NAME _____

EET 2259 Lab 12

Studying AC Circuits with LabVIEW

OBJECTIVES

-Write LabVIEW programs to generate AC waveforms of user-specified shape, amplitude, and frequency.

-Write LabVIEW programs to display and measure AC waveforms produced by external sources.

-Use LabVIEW to explore characteristics of AC circuits studied in previous electronics courses.

-Use LabVIEW to measure AC current.

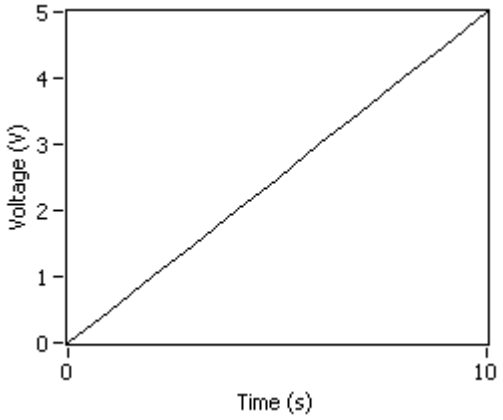
Part 1. Generating AC Waveforms

From previous labs, you know how to use LabVIEW's DAQ Assistant to generate constant DC voltages. Let's see now how to generate voltages that change over time. We'll learn two general approaches to this problem.

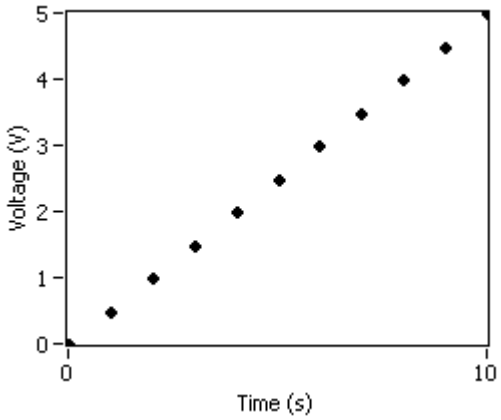
- The first approach uses the familiar functions on LabVIEW's **Functions > Programming > Numeric** palette and **Functions > Mathematics** palette to produce the values, point by point, of the waveform you're trying to generate.

- The second approach, which in general requires less work on your part, uses specialized functions and VIs on LabVIEW's **Functions > Programming > Waveform > Analog Waveform > Generation** palette. As the palette's name suggests, these functions and VIs are custom-designed to do exactly what we're trying to do here—namely, generate waveforms.

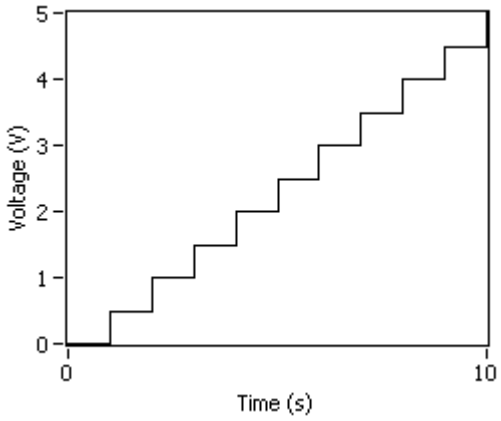
Let's start with the first approach. We'll use it to do something very simple: generate a voltage that starts at 0 V and increases steadily to 5 V over a period of 10 seconds, as shown below.



In generating any voltage waveform, it's important to think about the number of points you generate and how frequently you generate them. In the example above, suppose we generate a new voltage value once every second. Then the points that LabVIEW generates will look like this:

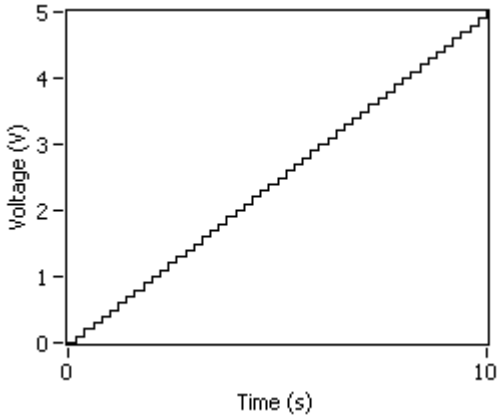


In your mind, you can connect these points to get a nice smooth line. But if you use the DAQ Assistant to send these points out to the real world, what you'll actually get is a waveform that is constant at 0 V for one second, then constant at 0.5 V for a second, and so on. Displayed on an oscilloscope, it will look like this:



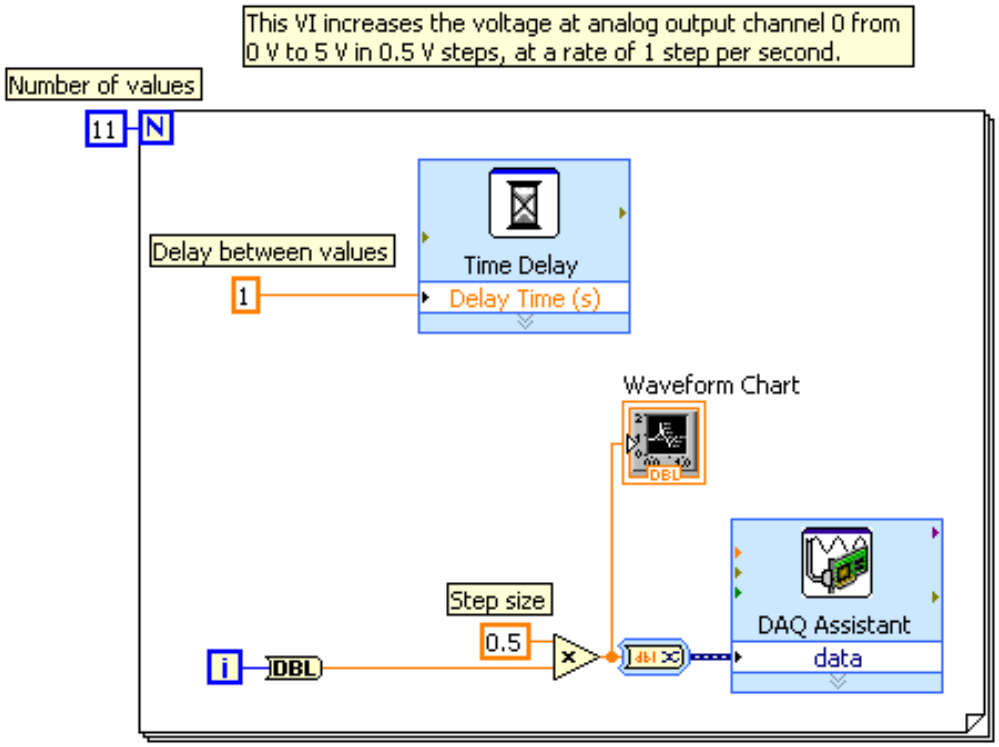
How can we modify this "staircase" pattern to look more like the nice smooth line shown in the first picture on the previous page? Answer: By generating more values that are spaced closer together in time. This will result in smaller steps in the staircase, and the waveform will be a closer approximation to a straight line.

For example, if we generate fives voltage values per second instead of just one per second, we'll get an output that looks like this:



Here you can still see that the output is a staircase. If we increase the number of voltage values even further, to ten values per second or 100 values per second, eventually the steps will become so small that your eye can no longer see the steps, and the result will look like a smooth line. Let's do it with LabVIEW!

1. Create a new VI whose front panel contains a waveform chart. Make its block diagram to look like this, with the DAQ Assistant configured to produce an analog voltage on channel ao0:



2. Run the program, and you should see that the waveform chart displays a line that rises from 0 V to 5 V over a period of 10 seconds.

3. Using a multimeter to measure the voltage on the myDAQ's analog output channel 0, run the program again. You'll find that although the waveform chart plots the values as a smooth line, the voltage actually increases in 0.5 V steps once every second.

4. To display this on an oscilloscope, we'll need to speed things up. Change the delay time from 1 second to 0.01 second. Also enclose the entire block diagram in a While Loop with a STOP button, so that the For Loop executes repeatedly. Run the program again, this time using an oscilloscope to display the output voltage. Adjust the oscilloscope's settings to display about two cycles of the waveform. You should see a very rough sawtooth wave. We could make it smoother by adjusting the step size, but we'll soon see a better way to generate waveforms. Save this VI as **Lab12GenerateSawtooth.vi** and **show me your working program**.

Next, let's generate a square wave, which is mathematically even simpler than a sawtooth wave.

1. Save a copy of your previous program under the new name **Lab12GenerateSquare.vi**. Modify it so that it generates a square wave with an amplitude of 5 V. Display your program's output on the oscilloscope. (*Hints*: Instead of needing a FOR Loop inside a WHILE Loop, this time you can just use a WHILE Loop. Use the old even-odd trick from previous labs to produce 0 V when the loop's iteration terminal is even, and 5 V when the iteration terminal is odd.)

2. On the front panel, add a knob that lets the user set the square wave's amplitude. Save this VI as **Lab12GenerateSquare.vi** and **show me your working program**.

Part 2. LabVIEW's Waveform-Generating Functions & VIs

Using LabVIEW's regular math functions, we could continue writing programs to generate other waveforms. For example, we could use the **Sine** function on the **Functions > Mathematics > Elementary > Trigonometric** palette to generate sine waves. But an easier approach is to use the specialized functions on the **Functions > Programming > Waveform > Analog Waveform > Generation** palette.

Before we use these, we must get familiar with a LabVIEW data type called the waveform data type. In this course you've used several *simple data types*, such as numeric, string, and Boolean. You've also used some *composite data types*: arrays and clusters. (These are called *composite data types* because they are groups made up of the simpler data types: for example, a cluster might contain one string, one Boolean, and two numerics.)

A waveform is another composite data type that always contains the following three pieces:

- The **data**, which is an array of floating-point numbers. These are the values of the waveform as time passes.
- The **start time**, which is the precise time when the first value was generated or measured.
- The **Δ*t*** (pronounced "delta t"), which is the time interval between the data values.

In the VIs that you wrote in the first part of this lab, you simply sent floating-point numbers, one at a time, to the waveform chart and the DAQ Assistant. (See the block diagram on the previous page.) The time delay in your loop controlled the spacing between these numbers.

In the following program, you will instead send a **waveform** to the waveform chart and the DAQ Assistant. As explained above, this waveform will contain an array of data values, as well as timing information (the start time and the Δ*t*).

1. Create a new VI whose front panel contains a waveform chart. On the VI's block diagram, place a **Sawtooth Waveform.vi** from the **Functions > Programming > Waveform > Analog Waveform > Generation** palette.

2. The Sawtooth Waveform.vi has seven input terminals and two output terminals. Connect the output named **signal out** to the Waveform Chart. Leave all of the inputs unwired; this causes it to use default values for all of the inputs.

3. Run your VI. On the waveform chart you should see a sawtooth waveform, squished together horizontally toward the left edge of the chart. Notice the time-stamps on the

chart's horizontal axis. The starting time should be 7:00:00.000 PM on December 31, 1903. (This is LabVIEW's default starting time that you get if you don't specify otherwise.) And the ending time should be five seconds later.

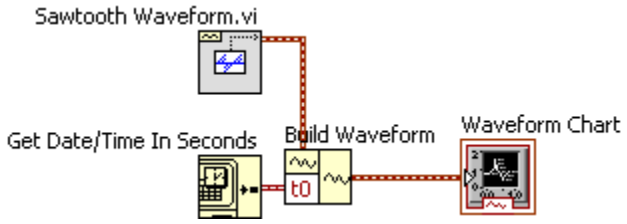
4. To stretch out the waveform horizontally, change the ending time from 7:00:05.000 PM to 7:00:01.000 PM by clicking on it and entering the new time. You should now be able to clearly see ten cycles of a sawtooth. Since there are ten cycles in one second, its frequency is 10 Hz. Also notice that its negative peak is -1 V and its positive peak is 1 V. (These are the default frequency and peak values that you get if you don't specify otherwise.)

5. Obviously, the default times are not correct. Let's fix that by telling LabVIEW to read the computer's clock when you run the VI. On the block diagram, place a **Get Date/Time In Seconds** function from the **Functions > Programming > Timing** palette.

6. Also place a **Build Waveform** function from the **Functions > Programming > Waveform** palette. Notice that its second input terminal is labeled **Y**. Click on this input terminal, and you'll be given the choice to change it from **Y** to **t0** or **dt**. These three choices correspond to the three components of a waveform discussed earlier:
   - Y is the array of data values.
   - t0 is the start time.
   - dt is the time interval between values, which we called Δt (or "delta t") above.

   We want to change the start time, so select t0.

7. Wire the new items so that your block diagram looks like this:



   By wiring the diagram in this way, you're telling LabVIEW to modify the waveform's start time so that it's equal to the current time from the computer's clock. (But you're not modifying the data values themselves or the time interval between the values.)

8. Run the VI again, and your sawtooth wave should look the same as before, but now the horizontal axis should show the correct times.

9. On the front panel, add a knob labeled **Amplitude** and a numeric control labeled **Frequency**. Wire these to the correct input terminals on the Sawtooth Waveform.vi so that the user can set the wave's amplitude and frequency. Also place a WHILE Loop with a STOP button around the entire block diagram.

10. Run your VI again with an amplitude of about 3 and a frequency of 20. Note that the resulting waveform has these new amplitude and frequency values.

11. Next, let's use a DAQ Assistant to send this waveform out to the real world where we can display it on an oscilloscope. Place a DAQ Assistant on the block diagram and configure it to perform analog output on Channel ao0.

12. This DAQ Assistant's configuration needs to be different from what you're used to. Until now, in the **Generation Mode** dropdown box, you've always selected **1 Sample (on Demand)**. That is the correct setting when you're sending one value at a time to the DAQ Assistant. But as we'll see below, the Sawtooth Waveform VI generates 1000 values at a time, and we want the DAQ Assistant to output those values over and over until we stop the program. So do this:
    - Set the Generation Mode to **Continuous Samples** instead of **1 Sample (on Demand)**.

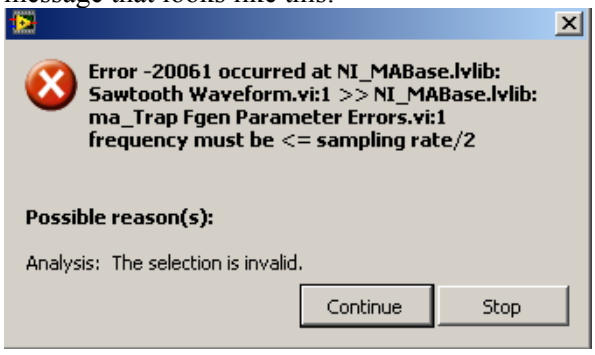
13. Whenever we use Continuous Samples, we also want to enable a feature called regeneration. Here's what this means: if your program is not running quickly enough to keep providing the hardware with new output values, the hardware can instead re-use some of the old output values, which are stored temporarily in a buffer. To enable this feature, click on the DAQ Assistant's **Advanced Timing** tab, and then select **Allow regeneration** in the drop-down box.

14. Click **OK** to close the DAQ Assistant's configuration window.

15. Place a **To DDT** on the block diagram. Again, we need to configure this differently from what you're used to. Until now, you've usually selected **Single scalar** for the **Input data type**. This time, though, we're sending a waveform to the DAQ Assistant. So select **Single waveform** instead of **Single scalar**. Then click the **OK** button.

16. Finally, run a wire from the **Build Waveform**'s output to the **To DDT**'s input. And run another wire from the **To DDT**'s output to the **DAQ Assistant**'s input. Run your VI, and the oscilloscope should display a sawtooth waveform. You should find that you can adjust the amplitude and frequency as the program runs.

17. While the program is running, try changing its frequency to 700. You should get an error message that looks like this:

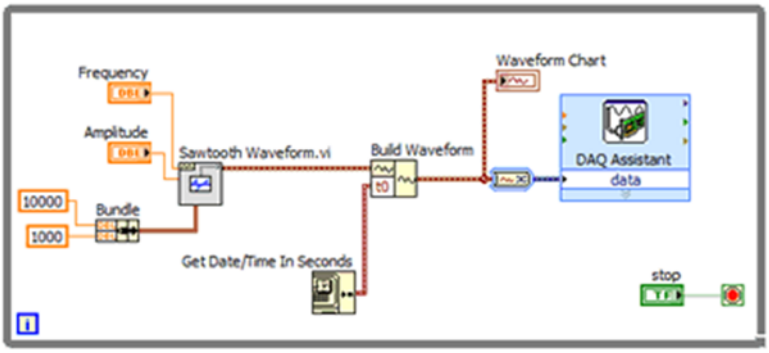


The heart of this message is the statement that "frequency must be <= sampling rate/2." The problem here is that we're violating the Nyquist Sampling Theorem, which you may have studied in another class. This famous theorem states that to accurately reproduce a signal, the signal's frequency must be less than or equal to the sampling rate divided by 2.

18. Let's fix this problem. Click the error message's Stop button to close the message. Next, notice that Sawtooth Waveform.vi has an input terminal called **sampling info**. Since we haven't connected anything to that terminal, LabVIEW is using its default sampling rate, which is 1000 samples per second. As long as our waveform's frequency is less than 500 Hz, we're satisfying the requirement that frequency must be less than half of the sampling rate. But once the frequency goes over 500 Hz, we'll get into trouble unless we increase the sampling rate above its default value of 1000.

19. In the Context Help window, look at the help for the Sawtooth Waveform.vi and then click the link labeled **Detailed help**. In the resulting help text you will see that the **sampling info** input terminal accepts a cluster of two values. The first of these values is the sampling rate, which we've just been discussing. The second value is the number of samples to generate, which also has a default value of 1000.

20. To fix our sampling problem, let's modify the block diagram so that the sampling rate is 10,000 samples per second instead of the default 1000. To do this, place a **Bundle** function and rewire the block diagram to look like this:

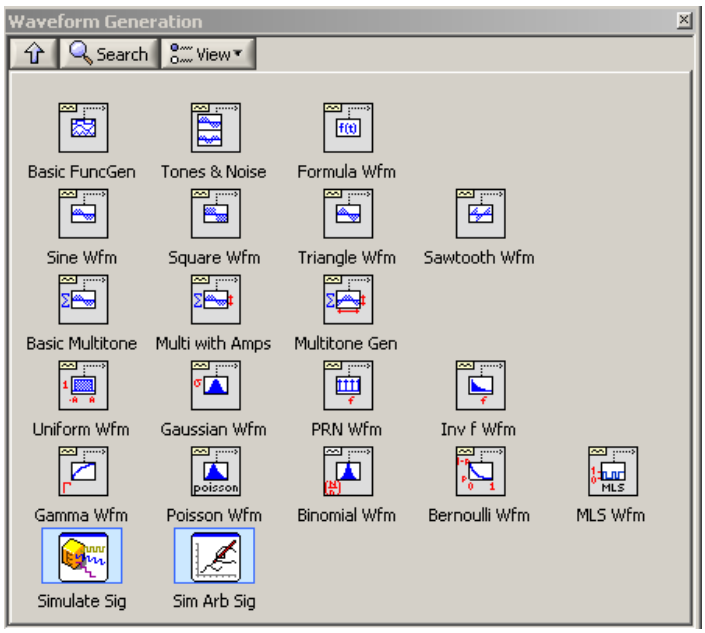


Note that we're not changing the default value of 1000 for the number of samples, but we are increasing the sampling rate above its default value of 1000.

21. Run your VI again with the front-panel frequency set to 700, and this time you shouldn't get an error message. Save this VI as **Lab12GenerateSawtooth_2.vi** and **show me your working program**.

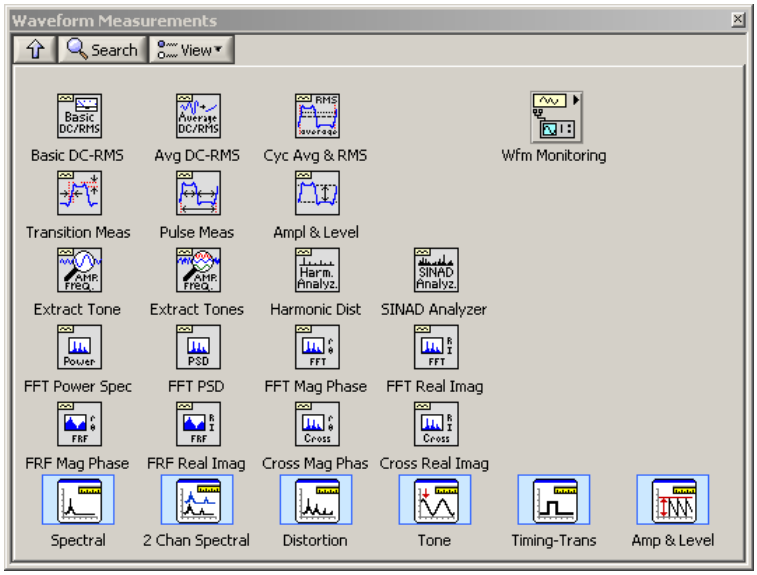
In the previous program you used Sawtooth Waveform.vi to generate a sawtooth waveform. Most of the other VIs on the **Functions > Programming > Waveform > Analog Waveform > Generation** palette (pictured on the next page) are similar to Sawtooth Waveform.vi. In particular, notice that the second row contains VIs for sine, square, and triangle waveforms. Also, the **Basic Function Generator.vi** in the first row can produce all four of the basic waveforms. The VIs in rows four and five are for generating different kinds of noise (uniform white noise, Gaussian noise, and so on).

Finally, the **Simulate Signal Express VI** in the last row provides "one-stop shopping" that combines many of the other VIs on this palette. With **Simulate Signal** you can choose among sine, square, triangle, or sawtooth waveforms, with the option of adding nine different kinds of noise to the waveform. We'll use this Express VI later in this lab.



Part 3. Measuring AC Waveforms

So far in this lab we've been generating AC waveforms and configuring the DAQ Assistant to perform analog output. Now let's look at using LabVIEW to measure AC waveforms that are produced externally. In this case we'll be performing analog input. The VIs we'll use are contained on the **Functions > Programming > Waveform > Analog Waveform > Measurements** palette (pictured below).



Note that this palette, like the one we just looked at, contains VIs (in the first 5 rows) and Express VIs (in the last row). In general, the Express VIs combine the functionality of two or more of the VIs, and they're usually simpler to use. So we'll use Express VIs for the rest of the lab.

With the items on this palette you can do some pretty advanced analysis, but let's stick to the basic measurements that you're familiar with from previous courses: frequency, peak voltage, peak-to-peak voltage, and rms voltage.

1. Create a new VI whose front panel contains a waveform graph. Place a DAQ Assistant on the block diagram and configure it to perform analog input on Channel ai0. Also, in the DAQ Assistant's Timing Settings, set the **Acquisition Mode** to **N Samples**.

2. Notice that **Samples to Read** is set to 100, and **Rate (Hz)** is set to 1k. Nyquist's Sampling Theorem applies here, just as it did above when we were generating waveforms. Whenever you measure a waveform, you must ensure that your sampling rate is at least twice as great as the highest frequency you expect to measure. And even though the theorem says "at least twice as great," it's generally better to use an even higher sampling rate—say, ten times as great as the highest frequency you expect to measure. In the steps below, we'll be measuring waveforms with frequencies up to 1 kHz. So in the DAQ Assistant, change **Rate (Hz)** from 1k to 10k.

3. Notice that the default value for Number of Samples that we're reading is 100. Since we increased the Rate by a factor of 10, let's do the same with the Number of Samples: increase it from 100 to 1000. The idea here is that we want to read enough samples to cover several cycles (let's say, 10 or more cycles) of our waveform. Doing this will average out any fluctuations that might occur in a single cycle, giving us more confidence in our measurement results than if we had just read enough samples to cover one cycle or part of a cycle. So let's do a simple calculation.

   I said above that we'll be measuring waveforms with frequencies up to 1 kHz. What is the period (cycle time) of a 1 kHz waveform? _____

   If our sampling rate is 10 kHz, how long will it take to read 1000 samples? _____

   How many cycles of a 1 kHz waveform will occur in this amount of time? _____

4. Click the DAQ Assistant's **OK** button.

5. Move the waveform graph inside the While Loop, and connect its input terminal to the DAQ Assistant's **data** output.

6. From the **Functions > Programming > Waveform > Analog Waveform > Measurements** palette, place an **Amplitude and Level Measurements** Express VI. Configure it to display **RMS**, **Negative Peak**, **Positive Peak**, and **Peak to Peak.** Then connect each of these outputs through a **From DDT** to a numeric indicator. Also connect this Express VI's **Signals** input to the DAQ Assistant's **data** output.

7. From the same palette, place a **Tone Measurements** Express VI. Configure it to display **Frequency.** Then connect its **Frequency** output through a **From DDT** to a numeric

indicator. Also connect this Express VI's **Signals** input to the DAQ Assistant's **data** output.

8. Set the red trainer's function generator to produce a sine wave with a frequency of 1 kHz and an rms voltage of 2 V. Verify these values using a multimeter or oscilloscope. Then connect the function generator's output and ground terminals to the myDAQ's AI0+ and AI0− terminals, respectively. *Since the trainer's function generator is a floating voltage source, don't forget also to run a jumper wire between the myDAQ's AGND and AI0− terminals, as you've done in the past.*

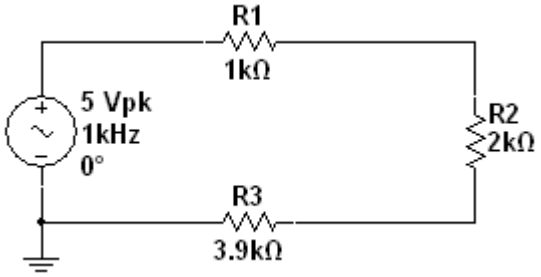
9. Run the program, and make sure that the values displayed on your program's front panel are correct. Save this VI as **Lab12MeasureAC.vi** and **show me your working program**.

_____

Part 4. An AC Voltage Divider

Knowing what you've learned in this lab, you can use LabVIEW to study many aspects of AC circuits that you've studied in earlier courses. As a simple example, let's use LabVIEW to verify the voltage-divider rule in a series AC circuit.

1. Using your knowledge of AC circuits, predict the peak voltage drops across R1 and R2 in the circuit shown below. Record your predictions in the spaces provided.



Predicted Peak Voltages: *V1*_____ *V2*_____

Hardware Setup:

2. Build the circuit on the breadboard. For the AC voltage source, **do not** use a function generator. Rather, use the myDAQ's analog output channel 0. In other words, use the myDAQ's AO0 terminal for the + terminal of the voltage source in the diagram above, and use the myDAQ's AGND terminal for the − terminal.

3. Also connect a pair of wires to measure the voltage drops across R1 and R2. In particular:
   - Use two wires to connect the ends of R1 to the myDAQ's AI0+ and AI0− analog inputs.
   - Use two more wires to connect the ends of R2 to the myDAQ's AI1+ and AI1− analog inputs.
   - *Since the myDAQ's analog output is a grounded voltage source, don't connect jumper wires between the myDAQ's AGND terminal and any of its other terminals.*

LabVIEW Program:

4. Create a new VI whose front panel contains a STOP button and two numeric indicators labeled **V1** and **V2**.

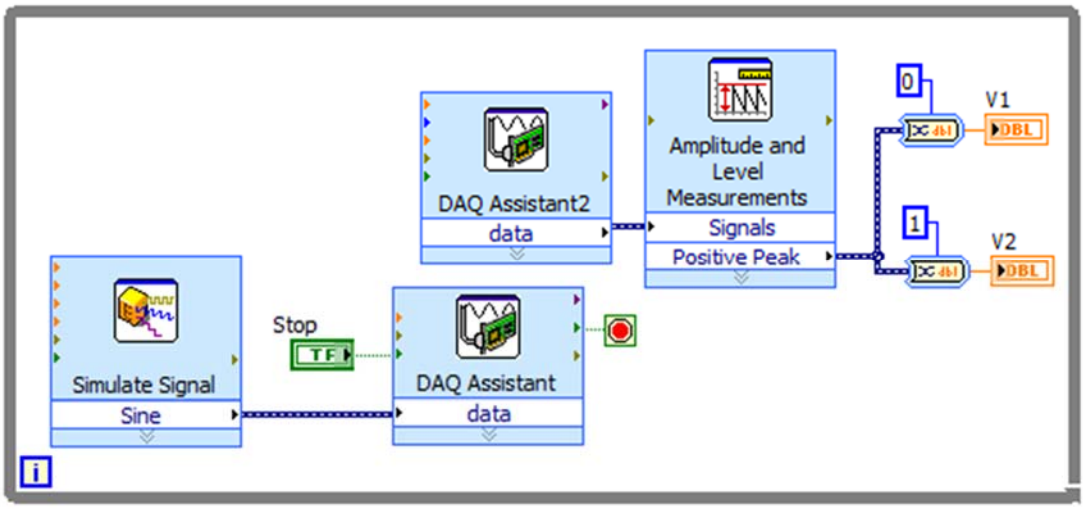
5. On the block diagram, place a **Simulate Signal** Express VI. Configure it to produce a 5 V pk, 1 kHz sine wave. To see how the number of samples per second (in the dialog box's **Timing** section) will affect the quality of your output waveform, notice how the simulated wave in the Preview window changes as you increase the number of samples per second: start with 2000 samples per second, then change it to 4000, then 8000, then 16000, and then 32000. With this value set to 32000, press **OK**.

6. Place a DAQ Assistant on the block diagram. Configure it to perform *continuous* analog output on channel ao0. (And remember from earlier that whenever we use continuous mode, we also want to allow regeneration.)

7. Place another DAQ Assistant on the block diagram. Configure it to perform differential-mode analog input voltage on two channels: ai0 and ai1. Set this DAQ Assistant's Acquisition Mode to **N Samples**, reading **1000** samples at a rate of **10 kHz**.

8. Place an **Amplitude and Level** Express VI on the diagram. Configure it to measure Positive Peak. Connect its output terminal to two **From DDTs**. Configure the first **From DDT** to read a single floating-point scalar from **Channel 0**. Configure the second one to read a single floating-point scalar from **Channel 1**.

9. Build the block diagram to look like this:



10. Run the program, and make sure that the values displayed on your program's front panel agree with your predictions from above. Also use the oscilloscope to display the sine wave being produced on the AO0 analog output. Save this VI as **Lab12ACVoltageDivider.vi** and **show me your working program**.

Part 5. Measuring AC Current

Up to now you haven't used LabVIEW to measure AC current, but you shouldn't have much trouble figuring out how to do so, using what you learned in Lab 11 about measuring DC current and what you've learned above about measuring AC voltages. Let's measure the current flowing through a 4.7-kΩ resistor when it has an AC voltage applied across it.

Hardware Setup:

1. Place a 4.7-kΩ resistor on a breadboard. In series with it place a 10-Ω sensing resistor. Connect the myDAQ's analog output AO0 across the series combination of these two resistors.

2. Run wires from the two ends of the sensing resistor to the myDAQ's AI0+ and AI0− analog inputs. *Since the myDAQ's analog output is a grounded voltage source, don't connect jumper wires between the myDAQ's AGND terminal and any of its other terminals.*

LabVIEW Program:

3. Create a new VI whose front panel contains a STOP button and a numeric indicator labeled **RMS Current**, configured to use engineering notation and 3 significant digits.

4. Suppose a 1-V pk, 1-kHz sinusoidal voltage is applied across a 4.7-kΩ resistor. How much **rms current** will pass through the resistor? (Hopefully you remember how to convert between peak values and rms values.)

   _____

5. On your VI's block diagram, place code that uses a **Simulate Signal** Express VI and a DAQ Assistant to generate a 1 V pk, 1 kHz sinusoidal voltage on the myDAQ's analog output AO0.

6. Place another DAQ Assistant that will measure the current flowing through the sensing resistor, and then place additional code to display the rms value of this current. To do this you'll need to combine elements from **lab11MeasureDCCurrent.vi** and **Lab12MeasureAC.vi**, so you may want to review these two programs. Also, **don't forget Nyquist's Theorem**: in your current-measuring DAQ Assistant, use a sampling rate equal to at least twice (better, ten times) the frequency of the signal you're measuring.

7. Run the program, and you should find that the rms current displayed on the VI's front panel is close to the value that you predicted above. Save this VI as **Lab12MeasureACCurrent.vi** and **show me your working program**.

   _____

Part 6. Review Questions

1. What are the names of the three pieces of a LabVIEW waveform?

2. In previous labs, when you've used a DAQ Assistant to perform **analog output**, you've set the DAQ Assistant's Generation Mode to

   _____.

   But in this lab (and in future labs), when you're using a DAQ Assistant to produce an AC waveform, you should instead set the DAQ Assistant's Generation Mode to

   _____.

3. When making the change discussed in the previous question, you should also change one of the DAQ Assistant's Advanced Timing settings to allow

   _____.

4. What is the minimum sampling rate that can be used to sample a waveform whose frequency is 25 kHz?

5. What is the name of the theorem that tells us how to answer the previous question?

6. In previous labs, when you've used a DAQ Assistant to perform **analog input**, you've set the DAQ Assistant's Acquisition Mode to

   _____.

   But in this lab (and in future labs), when you're using a DAQ Assistant to measure an AC waveform, you should instead set the DAQ Assistant's Acquisition Mode to

   _____.

***** This lab had 6 named programs for me to check. If you didn't finish all of these during class, finish them after class. Then upload all 6 programs, along with any related subVIs, to the website by the due date. Also turn in your lab sheets at the beginning of class.******