Lab(1)

Faculty of Engineering,
Ain Shams University
MCT-242, Fall 2014

December 2, 2014

NATIONAL INSTRUMENTS
LabVIEW

Ahmed Okasha
okasha1st@gmail.com

LAB(1) Objective:

- Get familiarized with LabVIEW and realize its capabilities and functionalities.
- Understanding Dataflow Programming
- Give general overview on LabVIEW graphical interface.
- Understand front panels, block diagrams, and connectors/icons
- Demonstrate LabVIEW as a programming language using several examples

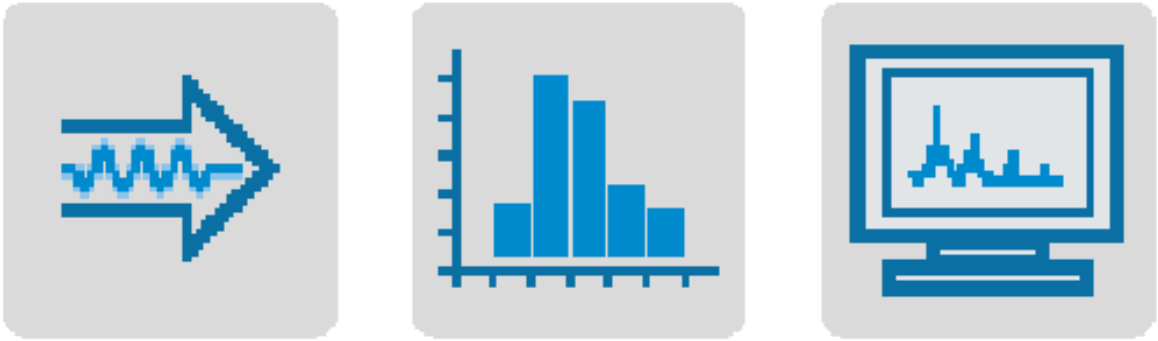
What is LabVIEW ?

- LabVIEW is a Graphical Programming (G-Programming) environment used by millions of engineers and scientists to develop sophisticated measurement, test, and control systems using intuitive graphical icons and wires that resemble a flowchart

LabVIEW, short for **L**aboratory **V**irtual **I**nstrument **E**ngineering **W**orkbench,

LabVIEW Capabilities

- LabVIEW is specifically designed to take measurements, analyze data, and present results to the user. And because it has such a versatile graphical user interface and is so easy to program with, it is also ideal for simulations, presentation of ideas, general programming, or even teaching basic programming concepts.



Acquire, Analyze, and Present

LabVIEW Capabilities (cont.)

- Libraries of signal processing, analysis, and control algorithms
- Libraries of communication, file I/O, and connectivity

In addition to the standard programming language constructs, LabVIEW contains functions for:

- String, array, and waveform manipulation
- Signal processing, including filters, windowing, spectral analysis, and transforms
- Mathematical analysis, including curve fitting, statistics, differential equations, linear algebra, and interpolation
- Report generation, file I/O, and database connectivity

Add-on packages supplement the core functionality for more specialized disciplines, such as:

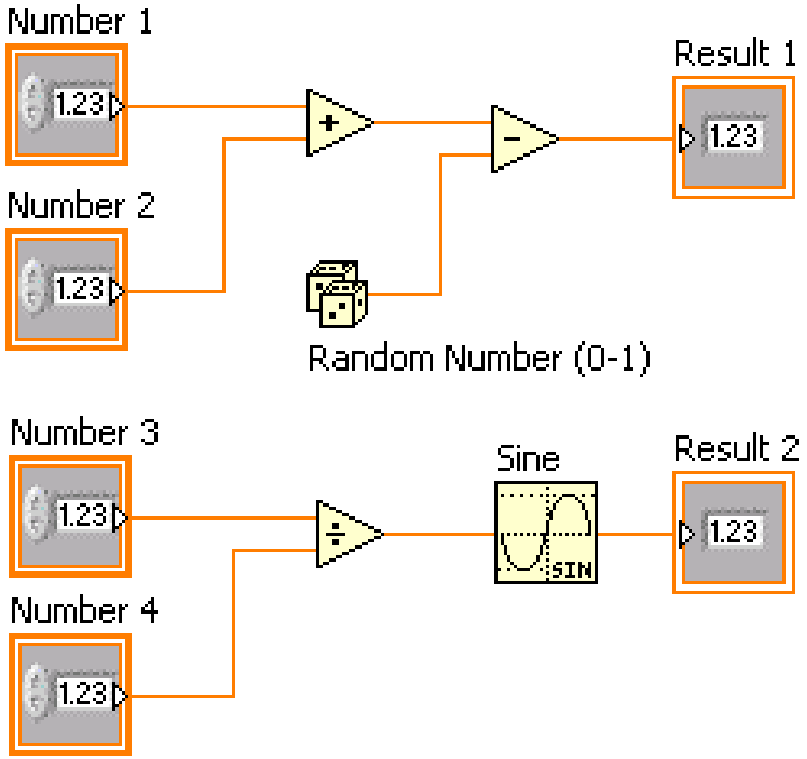
- Control design and simulation
- Sound and vibration analysis
- Machine vision and image processing
- RF and communication

Dataflow Programming

- The LabVIEW program development environment is different from standard C or Java development systems in one important respect: While other programming systems use text-based languages to create lines of code, LabVIEW uses a graphical programming language, often called "G," to create programs in a pictorial form called a block diagram.
- **Dataflow** means that functions execute only after receiving the necessary data.

Dataflow Programming (cont.)

- 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



LabVIEW Terms and Their Conventional Equivalents

LabVIEW	***Conventional Language***
VI	program
subVI	subroutine, subprogram, object
Front panel	user interface
Block diagram	program code
G	C, C++, Java, Pascal, BASIC, etc.
Control	Input
Indicator	Output

Two of The Numerous NI Data Acquisition Devices

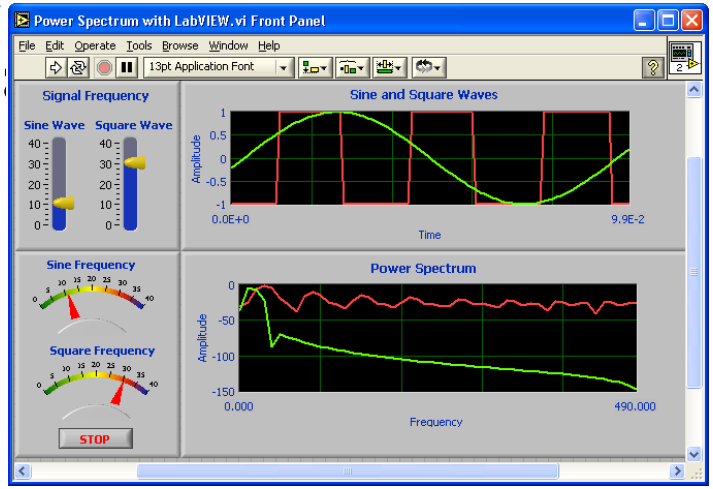
NI 6008

NI 6221

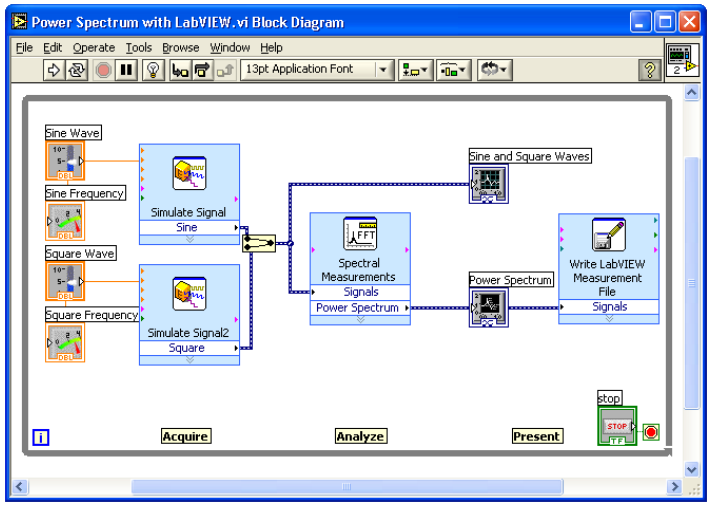
LabVIEW Programs Are Called Virtual Instruments (VIs)

Called VI because their appearance imitate actual physical instruments.

- **Front Panel**
  - **Controls = Inputs**
  - **Indicators = Outputs**

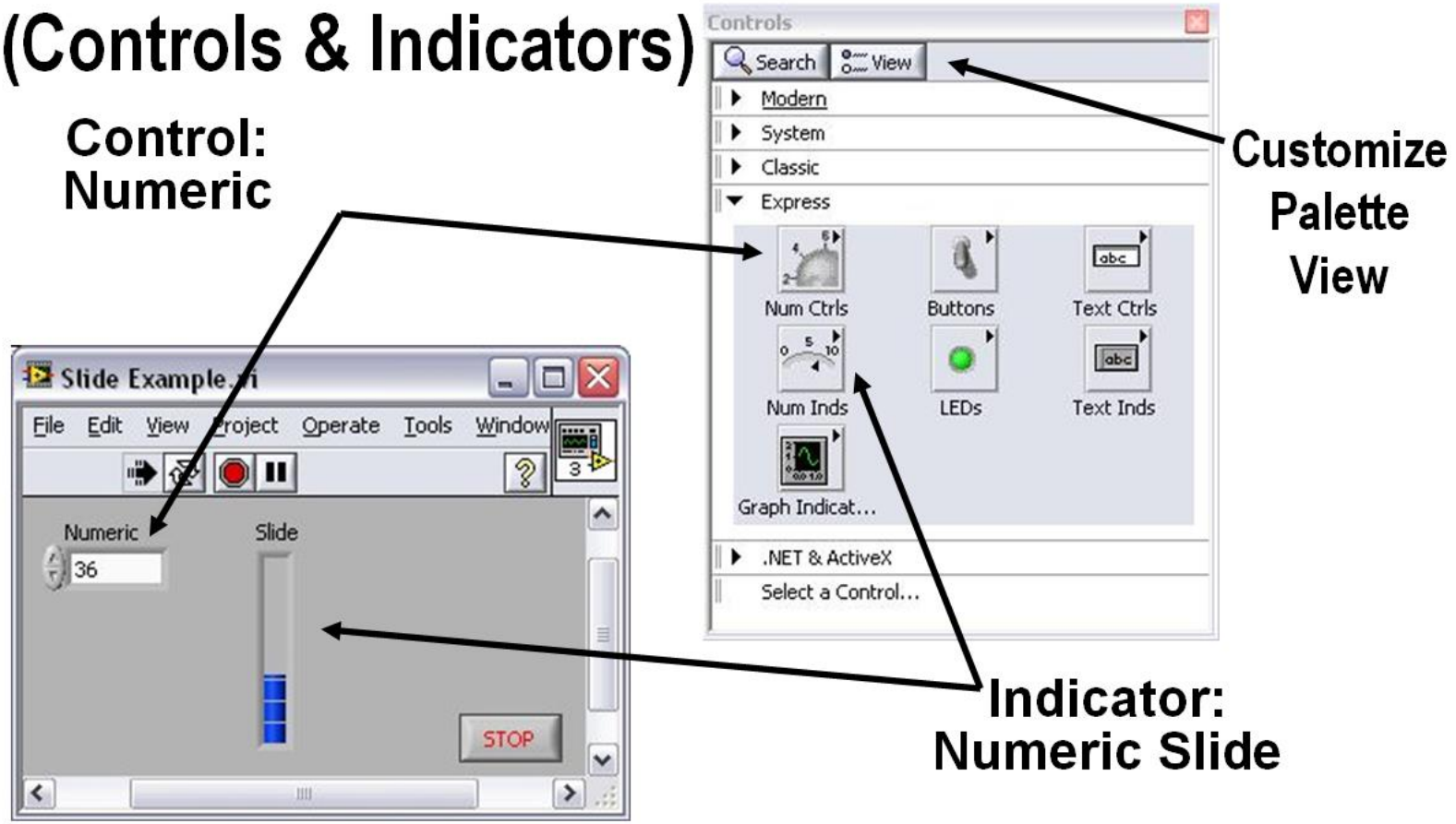


- **Block Diagram**
  - **Accompanying "program" for front panel**
  - **Components "wired" together**



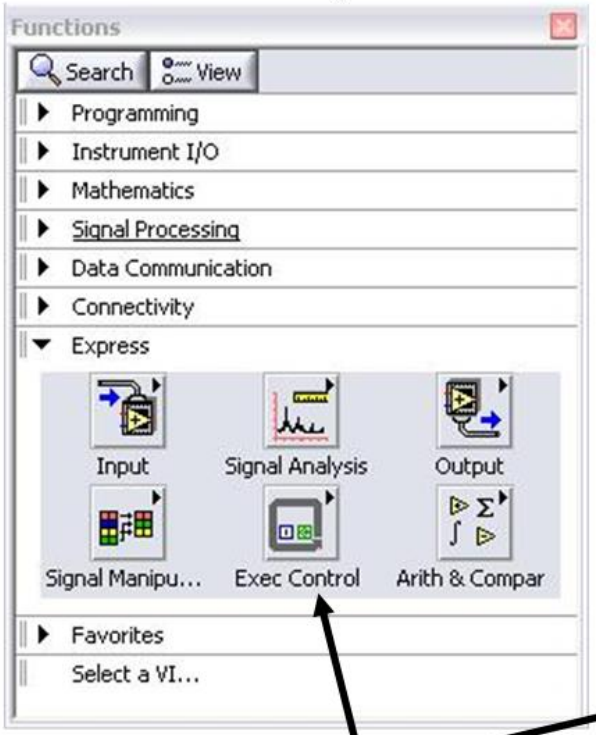
VI Front Panel

Controls Palette (Controls & Indicators) (Place items on the Front Panel Window)

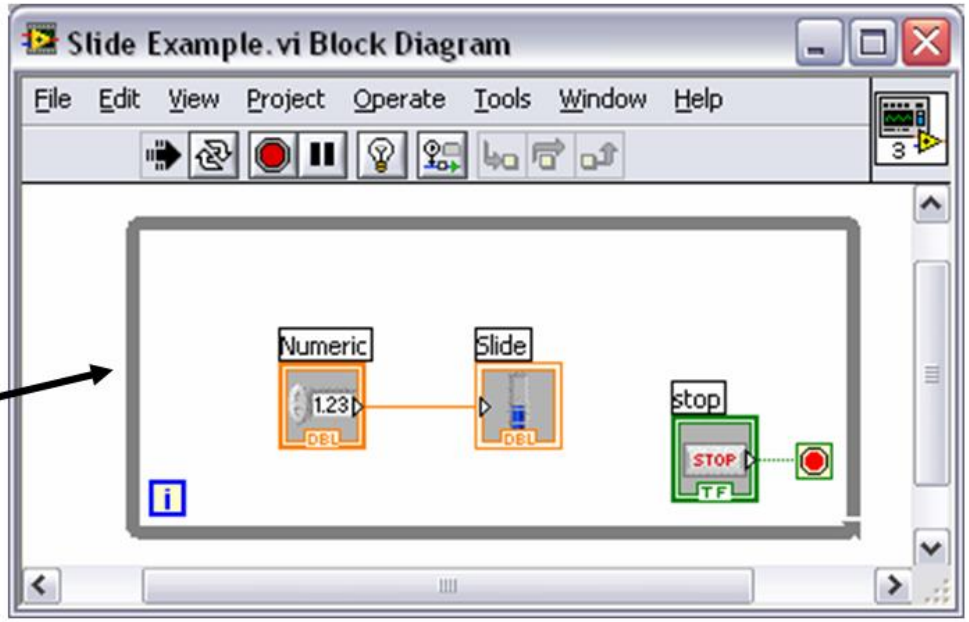


VI Block Diagram

Functions (and Structures) Palette



(Place items on the Block Diagram Window)



Structure: While Loop

Tools Palette

Tools Palette

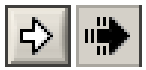
- Recommended: Automatic Selection Tool
- Tools to operate and modify both front panel and block diagram objects

Automatic Selection Tool

Automatically chooses among the following tools:

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

Status Toolbar

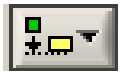

Run Button


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

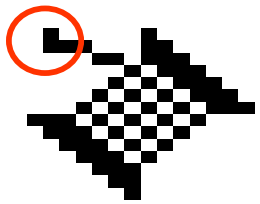

Step Into Button


Step Over Button

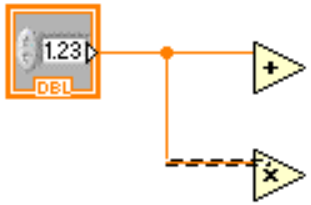

Step Out Button

Wiring Tips – Block Diagram

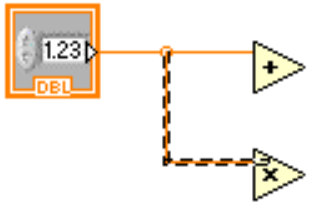
Wiring "Hot Spot"



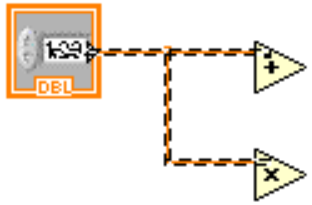
Click To Select Wires



single-click

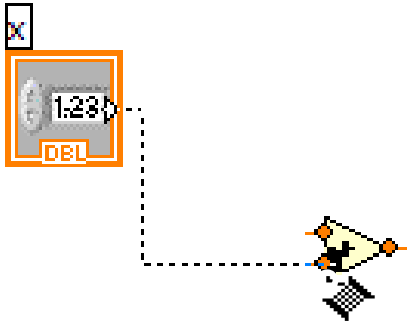


double-click

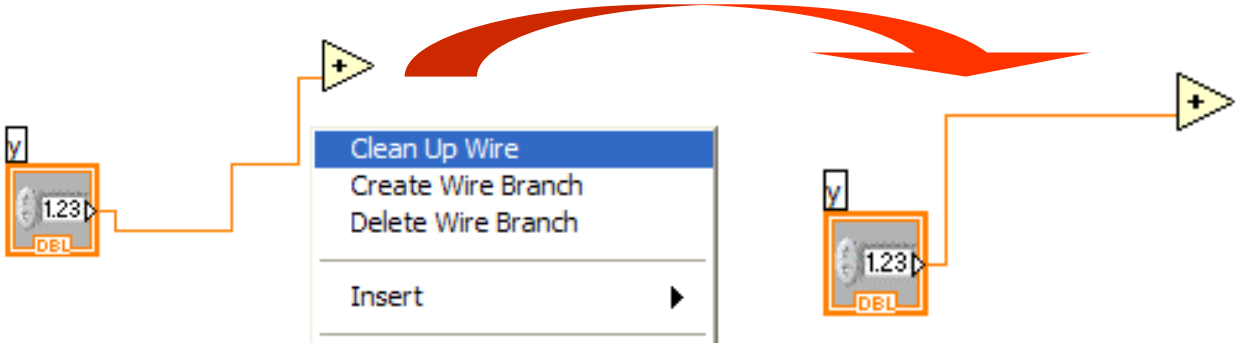


triple-click

Use Automatic Wire Routing



Clean Up Wiring



Debugging Techniques

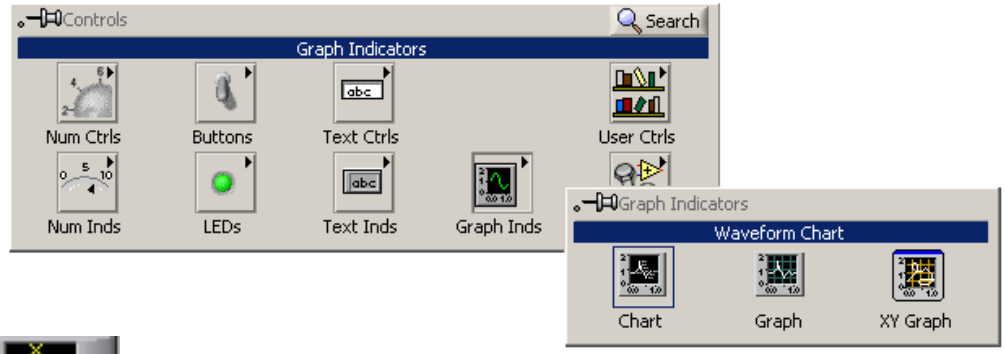
- Finding Errors

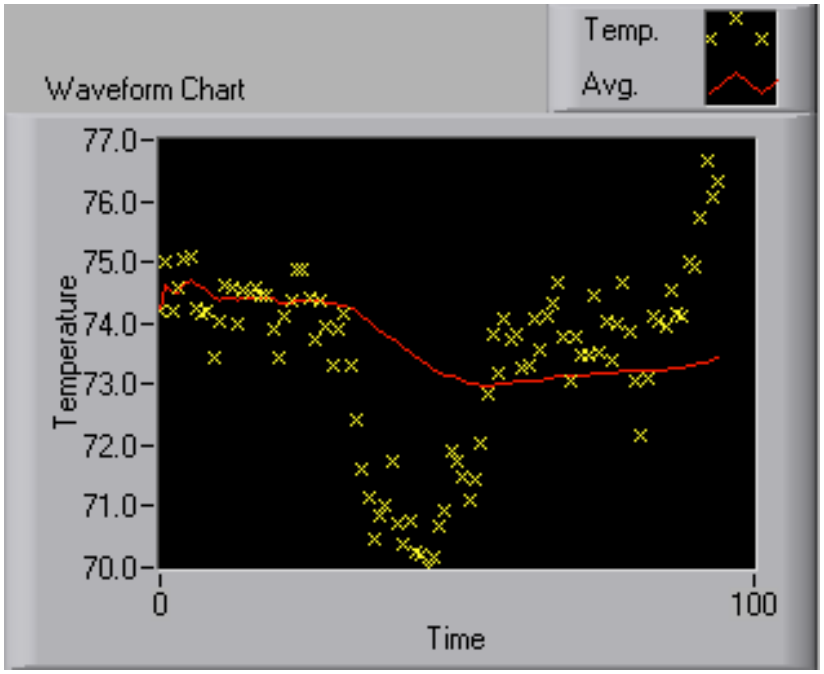
Click on broken Run button
Window showing error appears

- Execution Highlighting

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

Waveform Charts



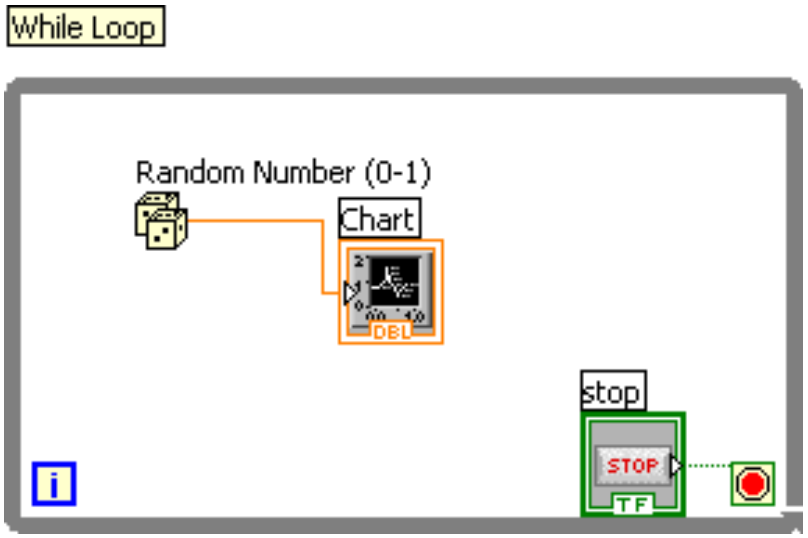


Waveform chart – special numeric indicator that can display a history of values

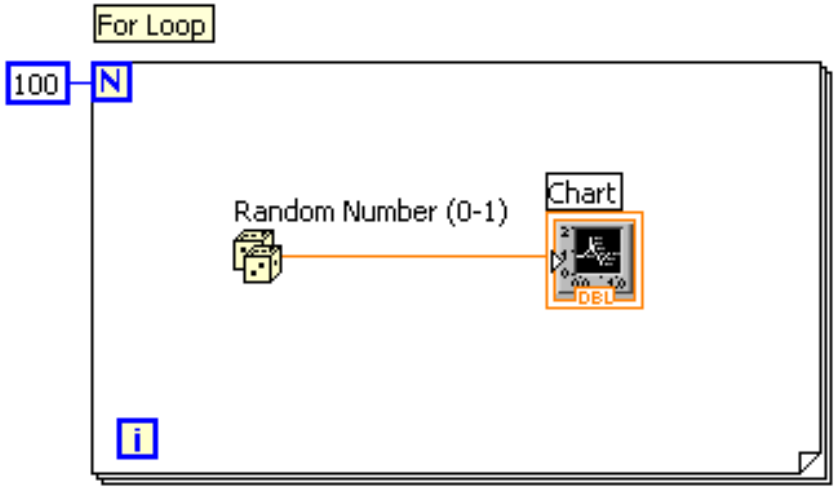
Controls >> Graph Indicators >> Waveform Chart

Loops

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

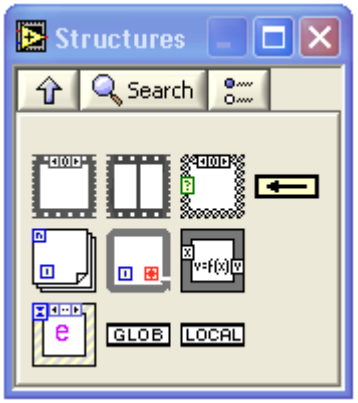


- For Loops
  - Have Iteration Terminal
  - Run According to input N of Count Terminal

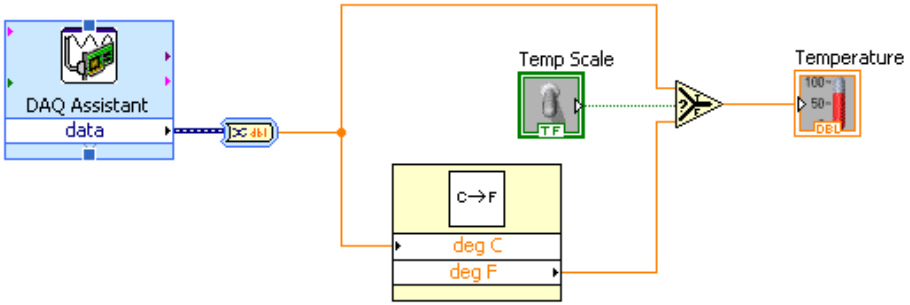


Loops (cont.)

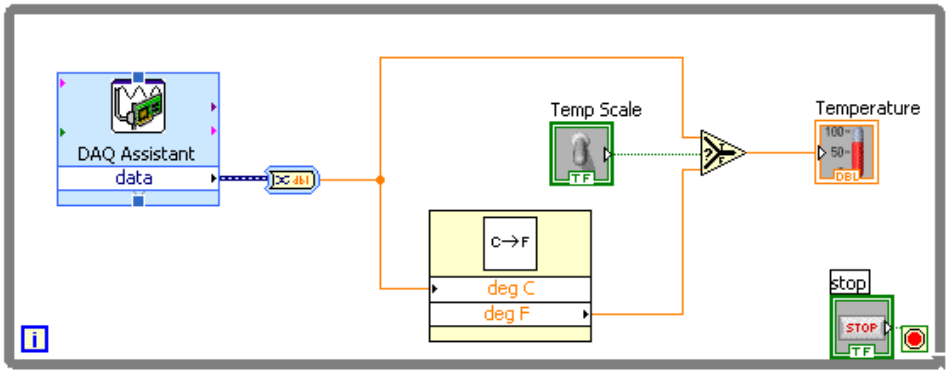
1. Select the loop



2. Enclose code to be repeated



3. Drop or drag additional nodes and then wire

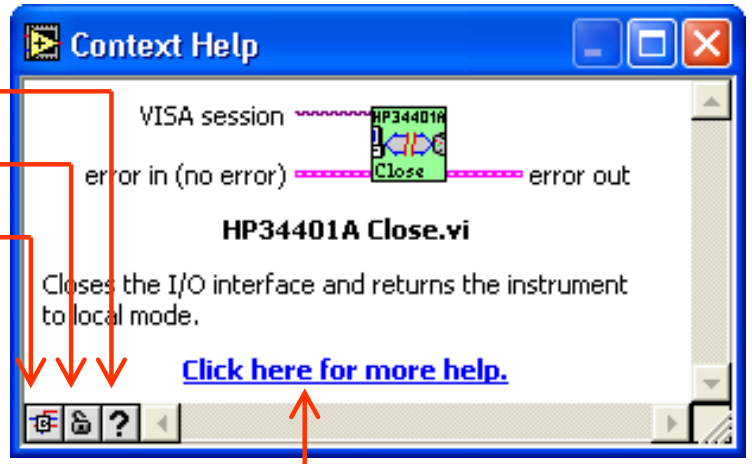


Help Options

To display the **Context Help** window, select **Help»Show Context Help** or press the <Ctrl-H> keys.

Context Help

- **Online help**
- **Lock help**
- **Simple/Complex Diagram help**
- **Ctrl + H**



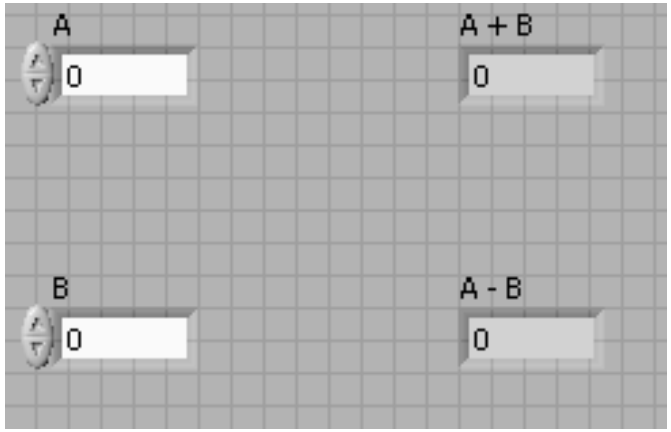
Online reference

- **All menus online**
- **Pop up on functions in diagram to access online info directly**

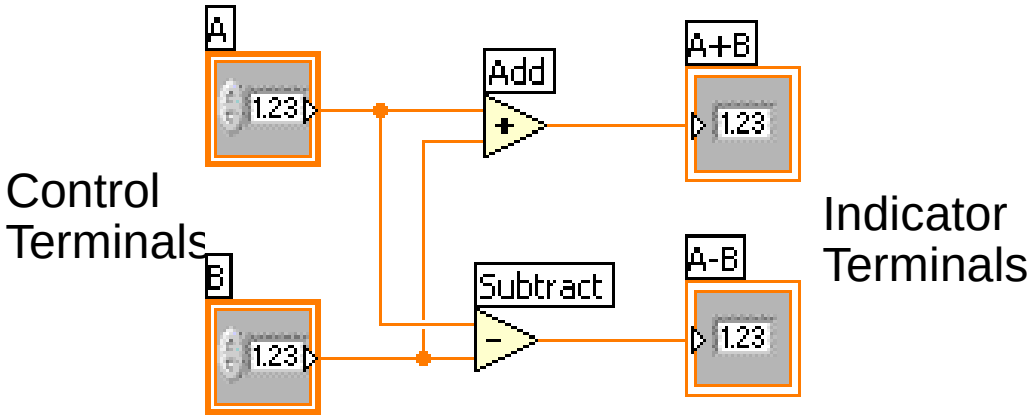
Example (1)

- Design a VI that sums two inputs A and B
- Solution:

Front Panel Window



Block Diagram Window



Example (2)

- Simulate a sine wave in which you can control its amplitude and frequency