LabVIEW Day 2: Other loops, Other graphs

Vern Lindberg

From now on, I will not include the Programming to indicate paths to icons for the block diagram. I assume you will be getting comfortable with the functions palette. I will include paths to Express or Mathematics.

1 Other loops, and timing

Open a new blank vi. On the block diagram, if you look at Structures you will see the While Loop that we used last time and several other structures. In this course we will eventually discuss

- For Loops
- Case Structures
- Sequence structures (flat or stacked)
- Formula Nodes

1.1 More on the While Loop

(a) Start by building a while loop. Inside the loop create an indicator for the index. Note that the icons and lines are colored blue, the color for integers.

Right click on the index indicator icon and scroll down to Representation. The indicator is of type I32, a 4 byte signed integer.

Place a Numeric→Random Number (the dice) in the loop and create an indicator. Check its representation using right click to see that it is DBL, an 8 byte floating point number.

(b) We do not operate at the speed of the computer, so we want to slow down the rate at which the loop recurs. Place a Timing→Wait (ms) in the loop and create a constant with a value of 1000 attached to the input. The program will do the following: Enter the loop. In parallel, display the index and generate and display the random number,
and start the timer. After 1000 ms, check to see if stop has been pushed, if not increment the index and repeat, if yes increment the index and stop.

You can see the action on the block diagram by clicking the lightbulb icon on the panel toolstrip and running the program.

Once done unclick the lightbulb.

(c) We may also want to see the final value of the index. Wire the index output to the boundary of the loop and a solid square appears—a non-indexed tunnel. Right click and Create an indicator.

Run the program and compare the final index to the value indicated inside the loop.

(d) Now use a different method to terminate the while loop. Delete the stop button and any broken wires. Deposit a Comparison→Greater or Equal? in the loop. Connect the top input to the index, and create a control to the bottom input. Connect the output to the stop sign.

Start with a control value of 2. The loop should stop when the index is greater than or equal to 2. See if you can play computer and predict what the final index should be.

(e) It would be nice to see all the random numbers that were produced, in an array. Wire the output of the random number generator to the boundary of the While. Initially the tunnel is solid and only passes the last value. Right click on the tunnel and Enable Indexing. Create an indicator at the tunnel.

On the front panel you will see an Array indicator that has an Index Display (initially 0 with up and down arrows to the left) and an array display, here an array of floating point numbers.

But you only see one element of the array by default. You can enter numbers 1, 2, 3 … into the Index Display, or use the up and down arrows, or you can hover over the array element, grab the big handles, and drag to display more than one number. Be careful, there are also small handles that adjust the width of the display cells.

(f) Another very important question needs to be answered—If we make the control 0, at the start $0 \geq 0$ is false, do the other elements of the loop execute?

Make the control 0 and run. How many random numbers are produced? Answer the question: if the loop condition is false, will the loop terminate (a) without ever executing the contents of the loop or (b) after computing the contents of the loop once?
1.1.1 Stop if True versus Continue if True

So far you have used the stop sign, and if you right click on it you will see that it is Stop If True. How does it work? The stop button is false until the button is pushed. The index, $i$, is less than the value, so the comparison is false. OR(false, false) is false, so the program continues. Eventually either the stop button becomes true, or the comparison becomes true. Then OR(true, false) is true and the loop stops.

(a) You can right click to make the condition Continue If True, and the stop sign turns into a continue icon. Do this, and also right click on the Compare and Replace→Comparison Palette→Less Than?. Now when you run it should act the same—repeating as long as $i < value$.

1.2 The For Loop

A second major loop is the For Loop. You could deposit Structures→For Loop on a new vi and add the elements that you did for the While Loop, but let’s be more efficient. Hover over the loop boundary, right-click and Replace with For Loop. You will see an index, $i$, and a For Count, $N$. The For Count tells how many times the loop is repeated. Create a control for it, it will appear outside the loop.

You had a random number generator wired to an array, and an indicator for the index. Outside the loop you had indicators for the array and for the last value of the index. Delete both of these, and the wires to the tunnels.

Now rewire the index to a tunnel on the boundary on the loop. You will notice that in For Loops, the default is for indexed tunnels. Right click on the tunnel for the last index, and Disable Indexing to have the tunnel turn back to solid. Create an indicator for the last value of the index.

Wire the random number to a tunnel, and create and array indicator for it.

Run the program with a For Control of value of 2—what is the final value of the index? How many times has the loop been executed?

Try a For Control of 0. How many array elements are produced? This should let you answer the question: if the For loop is executed 0 times, will the loop stop (a) without ever executing the contents of the loop or (b) after computing the contents of the loop once?
1 Other loops, and timing

1.3 Array Displays

Notice that the line carrying the real data in a 1D array is a thick orange line. If you hover over the indicator portion on the front panel, you will see either big or small handles. The big handles adjust the number of array elements shown and the small handles adjust the width of the box for each array element. Drag using the large handles vertically to show several boxes, then run the program with a For Count of 5. You should see 5 elements of the array, but they will appear to extend past the box.

Use the small handles to extend the box size to see all the digits in the answer. Default appears to be 6 digits after the decimal point.

![Array Indicators, and width of indicator cells](image)

**Fig. 1:** Array Indicators, and width of indicator cells

You can control the digits displayed by right clicking on an array element and using *Display Format*. The options should be self evident. Be sure to see the effect of unchecking *Hide Trailing Zeros* and *SI notation*.

Change the value of the Index Display and see what it does.

Figure 1 shows the results of non-indexed tunnels (only the final value shown), versus indexed tunnels (array of values shown), One dimensional arrays can be spread horizontally or vertically to show several values.

A potential problem arises if the width of the indicator cell is too small: only part of a datum is shown. Either widen the cell with handles, or adjust the Display Format of the indicator. For example if the value was in scientific notation and was $0.123456 \times 10^{-19}$ and the box was not wide enough we would see 0.12345 which could easily confuse us if we...
were expecting a small number.

1.4 Auto-indexed For Loops

Once we have an array, we may want to modify each element, or use each element to create another array. An Auto Indexed For Loop is a for loop with the N not connected. The loop uses the size of the input array to know how many times to repeat.

(a) Add a new FOR loop to the right of your existing loop and place a subtract in the new loop. Wire the array of randoms to the top terminal of the subtract, you will see an indexed tunnel appear. Create a constant of 0.5 and connect to the subtract. Wire the output of the subtract to a tunnel on the right side of the loop and Create an indicator. This now produces an array that goes from -0.5 to +0.5.

Because the array already has a size, the second For Loop needs no index, it just does its work on each element of the array.

(b) Also in the new loop place a Comparison→Greater than? and wire it to compare the random number to 0.5. Wire the output to the boundary of the loop and Create an indicator.

Run the program, look at both of the new arrays, and verify that you understand how it works. What is the type of the second array?

2 Charts and Graphs

There are 3 major types of plotting available in LabVIEW, Waveform Charts, Waveform Graphs, and XY Graphs.

Waveform displays are ideal for displaying data taken at regular intervals. Data taken at unequal intervals can only be plotted on an XY display.

2.1 Waveform Charts

Charts are named for strip chart recorders. Prior to computers, a strip chart recorder had a roll of paper that was moved horizontally through at some rate. The vertical position of a pen was set by the input voltage, and the pen created a permanent record on the paper. The speed at which the paper was fed could be varied.

Waveform Charts behave similarly, data is continually added, never erased, and the horizontal axis has no real scale other than the number of the point. If you know the spacing
in time of the data collection, you can figure out the meaning of the horizontal axis, but it is not obvious.

Graphs must be placed on the front panel. Modern→Graph→Waveform Chart places a waveform chart on the front panel, and its icon on the block diagram.

(a) Start with a new blank vi. On the front panel place a Waveform Chart. Go to the block diagram and surround the graph icon with a While Loop.

Control the speed of operation of the loop with a different timing function. Place a Timing→Time Delay icon in the loop, use the default settings, and connect a constant of 0.2 to it. (If the constant is blue, it is an integer and can’t have the desired value. Right click and change the Representation to DBL.)

(b) Figure out how to display the time in seconds inside the loop: hint, use the index.

(c) From the Express→Input place a Simulate Signal icon and add noise to it (the default is fine, Uniform White Noise of amplitude 0.6). Add a control for the frequency.

From Express→Signal Analysis place a Tone Measurement on the block diagram and check Frequency as the only output. Connect the data from the simulate signal to the input of the tone measurements.

(d) In order to illustrate the properties of the Waveform Chart and Waveform Graph, we will convert the DDT\(^1\) data into simple floating point data. Place a Express→Signal Manipulation→From DDT near the tone measurement output. In the configuration window, choose Single Scalar as the Resulting Data Type.

Wire the output of the Tone Measurement Frequency to the From DDT, and the output of the From DDT to the input of the chart.

Run the program for a couple of seconds with a frequency of 10 Hz. Stop it. Change the frequency to 20 Hz and run again. What happens on the chart?

(e) Now change the constant controlling the timing from 0.2 s to 2 s and run the program for another two seconds. Describe what happens to the horizontal spacing of the data points.

In a real chart recorder we would eventually run out of paper and have to replace the roll. For your Waveform Chart you can “replace the roll” by right clicking on the graph and Data Operations→Clear Chart.

Data for the strip chart are stored in a buffer of default length 1024 points. You can change the size of the buffer by right clicking on the chart and going to Chart History Length. Once the buffer is full, the oldest data is dropped and new data are added.

\(^1\) Dynamic Data Type
Later we will discuss means of saving all the data in an array or saving it to the hard drive.

2.2 Multiple Plots on a Chart

If we want two or more plots on a chart we bundle the data together before sending it to the chart.

(a) Double click on the Tone Measurements and check Amplitude as well as Frequency. Close the Configure window.

Duplicate the From DDT icon and wire to the Amplitude output of the Tone Measurement. Delete the wire from the first From DDT to the chart.

(b) Place a Cluster, Class, & Variant→Bundle on the block diagram, default number of inputs is 2, but it can be expanded. Wire the inputs to the Frequency and Amplitude outputs of the respective From DDTs, and wire the output to the chart.

(c) Return the delay to 0.2 s, and run. You should now have two traces on the graph.

2.3 Modes for Waveform Chart

There are three modes for the Waveform Chart chosen from Properties→Appearance→Update Mode. The default is Strip Chart mode where a new datum is added to the existing data on the chart.

The Scope Chart mode keeps the width of the window constant, for example 20 points. Initially the scale would run 0 to 20, and data would plot. Once point 20 is added, the scale would jump to 20 to 40, and data would plot. All data are still saved.

The Sweep Chart also keeps the number of points shown fixed. Once the scale, 0-20 for example, is shown, the scale is marked 20 to 40, data are plotted up to a red vertical line, and to the right of the line, the old data are shown. This makes a lot more sense if you see it.

2.4 Waveform Graphs

A chart is happy simply to add one point at a time. A graph requires a collection of points to plot, in this case that will be an array. For the moment we will produce the graph after all the data are collected, that is outside the loop.
(a) Wire the output of the From DDT for frequency to an indexed tunnel. Similarly wire the From DDT amplitude to an indexed tunnel.

Place another Waveform Chart and a Waveform Graph on the front panel. Place the icons for these outside the loop and connect their inputs to the tunnel.

(b) Set the frequency to 20 Hz and run the program you will see data appearing on the chart that is inside the loop, but not on the other chart and graph. Stop the program and all the data will be sent out of the loop and will create a display.

Set the frequency to about 40 Hz and run the program for a couple of seconds and stop. Compare the results of the chart and the graph.

Essentially, the graph just plots the data from the latest run, while the chart plots all the data.

2.5 Adding time information to the axis

If you look at the context help for the Waveform graph, you will see that there is a method of getting the timing information into the Waveform Graph.

“WDT (Waveform Data Type) includes timing info. Others default to 0 for $x_0$ and 1 for $\Delta x$. Combine timing information using a bundle node.”

In greater detail, this means that to pass detailed information to the graph, we create a bundle of starting time\(^2\) ($x_0$), time interval ($\Delta t$), and the data. The first two items are scalars, and the data is a 1D array. This particular bundle is called a Waveform Data Type.

(a) Unwire the graph and insert a Cluster, Class, & Variant→Bundle. Expand it so it has 3 inputs. Wire a constant 0 for start time to the top input.

Take the time increment from inside the loop to a tunnel and to the middle input of the bundle. Should this tunnel be indexed or non-indexed? Connect the array of to the bottom terminal. Connect the output of the bundle to the input of the graph.

(b) Run the program and you should see real time on the horizontal axis. If you double click on the axis label you can add the units, since it now has units.

(c) Change the wait time from 0.2 s to 0.5 sec and run again for about 5 seconds. Does the new waveform graph display time properly?

\(^2\) Sometimes the start time must be in the form of a time stamp, a more complicated representation that we do not need to worry about now.
2.6 Multiple Plots on one graph

Earlier I described how to get multiple plots on a single waveform chart. Things are slightly different when we have a waveform graph.

(a) Remove the wire connecting the 1D frequency array to the Bundle, and the wire connecting the Bundle output to the graph.

(b) Outside the loop, place an Array→Build Array icon and expand it to two inputs. Connect the 1D arrays for frequency and amplitude to the two inputs. Connect the output to the array input for the Bundle icon. (It is now a 2D array). Wire the output to the chart.

(c) Run the program and you should have two plots on your graph.

3 Things you should have mastered

- Different ways to terminate a while loop, stop if true, continue if true
- Timers: Wait (ms) Time Delay
- For Loop: index, for count
- Auto indexed For Loops
- Tunnels, indexed and non-indexed, default state for While and For loops
- 1D and 2D arrays
- Differences between Waveform Charts and Graphs
- Modes for Waveform Charts
- Adding timing information to a Waveform Graph
- Multiple plots on a single chart or graph