Lab experience 3: LabVIEW as a Function Generator

Objective

- Learn programming skills to create a LabVIEW-based function generator
- Evaluate the limits on signal output data collection through a USB-6008 device.

Catching up

Seniors have used analog output from LabVIEW as part of their feedback-controlled heaters in BIOEN 337, and the juniors will do the same project this quarter.

The juniors have conducted a lab exercise in BIOEN 327 that passed a signal through the 'line out' port in the computer sound card, to the 'line in' port in the same or another computer. The sound card provides an inexpensive, high frequency signal input and output function, but is worse than the USB-6008 at low frequencies (the sound card contains a high-pass filter) and has a smaller voltage range (the sound card is limited to ±1 V). Anyone interested in exploring the sound card as an I/O port is welcome to do part 2 of the BIOEN 327 sound card exercise.

Reading & new concepts

Each USB-6008 and USB-6009 device has two 12-bit digital-to-analog converters (DACs) that change their output value(s) when the appropriate commands are sent from the controlling computer. You may set up the device to receive a value for either or both channels, but if you set it up to receive both channels 0 and 1, you need to send an array of two values. If it is set up for both channels and you want to hold one constant, then you need to send the same value as you sent last time.

The USB-6008 and USB-6009 have only one output voltage range, 0 to 5 V. Unlike the analog input, there is no 'differential' analog output; both have the same GND reference that is defined as 0 V. Output values are updated on demand, such that there is no 'N samples' option. The device specifications state that the maximum output rate is 150 samples/second, but you can send commands to the device faster if you like.

Read section 5.10, DC Voltage Source, in the textbook. You should understand the steps but the programming exercise itself is optional. Note the use of the STOP input to the DAQ Asst on page 212. This feature is important because the DAQ changes its output *only when you tell it to* (or unplug it) so if it is going along at 3.14 V it will *stay* at 3.14 V when you stop the VI if the VI does not send a "0 V" signal first.

Do exercise parts 1(a) and 3(a) before coming to lab, if possible. Show the instructor your VI and calculations, then proceed to circuit building.

Exercises

Demonstrate each VI for the instructor, and record data or answer the questions in the few places where space has been left for them.

1. Create and test a LabVIEW-based sine generator

a) Create the VI whose block diagram is shown on page 218, with the modifications listed below. If you need instruction on the MathScript node, ask the instructor or refer to sections 4.1 & 4.2 in the textbook.

- Replace the millisecond 'wait' with 'Wait until next ms multiple'
- Replace the 10 ms constant with a front-panel control having a default of 50 ms.
- Add two controls and inputs to the MathScript node so you can modify the amplitude and offset of the sine wave. Edit the MathScript itself to include the amplitude and offset variables.
- Add an 'In range and coerce' function to ensure the data value is between 0 and 5 V. This function is in Programming >> Comparison.
- The error handler and OR function are optional; you may wire the STOP button directly to the loop terminate icon *and to the DAQ Asst stop*.

b) Set the amplitude and offset to 0 and measure the DAQ output voltage with a digital multimeter. Repeat with the offset at 2.4 V and 4.8 V. Compare the observed error to the 'Accuracy' value for analog output shown in the Specifications section of the USB-6008 web page. Note that to compare the two, the values should both be in volts or percent.

Observed:

Published:

c) Connect the AO 0 (analog output zero) terminal to an oscilloscope and verify that the output is the desired sine wave for various values of samples per cycle and loop period, amplitude and offset.

d) Decrease the samples per cycle and loop period until the transition from one output voltage to the next is clearly visible on the oscilloscope screen.

- Estimate the rate at which the voltage changes from one level to the next (V/ms).

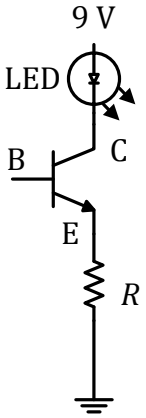
- What, if anything, do you observe as you push the loop rate higher than 150 Hz?

2. Use LabView to read a saved bio-signal from disk and regenerate the signal

a) Open 426_SignalRead_OutputOnce.vi in the G:\498F-599G\Lab03 folder, and use it to read, display, and output the signal that has been saved in that folder. Observe the signal on the oscilloscope. The signal sampling rate has been included in the file name so you can set the loop period appropriately.

b) Save the VI under a new file name and modify it so that the signal repeats continuously. This can be done by nesting the For loop in a While loop. In addition, you will want to put a second DAQ Asst outside the While loop; its data input should be 0, and the Error Out from the 1st DAQ Asst should be wired into the Error In of the 2nd DAQ Asst.

3. Control the light level from an LED using LabVIEW and a VCCS



This exercise uses a voltage-controlled current source (VCCS) to regulate the light produced by a light-emitting diode as a function of DAQ output voltage. It is necessary because the USB-600x does not have the current capacity to drive an LED. Do not connect an LED directly to the analog output terminals!

a) The simplest VCCS uses a 2N3904 or similar BJT and a resistor. In the figure at right, note that the DAC is connected to the base B, and $V_E \approx V_B - 0.6$ V. Let's start with some simple engineering calculations to select our components.

- Assume that we would like to operate the LED with currents $0 < I < 50$ mA. Choose a resistance for your VCCS that will ensure the LED current limit is not exceeded when the DAC voltage is at its maximum. Show your calculations.

- Determine the power that your resistor will dissipate when the DAC voltage is at its maximum.

- Find the data sheet for your transistor, for example G:\Datasheets\2N3904.pdf. Look for I_C in the Absolute Maximum Ratings table, and confirm that the transistor's current capacity is sufficient, *i.e.* greater than 50 mA.

- Find the transistor's DC Current Gain, h_{FE}, in the Electrical Characteristics table. This gain is the ratio of current through the emitter to current through the base. Verify that when the emitter current is 50 mA, the current from the DAC to the base will not exceed 5 mA, which is the maximum DAC drive current.

b) Create a VCCS using the components that you selected. Do not connect the LED directly from the analog output to ground!

b. Connect the DAC to the base of the transistor and run your sine-generator VI to verify proper system operation, *i.e.* that the LED brightness varies sinusoidally.

4. Now let's make a second circuit using an LED of a different color!

The circuit we are creating, with two alternating LEDs, is one essential component of the optical sensor in a pulse oximeter.

a) Build a second LED-BJT-R circuit in parallel with the first. Test it with the output from your existing VI.

b) Modify the VI to add a second output channel.

- Open the DAQ Asst and click the blue plus sign to add output channel AO1.
- Note that if you run it now, the VI will give an error because a single scalar is passed to the DAQ Asst while it expects one number for each channel.
- Revise the MathScript node to make its output an array in which the two values vary inversely, e.g. $y_1 = 2 + \sin(arg)$, $y_2 = 2 - \sin(arg)$. Since this is MATLAB code, you know that making *arg* into an array will cause *data* to be a 1-D array and the correct values will be passed to the DAQ output. Interestingly, the Waveform Chart will plot the values from the 1-D array as a single curve. To get two curves we need a 2-D array, which can be generated with the Build Array function.

c) Run your new VI and circuit to watch the LED brightness wax and wane.

d) Modify your VI such that the output function is a square wave rather than a sine wave, and that the +/− changes are opposite in sign for the two channels. There are many ways to do this, including but not limited to the following:

- Modify the MathScript code so the output value is `2+2*sign(sin(arg))`;
- Use the `square(x)` function in Mathematics >> Elementary >> Gating, and scale the loop index as desired to change the square wave period;
- Create the Waveform Simulator introduced in section 4.3 of the text.

e) Using the oscilloscope, observe the transition timing as the AO0 and AO1 channels switch from high to low and vice versa.

- In our VI, it appears that the write operation to the DAQ should produce the transitions at AO0 and AO1 simultaneously. Do you see any delay from one channel to the other?