LabView Tutorial
PHYS 210

LabView is a computer programming environment commonly used for data acquisition and analysis. It also can be used for general programming purposes. It is fundamentally different from standard programming languages such as C in that it uses *Graphical Programming*. It is very easy to learn from scratch and very convenient for writing simple programs.

This tutorial will illustrate 4 programs of progressively higher difficulty. The programs will interface with the sound card and will also introduce you to simple data acquisition and output generation.

Refer to the last color page for illustration of the graphical programming diagrams.
Italics below indicates specific directions.

1. Make a Beep

Start LabView Version 6.1 and click on "New VI" (VI stands for "visual interface").

The program opens two windows, the main VI and the VI Diagram. The Diagram contains all the programming elements.

Go to the Diagram window

First thing we need to do is to generate a waveform, an array of data that will represent the beep. LabView has many functions for waveform generation.

In the Window menu on the Diagram window click on "Show Functions Palette"

This will open a small window with many function groups. For more information on what each symbol means refer to the "Quick Reference" handout.

Chose the "Waveform" functions (middle column, fourth from the top)
Chose "Waveform Generation" (lower right corner)
Left-click on the icon 'Basic Function Generator" (upper left corner) and drag it to the diagram window

This is how you program most things, go through a series of menus, find the right function and drag it to the Diagram window.

Now in the Window menu click on "Show Tools Palette"

The tools allow you to manipulate the icons in the Diagram window

Click on the Wire tool (a little spool)
Move the wire tool over the icon of "Basic Function Generator"

You will see a bunch of connections, in different colors, that refer to different inputs and outputs of the function. The color refers to the data type of each input and output. For more information on data types refer to the “Wire Types” color section of the Quick Reference handout.

LabView can help you create appropriate inputs and outputs.

Move the wire tool over “Sampling Info” connection, right-click on it, go to “Create” section and select “Control”

“Control” refers to an element that appears on the main VI window and allows you to control the parameters of the program in real time. You will see an element appear on the main VI window.

The sampling info contains two important parameters for waveform generation, the sampling frequency called Fs and the number of samples (#s). The total duration of the waveform is the number of samples divided by the sampling frequency.

Go back to the Diagram window. To move the sampling info out of the way, select from the Tools palette the arrow tool (middle top). You can click and drag elements in the diagram using this tool. The “wires” that connect them indicate the flow of data.

With the wire tool, create Controls for signal type, frequency, and amplitude connections of the waveform generator.

The other input parameters can be left to their default values and don’t need to be specified.

Now go to the main VI window. Go to Windows menu and select “Show Controls Palette”

Here you can choose from a variety of control and display elements for the graphical interface to the program.

Go to the Graph section (right column, second from top) and select ‘Waveform Graph”
Drag and drop it on the VI window.

Go back to the Diagram window, you will see a new element. With the wire tool, connect the signal out of the Function generator to the Waveform Graph.

Now you are ready to test the program for the first time. Go back to the VI window and click on the circling arrows near the top left corner. This will run the program in a continuous loop.

A sine waveform should appear on the graph. You can change the frequency, the amplitude and the signal type of the waveform. Play a little with different setting.

Now we will add the sound output. To stop the program, click on the red button near the top of the window.

On the functions palette, make your way to the top level of functions by clicking on the up arrow near the top left corner. Select "Graphics and Sound" (right column, sixth from top), select Sound (bottom row) and drag "Snd Write Waveform" to the diagram.

Click on the sound format input and create a control on the front panel.

Now we need to take the output of the waveform and connect it to the sound function. However, there is a complication. The waveform is a different type of data then the input expected by the sound output. This is indicated by a different color of the wires. We need to convert the waveform data to a simple array of integer numbers first.

With the arrow tool over the brown wire going to the Waveform Graph, right-click the mouse and select Waveform Tools. Click on "Get Waveform Components".

Place the mouse over the new icon, you will see two circles appear, drag the bottom circle down until the icon is three times taller than it was initially.

Now you can see that the waveform contains 3 pieces of information, the initial time (t0), the time interval between points (dt) and the actually data of the waveform (Y).

Using the wire tool, connect the left side of the waveform conversion icon to the wire going to the Waveform Graph.

Now we need to do one more conversion. The "Y" output is a double floating number (orange) while we need an integer (blue) as the input to the sound function.

In the Functions Palette go to Numeric (middle column, top), select "Conversions", and drag "To Long Integer" (top row third from left) onto the diagram.

Now connect the "Y" output to the conversion icon and the output of the conversion to "mono 16-bit"input of the sound function.

After this diversion into some computer technicalities we are ready to run the program.

Go to the main VI window and start it in the loop mode.

Before you will get any sound, we need to set up various parameters to appropriate values.

Set the sampling frequency (Fs) to the same number as output rate of the sound format (11025 by default).
Set the frequency of the waveform to 1000 Hz, something that you can hear.
Set the amplitude of the waveform to 10000, this will produce sound of reasonable volume.
Finally, in the sound format, change the bits per sample to 16, since we used the "mono 16-bit" input to the sound function.

Your computer should start beeping now! If it doesn't, check the procedure again and then ask a TA for help.

The beeps are interrupted because the program stops after each iteration. You can change the frequency, amplitude and duration of the beep. Play a little with the program.

To see what is happening while the program is running, to back to the Diagram window and click on the "light bulb" button near the top. This will put the program into the debugging mode. Now you can see how different parts of the program are executed. You may notice that multiple parts of the program can be executed at the same time. The order of the execution does not depend on the position of the icons in the window, but on their logical dependencies.

2. Listen to the Beep

Now we like the computer to listen (using the microphone) to the beep that it creates. After thinking a little, you may realize that there is a problem, we don't have any way to synchronize the time when the computer makes a beep and the time when it turns on the microphone to listen. This is a common problem in computer control. It is difficult to make computers to something at a very specific time. So, generally when two tasks need to happen simultaneously or with a fixed delay, one needs to pay careful attention to the synchronization issue.

To solve this problem we will go to a slightly lower level of sound commands that allows better control of the timing.

In the diagram window, delete the sound output function by selecting it with the arrow tool and hitting the delete key.

In the Sound section of the Function Palette, go to the "Sound Output" (little speaker). Select SO Config, SO Write, SO Start, SO Wait and SO Clear functions and drag each one in turn to the diagram.

These functions separate all the steps needed for output of sound and will give us better control of the synchronization. If you want to learn more what each function does, right-click it and select Help.

Now connect the sound format control to the sound format input of the Config function.

The Config function outputs something called "Task ID" which is a common parameter that will tie all these functions together.

Connect the Task Out of each function to the Task In of the next function, in the order Config, Write, Start, Wait, and Clear.

The Task ID is not changed by each function, but by connecting them sequentially with wires you ensure that the functions will be executed in the proper order.

Finally, connect the waveform data to the mono 16-bit input of the Write function.

Try to execute the program. It should beep in exactly the same way as before.

Now we are ready to add the microphone input.

Go to the sound input part (microphone picture) and add functions SI Config, SI Start, SI Read and SI Clear to the diagram. Connect the functions sequentially with the Task ID wires.
You can also use the same sound info as you used for the output.

Go to the main VI window and add another waveform graph. This graph will display the sound recorded by the microphone.

Back in the diagram window, connect the "Waveform Graph 2" to the mono 16-bit output of the SI Read function.

You may worry that the wires initially have different color, but LabView is smart enough to change the type of the graph when it is possible.

Finally, we are at the point when we can synchronize the input and the output. We want to start outputting the sound at the same time as we start reading the microphone.

Rearrange the icons on the diagram so that SO Start and SI Start are right next to each other.
Now in the Functions Palette go to the Structures section. Drag a "Sequence Structure" to the Diagram window. It will make a little rectangle. Drag the corners of the rectangle so that it encompasses the SO Start and SI Start functions. If the rectangle appears on top of the other icons, delete it and start over. The SO Start and SI Start functions should be inside the rectangle and Task ID wires should pass through the sides. Make sure that no other wires except Task ID wires pass through the rectangle.

The sequence structure ensures that these two functions are executed at approximately the same time. This is what is called "software synchronization" and it is accurate only to a few milliseconds. For better accuracy it is often necessary to use "hardware" synchronization which involved physically changing a voltage on one of the control lines to trigger some process.

Now try to run the program. You should see an output display with the sound recorded by the microphone. You may also see microphone signals if you talk while the program is running.

3. Map Sound System Response

The next goal is to map out the response of the sound system as a function of frequency. We will measure the signal in the microphone for beeps of different frequencies. An ideal sound system should have a flat response independent of the frequency.

First we need some way to quantify the amplitude of the signal recorded by the microphone. There are a several ways of doing it, which we will explore later. Since the signal is oscillatory, a simple time average of the signal will not work. We will first use the root mean square (RMS) of the signal, which is a simple measure of the amplitude of the oscillations.

In the Functions Palette, go to the Mathematics section (right column, fifth from top) and select "Probability and Statistics" (left column, second from top). Drag RMS function to the diagram.

If you look carefully at the microphone display while the program is running you will notice that the beep does not last as long as the microphone recording. In addition, there is usually a distortion of the very beginning of the waveform. To get accurate measurements of the signal amplitude, we need to select the middle region of the beep, where the signal is clean.

In the Functions Palette go to the Array section (middle column, second from top) and drag Array Subset (third column, second from top) to the diagram. This will allow you to select a section of the array to analyze.

Connect the output of the SI Read function to the array input of the Array Subset function.
Create Controls on the main VI panel for the length and index inputs of the Array function.
Connect the output of the Array Subset function to the input of the RMS function.

The index specifies the beginning of the subset array of a given length. If you want, you can add another graph to display the section of the array that has been selected.

The RMS function outputs one number that is equal to the r.m.s. of the array. What we need to do is to step through a series of frequencies and make a plot of the r.m.s. as a function of the frequency.

In the Functions Palette go to the Structures section (top left corner) and drag "For Loop" to the diagram. Drag the For Loop rectangle to encompass all elements on the diagram. If it appears on top of the other icons, delete it and drag a new one from the Functions Palette.

Everything inside the "For Loop" will be executed repeatedly, the number of iterations is specified by "N" in the upper left corner.

We need to specify the start frequency, the step size and the number of steps for our mapping of the frequency response.

Go the main VI and in the Controls Palette select Numeric (upper left corner). Drag three "Digital control" icons to the main panel. In the Tools Palette, click on the text tool, denoted by symbol A. Change the names of the three controls to "Start Frequency", "Step Size" and "Number of Steps" (you can use any other names as well). The number of steps has to be an integer, right click on the numeric control, go to "Representation" and select "I32".

Go back to the Diagram window. You should see 3 new elements with the names you gave in the main VI. Drag all 3 icons outside of the For Loop rectangle. Connect the "Number of Steps" control to the "N" control of the For Loop.

Now we need to calculate the required frequency at each step. We want $f(i) = f_{start} + i \times (\text{step size})$.

In the Functions Palette, go to numeric and drag "+" and "×" icons to the diagram. Connect the icons with wires to create the equation given above. The iteration number "i" is available from the little blue icon in the lower left corner of the For Loop.

Finally, connect the output of the frequency equation to the frequency input of the Waveform generation function.

You will notice that the wires suddenly become dashed. This indicates that there is an internal inconsistency in the program. We now have two ways to determine the frequency, from the control on the front panel and the iteration step.

To fix this problem, right click on the frequency control icon and select "Change to Indicator". Now it will simply show the current frequency, but won't let you change it from the VI panel.

Now lets create a plot on the main VI panel that will show the frequency response.

From the Graph section of the Controls Palette select "Waveform Graph" and drag it to the main VI window. In the Diagram window, place the icon outside of the For Loop. Connect the output of the RMS function to the new waveform graph.

You are almost ready to run the program, just need to set appropriate parameters.

Set the Start Frequency to 100 (in units of Hz), the step to 100 and the number of steps to 30. Look at the graph that shows the microphone output. Select a region of the output that is within the duration of the beep and away from any transients. Put appropriate values in the Index and Length controls (something like initial index =300, length = 500).

Click on the single arrow button at the top of the window to run the program once. It will make a series of beeps of increasing frequency. When the program finishes running, you will see a display of the spectral response.

Experiment with different settings of the program. Make sure the microphone is not saturating, and the amplitude of the signal is less than +/- 32000, the maximum number that can be represented by a 16-bit integer.

4. Real Time Response Mapping

You may notice one annoying feature of the program. It does not display the response graph until it finishes stepping through all points. This can be a problem if you are trying to experiment with various parameters, it would be nice to see what is happening in real time.

We will modify the program so the response signal is updated point by point as the program takes data. Also, we will fix the x-axis of the graph to show the actual frequency at each point.

In the main VI, delete the spectral response waveform graph. In the Graph section of the Controls Palette, find the "XY Graph" and drag it to the window. In the Diagram window, drag the XY graph inside the "For Loop".

Now we need to setup a way to build the data point-by-point. The "XY Graph" requires a data type called a cluster. A cluster is similar to an array, except that its elements can have different data types, like integer or double. For the XY graph we need an array of clusters, each containing two elements, the X and Y coordinates. First, we will initialize the appropriate data structure.

In the Functions Palette, go to the Cluster section (right column, second from top) and select "Bundle". Place it somewhere to the left of the For Loop. Go to the Numeric Section and select "Numeric Constant" (bottom left corner). Drag two numeric constants to the diagram.

The data that we will use are real, so change the representation of the numeric constant to "DBL" by right-clicking on it. Wire the two numeric constants to the two inputs of the Bundle function. You don't need to change the values of the constants.

Now you created a cluster of two doubles and need to create an array of such clusters.

Go to the Array section of the Functions Palette and drag "Initialize Array" (second row first on the left) to the diagram. Connect the output of the Bundle to the "element" input of the array. Right-click on the "dimension size" input and create a constant. This specifies the length of the array and is equal to zero for now.

Go to the For Loop and right click on the left edge of the For Loop rectangle. Choose "Add Shift Register"

The shift register allows you to pass data from one iteration of the For Loop to the next.

Connect the output of the Initialize Array function to the shift register icon on the left-hand side of the For Loop.

Inside the For Loop we need to create a cluster with the new data after each execution of the loop and add it to the array.

From the Cluster functions, select Bundle and place it inside the For Loop. Connect the current frequency value to the upper input of the bundle and the RMS output to the lower input.

Go to the Array functions, drag function called "Insert Into Array" to the diagram and place it inside the For Loop. Connect the Shift Register on the left-hand side of the For Loop to the input array of the "Insert Into Array" function. Connect the output of the "Insert into Array" function to the Shift Register on the right-hand side of the For Loop. Connect the output of the Bundle Function to the input called "new element/subarray" of the "Insert Into Array" function. Finally, connect the XY graph to the wire running from the output of the "Insert Into Array" to the right-hand side Shift Register.

Now you are ready to run the program. You may find that the arrow button for running is "broken". This means that there are some inconsistencies in the program. You can delete any loose wires by pressing Ctrl-B. You may also have some elements left from when the RMS output wire was going through the wall of the For Loop. Delete these elements.

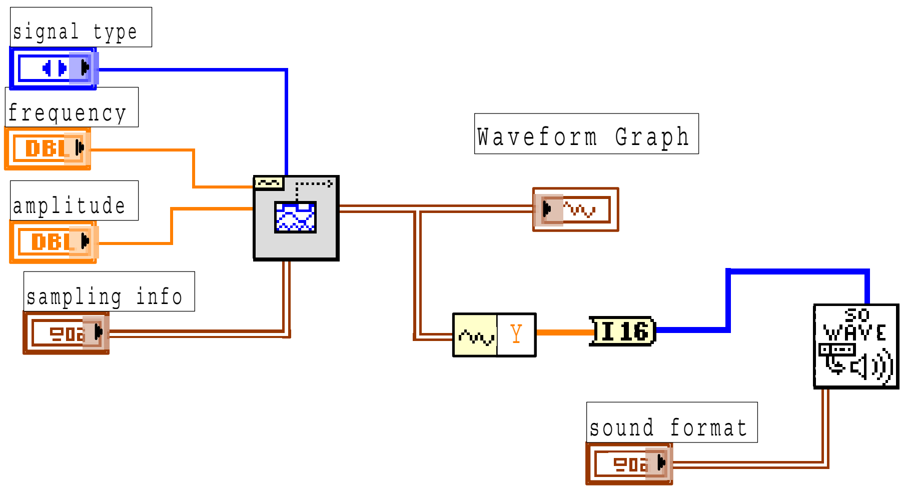
Now go to the main VI window and run the program. It should work similar to what it did before, except now the frequency response data are plotted in real time and the X-axis of the graph is in Hz. To change the name of the X-axis ("Time" by default) use the text tool ("A") and type in a new name, such as "Frequency (Hz)". You can also change the plot range of any graph simply by typing a different number in the axes labels.

5. Additional Exercises

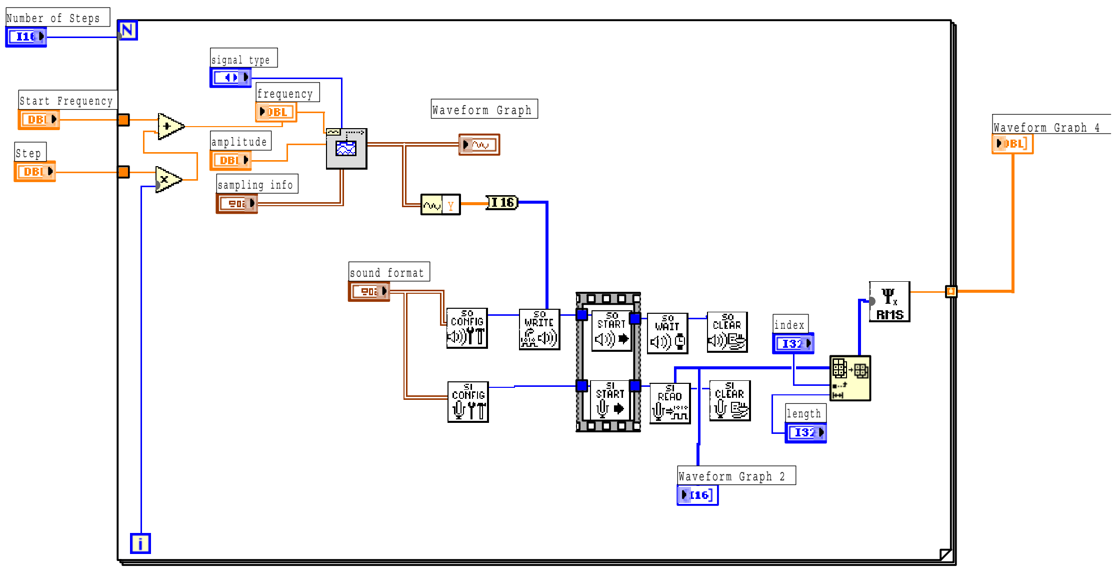
Modify the program to obtain a real-time continuous scrolling display of the microphone input.

Experiment with other possibilities, such as reversing the audio signal in time and playing it back or performing Fourier Transform of the data.

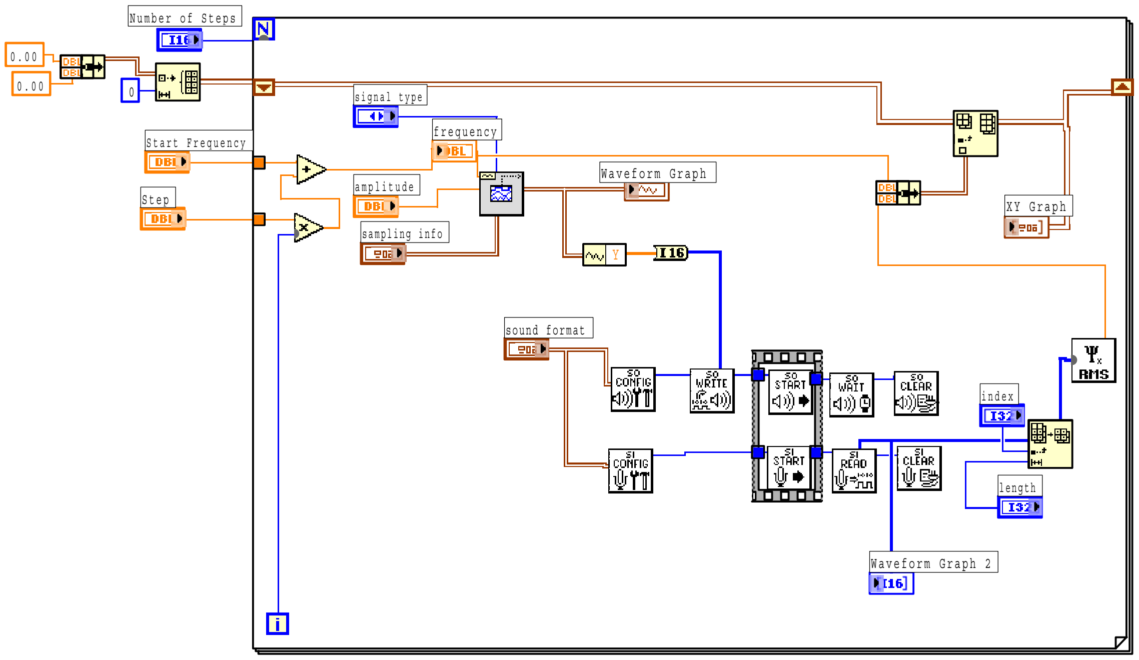
1) Make a Beep



3) Map Sound System Response



4) Real-Time Response Mapping


0.00
0.00
DBL
DBL
0
Number of Steps
I16
N
Start Frequency
DBL
Step
DBL
signal type
frequency
DBL
amplitude
DBL
sampling info
Waveform Graph
Y
I16
XY Graph
sound format
SO CONFIG
SO WRITE
SO START
SO WAIT
SO CLEAR
SI CONFIG
SI START
SI READ
SI CLEAR
index
I32
length
I32
RMS
Waveform Graph 2
[I16]
i