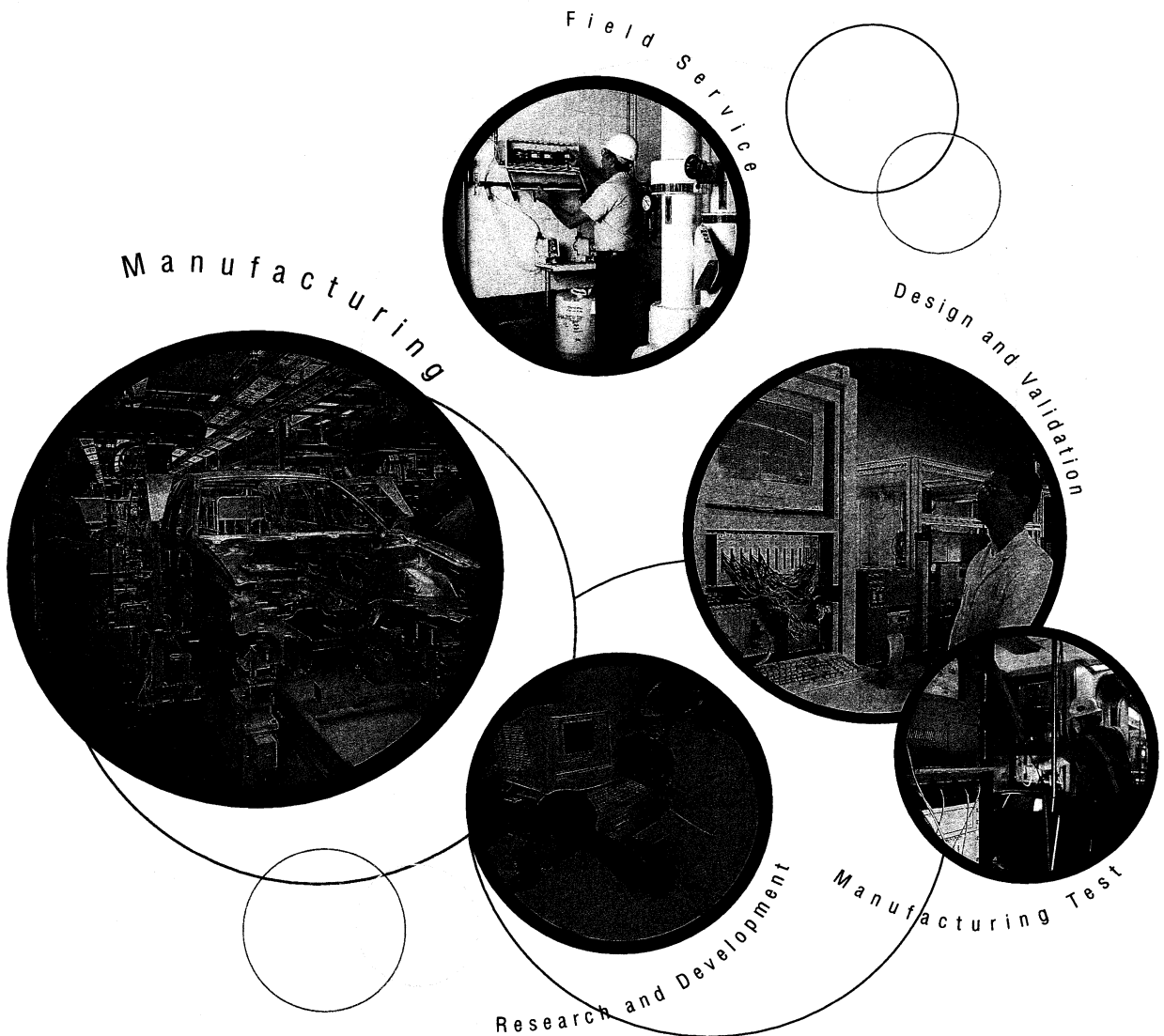




Proven Productivity

ni.com™



Networked

Measurement and Automation

LabVIEW™ – Proven Productivity

An In-Depth Look at Graphical Programming

September 2000 Edition
Part Number 350150F-01

© Copyright 2000 National Instruments Corporation.
All Rights Reserved.

Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

Trademarks

DataSocket™, LabVIEW™, Measure™, Measure Studio™, National Instruments™, ni.com™, PXI™, SCXI™, and VirtualBench™ are trademarks of National Instruments Corporation.

Product and company names mentioned herein are trademarks or trade names of their respective companies.

Worldwide Technical Support and Product Information

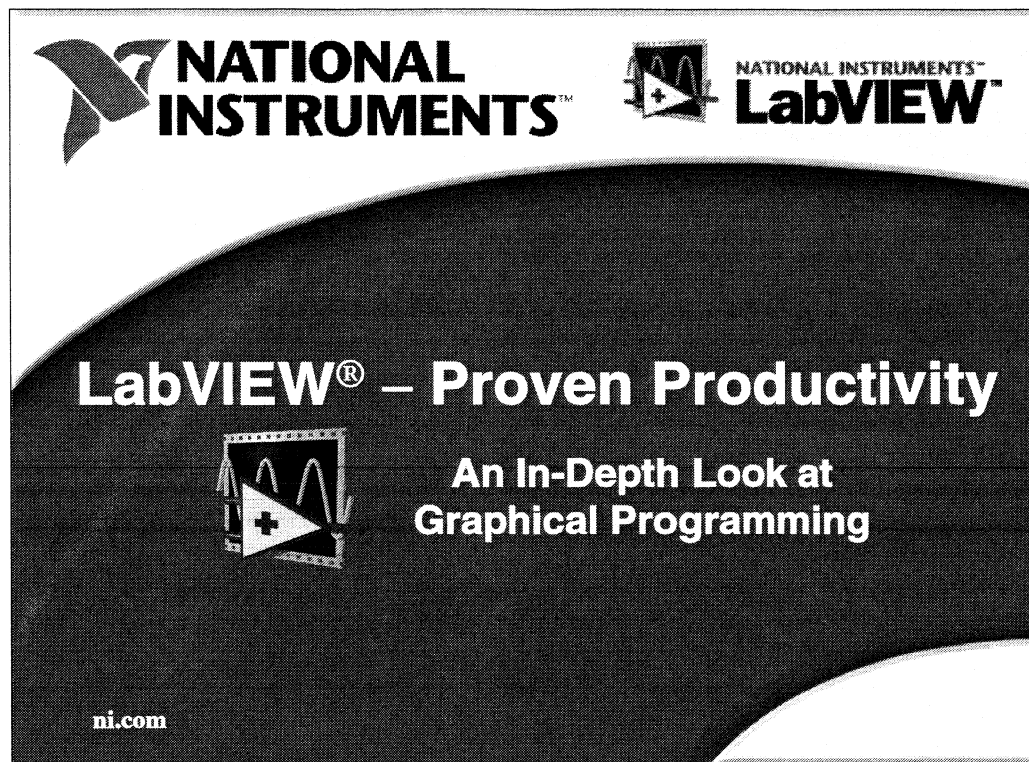
ni.com

National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 794 0100

Worldwide Offices

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 284 5011,
Canada (Calgary) 403 274 9391, Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521,
China 0755 3904939, Denmark 45 76 26 00, Finland 09 725 725 11, France 01 48 14 24 24,
Germany 089 741 31 30, Greece 30 1 42 96 427, Hong Kong 2645 3186, India 91805275406,
Israel 03 6120092, Italy 02 413091, Japan 03 5472 2970, Korea 02 596 7456, Mexico (D.F.) 5 280 7625,
Mexico (Monterrey) 8 357 7695, Netherlands 0348 433466, New Zealand 09 914 0488, Norway 32 27 73 00,
Poland 0 22 528 94 06, Portugal 351 1 726 9011, Singapore 2265886, Spain 91 640 0085, Sweden 08 587 895 00,
Switzerland 056 200 51 51, Taiwan 02 2528 7227, United Kingdom 01635 523545



Welcome to the LabVIEW Hands-On Seminar! This seminar is designed to give you an in-depth look at virtual instrumentation using graphical programming. We will discuss the concept of graphical programming and how you can use it to build powerful instrumentation and data acquisition systems. In a few moments, we will demonstrate the LabVIEW Development System to show you how easy it is to use and how powerful it can be for developing complete solutions quickly. In some exercises, you will build actual LabVIEW virtual instruments (VIs) that demonstrate the powerful and easy-to-use LabVIEW graphical programming paradigm. In other exercises, you will run completed LabVIEW VIs that demonstrate the flexibility and programming power available in LabVIEW.

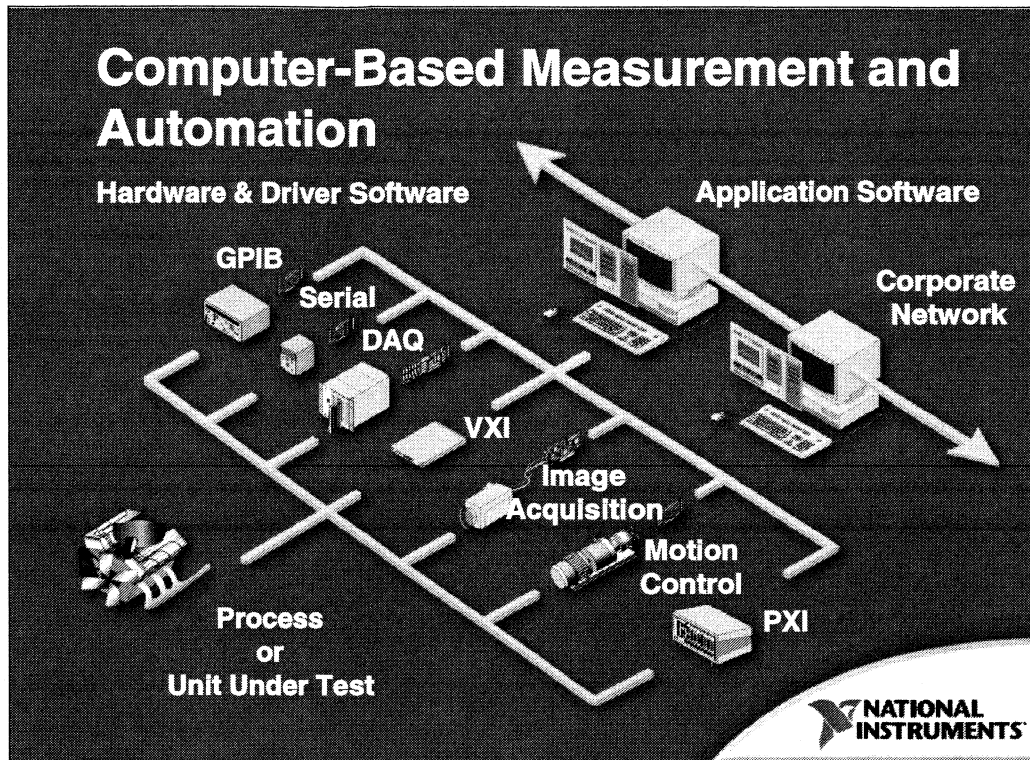
Agenda

- OverVIEW
- Data Acquisition
- Measurement Analysis
- Instrumentation
- Networking and the Internet
- Enterprise Integration
- The LabVIEW Platform
- Conclusion

ni.com



Here is a brief summary of some of the topics in LabVIEW we will focus on today.



National Instruments provides you with each of these types of computer-based measurement and automation hardware and software tools. You can take all of your measurements from computers using different combinations of I/O that are appropriate for your application, such as:

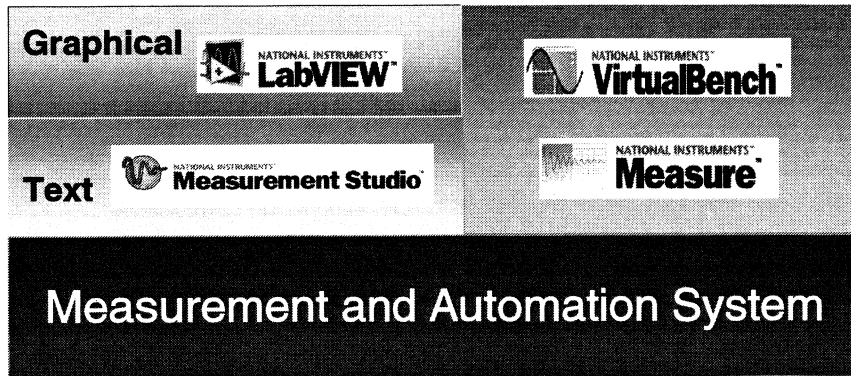
- Rack-n-stack GPIB instruments (NI makes the GPIB host interface)
- Serial instruments (NI makes RS-232 & RS-485 host interfaces)
- Data acquisition boards with and without signal conditioning
- VXI instruments (NI makes controllers, chassis, and DAQ boards)
- Image acquisition boards
- Motion control boards
- PXI/CompactPCI instruments (NI makes controllers, chassis, GPIB controllers, DAQ boards, and IMAQ boards)

The computers in a measurement and automation system are often networked together and today we will show you how LabVIEW allows you to quickly integrate networking and Internet technologies into your applications. LabVIEW excels at making you productive by allowing you to take measurements quickly and easily.

Virtual Instrumentation Software

Programming

Non-Programming

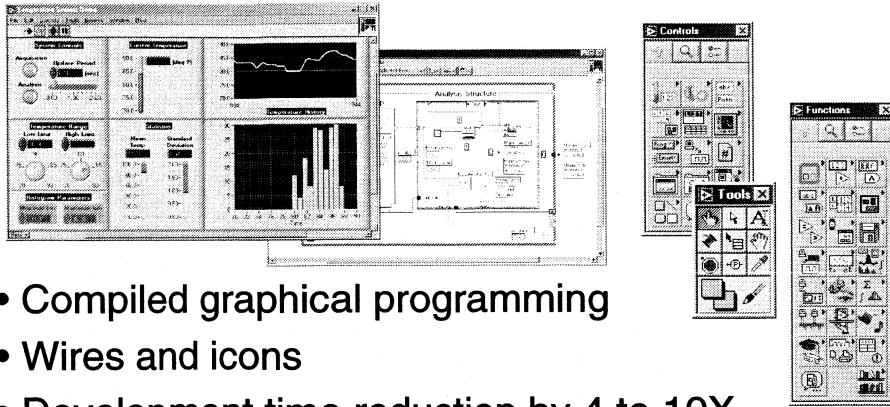


ni.com



LabVIEW is one member of our family of software tools and programming languages. It is a graphical programming language that lets you develop measurement and automation systems quickly and easily. If you prefer text-based programming, we have Measurement Studio, a complete ANSI C software development environment with measurement and automation libraries as well as tools for Visual Basic and Visual C++. VirtualBench and Measure are turnkey packages which do not require any programming. VirtualBench combines soft front panels with DAQ devices or computer-based instruments to deliver measurements comparable to those taken with benchtop instruments. Measure inserts measurements directly into cells of an Excel spreadsheet.

LabVIEW Programming



- Compiled graphical programming
- Wires and icons
- Development time reduction by 4 to 10X
- Full-fledged programming environment

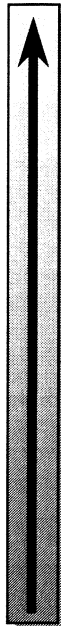
ni.com



LabVIEW is a compiled programming language just like other programming languages, such as Visual Basic or Visual C++. However, LabVIEW is a high-level language.

In text-based programming languages, you are as concerned about the code as you are about what you are trying to do. You must pay close attention to the syntax (commas, periods, semicolons, square brackets, curly brackets, round brackets, etc.) which may be quite tedious. LabVIEW is higher-level—it uses icons to represent subroutines, and you wire icons together to determine the flow of data through your program. It is similar to flow-charting your code as you are writing it. LabVIEW makes you productive because you can write your program in significantly less time than if you wrote it in a text-based programming language.

LabVIEW Product History

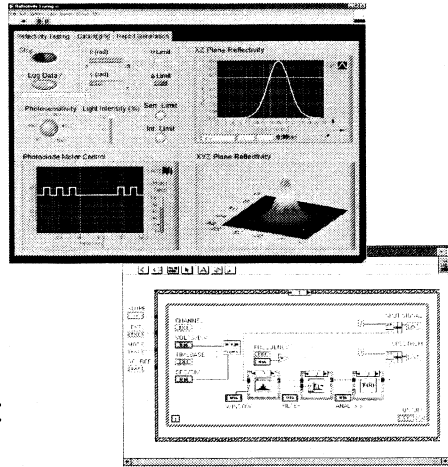


- August 2000** • LabVIEW 6/ Internet-ready measurement intelligence
- February 1999** • LabVIEW 5.1 3D graphs, Performance, Web tools
- March 1998** • LabVIEW 5.0 ActiveX, Multithreading, Undo
- June 1997** • LabVIEW 4.1 DAQ Wizards on Windows Platforms
- March 1996** • LabVIEW 4.0
- December 1994** • LabVIEW 3.1 Added HP-UX and Power Mac platforms
- August 1993** • LabVIEW 3.0 Multiplatform version of LabVIEW
- October 1992** • LabVIEW for Sun
- September 1992** • LabVIEW for Windows
- January 1990** • LabVIEW 2.0 for Macintosh
- October 1986** • LabVIEW 1.0 for Macintosh
- April 1983** • LabVIEW project started at National Instruments

LabVIEW 1.0 was first introduced on the Macintosh in October of 1986. Since then, we have worked hard to improve LabVIEW so that you can develop measurement and automation systems quicker and easier than ever before. Today, LabVIEW is the de facto standard for test and measurement programming languages. It has all the breadth and depth of a conventional programming language but it is fun to use and makes you highly productive by decreasing the time required to develop applications.

Programming Paradigm

- Front Panel
 - Graphical user interface
 - Controls and indicators
- Block Diagram
 - Source code window
 - Numerous functions
 - Rapid code development
 - Self-documenting



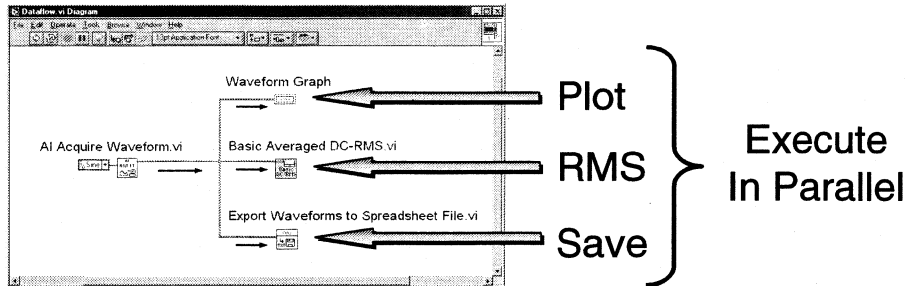
ni.com



LabVIEW programs are called “Virtual Instruments” or “VIs” for short. They are modeled after traditional benchtop instruments which generally possess a panel for inputs and outputs and user interaction and behind the panel is the instrument’s hardware.

LabVIEW Virtual Instruments are similar—they have a Front Panel which serves as the graphical user interface containing program inputs and outputs, and a graphical source code area called the Block Diagram. Program inputs on the Front Panel are called controls and indicators are program outputs. Every object on the Front Panel has an associated terminal on the Block Diagram that allows you to perform various operations and analysis on the data it contains or displays.

Dataflow Programming



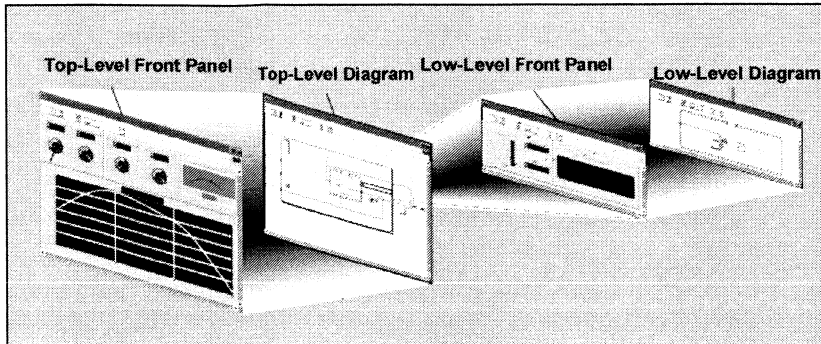
- Wires pass data (nonlinear)
- Data flows from sources to sinks
- Code can execute multiple operations in parallel

ni.com



LabVIEW is technically a dataflow programming language. This means that program data flows from a data source to one or more sinks and it propagates through the program in this fashion. Because of dataflow, LabVIEW is not linear like text-based languages and can execute multiple operations in parallel. For example, as you can see in this picture, we are plotting a waveform, calculating the RMS value, and saving the waveform to disk all in parallel.

Hierarchy of VIs



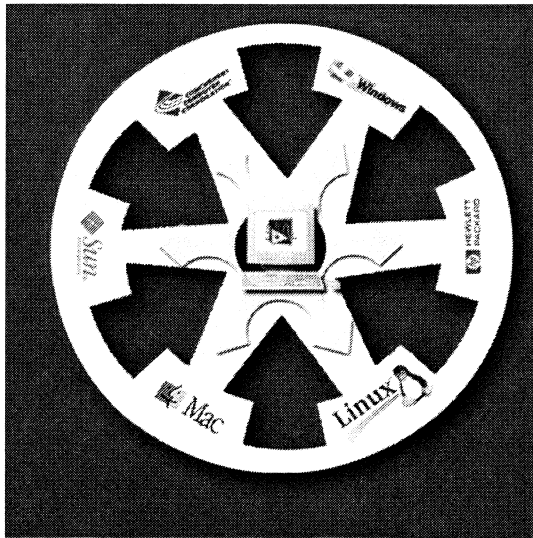
- Modular design
- Reusable building blocks
- Hierarchical system

ni.com



LabVIEW lends itself well to modular programming techniques—you can write VIs and use them as subroutines in a higher-level program. VIs used inside of other VIs are called subVIs.

Multiplatform Compatibility

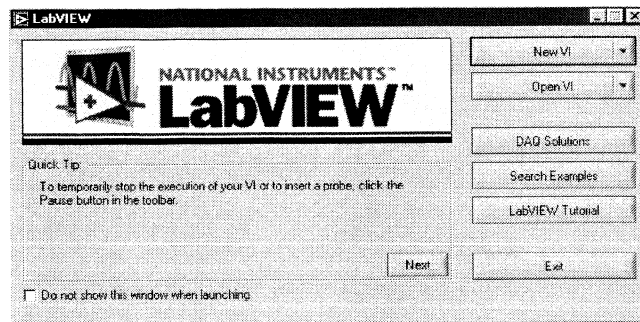


Migrate applications transparently between platforms



LabVIEW is a multiplatform programming language which means you can take the LabVIEW source code you have written on one platform and load it on any of the other supported platforms and it will run without any modifications. The only exception to this rule is if your LabVIEW code contains operating system-specific calls that will only execute on a given platform. An example of this is ActiveX Automation which is specific to the Windows operating systems.

Explore LabVIEW



ni.com



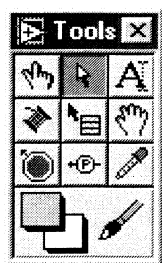
Now, let's take a look at LabVIEW. To launch it, double-click the LabVIEW shortcut on your desktop.










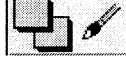
Exercise 1 – Create a simple LabVIEW VI

In this exercise, you will create a simple LabVIEW VI that tests an input value against a specified limit and lights an LED if the input value exceeds that limit. You will also create a subVI from this simple limit test which will allow you to use it in other VIs.

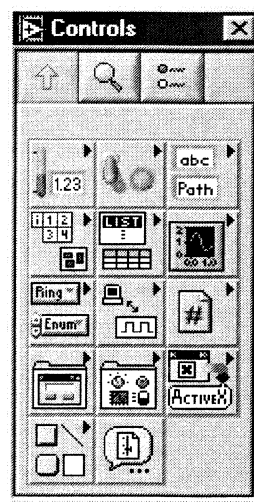
Below, you will find a list identifying each of the various tools found in the LabVIEW Tools palette to assist you as you complete these exercises. Also named are the controls and functions in the Controls and Functions palettes. For an explanation of these tools and palette contents, as well as other facets of the LabVIEW development environment, please consult the Appendix.















Tools Palette



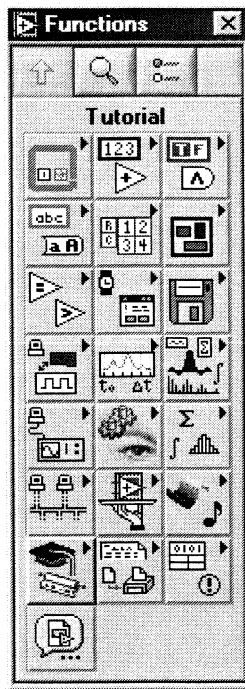
-  **Operating tool**
-  **Positioning tool**
-  **Labeling tool**
-  **Wiring tool**
-  **Object shortcut menu tool**
-  **Scrolling tool**
-  **Breakpoint tool**
-  **Probe tool**
-  **Color Copy tool**
-  **Coloring tool**























Controls Palette



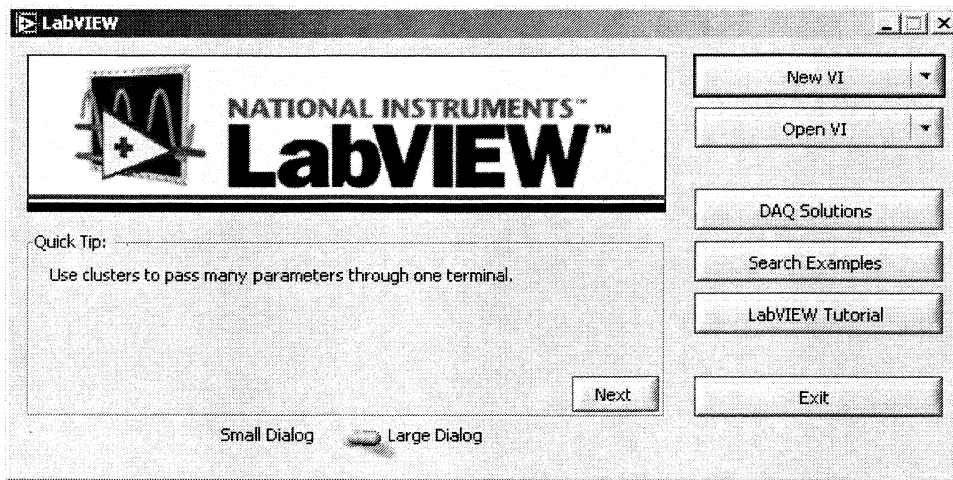
- | | | | |
|---|----------------------|---|----------------------------|
|  | Numeric |  | I/O |
|  | Boolean |  | Refnum |
|  | String/Path |  | Dialog Controls |
|  | Array/Cluster |  | Classic Controls |
|  | List/Table |  | ActiveX |
|  | Graph |  | Decorations |
|  | Ring/Enum |  | Select a Control... |

Functions Palette

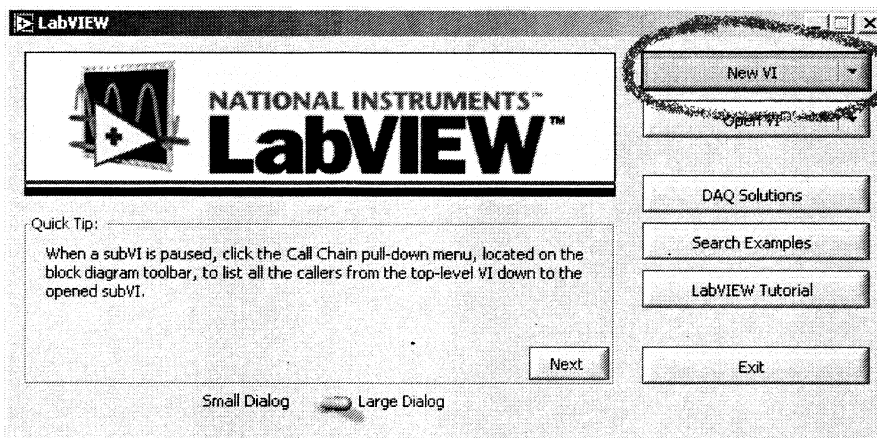


	Structures		Analyze
	Numeric		Instrument I/O
	Boolean		Motion and Vision
	String		Mathematics
	Array		Communication
	Cluster		Application Control
	Comparison		Graphics & Sound
	Time/Dialog		Tutorial
	File I/O		Report Generation
	Data Acquisition		Advanced
	Waveform		Select a VI...

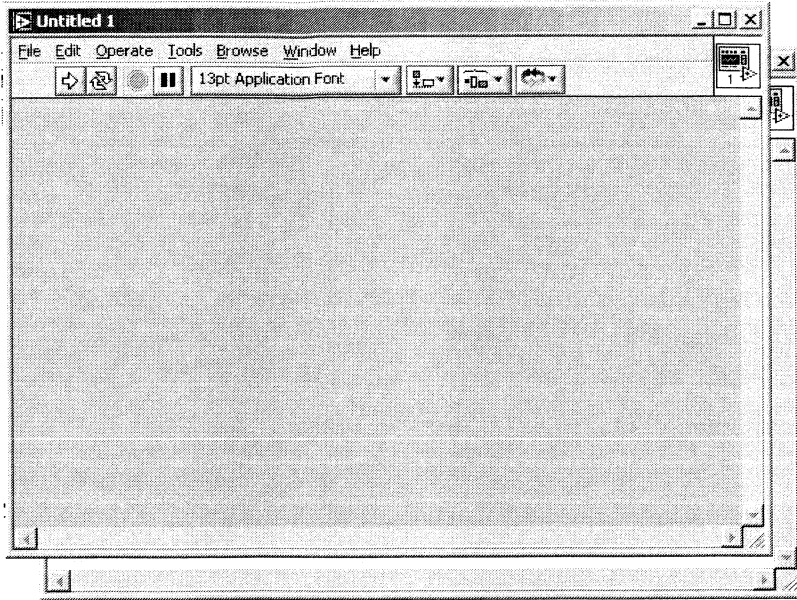
1. If you have not already done so, double-click on the **LabVIEW icon** on your desktop. Once you have launched LabVIEW, you will see a splash screen like the one pictured below:



2. Click **New VI...** in the upper right corner of the splash screen.

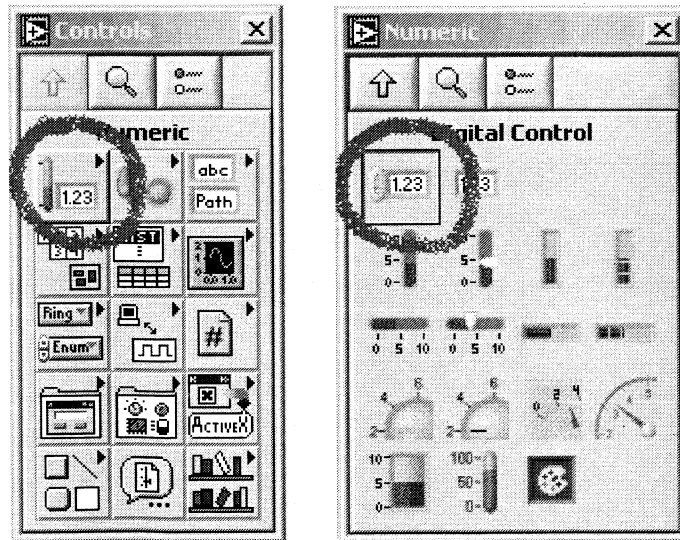


You will see **two new windows** appear as shown below:

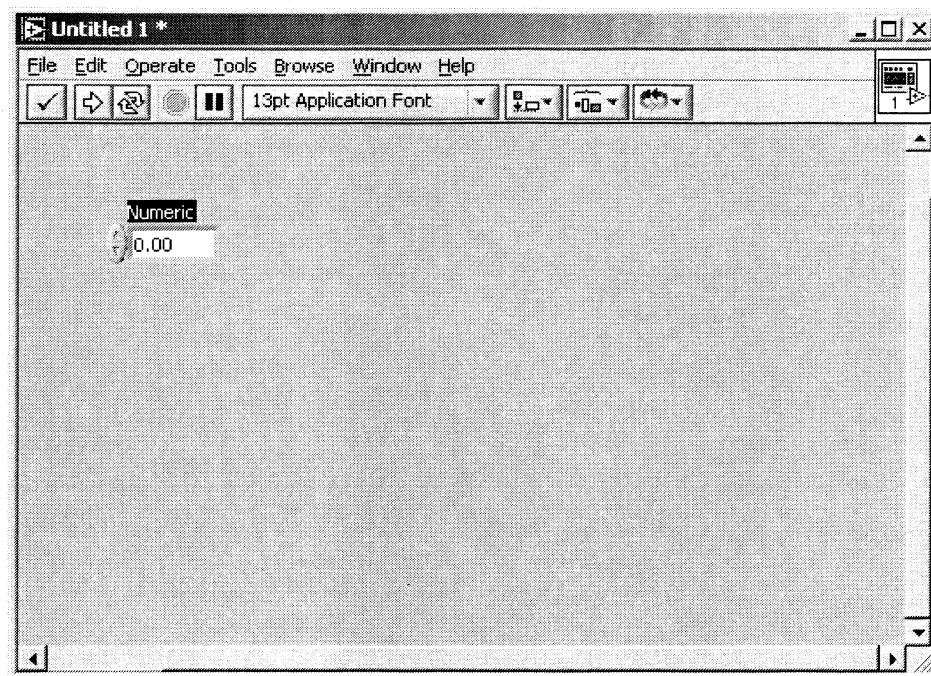


The gray window is the **Front Panel** and the white one is the **Block Diagram**.

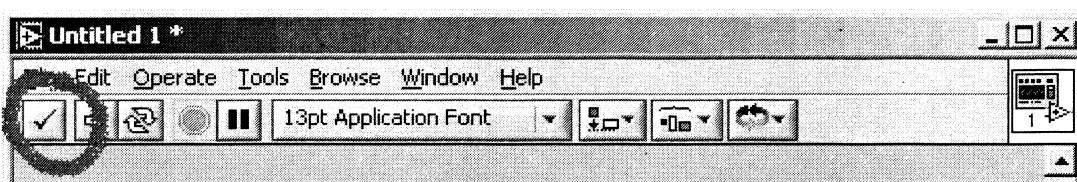
3. Place a **Digital Control** on the Front Panel by going to the **Controls** palette and clicking on the **Numeric** subpalette. In the Numeric subpalette, click on the **Digital Control** and drag it to the Front Panel and click once to drop it.



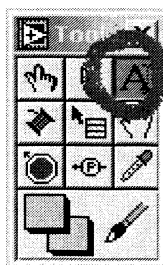
Your Front Panel should look like:



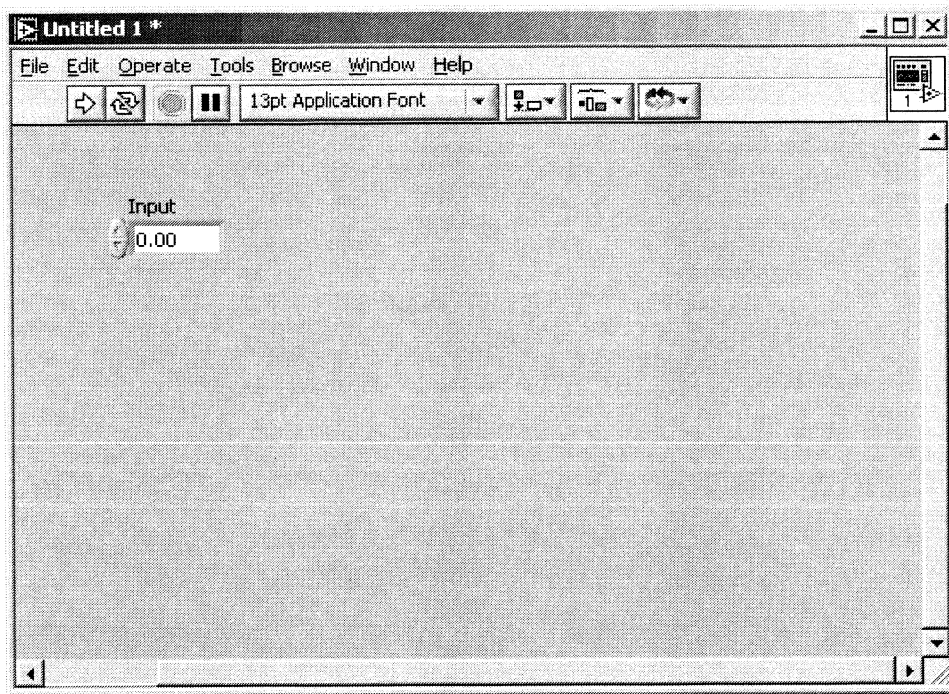
To rename the Digital Control from Numeric to Input, simply type in the word **Input** on the keyboard after you have placed down the Digital Control and **click** anywhere to complete the change. This will rename the Digital Control to **Input**. Alternatively, you can use the **Labeling** tool to rename controls by highlighting the existing control name and entering the new name from the keyboard. To save the name change, simply **click** anywhere outside of the label once or click on the **check mark** button in the toolbar as shown below:



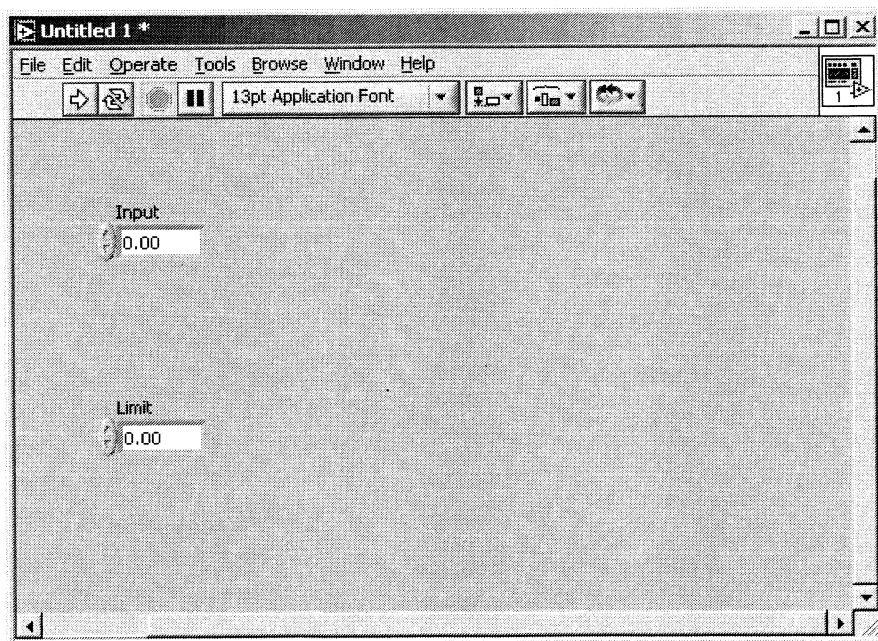
You can access the **Labeling** tool from the **Tools** palette:



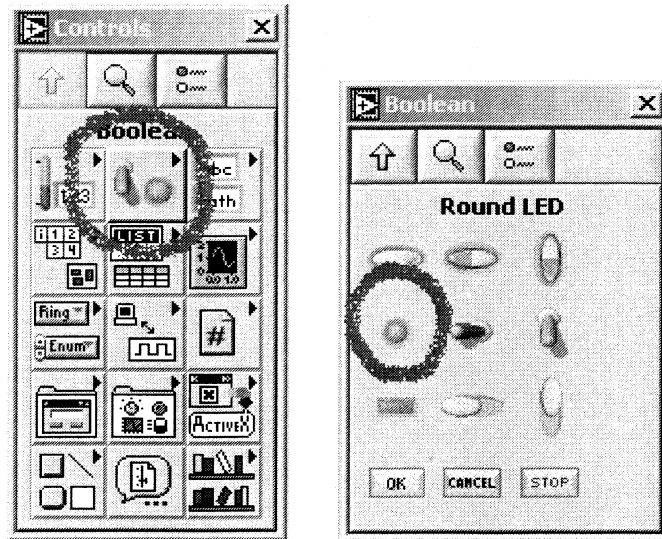
Your **Front Panel** should now look like the one below:



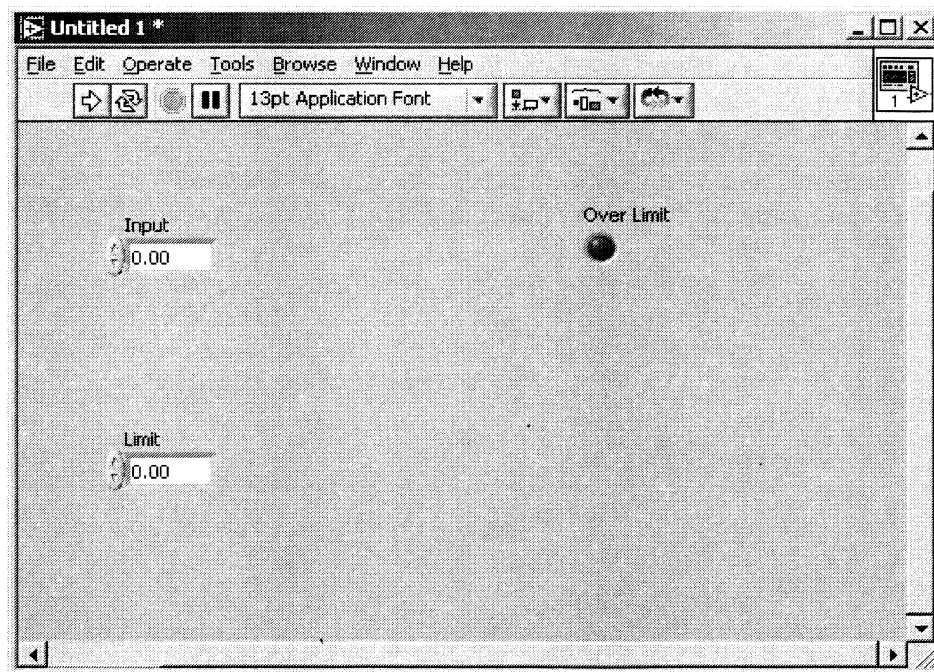
4. Press the **up arrow** on the Controls palette to return from the Numeric subpalette to the main Controls palette.
5. Repeat the same procedures as in the previous step to place another **Digital Control** on the Front Panel named **Limit**. Your Front Panel should look like:



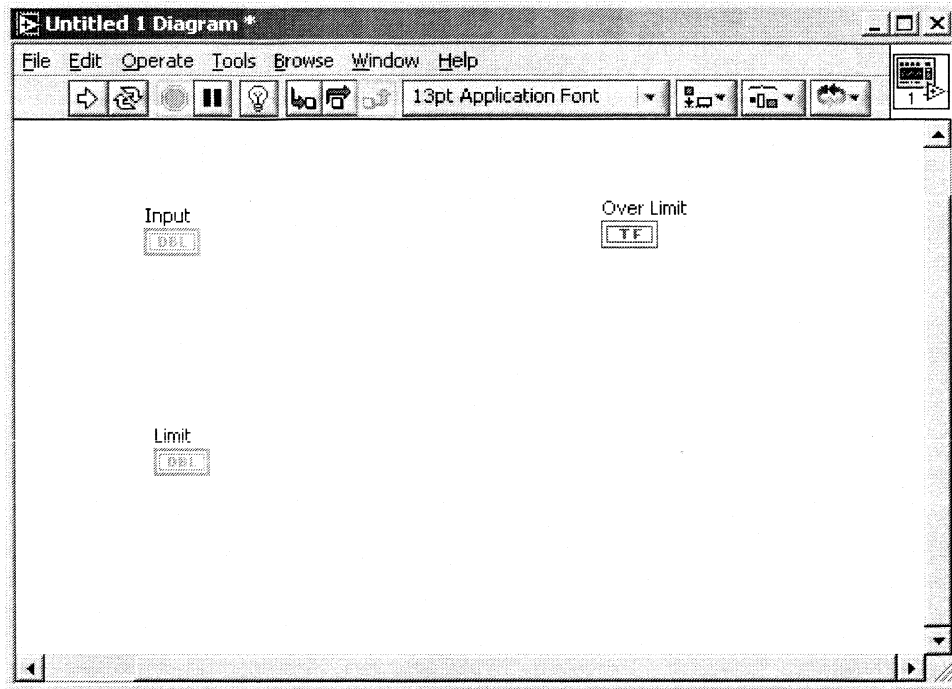
- Click on the **Boolean** subpalette of the Controls palette and place a **Round LED** on the Front Panel. Rename it to **Over Limit**.



Your VI should look like this now:

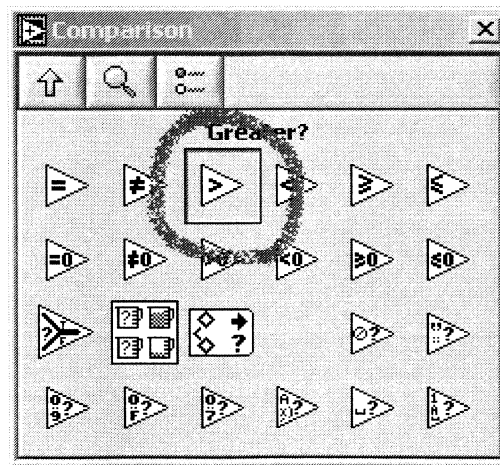
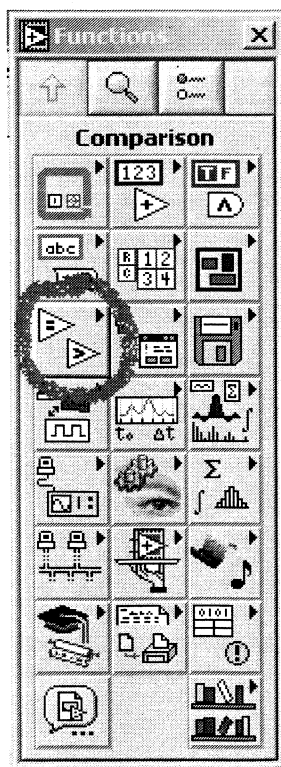


7. Switch to the **Block Diagram** by pressing Ctrl+E. It should already look like this:

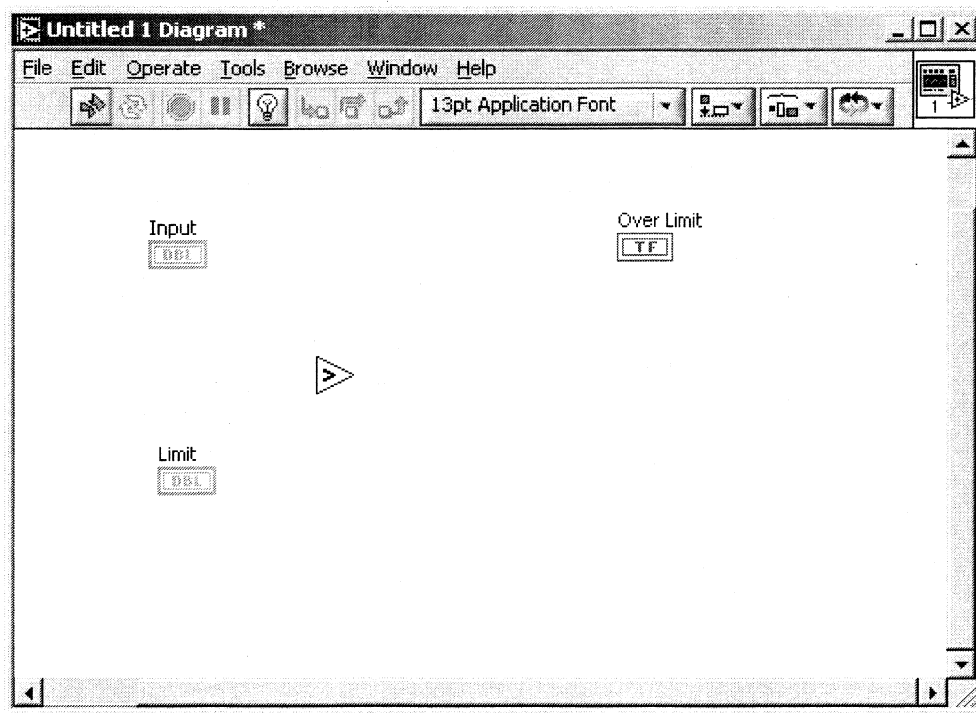


To place **functions** on the **Block Diagram**, you use the **Functions** palette in exactly the same manner as the Controls palette.

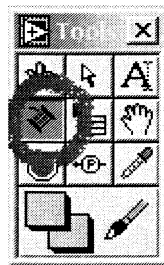
8. Place a **Greater?** function on the **Block Diagram** from the **Comparison** subpalette of the **Functions** palette. Be sure to hit the up arrow to return to the main Functions palette from the Comparison subpalette.



Your Block Diagram should look like:

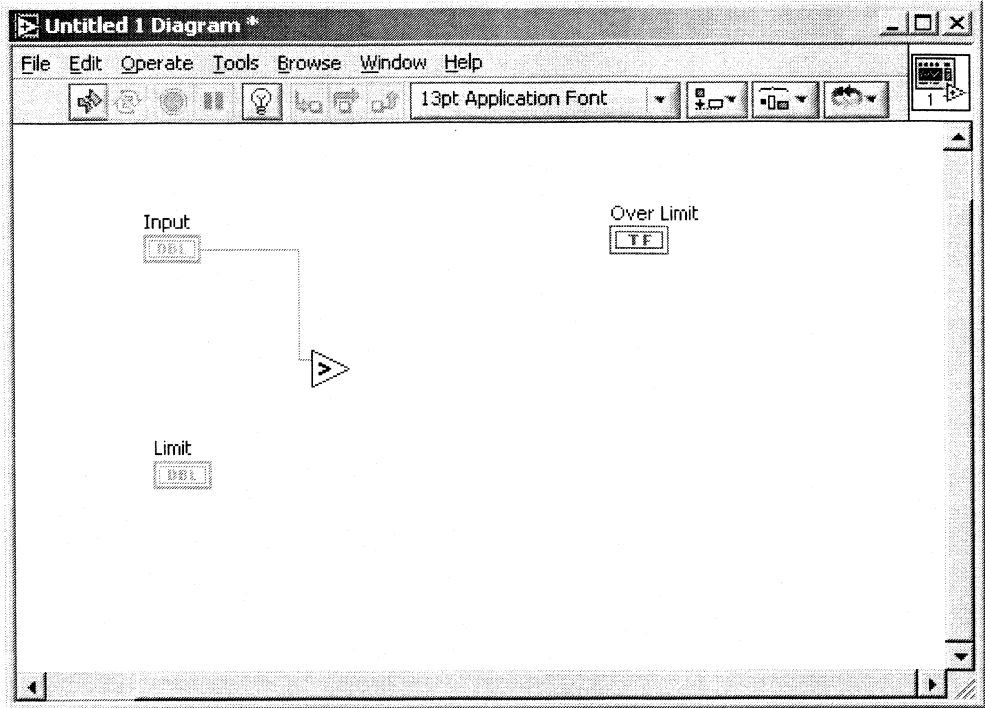


9. Wire the **Input** and **Limit** control terminals to the input terminals on the **Greater?** function using the **Wiring** tool. The Wiring tool can be found in the **Tools** palette:

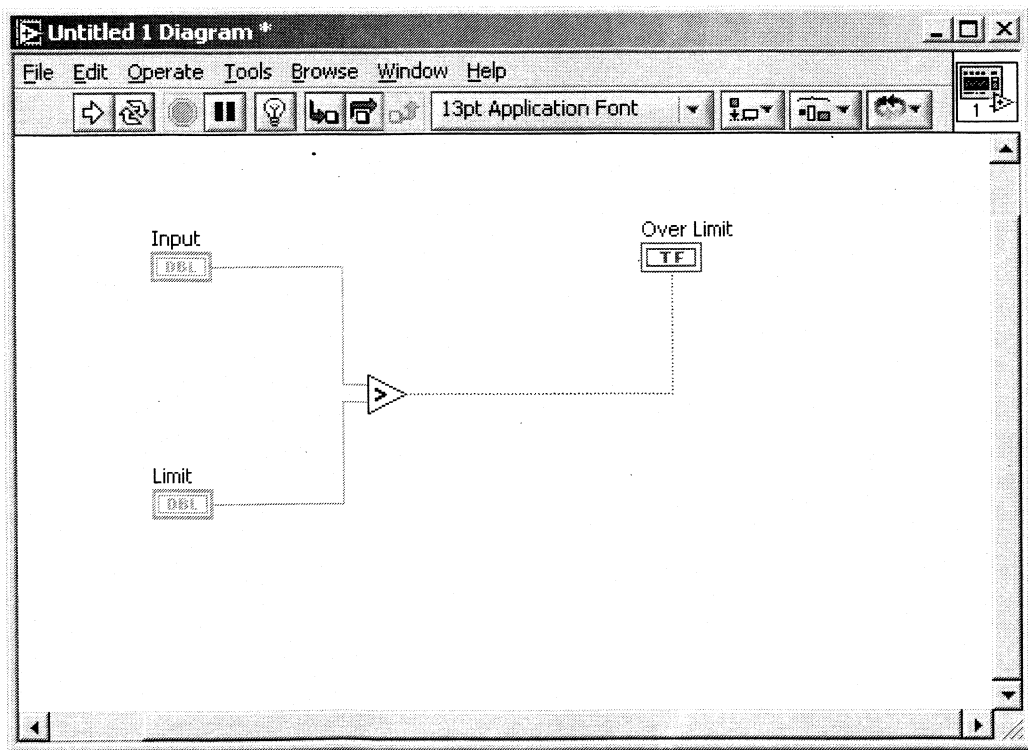


To wire the **Input** control terminal to the upper input on the **Greater?** function, place the **Wiring** tool over the orange **Input** control terminal and it will blink. **Click** once on the **Input** control terminal to anchor the beginning of the wire. **Drag** the **Wiring** tool close to the **Greater?** function and as you get close to it, its input and outputs will appear as small wire stubs. **Hover** with the **Wiring** tool over the upper **x** input on the **Greater?** function and the portion of the **Greater?** function attached to the **x** input will **blink** and turn **black**. To complete the wiring from the orange **Input** control terminal to the **x** input on the **Greater?** function, **click** once more. The wire segment that used to be dotted should now be **solid orange**.

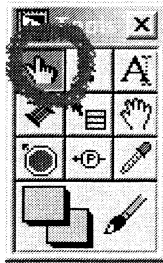
Your Block Diagram should look like this:



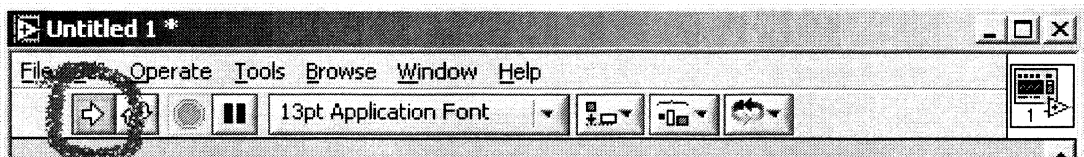
- Using the same wiring procedure as in the previous step, wire the orange **Limit** control terminal to the **y** input on the **Greater?** function and wire the **Greater?** output to the green **Over Limit** terminal. The complete Block Diagram should look like this:



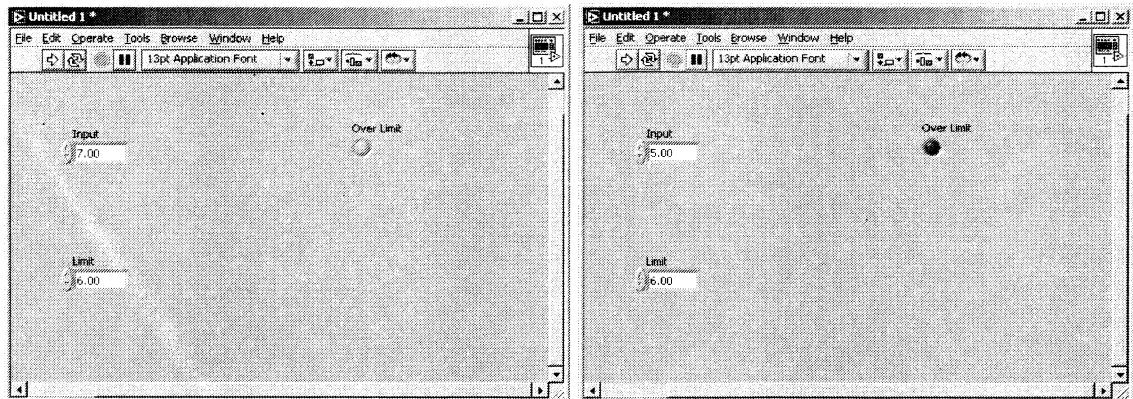
11. Go back to the **Front Panel** by pressing **Ctrl+E** and click on the **Operate** tool in the Tools palette:



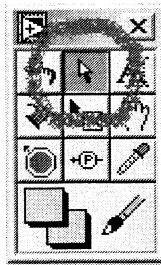
12. Change the numeric values inside the **Input** and **Limit** digital controls by either **clicking** once with the **Operate** tool inside them and entering new values or using the **Operate** tool to increment or decrement the current value using the **up and down arrows** next to each control. This VI compares the value of the **Input** control to the value of the **Limit** control. If the value of **Input** is **greater** than the value of **Limit**, the round LED labeled **Over Limit** will light. Otherwise, the Over Limit LED will not be lit. To **run** the VI and test whether the value of Input is over the limit, click on the **run arrow** in the toolbar.



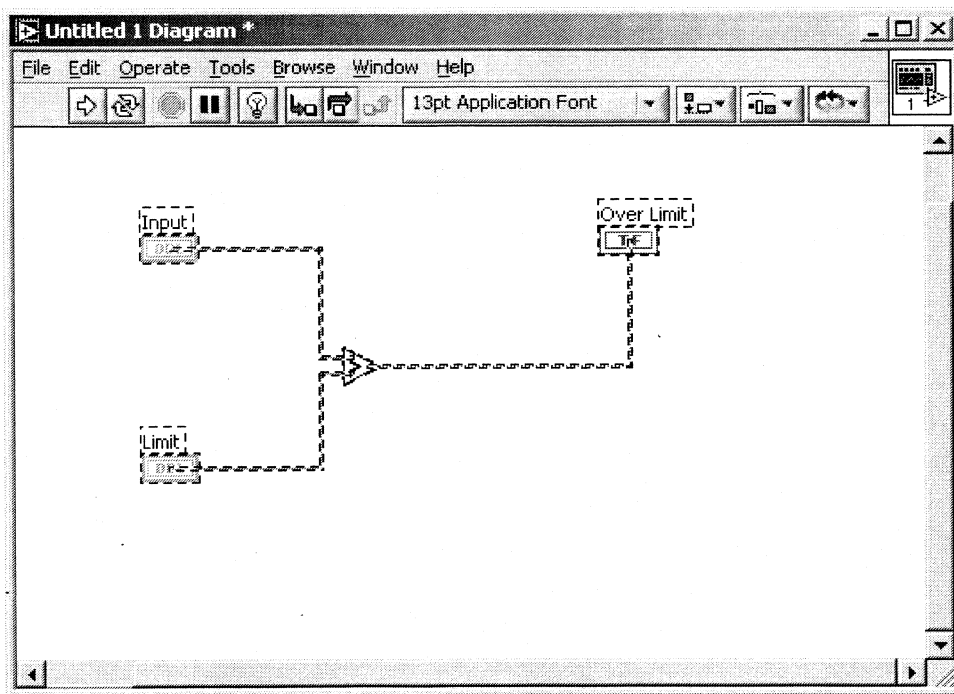
The two Front Panels below show both Input being over the limit and Input not being over the limit:



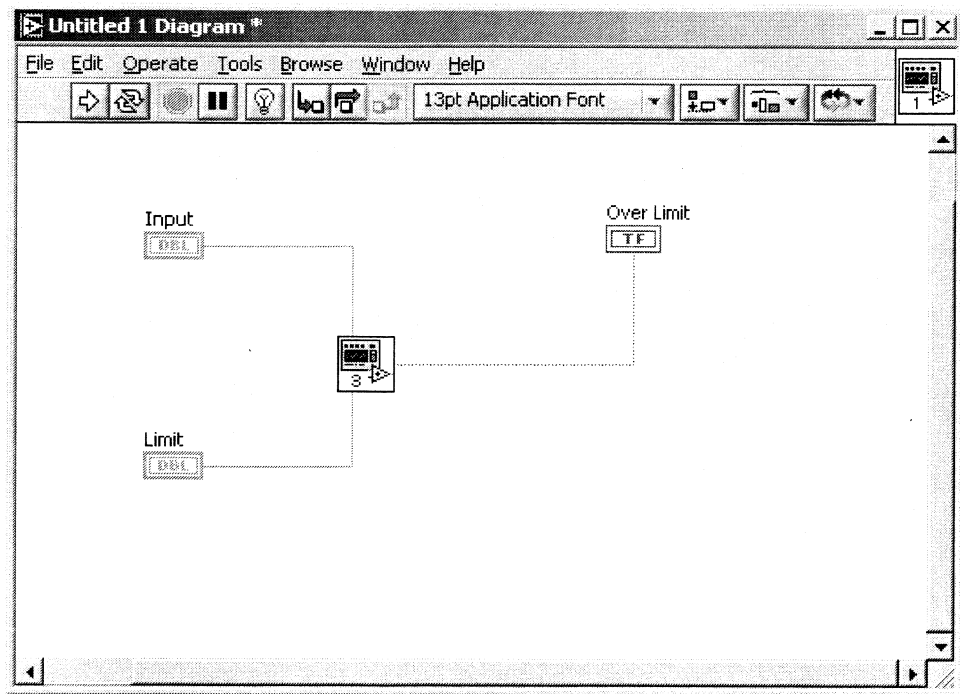
13. Go back to the **Block Diagram** by pressing **Ctrl+E** and then change to the **Selection** tool in the Tools palette.



14. **Click** the mouse button once and hold it down and **draw a box** around all of the objects on the Block Diagram and release the mouse button. All of the Block Diagram objects should now be highlighted:



15. From the **Edit** menu, choose **Create SubVI**. Your Block Diagram will then look like this:



What you have done is create a **subVI** from your limit comparison code. A subVI is simply a **VI** that you can use **inside** of **other VIs**. Each subVI in LabVIEW is represented by an **icon** when it is placed on the **Block Diagram** of another VI. SubVIs are powerful because they allow you to integrate **hierarchy** into your applications and allow you to **share code** components among different LabVIEW applications.

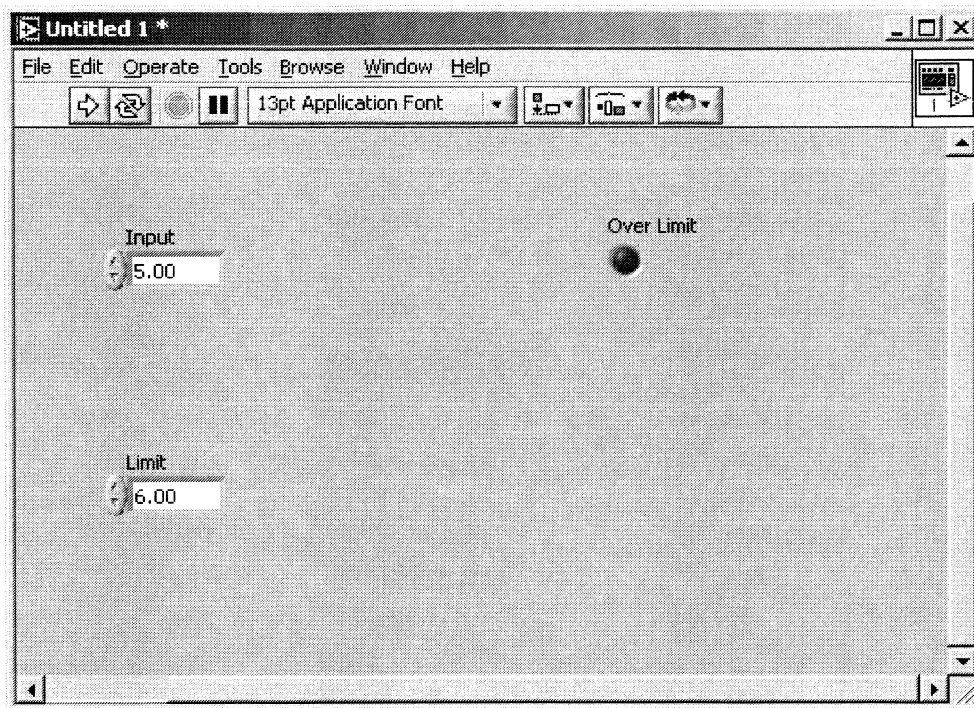
16. You do not need to save this VI that you have just created, but **leave it open** because you will use it in later exercises.

(OPTIONAL)

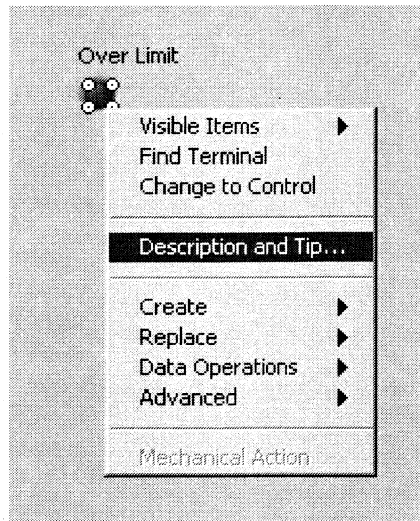
Exercise 2 – *Add a description and a tip*

In this exercise, you will build on the VI you previously created in Exercise 1 by adding a description and a tip strip to the LED on the Front Panel.

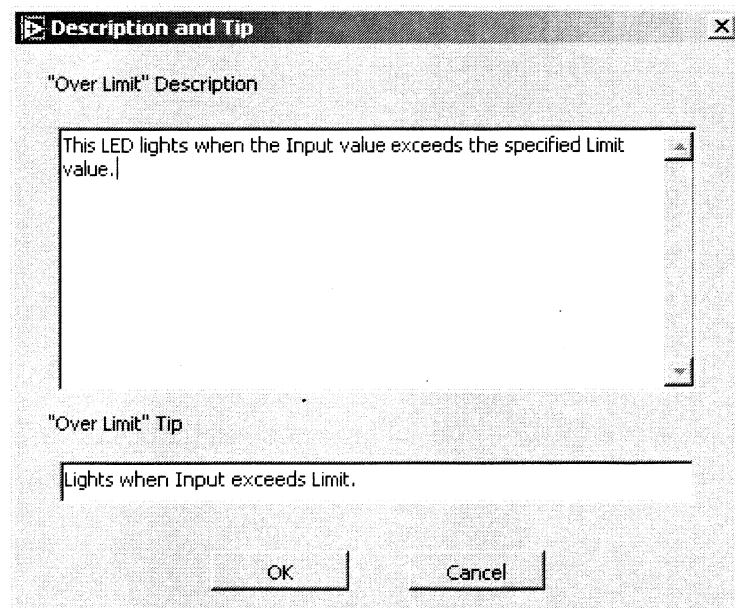
1. Go back to the **Front Panel** of the VI you created in **Exercise 1** which looks like this:



2. **Right click** on the **Over Limit** LED and choose **Description and Tip...** from the shortcut menu that appears.

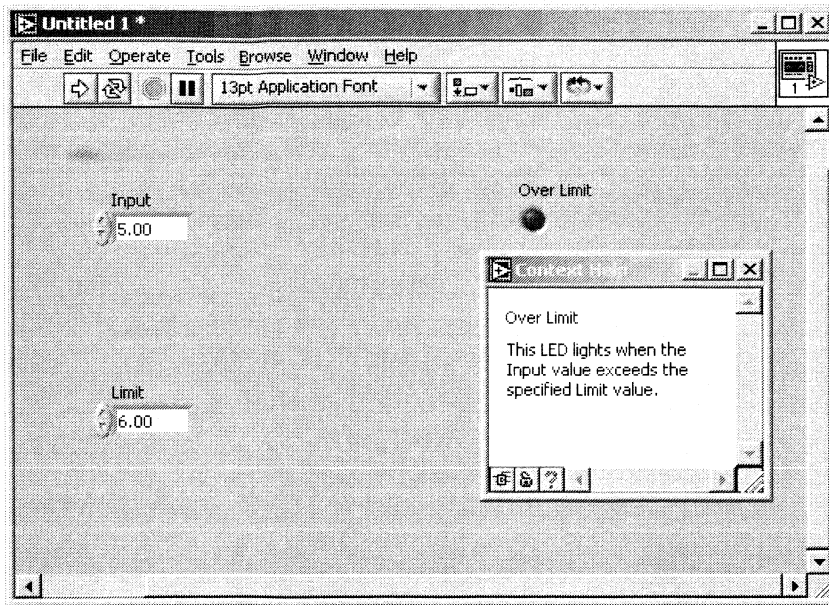


3. The window that appears allows you to give a **description** to the LED as well as define a **tip strip** that appears when the **Operate** tool hovers over it. Enter a description for the LED and a tip in the **Description and Tip** window. An example is provided below:

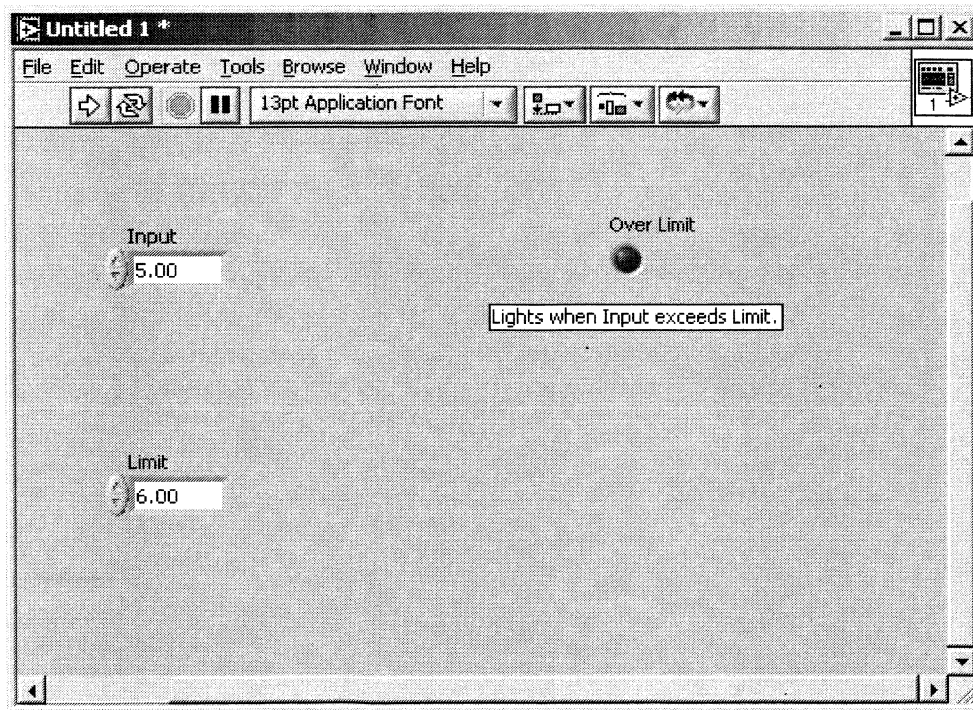


Click **OK** when you have finished entering the description and tip.

4. Press **Ctrl+H** to display the **Context Help** window. When you move the **cursor** over the **Over Limit** LED, you will see the contents of the **Context Help** window change to display the **name** of the LED as well as the **description** of the LED that you just entered.



- Using the **Tools** palette, change to the **Operate** tool and **hover** over the **Over Limit** LED. You will see a yellow **tip strip** appear which displays the tip that you entered earlier in the **Description and Tip** window.

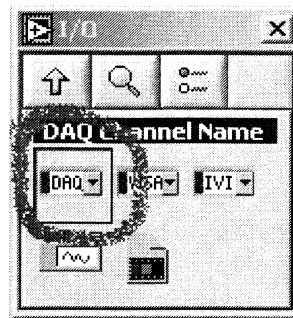
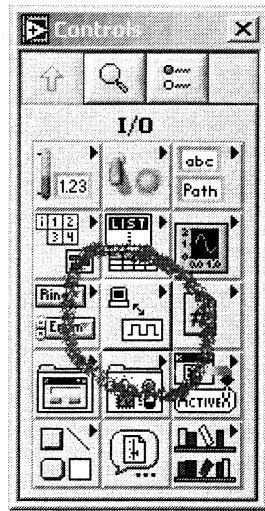


- Once again, leave this VI **open**, as you will need it in the **next** exercise.

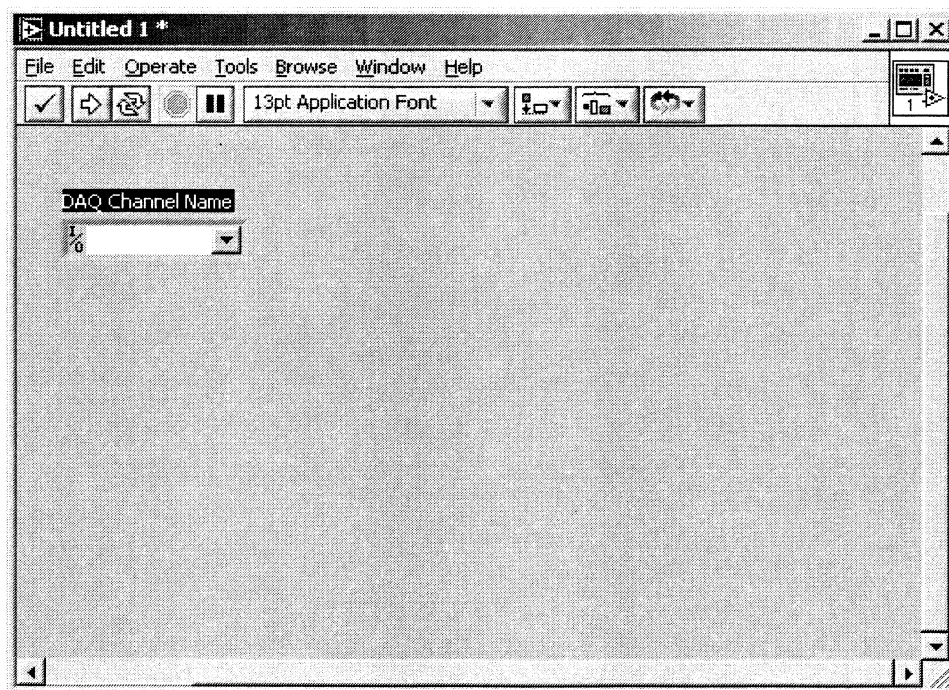
Exercise 3 – *Take temperature measurements*

In this exercise, you will create a new VI and take temperature readings from a DAQ channel you create and configure.

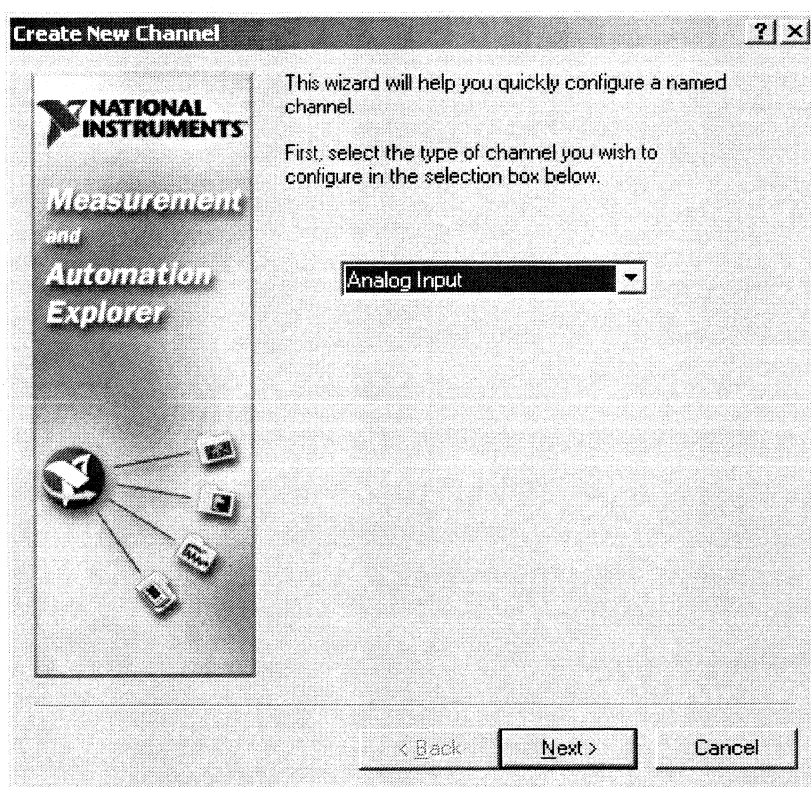
1. Create a **new VI** by pressing **Ctrl+N**.
2. Place a **DAQ Channel Name** control from the **I/O** subpalette of the Controls palette on the Front Panel. Press the up arrow on the controls palette to return from the **I/O** subpalette to the main controls palette.



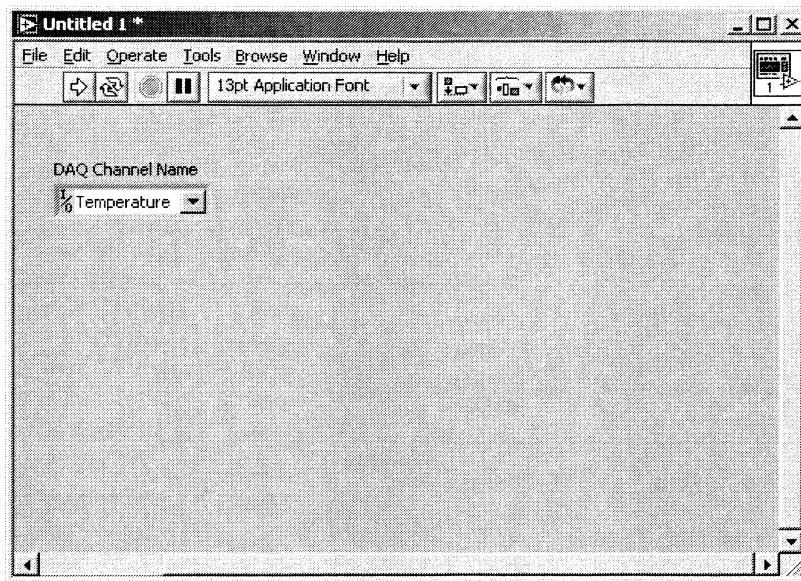
Your front panel should now look like the one below:



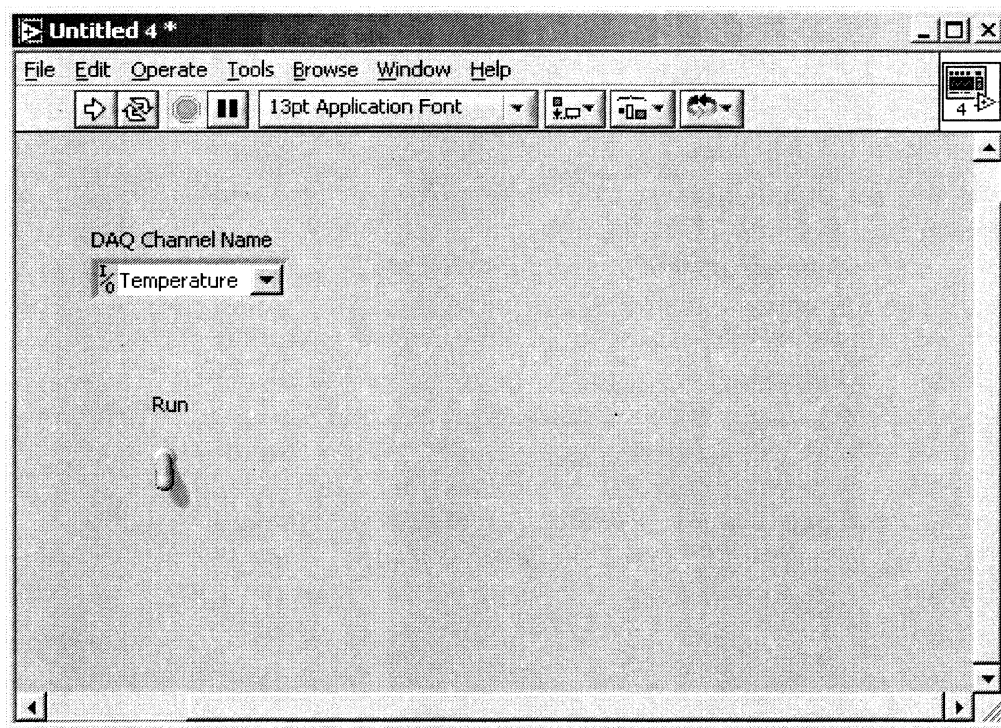
3. **Right click** on the DAQ Channel Name control and choose **New DAQ Channel...** from the shortcut menu. This will launch a new window titled **Create New Channel**.



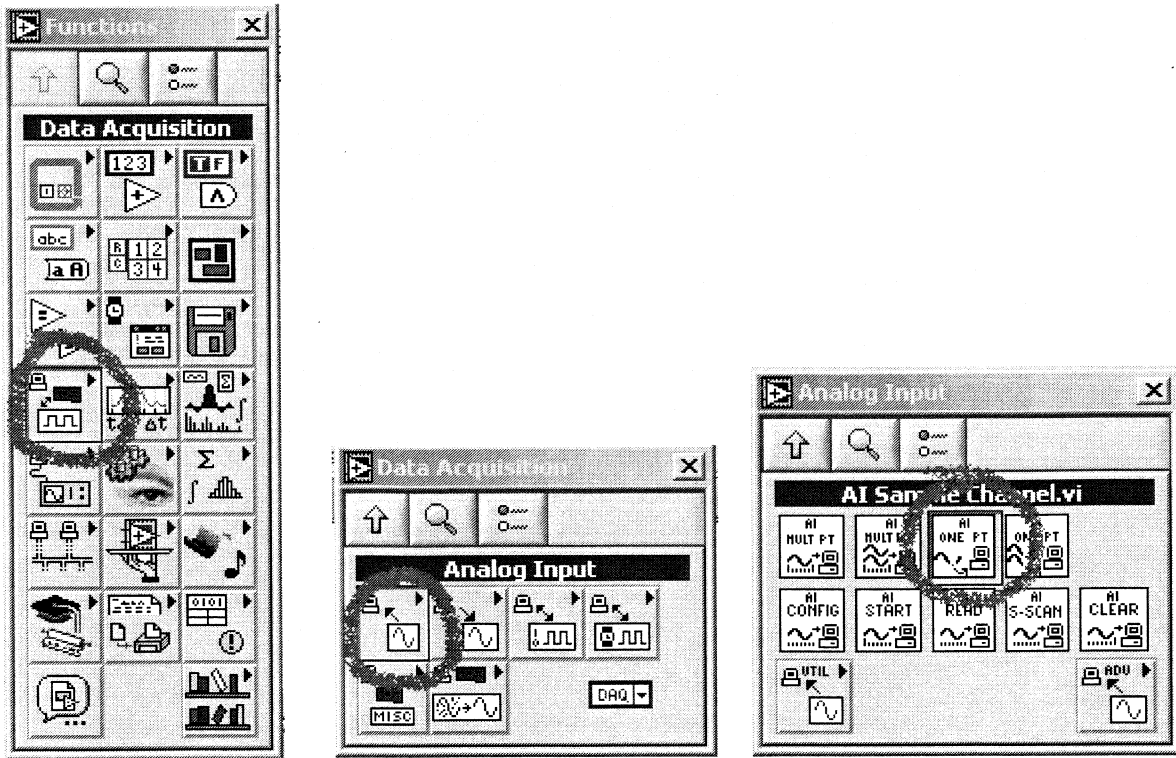
4. Select **Analog Input** and hit **Next>**. Give your channel the channel name **Temperature** and press **Next>**. Select **J Thermocouple** and click **Next>** once more. The units should be **Deg C** and the range from **0.0000** to **100.0000 Deg C**. Click on the **Next>** button and the next screen will automatically generate the physical voltage range of your thermocouple. Press **Next>** once more and then choose **DAQCard-1200** for your DAQ hardware and select channel **4**. Once you have clicked **Finish**, you will have successfully created a **new DAQ Channel Name** in Measurement and Automation Explorer. Now, when you wish to take a measurement inside of LabVIEW from the channel you just configured, simply select the name **Temperature** in a LabVIEW DAQ Channel Name control. Note that when you return to the LabVIEW Front Panel you will see that the **DAQ Channel Name** control on the Front Panel reads **Temperature** as shown on the following page.



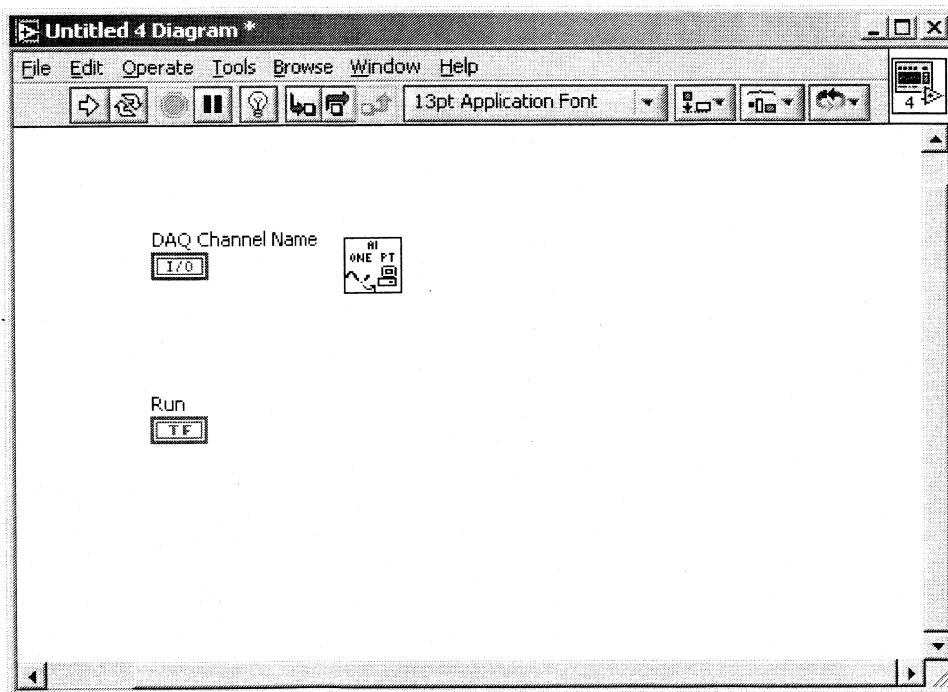
5. Place a **Vertical Toggle Switch** on the Front Panel from the **Boolean** subpalette of the **Controls** palette. Once you have placed it on the panel, type **Run** on the keyboard and the label of the switch will change from **Boolean** to **Run** or use the **Labeling** tool.



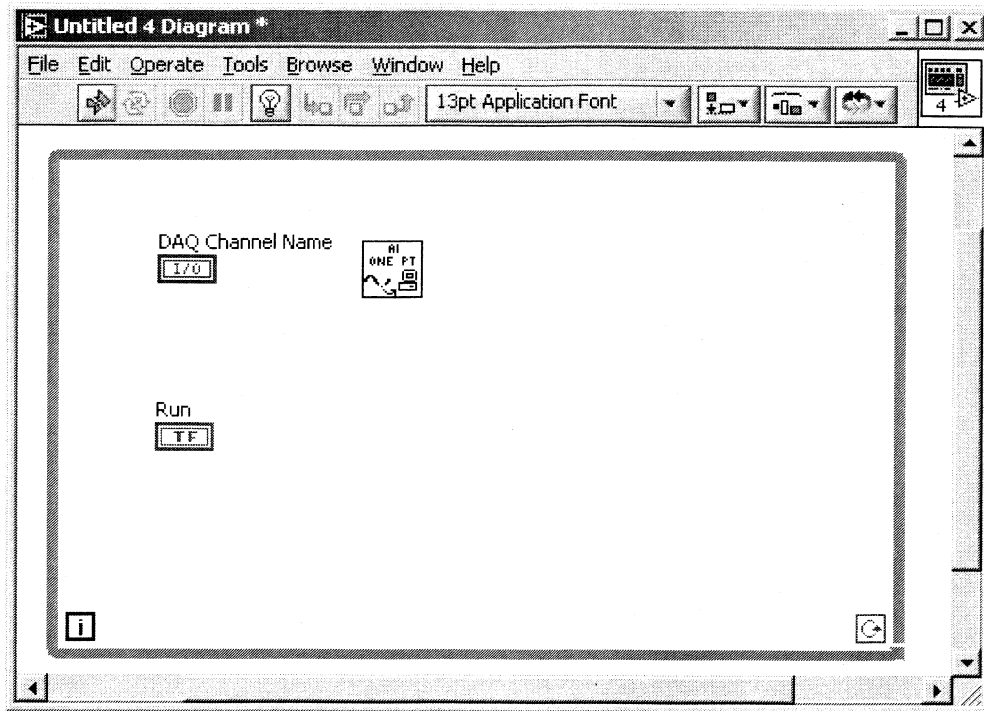
- Switch to the **Block Diagram** and place an **AI Sample Channel VI** down. You can find it by going through the **DataAcquisition>>Analog Input** subpalettes of the **Functions** palette. Push the up arrow twice on the Functions palette to return from the subpalettes to the main functions palette.



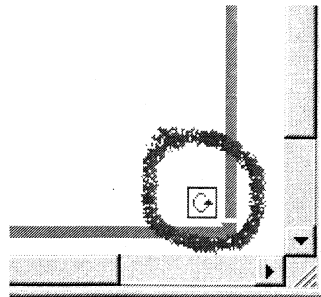
Your block diagram should look like:



7. Click on the **Structures** subpalette of the **Functions** palette (upper left corner) and click on the **While Loop**. **Left click** on the Block Diagram and hold the mouse button down while dragging. A box will appear as you move the mouse around. Enclose the **DAQ Channel Name** and **Run** control terminals and the **AI Sample Channel VI**. A thick gray box representing a **While Loop** will appear around them once you release the mouse button.

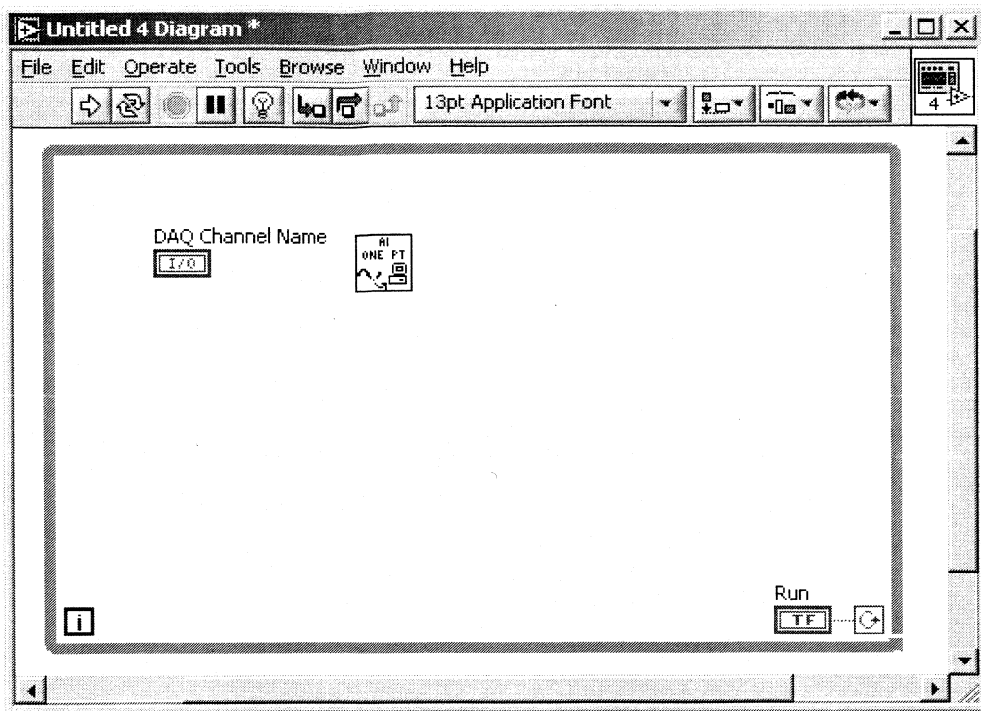


8. Drag the **Run** control terminal until it is adjacent to the **Conditional** terminal of the **While Loop** and press **Space Bar** while still pressing the mouse button. The **Conditional** terminal is in the lower right corner of the **While Loop** and is a circular green arrow as shown below:

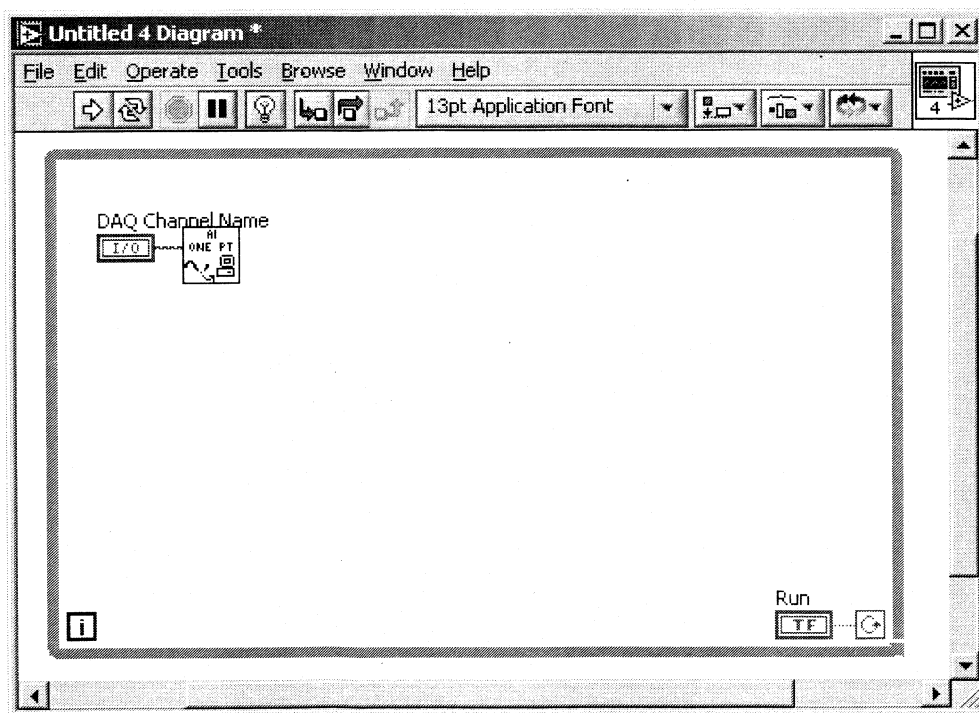


A thin, green wire automatically appears between the **Conditional** terminal and the **Run** control terminal, indicating that the **Run Toggle Switch** on the Front Panel controls the **While Loop's** execution. If the **Run Toggle Switch** is in the **up** position, the **While Loop** executes. Conversely, flipping it **down** stops execution. LabVIEW automatically wires objects together when you place them down or move them. You can toggle this

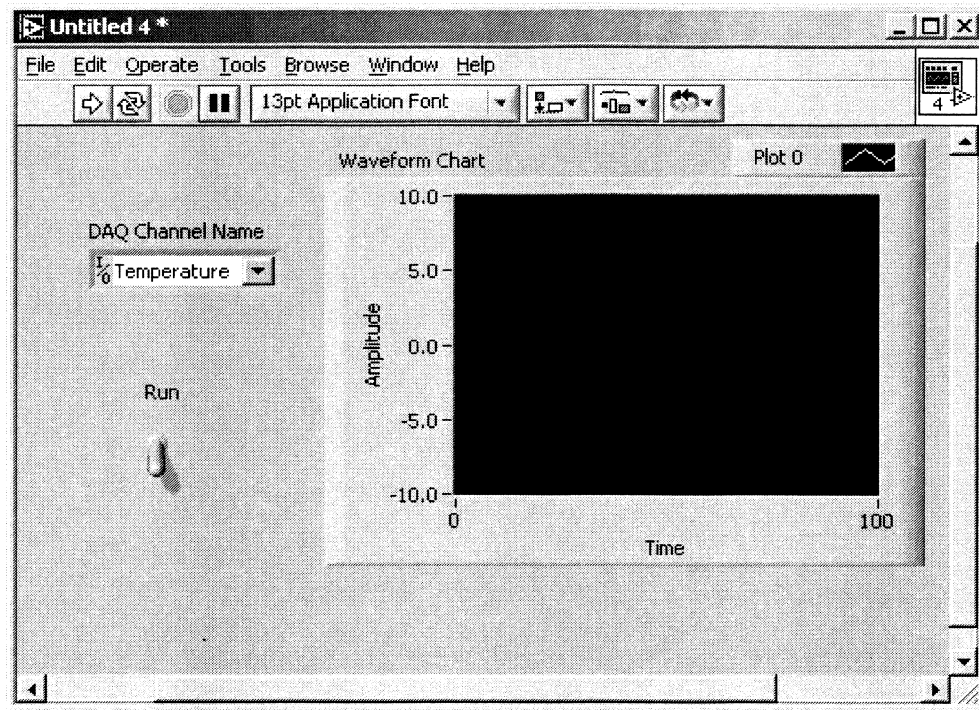
feature on and off by using the **Space Bar**. Your Block Diagram should look like the one below:



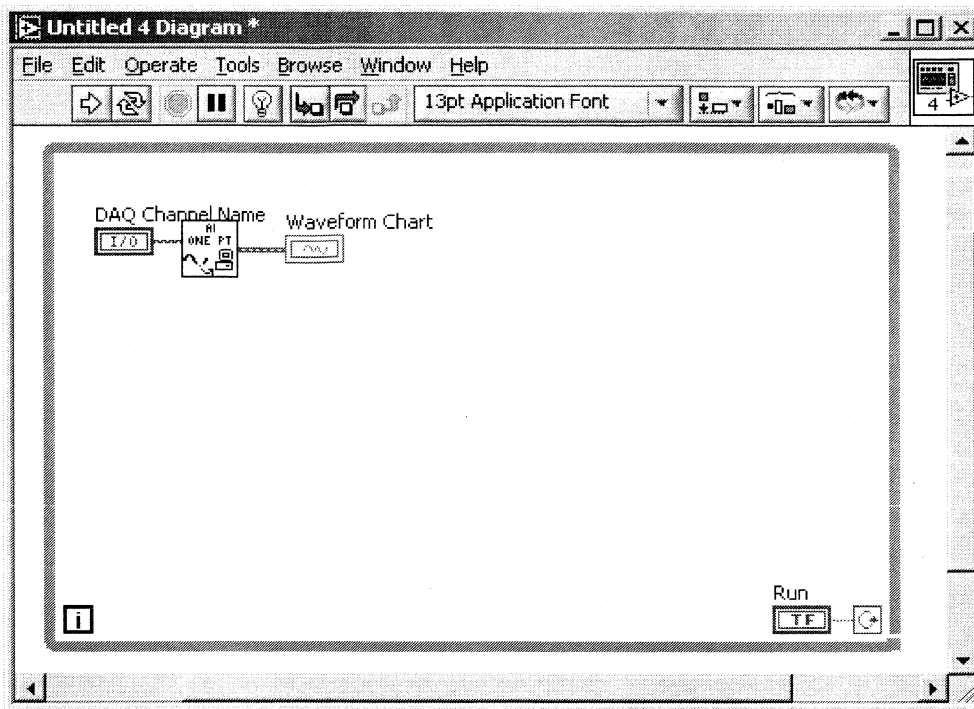
9. Wire up the **DAQ Channel Name** control to the channel input of **AI Sample Channel VI** using the same procedure for wiring as detailed in the previous step. The default convention in LabVIEW places inputs on the left side of functions and VIs and the outputs on the right side.



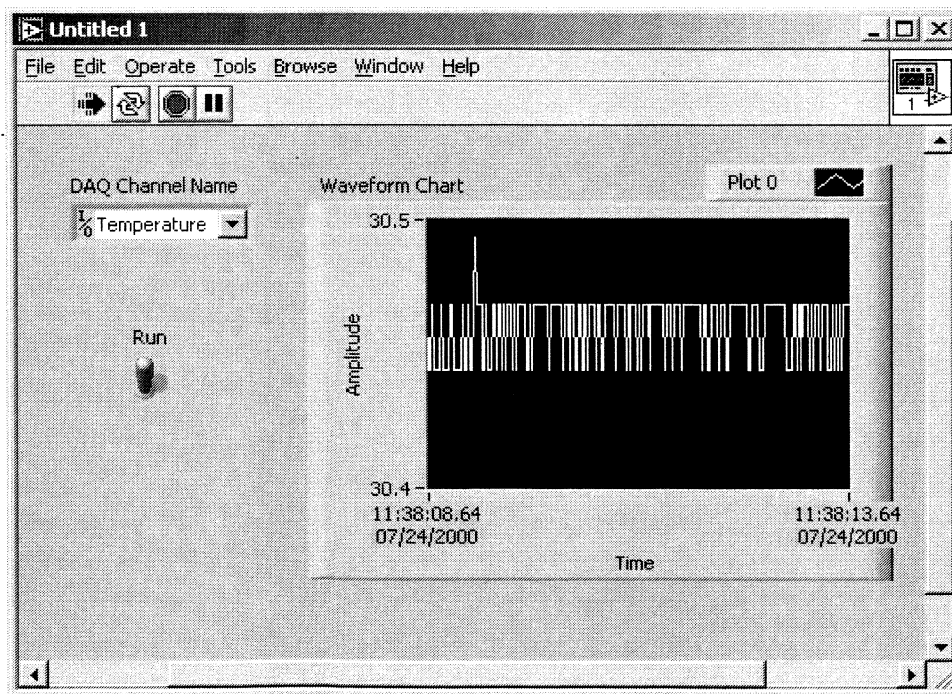
10. Go back to the Front Panel by pressing **Ctrl+E** or choosing **Window>>Show Panel** from the menu bar.
11. Click on the **Graph** subpalette of the **Controls** palette and choose **Waveform Chart** and place it on the Front Panel. Click the **up arrow** to return to the main Controls palette.



12. Wire the **Waveform Chart** terminal to the **sample** output of the **AI Sample Channel VI**. Remember you can use automatic wiring by **dragging** the **Waveform Chart** terminal next to the **sample** output of the **AI Sample Channel VI** and pressing **Space Bar**.



- Go back to the **Front Panel** by pressing **Ctrl+E** and flip the **Run Toggle Switch up** using the **Operate** tool. Autoscale the data in the **Waveform Chart** by **right clicking** on the **Y axis label** or **markers** and choosing **Autoscale Y**. Then, press the **run button** to **run** the VI and **acquire** temperature measurements continuously from your thermocouple. Switch the **Run Toggle Switch down** with the **Operate** tool to stop the VI.

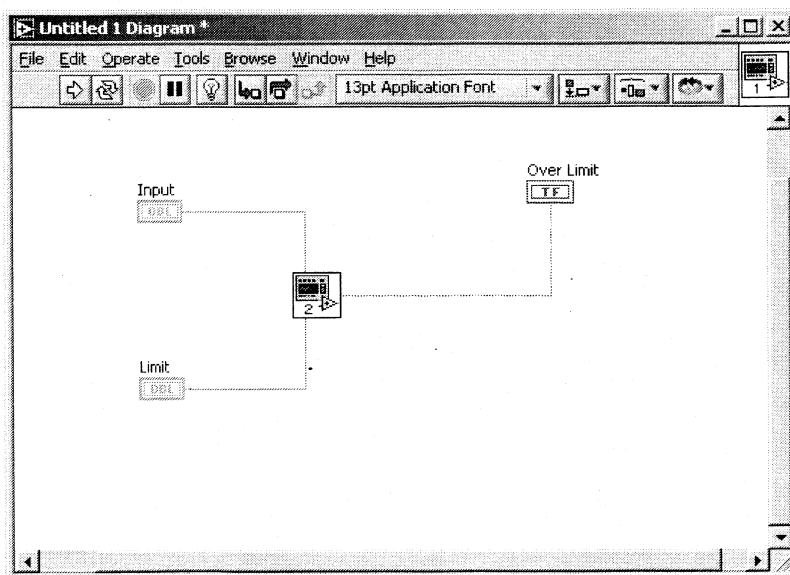


14. **Save** this VI by pressing **Ctrl+S** or choosing **File>>Save** in the **File** menu. Save it as **Exercise 3** in the **C:\National Instruments\LabVIEW\seminar\customer work** folder. Leave your VI open, as you will use it in the next exercise.

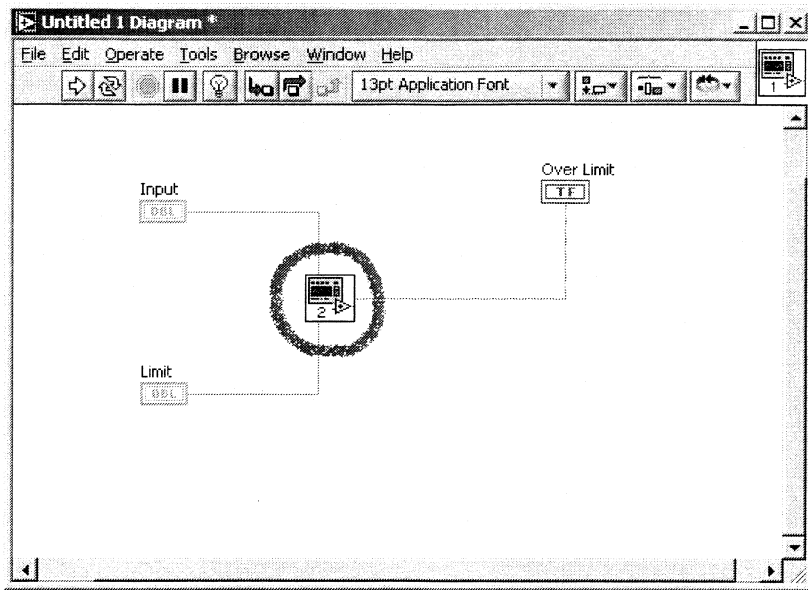
Exercise 4 – *Test temperature readings using a subVI*

In this exercise, you will use the subVI you created in Exercise 1 inside your VI taking temperature measurements to determine whether or not the readings exceed a specified limit.

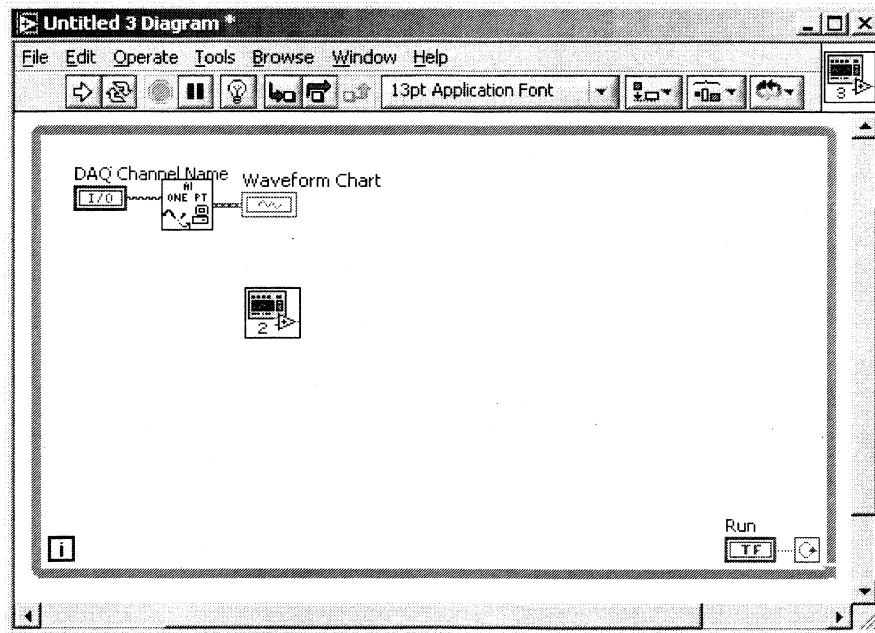
1. In LabVIEW, **subVIs** are VIs that you use inside of other VIs. A subVI is represented on a VI Block Diagram by an **icon**. In addition, subVIs have **connector panes** which contain their **input** and **output** terminals. You can wire controls, indicators, functions, and other VIs to these terminals. The **connectors** to a function or subVI become **visible** when you **hover** over them with the **Wiring** tool. All LabVIEW VIs, including top level VIs that call other subVIs have an icon and a connector pane. You will now embed the VI you previously created to test if an input exceeds a defined limit in the VI which acquires temperature measurements as a subVI.
2. First, return to the **Block Diagram** of the VI you created to test an input against a specified limit. It should look like this:



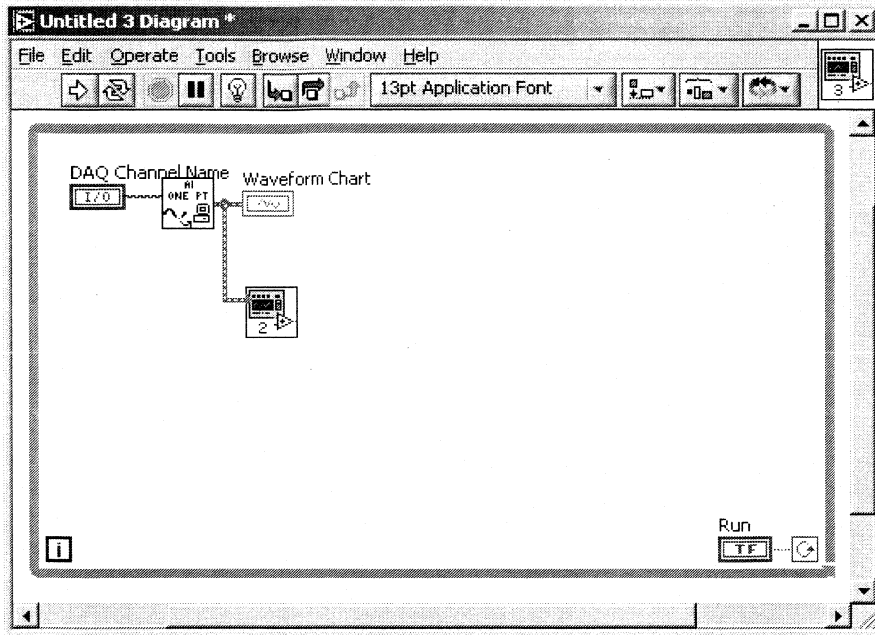
3. Drag the icon on the **Block Diagram** (circled in red below)



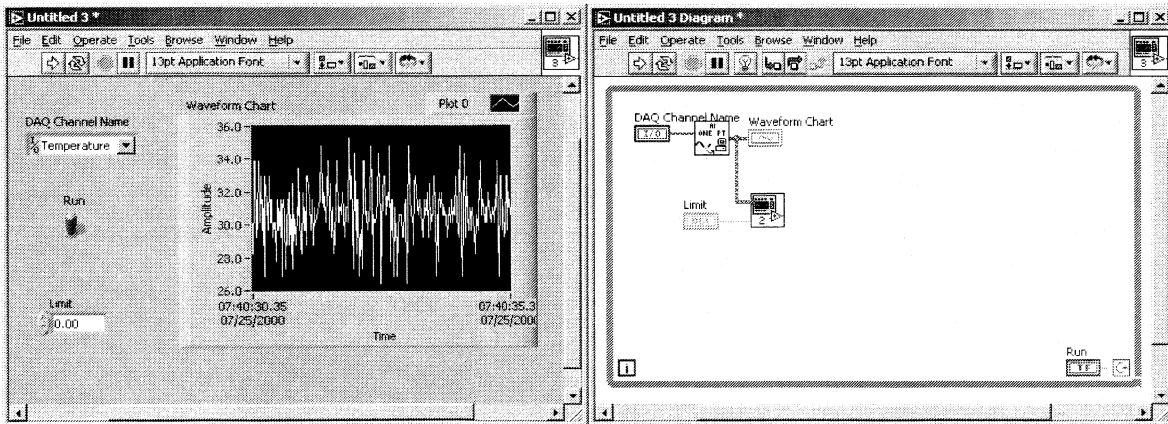
to the **Block Diagram** of your **Exercise 3 VI** that takes temperature measurements as shown below:



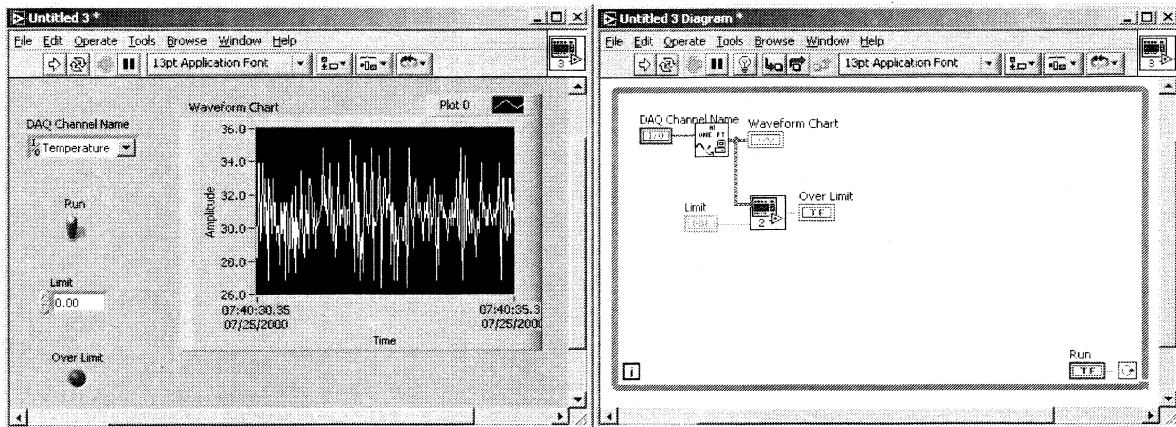
4. Wire the **output** of the **AI Sample Channel VI** to the **Input** terminal on the subVI you just placed on the diagram:



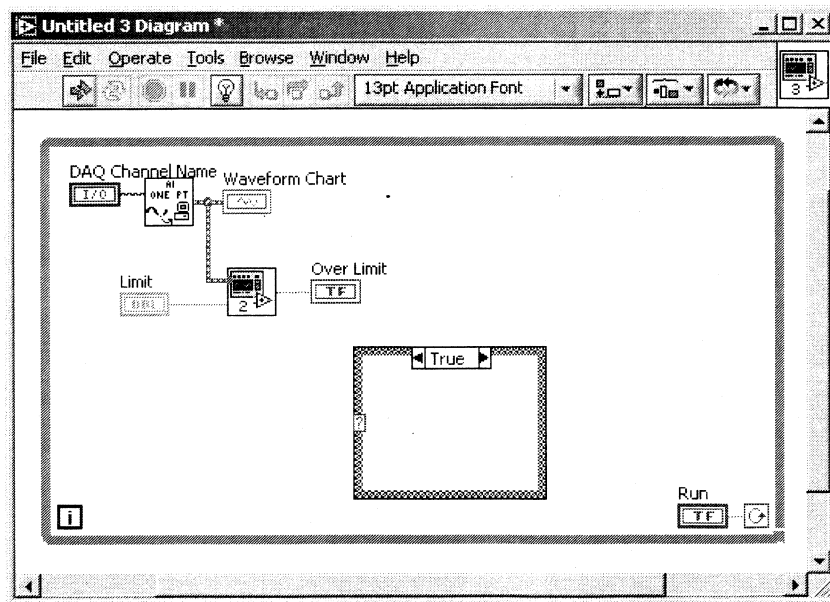
5. Return to the **Front Panel** and place down a **Digital Control** from the **Numeric** subpalette and label it **Limit**. Then, return to the **Block Diagram** and wire the orange **Limit** terminal to the **Limit** input on the subVI. The Front Panel and Block Diagram are both shown below:



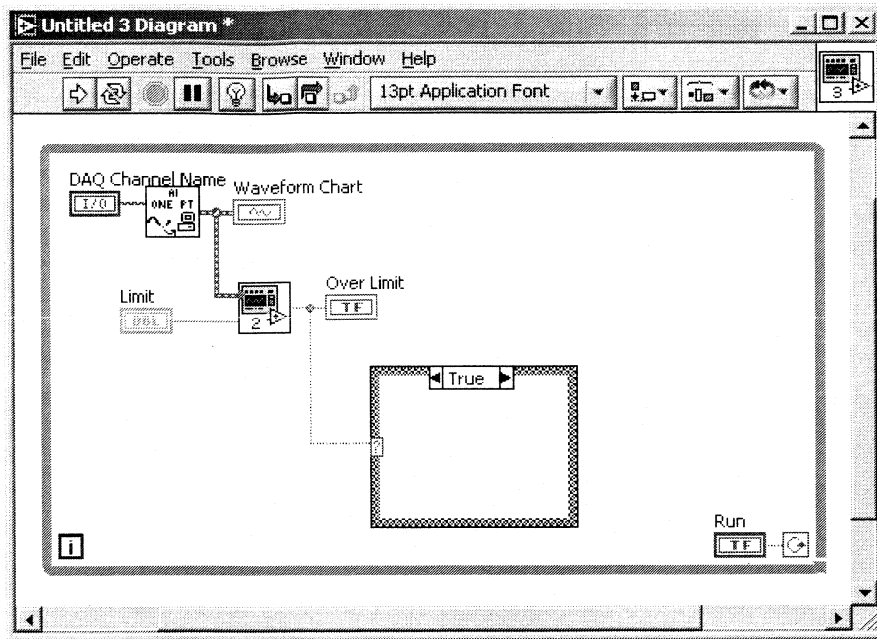
6. Add a **Round LED** to the **Front Panel** and label it **Over Limit**. Then connect its terminal on the **Block Diagram** to the output of the **subVI** as shown below:



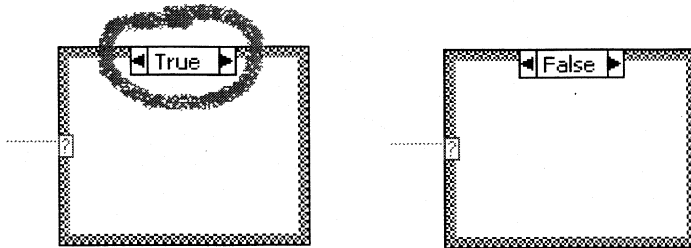
7. Make sure the **Run Toggle Switch** on the **Front Panel** is **up** and **run** the VI by pressing the **Run** button. You will see the **Over Limit LED** light because the temperature you are acquiring is greater than 0. Change the value of the **Limit** Digital Control with the **Operate** tool and observe the LED turn on and off as the **temperature** reading exceeds or falls below the value of **Limit**.
8. Return to the **Block Diagram** and place down a **Case** structure from the **Structures** subpalette of the **Functions** palette like the one pictured below:



- Wire the **output** of the **subVI** to the **Selection** terminal of the **Case** structure. The **Selection** terminal looks like a green question mark. Your **Block Diagram** should look like:

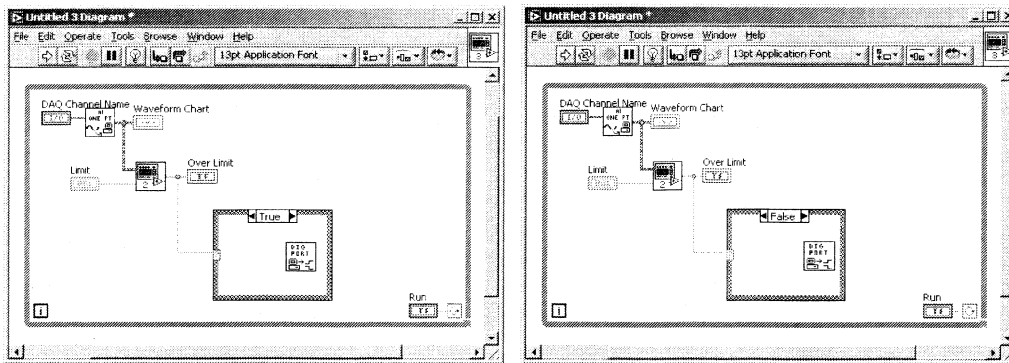


- Click on the **increment** or **decrement** arrows next to the **True** diagram identifier at the top border of the **Case** structure to switch to the **False** case and then switch back to the **True** case.

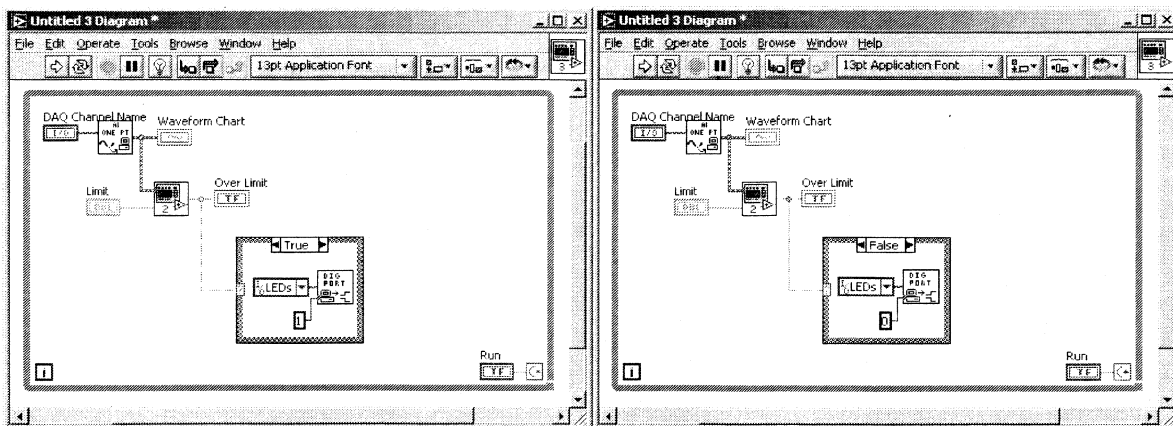


The **Case** structure has both a **True** and a **False** case, one of which executes when the structure executes. The **True** case executes if the input to the **Selection** terminal is a **Boolean true** and the **False** case executes if the **Selection** terminal receives a **Boolean false**. In this case, the subVI that checks the temperature reading against the Limit value outputs a true or a false depending on whether or not the limit has been exceeded. Since the output of this subVI is wired to the Selection input of the Case structure, the output of the temperature limit test dictates what type of action the Case statement executes.

11. Now you will add the code necessary to turn on an LED on the DAQ signal accessory box when the temperature reading exceeds the limit specified by the value of the Limit Digital Control. First, place a **Write to Digital Port VI** in both the **True** and the **False** case of the **Case** structure. You can find the **Write to Digital Port VI** in the **Data Acquisition>>Digital I/O** subpalette of the Functions palette.



12. In both the **True** and the **False** cases of the **Case** structure, wire up a **DAQ Channel Name** constant to the **digital channel** input on the **Write to Digital Port VI**. The **DAQ Channel Name** constant is in the **Data Acquisition** subpalette of the Functions palette. To access the digital lines on the DAQ card, **right click** on both **DAQ Channel Name** constants and choose **Select DAQ Class...>>Digital Output** from the shortcut menus. Then, switch to the **Operate** tool and **click** on the purple down arrows on the **DAQ Channel Name** constants and choose **LEDs** in both cases. Also wire a **Numeric** constant from the Numeric subpalette to the **pattern** input on the **Write to Digital Port VI** in the **True** and the **False** case. In the **True** case, change the value of the **Numeric** constant from 0 to 1 using the **Labeling** tool and leave the **Numeric** constant in the **False** case set to 0. Your Block Diagram should look like:



13. If you convert a decimal 0 or 1 into binary, the least significant bit is the only difference between the two numbers. Because of this fact, the **Numeric** constant wired to the **pattern** input on the **Write to Digital Port VI** writes either a 0 or a 1 to the first line, or least significant bit, of the port and all other lines are unaffected. If a 1 gets written to the port, the first line sees a logical 1. If a 0 gets written to the port, the same line sees a logical 0. In this way, you can toggle an LED on or off simply by writing to a digital port. **Run** the VI and note that both the **Over Limit** LED on the Front Panel and one of the four **LEDs** on the DAQ signal accessory box light when the temperature reading exceeds the value specified by the **Limit** Digital Control.

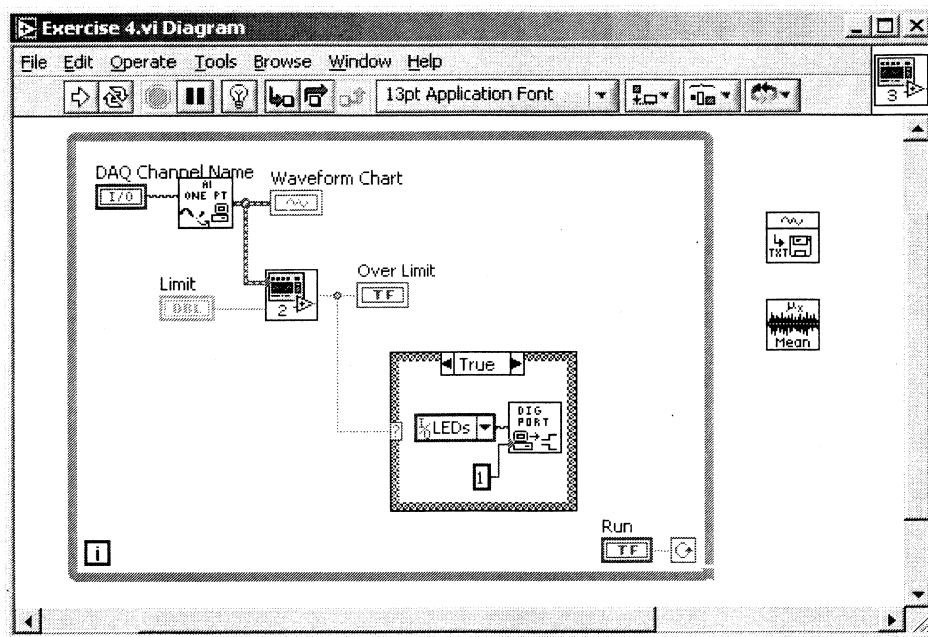
14. Save your VI in **C:\National Instruments\LabVIEW\seminar\customer work**. You will be prompted whether or not to save untitled subVIs. Press **OK** and then give the subVI in the VI you have just finished the name **subVI** in the file dialog that appears. Then, proceed to save the main VI as **Exercise 4** in the second file dialog.

(OPTIONAL)

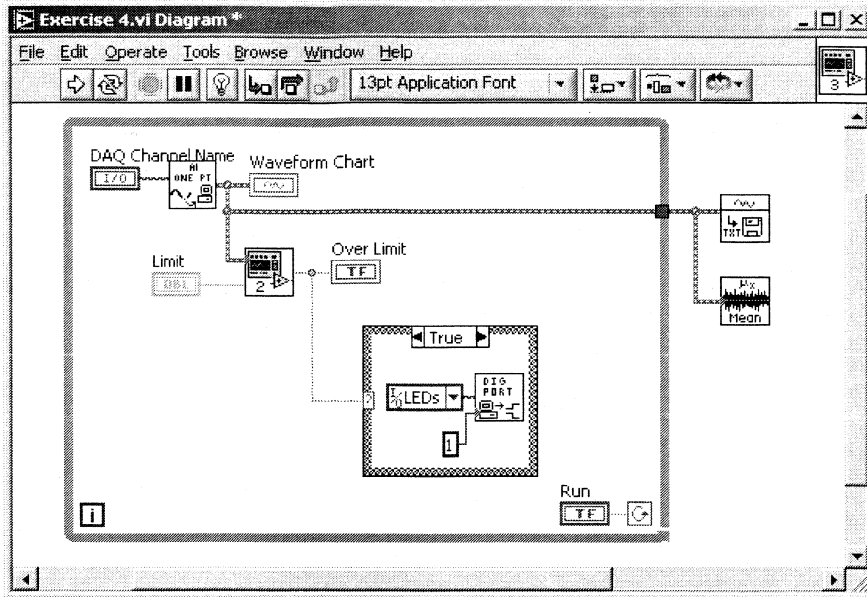
Exercise 5 – *Save temperatures to disk and calculate the mean*

In this exercise, you will use the VI you created in Exercise 4 and build on it so that you can save temperature readings to a spreadsheet file and calculate the average (mean) of the acquired readings.

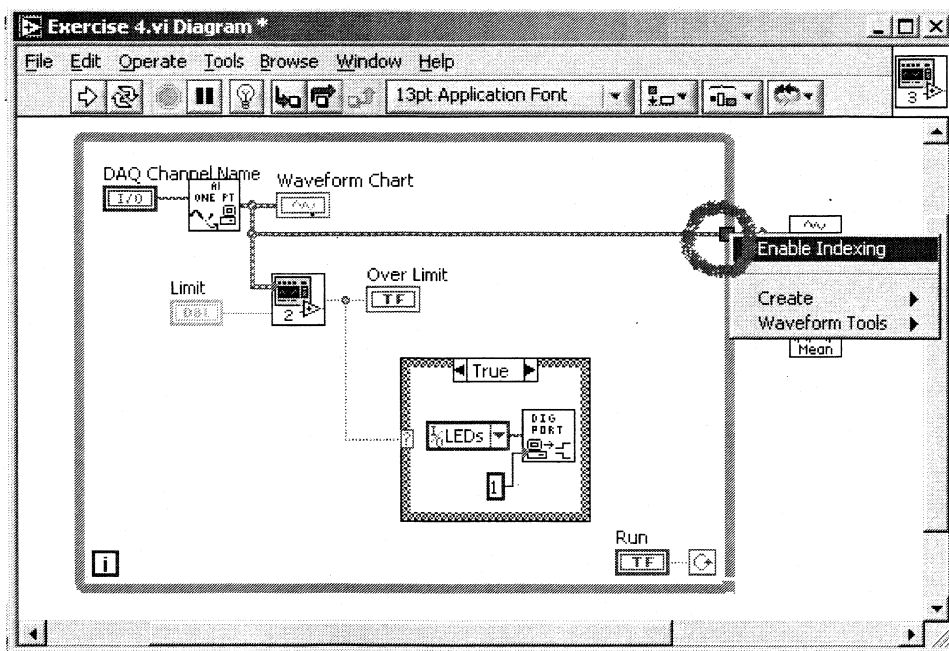
1. Switch to or open the VI you created in the last exercise called **Exercise 4.vi**
2. Add an **Export Waveforms to Spreadsheet File VI** and a **Mean VI** to the Block Diagram as shown below. You can find the Export Waveforms to Spreadsheet File VI in the **Waveform>>Waveform File I/O** subpalette and the Mean VI in the **Mathematics>>Probability and Statistics** subpalette of the Functions palette.



3. Wire the **output** of the **AI Sample Channel VI** to the **Waveforms** input on the **Export Waveforms to Spreadsheet File VI** and the **X** input on **Mean VI** as depicted below:

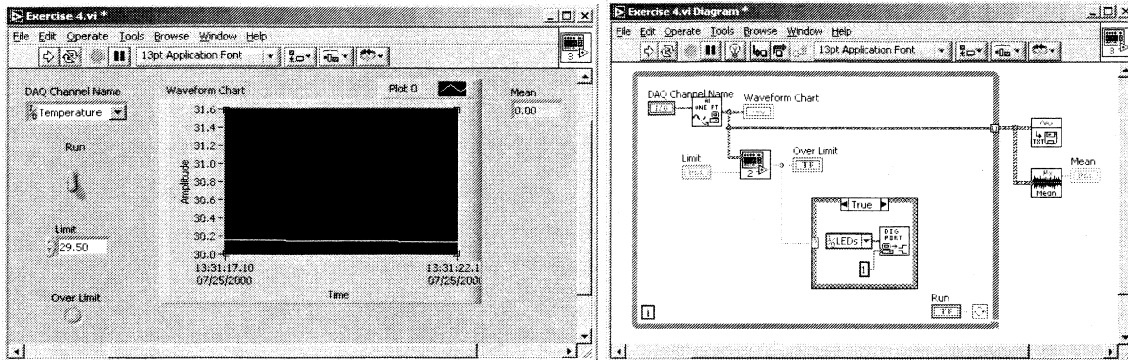


4. **Right click** on the **Tunnel** at the **While Loop**'s right-hand border and select **Enable Indexing**.



This allows all of the data generated inside the While Loop to automatically accumulate at the Tunnel after completion of the loop so that you can write all of the acquired temperature data to a spreadsheet file instead of just the data generated by the last iteration of the While Loop.

5. Create a **Digital Indicator** on the Front Panel (**Numeric** subpalette of the **Controls** palette), name it **Mean**, and **wire** its orange terminal on the **Block Diagram** to the **output** of Mean.vi.

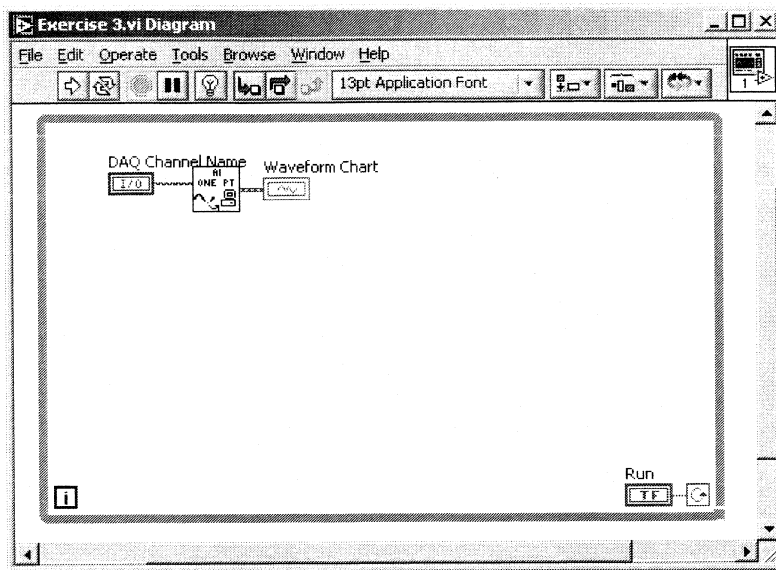


6. Make sure the **Run Toggle Switch** is **up** and then **run** the VI. When you flip the Run switch **down** to stop the VI, a dialog box will appear asking you to give a name to the file containing the exported waveforms. Save the file as **temperature.xls** in the **C:\National Instruments\LabVIEW\seminar\customer work** folder. Note that the **Mean** indicator on the Front Panel displays the mean of the acquired temperature data. Open the **C:\National Instruments\LabVIEW\seminar\customer work** folder using Windows Explorer and **double-click** on the **temperature.xls** icon to **launch** Excel, **open** the file, and **view** the data it contains. Note that the waveform file contains not only the data you acquired but timing information as well. **Save** your VI as **Exercise 5.vi** in the same **customer work** folder using the **Save As...** option in the **File** menu.

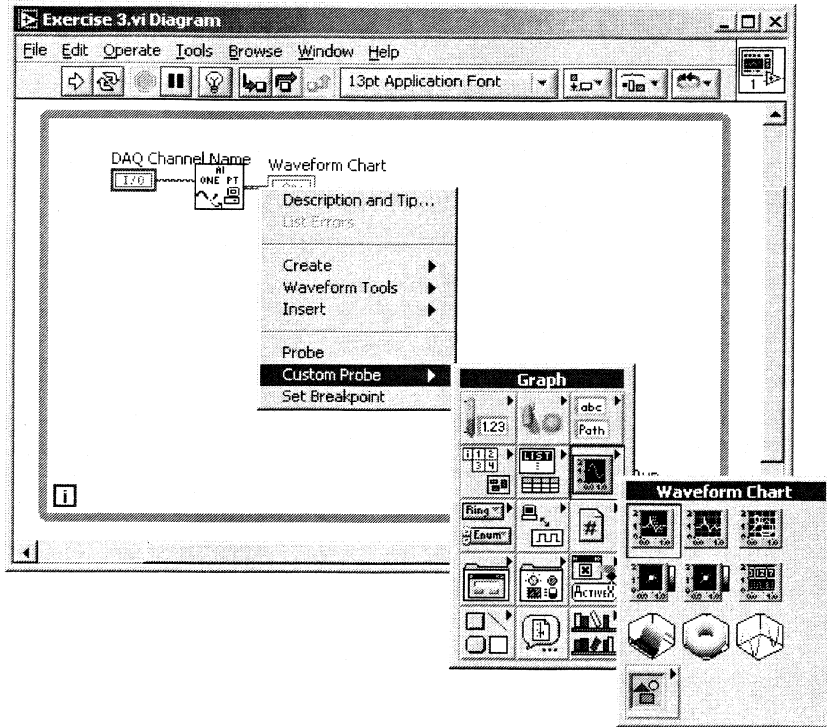
Exercise 6 – Use debugging tools

In this exercise, you will learn how to use the debugging tools available to you in LabVIEW to help you diagnose and correct problems with your VIs.

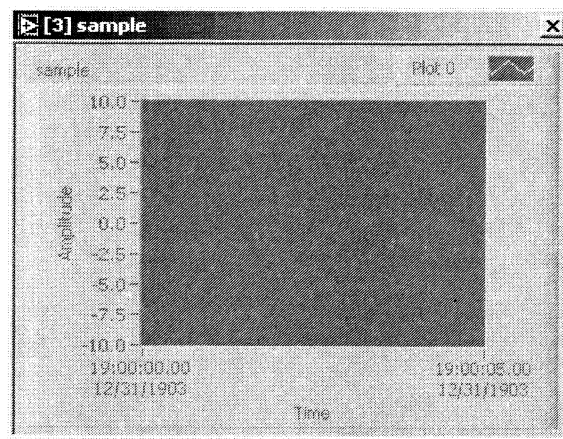
1. Open **C:\National Instruments\LabVIEW\seminar\customer work** and open the **Block Diagram** of **Exercise 3.vi**. It should look like this:



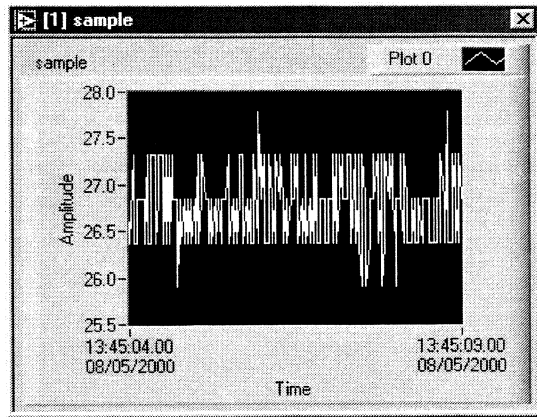
2. Be sure the **Run** Toggle Switch on the **Front Panel** is flipped **up**. **Right click** on the **brown wire** segment that connects the **AI Sample Channel VI** to the **Waveform Chart** and select **Custom Probe**. A **Controls** palette will appear. Click on the **Waveform Chart** inside the **Graph** subpalette as shown on the following page.



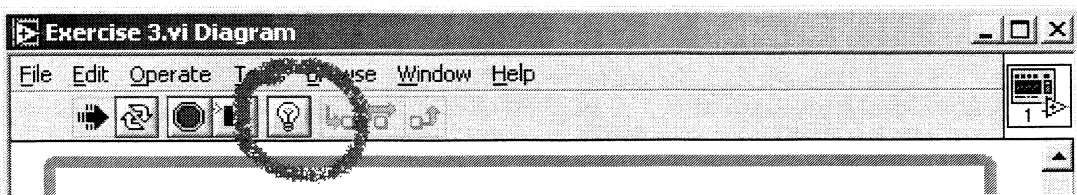
3. A probe will appear as a **floating window** that looks like this:



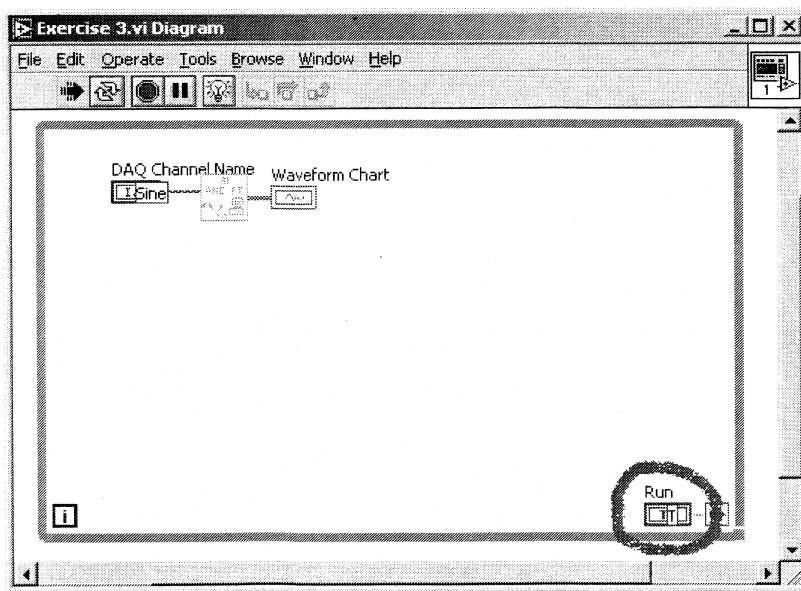
This probe will **display** the **data** flowing through the wire from the AI Sample Channel VI to the waveform chart as the VI **executes**. **Run** the VI and you will see **temperature readings** being plotted. This is a useful tool for examining data values at various points along your VI while it is executing.



4. Close the floating probe window and click on the lightbulb found in the toolbar of the VI Block Diagram as shown below:

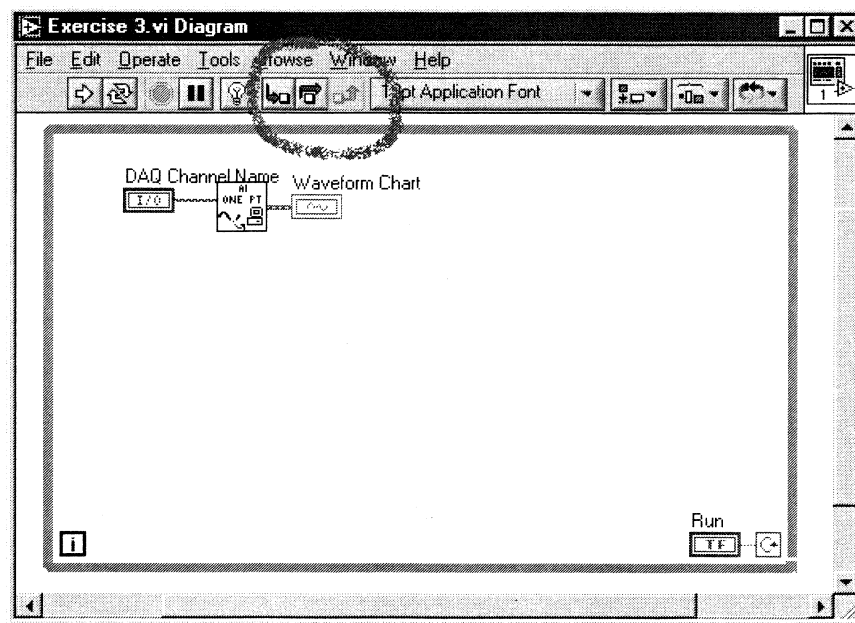


This is the **execution highlighting** option which allows you to **watch** the **execution** of your VI in slow motion. You will see **bubbles** traveling down the various **wires** indicating the **flow of data** through the VI, as well as the **current values** of the Front Panel **controls** floating over their respective **terminals** on the Block Diagram. For example, you should see a **T** displayed over the top of the green **Run terminal** in the **lower right corner** of the **While Loop** indicating that the present value of the **Run Toggle Switch** on the **Front Panel** is a Boolean **True**.



This tool is particularly useful for watching the flow of execution in your VI.

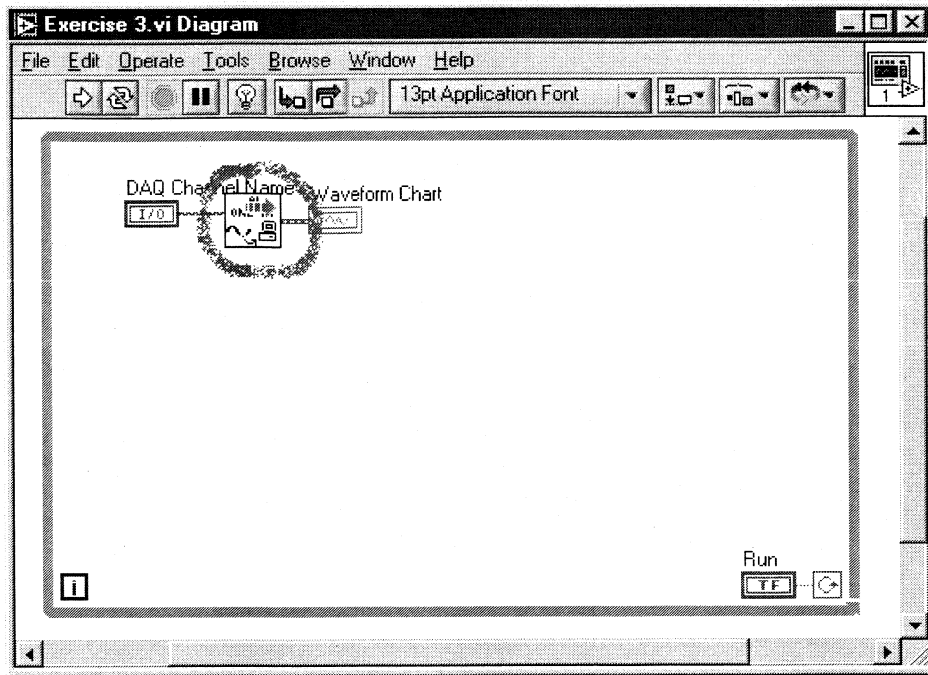
5. Click on the **lightbulb** on the **toolbar** once more to turn off execution highlighting. Return to the **Front Panel** using **Ctrl+E** and switch the **Run Toggle Switch** to the **down** position to **stop** the VI. Once you have stopped the VI, go back to the **Block Diagram**. You can **single-step** through a VI to view each **individual action** or operation of the VI on the **Block Diagram** as the VI runs. There are **three buttons** associated with **single-stepping** through a VI: **Step Into**, **Step Over**, and **Step Out**. All three are circled below in red. The **Step Into** button is on the **left**, **Step Over** is in the **middle**, and **Step Out** is on the **right**.



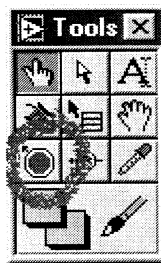
In LabVIEW terminology, a **node** is simply a **program execution element**. Nodes are analogous to statements, operators, functions, and subroutines in text-based programming languages. On the Block Diagram, **nodes** include **functions**, **structures**, and **subVIs**. The **Step Into** button **opens** a node and **pauses**. When you click the Step Into button again, it executes the first action and pauses at the next action of the subVI or structure. The **Step Over** button **executes** a node and **pauses** at the next node. **Step Out** finishes executing the current node and **pauses**.

6. Click the **Step Into** button once and the VI will pause execution at the **While Loop** which will cause it to **blink**. Press the **Step Into** button once more and program execution **pauses** at the **AI Sample Channel** subVI which will **blink**. Pressing the Step Into button would open up the AI Sample Channel VI and pause execution inside the subVI and successive presses of the single-stepping buttons would allow you to watch program execution in the subVI. Instead, press the **Step Over** button to **completely execute** the **AI Sample Channel** subVI and complete the first iteration of the While Loop. Execution pauses at the **While Loop** so it is once again **blinking**. However, this time when you press the **Step Into** button, the entire **Block Diagram** blinks, indicating that the VI has **finished executing**. Because the **Run Toggle Switch** on the **Front Panel**

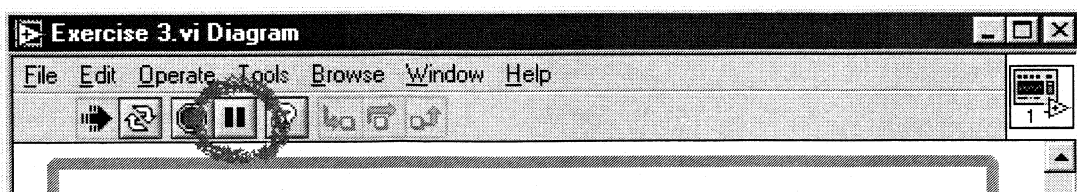
is **down**, the **While Loop** executes only **once** since the **Conditional** terminal receives a Boolean **False**. To **step out of the VI** after it has completed executing, click the **Step Out** button. One additional note is that **subVIs** that are currently **running** will have a **green** execution glyph on the **icon** like the one shown below:



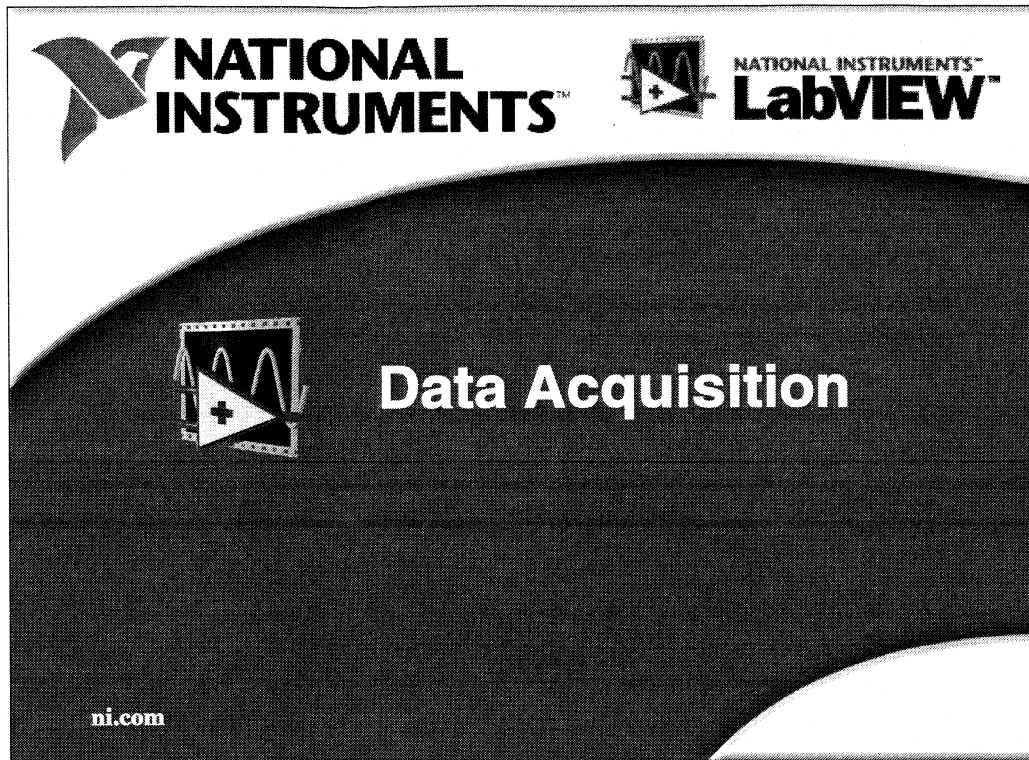
- To arbitrarily **pause** the execution of a VI at a given **subVI**, **node**, or **wire**, select the **Breakpoint** tool from the **Tools** palette and click on the object.



Then **run** the VI and its execution will be **suspended** at the spot where the **breakpoint** has been set. To **temporarily stop** the execution of your VI or insert a probe, click the **Pause** button in the **toolbar**.



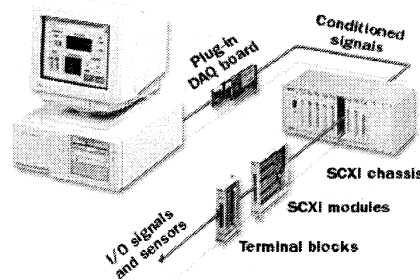
- Close** Exercise 3.vi **without** saving changes.



In many applications, plug-in data acquisition boards or computer-based instruments can completely replace traditional benchtop instrumentation. The LabVIEW data acquisition (DAQ) libraries contain functions for performing analog and digital I/O as well as counter/timer operations. Plug-in DAQ boards are ideal for high-speed and direct control applications. Because of their lower cost, they bring your cost per channel down significantly.

Data Acquisition Basics

- LabVIEW works with all NI data acquisition devices through NI-DAQ® driver software
- DAQ boards for:
 - Analog I/O
 - Digital I/O
 - Counter/Timers
- Signal conditioning through SCXI



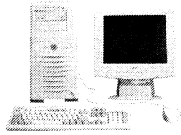
ni.com



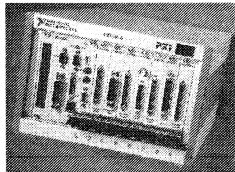
National Instruments data acquisition devices harness modern technologies like reduced noise designs, ASICs for increased reliability, hands-free calibration, and shielded I/O connectors for accurate measurement collection from sensors and transducers. Here is an example of a typical data acquisition system. Generally, you have a plug-in DAQ board inside a host PC or other measurement peripheral, such as a USB DAQ device. The DAQ device is cabled with a shielded cable to an SCXI chassis which provides signal conditioning to interface sensor and transducer outputs to the DAQ device. Inside the SCXI chassis are modules relevant to the application which provide signal conditioning, including signal amplification, filtering, switching, and isolation. On the front of the SCXI modules are terminal blocks to which you wire or connect your measurement sensors or transducers.

DAQ Host Platforms

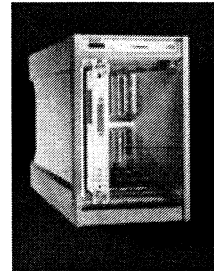
Desktop



Laptop

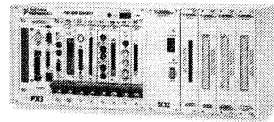


PXI



VXI

ni.com

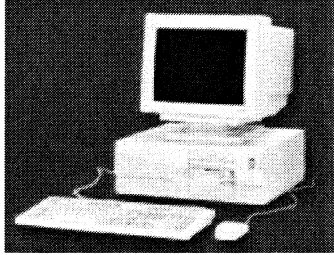


PXI/SCXI



Typical platforms today that host your DAQ devices include desktop PCs and laptops, PXI/CompactPCI rackmount computers, and VXIbus systems.

Why Desktops?



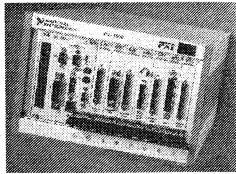
- Advantages
 - Latest technologies
 - Inexpensive
 - Readily available
- Disadvantages
 - Rackmounting difficult
 - Not rugged
 - Few expansion slots

ni.com

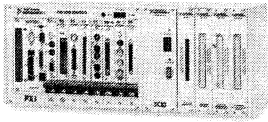


Let's evaluate the desktop PC as a host for DAQ systems since it is so common. Using a desktop PC is advantageous because they are so easy to obtain and are relatively inexpensive. Plus, they evolve rapidly to embrace the latest technological advancements. In effect, they provide the fastest processors at the lowest prices. However, desktop PCs are not rugged and are ill-suited for rackmounting. In addition, there are few expansion slots available for installing additional hardware, such as plug-in PCI DAQ boards.

Why PXI/CompactPCI?



PXI



PXI/SCXI

ni.com

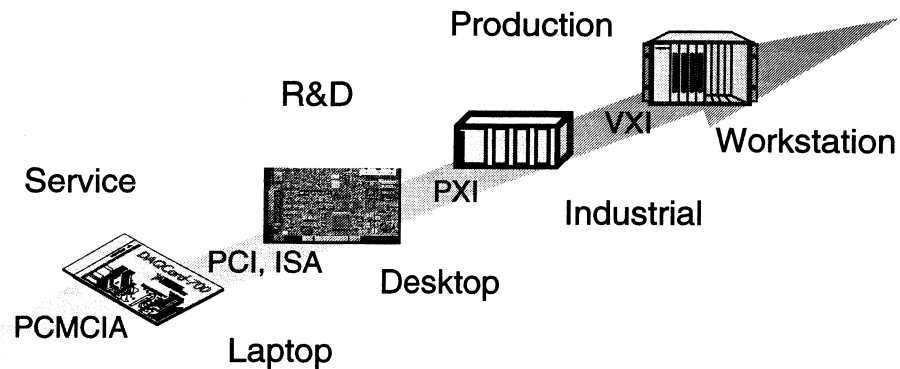
- Advantages
 - Well-suited for rackmounting
 - Rugged
 - More slots
 - Backplane timing and triggering
 - Embedded or external control
 - Same software as desktops
 - DAQ and SCXI in same chassis
- Disadvantages
 - Slightly more extensive than desktop



PXI/CompactPCI systems address these limitations of desktop PCs. PXI is rackmountable, rugged, modular instrumentation that uses a modular Eurocard architecture that has been proven in industrial settings for decades. This architecture allows up to eight PXI/CompactPCI boards to plug into the front of a PXI chassis. In addition, PXI/CompactPCI is well-suited for rackmount applications. PXI systems can be controlled through embedded computers or through direct control from a desktop PC or other PXI system using MXI-3. Desktop PCs and PXI systems run the same software—Windows operating systems—so that code that runs on a desktop will also run on your PXI embedded computer. Plus, PXI also has timing and triggering built directly into the backplane. This means that when you need to trigger acquisitions between boards, you don't have to have messy cabling cluttering the front of your chassis.

You can use a PXI chassis with National Instruments DAQ boards and Computer-based Instruments or you can use a combination PXI/SCXI chassis to hold both your DAQ boards and your signal conditioning modules in the same chassis. Another benefit of PXI is that you can use National Instruments boards or any other PXI or CompactPCI vendor's boards in the same system. PXI is the ultimate test and measurement platform because it gives you all of the timing and triggering benefits of VXI with the speed and cost-effectiveness of desktop PCs.

Scalable DAQ Solutions



Software runs unmodified on each platform

ni.com



Scalability is one of the most important benefits of National Instruments hardware and software. All DAQ products use the same common set of software functions that span all available platforms, buses, and boards. This is significant because today you might be writing code for a laptop with a DAQCard PCMCIA data acquisition card. But six months from now, you may be using a rackmount industrial computer or PXI system. With National Instruments hardware and software, your code migrates with the hardware. You can run the exact same code with any other data acquisition devices that have at least the same performance capabilities as the board for which you originally wrote your code. This preserves your investment in the code you write today, whether you program with LabVIEW, Measurement Studio, Visual C++, or Visual Basic. You always have a forward migration path for your code with National Instruments products.

DAQ Solution Wizard

Common Applications

ni.com

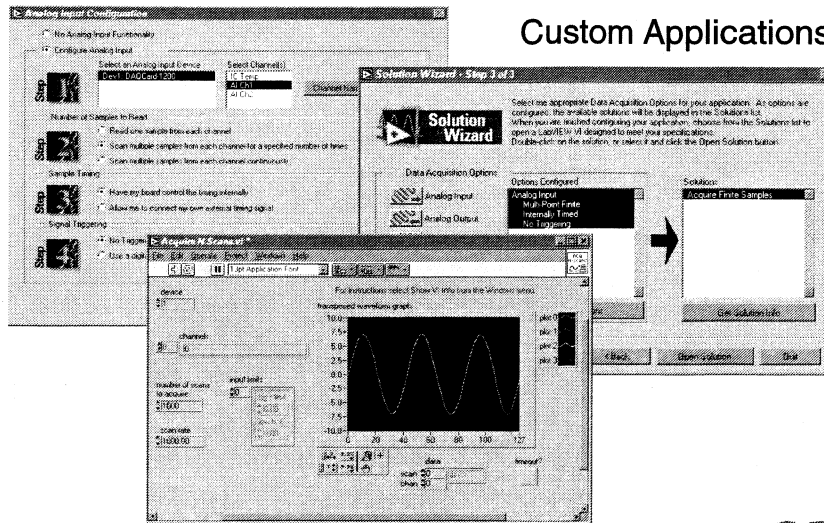
NATIONAL INSTRUMENTS

Many kinds of software packages perform data acquisition. Some are turnkey packages that acquire and present data but you cannot modify them to meet your specific requirements. Some are configurable, menu-driven packages that are easy to use but customization is not possible or is usually quite limited. Finally, traditional programming environments such as Visual C++ and Visual Basic offer the programming flexibility you need but you will likely expend a substantial amount of effort learning the language and programming even simple DAQ applications.

LabVIEW provides the perfect combination of programming ease and complete flexibility. The built-in DAQ Solution Wizard automatically generates applications to meet your needs. By filling in the blanks in a simple, menu-driven wizard, you literally choose your solution—whether you need a virtual oscilloscope, data logger, or temperature measurement system. These programs run immediately with no programming and the DAQ Solution Wizard automatically generates a VI for you complete with a Front Panel and Block Diagram source code. This enables you to modify the generated solution as your application needs change.

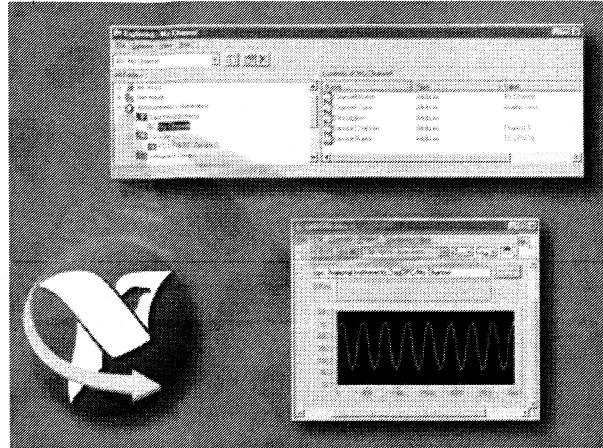
DAQ Solution Wizard

Custom Applications



You can also build custom DAQ solutions by specifying the analog I/O, digital I/O, or counter/timer characteristics that meet your system requirements. The DAQ Solution Wizard then creates a VI using the specifications you have selected. Once again, you can run the generated custom solution immediately. But most importantly, the generated solution is an actual LabVIEW VI that you can modify as your application needs change.

Measurement and Automation Explorer



ni.com



In the past, data acquisition system developers spent a large amount of time defining the signal types, connections, transducer equations, and unit conversions of the system before ever beginning actual development. However, Measurement and Automation Explorer greatly simplifies these tasks by providing access to all your National Instruments DAQ, GPIB, IMAQ, IVI, Motion, VISA, and VXI devices from a single location. With Measurement & Automation Explorer, you can configure your National Instruments hardware and software; add new channels, interfaces, and virtual instruments; execute system diagnostics; and view devices and instruments connected to your system.

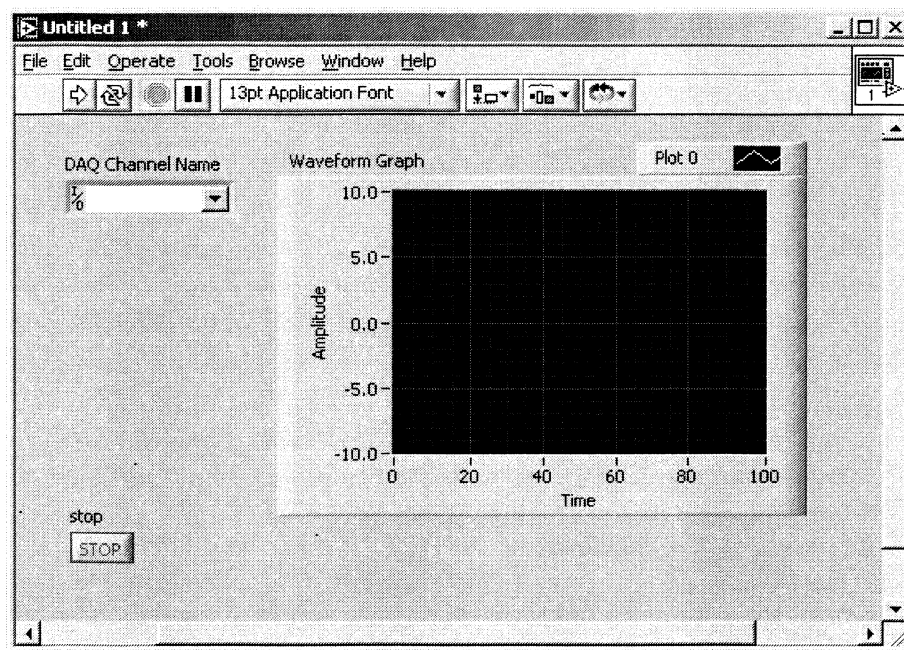
The Data Neighborhood category of Measurement and Automation Explorer provides access to all your virtual channels which are shortcuts to configured channels in your system. These virtual channels allow you to descriptively name physical channels and transparently apply modifications, such as scaling, to your measurements. The Scales category provides access to the custom scales you have configured for operating on acquired data. You can even view, launch, and update the National Instruments software installed on your system with the Software category.

Because of the tight integration between LabVIEW and Measurement and Automation Explorer, you can access, edit, and insert channels directly in Measurement and Automation Explorer from LabVIEW to decrease your application development time. Measurement and Automation Explorer makes you more productive by minimizing the time you spend configuring and accessing your system hardware so that you can spend time on more meaningful tasks like taking measurements.

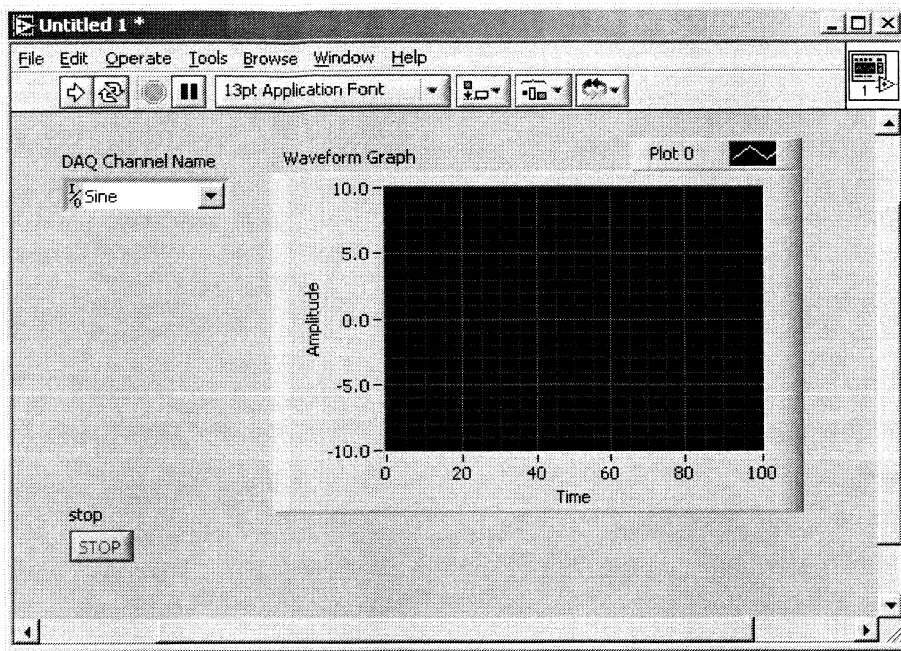
Exercise 7 – *Acquire a waveform and compute the RMS value*

In this exercise, you will acquire more than just a series of temperature readings. You will acquire an entire voltage waveform and then analyze it by computing the RMS value of the signal.

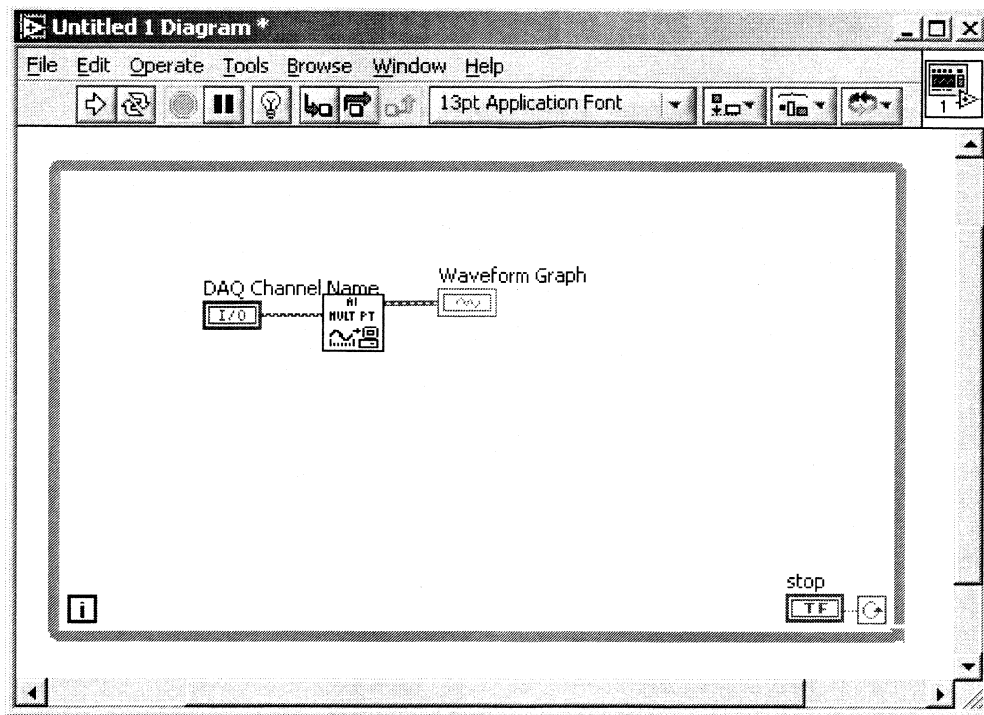
1. Physically **wire** the green **Analog In Channel 1** connector on the DAQ signal accessory to the **Sine** output connector of the **Function Generator**.
2. Create a new VI by pressing **Ctrl+N** and place a **DAQ Channel Name** control (I/O subpalette), **Waveform Graph** (Graph subpalette), and **Stop Button** (Boolean subpalette) from the **Controls** palette on the **Front Panel** like the one pictured below:



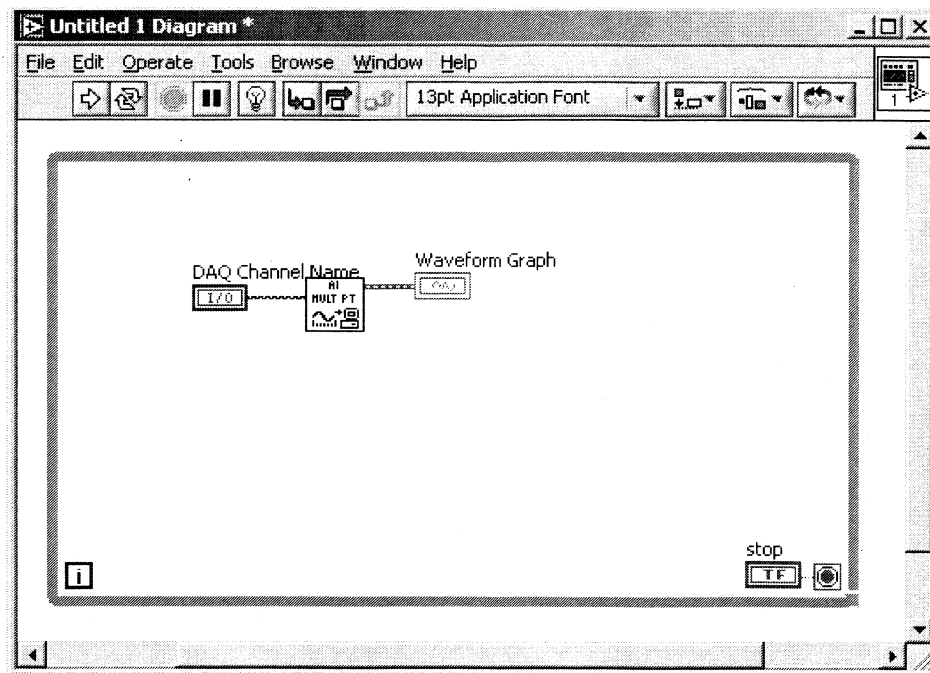
3. Using the **Operate** tool, select **Sine** in the **DAQ Channel Name** control:



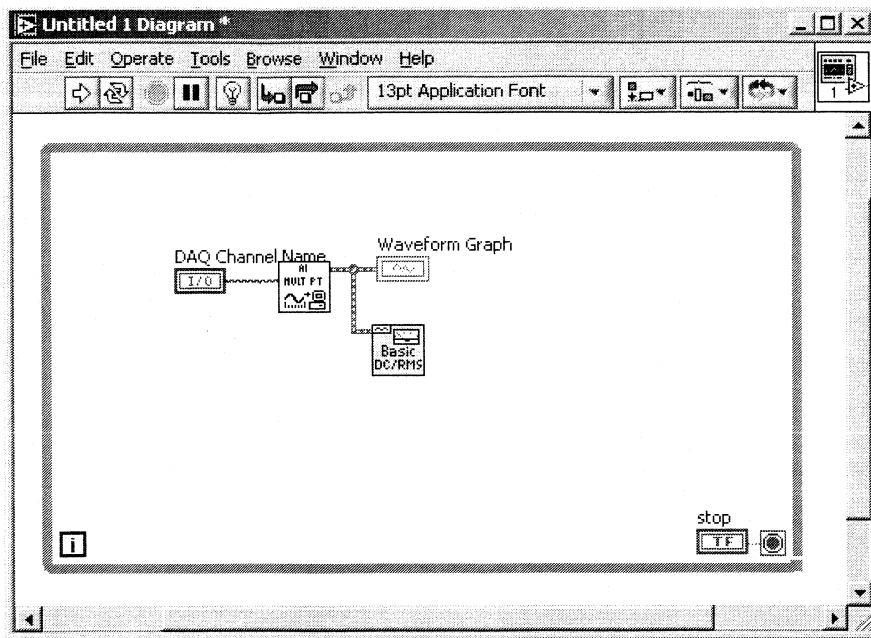
4. Construct the following **Block Diagram** as shown. You will need the **DAQ Channel Name**, **Stop Button**, and **Waveform Graph** terminals as well as a **While Loop** (Structures subpalette) and an **AI Acquire Waveform VI** (Data Acquisition>>Analog Input subpalette) from the **Functions** palette. Wire the **DAQ Channel Name** terminal to the **channel (0)** input on the **AI Acquire Waveform VI** and the **Waveform Graph** to the **waveform** output. In addition, wire the **Stop Button** terminal to the **Conditional** terminal of the **While Loop**.



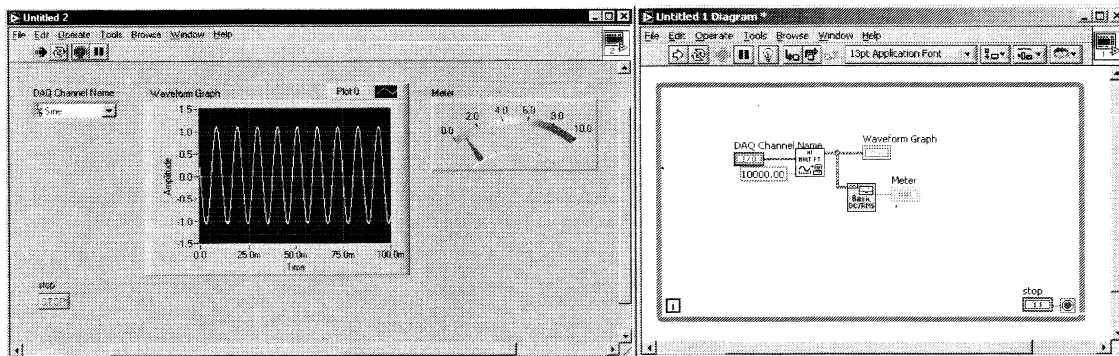
5. Click on the **Conditional** terminal of the **While Loop** with the **Operate** tool and it will turn into a red stop sign. This means that the **While Loop** will now **stop if true** instead of **continuing if true**.



6. Add a **Basic Averaged DC-RMS VI** from the **Analyze>>Waveform Measurements** subpalette of the **Functions** palette and wire the **signal in** input to the **output** of the **AI Acquire Waveform VI**.



- Return to the **Front Panel** and place a **Meter** down. Then, wire the **Meter** terminal on the **Block Diagram** to the **RMS** output of the **Basic Averaged DC-RMS VI**. Wire a **Numeric** constant (Numeric subpalette of the Functions palette) of **10000.00** to the **sample rate** input on the **AI Acquire Waveform VI** and **run** the VI. Note that the VI continuously acquires a waveform and calculates the RMS value of the sine wave. Press the **Stop Button** to stop the VI.

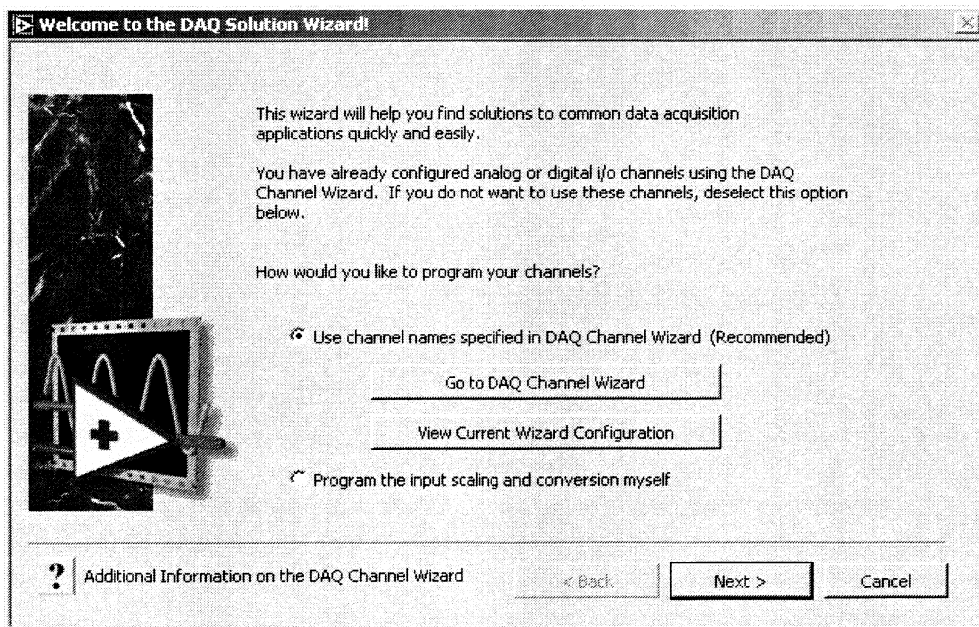


- Save your VI in the **C:\National Instruments\LabVIEW\seminar\customer work** folder and call it **Exercise 7.vi**

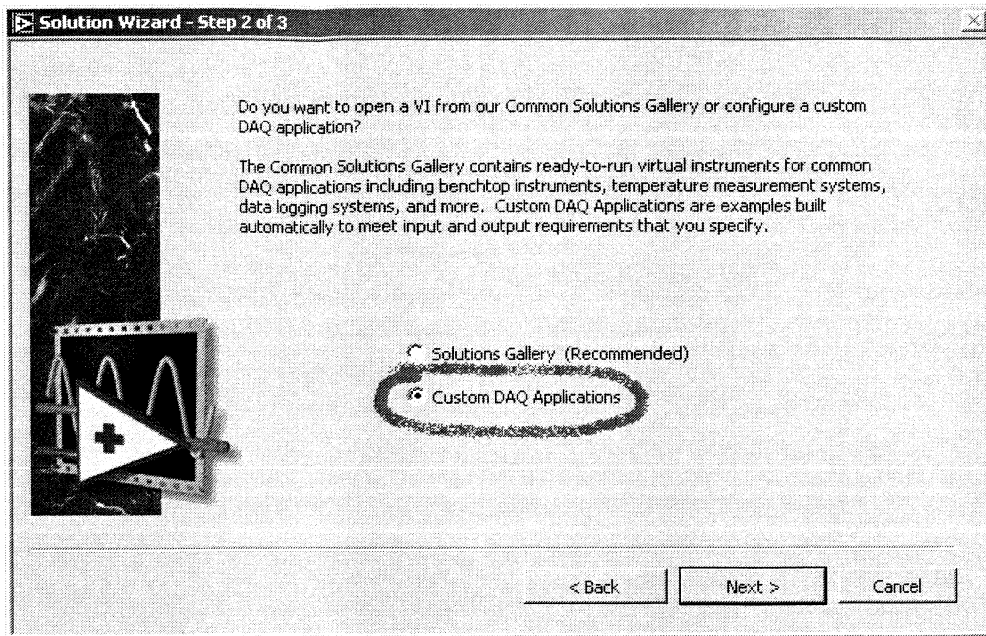
Exercise 8 – *Triggered acquisition with the DAQ Solution Wizard*

In this exercise, you will become familiar with the DAQ Solution Wizard and then use it to automatically generate a VI that acquires a waveform using a digital trigger.

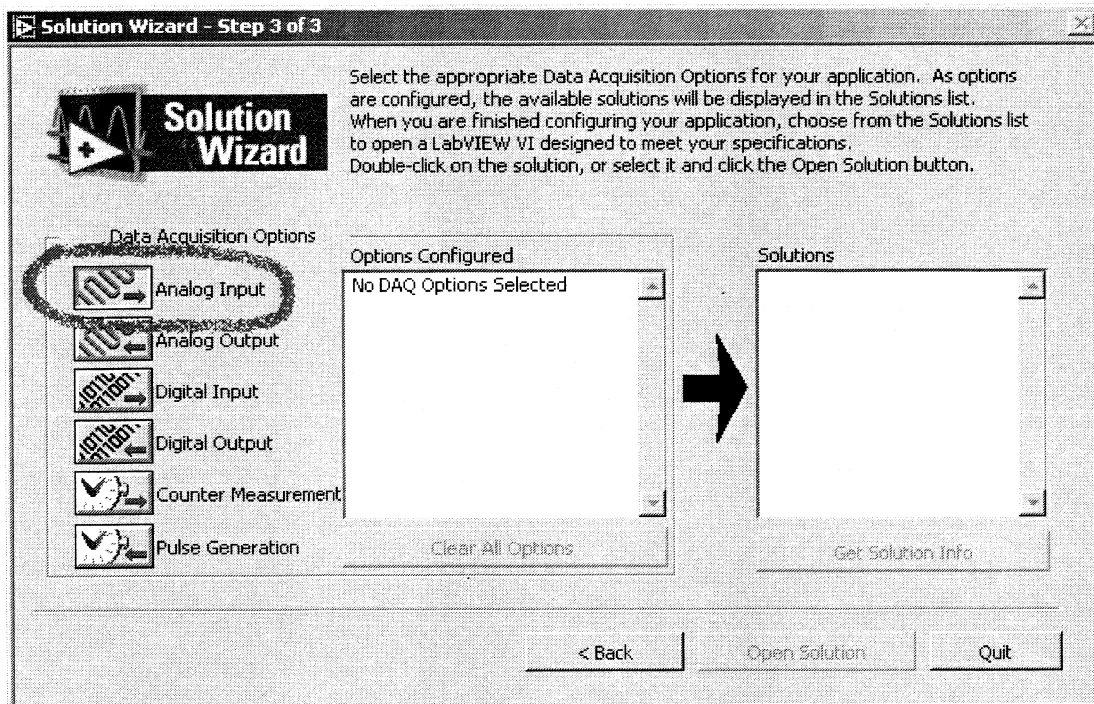
1. Launch the **DAQ Solution Wizard** by choosing **Data Acquisition>>DAQ Solution Wizard...** from the **Tools** menu in LabVIEW. Press **Next>** when the **DAQ Solution Wizard** window appears to use the hardware channels that have been previously configured on your computer.



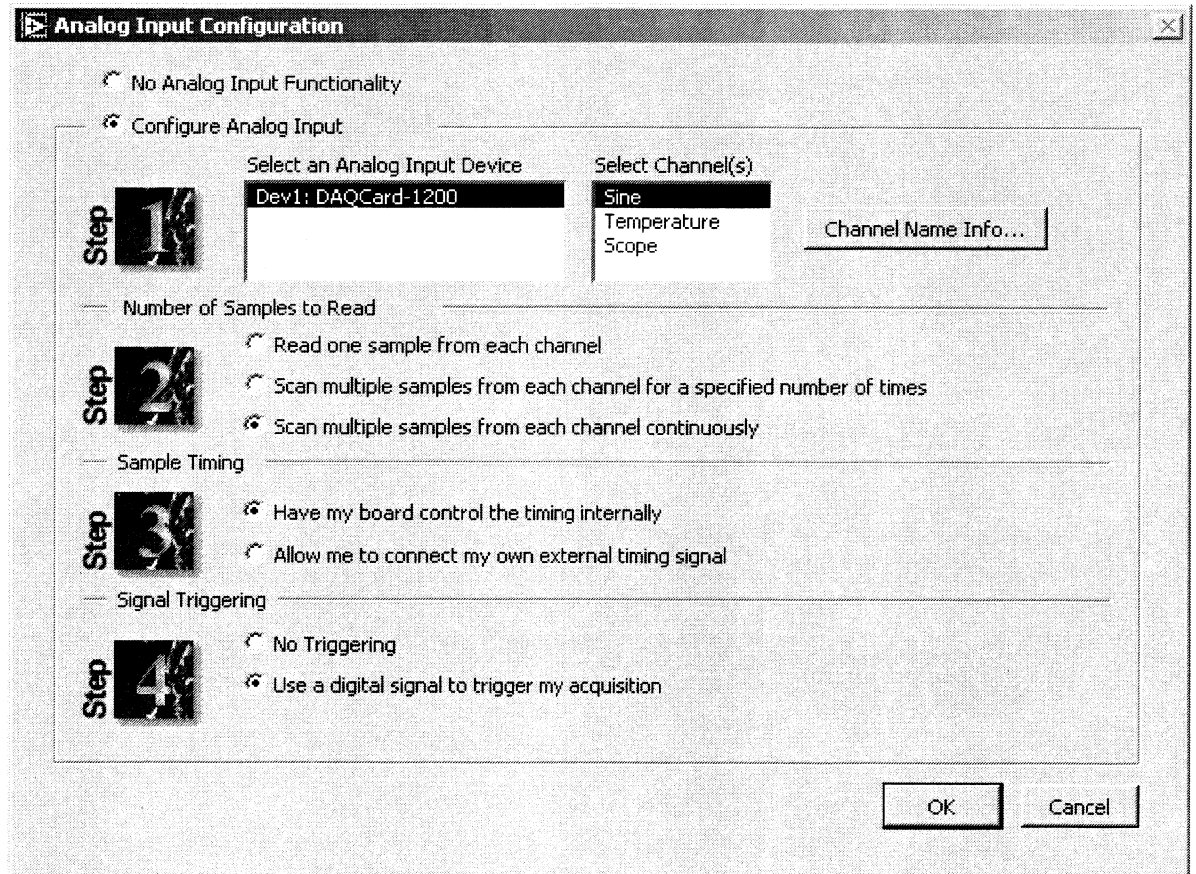
2. Click on the **Custom DAQ Applications** radio button in the next window and press **Next>**.



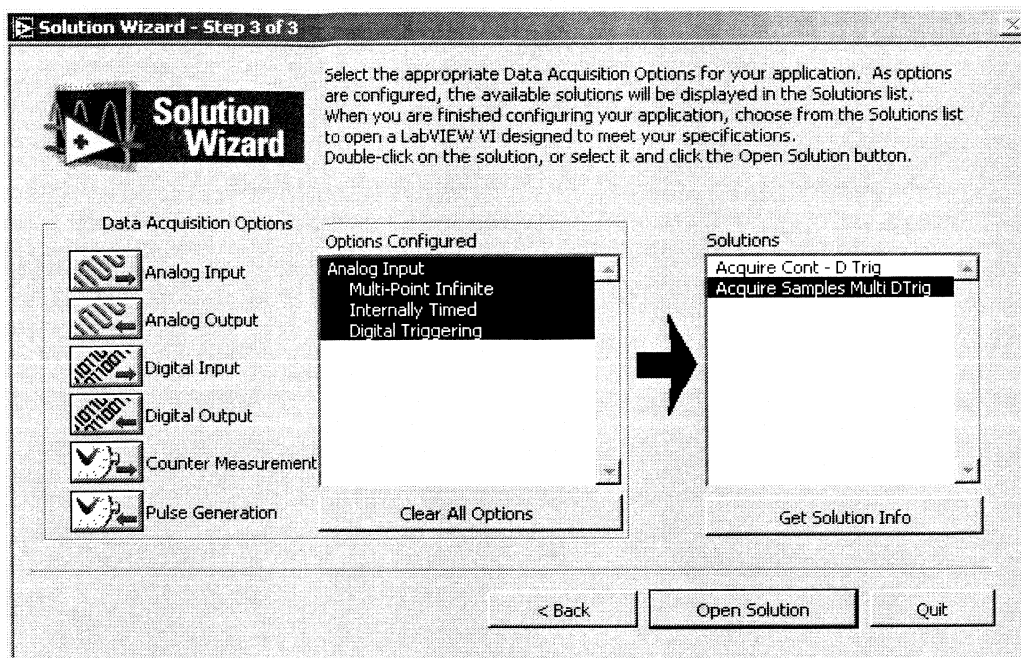
3. Click on **Analog Input** under the **Data Acquisition Options** in the third window that appears.



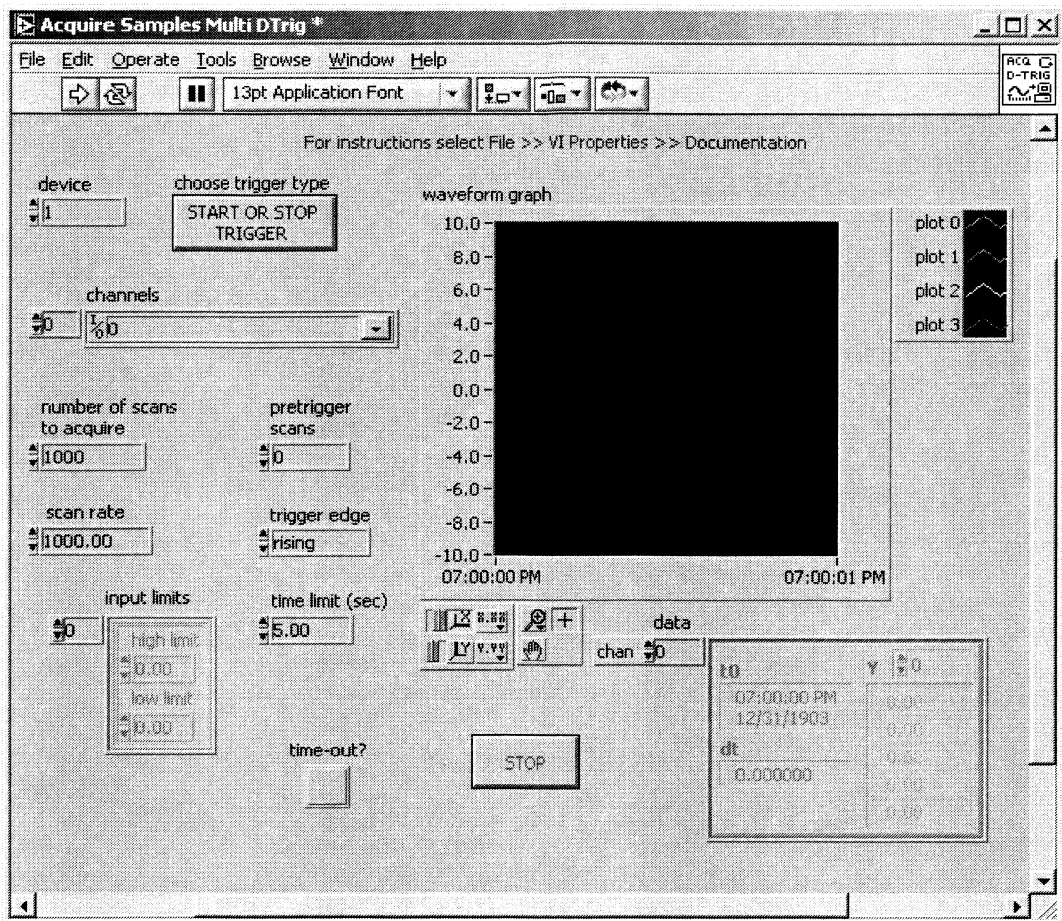
This will launch a new **Analog Input Configuration** window. In the **Step 1** section, select the **Sine** channel. In **Step 2**, click on the third radio button to scan **multiple samples from each channel continuously**. In **Step 3**, select **internal board timing** and choose to use a **digital trigger** in **Step 4**. The correct option selections are shown in the picture below. Press the **OK** button once you have entered these options and a new window will appear.



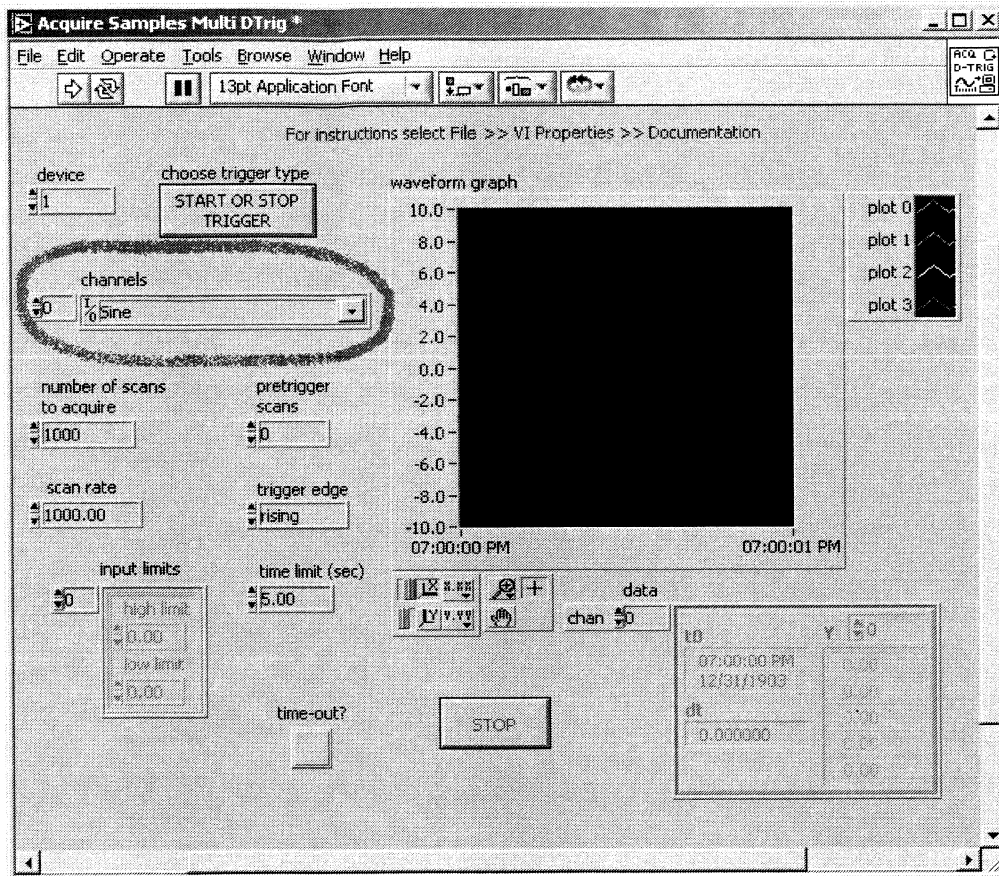
4. Click on the **Acquire Samples Multi DTrig** option in the **Solutions** box and then click the **Open Solution** button. This will open a new LabVIEW application.



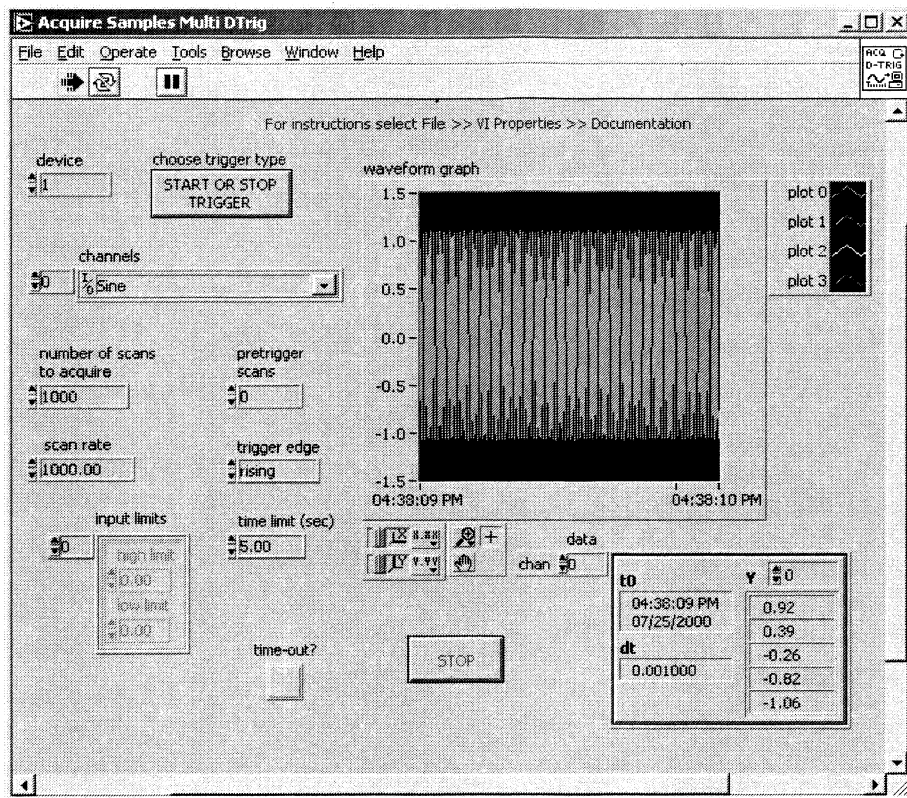
5. The LabVIEW application launched by the DAQ Solution Wizard looks like this:



With the **Operate** tool, click on the **DAQ Channel Name** control under the **channels** label and select the **Sine** channel.



6. **Right click** on the **Y-axis** scale of the **waveform graph** and choose **Autoscale Y**. Press the **run** button to run the VI. This VI acquires a waveform from the **Sine** channel when it receives a **digital trigger**. Push the square **Digital Trigger** button on the DAQ signal accessory to trigger an acquisition. This VI is programmed to acquire data every time you push the trigger. If a trigger is not reached in the time limit specified by the **time limit (sec)** control on the **Front Panel**, a time-out occurs and the **time-out?** LED lights. There is no need to restart the VI after you experience a time-out or acquire a waveform since the VI runs until you press the **Stop** button.



- Return to the **Solution Wizard** window and press **Quit** or **Cancel** to close it. Also **close** the **Acquire Samples Multi DTrig VI** without saving changes as well.



NATIONAL INSTRUMENTS™
LabVIEW™

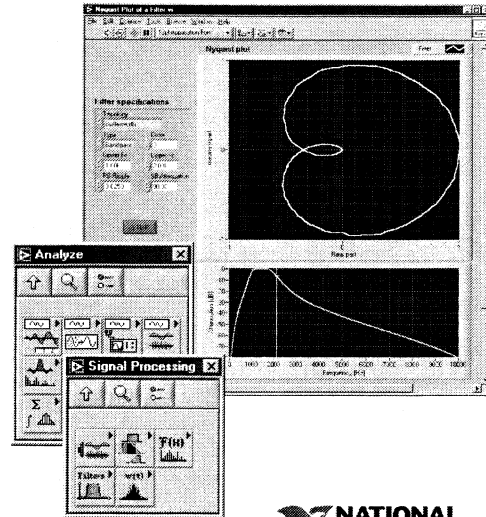


Measurement Analysis

ni.com

Measurement Analysis

- DC/RMS
- Tone detection and distortion
- Frequency analysis
- Signal generation
- Digital filtering
- Waveform monitoring



ni.com

 NATIONAL INSTRUMENTS

LabVIEW also has powerful measurement analysis capabilities that transform your raw data into meaningful measurements. These routines are high-level tools for signal generation, digital filtering, frequency analysis, limit testing, triggering, DC/RMS measurements, THD (Total Harmonic Distortion), and SINAD. Also included are functions for statistics, evaluations, regressions, linear algebra, and even curve fitting. These tools handle many of the details associated with common measurements for you. As a specific example of how these routines address details, the frequency analysis routines go beyond a standard FFT by including common ancillary tasks such as windowing, averaging, and scaling. By using these tools, you can speed up your development by avoiding coding and testing these routines yourself. The LabVIEW measurement analysis VIs work with both acquired and simulated data and include source code, allowing you to examine and modify them for customization, educational purposes, or validation. Not only do these analysis tools deliver powerful measurement functionality to your applications, but you can reconfigure them to fit your particular analysis needs. For instance, by changing the time constant on the DC/RMS measurement, you can trade speed for accuracy. These tools include default parameters for common circumstances, so in many cases these measurements will work “out of the box.”

DC/RMS Measurements

- DC/RMS
 - Averaged (linear or exponential)
 - Non-averaged
 - Trade between speed and accuracy

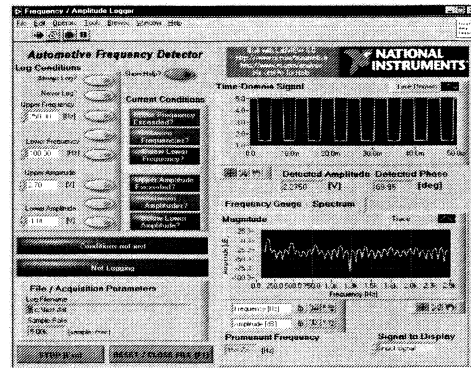
ni.com



LabVIEW has both averaged and non-averaged DC and RMS measurements. A key capability of these measurements is their flexibility—by modifying parameters you trade between speed and accuracy in your measurements. Example parameters include measurement time, averaging type, exponential averaging time constant, and window type.

Tone Detection and Distortion

- Extract prominent tone parameters
 - Frequency
 - Amplitude
 - Phase
- Compensate for spectral leakage
- Harmonic Distortion Analyzer
- SINAD



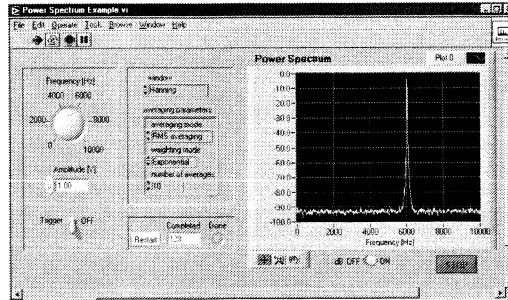
ni.com



Tone detection and distortion measurement VIs extract prominent tone parameters such as frequency, amplitude, and phase from time domain signals and encompass a proprietary algorithm that compensates for inaccuracies due to spectral leakage and the discrete nature of the Discrete Fourier Transform (DFT). The tone detection algorithm is also the basis for several distortion measurement VIs, including Harmonic Distortion Analyzer and SINAD (Signal + Noise + Distortion) tools.

Averaged Frequency Analysis

- Spectral analysis with averaging
 - RMS
 - Vector
 - Peak hold
- Reduce signal fluctuations
- Improve SNR
- Extract signal peaks



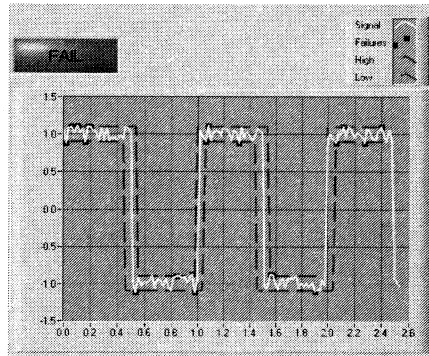
ni.com



The averaged frequency analysis tools in LabVIEW provide powerful spectral analysis with three types of averaging—RMS, vector, and peak hold. By applying averaging, you can reduce signal fluctuations, improve the signal to noise ratio (SNR), and extract spectral peaks.

Waveform Monitoring

- Pass/Fail tests
 - Time dependencies
 - Frequency dependencies
- Limit Testing
- Triggering
- Peak detection



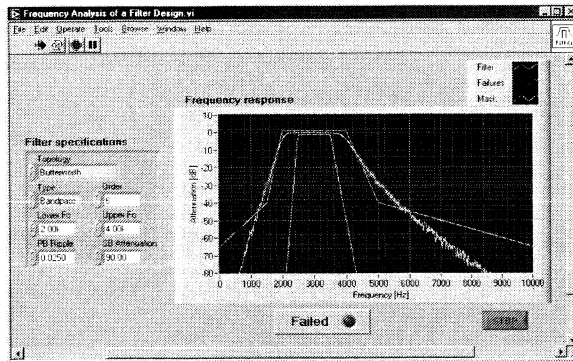
ni.com



To quickly set up pass/fail type tests with results that depend on the time or frequency content of a signal, use the LabVIEW waveform monitoring VIs. These VIs provide limit testing, triggering, and peak detection. Commonly applied in telecom and maintenance applications, the limit testing tools compare a test signal to a predefined mask that specifies high and low values as a function of time or frequency. Pass and fail states are output during a test depending on whether or not a test signal exceeds or goes below these mask values at a particular frequency. To ease the creation of masks, the measurement library includes two methods of specifying a mask: either as a set of arrays or as a function. The triggering capability found in the waveform monitoring VIs will be familiar to those with experience with digital oscilloscopes. It allows you to check for trigger conditions on a waveform and specify the level, hysteresis, and slope.

Digital Filtering

- Alter signal frequency characteristics
- Both IIR and FIR filters



ni.com



Digital filtering allows you to alter or remove unwanted frequency components of signals. In LabVIEW, you can apply both infinite impulse response (IIR) and finite impulse response (FIR) digital filters to signals. These filtering tools give you the flexibility to design a filter in several different ways. For instance, with an IIR filter, one method of choosing a filter is to select an order and a set of cutoff frequencies. For flexibility, you can also have the filter VI automatically choose an order based on passband and stopband widths and gains that you specify.

Signal Generation

- Standard signals
 - Sine
 - Square
 - Triangle, etc.
- Multi-tone signals
- Dual Tone Multi Frequency (DTMF) signals

ni.com

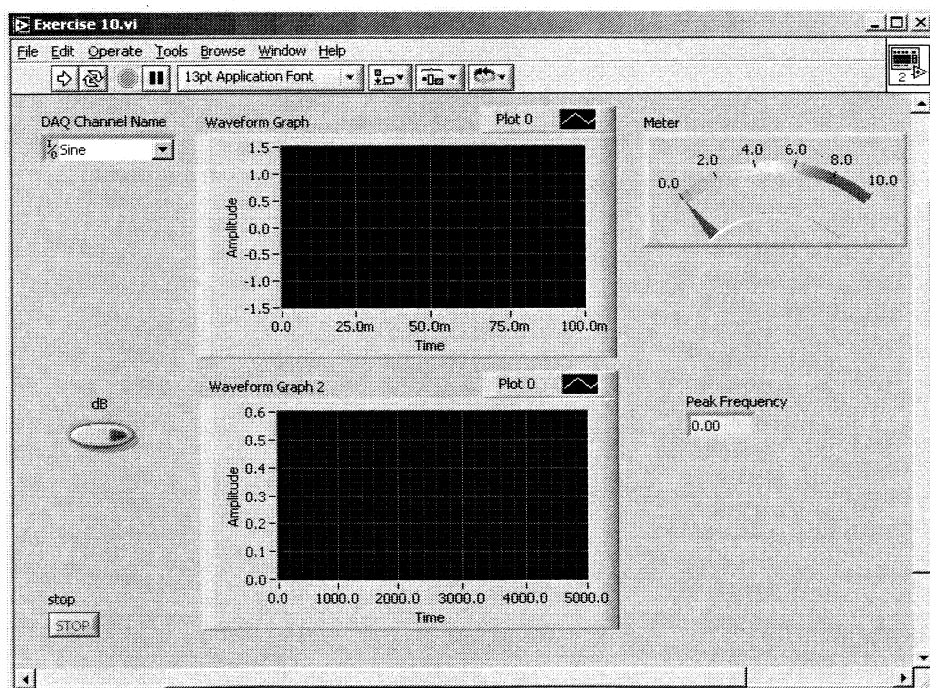


You can generate standard signals (sine, square, triangle, etc.) in LabVIEW. You can also generate multi-tone signals with control over the gain and phase of each component signal. With these types of signals, you can test frequency response in parallel, rather than one at a time, which can shorten your test time. Other applications include generation of Dual Tone Multi Frequency (DTMF) test signals for telephony and implementation of test stations that comply with standards that require multi-tone test.

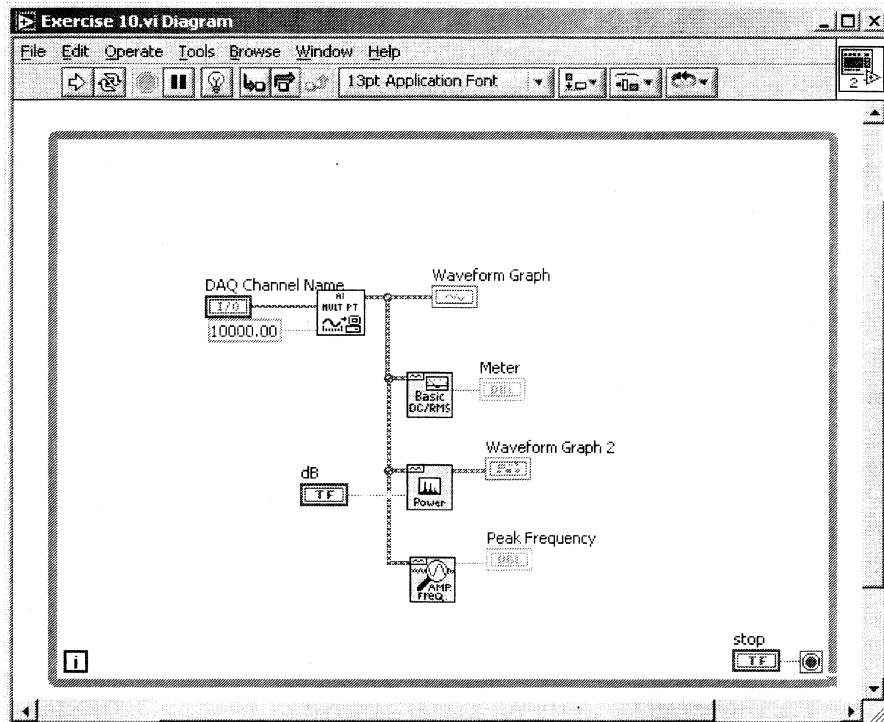
Exercise 9 – *Compute a power spectrum and perform tone detection*

In this exercise, you will build on Exercise 7 in which you acquired a waveform and computed the RMS value. You will add measurement analysis VIs to compute the power spectrum of the acquired signal and then perform tone detection to extract the peak frequency of the signal.

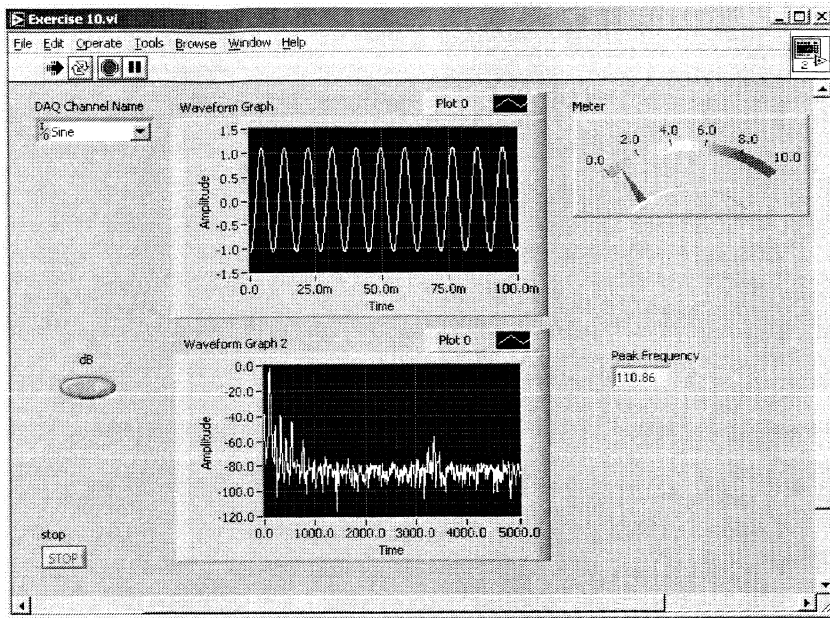
1. Make sure that the green **Analog In Channel 1** connector on the DAQ signal accessory is wired to the **Function Generator Sine** output.
2. Open the **C:\National Instruments\LabVIEW\seminar\customer work** folder and double click on **Exercise 7.vi**
3. Click on the **DAQ Channel Name** control with the **Operate** tool and select the **Sine** channel.
4. Add a **second Waveform Graph** from the Graph subpalette of the Controls palette to the **Front Panel** as shown. The label on the graph will automatically increment to **Waveform Graph 2**. Also add a **Digital Indicator** from the Numeric subpalette and label it **Peak Frequency**. Put a **Push Button** control named **dB** next to **Waveform Graph 2** and turn it on with the **Operate** tool. Make sure that **autoscaling** on **Waveform Graph 2** is enabled by **right clicking** on the **Y-axis** and making sure that **Autoscale Y** is **checked**. Your Front Panel should look like:



5. On the **Block Diagram**, add an **FFT Power Spectrum VI** and an **Extract Single Tone Information VI** (Analyze>>Waveform Measurements subpalette of the Functions palette). Wire their **time signal in** inputs to the **waveform** output of the **AI Acquire Waveform VI**. Wire the **power spectrum** output of the **FFT Power Spectrum VI** to **Waveform Graph 2** and the **detected frequency** output of the **Extract Single Tone Information VI** to the **Peak Frequency** Digital Indicator. Also wire the green **dB** Push Button control terminal to the **dB On** input of the **FFT Power Spectrum VI**. Your Block Diagram should look like the one pictured below:

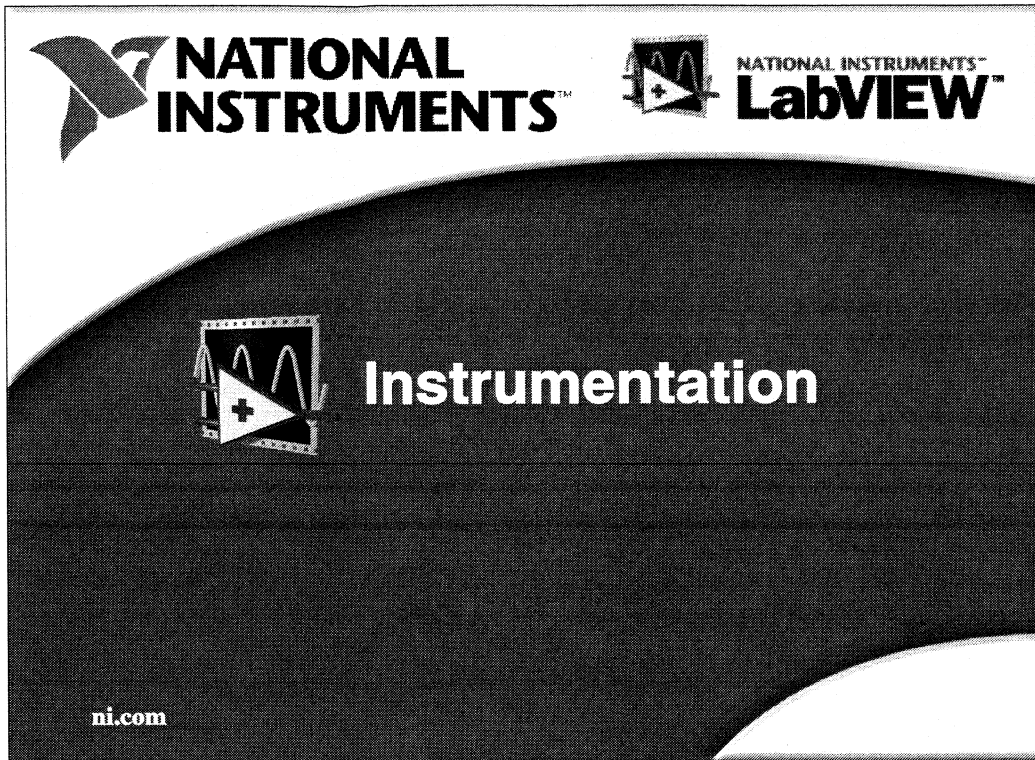


6. Turn the **dB Push Button** **on** and **Run** the VI. You should see something like this:



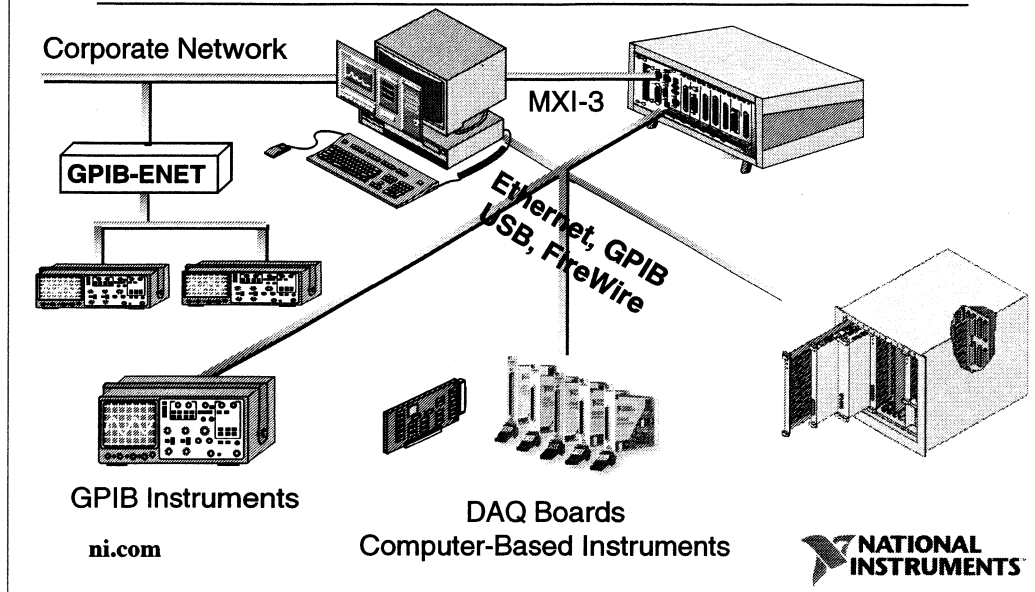
The **Extract Single Tone Information VI** extracts the **peak frequency** component in the **measured signal** and outputs its frequency to the **Peak Frequency Digital Indicator**. **Compare** this value to the **highest peak** displayed in **Waveform Graph 2** and note the correlation. Pushing the **dB Push Button** on the **Front Panel** controls whether the **power spectrum** in **Waveform Graph 2** is displayed in **decibels** or not. Turn the **Frequency Adjust knob** and move the **Frequency Range** switch for the **Function Generator** on the **DAQ signal accessory** to change the frequency **characteristics** of the **measured signal**. Note that the **power spectrum** in **Waveform Graph 2** and the value in the **Peak Frequency Digital Indicator** track any **changes** you make on the **DAQ signal accessory**.

7. **Save** your VI as **Exercise 9.vi** using the **Save As...** option in the **File** menu.



By using LabVIEW to communicate with standard benchtop instruments, you can utilize your existing equipment while still benefiting from virtual instrumentation. You can use the LabVIEW VISA, GPIB, VXI, and Serial libraries to control your instruments and create a single, unified system. In addition, there are more than 1600 GPIB, VXI, serial, and computer-based instrument drivers available free of charge from National Instruments for controlling your instruments.

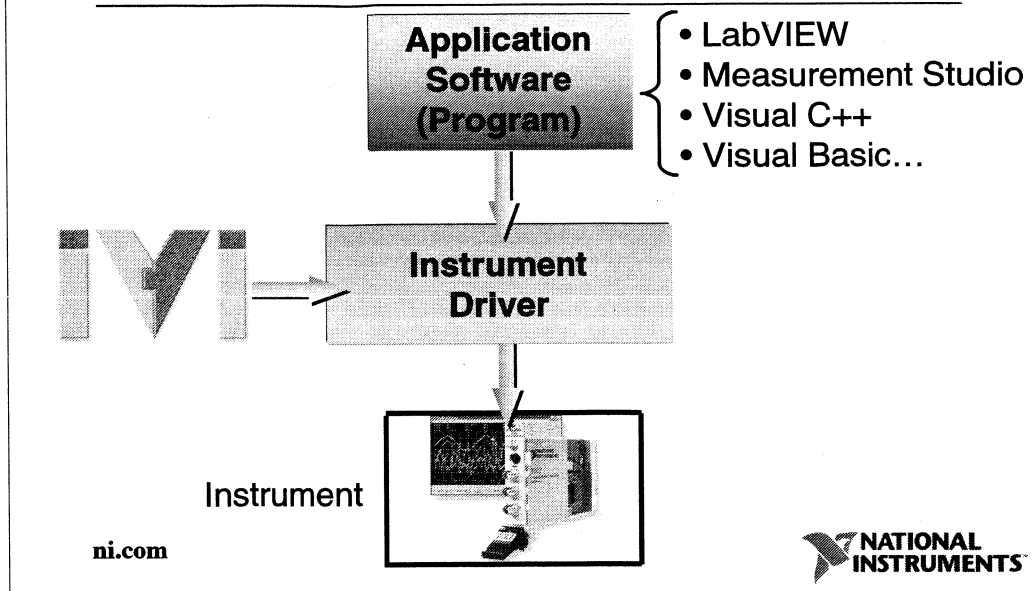
Modern Instrumentation Systems



Modern instrumentation systems are typically hosted by either a desktop PC or a rackmount PC, such as PXI. The majority of them have one or more of the following types of instruments as depicted in this picture:

- GPIB
- Plug-in data acquisition boards
- Computer-based instruments
- VXI instruments
- Networked GPIB instruments

Instrument Drivers and Application Software

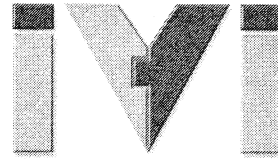


Instrument drivers, which are an important component of test systems today, perform the actual communication and control of the instrument hardware in your system. They give you high-level, easy-to-use programmatic access to the complex measurement capabilities of instruments through an application programming interface (API). Instrument drivers are available in several forms, ranging from functions callable from programming languages like Measurement Studio, Visual C++, and Visual Basic to VIs in LabVIEW. Instrument drivers eliminate the need to learn the complex, low-level programming commands for instruments. National Instruments offers instrument drivers for GPIB, VXI, serial, and computer-based instruments from more than 50 different vendors free of charge. An important category of instrument drivers is IIVI, Interchangeable Virtual Instruments, which make it possible to use the same software interface for different hardware instruments.

By using LabVIEW to communicate with standard benchtop instruments, you can utilize your existing equipment while still benefiting from virtual instrumentation. You can use the LabVIEW VISA, GPIB, VXI, and Serial libraries to control your instruments and create a single, unified system. In addition, there are more than 1600 GPIB, VXI, serial, CAMAC, and computer-based instrument drivers from more than 50 vendors available free of charge from National Instruments for controlling your instruments.

Interchangeable Virtual Instruments

- High performance
 - State-caching
- High productivity
 - Simulation
- Lower maintenance cost
 - Instrument interchangeability
- Two-tier architecture
 - Specific driver (one instrument)
 - Class driver (any instrument)



ni.com



Interchangeable Virtual Instruments (IVI™) is a new instrument driver technology that is defined by a standards body called the IVI Foundation. IVI allows you to communicate with instruments from multiple vendors and through multiple communication buses such as GPIB, serial, and VXI. IVI drivers provide many benefits including state-caching for unparalleled performance, instrument simulation for productivity, and low maintenance costs through instrument interchangeability. State-caching eliminates redundant instrument I/O by only sending to the instrument the commands necessary to incrementally change its state. You can also configure IVI drivers to run in a mode that simulates the actual instrument in software which is especially beneficial for system prototyping. IVI also lowers maintenance costs since you can exchange instruments of the same class regardless of manufacturer or bus connection in the event of instrument failure or calibration.

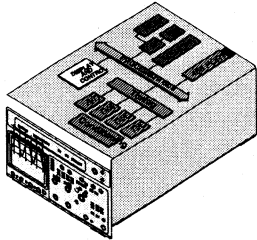
To provide for instrument interchangeability, IVI defines a two-tiered architecture:

- A specific driver tier with drivers that communicate directly with a specific instrument and are intended for use with only one instrument at a time
- A class driver tier with drivers designed to communicate with any instrument within a specific instrument class

Instrument class specifications are defined by the IVI Foundation to provide a consistent programming model. The currently defined classes are DMMs, scopes, switches, arbitrary waveform/function generators, and DC power supplies. Classes that are in development include RF signal generators, spectrum analyzers, power meters, and digital instruments.

Computer-Based Instruments

Stand-Alone Instrument

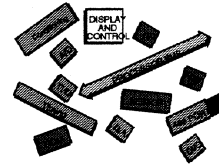
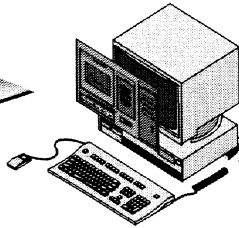


- Register-mapped I/O
- Limited expansion
- Fixed functionality
- External interface

Vendor Defined

ni.com

Computer-Based Instrument



- Memory-mapped I/O
- Network/Internet connectivity
- Online data processing/logging/trending
- Online report generation
- Expandable memory

User Defined

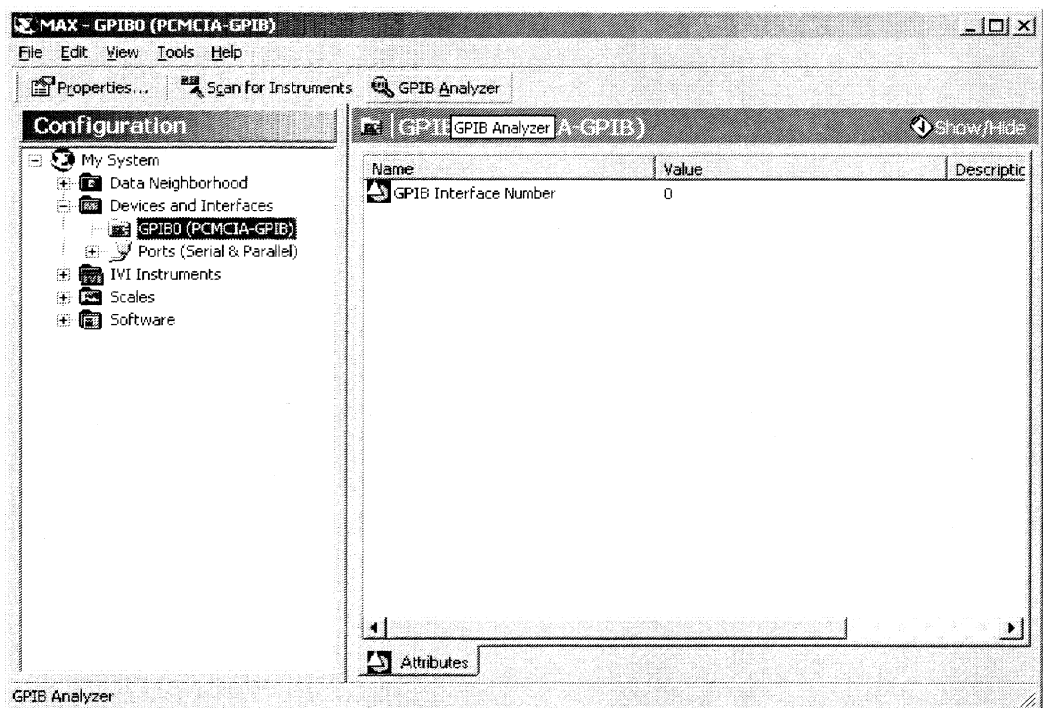


Today, stand-alone instruments are quite common. Most are dedicated, special purpose instruments, but some measurements can be taken significantly faster and for a lot less money with computer-based instruments. Stand-alone instruments have a fixed set of functionality defined entirely by the instrument manufacturer. These instruments pass information back to a host computer over an interface such as GPIB or serial which is typically quite slow. On the other hand, computer-based instruments are user-defined. This means that their complete functionality is determined by how you program them and what kind of analysis and display capabilities you wish to use for your measurements. With a computer-based instrument, your instrument is essentially the PC so you can connect to a network, monitor and/or control over the Internet, and even process or log data online. Computer-based instruments are extremely fast and 100% customizable. To increase measurement throughput, lower the cost of taking measurements, simplify multiple-instrument synchronization, and maximize performance, computer-based instruments are the right choice. The key to higher measurement throughput is speeding up the transfer of data to your computer. The advantage of PCI or PXI/CompactPCI plug-in devices is that they are much faster than GPIB-based instruments for data transfers (132 MB/s versus 8 MB/s). This disparity becomes increasingly evident when performing several I/O operations within a single test or program.

Exercise 10 – Query a GPIB instrument

In this exercise, you will programmatically query the NI Instrument Simulator, which simulates a GPIB instrument, for information about its manufacturer and hardware revision number.

1. Insert the **GPIB** card into the open **socket** on your laptop. Windows will **automatically** install the card for you through its plug-and-play services. Launch **Measurement and Automation Explorer** by **double clicking** on the **icon** on your desktop and **expand** the **Devices and Interfaces** category where you should see the PCMCIA GPIB card listed as shown below:

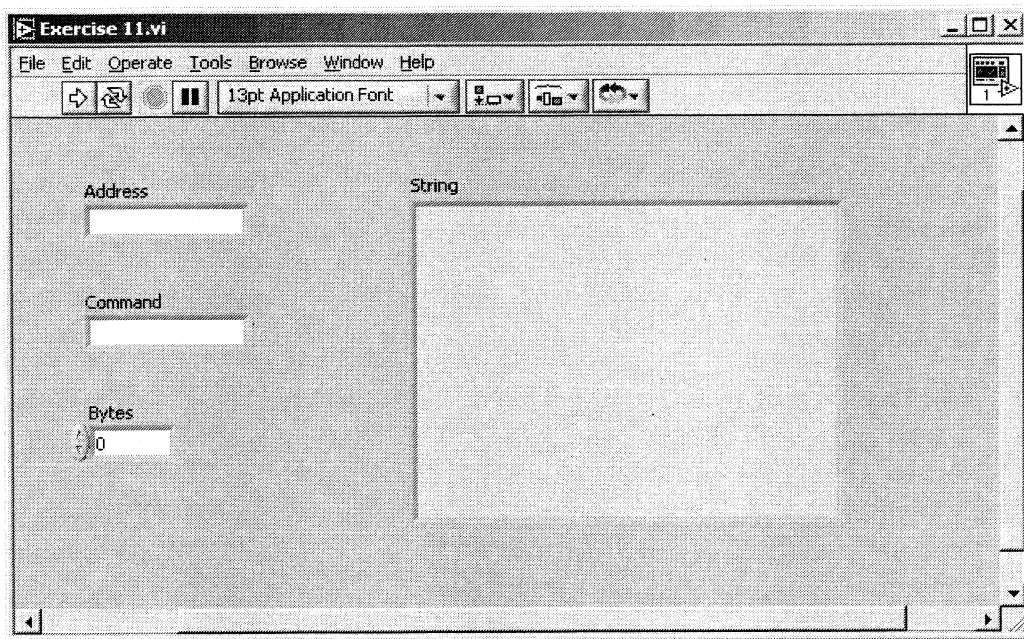


Close Measurement and Automation Explorer. You can now use LabVIEW to communicate with GPIB devices connected to this GPIB interface.

2. Make sure the **power** on the **NI Instrument Simulator** is **off** and then set it right side up on the table with the two **banks of light blue switches facing** you. The **6 green LEDs** should be **visible**. On the **left-most** bank of switches, switch the **first** and the **last** of the switches **down** toward the table and the remaining switches **up** toward the ceiling. You are using the proper switch bank if there is writing above and below the switches on the Instrument Simulator. The two **end** switches should be **closest** to the writing that says **GPIB ADDRESS** and all **other** switches should be nearest the **BAUD RATE** writing on the simulator. Once you have set these switches, turn the **power on**. After a few moments, the two LEDs labeled **POWER** and **READY** will light and you may proceed.

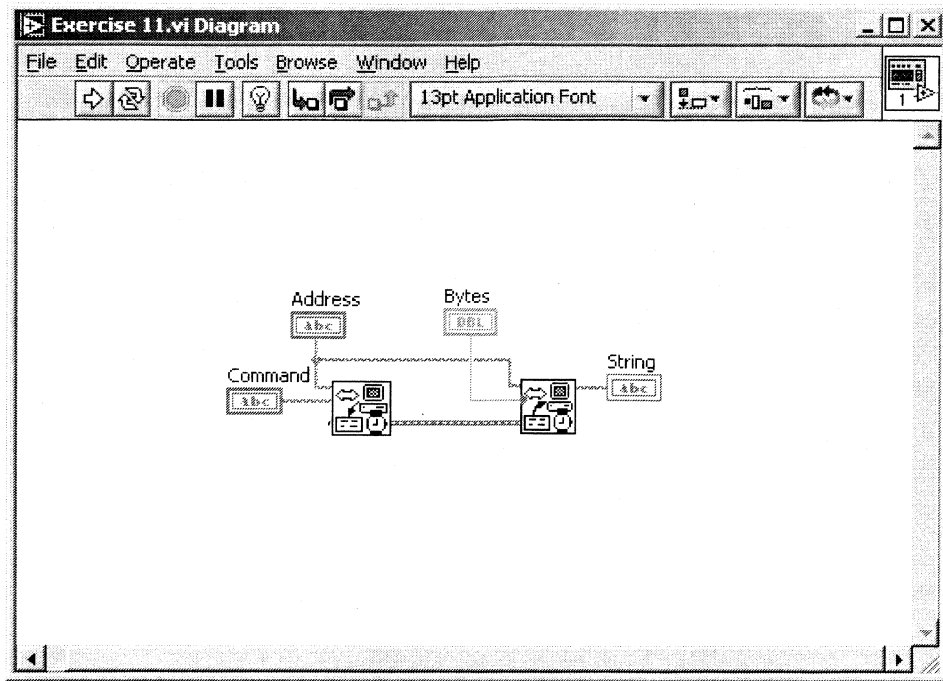
The NI Instrument Simulator simulates the behavior of a stand-alone GPIB or serial instrument. In this case, you will use it to simulate a GPIB instrument.

3. Create a **new VI** in LabVIEW by pressing **Ctrl+N**.
4. Since you can have more than one instrument on a GPIB bus, each instrument is identified by a **unique address**. In this case, the Instrument Simulator has an address of **1**. To control GPIB instruments, you send them **command strings** to perform various operations and then **read** back any requested data. You will now query the Instrument Simulator to return descriptive information about itself, such as the manufacturer's name and hardware revision number. Do this by placing two **String** controls on the **Front Panel** and name them **Address** and **Command**. The String controls can be found in the **String & Path** subpalette of the **Controls** palette. Also place down a **String** indicator from the same subpalette and **resize** it with the **Selection** tool so that it will hold several lines of text. Place a **Digital** control from the **Numeric** subpalette of the **Controls** palette on the **Front Panel** and name it **Bytes**. Your Front Panel should look like this:

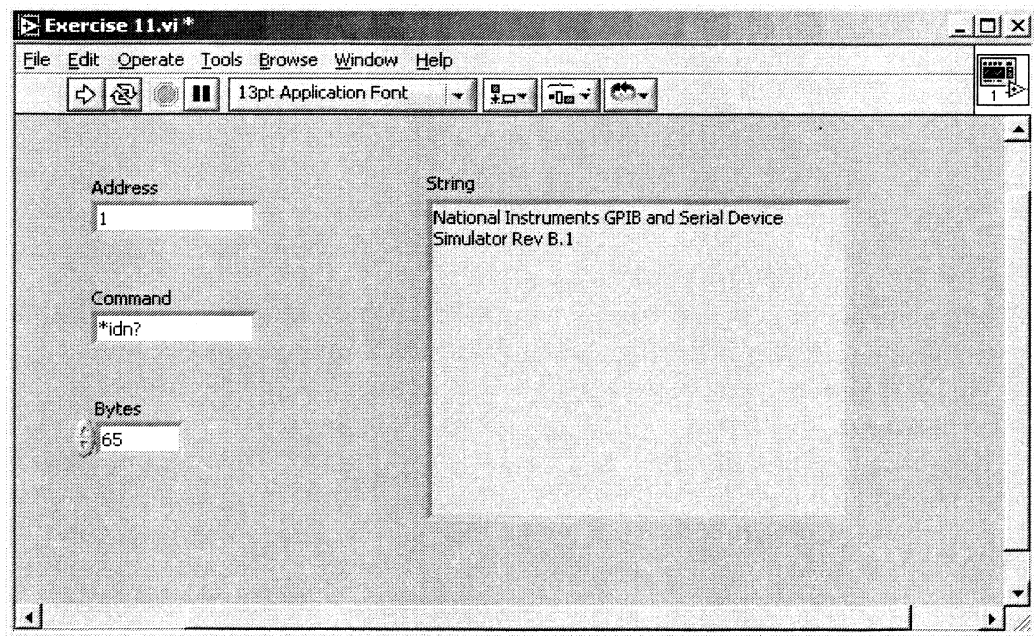


5. On the **Block Diagram**, place down a **GPIB Write** and a **GPIB Read** function (Instrument I/O>>GPIB subpalette of the Functions palette). Be sure to place the **GPIB Write** on the **left** of the **GPIB Read**. Wire the pink **Address** control terminal to the **address string** inputs on both the **GPIB Write** and the **GPIB Read** functions. Wire the **Command** string input to the **data** input on the **GPIB Write** and the **data** output of the **GPIB Read** to the **String** indicator. Once you have done that, wire the **Bytes** Digital Control terminal to the **byte count** input on the **GPIB Read** function. Using the principle of **dataflow**, wire the **error out** output of the **GPIB Write** to the **error in** input on the **GPIB Read** function to

guarantee that the write executes first. The completed **Block Diagram** should look like:



- Using either the **Operate** or the **Labeling** tool on the **Front Panel**, enter **1** for the address, ***idn?** for the command and **65** for the number of bytes to read back from the instrument. **Run** the VI and the **NI Instrument Simulator** will return **identification** information inside the **String** indicator on the **Front Panel**. You should see the following after running the VI:

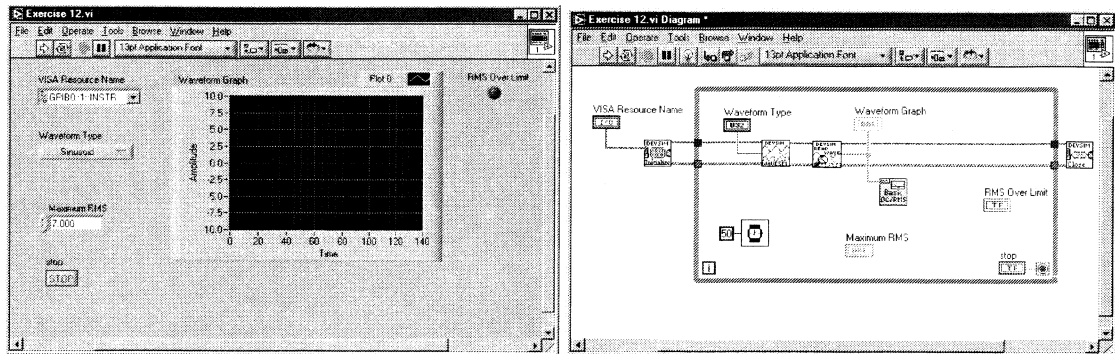


7. **Save** your VI as **Exercise 10.vi** in the **C:\National Instruments\LabVIEW\seminar\customer work** folder.

Exercise 11 – Use an Instrument Driver

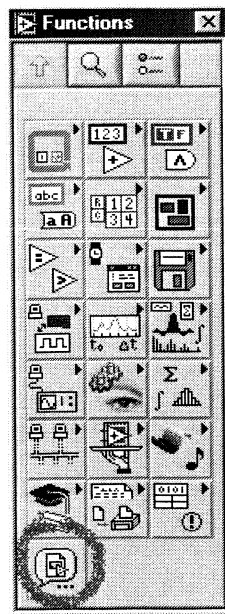
In this exercise, you will learn about the architecture of an instrument driver. You will also modify a ready-to-run example that controls the NI Instrument Simulator in order to be able to analyze the RMS value of acquired waveforms.

1. From the **C:\National Instruments\LabVIEW\seminar\solutions** folder, open **Exercise 11.vi** and open the **Block Diagram**.

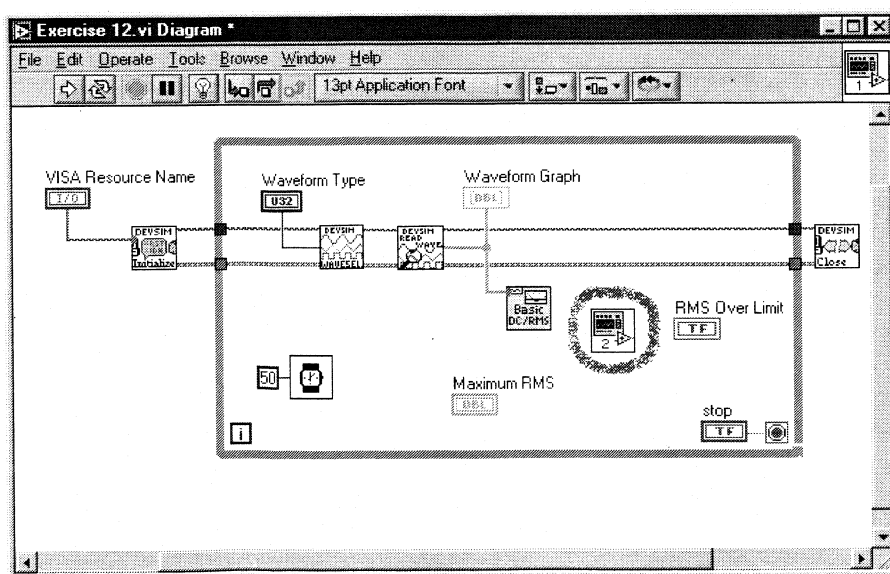


2. **Exercise 11.vi** is an example of a simplified **instrument driver** that has been created to **control** the **NI Instrument Simulator**. An instrument driver performs the **communication** and **control** of the **instruments** in your system through a **high-level programming interface**. Generally, an instrument driver gives you programmatic access to the **initialization**, **configuration**, and **operation** of an instrument. Study the **Block Diagram** for **Exercise 11.vi**. You will see that the first thing this VI does is **initialize** the NI Instrument Simulator. Then, it **configures** the simulator to **output** a specified **waveform type**. Once the output waveform has been selected, the waveform is **read** from the simulator and **displayed** in a graph. Both the waveform **configuration** and **collection** are inside a **While Loop** so that you can change waveform types and collect data from the NI Instrument Simulator **continuously**.
3. **Run** the VI. Notice that **changing the waveform type** results in a **different** signal being acquired from the Instrument Simulator. There may be a slight delay between the time you select a different waveform type and the simulator outputs the new waveform. The **subVIs** in this VI are built using **VISA** functions. VISA stands for **Virtual Instrument Software Architecture**. **VISA** is a **standard interface** for controlling **GPIO, VXI, serial**, and other types of **instruments**. The **VISA Resource Name** on the **Front Panel** is simply the VISA designation for the **NI Instrument Simulator**. The **Maximum RMS** and **RMS Over Limit** Front Panel objects are **not wired** to anything yet so they will **not affect** the behavior of the VI. However, they will be used in the following steps.
4. Quite often, instrument control applications involve some sort of **pass/fail** testing. In this case, you will calculate the **RMS value** of the acquired **waveform** and

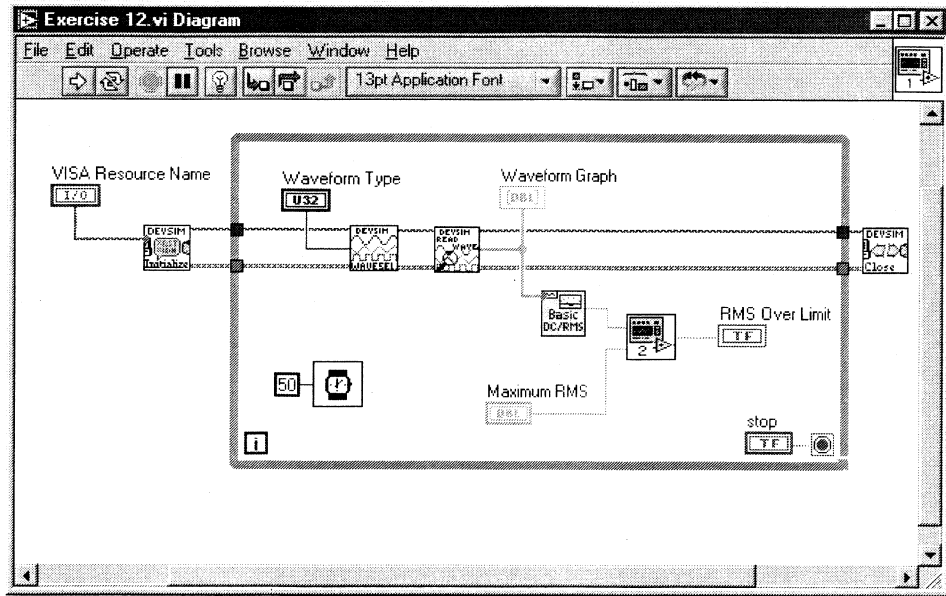
then determine whether or not it **exceeds** a specified **limit** by using **subVI.vi** which you created in an earlier exercise. To incorporate this type of pass/fail test into the VI, click on **Select a VI...** from the **Functions** palette, located in the **lower left corner** as shown below:



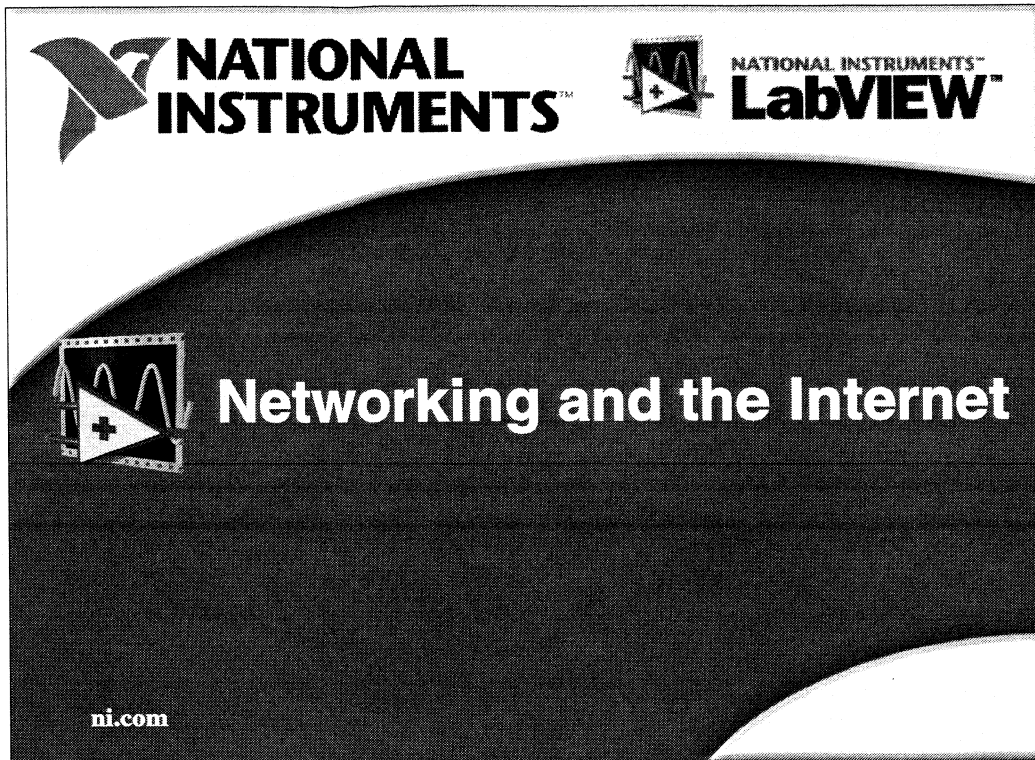
5. You will see a **file dialog** asking you for the VI to open. Browse to the **C:\National Instruments\LabVIEW\seminar\customer work** folder and **double click** on **subVI.vi** or **select it and press Open**. **Drag** the **subVI icon** that appears to the **Block Diagram** next to the **Basic Averaged DC-RMS VI** as shown below:



6. Wire the **RMS Value** output of the **Basic Averaged DC-RMS VI** to the **Input** terminal on the **subVI VI** and the **Maximum RMS Digital Control** terminal to the **Limit** input on the **subVI VI**. Then wire the **Over Limit** output of the **subVI VI** to the **RMS Over Limit LED** terminal so that your Block Diagram looks like this:



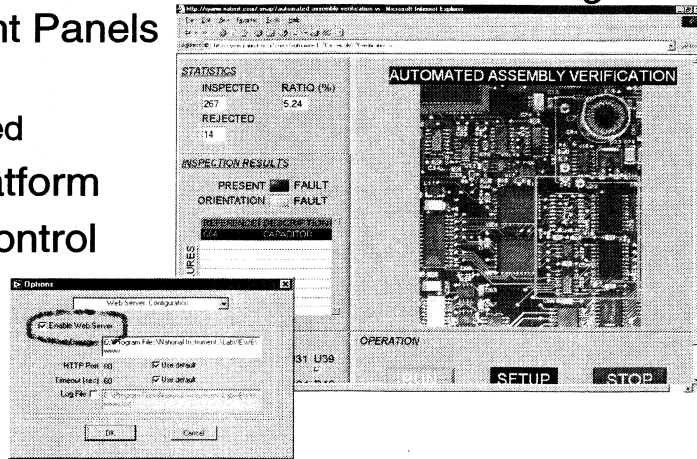
7. Return to the **Front Panel** by pressing **Ctrl+E** and **run** the VI. Once again, the NI Instrument Simulator will output waveforms that are displayed in the graph. The VI then **calculates** the **RMS** value of the acquired **waveform** and **compares** it to the **value** of the **Maximum RMS Digital Control**. If the **actual RMS** value **exceeds** this **maximum**, then the **RMS Over Limit LED lights**. Run the RMS limit test on a number of **different waveform types**. When you are finished running the VI, **close** it **without** saving changes.



There are obvious benefits to networking computers together, especially when you want to share information or distribute execution. Let's look now at some of the Internet and networking capabilities of LabVIEW.

Web Server

- Publish HTML documents and embed images of VI Front Panels
 - Static
 - Animated
- Cross-platform
- Access control

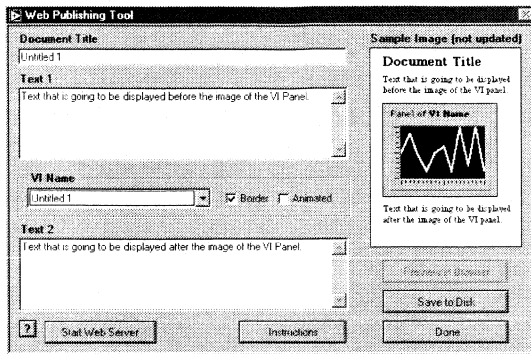


ni.com



For reporting or monitoring purposes, you may wish to publish images of your LabVIEW application's user interface to the Web. This is a trivial task with the built-in LabVIEW Web Server which publishes images of your VI front panels directly to the Web. The nice thing about the Web Server is that it publishes data in a format that is available on most common computing platforms. This means that you can publish reports from a machine with LabVIEW running on Linux, for example, and a client machine running Mac OS has the ability to display the generated reports. The built-in LabVIEW Web Server provides security features by allowing you to selectively block or grant access to various client machines.

Web Publishing Tool



- Borders and titles
- Text
- Front Panel images
 - Static
 - Animated

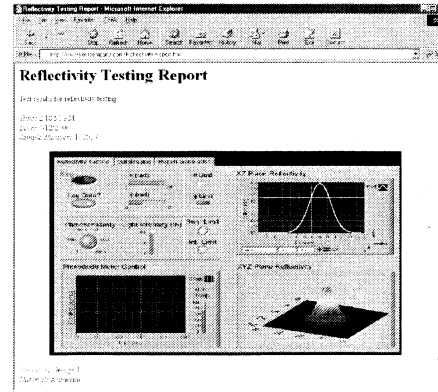
ni.com



If you need to not only publish images of your VI Front Panels, but also add additional embellishments, such as titles, text, and borders, then you can use the Web Publishing Tool. This tool allows you to create a new HTML document and embed static or animated images of the Front Panel in it or embed images of the front panel in an existing HTML document.

HTML Report Generation

- Embedded VI Front Panels
- Text
- Tables
- Graphics
- Bulleted lists
- Hypertext links



ni.com



If you need even more flexibility than the built-in Web Server and Web Publishing Tool already provide, you can use the LabVIEW report generation VIs to create HTML reports. Not only can you create report text and embed images of VI Front Panels in your reports, but you can also insert tables, bulleted lists, graphics, and even hypertext links.

Internet Developers Toolkit

- Advanced Web server
- E-mail
- FTP
- CGI
- Telnet

ni.com

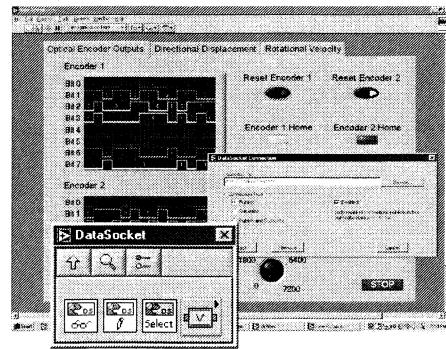


LabVIEW development systems have built-in Internet functionality as we have already explained. If you have more advanced Internet requirements, then you can use the Internet Developers Toolkit. The toolkit's Web server goes beyond the built-in LabVIEW Web server by providing more control over security levels. With the e-mail, FTP, Telnet, and CGI tools that are part of this toolkit, you can automatically send e-mails, send files using FTP, or dynamically decide upon content to present across the Web using Common Gateway Interface (CGI) programs from your LabVIEW applications.

DataSocket

- Optimized for streaming data
- Exchange data between one or more network computers
- URL-based
- Three participants
 - Publishers
 - DataSocket server
 - Subscribers
- Security
- Language and platform independent

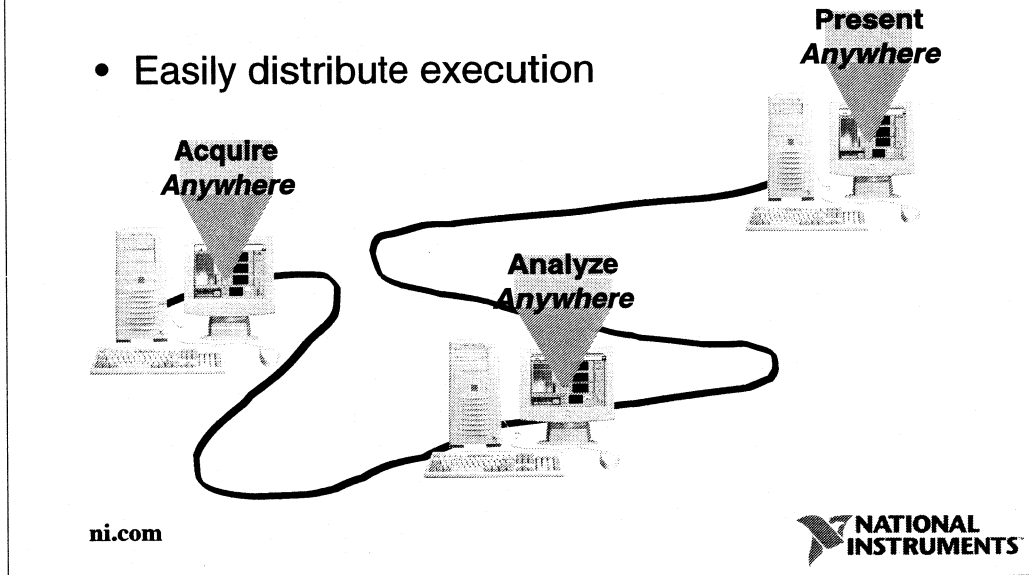
ni.com



One technology resident in LabVIEW that greatly facilitates data sharing is DataSocket. DataSocket is extremely useful because it simplifies streaming data between different applications on one computer or between a host of computers on a network. Although a variety of different technologies exist today to share data between applications, such as TCP/IP and DDE, most of these tools are not targeted specifically and optimized for streaming data like DataSocket. DataSocket applications require three participants—a publisher, the DataSocket server, and a subscriber. A publishing application uses the DataSocket API to write data to the DataSocket server. The publisher writes the data to an identifier, or tag, formatted like a Uniform Resource Locator (URL), which looks just like a Web address. A subscribing application then uses the same DataSocket API to read data from that URL on the server. Both the publishing and the subscribing applications are “clients” of the DataSocket server. These three participants can reside on the same machine or they can all run on completely different computers. The ability to run these DataSocket components on separate machines improves performance and provides security by isolating network connections from the measurement application. The DataSocket server can broadcast live measurement data at high rates across your network or across the Internet to several remote clients concurrently. It simplifies your measurement application by automatically managing the connections to clients. The DataSocket Server Manager gives you the ability to configure and manage the security of your applications by allowing you to set options such as access permission, number of connections, and predefined data items stored in the DataSocket server. Another major advantage of DataSocket is that it is both platform and language independent. This means that you can use a variety of programming languages to create both publishing and subscribing applications on many different operating systems. You can program DataSocket applications with LabVIEW, National Instruments Measurement Studio, Visual Basic, or simply use a standard Web browser. Another benefit of DataSocket is that it can not only connect to URLs created by DataSocket clients, but it can also directly connect to files using FTP or HTTP. Please note that the DataSocket server is only available on Windows platforms. In LabVIEW, you can publish and subscribe data from any user interface control with no programming and literally two mouse clicks. For added flexibility, you can programmatically post or retrieve data from the DataSocket server as well.

Remote VI Call

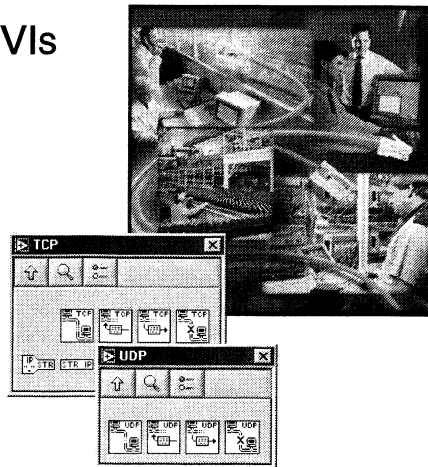
- Easily distribute execution



Distributed execution allows you to execute various tasks in your measurement system across several different computers so that no one machine becomes stressed under the heavy load of the required operations, thus improving system performance. Additionally, you can coordinate measurement operations across many computers into a single system. Also, you can acquire and reduce data at the computer taking the measurement which helps reduce network congestion. One common application of distributed execution is remote control. You can split execution so that the measurement system can reside on a remote machine and the controlling portion resides on a computer in a more comfortable or accessible location. Distributing execution provides some definite advantages as you can see. A typical measurement solution consists of an acquisition element, an analysis component, and some type of presentation capability. With distributed execution, you can break up these tasks and run them on different computers. No longer is it acquire, analyze, and present—it's acquire anywhere, analyze anywhere, and present anywhere. LabVIEW has a feature called the remote VI call that lets you simply select the LabVIEW code you wish to execute remotely. For text-based languages, like C or C++, this technique is referred to as a remote procedure call (RPC) or remote method invocation (RMI) in Java. Both of these techniques can be difficult to implement because they require specific knowledge of your network environment and require several programming steps. Many consider RPC an expert programming topic. However, by combining the remote VI call with LabVIEW's hierarchical nature, you can distribute as little or as much of your program as you wish with a minimal amount of programming. Not only can you distribute the acquisition of data, but also the analysis, decision logic, or any part of your program that can be encapsulated in a LabVIEW VI. Using remote VI calls, you can easily execute any VI on any computer. With a remote VI call, you can even execute a VI on a platform altogether different than the one on which the calling application runs. For instance, you may have an operation designed for a special configuration on a Linux machine that you would like to call from your main program running on Windows 2000. If you use the remote VI call, it makes no difference what processor and operating system are resident on the local and remote machines as long as LabVIEW exists on both.

TCP and UDP

- Both TCP and UDP VIs available



ni.com

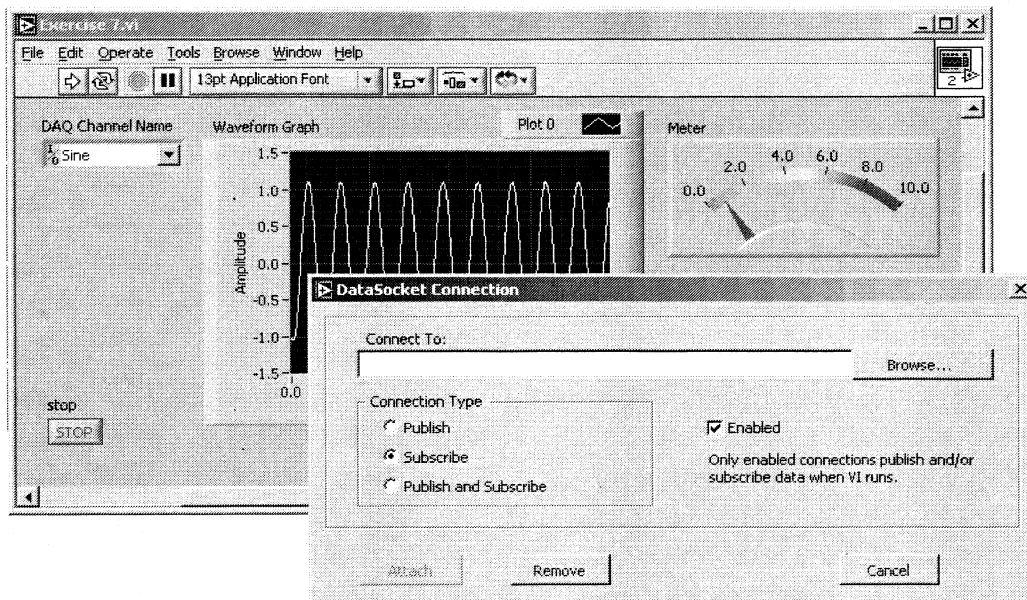


For users wishing to programmatically control the networking side of their applications, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) VIs are available in LabVIEW. TCP is a reliable, connection-based protocol which provides error detection and ensures that data arrives in order and without duplication. For these reasons, TCP is usually the best choice for network applications. Although UDP can give higher performance than TCP and does not require a connection, it does not guarantee data delivery.

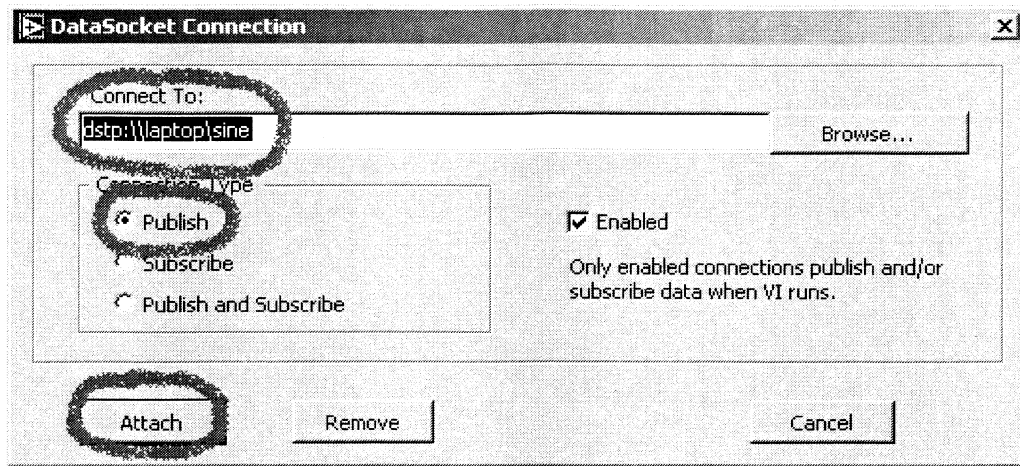
Exercise 12 – *Publish and subscribe to data using DataSocket*

In this exercise, you will use the VI you built in Exercise 7 to acquire waveform data. You will then publish that data directly from a graph on the Front Panel to a graph in a completely different VI with no programming.

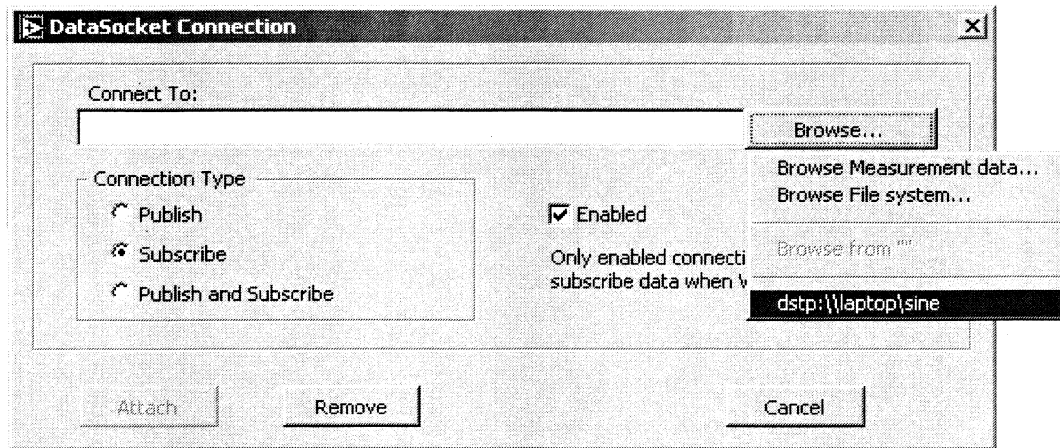
1. In the **C:\National Instruments\LabVIEW\seminar\customer work** folder, open **Exercise 7.vi**. Be sure that you select the **Sine** channel in the **DAQ Channel Name** control.
2. **Right click** on the **Waveform Graph** and choose **Data Operations>> DataSocket Connection...** from the shortcut menu. A DataSocket Connection window will appear as shown below.



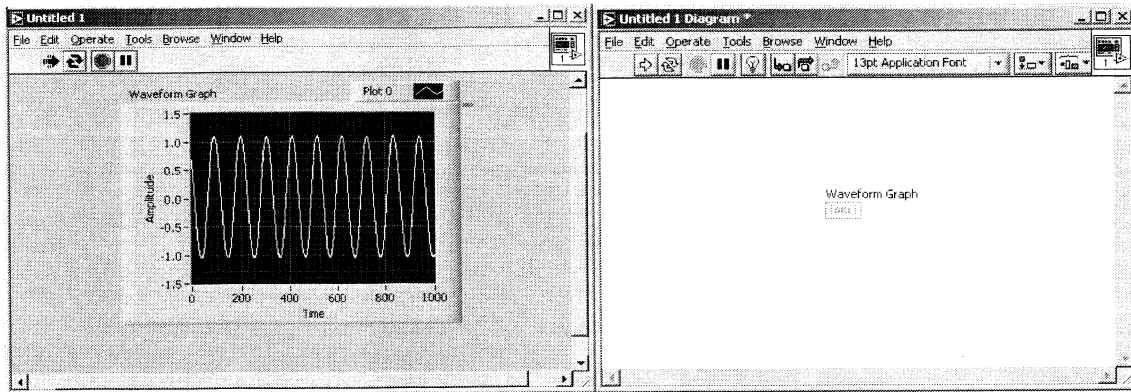
3. In the **DataSocket Connection** window, enter **dstp://laptop/sine** below the **Connect To:** heading, **click** on the **Publish** radio button, and then **press** the **Attach** button. This will **close** the **DataSocket Connection** window and set up a publishing connection named **Sine** between the **Waveform Graph** and the **DataSocket server**. **Run** the VI. DataSocket **URLs** are constructed simply by adding **dstp://** before your **computer name** followed by the **data item** name you give to the data you would like to publish. In this case, your machine name is **laptop** and you have given the data you wish to publish from the graph the data item name **sine**.



4. Open a new VI by pressing **Ctrl+N**. Place a **Waveform Graph** on the **Front Panel** (Graph subpalette of the Controls palette). **Right click** on the **Waveform Graph** and choose **Data Operations>>DataSocket Connection**. In the **DataSocket Connection** window that appears, click on the **Browse** button and select **dstp://laptop/sine** from the list. Then click the **Subscribe** radio button and press **Attach**. The DataSocket Connection window will then disappear.



5. Press the **Continuous Run** button and you will see **data** displayed in the **Waveform Graph** of the new VI. Remember that you did not have to do any programming to subscribe to the data! Notice that there is a single orange Waveform Graph terminal on the Block Diagram with nothing wired to it. The DataSocket Connection feature allows you to publish and subscribe to measurements with no programming.

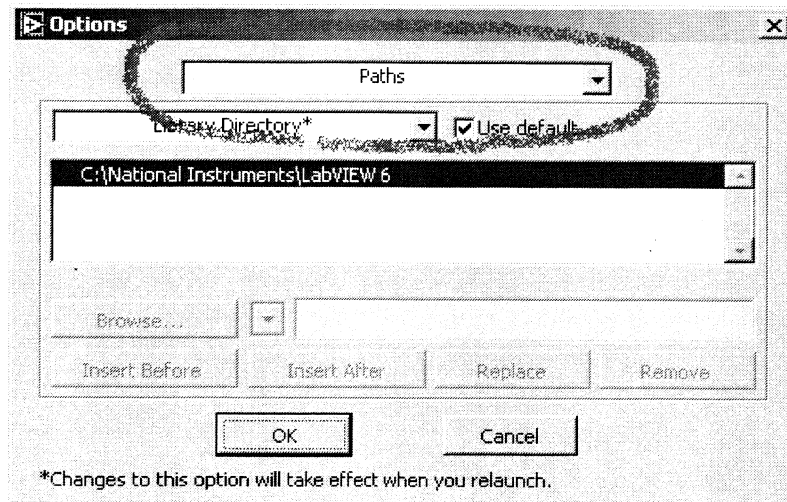


6. Close both **Exercise 7.vi** and the new **VI** you created **without** saving changes.

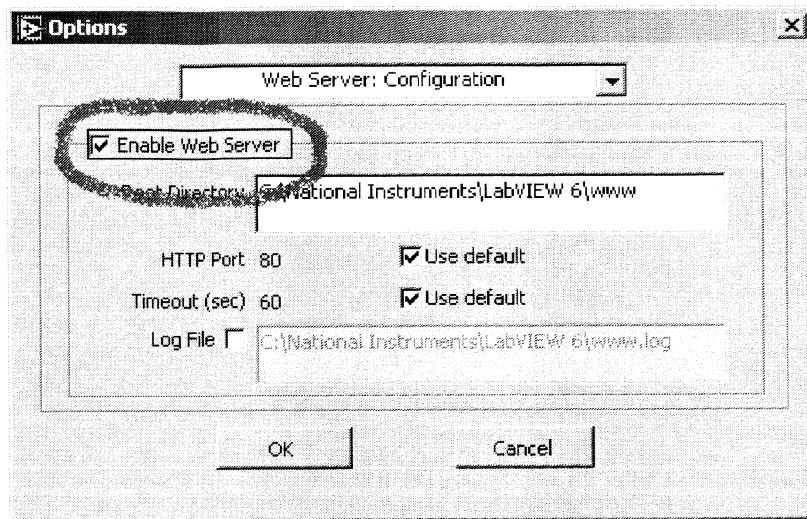
Exercise 13 – *Display an image of a Front Panel in a Web browser*

In this exercise, you will embed an image of the VI Front Panel you developed in Exercise 7 inside a Web browser using the built-in LabVIEW Web server.

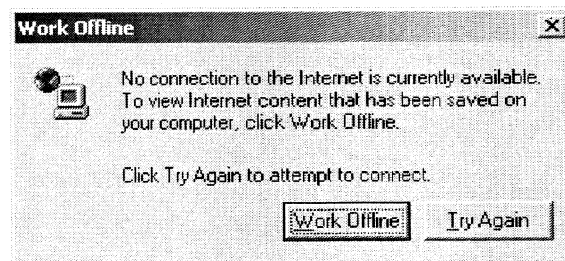
1. Open **C:\National Instruments\LabVIEW\seminar\customer work** and run **Exercise 7.vi**. Be sure that you select **Sine** in the **DAQ Channel Name** control **before** running the VI.
2. From the **Tools** menu, choose **Options...** which will launch a new **Options** window.



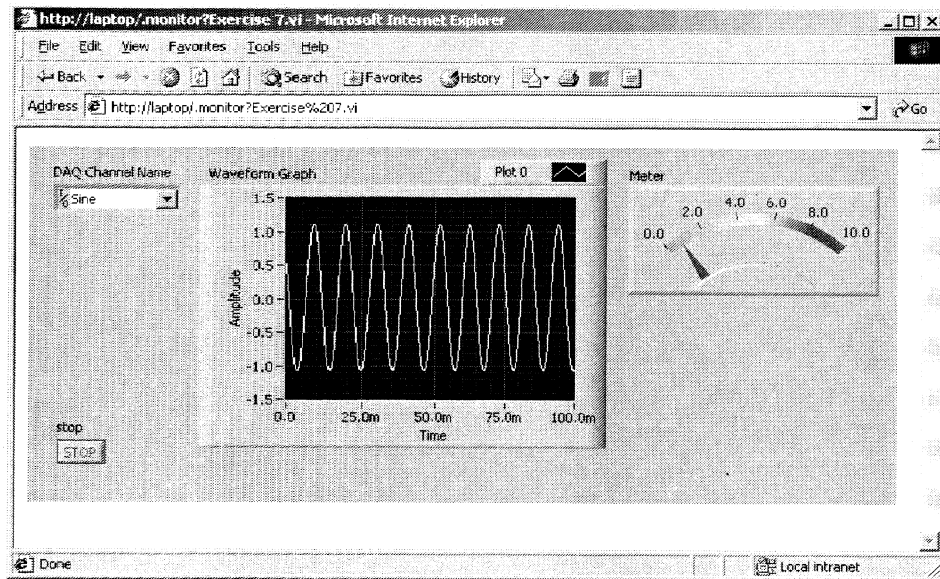
In the **Web Server: Configuration** options (accessed from the pull-down ring at the top of the window), **enable** the built-in Web server by clicking on the **Enable Web Server** check box as shown on the following page.



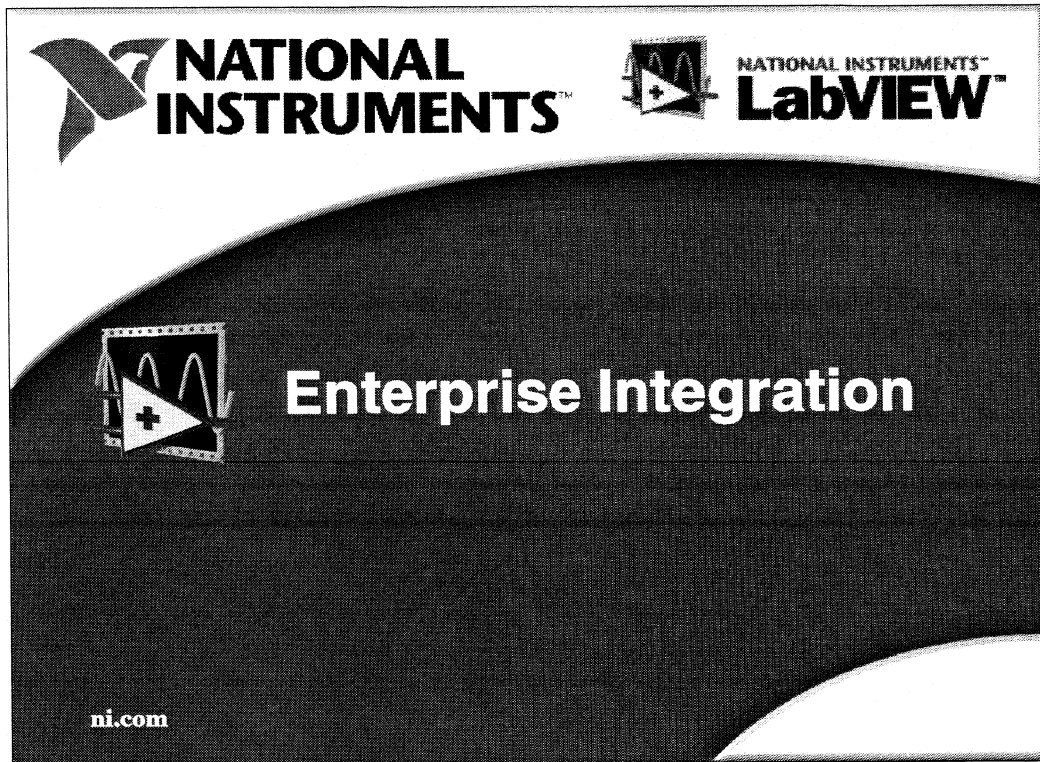
3. Launch **Internet Explorer** by choosing **Start>>Programs>>Internet Explorer** from the **Start Menu**. If you see a dialog box like the one pictured below, click on **Try Again**.



4. From the **Favorites** menu, click on **Snapshot of Exercise 7.vi** and you will see an **image** of Exercise 7's **Front Panel** displayed in the Web browser like the one below:

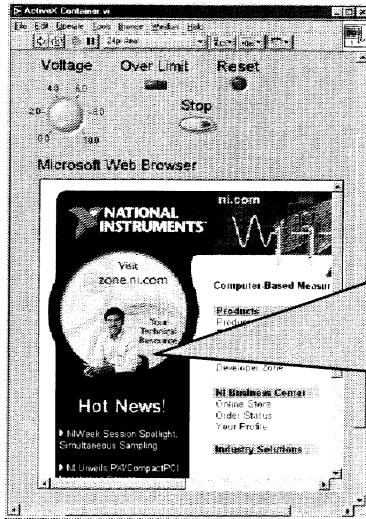


5. As you can see, you can literally publish images of your VIs to the Web simply by clicking in a check box and typing in a simple URL. **Close** Exercise 7.vi **without** saving changes.



Now, let's examine how LabVIEW can integrate with other applications and enterprise tools.

ActiveX Controls

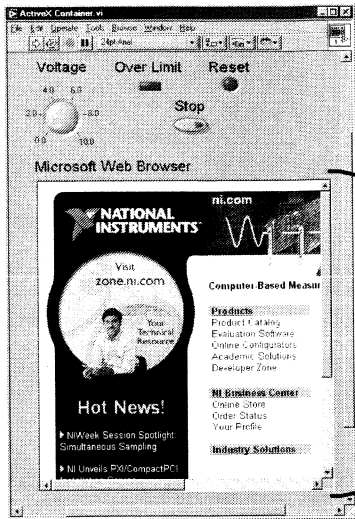


ni.com

- Components with specific functionality
- Require a *container*
- May include a user interface
- Have properties and events
 - Property pages
- Support remote and Internet access



ActiveX Containers



- Hold ActiveX controls
- Expose configuration programmatically or through property pages
- Handle events

ni.com



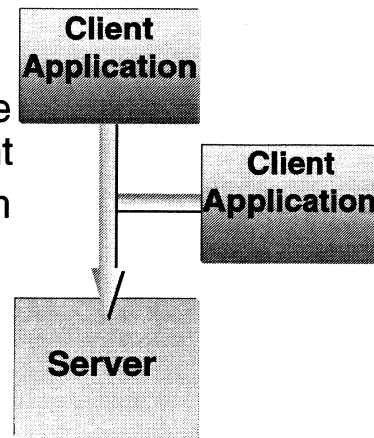
ActiveX is becoming the defacto way of integrating software components in the Windows environment. Let's take a look now at the ActiveX capabilities of LabVIEW. First, ActiveX controls are ActiveX components providing a service to the application in which they are used. They can be embedded in ActiveX containers like Visual Basic or LabVIEW. For example, a National Instruments data acquisition ActiveX control allows the application using it to have the service of a National Instruments data acquisition board. Many ActiveX controls include a user interface like a graph or table, but it is not required. ActiveX controls like the data acquisition control do not have a user interface, but instead can acquire data and share that data with a graph ActiveX control.

ActiveX controls can be configured through two mechanisms: through property pages or programmatically. *Property pages* are a series of dialogs available at configuration time that allow you to set the features of the control. Typical properties of the DAQ control would include the scan rate or the channel number. Most properties can also be set programmatically through Automation which we will discuss in a moment. Some ActiveX controls can trigger an *event* when they have finished an operation. An event essentially calls a predefined function in your application when the service is complete. Events provide many advantages. For example, you do not have to programmatically wait until an operation is complete before continuing, nor do you need to continually poll the control to see when it has finished. Since ActiveX is based on COM+ and COM+ allows you to extend control of your components over a network, you can control your ActiveX controls remotely.

Applications are ActiveX containers if they can hold or contain an ActiveX control and can manipulate the control through Automation or by configuring its property pages. Simply put, a container holds controls. LabVIEW is an ActiveX container. If an ActiveX control has a user interface, the container allows you to insert and place the control. The container provides some way of setting up or programming the control either through property pages or programmatically through Automation. Again, property pages are dialogs that allow you to set properties of the control. Programmatically these can be changed by accessing variables. In a few minutes we will see an example of configuring an ActiveX control programmatically in LabVIEW.

Automation Clients and Servers

- *Client*—software controlling or programming server
- *Server*—software providing the service or serving data to client
- Client accesses server through published API
- Many applications can use same server
- Servers rarely require UI interaction



ni.com

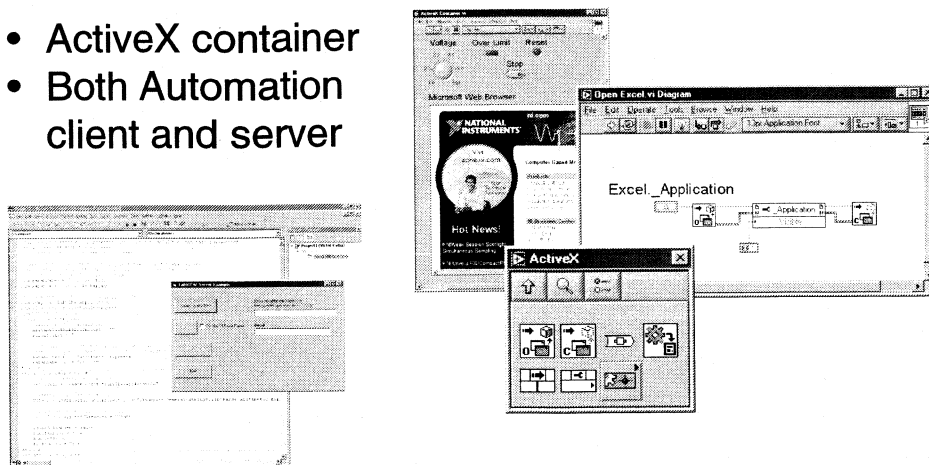


ActiveX automation is the programmatic interface to control an application or software service through scripting. Automation gives you access to the programmatic API of the application or service. The application that controls or scripts the application or service is the Automation client and the ActiveX component providing the service to be controlled is the Automation server. Again, servers provide data and services to clients.

Since Automation servers just serve data, they rarely require a user interface. If you control an application through Automation you will probably never see the user interface of the application unless you specifically program it to show it.

LabVIEW ActiveX and Automation

- ActiveX container
- Both Automation client and server



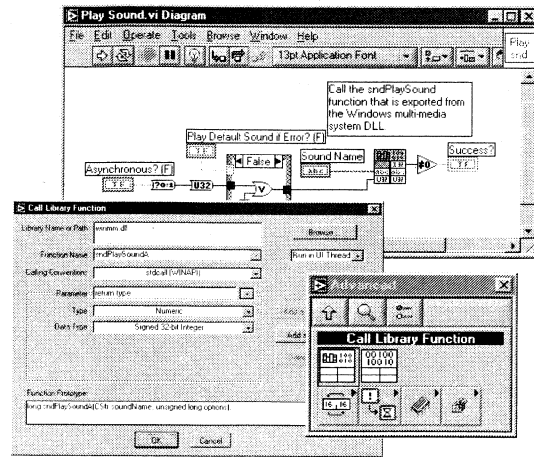
ni.com



In many applications, Automation becomes the glue that binds the services of many applications together. Automation is a powerful tool that allows you to integrate many tools and different applications into your LabVIEW programs. LabVIEW is both an Automation client and an Automation server for maximum programming flexibility. Because you can embed ActiveX controls in LabVIEW as well, it is an ActiveX container.

Call Shared Libraries (DLLs)

- Call external code
 - DLLs
 - Shared libraries
- Automatic display of function names and prototypes
- VIs automatically generated from Measurement Studio DLLs



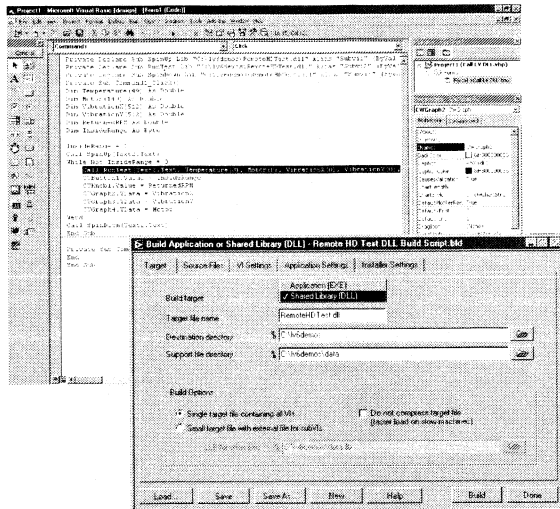
ni.com



One of the most common ways to share code today is through the use of a dynamic link library (DLL) or shared library. A DLL is a code library that is linked to programs when it is loaded or run rather than as the final phase of compilation. This means that the same block of code can be shared between several different tasks or applications rather than each one containing copies of the routines it uses.

LabVIEW capitalizes on this fact by allowing you to call external code in the form of DLLs on Windows platforms and shared libraries on Macintosh and UNIX platforms. Standard application development environments such as Microsoft Visual C++, Visual Basic, and National Instruments Measurement Studio can create DLLs and shared libraries for use in LabVIEW. Using the LabVIEW Call Library function, you integrate the external code directly into your Block Diagram. If you use Measurement Studio to build the DLL, LabVIEW can automatically generate VIs that correspond to the DLL functions through a Measurement Studio function panel. To simplify the process of integrating external code into LabVIEW, the function names and prototypes exported by the DLL are automatically displayed when you configure the Call Library function.

Generate Shared Libraries (DLLs)



ni.com

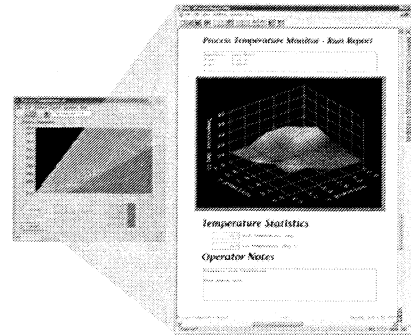


- Create DLLs and shared libraries through Application Builder
- Create build files for easy replication
- Create installers for easy distribution

To ensure that LabVIEW code easily integrates with different programming languages and enterprise tools, the LabVIEW Application Builder can generate a DLL or shared library from any VI. You can then easily incorporate these DLLs or shared libraries into other programming environments such as Microsoft Visual Basic or Visual C++ or National Instruments Measurement Studio. All you have to do to generate your DLL is simply complete a dialog box. You can even create build files that contain build instructions for easy replication of the DLLs and shared libraries you have already built. Additionally, you can create installers on Windows platforms for easy distribution of your DLLs.

HiQ Report Generation

- Publication-quality reports from LabVIEW
- Interactive report layout
- Load and edit HiQ scripts
- OpenGL 3D data visualization



ni.com

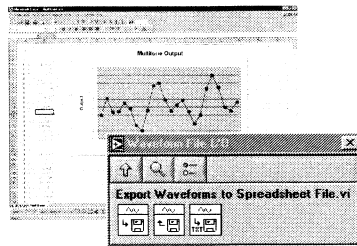
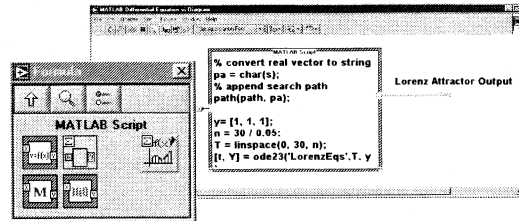


HiQ is a high-performance, interactive, problem-solving environment for analyzing, visualizing, and documenting real-world science and engineering problems. HiQ uses a virtual notebook interface and a programming language called HiQ-Script. The notebook has pages, sections, and tabs where you can place and arrange objects such as text, numerics, and graphs. From LabVIEW, you can launch HiQ, access and manipulate HiQ Notebooks to analyze and visualize data, run a HiQ-Script within a Notebook, and print publication-quality reports. By using the HiQ interactive notebook interface, you save time by designing the layout of your reports interactively without having to write any additional code.

With both LabVIEW and HiQ, you can take advantage of powerful OpenGL 3D data visualization capabilities for advanced data visualization. This means 3D line, point, surface, and contour plots with the ability to interactively rotate and visualize any kind of data. HiQ also includes hundreds of built-in functions for advanced data analysis. Although LabVIEW already contains formula and expression nodes for evaluation formulas directly in the Block Diagram, LabVIEW also allows you to load and edit HiQ scripts so that you can work with HiQ's mathematics functionality as well as its own.

Excel and MATLAB

- Create or load MATLAB scripts
- Export waveforms and arrays in Excel format



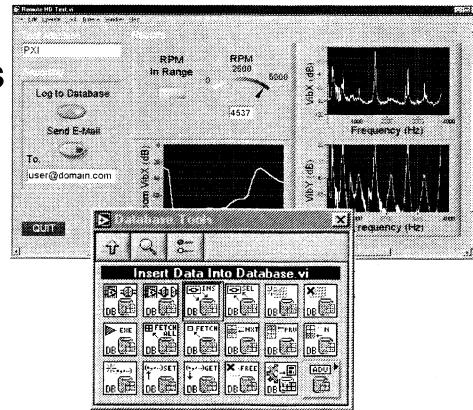
ni.com



Just as LabVIEW connects to HiQ for high-quality report generation capabilities, you can also run MATLAB scripts in LabVIEW if you have MATLAB installed. If you already have scripts written, you can import them directly into your code. If you do not already have scripts, you can actually create them using MATLAB syntax directly on the Block Diagram. Furthermore, LabVIEW can export both waveforms and arrays in standard spreadsheet format so that you can output your data and open it up directly in Microsoft Excel.

Databases

- Integration through add-on Database Tools



ni.com



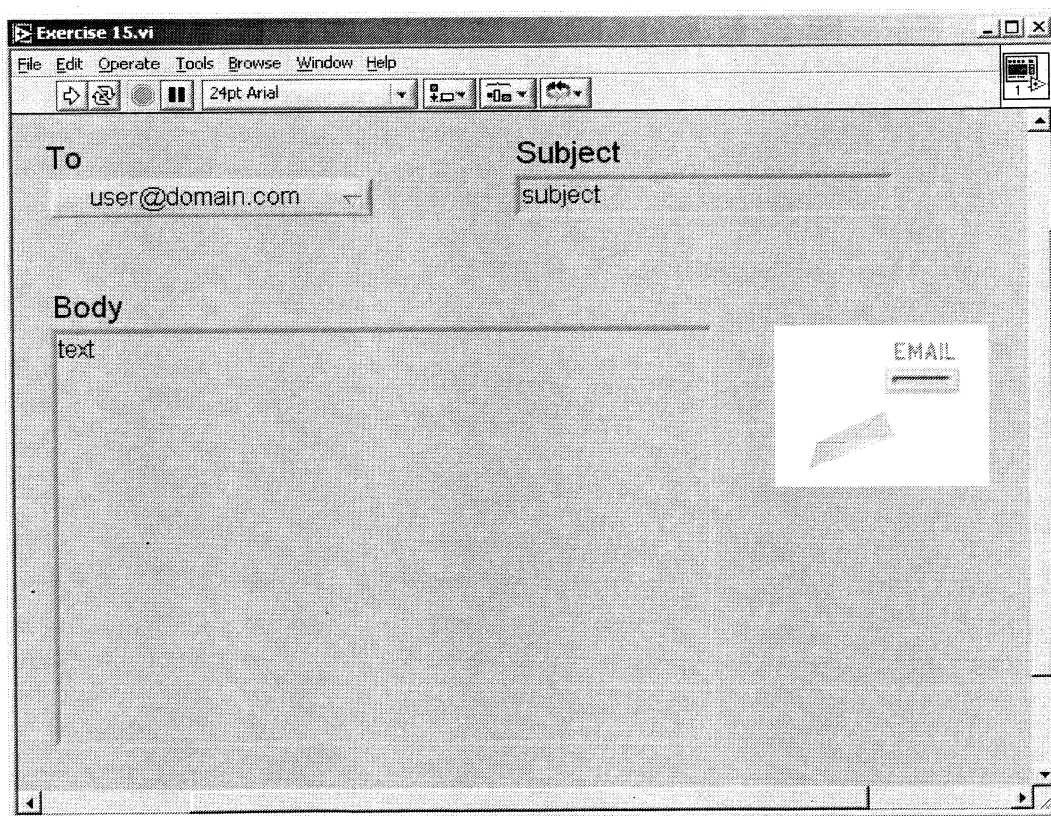
If you need to integrate databases into your measurement and automation applications, add-on tools are available. These database tools deliver high level, easy-to-use functions for integrating local and remote databases quickly and easily into your LabVIEW programs. With these tools, you can automatically insert data into and select data from databases, as well as perform a host of other common database operations. For programmers desiring advanced database functionality and flexibility in their applications, the Database Tools also allow you to execute SQL statements from within LabVIEW applications. We will discuss these database tools and their capabilities in more detail at a later time.

(OPTIONAL)

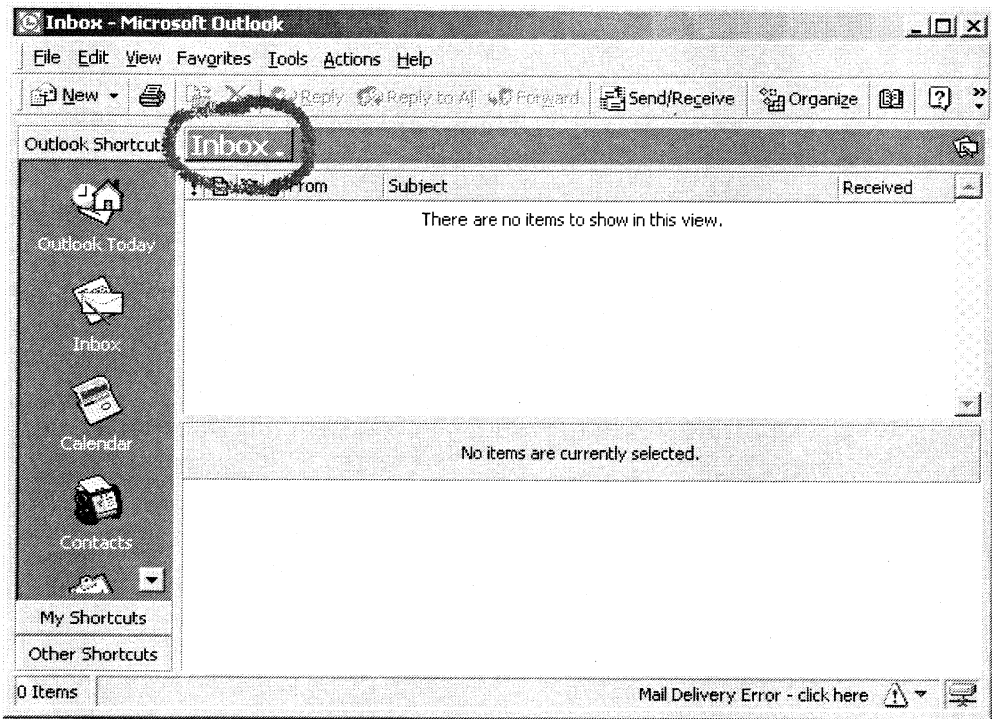
Exercise 14 – *Programmatically send an e-mail message*

In this exercise, you will use a ready-to-run VI to programmatically send and view an e-mail message. This VI is not part of the examples that ship with LabVIEW, but you can find it on Developer Zone by visiting <http://zone.ni.com>.

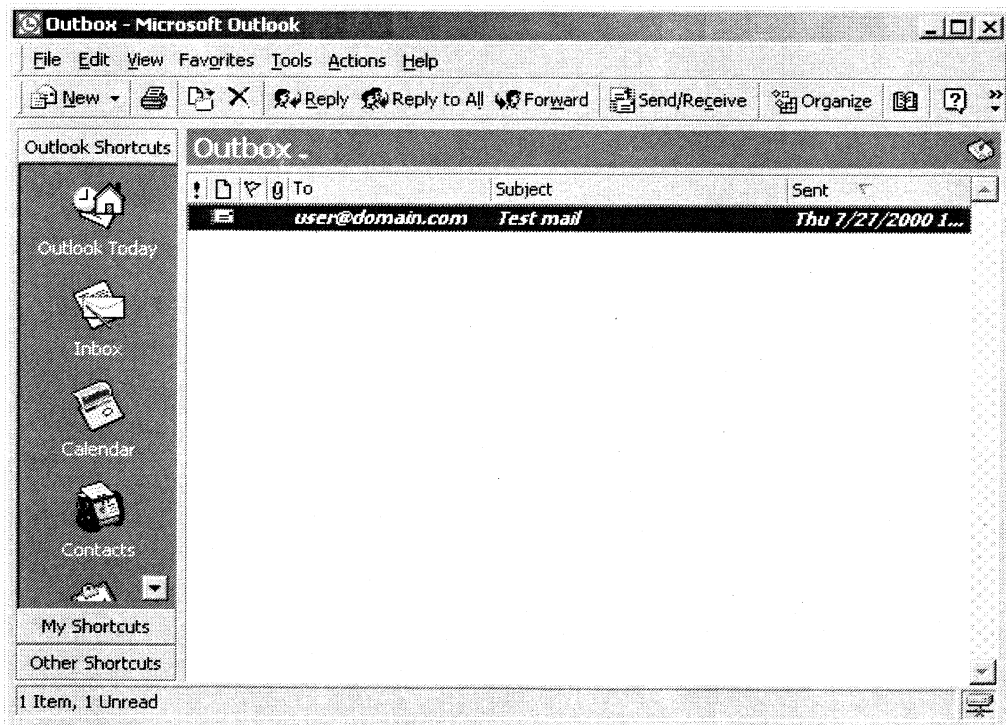
1. Open the **C:\National Instruments\LabVIEW\seminar\solutions** folder and launch **Exercise 14.vi**



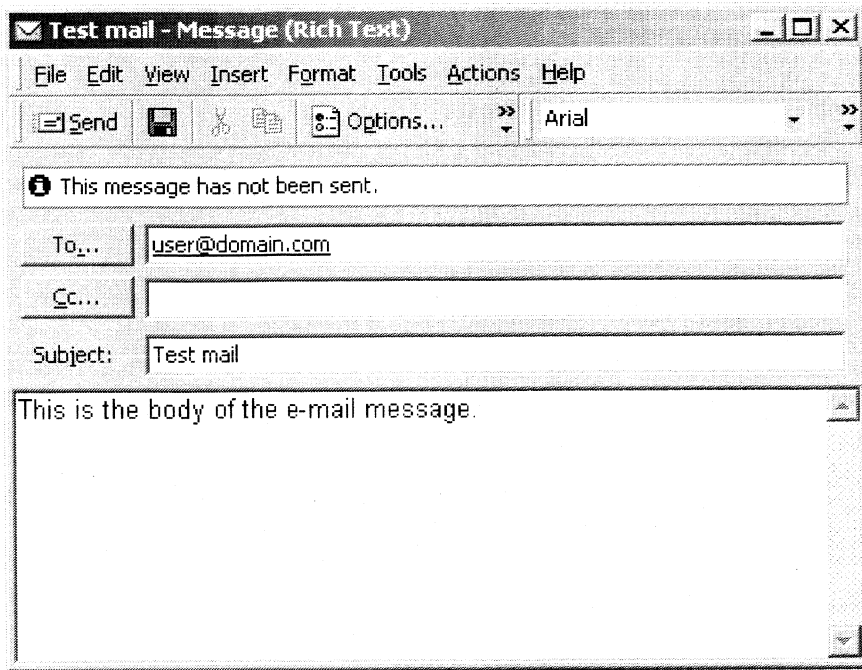
2. This VI **programmatically** sends an **e-mail** through **ActiveX Automation** and then launches **Microsoft Outlook** so you can view the **pending message**. To do this, **select** a recipient from the **To** menu ring with the **Operate** tool and enter a **subject** for your e-mail and the **text** to include in the **body** with either the **Operate** tool or the **Labeling** tool. Press the **run button** to run the VI. This will **create** the message and then **launch** Outlook. To **view** the outgoing message, **click** on the menu ring in Outlook like the one **circled** below in **red** and select **Outbox**.



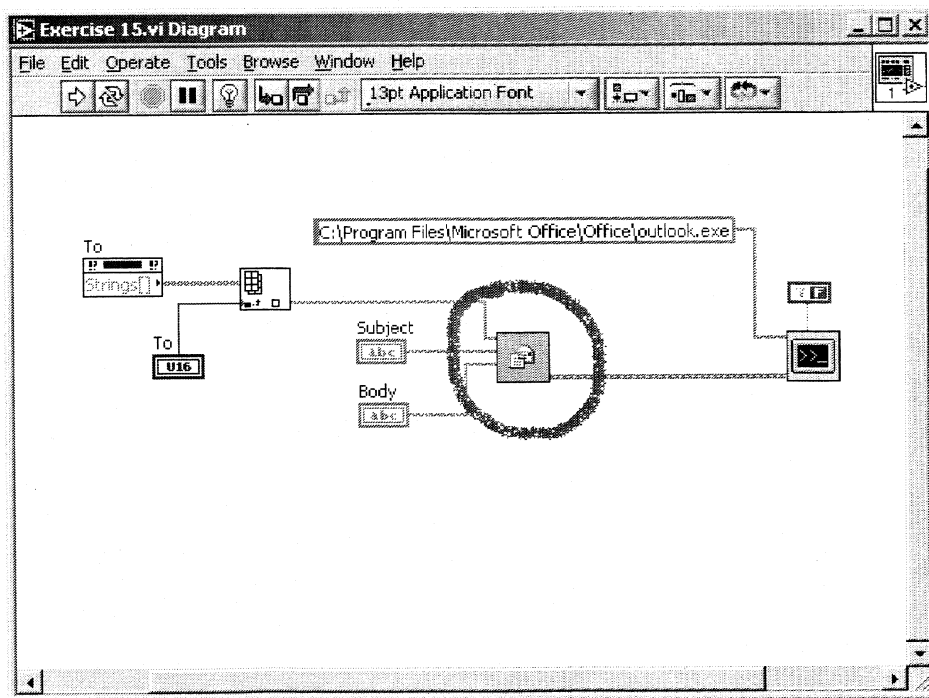
You will then see a **pending** message listed in the **Outbox** as shown in the picture below:



To **open** and view the **message** in its entirety, simply **highlight** it in the **Outbox** and press **Enter**.



3. If you wish to look at the **code** that generates the e-mail programmatically, return to **Exercise 14.vi** and **double click** on **Create Outlook Mail Message.vi** (circled in red below). This will **open up** the subVI's **Front Panel**. You can also view the **Block Diagram** of the Create Outlook Mail Message VI by pressing **Ctrl+E**.

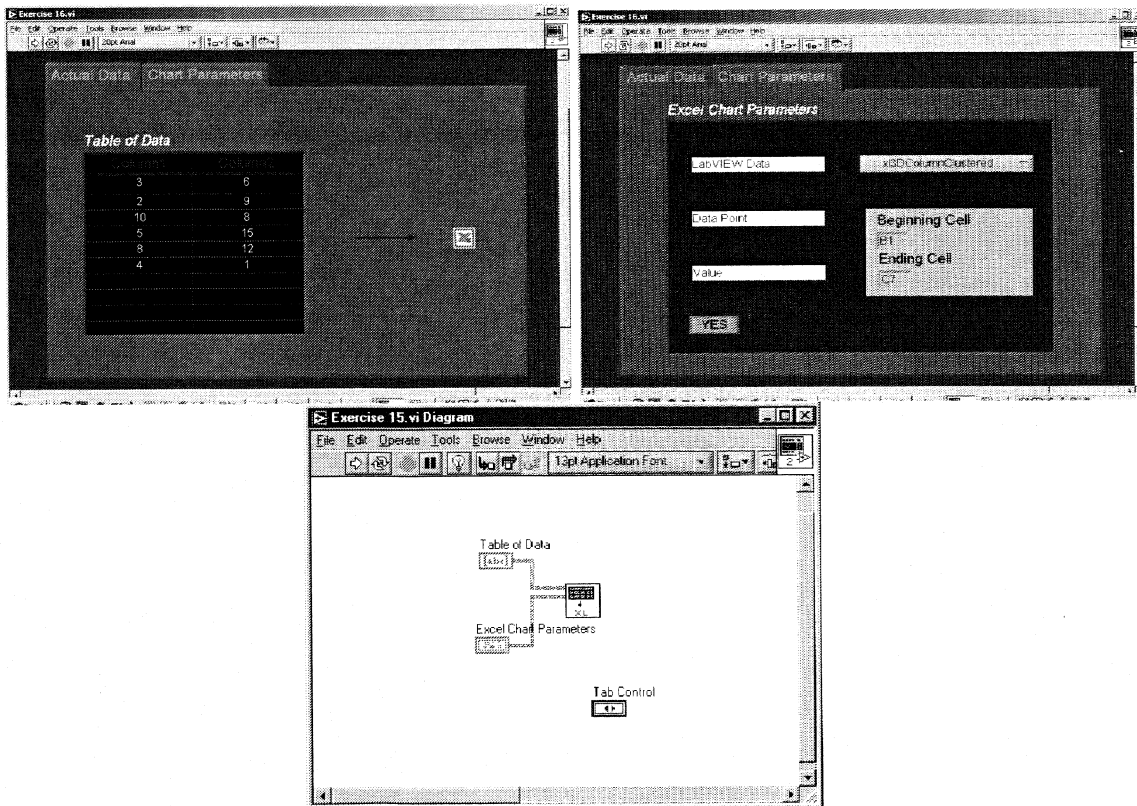


4. **Close Exercise 14.vi** and **Outlook** when you are finished. Do **not** save changes.

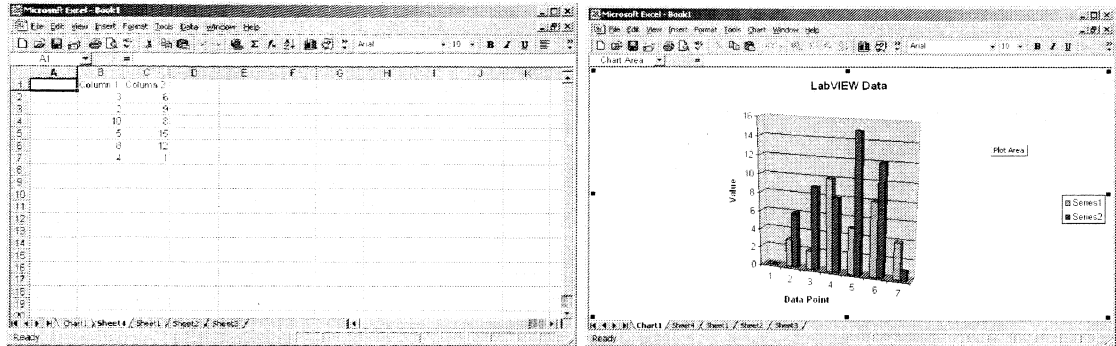
Exercise 15 – Export LabVIEW data to Excel and create a chart

In this exercise, you will run a VI that programmatically exports data from a LabVIEW table to a Microsoft Excel workbook and charts it. This VI is not part of the examples that ship with LabVIEW, but you can find it on Developer Zone by visiting <http://zone.ni.com>.

1. Go to **C:\National Instruments\LabVIEW\seminar\solutions** and double click **Exercise 15.vi**
2. This VI programmatically exports a **table** of data from **LabVIEW** to an **Excel** spreadsheet and then **charts** it. Once again, this is accomplished through **ActiveX Automation**. There is a **tab dialog** on the **Front Panel** of this VI. One tab contains the table of **data** to be **exported** to Excel. The other tab contains **parameters** that govern the **appearance** of the Excel **chart** that is generated. You can **switch** between the **tabs** of the tab dialog with the **Operate** tool. Below is a picture of the Front Panel displaying **each** of the **tab dialog's** pages as well as the accompanying **Block Diagram**.



3. Excel will **launch** and create a **chart** and a **worksheet** containing the data from the table in LabVIEW when you **run** the VI. You can see the worksheet **data** in Excel by clicking on the **Sheet4** tab in the **lower left** corner of the **workbook** and you can view the **chart** by clicking on the **Chart 1** tab.



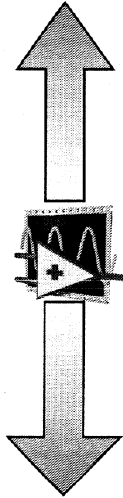
4. Close Excel and **Exercise 15.vi** and do **not** save changes.



The LabVIEW Platform

ni.com

The LabVIEW Platform



ni.com

- Developer Suite
- LabVIEW
- LabVIEW RT
- BridgeVIEW
- Machine Vision and Image Processing
- Add-on software



In addition to traditional LabVIEW capabilities, National Instruments offers a platform of LabVIEW functionality that includes real-time, datalogging and supervisory control, sound and vibration, and motion and vision.

LabVIEW Development Systems

- **Base Package**
 - Data acquisition
 - Serial, GPIB, VXI, computer-based instrument control
 - Data presentation
- **Full Development System**
 - LabVIEW Base Package
 - Measurement Analysis Libraries
 - MATLAB and HiQ integration (MATLAB not included)
 - 3D line, point, curve, and surface plots
 - Advanced report generation

ni.com



LabVIEW itself comes in three different development system options and is available on Windows 2000/NT/9x, Mac OS, Linux, Sun Solaris, and HP-UX. The Base Package includes all of the tools you need to develop a complete data acquisition or instrument control applications with basic data analysis. The Full Development System adds advanced measurement analysis libraries, report generation, custom graphics and animation, MATLAB and HiQ script integration, Web publishing, 3D plots, and the ability to call external code as DLLs and shared libraries to the Base Package. Please note that the LabVIEW Full Development System does not include MATLAB, but rather the ability to integrate MATLAB code into your LabVIEW applications. You must already have MATLAB installed on your system to use this feature.

LabVIEW Development Systems

- **Professional Development System**
 - LabVIEW Full Development System
 - Stand-alone executables and shared libraries (DLLs)
 - Code complexity metrics and differencing
 - Source code control
 - Quality and standards documentation

- **Software Subscription Program**

ni.com

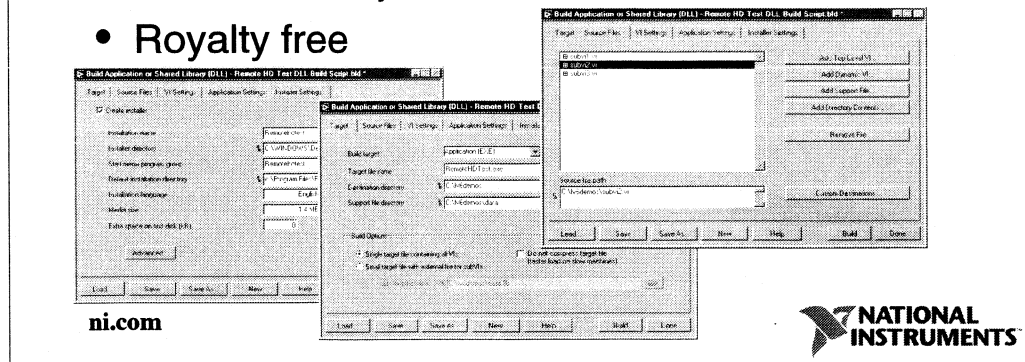


The LabVIEW Professional Development System facilitates the development of sophisticated, high-end systems for developers working in teams, users developing large suites of VIs, or programmers needing to adhere to stringent quality standards. The Professional Development System allows you to build both stand-alone executables and shared libraries (DLLs) and installers on Windows platforms for easy distribution. It also includes tools for source code control, code complexity measurements, and functional and cosmetic code differentiation through graphical differencing. To facilitate writing VIs in multiple languages, LabVIEW text can easily be exported to an external translation utility, translated, and then imported back into the VIs. Plus, there are documents discussing consistent programming style and steps to follow to meet internal regulations or other strict quality standards. For more information on LabVIEW, please visit ni.com/labview

The Software Subscription Program is an automatic maintenance service that provides you with free, automatic updates and upgrades for driver and application software, priority technical support, and discounts on customer education courses and materials.

Application Builder

- Generates executables or shared libraries (DLLs)
- Creates installers for distribution
- Installs directly into LabVIEW environment
- Royalty free



The LabVIEW Application Builder is a package for creating stand-alone executables and shared libraries (DLLs). It installs itself directly into your LabVIEW development system and enables you to generate stand-alone executables that you can run on any other computer running the same operating system. You can also use Application Builder to create shared libraries or DLLs on Windows platforms from any VI, which enables you to integrate your LabVIEW code into other development environments such as National Instruments Measurement Studio, Microsoft Visual Basic, or Visual C++. Once you have built your executables or shared libraries, you can create build scripts for easy replication and installers for easy distribution. Application Builder is part of the LabVIEW Professional Development System or is available for the other development systems as an add-on package. Distribution of your built applications does not require any licensing fees.

Enterprise Connectivity Toolset

- Internet Developers Toolkit
- SPC Toolkit
- Database Tools

ni.com



The LabVIEW Enterprise Connectivity Toolset delivers integrated tools for database connectivity, statistical process control, and Internet-enabling technologies to improve the quality of your products and decrease time to market. It includes the Internet Developers Toolkit, SPC Toolkit, and database tools for Windows platforms.

Internet Developers Toolkit

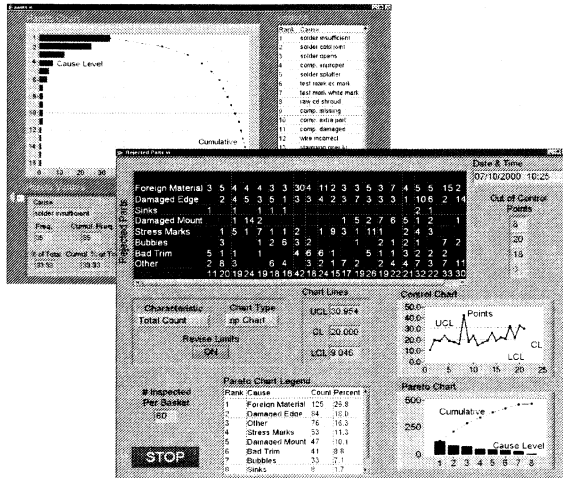
- Advanced Web server
- E-mail
- FTP
- CGI
- Telnet

ni.com



We have already mentioned that if you need to integrate Internet technologies such as e-mail, FTP, or CGI programming in your applications, the Internet Developers Toolkit is what you need. It also includes Telnet functionality as well.

SPC Toolkit



- Control charts
- Process statistics
- Pareto analysis

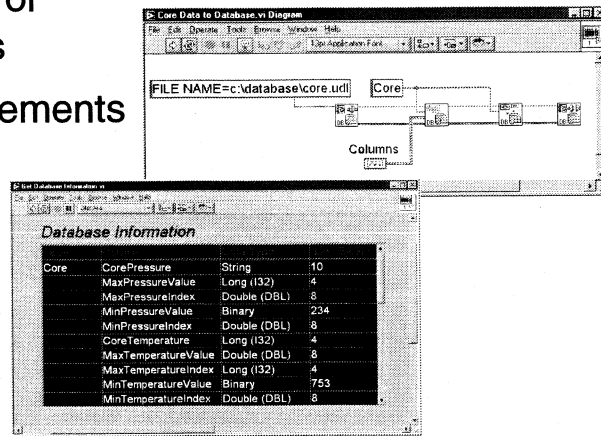
ni.com



The SPC Toolkit contains functions for process monitoring applications. Its functionality can be grouped into three different areas of SPC: control charts, process statistics, and Pareto analysis. The control chart libraries can compute and plot points for different types of attributes and variables charts and detect out-of-control points or process shifts with run rules. The process statistics VIs analyze process capability by evaluating if a process is normally distributed, calculating process capability ratios, and determining the fraction nonconforming for normally distributed processes. Pareto analysis allows you to display the relative importance of problems or assignable causes in your process. With the SPC Toolkit, you can display the number of occurrences or percentage occurrence of particular causes.

Database Tools

- Interact with local or remote databases
- Execute SQL statements
- Save, fetch, and update records
- Use ODBC
- Ship with ODBC drivers for many databases



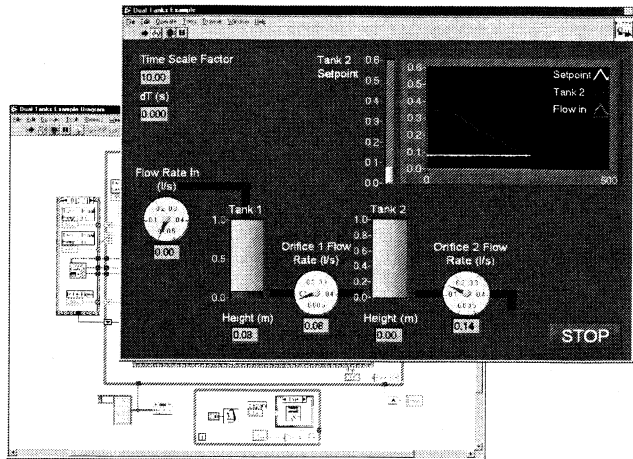
ni.com



The add-on database tools in the Enterprise Connectivity Toolset include a collection of functions for interaction with local and remote databases. You can save, fetch, and update data records as well as execute the complete set of SQL functions. The connectivity with databases comes through ODBC drivers and to ensure connectivity to most databases, the Database Tools ship with a number of ODBC database drivers.

PID Control Toolset

- PID Control
 - Autotuning
 - Gain scheduling
- Fuzzy Logic
 - Control strategies
 - Decision making



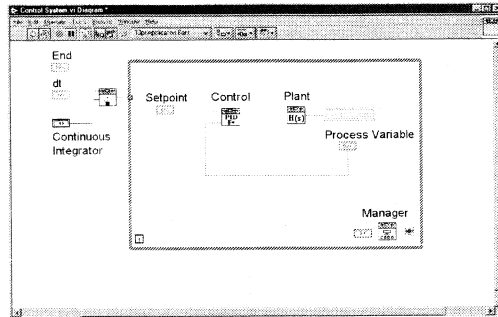
ni.com



To quickly develop automated control applications, the PID Control Toolset provides sophisticated control algorithms for PID and fuzzy logic control. The PID tools implement a wide range of PID algorithms and even feature autotuning and gain scheduling to improve system performance. For nonlinear or highly complex systems, the fuzzy logic tools accelerate development by implementing control strategies through simple linguistic rules. The tools can also be used for decision making, such as pattern recognition or fault diagnosis.

System Simulation and Design

- Common control elements
- Hardware integration
- Different system representations
- Dynamic system response
- Bode, Nyquist, and root-locus plots



ni.com

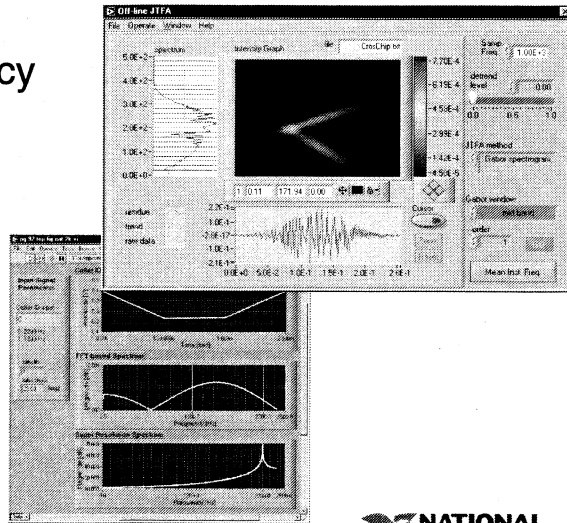


For simulation, design, analysis, and optimization of linear and nonlinear advanced control systems, you need the System Simulation and Design Toolset for LabVIEW which has both continuous and discrete capabilities. This Toolset not only includes symbols, blocks, and control elements commonly used in control engineering, but you can actually integrate DAQ hardware into your system for real-world control. To aid in advanced control design and optimization, you can transform between different representations of your system, view dynamic system responses, and generate Bode, Nyquist, and root-locus plots. There are even tools for managing, synchronizing, and monitoring real-time system behavior.

Signal Processing Toolset

- Joint time-frequency analysis
- Digital filter design
- Super-resolution spectral analysis
- Wavelet and filter bank design

ni.com

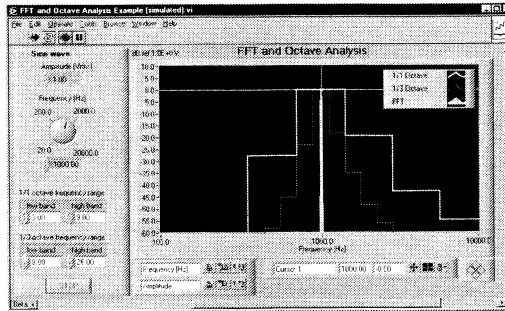


**NATIONAL
INSTRUMENTS**

The LabVIEW Signal Processing Toolset provides powerful tools for joint time-frequency analysis, digital filter design, super-resolution spectral analysis, and wavelet and filter bank design. The joint time-frequency portion of the toolset allows you to simultaneously examine the time and frequency domain representations of a signal. Quickly design lowpass, highpass, bandpass, and bandstop FIR and IIR filters interactively and output filter coefficients for use in LabVIEW and other applications. Super-resolution spectral analysis provides a model-based alternative to the FFT and delivers estimates of amplitude, phase, damping factor, and frequency of the damped sinusoidal components of a signal. The wavelet and filter bank design tools decompose a signal into multiple bands that represent the signal in terms of varying time and scales through a bank of filters. This decomposition facilitates extraction of signal features, noise reduction, and other operations.

Sound and Vibration

- Octave analysis
- Averaged frequency analysis
- Transient analysis
- Weighted filtering
- Calibration
- Sound level measurements
- Specialized displays

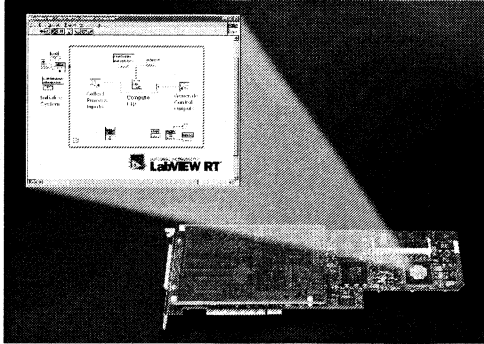


ni.com



For sound and vibration applications, the Sound and Vibration Toolset offers you averaged frequency analysis, transient analysis, octave analysis, sound level measurements, weighted filtering, and system calibration. The toolset also includes enhanced display tools such as waterfall displays and octave line and bar graphs.

LabVIEW RT



- Real-time data acquisition and control
- Download programs to dedicated hardware for deterministic, real-time performance

ni.com

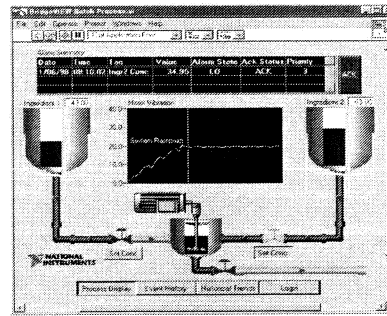


National Instruments LabVIEW RT delivers deterministic, real-time performance with an easy-to-use Windows-based development environment. Built on top of standard LabVIEW, you develop your LabVIEW RT applications with graphical programming, then download and run them on an independent hardware target with a real-time operating system. As a result, you can use all the powerful development tools of LabVIEW to develop real-time, reliable solutions.

If you need to run your LabVIEW code deterministically—that is, in a perfectly predictable and repeatable interval of time, then LabVIEW RT is what you need. LabVIEW RT is great for running real-time control loops—or any application that requires the code to execute in a precise amount of time.

BridgeVIEW

- High performance data logging and supervisory control
- System alarms and events
- PLC connectivity
- Standard OPC connectivity



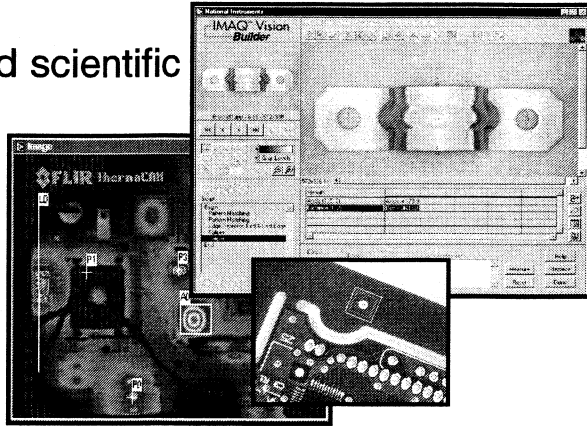
ni.com



BridgeVIEW is built on top of LabVIEW to handle high channel count and distributed applications. It adds easy networking, tag configuration, alarm and event management, historical datalogging, real-time trending, PLC connectivity, and OPC integration to the LabVIEW environment.

Machine Vision and Image Processing

- Machine vision and scientific imaging
 - Inspection
 - Gauging
 - Pattern matching
 - Color matching
 - Blob analysis and morphology
- Vision Builder
- Optical character recognition



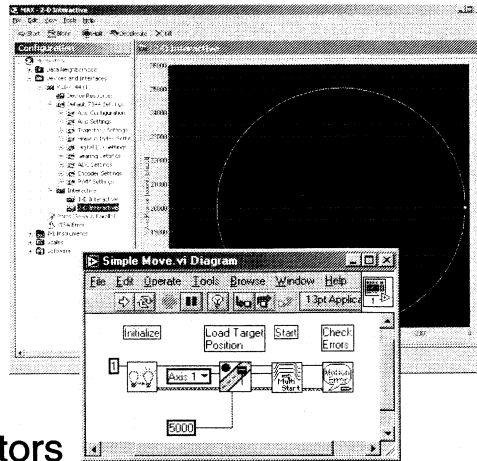
ni.com



LabVIEW also has machine vision and image processing tools for machine vision and scientific imaging applications. These tools include high-level vision processing and display functions and Vision Builder, an interactive environment for rapid prototyping of vision applications. You can accelerate your development time with functions for inspection, gauging, pattern matching, blob analysis and morphology, and even color matching. Optical character recognition software is also available for high-speed reading performance despite poor image quality, poor lighting, and degradations from industrial process variations.

Motion Control

- Easy-to-use
 - Configuration
 - VIs
- Powerful feature set
- Easy integration
 - Data acquisition
 - Vision
- Single and multi-axis
- Servo and stepper motors



ni.com



Motion control libraries are included with LabVIEW if you need to develop and implement motion control applications. These libraries allow you to work with single and multi-axis motion control systems and can control both stepper and servo motors. Interactive configuration through Measurement and Automation Explorer helps you get your motion systems up and running quickly and the easy-to-use VIs offer a powerful feature set. Since many motion systems do more than just motion control, National Instruments has focused on making sure that integration between your motion, vision, and data acquisition system is as simple as possible to give you a complete solution.

NI Developer Suite

- **Standard Edition**
 - LabVIEW
 - Measurement Studio
 - IVI Driver Toolset
 - HiQ
- **Professional Edition**
 - NI Developer Suite Standard Edition
 - Enterprise Connectivity Toolset
 - Signal Processing Toolset
 - PID Control Toolset
 - System Simulation and Design Toolset
 - Automation Symbols Toolset

ni.com



The NI Developer Suite is a subscription-based bundling of the most popular National Instruments software products that gives you the tools you need, when you need them. NI Developer Suite is an exceptional value because the cost of the entire Suite is much lower than the cost of all the individual components. Whether you need tools for instant connection to other programs, the Internet, or corporate databases, NI Developer Suite delivers the connectivity you need to provide communication throughout your organization. Several editions of NI Developer Suite are available to suit your specific software needs. The Standard Edition consists of LabVIEW, Measurement Studio, the IVI Driver Toolset, and HiQ. The Professional Edition includes all of the software in the Standard Edition plus the add-on toolsets that expand core LabVIEW capabilities which we have already discussed. The Professional Edition of NI Developer Suite literally saves you thousands of dollars over purchasing LabVIEW and all the add-on toolsets separately.

NI Developer Suite

- **Test Edition**
 - NI Developer Suite Professional Edition
 - TestStand
- **All editions include a one year software subscription free**

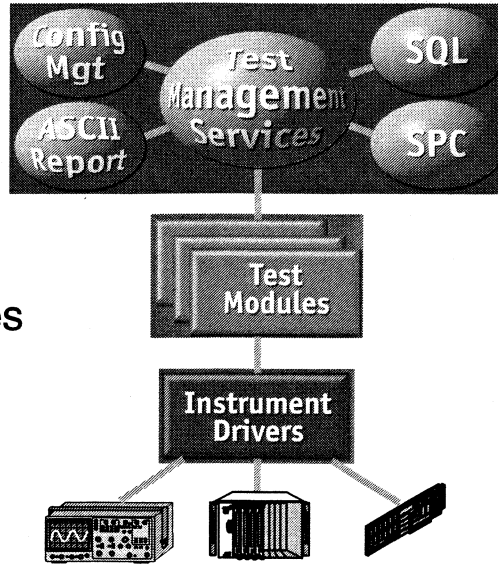
ni.com



The NI Developer Suite Test Edition is ideal for automated test applications. It builds on the NI Developer Suite Professional Edition by including TestStand for complete automated test management. A purchase of Developer Suite, any edition, gives you a one year software subscription free of charge. This means you receive a set of quarterly software CDs and priority technical support for one year at no extra cost so that you always have the most current versions of National Instruments software. To find out more information about Developer Suite, visit ni.com/suite

TestStand

- Off-the-shelf, customizable test executive
- Executes tests written in most popular languages
- Automatic report generation
 - Text
 - HTML
 - Database



TestStand is a customizable, off-the-shelf, high-speed test executive for organizing and executing automated tests. You can execute tests written in all of the most popular test languages, including LabVIEW, Measurement Studio, and Visual Basic. Additionally, you can call any C DLL or ActiveX server. TestStand provides automatic test report generation in text, HTML, and database formats. TestStand allows the test developer to customize almost every aspect of its execution and development environment.



Conclusion

ni.com

Benefits

- Rapid application development
- Tight integration between software and hardware
- High performance compiled execution
- Long-term compatibility
- Open development environment

ni.com



LabVIEW accelerates development over traditional programming by a factor of four and sometimes by as much as a factor of ten. With LabVIEW, you can prototype, design, and modify systems in a short amount of time. The tight integration between LabVIEW and measurement hardware and sensors means you can build applications in fewer steps and with less programming which leads to decreased development time. All LabVIEW applications execute at compiled speed for optimal performance and LabVIEW provides backward compatibility clear back to version 1.0 which preserves your investments in code development. With the open development environment of LabVIEW, you can connect to other applications, the Internet, DLLs and shared libraries, databases, DataSocket, TCP, and a host of other protocols. The bottom line is that LabVIEW simply makes you more productive so that you can focus on what you do best—taking measurements.

Additional Services



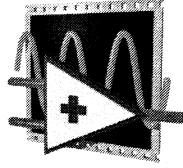
- Technical support
- Alliance Program
 - Systems integrators
- Customer education
 - Regional classes
 - On-site classes

ni.com



National Instruments offers world-class technical support and regional and on-site customer education classes to help you build powerful, high performance measurement and automation systems. A worldwide network of highly skilled Alliance Program members provides a wealth of systems integration expertise to help you develop systems if you have time constraints or limited technical resources.

Questions or Comments?



NATIONAL INSTRUMENTS™
LabVIEW™

ni.com



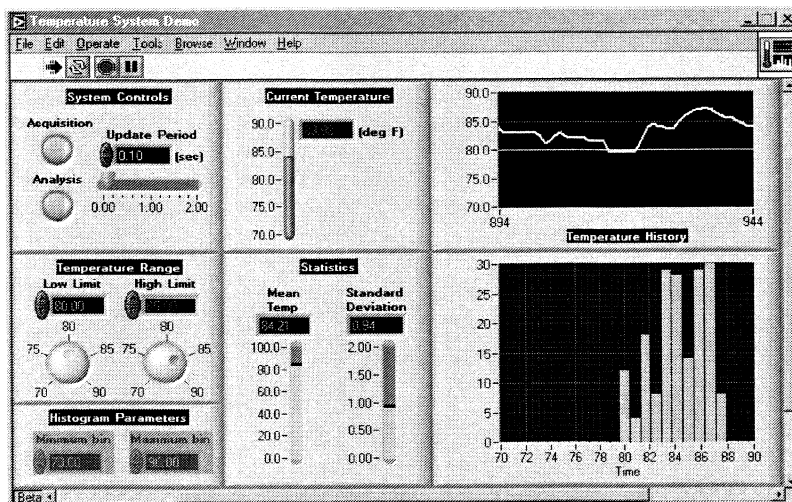
Thanks for coming to the LabVIEW Hands-On seminar!! We hope that you have learned why LabVIEW is a powerful tool for writing measurement and automation programs in a short amount of time.

Appendix

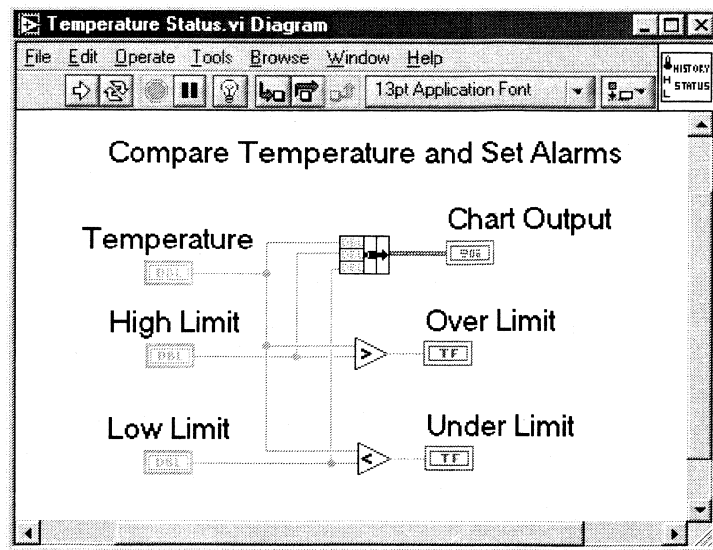
The LabVIEW Environment

Panel and Diagram Windows

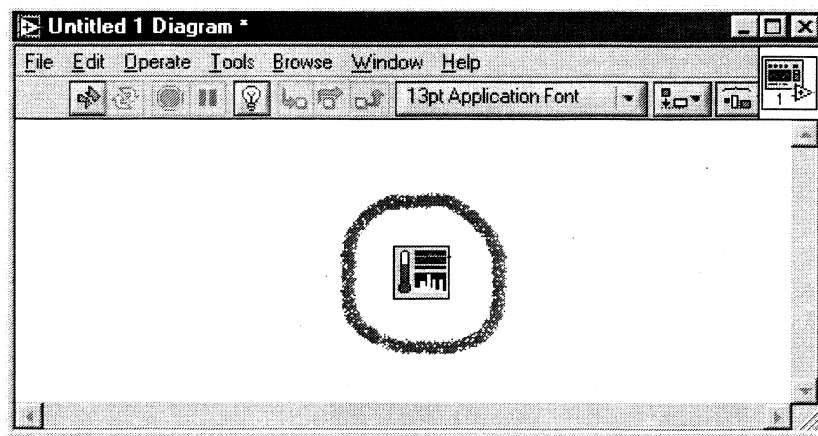
LabVIEW programs are called virtual instruments (VIs). VIs have three main parts: the front panel, the block diagram, and the icon/connector. The front panel is your means of setting input values and viewing outputs from the VI block diagram. Because the front panel is analogous to a front panel of a real instrument, the inputs are called controls and the outputs are called indicators. You can use a variety of controls and indicators, such as knobs, switches, buttons, charts, graphs, and so on to make the front panel easily identifiable and understandable. An example of a VI front panel is shown below:



Each front panel has an accompanying block diagram, which is the VI program. You build the block diagram using graphical programming. You can think of the block diagram as source code. The components of the block diagram represent program nodes; for example, For Loops, Case structures, and arithmetic functions. The components are “wired” together to define the flow of data within the block diagram. The block diagram for a VI is shown on the following page.

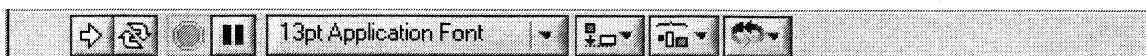


You use the icon/connector to turn a VI into an object (subVI) that you can use as a “subroutine” in the block diagrams of other VIs. The icon graphically represents the VI in the block diagram of other VIs. The connector *terminals* determine where you must wire the inputs and outputs on the icon. The terminals are analogous to subroutine parameters. They correspond to the controls and indicators on the VI front panel. The following illustration shows the icon for a subVI on the block diagram of another VI (circled in red).




Front Panel Toolbar


Both the front panel and block diagram windows contain a toolbar of command buttons and status indicators that you use for controlling the VI. One of two toolbars is available, depending on whether you are working in the Panel or Diagram window. The following toolbar appears at the top of the Panel window.





Below is a description of the most commonly used command buttons on the front panel toolbar.

 **The Run button.** You click on it to run the VI.

 **The Broken Run button.** This button replaces the Run button and indicates that the VI cannot compile due to errors. To find out why, click on this button. A pop-up window lists all errors.

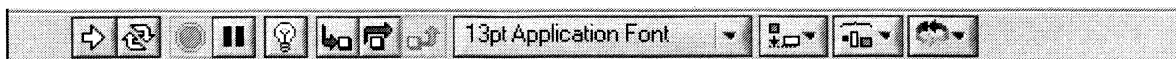
 **The Continuous Run button.** Click on it to execute the VI repeatedly. Click on this button again to disable continuous running.

 While the VI is executing, the **Abort Execution button** appears. Click on this button to halt the VI immediately. *You should avoid using the Abort Execution button to terminate your VI, and either let the VI execute to completion or design a method to terminate the VI programmatically. If you do not do this, your VI could be left in an unknown state.*


 **The Pause/Continue button.** This button pauses VI execution. To continue from pause mode, press the button again, and the VI continues execution.


Block Diagram Toolbar


The block diagram toolbar contains most of the same buttons as the front panel toolbar, in addition to four debugging features as shown below:




Following is a brief description of the command buttons used for debugging:

 **The Execution Highlighting button.** Used for debugging to see the flow of data through the block diagram. Click on it again to disable execution highlighting.

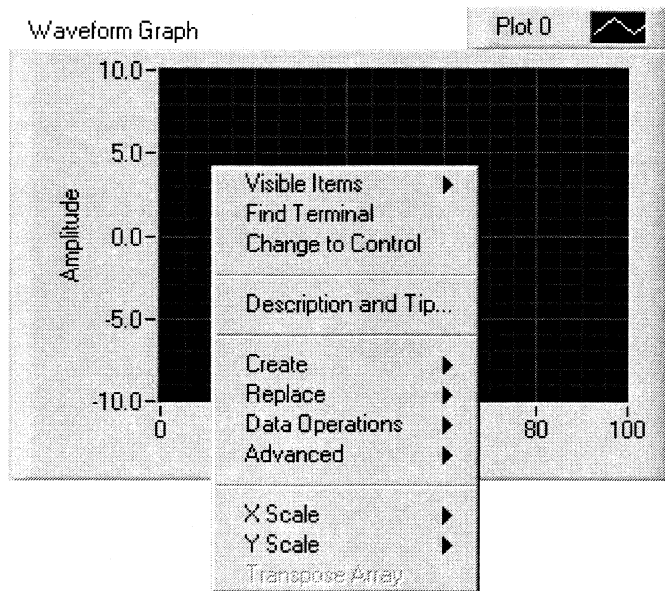
 **The Step Into button.** Click on the Step Into button to step into a loop, subVI, structure, or other node. In LabVIEW terminology, a node is simply a program execution element. Nodes are analogous to statements, operators, functions, and subroutines in text-based programming languages. By stepping into a node, you are ready to single step inside that node.

 **The Step Over button.** You click on this button to step over a loop, subVI, or other block diagram node. By stepping over a node, you execute the node without single stepping through the node.

 **The Step Out button.** Click on the Step Out button to step out of a loop, subVI, or other block diagram node. By stepping out of a node, you complete single stepping through the node and go to the next node.

Shortcut Menus

Nearly all the objects you use to build VIs have shortcut menus for selection and modification. You access shortcut menus by holding down the right mouse button when the cursor is on the desired panel or object. Below is a picture of the shortcut menu for a waveform graph.

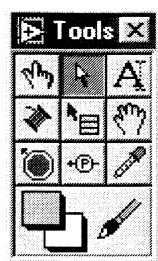


Palettes

LabVIEW has graphical, floating palettes to aid in creating and operating VIs. The three palettes include the Tools, Controls, and Functions palettes.

Tools Palette

You can create, modify, and debug VIs using the tools located in the floating Tools palette. If the Tools palette is not visible, select Show Tools Palette from the Windows menu to display the palette. After you select a tool from this menu, the mouse cursor takes its shape. Place any tool found in the Tools palette over a subVI or function icon to display information pertaining to that subVI or function in the Context Help window. You first must select Show Context Help from the Help menu or press Ctrl+H.



Operating tool. Use the Operating tool to manipulate the values of front panel controls and indicators. The tool changes to a text icon when it passes over a text-based control, such as a digital or string control.



Positioning tool. Use the Positioning tool to select, move, or resize objects. The tool changes to an icon that looks like the corner of a picture frame when it passes over a corner of a resizable object.



Labeling tool. Use the Labeling tool to enter text into labels.



Wiring tool. Use the Wiring tool to wire objects together on the block diagram. Place the Wiring tool over a wire to display the data type of the wire in the Context Help window. You first must select Show Context Help from the Help menu or press Ctrl+H.



Object shortcut menu tool. Use the object shortcut menu tool to open an object's shortcut menu with the left mouse button.



Scrolling tool. Use the Scrolling tool to scroll through windows without using scrollbars.



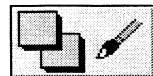
Breakpoint tool. Use the Breakpoint tool to set breakpoints on subVIs, wires, functions, and structures to temporarily suspend VI execution.



Probe tool. Use the Probe tool to create probes on wires in the block diagram for viewing data flowing through them while a VI is executing.



Color Copy tool. Use the Color Copy tool to copy colors for pasting with the Coloring tool.



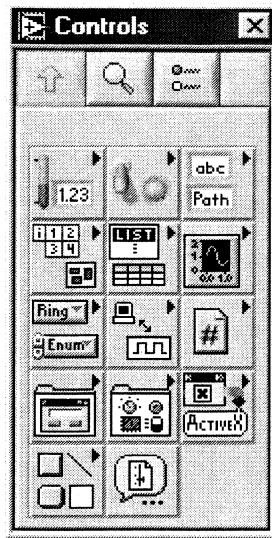
Coloring tool. Use the Coloring tool to color an object. It also displays the foreground and background of the object.

Controls and Functions Palettes

The Controls and Functions palettes consist of top-level icons representing subpalettes, giving access to a full range of available objects that you can use in creating a VI. You can access the subpalettes by clicking on the top-level icon. You also can convert the subpalette to a floating palette that remains on your screen by tacking down the pushpin at the top left corner of the subpalette.

Controls Palette

You add controls and indicators to the front panel via the Controls palette. Each option in the palette displays a subpalette of available controls and indicators for that selection. If the Controls palette is not visible, you can open the palette by selecting Show Controls Palette from the Windows menu. You also can access the Controls palette by right clicking on an open area in the front panel window. Then you can tack down the Controls palette into a floating palette by clicking on the pushpin on the top left corner of the palette. *The Controls palette is available only when the front panel window is active.*



Numeric subpalette. Consists of controls and indicators for numeric data.



Boolean subpalette. Consists of controls and indicators for Boolean (TRUE or FALSE) values.



String & Path subpalette. Consists of controls and indicators for strings and paths.



Array & Cluster subpalette. Consists of controls and indicators that group sets of data types.



List & Table subpalette. Consists of controls and indicators for listboxes and tables.



Graph subpalette. Consists of indicators to plot data in graphs or real-time charts.



Ring & Enum subpalette. Consists of controls and indicators for fixed string or numeric data.



I/O subpalette. Consists of controls and indicators for accessing instruments and data acquisition hardware connected to your system.



Refnum subpalette. Consists of controls and indicators for refnums.



Dialog Controls subpalette. Contains the graphical objects used to create dialog boxes that conform their appearance to that of the operating system on which they are running.



Classic Controls subpalette. Contains low-color versions of the standard controls and indicators.



ActiveX subpalette. Consists of controls and indicators that allow ActiveX Container and Automation capabilities.



Decorations subpalette. Consists of graphical objects for customizing front panel displays.



Select a Control... subpalette. Displays a dialog box to load custom controls.



Array subpalette. Consists of functions to create and process arrays, which are sets of data with the same data type.



Cluster subpalette. Consists of functions to create and manipulate clusters, which are groups of data composed of dissimilar data types.



Comparison subpalette. Consists of functions to compare numbers, Booleans (TRUE or FALSE), and strings.



Time & Dialog subpalette. Consists of functions for dialog windows, timing, and error handling.



File I/O subpalette. Consists of functions and VIs for File I/O.



Data Acquisition subpalette. Consists of VIs for performing analog input, analog output, digital I/O, counter measurements, calibration and configuration, and signal conditioning operations using data acquisition devices.



Waveform subpalette. Consists of functions and VIs for performing waveform operations.



Analyze subpalette. Consists of VIs for performing various types of analysis, such as waveform monitoring and conditioning.



Instrument I/O subpalette. Consists of VIs for communicating with GPIB and serial instruments connected to your system.



Motion and Vision subpalette. Consists of VIs for performing motion control or image acquisition operations.



Mathematics subpalette. Consists of VIs for performing advanced mathematics, such as curve fitting, linear algebra, calculus, and probability and statistics.



Communication subpalette. Consists of networking and communication VIs for TCP, UDP, ActiveX, and DataSocket.



Application Control subpalette. Consists of functions and VIs to programmatically control your VIs and LabVIEW as an application. Includes functionality for performing actions such as printing programmatically, changing LabVIEW menus, showing the Context Help window, and stopping or exiting LabVIEW.



Graphics & Sound subpalette. Consists of VIs for creating and modifying specialized plots and custom graphics. Also contains VIs for recording and generating sound.



Tutorial subpalette. Consists of VIs used in the LabVIEW Tutorial and other LabVIEW examples.



Report Generation subpalette. Consists of tools for creating and modifying printed or HTML reports of VIs and their output.



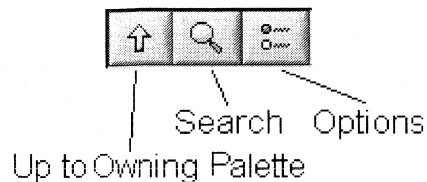
Advanced subpalette. Consists of miscellaneous functions for performing advanced operations such as data manipulation, synchronization, and integration of external code into LabVIEW VIs.



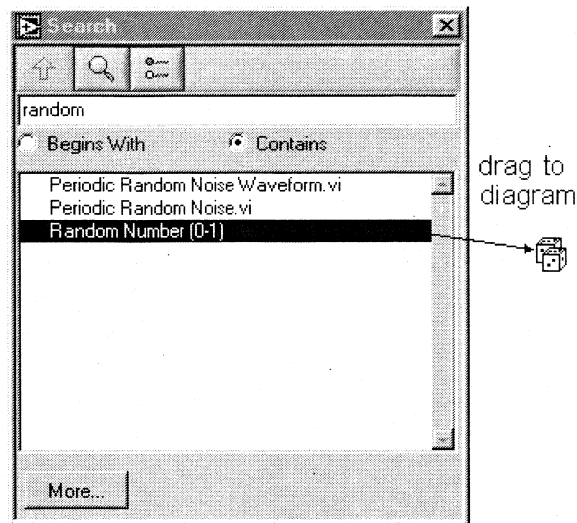
Select a VI... subpalette. Consists of a dialog box for inserting subVIs into the current VI.

Controls/Functions Browser

The easiest way to find the objects you want in the Controls or Functions palettes is to use the Controls/Functions Browser. When you select Show Controls Palette or Show Functions Palette from the Window menu, the appropriate floating palette appears on the screen. If you know where a particular function is located, you can go to that subpalette and select the item. However, if you are not sure where an object is located, you can use the browser tools at the top of the palette as shown below:



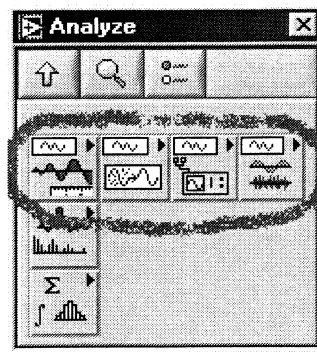
For example, suppose you are searching for the Random Number function. Click on the Search button and start typing Random Number into the string box at the top of the search window. The Controls/Function Browser lists all matching items that either start with or contain the search string you entered. Double-clicking on a listing in the search window locates the desired object in the appropriate subpalette. From the subpalette, you can then drag the desired object to the front panel or diagram to place it. Additionally, you can simply select a listing in the search window and drag it directly to the front panel or block diagram to place down the object it describes as shown below:



Pressing the Up to Owning Palette button returns you to the subpalette directly above the current subpalette in the Controls or Functions palette hierarchies.

LabVIEW Measurement Analysis Capabilities

The measurement analysis VIs can be found in four palettes on the top row of the LabVIEW Analyze palette (4th row, 3rd column). These new palettes are Waveform Measurement, Waveform Generation, Waveform Monitoring, and Waveform Filtering.



Overview

LabVIEW measurement analysis routines are high-level tools that handle many of the details associated with common measurements for signal generation, digital filtering, frequency analysis, limit testing, triggering, DC / RMS measurement, THD (Total Harmonic Distortion), and SINAD. As a specific example of how these routines address measurement details, Frequency analysis routines go beyond a standard FFT by including common ancillary tasks such as windowing, averaging, and scaling. By using these tools, you can speed up your development because you can avoid coding and testing these routines yourself.

Source Code Included

The Measurement Analysis VIs were developed using VIs from the standard LabVIEW Advanced Analysis Library. The VIs include source code, allowing you to examine or modify them for customization, educational purposes, or validation.

Parameterized Measurements

You can easily reconfigure these measurements to fit your particular analysis needs. For instance, by changing the time constant on the DC/RMS measurement, you can trade speed for accuracy. In addition, the tools include default parameters for common circumstances, so in many cases these measurements will work “out of the box.”

Tool Highlights

- 1) **DC/RMS Measurements**—tools for averaged and non-averaged DC/RMS measurements. A key capability of these VIs is their flexibility—by modifying parameters, you tradeoff between speed and accuracy in your measurements. Example parameters include measurement time, averaging type (exponential or linear), an exponential averaging time constant, and a window type.
- 2) **Accurate Tone Detection and Distortion Measurements**—tools for extracting prominent tone parameters such as frequency, amplitude, and phase from a time domain signal. These VIs include a proprietary algorithm that compensates for inaccuracies due to spectral leakage

and the discrete nature of the Discrete Fourier Transform (DFT). The tone detection algorithm is also the basis for several distortion measurement VIs including Harmonic Distortion Analyzer and SINAD (Signal + Noise + Distortion) tools.

- 3) **Averaged Frequency Analysis**—spectral analysis with three types of averaging: RMS, vector, and peak hold. By applying averaging, you can reduce signal fluctuations, improve the signal to noise ratio (SNR), and extract spectral peaks.
- 4) **Waveform Monitoring—Limit Testing, Triggering, and Peak Detection**—allow you to quickly setup pass/fail type tests with results that depend on the time or frequency content of a signal. Commonly applied in telecom and maintenance applications, these tools compare a test signal with a predefined mask that specifies high and low values as a function of time or frequency. Pass and fail states are output during a test depending on whether or not a test signal exceeds or goes below these mask values at a particular frequency. To ease the creation of masks, the measurement library includes two methods of specifying a mask: either as a set of arrays or as a function. Waveform monitoring VIs also include tools for triggering and peak detection of waveforms. The triggering VI will be familiar if you have experience with digital oscilloscopes. It allows you to check for trigger conditions on a waveform. Inputs allow you to specify level, hysteresis, and slope.
- 5) **Filtering**—allow you to apply infinite impulse response (IIR) and finite impulse response (FIR) digital filters to signals. These tools feature the flexibility to design a filter in several different ways. For instance, with an IIR filter one method of choosing a filter is to select an order and a set of cutoff frequencies. For flexibility, you can also have the filter VI automatically choose an order based on passband and stopband widths and gains that you specify.
- 6) **Signal Generation**—generate standard signals (sine, square, triangle, etc.). You can also generate multi-tone signals with control over the gain and phase of each component signal. With such signals, you can test frequency response in parallel, rather than one at a time, which can shorten your test time. Other applications include generation of Dual Tone Multi Frequency (DTMF) test signals for telephony and implementation of test stations that comply with standards that require multi-tone test.

Tool Specifics

- **Mask Testing**

Common application: Pass/Fail testing of telecom signals. To ensure the signal integrity of a digital signal, engineers commonly apply limit testing to check the physical parameters of triggered digital pulses.

Other applications: Anywhere you need to judge signal quality based on limits.

Benefits: Limit testing is handy because it can simultaneously check multiple parameters. With limit testing, you can simultaneously verify physical parameters such as risetime, falltime, pulse width, amplitude, overshoot/undershoot, and leading/trailing edge jitter.

Implementation: At its most basic level, limit testing involves comparison of an input signal with one or more “Masks.” Each mask defines high or low limits as a function of time or other independent variables. Limit tests determine pass or fail conditions depending on if the input signal exceeds or goes below each mask.

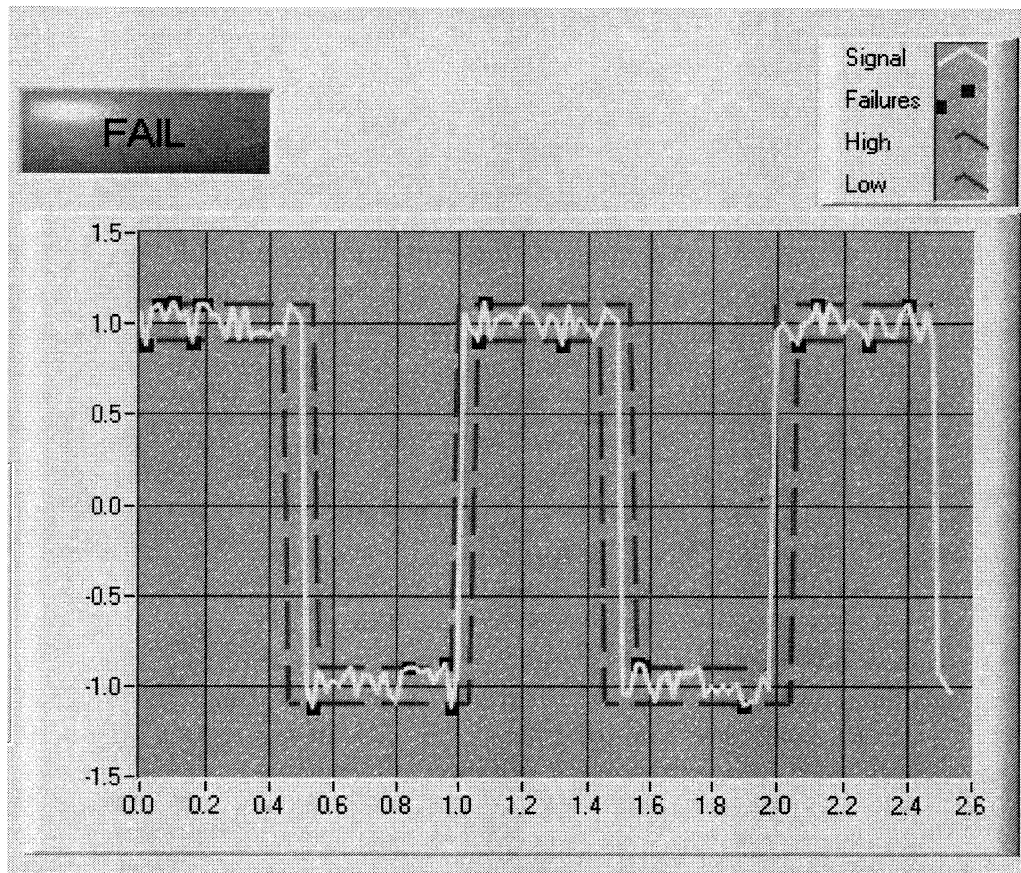


Figure 1: This limit test checks the input signal (green, solid) against high and low limits (red, dashed). It fails because some portions of the input exceed or go below these limits (blue squares).

LabVIEW includes three VIs for limit testing. These VIs allow you to define limits (masks) and apply the limit test. Users can specify limits by inputting numbers or using a formula. The VIs allow limit testing on time-domain, frequency-domain, or any other 1D (XY) signal domain. The Limit Testing VI compares the input signal to high and low limits. By cascading these VIs, users can perform multi-limit testing, which compares the input signal to two or more high/low limits. Such testing is useful for implementing tests with different levels of severity, for instance Pass/Fail/Warning.

- **Tone Detection Measurements**

Applications: One simple application is as a replacement for a Stand-alone Frequency Counter. The general idea is to check for the presence of a particular frequency in a spectrum. Examples include determining the rotational speed of a rotating device or decoding a DTMF signal for telephony.

Benefits: LabVIEW’s tone detection algorithm compensates for aliasing and spectral leakage, which can reduce the error in the estimate and measurement time.

Implementation: LabVIEW contains a VI for Tone Detection called *Extract Single Tone Information.vi*. Given a time domain signal provided as a waveform, it can determine the peak (prominent tone) in that signal. Other inputs include search parameters that allow you to specify a center frequency and a search range to look for a tone in a particular area of the spectrum and a selector that allows you to specify what (besides the frequency estimate of the tone) to output. The VI outputs the frequency/phase/amplitude estimates of the prominent tone, and can also output the input signal, the detected tone (filtered from the input signal) or the input signal with the tone removed. *Extract Single Tone Information.vi* uses a proprietary tone detection algorithm to improve speed. In addition, it applies windowing (specifically a Hanning window) to reduce error due to spectral leakage. This VI joins an older VI called *Power and Frequency Estimate.vi* that works with arrays and uses a different tone detection algorithm.

- **DC/RMS Measurements**

Applications: DC measurements define the value of a static slowly moving signal. Typically you expect this value to be somewhat independent of the measurement time (within a time window). RMS (root mean square) measurements give you a representation of the energy of a dynamic signal—a signal with relatively fast changes such as noise or periodic signals.

Benefits: To understand the benefits of the LabVIEW DC/RMS measurement VIs, you must understand some “gray areas” associated with DC and RMS measurements. Real world DC signals are never purely DC, they often include some dynamic component. In addition, real world AC signals on which you might want to perform an RMS measurement often include some DC component (commonly called an offset).

Error sources in DC measurements	Error sources in RMS measurements
Noise (of any kind)	DC (offset error)
Hum (50 or 60 Hz main power)	Noise
Time drift (temperature)	Hum
	Spikes (industrial noise)

Luckily, you can compensate for such errors with analysis techniques, such as those found in LabVIEW.

Implementation: LabVIEW ships with VIs for DC/RMS measurement: Basic Averaged DC/RMS.vi and Averaged DC/RMS. The highlights of these VIs include the ability to balance accuracy needs with measurement speed by controlling averaging time and selecting a time-domain window as well as block (array) or single point operation. The Basic Averaged DC/RMS VI outputs DC and RMS measurements as single values, while the Advanced Averaged DC/RMS VI outputs results as waveforms and allows more control over windowing.

- ***Frequency Analysis Measurements***

Applications: Anytime you want to control the frequency content of a signal. Frequency analysis is common used in analysis of sound, vibration, biological signals, communication, and many other areas.

Benefits: As high-level waveform tools, these VIs avoid added programming often involved in working with lower-level tools. For instance, if you were to directly call an FFT routine, you would likely also want to remove negative frequencies, calculate an averaged spectrum, determine magnitudes, apply a time-domain window, and scale the output (dB or linear). With these functions, you can control and apply these extras simply by setting input parameters. Another highlight of these VIs is the use of an FFT algorithm that is a prime-factor variety that speeds up calculations for common array sizes (non power of two). To reduce variations or improve the signal to noise ratio, you can optionally apply averaging. All of the Frequency Analysis tools offer three types of averaging:

- **RMS Averaging**—Can reduce signal variations, but doesn't affect the noise floor.
- **Vector Averaging**—Can reduce the noise floor (improve the signal to noise ratio) for triggered signals
- **Peak Hold**—Retains the RMS peak levels of the averaged quantities. Peak hold is performed at each frequency line separately, retaining RMS peak levels from one FFT record to the next. Useful in locating the prominent tone.

Implementation: Eight VIs all produce ready-to-graph results. Tools include:

- Power Spectrum
- Power Spectral Density
- FFT Spectrum (Magnitude + Phase Outputs)
- FFT Spectrum (Real + Imaginary Outputs)
- Frequency Response Function (Magnitude + Phase Outputs)
- Frequency Response Function (Real + Imaginary Outputs)
- Cross Spectra (Magnitude + Phase Outputs)
- Cross Spectra (Real + Imaginary Outputs)

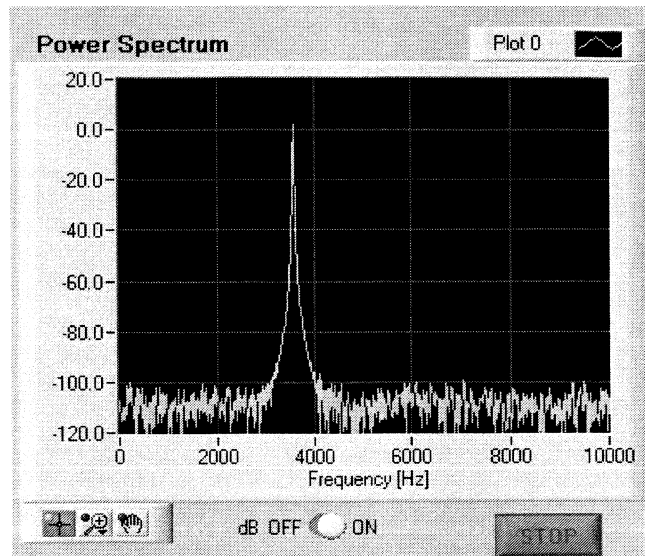
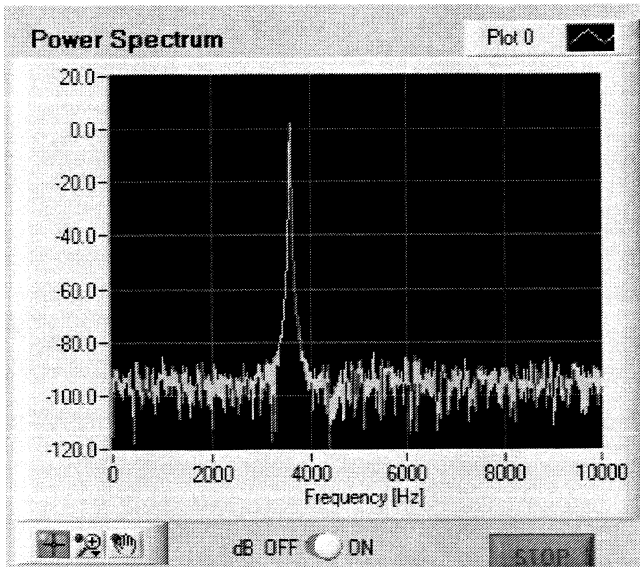


Figure 2: Vector averaging can reduce the noise floor of a signal. Note the improvement in the signal on the right.

- **Signal Generation**

Applications: Useful for stimulus/response tests, such as determining a transfer function. Also beneficial anywhere you need to generate a simulated signal.

Benefits:

- Work directly with NI-DAQ analog output functions
- Define common signals (sine, square, triangle, noise, and others) by parameters
- Define custom signals by specifying a function
- Multitone signal generation with control over phase relationship between tones

Implementation: The signal generation VIs include:

- Function Generator—a single VI that can generate sine, triangle, square, and sawtooth waves
- Formula Waveform—creates a waveform based on a user-defined formula
- Sine Waveform/Triangle/Square/Sawtooth Waveform—Four VIs that break up the action of the Function Generator VI
- Basic Multitone/Basic Multitone with Amplitudes/Multitone Generator—Three VIs for generating signals containing multiple sinusoidal frequency components
- Uniform/Gaussian White Noise, Periodic Random Noise Waveform—Three VIs for generation of noise signals

- **Filtering**

Applications: Frequency-selective filtering of waveform signals.

Benefits: Perform JIT (Just In Time) filter design, which means that LabVIEW can design filters based on a set of logically-selected design parameters. Specifically, with an IIR filter, you can specify cutoff frequencies (and other design parameters) and LabVIEW will automatically select

an optimal order filter that fits your requirements. Another highlight of these VIs is that they are polymorphic, meaning that they automatically handle either single or arrays of waveforms.

Implementation:

- FIR (Finite Impulse Response)—designs and applies FIR (linear phase) filters to waveforms
- IIR (Infinite Impulse Response)—designs and applies IIR (non-linear phase) filters to waveforms
- Scaled Window—Applies a scaled time-domain window to waveform(s) to reduce spectral leakage



ni.com/seminars (512) 794-0100 • Fax (512) 683-5794 • info@ni.com

 This document represents a commitment from National Instruments to the environment.

© Copyright 2000 National Instruments Corporation. All rights reserved. Product and company names listed are trademarks or trade names of their respective companies.



350150F-01 Sep00