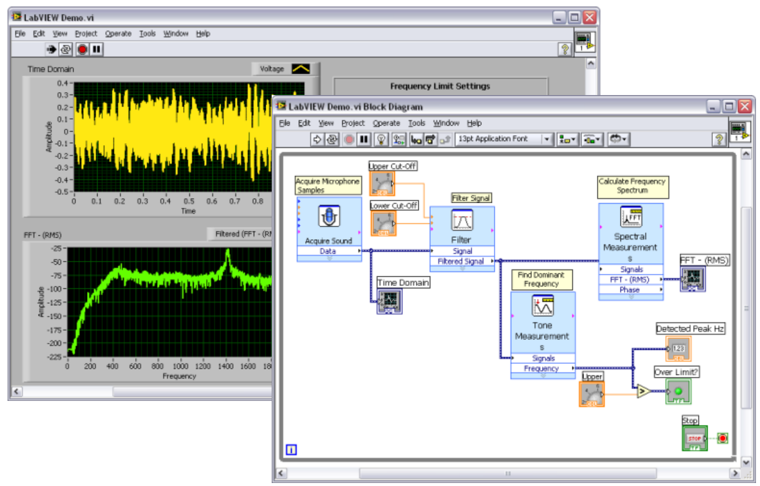
Introduction to LabVIEW
For Use in Embedded System Development



UC Berkeley EE249

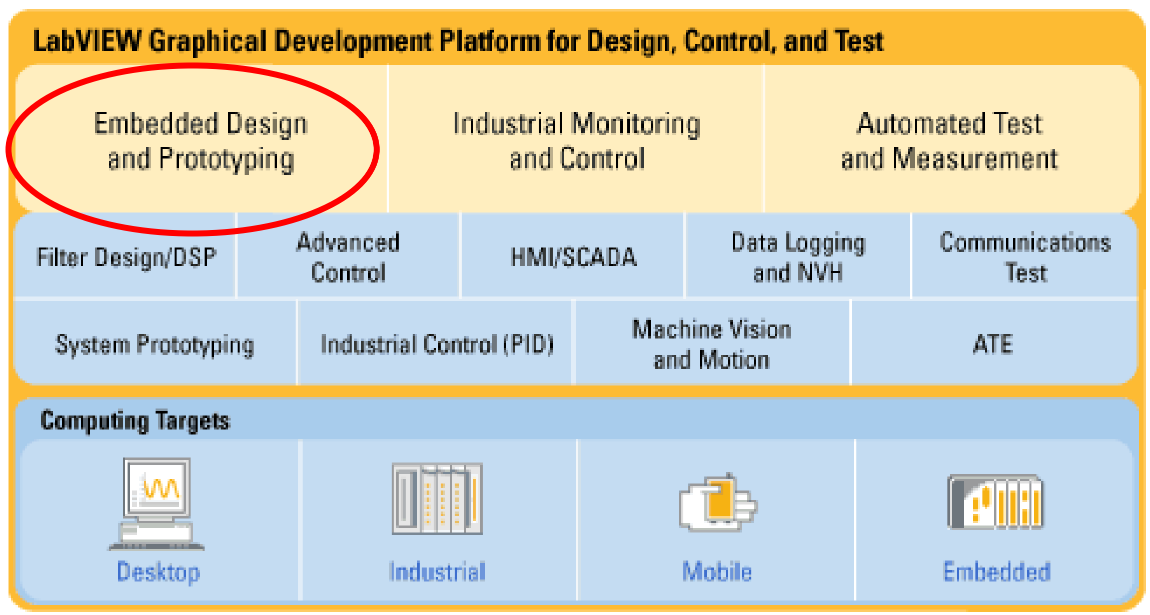
Hugo.Andrade@ni.com

Lab Goals

- Become comfortable with the LabVIEW environment
- Ability to use LabVIEW to solve problems that arise *during the analysis, design, prototype and deployment of Embedded Systems*
- 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
  - Modeling tools
  - Targets and Deployment

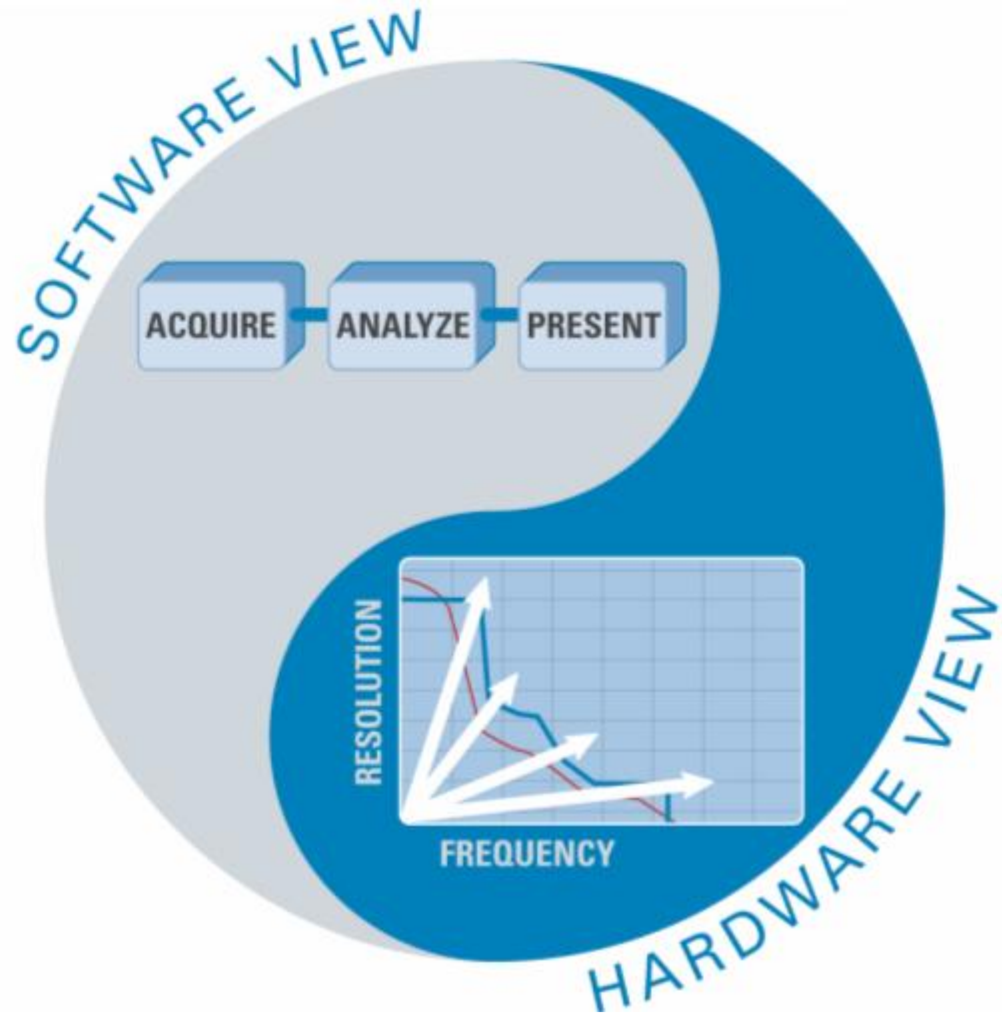
LabVIEW Graphical Development System

- Graphical Programming Environment
- Compile code for multiple OS and devices
- Useful in a broad range of applications



NATIONAL INSTRUMENTS

The Virtual Instrumentation Approach

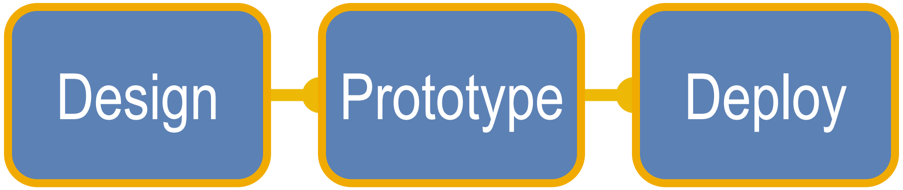


Virtual Instrumentation Applications

- **Analysis and Design**
  - Simulation
  - Signal and Image Processing
  - Embedded System Programming
    - (PC, DSP, FPGA, Microcontroller)
  - Prototyping
  - And more…
- **Control**
  - Automatic Controls and Dynamic Systems
  - Mechatronics and Robotics
  - And more…
- **Measurement/Test**
  - Circuits and Electronics
  - Measurements and Instrumentation

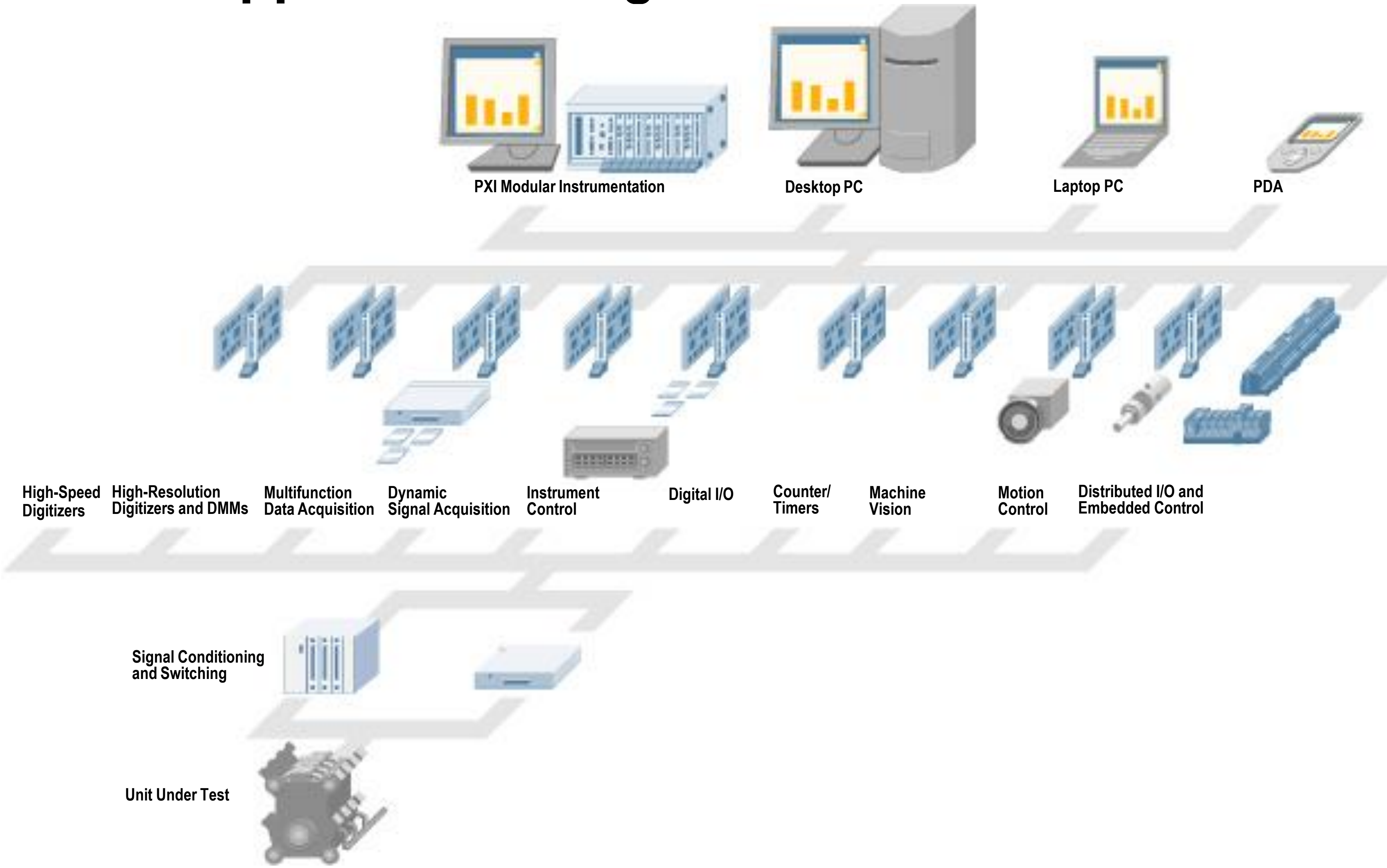
…

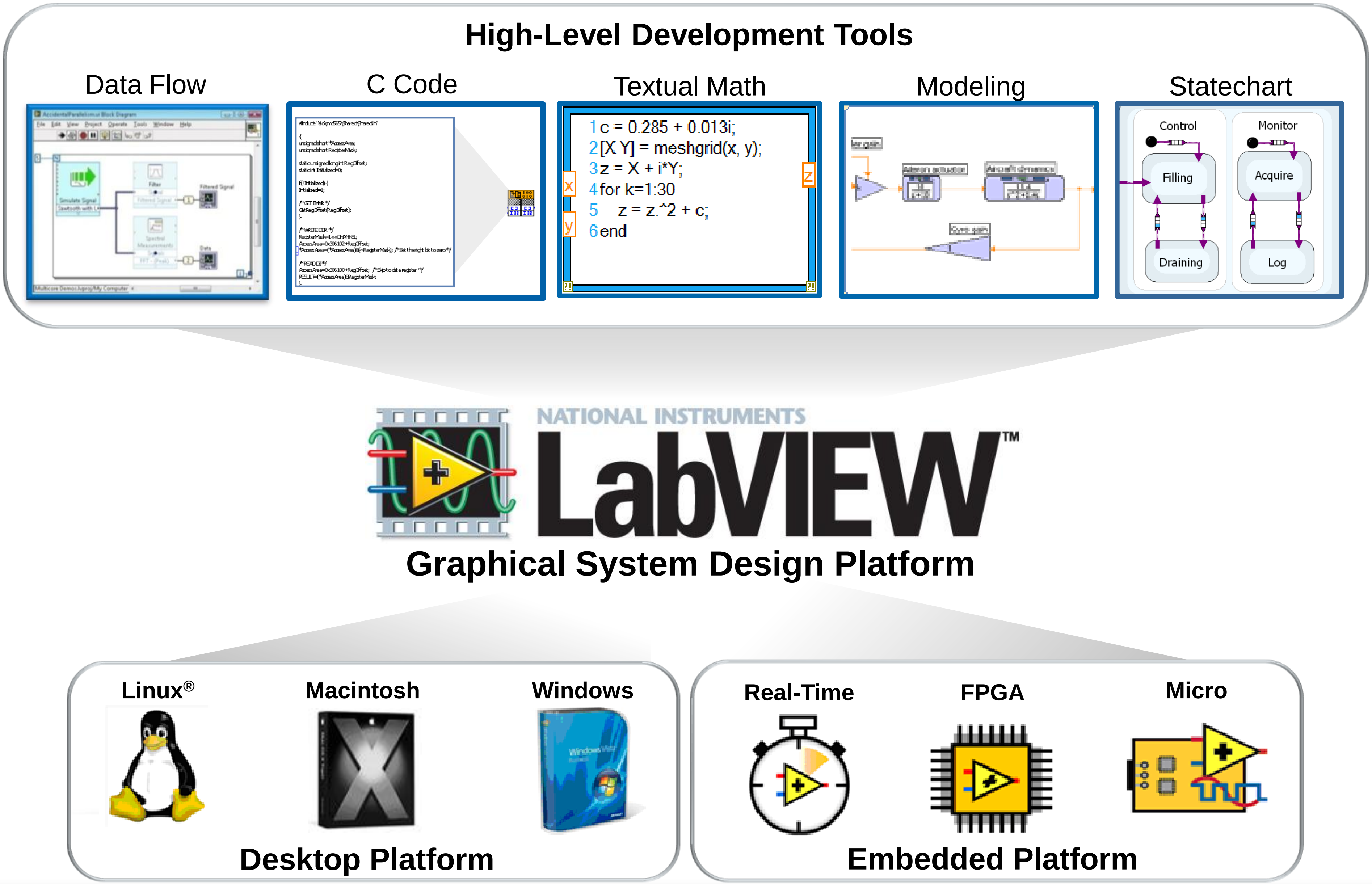
A single graphical development platform



NATIONAL INSTRUMENTS LabVIEW™

The NI Approach – Integrated Hardware Platforms




High-Level Development Tools
Data Flow
C Code
Textual Math
Modeling
Statechart
1 c = 0.285 + 0.013i;
2 [X Y] = meshgrid(x, y);
3 z = X + i*Y;
4 for k=1:30
5 z = z.^2 + c;
6 end
Control
Monitor
Filling
Acquire
Draining
Log
NATIONAL INSTRUMENTS
LabVIEW™
Graphical System Design Platform
Linux®
Macintosh
Windows
Desktop Platform
Real-Time
FPGA
Micro
Embedded Platform

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

A. Setting Up Your Hardware

- Data Acquisition Device (DAQ) Track A
  - Actual USB, PCI, or PXI Device
  - Configured in MAX

- Simulated Data Acquisition Device (DAQ) Track B
  - Software simulated at the driver level
  - Configured in MAX
- Sound Card Track C
  - Built into most computers

What type of device should I use?

	Sound Card*	NI USB DAQ	NI PCI DAQ	Instruments*
AI Bandwidth	8–44 KS/s	10–200 KS/s	250 K–1.2 Ms/s	20kS/s–2 GS/s
Accuracy	12–16 bit	12–16 bit	14–18 bit	12–24 bit
Portable	x	x	—	some
AI Channels	2	8–16	16–80	2
AO Channels	2	1–2	2–4	0
AC or DC	AC	AC/DC	AC/DC	AC/DC
Triggering	—	x	x	x
Calibrated	—	x	x	x

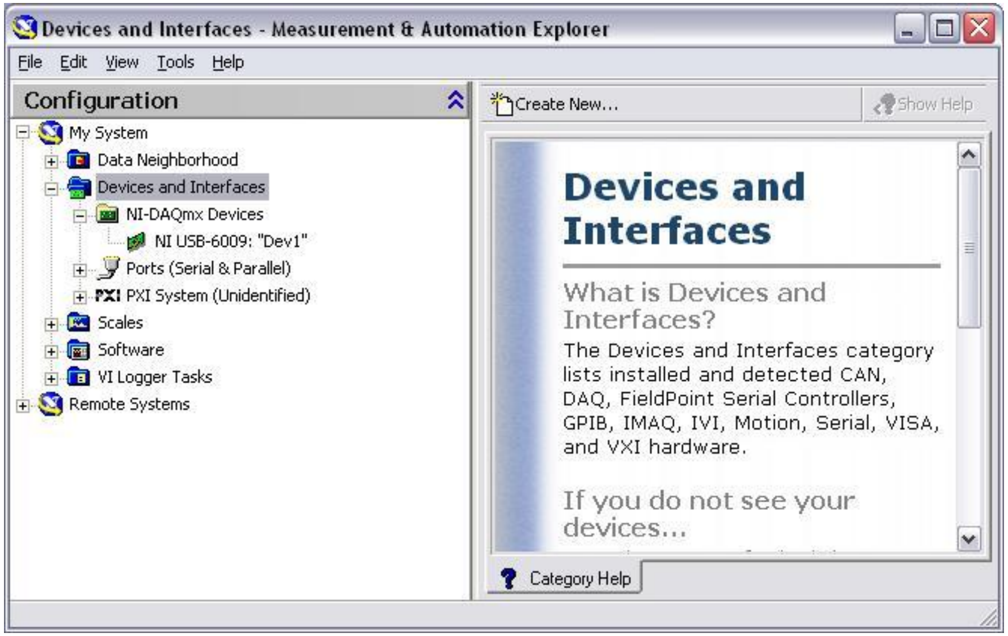
* The above table may not be representative of all device variations that exist in each category

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.

Icon Found on

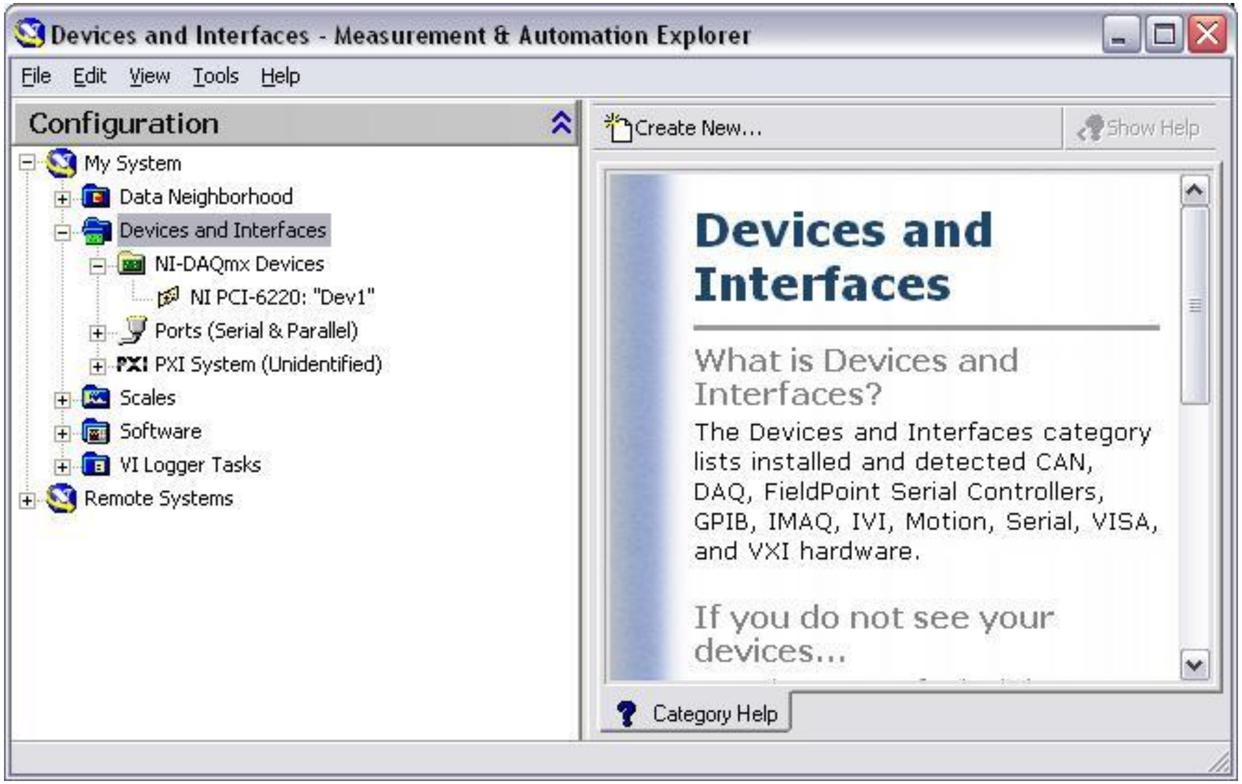
Windows Desktop



Track B

Exercise 1 – Setting Up Your Device

- Use Measurement and Automation Explorer (MAX) to:
  - Configure and test your Simulated Data Acquisition (DAQ) device



Open and Run LabVIEW

Start»All Programs»National Instruments LabVIEW 8.5

Startup Screen:

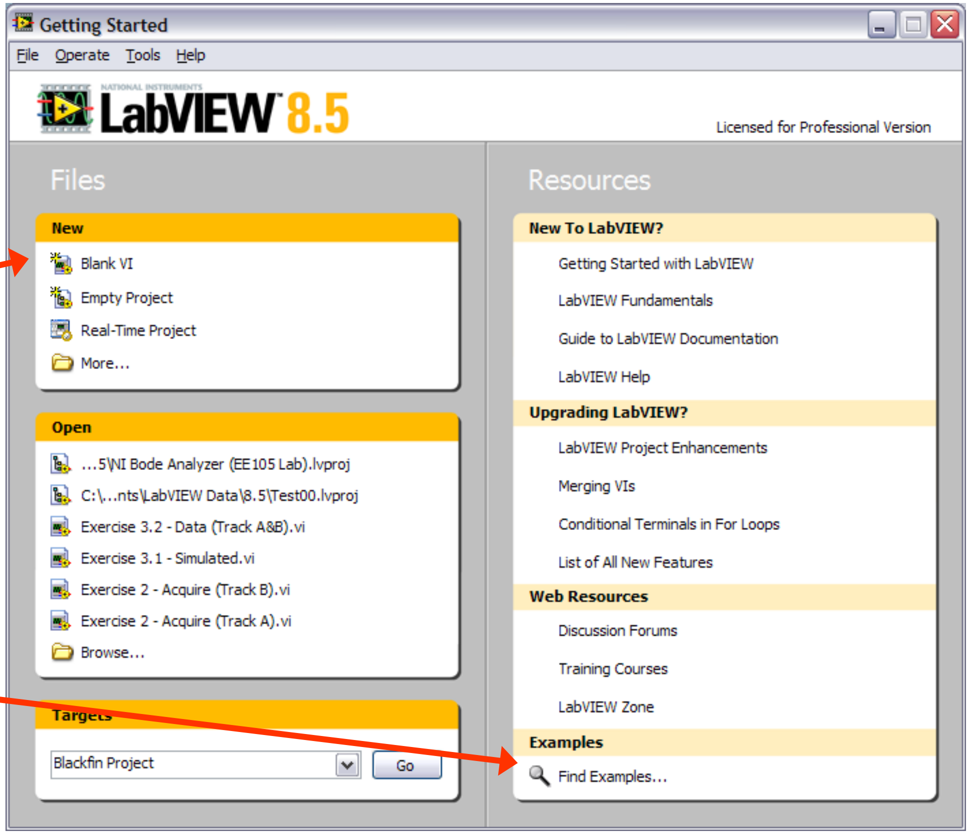
Start from a Blank VI:

New»Blank VI

or

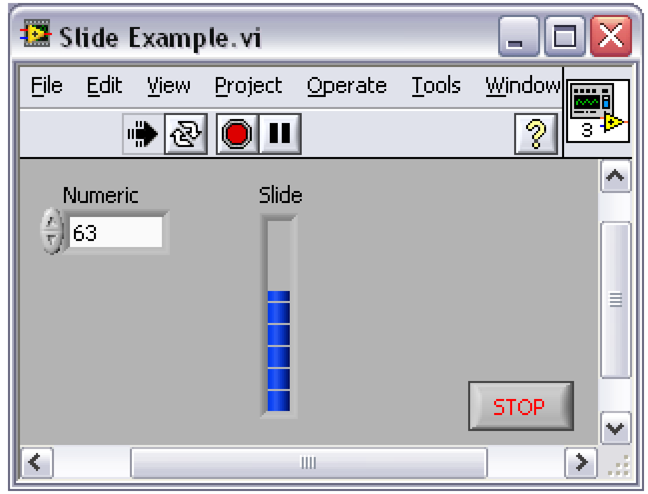
Start from an Example:

Examples»Find Examples…



LabVIEW Programs Are Called Virtual Instruments (VIs)

Each VI has 2 Windows

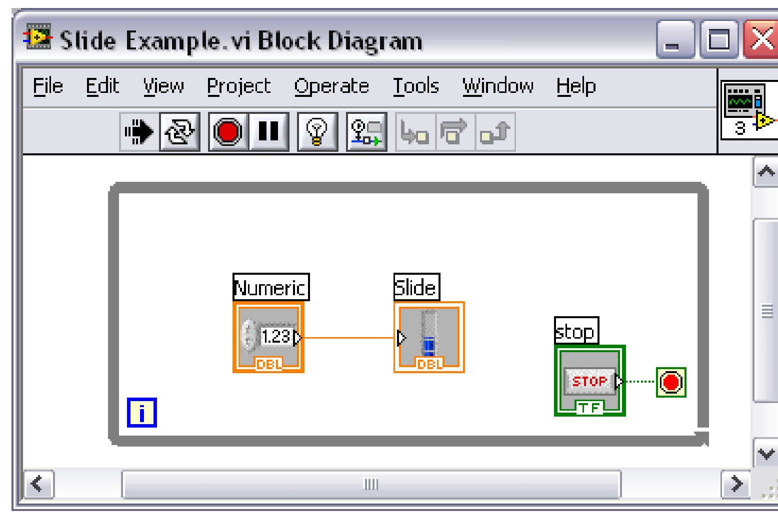


Front Panel

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

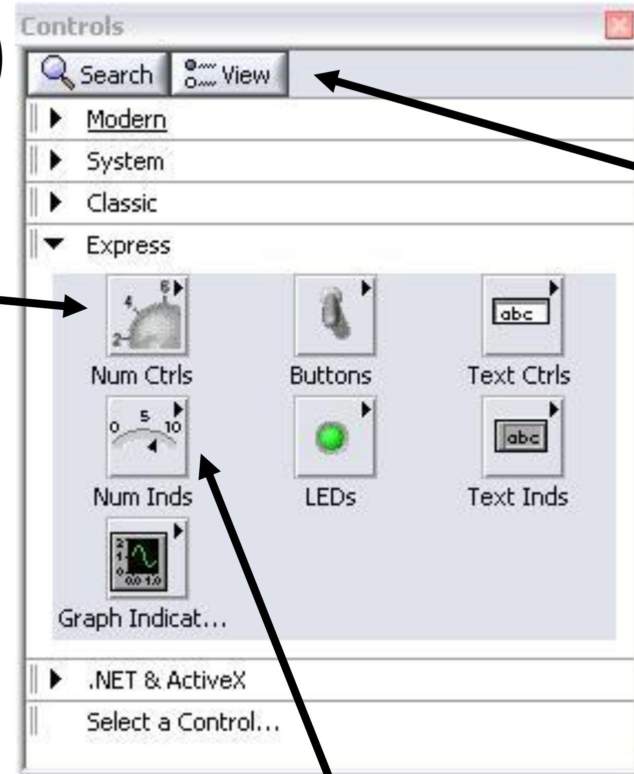
Block Diagram

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



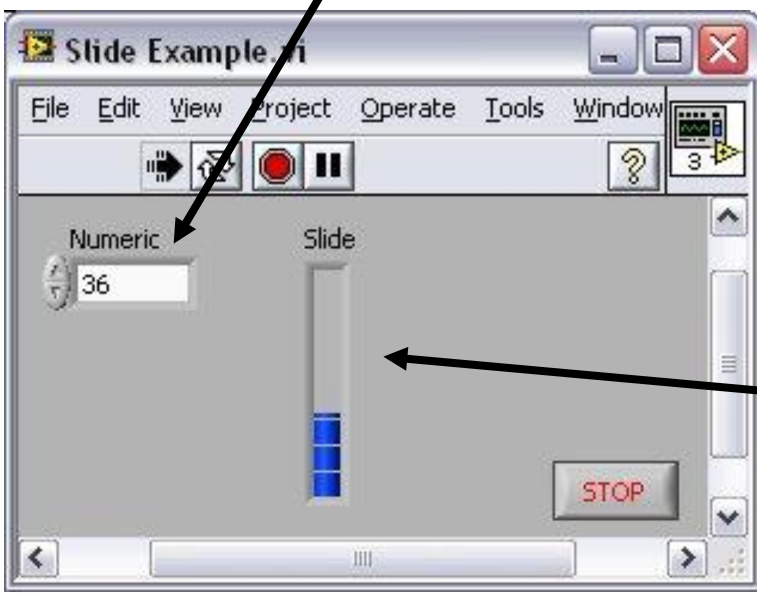
Controls Palette (Controls & Indicators)

(Place items on the Front Panel Window)



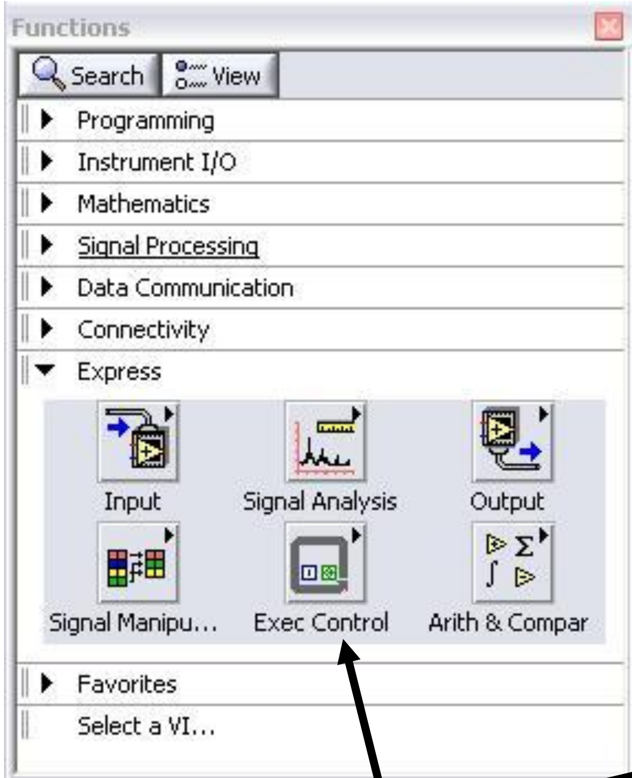
Customize Palette View

Control: Numeric

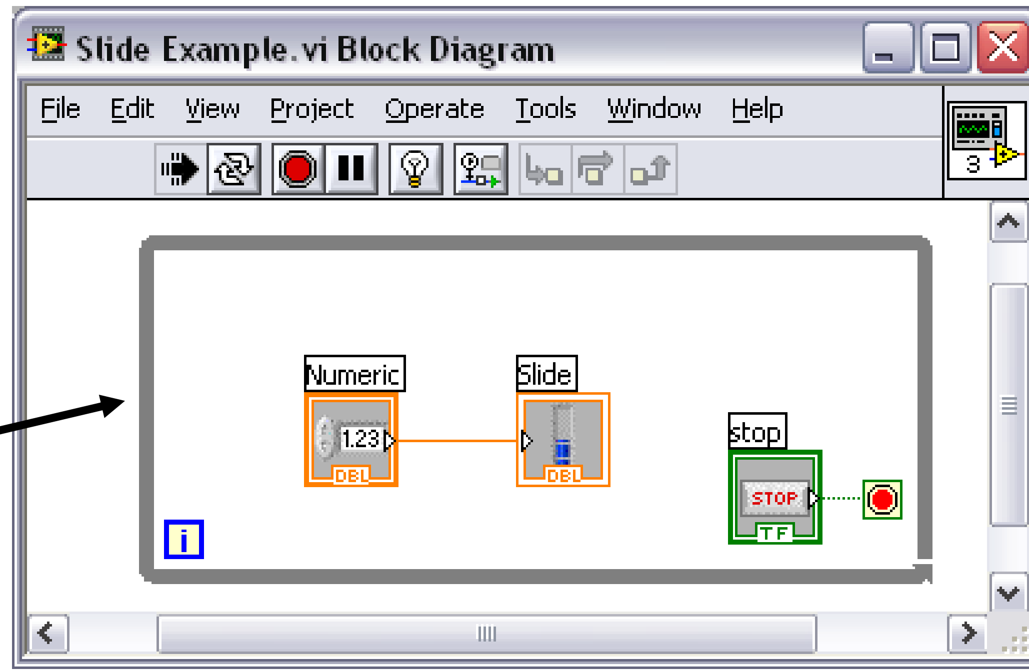


Indicator: Numeric Slide

Functions (and Structures) Palette

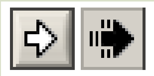


(Place items on the Block Diagram Window)



Structure:
While Loop

Status Toolbar


Run Button

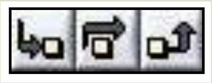

Continuous Run Button


Abort Execution

Additional Buttons on the Diagram Toolbar

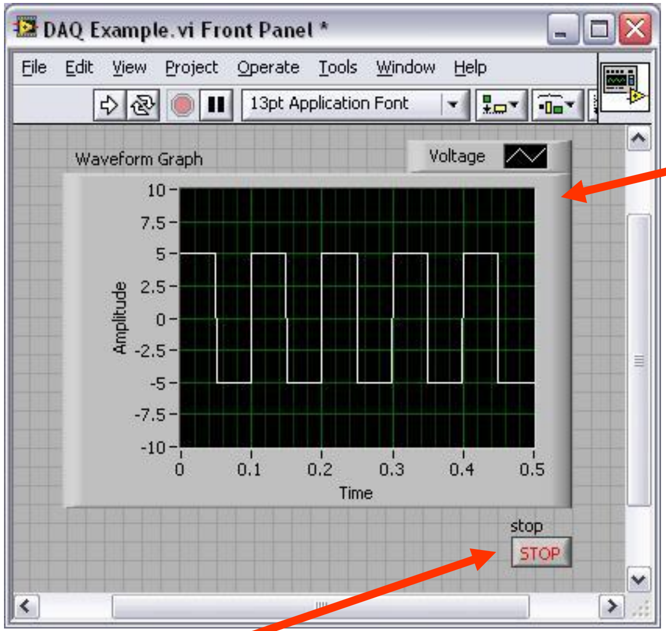

Execution Highlighting Button


Retain Wire Values Button


Step Function Buttons

Demonstration 1: Creating a VI

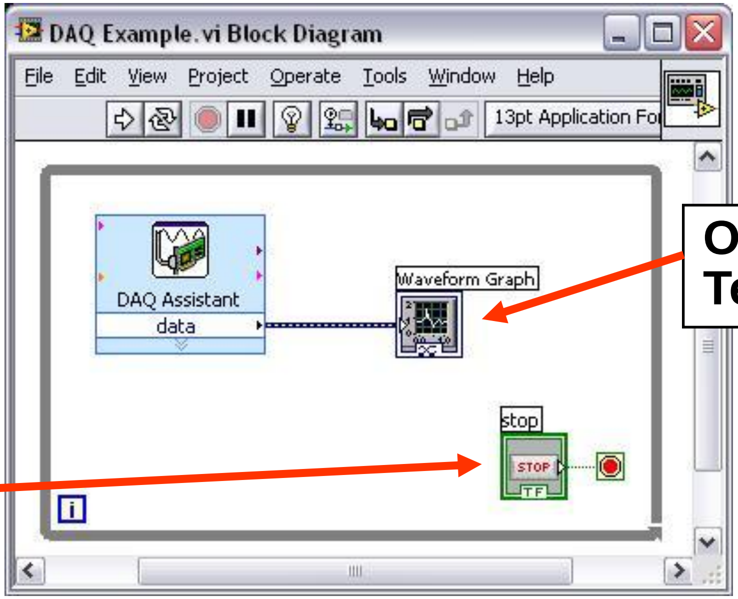
Front Panel Window



Graph Indicator

Boolean Control

Block Diagram Window

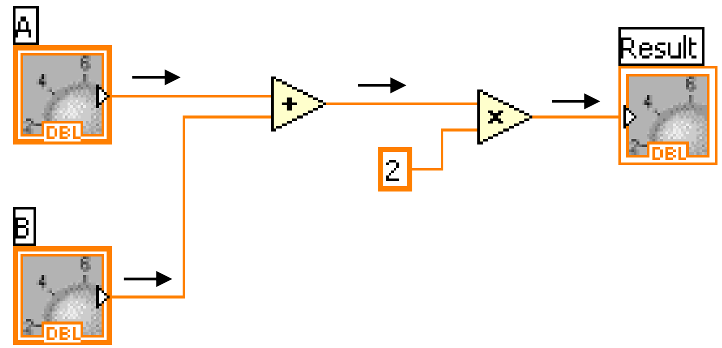


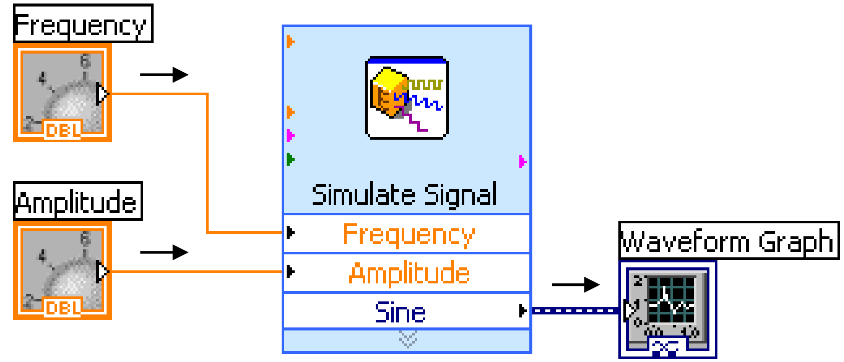
Output Terminal

Input Terminals

Dataflow Programming

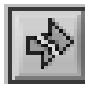
- Block diagram execution
  - Dependent on the flow of data
  - Block diagram does NOT execute left to right
- Node executes when data is available to ALL input terminals
- Nodes supply data to all output terminals when done





Debugging Techniques

- **Finding Errors**

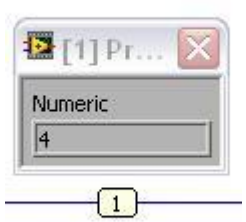


Click on broken **Run** button.
Window showing error appears.

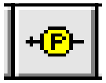
- **Execution Highlighting**

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.

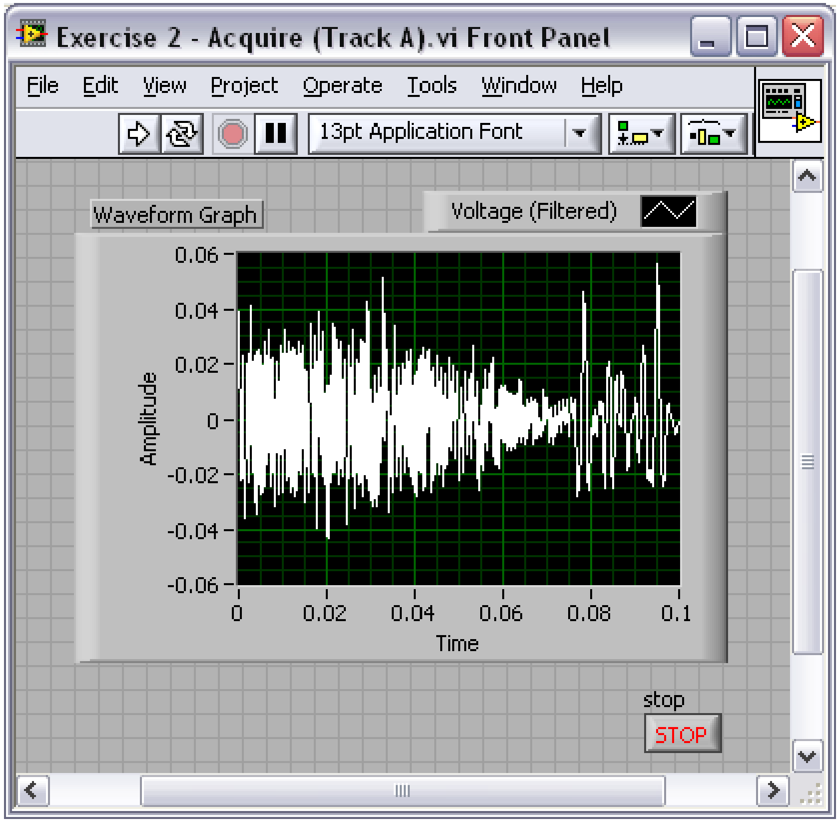


You can also select Probe tool from Tools palette and click on wire.

Exercise 2 – Acquiring a Signal with DAQ

Track A&B

- Use a LabVIEW template to:
  - Acquire a signal from your DAQ device



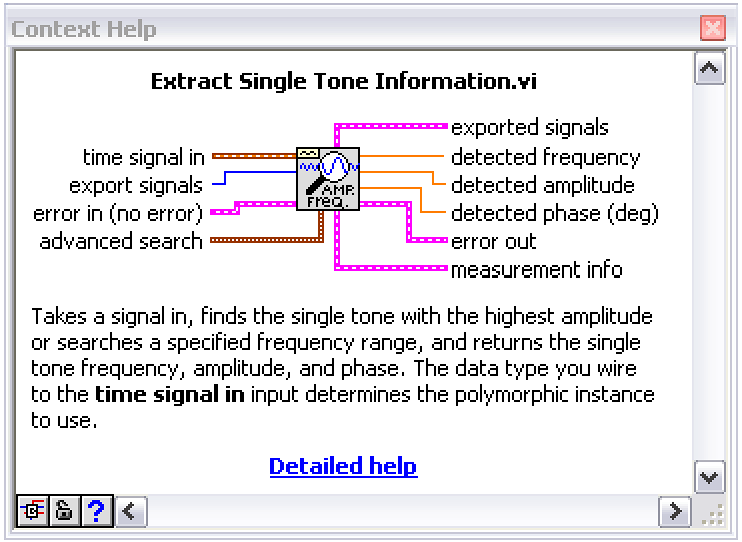
This exercise should take 15 minutes.

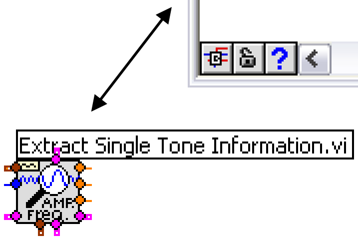
Context Help Window

- **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 "**<u>Detailed Help</u>**." on the context help window





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.

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

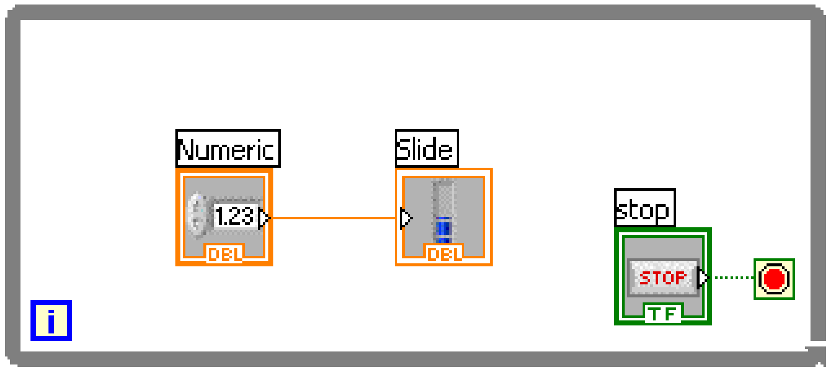
- Case Structure
- Select (simple If statement)
- File I/O

Loops

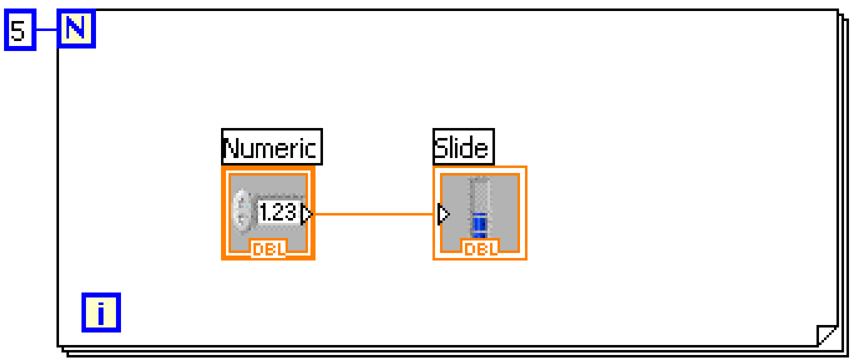
- While Loops
  - i erminal counts iteration
  - Always runs at least once
  - Runs until stop condition is met

- For Loops
  - i erminal counts iterations
  - Run according to input **N** of count terminal N

While Loop

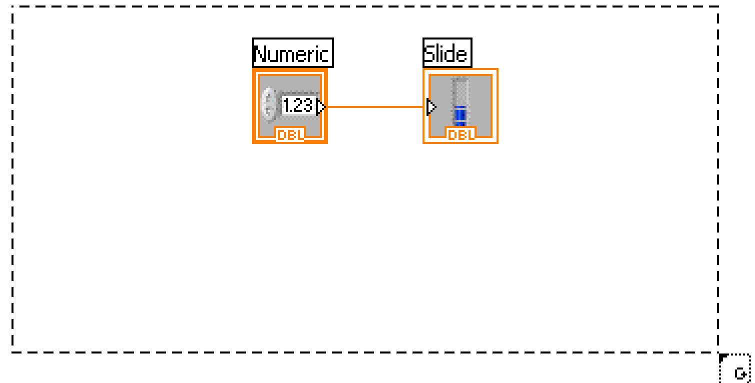


For Loop

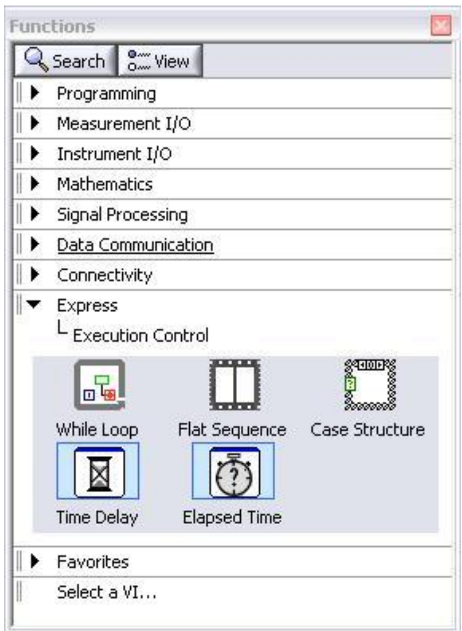


Drawing a Loop

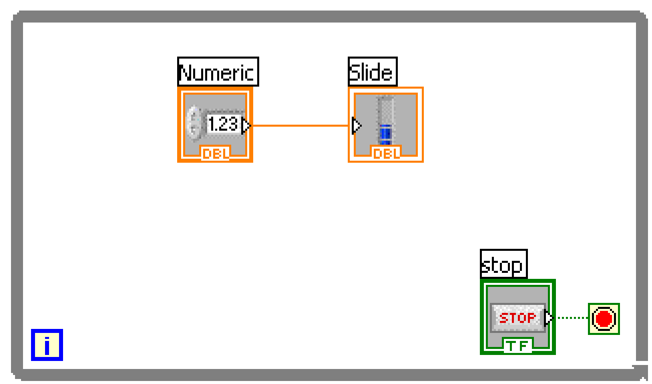
2. Enclose code to be repeated



1. Select the structure

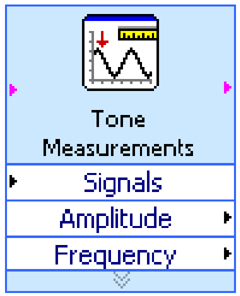


3. Drop or drag additional nodes and then wire

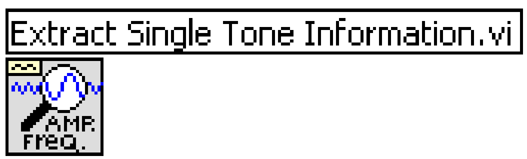


3 Types of Functions (from the Functions Palette)

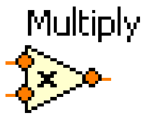
Express VIs: interactive VIs with configurable dialog page **(blue border)**



Standard VIs: modularized VIs customized by wiring **(customizable)**



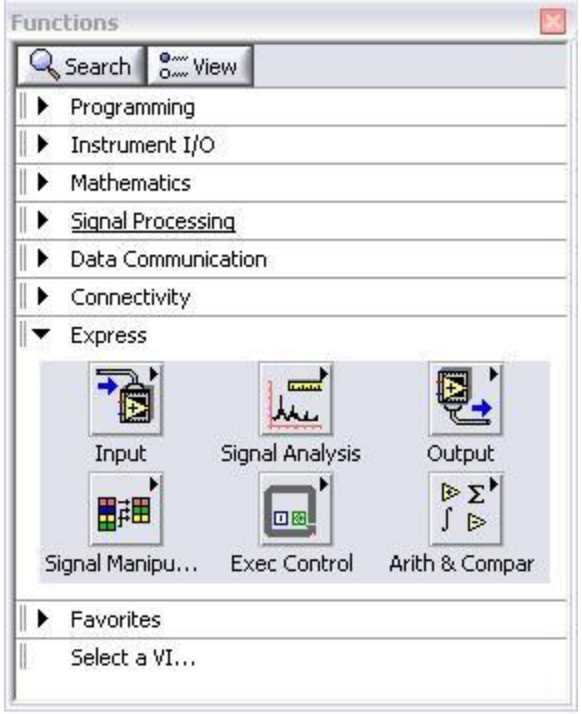
Functions: fundamental operating elements of LabVIEW; no front panel or block diagram **(yellow)**



What Types of Functions are Available?

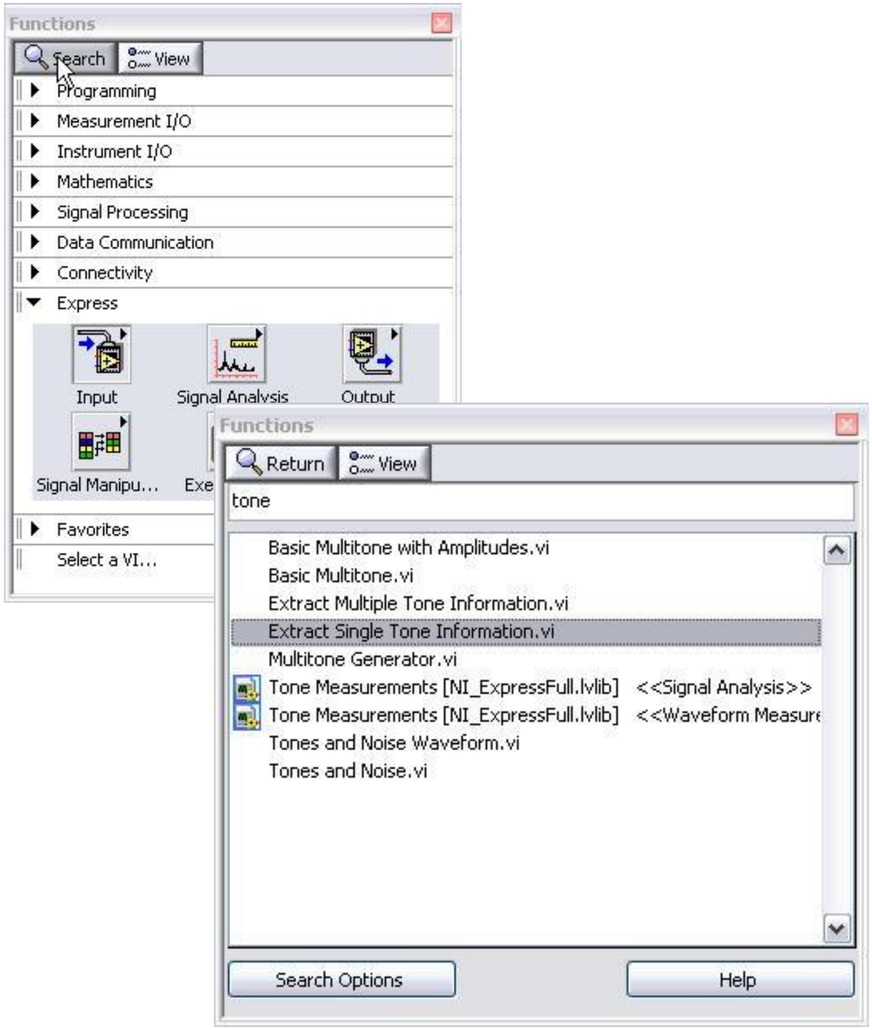
- **Input and Output**
  - Signal and Data Simulation
  - Acquire and Generate Real Signals with DAQ
  - Instrument I/O Assistant (Serial & GPIB)
  - ActiveX for communication with other programs
- **Analysis**
  - Signal Processing
  - Statistics
  - Advanced Math and Formulas
  - Continuous Time Solver
- **Storage**
  - File I/O

Express Functions Palette



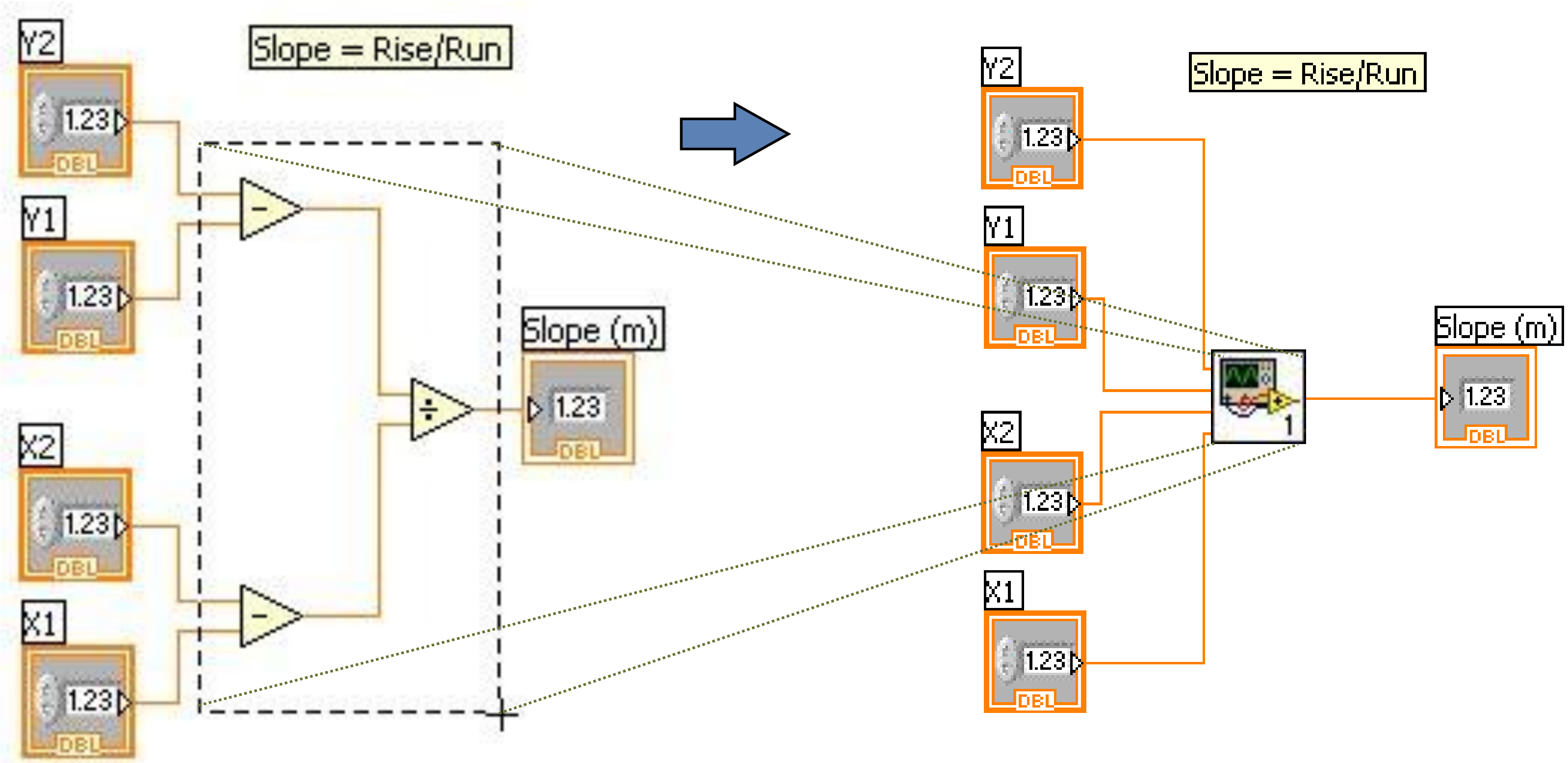
Searching for Controls, VIs, and Functions

- 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



Create SubVI

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



LabVIEW Functions and SubVIs operate like Functions in other languages

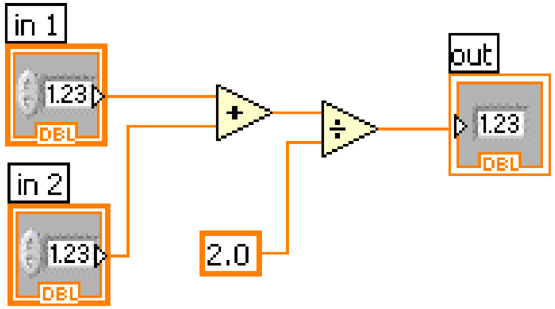
Function Pseudo Code

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

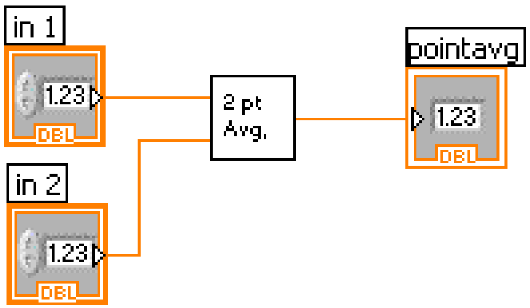
Calling Program Pseudo Code

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

SubVI Block Diagram



Calling VI Block Diagram

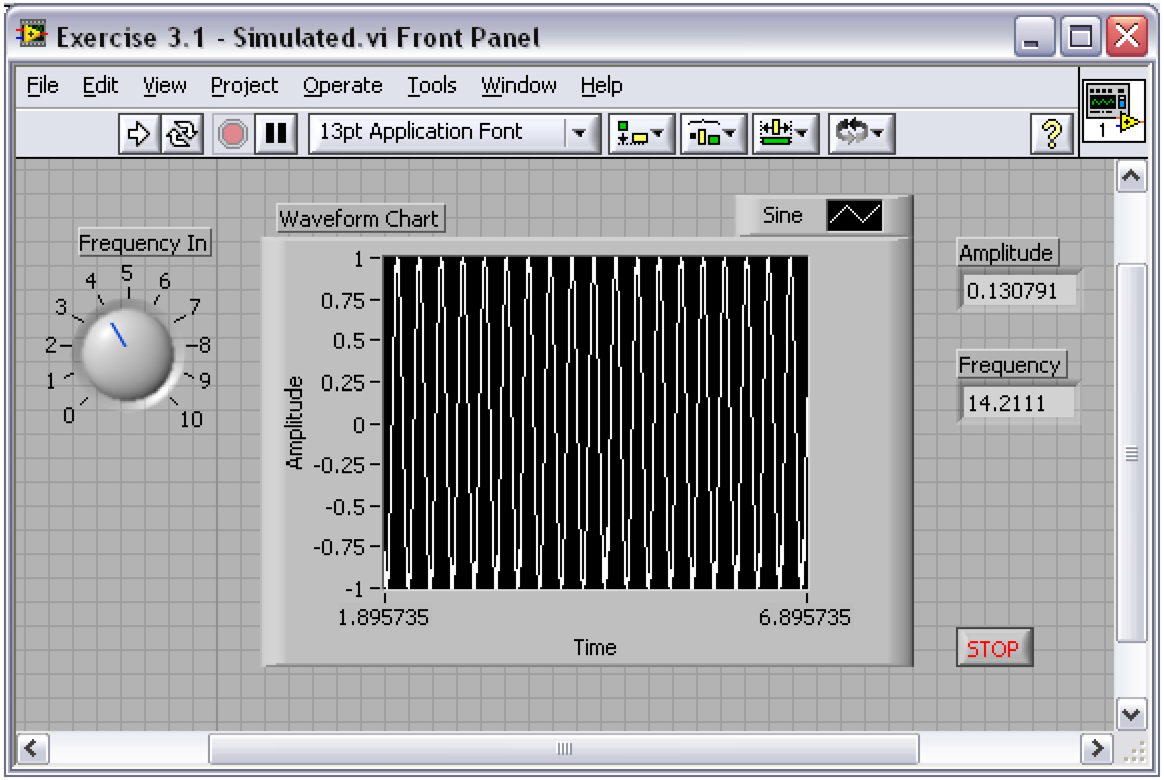


Track A,B,&C

Exercise 3.1 – Analysis

- Use LabVIEW Express VIs to:
  - Simulate a signal and display its amplitude and frequency

This exercise should take 15 minutes.

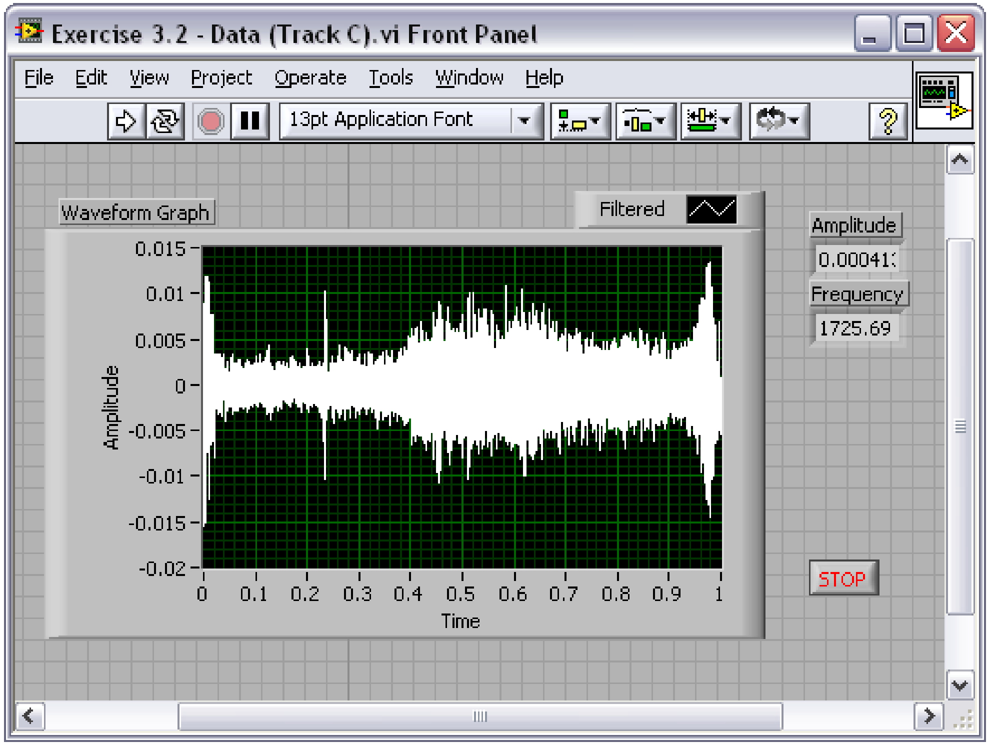


Track A&B

Exercise 3.2 – Analysis

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

This exercise should take 15 minutes.

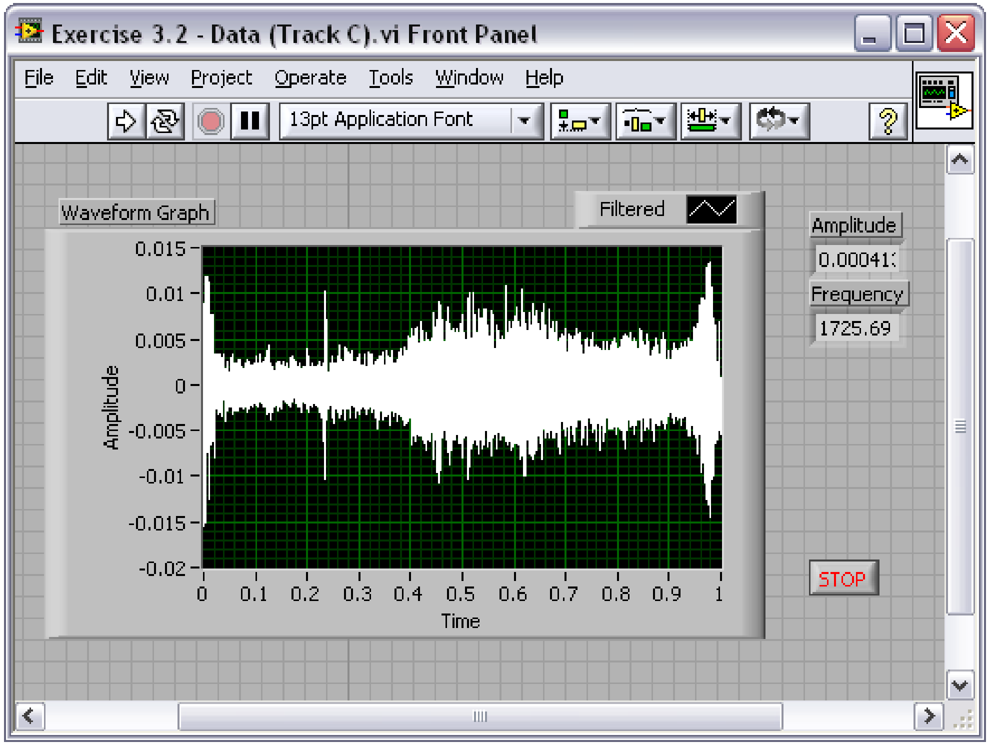


Track C

Exercise 3.2 – Analysis

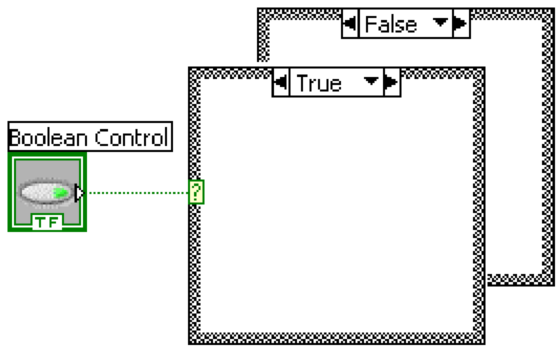
- Use LabVIEW Express VIs to:
  - Acquire a signal and display its amplitude and frequency

This exercise should take 15 minutes.

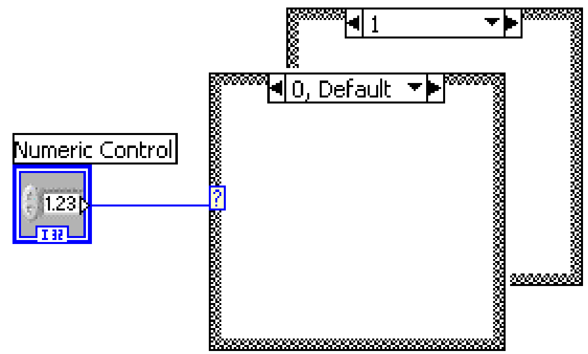


How Do I Make Decisions in LabVIEW?

1. Case Structures

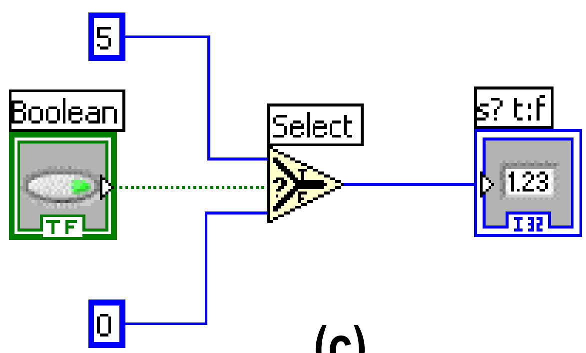


(a)



(b)

2. Select

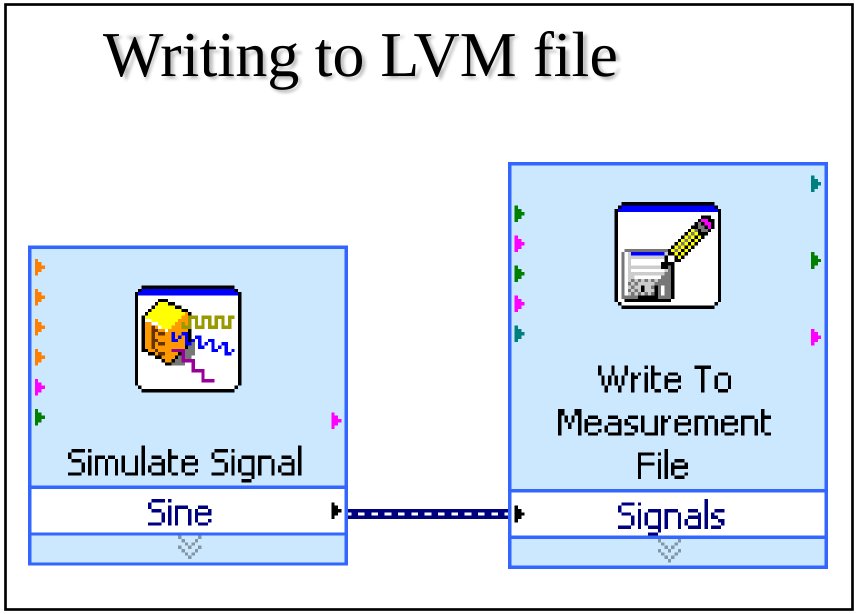


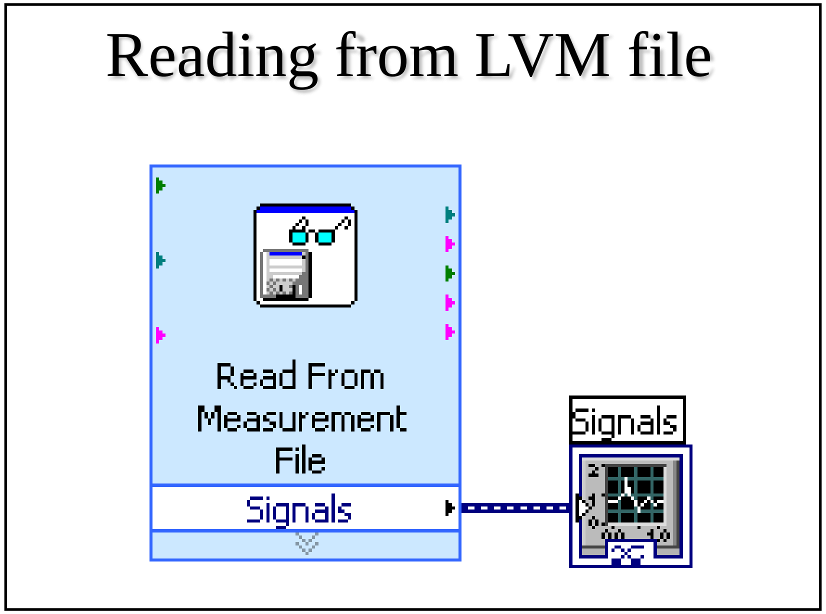
(c)

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)



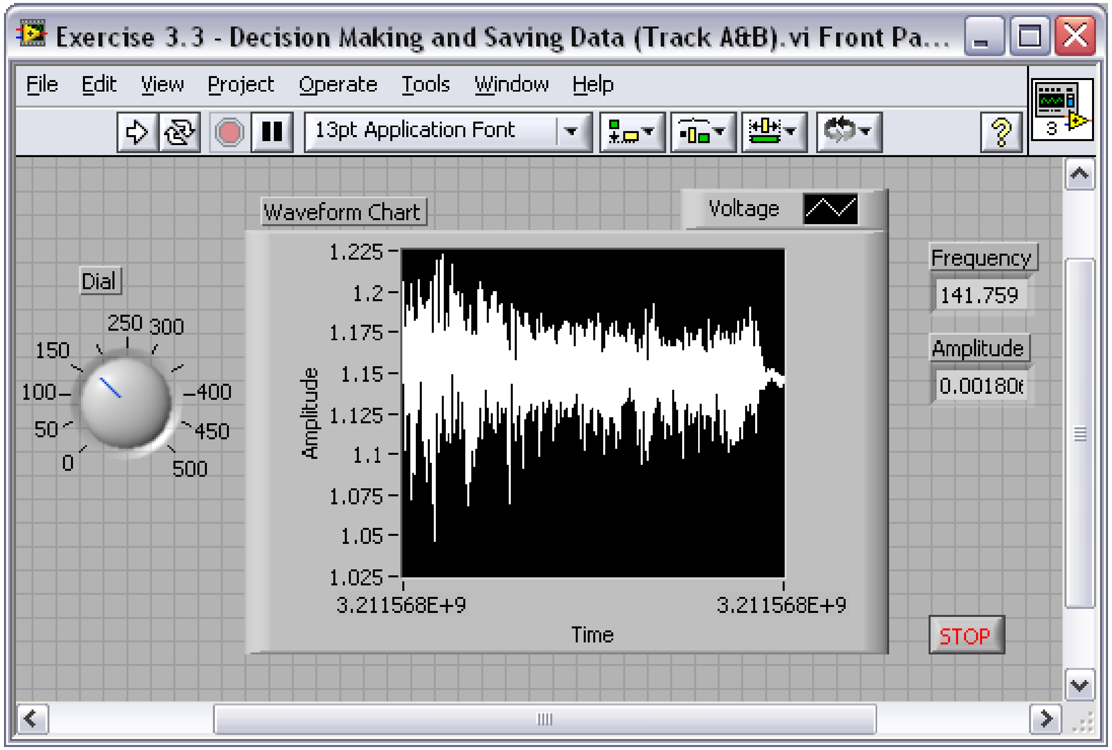


Track A,B,C

Exercise 3.3 – Decision Making and Saving Data

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

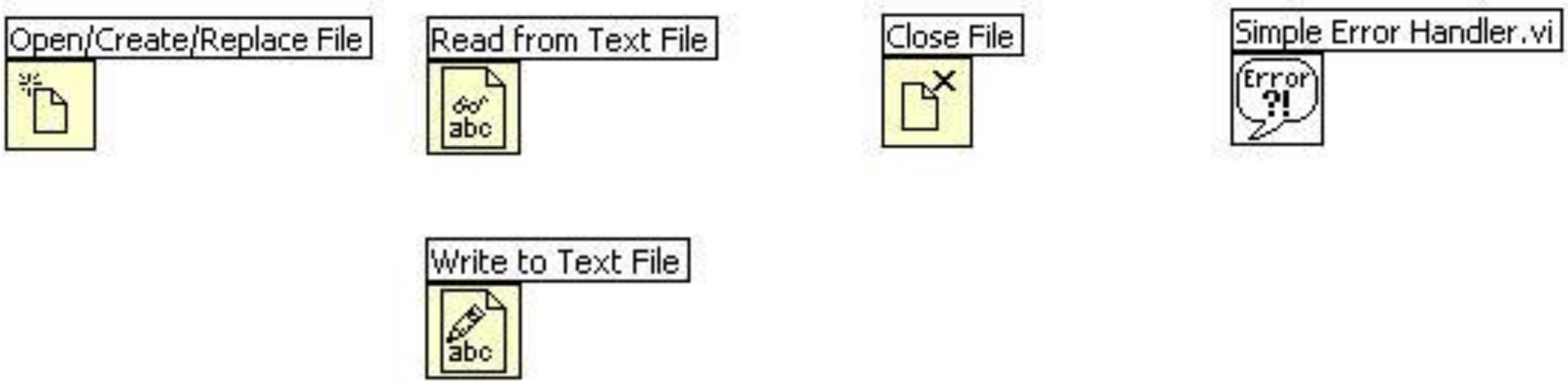
This exercise should take 15 minutes.



NATIONAL INSTRUMENTS™

File I/O Programming Model – Under the hood

Open/ Create/ Replace File → Read and/or Write to File → Close File → Check for Errors



Section III – Presenting your Results

A. Displaying Data on the Front Panel

- Controls and Indicators
- Graphs and Charts
- Loop Timing

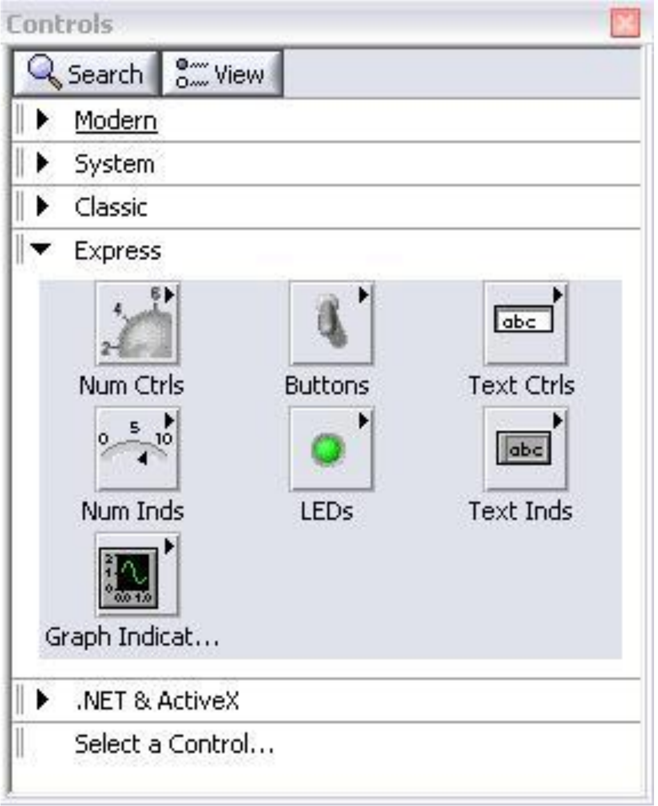
B. Signal Processing

- 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

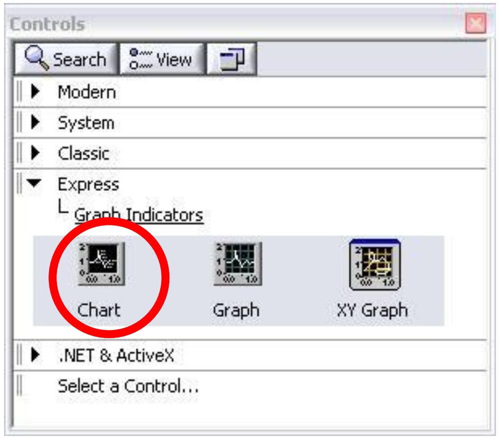
Express Controls Palette



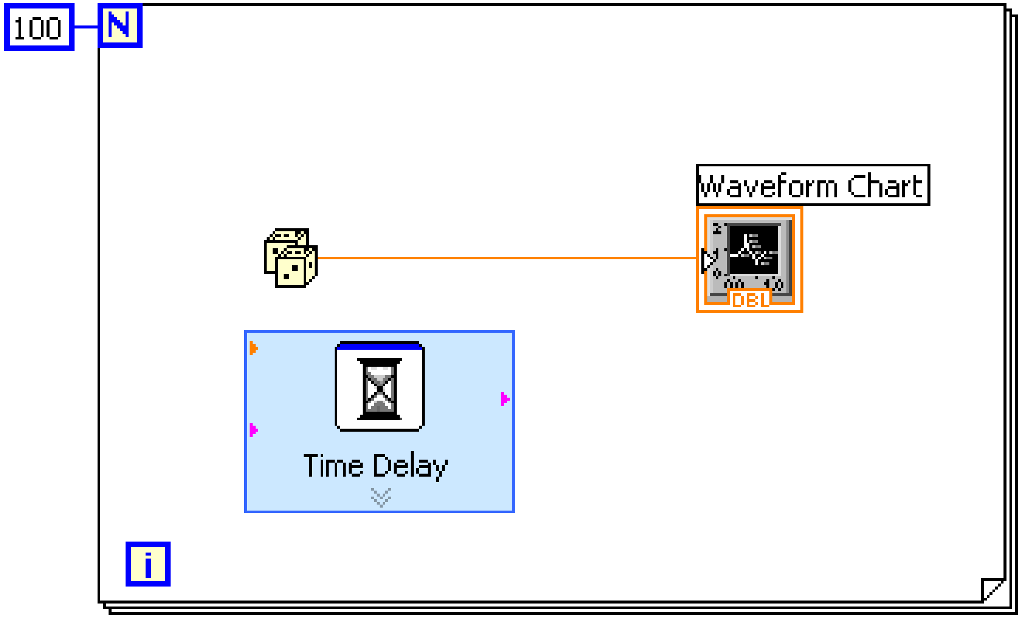
Charts – Add 1 data point at a time with history

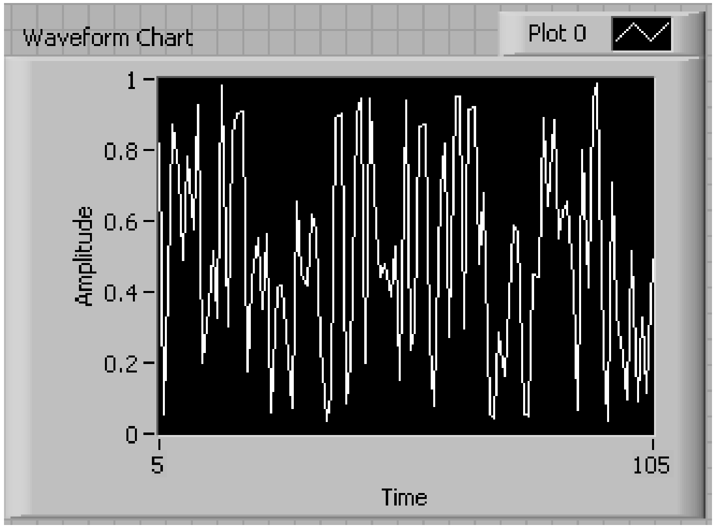
Waveform chart – special numeric indicator that can display a history of values

- Chart updates with each individual point it receives



Functions»Express»Graph Indicators»Chart

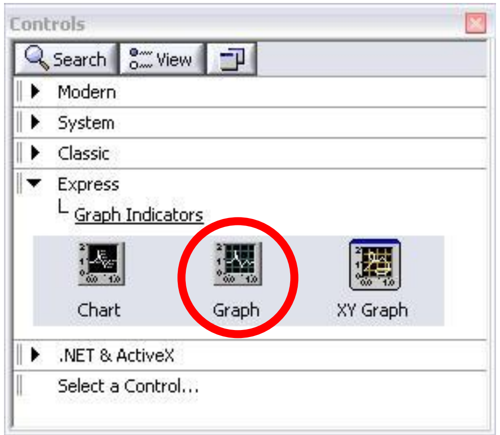




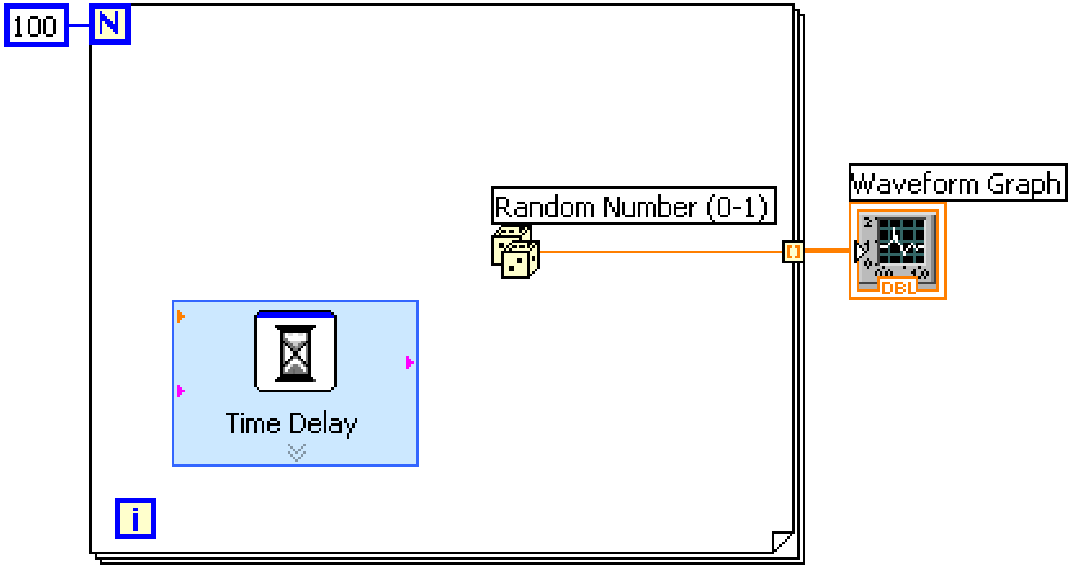
Graphs – Display many data points at once

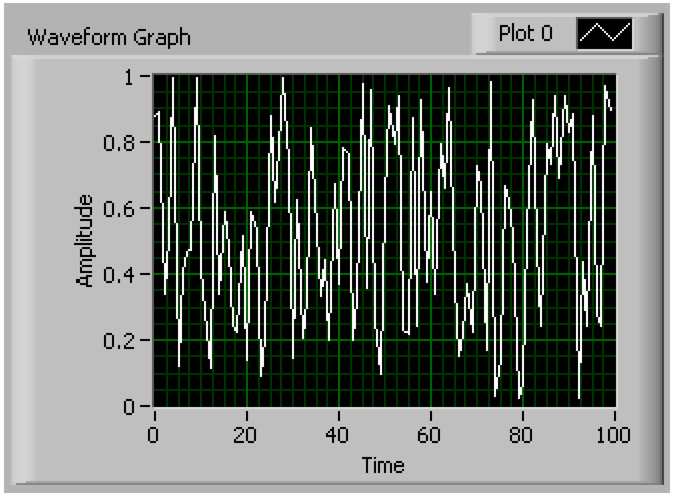
Waveform graph – special numeric indicator that displays an array of data

- Graph updates after all points have been collected
- May be used in a loop if VI collects buffers of data



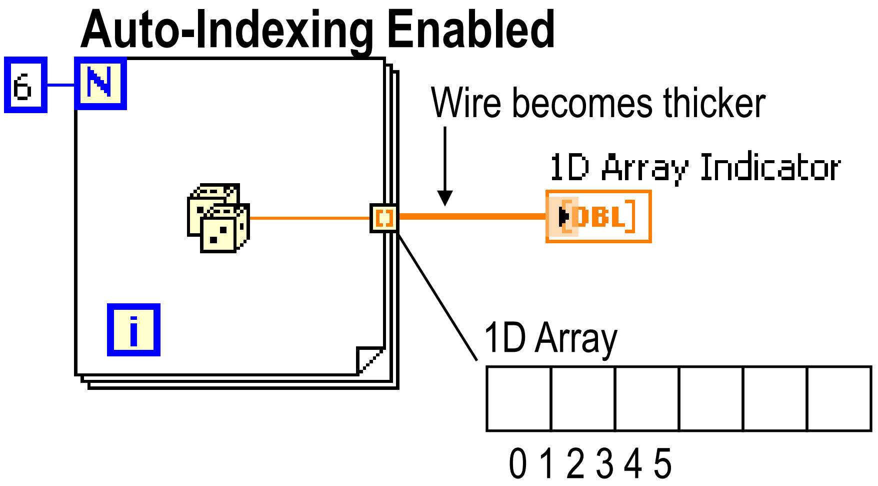
Functions»Express»Graph Indicators»Graph

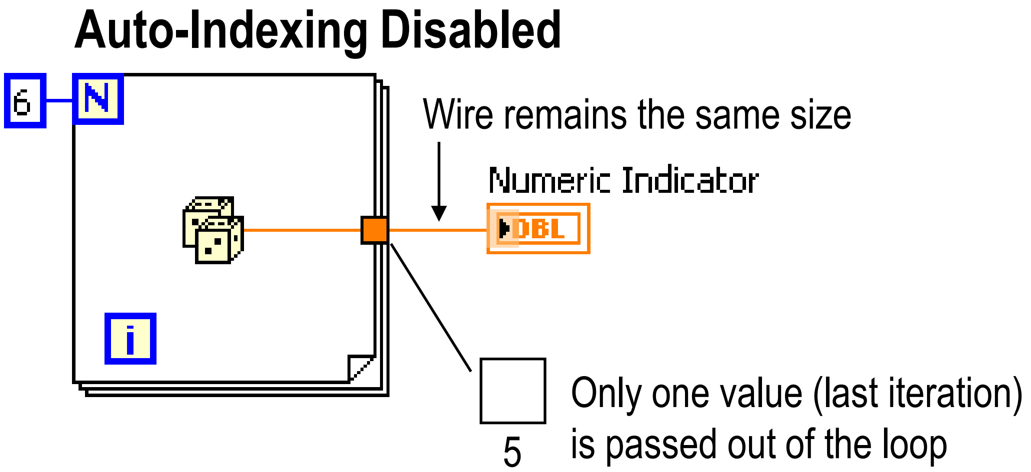




Building Arrays with Loops (Auto-Indexing)

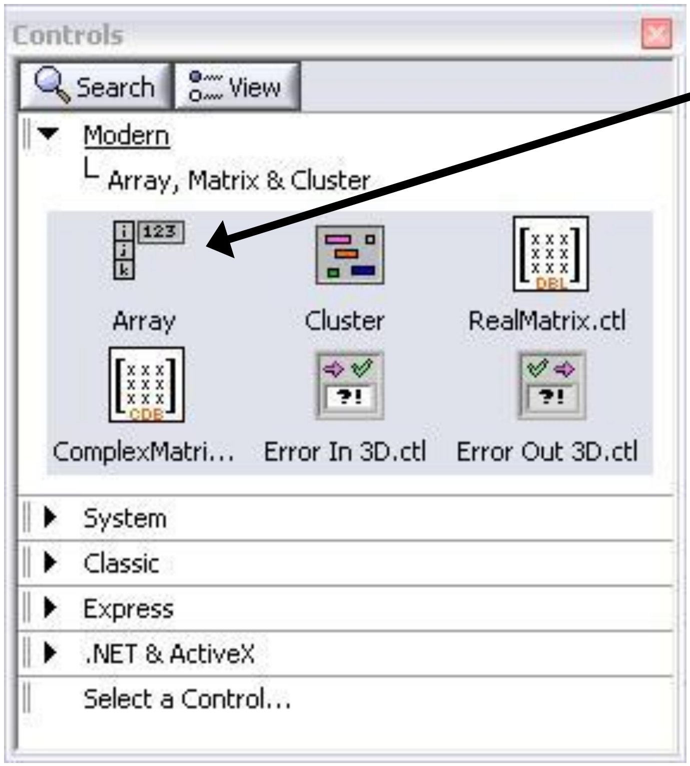
- Loops can accumulate arrays at their boundaries with auto-indexing
- For Loops auto-index by default
- While Loops output only the final value by default
- Right-click tunnel and enable/disable auto-indexing



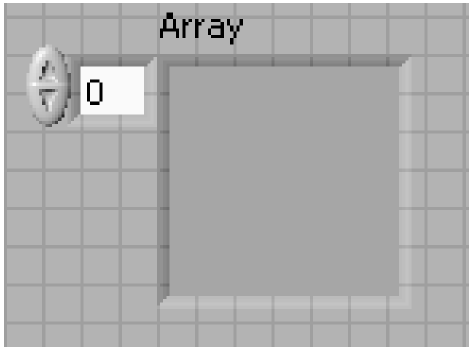


Creating an Array (Step 1 of 2)

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

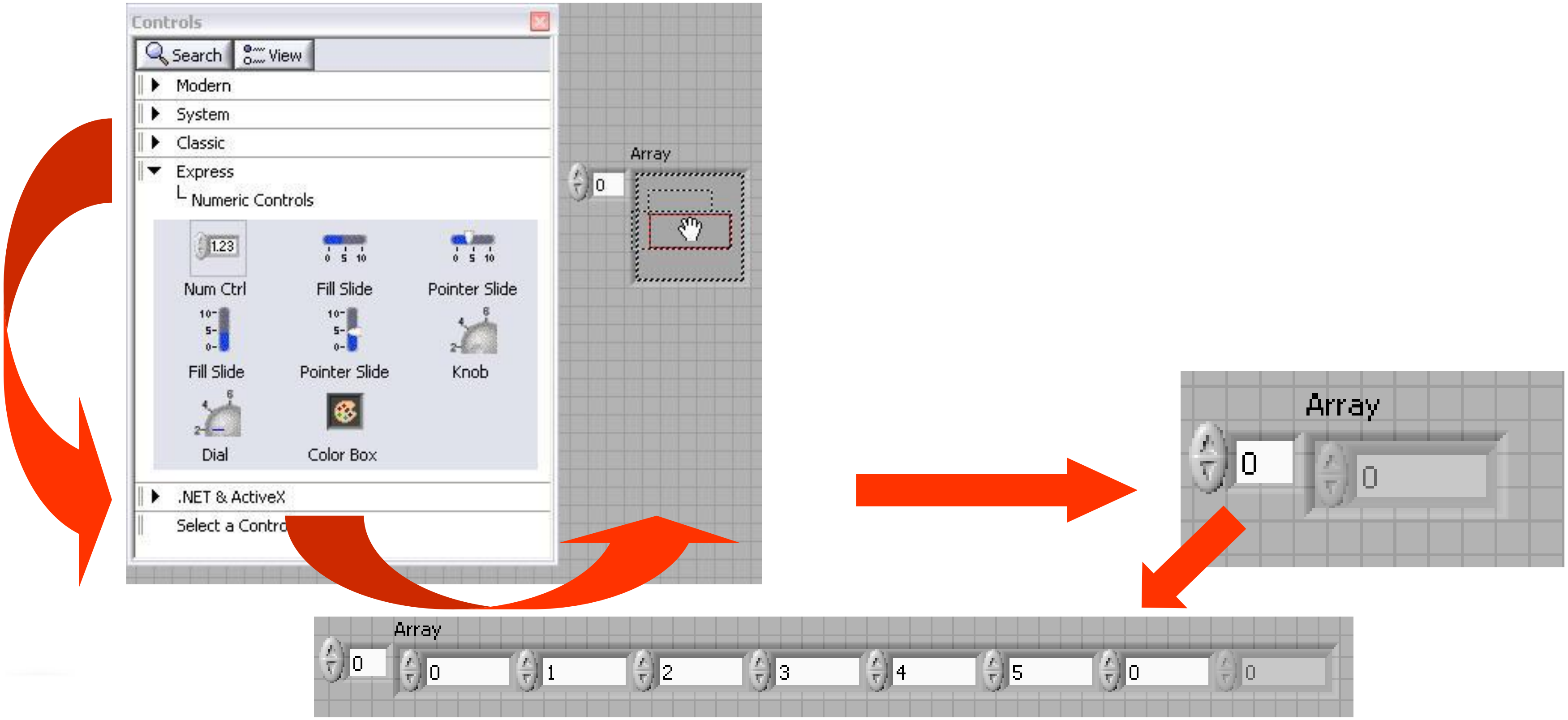


Drop it on the Front Panel.



Create an Array (Step 2 of 2)

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

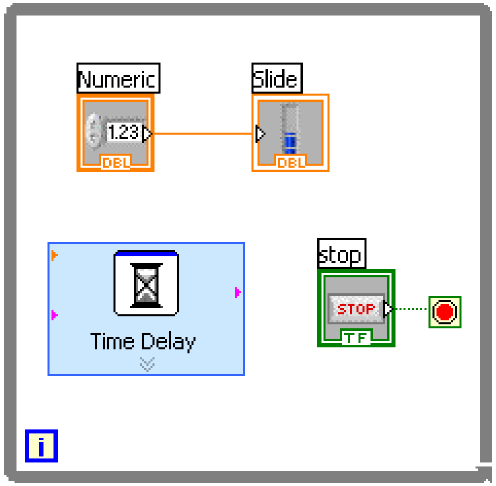


NATIONAL INSTRUMENTS

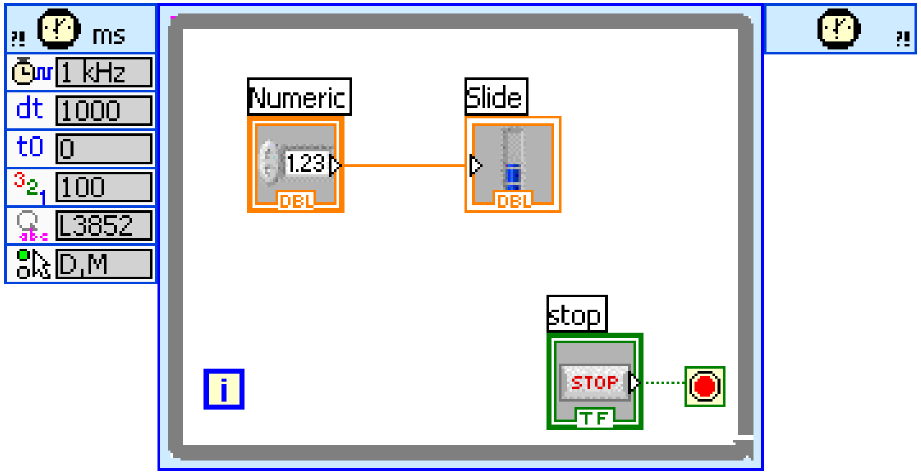
How Do I Time a Loop?

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.*



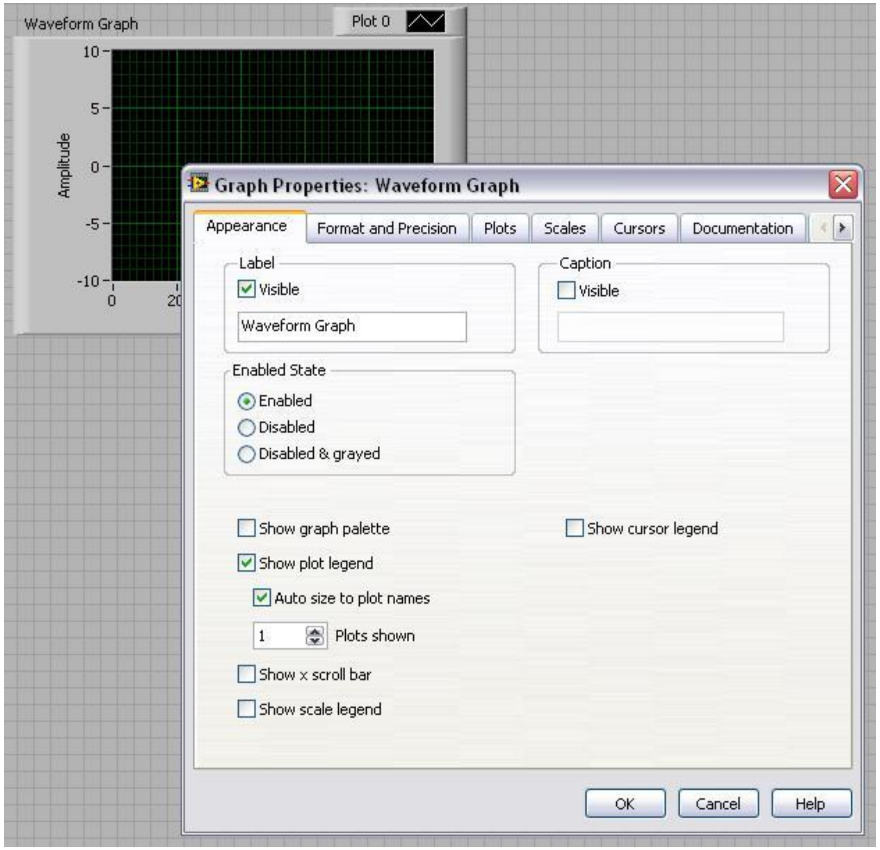
Time Delay



Timed Loop

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

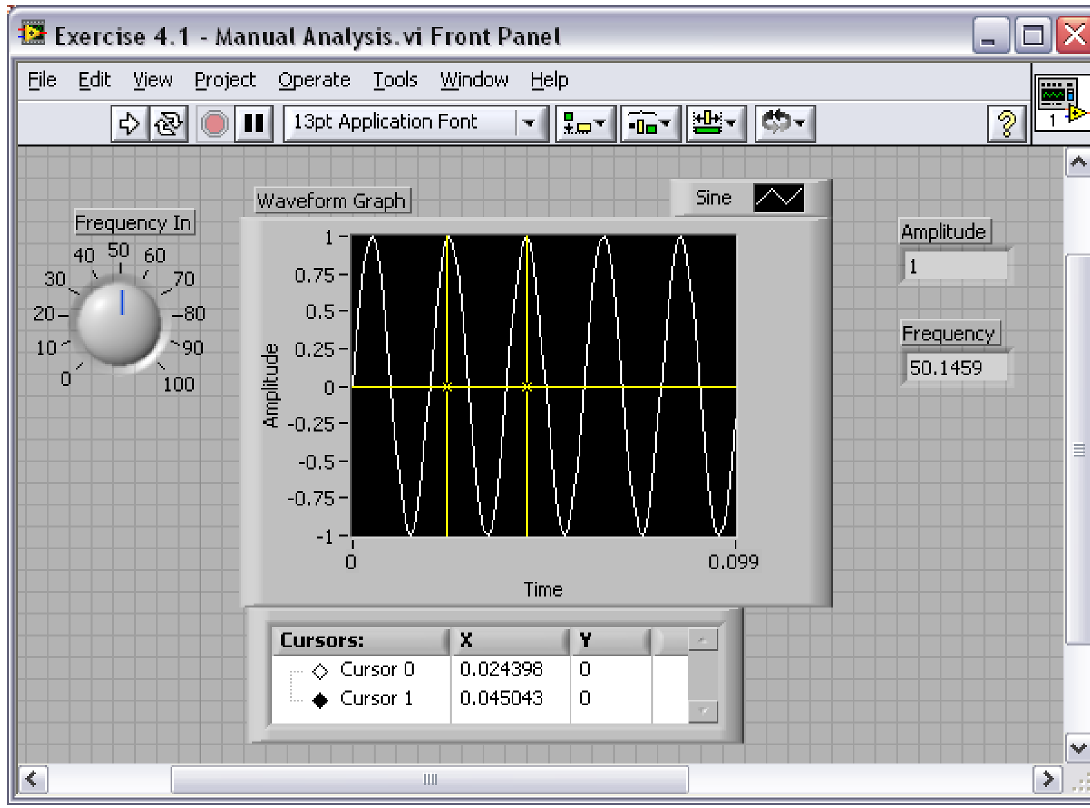


Exercise 4.1 – Manual Analysis

Track A,B,&C

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

This exercise should take 15 minutes.



NATIONAL INSTRUMENTS

Textual Math in LabVIEW

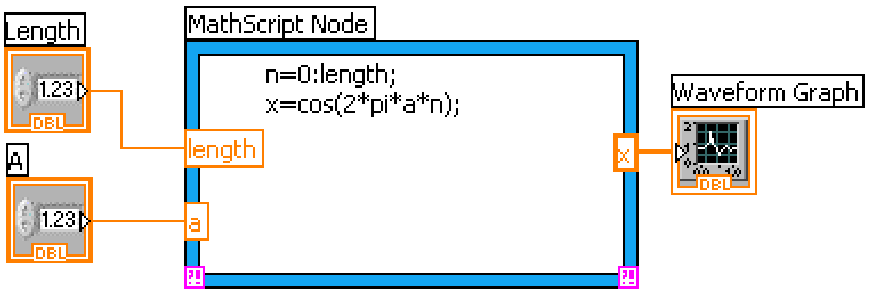
- Integrate existing scripts with LabVIEW for faster development
- Interactive, easy-to-use, hands-on learning environment
- Develop algorithms, explore mathematical concepts, and analyze results using a single environment
- Freedom to choose the most effective syntax, whether graphical or textual within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

Supported Math Tools:

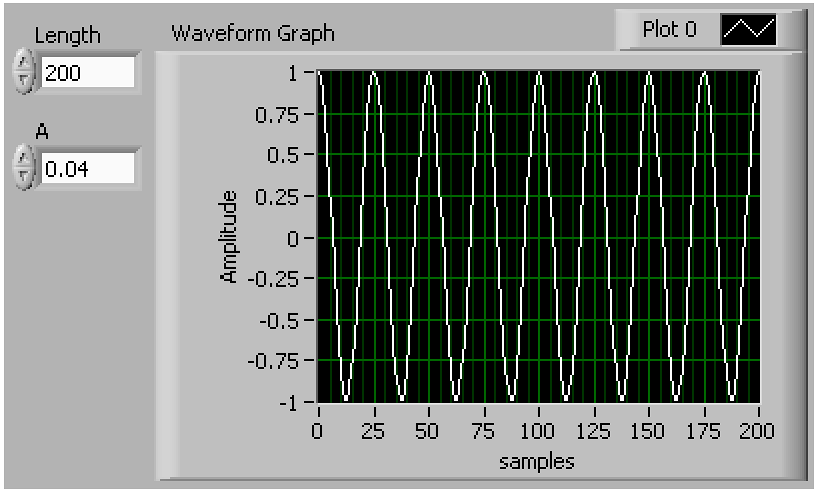
MathScript script node	MathSoft software
Mathematica software	MATLAB® software
Maple software	Xmath software

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



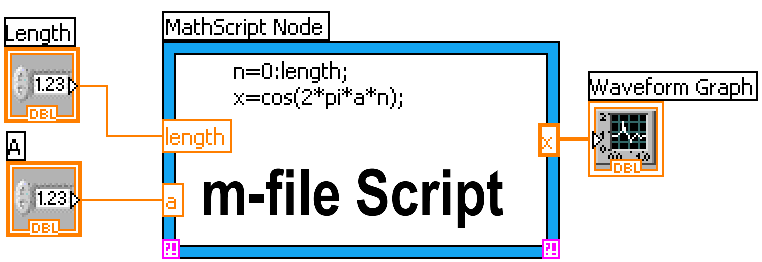
(Functions»Programming» Structures»MathScript)

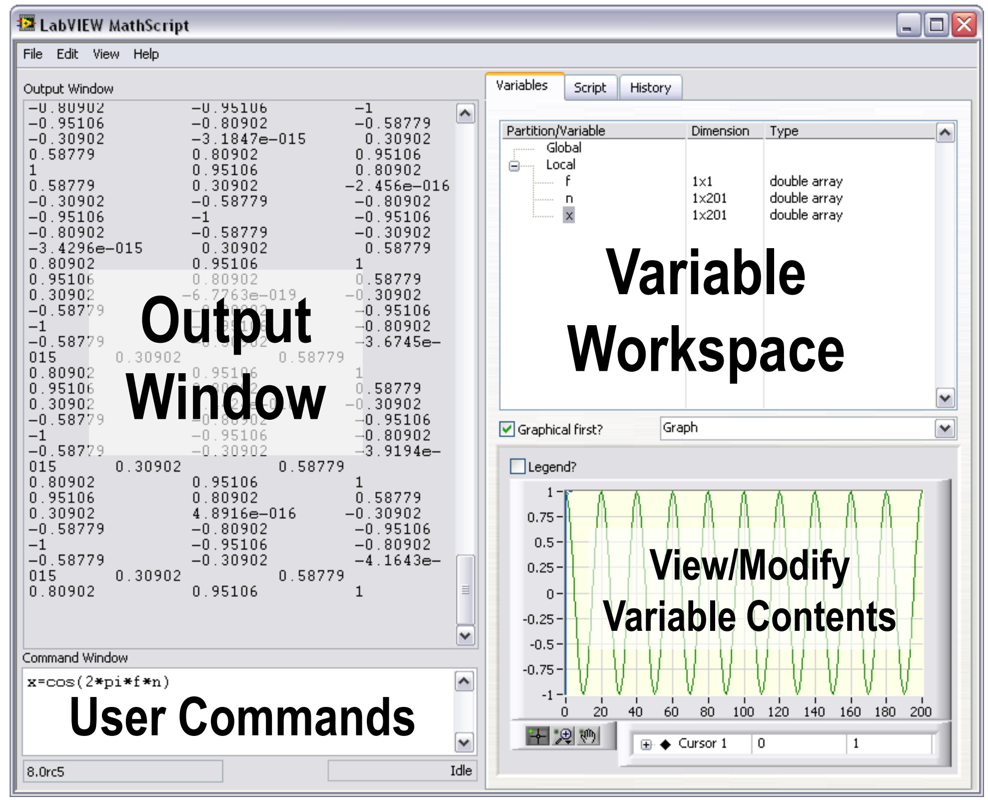


Prototype your equations in the interactive **MathScript Window**.

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





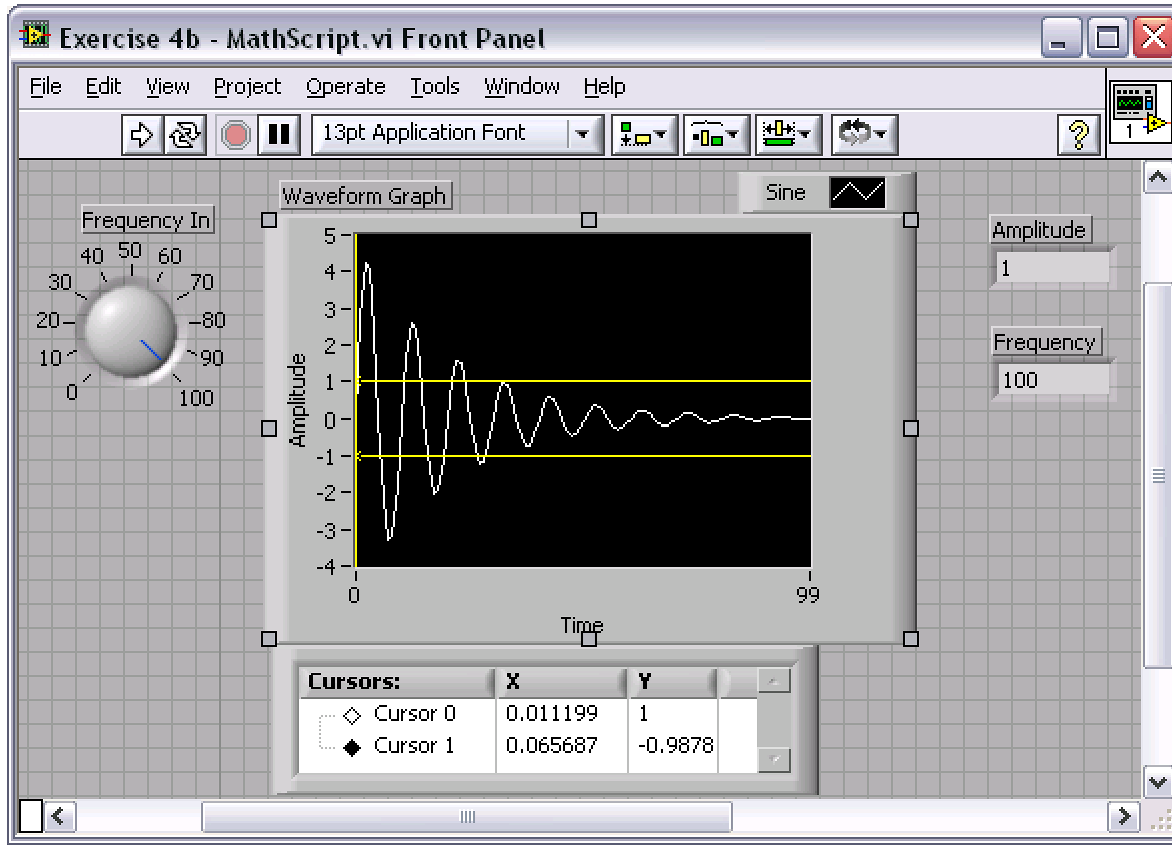
(LabVIEW»Tools»MathScript Window)

Exercise 4.2 – Using MathScript

Track A,B,&C

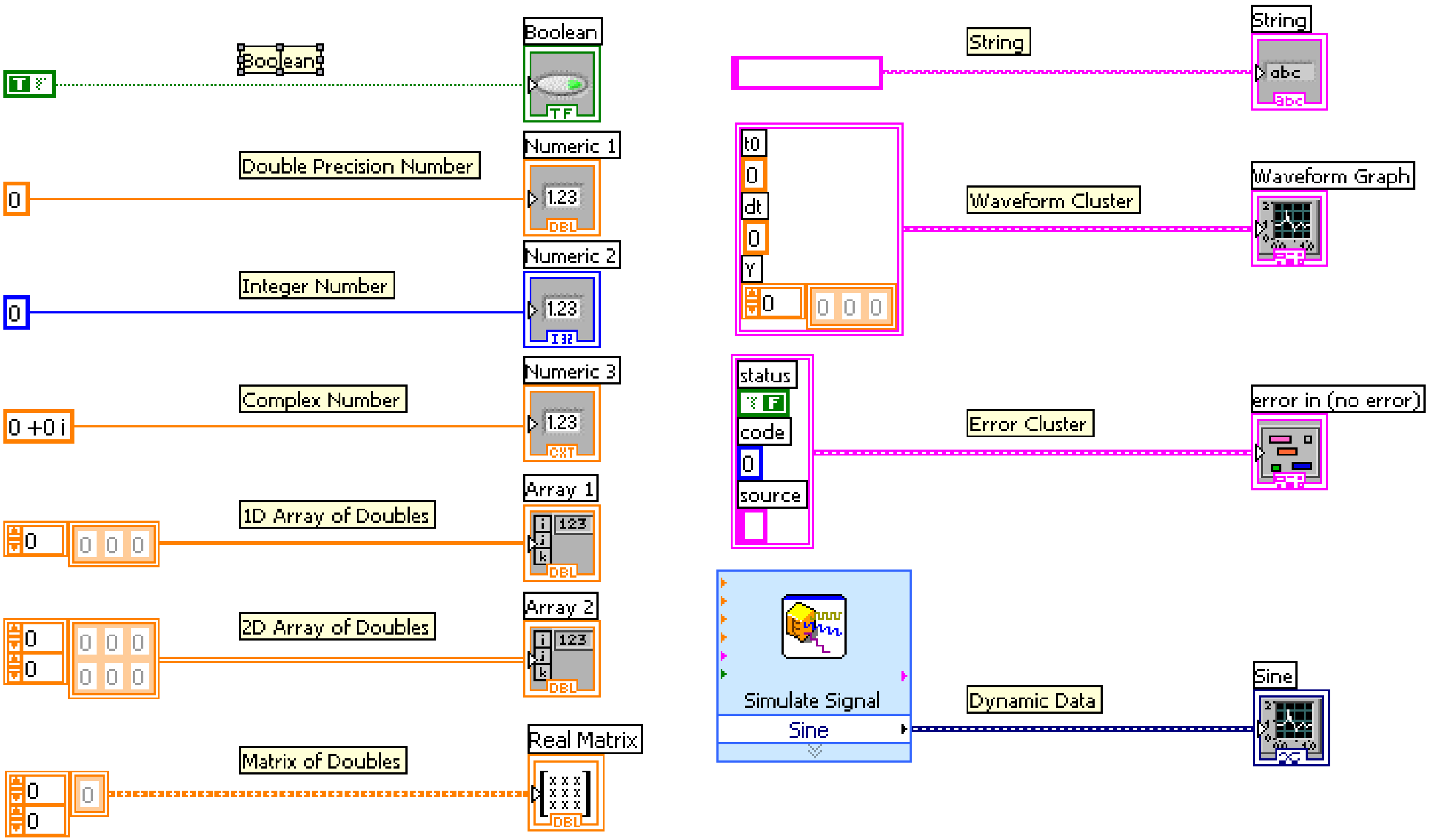
Use the MathScript Node and Interactive Window to process the acquired signal (logarithmic decay) in the MathScript and save the script.

This exercise should take 25 minutes.



NATIONAL INSTRUMENTS

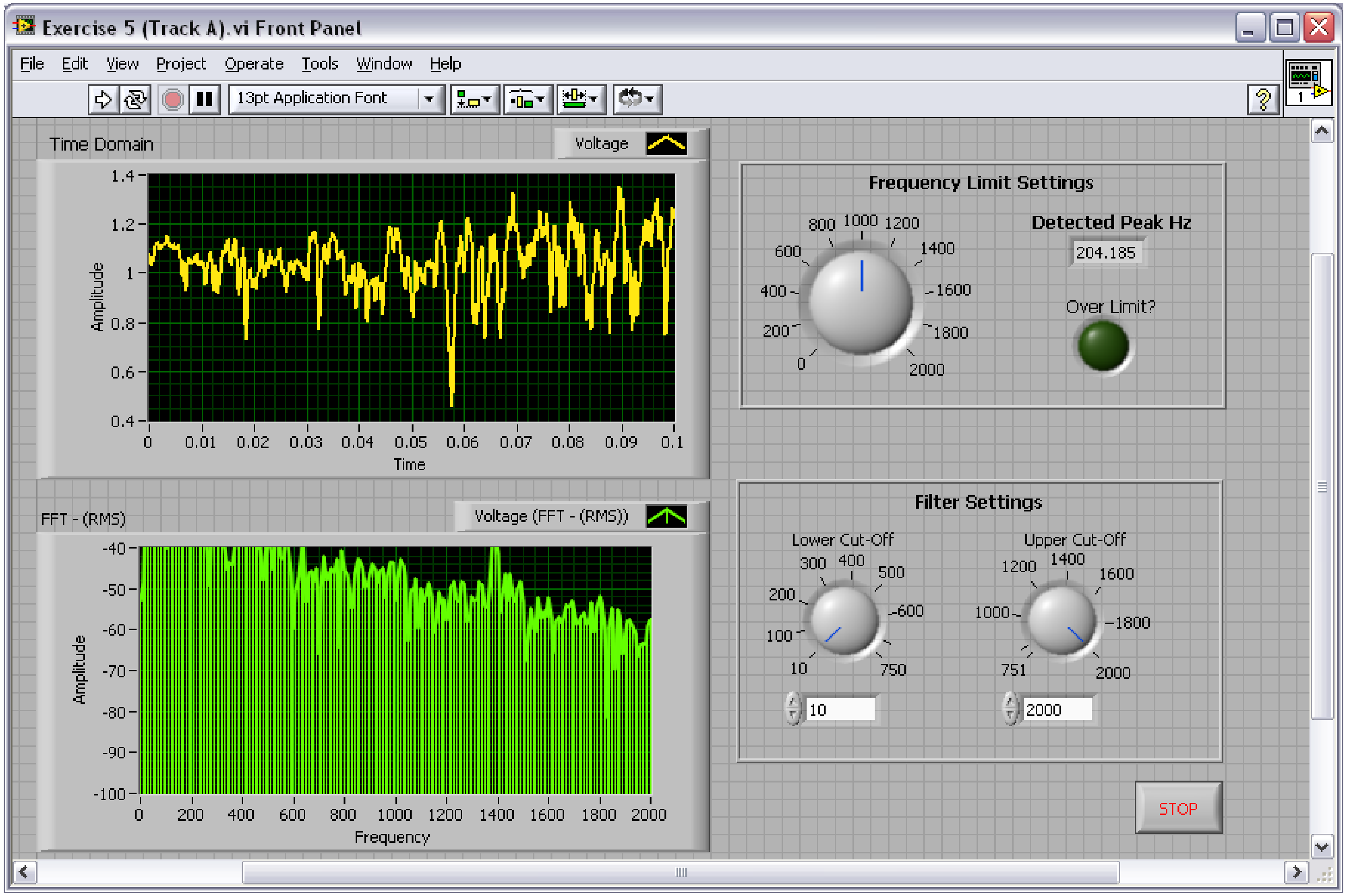
Review of Data Types Found in LabVIEW



Track A,B,&C

Exercise 5 – Apply What You Have Learned

This exercise should take 20 minutes.



Section IV – Advanced Data Flow Topics (optional)

A. Additional Data types

- Cluster

B. Data Flow Constructs

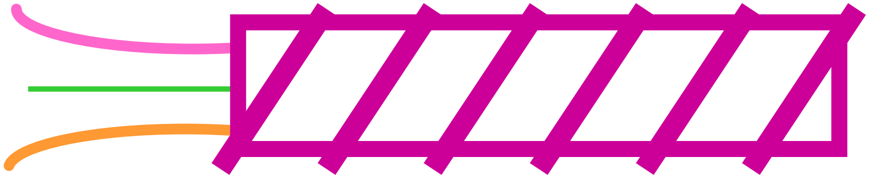
- Shift Register
- Local Variables

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**

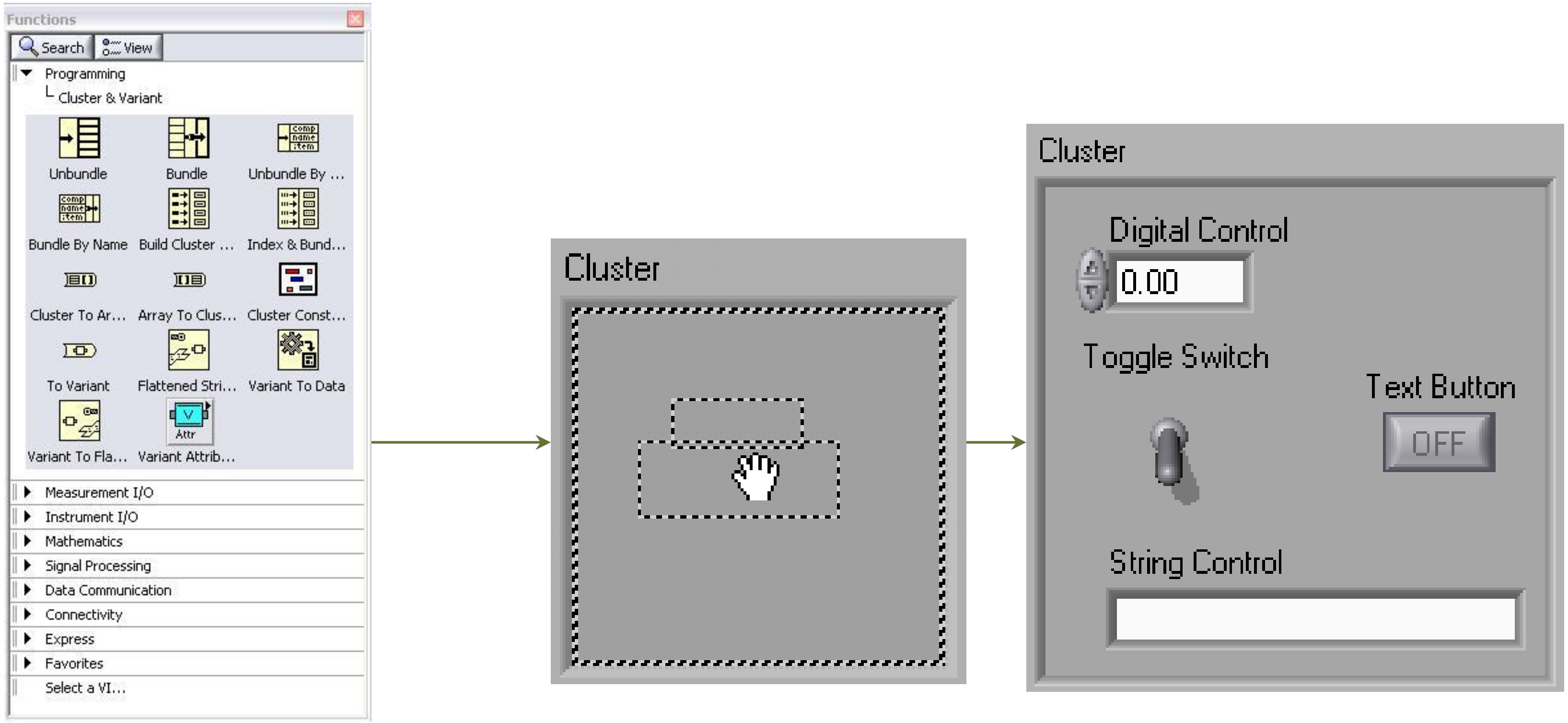


Creating a Cluster

1. Select a **Cluster** shell.

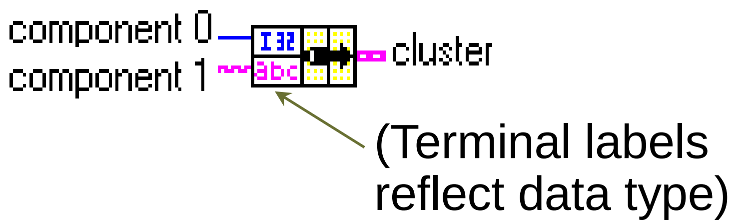
2. Place objects inside the shell.

Controls»Modern»Array, Matrix & Cluster



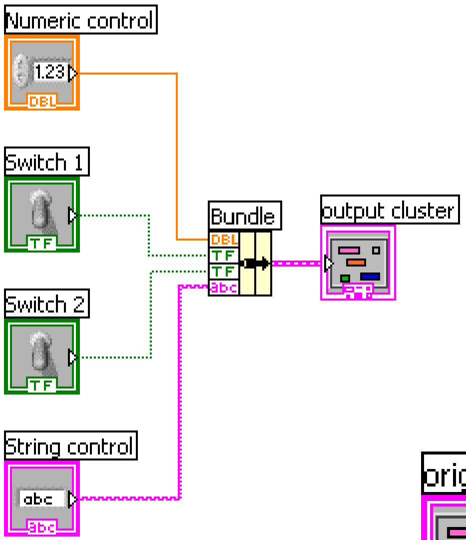
Cluster Functions

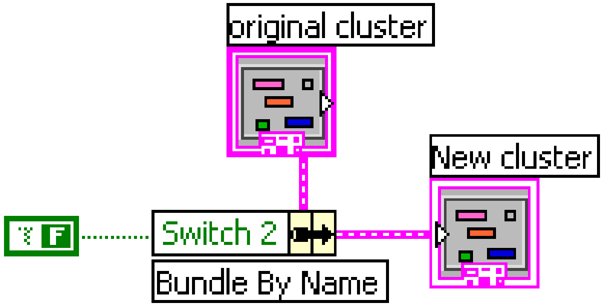
- In the **Cluster & Variant** subpalette of the **Programming** palette
- Can also be accessed by right-clicking the cluster terminal



Bundle

Bundle By Name

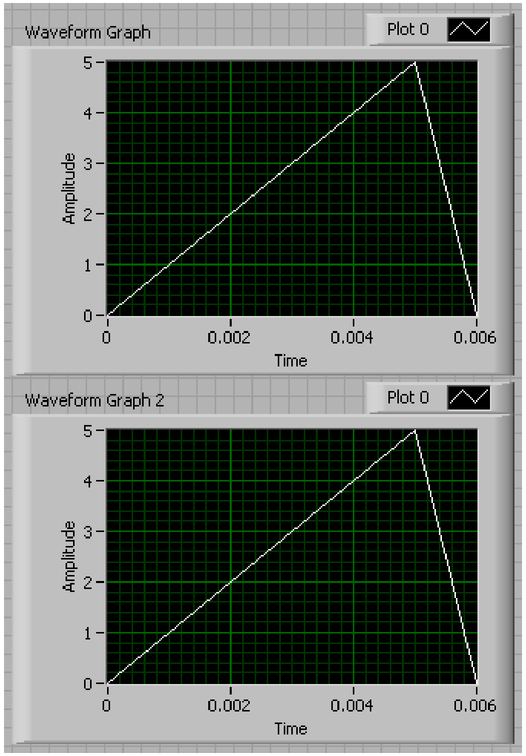




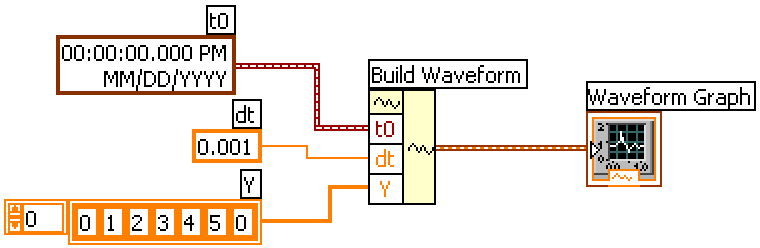
Using Arrays and Clusters with Graphs

The Waveform Datatype contains 3 pieces of data:

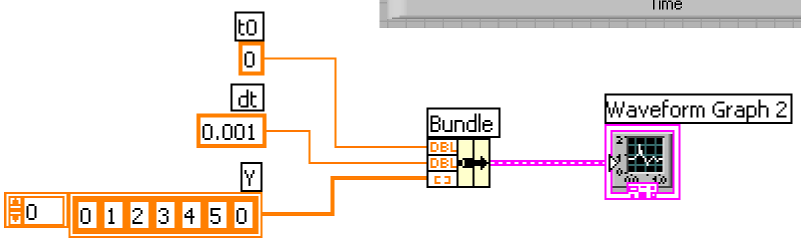
- t0 = Start Time
- dt = Time between Samples
- Y = Array of Y magnitudes



Two ways to create a Waveform Cluster:



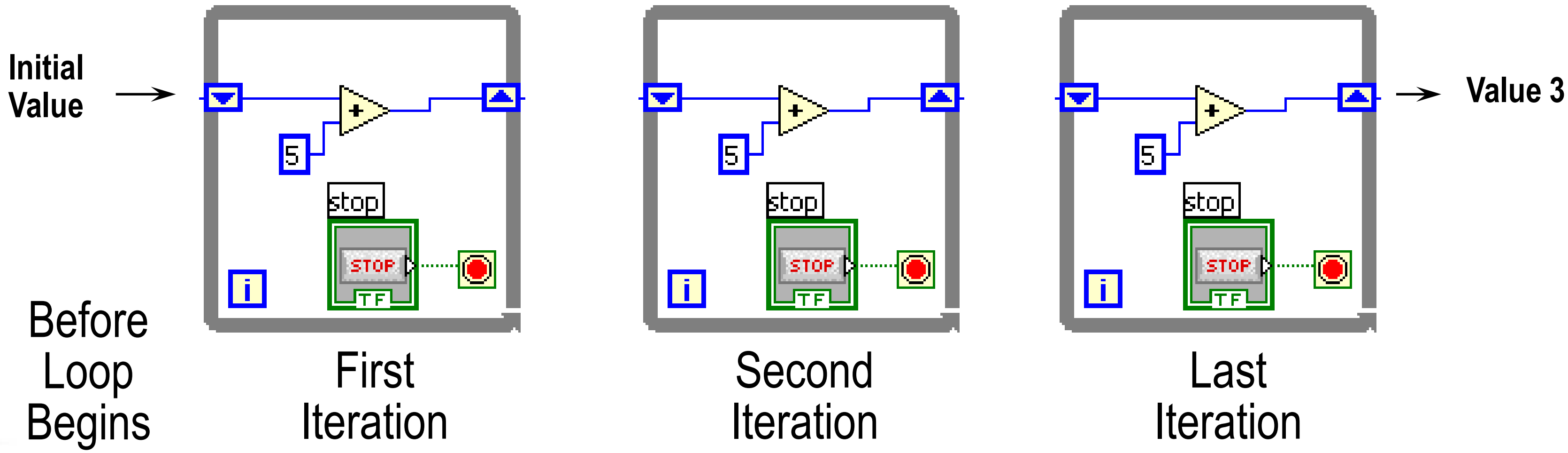
Build Waveform (absolute time)



Cluster (relative time)

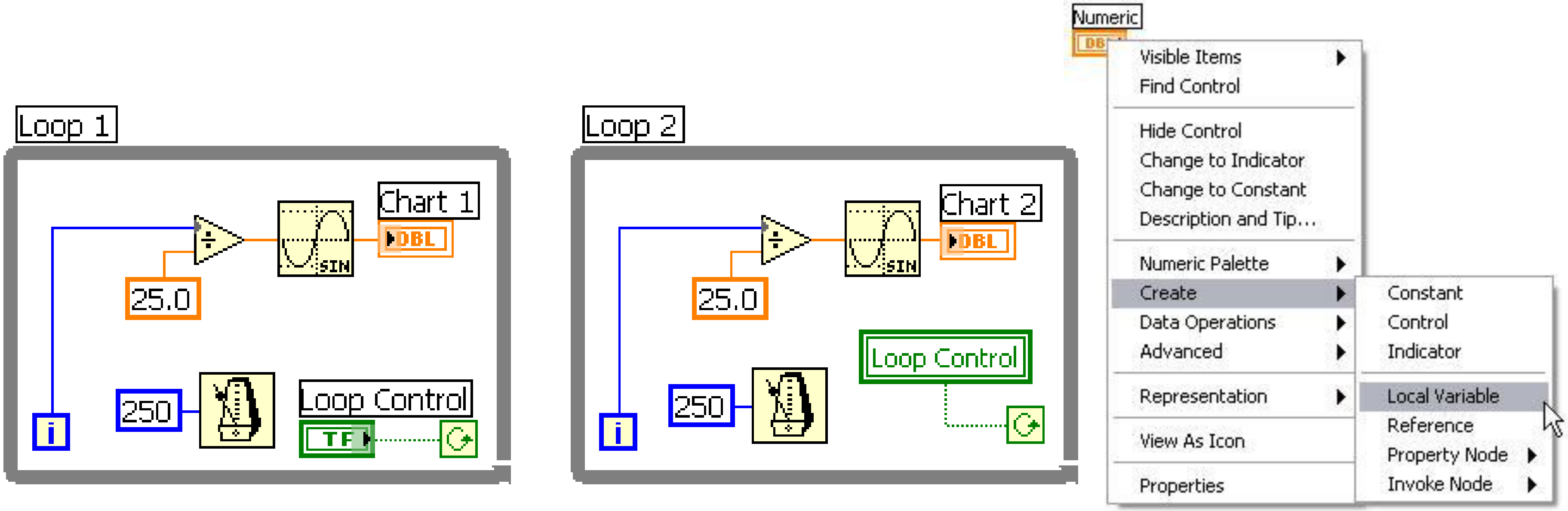
Shift Register – Access Previous Loop Data

- 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

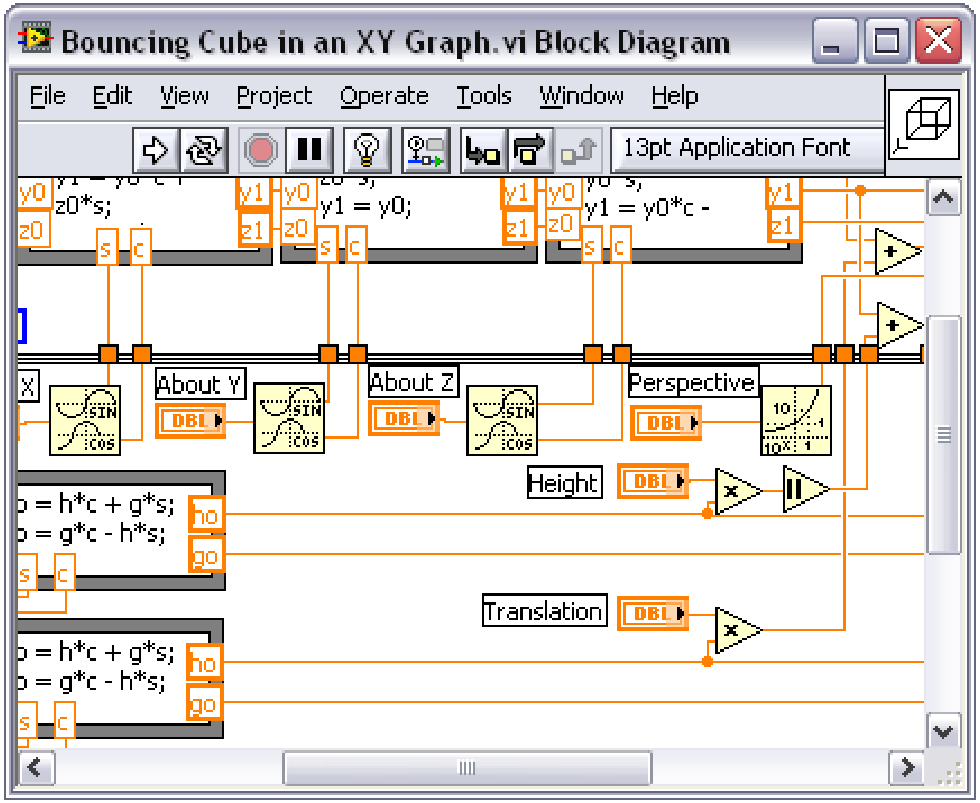


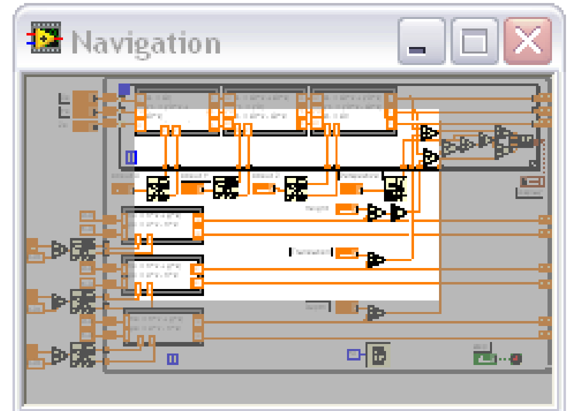
Local Variables

- 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



LabVIEW Navigation Window



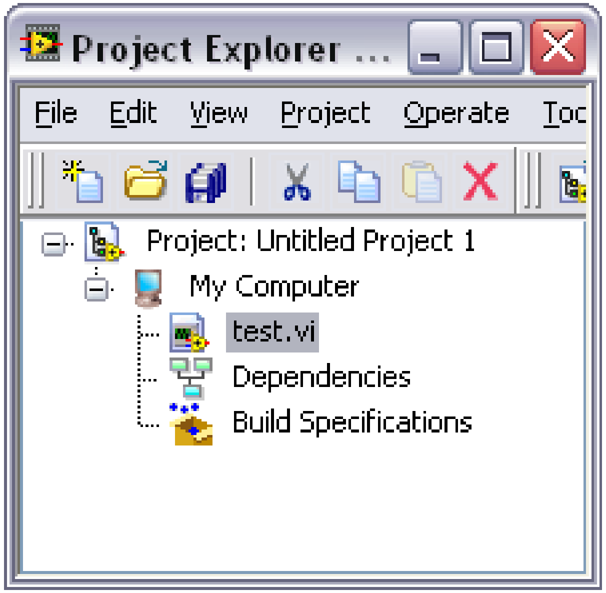


- Shows the current region of view compared to entire Front Panel or Block Diagram
- Great for large programs

* Organize and reduce program visual size with subVIs

LabVIEW Project

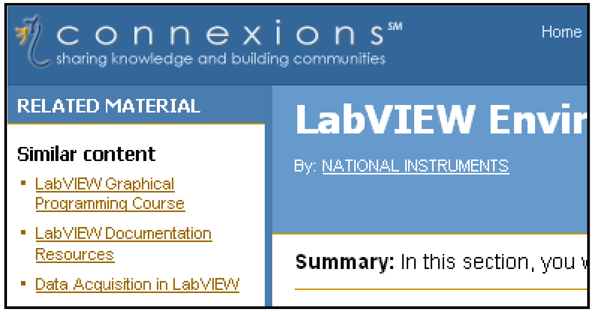
- Group and organize VIs
- Hardware and I/O management
- Manage VIs for multiple targets
- Build libraries and executables
- Manage large LabVIEW applications
- Enable version tracking and management



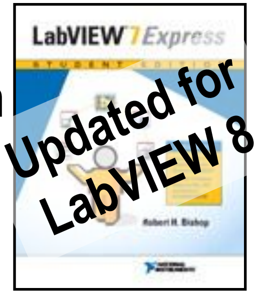
(**LabVIEW»Project»New**)

Additional Resources

- NI Academic Web & Student Corner
  - http://www.ni.com/academic
- Connexions: Full LabVIEW Training Course
  - www.cnx.rice.edu
  - Or search for "LabVIEW basics"



- LabVIEW Certification
  - LabVIEW Fundamentals Exam (free on www.ni.com/academic)
  - Certified LabVIEW Associate Developer Exam (industry recognized certification)
- Get your own copy of LabVIEW Student Edition
  - www.ni.com/academic



By Robert H Bishop.

Published by Prentice Hall.

Section V – Modeling Tools

A. Simulation Diagram - Continuous time

- Simple model (integration)
- Feedback
- Subsystems

B. State Charts (optional)

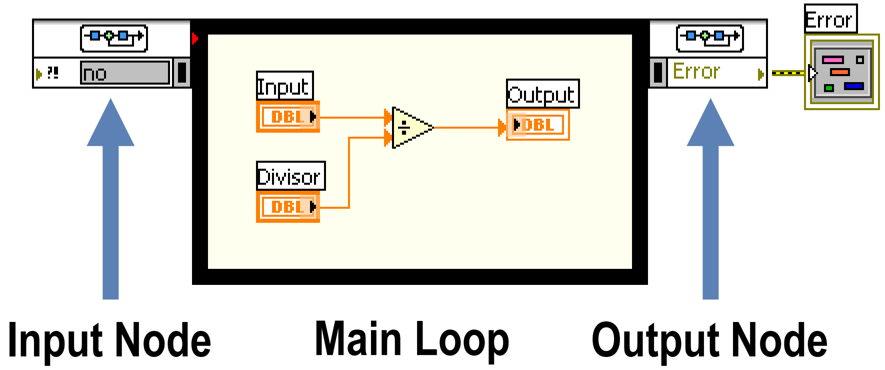
The Design Process

1. **Modeling** – Identify a mathematical representation of the plant
2. **Control Design** – Choose a control method and design a controller
3. **Simulation** – Employ a point-by-point approach to simulate the system timing with a solver
4. **Tuning and Verification** – Introduce real-world nonlinearities, tune, and verify the control algorithm
5. **Deployment** – Implement the finalized control system

LabVIEW Simulation Module

- Develop dynamic systems such as motor controllers and hydraulic simulators with LabVIEW
- Implement your dynamic systems with real-time I/O using built-in LabVIEW data acquisition functions
- Simulate linear, nonlinear, and discrete systems with a wide array of solvers
- Deploy dynamic systems to real-time hardware with the NI LabVIEW Real-Time Module
- Translate models from The MathWorks, Inc. Simulink® into LabVIEW with built-in utility

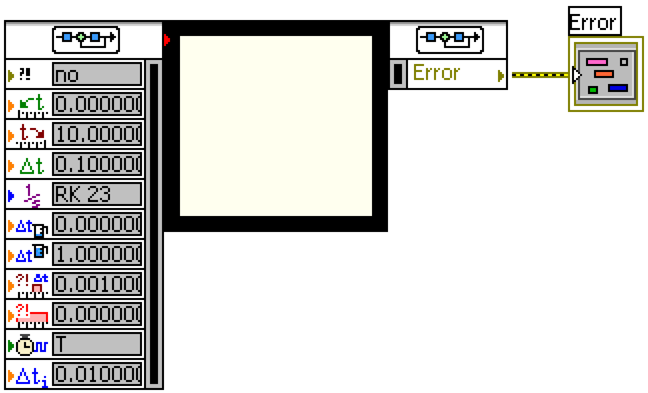
The Simulation Loop



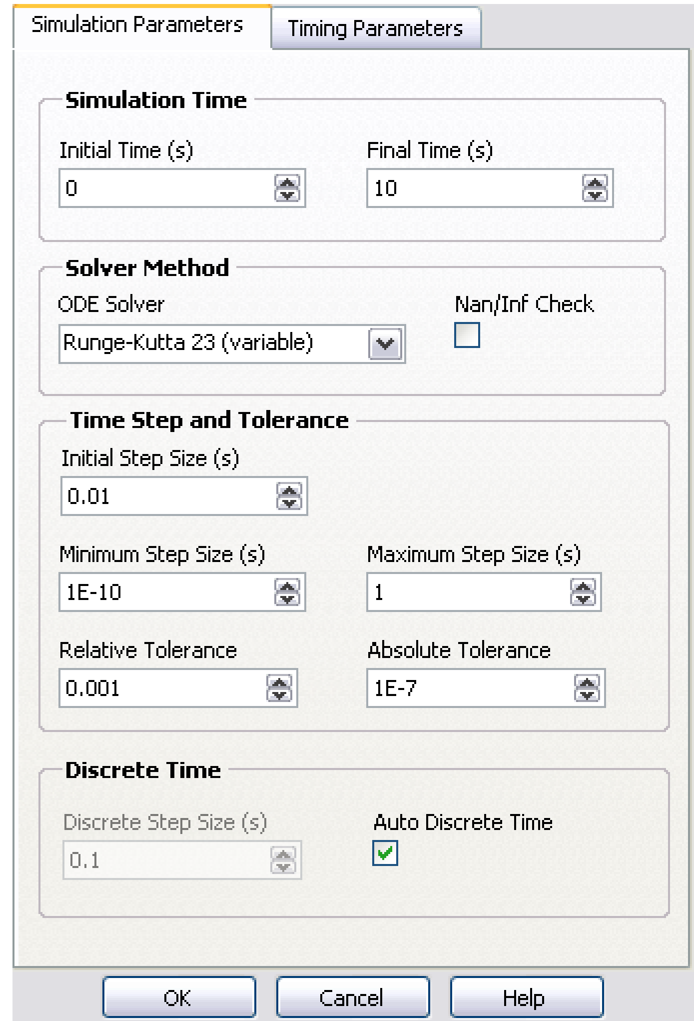
- Built in Differential Equation Solver allows continuous-time system
- Similar to a While Loop with a predefined time period
- Installed with Simulation Module
- Double-click Input Node to configure simulation parameters
- Create an indicator on the Output Node to display Simulation errors

Simulation Loop Parameters

- Drag left node to show current parameters and provide inputs for run-time simulation configuration



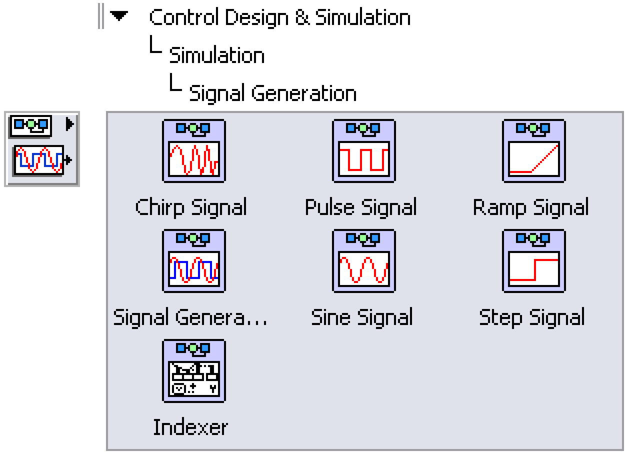
- Double-click Input Node to configure simulation parameters



Generating Simulation Input

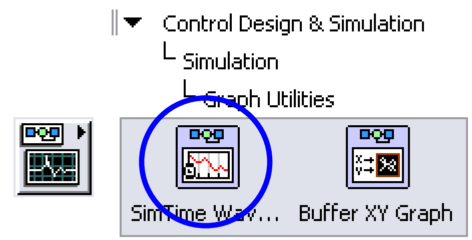
Simulations can utilize a wide variety of signal sources:

- Simulated Signals
  - Step Input
  - Impulse
  - Front Panel User Input
- Real World signals
  - Data Acquisition Hardware

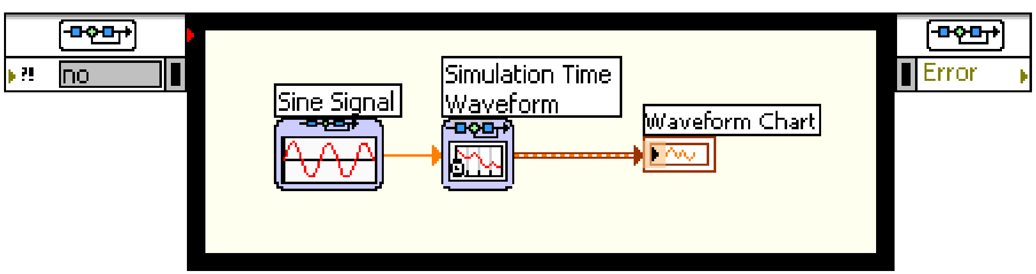


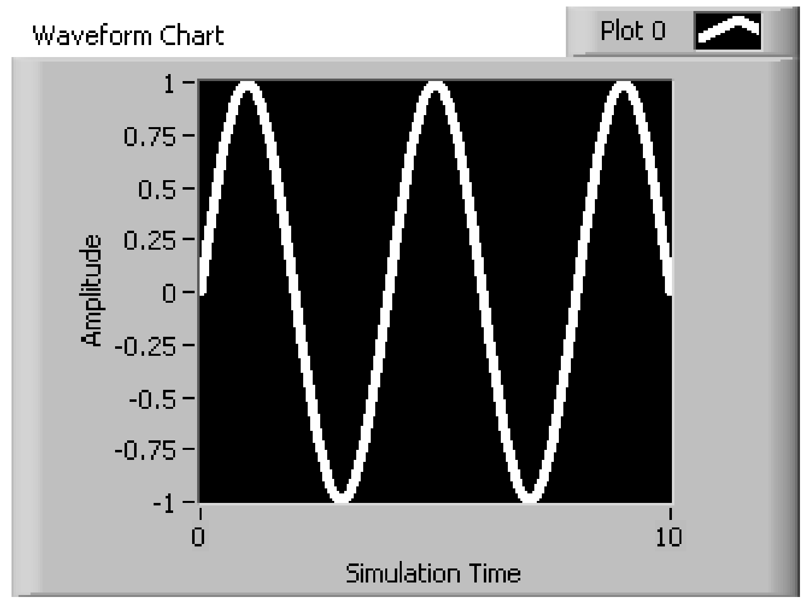
Capturing Simulation Output

- Use the Graph Utilities functions to plot one or more signals

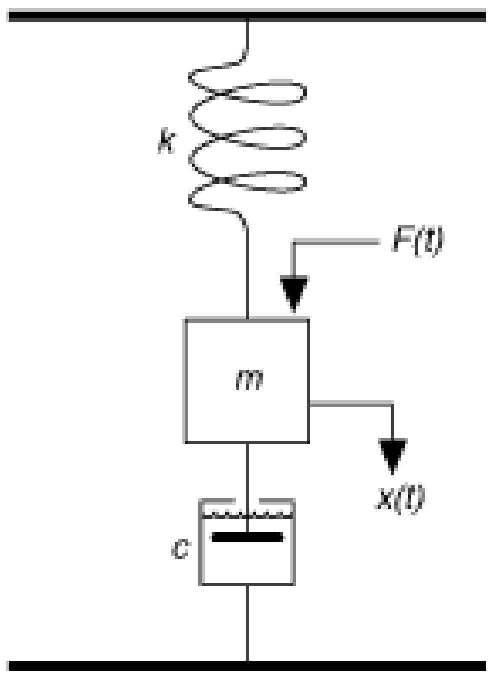


- Plots are updated as the Simulation Loop executes





Exercise 6: Compute and view the position *x*(*t*) of the mass

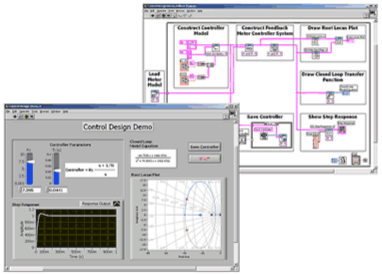


- $F(t) – cx'(t) – kx(t) = mx''(t)$
- *c* is the damping constant of the spring
- *k* is the stiffness of the spring

- Construct a simulation diagram that iterates the following steps over a period of time.
- Divide a known force by a known mass to calculate the acceleration of the mass.
- Integrate acceleration to calculate the velocity of the mass.
- Integrate velocity to calculate the position of the mass.
- Iterate over different stiffness values to see effect

Where Can I Learn More?

We have only begun to explore the many opportunities for control and simulation within LabVIEW. Learn more by visiting the following links:



System Identification Toolkit:
- http://sine.ni.com/nips/cds/view/p/lang/en/nid/13853

Control Design Toolkit:
- http://sine.ni.com/nips/cds/view/p/lang/en/nid/13854

Simulation Module:
- http://sine.ni.com/nips/cds/view/p/lang/en/nid/13852

LabVIEW Real-Time Module:
- http://www.ni.com/realtime

Data Acquisition and Control Hardware:
- http://www.ni.com/dataacquisition

CompactRIO Real-Time Platform:
- http://www.ni.com/compactrio

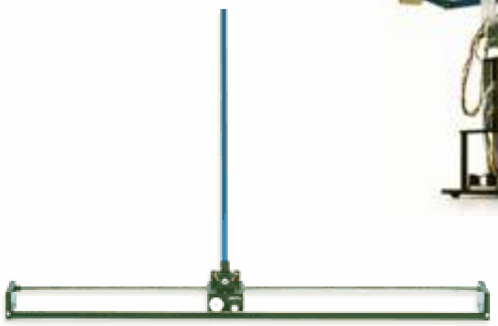
Educational Control Partners

Quanser – www.quanser.com

- LabVIEW based curriculum and solutions
- Linear, rotary, mechatronic and specialty control experiments
- Uniquely modular, allowing multiple configurations for a wide range of experiments

Quanser QNET – 010
DC Motor Control

Quanser QNET – 011
Rotary Inverted Pendulum



Modular
Linear Pendulum

3 Degree of
Freedom
Helicopter

Educational Control Partners

Educational Control Products (ECP) – www.ecpsystems.com

- LabVIEW control templates
- Intuitive systems provide unparalleled flexibility and dynamic fidelity
- In use at over 400 universities and industrial sites world-wide
- Proven to accelerate student learning while saving instructor time

ECP Model 220 Industrial Plant

ECP Model 730 Magnetic Levitation

ECP Model 750 Gyroscope

ECP Model 205 Torsional Plant

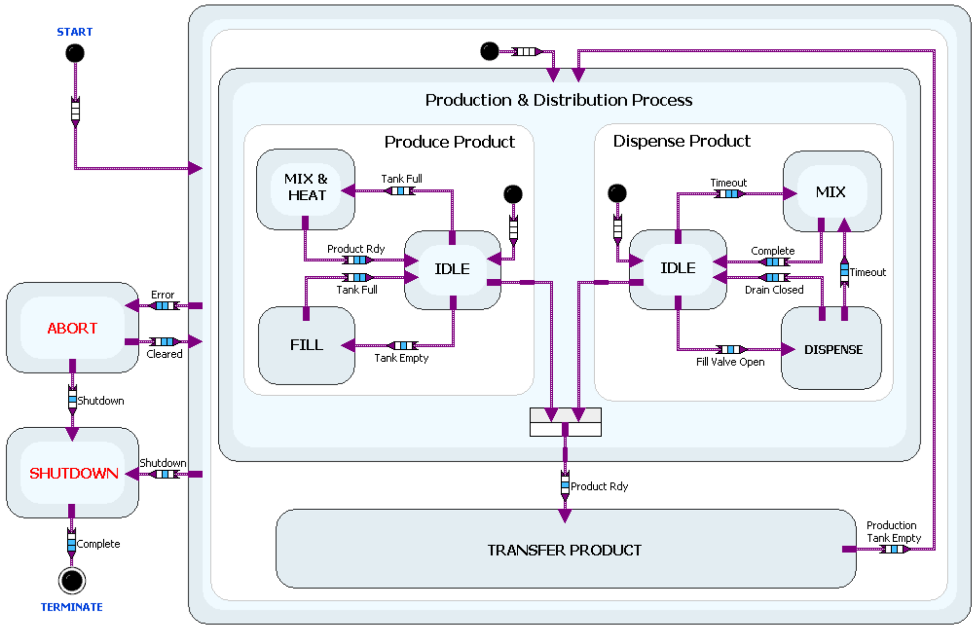
Additional Resources

- NI Academic Controls Web
  - http://www.ni.com/academic/controls
- LabVIEW Student Edition DVD with Control Design and Simulation
  - http://www.academicsuperstore.com/ search: LabVIEW
  - Part Number: 752412

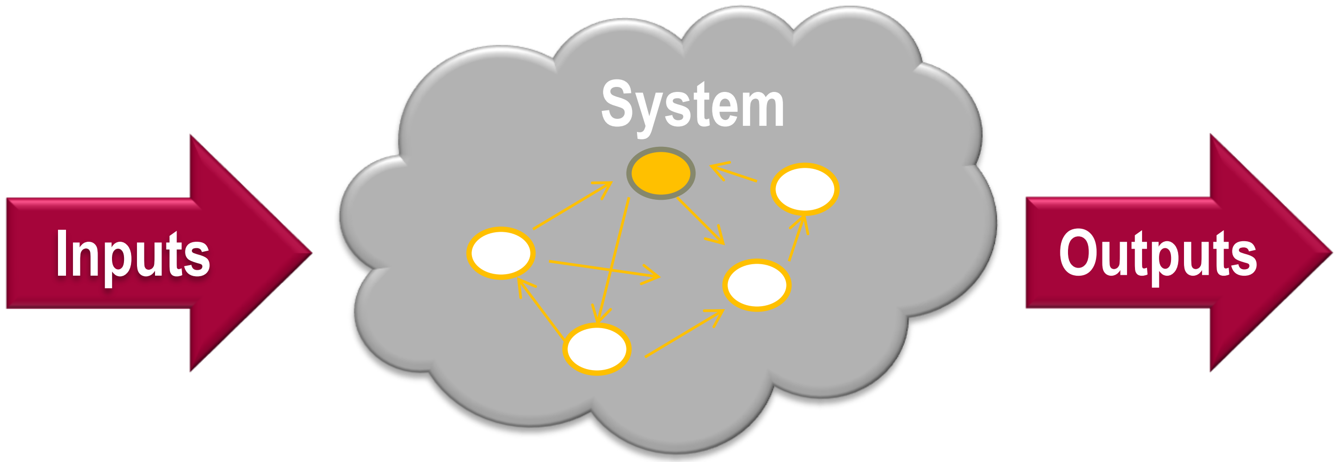
- Connexions: Full LabVIEW Introductory Course
  - www.cnx.rice.edu
  - Or search for "LabVIEW basics"

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

Developing Applications with the NI LabVIEW Statechart Module



What are Statecharts?

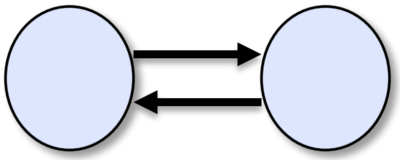


Statecharts are visual representations of reactive (event-based) systems.

Differences between Statecharts and FSMs

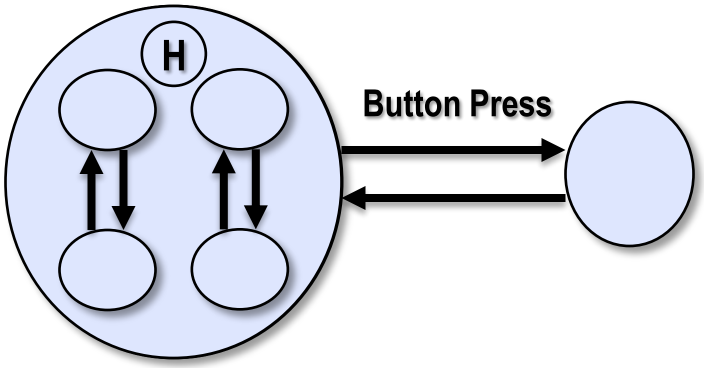
Both contain the same basic concepts:

- States
- Transitions



Statechart adds additional concepts:

- Hierarchy
- Concurrency
- Event-based paradigm
- Pseudostates & Connectors



Based on the UML statechart diagram specification

Reactive Systems

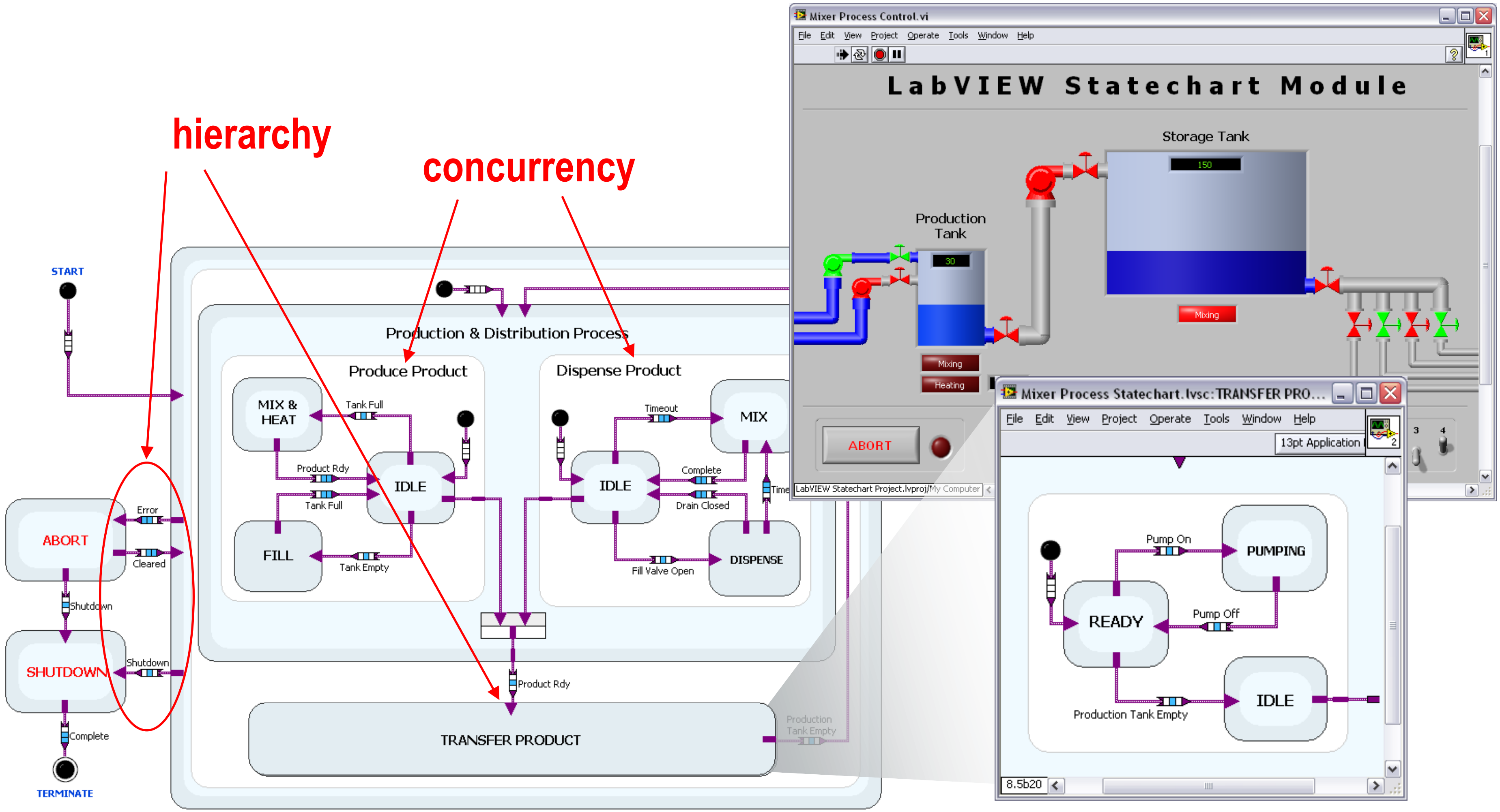
- Communication systems
- Digital protocols
- Control applications
  - Sequential logic
  - Batch processing
  - Event response
  - Non-linear control
- User-interface implementation
- System modeling for virtual prototyping (simulation)

State Machines

Statechart Benefits

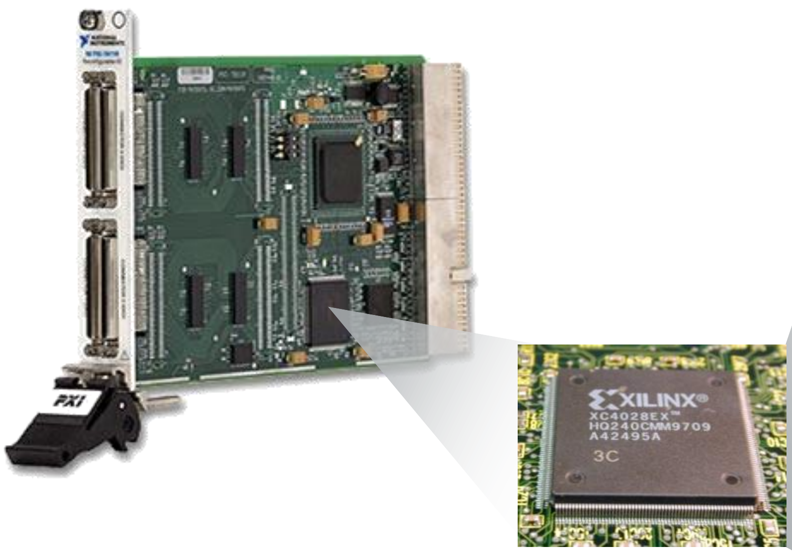
- Abstraction
  - Simple semantics to represent complex systems
  - System-level view
  - Self-documenting

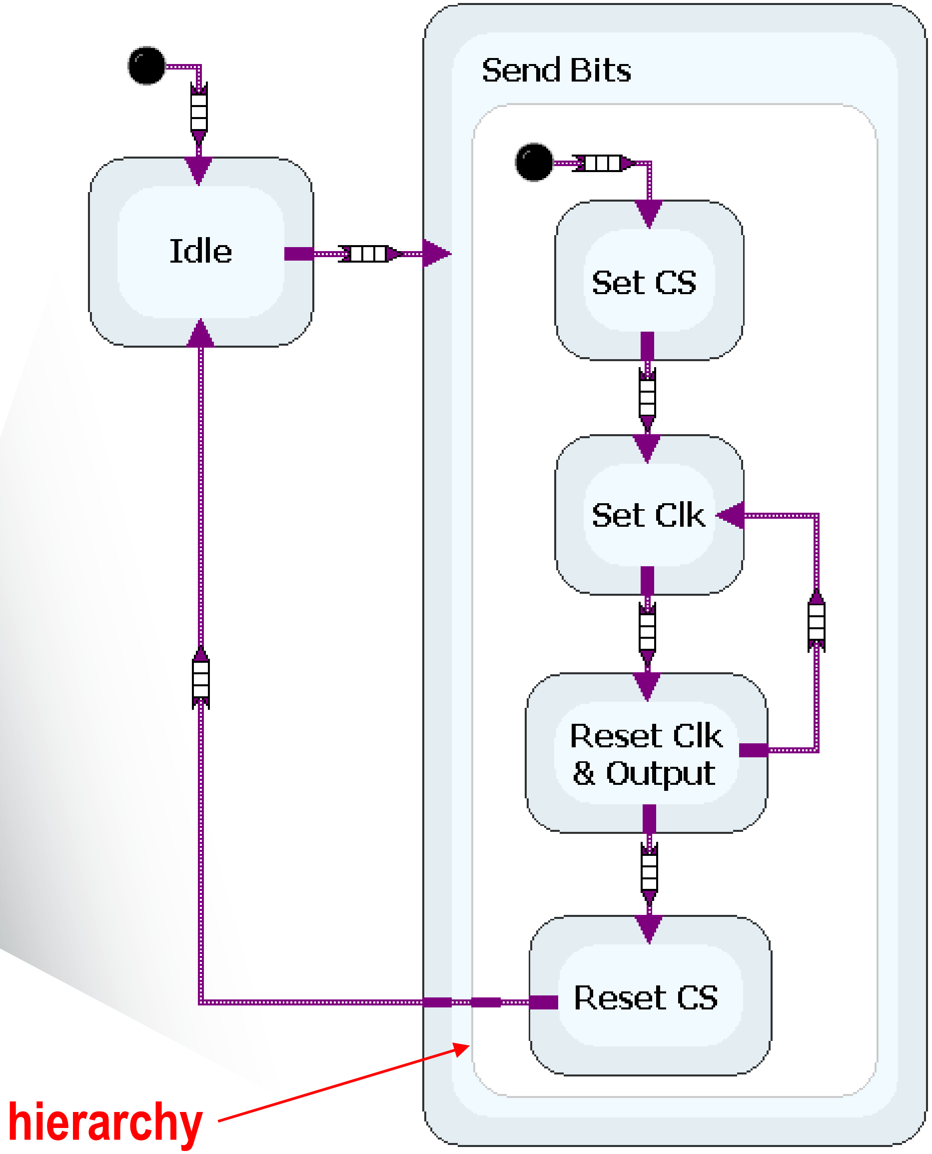
Machine & Process Control



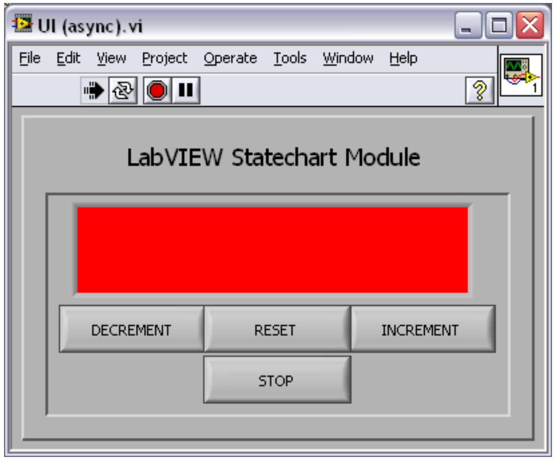
NATIONAL INSTRUMENTS

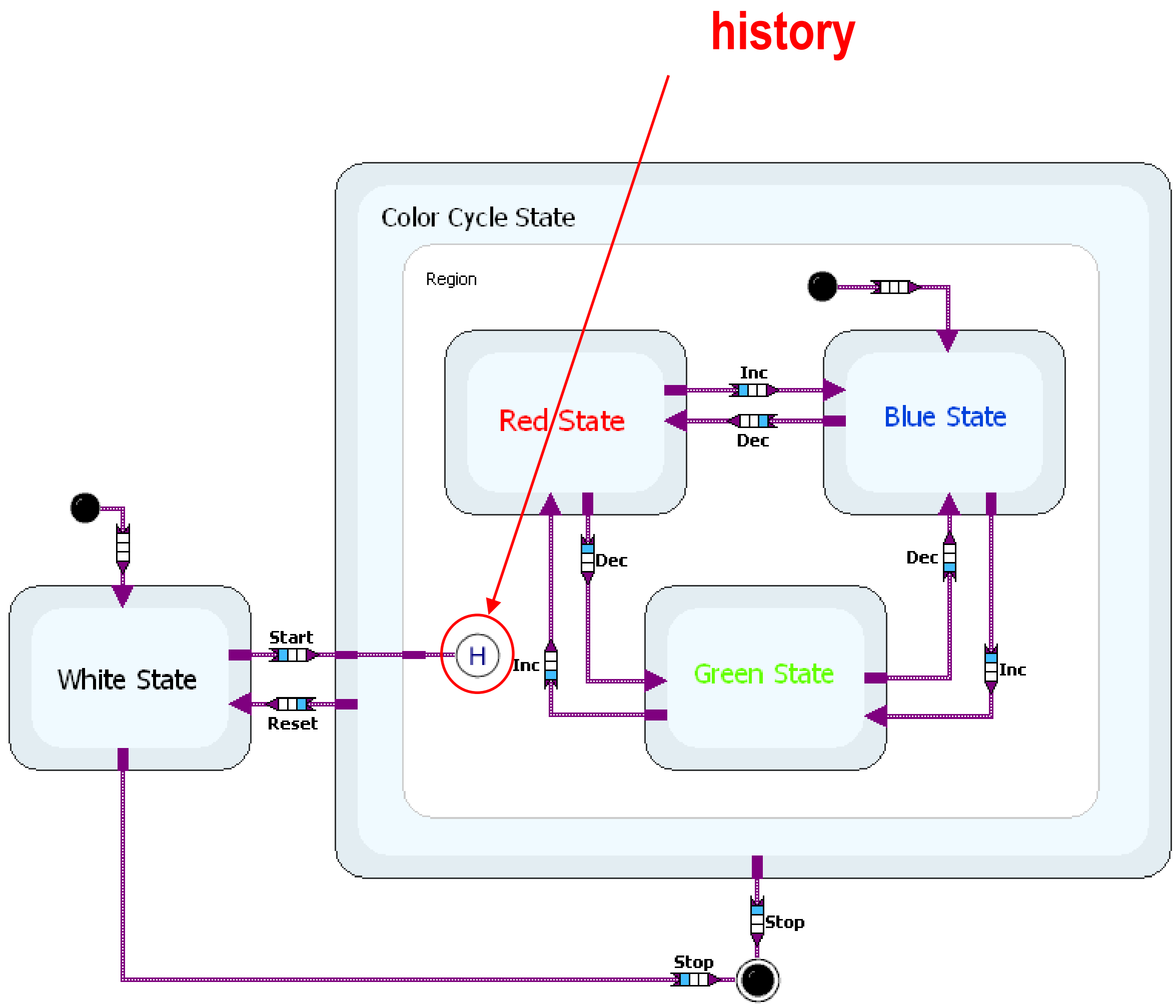
FPGA Logic


PXI
XILINX
XC4028EX
HQ240CMM9709
A42495A
3C


Idle
Send Bits
Set CS
Set Clk
Reset Clk
& Output
Reset CS
hierarchy

User Interfaces



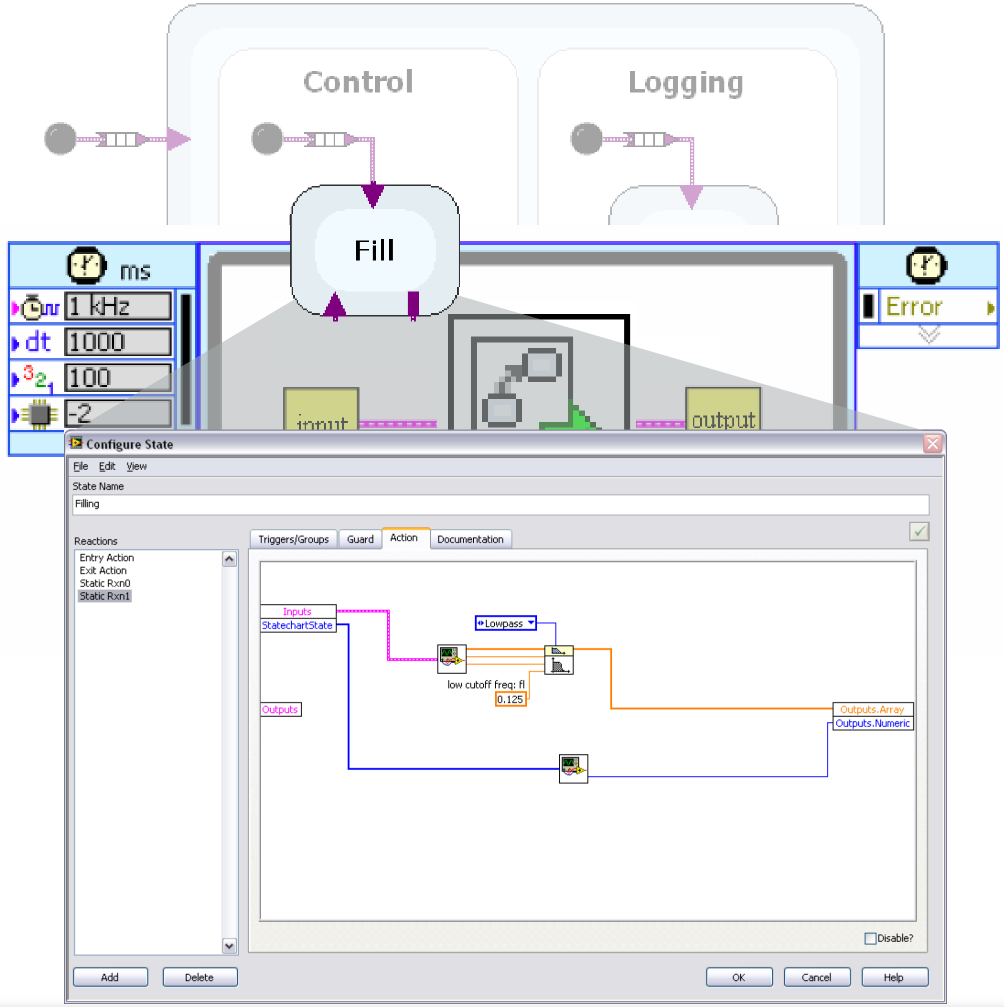


Statechart Benefits

- Abstraction
  - Simple semantics to represent complex systems
  - System-level view
  - Self-documenting
- Scalability
  - Easily extend applications
  - Open software platform
- Automatic Code Generation
  - LabVIEW Embedded Technology

LabVIEW Statechart Development

1. Build statechart
2. Define transitions and states
3. Generate statechart subVI
4. Place in LabVIEW block diagram



NATIONAL INSTRUMENTS

Example – Ceiling Fan

- Triggers
  - Power switch
  - Fan toggle
  - Light toggle
- Outputs
  - Light
  - Fan speed

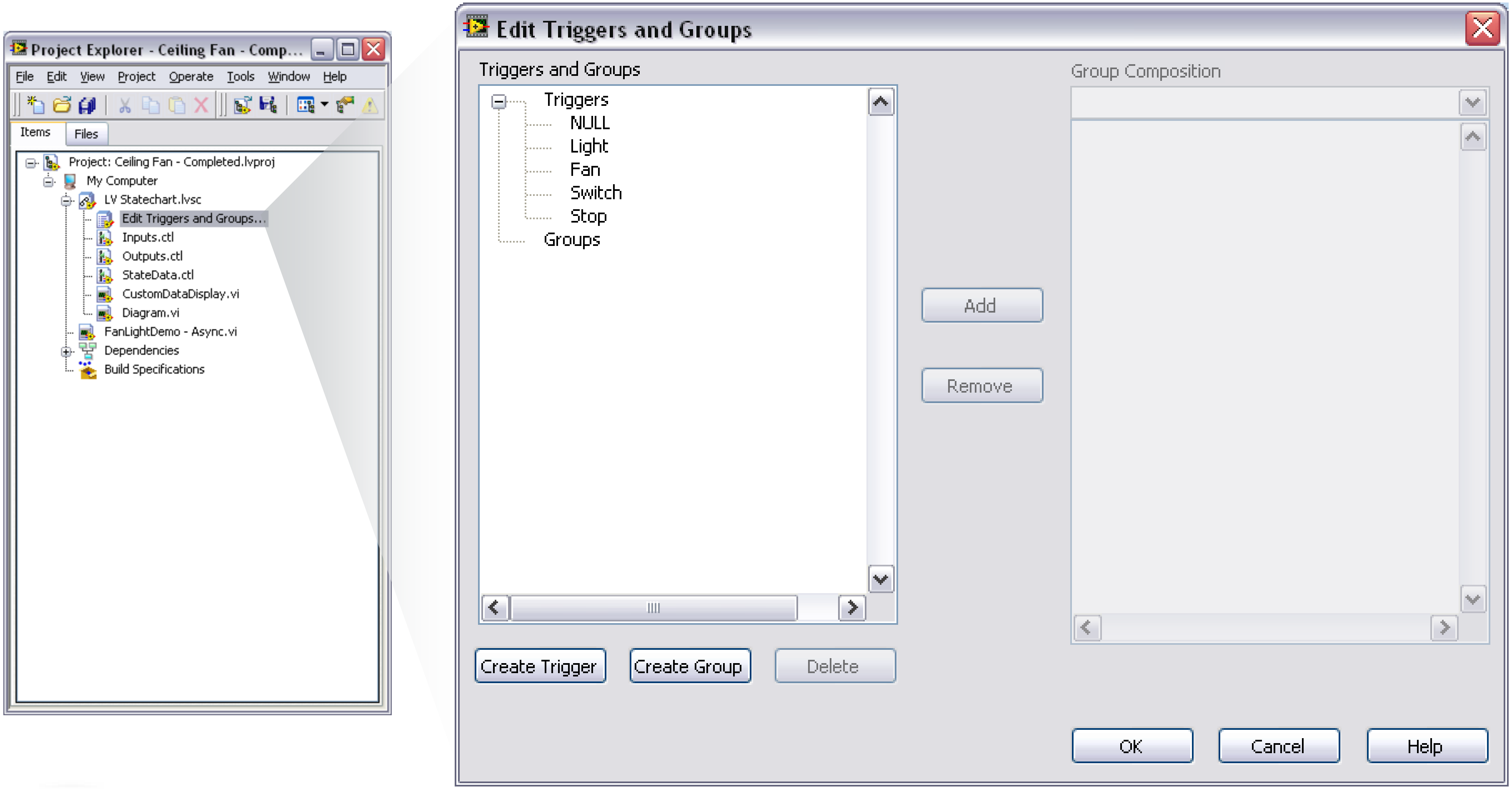
<table>
<tr><th colspan="2">Power</th><th colspan="2">No Power</th></tr>
<tr><td>Fan</td><td>Light</td><td>Fan</td><td>Light</td></tr>
<tr><td>High</td><td rowspan="2">on</td><td colspan="2" rowspan="4">off</td></tr>
<tr><td>medium</td></tr>
<tr><td>low</td><td rowspan="2">off</td></tr>
<tr><td>off</td></tr>
</table>

Example – Ceiling Fan

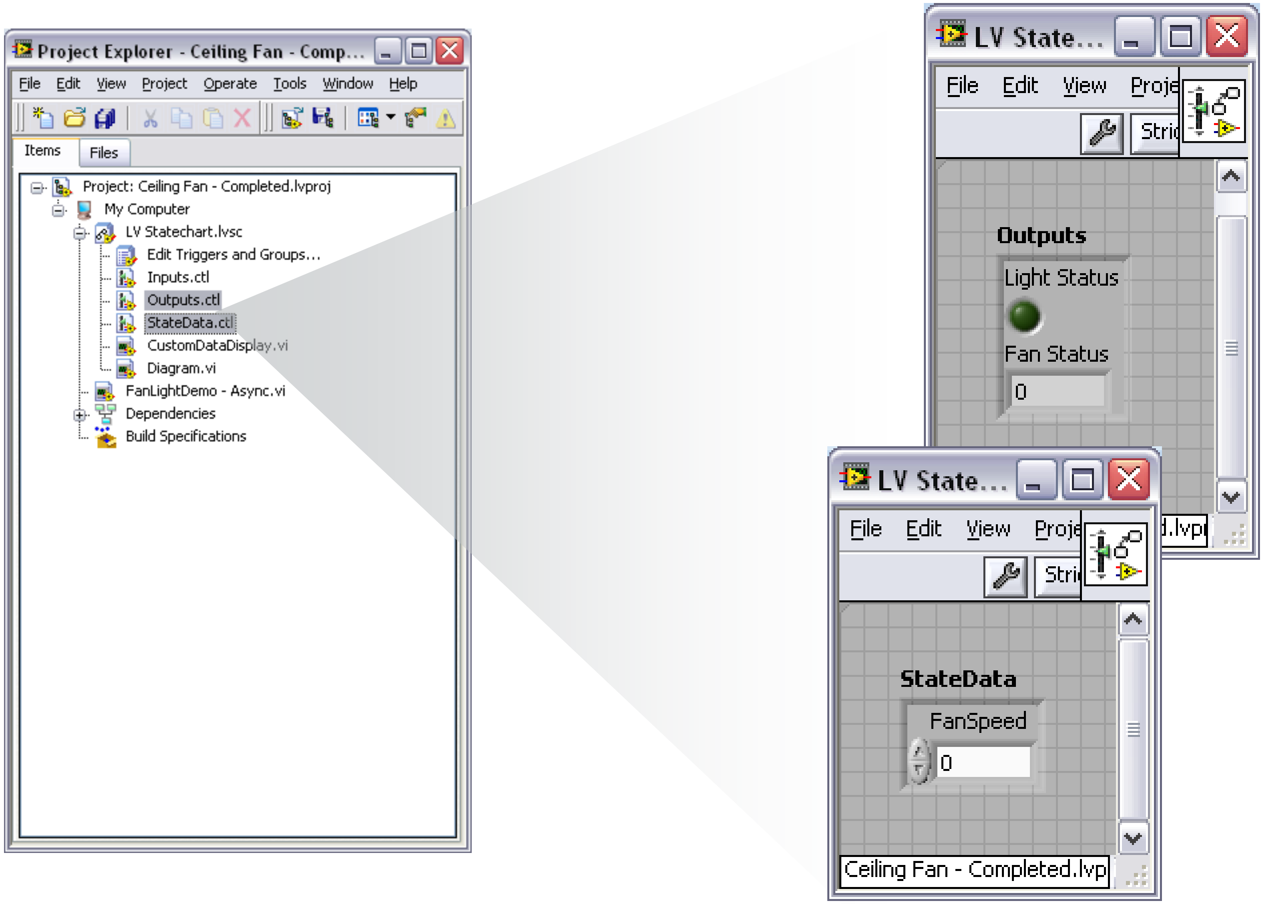
- Triggers
  - Power switch
  - Fan toggle
  - Light toggle
- Outputs
  - Light
  - Fan speed
- Internal Data
  - Fan Speed

<table>
<tr><th colspan="2">Power</th><th colspan="2">No Power</th></tr>
<tr><td>Fan</td><td>Light</td><td>Fan</td><td>Light</td></tr>
<tr><td>on</td><td>on</td><td colspan="2" rowspan="2">off</td></tr>
<tr><td>off</td><td>off</td></tr>
</table>

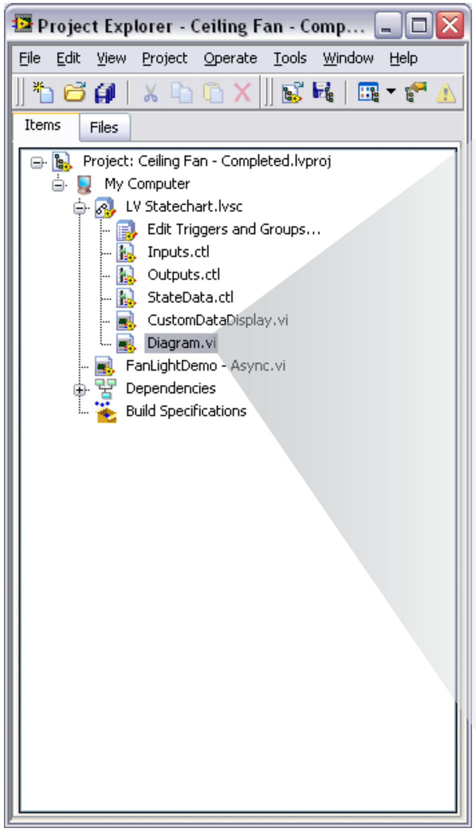
1. Build Statechart

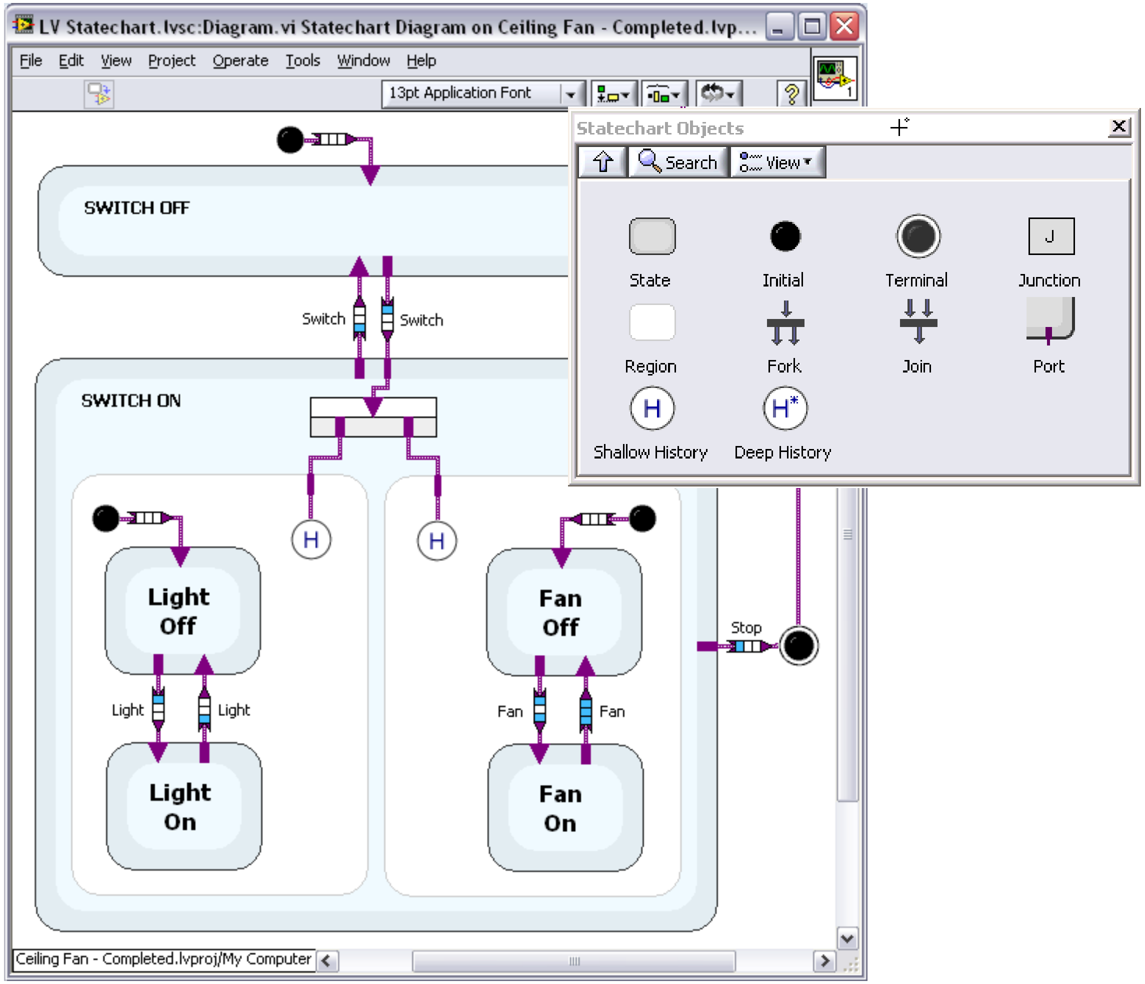


1. Build Statechart



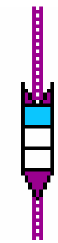
1. Build Statechart





2. Define Transitions and States

- Each Transition contains three components
  - Trigger – events that cause a transition
  - Guard – logic that can prevent a transition
  - Action – what happens when you transition

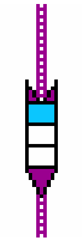


If the doorbell rings and an adult is home, answer the door.

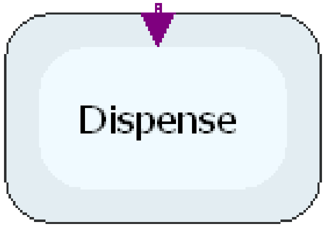
```
Curr State - DOOR CLOSED
 Trigger   - doorbell ring
  Guard     - adult home?
   Action    - open door
 New State  - DOOR OPEN
```

2. Define Transitions and States

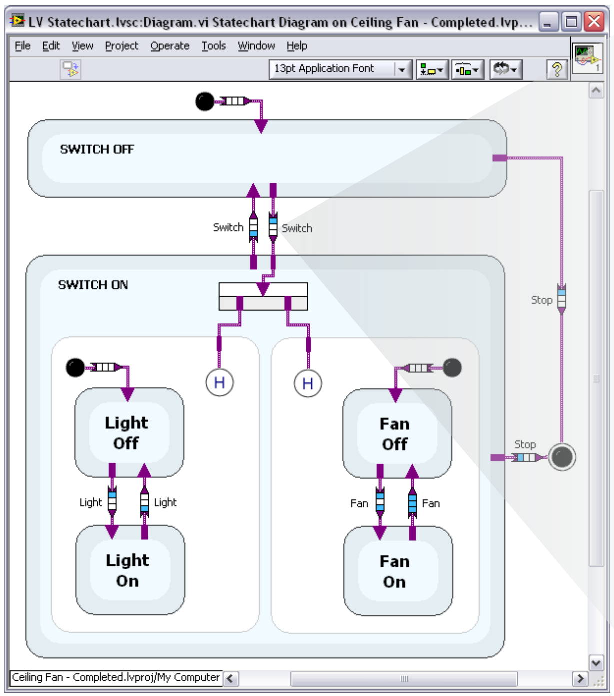
- Each Transition contains three components
  - Trigger – events that cause a transition
  - Guard – logic that can prevent a transition
  - Action – what happens when you transition



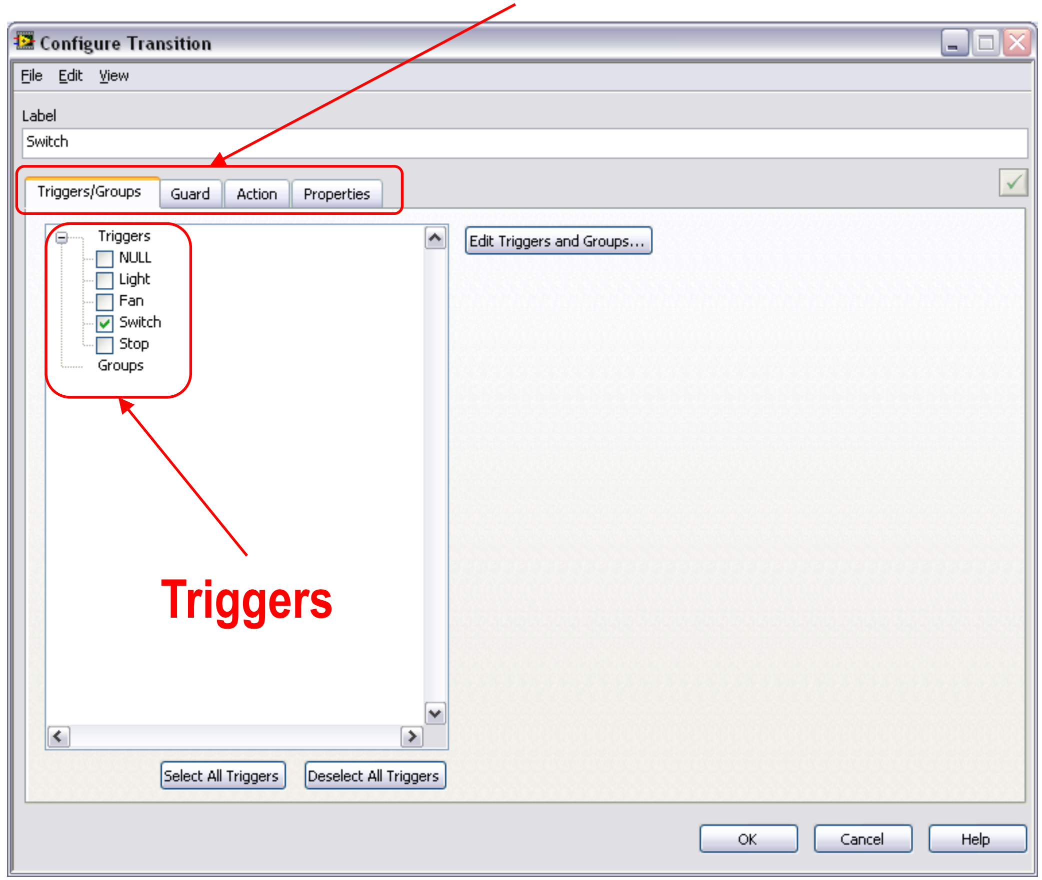
- Each state contains three types of actions
  - Entry – what happens when you get there
  - Exit – what happens when you leave
  - Static – what happens while you are there



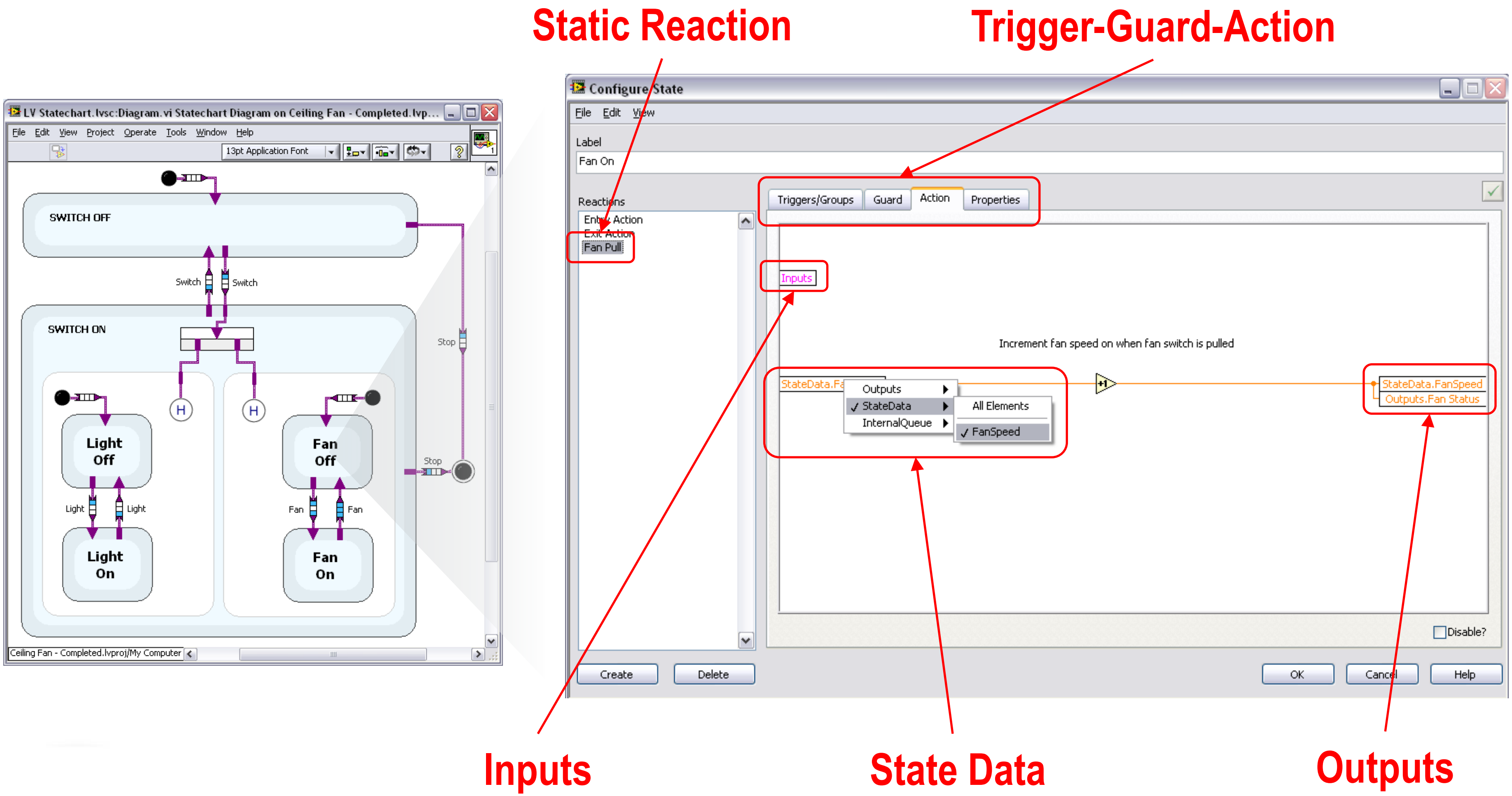
2. Define Transitions and States



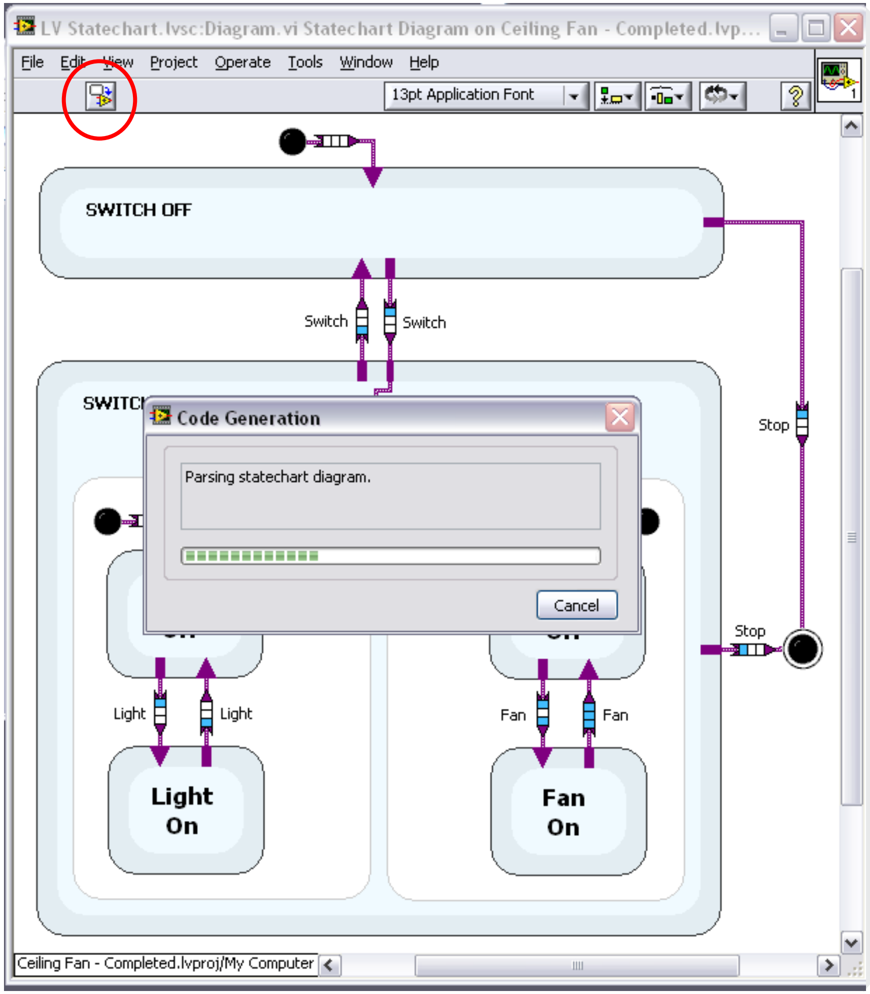
Trigger-Guard-Action

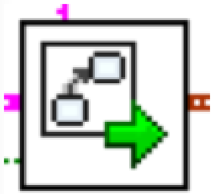


2. Define Transitions and States

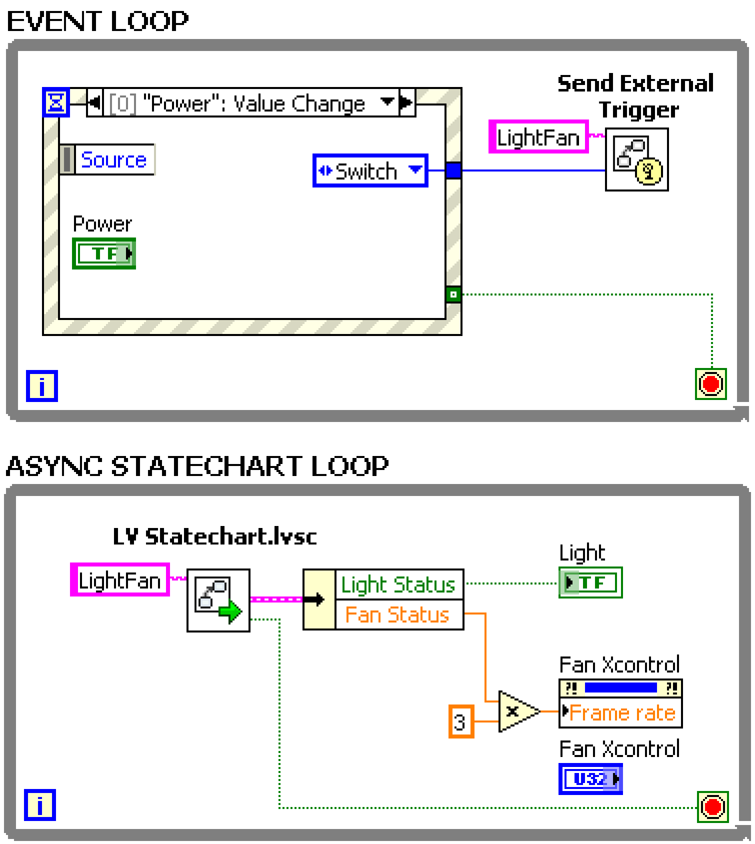


3. Build Statechart SubVI





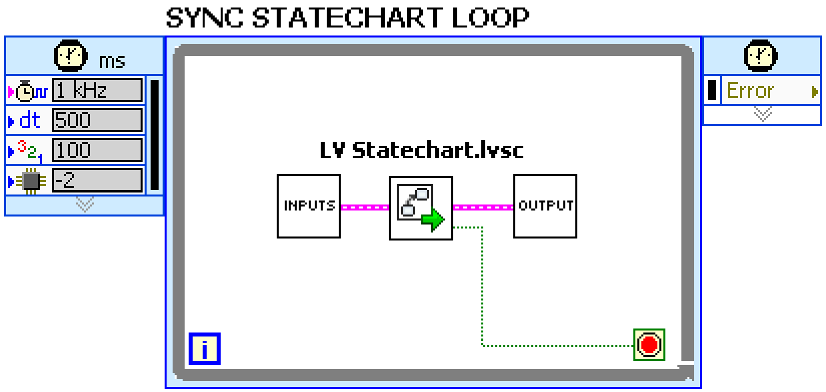
4. Place in LabVIEW Block Diagram



Asynchronous Usage

- User interface
- Interruption handling
- Modeling event driven systems

4. Place in LabVIEW Block Diagram

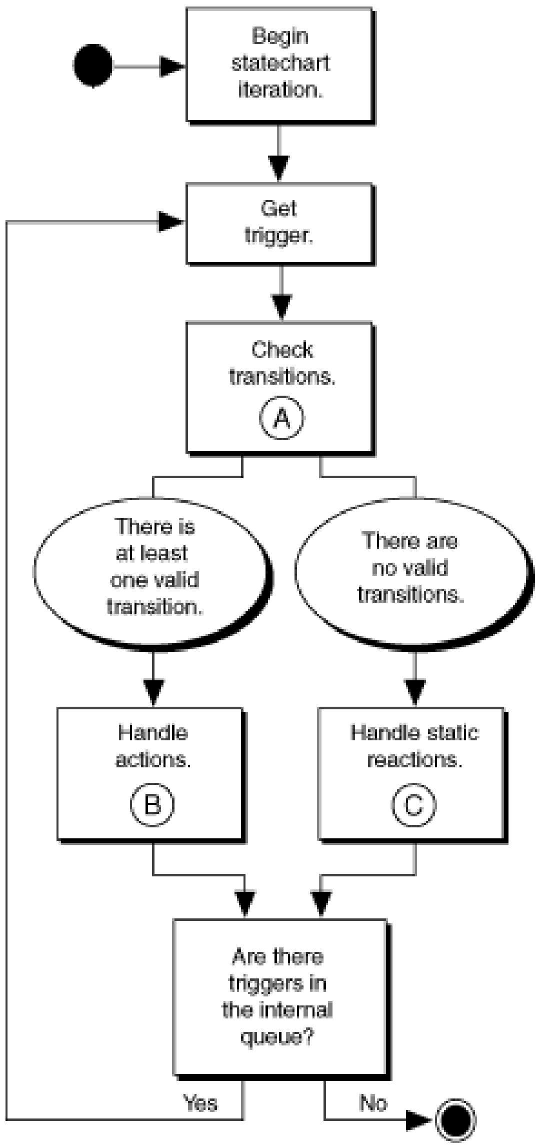


Synchronous Usage

- Embedded applications
- Communication protocols
- Control implementations

Statechart Execution

- Evaluate the trigger/guard logic for the transitions leaving the current state(s)
- On first valid transition:
  - Execute the exit action(s) for the current state(s)
  - Execute the transition action
  - Execute the entry action(s) for all state(s) being transitioned to
- If no transitions are valid:
  - Evaluate the trigger/guard logic for all static reactions configured for the current state
  - Execute the action code for all valid reactions



DEMO

What to do next?

- Visit ni.com/statechart
  - Demo videos
  - Statecharts 101 whitepaper
  - Statecharts with LabVIEW FPGA whitepaper
  - Try the LabVIEW Statechart Module online
- Demonstration from local Field Engineer

Section VI – Targets and Deployment

A. LabVIEW Real-time

B. LabVIEW FPGA

C. LabVIEW Microprocessor SDK