

# BE/EE189 Design and Construction of Biodevices

## Lecture 1



# LabVIEW Programming – Basics

- Virtual instrument and LabVIEW
- The LabVIEW development environment
- Basic programming with LabVIEW
- Navigation window
- Property nodes
- Cleaning up the block diagram



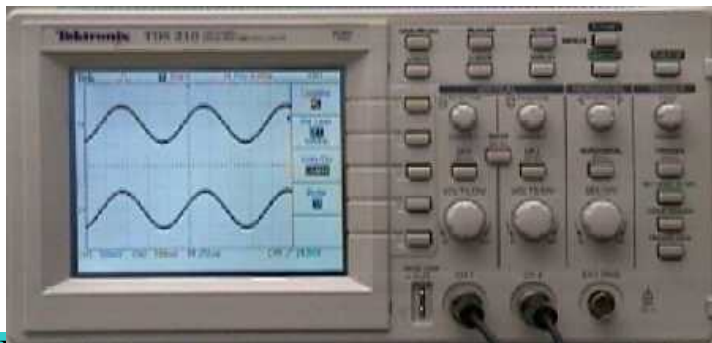
# Virtual Instrument

- Virtual instrumentation can be defined as:
  - *A layer of software and/or hardware added to a general purpose computer in such a fashion that users can interact with the computer as though it were their own custom-designed traditional electronic instrument.*

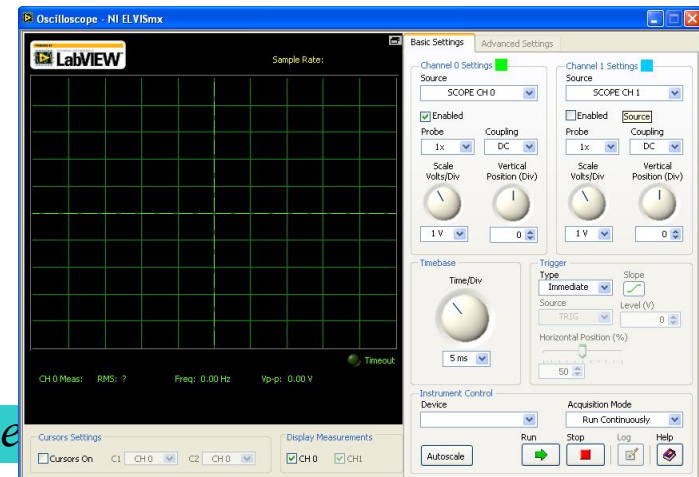
*From Virtual Bio-instrumentation by Jon B. Olansen and Eric Rosow*

- LabVIEW programs are called virtual instruments (VIs).

Real Oscilloscope



Virtual Oscilloscope



# LabVIEW Graphical Development System

- Graphical programming environment – different from text-based programming language such as C or Fortran.
- Compile code for multiple OS and devices





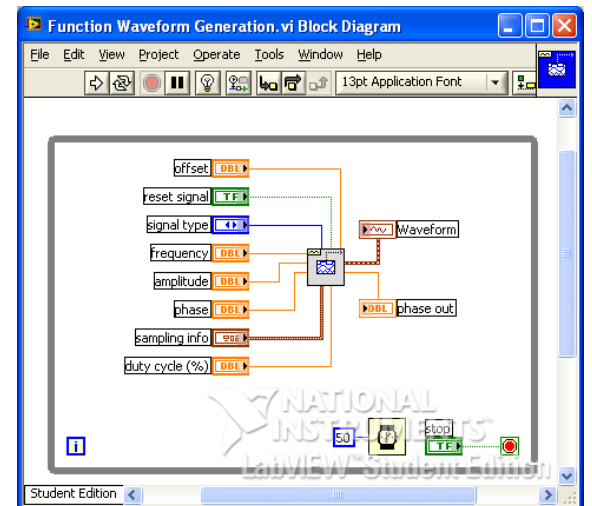
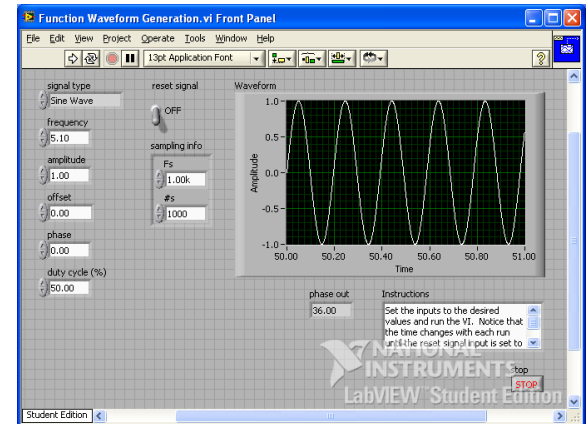
# Virtual Instrumentation Applications

- Design
  - Signal and image processing
  - Embedded system programming (PC, DSP, FPGA, etc.)
  - Simulation and prototyping
- Control
  - Automatic controls and dynamic systems
  - Mechatronics and robotics
- Measurements
  - Circuits and electronics
  - Measurements and instrumentation



# VI Programming Environment

- **Front panel:** interface to your VI program
  - Controls = inputs
  - Indicators = outputs
- **Block diagram:** program code in a graphical form
  - Terminals corresponding to front panel controls and indicators
  - Constants, functions, subVIs, structures
  - Wires connect components together



# Front Panel and Controls Palette

The image shows the LabVIEW interface for a 'Function Waveform Generation.vi'. The front panel includes several controls and indicators:

- Numeric control:** A slider for 'signal type' set to 'Sine Wave'.
- Boolean control:** A toggle switch for 'reset signal' set to 'OFF'.
- Numeric controls:** Input fields for 'frequency' (5.10), 'amplitude' (1.00), 'offset' (0.00), 'phase' (0.00), and 'duty cycle (%)' (50.00).
- Numeric indicator:** An output field for 'phase out' showing the value 36.00.
- Waveform graph:** A graph showing a sine wave with 'Amplitude' on the y-axis (ranging from -1.0 to 1.0) and 'Time' on the x-axis (ranging from 50.00 to 51.00).
- Controls Palette:** A floating window showing various control types such as 'Num Ctrls', 'Buttons', 'Text Ctrls', 'User Ctrls', 'Num Inds', 'LEDs', and 'Graph Indicat...'. The 'Express' section is expanded.

Arrows point from the text labels to the corresponding elements in the LabVIEW interface.

- Controls and indicators can be created using the controls palette.

# Block Diagram and Functions Palette

While loop

Numeric control terminal

Waveform graph terminal

Numeric indicator terminal

Wait function

Numeric constant

Functions

- Express
  - Input
  - Signal Analysis
  - Output
  - Sig Manip
  - Exec Control
  - Arith & Compar
- Favorites
- User Libraries
  - Express User ...
- Select a VI...

- Functions, execution controls, etc. can be created using the functions palette.

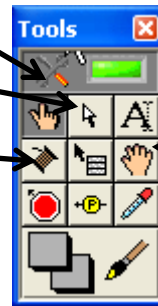
# Tools Palette

- A tool is a special operation mode of the mouse cursor. You use tools to perform specific editing functions.
- Automatic tool selection: labVIEW will automatically select the corresponding tool as you move the cursor over objects on either the front panel or the block diagram.

Automatic tool selection

Position tool

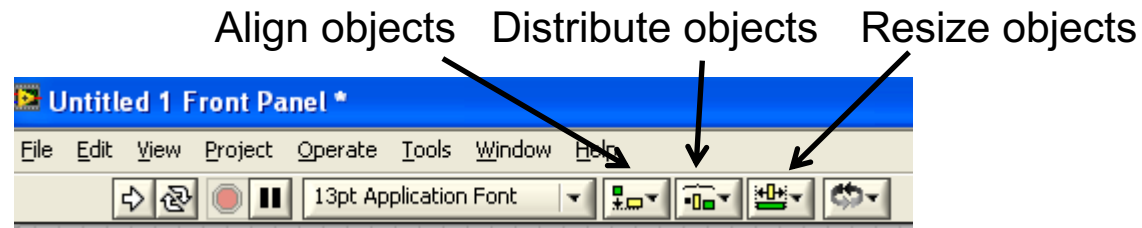
Wiring tool



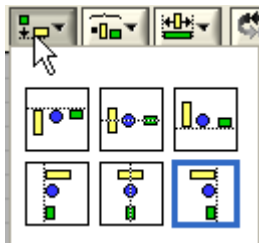
Scrolling tool

# Aligning Objects

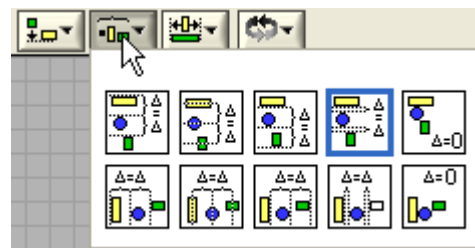
- After selecting the desired objects for alignment, you can align, distribute, or resize them – make things neat and pretty.



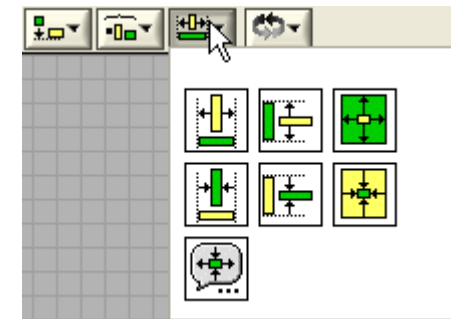
Align objects menu



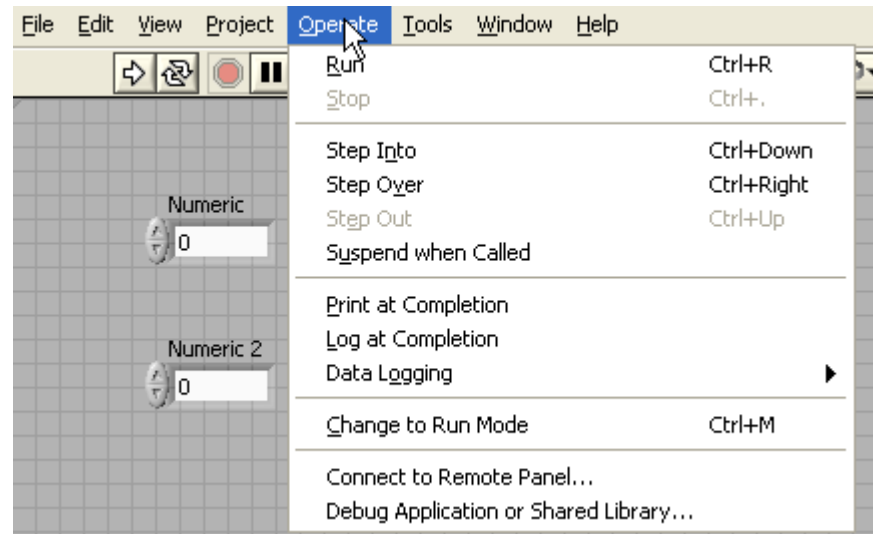
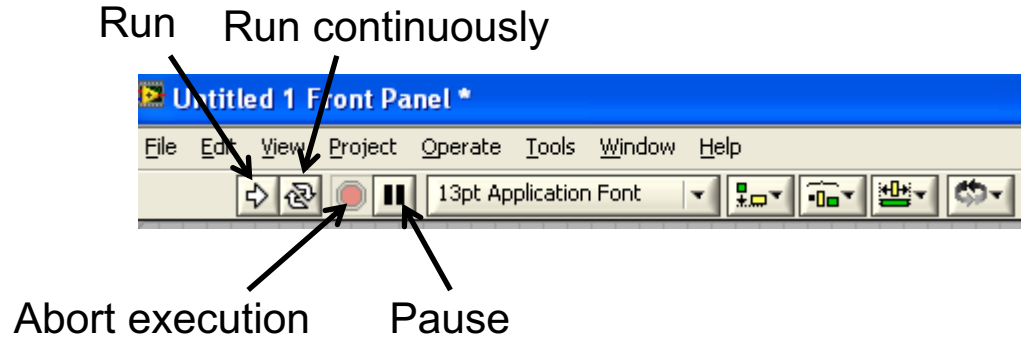
Distribute objects menu



Resize objects menu

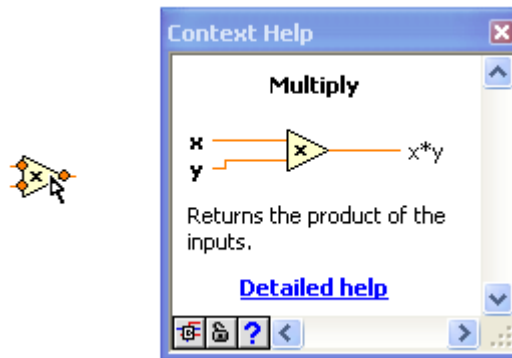


# Execution Control



# Context Help Window

- “Show Context Help” (Ctrl+H) – show context help associated with the selected objects.





# Basic Concept of LabVIEW Programming

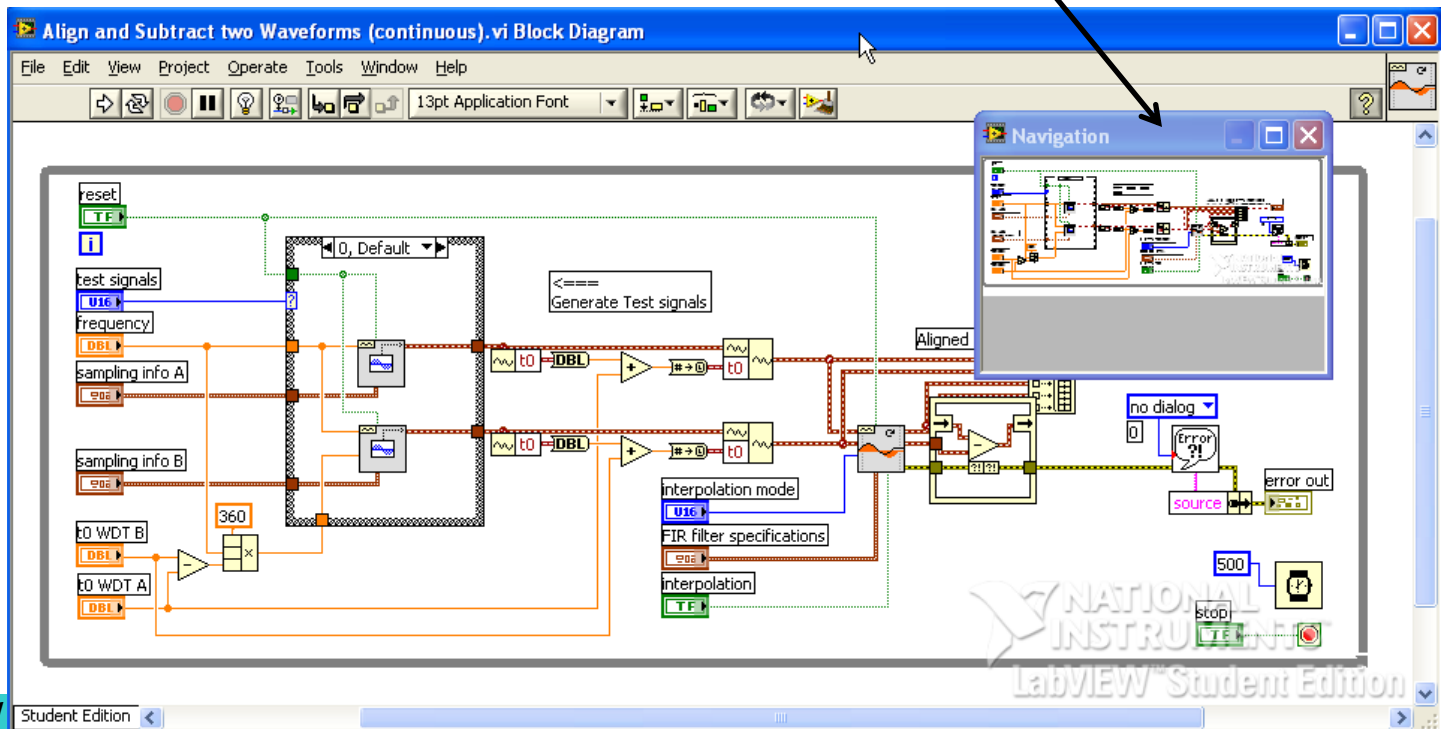
- **Modular programming:** a given task is divided into a series of simpler subtasks which is implemented separately and then assembled.
- **Data flow programming:** the icons (subtasks) in the block diagram are wired together to allow data flow. The execution of a VI is governed by the data flow.



# Navigation Window

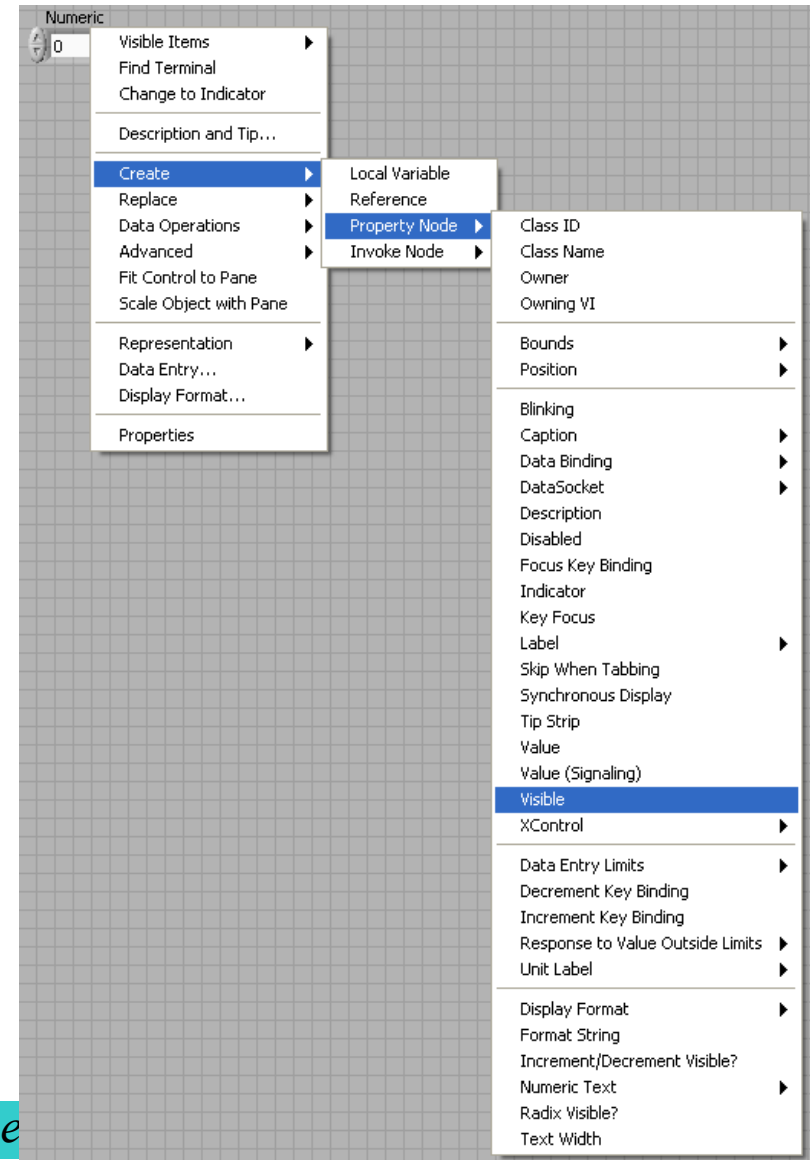
- For complicated VIs, the navigation window can display an overview of the active front panel in edit mode or the active block diagram.

Navigation window



# Property Nodes

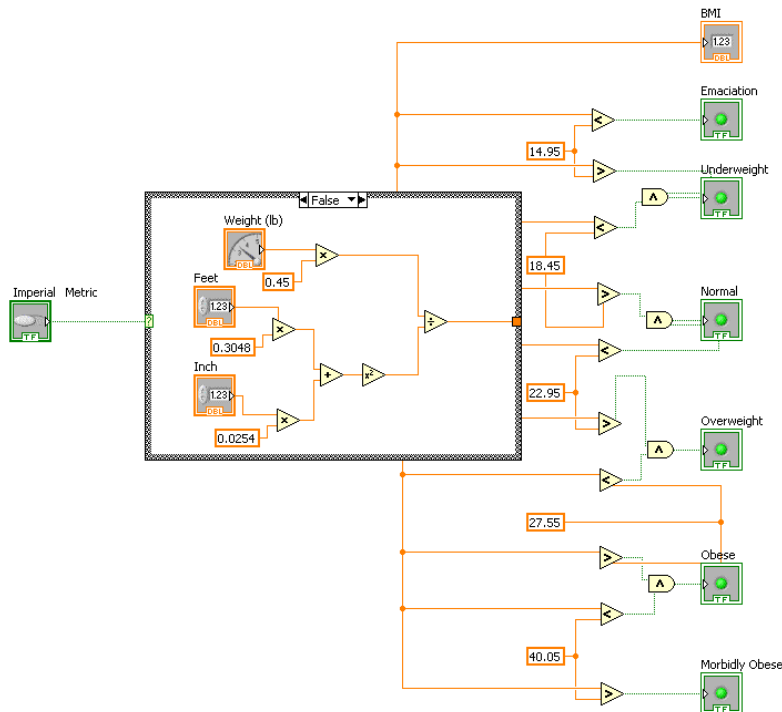
- Allow you to *set* or *get* the properties of objects. For example, in some applications, you might want to make a front panel object vanish while the VI is running if a Boolean input is True.



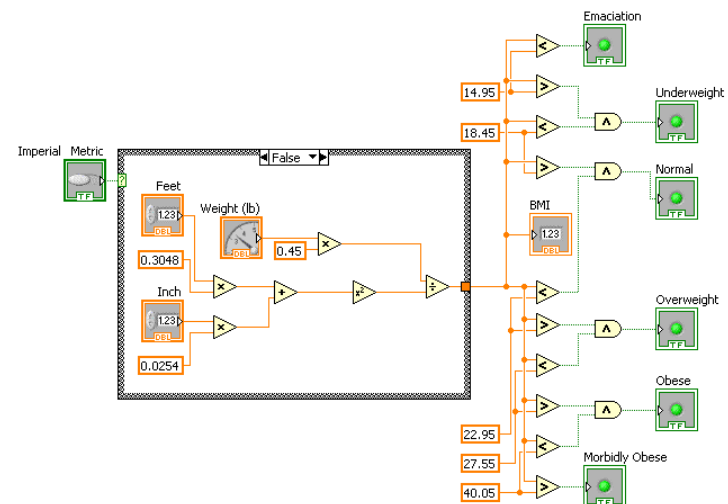
# Clean Up the Block Diagram

- Cleaning up for easier-to-understand diagrams and for debug purposes

Before cleanup

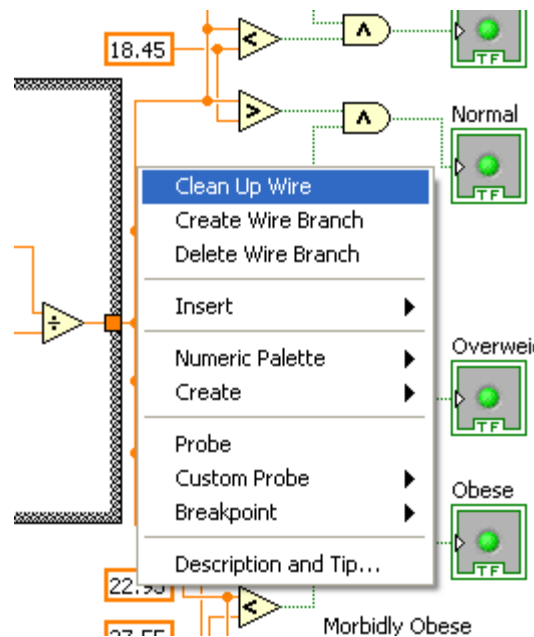


After cleanup



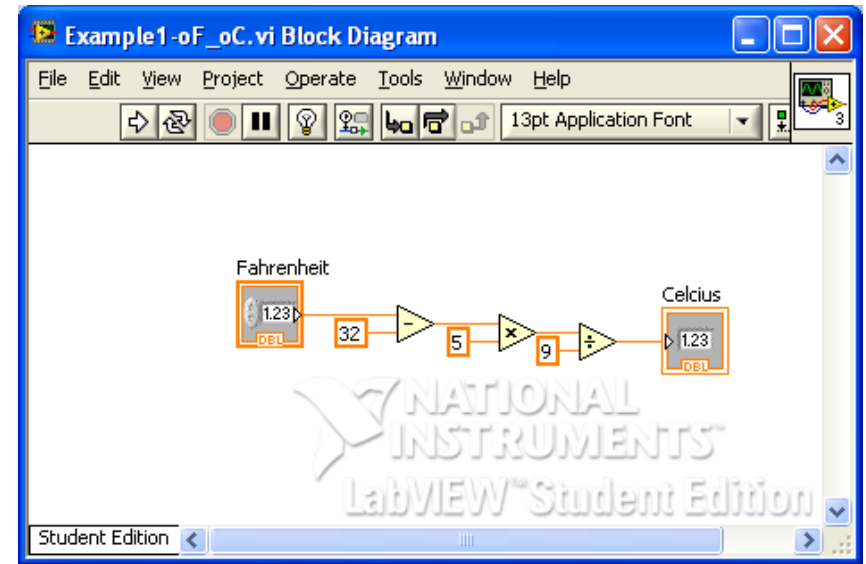
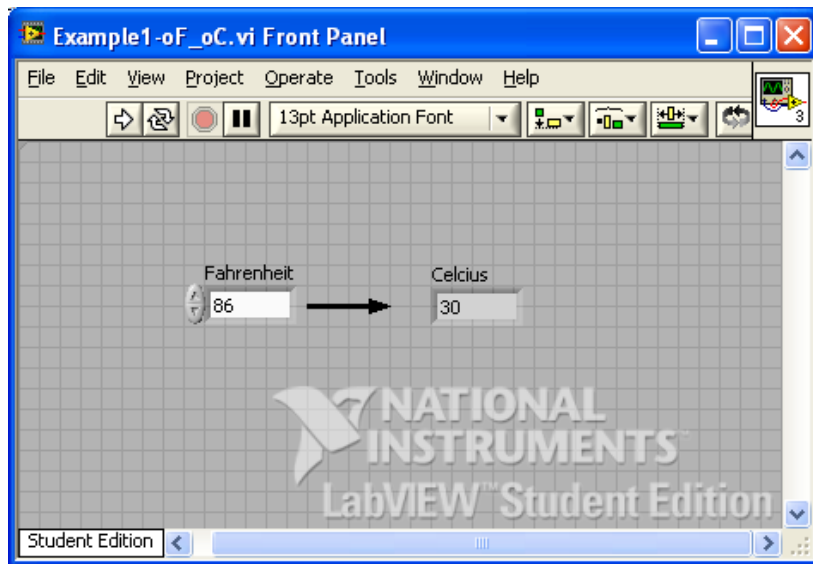
# Clean Up Wire

- You can clean up a wire by right clicking it, and select “Clean Up Wire” – Very useful.

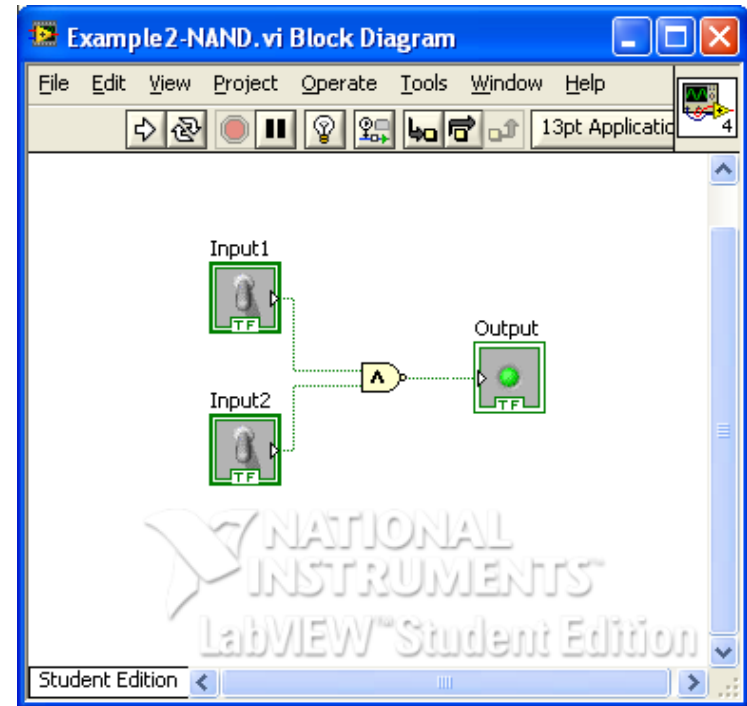
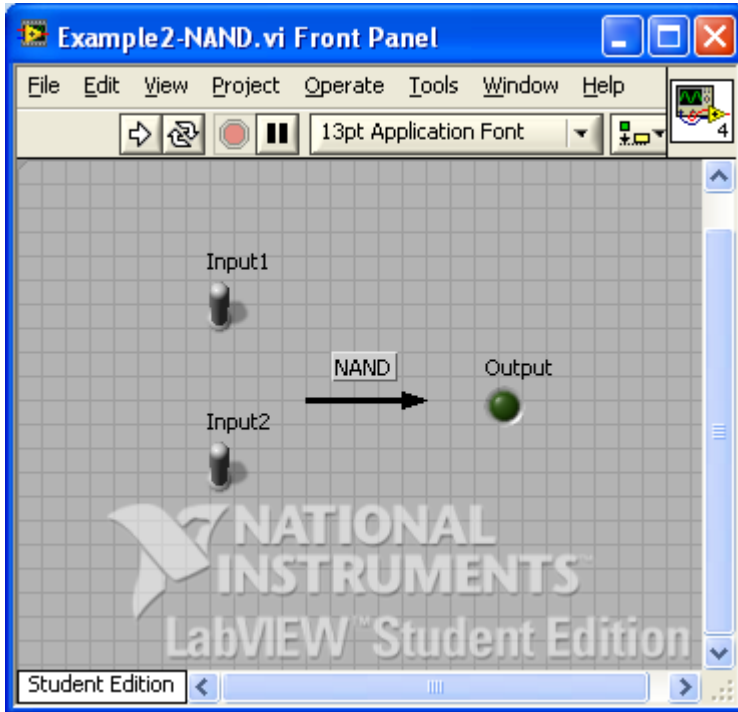


# Example 1.1 – °F to °C Converter

- $[\text{°C}] = ([\text{°F}] - 32) \times 5/9$



# Example 1.2 – NAND Gate



# BE/EE189 Design and Construction of Biodevices

## Lecture 2





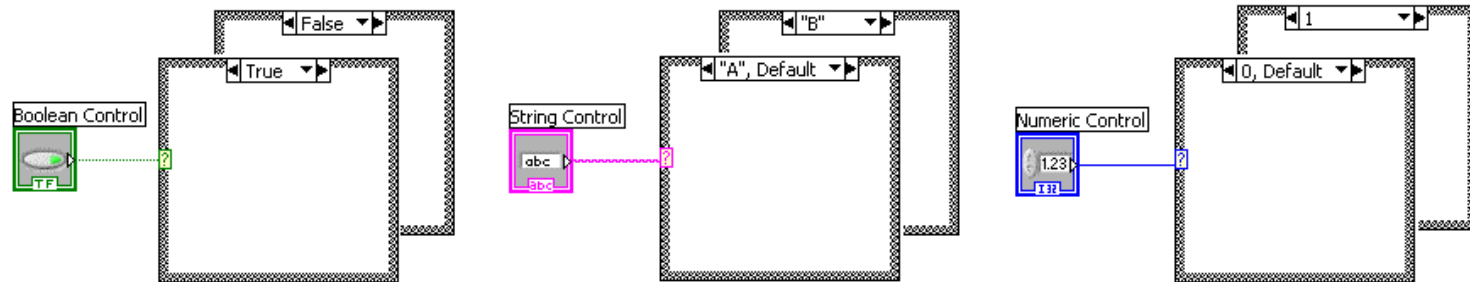
# LabVIEW Programming – More Basics, Structures, Data Types, VI

- Case structure
- Debugging techniques
- Useful shortcuts
- Data types in labVIEW
- Concept of subVI
- Creating a subVI
- Using a VI as a subVI
- Error checking and error handling
- The VI hierarchy window



# Case Structure

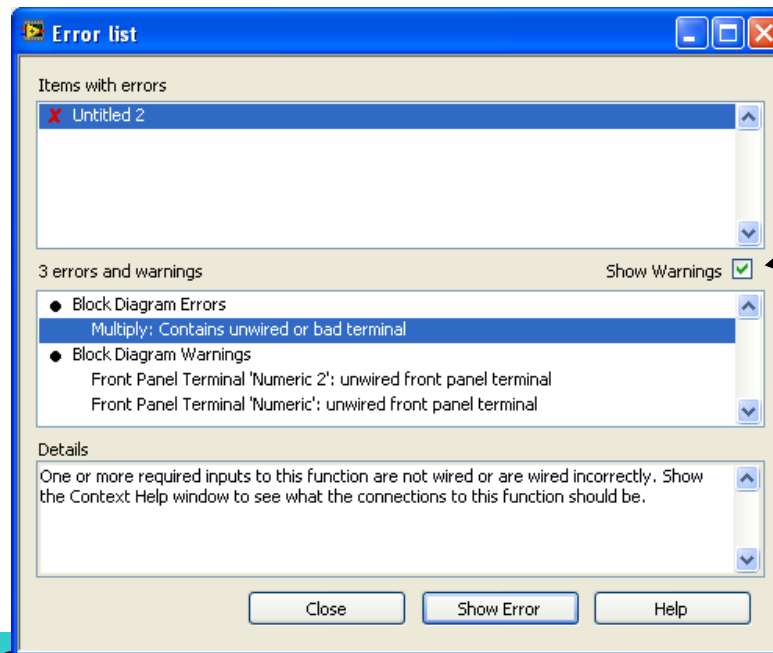
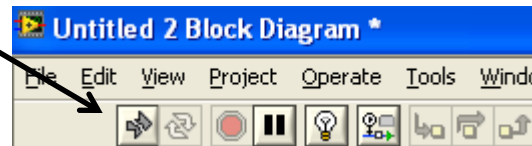
- CASE structure has two or more subdiagrams.
- Only one subdiagram to execute at a time based on the value of the selector.
- Each subdiagram must provide output value for the CASE structure.



# LabVIEW Debugging Techniques – Finding Errors

- Click on the broken run button to show the error list

Broken run button

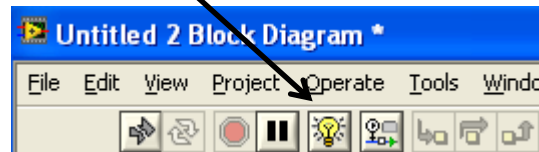


Show warnings

# LabVIEW Debugging Techniques – Highlight Execution

- Highlight execution can be used to show the animation of the VI execution
- Will reduce performance

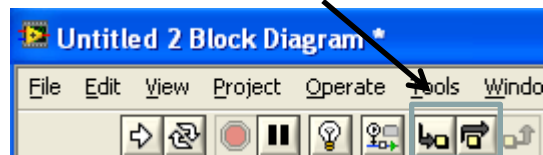
Highlight execution button



# LabVIEW Debugging Techniques – Single-Stepping

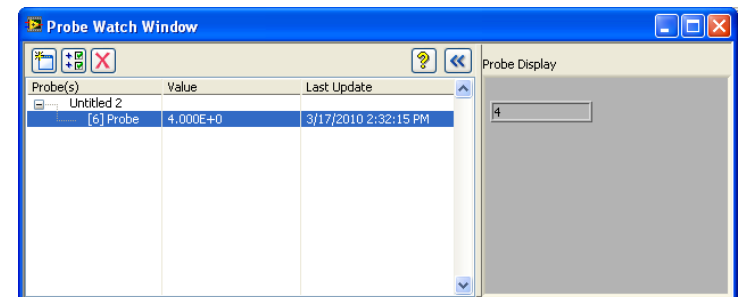
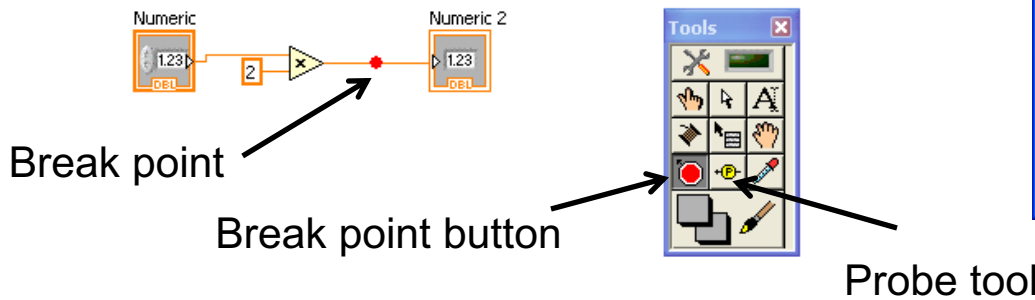
- In the single-step mode, you can either “step into” or “step over” the node in the block diagram.

Single-stepping buttons



# LabVIEW Debugging Techniques – Breakpoints and Probes

- You can halt execution at certain locations by using the breakpoint.
- Use the probe tool to view data as it flows through a block diagram wire. A probe watch window will display the current data value.



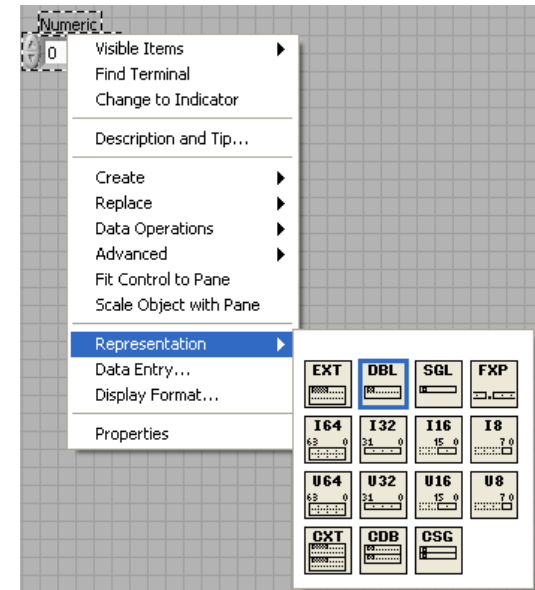
# Useful Shortcuts

- Ctrl-S: Save a VI
- Ctrl-R: Run a VI
- Ctrl-E: Toggle between the front panel and the block diagram
- Ctrl-H: Toggle the **Context Help** window on and off
- Ctrl-B: Remove all bad wires
- Ctrl-W: Close the active window
- Ctrl-F: Find objects and VIs



# Data Types

- Numeric
  - integer: signed, unsigned + precision → I64, I32, I16, I8, U64, U32, U16, U8
  - floating-point number: single, double, extended precision
- String: a sequence of characters
- Boolean: true or false



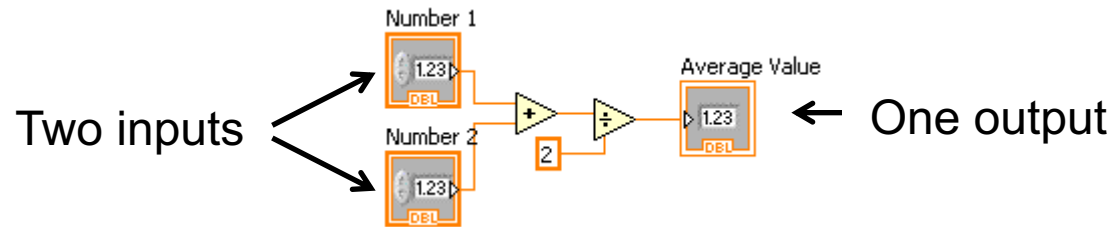


# SubVI

- A subVI is a stand-alone VI that is called by other VIs, similar to a subroutine or function in text based programming languages.
- Remember **Modular programming**: a given task (top-level VI) is divided into a series of simpler subtasks (functions or subVIs) which is implemented separately and then assembled.

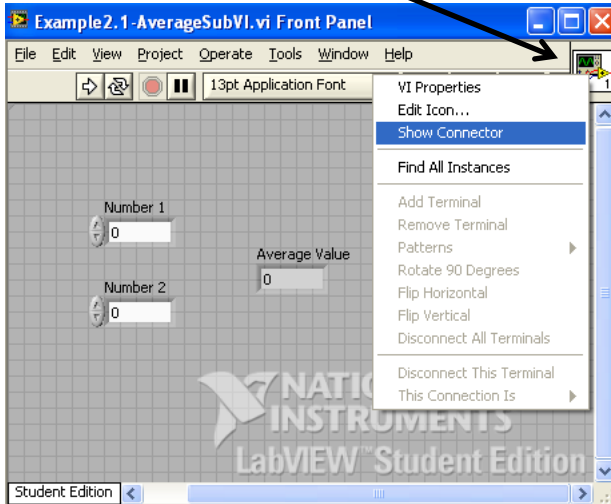
# Creating subVI – 1: Develop an independent VI

- Define the input and output of your subVI, and develop the VI.

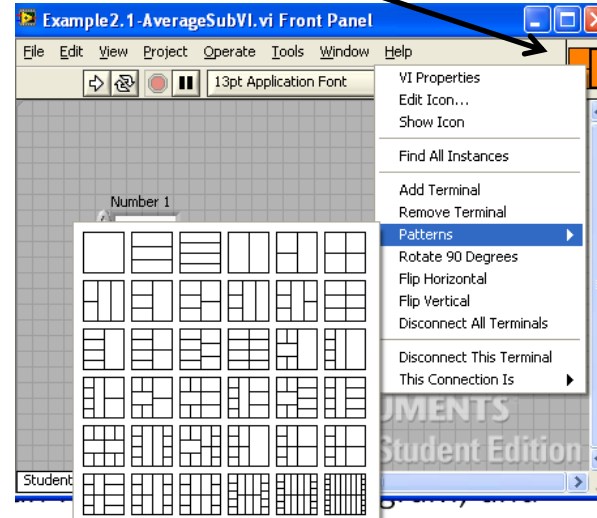


# Creating subVI – 2: Assigning Input and Output

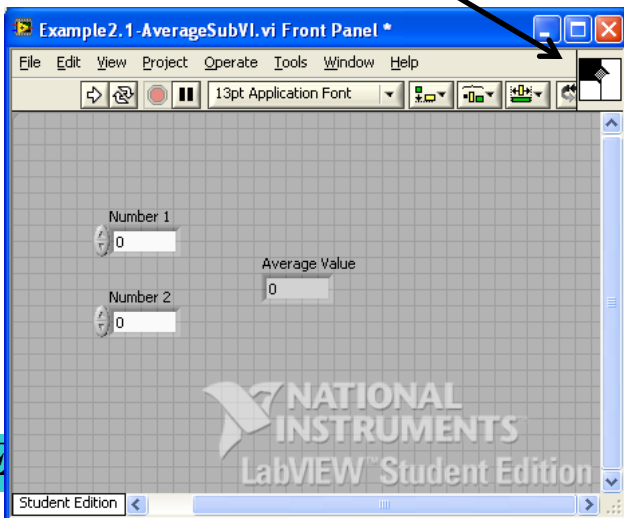
Right-click on the icon plane, select “Show Connector”



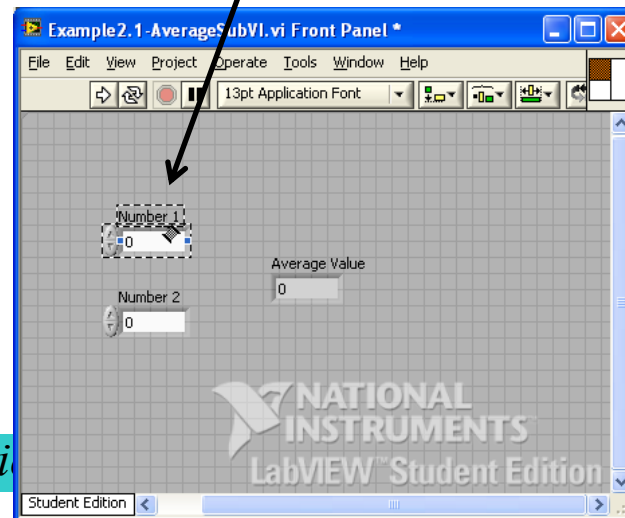
Right-click on the icon plane, select “Patterns” to define number of inputs and outputs



Click on the terminal with the wiring tool



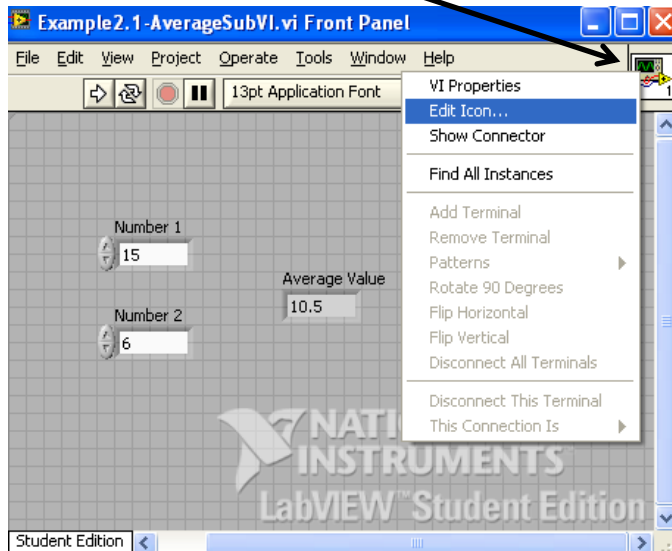
Then click on the control or indicator with the wiring tool



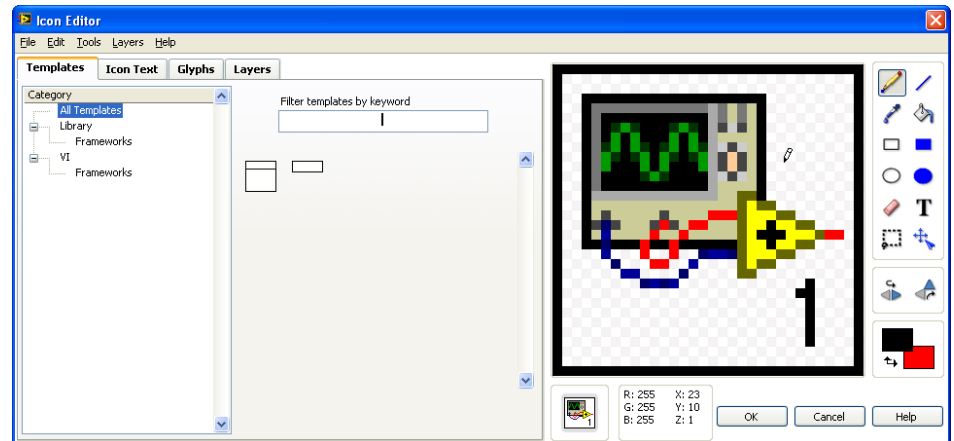
# Creating subVI – 3: Edit Icon

- A subVI is represented by an icon in the block diagram, and you can customize the icon picture.

Right-click on the icon plane, select “Edit Icon...”

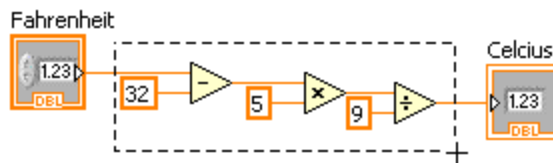


Edit the icon with the icon editor

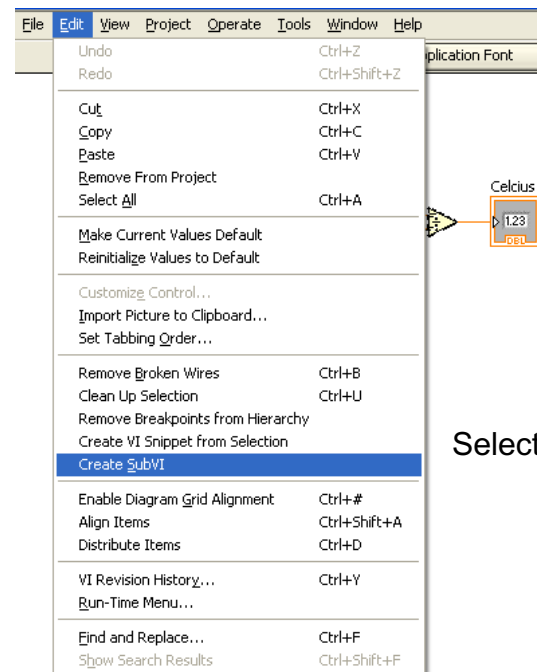


# Creating a subVI from a selection

- You can select components of the main VI and group them into a subVI

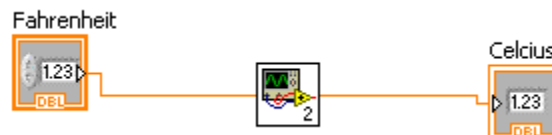


Select components



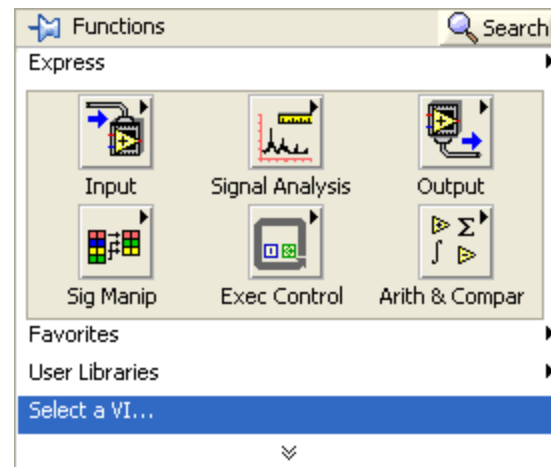
Select "Create SubVI"

SubVI created



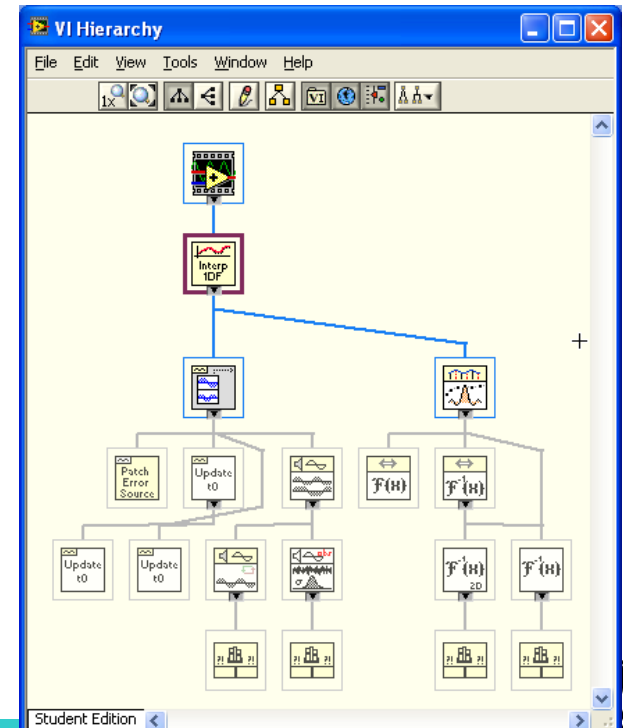
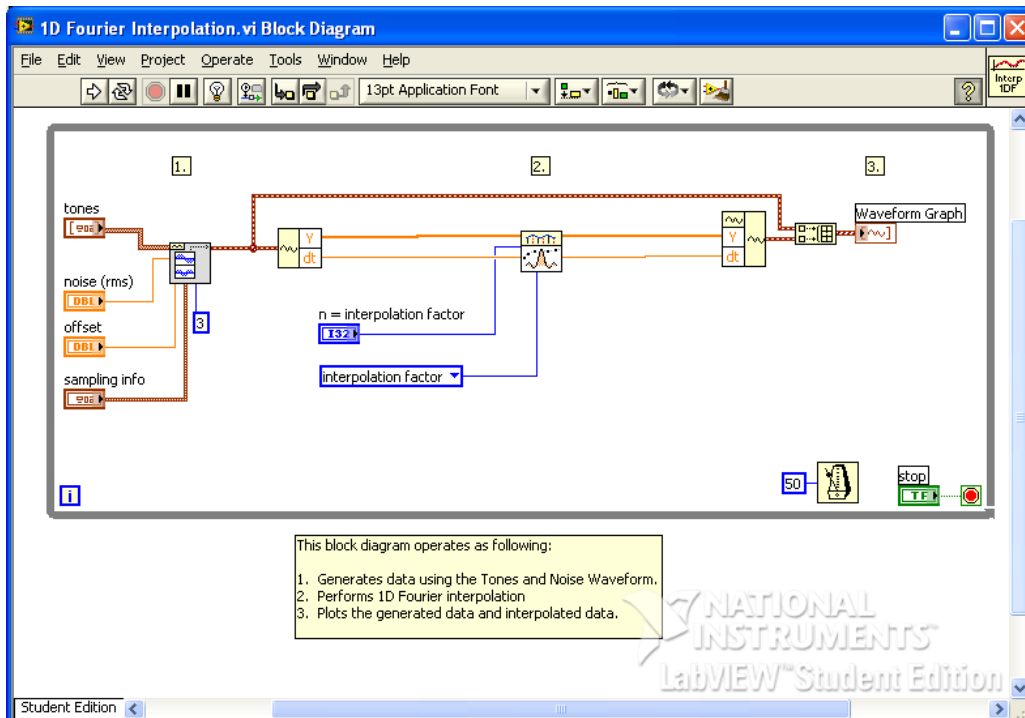
# Calling SubVI

- In the functions palette, select “Select a VI...”, and choose the VI that you developed as a subVI. Just like you add a built-in function.



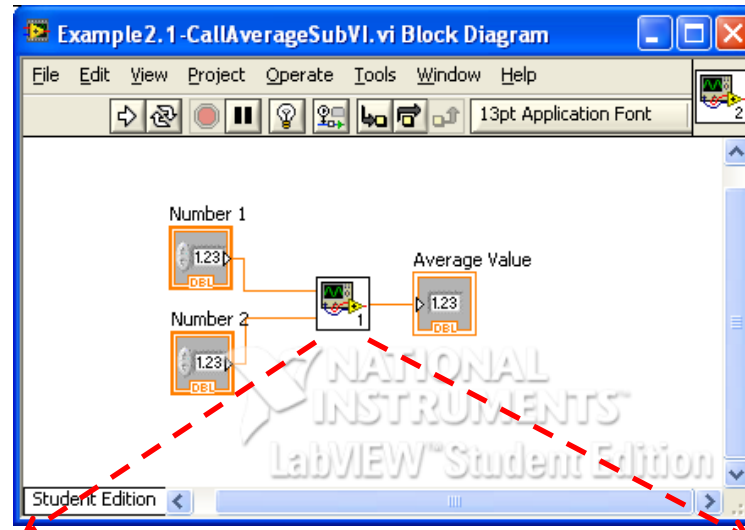
# The VI Hierarchy Window

- Display a graphical representation of the hierarchical structure of all VIs in memory and shows the dependencies of top-level VIs and sub VIs (View >> VI Hierarchy).

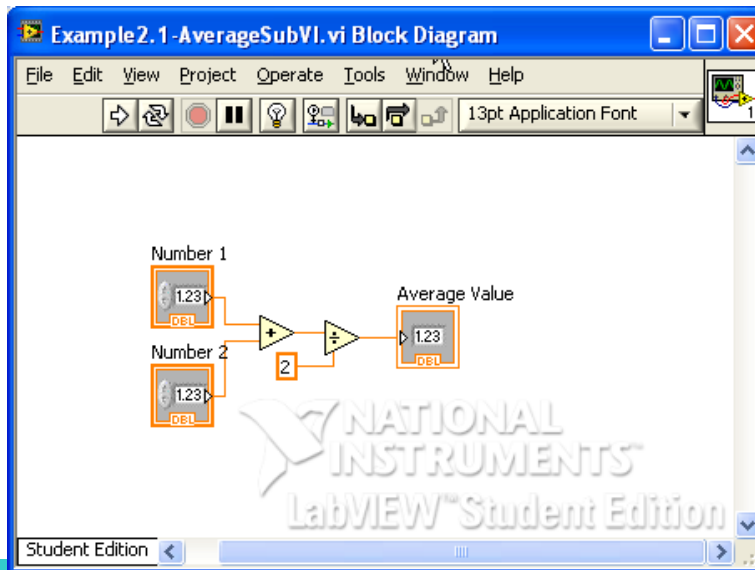


# Example – SubVI to Calculate Average Value

Top-level VI



SubVI





# Work Example 2.1 – BMI Calculation and Display

- Body mass index (BMI) = body weight (kg)/(height(m))<sup>2</sup>

Category	BMI range – kg/m <sup>2</sup>
Emaciation	less than 14.9
Underweight	from 15 to 18.4
Normal	from 18.5 to 22.9
Overweight	from 23 to 27.5
Obese	from 27.6 to 40
Morbidly Obese	greater than 40

[http://en.wikipedia.org/wiki/Body\\_mass\\_index](http://en.wikipedia.org/wiki/Body_mass_index)



# Work Example 2.1 – BMI Calculation and Display

The screenshot shows a LabVIEW front panel titled "Example3-BMI.vi Front Panel". The interface includes a menu bar (File, Edit, View, Project, Operate, Tools, Window, Help) and a toolbar with various icons. The main display area is divided into two sections: Imperial and Metric. The Imperial section shows a weight gauge (lb) with a needle pointing to approximately 150, and height input fields for Feet (5) and Inch (8). The Metric section shows a weight gauge (kg) with a needle pointing to approximately 75, and a height bar (cm) with a blue bar reaching approximately 180. Below these sections, the BMI is displayed as 21.982. A row of six indicator lights represents BMI categories: Emaciation, Underweight, Normal (lit), Overweight, Obese, and Morbidly Obese. The National Instruments LabVIEW Student Edition logo is visible in the bottom right corner.

# Work Example 2.2 – SubVI for Evaluating Blood Pressure

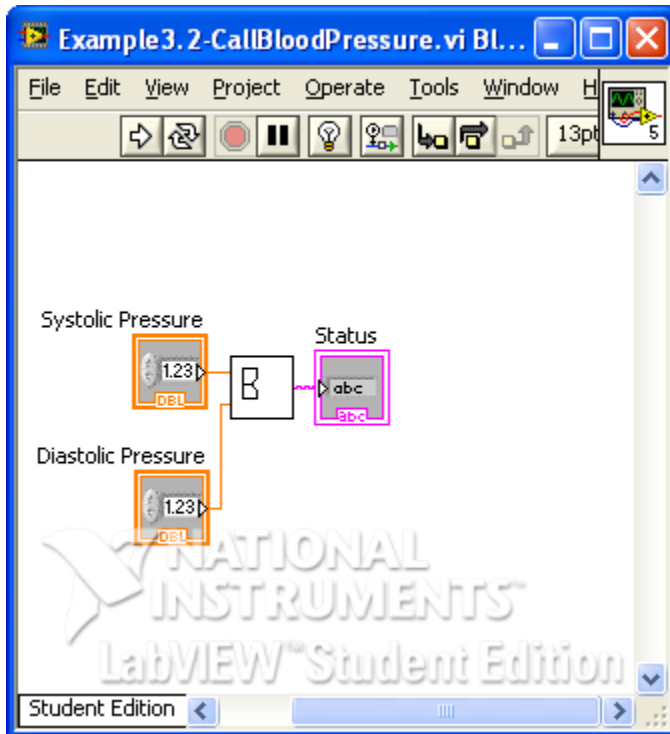
- For input systolic and diastolic pressure, output the status.
- Your systolic and diastolic numbers may not be in the same blood pressure category. In this case, the more severe category is the one you're in.

Category	Systolic (top number)		Diastolic (bottom number)
Normal	Less than 120	<i>And</i>	Less than 80
Prehypertension	120–139	<i>Or</i>	80–89
High blood pressure			
Stage 1	140–159	<i>Or</i>	90–99
Stage 2	160 or higher	<i>Or</i>	100 or higher

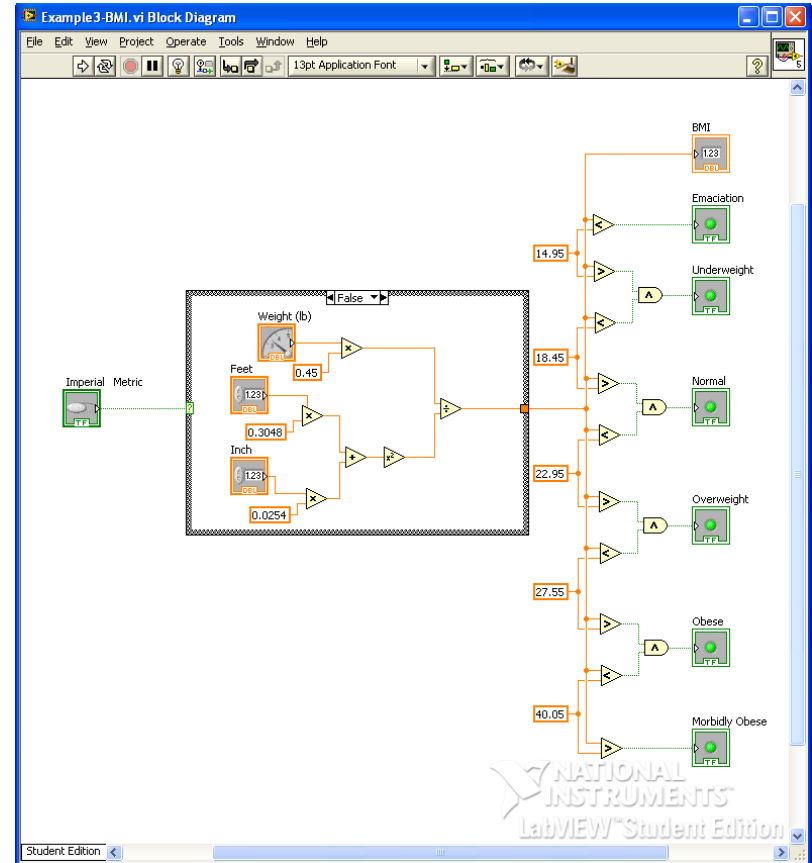
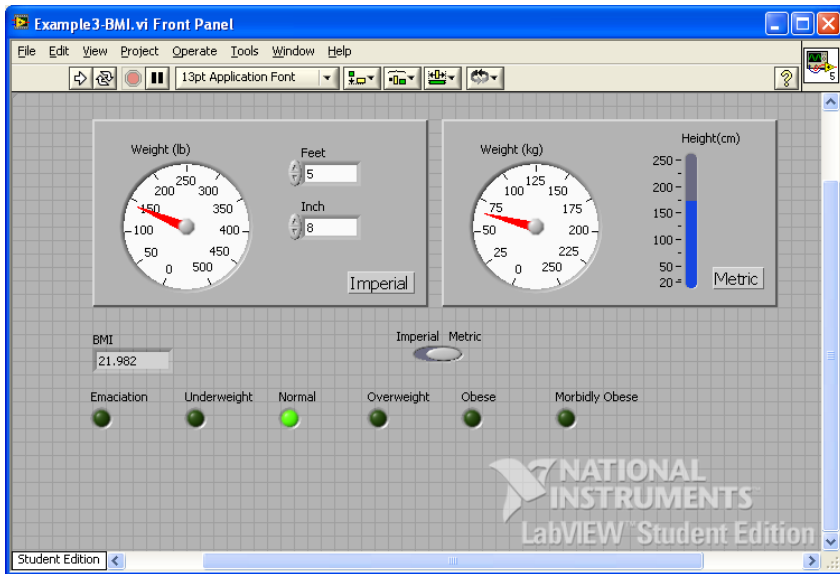
# Work Example 2.2 – SubVI for Evaluating Blood Pressure

Top-level VI

What should the SubVI look like?

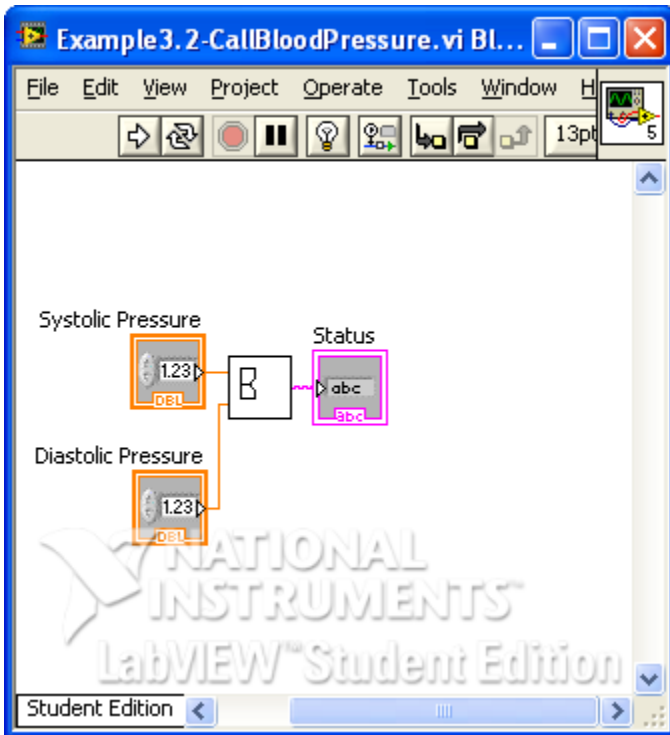


# Answer

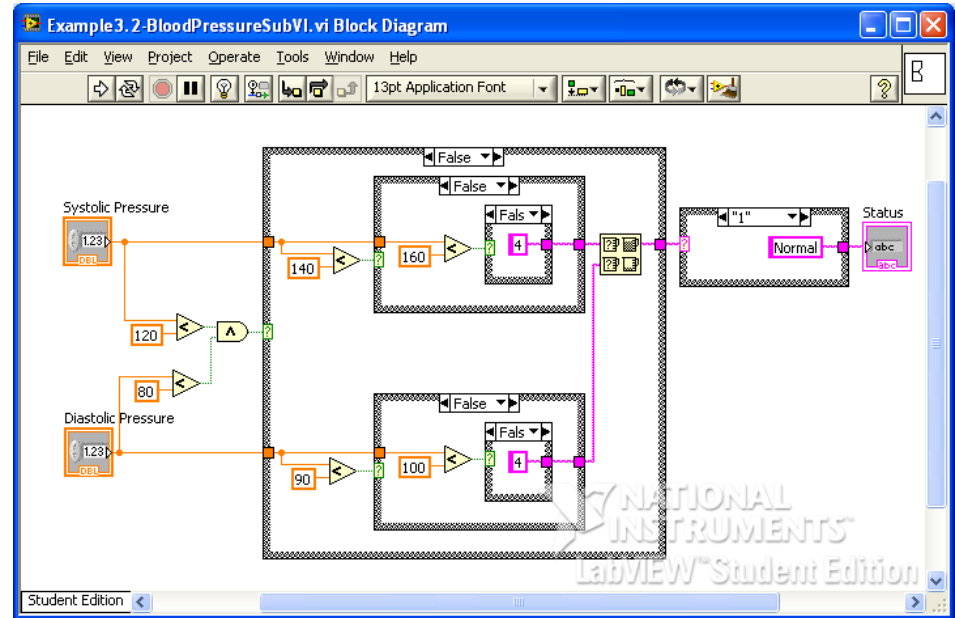


# Answer

Top-level VI



SubVI



# BE/EE189 Design and Construction of Biodevices

## Lecture 3



# LabVIEW Programming – Error Handling & Structures

- Error Handling
- For loop, while loop
- Sequence structure
- Timing control
- Formula node
- Local variables



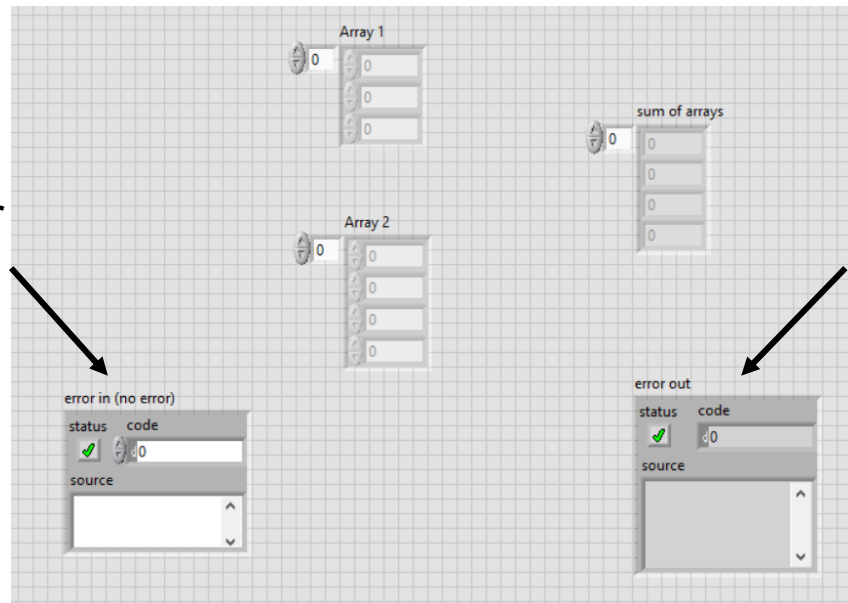


# Error Handling

- LabVIEW allows you to define error codes and messages for your subVIs
- These errors can be passed along to and from multiple subVIs
- All NI-produced subVIs contain an error in and error out node

## Error In:

The error in indicator reads errors passed to it by previous VIs. The default is “no error”.

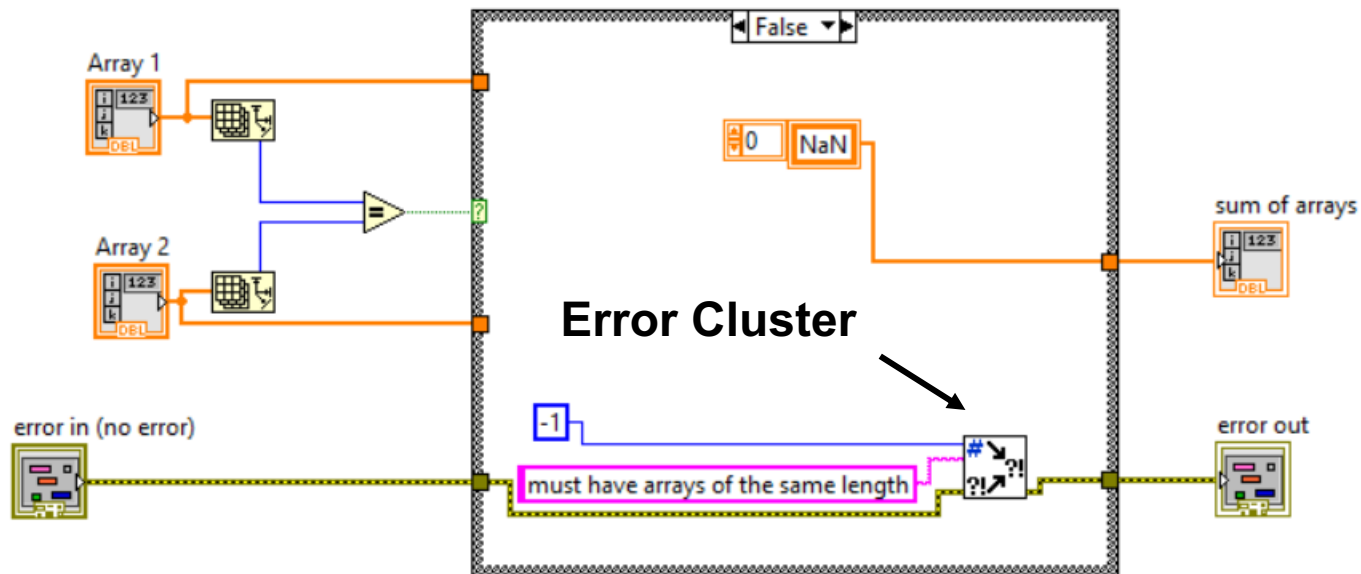


## Error Out:

The error out indicator reads errors from previous VIs and the current VI.

# Error Handling

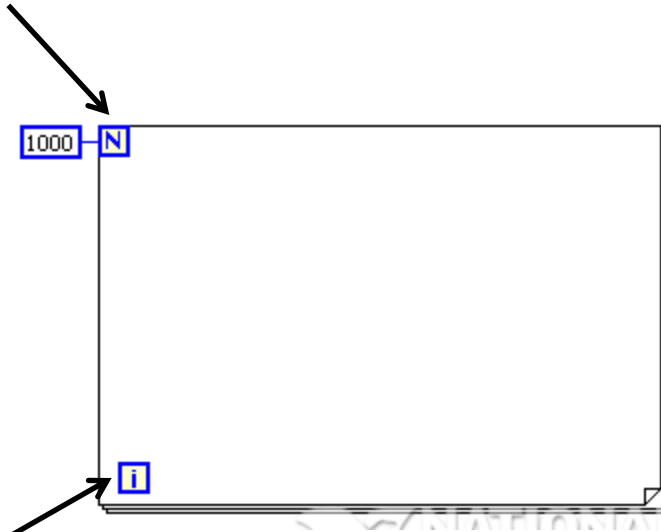
- Error codes (integer) and messages (string) are added in using the “error cluster” function in the “Dialog & User Interface”.



# For Loop

## Count terminal:

Number of times you want the loop to execute; Available for use inside the for loop.



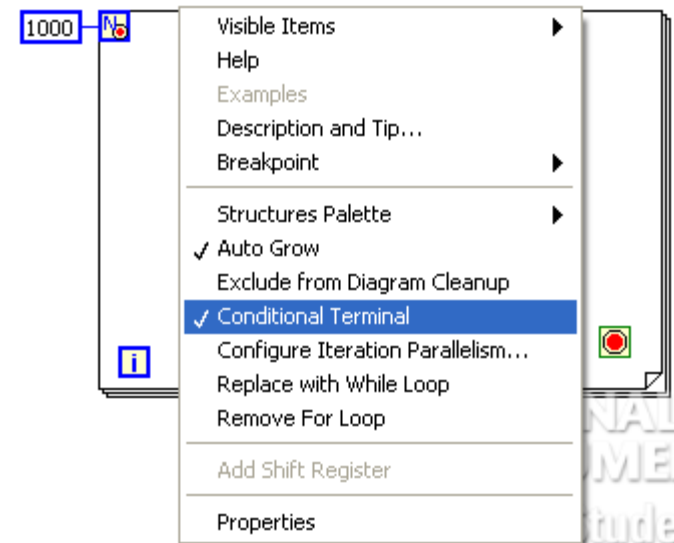
## Iteration terminal:

Number of times the loop has executed; Available for use inside the for loop.

Note: **starting from 0**

## Conditional terminal:

A for loop with a conditional terminal executes until the condition occurs or until all iterations complete



# While Loop

**Iteration terminal:**  
Number of times the  
loop has executed;  
Available for use  
inside the for loop.  
Note: **starting from 0**



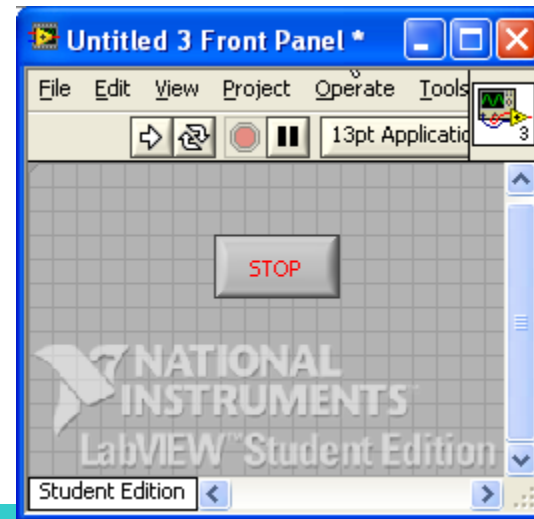
**Conditional terminal:**  
Loop executes until input is TRUE

Selected from Functions>>Programming>>Structures

A while loop with a button to stop execution

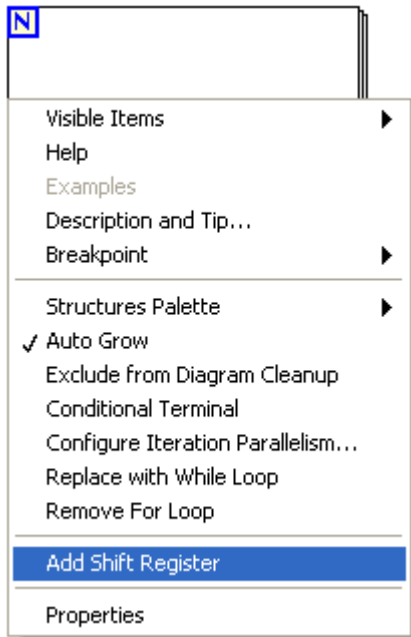


Selected from Functions>>Express>>Execution Control

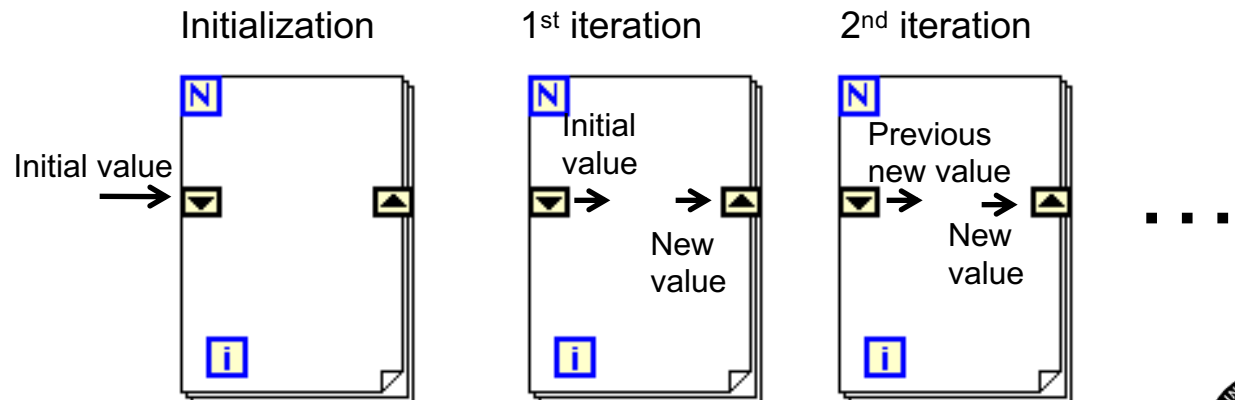
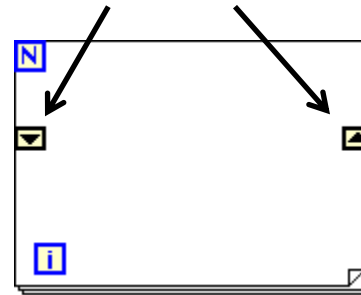


# Shift Registers

- Shift registers transfer values from one iteration of a For Loop or While Loop to the next.

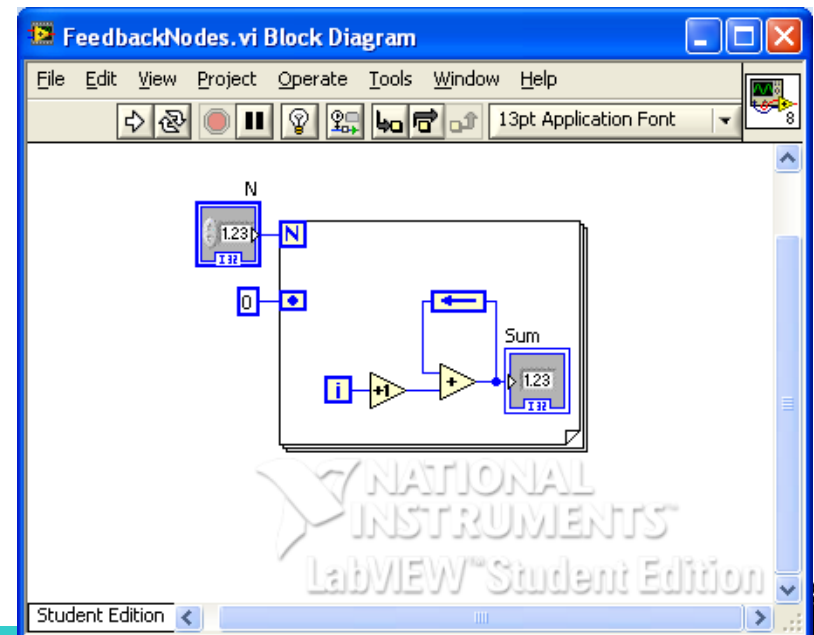
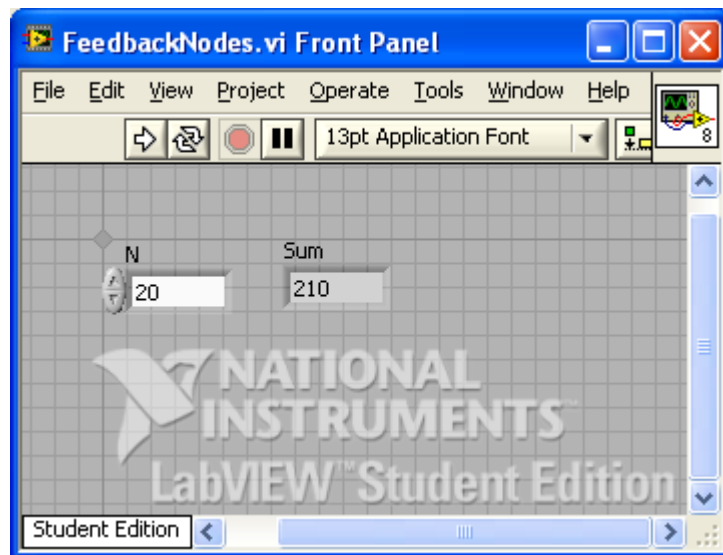


Shift registers



# Feedback Nodes

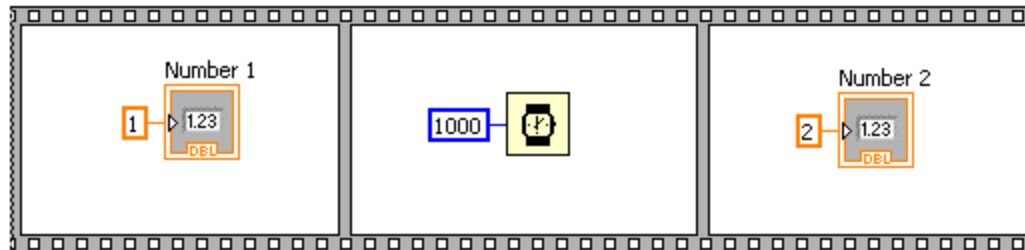
- Similar to shift registers.
- Use the feedback node to avoid unnecessarily long wires.
- Can configure multiple delays.
- Example: calculate  $1+2+\dots+N$ .



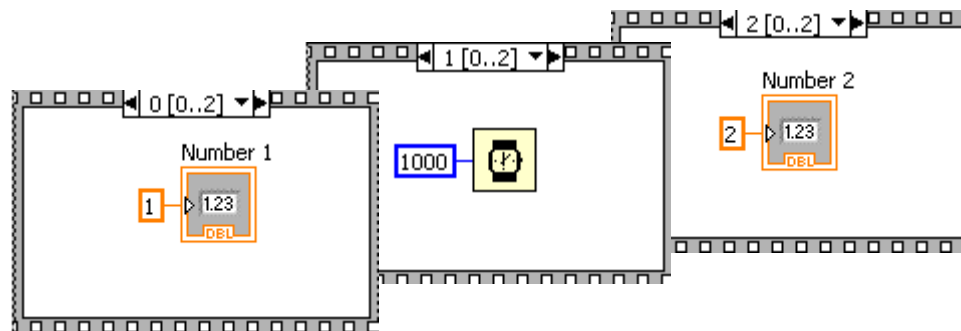
# Sequence Structure

- The **sequence structure** executes subdiagrams sequentially.
- Two classes: **flat sequence** and **stacked sequence**.

Flat sequence

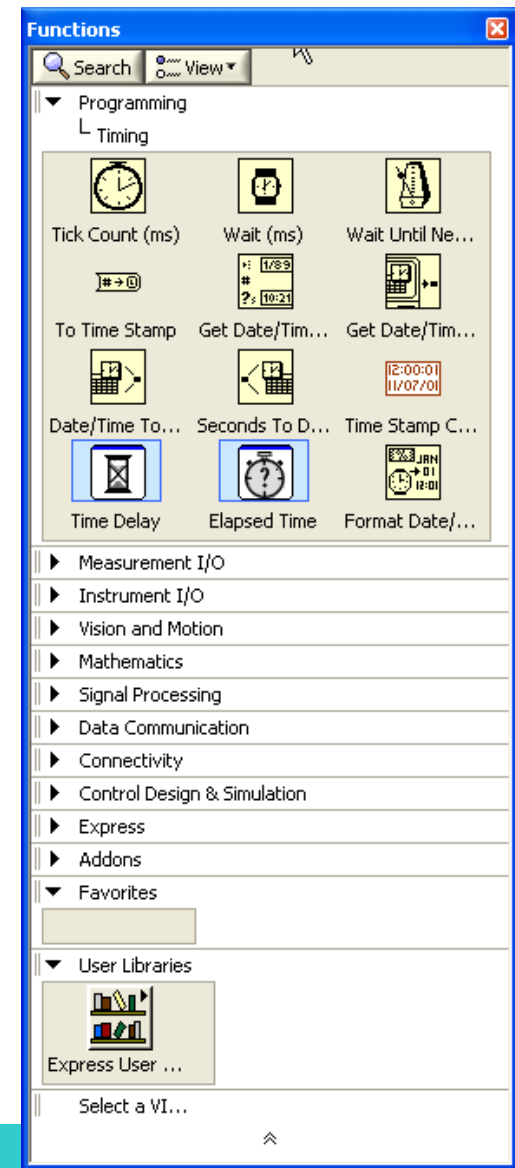
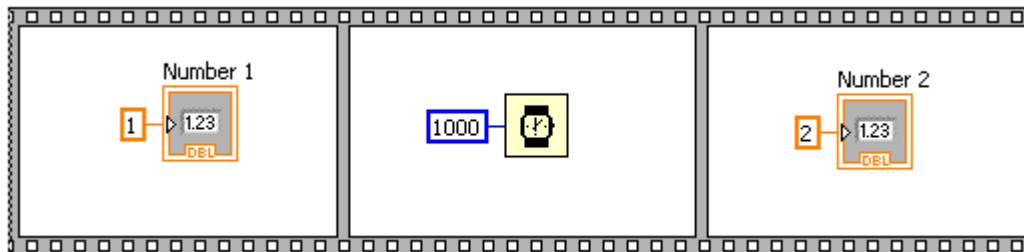


Stacked sequence



# Timing Control

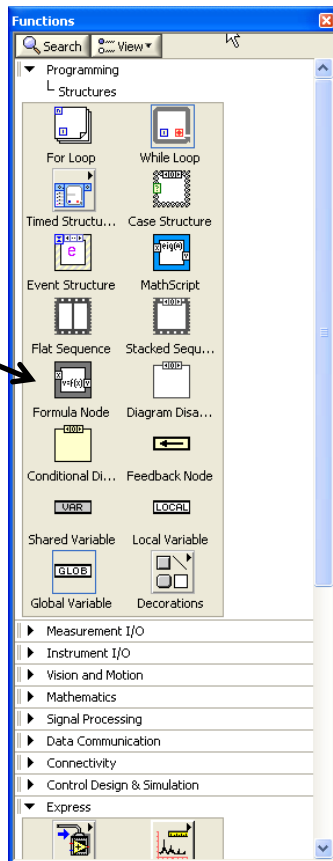
- **Functions>>Programming>>Timing**
- Example: wait 1 second before executing next step.



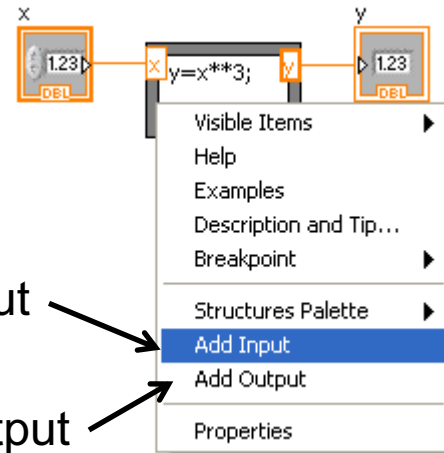


# Formula Node

- Allows you to program one or more algebraic formulas.



Formula node



Add input

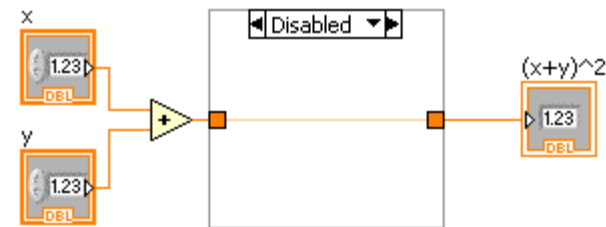
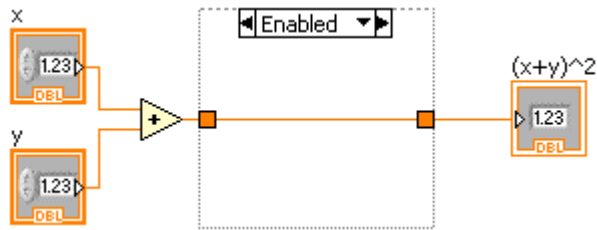
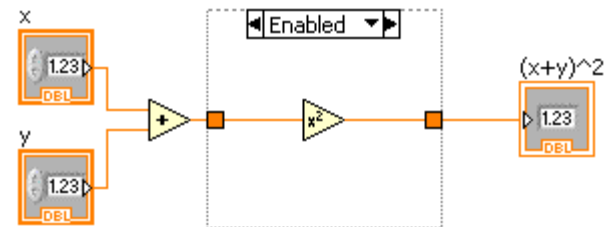
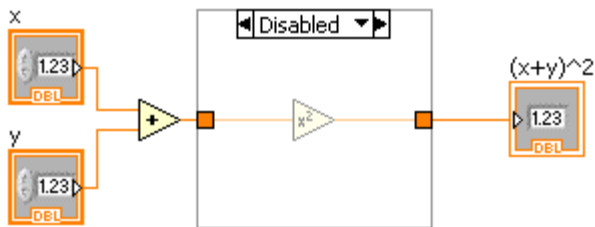
Add Output

Example: calculate  $y=x^3$



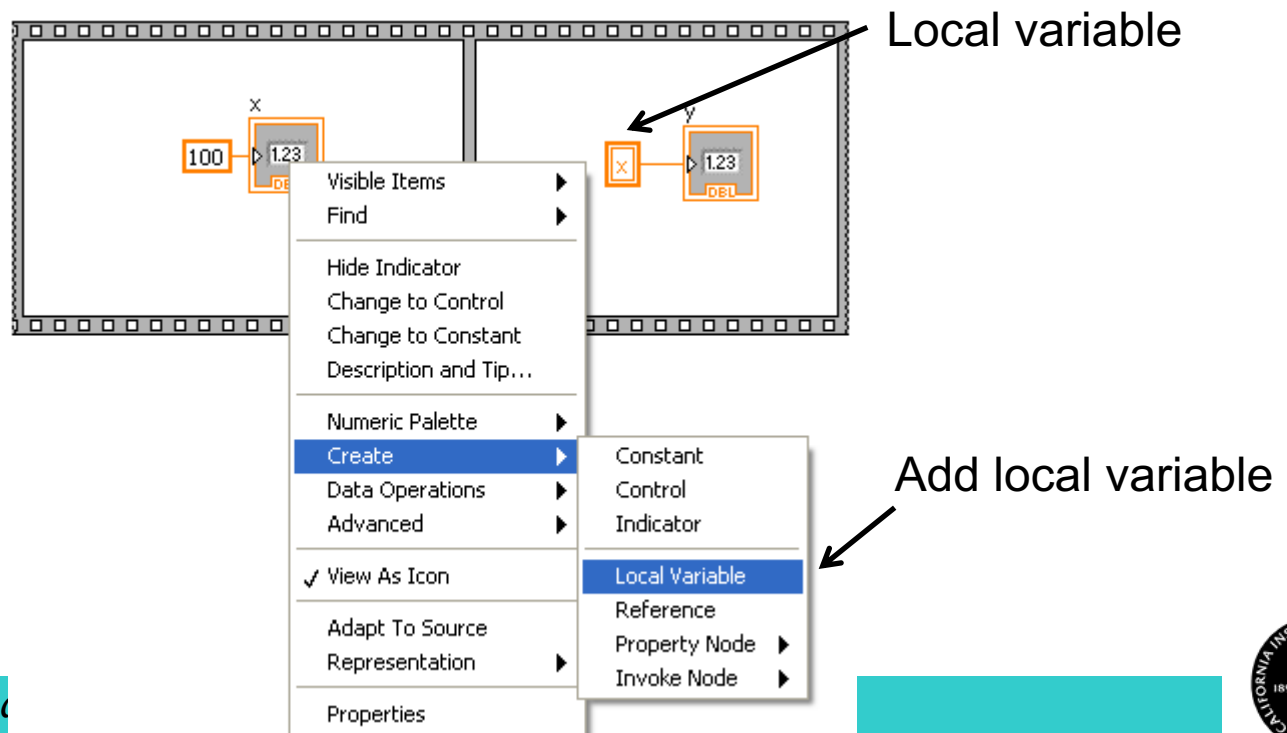
# Diagram Disable Structure

- To disable specific sections of code, equivalent to commenting out code in a text-based Programming language.
- The disabled codes can be enabled by select “Enable This Subdiagram”.

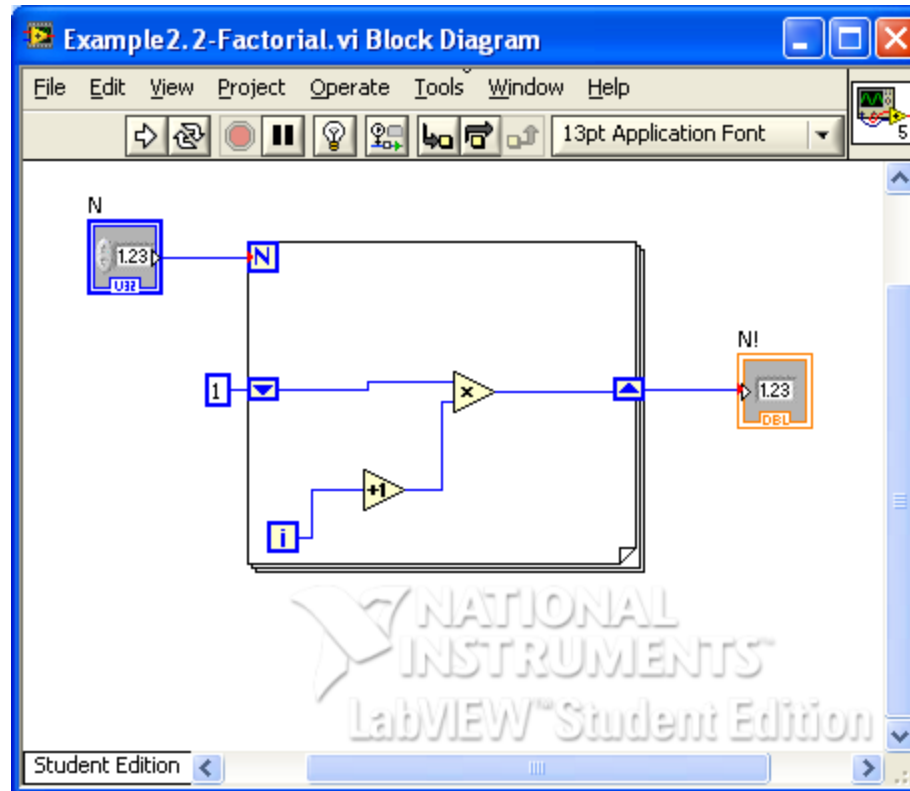


# Local Variables

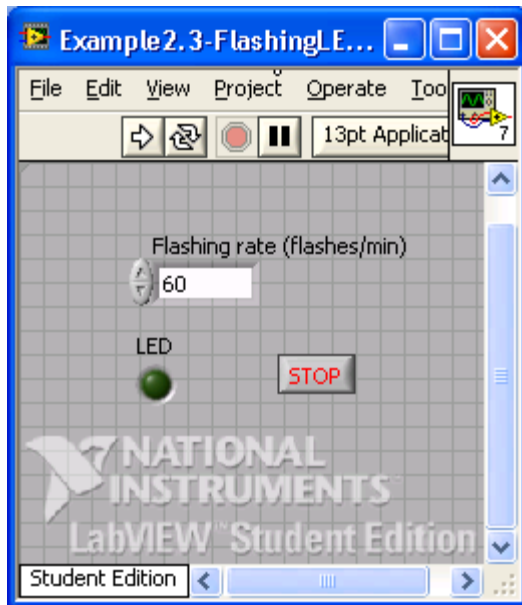
- Each front panel object has only one corresponding block diagram terminal. You can use **local variables** to access (read or write) front panel objects from more than one location in a single VI.



# Work Example 3.1 – Calculate the factorial n!

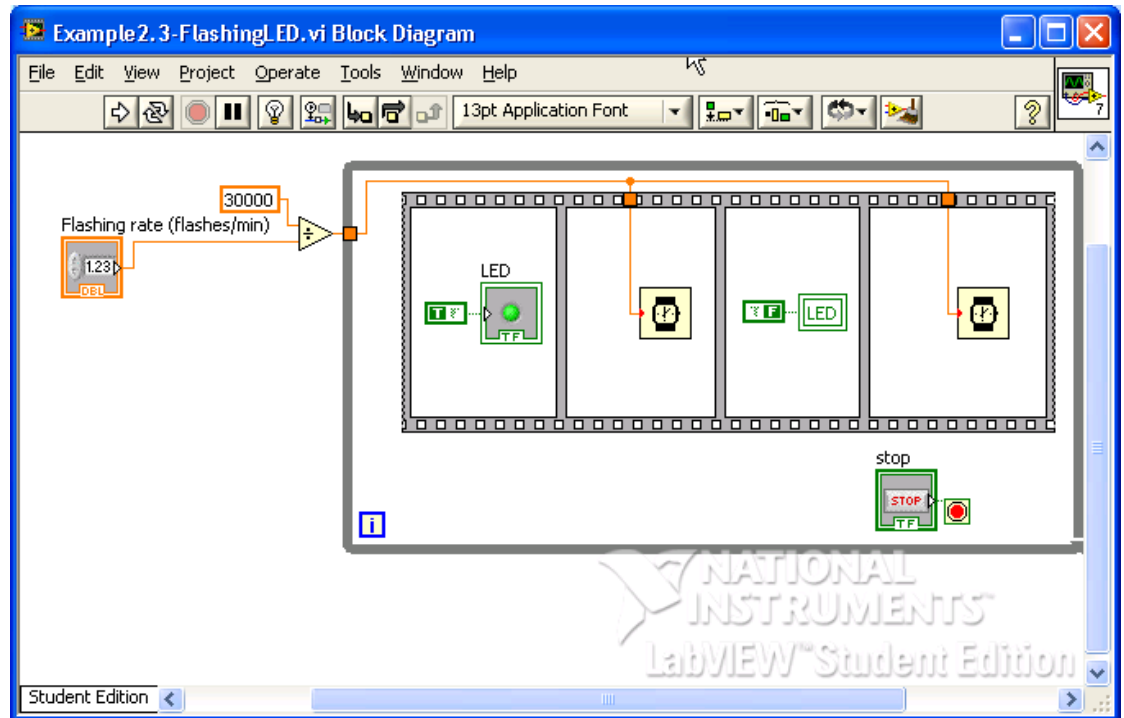


# Work Example 3.2 – Flashing LED



Block diagram?

# Answer



# BE/EE189 Design and Construction of Biodevices

## Lecture 4



# LabVIEW Programming – Arrays, Clusters, Matrix, Chart and Graph

- Arrays
- Polymorphism
- Clusters
- Matrix
- Memory Usage
- Waveform charts
- Waveform graphs
- XY graphs
- Math plots



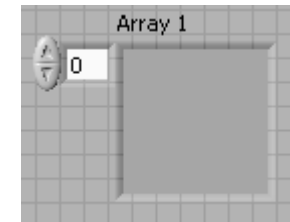
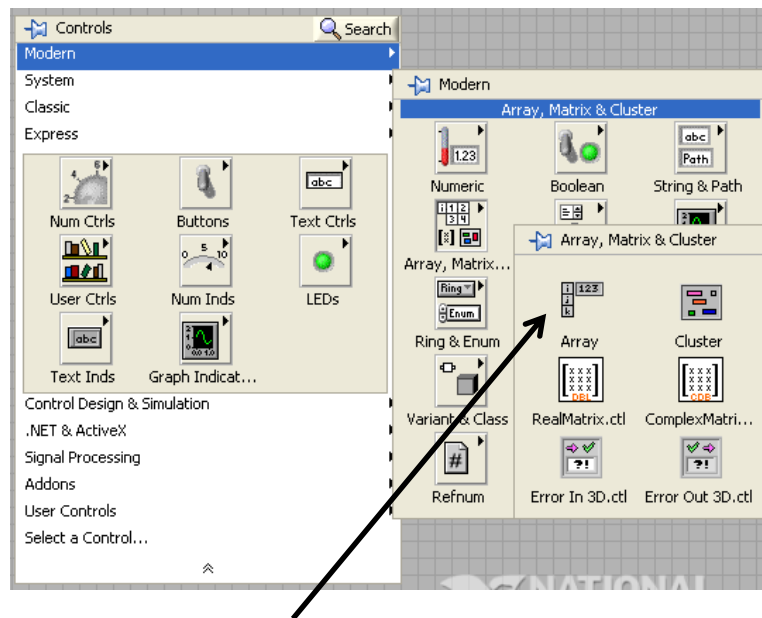


# Array

- Array – a *variable-sized* collection of data elements that are all the *same type*.
- Array can have one or more dimensions.
- If memory permitted, each dimension can have up to  $2^{31}-1$  elements.
- Cannot create an array of arrays, charts, or graphs.
- Can create an array of clusters which has one or more arrays.
- The index is zero-based.

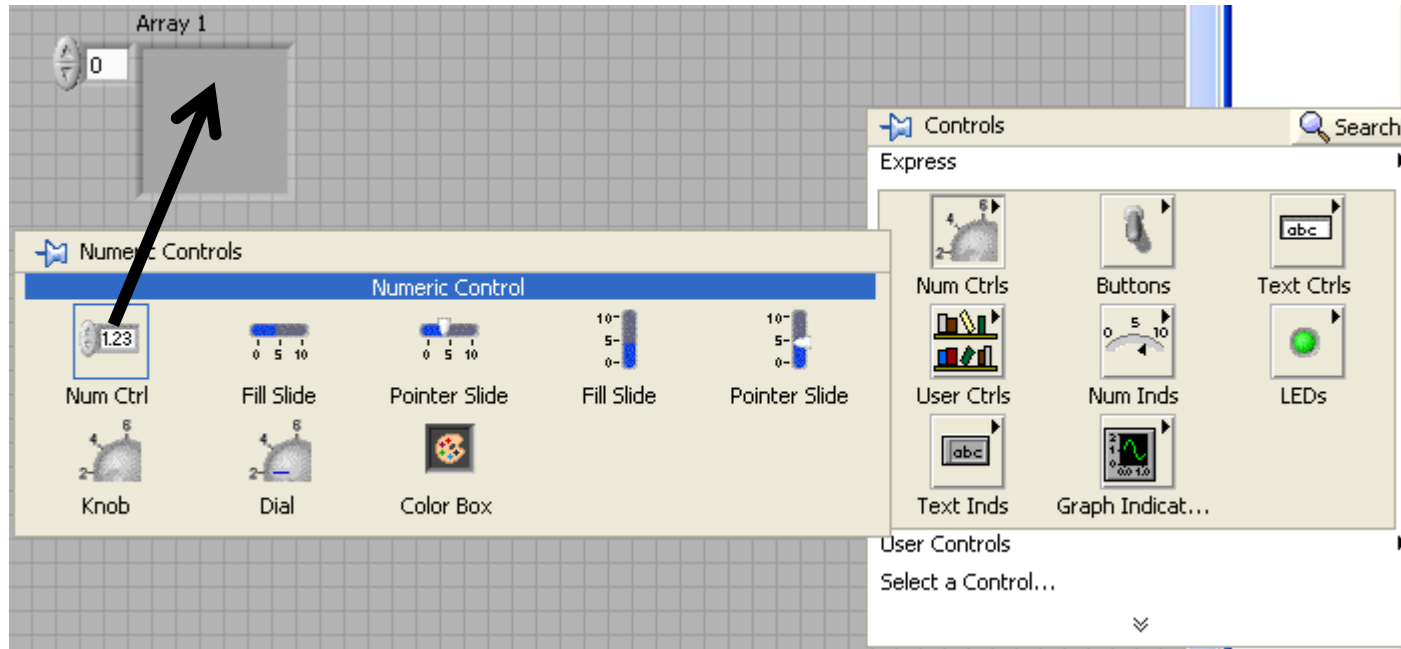


# Creating an Array – 1: Add an Array Shell

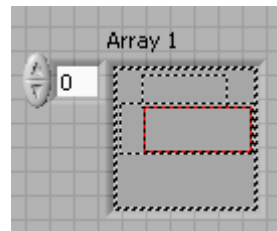


Array control

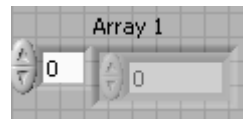
# Creating an Array – 2: Place data object into shell



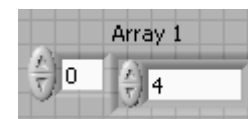
Add object



Becoming..



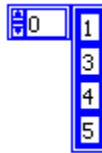
Set value



# Creating an Array Constant

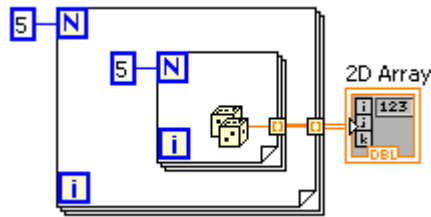
- Similar as previous procedures, except it happens in the block diagram.

An array constant



# Creating Arrays with Loops

- Example: create a 2D Arrays of random numbers



5x5 Array of random numbers

A screenshot of a 5x5 array of random numbers. The array is displayed in a grid format with a '2D Array' label at the top. The values are as follows:

0	0.48227	0.62126	0.73488	0.32592	0.40114	0
0	0.66592	0.27689	0.25197	0.42134	0.66651	0
	0.21557	0.10760	0.19329	0.12558	0.78125	0
	0.66715	0.62882	0.90216	0.08494	0.11220	0
	0.42717	0.77217	0.82293	0.39883	0.26359	0
0	0	0	0	0	0	0

# Array Functions

Functions

- Programming
- Measurement I/O
- Instrument I/O
- Vision and Motion

Programming

- Array
- Cluster, Class...
- Boolean
- String
- Timing
- Dialog & User...
- Waveform
- Application C...
- Graphics & So...
- Report Gener...

Array

- Array Size
- Index Array
- Replace Subset
- Insert Into Ar...
- Delete From ...
- Initialize Array
- Build Array
- Array Subset
- Max & Min
- Reshape Array
- Sort 1D Array
- Search 1D Ar...
- Split 1D Array
- Reverse 1D A...
- Rotate 1D Ar...
- Interpolate 1...
- Threshold 1D ...
- Interleave 1D...
- Decimate 1D ...
- Transpose 2D...
- Array Constant
- Array To Clus...
- Cluster To Ar...
- Array to Matrix
- Matrix to Array
- Matrix

Array size

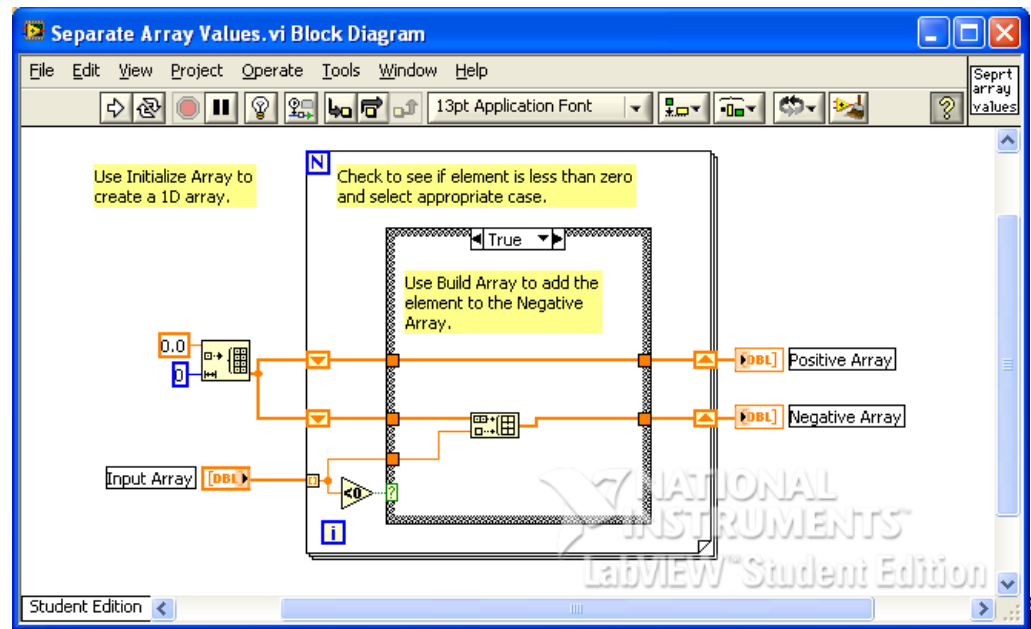
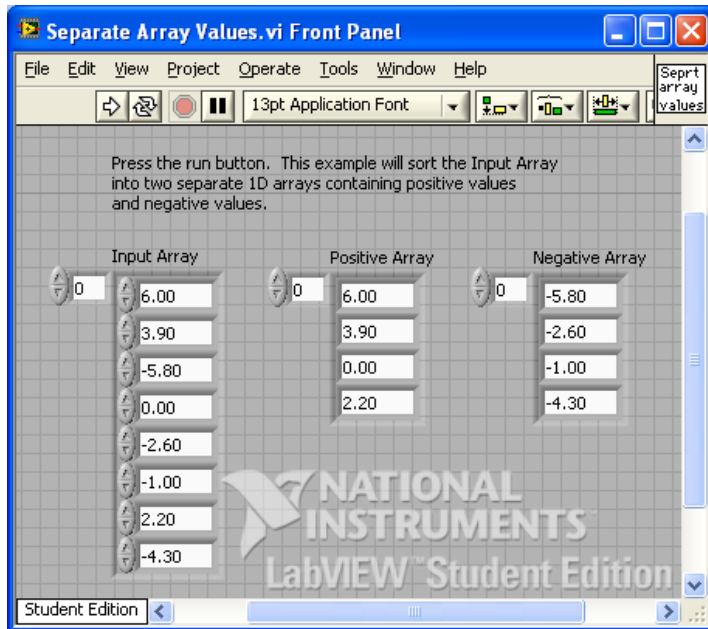
Initialize array

Index array

Array subset

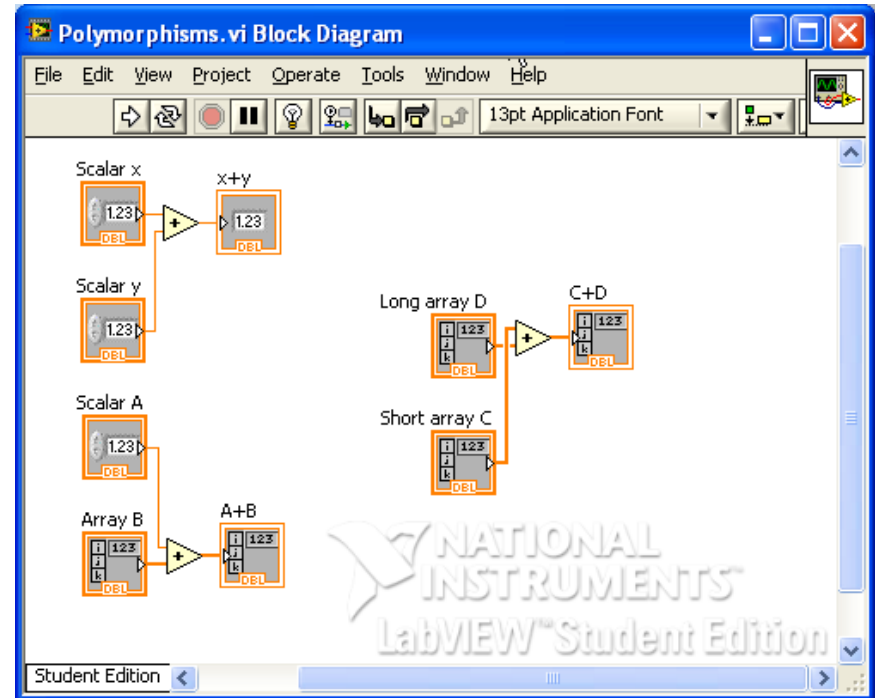
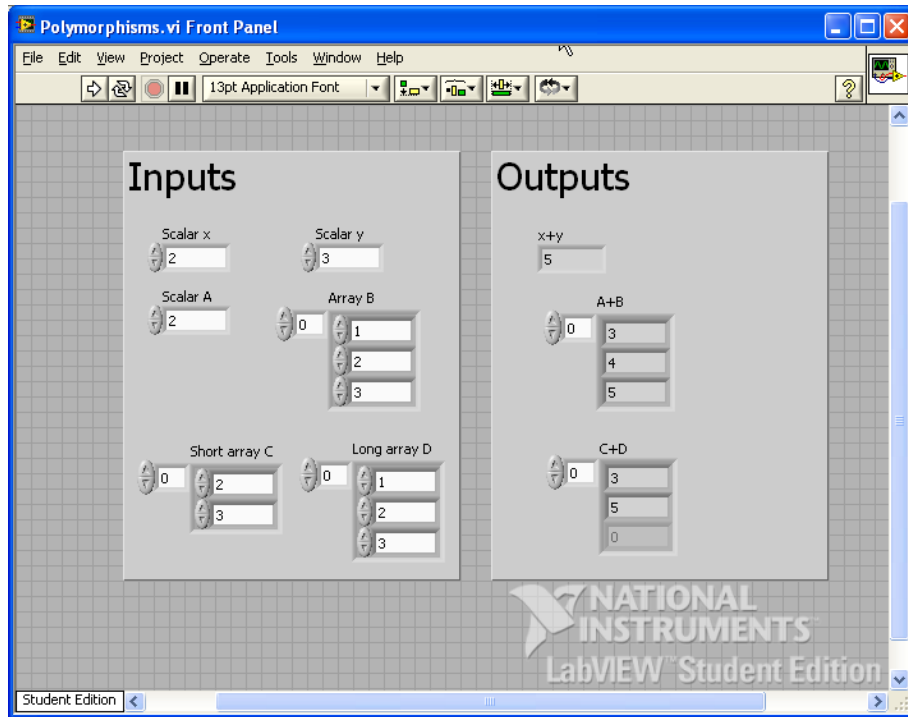
# Example – Separate Array Values

- Takes an input array that contains a mixture of positive and negative values and separates that array into two smaller arrays - one containing just the negative values and one containing just the positive values. (From LabVIEW examples)



# Polymorphism

- Polymorphism is the ability of certain LabVIEW functions to accept inputs of different dimensions and representations.



Notice that the sum of a short array and a long array is a short array.

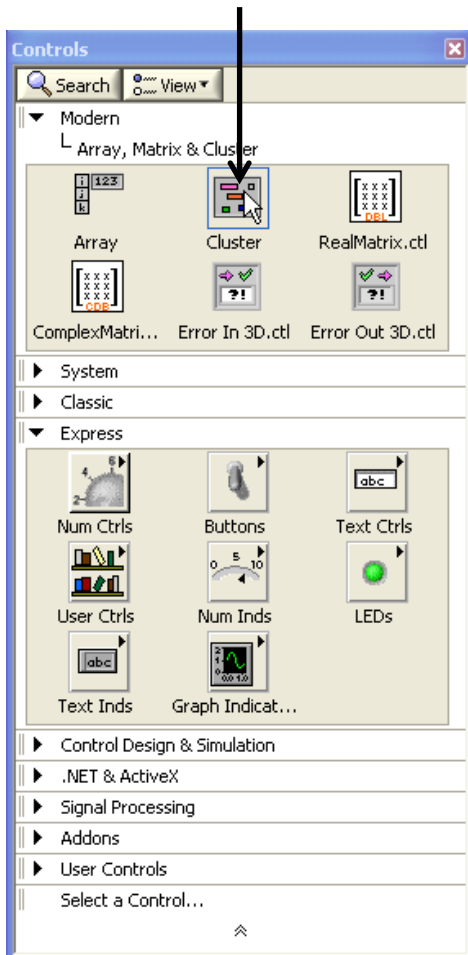


# Cluster

- Cluster – a *fixed-sized* collection of data elements of *mixed types*.
- Similar to *struct* in C
- Elements must be either all controls or all indicators

# Creating a Cluster

1: Add a cluster shell



2: Placing objects in the cluster shell

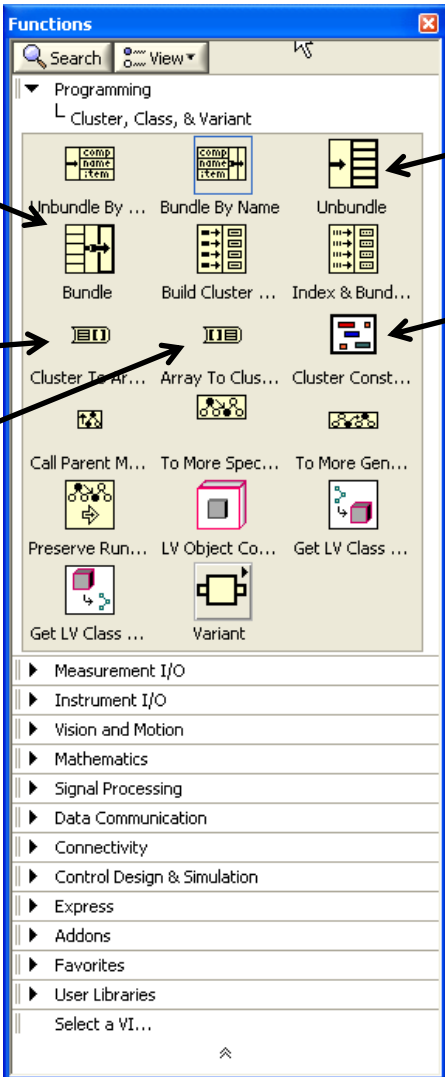


In block diagram



- A cluster constant can be created in the block diagram in a similar way.

# Cluster Functions



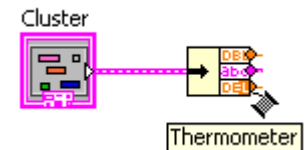
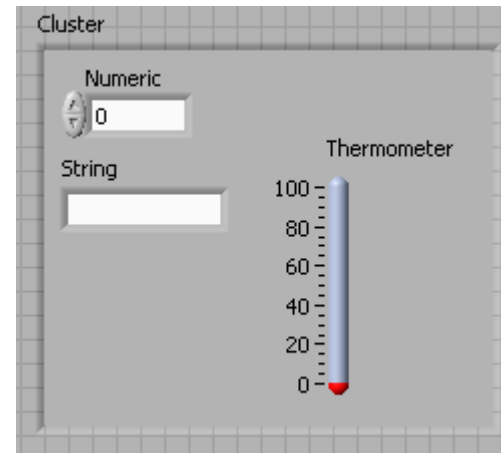
The screenshot shows the LabVIEW Functions palette with the following annotations:

- Bundle**: Points to the 'Bundle' icon in the 'Cluster, Class, & Variant' section.
- Unbundle**: Points to the 'Unbundle' icon in the 'Cluster, Class, & Variant' section.
- Cluster to array**: Points to the 'Cluster To Ar...' icon.
- Array to cluster**: Points to the 'Array To Clus...' icon.
- Cluster constant**: Points to the 'Cluster Const...' icon.

The Functions palette includes the following categories and items:

- Programming
  - Cluster, Class, & Variant
    - Unbundle By ...
    - Bundle By Name
    - Unbundle
    - Bundle
    - Build Cluster ...
    - Index & Bund...
    - Cluster To Ar...
    - Array To Clus...
    - Cluster Const...
    - Call Parent M...
    - To More Spec...
    - To More Gen...
    - Preserve Run...
    - LV Object Co...
    - Get LV Class ...
    - Variant
  - Measurement I/O
  - Instrument I/O
  - Vision and Motion
  - Mathematics
  - Signal Processing
  - Data Communication
  - Connectivity
  - Control Design & Simulation
  - Express
  - Addons
  - Favorites
  - User Libraries
  - Select a VI...

Unbundle example

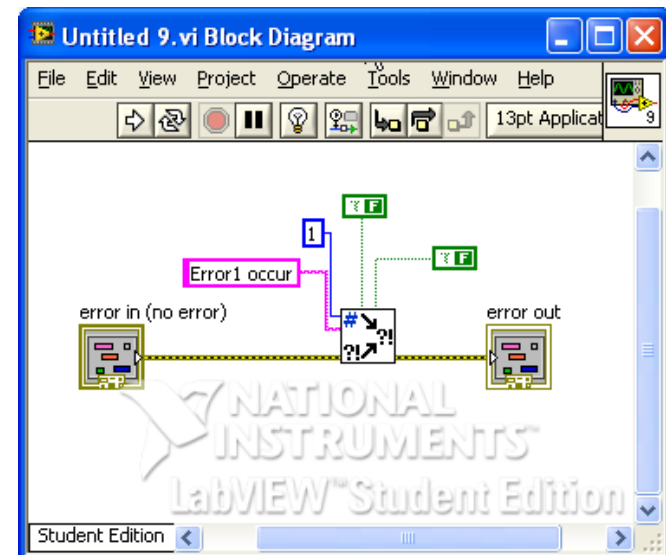
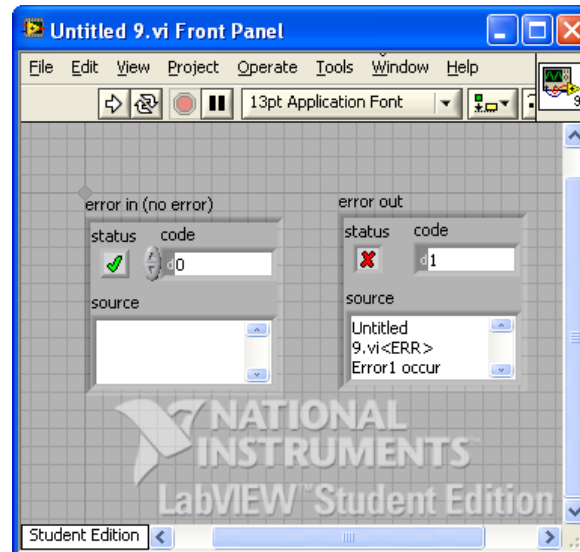
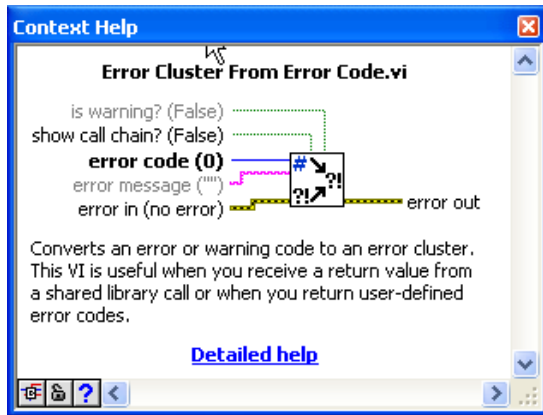


# Error Handling

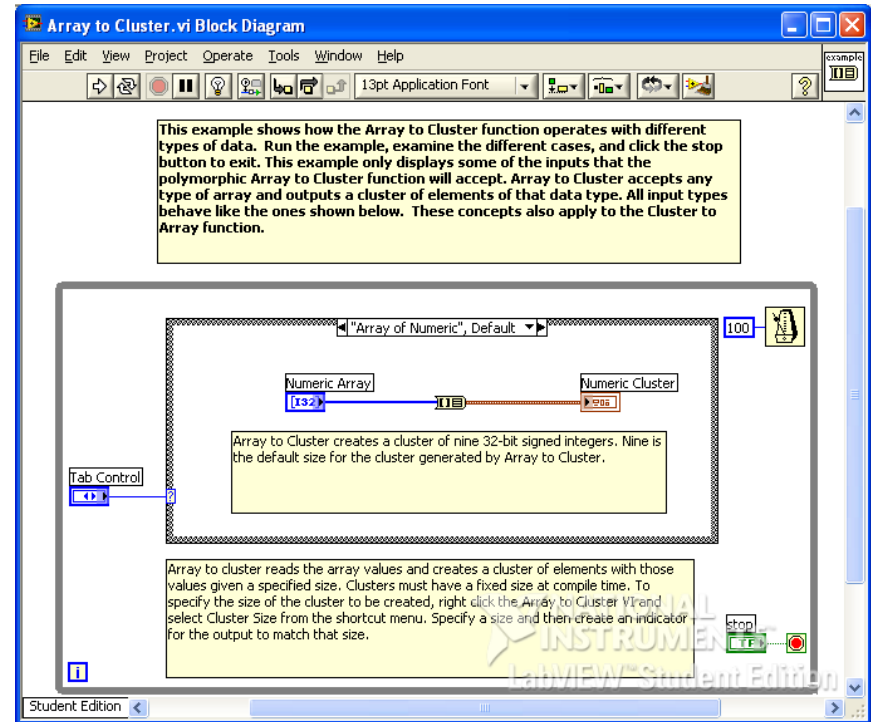
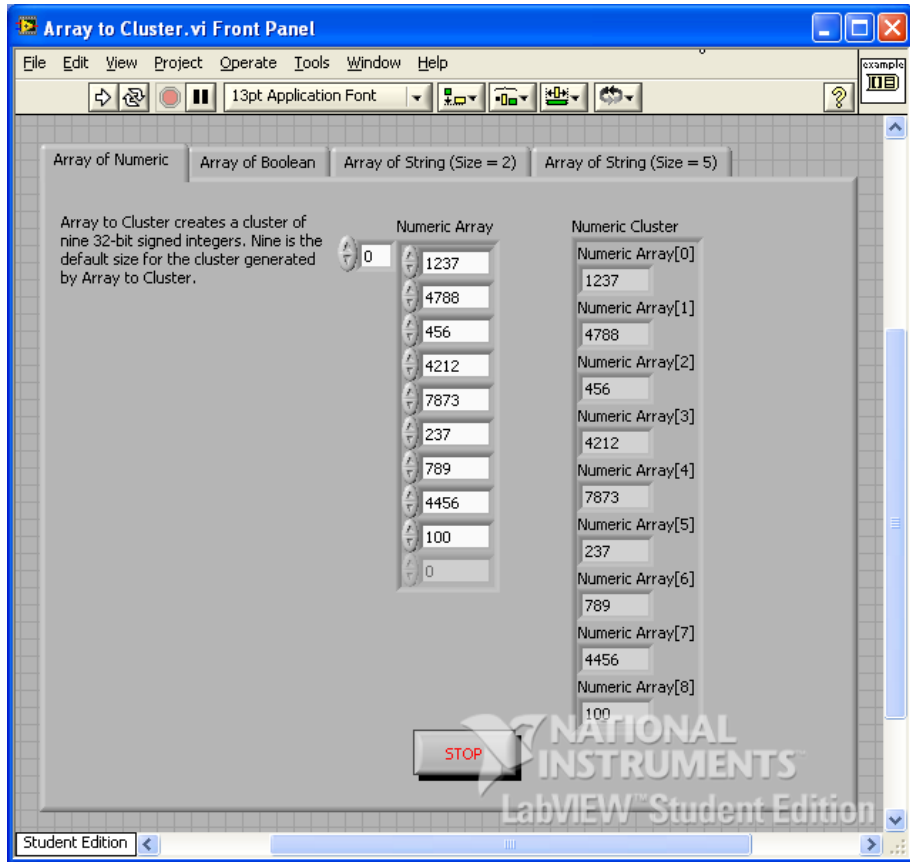
- Automatic error handling – displaying an error dialog box when error occurred. Each error has a numeric code and an error message. Can be disabled in *File >> VI Properties >> Execution*.
- Manual error handling: using error-handling VIs and functions on the **Dialog & User Interface** subpalette (found on the **Programming** palette).

# Error Clusters

- Error-cluster controls and indicators are used to create error inputs and outputs.
- Programming >> Dialog & User Interface >> Error Cluster From Error Code.

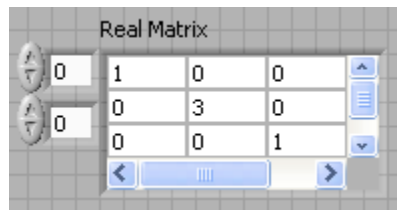


# Work Example – Array to Cluster (From LabVIEW Examples)



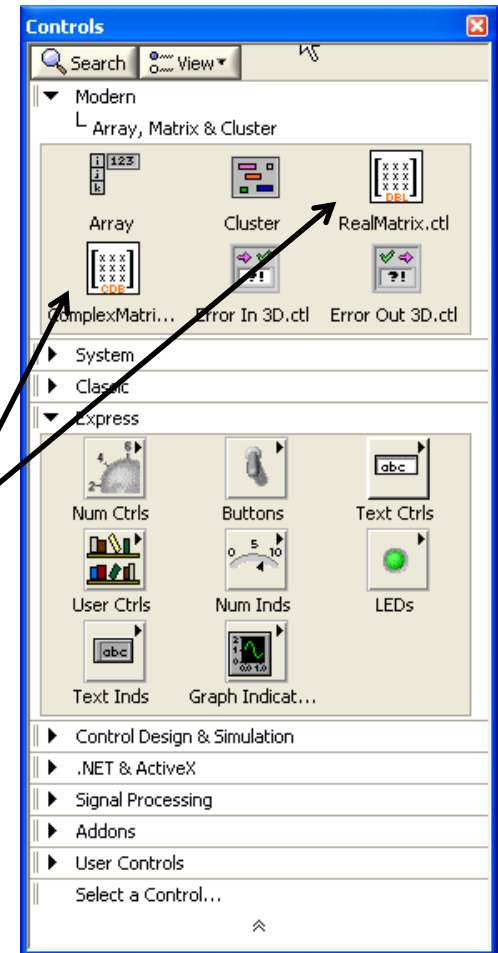
# Matrix

- Matrix can be used in place of two-dimensional arrays.
- Advantages: can use many linear algebra operations based on efficient matrix algorithms.



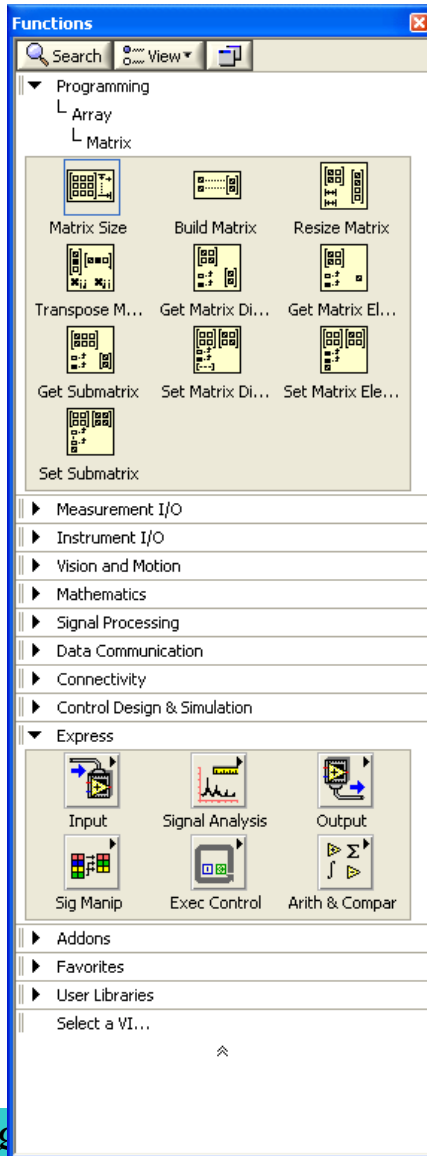
Complex matrix

Real matrix

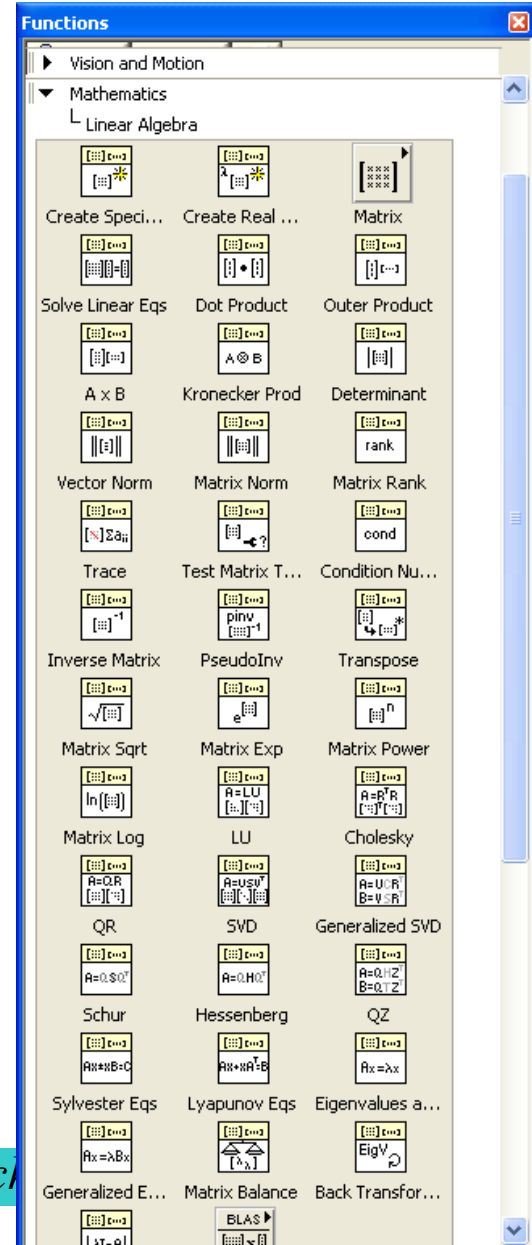


# Matrix Functions

Programming  
 >>Array>>Matrix



Mathematics>>  
 Linear Algebra





# Memory Usage

- Automatic memory handling – no need to worry when working with small sets of data.
- Some hints for working with large data sets:
  - Initialize large data sets, other than dynamically creating them.
  - Breaking a VI into subVIs.
  - Do not overuse local variables.
  - Unless needed, do not display large arrays and strings on open front panel.
  - Use consistent data types for array.
  - Refrain from using complicated hierarchical data types, such as clusters or arrays of clusters containing large arrays or strings.
  - Unless needed, do not use transparent and overlapped front panel objects.

# Charts and Graphs

Waveform chart

Waveform graphs

XY graph

2D math plots

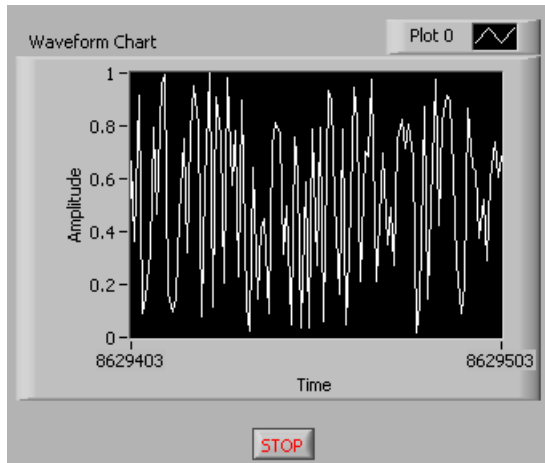
3D math plots

The screenshot shows a 'Controls' palette with a search bar and a 'View' dropdown. The 'Modern' category is expanded, showing a grid of chart icons. The 'Express' category is also visible below. The 'System', 'Classic', and 'Control Design & Simulation' categories are collapsed. The 'User Controls' category is also collapsed.

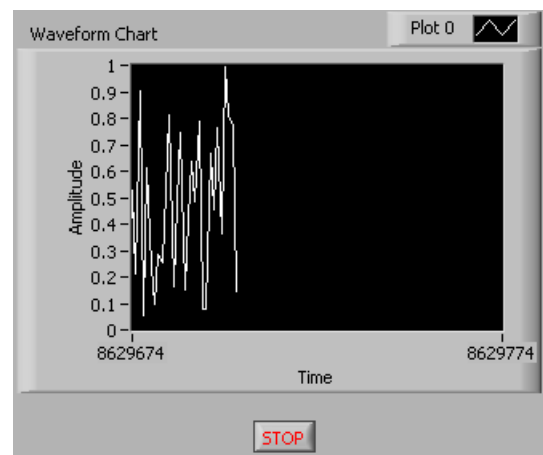
# Waveform charts

- Plot new data as they become available.
- Three update modes: **strip chart**, **scope chart**, and **sweep chart** .

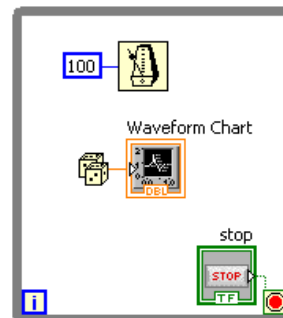
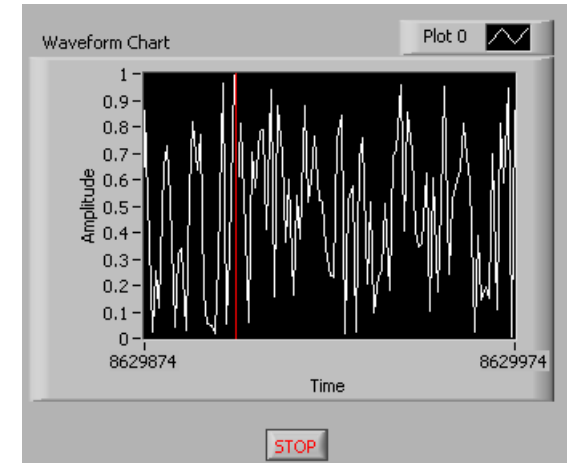
Strip chart



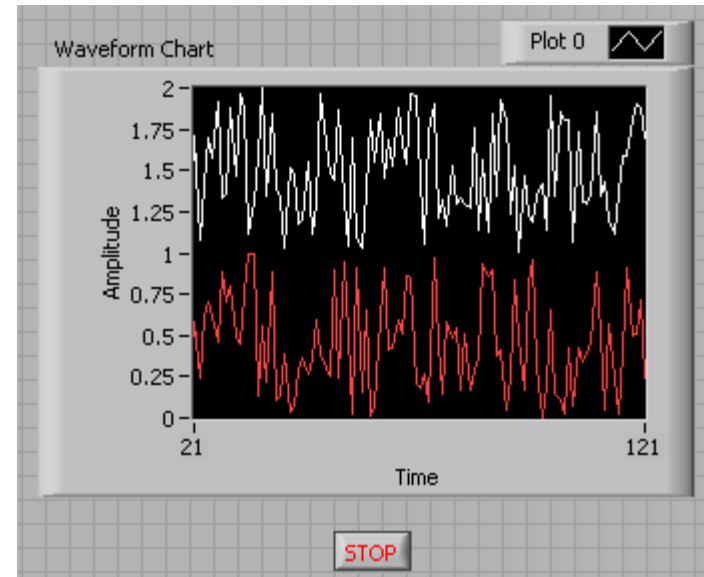
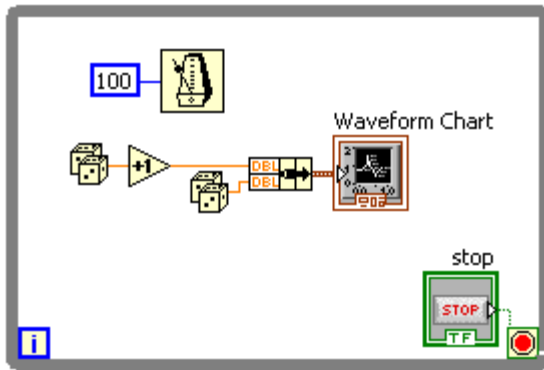
Scope chart



Sweep chart

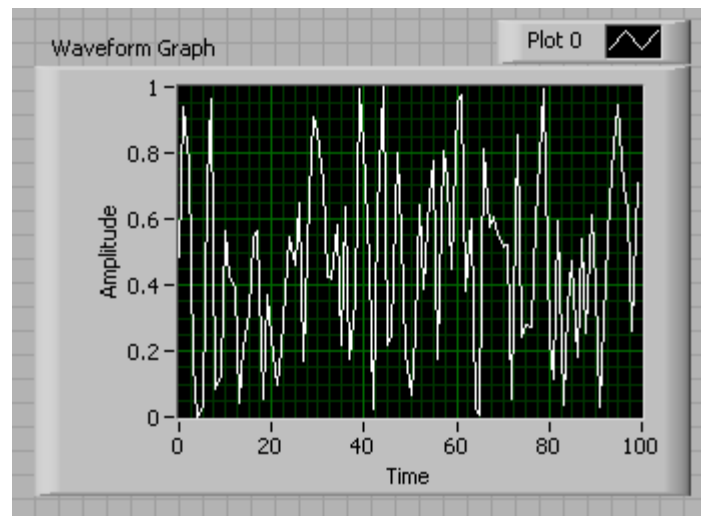
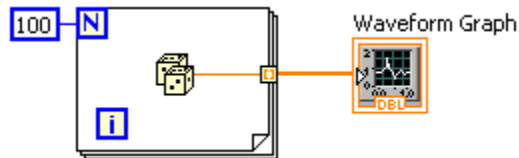


# Multi-plot Waveform Charts



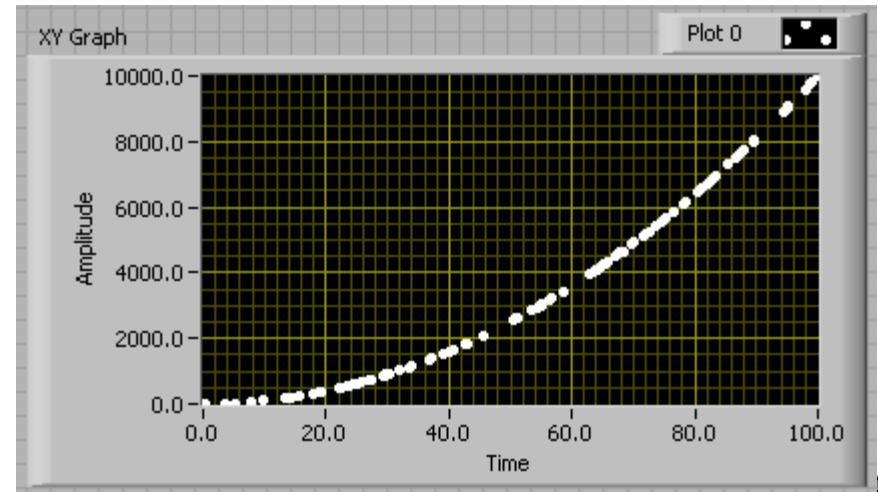
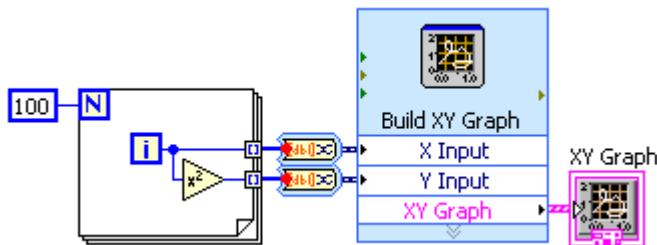
# Waveform Graphs

- Plot existing arrays of data all at once – different from waveform charts.



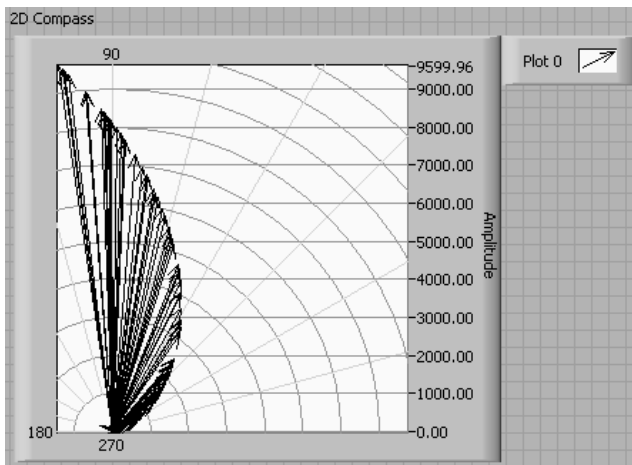
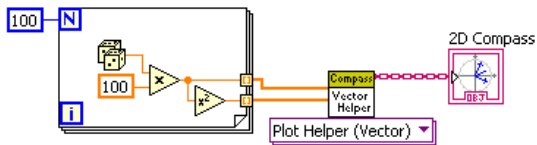
# XY Graphs

- Waveform graph is ideal for plotting evenly sampled waveforms.
- XY graph is more suitable for situations where you want to specify points using their (x, y) coordinates.
- Controls>>Express>>Graphs Indicators

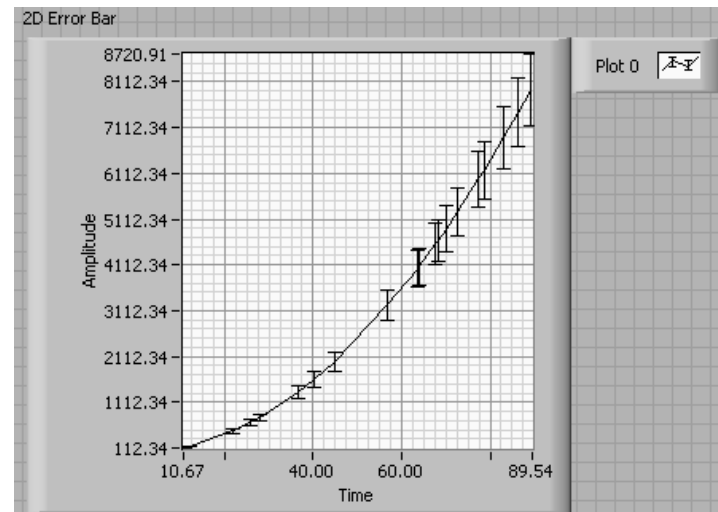
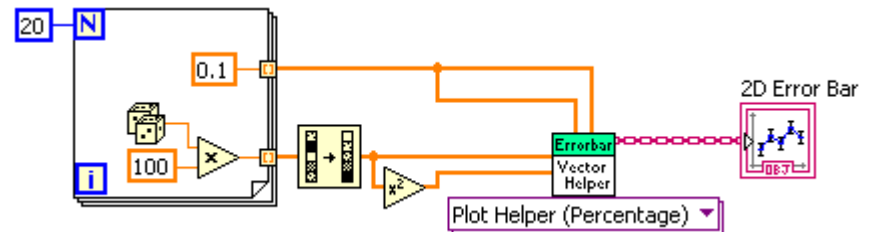


# 2D Math Plots

## 2D Compass



## 2D Error Bar

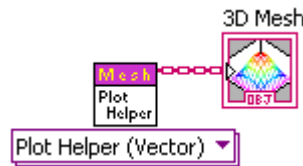
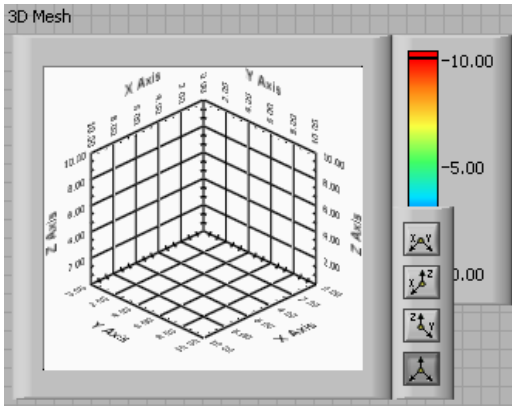


Others: 2D Feather and XY Plot Matrix

# 3D Math Plots

- To visualize data in three dimensions.
- Eleven types are available: Bar, Comet, Stem, Pie, Scatter, Surface, Contour, Mesh, Waterfall, Quiver, and Ribbon.

## Example



**Context Help**

**3D Mesh Datatype.lvclass:Plot Helper.vi**

3D Plot lvclass Array in 3D Plot lvclass Array out  
 x vector error out  
 y vector   
 error in (no error)   
 z matrix   
 Plot ID

Setup VI for 3D Mesh Plot. Converts mesh plot data to generic 3D data..

The **Vector** instance creates a datapoint from the *i*-th element of the **x vector** input, *j*-th element of the **y vector** and *ij*-th element of the **z matrix**.

The **Matrix** instance creates a datapoint from the *ij*-th element of each of the **matrix** inputs.

If only **z matrix** is provided, each *ij*-th element of *z* is plotted against *i* and *j*.

Successive calls to this VI with the same 3D Plot input with different **Plot ID values will create multiple plots.**

**Controls**

Search View

Modern

Graph

3D Graph

Scatter Bar

Pie Stem

Ribbon Contour

Quiver Comet

Surface Mesh

Waterfall Surface Graph

Parametric Gr... Line Graph

System

Classic

Express

Num Ctrls Buttons

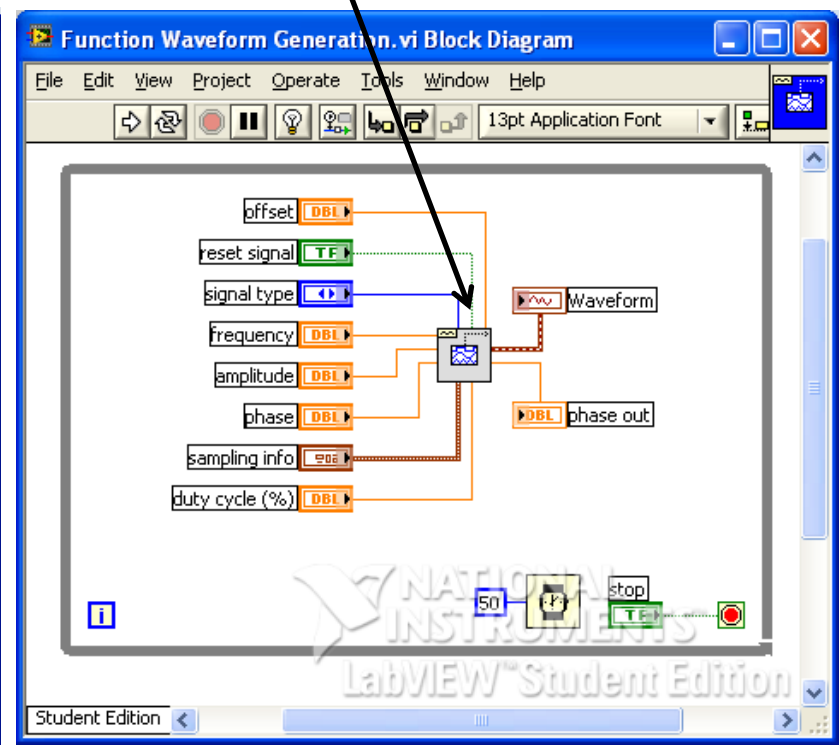
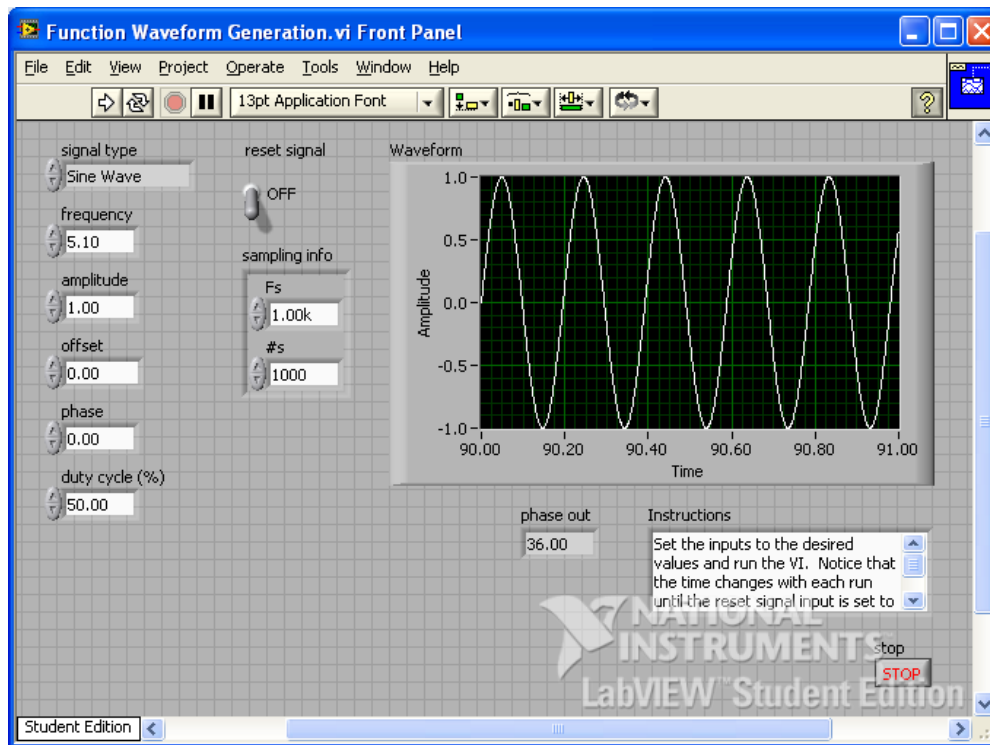
Text Ctrls User Ctrls

Num Inds LEDs



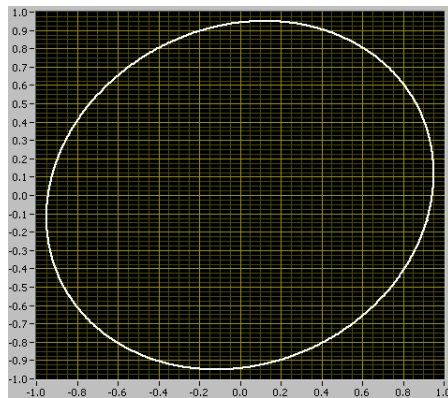
# Example – Function Waveform Generation

Functions palette>>Signal processing>>Waveform generation>>Basic function generator

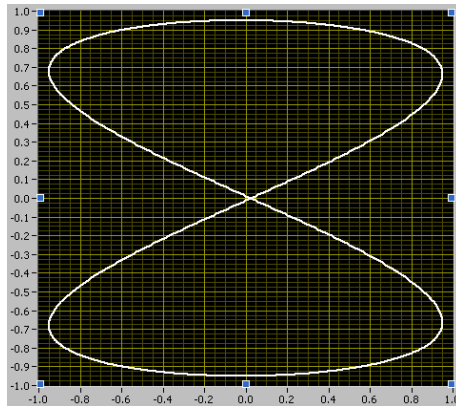


# Example – Lissajous Curve

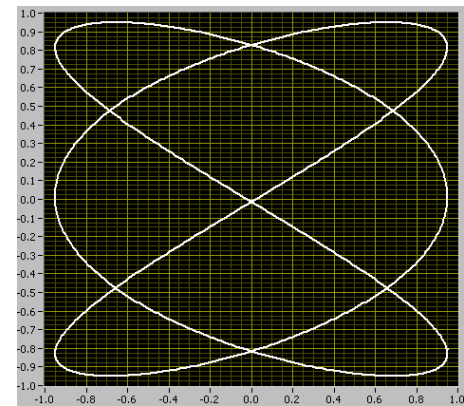
- Lissajous curve is an XY graph of  $x = A \sin(at + \delta)$ ,  $y = B \sin(bt)$ ,  
The appearance of the figure is very sensitive to the ratio  $a/b$
- Examples



$a=b$



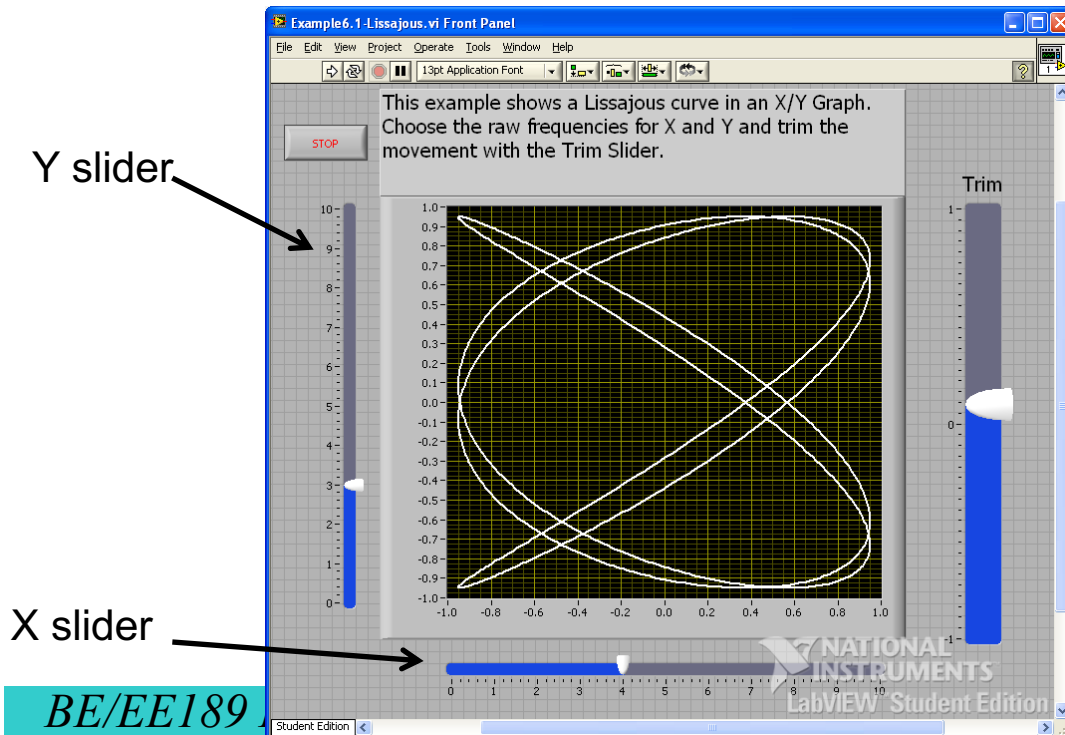
$a=2b$



$a/b = 3/2$

# Example – Lissajous Curve

- In this example, we change the frequency of X and Y signal and plot the Lissajous curve. The trim value is used to slightly adjust the frequency of X signal to see the movement of the curve.



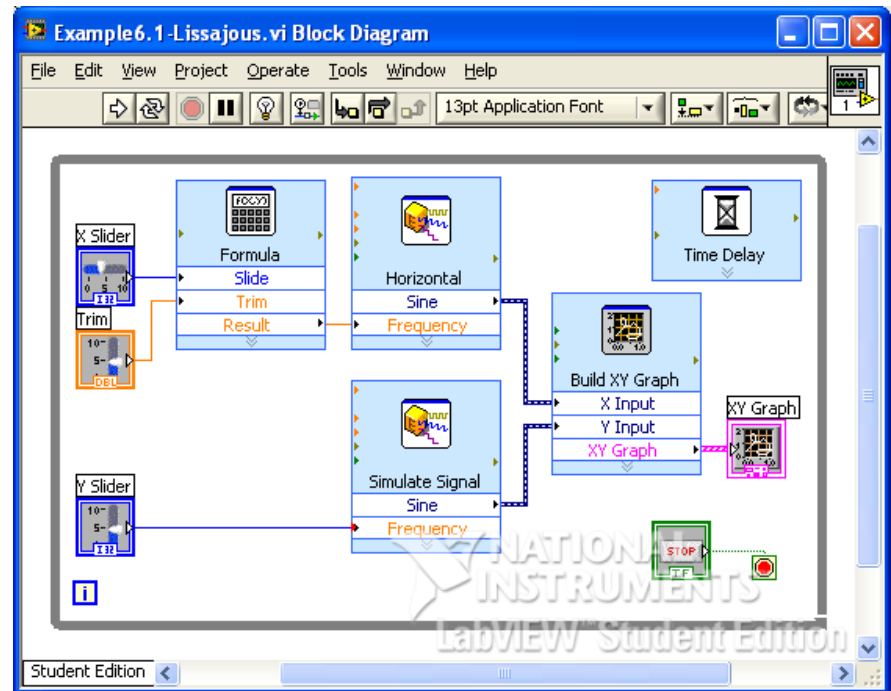
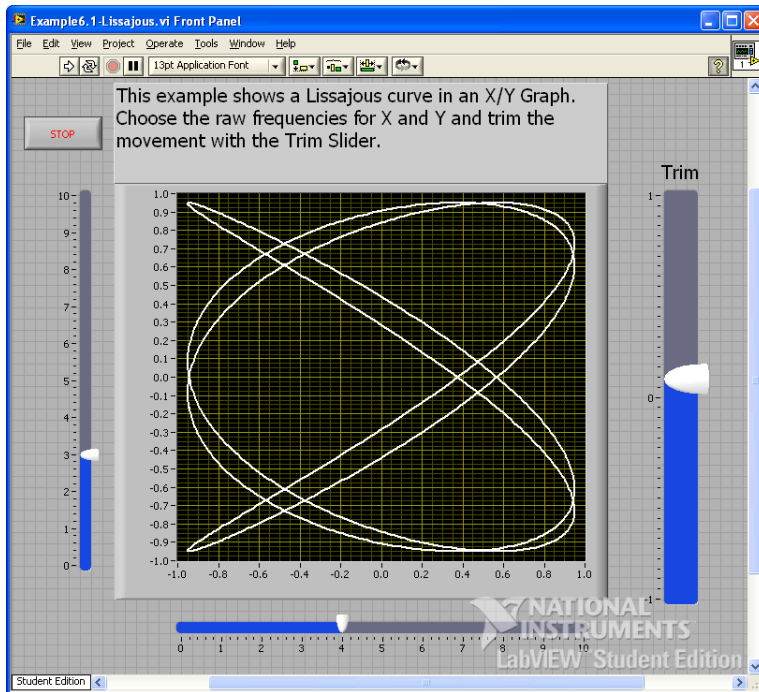
X frequency = X slider + Trim / 200

Y frequency = Y slider

Functions palette >> Express >> Signal analysis >> Simulate signal



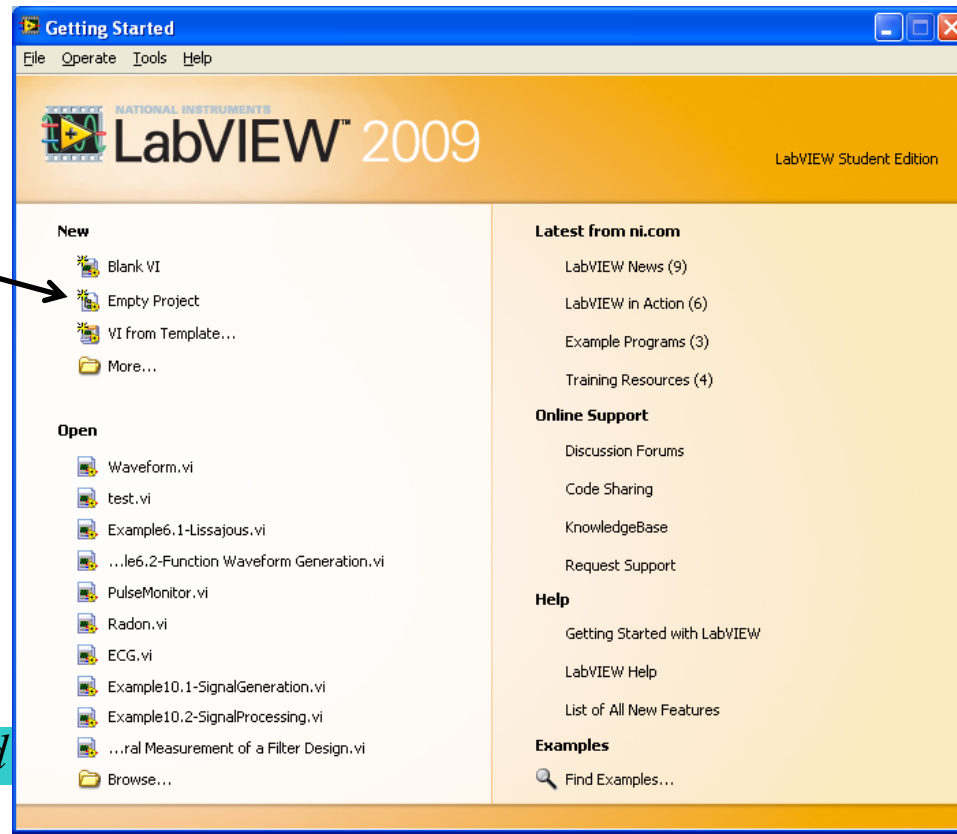
# Example – Lissajous Curve (From LabVIEW Examples)



# Stand-alone Application

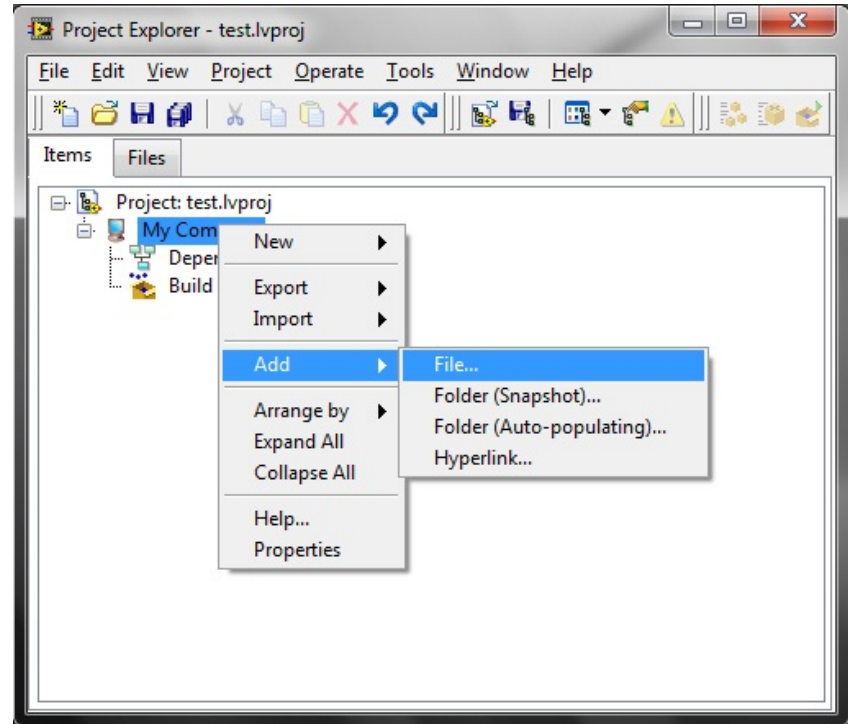
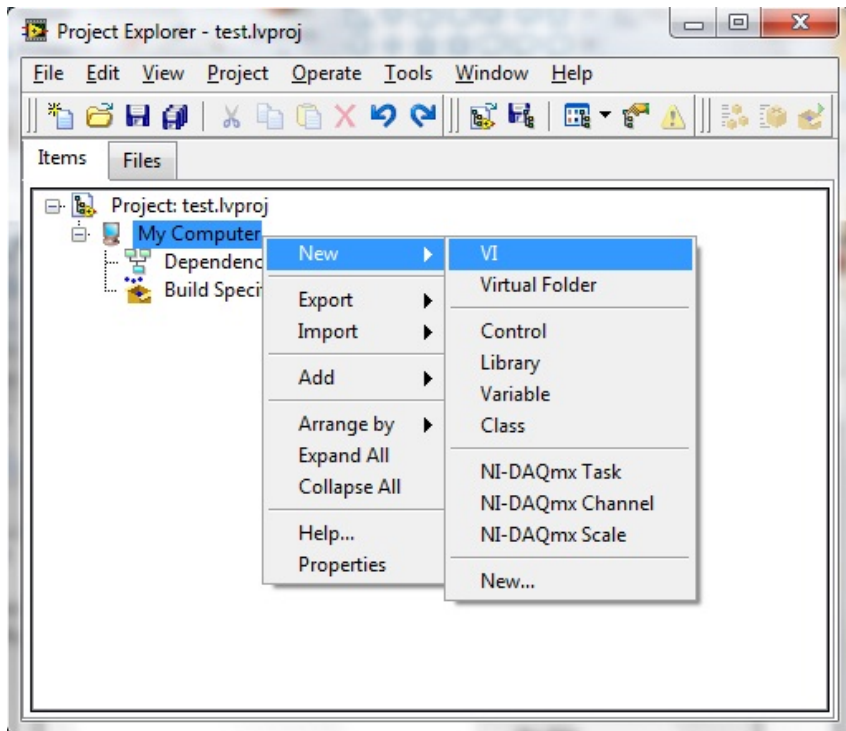
- You can use the Application Builder to build a stand-alone application.
- You need to use a LabVIEW project for the building.

New empty project

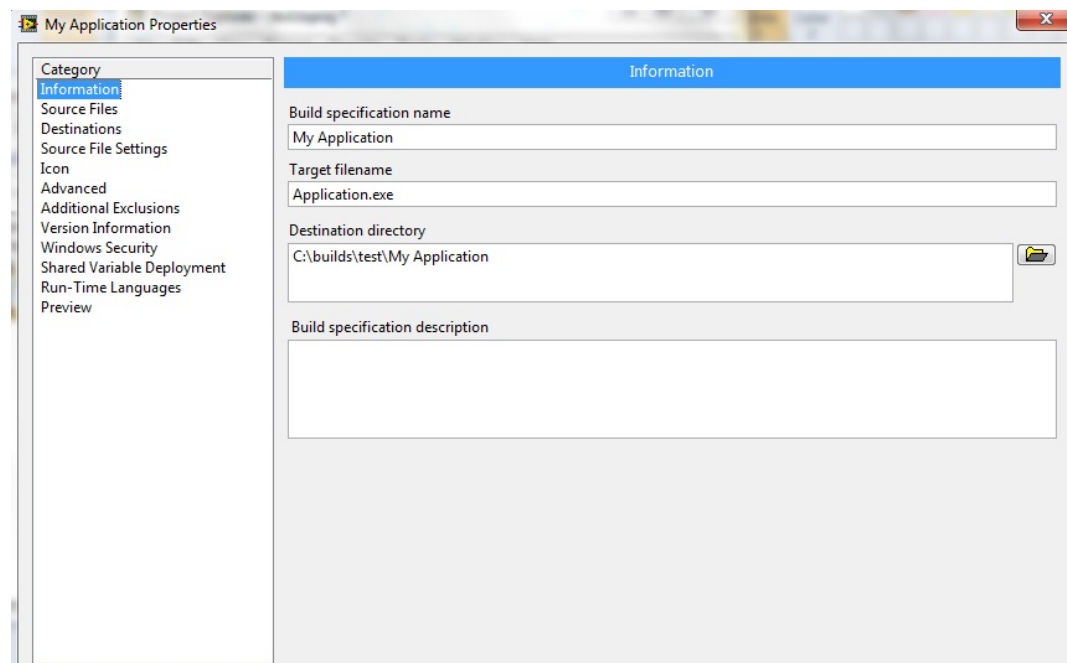
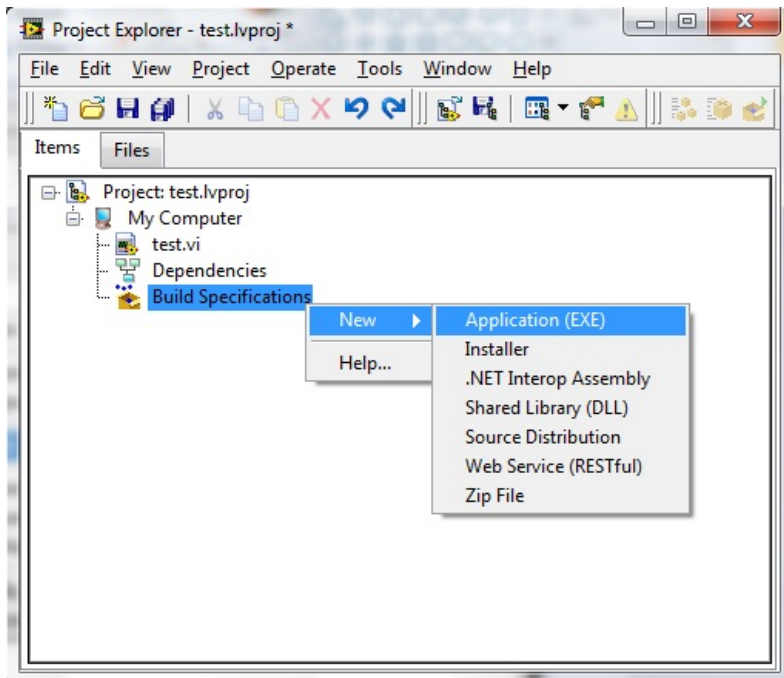


# Stand-alone Application – Add VI to Project

- You can either create a new VI or add existing VI to the project.

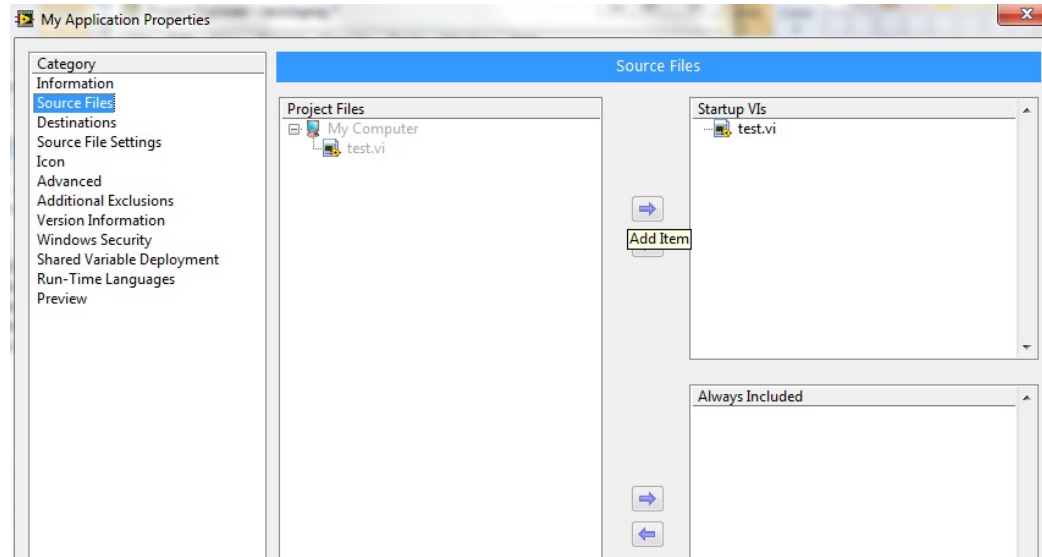


# Stand-alone Application – Create an Application

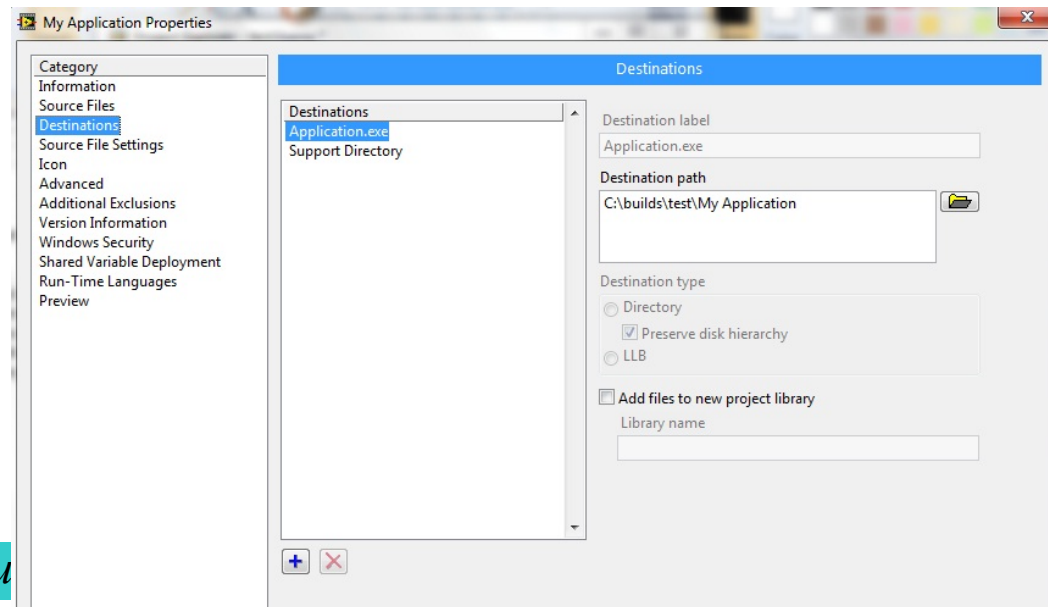


# Stand-alone Application – Configure the Application

Specify startup VI



Choose destination

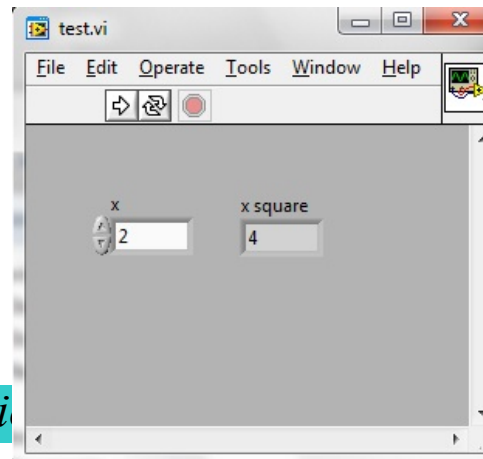




# Stand-alone Application – Build

The image shows two windows from the LabVIEW IDE. On the left is the Project Explorer for a project named 'test.lvproj'. The tree view shows 'My Computer' containing 'test.vi', 'Dependencies', 'Build Specifications', and 'My Application'. A context menu is open over 'My Application', with 'Build' selected. On the right is the 'My Application Properties' dialog box. The 'Destinations' tab is active, showing a list of destinations with 'Application.exe' selected. The 'Destination label' is 'Application.exe' and the 'Destination path' is 'C:\builds\test\My Application'. The 'Destination type' is set to 'Directory' with 'Preserve disk hierarchy' checked. At the bottom right of the dialog, there is a large 'Build' label with a downward arrow pointing to the 'Build' button.

Stand-alone application (\*.exe)



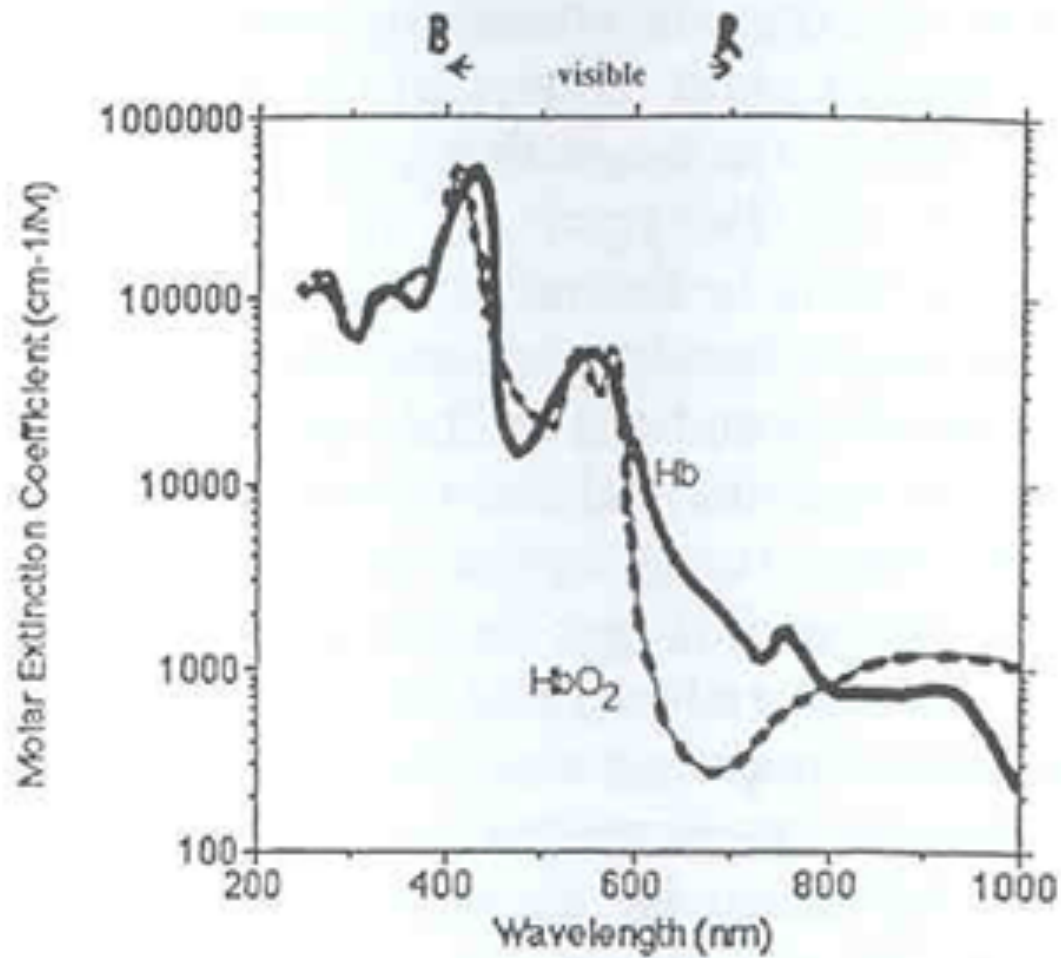


Figure 11 UV-visible Spectra of oxy- (HbO<sub>2</sub>) and deoxyhaemoglobin (Hb)

# BE/EE189 Design and Construction of Biodevices

## Lecture 5



# LabVIEW Programming – Data acquisition

- DAQ system
- Signals and signal conditioning
- Nyquist frequency
- NI ELVIS II
- NI-DAQmx and DAQ assistant



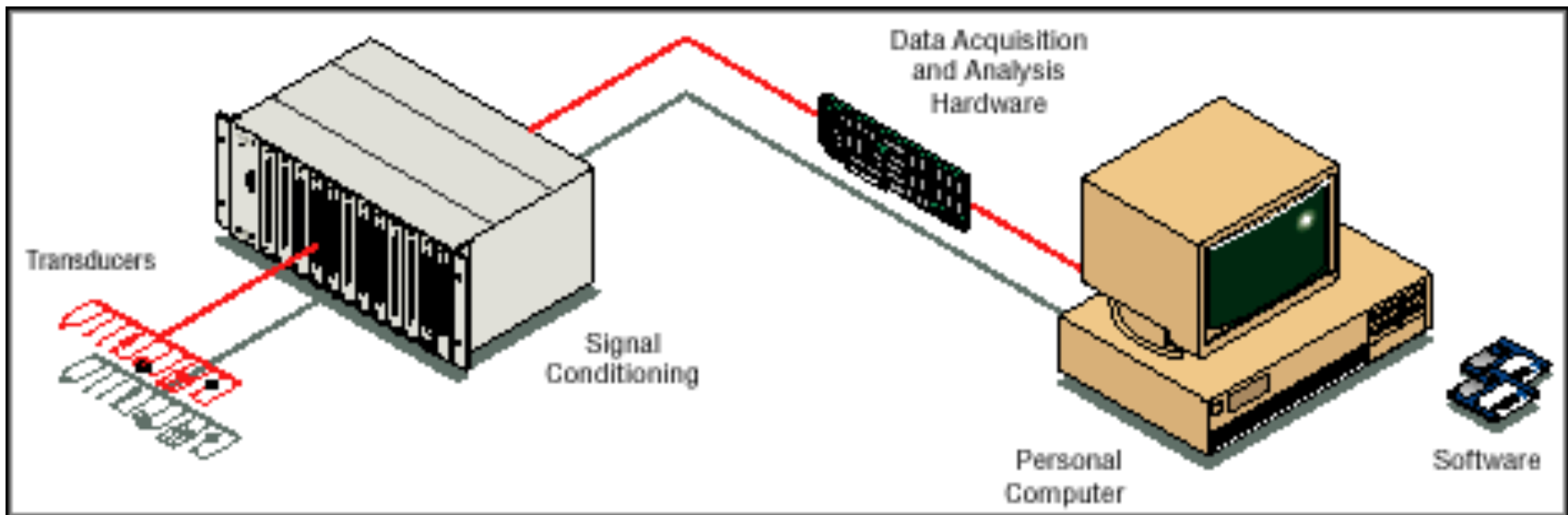
# LabVIEW Programming – NI-DAQmx, Strings and File I/O

- Using NI-DAQmx VIs
- Strings
- File I/O



# DAQ System

- Data acquisition (DAQ) is the measurement or generation of electrical signals.



# Types of Signals and Signal Conditioning

- Digital signals: On-Off, Pulse Train
- Analog signals
  - DC: static or slow changing signals
  - AC: fast changing signals
- Signal conditioning: manipulating the signal before further processing, e.g.,
  - Amplification
  - Filtering
  - Isolation



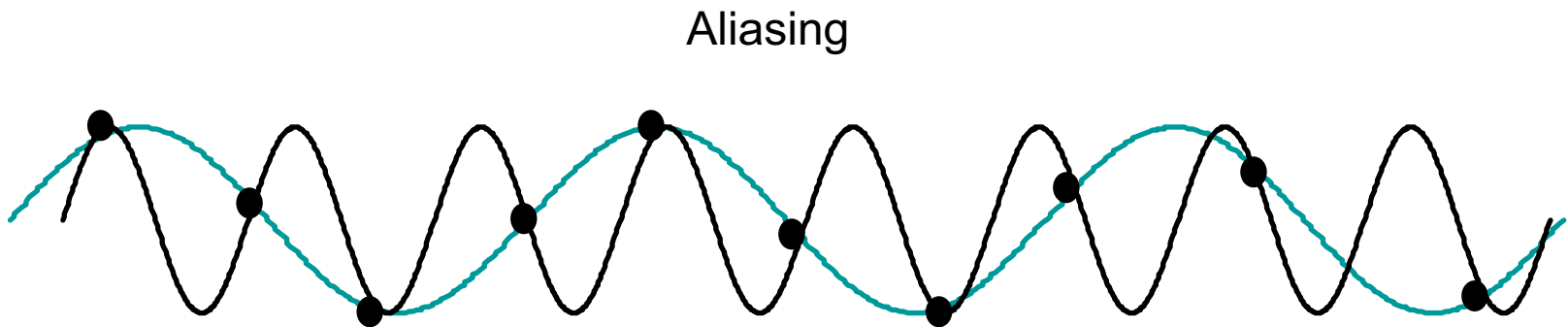
# Common Transducers

Phenomenon	Transducer
Temperature	Thermocouples Resistance temperature detectors (RTDs) Thermistors Integrated circuit sensor
Light	Vacuum tube photosensors Photoconductive cells
Sound	Microphone
Force and pressure	Strain gauges Piezoelectric transducers Load cells
Position (displacement)	Potentiometers Linear voltage differential transformer (LVDT) Optical encoder
Fluid flow	Head meters Rotational flowmeters Ultrasonic flowmeters
pH	pH electrodes

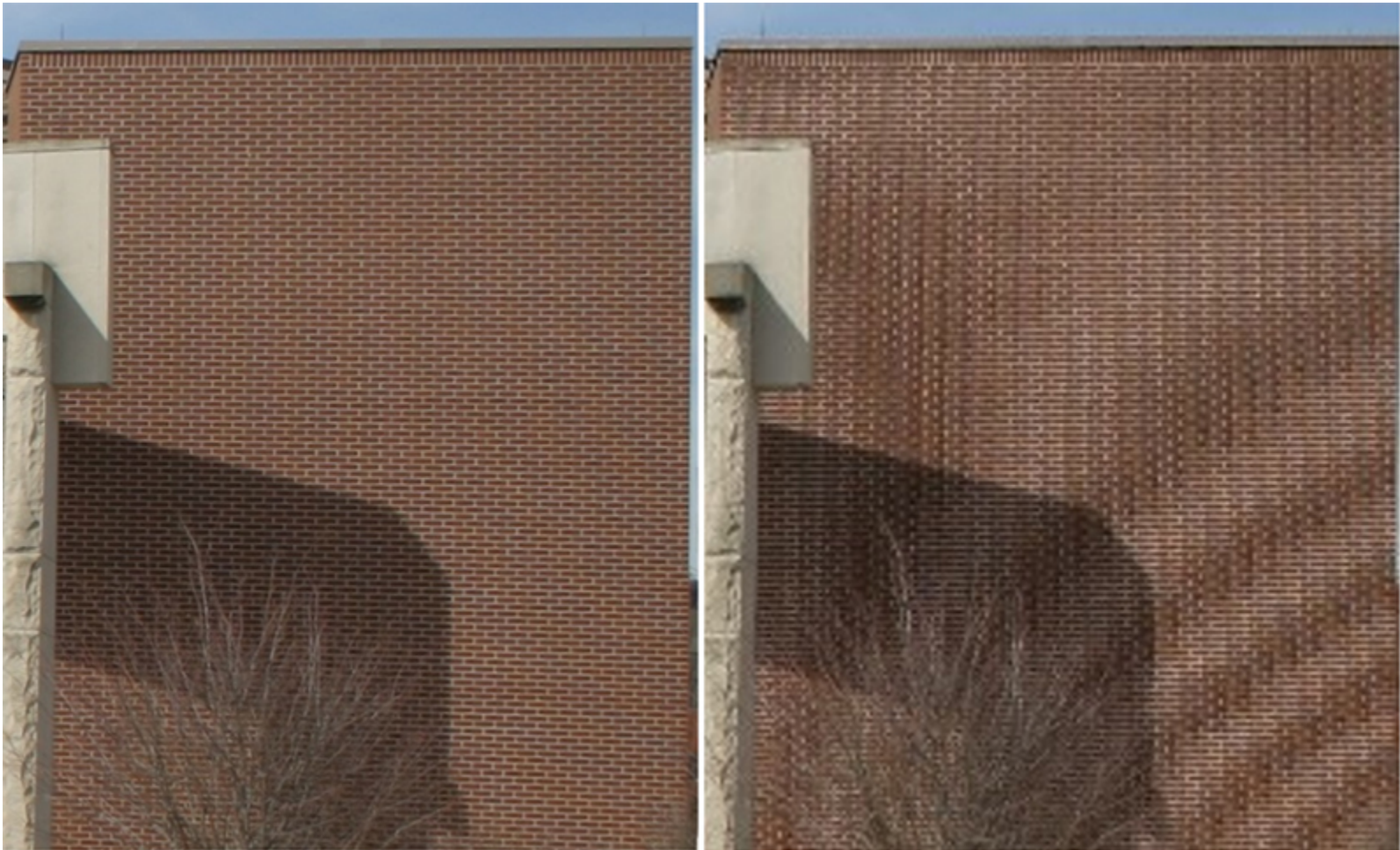


# Sampling and Nyquist Frequency

- For a given sampling rate, we can only recover signals with maximum frequency less than the Nyquist frequency, which is half of the sampling rate.
- Aliasing will occur if the maximum signal frequency is larger than the Nyquist frequency.

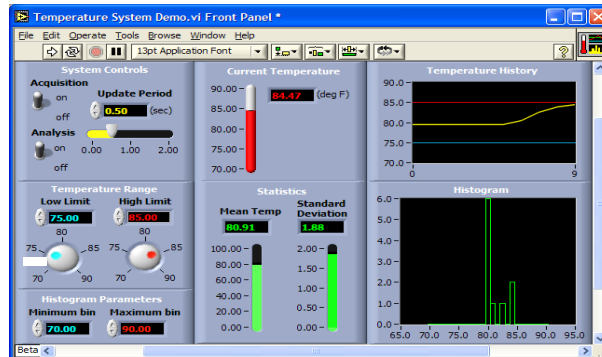


# Aliasing

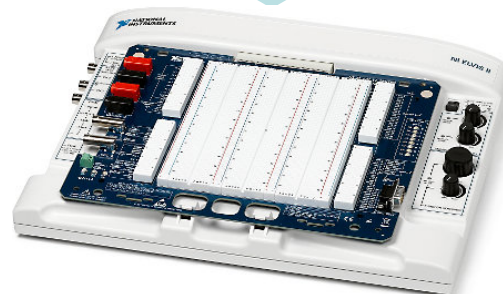


# NI Data Acquisition Framework

## LabVIEW



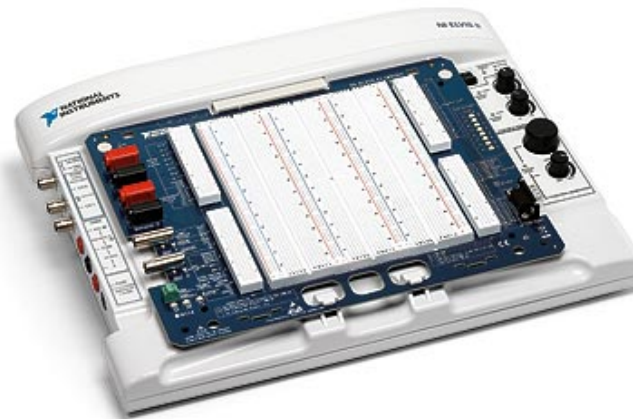
## NI-DAQmx



We will use NI ELVIS II

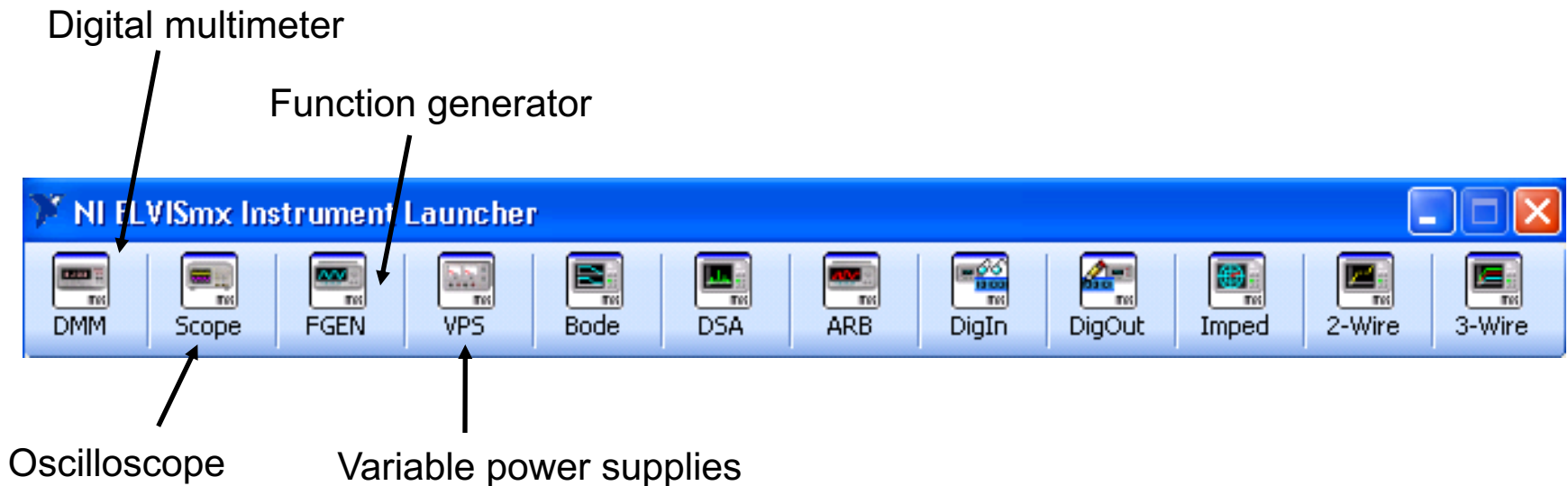
# NI ELVIS II

- ELVIS - Educational Laboratory Virtual Instrumentation Suite
- High-speed USB plug-and-play connectivity
- 12 virtual instruments: oscilloscope, digital multimeter, function generator, variable power supply, etc.
- Bread board for circuit prototyping.



# Virtual Instruments of NI ELVIS II

## Instrument launcher



# NI ELVIS II - Circuit Prototyping

- Fixed power supply: +5V, +/-15V.
- Variable power supply: 0 to 12V, 0 to -12V.
- 16 single-ended, 16-bit analog input, maximum 1.25 MS/s sampling rate.
- Two 16-bit analog outputs (2.8 MS/s); 24 digital I/O.
- LEDs for indication.

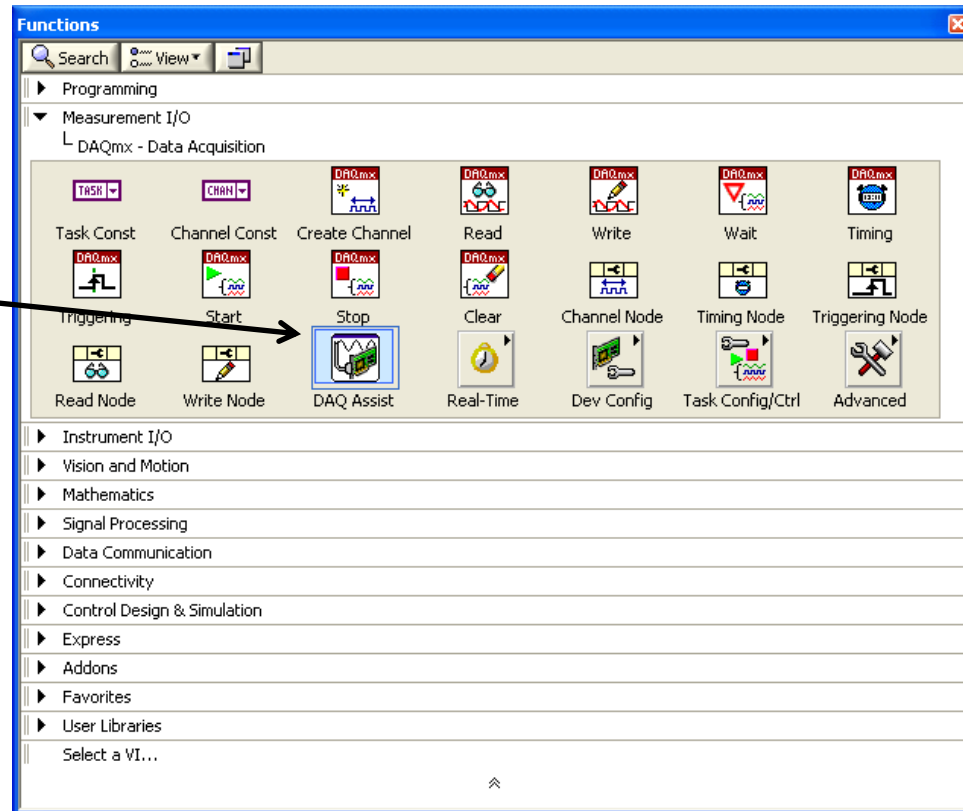
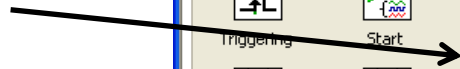


# NI-DAQmx

- NI-DAQmx: a DAQ driver architecture with significant improvement over previous NI-DAQ drivers.
- Physical channel: a terminal or pin at which an analog or digital signal is measured or generated
- Virtual channel: a collection of property settings that can include a name, a physical channel, input terminal connections, the type of measurement or generation, and scaling information.
- Task: a collection of one or more virtual channels with timing, triggering, and other properties.

# NI-DAQmx-Data Acquisition Palette

DAQ Assistant





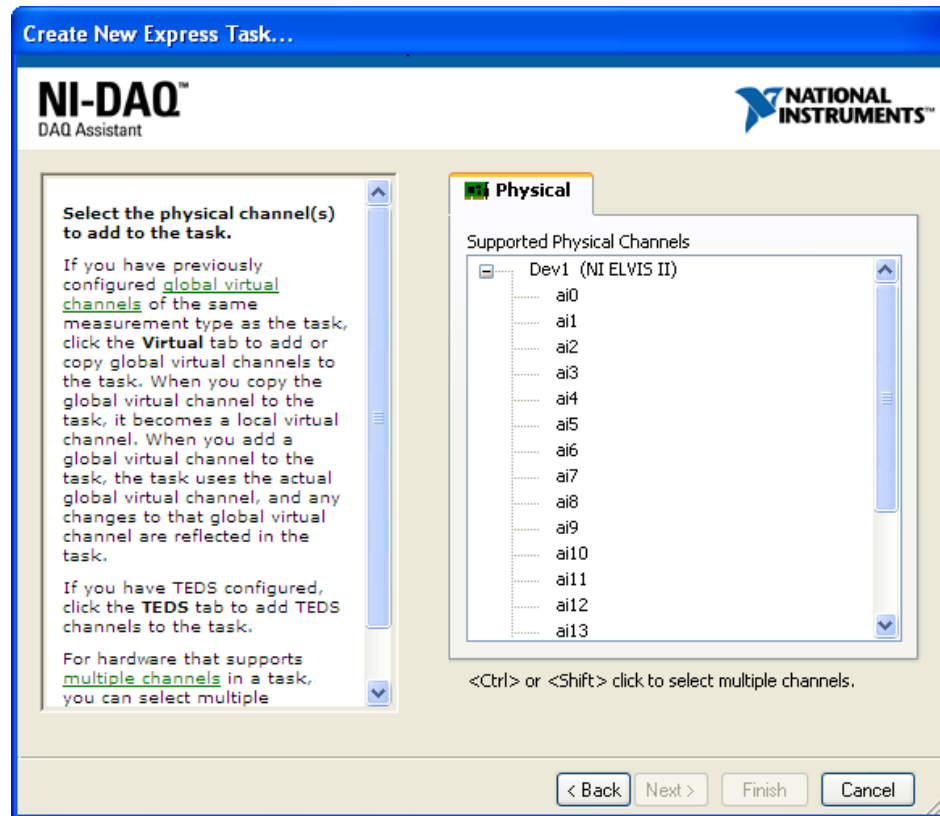
# Using DAQ Assistant – 1

- You can acquire or generate signals
- Example: select “Acquire Signals >> Analog Input >> Voltage” for acquiring analog voltage signal.



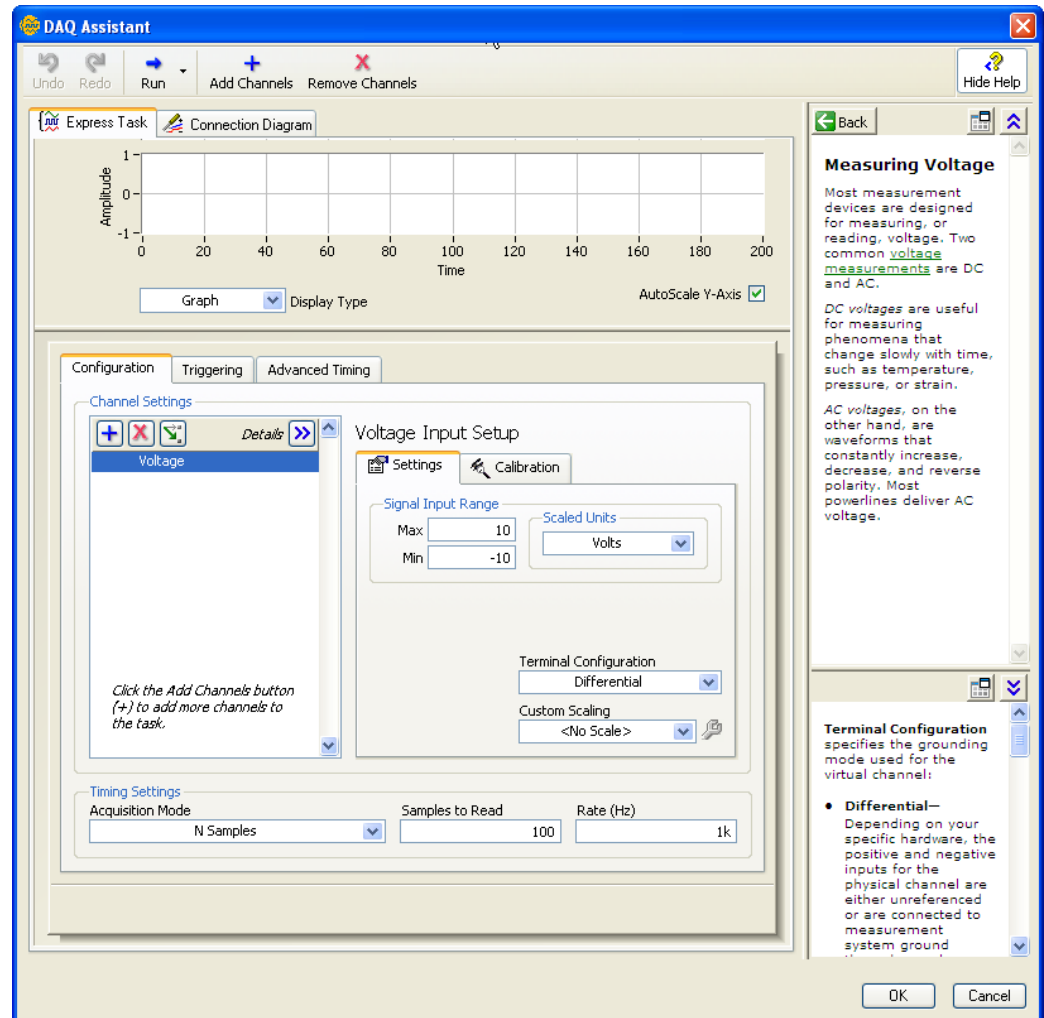
# Using DAQ Assistant – 2

- Select the physical channel



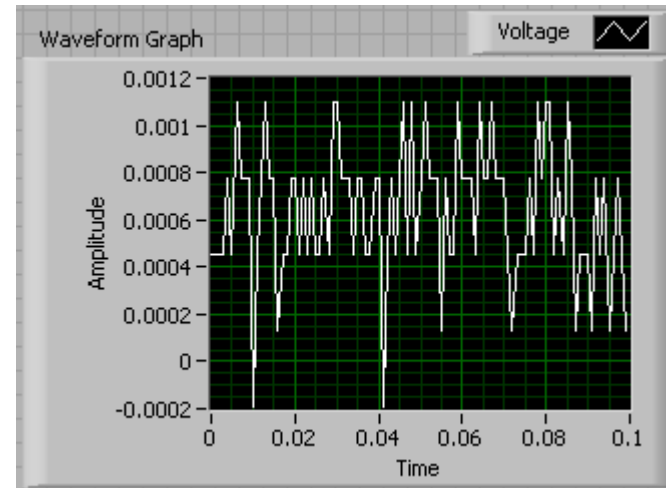
# Using DAQ Assistant – 3

- Configuring the channel settings and testing the DAQmx task

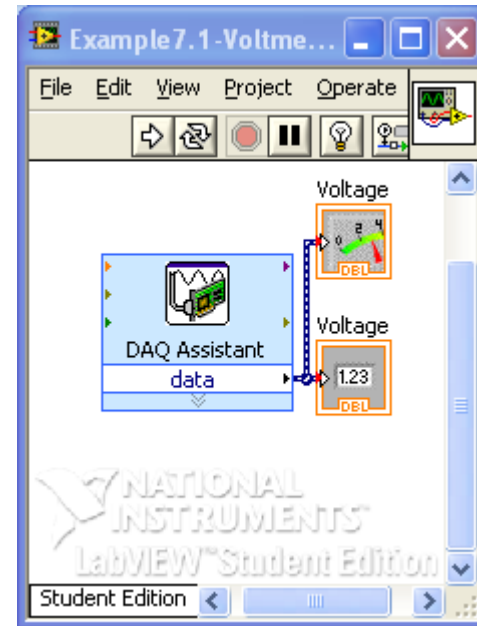
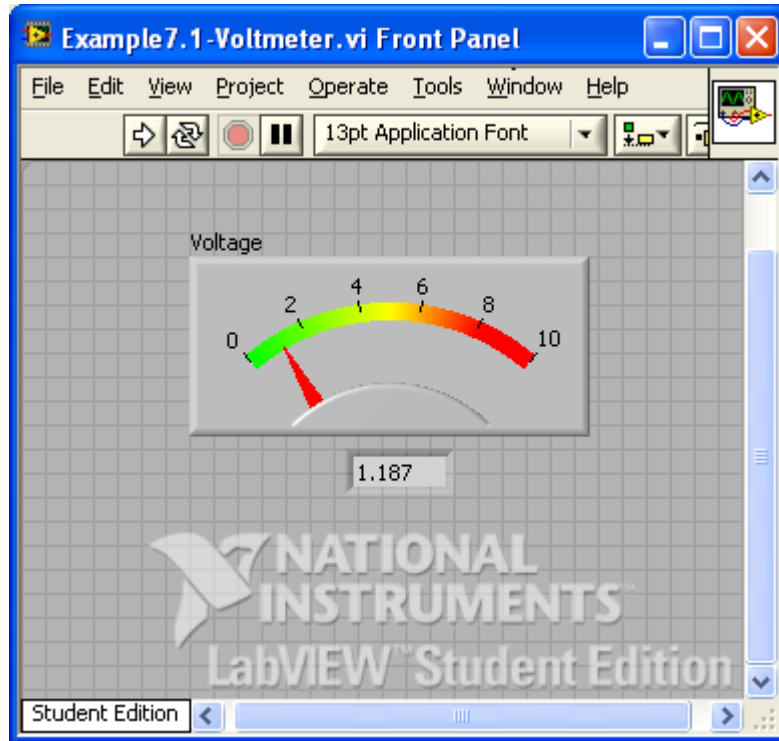


# Using DAQ Assistant – 4

- The output can be displayed in a waveform graph

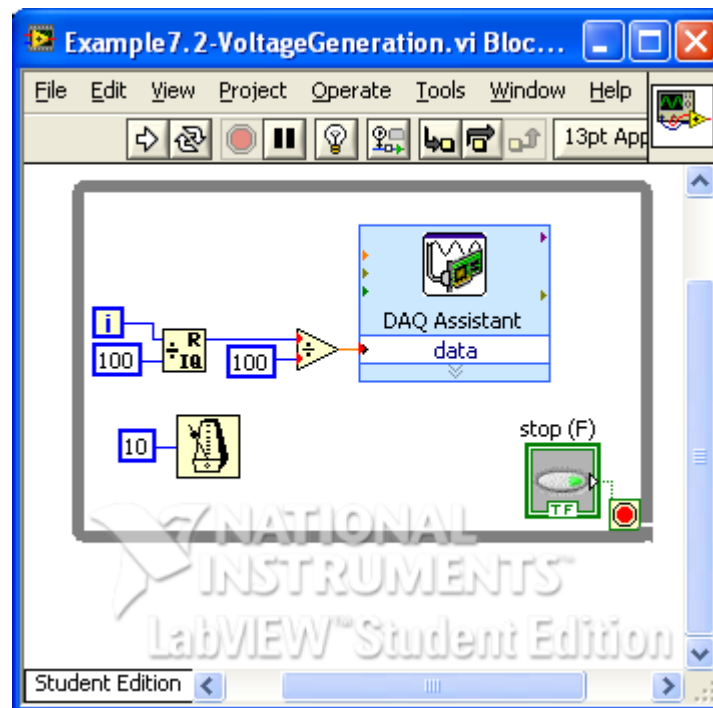


# Work Example 5.1 – Voltmeter



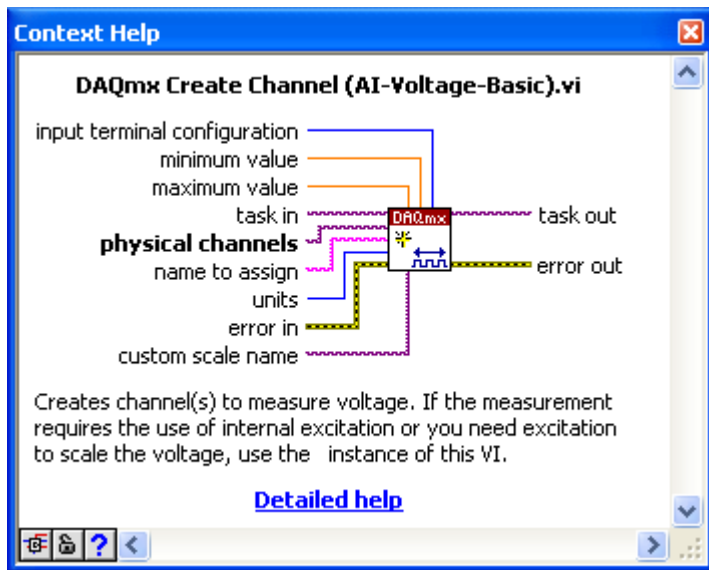
# Work Example 5.2 – Voltage Generation

- Generate a sawtooth waveform

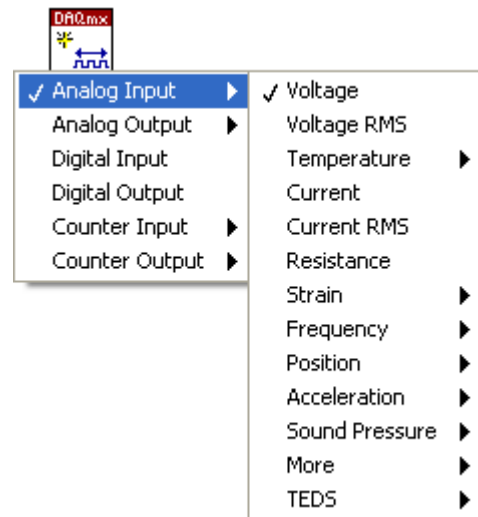


# NI-DAQmx VIs – Create Virtual Channel

- DAQmx Create Channel: Creates a virtual channel or set of virtual channels and adds them to a task. If you do not specify a task, NI-DAQmx creates a task for you and adds the virtual channels this VI creates to that task



## Select channel types

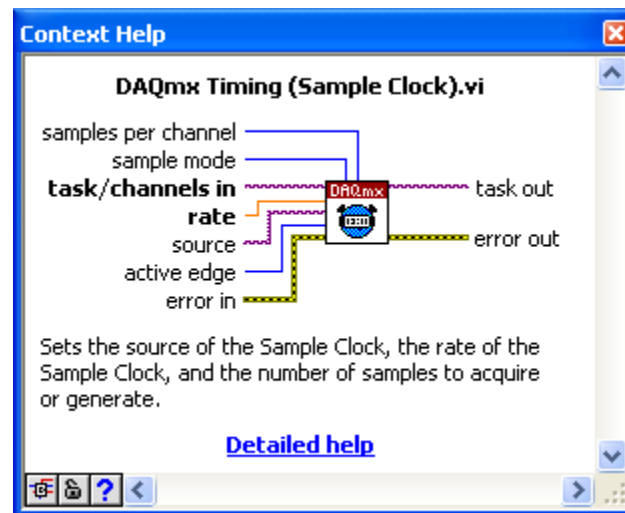


## Examples



# NI-DAQmx VIs – Timing

- Configures the number of samples to acquire or generate and creates a buffer when needed. Specify the sampling rate or use an external clock.





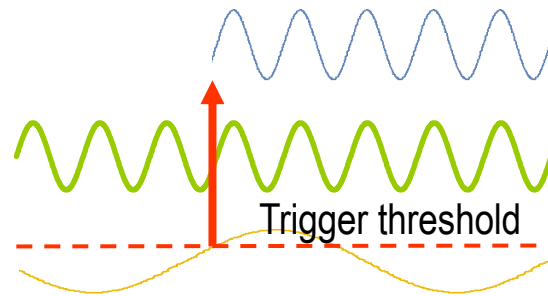
# NI-DAQmx VIs – Trigger

- Configures triggering for the task.
- Analog trigger

Acquired Signal

Input Signal

Trigger Signal

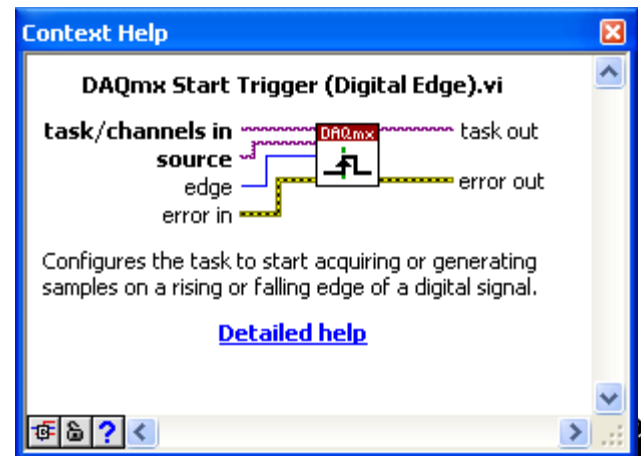
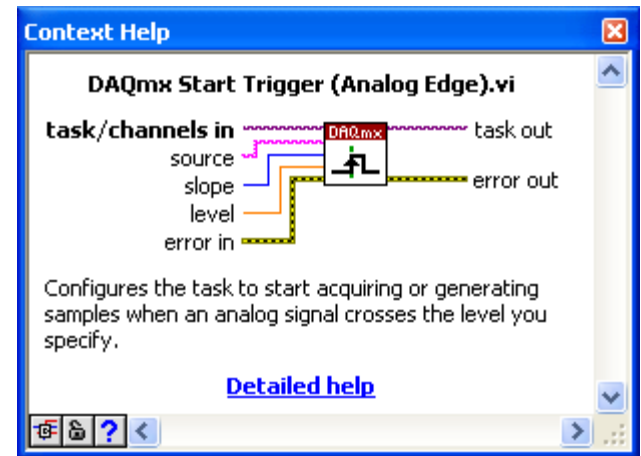
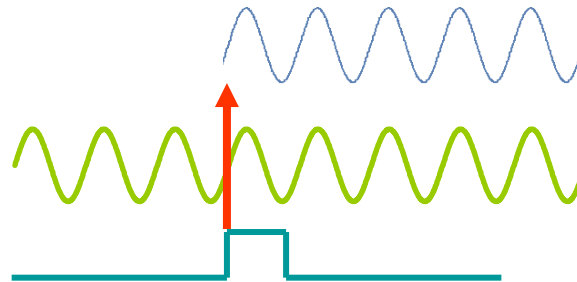


- Digital trigger

Acquired Signal

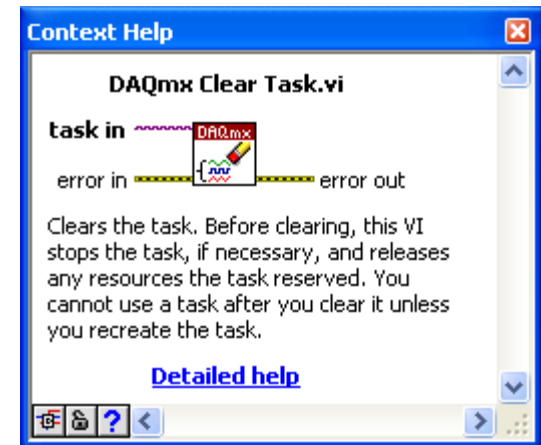
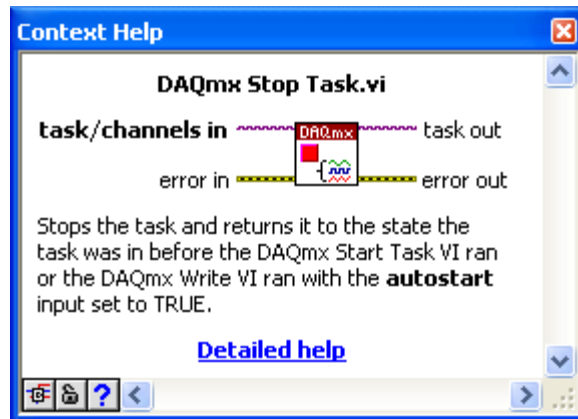
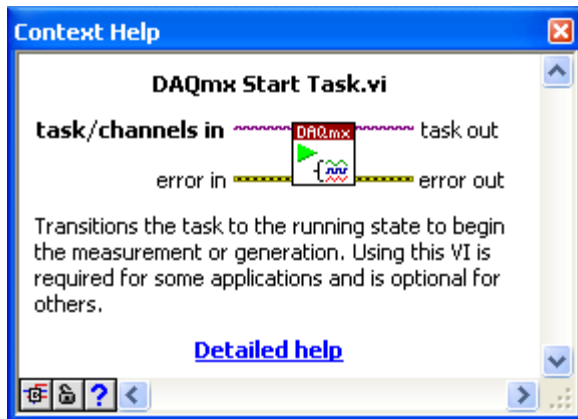
Input Signal

Trigger Signal



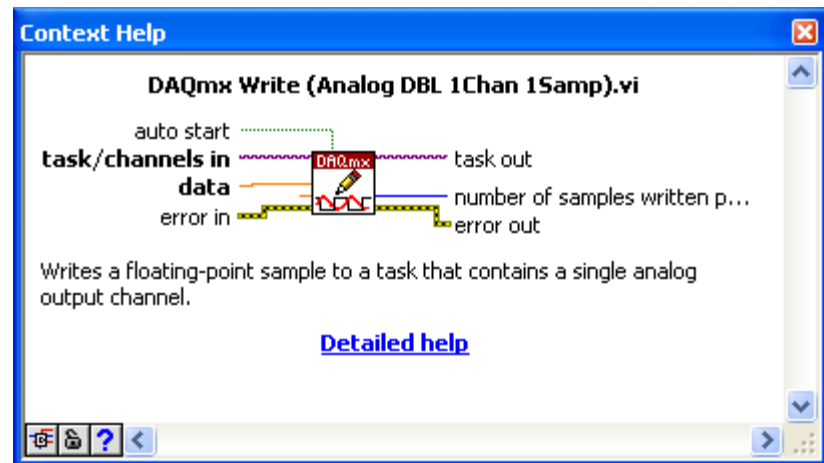
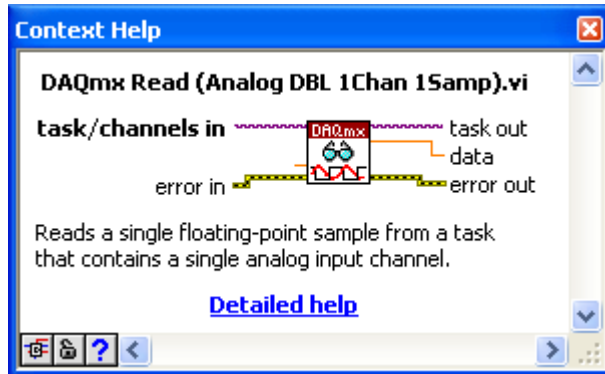
# NI-DAQmx VIs – Start, Stop, and Clear

- To start, stop or clear the task

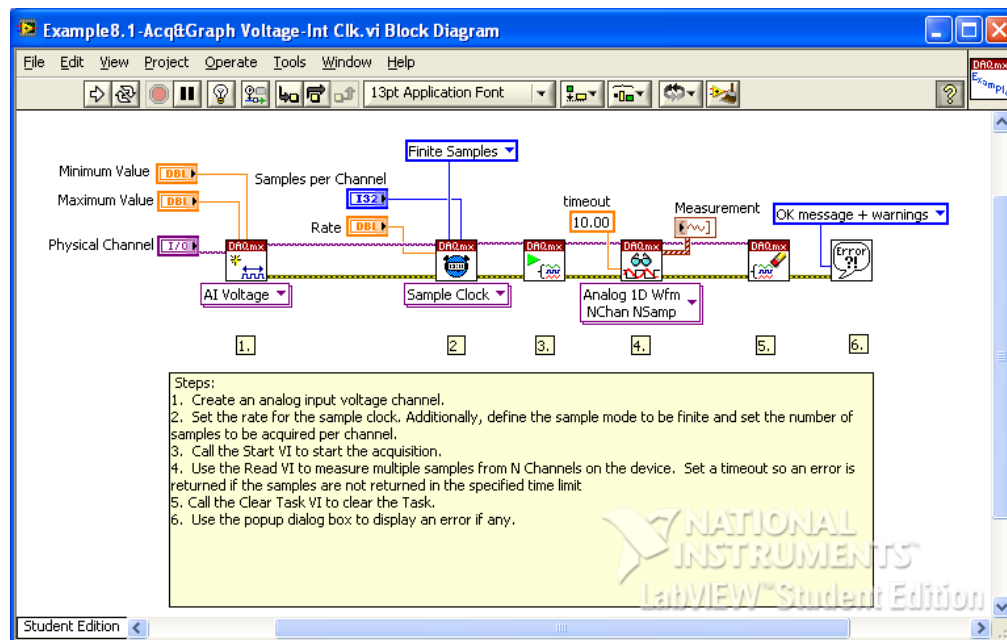
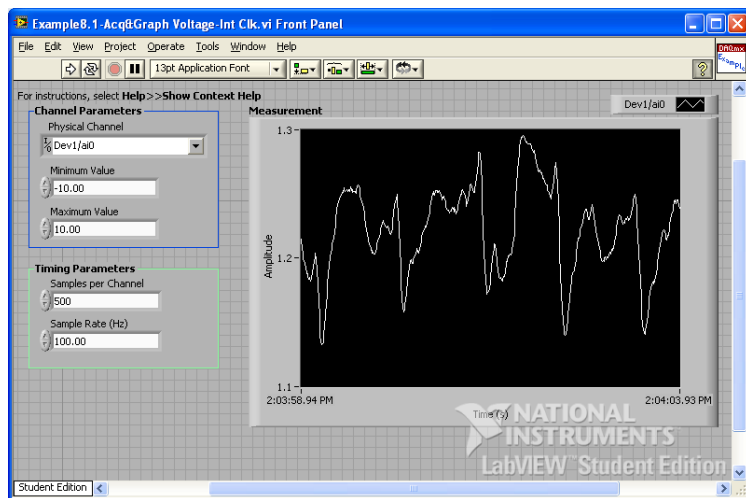


# NI-DAQmx VIs – Read, Write

- To read from or write to the task/channels.



# Example – Acq&Graph Voltage-Int Clk (From LabVIEW Examples)



### Example 8.1 - Acq&Graph Voltage-Int Clk.vi Block Diagram

The block diagram illustrates the acquisition and graphing of voltage data. It starts with an AI Voltage channel (Step 1). The Sample Clock rate is set to 10.00 (Step 2), and the sample mode is set to 'Finite Samples' with a specified number of samples per channel. The acquisition is initiated by the Start VI (Step 3). The Read VI (Step 4) measures multiple samples from N Channels on the device, with a timeout of 10.00. The measurement is displayed as an Analog 1D Wfm NChan NSamp. The Clear Task VI (Step 5) is used to clear the task. Finally, an error message is displayed (Step 6) if any errors occur.

**Steps:**

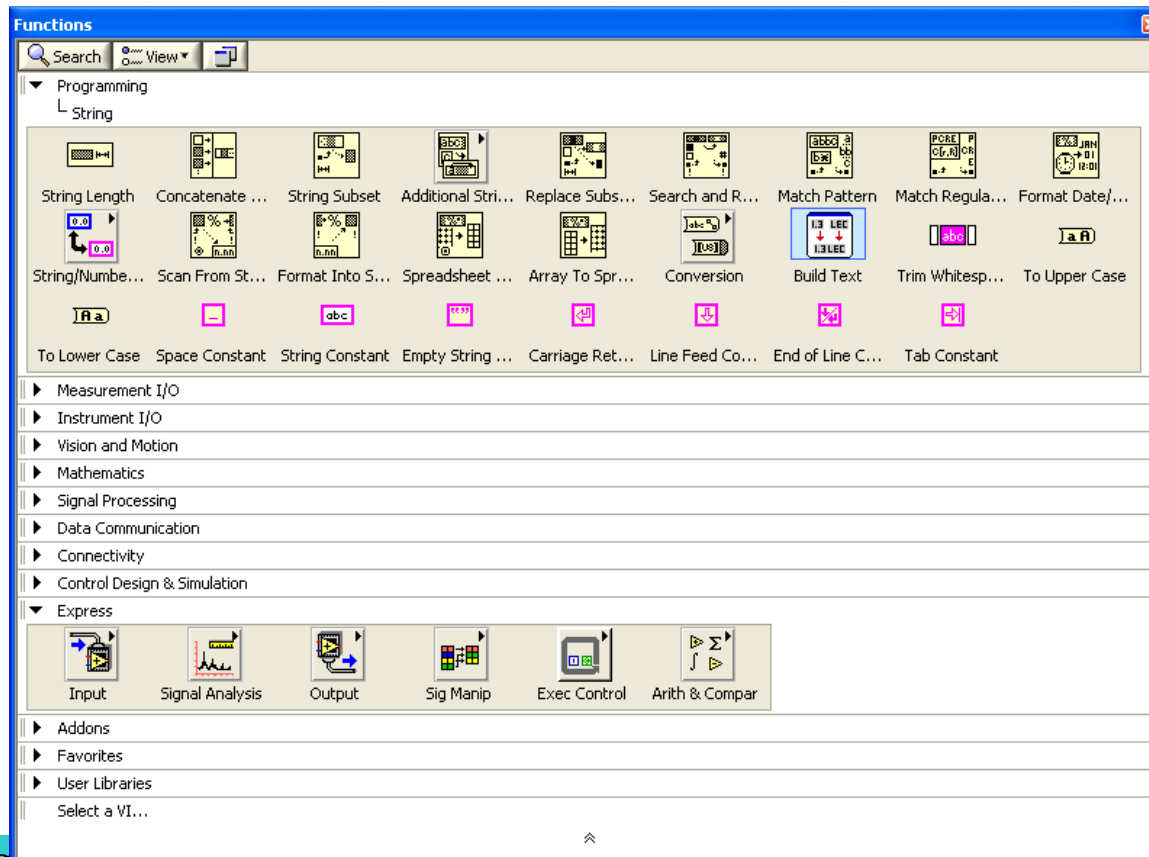
1. Create an analog input voltage channel.
2. Set the rate for the sample clock. Additionally, define the sample mode to be finite and set the number of samples to be acquired per channel.
3. Call the Start VI to start the acquisition.
4. Use the Read VI to measure multiple samples from N Channels on the device. Set a timeout so an error is returned if the samples are not returned in the specified time limit.
5. Call the Clear Task VI to clear the Task.
6. Use the popup dialog box to display an error if any.

NATIONAL INSTRUMENTS  
LabVIEW™ Student Edition

Student Edition

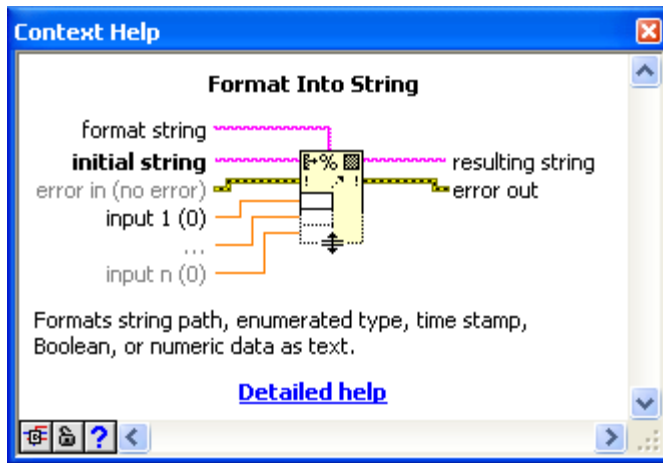
# Strings

- String: a sequence of characters that can be displayable or nondisplayable.

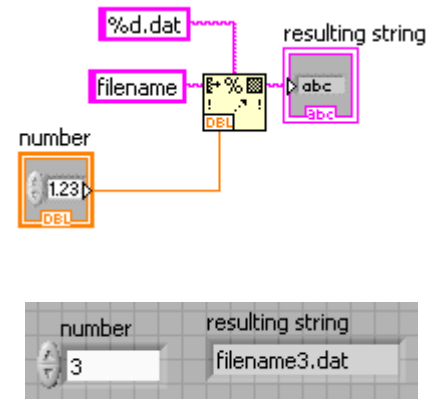


# Format into String

- The format string is similar to those in C except some additional features.

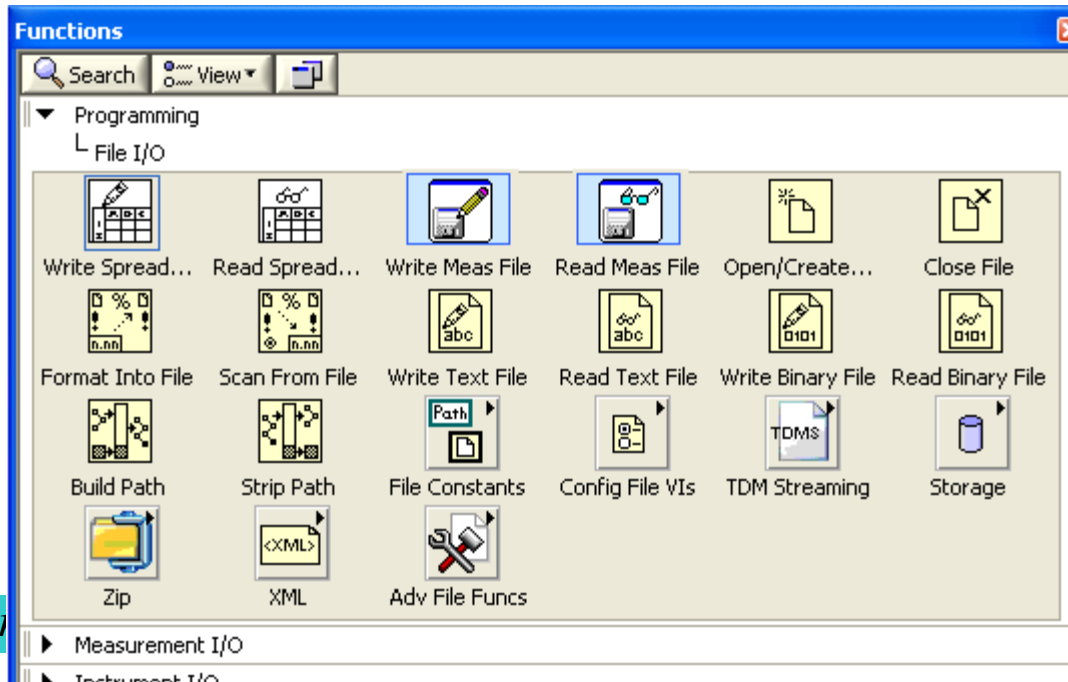


## Example



# File I/O

- File I/O operations pass data to and from files
  - Opening and closing data files.
  - Reading data from and writing data to files.
  - Reading from and writing to spreadsheet-formatted files.
  - Moving and renaming files and directories.
  - Changing file characteristics.



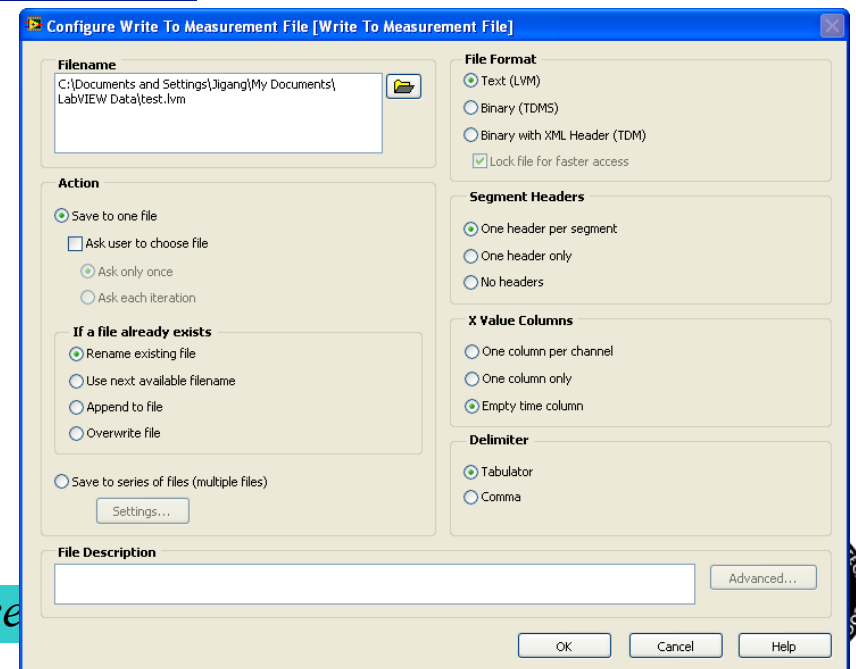
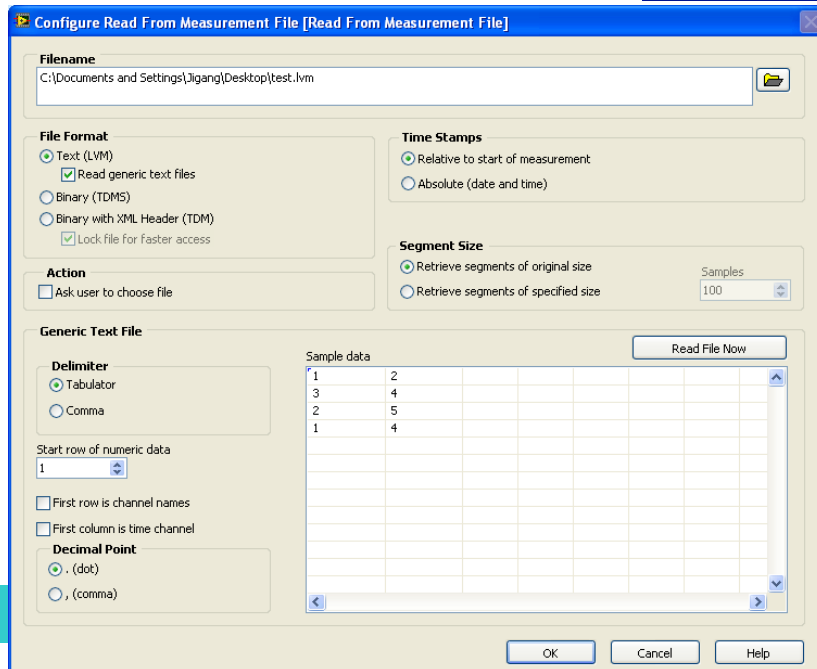


# Read from and Write to Measurement File

- Use these express VIs for easy writing and reading

Example lvm file

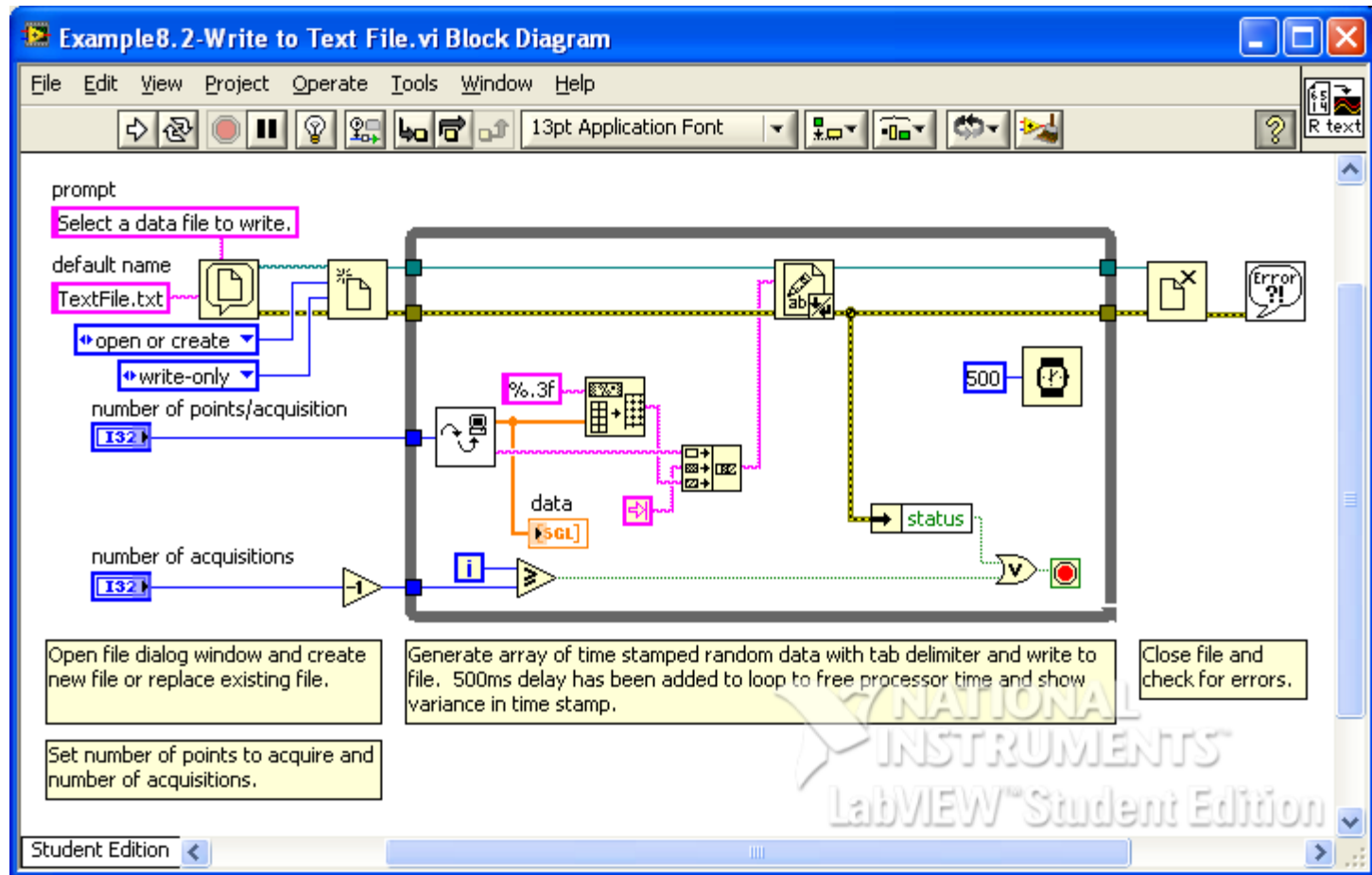
```
File Edit Format View Help
1      2
3      4
2      5
1      4
```



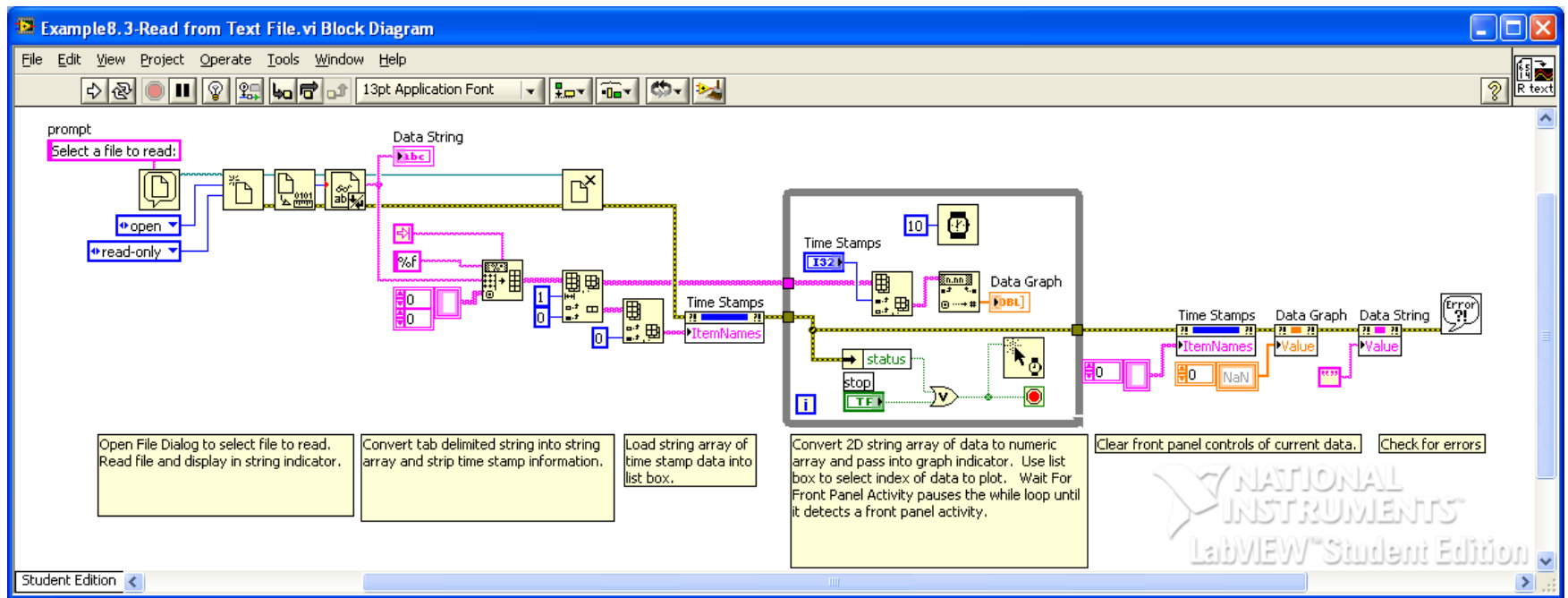
odevice

OTONEL

# Example – Write to Text File (From LabVIEW Examples)



# Example – Read from Text File (From LabVIEW Examples)



# BE/EE189 Design and Construction of Biodevices

## Lecture 6



# LabVIEW Programming – MathScript, Matlab, Curve Fitting, and FFT

- MathScript RT module
- Matlab integration
- Curve fitting
- Signal processing – transforms



# LabVIEW Programming – Analysis and Signal Processing

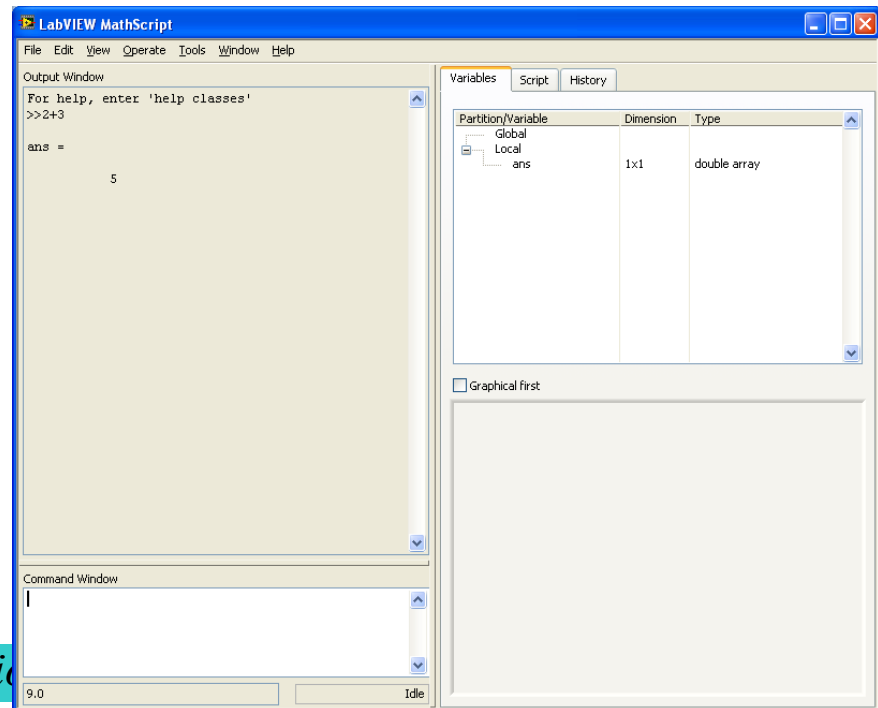
- Differential Equations
- Integration and Differentiation
- Signal generation and processing



# MathScript RT Module

- Provides access to a text-based math-oriented language with a command-prompt from within the LabVIEW development environment.
- Similar to Matlab in syntax.

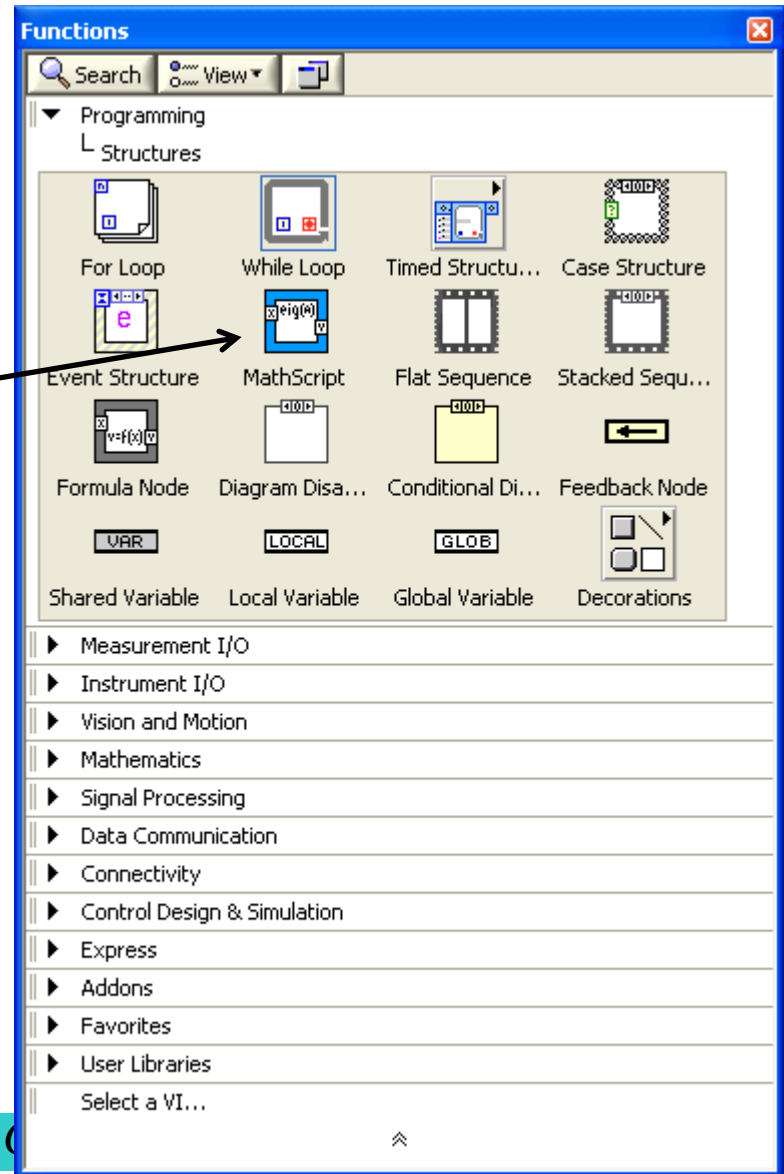
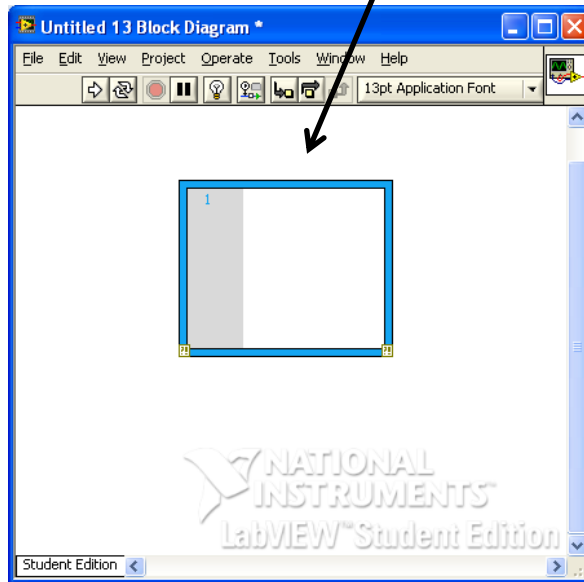
MathScript interactive window



# MathScript Nodes

- MathScript can be integrated into the VI using the MathScript Nodes.

MathScript Node

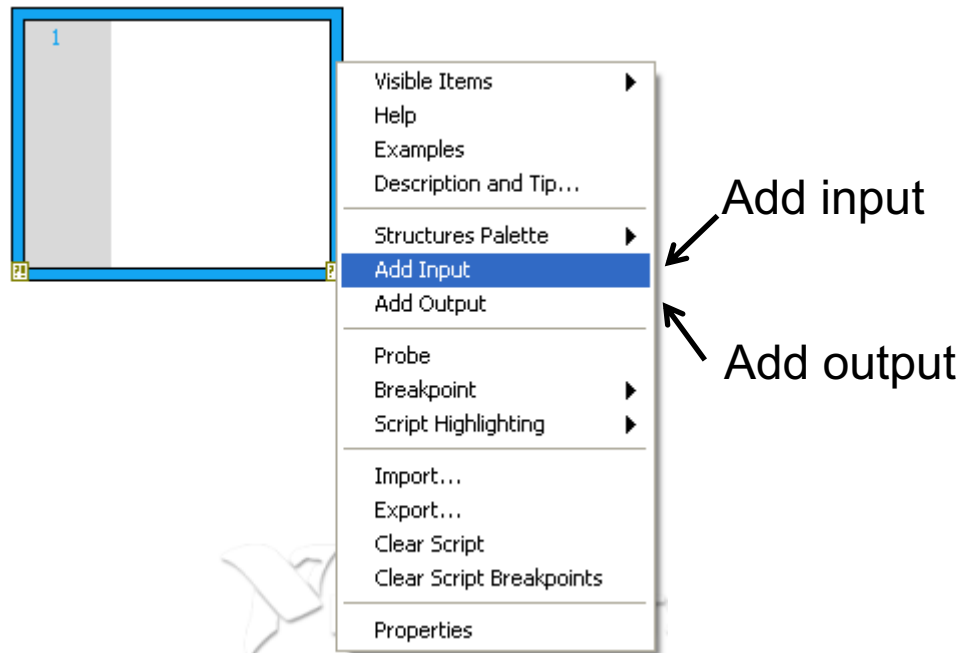




# MathScript Node – Input and Output

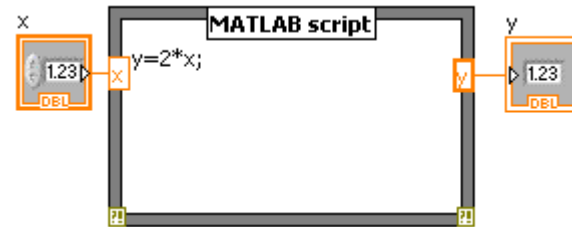
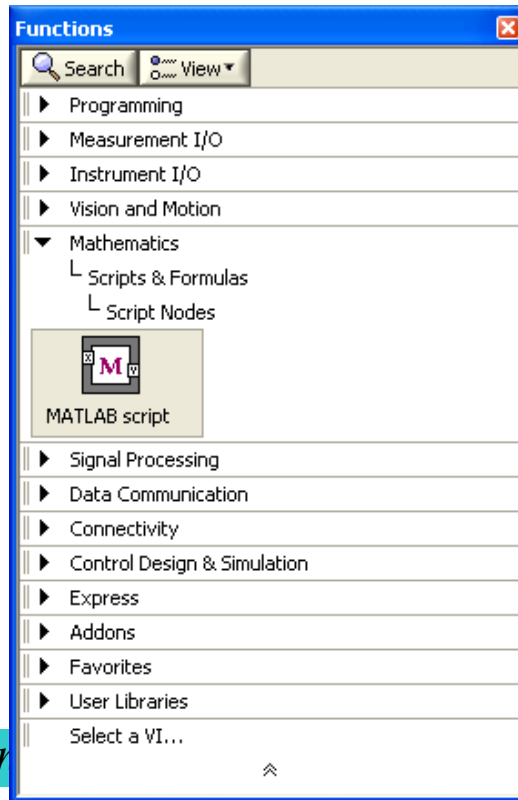
- To interact with the rest of VI, the MathScript Node usually need to specify input and output.

Right click the MathScript Node

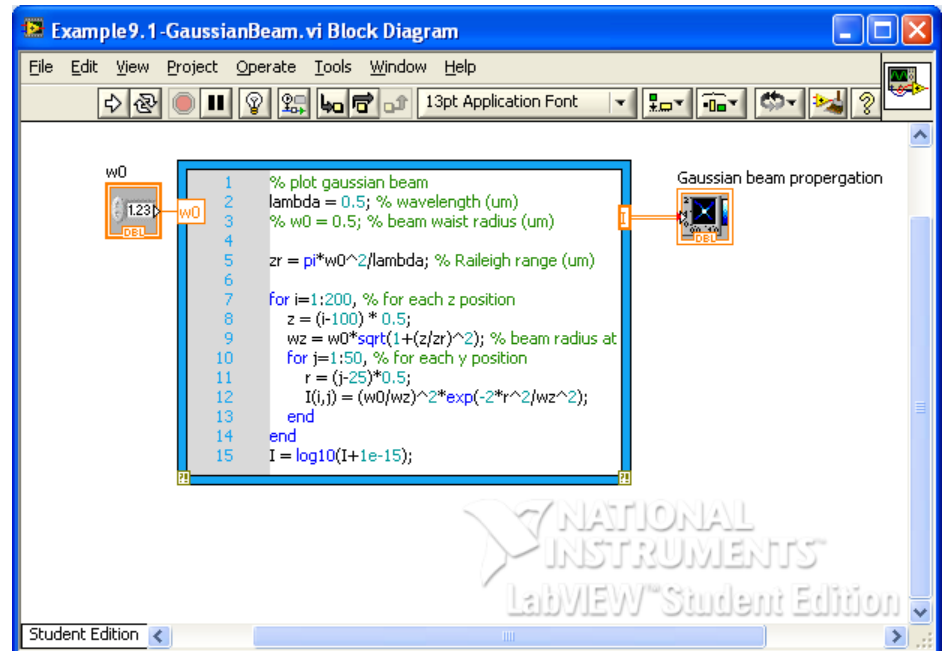
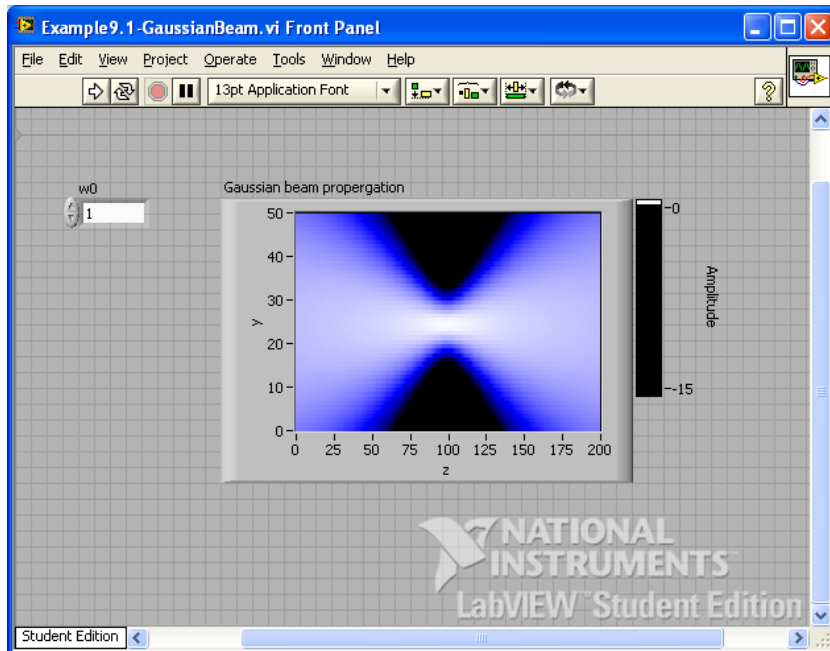


# Matlab Integration

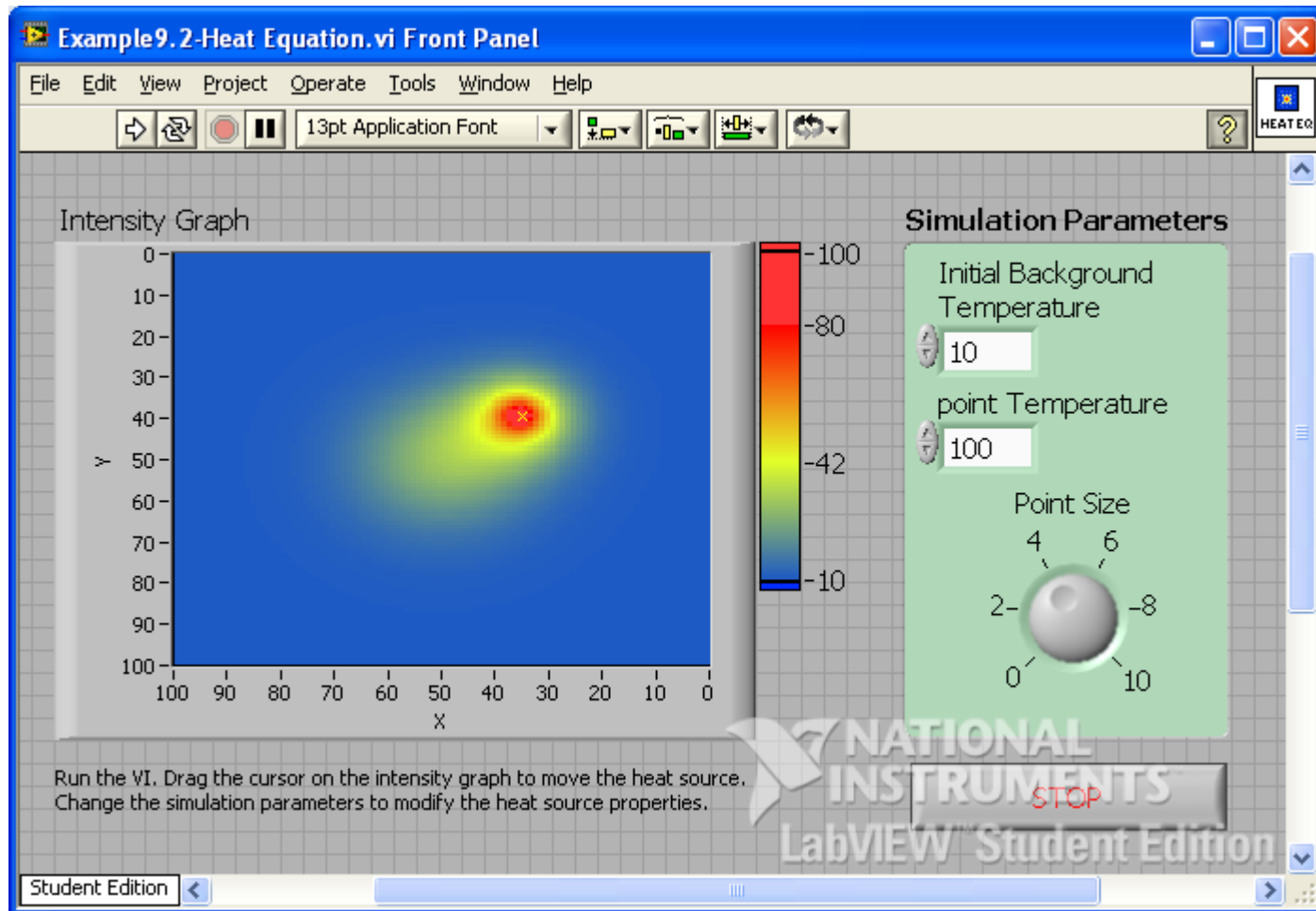
- Matlab script can be used directly in LabVIEW, similar to the MathScript Node.
- Matlab must be installed.



# Example – Gaussian Beam Propagation



# Example – Heat Transfer (From LabVIEW Example)



# Curve Fitting

- Least squares method

Error      Function      Observed data

↓           ↓           ↓

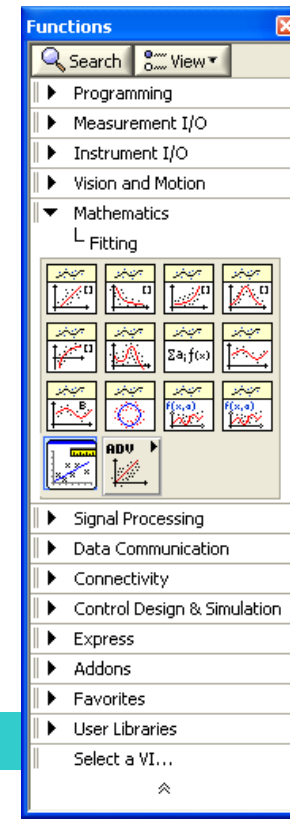
$$e(a) = [f(x, a) - y(x)]^2$$

↑

coefficients

Jacobian equation:  $\frac{\partial e(a)}{\partial a} = 0$

- Curve-fitting in LabVIEW:
  - Linear fit
  - Exponential fit
  - General polynomial fit
  - General linear fit
  - Nonlinear Levenberg-Marquardt fit
  - B-spline fit



# Fitting VIs

**Context Help**

**Linear Fit.vi**

Y  
X  
Weight  
tolerance  
method  
parameter bounds

Best Linear Fit  
slope  
intercept  
error  
residue

Returns the linear fit of a data set ( $X$ ,  $Y$ ) using the Least Square, Least Absolute Residual, or Bisquare method.

[Detailed help](#)

The diagram shows a central icon for Linear Fit.vi with several input and output lines. Inputs on the left include Y, X, Weight, tolerance, method, and parameter bounds. Outputs on the right include Best Linear Fit, slope, intercept, error, and residue.

**Context Help**

**Exponential Fit.vi**

Y  
X  
Weight  
tolerance  
method  
parameter bounds

Best Exponential Fit  
amplitude  
damping  
offset  
error  
residue

Returns the exponential fit of a data set ( $X$ ,  $Y$ ) using the Least Square, Least Absolute Residual, or Bisquare method.

[Detailed help](#)

The diagram shows a central icon for Exponential Fit.vi with several input and output lines. Inputs on the left include Y, X, Weight, tolerance, method, and parameter bounds. Outputs on the right include Best Exponential Fit, amplitude, damping, offset, error, and residue.

**Context Help**

**Gaussian Peak Fit.vi**

initial guess  
Y  
X  
Weight  
tolerance  
method  
parameter bounds

offset  
Best Gaussian Fit  
amplitude  
center  
standard deviation  
error  
residue

Returns the Gaussian fit of a data set ( $X$ ,  $Y$ ) using the Least Square, Least Absolute Residual, or Bisquare method.

[Detailed help](#)

The diagram shows a central icon for Gaussian Peak Fit.vi with several input and output lines. Inputs on the left include initial guess, Y, X, Weight, tolerance, method, and parameter bounds. Outputs on the right include offset, Best Gaussian Fit, amplitude, center, standard deviation, error, and residue.

**Context Help**

**B-Spline Fit.vi**

# of control points  
Y  
X  
Weight  
degree  
parameter selection

Best BSpline Fit Y  
Best BSpline Fit X  
error  
residue

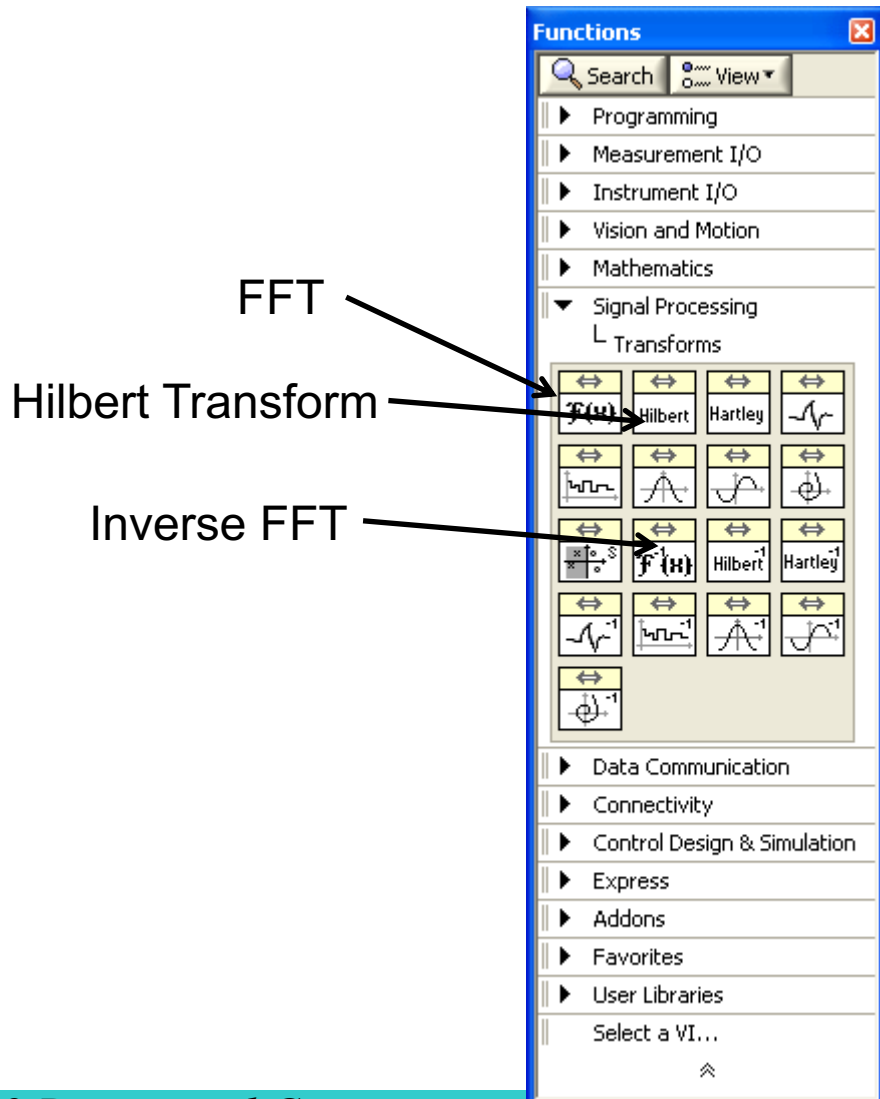
Uses B-spline fitting to smooth a data set ( $X$ ,  $Y$ ).

[Detailed help](#)

The diagram shows a central icon for B-Spline Fit.vi with several input and output lines. Inputs on the left include # of control points, Y, X, Weight, degree, and parameter selection. Outputs on the right include Best BSpline Fit Y, Best BSpline Fit X, error, and residue.

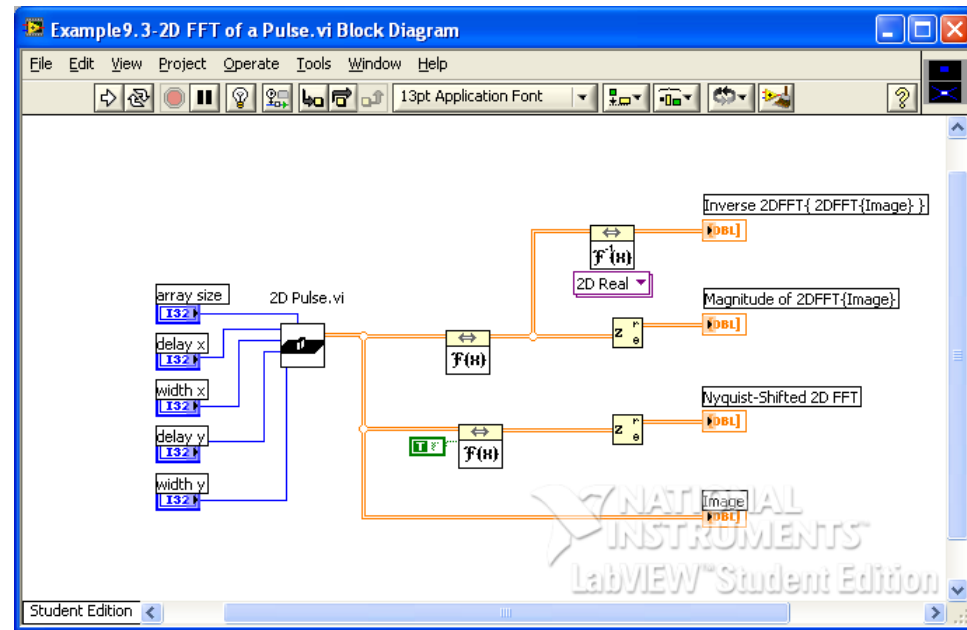
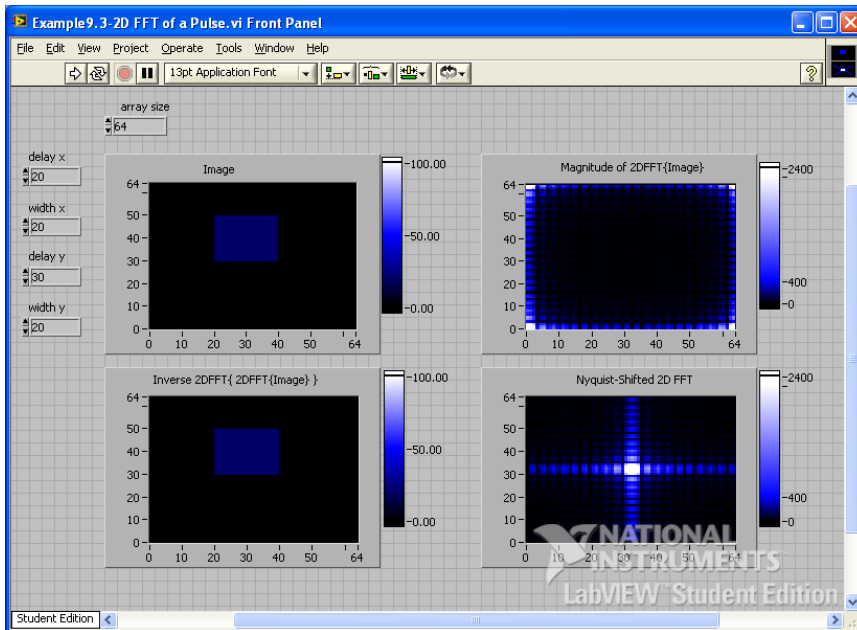


# Signal Processing – Transforms



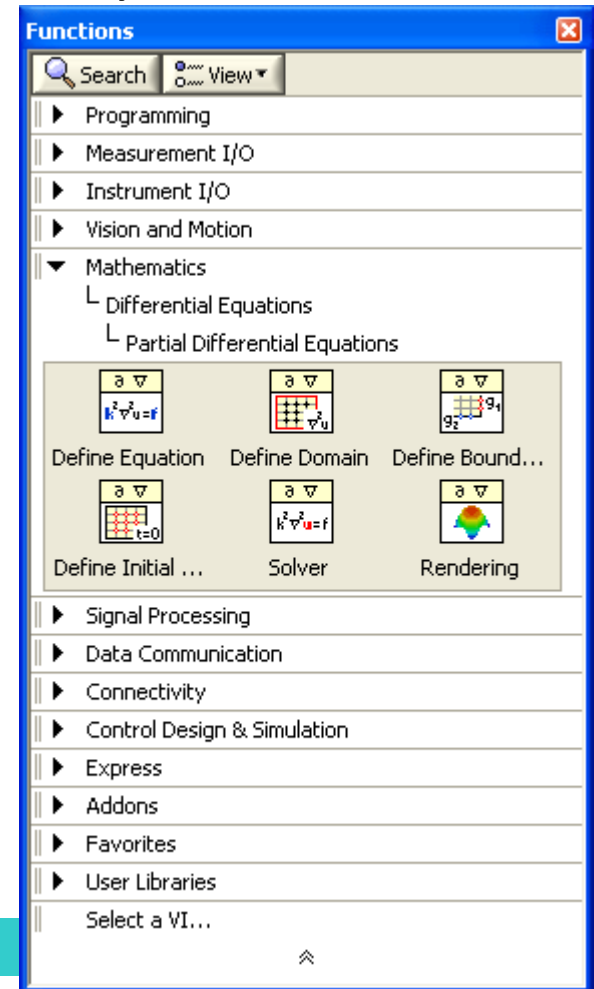
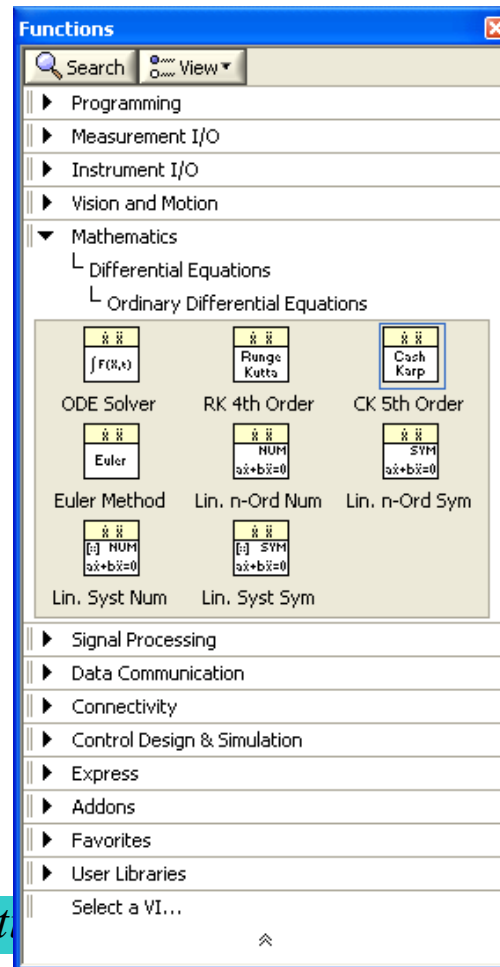
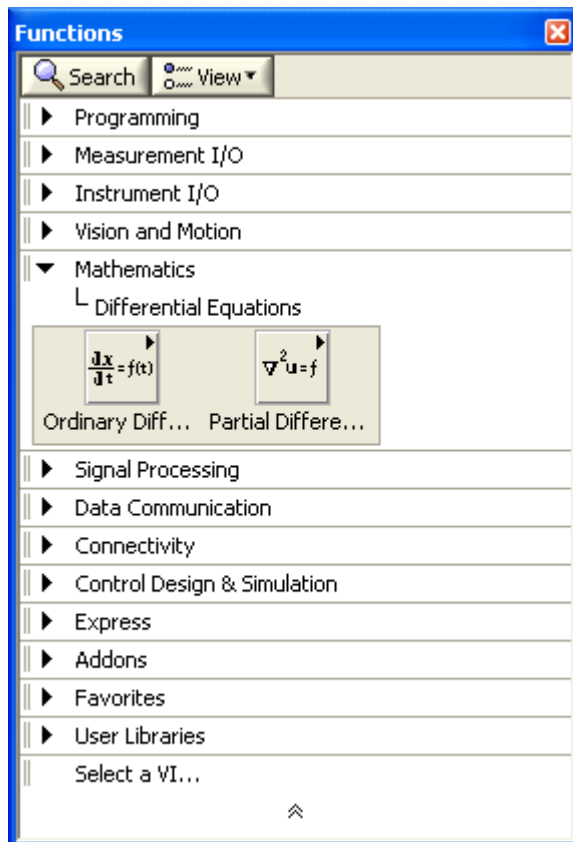


# Example – 2D FFT of a Pulse (From LabVIEW Example)



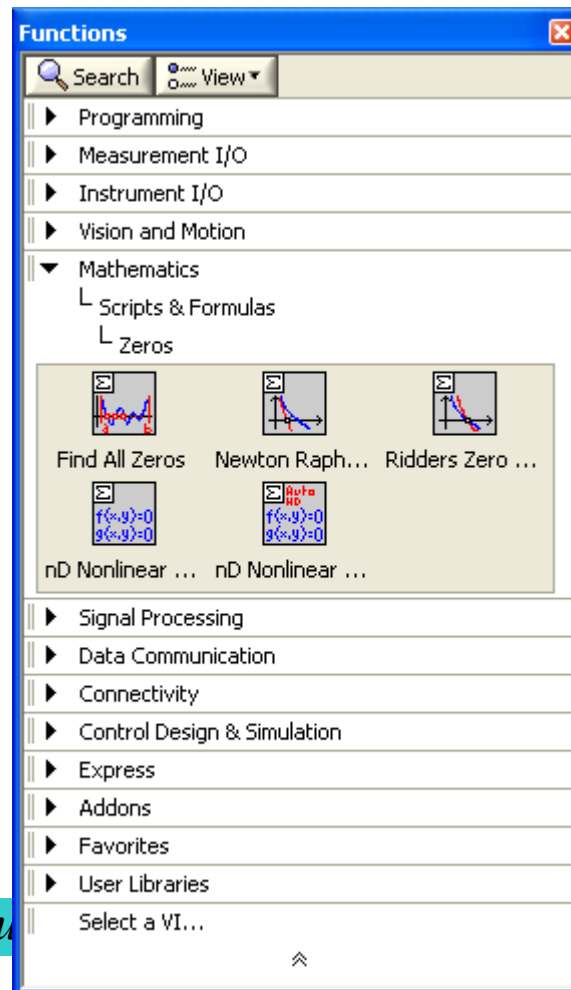
# Differential Equations

- LabVIEW can solve ODEs and PDEs numerically.



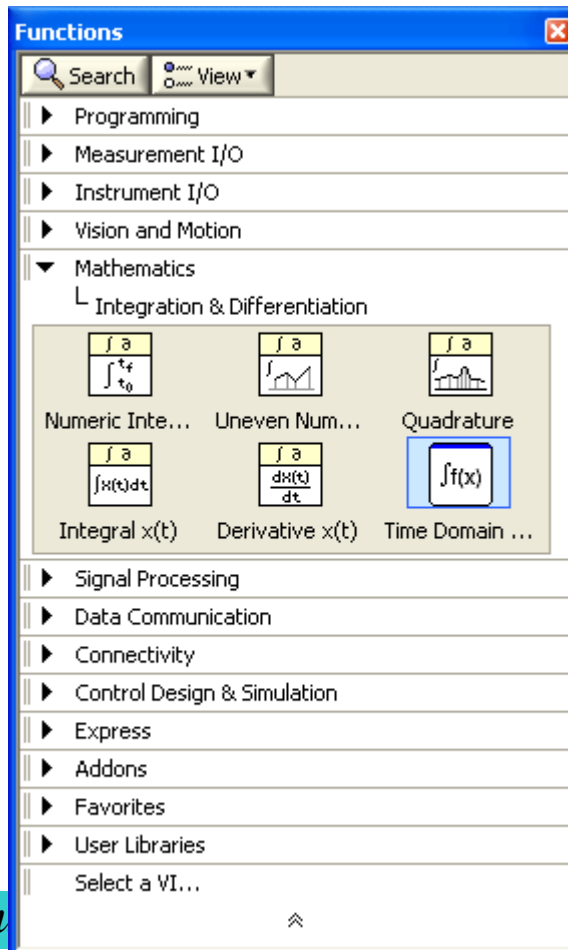
# Finding Zeros of Functions

- LabVIEW provides VIs that can be used to compute zeros of functions.



# Integration and Differentiation

- LabVIEW provides VIs for integration and differentiation.



# Signal Generation

- For testing algorithms and other purposes when real-world signals are not available.
- Signal can be generated by
  - Mathematical equations
  - Arrays of data points
  - Signal generation Vis for common signals

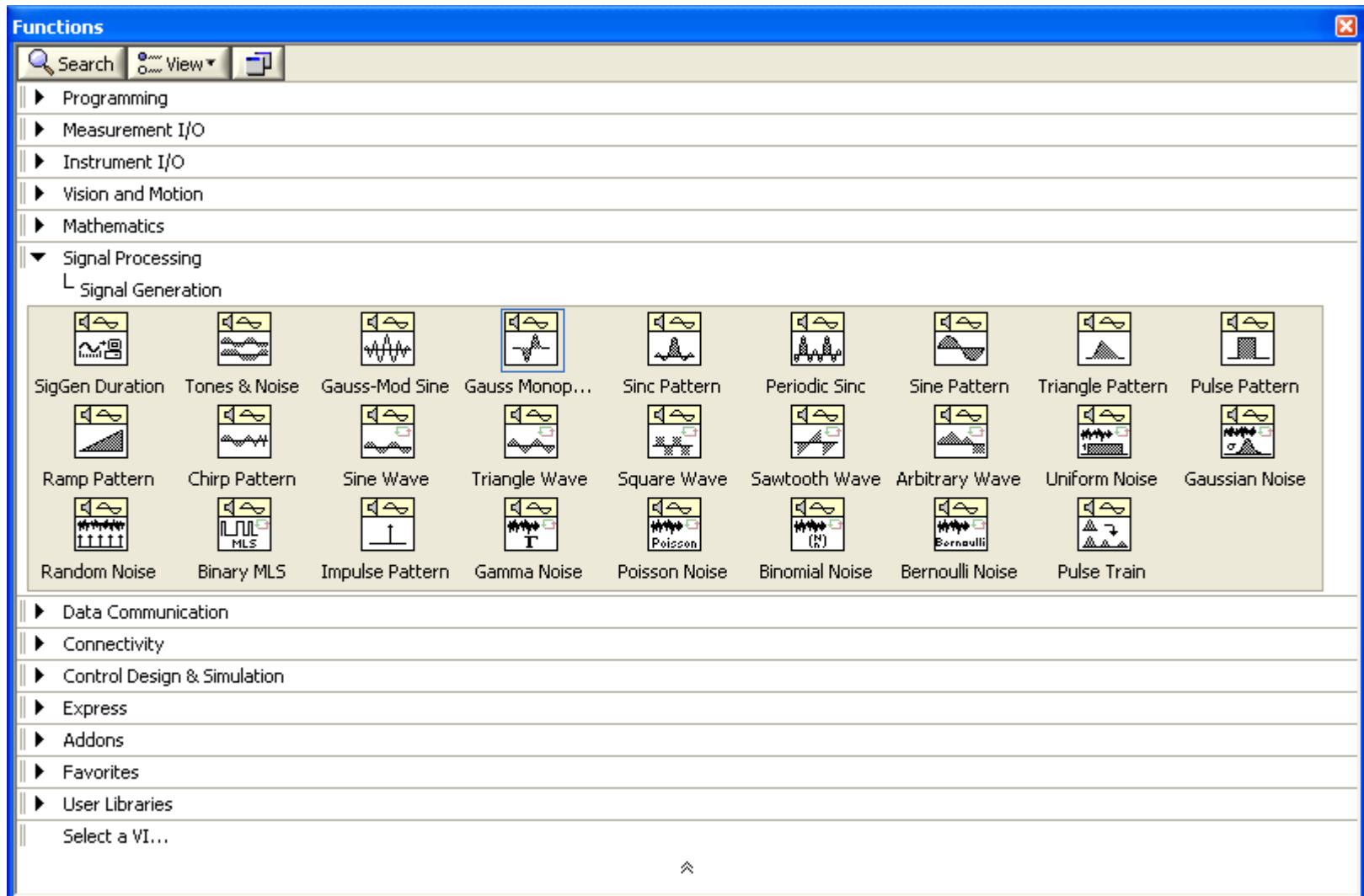


# Normalized Frequency

- Also called **digital frequency**.

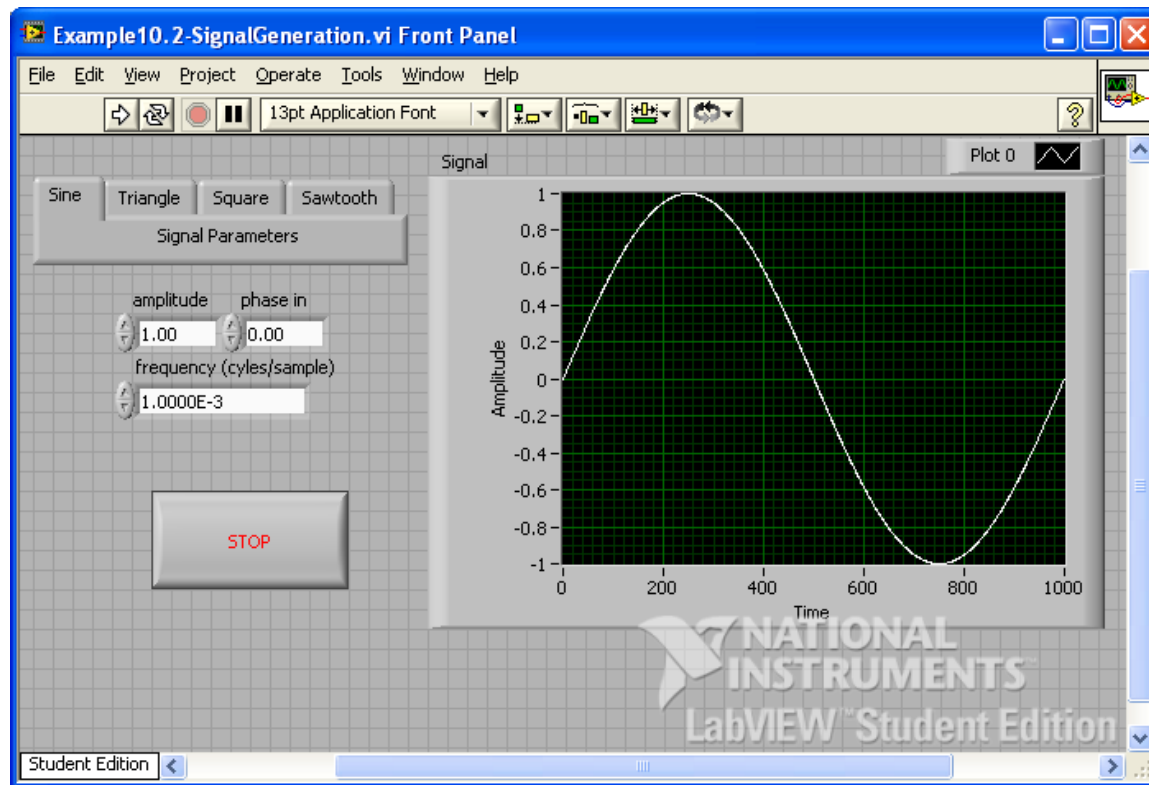
$$f = \text{Normalized frequency} = \frac{\text{Analog frequency}}{\text{Sampling frequency}}$$

# Signal Generation VIs



# Example – Signal Generation

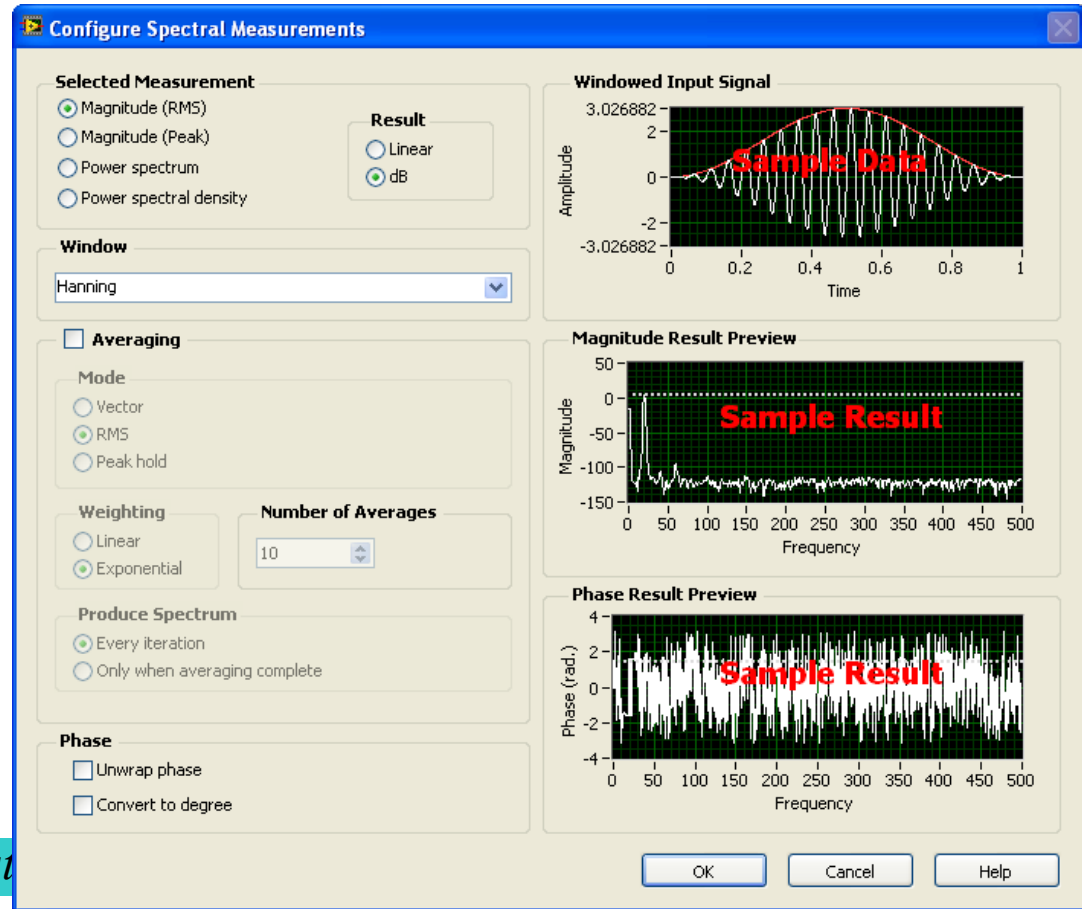
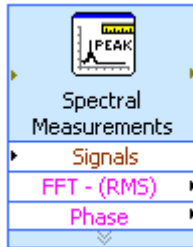
- Generate sine, triangle, square, and sawtooth signal.





# Signal Processing – Spectral Measurements Express VI

- The spectral measurements Express VI performs spectral measurements, such as spectral power density.



**Configure Spectral Measurements**

**Selected Measurement**

- Magnitude (RMS)
- Magnitude (Peak)
- Power spectrum
- Power spectral density

**Result**

- Linear
- dB

**Window**

Hanning

**Averaging**

**Mode**

- Vector
- RMS
- Peak hold

**Weighting**

- Linear
- Exponential

**Number of Averages**

10

**Produce Spectrum**

- Every iteration
- Only when averaging complete

**Phase**

- Unwrap phase
- Convert to degree

**Windowed Input Signal**

Amplitude vs Time

**Magnitude Result Preview**

Magnitude vs Frequency

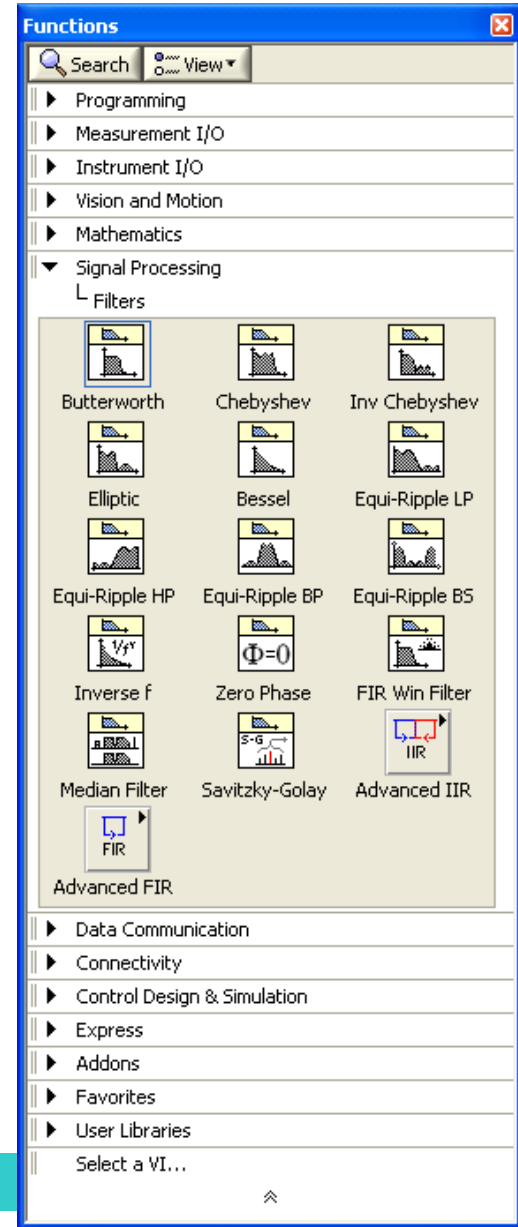
**Phase Result Preview**

Phase (rad.) vs Frequency

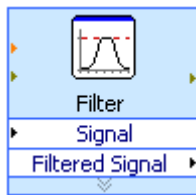
OK Cancel Help

# Signal Processing – Filtering

- LabVIEW can be used to implement digital filters
  - Finite impulse response (FIR) filters
  - Infinite impulse response (IIR) filters



# Signal Processing – Filter Express VI



**Configure Filter [Filter]**

**Filtering Type**

Lowpass

**Filter Specifications**

Cutoff Frequency (Hz)  
100

High cutoff frequency (Hz)  
400

Finite impulse response (FIR) filter

Taps  
29

Infinite impulse response (IIR) filter

Topology  
Butterworth

Order  
3

**Input Signal**

Amplitude vs. Time plot showing Sample Data.

**Result Preview**

Amplitude vs. Time plot showing Sample Result.

**View Mode**

Signals  Show as spectrum

Transfer function

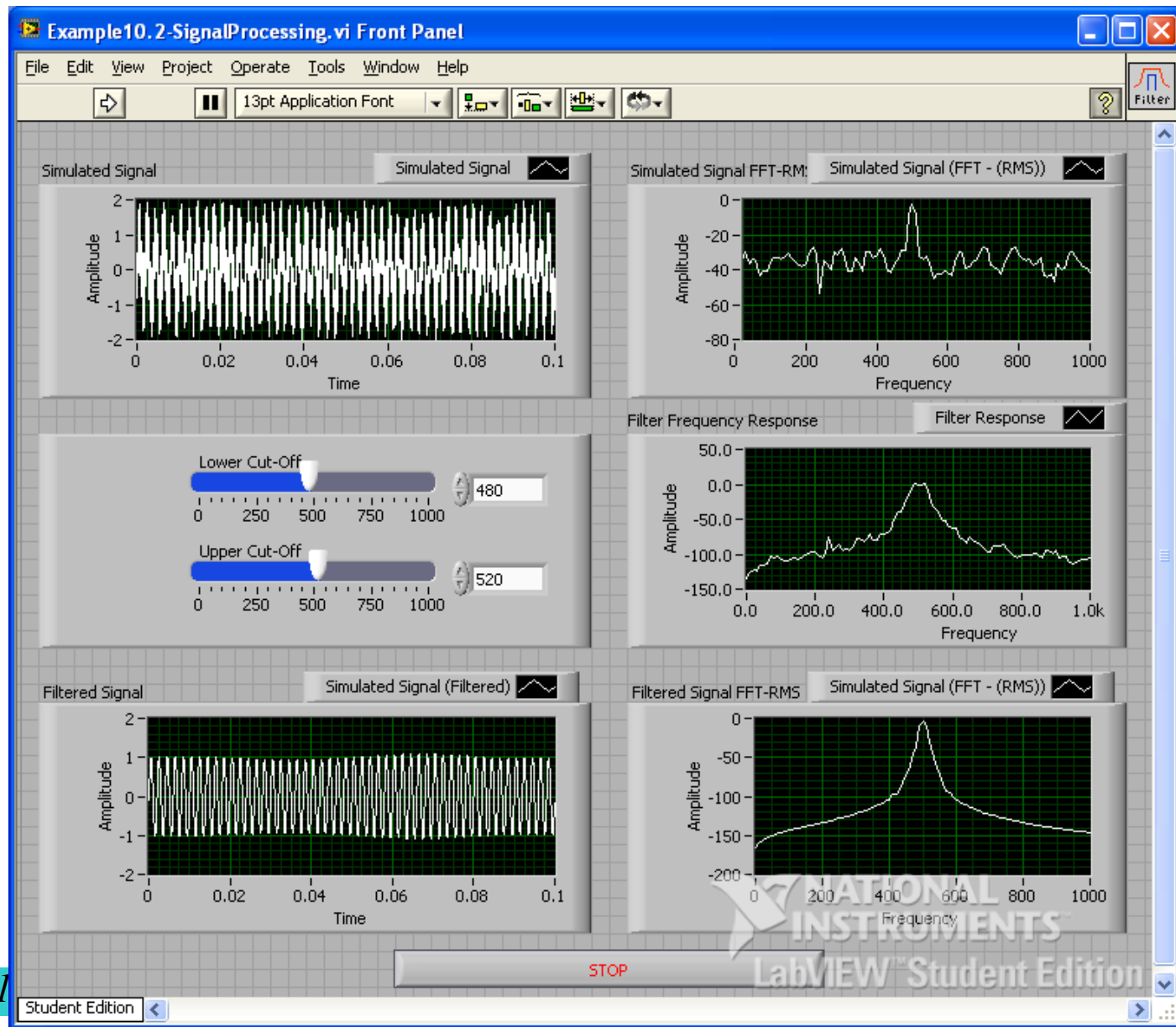
**Scale Mode**

Magnitude in dB

Frequency in log

OK Cancel Help

# Example – Signal Processing



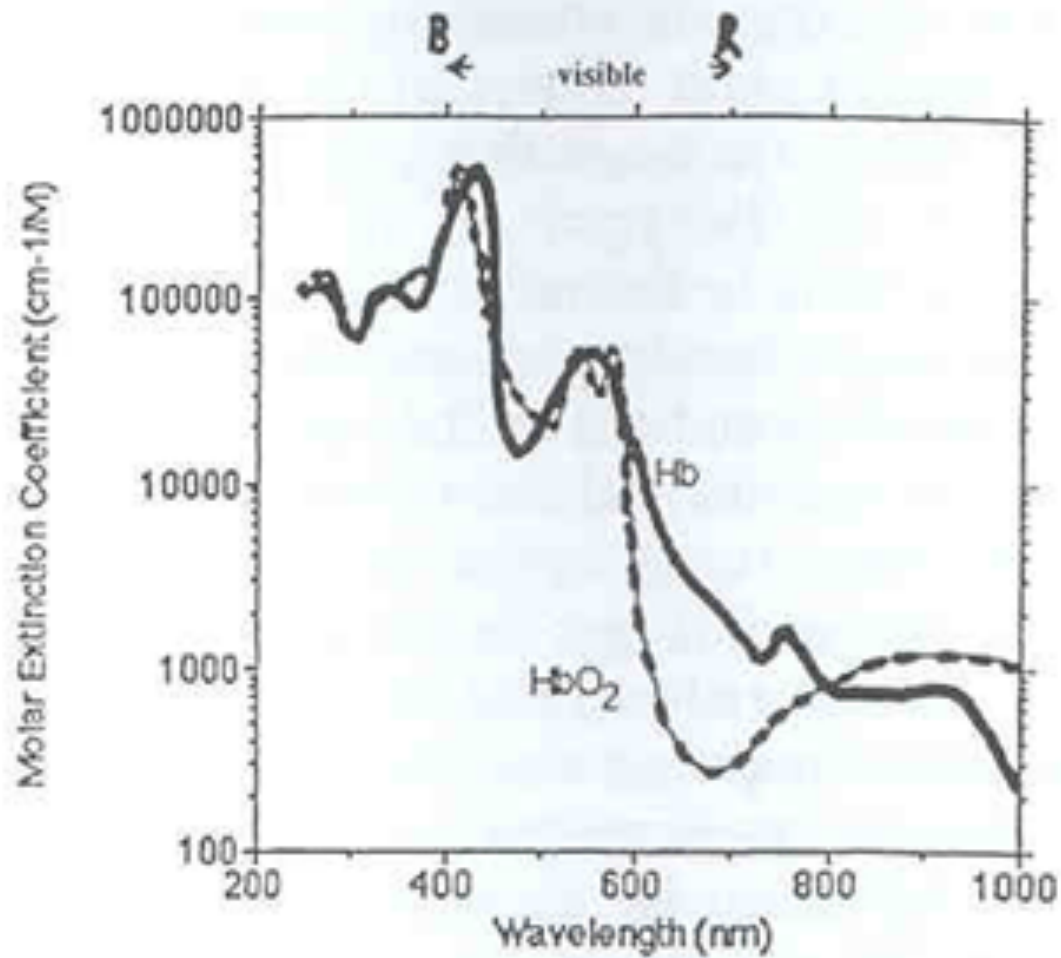
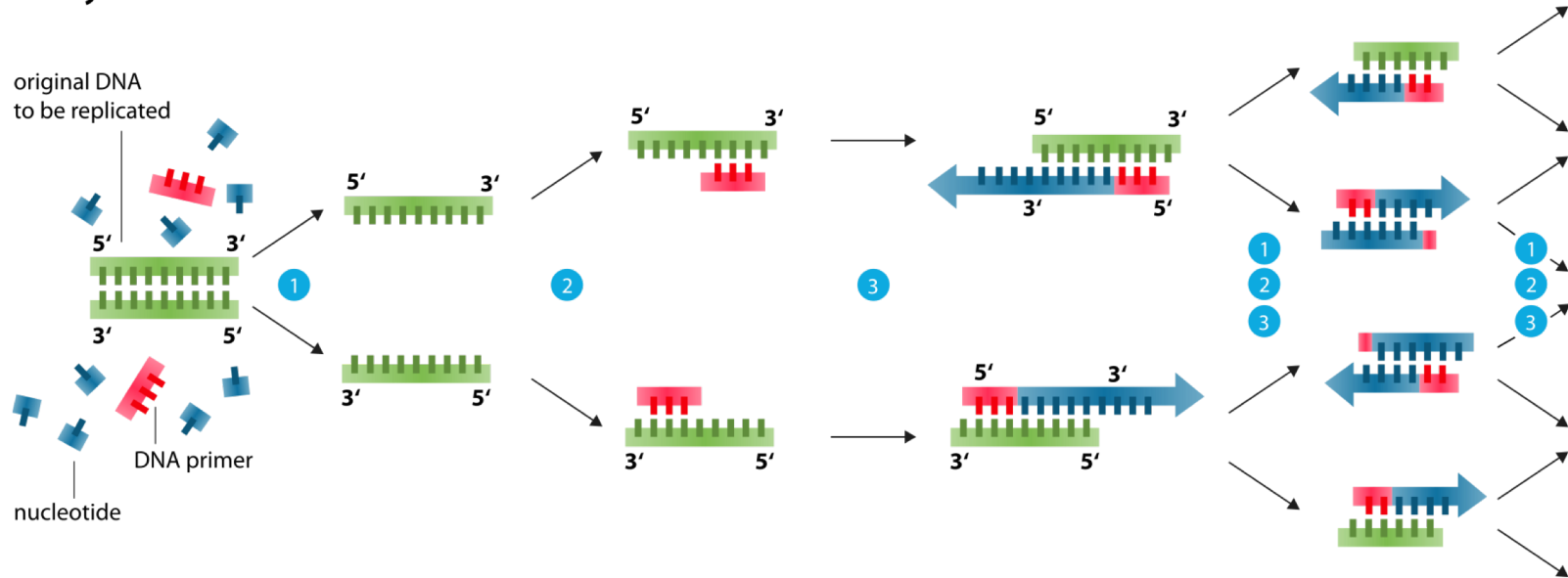
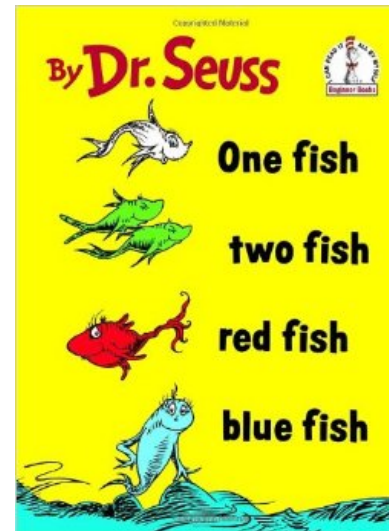


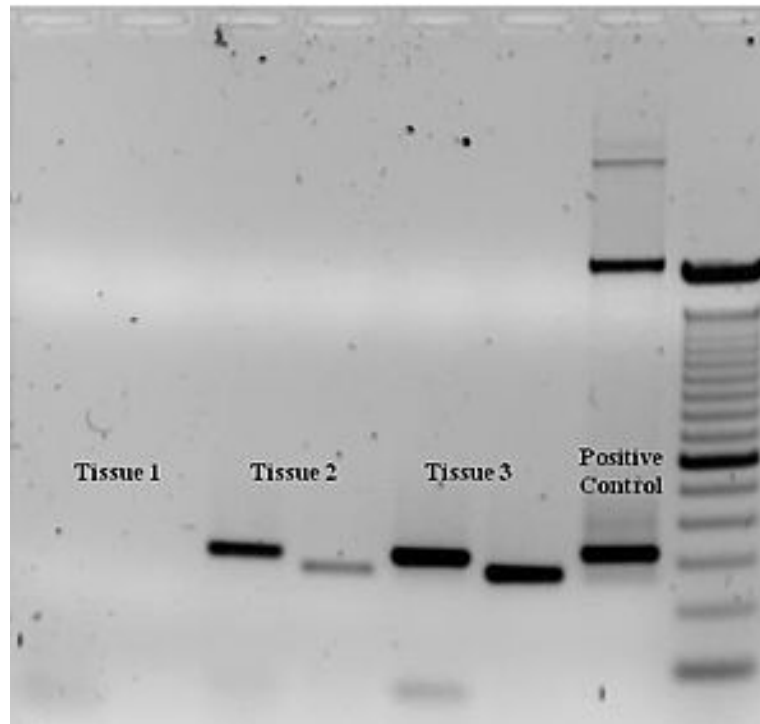
Figure 11 UV-visible Spectra of oxy- (HbO<sub>2</sub>) and deoxyhaemoglobin (Hb)

# Polymerase chain reaction - PCR



- 1 **Denaturation** at 94-96°C
- 2 **Annealing** at ~68°C
- 3 **Elongation** at ca. 72 °C





Ethidium bromide-stained PCR products after [gel electrophoresis](#). Two sets of primers were used to amplify a target sequence from three different tissue samples. No amplification is present in sample #1; DNA bands in sample #2 and #3 indicate successful amplification of the target sequence. The gel also shows a positive control, and a DNA ladder containing DNA fragments of defined length for sizing the bands in the experimental PCRs.