; while many ultra-fast devices use the "NIM" standard voltage levels, the old standard has been "Two-Transistor Logic," or TTL. – *Wikipedia* states:

> Standard TTL circuits operate with a 5-volt power supply. A TTL input signal is defined as "low" when between 0 V and 0.8 V with respect to the ground terminal, and "high" when between 2 V and 5 V, and if a voltage signal ranging between 0.8 V and 2.0 V is sent into the input of a TTL gate, it is considered "uncertain."

Armed with an appropriate DAQ (capable of hardware timing), proceed as follows: Under the **Counter I/O** tab, for an available counter (*e.g.*, **Channel Name>>Dev1/ctr0**) select **Mode>>Pulse Train Generation**. For **Pulse Terminal**, select the output pin for the counter being used (*e.g.*, the output pin for *counter0* on the PCIe-6251/ELVIS I is called CTR 0 OUT and is the PFI 12 pin). Connect the pin selected by **Pulse Terminal**, along with the D GND pin, to the input of an oscilloscope (or an AI channel of your DAQ device). Press **Start** and verify that a TTL-compatible digital waveform is being created with the frequency that you have input on the front-panel Frequency control.

5. **Creating a Simulated Device** There will be occasions when you want to work on your programming, when a DAQ device is not available to you. For those situations, you can create a software replica (called a *simulated device*), using MAX.
    a. Create a USB-6002 simulated device as follows: in MAX, right-click on **Devices and Interfaces** under the **My System** heading and select **Create New…**. In the dialog window that appears, select **Simulated NI-DAQmx Device or Modular Instrument**, and press **Finish**. Then, in the **Create Simulated NI-DAQmx Device** window, choose **USB DAQ>>USB-6002**, and press **OK**.
    b. Confirm that the USB-6002 simulated device now appears in the list under **Devices and Interfaces**, in the same manner that a real DAQ device would, except that simulated devices are denoted by yellow icons.
    c. Create a **Test Panels** window for the simulated device, then choose the **Analog Input** tab and select **Mode>>Finite**. When you press **Start**, the chart will display a simulated acquisition at the specified AI channel, of one cycle of a noisy sine wave. If **All Input** is selected, then the digital lines will count in binary, under the **Digital I/O** tab.