Introduction to LabVIEW

GRAPHICAL PROGRAMMING
FOR ENGINEERS AND SCIENTISTS
How to Succeed in EE 20 Lab

• Work as a group of 2
• Read the lab guide thoroughly
• Use help function and help pages in LabVIEW
• Do the Pre-Lab before you come to the lab
• Don’t do the post-lab independent section at the last minute
Lab Goals

• Become comfortable with the LabVIEW environment and data flow execution
  – Front panels
  – Block diagrams
  – Functions and Controls Palettes
  – Use built-in LabVIEW functions
• Use LabVIEW to solve problems
• Learn LabVIEW concepts
  – Finding and using math and complex analysis functions
  – Working with data types, such as arrays and clusters
  – Displaying results

This lab prepares you to do the following:
• Use LabVIEW to create applications.
• Understand front panels, block diagrams, and icons and connector panes.
• Use built-in LabVIEW functions.
Section I – LabVIEW Environment

A. Components to a LabVIEW Application
   • Front Panel
   • Block Diagram

B. Programming Environment
   • Controls Palette
   • Functions Palette
   • Tools Palette
   • Status Toolbar

C. Additional Help
   • Context Help Window
   • Tips for Working in LabVIEW
LabVIEW

LabVIEW is a graphical programming language that uses icons instead of lines of text to create applications. In contrast to text-based programming languages, where instructions determine program execution, LabVIEW uses dataflow programming, where the flow of data determines execution order.

LabVIEW Example Finder

LabVIEW features hundreds of example VIs you can use and incorporate into VIs that you create. In addition to the example VIs that are shipped with LabVIEW, you can access hundreds of example VIs on the NI Developer Zone (zone.ni.com). You can modify an example VI to fit an application, or you can copy and paste from one or more examples into a VI that you create.
LabVIEW programs are called virtual instruments (VIs).

Controls are inputs and indicators are outputs.

Each VI contains three main parts:

- **Front panel** – How the user interacts with the VI
- **Block diagram** – The code that controls the program
- **Icon/connector** – The means of connecting a VI to other VIs

In LabVIEW, you build a user interface by using a set of tools and objects. The user interface is known as the front panel. You then add code using graphical representations of functions to control the front panel objects. The block diagram contains this code. In some ways, the block diagram resembles a flowchart.

You interact with the front panel when the program is running. You can control the program, change inputs, and see data updated in real time. Controls are used for inputs such as adjusting a slide control to set an alarm value, turning a switch on or off, or stopping a program. Indicators are used as outputs. Thermometers, lights, and other indicators display output values from the program. These may include data, program states, and other information.

Every front panel control or indicator has a corresponding terminal on the block diagram. When you run a VI, values from controls flow through the block diagram, where they are used in the functions on the diagram, and the results are passed into other functions or indicators through wires.
Use the Controls palette to place controls and indicators on the front panel. The Controls palette is available only on the front panel. To view the palette, select View»Controls Palette. You also can display the Controls palette by right-clicking an open area on the front panel. Tack down the Controls palette by clicking the pushpin on the top left corner of the palette.
Use the **Functions** palette to build the block diagram. The **Functions** palette is available only on the block diagram. To view the palette, select **View»Functions Palette**. You also can display the **Functions** palette by right-clicking an open area on the block diagram. Tack down the **Functions** palette by clicking the pushpin on the top left corner of the palette.
Searching for Controls, VIs, and Functions

- Palettes are filled with hundreds of VIs
- Press the search button to index all VIs for text searching
- Click and drag an item from the search window to the block diagram
- Double-click an item to open the owning palette

Use the buttons on top of the palette windows to navigate, search, and edit the palettes.
You can search for controls, VIs, and functions that either contain certain words or start with certain words. Double-clicking a search result opens the palette that contains the search result. You also can click and drag the name of the control, VI, or function directly to the front panel or block diagram.
When you create an object on the front panel, a terminal is created on the block diagram. These terminals give you access to the front panel objects from the block diagram code. Each terminal contains useful information about the front panel object it corresponds to, and uses color and symbols to provide information about the data type. For example, floating data types are represented with the color orange, whereas integer data types are blue. Boolean terminals are often green with TF lettering.

In general, blue terminals should wire to blue terminals, green to green, and so on. This is not a hard-and-fast rule; you can use LabVIEW to connect an orange terminal (floating point data) to a blue terminal (integer data), coercing the data. But in most cases, look for a match in colors.

Controls have a thick border and an arrow on the right side. Indicators have a thin border and an arrow on the left side. Logic rules apply to wiring in LabVIEW: Each wire must have one (but only one) source (or control), and each wire may have multiple destinations (or indicators).
Dataflow Programming

- Block diagram execution
  - Dependent on the flow of data
  - Block diagram does NOT execute left to right

Note:
- Node executes when data is available to ALL input terminals
- Nodes supply data to all output terminals when done

LabVIEW follows a dataflow model for running VIs. A block diagram node executes when all its inputs are available. When a node completes execution, it supplies data to its output terminals and passes the output data to the next node in the dataflow path. Visual Basic, C++, JAVA, and most other text-based programming languages follow a control flow model of program execution. In control flow, the sequential order of program elements determines the execution order of a program.

Consider the block diagram above. It adds two numbers and then multiplies by 2 from the result of the addition. In this case, the block diagram executes from left to right, not because the objects are placed in that order but because one of the inputs of the Multiply function is not valid until the Add function has finished executing and passed the data to the Multiply function. Remember that a node executes only when data are available at all of its input terminals, and it supplies data to its output terminals only when it finishes execution. In the second piece of code, the Simulate Signal Express VI receives input from the controls and passes its result to the graph.

You may consider the add-multiply and the simulate signal code to coexist on the same block diagram in parallel. This means that they begin executing at the same time and run independently of one another. If the computer running this code had multiple processors, these two pieces of code could run independently of one another (each on its own processor) without any additional coding.
If you enable the automatic selection tool and you move the cursor over objects on the front panel or block diagram, LabVIEW automatically selects the corresponding tool from the Tools palette. To view the Tools palette, select View » Tools Palette. You also can display the Tools palette by holding shift and right-clicking an open area on either the front panel or block diagram. Open the Tools Palette Toggle automatic selection tool by clicking the Automatic Selection Tool button in the Tools palette.

Use the Operating Tool to change the values of a control or select the text within a control.

Use the Positioning/Resizing Tool to select, move, or resize objects. The Positioning Tool changes shape when it moves over a corner of a resizable object.

Use the Labeling Tool to edit text and create free labels. The Labeling Tool changes to a cursor when you create free labels.

Use the Wiring Tool to wire objects together on the block diagram.

Other important tools:
• Click the Run button to run the VI. While the VI runs, the Run button appears with a black arrow if the VI is a top-level VI, meaning it has no callers and therefore is not a subVI.

• Click the Continuous Run button to run the VI until you abort or pause it. You also can click the button again to disable continuous running.

• While the VI runs, the Abort Execution button appears. Click this button to stop the VI immediately.

Note: Avoid using the Abort Execution button to stop a VI. Either let the VI complete its data flow or design a method to stop the VI programmatically. By doing so, the VI is at a known state. For example, place a button on the front panel that stops the VI when you click it.

• Click the Pause button to pause a running VI. When you click the Pause button, LabVIEW highlights on the block diagram the location where you paused execution. Click the Pause button again to continue running the VI.

• Select the Text Settings pull-down menu to change the font settings for the VI, including size, style, and color.

• Select the Align Objects pull-down menu to align objects along axes, including vertical, top edge, left, and so on.

• Select the Distribute Objects pull-down menu to space objects evenly, including gaps, compression, and so on.

• Select the Resize Objects pull-down menu to change the width and height of front panel objects.
Debugging Techniques

• Finding Errors
  Click on broken Run button.
  Window showing error appears.

• Execution Highlighting
  Click on Execution Highlighting button; data flow is animated using bubbles. Values are displayed on wires.

• Probes
  Right-click on wire to display probe; it shows data as it flows through wire segment.
  You can also select Probe tool from Tools palette and click on wire.

When your VI is not executable, a broken arrow is displayed in the Run button in the palette.

• Finding Errors: To list errors, click on the broken arrow. To locate the bad object, click on the error message.

• Execution Highlighting: Animates the diagram and traces the flow of the data, allowing you to view intermediate values. Click on the light bulb on the toolbar.

• Probe: Used to view values in arrays and clusters. Click on wires with the Probe tool or right-click on the wire to set probes.

• Retain Wire Values: Used with probes to view the values from the last iteration of the program.

• Breakpoint: Sets pauses at different locations on the diagram. Click on wires or objects with the Breakpoint tool to set breakpoints.
The Context Help window displays basic information about LabVIEW objects and updates as you move the cursor over each object. Objects with context help information include VIs, functions, constants, structures, palettes, properties, methods, events, and dialog box components.

To display the Context Help window, select Help » Show Context Help, press the <Ctrl-H> keys, or press the Show Context Help Window button in the toolbar.

Connections for a subVI or function are displayed in Context Help:

**Required** – bold
**Recommended** – normal
**Optional** – dimmed
Section II – Elements of Typical Programs

A. Data Types

B. Loop Structures
   • While Loop
   • For Loop

C. Conditional Statements
   • Case Structure
   • Select (simple If statement)
LabVIEW uses many common data types – Boolean, numeric, strings, clusters, and so on.

To access functions specific to certain data types, you can refer to the appropriate Sub-Palette from the Functions Palette.
Block Diagram – Wires

- Transfer data between block diagram objects through wires
- Wires are different colors, styles, and thicknesses, depending on their data types
- A broken wire appears as a dashed black line with a red X in the middle

<table>
<thead>
<tr>
<th></th>
<th>DBL Numeric</th>
<th>Integer Numeric</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1D Array</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2D Array</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In LabVIEW, you use wires to connect multiple terminals together to pass data in a VI. You must connect the wires to inputs and outputs that are compatible with the data that is transferred with the wire. For example, you cannot wire an array output to a numeric input. In addition, the direction of the wires must be correct. You must connect the wires to only one input and at least one output. For example, you cannot wire two indicators together. The components that determine wiring compatibility include the data type of the control and/or the indicator and the data type of the terminal.
Block Diagram – Wiring Tips

- Press `<Ctrl-B>` to delete all broken wires
- Right-click and select **Clean Up Wire** to reroute the wire
B. Loop Structures

• While Loop
  – Terminal counts iterations
  – Always runs at least once
  – Runs until stop condition is met

• For Loop
  – Terminal counts iterations
  – Runs according to input $N$ of count terminal

Both the While and For Loops are located on the Functions>Structures palette. The For Loop differs from the While Loop in that the For Loop executes a set number of times. A While Loop stops executing the subdiagram only if the value at the conditional terminal exists.

**While Loops**
Similar to a Do loop or a Repeat-until loop in text-based programming languages, a While Loop, shown at the top right, executes a subdiagram until a condition is met. The While Loop executes the subdiagram until the conditional terminal, an input terminal, receives a specific Boolean value. The default behavior and appearance of the conditional terminal is *Stop If True*. When a conditional terminal is *Stop If True*, the While Loop executes its subdiagram until the conditional terminal receives a TRUE value. The iteration terminal (an output terminal), shown at left, contains the number of completed iterations. The iteration count always starts at zero. During the first iteration, the iteration terminal returns 0.

**For Loops**
A For Loop, shown above, executes a subdiagram a set number of times. The value in the count terminal (an input terminal) represented by the N, indicates how many times to repeat the subdiagram. The iteration terminal (an output terminal), shown at left, contains the number of completed iterations. The iteration count always starts at zero. During the first iteration, the iteration terminal returns 0.
Drawing a Loop

1. Select the structure

2. Enclose code to be repeated

3. Drop or drag additional nodes and then wire

Place loops in your diagram by selecting them from the **Functions** palette:

- When selected, the mouse cursor becomes a special pointer that you use to enclose the section of code you want to include in the While Loop.
- Click the mouse button to define the top-left corner and then click the mouse button again at the bottom-right corner. The While Loop boundary appears around the selected code.
- Drag or drop additional nodes in the While Loop if needed.
C. Conditional Statements

1. Case Structures

   ![Diagram of Case Structure]

Case Structure
The case structure has one or more subdiagrams, or cases, one of which executes when
the structure executes. The value wired to the selector terminal determines which case to
execute and can be Boolean, string, integer, or enumerated type. Right-click the structure
border to add or delete cases. Use the Labeling tool to enter value(s) in the case selector
label and configure the value(s) handled by each case. It is found at Functions»
Programming»Structures»Case Structure.

2. Select

   ![Diagram of Select Function]

Select
Returns the value wired to the t input or f input, depending on the value of s. If s is
TRUE, this function returns the value wired to t. If s is FALSE, this function returns the
value wired to f. The connector pane displays the default data types for this polymorphic
function. It is found at Functions»Programming»Comparison»Select.

- **Example A**: Boolean input. Simple if-then case. If the Boolean input is TRUE, the true
case executes; otherwise the FALSE case executes.

- **Example B**: Numeric input. The input value determines which box to execute. If out of
range of the cases, LabVIEW chooses the default case.

- **Example C**: When the Boolean passes a TRUE value to the Select VI, the value 5 is
passed to the indicator. When the Boolean passes a FALSE value to the Select VI, 0 is
passed to the indicator.
Section III–Presenting Your Results

• Displaying Data on the Front Panel
  • Controls and Indicators
  • Graphs and Charts
What Types of Controls and Indicators Are Available?

- **Numeric Data**
  - Number Input and Display
  - Analog Sliders, Dials, and Gauges
- **Boolean Data**
  - Buttons and LEDs
- **Array and Matrix Data**
  - Numeric Display
  - Chart
  - Graph
  - XY Graph
  - Intensity Graph
  - 3D Graph: Point, Surface, and Model
- **Decorations**
  - Tab Control
  - Arrows
- **Other**
  - Strings and Text Boxes
  - Picture/Image Display
  - ActiveX Controls

Controls and indicators are front panel items that you can use to interact with your program to provide input and display results. You can access controls and indicators by right-clicking the front panel.

In addition, you obtain additional controls and indicators when you install toolkits and modules.

For example, when you install the control design tools, you can use specialized plots, such as Bode and Nyquist, that are not available by default.
The waveform chart is a special numeric indicator that displays one or more plots. It is located on the Controls»Modern»Graph palette. Waveform charts can display single or multiple plots. The following front panel shows an example of a multiplot waveform chart.

You can change the minimum and maximum values of either the x-axis or y-axis by double-clicking on the value with the labeling tool and typing the new value. Similarly, you can change the label of the axis. You can also right-click the plot legend and change the style, shape, and color of the trace that is displayed on the chart.
Graphs – Display Many Data Points at Once

Waveform graph – special numeric indicator that displays an array of data
• Graph updates after all points have been collected
• May be used in a loop if VI collects buffers of data

Controls»Express»Graph Indicators»Graph

ni.com

Graphs are very powerful indicators in LabVIEW. You can use these highly customizable tools to concisely display a great deal of information.

With the properties page of the graph, you can display settings for plot types, scale and cursor options, and many other features of the graph. To open the properties page, right-click the graph on the front panel and choose Properties.

You can also create technical-paper-quality graphics with the “export simplified image” function. Right-click the graph and select Data Operations»Export Simplified Image…
Tips for Working in LabVIEW

• Keystroke Shortcuts
  – <Ctrl-H> – Activate/Deactivate Context Help Window
  – <Ctrl-B> – Remove Broken Wires from Block Diagram
  – <Ctrl-E> – Toggle between Front Panel and Block Diagram
  – <Ctrl-Z> – Undo (also in Edit menu)
  – <Right-Click> – Open Controls/Functions Palette
• Tools»Options… – Set Preferences in LabVIEW

LabVIEW has many keystroke shortcuts that make working easier. The most common shortcuts are listed above.