Introduction to LabVIEW

GRAPHICAL PROGRAMMING
FOR ENGINEERS AND SCIENTISTS

LabVIEW™ 8
Course Goals

- Become comfortable with the LabVIEW environment and data flow execution
- Ability to use LabVIEW to solve problems
- LabVIEW Concepts
  - Acquiring, saving and loading data
  - Find and use math and complex analysis functions
  - Work with data types, such as arrays and clusters
  - Displaying and printing results

This is a list of the objectives of the course.

This course 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.
- Create and save programs in LabVIEW so you can use them as subroutines.
- Create applications that use plug-in DAQ devices.

This course does not describe any of the following:
- Programming theory
- Every built-in LabVIEW function or object
- Analog-to-digital (A/D) theory

NI does provide free reference materials on the above topics on ni.com.

The LabVIEW Help is also very helpful:
LabVIEW » Help » Search the LabVIEW Help…
Virtual Instrumentation

For more than 25 years, National Instruments has revolutionized the way engineers and scientists in industry, government, and academia approach measurement and automation. Leveraging PCs and commercial technologies, virtual instrumentation increases productivity and lowers costs for test, control, and design applications through easy-to-integrate software, such as NI LabVIEW, and modular measurement and control hardware for PXI, PCI, USB, and Ethernet.

With virtual instrumentation, engineers use graphical programming software to create user-defined solutions that meet their specific needs, which is a great alternative to proprietary, fixed functionality traditional instruments. Additionally, virtual instrumentation capitalizes on the ever-increasing performance of personal computers. For example, in test, measurement, and control, engineers have used virtual instrumentation to downsize automated test equipment (ATE) while experiencing up to a 10 times increase in productivity gains at a fraction of the cost of traditional instrument solutions.
National Instruments LabVIEW is an industry-leading software tool for designing test, measurement, and control systems. Since its introduction in 1986, engineers and scientists worldwide who have relied on NI LabVIEW graphical development for projects throughout the product design cycle have gained improved quality, shorter time to market, and greater engineering and manufacturing efficiency. By using the integrated LabVIEW environment to interface with real-world signals, analyze data for meaningful information, and share results, you can boost productivity throughout your organization. Because LabVIEW has the flexibility of a programming language combined with built-in tools designed specifically for test, measurement, and control, you can create applications that range from simple temperature monitoring to sophisticated simulation and control systems. No matter what your project is, LabVIEW has the tools necessary to make you successful quickly.
Virtual Instrumentation Applications

Virtual instrumentation is applicable in many different types of applications, starting from design to prototyping and deployment. The LabVIEW platform provides specific tools and models to solve specific applications ranging from designing signal processing algorithms to making voltage measurements and can target any number of platforms from the desktop to embedded devices – with an intuitive, powerful graphical paradigm.

With version 8, LabVIEW scales from design and development on PCs to several embedded targets from ruggedized toaster size prototypes to embedded systems on chips. **LabVIEW streamlines system design with a single graphical development platform.** In doing so, LabVIEW encompasses better management of distributed, networked systems because as the targets for LabVIEW grow varied and embedded, you will need to be able to more easily distribute and communicate between various LabVIEW code pieces in your system.
Integrated Hardware Platforms

A virtual instrument consists of an industry-standard computer or workstation equipped with powerful application software, cost-effective hardware such as plug-in boards, and driver software, which together perform the functions of traditional instruments.

Virtual instruments represent a fundamental shift from traditional hardware-centered instrumentation systems to software-centered systems that exploit the computing power, productivity, display, and connectivity capabilities of popular desktop computers and workstations.

Although the PC and integrated circuit technology have experienced significant advances in the last two decades, software truly offers the flexibility to build on this powerful hardware foundation to create virtual instruments, providing better ways to innovate and significantly reduce cost. With virtual instruments, engineers and scientists build measurement and automation systems that suit their needs exactly (user-defined) instead of being limited by traditional fixed-function instruments (vendor-defined).
Section I – LabVIEW Environment

A. Getting Data into your Computer
   • Data Acquisition Devices
     – NI-DAQ
     – Simulated Data Acquisition
     – Sound Card

B. LabVIEW Environment
   • Front Panel / Block Diagram
   • Toolbar / Tools Palette

C. Components of a LabVIEW Application
   • Creating a VI
   • Data Flow Execution

D. Additional Help
   • Finding Functions
   • Tips for Working in LabVIEW
What type of device should I use?

There are many types of data acquisition and control devices on the market. A few have been highlighted above. The trade-off usually falls between sampling rate (samples/second), resolution (bits), number of channels, and data transfer rate (usually limited by “bus” type: USB, PCI, PXI, etc.). Multifunction DAQ (data acquisition) devices are ideal because they can be used in a wide range of applications.

**USB-6008 & USB-6009 Low-Cost USB DAQ**

The National Instruments USB-6009 provides basic data acquisition functionality for applications such as simple data logging, portable measurements, and academic lab experiments. The NI USB-6008 and NI USB-6009 are ideal for students. Create your own measurement application by programming the NI USB-6009 using LabVIEW and NI-DAQmx driver software for Windows. For Mac OS X and Linux users, download and use the NI-DAQmx Base driver.

**NI USB-6009 Specifications:**

- Eight 14-bit analog inputs
- 12 digital I/O lines
  - 2 analog outputs
  - 1 counter

http://www.ni.com/daq/
What is MAX?

• MAX stands for Measurement & Automation Explorer.
• MAX configures and organizes all your National Instruments DAQ, PCI/PXI instruments, GPIB, IMAQ, IVI, Motion, VISA, and VXI devices.
• Used for configuring and testing devices.

The next level of software we are concerned with is called Measurement & Automation Explorer (MAX). MAX is a software interface that gives you access to all of your National Instruments DAQ, GPIB, IMAQ, IVI, Motion, VISA, and VXI devices. The shortcut to MAX will be placed on your desktop after installation. A picture of the icon is shown above. MAX is mainly used to configure and test your National Instruments hardware, but it does offer other functionality such as checking to see if you have the latest version of NI-DAQ installed. When you run an application using NI-DAQmx, the software reads the MAX configuration to determine the devices you have configured. Therefore, you must configure DAQ devices first with MAX.

The functionality of MAX is broken into seven categories:
• Data Neighborhood
• Devices and Interfaces
• IVI Instruments
• Scales
• Historical Data
• Software
• VI Logger Tasks

For this course, we will focus on Data Neighborhood, Devices and Interfaces, Scales, and Software. We will now step through each one of these categories and learn about the functionality each one offers.
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.

You can purchase several add-on software toolkits for developing specialized applications. All the toolkits integrate seamlessly in LabVIEW. Refer to the National Instruments Web site for more information about these toolkits.

LabVIEW also includes several wizards to help you quickly configure your DAQ devices and computer-based instruments and build applications.

LabVIEW Example Finder

LabVIEW includes hundreds of example VIs you can use and incorporate into VIs that you create. In addition to the example VIs that ship with LabVIEW, you also 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)

Each VI has 2 Windows

Front Panel
- User Interface (UI)
  - Controls = Inputs
  - Indicators = Outputs

Block Diagram
- Graphical Code
  - Data travels on wires from controls through functions to indicators
  - Blocks execute by Dataflow

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 – 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.

Users interact with the Front Panel when the program is running. Users 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 to stop 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 a VI is run, 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 Window»Show 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.
Functions (and Structures) 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 **Window»Show 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.
**Tools Palette**

- Recommended: Automatic Selection Tool
- Tools to operate and modify both front panel and block diagram objects

Automatically chooses among the following tools:

- Operating Tool
- Positioning/Resizing Tool
- Labeling Tool
- Wiring Tool

If automatic tool selection is enabled and you move the cursor over objects on the front panel or block diagram, LabVIEW automatically selects the corresponding tool from the Tools palette. Toggle automatic tool selection by clicking the **Automatic Tool Selection** 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 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:

- Scrolling Tool
- Breakpoint Tool
- Probe Tool
- Color Copy Tool
- Coloring Tool
- Shortcut Menu Tool
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.
Demonstration 1: Creating a VI

When you create an object on the Front Panel, a terminal will be 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. For example, the color and symbols provide information about the data type. For example: The dynamic data type is a polymorphic data type represented by dark blue terminals. Boolean terminals are 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; LabVIEW will allow a user to connect a blue terminal (dynamic data) to an orange terminal (fractional value), for example. But in most cases, look for a match in colors.

Controls have an arrow on the right side and have a thick border. Indicators have an arrow on the left and a thin border. 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
- 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 co-exist on the same block diagram in parallel. This means that they will both begin executing at the same time and run independent of one another. If the computer running this code had multiple processors, these two pieces of code could run independent of one another (each on its own processor) without any additional coding.
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 and 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 in conjunction with probes to view the values from the last iteration of the program.

• Breakpoint: Set pauses at different locations on the diagram. Click on wires or objects with the Breakpoint tool to set breakpoints.
Context Help Window

• **Help»Show Context Help**, press the <Ctrl+H> keys
• Hover cursor over object to update window

Additional Help

– Right-Click on the VI icon and choose Help, or
– Choose “**Detailed Help**.” on the context help window

The **Context Help window** displays basic information about LabVIEW objects when 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 displayed in Context Help:

**Required** – bold
**Recommended** – normal
**Optional** – dimmed

Additional Help

• **VI, Function, & How-To Help is also available.**
  – **Help» VI, Function, & How-To Help**
  – Right-click the VI icon and choose Help, or
  – Choose “**Detailed Help**.” on the context help window.

• **LabVIEW Help – reference style help**
  – **Help»Search the LabVIEW Help…**
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)

• Tools » Options… – Set Preferences in LabVIEW

• VI Properties – Configure VI Appearance, Documentation, etc.

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

While the Automatic Selection Tool is great for choosing the tool you would like to use in LabVIEW, there are sometimes cases when you want manual control. Once the Automatic Selection Tool is turned off, use the Tab key to toggle between the four most common tools (Operate Value, Position/Size/Select, Edit Text, Set Color on Front Panel and Operate Value, Position/Size/Select, Edit Text, Connect Wire on Block Diagram). Once you are finished with the tool you choose, you can press <Shift+Tab> to turn the Automatic Selection Tool back on.

In the Tools » Options… dialog, there are many configurable options for customizing your Front Panel, Block Diagram, Colors, Printing, and much more.

Similar to the LabVIEW Options, you can configure VI specific properties by going to File » VI Properties… There you can document the VI, change the appearance of the window, and customize it in several other ways.
Section II – Elements of Typical Programs

A. Loops
   • While Loop
   • For Loop

B. Functions and SubVIs
   • Types of Functions
   • Creating Custom Functions (SubVI)
   • Functions Palette & Searching

C. Decision Making and File IO
   • Case Structure
   • Select (simple If statement)
   • File I/O
Loops

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

• For Loops
  – Terminal counts iterations
  – Run 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 sub diagram 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.
Place loops in your diagram by selecting them from the Structures palette of the Functions palette:

- When selected, the mouse cursor becomes a special pointer that you use to enclose the section of code you want to repeat.
- Click the mouse button to define the top-left corner, click the mouse button again at the bottom-right corner, and the While Loop boundary is created around the selected code.
- Drag or drop additional nodes in the While Loop if needed.
3 Types of Functions (from the Functions Palette)

Express VIs: interactive VIs with configurable dialog page (blue border)

Standard VIs: modularized VIs customized by wiring (customizable)

Functions: fundamental operating elements of LabVIEW; no front panel or block diagram (yellow)

LabVIEW 7.0 introduced a new type of subVI called Express VIs. These are interactive VIs that have a configuration dialog box that allows the user to customize the functionality of the Express VI. LabVIEW then generates a subVI based on these settings.

SubVIs are VIs (consisting of a front panel and a block diagram) that are used within another VI.

Functions are the building blocks of all VIs. Functions do not have a front panel or a block diagram.
**What Types of Functions are Available?**

- **Input and Output**
  - Signal and Data Simulation
  - Acquire and Generate Real Signals with DAQ
  - Instrument I/O Assistant (Serial & GPIB)
  - ActiveX for communication with other programs

- **Analysis**
  - Signal Processing
  - Statistics
  - Advanced Math and Formulas
  - Continuous Time Solver

- **Storage**
  - File I/O

LabVIEW includes several hundreds of pre-built functions that help you to acquire, analyze, and present data. You would generally use these functions as outlined on the slide above.

**LabVIEW Toolkits**

Additional toolkits are available for adding domain specific functionality to LabVIEW. These toolkits include:

<table>
<thead>
<tr>
<th>Application Deployment and Targeting Modules</th>
<th>Signal Processing and Analysis</th>
<th>Control Design and Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>* LabVIEW PDA Module</td>
<td>* Sound and Vibration Toolkit</td>
<td>* Control Design and Simulation Bundle</td>
</tr>
<tr>
<td>* LabVIEW Real-Time Module</td>
<td>* Advanced Signal Processing Toolkit</td>
<td>* LabVIEW Real-Time Module</td>
</tr>
<tr>
<td>* LabVIEW FPGA Module</td>
<td>* Modulation Toolkit</td>
<td>* System Identification Toolkit</td>
</tr>
<tr>
<td>* LabVIEW Vision Development Module</td>
<td>* Spectral Measurements Toolkit</td>
<td>* Control Design Toolkit</td>
</tr>
<tr>
<td></td>
<td>* Order Analysis Toolkit</td>
<td>* LabVIEW Simulation Module</td>
</tr>
<tr>
<td></td>
<td>* Digital Filter Design Toolkit</td>
<td>* State Diagram Toolkit</td>
</tr>
<tr>
<td>Embedded System Deployment</td>
<td>Software Engineering and Optimization Tools</td>
<td>Image Processing and Acquisition</td>
</tr>
<tr>
<td>* DSP Test Integration Toolkit</td>
<td>* Execution Trace Toolkit for LabVIEW Real-Time</td>
<td>* LabVIEW Vision Development Module</td>
</tr>
<tr>
<td>* Embedded Test Integration Toolkit</td>
<td>* Express VI Development Toolkit</td>
<td>* NI Vision Builder for Automated Inspection</td>
</tr>
<tr>
<td>* Digital Filter Design Toolkit</td>
<td>* State Diagram Toolkit</td>
<td>* NI-IMAQ for IEEE 1394</td>
</tr>
<tr>
<td>* LabVIEW FPGA Module</td>
<td>* VI Analyzer Toolkit</td>
<td></td>
</tr>
</tbody>
</table>

http://www.ni.com/toolkits/
Searching for Controls, VIs, and Functions

- Palettes are filled with hundreds of VIs
- Press the search button to index the 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.
Create SubVI

- Enclose area to be converted into a subVI.
- Select Edit » Create SubVI from the Edit Menu.

Creating SubVIs

After you build a VI, you can use it in another VI. A VI called from the block diagram of another VI is called a subVI. You can reuse a subVI in other VIs. To create a subVI, you need to build a connector pane and create an icon.

A subVI node corresponds to a subroutine call in text-based programming languages. A block diagram that contains several identical subVI nodes calls the same subVI several times.

The subVI controls and indicators receive data from and return data to the block diagram of the calling VI. Click the Select a VI icon or text on the Functions palette, navigate to and double-click a VI, and place the VI on a block diagram to create a subVI call to that VI.

A subVI input and output terminals and the icon can be easily customized. Follow the instructions below to quickly create a subVI.

Creating SubVIs from Sections of a VI

Convert a section of a VI into a subVI by using the Positioning tool to select the section of the block diagram you want to reuse and selecting Edit » Create SubVI. An icon for the new subVI replaces the selected section of the block diagram. LabVIEW creates controls and indicators for the new subVI, automatically configures the connector pane based on the number of control and indicator terminals you selected, and wires the subVI to the existing wires.

See Help » Search the LabVIEW Help… » SubVIs for more information.
# LabVIEW Functions and SubVIs operate like Functions in other languages

## Function Pseudo Code

```c
function average (in1, in2, out) {
    out = (in1 + in2)/2.0;
}
```

## Calling Program Pseudo Code

```c
main {
    average (in1, in2, pointavg)
}
```

## SubVI Block Diagram

![SubVI Block Diagram](image)

## Calling VI Block Diagram

![Calling VI Block Diagram](image)

A subVI node corresponds to a subroutine call in text-based programming languages. The node is not the subVI itself, just as a subroutine call statement in a program is not the subroutine itself. A block diagram that contains several identical subVI nodes calls the same subVI several times. The modular approach makes applications easier to debug and maintain.

The functionality of the subVI does not matter for this example. The important point is the passing of two numeric inputs and one numeric output.
How Do I Make Decisions in LabVIEW?

1. Case Structures

The Case Structure has one or more subdiagrams, or cases, exactly 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

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 will execute; otherwise the FALSE case will execute.
- **Example b:** Numeric input. The input value determines which box to execute. If out of range of the cases, LabVIEW will choose 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.
File I/O

File I/O – passing data to and from files
• Files can be binary, text, or spreadsheet
• Write/Read LabVIEW Measurements file (*.lvm)

Use LabVIEW measurement data files to save data that the Write Measurement File Express VI generates. The LabVIEW data file is a tab-delimited text file you can open with a spreadsheet application or a text-editing application. In addition to the data an Express VI generates, the .lvm file includes information about the data, such as the date and time the data was generated.

File I/O operations pass data from memory to and from files. In LabVIEW, you can use File I/O functions to:
• Open and close data files
• Read data from and write data to files
• Read from and write to spreadsheet-formatted files
• Move and rename files and directories
• Change file characteristics
• Create, modify, and read a configuration file
• Write to or read from LabVIEW Measurements files.

In the next example we will examine how to write to or read from LabVIEW Measurements files (*.lvm files).
File I/O Programming Model – Under the hood

Programming Model for the Intermediate File VIs

This same programming model applies to data acquisition, instrument control, file I/O, and most other communication schemes. In most instances you will open the file or communication channel, read and write multiple times, and then the communication will be closed or ended. It is also good programming practice to check for errors at the end. Remember this programming model when you move on to more advanced programming or look inside DAQ, communication, or file I/O Express VIs.

File I/O VIs and Functions

Use the File I/O VIs and functions to open and close files, read from and write to files, create directories and files you specify in the path control, retrieve directory information, and write strings, numbers, arrays, and clusters to files.

Use the high-level File I/O VIs located on the top row of the palette to perform common I/O operations, such as writing to or reading from various types of data. Acceptable types can include characters or lines in text files, 1D or 2D arrays of single-precision numeric values in spreadsheet text files, 1D or 2D arrays of single-precision numeric values in binary files, or 16-bit signed integers in binary files.

Use low-level File I/O VIs and functions located on the middle row of the palette, and the Advanced File Functions to control each file I/O operation individually.

Use the principal low-level functions to create or open, write data to, read data from, and close a file. You also can use low-level functions to create directories; move, copy, or delete files; list directory contents; change file characteristics; or manipulate paths.

Refer to the NI Developer Zone for more information about choosing a file format.
Section III – Presenting your Results

A. Displaying Data on the Front Panel
   • Controls and Indicators
   • Graphs and Charts
   • Loop Timing

B. Signal Processing
   • MathScript
   • Arrays
   • Clusters
   • Waveforms
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 & 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 allow the user to interact with your program to both provide input and display results. You can access Controls and Indicators by right-clicking the front panel.

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

For example, when you install the Control Design tools, you will get specialized plots such as Bode and Nyquist plots that are not available by default.
Charts – Add 1 data point at a time with history

**Waveform chart** – special numeric indicator that can display a history of values

- Chart updates with each individual point it receives

**Functions** » **Express** » **Graph Indicators** » **Chart**

The waveform chart is a special numeric indicator that displays one or more plots. The waveform chart 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 multi-plot waveform chart.

You can change the min and max values of either the x 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

**Functions»Express»Graph Indicators»Graph**

Graphs are very powerful indicators in LabVIEW. They are highly customizable, and can be used to concisely display a great deal of information.

The properties page of the graph allows you to 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**.

Graphs also allow you to create technical paper quality graphics with the “export simplified image” function. Right-click the graph, select **Data Operations»Export Simplified Image**…
Building Arrays with Loops (Auto-Indexing)

- Loops can accumulate arrays at their boundaries with auto-indexing
- For Loops auto-index by default
- While Loops output only the final value by default
- Right-click tunnel and enable/disable auto-indexing

For Loops and While Loops can index and accumulate arrays at their boundaries. This is known as auto-indexing.
- The indexing point on the boundary is called a tunnel.
- The For Loop default is auto-indexing enabled.
- The While Loop default is auto-indexing disabled.

Examples:
- Enable auto-indexing to collect values within the loop and build the array. All values are placed in array upon exiting loop.
- Disable auto-indexing if you are interested only in the final value.
Creating an Array (Step 1 of 2)

From the Controls»Modern»Array, Matrix, and Cluster subpalette, select the **Array** icon.

To create an array control or indicator as shown, select an array on the Controls»Modern»Array, Matrix, and Cluster palette, place it on the front panel, and drag a control or indicator into the array shell. If you attempt to drag an invalid control or indicator such as an XY graph into the array shell, you are unable to drop the control or indicator in the array shell.

You must insert an object in the array shell before you use the array on the block diagram. Otherwise, the array terminal appears black with an empty bracket.
Create an Array (Step 2 of 2)

1. Place an Array Shell.
2. Insert datatype into the shell (i.e. Numeric Control).

To add dimensions to an array one at a time, right-click the index display and select Add Dimension from the shortcut menu. You also can use the Positioning tool to resize the index display until you have as many dimensions as you want.

1D Array Viewing a Single Element:

1D Array Viewing Multiple Elements:

2D Array Viewing a Single Element:

2D Array Viewing Multiple Elements:
How Do I Time a Loop?

1. Loop Time Delay
   - Configure the Time Delay Express VI for seconds to wait each iteration of the loop (works on For and While loops).

2. Timed Loops
   - Configure special timed While loop for desired $dt$.

**Time Delay**
The Time Delay Express VI delays execution by a specified number of seconds. Following the rules of Data Flow Programming, the while loop will not iterate until all tasks inside of it are complete, thus delaying each iteration of the loop.

**Timed Loops**
Executes each iteration of the loop at the period you specify. Use the Timed Loop when you want to develop VIs with multi-rate timing capabilities, precise timing, feedback on loop execution, timing characteristics that change dynamically, or several levels of execution priority.

Double-click the Input Node or right-click the Input Node and select **Configure Timed Loop** from the shortcut menu to display the Loop Configuration dialog box, where you can configure the Timed Loop. The values you enter in the Loop Configuration dialog box appear as options in the Input Node.

**Wait Until Next ms Multiple**
Waits until the value of the millisecond timer becomes a multiple of the specified **millisecond multiple**. Use this function to synchronize activities. You can call this function in a loop to control the loop execution rate. However, it is possible that the first loop period might be short. This function makes asynchronous system calls, but the nodes themselves function synchronously. Therefore, it does not complete execution until the specified time has elapsed.

*Functions* » Programming » Timing » Wait Until Next ms Multiple
Control & Indicator Properties

- Properties are characteristics or qualities about an object
- Properties can be found by right clicking on a Control or Indicator

  - Properties Include:
    - Size
    - Color
    - Plot Style
    - Plot color

- Features include:
  - Cursors
  - Scaling

Properties are all the qualities of a front panel object. With properties, you can set or read such characteristics as foreground and background color, data formatting and precision, visibility, descriptive text, size and location on the front panel, and so on.
Textual Math in LabVIEW

- Integrate existing scripts with LabVIEW for faster development
- Interactive, easy-to-use, hands-on learning environment
- Develop algorithms, explore mathematical concepts, and analyze results using a single environment
- Freedom to choose the most effective syntax, whether graphical or textual within one VI

Supported Math Tools:
- MathScript script node
- Mathematica software
- Maple software
- MathSoft software
- MATLAB® software
- Xmath software

Overview

With the release of National Instruments LabVIEW 8, you have new freedom to choose the most effective syntax for technical computing, whether you are developing algorithms, exploring DSP concepts, or analyzing results. You can instrument your scripts and develop algorithms on the block diagram by interacting with popular third-party math tools such as The MathWorks Inc. MATLAB software, Mathematica, Maple, Mathcad, IDL and Xmath. Use of these math tools with LabVIEW is achieved in a variety of ways depending on the vendor as listed below:

Native LabVIEW textual math node:
- MathScript node, Formula node

Communication with vendor software through LabVIEW node:
- Xmath node, MATLAB script node, Maple* node, IDL* node

Communication with vendor software through VI Server:
- Mathematica* VIs, and Mathcad* VIs

In LabVIEW 8, you can combine the intuitive LabVIEW graphical dataflow programming with MathScript, a math-oriented textual programming language that is generally compatible with popular m-file script language.

*LabVIEW toolkit specific to the math tool must be installed.
Math with the MathScript Node

- Implement equations and algorithms textually
- Input and Output variables created at the border
- Generally compatible with popular m-file script language
- Terminate statements with a semicolon to disable immediate output

Prototype your equations in the interactive MathScript Window.

The MathScript Node enhances LabVIEW by adding a native text-based language for mathematical algorithm implementation in the graphical programming environment. M-file scripts you’ve written and saved from the MathScript window can be opened and used in the MathScript node. M-file scripts you created in other math software will generally run as well. The MathScript allows you to pick the syntax you are most comfortable with to solve the problem. Equations can be instrumented with the MathScript Node for parameter exploration, simulation, or deployment in a final application.

The MathScript Node:
- Located in the Programming » Structures subpalette.
- Resizable box for entering textual computations directly into block diagrams.
- To add variables, right-click and choose Add Input or Add Output.
- Name variables as they are used in formula. (Names are case sensitive.)
- The data type of the output can be changed by right-clicking the input or output node.
- Statements should be terminated with a semicolon to suppress output.
- Ability to import & export m-files by right-clicking on the node.
The Interactive MathScript Window

- Rapidly develop and test algorithms
- Share Scripts and Variables with the Node
- View /Modify Variable content in 1D, 2D, and 3D

The MathScript Window provides an interactive environment where equations can be prototyped and calculations can be made. The MathScript Window and Node share a common syntax and global variables making the move from prototype to implementation seamless. The data preview pane provides a convenient way to view variable data as numbers, graphically, or audibly (with soundcard support).

Help for MathScript
Help for the environment can be accessed using the Mathscript Interactive Environment Window. Type Help in the command window for an introduction to MathScript help. Help followed by a function will display help specific to that function.

Features of the interactive MathScript Window:

- Prototype equations and formulas through the command Window
- Easily access function help by typing Help <function> in the Command Window
- Select a variable to display its data in the Preview Pane and even listen to the result
- Write, Save, Load, and Run m-files using the Script tab
- Share data between the MathScript Node in LabVIEW and the MathScript Window using Global Variables
- Advanced plotting features and image export features
LabVIEW utilizes many common datatypes. These Datatypes include:

- **Boolean**: Numeric, Arrays, Strings, Clusters, and more.
- The color and symbol of each terminal indicate the data type of the control or indicator. Control terminals have a thicker border than indicator terminals. Also, arrows appear on front panel terminals to indicate whether the terminal is a control or an indicator. An arrow appears on the right if the terminal is a control, and an arrow appears on the left if the terminal is an indicator.

**Definitions**

- **Array**: Arrays group data elements of the same type. An array consists of elements and dimensions. Elements are the data that make up the array. A dimension is the length, height, or depth of an array. An array can have one or more dimensions and as many as \(2^{31} - 1\) elements per dimension, memory permitting.
- **Cluster**: Clusters group data elements of mixed types, such as a bundle of wires in a telephone cable, where each wire in the cable represents a different element of the cluster.

See **Help>Search the LabVIEW Help…** for more information. The *LabVIEW User Manual* on ni.com provides additional reference for data types found in LabVIEW.
Section IV – Advanced Data Flow Topics (optional)

A. Additional Data types
   • Cluster

B. Data Flow Constructs
   • Shift Register
   • Local Variables

C. Large Application Development
   • Navigator Window
   • LabVIEW Projects
Introduction to Clusters

• Data structure that groups data together
• Data may be of different types
• Analogous to *struct* in C
• Elements must be either all controls or all indicators
• Thought of as wires bundled into a cable
• **Order is important**

Clusters group like or unlike components together. They are equivalent to a *record* in Pascal or a *struct* in C.

Cluster components may be of different data types.

**Examples:**

- Error information—Grouping a Boolean error flag, a numeric error code, and an error source string to specify the exact error.
- User information—Grouping a string indicating a user’s name and an ID number specifying their security code.

All elements of a cluster must be either controls or indicators. You cannot have a string control and a Boolean indicator. Clusters can be thought of as grouping individual wires (data objects) together into a cable (cluster).
Creating a Cluster

1. Select a Cluster shell.
2. Place objects inside the shell.

Controls » Modern » Array, Matrix & Cluster

Cluster front panel object can be created by choosing Cluster from the Controls » Modern » Array, Matrix & Cluster palette.

- This option gives you a shell (similar to the array shell when creating arrays).
- You can size the cluster shell when you drop it.
- Right-click inside the shell and add objects of any type.

**Note: You can even have a cluster inside of a cluster.**

The cluster becomes a control or an indicator cluster based on the first object you place inside the cluster.

You can also create a cluster constant on the block diagram by choosing Cluster Constant from the Cluster palette.

- This gives you an empty cluster shell.
- You can size the cluster when you drop it.
- Put other constants inside the shell.

**Note:** You cannot place terminals for front panel objects in a cluster constant on the block diagram, nor can you place “special” constants like the Tab or Empty String constant within a block diagram cluster shell.
Cluster Functions

- In the Cluster & Variant subpalette of the Programming palette
- Can also be accessed by right-clicking the cluster terminal

The terms Bundle and Cluster are closely related in LabVIEW.

Example: You use a Bundle Function to create a Cluster. You use an Unbundle function to extract the parts of a cluster.

**Bundle** function—Forms a cluster containing the given objects (explain the example).

**Bundle by Name** function—Updates specific cluster object values (the object must have an owned label).

**Note:** You must have an existing cluster wired into the middle terminal of the function to use Bundle By Name.
Using Arrays and Clusters with Graphs

The Waveform Datatype contains 3 pieces of data:
• $t_0$ = Start Time
• $dt$ = Time between Samples
• $Y$ = Array of $Y$ magnitudes

Two ways to create a Waveform Cluster:

- **Build Waveform (absolute time)**: Builds a waveform or modifies an existing waveform with the start time represented as an absolute TimeStamp. Time Stamps are accurate to real-world time & date and are very useful for real-world data recording.
- **Cluster (relative time)**: Builds a waveform or modifies an existing waveform with a relative time stamp. The input to $t_0$ is a DBL. Building waveforms using the bundle allows data to be plotted on the negative X (time) axis.

The waveform data type carries the data, start time, and $\Delta t$ of a waveform. You can create waveforms using the Build Waveform function. Many of the VIs and functions you use to acquire or analyze waveforms accept and return the waveform data type by default. When you wire a waveform data type to a waveform graph or chart, the graph or chart automatically plots a waveform based on the data, start time, and $\Delta t$ of the waveform. When you wire an array of waveform data types to a waveform graph or chart, the graph or chart automatically plots all the waveforms.
Shift register – Access Previous Loop Data

- Available at left or right border of loop structures
- Right-click the border and select **Add Shift Register**
- Right terminal stores data on completion of iteration
- Left terminal provides stored data at beginning of next iteration

**Shift registers** transfer data from one iteration to the next:
- Right-click on the left or right side of a For Loop or a While Loop and select Add Shift Register.
- The right terminal stores data at the end of an iteration. Data appears at the left terminal at the start of the next iteration.
- A shift register adapts to any data type wired into it.

An input of 0 would result in an output of 5 the first iteration, 10 the second iteration and 15 the third iteration. Said another way, shift registers are used to retain values from one iteration to the next. They are valuable for many applications that have memory or feedback between states. The feedback node is another representation of the same concept. (pictured below) Both programs pictured behave the same.

See Help > Search the LabVIEW Help… for more information.
Local Variables

- Local Variables allow data to be passed between parallel loops.
- A single control or indicator can be read or written to from more than one location in the program
  - Local Variables break the dataflow paradigm and should be used sparingly

Sometimes you may need to access a front panel object from more than one place on the block diagram or to pass data between structures that cannot be connected by a wire. To accomplish these tasks, you would use a local variable.

Local variables are located in the Structures subpalette of the Functions palette.

When you place a local variable on the diagram, it contains by default the name (owned label) of the first object you placed on the front panel.

You use a local variable by first selecting the object you want to access. You can either click on the local variable with the Operating tool and select the object (by owned label) you want to access, or pop up on the local variable and choose the object from the Select Item menu.

Next, you must decide to either read or write to the object. Right click on the local variable and choose Change To Read or Change to Write.
LabVIEW Navigation Window

- Shows the current region of view compared to entire Front Panel or Block Diagram
- Great for large programs

* Organize and reduce program visual size with subVIs

Select **View»Show Navigation Window** to display this window.

Use the window to navigate large front panels or block diagrams. Click an area of the image in the **Navigation Window** to display that area in the front panel or block diagram window. You also can click and drag the image in the **Navigation Window** to scroll through the front panel or block diagram.
LabVIEW Project

- Group and organize VIs
- Hardware and I/O management
- Manage VIs for multiple targets
- Build libraries and executables
- Manage large LabVIEW applications
- Enable version tracking and management

(LabVIEW»Project»New)

LabVIEW Project

Use projects to group together LabVIEW files and non-LabVIEW files, create build specifications, and deploy or download files to targets. A target is a device or machine on which a VI runs. When you save a project, LabVIEW creates a project file (.lvproj), which includes configuration information, build information, deployment information, references to files in the project, and so on.

You must use a project to build stand-alone applications and shared libraries. You also must use a project to work with an RT, FPGA, or PDA target. Refer to the specific module documentation for more information about using projects with the LabVIEW Real-Time, FPGA, and PDA Modules.

Project-style LabVIEW Plug and Play instrument drivers use the project and project library features in LabVIEW 8.0. You can use project-style drivers in the same way as previous LabVIEW Plug and Play drivers.

Project Explorer Window

Use the Project Explorer window to create and edit projects. Select File»New Project to display the Project Explorer window. You also can select Project»New Project or select File»New and then select Empty Project in the New dialog box to display the Project Explorer window.
Additional Resources

- NI Academic Web & Student Corner
  - [http://www.ni.com/academic](http://www.ni.com/academic)

- Connexions: Full LabVIEW Training Course
  - [www.cnx.rice.edu](http://www.cnx.rice.edu)
  - Or search for “LabVIEW basics”

- LabVIEW Certification
  - LabVIEW Fundamentals Exam (free on [www.ni.com/academic](http://www.ni.com/academic))
  - Certified LabVIEW Associate Developer Exam (industry recognized certification)

- Get your own copy of LabVIEW Student Edition
  - [www.ni.com/academic](http://www.ni.com/academic)

Published by Robert H Bishop.
Prentice Hall. 

© National Instruments Corp. Introduction to LabVIEW
Exercise 1 – Setting Up Your Device

In this exercise, you will use Windows utilities to verify your sound card and prepare it for use with a microphone.

1. Prepare your microphone for use. Double-click the volume control icon on the task bar to open up the configuration window. The sound configuration window can also be found from the Windows Control Panel: Start Menu » Control Panel » Sounds and Audio Devices » Advanced.

2. If you do not see a microphone section, go to Options » Properties » Recording and place a checkmark in the box next to Microphone. This will display the Microphone volume control. Click “OK”.

3. Uncheck the Mute box if it is not already unchecked. Make sure that the volume is turned up.

4. Close the volume control configuration window.

5. Open the Sound Recorder by selecting Start » Programs » Accessories » Entertainment » Sound Recorder.

6. Click the record button and speak into your microphone. Notice how the sound signal is displayed in the Sound Recorder.

7. Click stop and close the Sound Recorder without saving changes when you are finished.
Exercise 2 – Acquiring a Signal with the Sound Card

- Use LabVIEW to:
  - Acquire a signal from your sound card

This exercise should take 15 minutes.
Exercise 2 – Acquiring a Signal with the Sound Card

Complete the following steps to create a VI that acquires data from your sound card.

1. Launch LabVIEW.
2. In the Getting Started window, click the Blank VI link.
3. Display the block diagram by pressing <Ctrl+E> or selecting Window»Show Block Diagram.
4. Place the Acquire Sound Express VI on the block diagram. Right-click to open the functions palette and select Express»Input»Acquire Sound. Place the Express VI on the block diagram.
5. In the configuration window under #Channels, select 1 from the drop-down list and click “OK”.
6. Place the Filter Express VI to the right of the Acquire Signal VI on the block diagram. From the functions palette, select Express»Signal Analysis»Filter and place it on the block diagram. In the configuration window under Filtering Type, choose “Highpass.” Under Cutoff Frequency, use a value of 300 Hz. Click “OK.”
7. Make the following connections on the block diagram by hovering your mouse over the terminal so that it becomes the wiring tool and clicking once on each of the terminals you wish to connect:
   a. Connect the “Data” output terminal of the Acquire Signal VI to the “Signal” input of the Filter VI.
   b. Create a graph indicator for the filtered signal by right-clicking on the “Filtered Signal” output terminal and choose Create»Graph Indicator.
8. Return to the front panel by pressing <Ctrl+E> or Window»Show Front Panel.
9. Run your program by clicking the run button. Hum or whistle into your microphone and observe the data you acquire from your sound card.
10. Save the VI as “Exercise 2 – Acquire.vi” in the Exercises folder.
11. Close the VI.
Exercise 3.1 – Analysis

• Use LabVIEW Express VIs to:

  – Simulate a signal and display its amplitude and frequency

This exercise should take 15 minutes.
Exercise 3.1 – Analysis

Create a VI that produces a sine wave with a specified frequency and displays the data on a Waveform Chart until stopped by the user.

1. Open a blank VI from the Getting Started screen.
2. Place a chart on the front panel. Right-click to open the controls palette and select Controls»Modern»Graph»Waveform Chart.
3. Place a dial control on the front panel. From the controls palette, select Controls»Modern »Numeric»Dial. Notice that when you first place the control on the front panel, the label text is highlighted. While this text is highlighted, type “Frequency In” to give a name to this control.
4. Go to the block diagram (<Ctrl+E>) and place a while loop down. Right-click to open the functions palette and select Express»Execution Control»While Loop. Click and drag on the block diagram to make the while loop the correct size. Select the waveform chart and dial and drag them inside the while loop if they are not already. Notice that a stop button is already connected to the conditional terminal of the while loop.

5. Place the Simulate Signal Express VI on the block diagram. From the functions palette, select Express»Signal Analysis»Simulate Signal and place it on the block diagram inside the while loop. In the configuration window under Timing, choose “Simulate acquisition timing.” Click “OK.”

6. Place a Tone Measurements Express VI on the block diagram (Express»Signal Analysis»Tone Measurements). In the configuration window, choose Amplitude and Frequency measurements in the Single Tone Measurements section. Click “OK.”
7. Make the following connections on the block diagram by hovering your mouse over the terminal so that it becomes the wiring tool and clicking once on each of the terminals you wish to connect:
   a. Connect the “Sine” output terminal of the Simulate Signal VI to the “Signals” input of the Tone Measurements VI.
   b. Connect the “Sine” output to the Waveform Chart.
   c. Create indicators for the amplitude and frequency measurements by right-clicking on each of the terminals of the Tone Measurements Express VI and selecting Create » Numeric Indicator.
   d. Connect the “Frequency In” control to the “Frequency” terminal of the Simulate Signal VI.

8. Return to the front panel and run the VI. Move the “Frequency In” dial and observe the frequency of the signal. Click the stop button once you are finished.

9. Save the VI as “Exercise 3.1 – Simulated.vi”.

10. Close the VI.

Notes

• When you bring up the functions palette, press the small push pin in the upper left hand corner of the palette. This will tack down the palette so that it doesn’t disappear. This step will be omitted in the following exercises, but should be repeated.
Exercise 3.2 – Analysis

• Use LabVIEW Express VIs to:
  – Acquire a signal and display its amplitude and frequency

This exercise should take 15 minutes.
Exercise 3.2 – Analysis

Create a VI that measures the frequency and amplitude of the signal from your sound card and displays the acquired signal on a waveform chart. The instructions are the same as in Exercise 3.1, but the Sound Signal VI is used in place of the Simulate Signal VI. Try to do this without following the instructions!

1. Open a blank VI.
2. Go to the block diagram and place a While Loop down (Express»Execution Control»While Loop).
3. Place the Acquire Sound Express VI on the block diagram (Express»Input»Acquire Sound).
4. Place a Filter Express VI on the block diagram. In the configuration window choose a highpass filter and a cutoff frequency of 300 Hz.
5. Place a Tone Measurements Express VI on the block diagram (Express»Signal Analysis»Tone). In the configuration window, choose Amplitude and Frequency measurements in the Single Tone Measurements section.
6. Create indicators for the amplitude and frequency measurements by right-clicking on each of the terminals of the Tone Measurements Express VI and selecting Create»Numeric Indicator.
7. Connect the “Data” terminal of the Acquire Sound Express VI to the “Signal” input of the Filter VI.
8. Connect the “Filtered Signal” terminal of the Filter VI to the “Signals” input of the Tone Measurements VI.
9. Create a graph indicator for the Filtered Signal by right-clicking on the “Filtered Signal” terminal and selecting Create»Graph Indicator.
10. Return to the front panel and run the VI. Observe the signal from your sound card and its amplitude and frequency. Hum or whistle into the microphone and observe the amplitude and frequency you are producing.
11. Save the VI as “Exercise 3.2-Data.vi”. Close the VI.
Exercise 3.3 – Decision Making and Saving Data

• Use a case structure to:
  – Make a VI that saves data when a condition is met

This exercise should take 15 minutes.
Create a VI that allows you to save your data to file if the frequency of your data goes below a user-controlled limit.

1. Open Exercise 3.2 – Data.vi.
2. Go to File > Save As... and save it as “Exercise 3.3 – Decision Making and Saving Data”. In the “Save As” dialog box, make sure substitute copy for original is selected and click “Continue...”.
3. Add a case structure to the block diagram inside the while loop (Functions > Programming > Structures > Case Structure).
4. Inside the “true” case of the case structure, add a Write to Measurement File Express VI (Functions > Programming > File I/O > Write to Measurement File).
   a. In the configuration window that opens, choose “Save to series of files (multiple files).” Note the default location your file will be saved to and change it if you wish.
   b. Click “Settings...” and choose “Use next available file name” under the Existing Files heading.
   c. Under File Termination choose to start a new file after 10 segments. Click “OK” twice.
5. Add code so that if the frequency computed from the Tone Measurements Express VI goes below a user-controlled limit, the data will be saved to file. Hint: Go to Functions > Programming > Comparison > Less?
6. Remember to connect your data from the DAQ Assistant or Acquire Sound Express VI to the “Signals” input of the Write to Measurement File VI. If you need help, refer to the solution to this exercise.
7. Go to the front panel and run your VI. Vary your frequency limit and then stop the VI.
8. Navigate to My Documents > LabVIEW Data and open one of the files that was saved there. Examine the file structure and check to verify that 10 segments are in the file.
9. Save your VI and close it.
Exercise 4.1 – Manual Analysis

- Use the cursor legend on a graph to:
  - Verify your frequency and amplitude measurements

This exercise should take 15 minutes.
Exercise 4.1 – Manual Analysis

Create a VI that displays simulated data on a waveform graph and measures the frequency and amplitude of that data. Use cursors on the graph to verify the frequency and amplitude measurements.

1. Open Exercise 3.1 – Simulated.vi.
2. Save the VI as “Exercise 4.1 – Manual Analysis.vi”.
3. Go to the block diagram and remove the While Loop. Right-click the edge of the loop and choose Remove While Loop so that the code inside the loop does not get deleted.
4. Delete the stop button.
5. On the front panel, replace the waveform chart with a waveform graph. Right-click the chart and select Replace»Modern»Graph»Waveform Graph.
6. Make the cursor legend viewable on the graph. Right-click on the graph and select Visible Items»Cursor Legend.
7. Change the maximum value of the “Frequency In” dial to 100. Double-click on the maximum value and type “100” once the text is highlighted.
8. Set a default value for the “Frequency In” dial by setting the dial to the value you would like, right-clicking the dial, and selecting Data Operations»Make Current Value Default.
9. Run the VI and observe the signal on the waveform graph. If you cannot see the signal, you may need to turn on auto-scaling for the x-axis. Right-click on the graph and select X Scale»AutoScale X.
10. Change the frequency of the signal so you can see a few periods on the graph.
11. Manually measure the frequency and amplitude of the signal on the graph using cursors. To make the cursors display on the graph, click on one of the three buttons in the cursor legend. Once the cursors are displayed, you can drag them around on the graph and their coordinates will be displayed in the cursor legend.

12. Remember that the frequency of a signal is the reciprocal of its period \( f = \frac{1}{T} \). Does your measurement match the frequency and amplitude indicators from the Tone Measurements VI?
13. Save your VI and close it.
Exercise 4.2 – Using MathScript
Use the MathScript Node and Interactive Window to process the acquired signal (logarithmic decay) in the MathScript and save the script.

This exercise should take 25 minutes.
Exercise 4.2 – MathScript

Create a VI that uses the MathScript Node to alter your simulated signal and graph it. Use the Interactive MathScript Window to view and alter the data and then load the script you have created back into the MathScript Node.

2. Save the VI as “Exercise 4.2 – MathScript.vi”.
3. Go to the block diagram and delete the wire connecting the Simulate Signal VI to the Waveform Graph.
5. Right-click on the left border of the MathScript Node and select Add Input. Name this input “In” by typing while the input node is highlighted black.
6. Right-click on the right border of the MathScript Node and select Add Output. Name this output “Out”.
7. Convert the Dynamic Data Type output of the Simulate Signals VI to a 1D Array of Scalars to input to the MathScript Node. Place a Convert from Dynamic Data Express VI on the block diagram (Express Signal Manipulation Convert from Dynamic Data). By default, the VI is configured correctly so click “OK” in the configuration window.
8. Wire the “Sine” output of the Simulate Signal VI to the “Dynamic Data” input of the Convert from Dynamic Data VI.
9. Wire the “Array” output of the Convert from Dynamic Data VI to the “In” node on the MathScript Node.
10. In order to use the data from the Simulate Signal VI in the Interactive MathScript Window it is necessary to declare the input variable as a global variable. Inside the MathScript Node type “global In;”.
11. Return to the front panel and increase the frequency to be between 50 and 100. Run the VI.
12. Open the Interactive MathScript Window (Tools MathScript Window…).
13. In the MathScript Window, the Command Window can be used to enter in the command that you wish to compute. In the Command Window, type “global In” and press “Enter”. This will allow you to see the data passed to the variable “In” on the MathScript Node.
14. Notice that all declared variables in the script along with their dimensions and type are listed 
on the “Variables” tab. To display the graphed data, click once on the variable In and change 
the drop down menu from “Numeric” to “Graph”.

![LabVIEW interface](image1.png)

15. Use the graph palette to zoom in on your data.

![Graph zoom](image2.png)

16. Right-click on “Cursor 1” and choose **Bring to Center**. What does this do?

17. Drag the cursor around. The cursor will not move if the zoom option is selected.

18. Right-click on the graph and choose **Undock Window**. What does this do? Close this new 
window when you are finished.

© National Instruments Corp. Introduction to LabVIEW 69
19. Multiply the data by a decreasing exponential function. Follow these steps:
   a. Make a 100 element array of data that constitutes a ramp function going from 0.01 to 5 by typing “Array = [0.01:0.05:5];” in the Command Window and pressing Enter. What type of variable is “Array”? 
   b. Make an array containing a decreasing exponential. Type “Exp = 5*exp(-Array);” and press Enter.
   c. Now multiply the Exp and In arrays element by element by typing “Out = In.*Exp;” and pressing Enter.
   d. Look at the graph of the variable “Out”.

20. Go to the History tab and use Ctrl-click to choose the 4 commands you just entered. Copy those commands using <Ctrl-C>.

21. On the Script tab, paste the commands into the Script Editor using <Ctrl-V>.

22. Save your script by clicking “Save” at the bottom of the window. Save it as “myscript.txt”


24. Return to the block diagram of Exercise 4.2 – MathScript. Load the script you just made by right-clicking on the MathScript Node border and selecting Import… Navigate to myscript.txt, select it, and click “OK”.

25. Right-click on the variable “Out” and select Choose Data Type»1D-Array»DBL 1D. Output data types must be set manually on the MathScript Node.

26. Wire “Out” to the Waveform Graph.

27. Return to the front panel and run the VI. Does the data look like you expect?

25. Save and close your VI.
Exercise 5 – Apply What You Have Learned

This exercise should take 20 minutes.
Exercise 5 – Apply What You Have Learned

In this exercise, you will create a VI that uses what you have learned. Design a VI that does the following:

1. Acquire data from your device (your sound card) and graph it.
2. Filter that data using the Filter Express VI (Functions»Express»Signal Analysis»Filter). There should be a front panel control for a user configurable cut-off frequency.

3. Take a Fast Fourier Transform to get the frequency information from the filtered data and graph the result. Use the Spectral Measurements Express VI (Functions»Express»Signal Analysis»Spectral).

4. Find the dominant frequency of the filtered data using the Tone Measurements Express VI.

5. Compare that frequency to a user inputted limit. If the frequency is over that limit, light up an LED.

(End of Exercise)