ICFA

REGIONAL INSTRUMENTATION SCHOOL

Istanbul, TURKEY
June 17-28, 2002

LABORATORY COURSE ON
INTERFACES AND DATA ACQUISITION

by

Selcuk Cihangir and Simon Kwan, *Fermilab, USA*
Marleigh Sheaff, *University of Wisconsin, USA*
Kerem Cankocak, *Mugla University, TURKEY*

Contents

Introduction

This is a course to introduce the students to LabView as a Data Acquisition (DAQ) and Analysis program. It is a product of a USA company called National Instruments (NI). **LabVIEW** (Laboratory Virtual Instrument Engineering Workbench) is a development environment based on graphical programming. LabVIEW uses terminology, icons, and ideas familiar to technicians, scientists, and engineers. LabVIEW relies on graphical symbols rather than text-based language to describe programming actions. LabView lends itself to a wide range of applications in science, education and industry. The scope of these applications varies from controlling a few instruments to monitoring assembly lines to test beam experiments in particle physics. In this course, students will follow step by step instructions to write programs first to become familiar with some of the basics of LabView. Then they will construct two simple DAQ setups utilizing LabView. Some debugging and DAQ system management utilities will be introduced at the end of the course.

Students are expected to do some investigative work on the programs by experimenting with menus, online help topics and notes, and the NI website (www.ni.com). Various Activity sessions will direct some of this work. Additional examples, online or on the course CD, are expected to supplement the course.

The duration of the course is insufficient to cover a lot of subjects and details of LabView. The hope is the students will appreciate the versatility, power and the convenience of LabView to use it in their future studies and research.

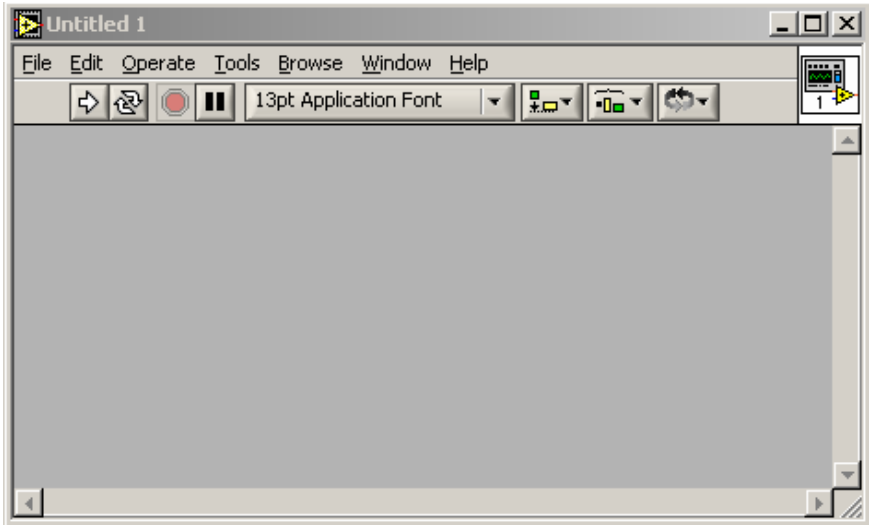
I. LABVIEW BASICS

The LabView basics can be itemized as follows:

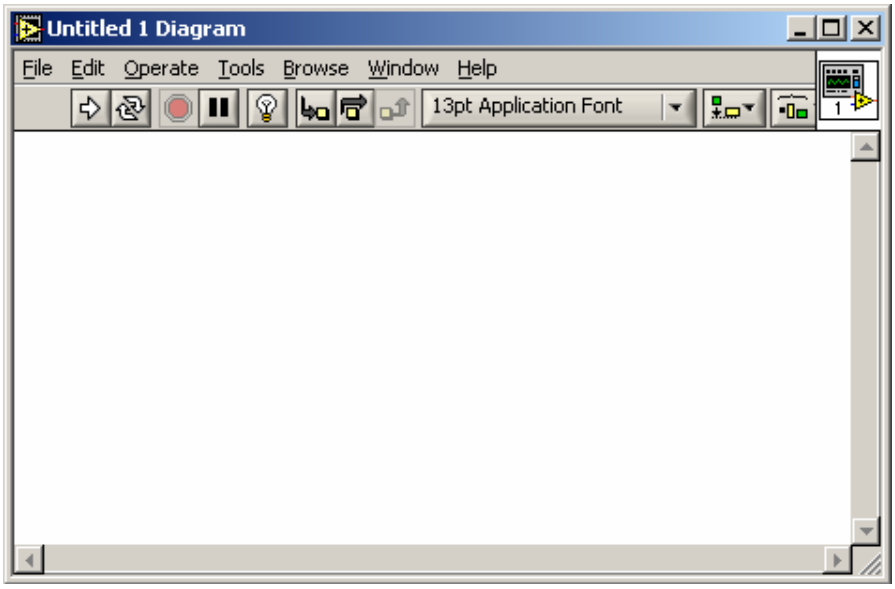
a. Graphical Language (G-Language) programs called Virtual Instruments (VIs)
b. Each VI has two main components:
   - Front Panel - How the user interacts with the VI.
   - Block Diagram - The code that contains graphical representations of functions to control the Front Panel objects may contain sub-VI's.
c. In Front Panel, means of interactions are:
   - Controls = Inputs to the program (numerals, strings, etc.)
   - Indicators = Outputs of the program (numerals, strings, plots, etc.)
d. In Block Diagrams are:
   - Icons – Means of representing operations and sub-VIs, have terminals defining inputs and outputs to the operations, the sub-VIs and representations of controls and indicators in Front Panel, are "wired" to each other or to constants.
   - Wires – Means of connecting operations and VIs to other operations, VIs, inputs and outputs, controls and indicators.

LabView Palettes are menu windows, collection of operations on and with the icons.

Activity1:
Start LabView (from the Start Menu, for instance), click on "New VI" to open an empty Front Panel and a Block Diagram. Right-click on the Front Panel to see the "Controls" palette, and on the Block Diagram to see the "Functions" palette. Observe the third palette called "Tools". Left-click on the menu items to see what they are. Some are conventional, some are specific to LabView. Try to figure out how to make the Controls, Functions and Tools palettes permanent on the desktop if they are not already.



A LabView Front Panel



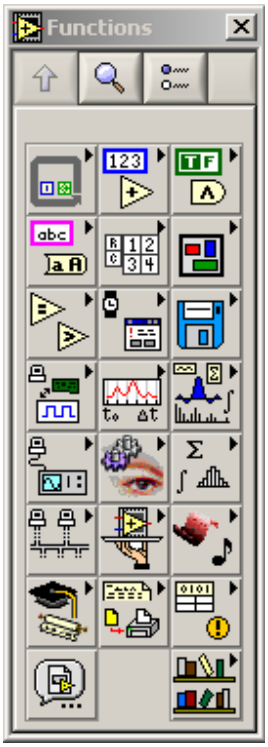
A LabView Block Diagram

II. LABVIEW PALETTES

Tool Palette: Bring the curser on the small icons to see the pop-up names for each tool, such as Operating Tool, Positioning Tool, Labeling Tool, Wiring Tool, and others. With clicking on the tool icon, the cursor takes the shape of the icon when it is on the Front Panel or the Block Diagram.

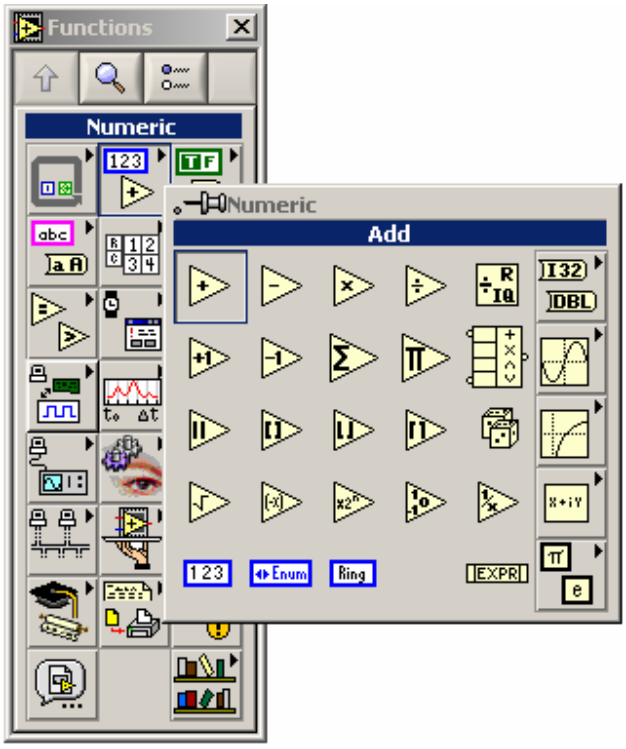
Control Palette: Bring the cursor on the icons to see the pop-up names and sub-icons and their names on this palette, such as Numeric, Boolean, String/Path, Array/Cluster, Graph, etc. Experiment with the small icons at the top of the palette.

Notice that this palette is highlighted and operational when the Front Panel is active.



Function Palette: Bring the cursor on the icons to see the pop-up names and sub-icons and their names on this palette, such as Structures, Numeric, Boolean, String, Array, Cluster, etc. Experiment with the small icons at the top of the palette.

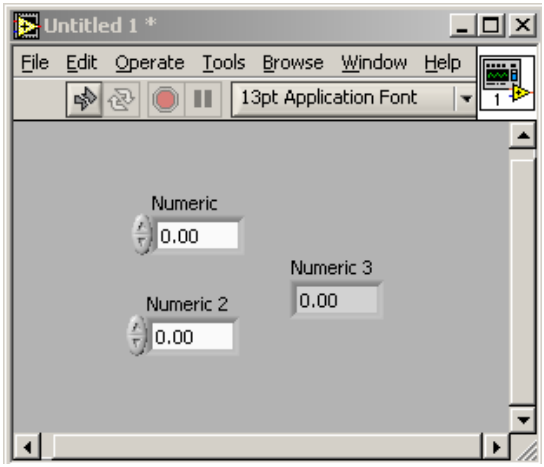
Notice that this palette is highlighted and operational when the Block Diagram is active.

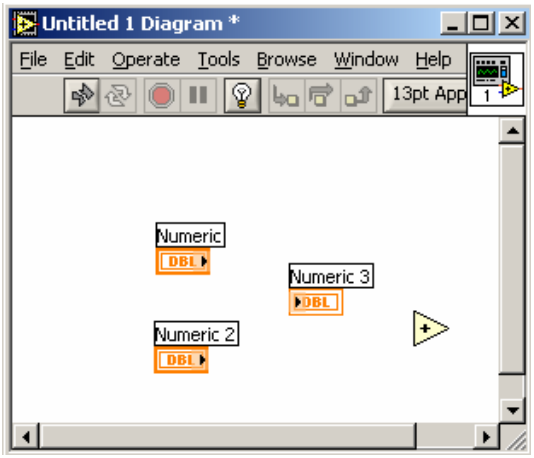


Bring the cursor on one of the functions on the Function Palette, for instance to produces the following selection. Bring the cursor on any icon on the pop-up box to see the names of the icons. These are operations that are used in Block Diagrams, the program itself. To bring or attach the operation (icon) on a Block Diagram, right-click on the icon, bring the curser on the Block Diagram and right-click again.

Same way, one can attach an icon from the Control Palette to a Front Panel.

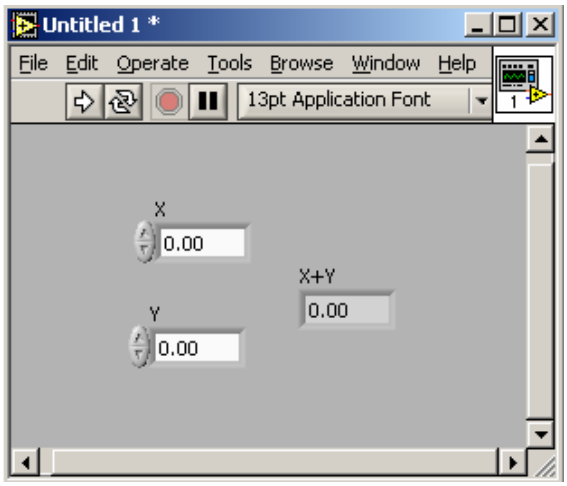
Activity 2:
Start LabView and open a new Front Panel and its Block Diagram. Attach some icons, say two numeric controllers and one numeric indicator to the Front Panel. Notice the corresponding icons appearing on the Block Diagram. Attach an Add icon to the Block Diagram. Notice that this does not produce anything new on the Front Panel. You now have the following windows:

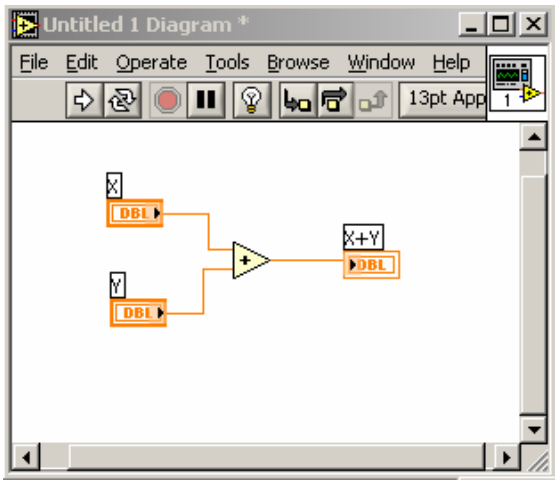




Notice the broken arrows on the top-left corners of each window, indicating that something is wrong. Click on one of them and study the dialog box that appears.

Using Tools Palette/Labeling Tool (click on the tool, then click on the label) change the labels of the items in the Front Panel. Notice that the label changes in the Block Diagram. Use Positioning tool to move the icons around. Use the Wiring Tool to connect the Block Diagram icons as follows (click on the wiring tool, then click on the icon to wire from and then click on the icon to wire to)





Notice that the broken arrow is not anymore. That means you can run this VI. Do so by choosing X and Y values (controllers or input values), and read the sum on the X+Y output or indicator.

You can save your work by Save or Save As in File menu on either window, and give it a name (you do not need to save this VI). A VI is closed when the Front Panel is closed. Closing the Block Diagram first does not prompt any action on the operation system level, i.e. save the changes, etc.

III. PROGRAMMING WITH LABVIEW

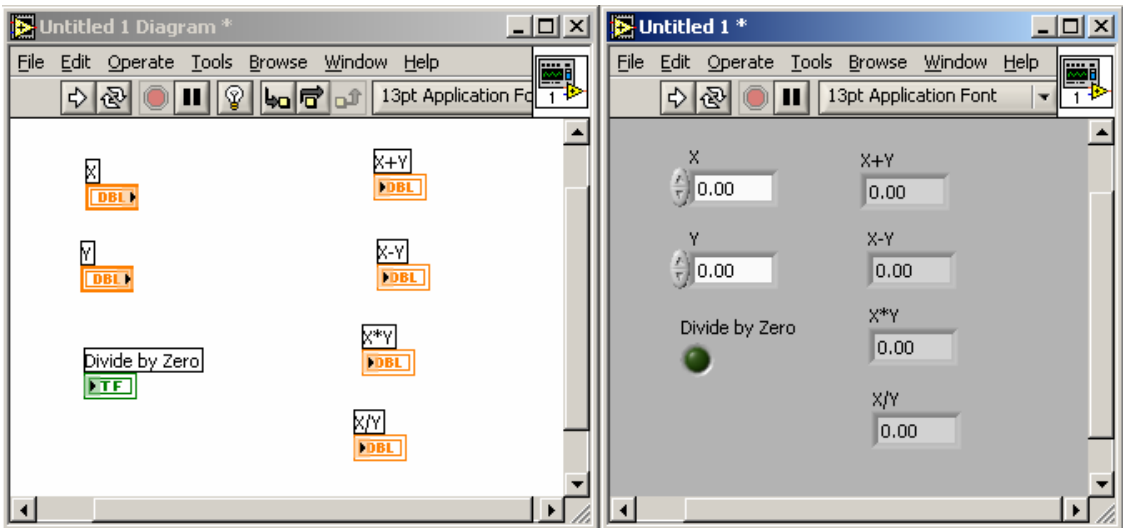
a) Basics: A Simple Calculator

The object of this exercise is to practice what we learned so far before we are introduced to new concepts in the following activities. We will construct a LabView program doing addition, subtraction, multiplication and division with two numbers and displays the answers. In division, if the divider is zero, a warning light comes on.

1) Open a New VI (either from LabView startup window or from File menu if you already have any VI open). If the palettes (Tools, Controls and Functions) are not open, open them from Windows menu of the Front Panel or the Block Diagram. You can also pop them by right clicking on the panels.

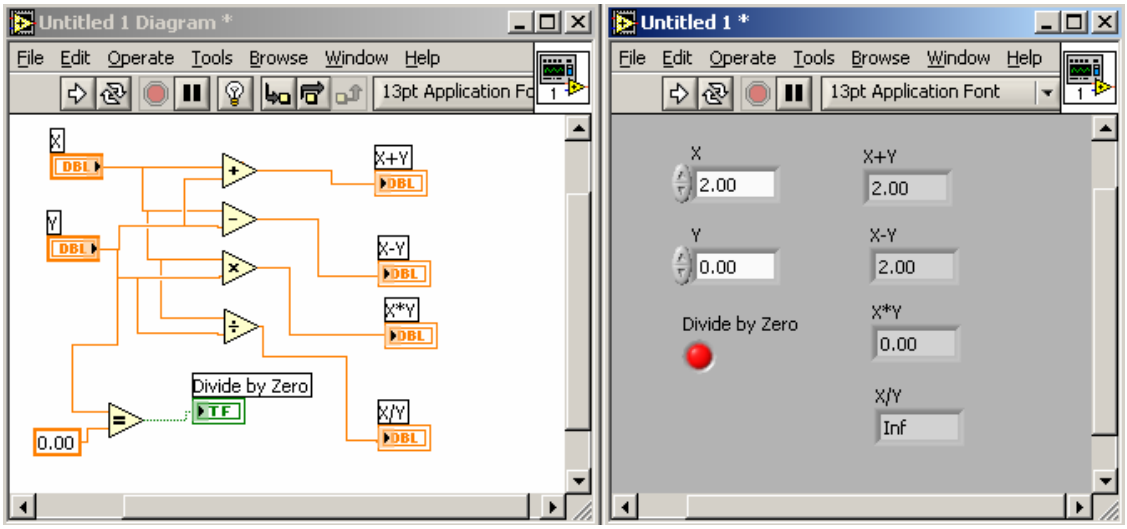
2) From the Controls/Numeric drop two Digital Controls on the Front Panel (click on the icon, bring the cursor on the Front Panel and click again) and label them (with Labeling tool) X and Y. These are the inputs to the four arithmetic operations.

3) Same way, drop four Digital Indicators and label them X+Y, X-Y, X*Y and X/Y. These are the outputs to the operations. Also drop from the Controls/Boolean a Round LED and label it as Divide by Zero. You now have the following windows:



4) We now need to perform the four operations on the inputs (in the Block Diagram) by bringing in the corresponding icons from the Functions palette. From the Functions/Numeric drop the Add, Subtract, Multiply and Divide icons to the Block Diagram and wire (with the Wiring tool) the inputs (the Digital controls X and Y) to the input terminals (on the left side of the icons) of the each operation icon. Notice that when the Wiring tool is over the icons, the terminals appear and blink. Wire the output terminals of the operations to the outputs (the Digital indicators X+Y, X-Y, X*Y and X/Y). You can have more than one wire out from an indicator or control.

5) For the warning light, from Functions/Comparison drag an Equal? icon to the Block Diagram, wire Y and a Numeric Constant 0 (zero) to the terminals of the Equal?. You can drop the Numeric Constant from Functions/Numeric and set it equal to zero with Labeling tool or Operating tool. You can also right click on the Equal? icon, choose from the pop-up menu create/constant and set the constant to zero. The output of the Equal? is a Boolean True or False, which you connect to the Divide by Zero (Boolean LED). Change the color to red when the output is true by using the Coloring tool. With the Operating tool, you can select the input values and run the VI by pressing the Run arrow on either Front Panel or the Block Diagram. You can save the VI with a name of your choice. It is called SimpleCalculator.vi in the course CD. You should have the following windows for your program:



Activity 3:
From the Help menu, choose the Show Context Help to see a help-window. Bring the cursor to any item on the Front Panel or the Block Diagram (click on the window to highlight the panel or the diagram) and observe the help window. Repeat after changing the cursor to the Wiring tool.

b) Structures: For Loop

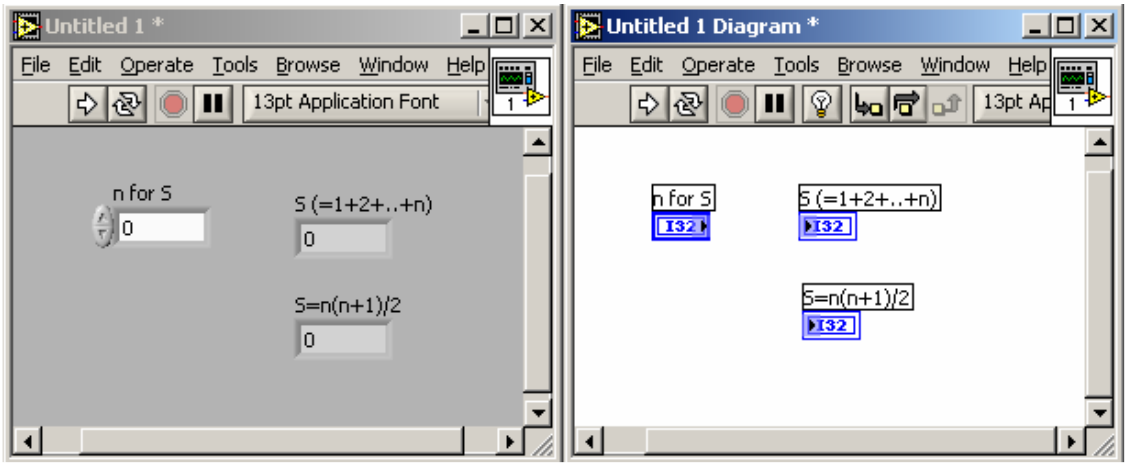
Subjects to learn: For Loop Structure, Shift Registers.
Function to learn: Increment.

The object of this exercise is to practice with the For Loop Structure. For Loop is the "DO" loop in Fortran. In this exercise, we will add the integers from 1 to n and display the sum. We will also calculate the formula that gives this sum to see that they are the same.

1) Open a New VI (either from LabView startup window or from File menu if you already have any VI open). If the palettes (Tools, Controls and Functions) are not open, open them from Windows menu of the Front Panel or the Block Diagram. You can also poop them by right clicking on the panels.

2) From the Controls/Numeric drop one Digital Control on the Front Panel (click on the icon and then click on the Front Panel) and label it (with Labeling tool) "n for S", S for sum.

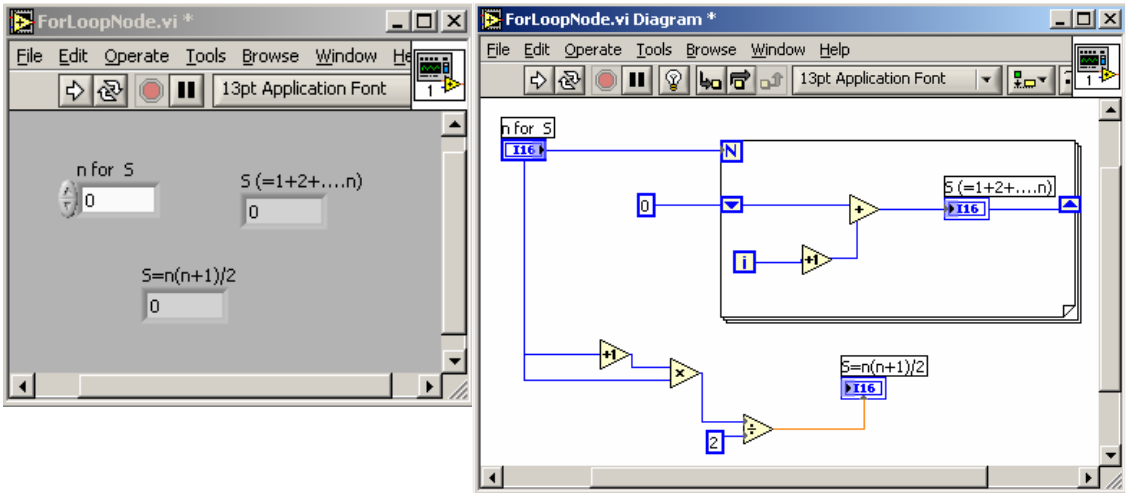
3) Same way, drop two Digital Indicators and label them "S (=1+2+..+n)" and "S=n(n+1)/2", the formula for the sum. Right click on these items, choose Representation from the pop-up menu and then choose I32, for integer. You now have the following windows:



4) We now need to perform the sum and evaluate the formula (in the Block Diagram). From the Functions/Structures click on the For Loop, bring the cursor on to the Block Diagram and click and drag the cursor to expand the loop. Wire (with the Wiring tool) the "n for S" to the N-box at upper left corner of the loop. Right-click on the left or right edge of the loop and choose "Add Shift Register". Wire a constant 0 (zero) the shift register on the left. This initializes the register. From Functions/Numeric bring in Add and wire the left shift register to one of the inputs of Add and the I-box in the loop to the other input after adding 1 to it (by Functions/Numeric/Increment). This is because the

value of I-box varies from 0 (zero) to N-1 during the For Loop. Bring in the Block Diagram the indicator "S =(1+2+..+n)" and wire the output terminal of the Add to it. Wire it also to the shift register on the right. With the shift register, the value of the indicator "S =(1+2+..+n)" is carried over to the next loop.

5) For evaluating the formula, from Functions/Numeric bring in Multiply and Divide to the Block Diagram, wire "n for S" to Multiply's both inputs after adding 1 with Increment to one of them. Divide the output by 2 with the Divide and wire the output of Divide to "S=n(n+1)/2". With the Operating tool, you can select the input value and run the VI by pressing the Run arrow on either Front Panel or the Block Diagram. You can save the VI with a name of your choice. It is called ForLoopNode.vi in the course CD. The program looks as follows:



c) Structures: Sequence

Subjects to learn: Sequence Structure, a graphing technique, Local Variables.
Function to learn: Sine, Wait (timing), Bundle, Waveform Graph, Build Array.

The object of this exercise is to practice with the Sequence Structure. In this structure, operations contained in a series of boxes are processed sequentially, starting from the box numbered 0 (zero) to the last one. In this exercise, we will evaluate Sine and square-Sine of angles from 0 to 10π (in increment of 0.1π) and display them in two separate graphs and then together on a third graph. We will do these in sequences separated from each other by 3 seconds.

1) Open a New VI and from the Controls/Graph drop three Waveform Graph Indicators on the Front Panel. Label them Sin X, Sin^2 X and Together. In the Block Diagram, drop in from Functions/Structures a Sequence structure while keeping the graph icons outside. Resize it if necessary. Right-click on the top edge of the Sequence box and choose Add Frame After to add a frame (Add Frame Before would to it also). This way,

add frames to have 5 frames (labeled automatically from 0 to 4). Practice to switch to frames using the small arrows on the top edge of the box.

2) We will evaluate Sine of angles in the first sequence of the structure. Choose the sequence 0, bring in the Sin X graph indicator in it. Bring in a For Loop and wire 101 to N. Outside the For Loop, evaluate $\pi/10$ using π (from Functions/Numeric/Additional Numeric Constants/ Pi). Bring in Sine from Functions/Numeric/Trigonometric into the For Loop, evaluate the angle value $I*\pi/10$ using I-box (I=0,1,…,100) and Multiply function. Wire the angle value to the input of Sine function.

3) From Help menu choose Show Context Help, bring the cursor on the Sin X in the Block Diagram to see the description of this indicator, Waveform Graph. As the description suggests, bring in a Bundle from Functions/Clusters, right-click on it to Add Input (so that you have three inputs). Wire a constant 0.0 to the first input, wire $\pi/10$ that you evaluated outside the For Loop to the second input, and the output of the Sine to the third input. Make sure that at the exit of the For Loop for the Sine value is indexed (i.e. when you right-click on the exit box, you see in the pop-up menu the choice Disable Indexing, meaning Indexing is already enabled). Wire the output of the Bundle to the graph Sin X.

4) Go to the second sequence (labeled 1), bring in a Wait from Functions/Time & Dialog and wire to it 3000 (3 seconds). Do the same to the sequence labeled 3.

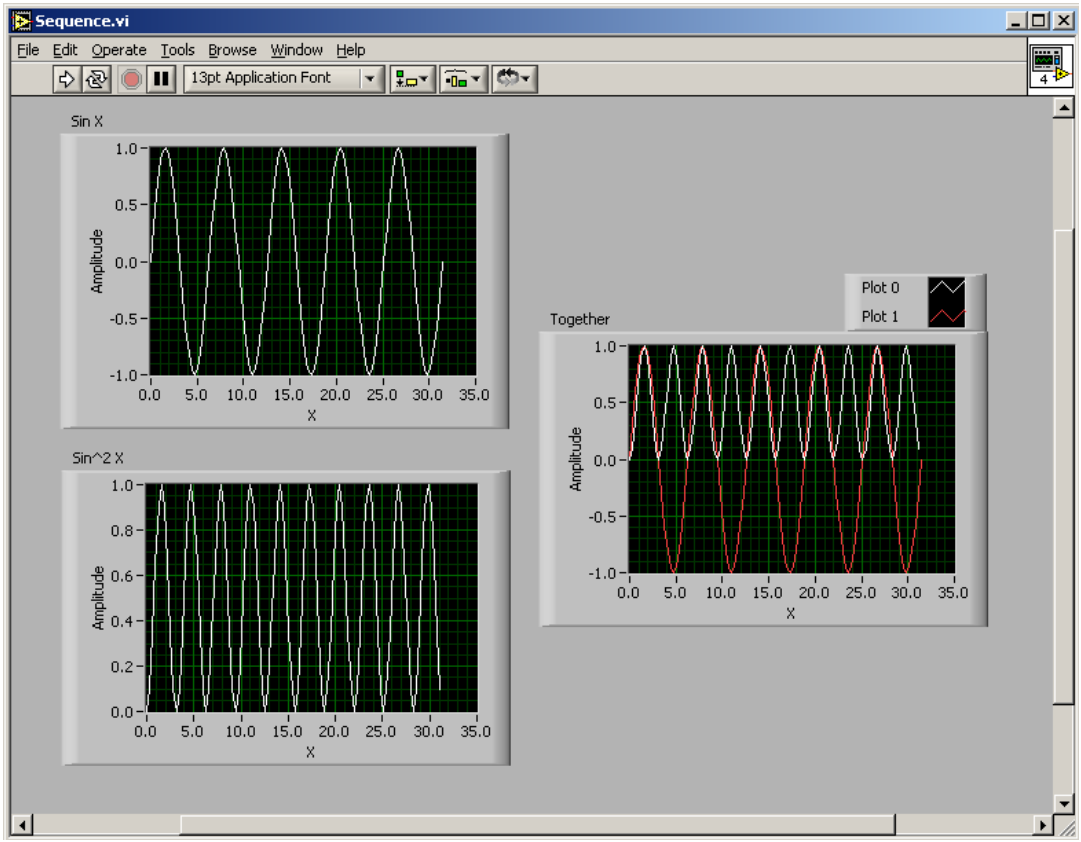
5) In sequence 2, duplicate the sequence 0, add another Sine and multiple two Sine functions by using a Multiply function and wire the output of Multiply to the third input of Bundle. Bring in the Sin^2 X icon into the sequence and wire the output of Bundle to it.

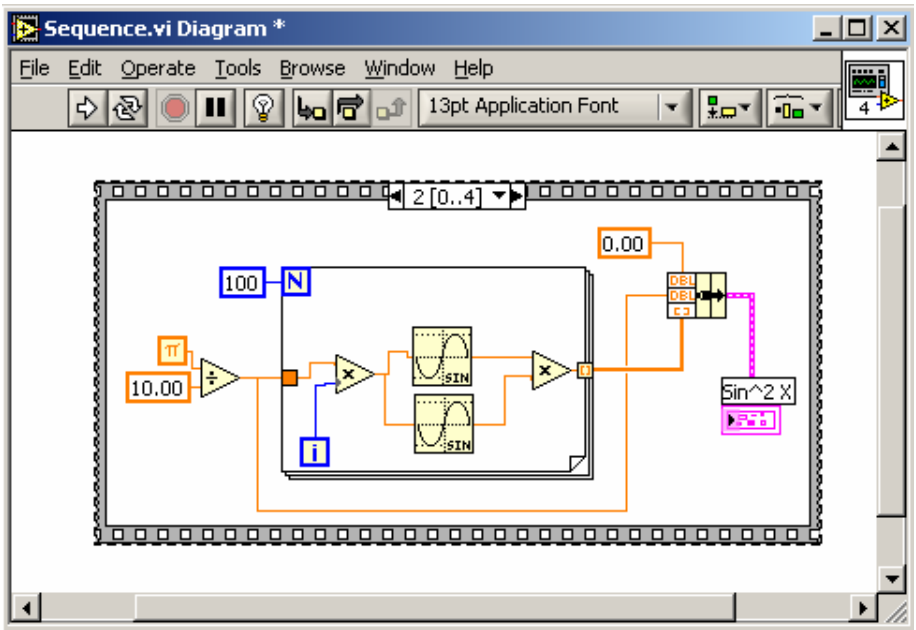
6) For plotting the Sin X and Sin^2 X on the same graph in the last sequence, we need to have the copies of them in that sequence. Go to the sequence that these graphs reside and right-click on them to choose Create/Local Variable from the pop-up menu. Drag them out of the box and bring to the last sequence. Right click on them to choose Change to Read to make them readable rather then writeable, if they are not readable already. Notice the change on the borders of the icon. Add a Build Array using Functions/Arrays, wire Sin X and Sin^2 X local variable icons to its inputs (you need two inputs, add inputs if necessary by right-clicking on Bundle and choosing Add Input from the pop-up menu). Wire the output of Build Array to the Together graph that you also bring in to this sequence. Run the program to see the results.

Activity 4:
Modify the program to add a light that comes on Red during the three seconds wait period, and turns Green at other times. Also add a control to choose the range of angles.

Activity 5:
Explore options with the graphs by right-clicking on the plots (in Front Panel) and choosing the items in the pop-up menu.

You can save the VI with a name of your choice. It is called Sequence.vi in the course CD. Now you have the following windows:





d) Structures: Case

Subjects to learn: Case Structure.
Function to learn: Random Number Generator, Greater or Equal?.

The object of this exercise is to practice with the Case Structure. In this structure, operations contained in a series of boxes are processed for the cases they are labeled with. The cases can be True or False, or numeric starting from the case numbered 0 (zero) to the last one. In this exercise, we will create a random number, check to see if it is less than 0.5. If it s, a red light comes on, otherwise a green light comes on.

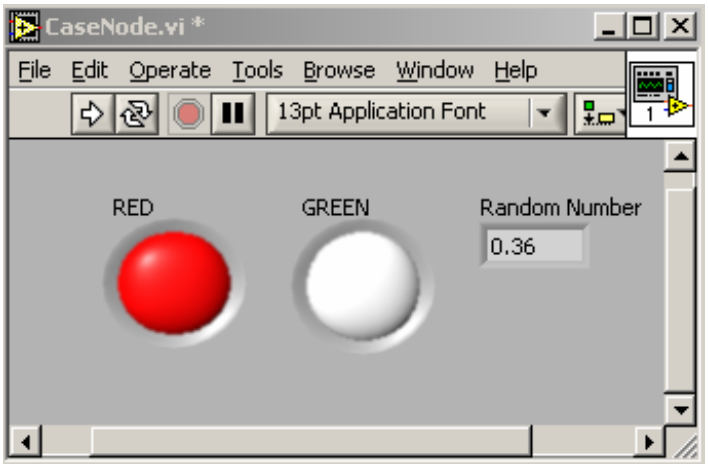
1) Open a New VI and from the Controls/Boolean drop two Round LED Indicators on the Front Panel. If they are not indicators, change to indicators by right-clicking on them and choosing Change to Indicator on the pop-up menu. Label them RED and GREEN. With Set Color tool, color them white. Also drop a numeric constant indicator and label it Random Number. In the Block Diagram, drop in a Random Number (0-1) from Functions/Numeric and two Case Structures from Functions/Structures while keeping other icons outside. Resize them if necessary. As default, the cases are True and False.

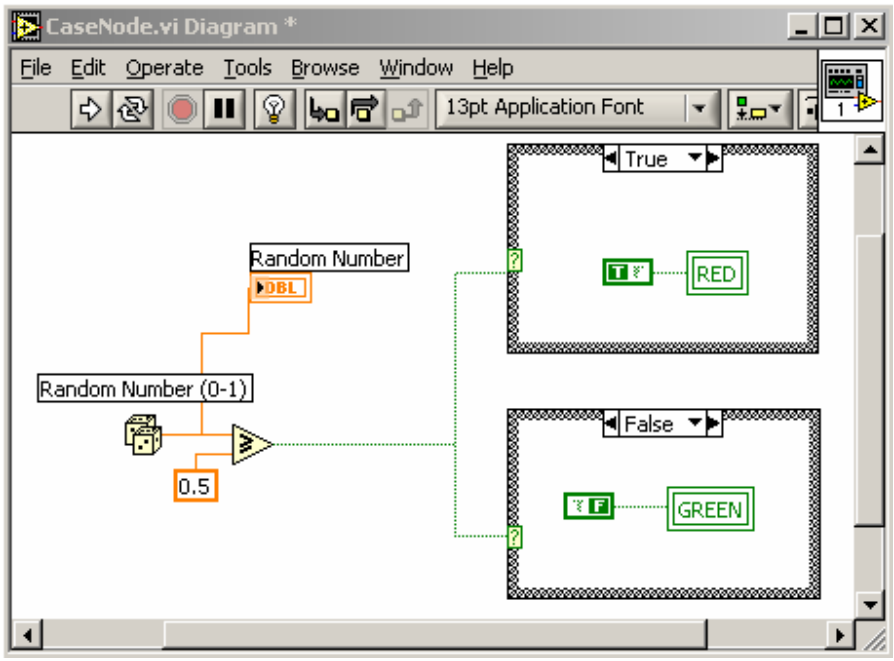
2) Bring in a Greater or Equal? from Functions/Comparison and wire the output of Random Number generator to one of its input and a constant (=0.5) to the other input. Wire the output to the question marks on the left edge of the Case Structure boxes via two separate wires. Choose on one of the case structures True (with the arrows on the top edge) and bring in the boolean indicator labeled GREEN in it. On the other structure, bring in the RED indicator while it is False. Wire the Random Number generator to the Random Number indicator with a second wire from the generator.

3) Create local variables of each indicators, bring them to the other case boxes in each structure. Right-click on the local variables, create a constant (boolean as default), by using Operate Value tool, switch the constants to the same case as the case of the Case Structure the variables are in, and wire them to the local variables. Run the program and change the colors of the lights accordingly (RED is red when the random number is less than 0.5, white otherwise; and, GREEN is green when the number is greater or equal to 0.5 and white otherwise).

Activity 6:
Modify the case of a Case Structure to have numeric cases by wiring an integer constant to the question mark box. Add more cases or delete by right-clicking on the case indicator on the top edge of the structure and choosing the necessary option from the pop-up menu. Undo the changes when finished using Edit menu repeatedly.

You can save the VI with a name of your choice. It is called CaseNode.vi in the course CD. Now you have the following windows:





e) Structures: Formula Node

Subjects to learn: Formula Node Structure.

The object of this exercise is to practice with the Formula Structure. In this structure, one can evaluate mathematical formulae and expressions similar to C-language. There are built-in functions such as abs, sine, log to operate on input parameters that are introduced to the structure on the left edge of the structure box. The outputs are accessed on the right edge of the box.

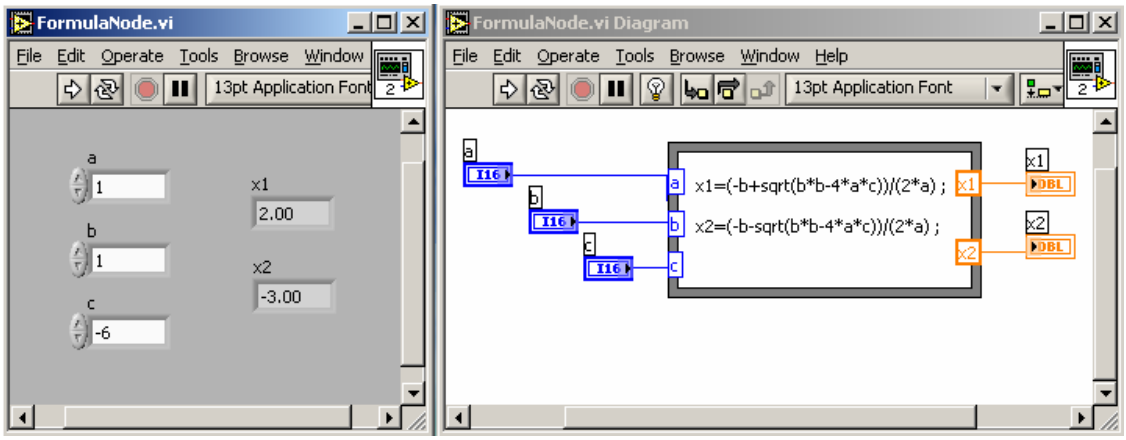
Activity 7:
After dropping in a Formula Node structure in a Block Diagram, use Show Context Help to get more information about this structure.

In this exercise, we will solve a quadratic equation $ax^2+bx+c=0$ for given a, b, and c.

1) Open a New VI and from the Controls/Numeric drop three Digital Controls and two Digital Indicators on the Front Panel. Label the controls a, b, c, and the indicators x1 and x2. Change the controls to I32 by right-clicking on them and choosing Representation and then I32 from the pop-up menu.

2) Bring in to the Block Diagram a Formula Node from Functions/Structures and right-click on the left edge to add inputs from the pop-menu, and on the right edge to add outputs. Notice the differences in the borders of the inputs and outputs. Wire the controls a, b and c to the inputs, and the indicators x1 and x2 to the outputs of the structure. Type in the structure the expressions to calculate x1 and x2, i.e. x1=(-b+sqrt(b*b-4*a*c))/(2*a) and x1=(-b-sqrt(b*b-4*a*c))/(2*a). End each line with a ";".

3) Choose values for a, b and c, and run the program. You can save the VI with a name of your choice. It is called FormulaNode.vi in the course CD. Now you have the following windows:



f) Arrays and Clusters, and Polymorphism

Subjects to learn: Polymorphism, Arrays and Clusters.

An array is a collection of data elements of the same type, i.e. numbers, Booleans, strings, etc., all controls or indicators. A simple example is an array of numbers in one or more dimensions (indices). A cluster is group of data of different types. For instance, we can have numeric and string variables in a cluster, but not in an array.

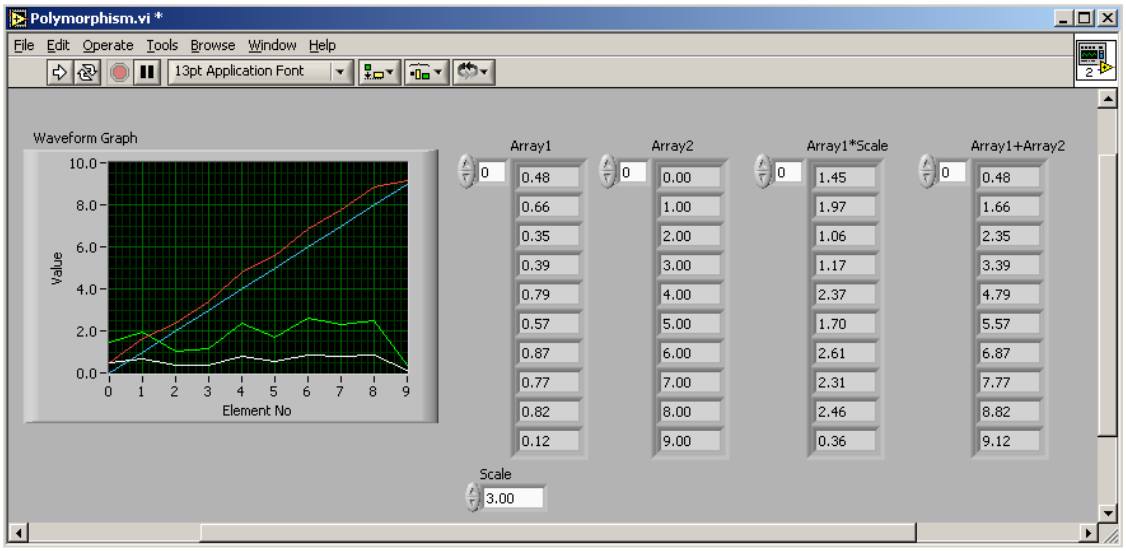
Polymorphism is a principle that the inputs to arithmetic functions (such as Add, Multiply, etc) can be of different size and representation. For instance, we can Add an array and a numeric constant, or Multiply two arrays or an array and a constant by one operation as opposed to operating on each element of the arrays.

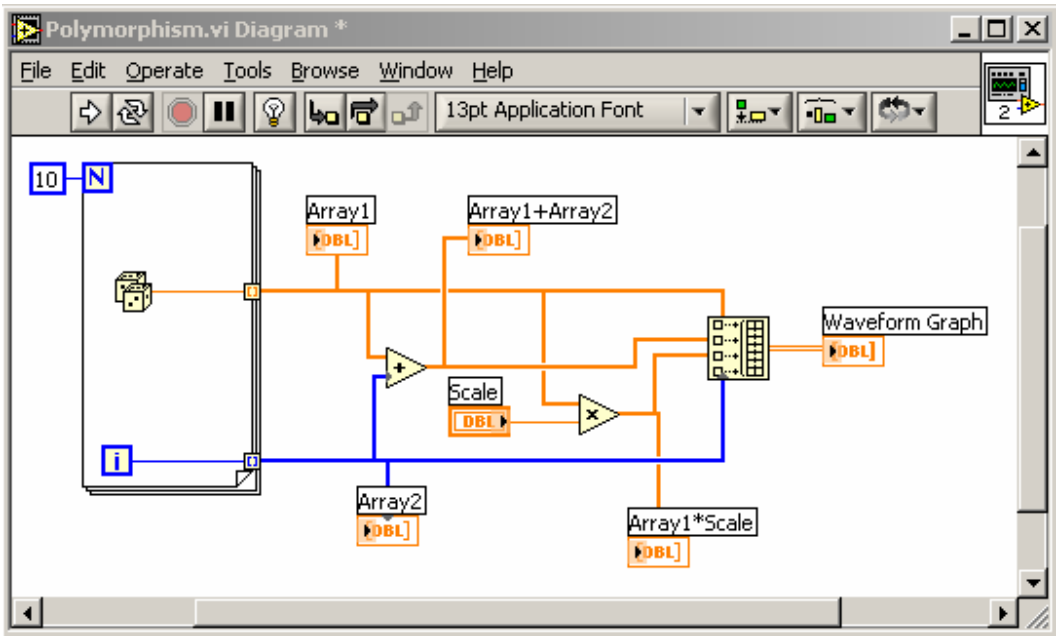
The object of this exercise is to practice with arrays while demonstrating polymorphism.

1) Open a New VI and from the Controls/Array&Cluster drop four Array elements and from the Controls/Numeric one Digital Control into the Front Panel. Label the arrays Array1, Array2, Array1*Scale and Array1+Array2, the digital control Scale. Drop a Numeric Indicator into each Array box (and notice the change). Resize the Arrays to show 10 elements, all grayed. Also drop a Waveform Graph into the Front Panel. Change the x-label to Element No and the y-label to Value. Right click on the graph and on the x-axis and the y-axis to study the pop-up menus.

2) Bring in to the Block Diagram a Random Number in a For Loop and Build Array outside of it. Resize the Build Array to have four inputs (by the Position/Size tool or by right-clicking on the inputs and choosing Add Input from the pop-up menu). Bring in also an Add and a Multiply functions

3) Wire a constant 10 to the N of the For Loop, Random Number to Array1, and I-box of the For Loop to Array2. Make sure at the exit of the For Loop the indexing is enabled. This makes the element of Array1 the random numbers generated, and those of Array2 integers from 0 to 9. Then wire Array1 and Array2 to the inputs of Add, Array1 and Scale to the inputs of Multiply. Wire Array1, Array2, and outputs of Add and Multiply to the inputs of Build Array and also the outputs of Add and Multiply to Array1+Array2 and Array1*Scale, respectively. And finally wire the output of the Build Array to the Waveform Graph. Run the program. You can save the VI with a name of your choice. It is called Polymorphism.vi in the course CD. Now you have the following windows:





Activity 8:
Add a fifth array indicator, calculate Array1*Array2 on this array and plot the elements on the same graph.

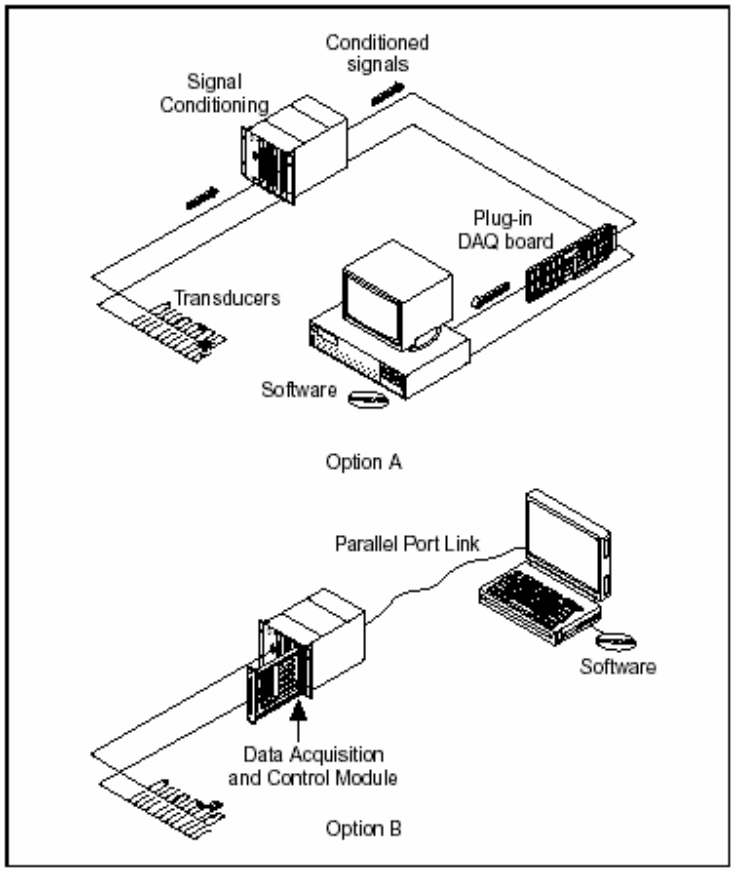
IV. DAQ WITH LABVIEW

Major components of a Data Acquisition system are instruments/devices, a computer and its application software, a hardware interface between the instruments and the computer hardware, and a software interface (driver) to carry on the data flow between the instruments and the computer. Some instruments further require a protocol/bus to communicate with the hardware interface. A schematic view of two possible setups is shown in the figure below. We make measurements with instruments. Instrumentation helps science and technology progress. Scientists and engineers around the world use instruments to observe, control, and understand the physical universe. In LabView, the programs and the operations/functions are called Virtual Instruments (VIs). The meaning of this expression becomes clearer when we do DAQ.

a) What Is Virtual Instrumentation?

Virtual instrumentation is defined as combining hardware and software with industry-standard computer technologies to create user-defined instrumentation solutions. We can use virtual instrumentation to create a customized system for test, measurement, and industrial automation by combining different hardware and software components. DAQ systems are concerned with the acquisition, analysis, and presentation of measurements and other data we collect. Acquisition is the means by which physical

signals, such as voltage, current, pressure, and temperature, are converted into digital formats and brought into the computer. Popular methods for acquiring data include plug-



Schematic view of a DAQ system for a desk-top computer (option A) where the hardware interface is plugged in the computer, and for a lap-top computer (option B) where the hardware interface is housed outside the computer.

in DAQ and instrument devices, GPIB instruments, VXI instruments, and RS-232 instruments. We will use the first two, plug-in DAQ and GPIB device/instruments, methods in this course. Data analysis transforms raw data into meaningful information. This can involve such things as curve fitting, statistical analysis, frequency response, or other numerical operations. Data presentation is the means for communicating with the system in an intuitive, meaningful format such as graphs and plots.

b) How Do Computers Talk to DAQ Devices?

Before a computer-based system can measure a physical signal, a sensor or transducer must convert the physical signal into an electrical one, such as voltage or current. A plug-in DAQ device, in our case the National Instruments' PCI-6024E board, is one of the components of the DAQ system. Unlike most stand-alone instruments, we cannot always directly connect signals to a plug-in DAQ device. In these cases, we use accessories to condition the signals before the plug-in DAQ device converts them to digital information. In our setup we have NI Signal Accessory. The software, NI-DAQ along with LabView in our case, controls the DAQ system by acquiring the raw data, analyzing the data, and presenting the results. The computer receives raw data. Software

takes the raw data and presents it in a form the user can understand. Software manipulates the data so it can appear in a graph or chart, or in a file for report. The software also controls the DAQ system, telling the DAQ device when to acquire data, as well as from which channels to acquire data.

Typically, DAQ software includes drivers (NI-DAQ, NI-488.2, etc.) and application (LabView) software. Drivers are unique to the device or type of device and include the set of commands the device accepts. Application software (such as LabVIEW) sends the commands to the drivers, for instance to acquire a thermocouple reading and return the reading, then displays and analyzes the data acquired. LabVIEW includes a set of VIs that let us configure, acquire data from, and send data to DAQ devices. This saves us the trouble of having to write those programs ourselves.

c) How Do Computers Talk to GPIB Instruments?

The fundamental task of an instrument is to measure some natural phenomenon. Unlike in DAQ devices, the signal the computer ultimately receives from a GPIB instrument requires no conditioning. GPIB is one of the common types of instruments (the others being Serial Port, VXI, PXI, Computer-based instruments, etc.) All external instruments communicate with the computer through a bus (GPIB bus in our case) where a communication protocol has been defined. The hardware interface of this bus to the computer hardware is a GBPI card that is plugged in the computer. It has its software interface (driver) called NI-488.2 in our system. The instruments have a set of general commands that they understand. They also have instrument-specific commands. The user writes an application program with an application software (such as LabView) that sends commands to and receives data from the instruments, analyze the data and presents the information in form of plots, graphs, etc.

LabView has a collection of functions and VIs for all these purposes. Some (analysis and presentation of data) we are familiar with already. We will learn about the others (communicating with the instruments) next. There are tools called Instrument Drivers, which are collections of functions and VIs that implement the commands necessary to perform the instrument's operations. LabVIEW instrument drivers simplify instrument programming to high-level commands, so we do not need to learn the low-level instrument-specific syntax needed to control our instruments. Instrument drivers are not necessary to use our instrument. In fact, in the program we will develop later we will not use any. They are merely time savers to help us develop our project so we do not need to study the instrument manual before writing a program. Instrument drivers create the instrument commands and communicate with the instrument over the GPIB bus (or any other bus). LabVIEW provides more than 700 instrument drivers from more than 50 vendors and they are accessible through NI web site.

d) Data Acquisition with a DAQ board

The object of this exercise is to practice with the DAQ functions. We will define two DAQ channels and read the data from them. We will display the data and plot them

in real-time. The channels are for measuring the room temperature, one with a thermocouple, one with a temperature sensor. These devices are mounted on a signal processor called NI Signal Accessory and they have assigned channel numbers on the plug-in DAQ board. Temperature sensor is actually a voltage level such that Temperature=100*Volt.

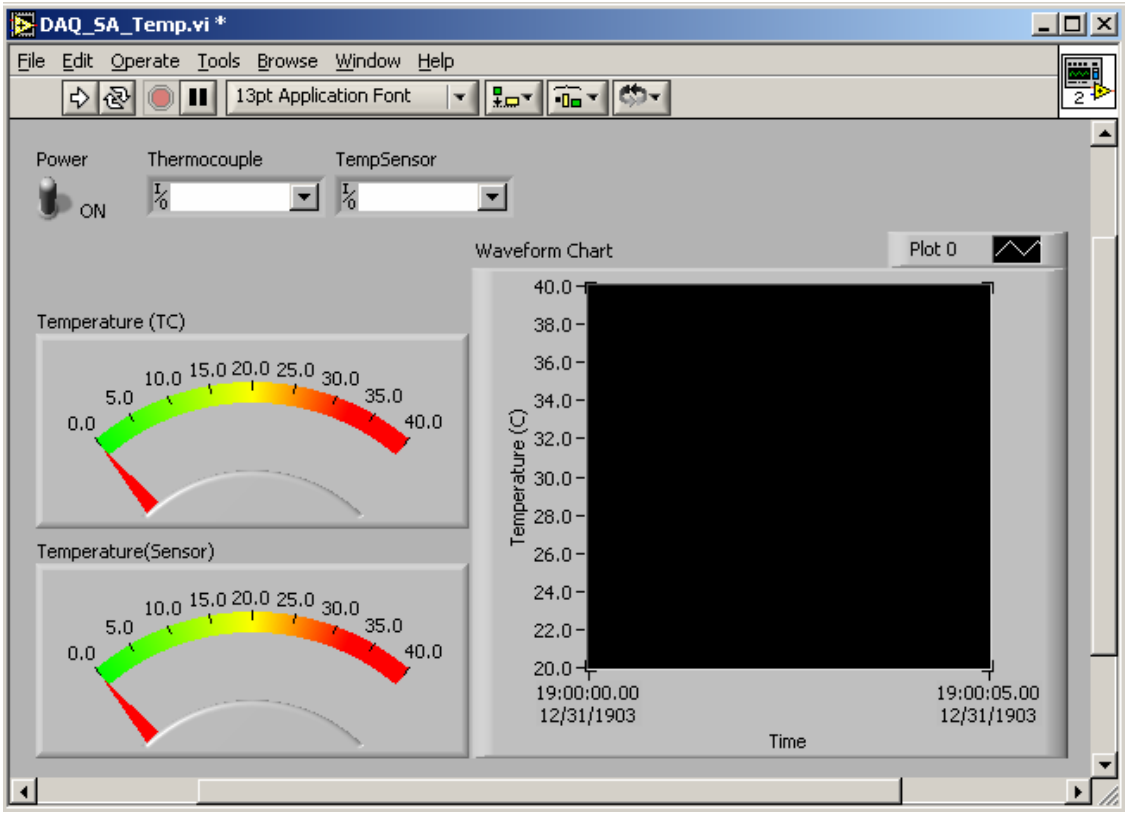
1) Open a New VI and from the Controls/I/O drop two DAQ Channel Name on the Front Panel. If they are not controls, change to controls by right-clicking on them and choosing Change to Control on the pop-up menu. Label them Thermocouple and TempSensor. Also drop two Meter indicators from Controls/Numeric and label them Temperature(TC) and Temperature(Sensor). Drop in a Waveform Chart and label the y-axis Temperature and change the scale to (20.0 – 40.0). Right-click on the x-axis scale and choose Formatting. In the dialog window's Format box, choose Time&Date. Add a Boolean Vertical Toggle Switch, name it Power, and show its Boolean Text and change the Mechanical Action to Switch When Pressed (both by right-clicking on the switch and choosing the appropriate item in the pop-up menu).

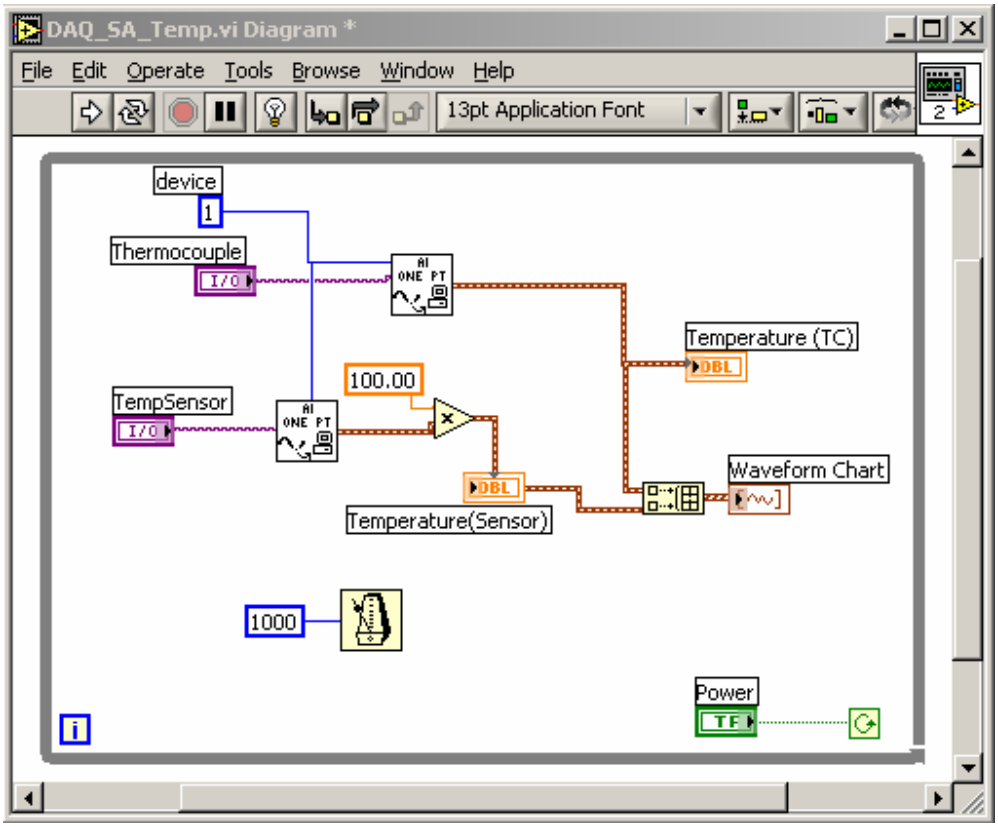
2) Right-click on the "DAQ Channel Name" Thermocouple and choose "New DAQ Channel" from the pop-up menu. This brings up a new dialog window. Choose Analog Input and click Next. Type a Channel Name (Thermocouple, for instance) and click Next. Select J-Thermocouple on the selection window, click Next three times. Each time notice the various assignments to various parameters of the channel. At the last window, notice the DAQ Hardware Used and choose a channel number 4. Click Finish. Do the same for the "DAQ Channel Name" TempSensor with appropriate name and Voltage as type of measurement. Choose 0 (zero) for channel number. We now defined two channels for the DAQ. We see these names and choose them for any DAQ Channel Name control by right-clicking on them and selecting the name in the pop-up list.

3) On the Block Diagram, include everything into a While Loop and wire Power to its Conditional Terminal at the bottom-right corner. From Functions/Data Acquisition/ Analog Input bring in two AI Sample Channel.vi's. Wire constant 1 (for Device number 1) to the device inputs of them (use Show Context Help to guide yourself), and wire Thermocouple and TempSensor to the channel(0) inputs of them. Wire their outputs (after multiplying the one for TempSensor with 100 with a Multiply function) to a Build Array and wire the output of Build Array to the Waveform Chart to have both plots on the same chart. Branch out from the outputs to the meters Temperature(TC) and Temperature(Sensor). And finally add a Wait Until Next ms Multiple and wire a constant 1000 (1 second) to it. This makes the loop to run, and therefore the temperature measurement to take place, once every second.

4) Run the program. Touch the tip of the thermocouple and/or the temperature sensor to see the change in the readings indicating your body temperature. Observe the meters and the jitter in the temperature readings.

You can save the VI with a name of your choice. It is called DAQ_SA_Temp.vi in the course CD. Now you have the following windows:


DAQ_SA_Temp.vi *
File Edit Operate Tools Browse Window Help
13pt Application Font
Power
ON
Thermocouple
TempSensor
Waveform Chart
Plot 0
Temperature (TC)
0.0 5.0 10.0 15.0 20.0 25.0 30.0 35.0 40.0
Temperature(Sensor)
0.0 5.0 10.0 15.0 20.0 25.0 30.0 35.0 40.0
40.0 38.0 36.0 34.0 32.0 30.0 28.0 26.0 24.0 22.0 20.0
Temperature (C)
19:00:00.00
12/31/1903
19:00:05.00
12/31/1903
Time


DAQ_SA_Temp.vi Diagram *
File Edit Operate Tools Browse Window Help
13pt Application Font
device
1
Thermocouple
I/O
AI ONE PT
Temperature (TC)
DBL
100.00
TempSensor
I/O
AI ONE PT
Waveform Chart
DBL
Temperature(Sensor)
1000
Power
TF

e) Data Acquisition with GPIB Instruments

The object of this exercise is to use two GPIB instruments, a Digital Multi Meter (DMM) and a Power Supply (PS), for acquiring data to observe Ohm's Law. We will measure the current over a resistor for a set of voltages and plot the current versus voltage.

The GPIB board (PCI-GPIB) is already plugged in the PC and the instruments are connected to it via GPIB cables. The software interface is the program NI-488.2 and the application program is LabView.

We will develop the steps of this exercise during the course since the instruments were not available during production of the course material and this course manual. Other than the GPIB functions, which are in Functions/Instrument I/O, we are familiar with all the functions and the techniques needed.

V. LABVIEW TOOLS

a) Debugging Tools

LabView provides means of debugging a program and detecting the errors that cause it not to run. If all is fine, the Run arrows at top-left corner of the Front Panel and the Block Diagram are clear (not shaded or broken). The program runs. If there is any error, these arrows are broken and shaded gray. If we click on one, an Error list window appears with all the information on the errors on the program. If we highlight an error in the list by clicking on it, the details are given in a sub-window. Show Error button at the bottom will show us the error on the Block Diagram.

Activity 9:
Open any of the programs you wrote during the course, and create an error (and see that the Run arrow is broken) by deleting a wire, for instance. Click on the broken Run arrow and study the Error list, see the details and show the error on the Block Diagram.

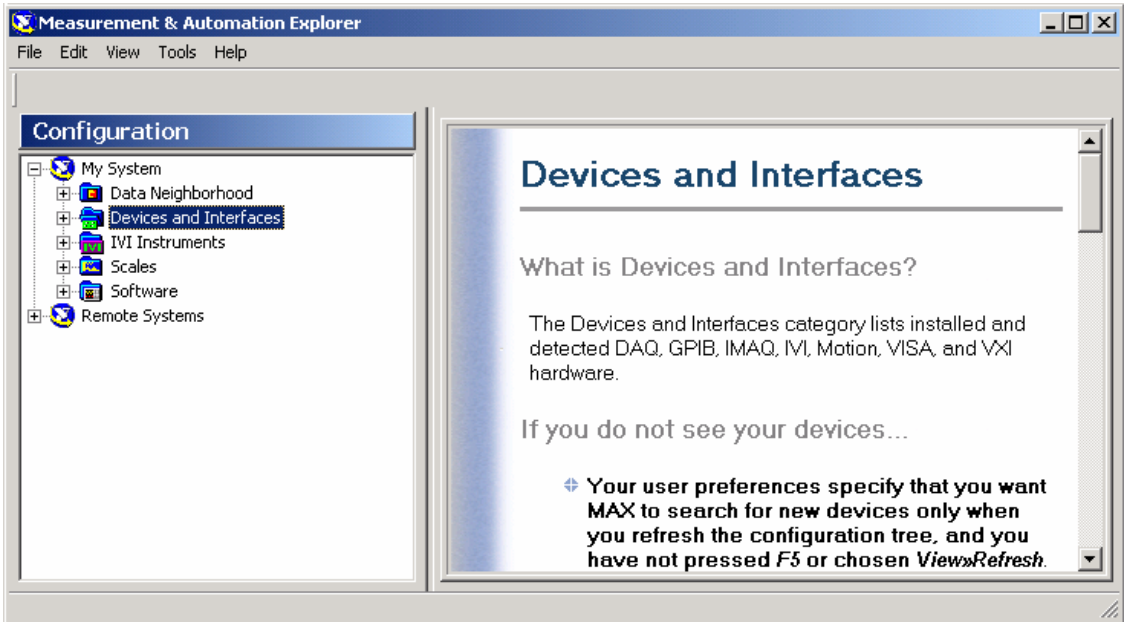
We can debug the program by running it step by step or by running a group of steps, stopping the program at the end of each step. We can display the inputs and outputs of each function in each step allowing us to investigate the results for consistency.

Step by step running is done by using the Run Control buttons, three buttons with right-angle arrows under the menu bar in Block Diagrams. If we bring the cursor on these buttons, the pop-up description windows explain what they do. We can display the outputs of each function by clicking on the Highlight Execution button (one with a light bulb). We can probe the value of data at any point of the program, for instance at an input of a function, by placing a probe on the wire at that location. This is done by choosing Probe Data on the Tool Palette and clicking on the location. A probe window appears where the value of the parameter is displayed.

Activity 10:
Open a program, click on the Run Control buttons, one at a time, to run the program. Click on the Highlight Execution button to display data at the outputs of functions. Use Probe Data tool for the same purpose.

b) Measurement and Automation Explorer

Measurement & Automation Explorer (MAX) provides access to all our National Instruments DAQ, GPIB, and other devices. With MAX, we can configure our National Instruments hardware and software, add new channels, interfaces, and virtual instruments, execute system diagnostics, and view the devices and instruments connected to our system. We can access to MAX from the Tools menu of a Front Panel or a Block Diagram. A typical MAX window looks as follow:

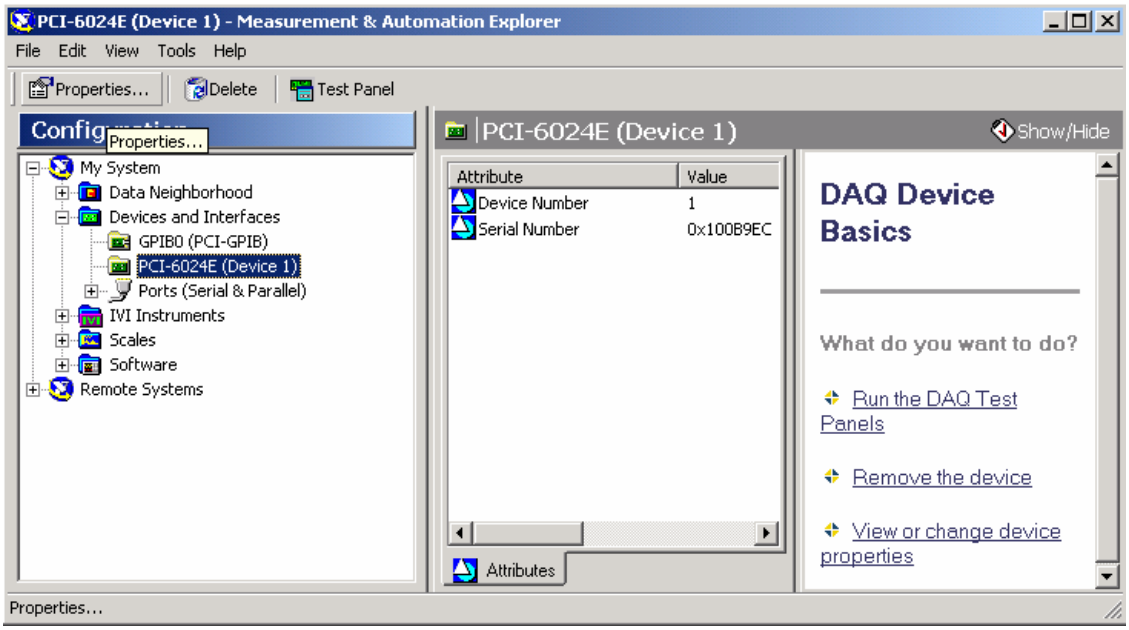


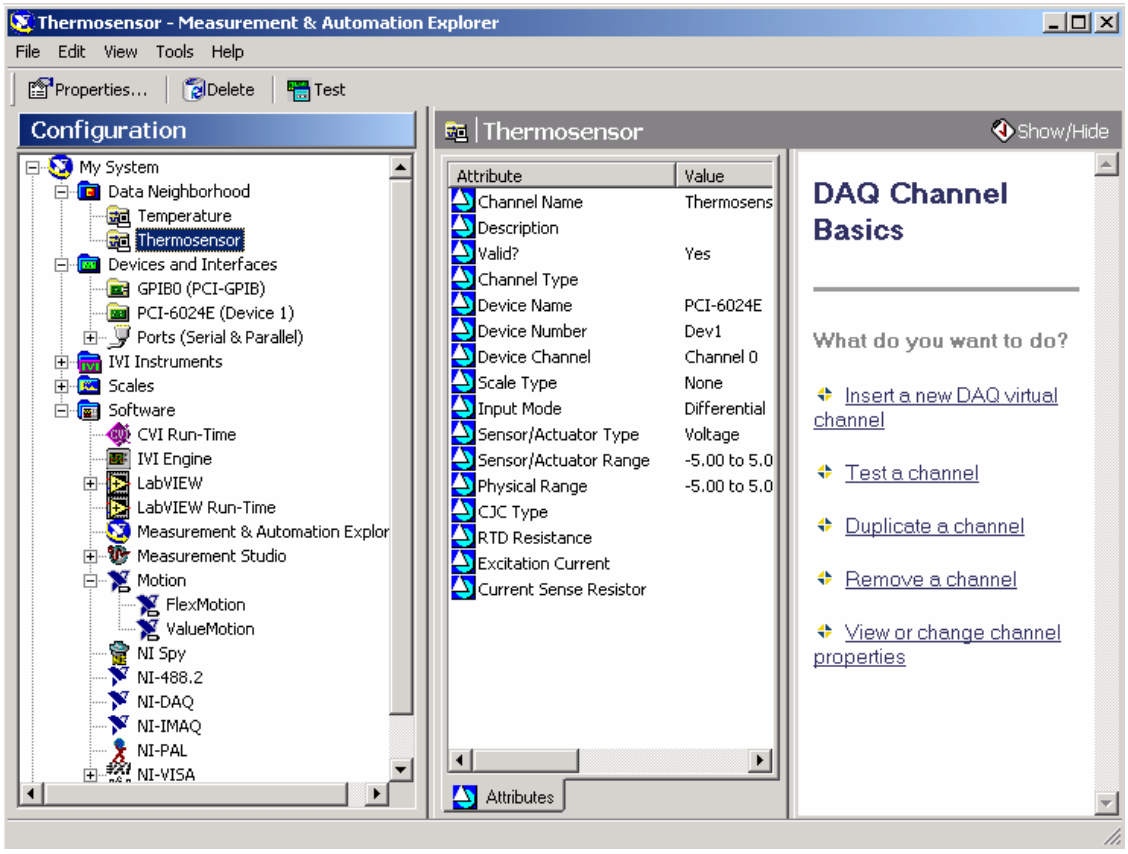
Left side is the Configuration Tree, which lists the items (hardware and software) in the system in our PC. Right side is the information/help window. If we choose, for instance Devices and Interfaces from the Configuration Tree and click on the + sign next to it we get a list of the devices in the system. By choosing a device from the list we see all the information, tools and help items related to that device on the information/help window. Right-clicking on them brings up a pop-up menu with various utility options.

Activity 11:
Start MAX, choose the DAQ device we have (PCI-6024E), study the menu items and options. Do the same for the GPIB device (PCI-GPIB) and study the instruments on this bus. Right-click on the instruments and study the pop-up menu. Study the Data

Neighborhood and Software items in the Configuration Tree. Do you recognize any of the names?

Below some of the windows you might be seeing:





c) LabView Examples

We can access to the examples provided with the LabView package by clicking on the Find Examples button on the LabView startup window or by choosing Find Examples on the Help menu of a Front Panel or a Block Diagram. In the dialog window that appears select Browse and click on the Task button. Double click on any folder in the middle section of the window, double click on any sub-folder to see the list of VIs for that category and sub-category. Open the VI's to study them. A typical window with one set of sub-folders and VIs in one sub-folder is shown below. The course CD contains some other examples in the folder Additional Examples.