Introduction to LabVIEW

GRAPHICAL PROGRAMMING
FOR ENGINEERS AND SCIENTISTS

ni.com
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.

http://www.ni.com
LabVIEW Programs: Virtual Instruments (VIs)

Front Panel
  • Controls = Inputs
  • Indicators = Outputs

Block Diagram
  • Accompanying “program” for front panel
  • Components “wired” together
LabVIEW programming style

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**.
Express VIs, VIs and Functions

- **Express VIs**: interactive VIs with configurable dialog page
- **Standard VIs**: modularized VIs customized by wiring
- **Functions**: fundamental operating elements of LabVIEW; no front panel or block diagram
Labview Functions

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
Controls and Functions Palettes

Controls Palette
(Front Panel Window)

Functions Palette
(Block Diagram Window)
Status Toolbar

Run Button
Continuous Run Button
Abort Execution
Pause/Continue Button
Text Settings
Align Objects
Distribute Objects
Reorder
Resize front panel objects

Additional Buttons on the Diagram Toolbar
Execution Highlighting Button
Step Into Button
Step Over Button
Step Out Button
Help Options

Context Help
- Online help
- Lock help
- Simple/Complex Diagram help
- Ctrl + H

Online reference
- All menus online
- Pop up on functions in diagram to access online info directly
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.
Creating a VI

Front Panel Window

Block Diagram Window

Control Terminals

Indicator Terminals
Creating a VI – Block Diagram
Dataflow Programming

• Block diagram executes 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
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.

• Probe

  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
SubVIs

- What is a subVI?
- Making an icon and connector for a subVI
- Using a VI as a subVI
SubVIs

• A SubVI is a VI that can be used within another VI
• Similar to a subroutine
• Advantages
  – Modular
  – Easier to debug
  – Don’t have to recreate code
  – Require less memory
Icon and Connector

- An icon represents a VI in other block diagrams
- A connector shows available terminals for data transfer
Steps to Create a SubVI

- Create the Icon
- Create the Connector
- Assign Terminals
- Save the VI
- Insert the VI into a Top Level VI
Create the Icon

• Right-click on the icon in the block diagram or front panel
Create the Connector

Right click on the icon pane (front panel only)
Assign Terminals
Save The VI

• Choose an Easy to Remember Location

• Organize by Functionality
  – Save Similar VIs into one directory (e.g. Math Utilities)

• Organize by Application
  – Save all VIs Used for a Specific Application into one directory or library file (e.g. Lab 1 – Frequency Response)
    • Library Files (.llbs) combine many VI’s into a single file, ideal for transferring entire applications across computers
Insert the SubVI into a Top Level VI

Accessing user-made subVIs
Functions >> All Functions >> Select a VI
Or
Drag icon onto target diagram
Loops and Charts

- For Loop
- While Loop
- Charts
- Multiplots
Loops

• While Loops
  – Have Iteration Terminal
  – Always Run at least Once
  – Run According to Conditional Terminal

• For Loops
  – Have Iteration Terminal
  – Run According to input N of Count Terminal
Loops

1. Select the loop
2. Enclose code to be repeated
3. Drop or drag additional nodes and then wire
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.
Case Structures

• In the Structures subpalette of Functions palette
• Enclose nodes or drag them inside the structure
• Stacked like a deck of cards, only one case visible

Functions >> Execution control
Case Structures
State machine with Labview

State Machines

- While Loop
- Case Structure
- Shift Register
State Machines Transitions

- Several programming techniques exist for transitioning from state to state in LabVIEW using State Machines.
- Default transition implies that after one state, another state always follows.
- Transitions between two potential states can be handled by a Select Function.
Sequence Structures

• In the **Execution Control** subpalette of Functions palette
• Executes diagrams sequentially
• Right-click to add new frame
Waveform chart – special numeric indicator that can display a history of values

Controls >> Graph Indicators >> Waveform Chart
Wiring Data into Charts

Single Plot Charts

Multiplot Charts
Graphs

- Selected from the Graph palette of Controls menu
Controls>>All Controls>>Graphs

**Waveform Graph** – Plot an array of numbers against their indices

**Express XY Graph** – Plot one array against another

**Digital Waveform Graph** – Plot bits from binary data
Graphs

Right-Click on the Graph and choose Properties to Interactively Customize
Arrays & File I/O

• Build arrays manually
• Have LabVIEW build arrays automatically
• Write to a spreadsheet file
• Read from a spreadsheet file
Adding an Array to the Front Panel

From the **Controls >> All Controls >> Array and Cluster** subpalette, select the **Array**

Drop it on the screen.
Adding an Array

Place data object into shell (i.e. Numeric Control)
Building an Array

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
Creating an Array with a Loop

• Loops accumulate arrays at their boundaries
Array Functions – Basics

Functions >> All functions >> Array
Array Functions – Build Array
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)

Writing to LVM file

Reading from LVM file
Write LabVIEW Measurement File

- Includes the open, write, close and error handling functions
- Handles formatting the string with either a tab or comma delimiter
- Merge Signals function is used to combine data into the dynamic data type
Strings

- A string is a sequence of displayable or nondisplayable characters (ASCII)
- Many uses – displaying messages, instrument control, file I/O
- String control/indicator is in the Controls » Text Control or Text Indicator
Clusters

- Data structure that groups data together
- Data may be of different types
- Analogous to \textit{struct} in C
- Elements must be either all controls or all indicators
- Thought of as wires bundled into a cable
Creating a Cluster

1. Select a Cluster shell

2. Place objects inside the shell

Controls >> All Controls >> Array & Cluster
Cluster Functions

- In the **Cluster** subpalette of the **Functions>>All functions** palette
- Can also be accessed by right-clicking on the cluster terminal

(Bundle By Name)

(Terminal labels reflect data type)
Cluster Functions

Unbundle

Unbundled cluster in the diagram

Unbundle By Name

Unbundled cluster in the diagram
Error Clusters

- Error cluster contains the following information:
  - **Boolean** to report whether error occurred
  - **Integer** to report a specific error code
  - **String** to give information about the error
Error Handling Techniques

• Error information is passed from one subVI to the next
• If an error occurs in one subVI, all subsequent subVIs are not executed in the usual manner
• Error Clusters contain all error conditions
• Automatic E
Formula Nodes

- In the Structures subpalette
- Implement complicated equations
- Variables created at border
- Variable names are case sensitive
- Each statement must terminate with a semicolon (;)
- Context Help Window shows available functions

Note: semicolon
Review of Data Types Found in LabVIEW
Printing & Documentation

• Print From File Menu to Printer, HTML, Rich Text File
• Programmatically Print Graphs or Front Panel Images
• Document VIs in VI Properties » Documentation Dialog
• Add Comments Using Free Labels on Front Panel & Block Diagram
Printing

• **File » Print**… Gives Many Printing Options
  – Choose to Print Icon, Front Panel, Block Diagram, VI Hierarchy, Included SubVIs, VI History

• **Print Panel.vi** (Programmatically Prints a Front Panel)
  – **Functions » All Functions » Application Control**

• **Generate & Print Reports** (**Functions » Output » Report**)
Documenting VIs

• VI Properties » Documentation
  – Provide a Description and Help Information for a VI

• VI Properties » Revision History
  – Track Changes Between Versions of a VI

• Individual Controls » Description and Tip…
  – Right Click to Provide Description and Tip Strip

• Use Labeling Tool to Document Front Panels & Block Diagrams
Data Acquisition

- Data acquisition (DAQ) basics
- Connecting Signals
- Simple DAQ application
What Types of Instruments Can Be Controlled?

- GPIB
- Serial
- Modular Instruments
- PXI Modular Instruments
- Image Acquisition
- Motion Control
- USB
- Ethernet
- Parallel Port
- CAN
Serial

- Serial communication transmits one bit at a time over a transmission line
- Usually does not require external hardware
- Four parameters: baud rate, data bits, parity bit, stop bits
Data Acquisition in Labview

Traditional NI-DAQ
Specific VIs for performing:
• Analog Input
• Analog Output
• Digital I/O
• Counter operations

NI-DAQmx
Next generation driver:
• VIs for performing a task
• One set of VIs for all measurement types
Data Acquisition

Temperature Acquisition using the DAQ Assistant
Hardware Connections

BNC-2120  SCB-68

SC-2075  NI-ELVIS

SCB-68
Virtual Instrumentation Applications

- **Design**
  - Signal and Image Processing
  - Embedded System Programming
    - (PC, DSP, FPGA, Microcontroller)
  - Simulation and Prototyping
  - And more...

- **Control**
  - Automatic Controls and Dynamic Systems
  - Mechatronics and Robotics
  - And more...

- **Measurements**
  - Circuits and Electronics
  - Measurements and Instrumentation
  - And more...

A single graphical development platform

[LabVIEW]
Next step:

LABVIEW FPGA