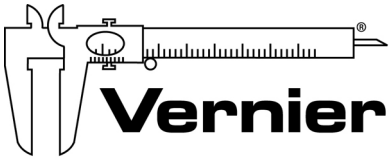
Hands-On Introduction to NI LabVIEW™ with Vernier®



Measure. Analyze. Learn.™

Vernier Software & Technology
13979 S.W. Millikan Way • Beaverton, OR 97005-2886
Toll Free (888) 837-6437 • (503) 277-2299 • FAX (503) 277-2440
info@vernier.com • www.vernier.com

Hands-On Introduction to NI LabVIEW™ with Vernier®

Proper safety precautions must be taken to protect teachers and students during the activities described herein. Neither the authors nor the publisher assumes responsibility or liability for the use of material described in this publication. It cannot be assumed that all safety warnings and precautions are included.

Published by
Vernier Software & Technology
13979 S.W. Millikan Way
Beaverton, OR 97005-2886
Toll free (888) 837-6437
(503) 277-2299
FAX (503) 277-2440
info@vernier.com
www.vernier.com

978-1-929075-21-8

Second Edition
Second Printing
Printed in the United States of America

Contents

Appendices

Preface

This book provides a hands-on introduction to National Instrument's LabVIEW Education Edition[1] software. It uses the Vernier SensorDAQ or LabQuest interfaces and Vernier sensors for data collection and analysis as you learn the basics of LabVIEW. We are convinced that learning LabVIEW while working with sensors is a great way to introduce students to the field of engineering. It is also a lot more fun than working with simulated data.

In 2003, we started conducting workshops with National Instruments in which we used LabVIEW with Vernier sensors. In these training courses, we introduced the basics of LabVIEW programming, followed by exercises that showed how to use Vernier interfaces and sensors for measurement and automation. The training culminated with a hands-on project. We take the same approach in our two books introducing engineering students to LabVIEW.

This book is designed to introduce students to LabVIEW programming. It begins with opening and running a fairly complete data-collection program written in LabVIEW. This is followed by seven chapters describing the basics of LabVIEW programming. Topics include the front panel and block diagram, dataflow programming, working with controls and functions, LabVIEW structures, and creating subVIs. Each of these chapters has two parts. First we explain the new material and at the end of the chapter there is a hands-on exercise with step-by-step instructions. In the exercise, students build a program that performs simple data acquisition with a Vernier sensor.

The LabVIEW material in this book is based on National Instrument's *Introduction to LabVIEW 8 in 6 Hours* document, as well as their online student training entitled *Getting Started with NI LabVIEW Student Training* (a compilation of tutorial, video, and exercises).

The only sensors required for the exercises and projects in this book are a temperature sensor and a microphone. One chapter, written exclusively for SensorDAQ, also requires a voltage probe. These sensors were chosen because they are inexpensive and can be used in many different ways. See *Appendix C* for additional details about the Vernier equipment used in this book.

The last chapter of this book presents the student with two open-ended projects. Creating a project allows students to consider design requirements, manage time constraints, work through troubleshooting challenges, and manage all of the other factors involved in real-world situations. The exercises provide the foundation, and the projects show the possibilities for the next level of use of these tools. Also, the projects are fun and motivational for students.

The first printing of this book was written for a Vernier SensorDAQ interface. Vernier worked with National Instruments to design the SensorDAQ interface. It was designed specifically for engineering education and automatically detects and configures Vernier sensors when used with LabVIEW. The result is that students can get started very quickly developing data collection VIs. The SensorDAQ also has screw terminals for connections to other inputs and outputs.

In this edition of the book, we have added support for the use of Vernier LabQuest interfaces. We have developed LabVIEW drivers and examples to allow most of the activities in this book

[1] Although we recommend LabVIEW Education Edition, other versions of LabVIEW that can communicate with Vernier interfaces and sensors may be used with this book. Note that the screenshots may look slightly different if you use a different version of LabVIEW.

to be done just as they are performed with the SensorDAQ. The only chapter that cannot be done using a LabQuest interface is Chapter 8, because it requires the use of the screw terminals on the SensorDAQ.

We feel it is important for teachers to read the information presented in the appendices. The appendices include valuable information that can help you become more comfortable with your initial use of Vernier sensors and LabVIEW software. Here is a short summary of the information available in each appendix:

- *Appendix A* provides instructions on how to use the files found on the CD.
- *Appendix B* provides a list of the equipment and supplies used in these activities.
- *Appendix C* provides information on Vernier products for Engineering Education.
- *Appendix D* provides information on the required software.
- *Appendix E* provides a brief discussion of the screenshots and how they may differ from what you see when you open LabVIEW.
- *Appendix F* describes common errors that occur when working with the SensorDAQ Express VI and the DAQ Assistant Express VI.
- *Appendix G* provides a summary of an actual classroom experience using this book to teach LabVIEW.

We recently completed a second book, *Engineering Projects with NI LabVIEW and Vernier*, which is intended for teaching engineering education in college or advanced high school classes. *Engineering Projects with NI LabVIEW and Vernier* was designed to follow this book, as it assumes some experience with LabVIEW programming. It offers a number of open-ended projects that introduce engineering concepts, common electrical circuits, designing simple fixtures, and more advanced LabVIEW programming to students. We feel these projects are a great way to get students excited about engineering.

The material for this book was compiled and created by Sam Swartley. He has been working with LabVIEW since version 5.1 and was the Vernier team leader for the development of the SensorDAQ interface. This edition was modified by Dave Vernier, who has been using LabVIEW since version 4. Edits, feedback, and material for the book also came from Steve Decker, a high school engineering teacher in Portland, and member of the Vernier Engineering Education department. Michele Perrin, a former engineering, math, and science teacher in Saint Louis, Stephanie Brierty, K-12 Product Marketing Engineer at National Instruments, and Peter Tampas, Professor Emeritus of Electrical Engineering Technology, Michigan Technological University also provided valuable feedback and edits. We are thankful to Gretchen Stahmer DeMoss, John Wheeler, and Christine Vernier for the help they provided in proofreading this book. Special thanks to Ray Almgren and Ravi Marawar of National Instruments, and David Vernier, CEO and founder of Vernier Software & Technology, for supporting this project and for their continued support of STEM education.

Chapter
1

Getting Started

ABOUT LABVIEW

LabVIEW is a graphical programming language used by millions of engineers and scientists to develop sophisticated measurement, test, and control programs. LabVIEW offers integration with thousands of hardware devices and provides hundreds of built-in libraries for advanced analysis and data visualization. The LabVIEW platform is scalable across multiple targets and operating systems, and since its introduction in 1986, has become an industry leader.

Educators use LabVIEW for teaching engineering, student design projects, and research projects. LabVIEW allows hands-on investigation of engineering concepts by acquiring a signal, performing analysis, and visualizing the data.

LabVIEW is a graphical programming language that uses icons instead of lines of text to create programs. In contrast to text-based programming languages, where instructions determine program execution, LabVIEW uses dataflow programming, where the flow of data determines execution order.

LabVIEW programs/subroutines are called virtual instruments (VIs). A LabVIEW VI represents a fundamental shift from traditional hardware-centered instrumentation systems to software-centered systems that exploit the computing power, productivity, display, and connectivity capabilities of popular desktop computers and workstations. With LabVIEW VIs, engineers and scientists build measurement and automation programs that suit their needs exactly (user-defined) instead of being limited by traditional fixed-function instruments (vendor-defined).

The LabVIEW Education Edition software helps teachers bring Science, Technology, Engineering, and Math (STEM) to life through hands-on learning. With LabVIEW, you can quickly build a program to log data, power a robot, or analyze information. The new LabVIEW Education Edition was designed in conjunction with Tufts Center for Engineering Education and Outreach to meet the needs of engineering educators, and works seamlessly with products such as Vernier SensorDAQ, Vernier LabQuest interfaces, Vernier Go! sensors, and the LEGO NXT Intelligent Brick.

ABOUT VERNIER SENSORS AND INTERFACES

Vernier Software & Technology produces a wide variety of sensors for science and engineering education. These sensors work with several different “lab interfaces,” which are electronic devices that connect to the USB port of a computer. The interface performs the analog to digital conversion and timing for data collection. It may also control analog or digital output lines. This book supports Vernier SensorDAQ and all Vernier LabQuest interfaces. Each of these interfaces has three or four connectors for analog sensors. Examples of analog sensors available from Vernier include Voltage, Microphone, Temperature, pH, Force, and Heart Rate Sensors. LabQuest interfaces also have two connectors for digital sensors. SensorDAQ has one connector for digital sensors. Examples of digital sensors include Motion Detectors, Photogates, Rotary Motion Sensors, and Radiation Monitors. Vernier Digital Control Unit (DCU) can also be connected and used to provide sufficient current for controlling external electrical devices.

In addition to the analog and digital sensor connectors, SensorDAQ includes a screw terminal connector. The screw terminal channels include digital input/output, analog output, counter/timer, +5 volt line, and two analog input channels. These connectors can be used to build circuits, create custom sensors, control RC servo motors, turn on electronic devices, and more.

Exercise
1

Open and Run an Example VI

In this exercise, you will open and run an advanced LabVIEW example VI that performs data logging with a Vernier Stainless Steel Temperature Probe connected to the SensorDAQ or LabQuest interface. Data analysis can then be performed on the logged data. In addition, this example provides feedback that the interface and sensor are properly connected.

Most of the examples in this book and on the Vernier web site are very simple. They are kept simple so that they are easier to understand. This VI is different; it is a fairly advanced example that displays some of the power of LabVIEW. It is designed as a program that can be used in the classroom for data logging and analysis.

OBJECTIVES

In this exercise, you will

- Open and run an example VI.
- Take measurements.
- Receive feedback on your hardware connection.

MATERIALS

SensorDAQ or LabQuest interface
Vernier Stainless Steel Temperature Probe
LabVIEW
computer
USB cable

PROCEDURE

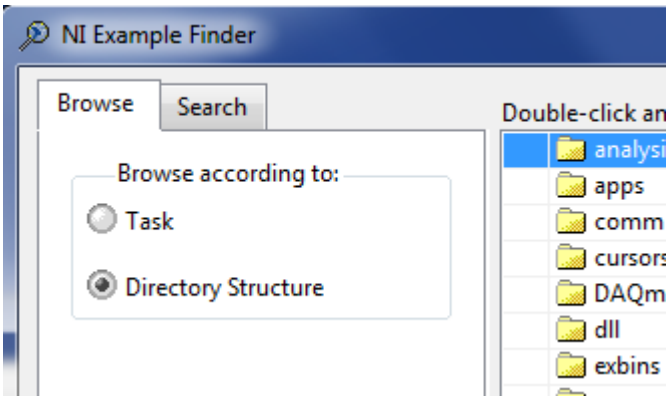
Part I Connect Hardware

1. Connect the USB cable to the SensorDAQ or LabQuest interface.
2. Connect the other end of the USB cable to any available USB port on your computer. If you are using a LabQuest 2 or the original LabQuest, you must press the Power button to activate the interface. The SensorDAQ and LabQuest Mini are powered on whenever they are connected to the computer. If you are using a SensorDAQ, the green LED (next to the USB cable port) should be blinking. If you are using a LabQuest Mini, the orange LED should be on.
3. Connect the Temperature Probe to Ch. 1.

Part II Start LabVIEW and Collect Data

4. Start LabVIEW.

5. Launch the NI Example Finder by choosing Find Examples from the Help menu.
6. Select Directory Structure in the upper-left corner. From the list of folders, choose Vernier.

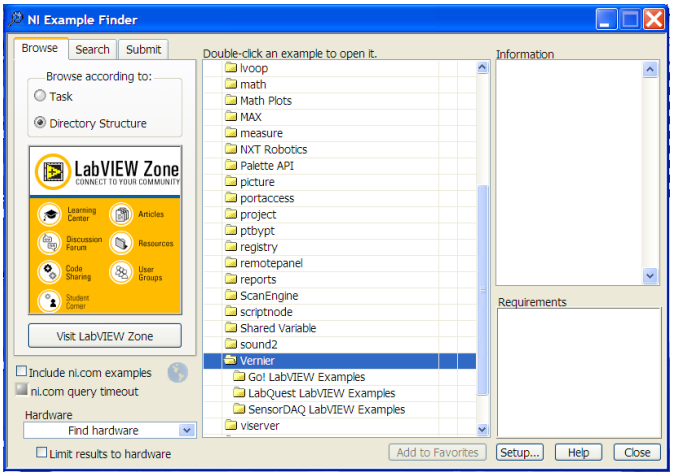


SensorDAQ

If you are using SensorDAQ, open the folders SensorDAQ LabVIEW Examples ► Log and Analyze Data ► Log with Analysis and single-click SensorDAQ Logger.vi to view the description of this example in the Information box.

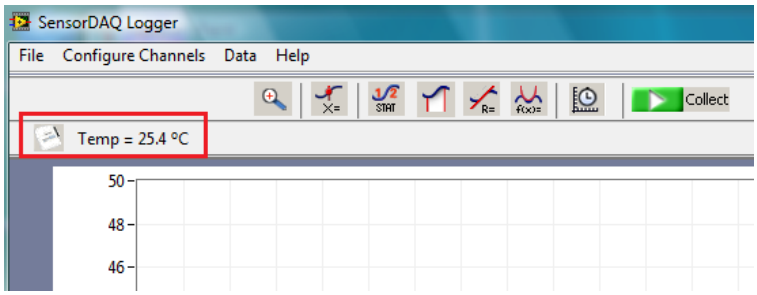
LabQuest Interface

If you are using a LabQuest interface, open the folders LabQuest LabVIEW Examples ► Log and Analyze Data ► Log with Analysis and single-click the LQ Logger.vi to view the description of this example in the Information box.



7. Double-click the Logger file to open it.
8. Start the example by clicking LabVIEW's Run arrow,  , in the upper-left corner.

Tip: If LabVIEW has successfully detected the interface, you will see an interface icon in the upper-left corner (see below), along with the sensor reading. In the example below, a Temperature Probe was connected.



9. Click Collect, . LabVIEW will begin plotting data in the graph.

10. Once data have been collected, click and drag on the graph to highlight data. Click the various analysis buttons in the toolbar (Zoom, Examine, Statistics, Integrate, Linear Fit, and Curve Fit) to study your data.

11. Stop this VI by choosing Exit from the File menu.

12. When you are ready to move to the next chapter, close LabVIEW.

EXTENSIONS

1. Click Data Collection, . Modify the length and sampling rate, and then collect some new data.

2. Store several runs of data and analyze them using the Linear Fit, Statistics, and Examine analysis features.

3. The Data Collection window has a triggering tab that allows you to set a data-collection trigger. Collect data that is triggered to start when the temperature has reached a user-defined limit.

4. Open the NI Example Finder by choosing Find Examples from the Help menu. Open and run the program called “Tangent Analysis.vi” that is found in the Log with Analysis folder.

Chapter
2

LabVIEW Environment

LabVIEW programs are called virtual instruments (VIs). Each VI has two windows: a front panel that contains the user interface, and a block diagram with the graphical code. This chapter discusses how to operate within these two windows; as with each chapter in this book, it is best to read the chapter through and then to perform the activity that follows. During the activity you will be able to try out the features described in the chapter.

FRONT PANEL

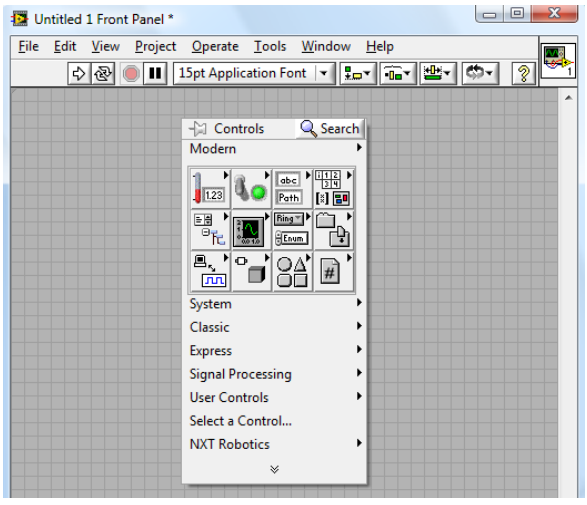
When you open a new or existing VI, the front panel window of the VI appears and functions as the graphical user interface (GUI) of a VI. Controls and indicators are placed on the front panel, and are the interactive input and output terminals respectively. They are automatically given corresponding terminals on the block diagram. The front panel controls and indicators are collectively referred to as objects. Refer to the "Block Diagram" section below for more information on block diagram terminals.

LabVIEW programmers design the front panel as they develop the VI. Users interact with the front panel when the program is running. Users can control the program, change inputs, and see data updated in real time. It is important for a VI to have an intuitive and easy-to-read front panel. The front panel is essentially the gateway for all user input and output of a VI; therefore, it is essential that the programmer has a good grasp of how to effectively design a front panel.

The front-panel window contains a toolbar across the top and a Controls palette that you can access by right-clicking anywhere on the front panel. You can recognize the front panel by its grey background and grid.

Front Panel Controls Palette

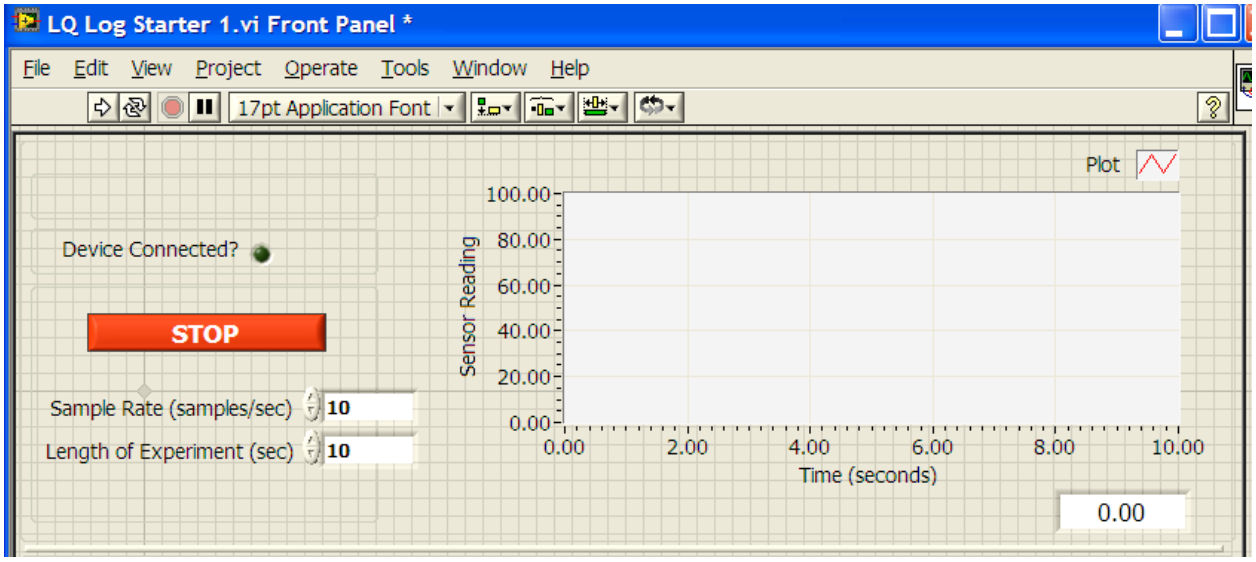
Use the Controls palette to place controls and indicators on the front panel. The Controls palette is available only on the front panel. To view the palette, choose Controls Palette from the View menu. You can also display the Controls palette by right-clicking an open area on the front panel. Tack down the Controls palette by clicking the pushpin on the upper-left corner of the palette. This will ensure that the Controls palette is always visible on the front panel.



After opening the Controls palette, use it to place controls and indicators on the front panel by dragging and dropping the icons from the palette into the front panel workspace.

Tip: Using the pushpin to pin the Controls palette to the front panel provides a View button to change what controls and indicators are visible on the Controls palette, and a Search button if you have trouble finding a specific control or indicator. You can also have multiple palettes open to have your most commonly used subpalettes available simultaneously.

Controls (knobs, push buttons, dials, and other input devices) are the interactive input terminals, while indicators (graphs, LEDs, and other displays) are the interactive output terminals of the VI. Controls simulate instrument input devices and supply data to the block diagram of the VI. Indicators simulate instrument output devices and display data the block diagram acquires or generates.



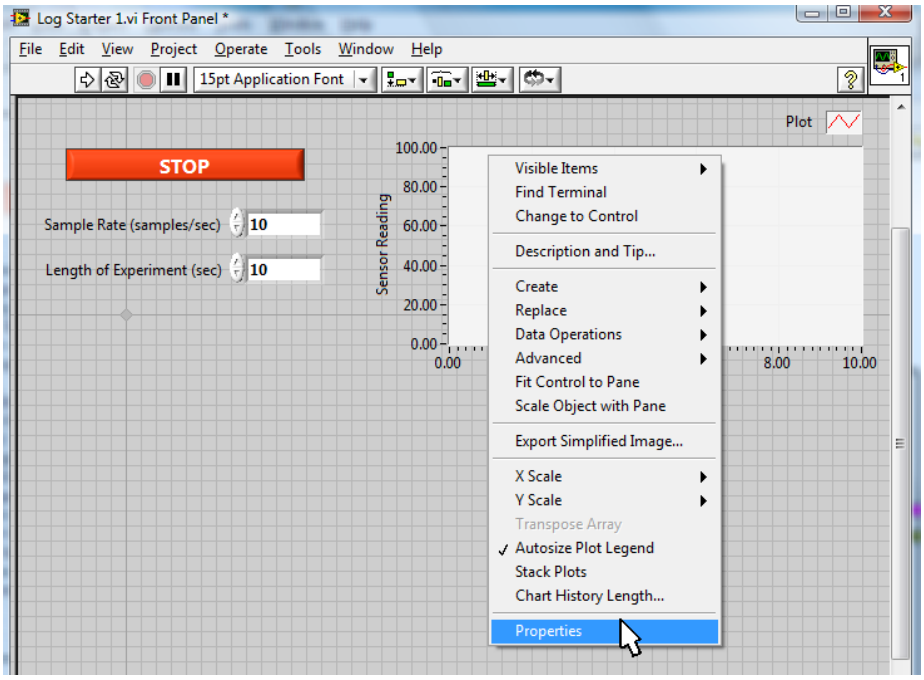
The figure above has three controls labeled STOP, Sample Rate, and Length of Experiment. The user can click the STOP button to change this control's value from True to False. The Sample Rate and Length of Experiment controls can be adjusted by clicking the up/down arrows, or by typing new values in the text box.

There are two indicators: a waveform chart and an LED labeled Device Connected. The user can see the value generated by the VI (in the block diagram) on the chart indicator. In addition, the user will have feedback from the LED on whether the SensorDAQ or LabQuest interface is connected (LED will turn bright green) or not connected (LED remains dark green).

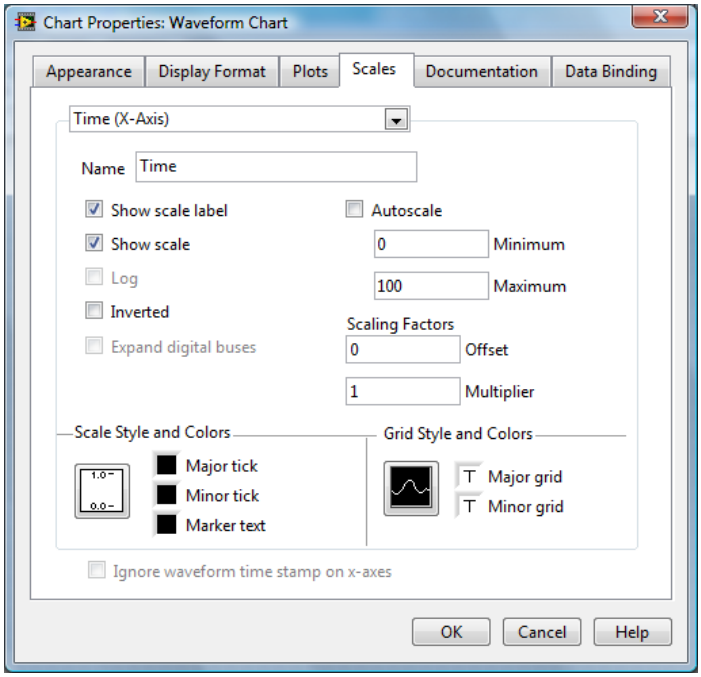
Front Panel Control Properties

All LabVIEW objects (e.g., controls and indicators) have associated shortcut menus and property dialog boxes. As you create a VI, use the shortcut menu items and/or the properties dialog box to change the appearance and/or behavior of front panel and block diagram objects. To access the

shortcut menu, right-click the object you want to modify. To access the Properties dialog box, select Properties from the shortcut menu.



For example, in the figure above, the user might modify some chart properties (such as the chart's grid styles and axis labels) by right-clicking the chart, and selecting Properties from the shortcut menu.



After clicking on Properties, the Properties dialog box opens and provides a way to modify the chart. The tab shown in the figure above provides scale properties that can easily be modified.

Front Panel Window Toolbar

Each window has a toolbar associated with it. Use the front panel window toolbar buttons to run and edit the VI. The following toolbar appears on the front panel window.

 Click the Run button to run your VI. You can run a VI if the Run button appears as a solid white arrow, shown at left.

 The Run button appears broken when the VI you are creating or editing contains errors. If the Run button still appears broken after you finish wiring the block diagram, the VI is broken and cannot run. Click this button to display the Error List window, which lists all errors and warnings.

 Click Run Continuously to run the VI until you abort or pause execution. You can also click the button again to disable continuous running. Be aware that this is not a good method for running a VI continuously; you should instead create code that provides this feature.

 While the VI runs, the Abort Execution button appears. Click this button to stop the VI immediately if there is no other way to stop the VI. If more than one running top-level VI uses the VI, the button is dimmed.

Caution: Avoid using the Abort Execution button. It stops the VI immediately before it finishes the current iteration. Aborting a VI that uses external resources, such as external hardware, might leave the resources in an unknown state by not resetting or releasing them properly. Design VIs with a STOP button to avoid this problem.

 Click Pause to pause a running VI. When you click the Pause button, LabVIEW highlights on the block diagram the location where you paused execution, and the Pause button appears red. Click the Pause button again to continue running the VI. Again, be aware that this might leave resources in a troubled state if you have external hardware.

Select the Text Settings pull-down menu to change the font settings for the selected portions of your VI, including size, style, and color.

 Click the Align Objects pull-down menu to align objects along axes, including vertical and horizontal edges and centers.

 Click the Distribute Objects pull-down menu to space multiple front panel objects evenly.

 Click the Resize Objects pull-down menu to resize multiple front panel objects to the same size.

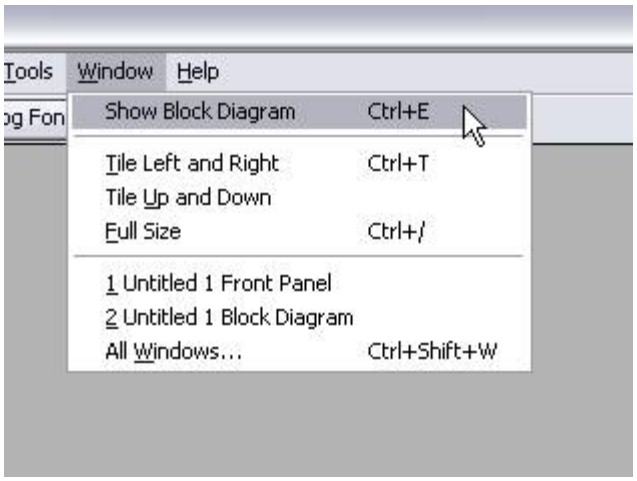
 Click the Reorder pull-down menu when your objects overlap each other and you want to define which one is in front or back of another. Select one of the objects with the Positioning tool and then select from Move Forward, Move Backward, Move To Front, and Move To Back.

 Click the Show Context Help Window button to toggle the display of the context help window. Context Help displays relevant information of an object when you hover your mouse over the object.

 The Enter Text button only appears (on the far left side of the toolbar) when the user has typed on the front panel. Pressing the Enter Text button finalizes the text or value changes. The Enter Text button disappears when you click it. Instead of pressing the Enter Text button, the user may also press the <Enter> key or click the front panel or block diagram workspace.

BLOCK DIAGRAM

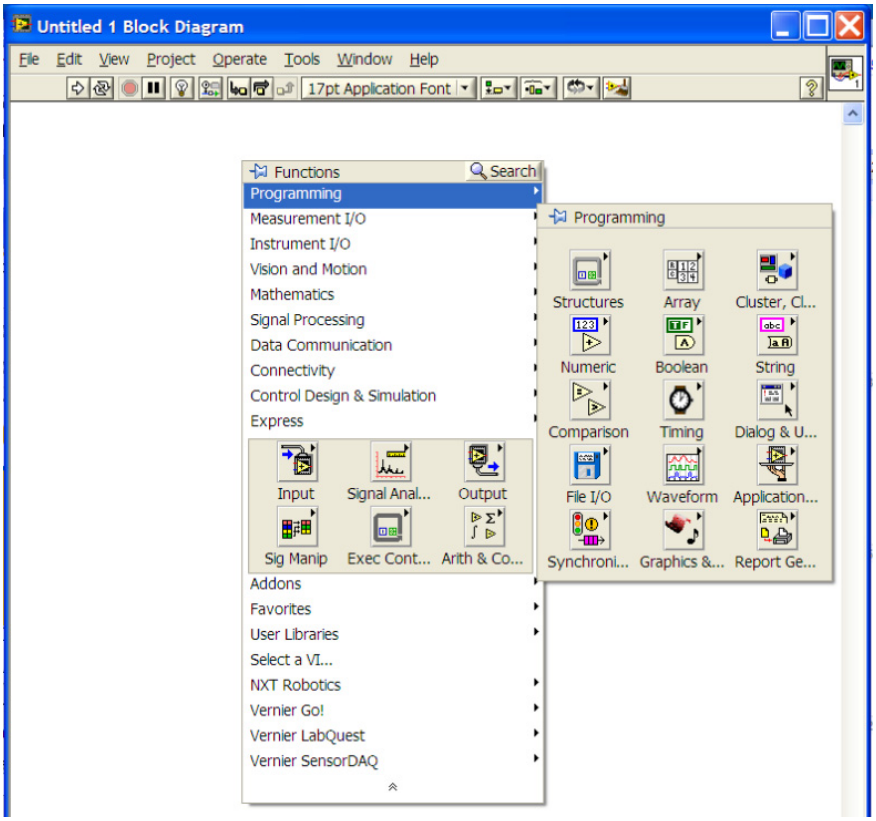
When you create or open a new VI, the front panel opens automatically. To bring up the block diagram, choose Show Block Diagram from the Window menu. Additionally, you can toggle between the block diagram and the front panel by pressing <Ctrl-E>.



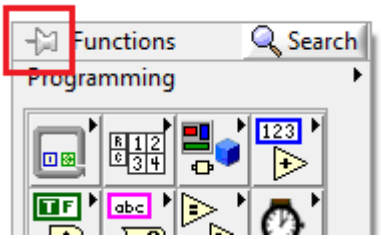
In LabVIEW, you build a user interface with controls and indicators on the front panel. The programmer must then add code using graphical representations of functions to control these front panel objects. The block diagram contains this code. In some ways, the block diagram resembles a flowchart. The concept of the block diagram is to separate the graphical source code from the user interface in a logical and simple manner. You can recognize the block diagram by its white background.

Block Diagram Functions Palette

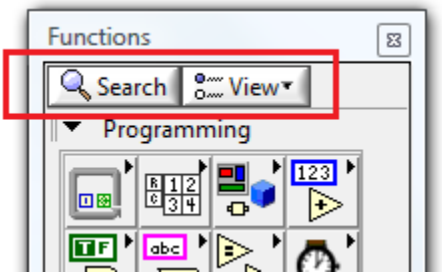
Block diagram objects include terminals (linked to front panel objects), functions, constants, structures, subVIs, and wires that transfer data among other block diagram objects. To place objects on the block diagram, simply drag and drop them from the Functions palette. The Functions palette automatically appears when you right-click anywhere on the block diagram workspace or choose Functions Palette from the View menu. It contains functions, constants, structures, and some subVIs.



The Functions palette appears when you right-click, and will automatically disappear when you left-click on something new. If you prefer to keep the Functions palette visible at all times, there is a pushpin on the top-left corner of the palette that can be clicked.



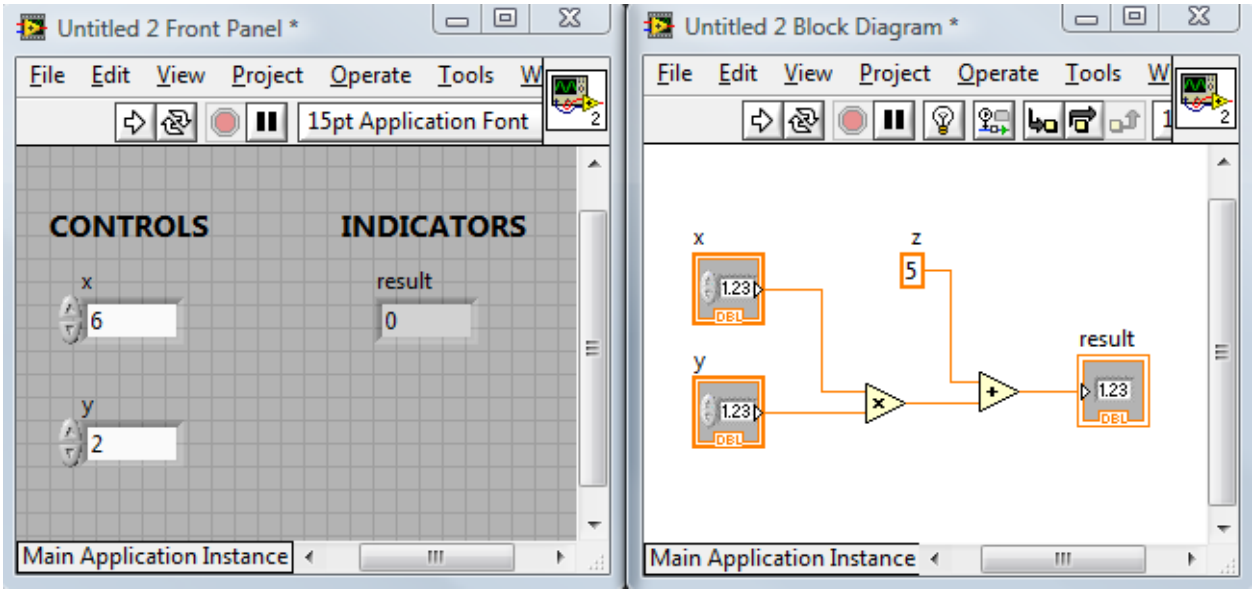
Clicking the pushpin will tack down the palette menu. Tacking the palette menu also provides a View button to change the subpalettes that are visible on the Functions palette, and a Search button if you have trouble finding a function.



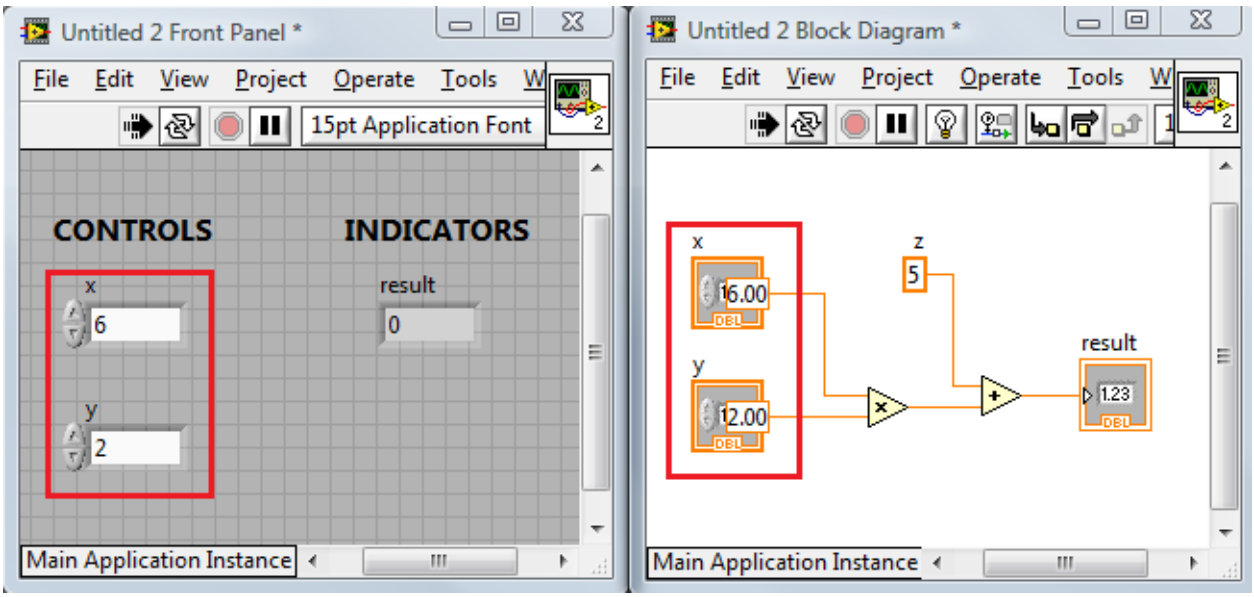
Tip: It can be very helpful to only make a select number of subpalettes visible. Consider only showing the Programming, Express, and Vernier palettes when you are first using the program.

Block Diagram Controls, Indicators and Constants

Front panel objects appear as terminals on the block diagram, and although they appear on separate windows, the two are linked. Block diagram control terminals receive their values from the front panel and pass these values into the block diagram logic. Block diagram indicator terminals receive their values from the block diagram logic and pass their values to the front panel. Constants do not appear on the front panel window. A constant only exists on the block diagram and simply passes a value within the block diagram logic.

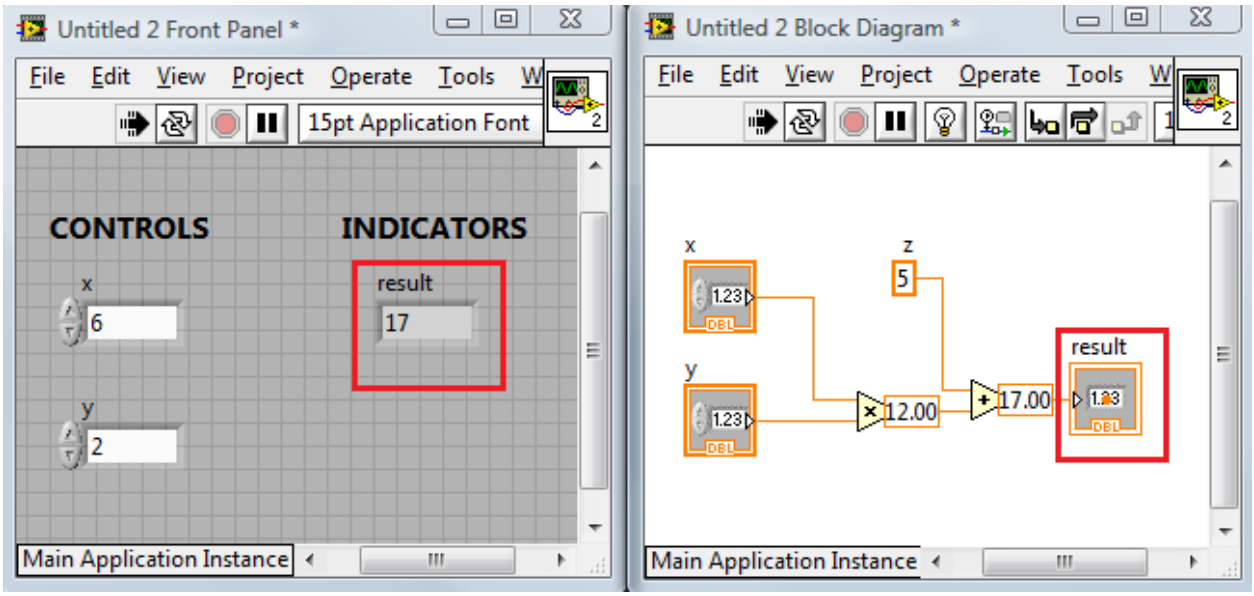


In the figure above, the front panel and block diagram are both shown prior to running this VI. When this VI is run, the data values of the front panel controls, x and y, enter the block diagram through their respective control terminals, x and y.



The data then flow from the terminals to the multiply function. When the multiply function completes the calculation, the new data value flows through the wire to the addition function,

where the constant z, with a value of 5, is added. The new data value leaves the addition function and flows to the indicator terminal called result.

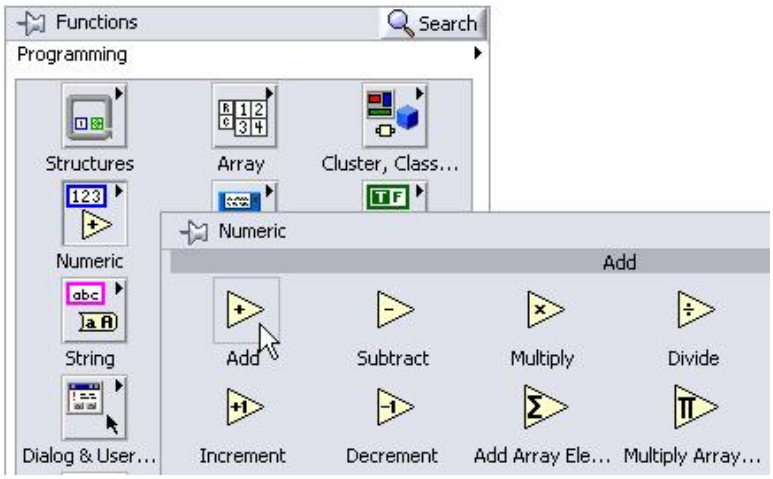


At this point, the front panel indicator linked to the result terminal is updated with the value calculated by the block diagram logic.

Notice that the x and y block diagram terminals look different from the result terminal. There are two distinguishing characteristics between a control and an indicator on the block diagram. The first is an arrow on the terminal that indicates the direction of data flow. The controls have arrows showing the data leaving the terminal, whereas the indicator has an arrow showing the data entering the terminal. The second distinguishing characteristic is the border around the terminal. Controls have a thick border and indicators have a thin border.

Block Diagram Functions

Functions are the fundamental operating elements of LabVIEW. Functions have input and output terminals for passing data in and out. You can tell if a block diagram object is a function by the pale yellow background on its icon. The Functions palette has functions arranged in groups based on the type of function they perform; for example, look in the Numeric subpalette for functions that perform numeric operations.

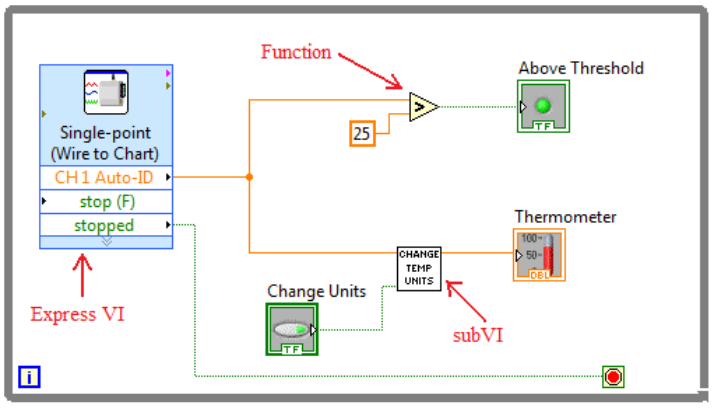


There are many different types of functions. Remember that a function has a pale yellow background like the functions shown below.

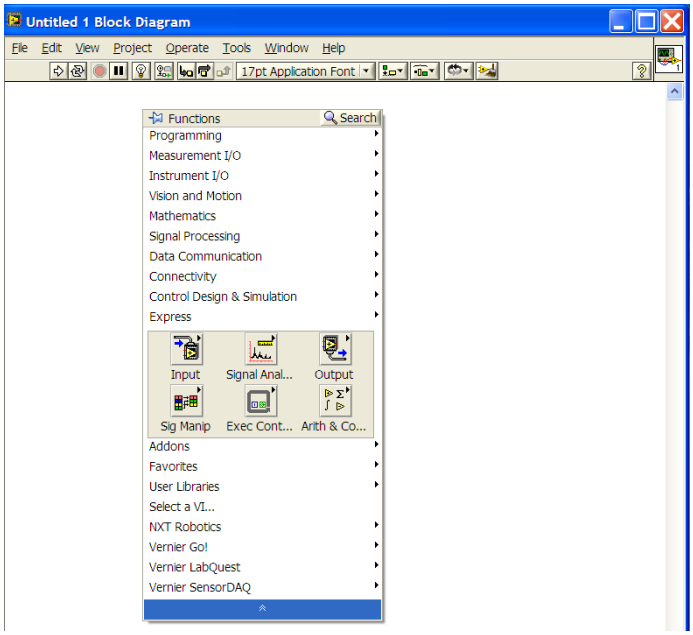
Block Diagram SubVIs and Express VIs

SubVIs are VIs that you create to use inside another VI, or that you access on the Functions palette. Any VI has the potential to be used as a subVI. When you double-click a subVI that is on the block diagram, its front panel window appears and you can access its block diagram. The icon for a subVI can be customized, but in most cases subVIs appear as square blocks. SubVIs are discussed in detail in Chapter 7.

Express VIs appear on the block diagram as expandable nodes with icons surrounded by a blue field. The primary benefit of Express VIs is their interactive configurability through a dialog box that is similar to a wizard or setup assistant. This dialog box appears when you place the Express VI on the block diagram. To re-open the dialog box, simply double-click the Express VI or right-click the Express VI and select Properties from the shortcut menu. You can also configure the Express VI by wiring values to terminals of the Express VI on the block diagram.

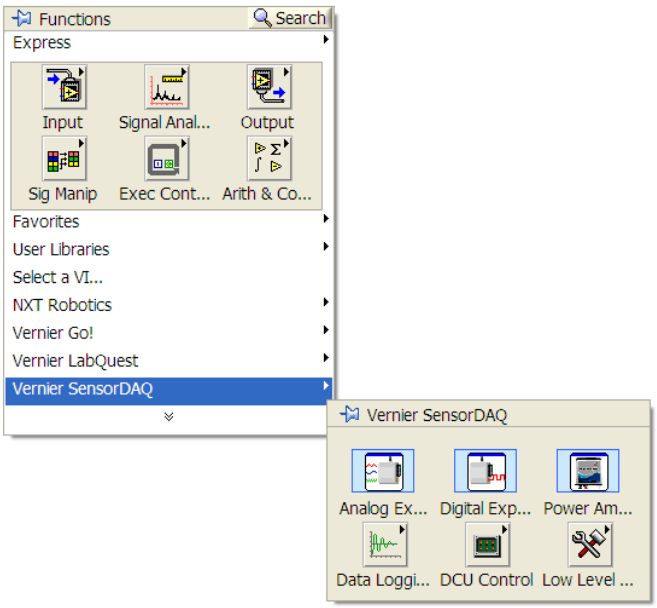


Special palettes for data collection with Vernier interfaces are available. They include standard and Express VIs. They are listed at the bottom of the list when you display the Functions palette.



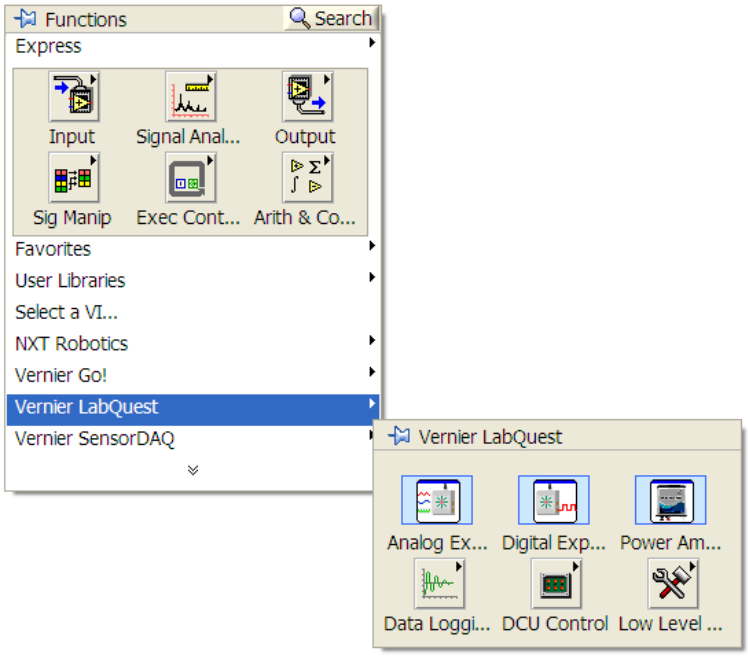
Using SensorDAQ

The SensorDAQ palette contains Express VIs and subVIs. When you are building a custom VI that communicates with the SensorDAQ, you will use these tools.



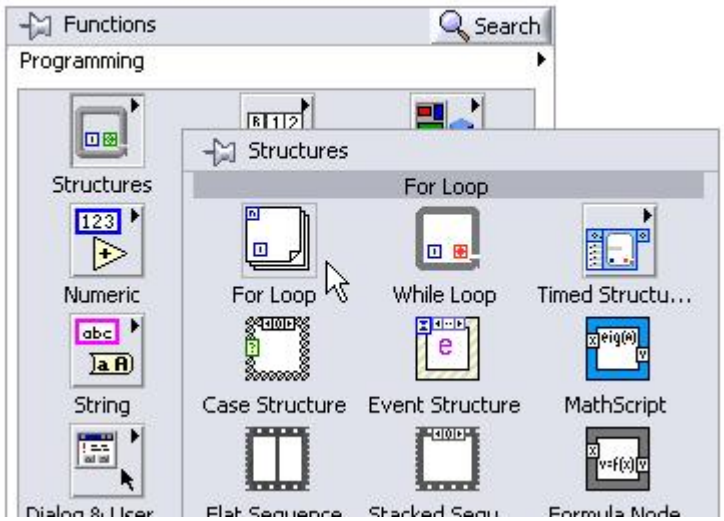
Using a LabQuest Interface

The LabQuest palette contains Express VIs and subVIs for data collection using any of the Vernier LabQuest interfaces.

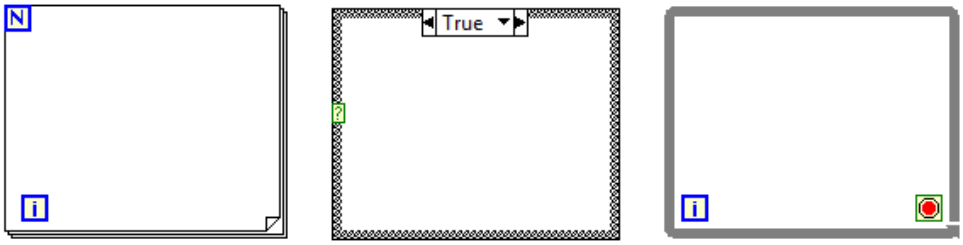


Block Diagram Structures

Structures, which include For Loops, Case Structures, and While Loops, are used for process control. The While Loop is examined in more detail in Chapter 4. The Case Structure is examined in more detail in Chapter 6. You can open the Structures subpalette from the Functions palette under Programming.



Below are a For Loop, Case Structure, and While Loop, as they appear on the block diagram.



Block Diagram Window Toolbar

When you open a VI, the following toolbar appears on the block diagram. Many of these buttons are the exact same as those found on the front panel window toolbar, and were covered on pages 2–4 and 2-5. The buttons not covered in that section, which only appear on the block diagram toolbar, are described below.

Click the Highlight Execution button to see the flow of data through the block diagram. Click the button again to disable execution highlighting.

Click Retain Wire Values button to save the wire values at each point in the flow of execution so that when you place a probe on a wire (the wire can be probed by right-clicking on the wire and selecting Probe from the shortcut menu), you can immediately obtain the most recent value of the data that passed through the wire.

Click the Step Into button to single-step into a loop, subVI, and so on. Single-stepping through a VI steps through the VI node to node. Each node blinks to denote when it is ready to execute. By stepping into the node, you are ready to single-step inside the node.

Click the Step Over button to step over a loop, subVI, and so on. By stepping over the node, you execute the node without single-stepping through the node.

Click the Step Out button to step out of a loop, subVI, and so on. By stepping out of a node, you complete single-stepping through the node and go to the next node.

Click the Clean Up Diagram button to re-route all existing wires and reorder objects on the block diagram for a cleaner, more readable block diagram.

TOOLS PALETTE

Working in LabVIEW requires the use of different tools. You will notice that as you move your cursor over LabVIEW front panel and block diagram objects, the cursor automatically changes icons. These different cursor icons represent LabVIEW tools. For example, if the cursor is near a block diagram terminal, the cursor icon will automatically change to the wiring tool. If the cursor is near a front-panel numeric control, the cursor icon will automatically change to the operating tool.

LabVIEW is configured to automatically select the tool based on the cursor location. In some cases, you will need to disable automatic tool selection and manually select the tool. To do this, you will need access to the LabVIEW tools. Select View ► Tools Palette to display the Tools palette.

Select your tool (such as the coloring tool) by clicking on the Tools palette button that represents the desired tool. When you click on a tool button, the automatic tool selection becomes disabled; therefore, when you are finished using the selected tool you must click the Automatic Tool Selection button to enable this feature again.

 If automatic tool selection is enabled and you move the cursor over objects on the front panel or block diagram, LabVIEW automatically selects the corresponding tool from the Tools palette. Toggle automatic tool selection by clicking the Automatic Tool Selection button in the Tools palette.

 Use the Operating tool to change the values of a control or to select the text within a control.

 Use the Positioning tool to select, move, or resize objects. The Positioning tool changes shape when it moves over a corner of a resizable object.

 Use the Labeling tool to edit text and create free labels. The Labeling tool changes to a cursor when you create free labels.

 Use the Wiring tool to wire objects together on the block diagram.

 Use the Shortcut Menu tool to open the shortcut menu of an object.

 Use the Scrolling tool to scroll the window without using the scroll bars.

 Use the Breakpoint tool to pause execution at a specified location of the block diagram.

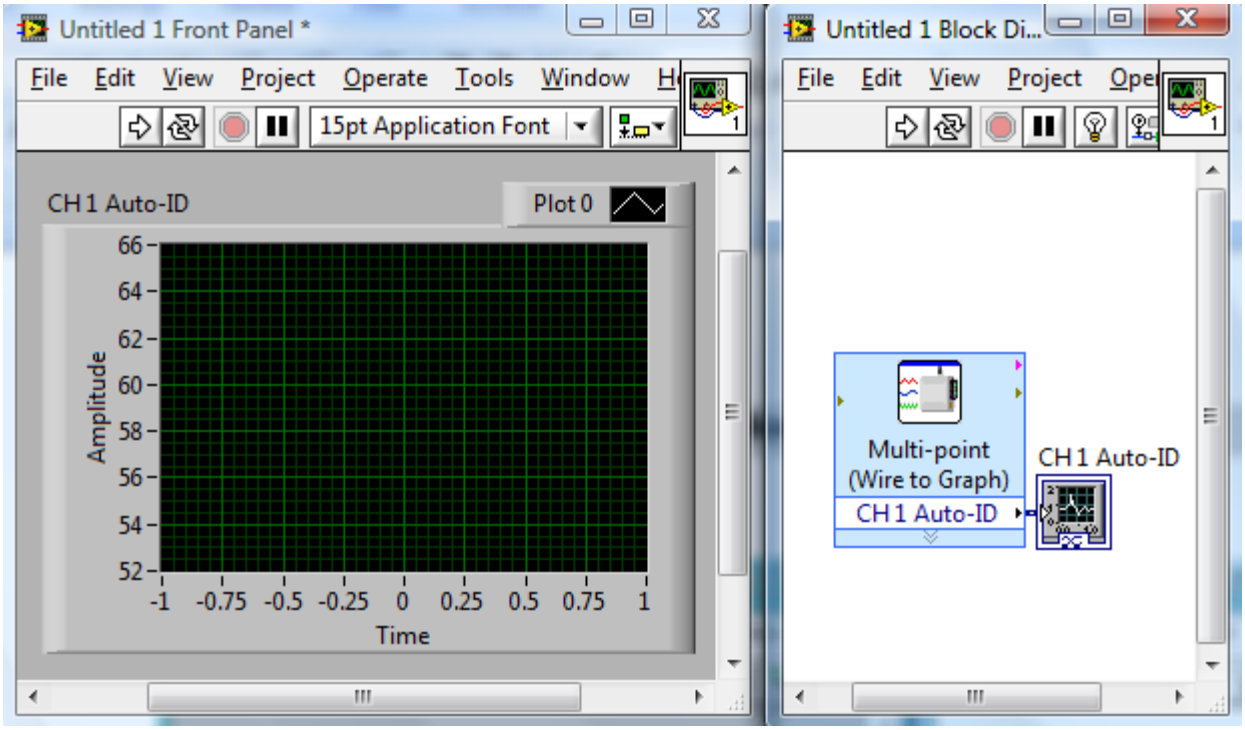
 Use the Probe tool on a wire to check the value of the wire as the program runs.

 Use the Color Copy tool to copy the color into the Coloring Tool palette.

 Use the Coloring tool to set the foreground and background color of front panel objects, front panel panes, and block diagram workspaces.

Exercise
2

Read Microphone Data



Completed front panel and block diagram

In this exercise, you will create a simple program using the Analog Express VI to collect data for a length of 0.1 second at a rate of 10,000 samples/second. This timing configuration results in 1001 data points. It is important to note how the data are displayed in this example. Namely, this is an example of multi-point data collection, where all data points are displayed on the graph following the collection of all 1001 data points. The Express VI is designed to return data in this multi-point method whenever the data-collection rate is greater than 200 samples/second. At slower rates, data are returned on a point-by-point basis.

OBJECTIVES

In this exercise, you will

- Create a LabVIEW VI.
- Become familiar with the LabVIEW environment.
- Acquire Microphone data.
- Display data in a graphical form.

MATERIALS

SensorDAQ or LabQuest interface
USB cable
computer
LabVIEW
Vernier Microphone

COMPUTER PROCEDURE

Part I Connect Equipment

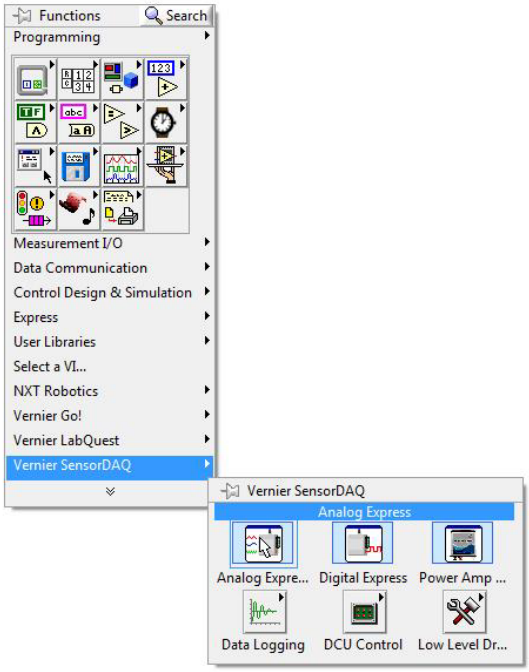
1. Connect the USB cable to the SensorDAQ or LabQuest interface.
2. Connect the other end of the USB cable to any available USB port on your computer. If you are using a LabQuest interface with a power button, turn it on.
3. Connect the Microphone to Ch. 1.

Part II Start LabVIEW and Create a VI to Collect Data

4. Start LabVIEW.
5. In the Getting Started window, click the Blank VI link in the New category.
6. View the block diagram by choosing Show Block Diagram from the Window menu (or use the <Ctrl-E> shortcut).
7. Place an Analog Express VI in the block diagram workspace.

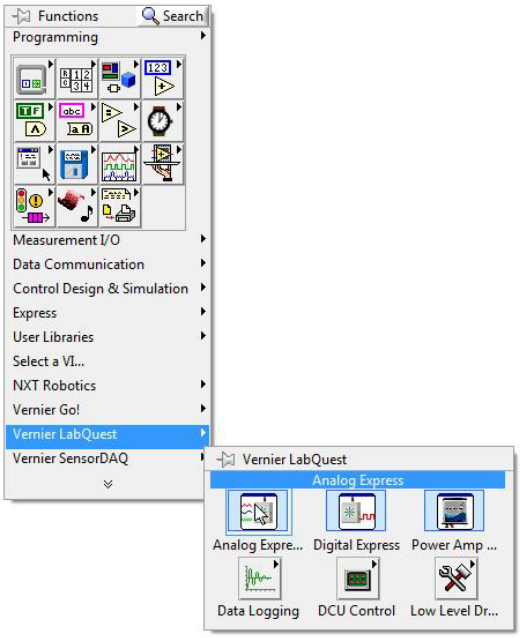
   #### SensorDAQ

   If you are using a SensorDAQ, right-click in the block diagram workspace and select Vernier SensorDAQ from the Functions palette. Click and drag the Analog Express VI to the block diagram workspace.



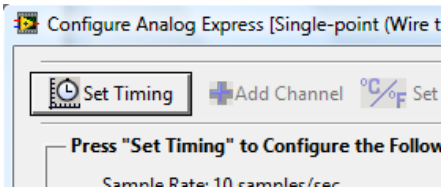
LabQuest Interface

If you are using a LabQuest interface, right-click in the block diagram workspace and select Vernier LabQuest from the Functions palette. Click and drag the Analog Express VI to the block diagram.

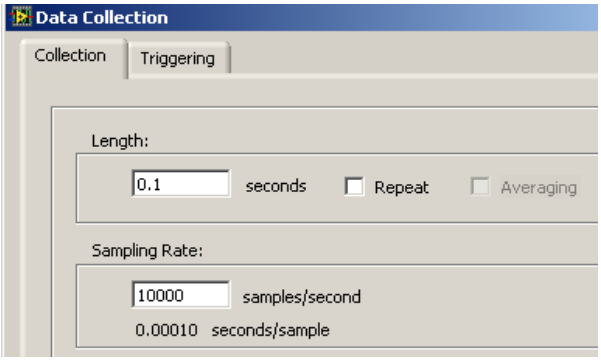


8. After dragging the Express VI from the palette to the block diagram workspace, the Express VI's configuration window will open. Note that this step can be slow, depending on your computer.

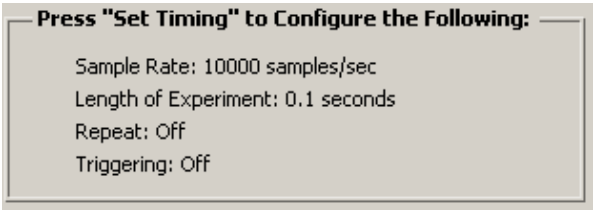
9. Click the Set Timing button, located in the upper-left corner of the configuration dialog.



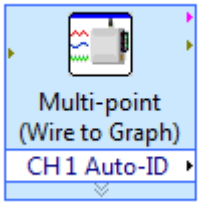
10. Set up data collection for a length of 0.1 seconds and a sampling rate of 10,000 samples/second.



11. Click Done to close the Set Timing dialog. The Express VI Configuration should now be updated with the new settings.

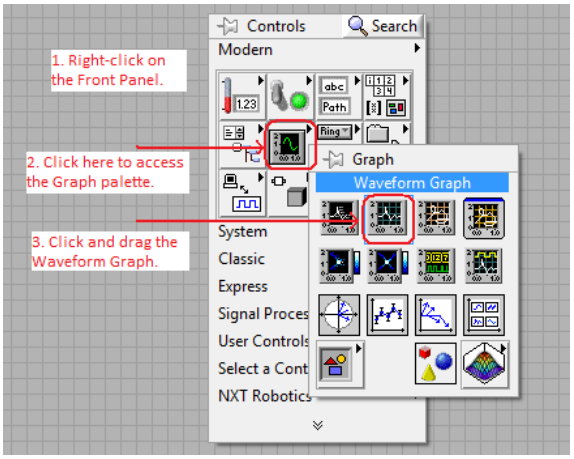


12. Click OK to close the Express VI's Configuration dialog. The Express VI will now be located in your block diagram workspace.

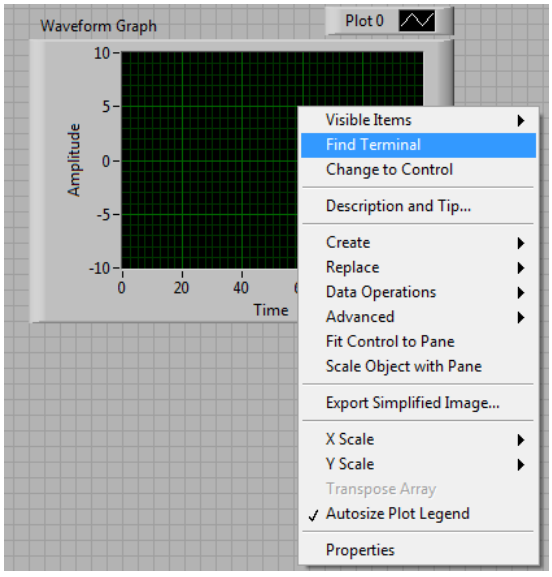


13. View the front panel by choosing Show Front Panel from the Window menu (or use the <Ctrl-E> shortcut).

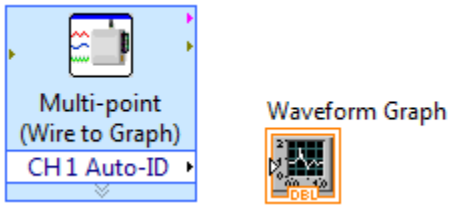
14. Place a Waveform Graph on the front panel. Access the graph by right-clicking in the front panel workspace and finding the Modern ► Graph palette.



15. Right-click the Graph and select Find Terminal. This will take you to the Graph's terminal in the block diagram workspace. You can also double-click the graph to find the corresponding terminal.



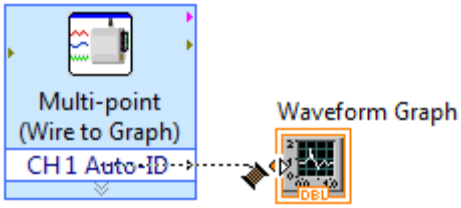
16. Move the graph terminal icon (click and drag the icon with your mouse) and place it to the right of the Analog Express VI. Note that the icon distinguishes this as an indicator (a thin border and an arrow showing the data entering the terminal).



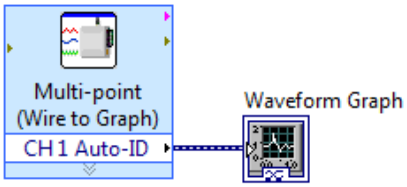
17. Wire the Analog Express VI CH 1 Auto-ID data output terminal to the Graph's terminal icon.

Tip: As you move your cursor close to the terminal icon, the cursor will automatically configure itself as a wiring tool. Click directly on the terminal and drag the wiring tool to the other terminal to make the full wire connection.

Tip: The Graph terminal icon changes automatically to match the wire's data type. Wires and data types are explained in more detail in Chapter 3.

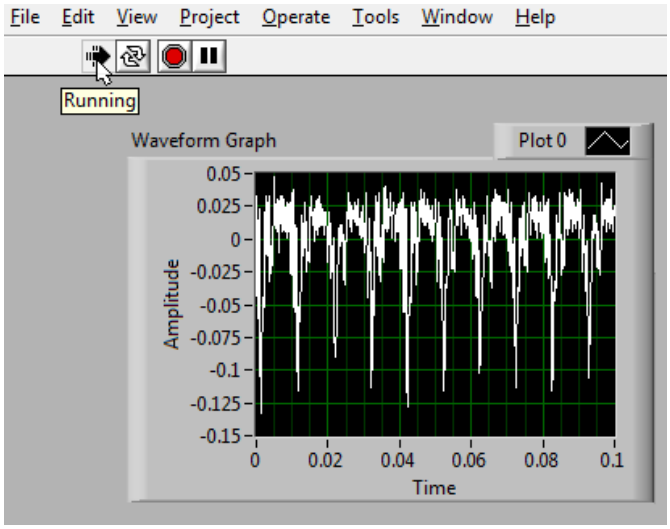


18. The block diagram is now complete with an Express VI to read the sensor and output the value through the CH 1 Auto-ID terminal. This sensor value is wired to a front panel Graph terminal, providing the user with a front panel display of the sensor reading.

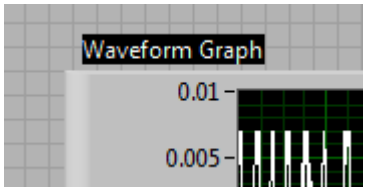


19. View the front panel by choosing Show Front Panel from the Window menu (or use the <Ctrl-E> shortcut).

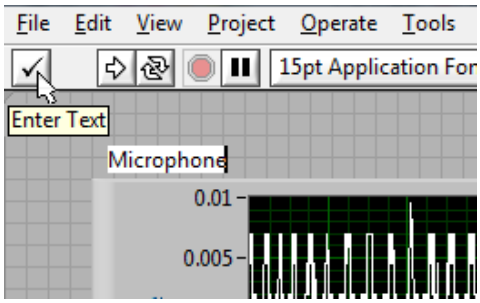
20. Hum into the microphone as you click the white Run arrow on the left side of the Toolbar to run the program.



21. Rename the graph by locating the cursor directly over the graph's label called Waveform Graph, and double-clicking the label to highlight both words.



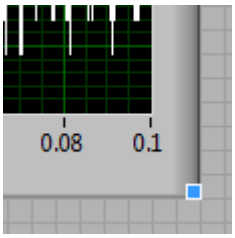
22. Enter “Microphone” as the new label for the graph, and click the Enter Text button.



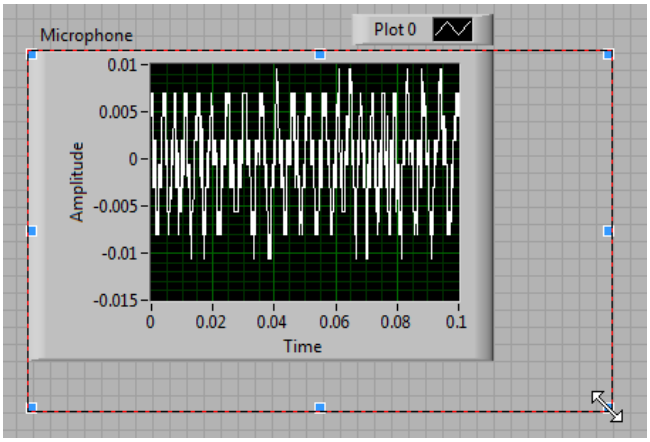
23. View the block diagram using <Ctrl-E> and note the graph terminal icon label has been modified with the new name.

24. View the front panel using <Ctrl-E>.

25. Place your cursor on the bottom-right corner of the graph window and note the blue squares in the corners and midpoint.

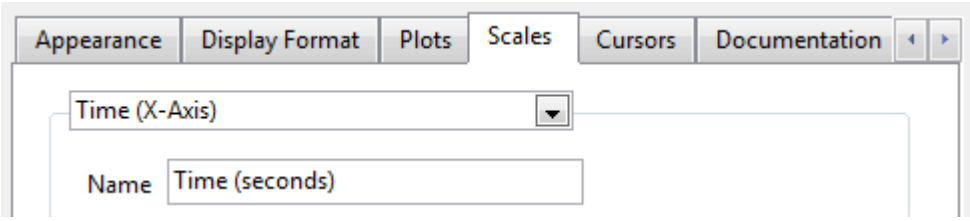


26. Click and drag the bottom-right corner of the graph (on the blue square) to resize the graph window.

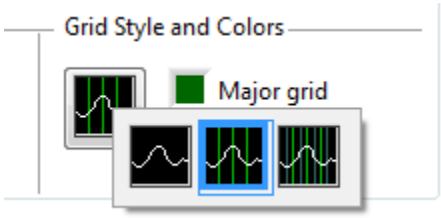


27. Right-click the graph and select Properties.

28. Click the Scales tab and change the Name of the x-axis scale to Time (seconds).

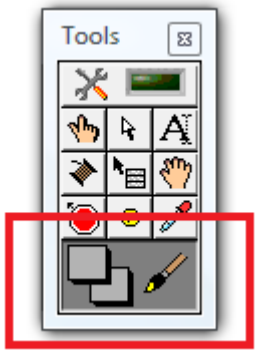


29. On the same tab, click the Grid Style icon and select the center icon to display the Major grid (the minor grid is made translucent).



30. Select OK to close the Properties dialog box.

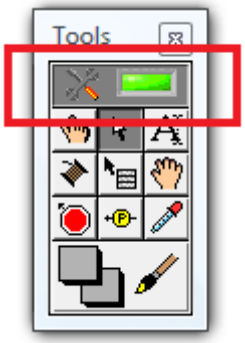
31. Select the Coloring tool from the Tools palette.



Tip: If the Tools Palette is not visible, choose Tools Palette from the View menu.

32. Locate the cursor (your cursor should look like a paint brush) on the border of the graph and right-click to view the color palette. Select a new color from the color palette.

33. Selecting the Coloring tool disables the automatic tool selection; therefore, you must click the Automatic Tool Selection tool button to enable this feature again.



EXTENSIONS

1. Run the VI using the Run Continuously toolbar button.

2. The y-scale is set to Autoscale the plot. Turn off Autoscale and configure the Y scale with appropriate minimum and maximum scale values.

3. Study the sound of your voice when you make an “o” sound versus an “e” sound.

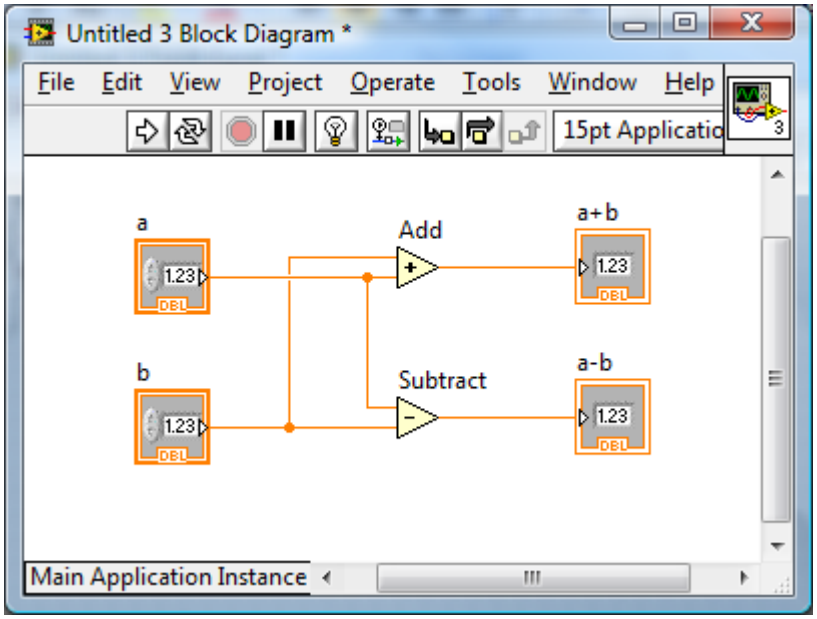
4. The front panel control palette has a subpalette called Decorations. Choose some decorations for your front panel, and use the Color tool to modify the color of the decorations.

Chapter
3

LabVIEW Programming

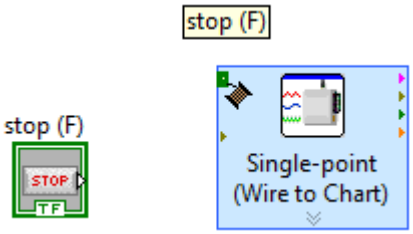
WIRES

Think of wires as a path for data to flow. Data comes into block diagram objects through a wire and can leave only through a wire. In the figure below, wires connect the control and indicator terminals to the Add and Subtract functions. As you can see, each wire has a single data source or starting point. However, you can branch off one wire, represented by a dot on the wire, and wire it to many VIs and functions. By branching off the main wire, you can send data to multiple destinations. Note that the color, style, and thickness of wires changes depending on the type of data the wire is transmitting.



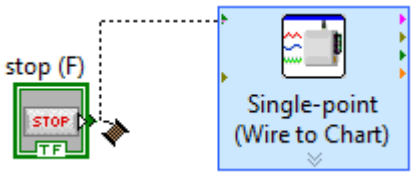
Logic rules apply to wiring in LabVIEW: each wire must have one (but only one) source (or control), and each wire may have multiple destinations (or indicators). For example, you cannot wire two indicators together, but you may wire one control to two indicators.

When you pass the Wiring tool over a terminal, the terminal blinks and a tip strip appears with the name of the terminal. Use this feature to ensure that you are selecting the correct terminal before wiring.



To wire objects together, pass the Wiring tool over the first terminal, click on the terminal, and then drag the cursor to the second terminal. If the Tools palette is on auto-select, the wiring tool automatically appears when you hold the mouse over a terminal. Click again on the destination terminal to terminate the wire. You may also click the mouse before you get to the second

terminal. This will pin the wire to the block diagram at the location at which you click. This allows you to make a turn in the wire path.



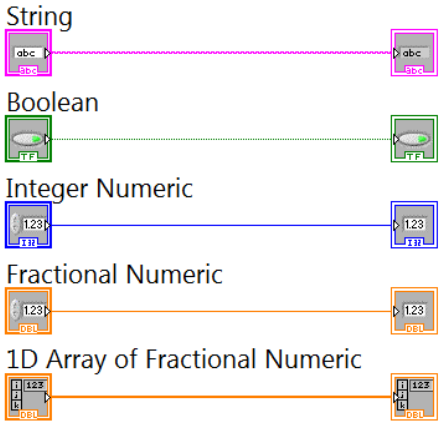
After wiring, you may want to clean up the path of the wire. Move your cursor to the wire. When the cursor turns into an arrow, click on the wire and drag the wire to relocate it. In addition, you can right-click the wire and select Clean Up Wire from the shortcut menu. LabVIEW automatically chooses a path for the wire. And finally, the wires and objects on a block diagram can be organized using the Clean Up Diagram button on the toolbar.

DATA TYPES

When you create an object on the front panel, a terminal will be created on the block diagram. These terminals give you access to the front panel objects from the block diagram code. Each terminal icon contains useful information about the front panel object it corresponds to. For example, the color and symbols provide information about the data type. The dynamic data type, used by many Express VIs, is a data type represented by dark blue terminals; Boolean terminals are green with TF lettering; and Strings are pink.

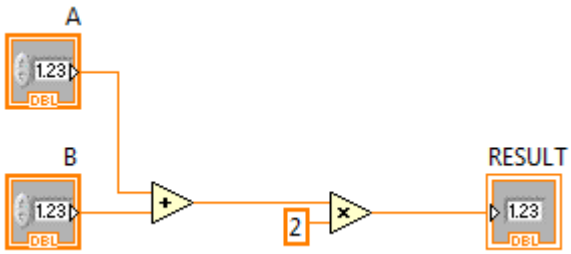
Every wire also has a type based on the data that it is transmitting. The data type of a wire determines which object, indicators, or functions you can connect a wire to. For example, if a Boolean switch has a green border, you can wire that switch to any input terminal with a green label. Note that the wire will also be green, reflecting the Boolean data type. If a knob has an orange border, you can wire a knob to any input terminal with an orange label and the wire will be orange. You cannot wire an orange knob to an input terminal with a green label because the data types are not compatible. In most cases, look for a match in color, but this is not a hard-and-fast rule; LabVIEW will allow a user to connect an Express VI's dark blue terminal to an orange terminal that represents a real number (fractional numeric value), for example.

The figure below shows the correct wiring of common data types. Note that when working in LabVIEW, the color of the wire and control matches the color of the input terminal.

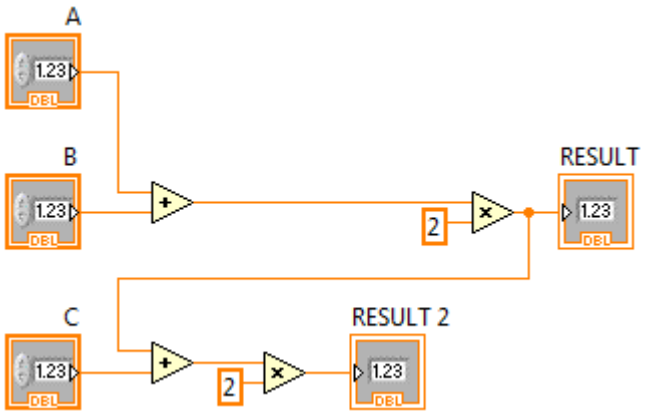


DATAFLOW PROGRAMMING

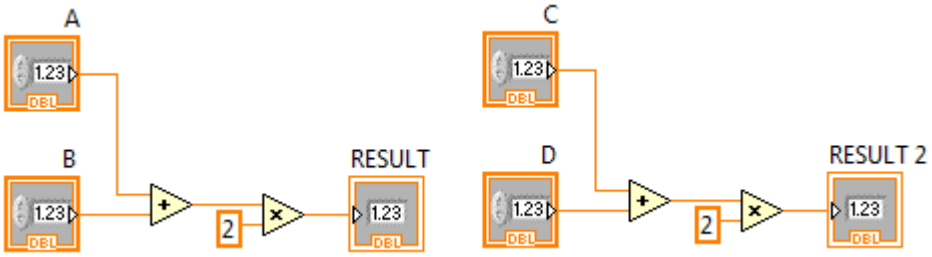
LabVIEW follows a dataflow model for running VIs. A block diagram object executes when all its inputs are available. When an object completes execution, it supplies data to its output terminals and passes the output data to the next object in the dataflow path.



Consider the block diagram in the figure above. In this program, the block diagram executes from left to right, not because the objects are placed in that order, but because the upper input of the Multiply function is not valid until the Add function has finished executing and passed the data to the Multiply function. Remember that an object executes only when data are available at all of its input terminals, and it supplies data to its output terminals only when it finishes execution.



In the figure above, the flow of the code dictates that the addition of C with RESULT does not occur until A and B have been added together and this sum is multiplied by 2. This means that RESULT will happen before RESULT 2. The length of the wires does not slow or speed the code.



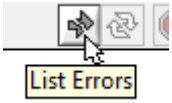
And in this final example, the code that generates RESULT 2 is to the right of the code that generates RESULT, but the position of code does not matter. What matters is the flow of the code and whether an object has received all of the required data input. Both will begin executing at the same time and run independent of one another. The two results will happen at

approximately the same time.

DEBUGGING TECHNIQUES

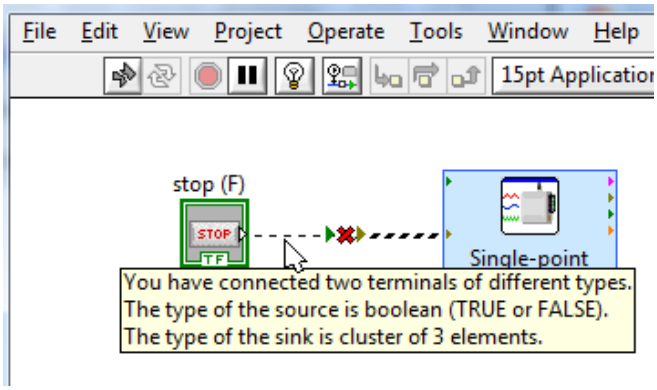
LabVIEW has several useful tools for debugging a program, a few of which are discussed below.

When your VI is not executable, a broken arrow is displayed in the Run button in the palette. To list errors, click the broken arrow. An Error List dialog box will appear. To locate the bad object, double-click the error message that is displayed in the dialog box.



Broken wires (wires that are not connected properly) prevent your VI from running. A broken wire appears as a dashed black line with a red X in the middle, as shown below.

Broken wires occur for a variety of reasons, such as wiring two objects with different or incompatible data types; for example, you cannot wire a Boolean control to a numeric input terminal. The data in the wire is a Boolean and the input is expecting a numeric value, so the data types are not compatible.



You must manually clean up a broken wire by selecting it and deleting it. A quick way to get rid of all broken wires is to use the shortcut <Ctrl-B> or choose Remove Broken Wires from the Edit menu.

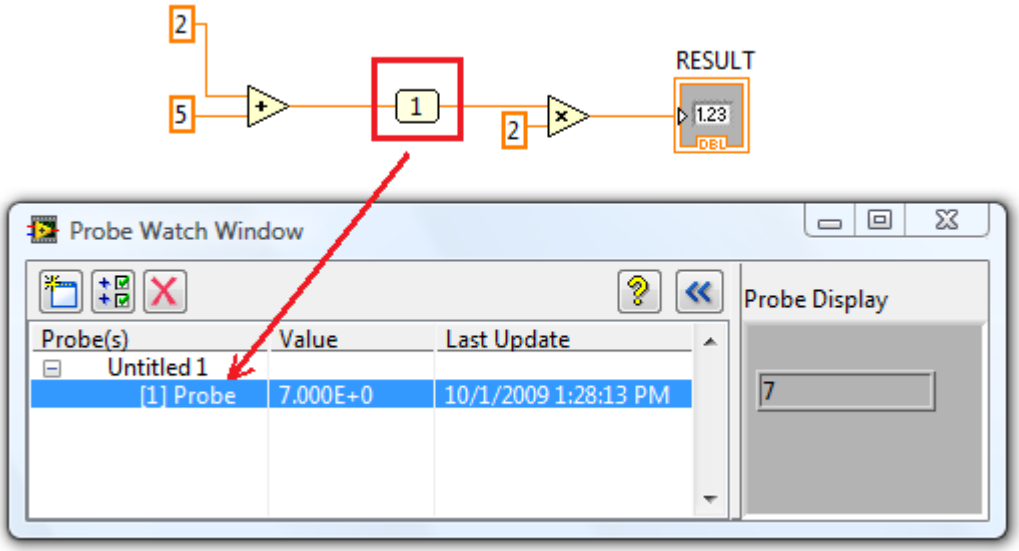
When trying to visualize the data flow of a particular VI, a very useful tool is Highlight Execution. Implement execution highlighting by clicking the Highlight Execution button on the toolbar.

When Highlight Execution is selected, the icon will change to represent a bulb turned on.

Now, when you run the VI, observe the data flow of the program's block diagram. Not only does execution highlighting show the sequence of execution, it also slows down the speed of execution so that it is visible to the user. Beware that this execution speed impacts the overall performance of the VI, and should be turned off to return execution to normal speeds.

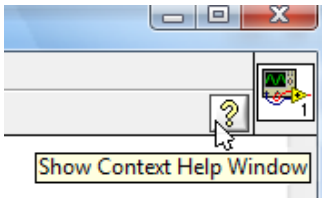
Execution highlighting allows you to view the data as it flows through the wires, but sometimes it is not easy to view just a single wire's data with execution highlighting. In addition, it may be necessary to view a wire's data as a program runs at full speed. To view a wire's data as a program runs at full speed, click the wire with the Probe Data tool that is found in the Tools Palette (highlighted in the figure below), or by selecting Probe from the shortcut menu that appears by right-clicking the wire.

After placing a probe on a wire, the Probe Watch Window dialog box appears. This window will display the value of the wire as the program runs. Note in the figure below that a probe was placed on the wire coming out of the Addition function, and it created a yellow box with a number 1. This corresponds with the probe number in the dialog box.

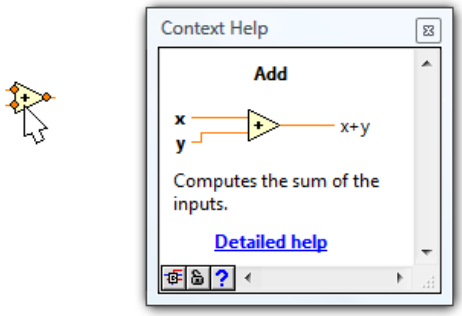


CONTEXT HELP

The Context Help window displays basic information about LabVIEW objects when you hover the mouse over each object. Open the Context Help window by choosing Show Context Help from the Help menu, pressing the <Ctrl-H> keys on the keyboard, or by clicking on the Show Context Help Window button on the toolbar, as seen in the figure below.



Hover your mouse over the different wires, controls, and indicators to see their data types in the Context Help window. Hover your mouse over the functions to see a brief explanation of their functions and the inputs and outputs they accept. Click on the Detailed Help link at the bottom of the Context Help window to get a more detailed description of the function and its inputs and outputs.



TIPS FOR WORKING IN LABVIEW

LabVIEW has many keystroke shortcuts that make working easier. The most common shortcuts are listed below.

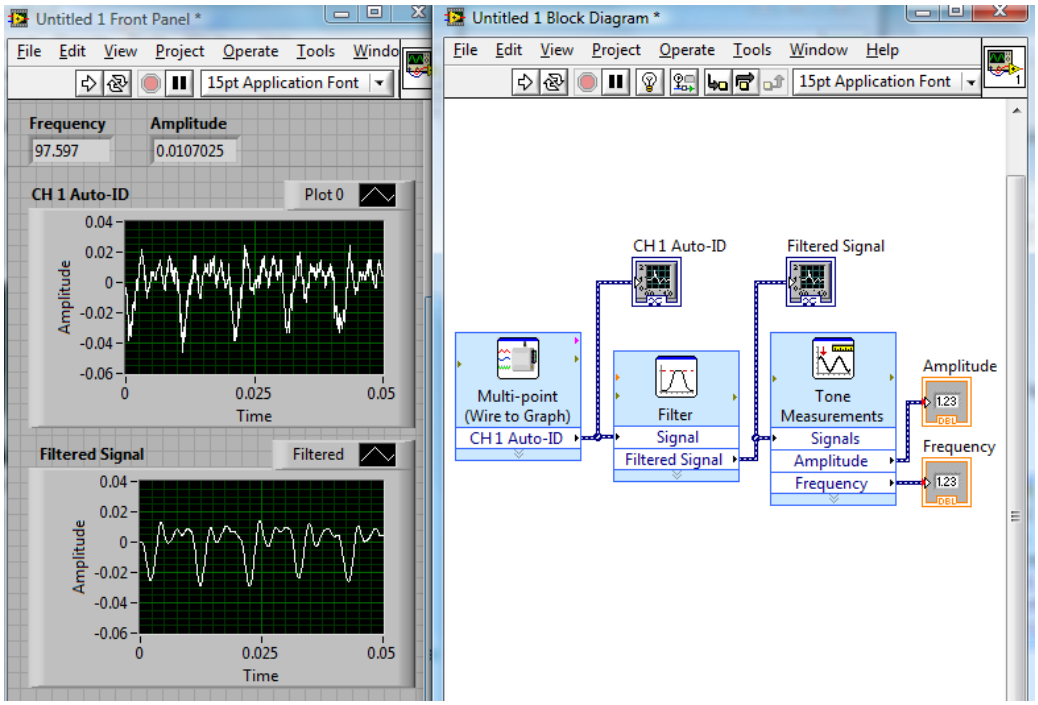
<Ctrl-E>	Toggle between the front panel and block diagram
<Ctrl-T>	Tiles the front panel and block diagram to easily see both windows side by side
<Ctrl-B>	Remove broken wires from the block diagram
<Ctrl-H>	Display and close the context help window
<Ctrl-Z>	Undo

While the Automatic Selection Tool is great for choosing the tool you would like to use in LabVIEW, there are sometimes cases when you want manual control. Once the Automatic Selection Tool is turned off, manually select the appropriate tool. Once you are finished with the tool you choose, you can click the Automatic Tool Selection button on the Tools palette, or press <Shift+Tab> to turn the Automatic Selection Tool back on.

In the Options dialog from the Tools menu, there are many configurable options for customizing your front panel, block diagram, colors, printing, and much more.

Exercise
3

Analyze Microphone Data



Completed front panel and block diagram

In the following steps, you will create a program using the Analog Express VI to collect microphone data for a length of 0.05 second at a rate of 10,000 samples/second. The raw microphone data will be displayed on a graph. The program will then apply a low-pass filter to the raw data to reduce the amplitude of the high frequency noise, thus smoothing out the data set. The filtered data are then analyzed to determine the frequency and amplitude of the tone. The filtering and analysis will be programmed using LabVIEW Signal Analysis Express VIs.

OBJECTIVES

In this exercise, you will

- Create a LabVIEW VI.
- Incorporate LabVIEW Express Analysis functions.
- Use the Execution Highlight feature to study data flow.
- Display data using numeric and graphical indicators.

MATERIALS

SensorDAQ or LabQuest interface
USB cable
computer
LabVIEW
Vernier Microphone

PROCEDURE

Part I Connect Equipment

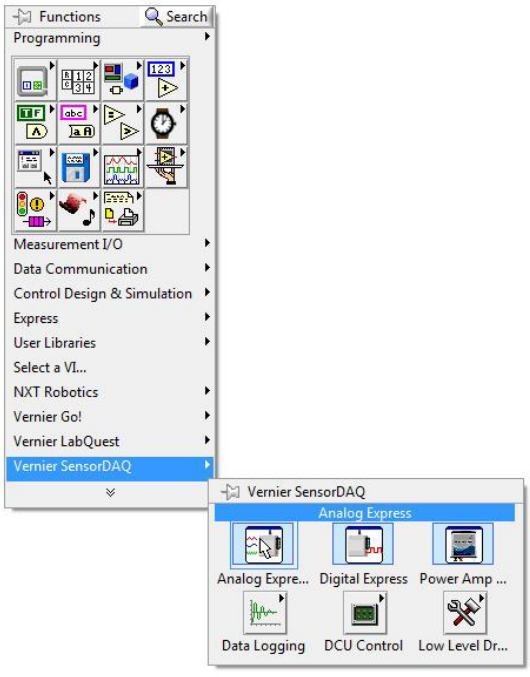
1. Connect the USB cable to the SensorDAQ or LabQuest interface.
2. Connect the other end of the USB cable to any available USB port on your computer. If you are using a LabQuest interface with a power button, turn it on.
3. Connect the Microphone to Ch. 1.

Part II Start LabVIEW and Create a VI to Collect Data

4. Start LabVIEW.
5. In the Getting Started window, click the Blank VI link in the New category.
6. View the block diagram by choosing Show Block Diagram from the Window menu (or use the <Ctrl-E> shortcut).
7. Place an Analog Express VI in the block diagram workspace.

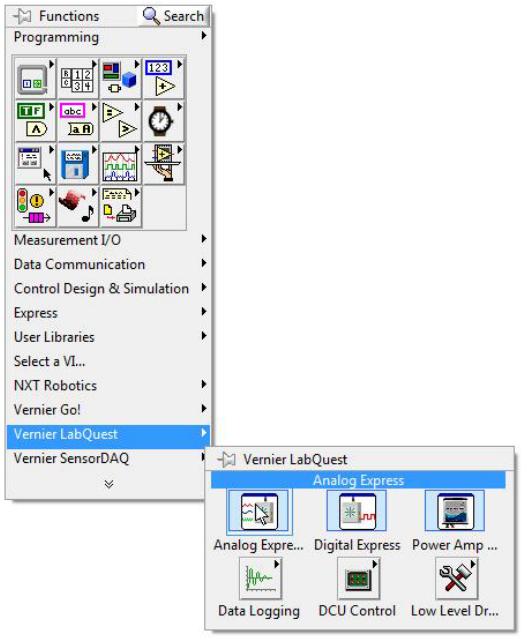
SensorDAQ

If you are using a SensorDAQ, right-click in the block diagram workspace and select Vernier SensorDAQ from the Functions palette. Click and drag the Analog Express VI to the block diagram workspace.



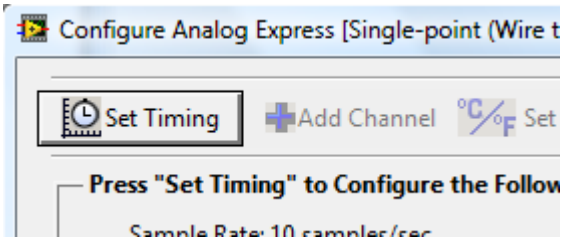
LabQuest Interface

If you are using a LabQuest interface, right-click in the block diagram workspace and select Vernier LabQuest from the Functions palette. Click and drag the Analog Express VI to the block diagram.



8. After dragging the Express VI from the palette to the block diagram workspace, the Express VI's configuration popup will open. Note that this step can be slow, depending on your computer.

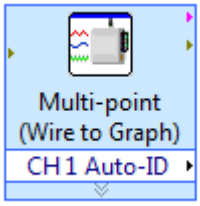
9. Click the Set Timing button, located in the upper-left corner of the configuration dialog.



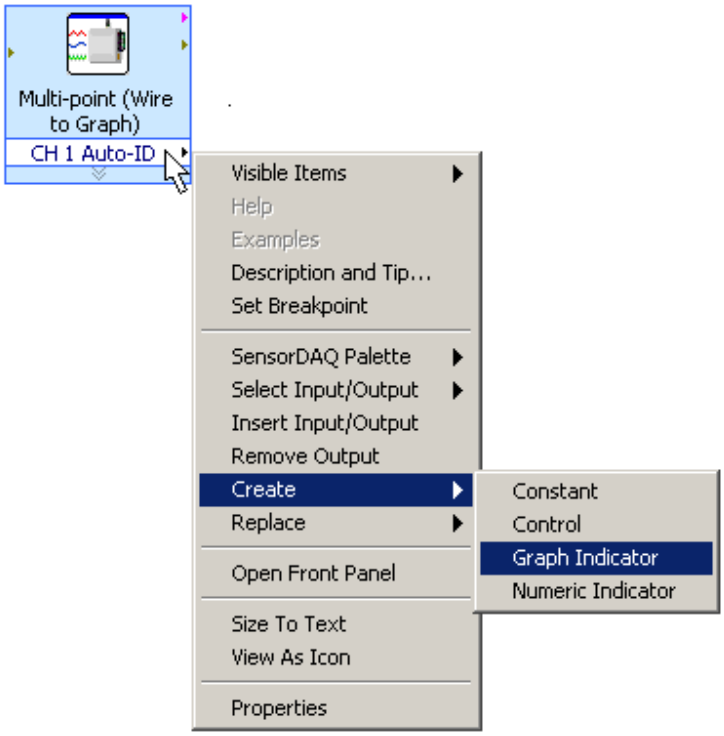
10. Set up the timing with a length of 0.05 second and a sample rate of 10,000 samples/second.

11. Click Done to close the Set Timing window. The Express VI Configuration should now be updated with the new settings.

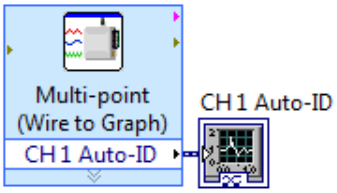
12. Select OK to close the Express VI's Configuration window. The Express VI will now be located in your block diagram workspace.



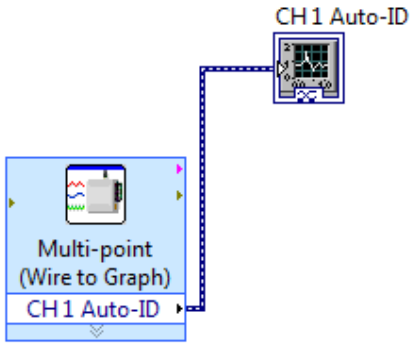
13. Create a graph for the front panel. This can be done by going to the front panel and selecting a graph from the Controls Palette. It can also be created in the block diagram workspace. To create it from the block diagram, right-click on the Express VI's "CH 1 Auto-ID" output terminal, and select Create ► Graph Indicator from the shortcut menu.



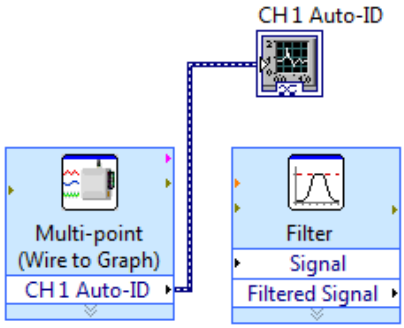
14. The graph is created, labeled and wired.



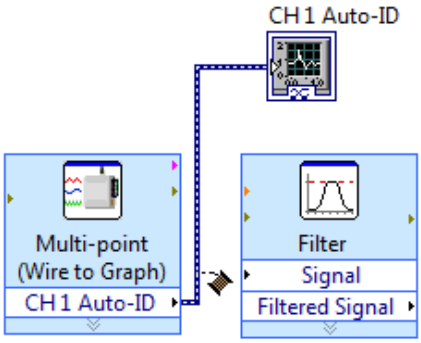
15. Move the graph terminal (click and drag to move an object) to make room to branch off of the wire.



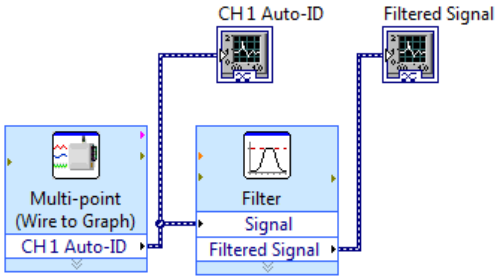
16. Place the Filter Express VI to the right of the Analog Express VI on the block diagram. From the Functions palette, choose Express ► Signal Analysis ► Filter and place it on the block diagram. When the configuration dialog opens, select "LowPass" as the Filtering Type, and 300 Hz for the Cutoff Frequency value. Click OK.



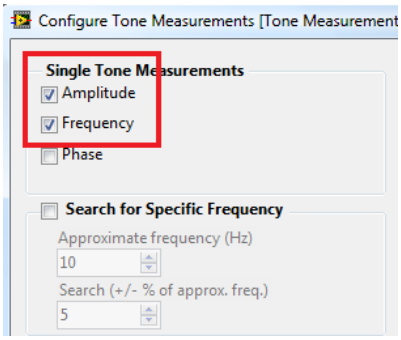
17. Bring the cursor to the wire. When the cursor changes to the Connect Wire tool, click and drag to create a wire branch from the wire to the Signal input terminal.



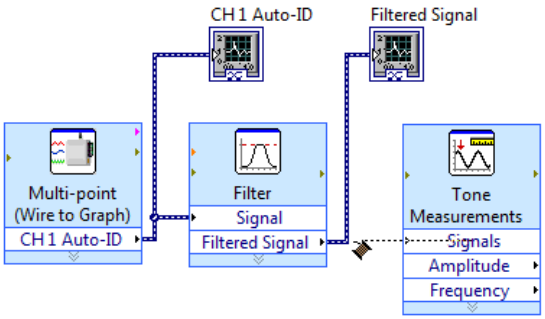
18. Create a graph indicator for the Filtered Signal output by right-clicking the Filtered Signal terminal and selecting Create ► Graph Indicator from the shortcut menu.



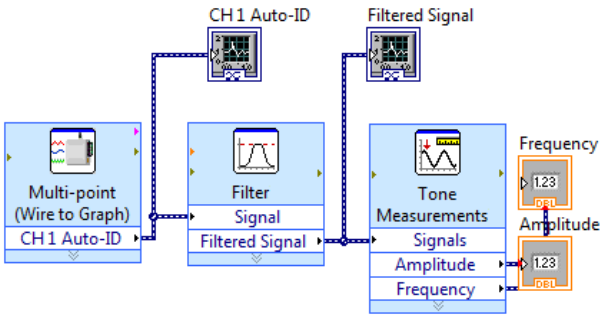
19. Place a Tone Measurements Express VI on the block diagram (Express ► Signal Analysis ► Tone Measurements). In the configuration dialog, choose Amplitude and Frequency measurements in the Single Tone Measurements section. Click OK.



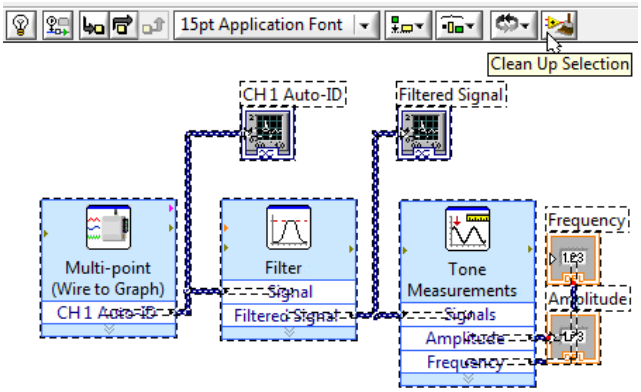
20. Wire the Filtered Signal to the Signals input terminal.



21. Create indicators for the amplitude and frequency measurements by right-clicking on each of the terminals of the Tone Measurements Express VI and selecting Create ► Numeric Indicator.

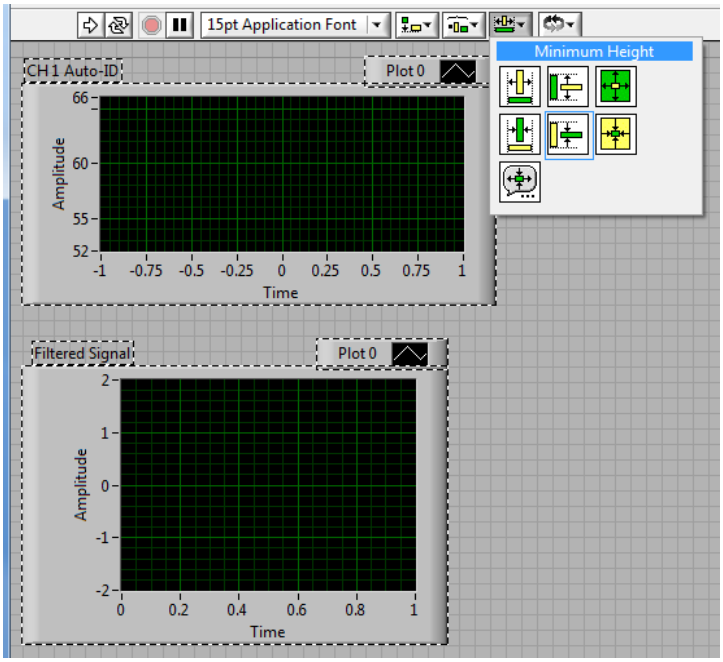


22. The block diagram objects and wires should be cleaned up. Select all, or a portion, of the block diagram and click the Clean Up Diagram toolbar button. Otherwise, manually move objects and wires.

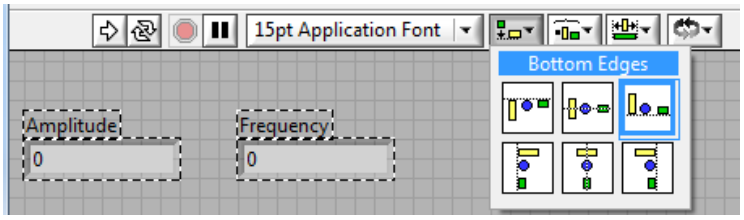


23. View the front panel using the shortcut <Ctrl-E>.

24. Resize one graph to increase the width and decrease the height. Then select both graphs and use the Resize Object toolbar button to make the other graph match your changes to the first graph (select Maximum Width and Minimum Height).

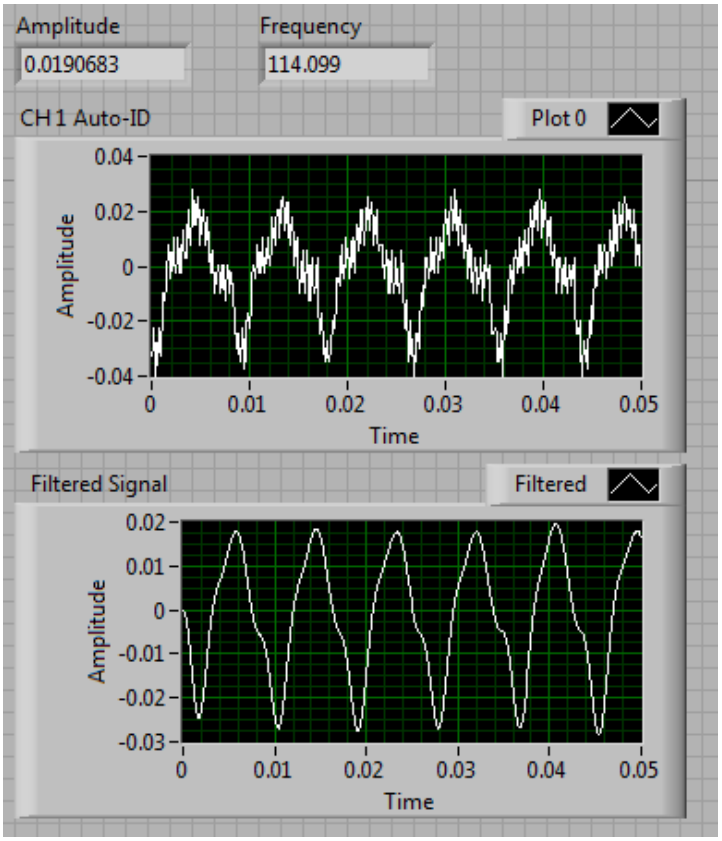


25. Move the two numeric indicators above the top graph and increase their width the same amount. Make sure they are lined up by using the Align Objects toolbar button to align their bottom edges.



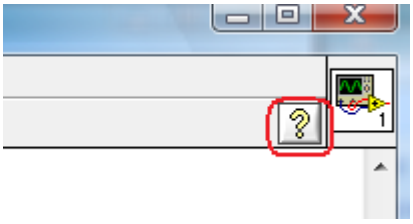
26. Begin humming into the microphone.

27. Click the white Run arrow on the left side of the Toolbar to run the program.



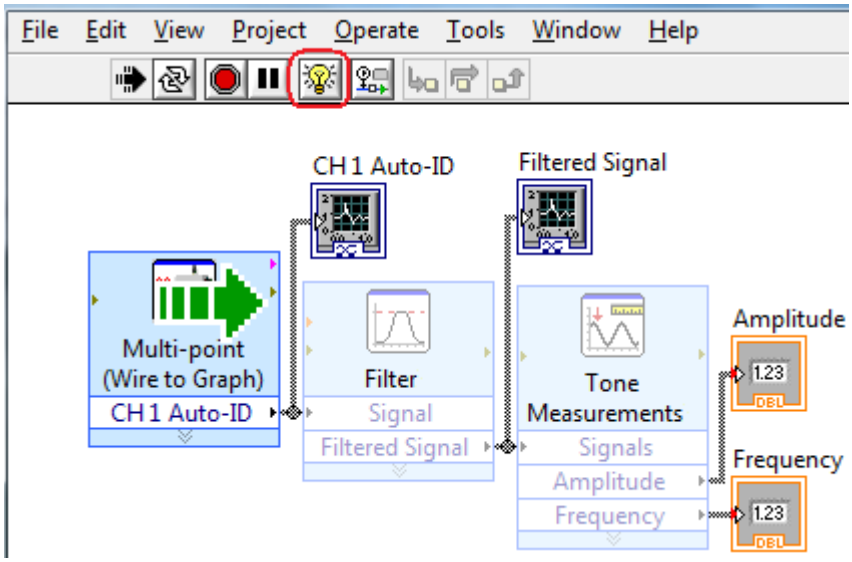
28. View the block diagram using the shortcut <Ctrl-E>.

29. Activate the Context Help Window (if it is not already) by clicking the Context Help Window button in the Toolbar.



30. Hover the cursor over the 3 Express VIs to view the Context Help Window information. Close the Context Help Window using the shortcut <Ctrl-H>.

31. Click the Highlight Execution debugging tool and then run the VI from the block diagram workspace by clicking on the white Run arrow. The order of execution, data flow, and data values should all be visible as the program runs.



EXTENSIONS

1. Turn highlight execution off. Display the value of a wire as the program runs at full speed using the Probe tool.

2. Run the VI continuously and experiment with sounds, such as vowel sounds or playing a musical instrument.

3. Modify the format of the front panel numeric indicators to provide one digit of precision for the Frequency and three digits of precision for the Amplitude.

4. Modify the filter configuration to observe how it affects the signal.

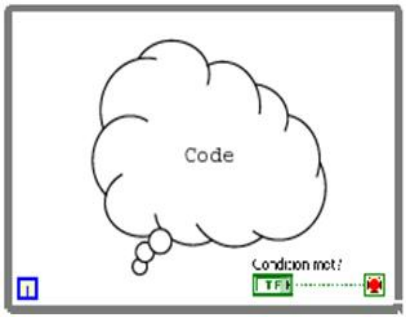
Chapter
4

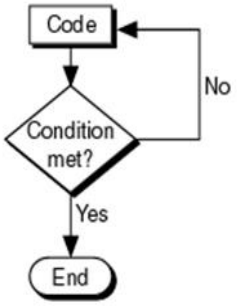
Working with Loops

LOOPS

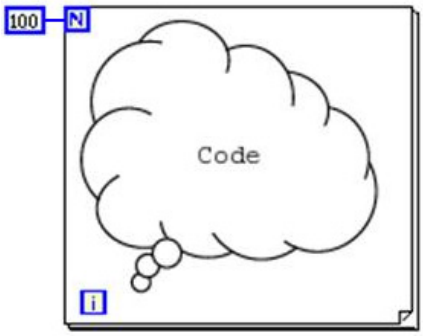
Two common structures used in LabVIEW programming are the While Loop and the For Loop. Both While and For Loops are located on the Functions ► Structures palette.

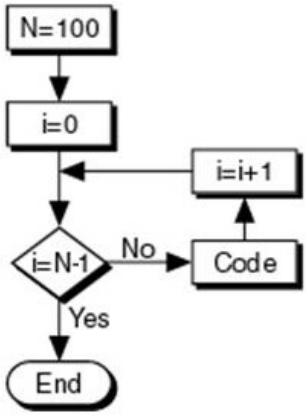
A While Loop is a control flow statement used to execute a block of LabVIEW code repeatedly until a given Boolean condition is met. First, you execute the code, and then the conditional terminal is evaluated. A While Loop does not have a set iteration count; thus, a While Loop executes indefinitely if the condition never occurs.





A For Loop is a control flow statement used to execute code a set number of times. The value in the count terminal (an input terminal represented by the N), indicates how many times to repeat the code.

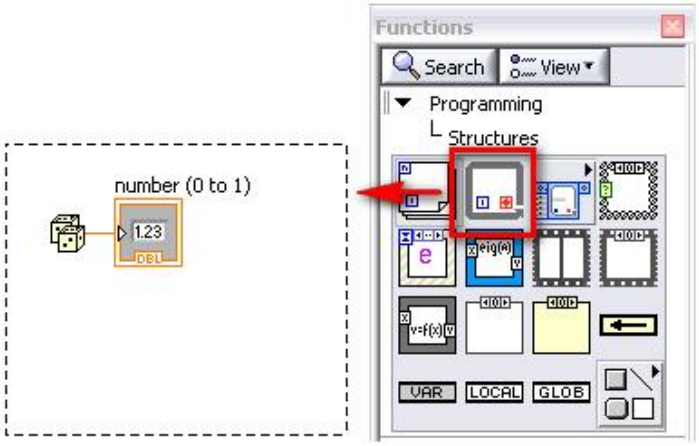




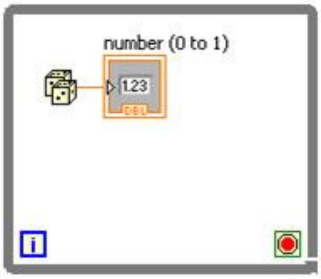
Both the While Loop and For Loop have an iteration terminal, i. The iteration terminal is an output terminal that contains the number of completed iterations. The iteration count always starts at zero. Therefore, the first time the code executes, the iteration terminal returns 0; the second time, the code executes the iteration terminal returns 1. If you repeat the code 100 times, the iteration terminal would end with a value of 99.

To place a loop on the block diagram, you will first select the loop from the Structures palette, and then use the cursor to drag a selection rectangle around (lasso) the section of the block diagram you want to repeat. For example, select the While Loop structure, and use the cursor to

drag a selection rectangle around code that generates a random number. The Random Number (0–1) function is found in the Numeric palette.

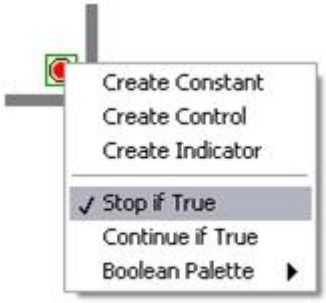


When you release the mouse button, a While Loop boundary encloses the section you have selected. The loop can be resized as necessary and additional code can be dropped inside the loop.

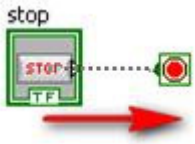


The diagram above is not complete because the While Loop's conditional terminal must be wired. The conditional terminal defines when the loop stops. There are two settings for the conditional terminal: Stop if True, , and Continue if True, . When set to Stop if True, the While Loop runs until a Boolean value of true is sent to the conditional terminal. If the conditional terminal is configured to Continue if True, the While Loop runs only if a Boolean value of true is sent to the terminal.

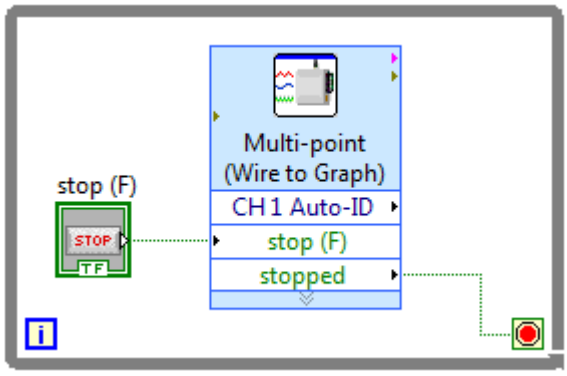
To switch the conditional terminal between Continue if True and Stop if True, right-click on the conditional terminal, and check the corresponding setting. In most cases, it is best to leave the conditional terminal in the default configuration of Stop if True. This keeps the While Loop logic consistent.



Wire a Boolean control (such as a STOP button) to the conditional terminal so that you can control the execution of the While Loop. An easy way to add a STOP button to a While Loop is to right-click on the conditional terminal and select Create ► Control from the shortcut menu. When the STOP button is pressed, a true value is passed to the conditional terminal causing the While Loop to stop execution. You can wire any Boolean data to the conditional terminal to control the execution of a While Loop.

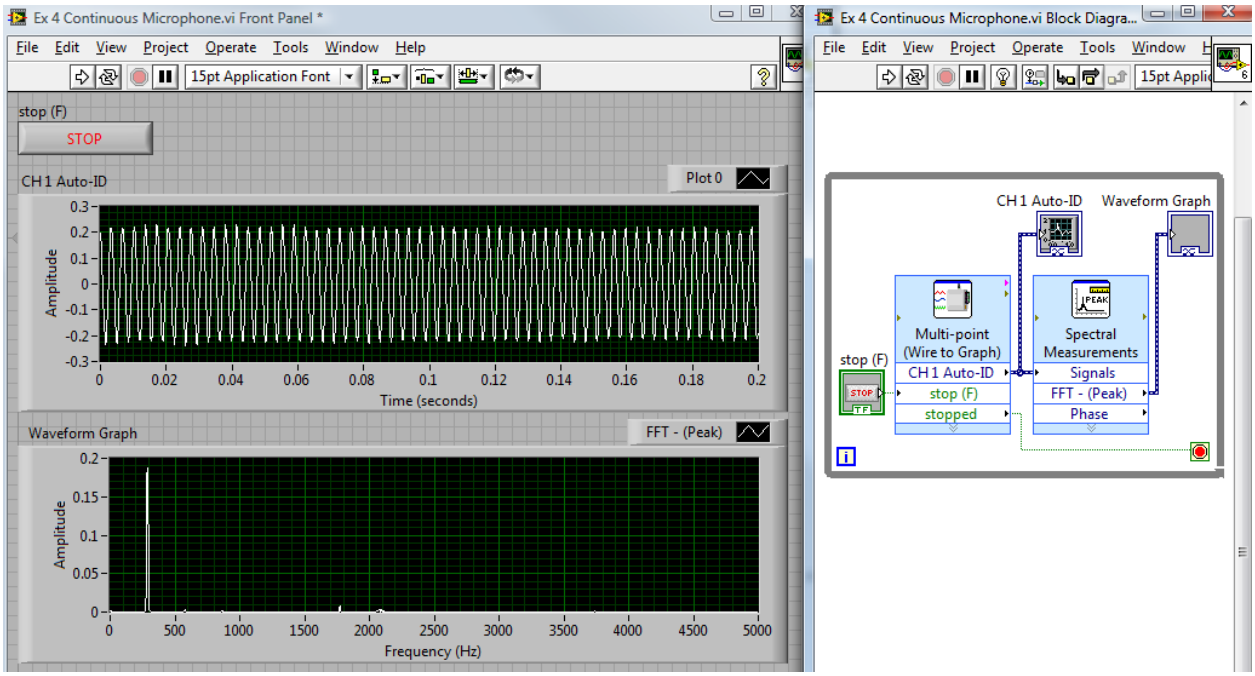


When working with hardware, it may be critical to stop the hardware prior to stopping the execution of the While Loop. For example, if there is code within the loop to continually collect data, and the loop is terminated without properly stopping the data collection process first, the hardware will not be properly shut down. The proper method would be to stop the hardware, and then terminate the execution of the While Loop as displayed in the example code below.



Exercise
4

Continuously Read and Analyze Microphone Data



Completed front panel and block diagram

In this exercise, you will create a simple program using the Analog Express VI and the Spectral Measurements Express VI to continuously collect microphone data and continuously perform an FFT analysis on the data. A Fast Fourier Transform (FFT) is a mathematical algorithm commonly used in digital signal processing to determine the dominant frequencies in a complex waveform.

OBJECTIVES

In this experiment, you will

- Create a LabVIEW VI to continuously collect and analyze data.
- Incorporate LabVIEW Express Analysis functions.
- Use a While Loop to keep collecting and analyzing data until stopped by the user.

MATERIALS

SensorDAQ or LabQuest interface
USB cable
computer
LabVIEW
Vernier Microphone
tuning fork

PROCEDURE

Part I Connect Equipment

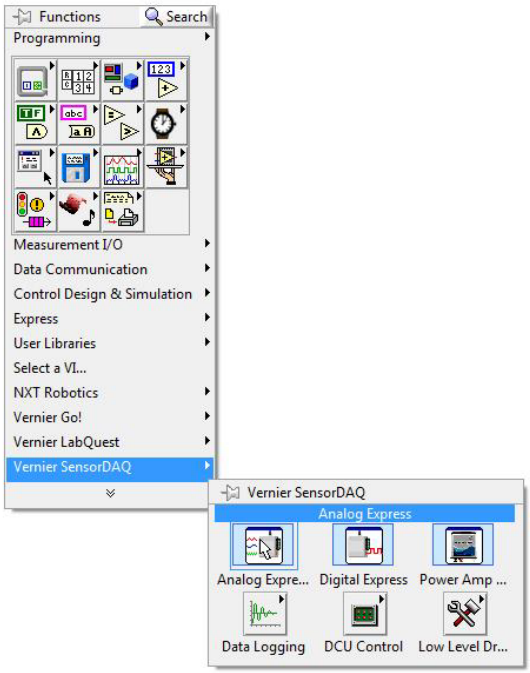
1. Connect the USB cable to the SensorDAQ or LabQuest interface.
2. Connect the other end of the USB cable to any available USB port on your computer. If you are using a LabQuest interface with a power button, turn it on.
3. Connect the Microphone to Ch. 1.

Part II Start LabVIEW and Create a VI to Collect Data

4. Start LabVIEW.
5. In the Getting Started window, click the "Blank VI" link under the New category.
6. View the block diagram by choosing Show Block Diagram from the Window menu (or use the <Ctrl-E> shortcut).
7. Place an Analog Express VI in the block diagram workspace.

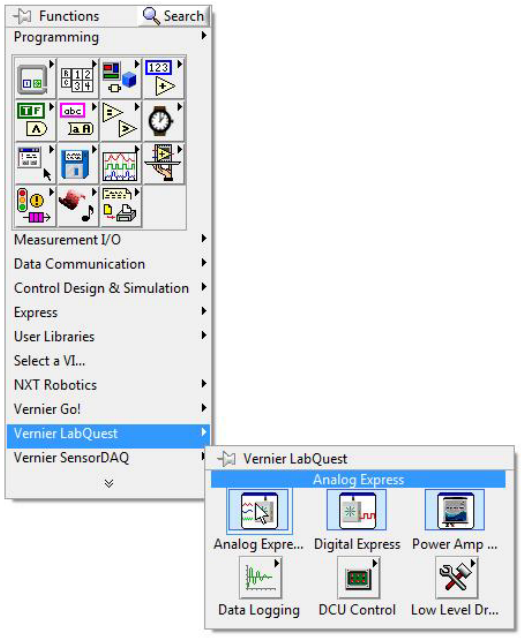
SensorDAQ

If you are using a SensorDAQ, right-click in the block diagram workspace and select Vernier SensorDAQ from the Functions palette. Click and drag the Analog Express VI to the block diagram workspace.



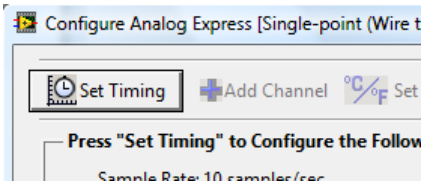
LabQuest Interface

If you are using a LabQuest interface, right-click in the block diagram workspace and select Vernier LabQuest from the Functions palette. Click and drag the Analog Express VI to the block diagram.

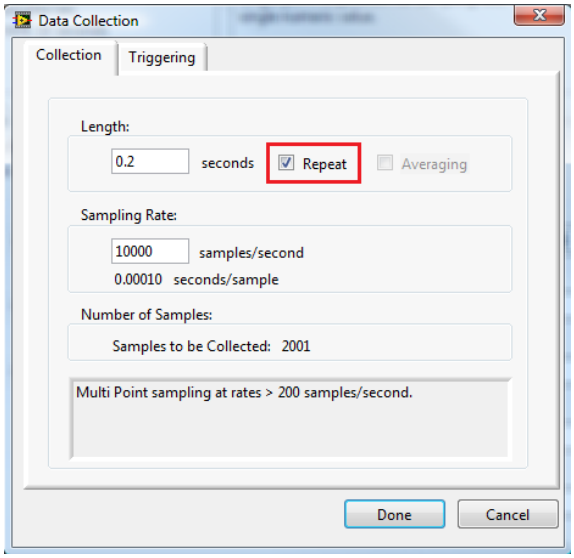


8. After dragging the Express VI from the palette to the block diagram workspace, the Express VI's configuration dialog will open. Note that this step can be slow, depending on your computer.

9. Click the Set Timing button, located in the top-left corner of the configuration dialog.



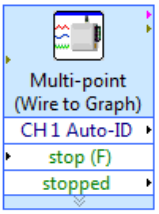
10. First check the Repeat option, and then set the timing with a length of 0.2 second and a sample rate of 10,000 samples/second.



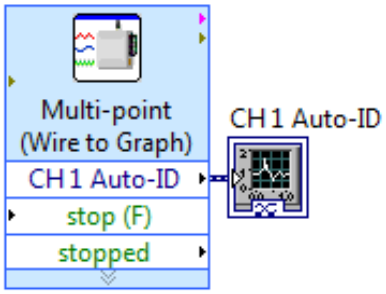
Tip: Repeat means that data collection is repeated. In this case, because the sample rate is greater than 200 samples/second, the Express VI returns data as a multi-point packet containing all of the data points. Checking Repeat configures the interface to stream 2001 samples every 0.2 seconds. This configuration requires that you place the Express VI in a loop to continually read those samples. If Repeat was not checked, the interface would collect the 2001 samples and then stop.

11. Click Done to close the Set Timing window. The Express VI Configuration should now be updated with the new settings.

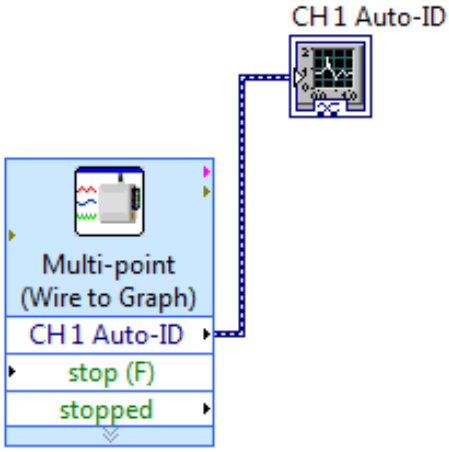
12. Select OK to close the Express VI's Configuration dialog. The Analog Express VI will now be located in your block diagram workspace.



13. Right-click the Express VI's "CH 1 Auto-ID" output terminal and select Create ► Graph Indicator from the shortcut menu.

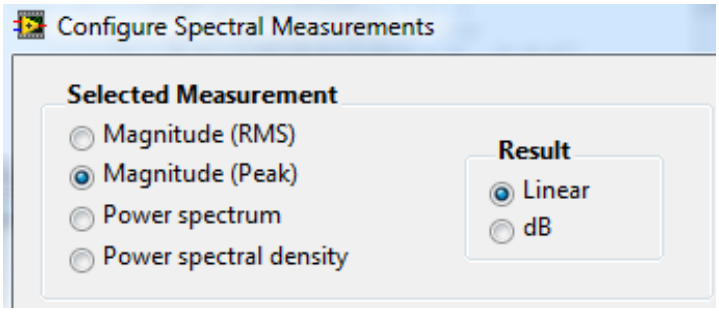


14. Move the graph terminal (click and drag to move an object) to make room to branch off of the wire.

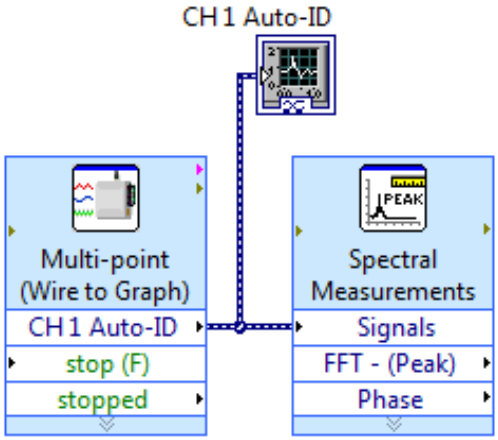


15. Place the Spectral Measurements Express VI to the right of the Analog Express VI on the block diagram. From the Functions palette, select Express ► Signal Analysis ► Spectral Measurements and place it on the block diagram.

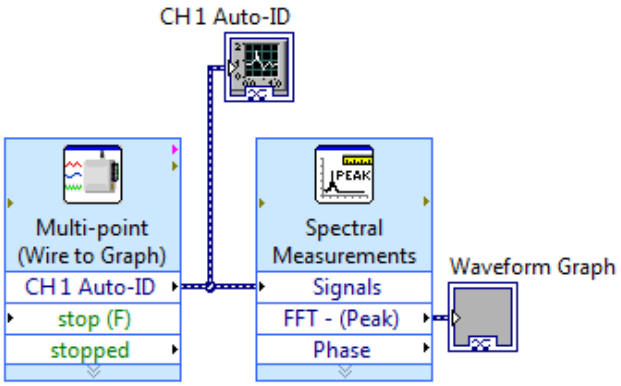
16. In the Configure Spectral Measurements window, select Magnitude (Peak) for the Selected Measurement and Linear for the Result. Leave the rest of the configuration window as the default values. Select OK to close the Express VI.



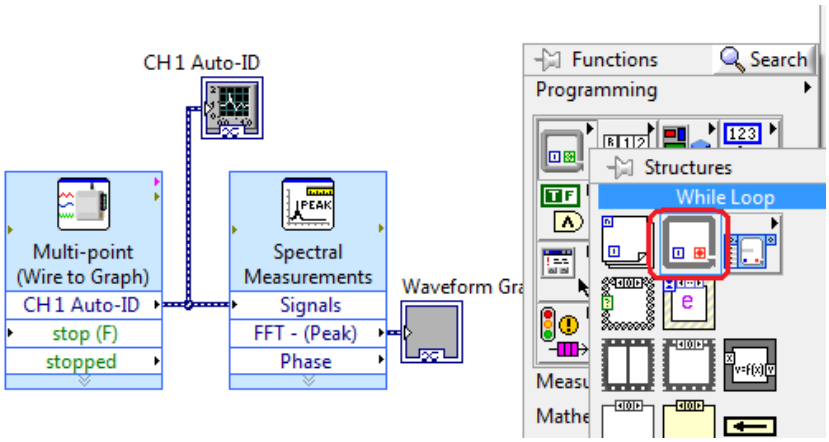
17. Create a wire branch from CH 1 Auto-ID to the Signals input terminal.



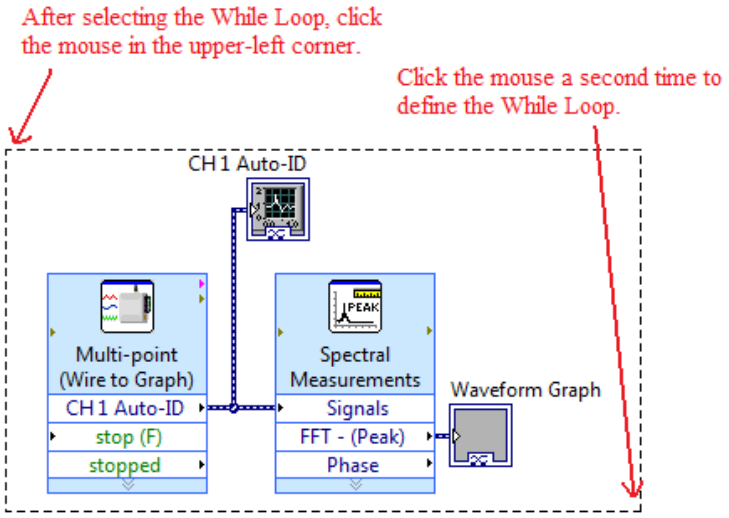
18. Create a Graph Indicator for the FFT output terminal (right-click on the FFT-(Peak) terminal to access the shortcut menu).



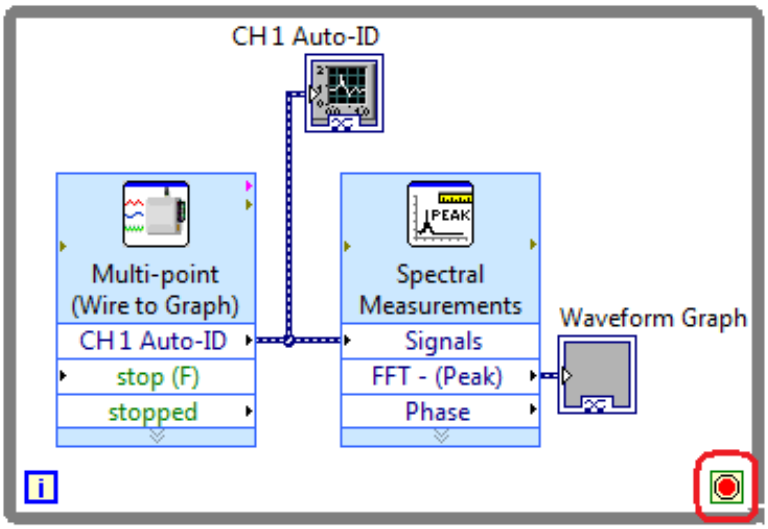
19. Select the While Loop from the Programming ► Structures palette.



20. The cursor will become a special pointer that you use to enclose the code. Click the mouse to define the upper-left corner. Drag the mouse to the location of the bottom-right corner, and click again to create the While Loop.

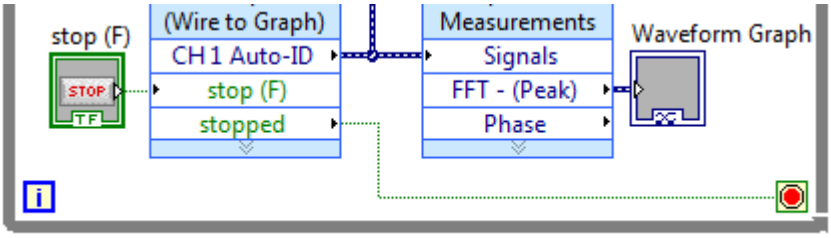


21. The While Loop generates a conditional terminal that is set with the default condition to Stop if True.



22. Right-click the Express VI's "stop (F)" terminal and select Create ► Control. You may have to resize the While Loop to provide room for this control.

23. Wire the Express VI's "stopped" terminal to the While Loop's conditional terminal.

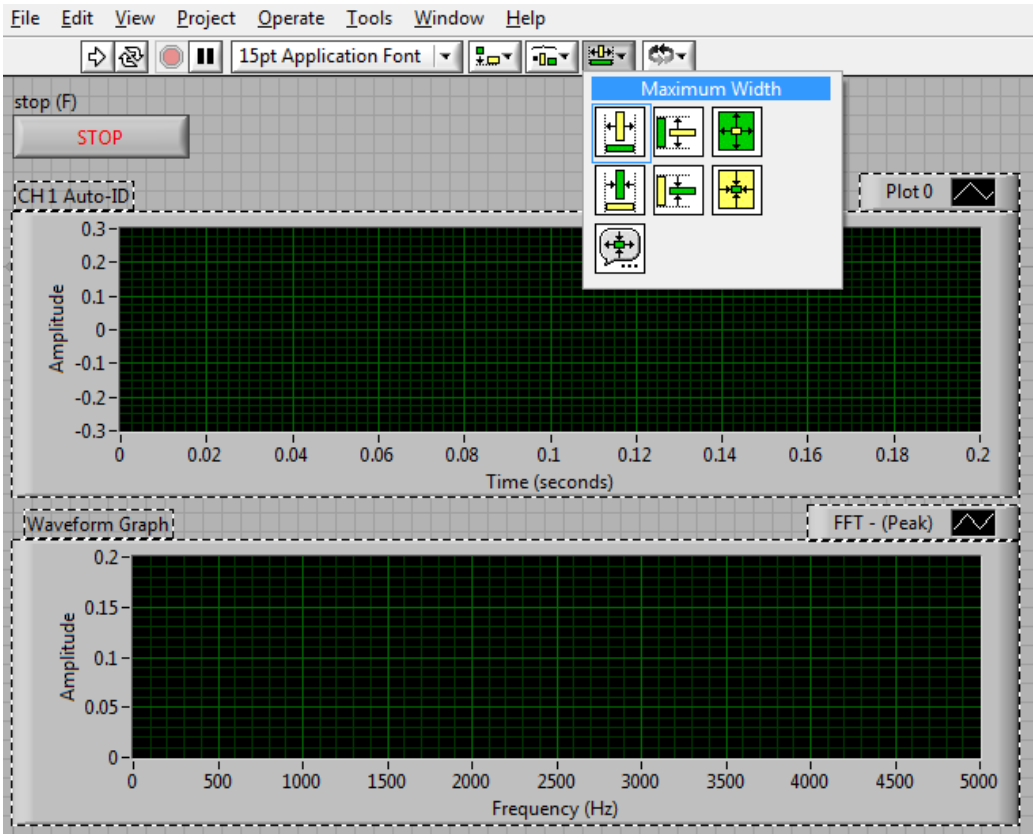


Note: It is important to stop the Express VI and the While Loop in this manner. The data flow of this program insures that the Analog Express VI is stopped before the While Loop is terminated. If the STOP button were wired directly to the While Loop's conditional terminal, the loop would be terminated properly, but the Express VI would not have any input telling it to shut down the interface hardware. This could cause problems with the device the next time it is run; therefore, the order of the program must first stop the Analog Express VI and then stop the While Loop.

24. View the front panel using the <Ctrl-E> shortcut.

25. Rename the x-axis legend of the Waveform Graph to "Frequency (Hz)" and move the graphs and STOP button, if necessary.

26. Resize the two graphs to make them longer. Highlight the two graphs and use the Resize Objects toolbar button to make them the same size.



27. Run the VI. Strike the tuning fork against a soft object (sole of a shoe, for example), and hold it near the Microphone.

28. Click the STOP button to end the program.

EXTENSIONS

1. Analyze your voice while saying the vowels “e” and “o,” and also while singing the vowel sounds.

2. Create an indicator to monitor the number of iterations made by the loop. Recall that the iterations of a loop start at zero, so you must add one to the iteration count.

3. Modify the program to execute the code exactly 25 iterations using a For Loop. To properly stop the program, the Express VI must be stopped prior to ending the program. Do not use a stop button to stop the Express VI; instead use the iteration count and the Equal? Function to stop the Express VI on the very last iteration of the For Loop.

4. In Extension 3, the count input for the For Loop is a constant. Change this constant to a front panel control.

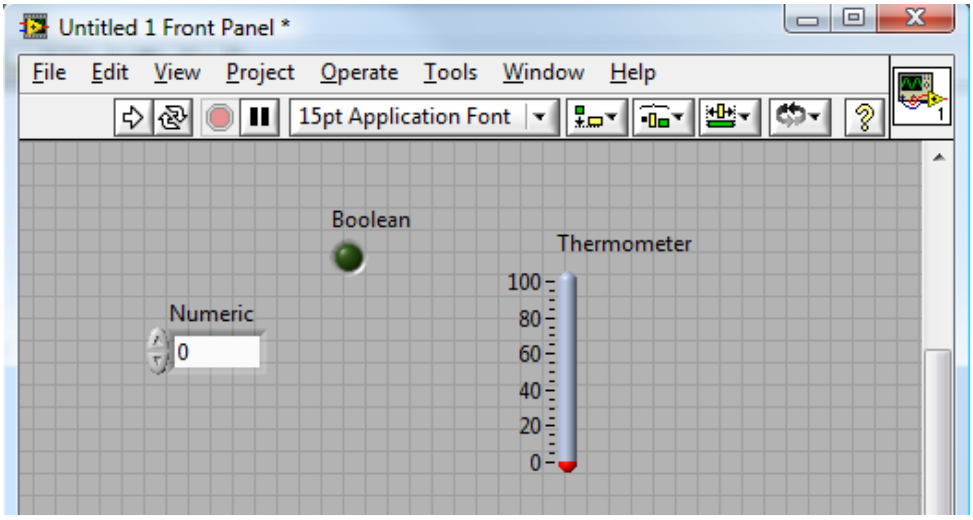
Chapter
5

Presenting Results

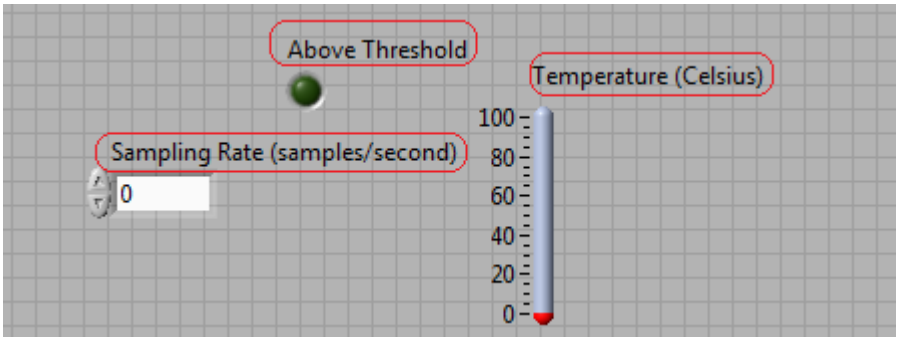
FRONT PANEL DESIGN

As discussed in previous sections, controls and indicators are front panel items that allow the user to interact with your program to both provide input and display results. When designing your front panel, you access controls and indicators by right-clicking the front panel to show the controls palette.

It is important to realize that the front panel gives the user a way to interact with the source code. It allows the user to change the values passed to the source code and see the data that the source code computes. When designing the user interface, choose controls and indicators that are appropriate for the task you want to perform. For example, when you want to set the sampling rate, use a numeric control that shows the increment/decrement button. When you want to indicate that a threshold value has been met, choose a Boolean LED indicator. When you want to display temperature, choose a chart or thermometer indicator.

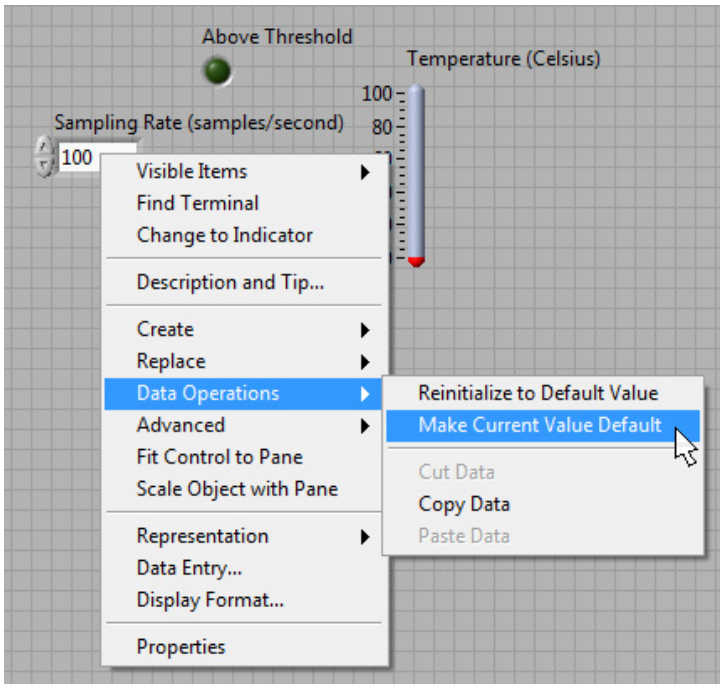


Make sure to label controls and indicators clearly. These labels help users identify the purpose of each control and indicator. Also, labeling helps you document your code on the block diagram. Control and indicator labels correspond to the names of terminals on the block diagram.

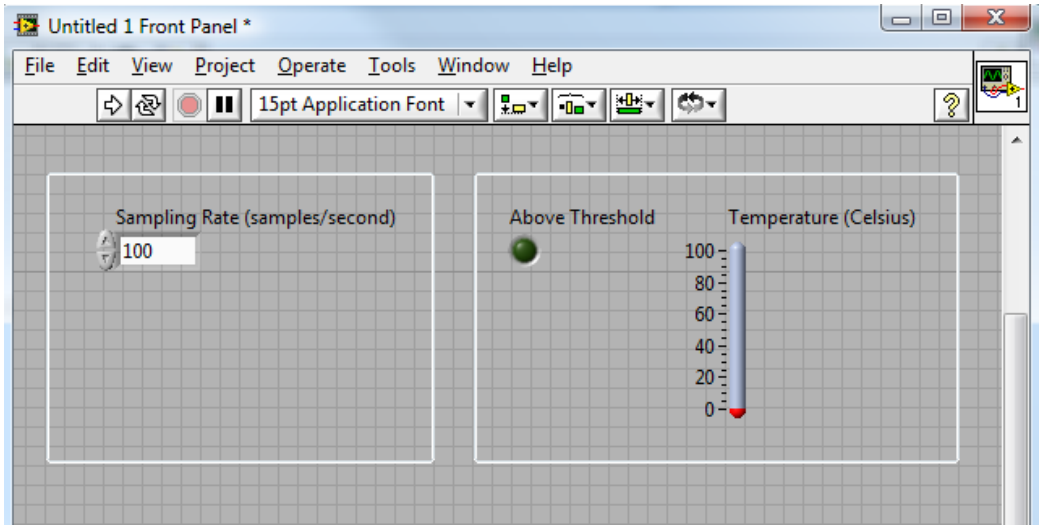


Spend time designing your front panel. LabVIEW offers many options for setting the properties of controls and indicators. You can view these properties by right-clicking the object and

browsing through the shortcut menu and submenus. For example, sometimes you might want your LabVIEW VI to open with default values for your controls. In this example, if you want the Sample Rate control to load with a default value of 100 samples/second, first change the value of this control to 100 and then set the property that makes the current value as the default value.



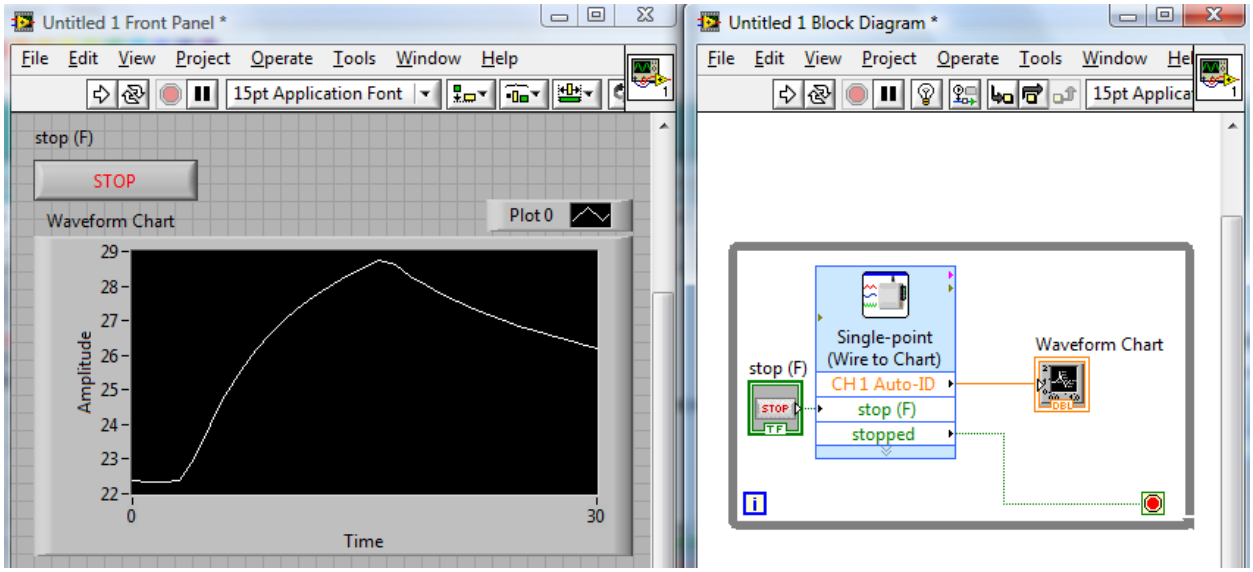
White space and alignment are probably the most important techniques for grouping and separation; the more items that your eye can find on a line, the cleaner and more cohesive the organization seems. When items are on a line, the eye follows the line from left to right or top to bottom. When you design the front panel, consider how users interact logically with the VI. Group controls and indicators, and consider the use of decorative borders (found in the Decoration subpalette). Leave some blank space between objects to make the front panel easier to read. Blank space also prevents users from accidentally clicking the wrong control or button. In general, spend some time spacing and aligning objects to provide users with a clean, easy-to-use interface.



GRAPHS AND CHARTS

When displaying sensor data, the most commonly used indicators are the graph and chart. A plot line on a chart or graph is a powerful visual display of your sensor measurements. Understanding the differences between a chart and graph is important in order to properly design your LabVIEW program.

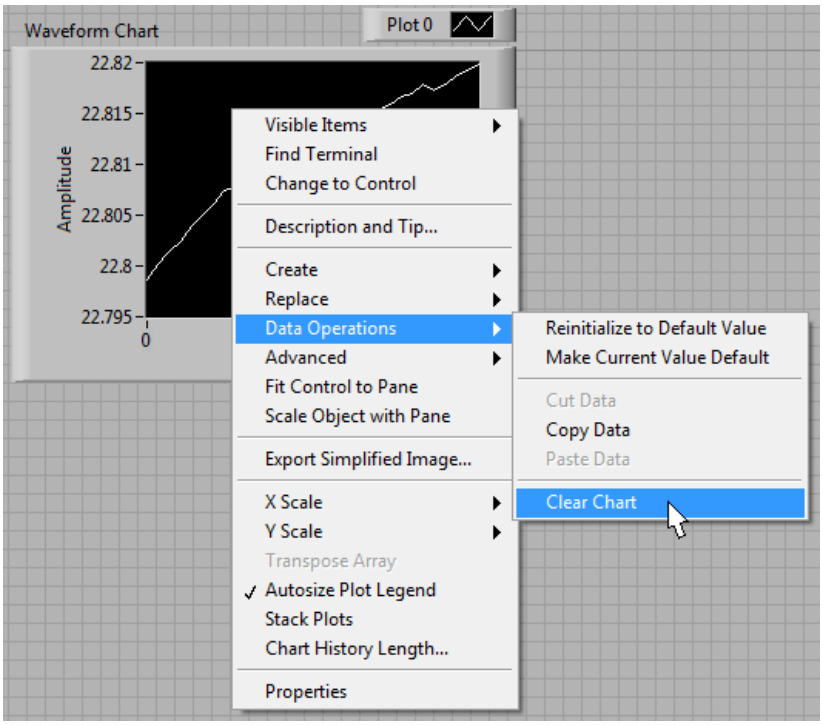
In general, a chart is used when performing slow data acquisition, updating the plot every time a data point is retrieved (point-by-point data acquisition). The plot on a chart is appended with the new data point to create a history. This allows the user to see the current reading or measurement in context with data previously acquired. When more data points are added than can be displayed on the chart, the chart scrolls so that new points are added to the right side of the chart, while old points disappear to the left.



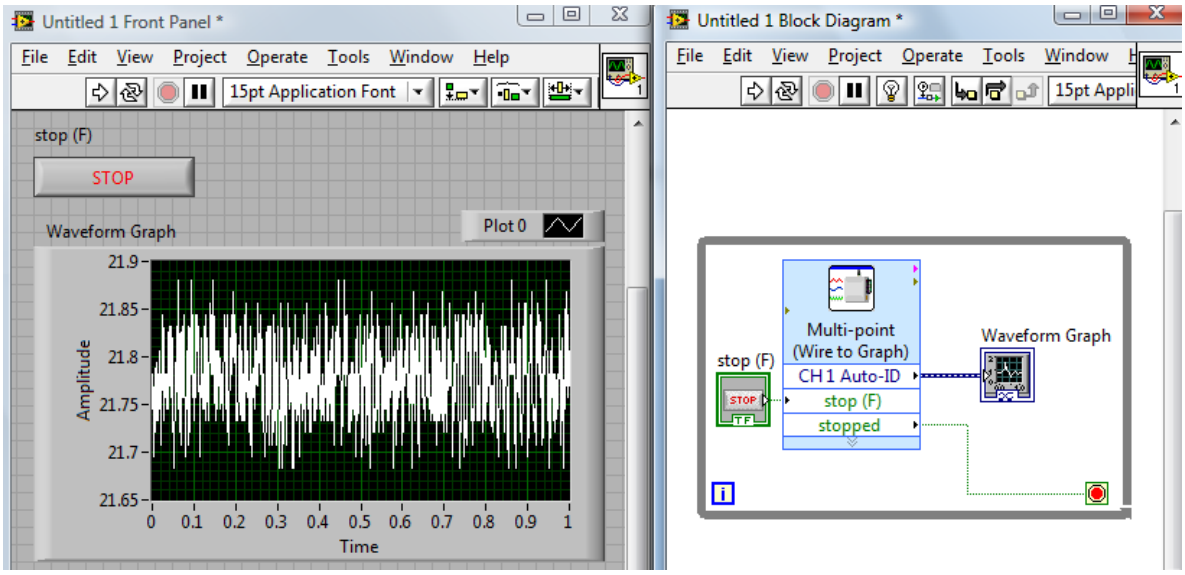
The program shown in the figure uses the Analog Express VI to collect single-point data at a rate of 1 sample/second for 30 seconds. With each iteration of the While Loop, a new data point is collected and appended to the plot line of the front panel chart.

An important thing to realize about a chart is that the history of data points is stored on the plot between runs of a LabVIEW VI. This means that if you run a VI and have a 30 second plot on your chart, the next time you run the VI, the new data points will be appended. The plot does not

automatically clear and start over at time of zero when you click the Run button. One method to clear the chart between runs is to go to the shortcut menu, and select Data Operations ► Clear Chart.



A graph, on the other hand, does not store previous data points. When the data are plotted, the graph discards the previously plotted data and displays only the new data. Only showing the new data makes a graph the correct choice of indictor when you are collecting and displaying large sets of data points (multi-point data acquisition) at a fast rate. A chart would not work for fast, multi-point acquisition, because it would continually be appending these large data sets. This creates memory issues, and does not provide the user with a helpful visual representation of the data.

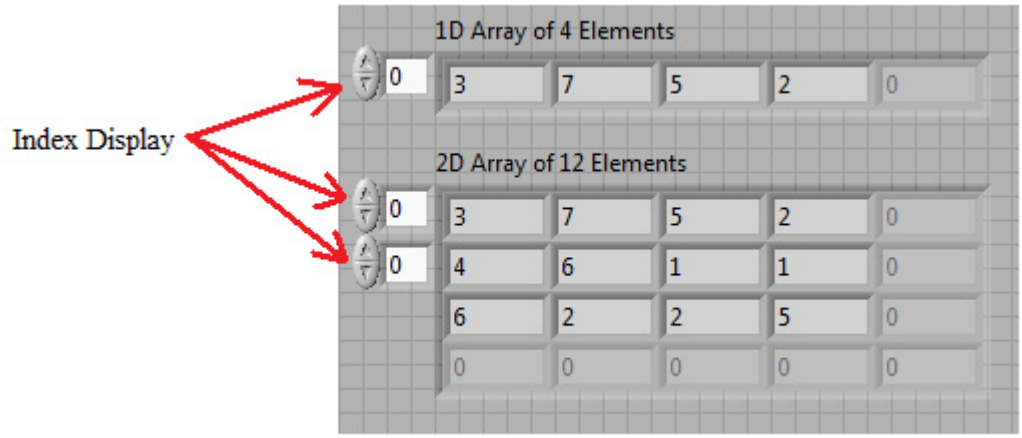


The program shown in the figure uses the Analog Express VI to repeatedly collect multi-point data at a rate of 1000 samples/second and a length of 1 second. When the loop iterates, the graph is updated with the most recent 1000 data points, providing the user with a constantly updated snapshot of the data.

The default look of a chart window is black with no gridlines. The default window of the graph window is with gridlines. However, these are properties that can be modified. In fact, the two indicators may be very hard to tell apart. If you are unsure, access the shortcut menu (right-click on the object) and open the Properties dialog box. The title of the dialog box will tell you whether it is a graph or a chart.

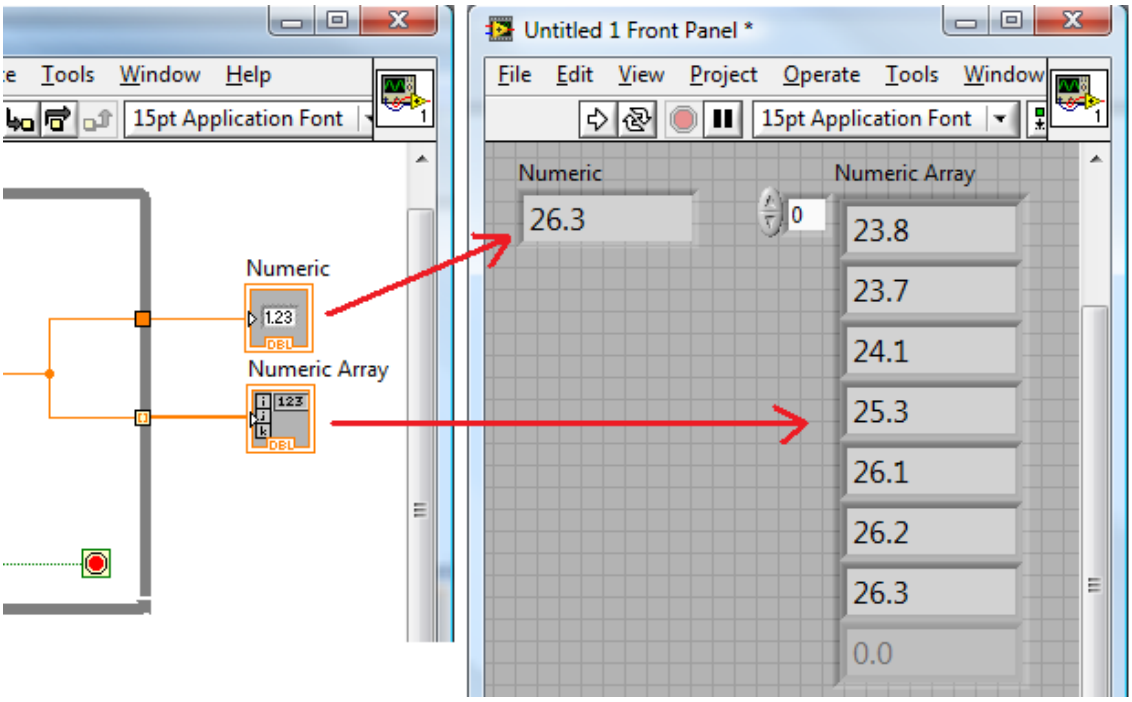
BUILDING ARRAYS WITH LOOPS

An array is an ordered, indexed list of data elements of the same type, similar to a matrix. An array, which consists of elements and dimensions, is either a control or an indicator—it cannot contain a mixture of controls and indicators. Elements are the data or values contained in the array. A dimension is the length, height, or depth of an array. Arrays are very helpful when you are working with a collection of similar data, and when you want to store a history of repetitive computations.



The figure above shows two front panel array indicators with numeric data elements. Array elements are ordered. The index display has up/down arrows that allow you to navigate to a particular element in the array. In LabVIEW software, the array index is zero-based. This means that if a one-dimensional (1D) array contains n elements, the index range is from 0 to n – 1, where index 0 points to the first element in the array, and index n – 1 points to the last element in the array.

For Loops and While Loops can index and accumulate arrays at their boundaries. This is known as auto-indexing. The indexing point on the boundary, called a tunnel, is created every time a wire crosses a loop boundary. The For Loop default is auto-indexing enabled. The While Loop default is auto-indexing disabled. The state of this tunnel is changed with the shortcut menu that appears when you right-click on the tunnel. Visually, the tunnel is solid when auto-indexing is disabled, and bracketed when auto-indexing is enabled. The wire exiting the loop is thicker when you use auto-indexing, because the wire contains an array instead of a single value.



An example of the use of auto-indexing would be to perform single-point data acquisition within a loop (temperature data, for example), and build the array by wiring the data point to the loop's border and auto-indexing the tunnel. Every data point that occurred (one point for each loop iteration) is stored in the tunnel. Upon exiting the loop, all data points are now stored in an array. This array can then be used in the next step of your program to perform analysis on the data or save to a file.

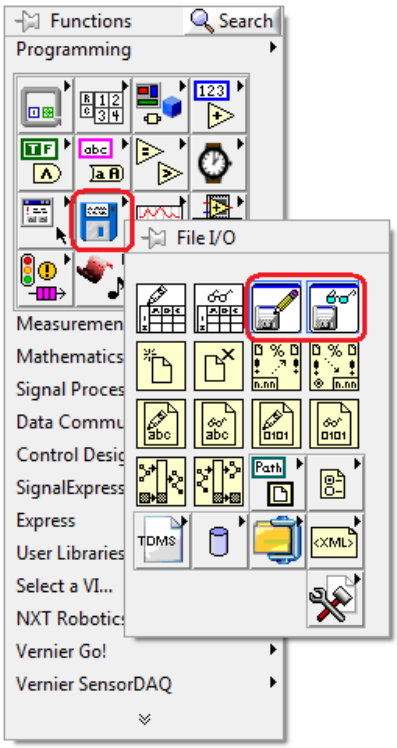
HIGH LEVEL FILE I/O FUNCTIONS

File I/O operations pass data from memory to and from files. High Level File I/O functions provide a higher level of abstraction to the user by opening and closing the file automatically before and after reading or writing data.

In LabVIEW, you can use File I/O functions to perform many operations on files, including:

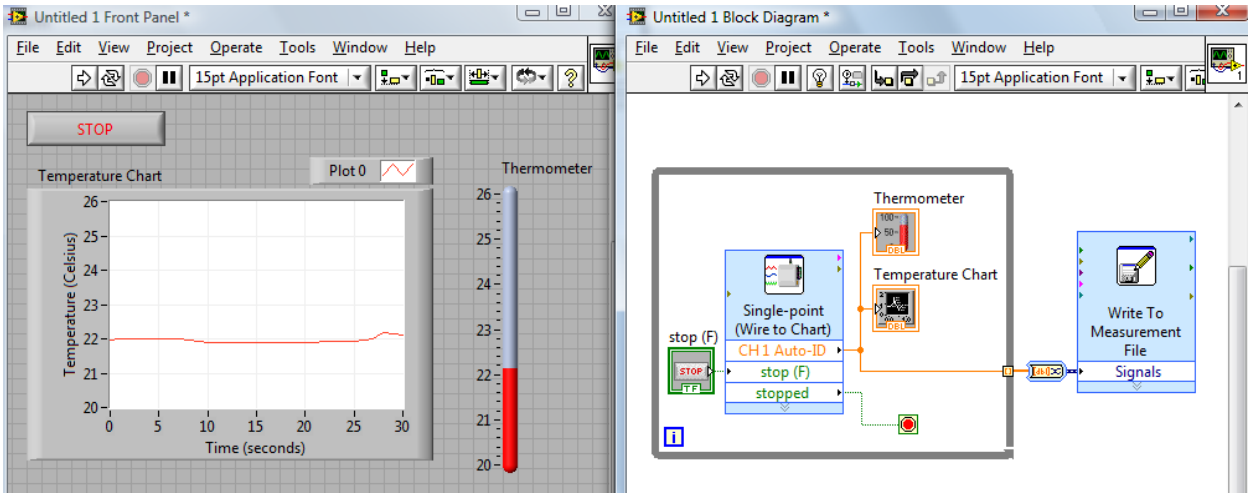
- Open and close data files
- Read data from and write data to files
- Read from and write to spreadsheet-formatted files
- Move and rename files and directories
- Change file characteristics
- Create, modify, and read a configuration file

The Write to Measurement File Express VI and Read from Measurement File Express VI are easy-to-use Express VIs that are excellent for simple applications. Use these to write data to a text-based measurement file (.lvm) or a binary measurement file (.tdm) format, and read back the data from the generated measurement file. You can specify the file name, file format, and segment size.



Exercise
5

Read Temperature Data



Completed front panel and block diagram

In this exercise, you will create a program using the Analog Express VI to collect temperature data and display it on a chart and thermometer. After all data have been collected, they are saved to a file using the Write To Measurement File Express VI.

OBJECTIVES

In this exercise, you will

- Create a LabVIEW VI to collect and display point-by-point temperature data.
- Incorporate LabVIEW Express File functions.
- Use a While Loop tunnel configured to auto-index to build an array of all data points.

MATERIALS

SensorDAQ or LabQuest interface
USB cable
computer
LabVIEW
Vernier Stainless Steel Temperature Probe

PROCEDURE

Part I Connect Equipment

1. Connect the USB cable to the SensorDAQ or LabQuest interface.

2. Connect the other end of the USB cable to any available USB port on your computer. If you are using a LabQuest interface with a power button, turn it on.

3. Connect the Temperature Probe to Ch. 1.

Part II Start LabVIEW and Create a VI to Collect Data

4. Start LabVIEW.

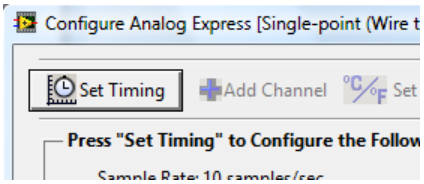
5. From the Getting Started window, click the Blank VI link in the New category.

6. View the block diagram using the <Ctrl-E> shortcut.

7. Drag the Analog Express from the SensorDAQ or LabQuest palette to the block diagram. Access this palette by right-clicking the block diagram workspace.

8. After dragging the Express VI from the palette to the block diagram workspace, the Express VI's configuration popup will open. Note that this step can be slow, depending on your computer.

9. Click the Set Timing button, located in the upper-left corner of the configure dialog.

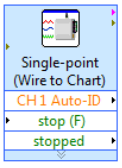


10. Set the timing with a length of 30 seconds and a sample rate of 1 sample/second.

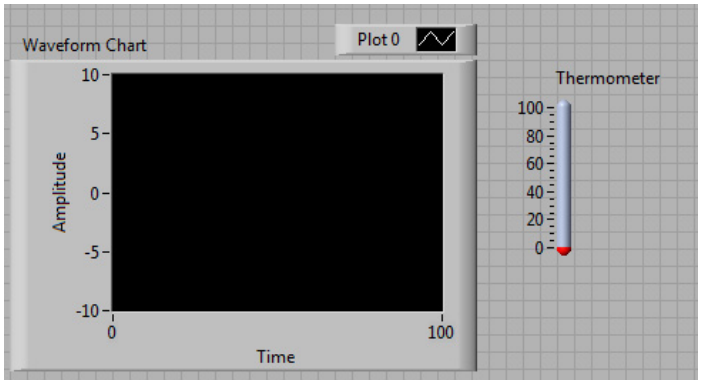
Tip: Repeat is not selected. Repeat would be selected if, at the end of our 30 second data collection run, we did not want data collection to end. In this example, we do want data collection to end after 30 seconds. Selecting Repeat makes sense for very short and fast data collection runs, such as fast collection with a microphone. In most cases, it does not make sense for slow data collection. If you want to repeat a 30 second data collection, it is better to simply change the length to 60 seconds.

11. Click Done to close the Set Timing popup. The Express VI Configuration should now be updated with the new settings.

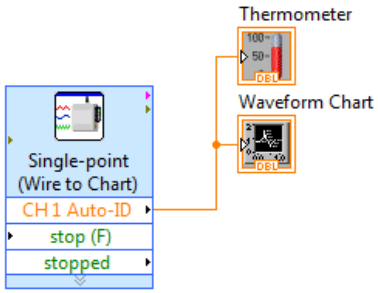
12. Click OK to close the Analog Express VI's Configuration window. The Analog Express VI will now be located in your block diagram workspace.



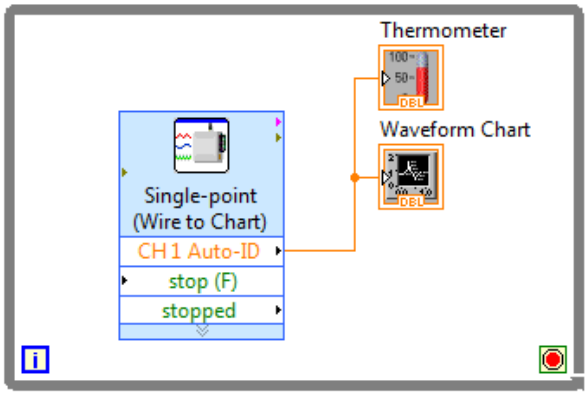
13. View the front panel using <Ctrl-E> and add a chart and thermometer.



14. Go to the block diagram and wire the CH 1 Auto-ID output terminal to the chart and thermometer.



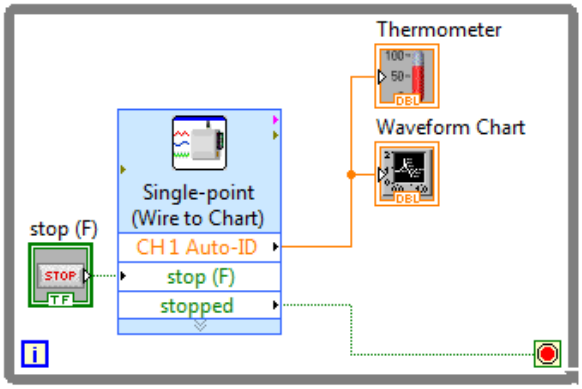
15. Place this code within a While Loop.



Tip: The Analog Express VI has been configured to collect a single data point every second for 30 seconds. The goal of this program is to show the user every data point as it occurs (as compared to a program that collects all of the data first, and then shows the data to the user after 30 seconds). This cannot be done without the use of a While Loop. Express VIs, subVIs, functions, and other objects on a block diagram do not automatically repeat; but placed inside a loop, the code is forced to repeat. On the first iteration of the loop, the Express VI reads a single data point. The loop repeats and the Express VI reads the second data point. The loop continues to repeat the code until the loop is stopped.

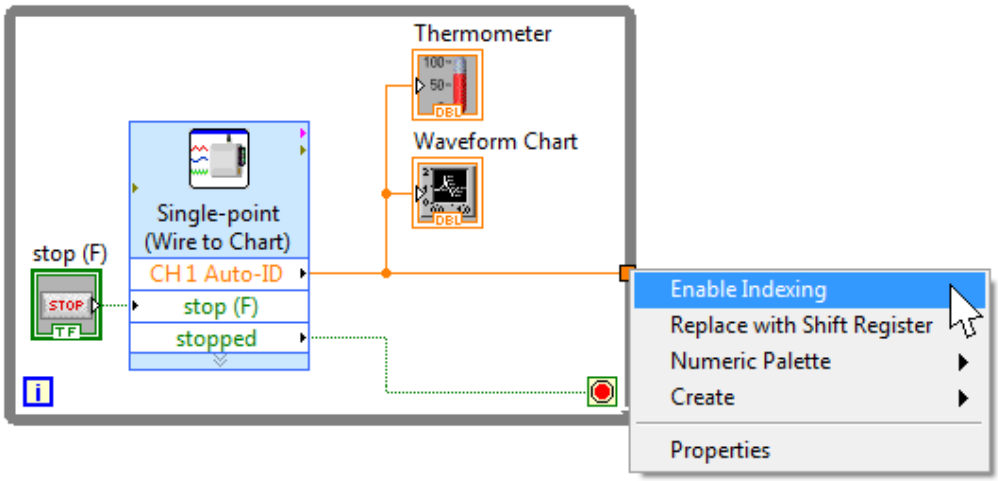
16. Right-click the Express VI's "stop (F)" terminal and select Create ► Control to create a STOP button control for the Express VI.

17. Wire the Express VI's "stopped" terminal to the While Loop's conditional terminal.



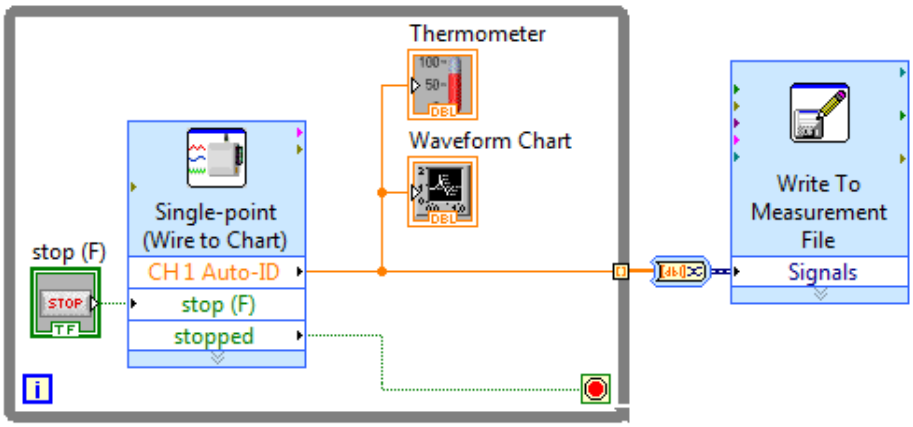
Tip: Proper shutdown of the interface hardware requires that you stop the Express VI prior to ending the program. This is accomplished by wiring the STOP button into the Express VI, rather than wiring it to the While Loop's conditional terminal. The Express VI's "stopped" output terminal will send a Boolean value of true (to stop the While Loop) when data collection is completed. This occurs when the user clicks the STOP button, or when all data points have been collected, as configured by the user.

18. Build an array of the data points by wiring the data wire to the border of the While Loop. Right-click the Loop Tunnel and select enable indexing.



19. Place the Write To Measurement File Express VI (found in the Programming ► File I/O palette) on the block diagram outside the While Loop. Use the default configuration and make a note of the default file name and where the file will be saved. Modify this name and location, if necessary.

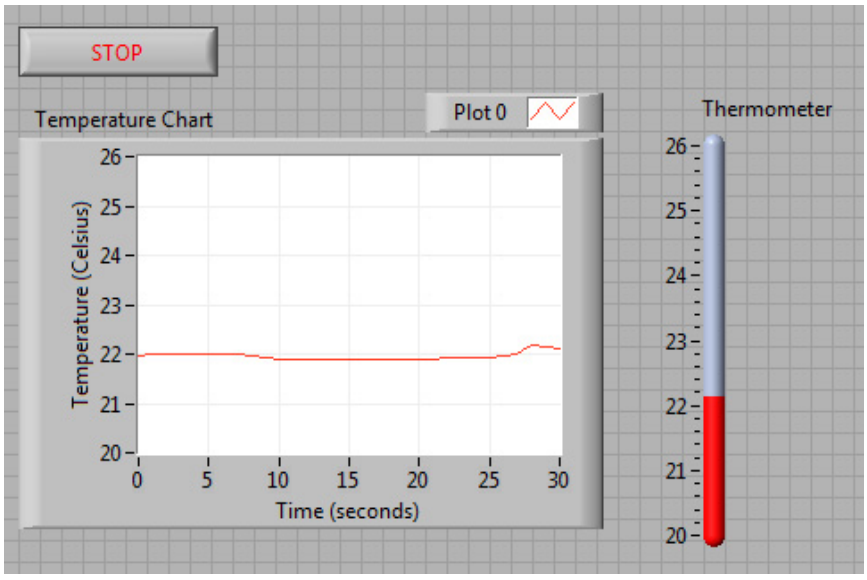
20. Wire from the loop tunnel to the Signals input terminal.



Tip: Most Express VIs accept and/or return the dynamic data type (dark blue terminal and wire). This special data type will automatically convert other data types when wired into a dynamic data terminal. In this example, the 1D array from the tunnel is automatically converted to dynamic data. The conversion occurs with a small Express VI that is automatically inserted into the wire. This Express VI  is called Convert to Dynamic Data. It is not the typical large blue box; however, you can still double-click on it to open a configuration dialog box when required.

21. View the front panel using <Ctrl-E>.

22. Spend some time cleaning up the front panel user interface. Align objects, provide new axis labels, change the x-axis to have a maximum value of 30 to match the data-collection length, provide proper labels, use the paint brush to add appropriate colors, etc.



23. Run the VI.

24. Only click the STOP button to end data collection prematurely. Otherwise, after 30 seconds, the Analog Express VI will send out a Boolean with a True value (from the Express VI's stopped terminal) to stop the While Loop.

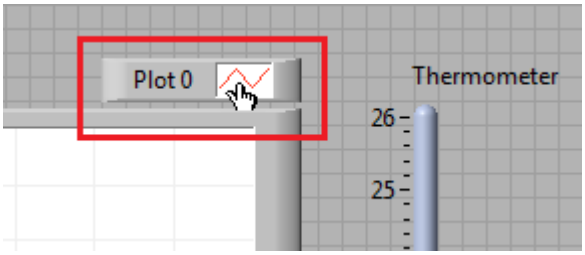
25. Locate the saved data file. If needed, go back to the Write To Measurement File Express VI and open it (double-click an Express VI to open the configuration dialog box) to see how it is configured to save the file. Once the file has been found, you can open it with a spreadsheet application or a text-editing application.

26. Run the VI a second time. Note that the new data are appended to the plot of the first run. Right-click the chart and select Data Operations ► Clear Chart. Run the VI again and note that the new data are starting from time zero.

EXTENSIONS

1. Clear the chart first. Go to the block diagram and change the sample rate to 10 samples/second, and then run the program. Notice that the chart x-axis is not configured correctly. Namely, each data point appears at a time interval of 1 second. The chart in this example is not smart enough to know that you have modified the sample rate. It is up to you, the programmer, to modify the chart properties that allow the data to display on the x-axis at the proper interval. Right-click the chart and select the Properties dialog box. Modify the x scale Multiplier property so that the scale is properly configured to display the data at an interval that matches the sample rate.

2. At the upper-right corner of the chart is a plot legend. Bring your cursor to the icon that looks like a plot line that appears next to the "Plot0" legend and click on it. A dialog provides a list of properties for modifying the plot line. Modify the plot style to only display the data points with a solid, round, green point. With this change, the data points will not be connected with a plot line.



3. Study the insulating properties of different socks by comparing the rate of cooling of a bottle of warm water with a sock insulating the outside of the bottle.

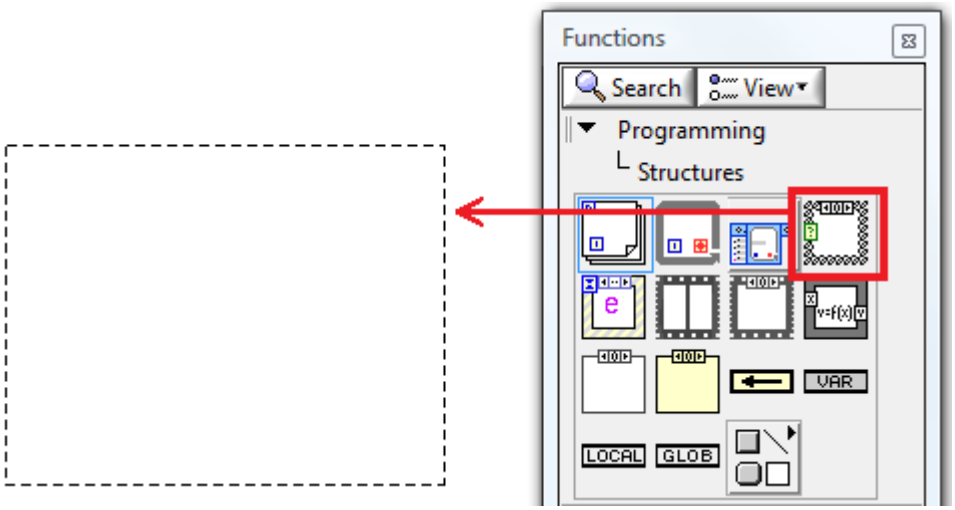
Chapter
6

Decision Making

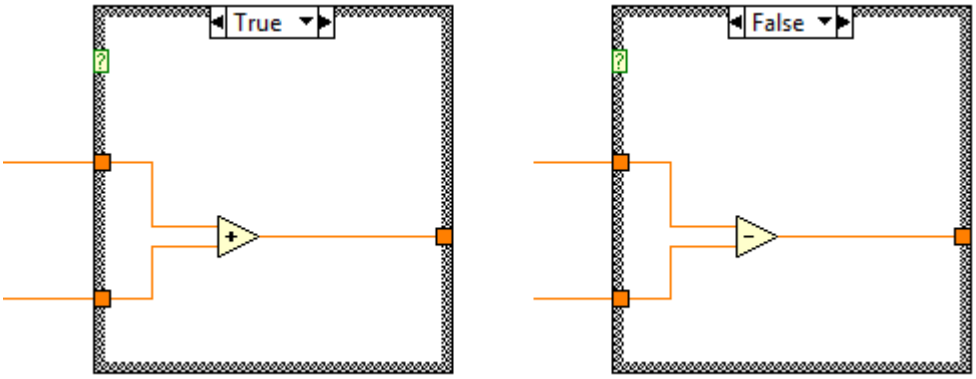
CASE STRUCTURE

When creating a program, you may want to execute (or bypass) code based on an input. The input may be user input (for example, pushing a front panel button) or input that is a result of the code in your program (for example, the reading of a sensor).

The Case Structure is a programming structure that has one or more subdiagrams, or cases, only one of which executes when the structure executes. The value wired to the selector terminal determines which case to execute. This structure is located on the Functions ► Structures palette. First, select the structure from the Structures palette, then use your cursor to drag a selection rectangle to size the structure.

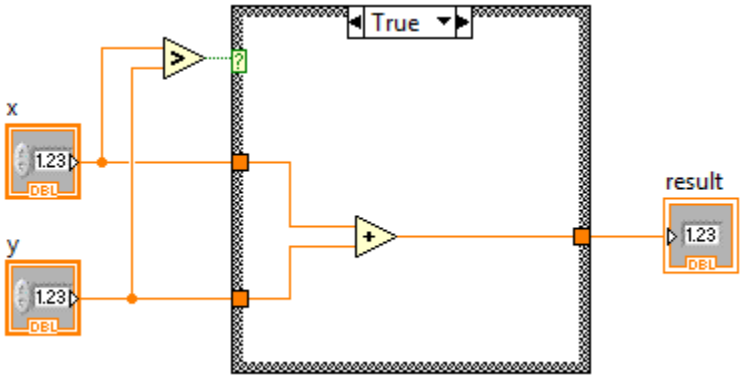


When you release the mouse button, the Case Structure boundary is set. The structure can be resized as necessary. Then add your code inside each subdiagram, or case, of the structure. The default is two cases, but there can be multiple cases. The hidden cases can be viewed by clicking the right or left arrows at the top of the case structure.



The diagram above is not complete because the case selector terminal has not been wired. The case selector terminal appears as a small question mark (?) on the left side of the case structure. The input value wired to the selector terminal determines which case to execute and can be

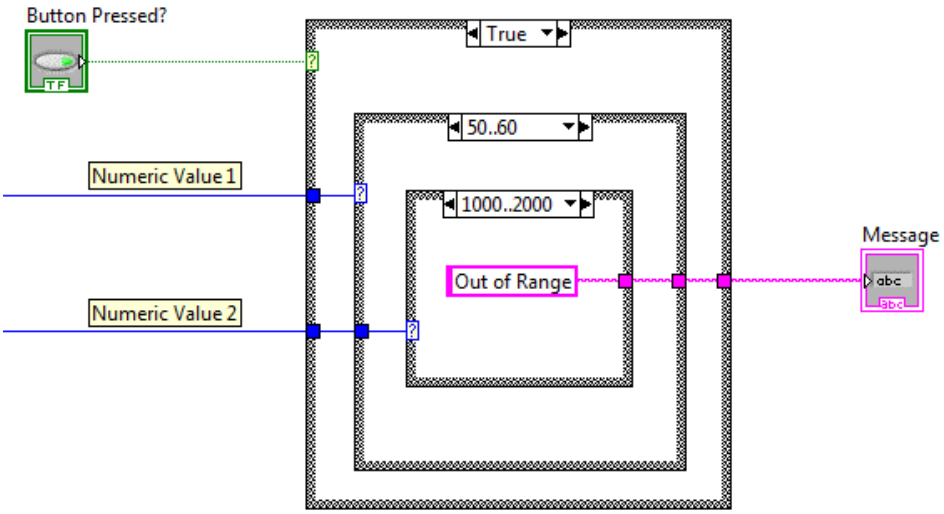
Boolean values, string, integer, or enumerated type (a control that provides a list of items). A case structure with a Boolean wired to its selector terminal has a maximum of two cases; all other data types allow two or more cases.



The figure above is now completely wired. The case selector input in this example is a Boolean value that is based on the result of a comparison of the two numeric values, x and y. If x is greater than y, the True case is executed and the two values are added. If x is not greater than y, the False case is executed (this case is not shown in the figure above), and the y numeric is subtracted from x.

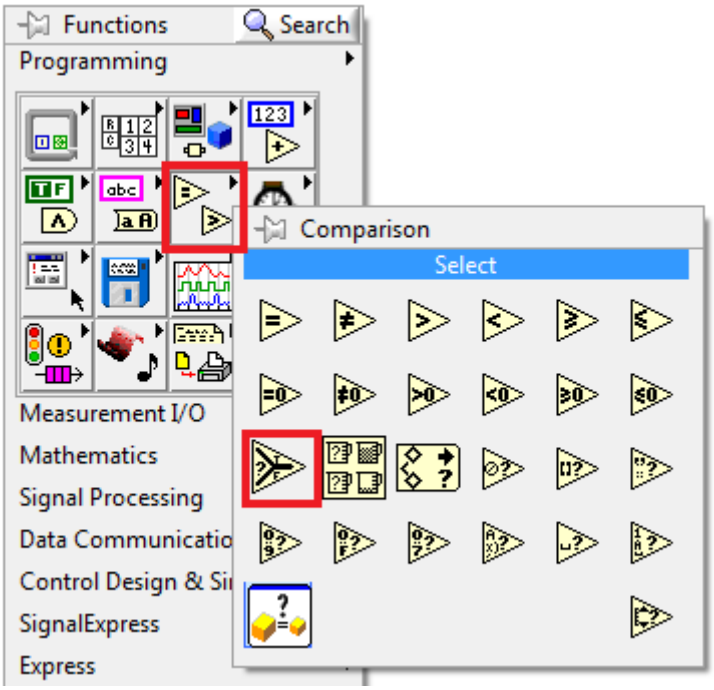
It is important to note that the Case Structure can also accommodate lists and ranges of values. For lists, use commas to separate values. For numeric ranges, specify a range as 10..20, meaning all numbers from 10 to 20 inclusively. You can also use open-ended ranges. For example, ..100 represents all numbers less than or equal to 100, and 100.. represents all numbers greater than or equal to 100. For string ranges, a range of a..c includes all strings beginning with a or b, but not c. A range of a..c,c includes the ending value of c.

Placing a Case Structure within a Case Structure (i.e., creating a nested Case Structure) increases the decision making power. For example, you may need to use two numeric values to determine what message to send to the user, but you only want to send the message if the user has pressed a button.

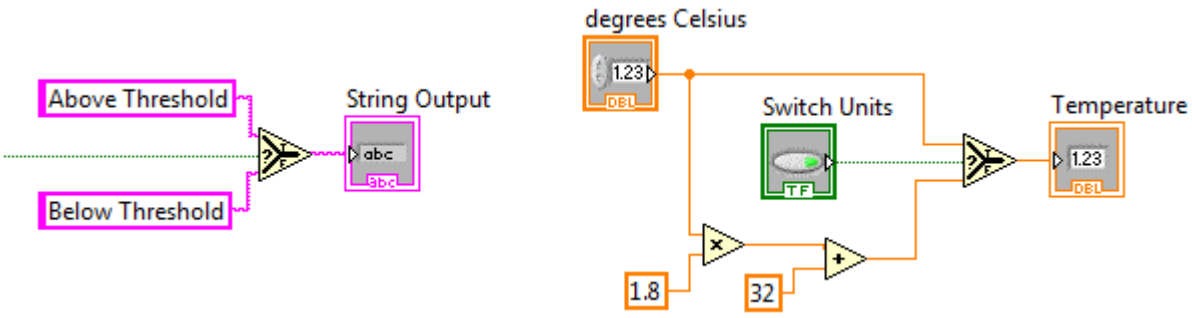


SELECT

The Select function is found in the Functions ► Programming ► Comparison palette.



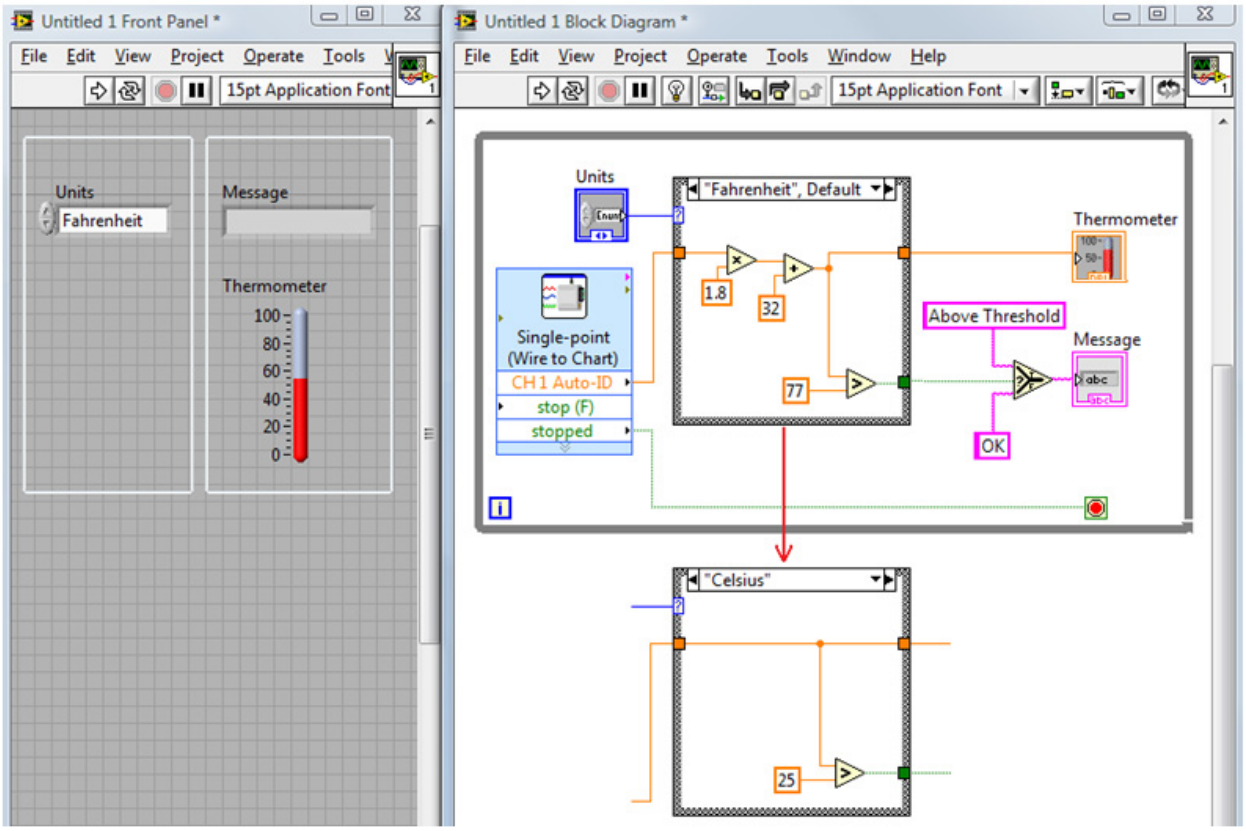
This function returns one of two values, depending on the value of the conditional input. If the conditional input is TRUE, this function returns the value wired to the "t" terminal. If it is FALSE, this function returns the value wired to the "f" terminal. In other words, your program selects one of two values.



The figure above shows two examples of using the Select function. In the example on the left, the Boolean value input determines what string is passed to the String Output. If the Boolean value is True, the String Output indicator will display the "Above Threshold" string. If it is False, it will display the "Below Threshold" string. In the example on the right, the Boolean control called "Switch Units" determines if the value is displayed in Celsius or Fahrenheit units.

Exercise
6

Above Threshold Warning of Temperature Data



Completed front panel and block diagram. The hidden case "Celsius" is also shown in this picture.

In the following exercise, you will create a program using the Analog Express VI that records temperature data. The code will allow the user to display the data in Celsius or Fahrenheit units. In addition, a threshold value is compared to the temperature measurement and a message is displayed to alert the user when the temperature is above the threshold.

OBJECTIVES

In this exercise, you will

- Create a LabVIEW VI to collect temperature data.
- Become familiar with an Enumerated control.
- Warn the user when the temperature is above a threshold value.
- Incorporate code that makes decisions based on user input and numeric comparison.

MATERIALS

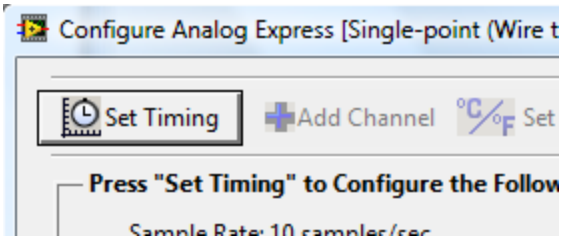
SensorDAQ or LabQuest interface
USB cable
computer
LabVIEW
Vernier Stainless Steel Temperature Probe

Part I Connect Equipment

1. Connect the USB cable to the SensorDAQ or LabQuest interface.
2. Connect the other end of the USB cable to any available USB port on your computer. If you are using a LabQuest interface with a power button, turn it on.
3. Connect the Temperature Probe to Ch. 1.

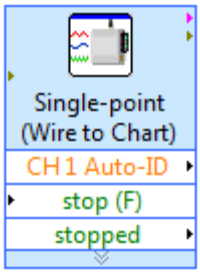
Part II Start LabVIEW and Create a VI to Collect Data

4. Start LabVIEW.
5. In the Getting Started window, click the Blank VI link in the New category.
6. View the block diagram using the <Ctrl-E> shortcut.
7. Click and drag the Analog Express VI from the SensorDAQ or LabQuest pallet to the block diagram. Access this palette by right-clicking the block diagram workspace.
8. After dragging the Express VI from the palette to the block diagram workspace, the Express VI's configuration popup will open. Note that this step can be slow, depending on your computer.
9. Click the Set Timing button, located in the upper-left corner of the configuration window.



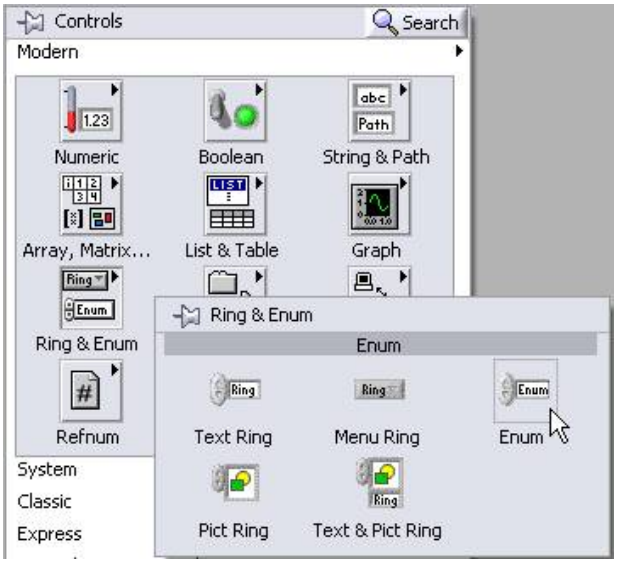
10. Set the timing with a length of 20 seconds and a sample rate of 2 samples/second.
11. Click Done to close the Set Timing window. The Express VI Configuration should now be updated with the new settings.

12. Click OK to close the Express VI's Configuration window. The Analog Express VI will now be located in your block diagram workspace.



13. View the front panel using the <Ctrl-E> shortcut.

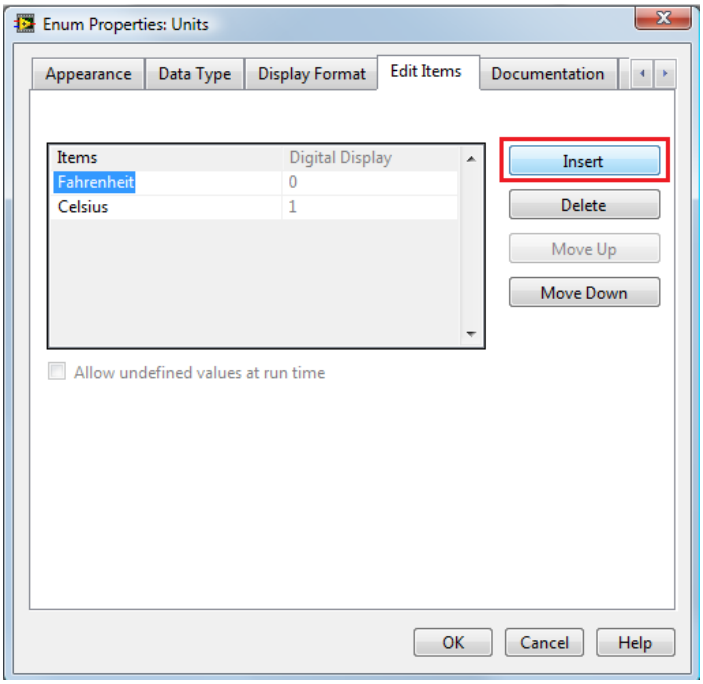
14. Place an Enum control on the front panel. This control is located in Controls ► Modern ► Ring & Enum. Rename the control "Units."



Tip: An enumerated type control, or Enum, contains lists of strings through which you can cycle. When you wire an enumerated type control to the selector terminal of a Case structure, LabVIEW matches the cases to the string values of items in the control, not the numeric values. In addition, you can right-click the Case Structure, and select Add Case for Every Value, to create a case for the string value of each item in the control.

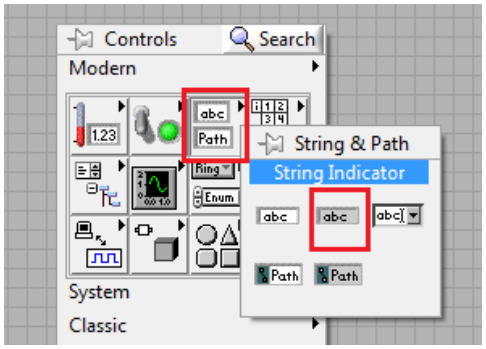
15. Right-click the Enum and select Edit Items from the menu.

16. Click the Insert button and add the following Items: Celsius and Fahrenheit.



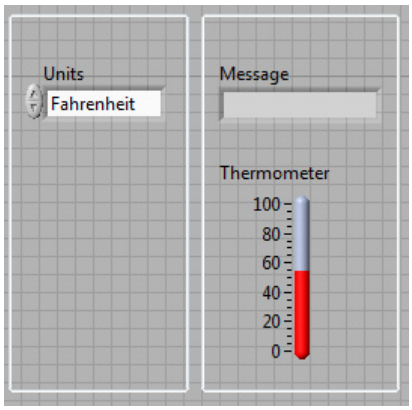
17. Click OK.

18. Add a string indicator to the front panel and name it "Message".



19. Add a thermometer indicator to the front panel.

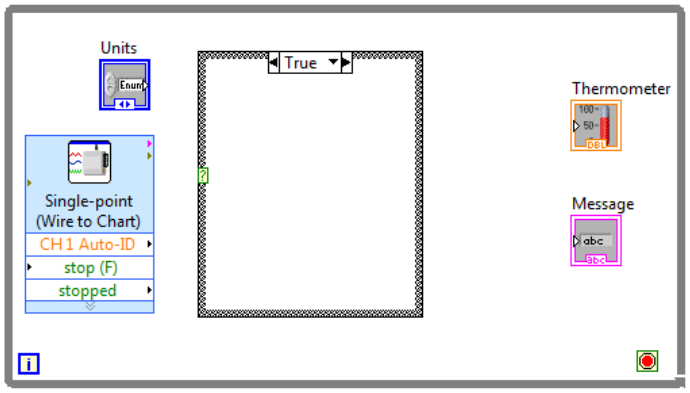
20. Clean up your front panel control and indicators.



21. View the block diagram using the <Ctrl-E> shortcut.

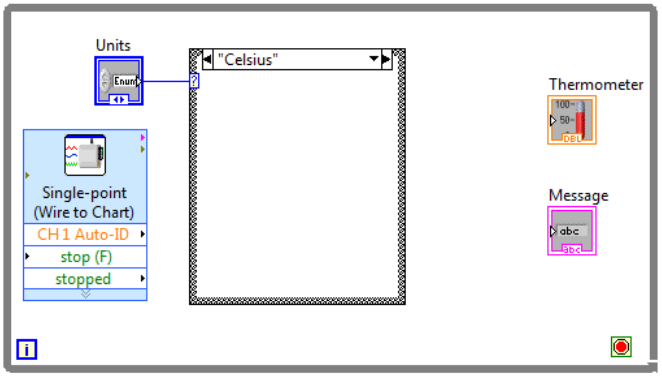
22. Encircle all of the code with a While Loop and move the two indicators to the right side of the loop. Move the Analog Express VI and Units control to the left side of the loop.

23. Add a Case Structure.



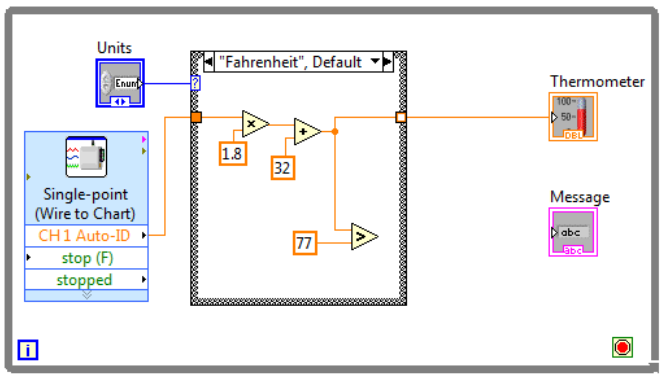
24. Click and drag the selector terminal (the small question mark (?) on the left side of the case structure) up to the upper-left corner of the Case Structure to be closer to the Units control.

Wire the Units control to the Case Structure, and note how the name of the two cases changes to match the input.

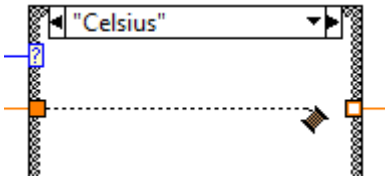


25. Go to the Fahrenheit case and add code to change the measurement value to Fahrenheit. Wire the result to the Thermometer. In addition, perform a greater than comparison of the reading to a value of 77 (degrees Fahrenheit).

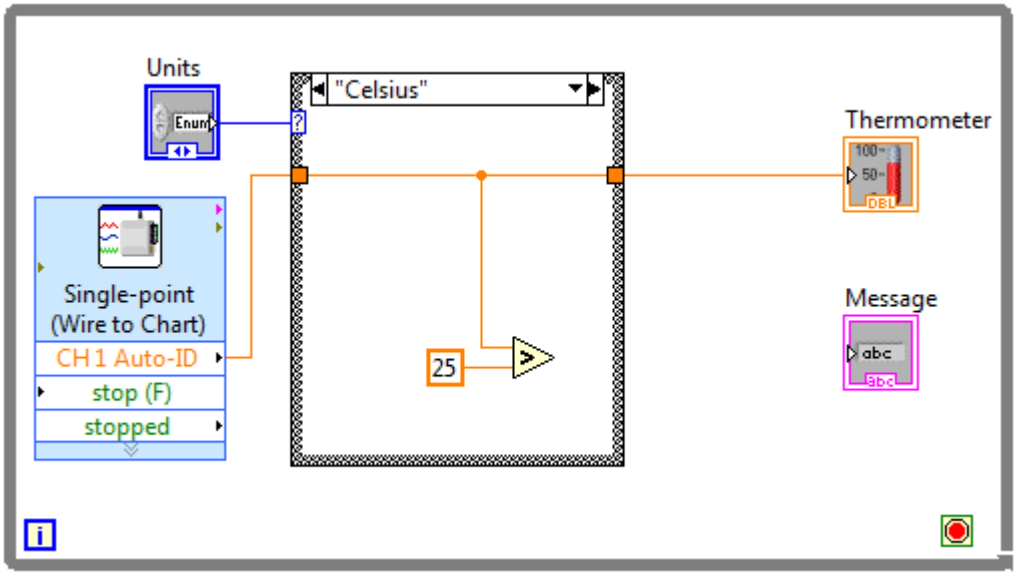
Tip: The tunnel of a Case Structure must be solid. A solid tunnel signifies that every case has data wired out of the Case Structure. A Case Structure cannot allow some cases to pass data out of the structure, while other cases do not. All cases must wire data out of the tunnel; otherwise, the tunnel will appear hollow and the VI will have a broken arrow and will not run.



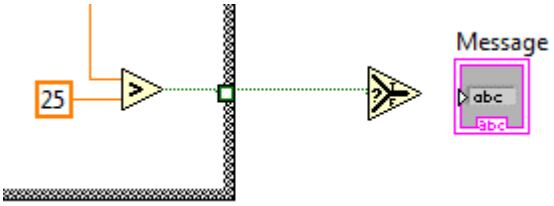
26. Change to the Celsius case and create a wire path from the input tunnel (where the measurement enters the case structure) to the output tunnel (where it exits). The tunnel will change from appearing hollow to appearing solid in color.



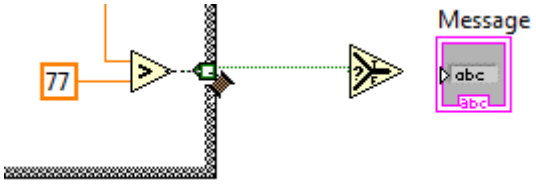
27. Add code to the Celsius case to perform a greater than comparison of the reading to a value of 25 (degrees Celsius).



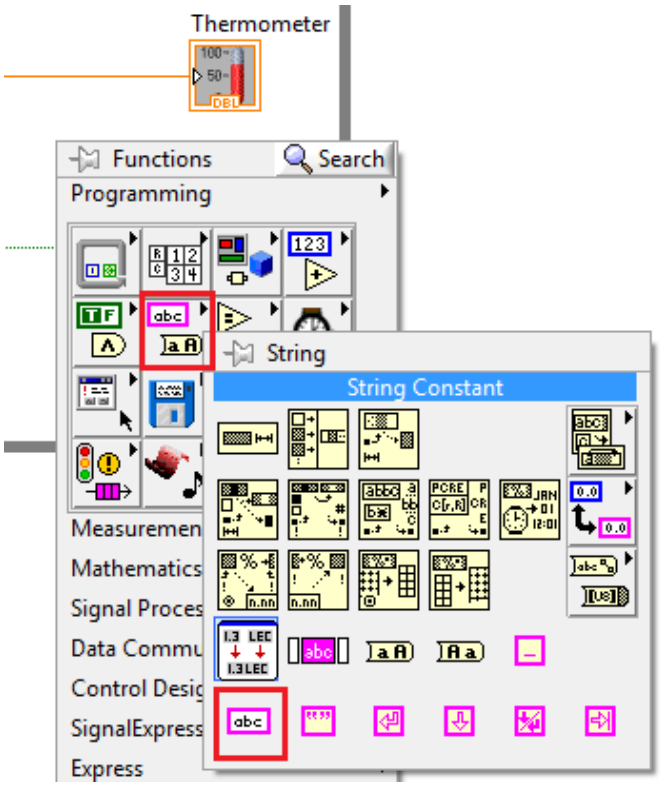
28. Add a Select function (found in the Programming ► Comparison palette) to the right of the Case Structure, in front of the Message indicator. Wire the result of the Greater? function in the Celsius case to the conditional input of the Select Function.



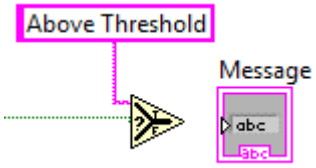
29. Go to the Fahrenheit case and wire the result of the Greater? function to the output tunnel. The tunnel will change from appearing hollow to appearing solid in color.



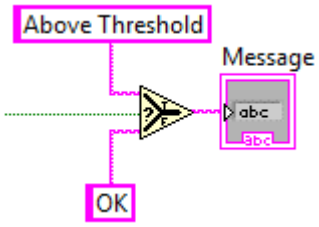
30. Add a String Constant function to the block diagram workspace next to the Select function. A String Constant is found in the Programming ► String palette.



31. When the String Constant is placed on the block diagram workspace, it will not contain any text. Type the words "Above Threshold" and wire the String Constant to the "t" input of the Select function.

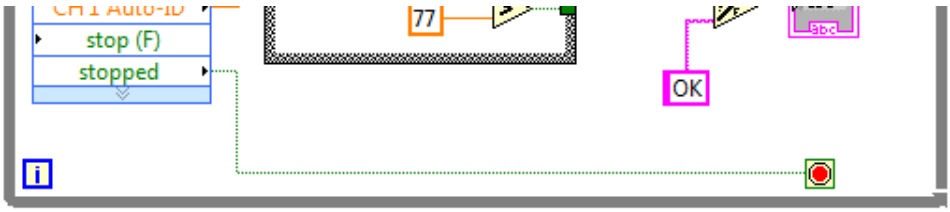


32. Add a second String Constant with "OK" and wire it to the "f" input of the Select function. Wire the result of the Select function to the Message indicator.

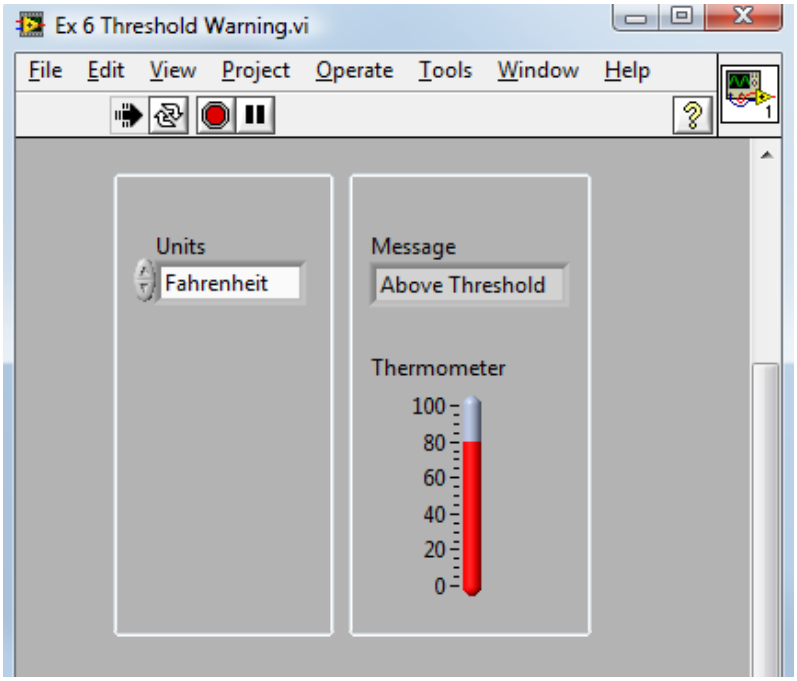


33. Wire the Analog Express VI's stopped output terminal to the While Loop's conditional terminal.

Tip: In this example, a STOP button is not wired into the Express VI's "stop (F)" input terminal. This means the Express VI will only send a Boolean true value when data collection is completed (as configured by the user in the Express VI). A STOP button may be added to provide the user with a means to cancel data collection prematurely.



34. Go to the front panel and run the VI. Change the Units and observe the reading on the Thermometer. Heat and cool the Temperature Probe so that the reading goes above and below the threshold.



EXTENSIONS

1. The threshold limit value is currently a numeric constant to compare against the Celsius reading, and a second constant to compare against the Fahrenheit reading. Modify the program so that there is only a single threshold value in degrees Celsius, and it is a front panel user control. Provide a front panel indicator that shows the threshold value in degrees Fahrenheit.

2. Add a lower limit threshold control and a lower limit message of "Below Threshold". There will now be a total of three possible messages: Above Threshold, OK, and Below Threshold.

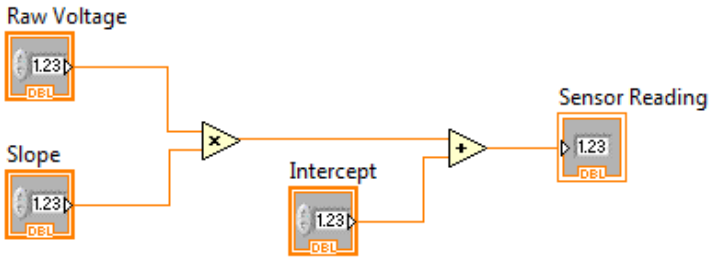
Chapter
7

SubVIs

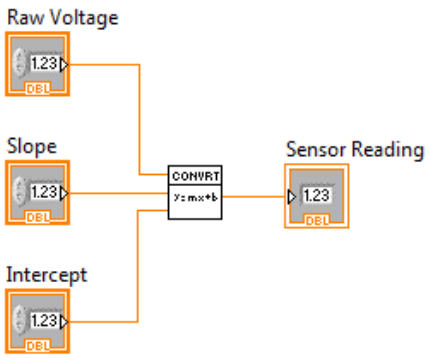
CREATING SUBVIS

After you build a VI, you can use it in the block diagram of another VI. A VI called from the block diagram of another VI is called a subVI. Creating and using subVIs within your LabVIEW program is essential for keeping your code readable, reusable, and easy to modify. In addition, it is helpful for debugging a program. In other programming environments, sub VIs would be called "subroutines."

A subVI works best if it is code that is focused on performing a single discrete task. Again, this helps for readability, reusability, and debugging.



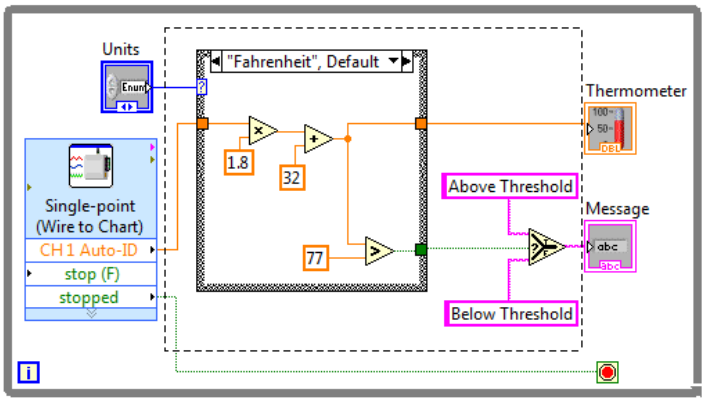
For example, in the figure above, the code is taking the raw voltage reading of a sensor and computing the reading in proper sensor units using a linear conversion. This conversion code is useful because it performs a discrete task that could be used in many other VIs. A subVI could be created from this code, and then used in the block diagram of other VIs that require this linear conversion. The figure below performs the same task, but performs the conversion within the code of the subVI.



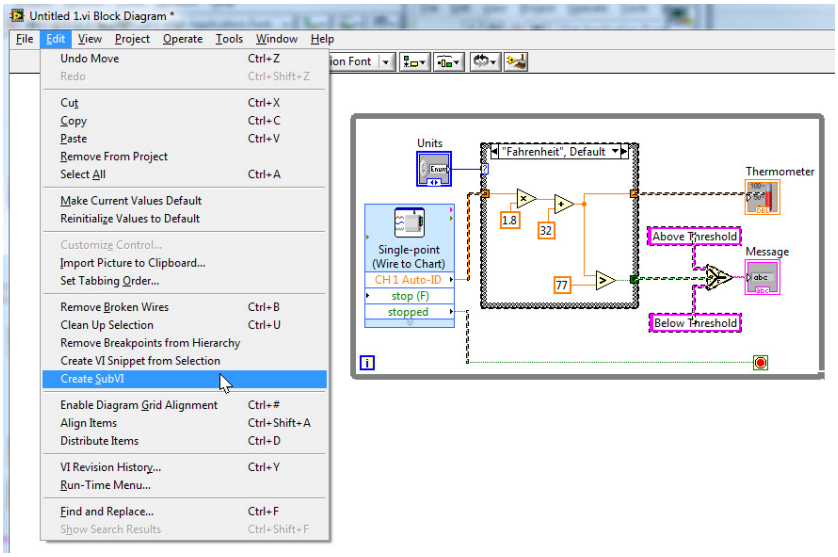
When you create a subVI and use it, you see an icon within your block diagram that represents the subVI. SubVIs are like VIs in that they contain front panels and block diagrams, but you call them from within a VI. The block diagram of a subVI may also contain subVIs, and the block diagram of these subVIs can contain subVIs. There is no limit to the number of nested subVIs.

There are two methods for creating subVIs. One is to open a New Blank VI and design that VI to be used as a subVI. The steps to convert a VI into a usable subVI include customizing the icon for the subVI and configuring the input and output terminals for your block diagram wiring. The steps for this method are explained in the next section.

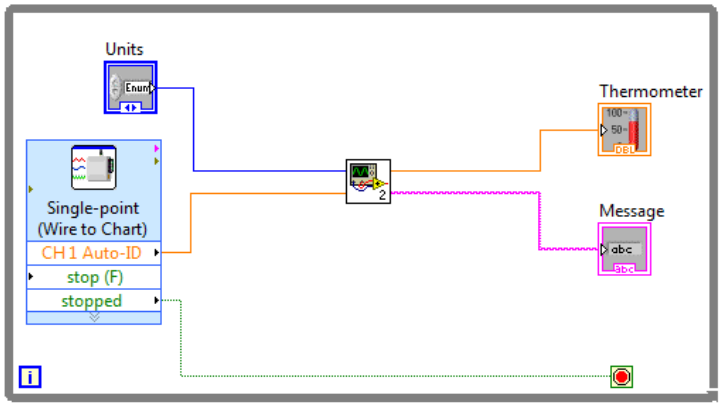
The other method is to convert a section of an existing VI into a subVI. To do this, start by using the Positioning tool to select the section of the block diagram you want to turn into a subVI.



After the selection has been made, choose Create SubVI from the Edit menu.

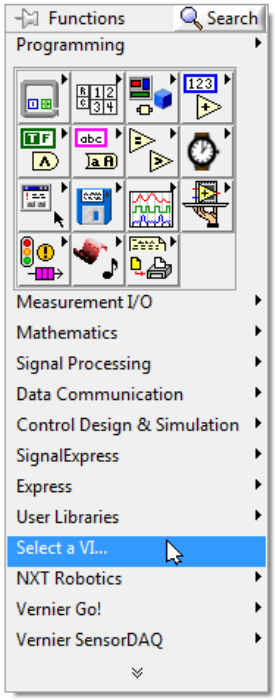


The selected code is automatically turned into a subVI. LabVIEW creates controls and indicators for the new subVI, configures the connector pane based on the number of control and indicator terminals you selected, and wires the subVI to the existing wires. This is now a functional subVI.



It is good practice to open the front panel of this subVI and clean it up. To open the front panel of a subVI from the calling VI, double-click the subVI on the block diagram. Like any other VI, to view the block diagram use the <Ctrl-E> shortcut. To open the front panel without opening the calling VI, simply open it like any other VI, by choosing Open from the File menu.

When you have created a subVI and saved it, it will be available to be used again. To access the subVI, go to the block diagram workspace, open the Functions palette, and choose Select a VI.



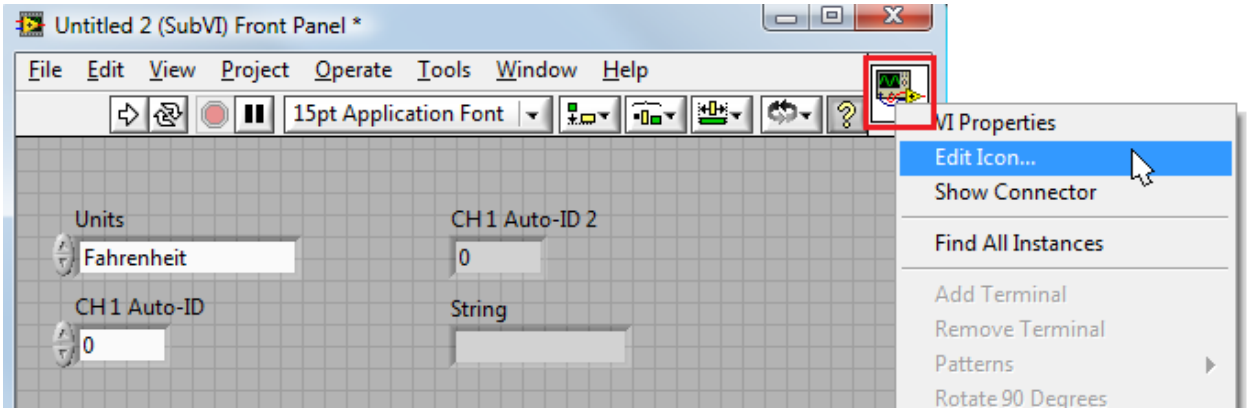
Browse to your subVI, and select it to drop it into the block diagram of your VI. If you edit and save a subVI, the changes affect all future calls to the subVI, not just the current instance. The subVI will contain all the saved changes, even if the calling VI is not open at the time of the changes. Once the calling VI is opened, the subVI will be loaded onto the block diagram with any changes previously made and saved.

It is important to note that the subVI is a separate file; therefore, if you are sharing a LabVIEW program that contains a subVI, you must send both the program and the subVI file. If you only send the main VI, and a user on a different computer tries to open it, it will open with a broken Run button and an error message asking for the missing subVI file.

ICON

SubVIs show up on the block diagram as an icon. LabVIEW provides a default icon that looks like the icon below. This icon is also viewed when you open the front panel of the subVI. It is in the upper-right corner.

The icon can be customized using the Icon Editor dialog box. Access the Icon Editor by double-clicking the icon in the upper-right corner of the front panel. You can also access the Icon Editor by right-clicking the icon in the upper-right corner of the front panel or block diagram, and choosing Edit Icon.

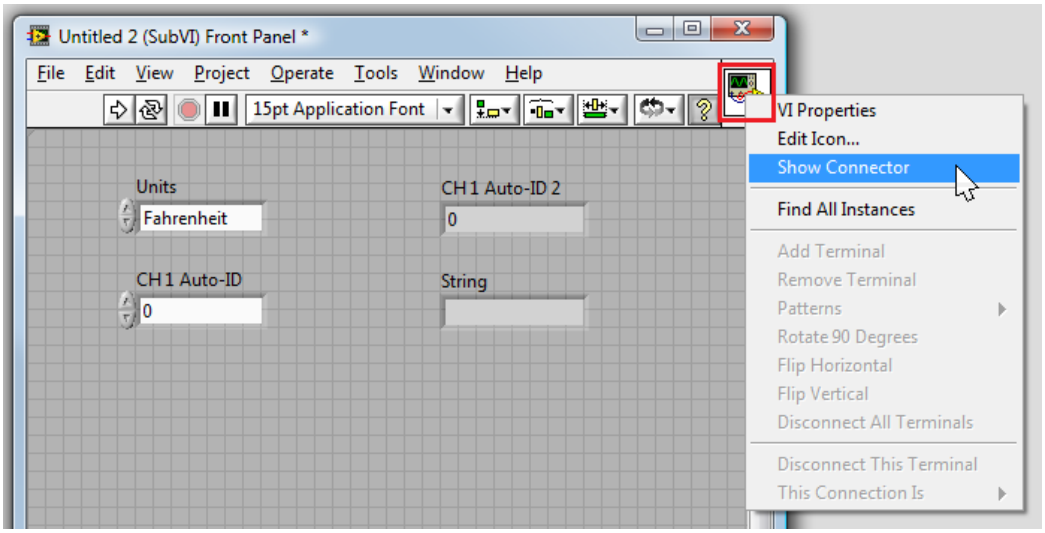


Once you open the Icon Editor, you have many tools for creating a custom icon or importing an image. It is important to think about the design of your icon so that it contributes to the overall understanding of how your program works. An icon designed with descriptive text may be the most useful. Graphics can be helpful, but be aware that they may also cause confusion if they are not meaningful.

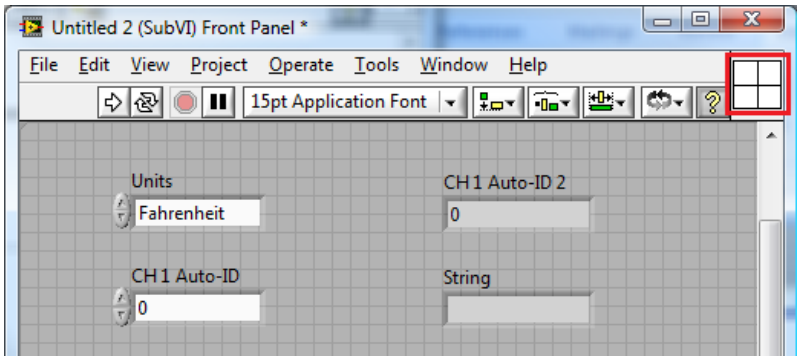
CONNECTOR PANE

The connector pane is a visual representation of the subVI terminals. The terminals correspond to the front panel controls and indicators of the VI. When a VI is run, you have direct access and control of the front panel controls and indicators. But when the VI is used as a subVI, the front panel controls and indicators are accessed without the use of a visible front panel. Instead, the connector pane of the subVI, linked to the front panel controls and indicators, allows wires to be connected as inputs and outputs. This allows data from the calling VI to be passed into and out of the subVI.

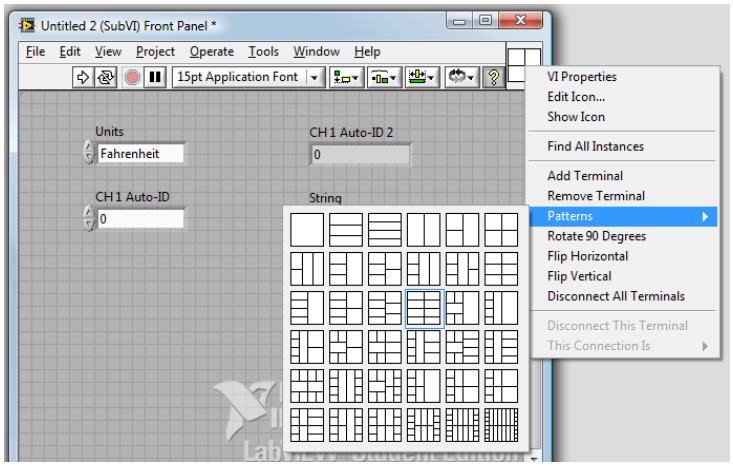
The connector pane is viewed and configured by right-clicking the icon in the upper-right corner of the front panel, and selecting Show Connector from the menu.



The icon will change to show the connector pane pattern. Each pattern has boxes that represent the location of terminals on the icon. You will use those boxes to assign inputs and outputs.



You can select a different pattern by right-clicking the connector pane, and selecting Patterns from the menu.

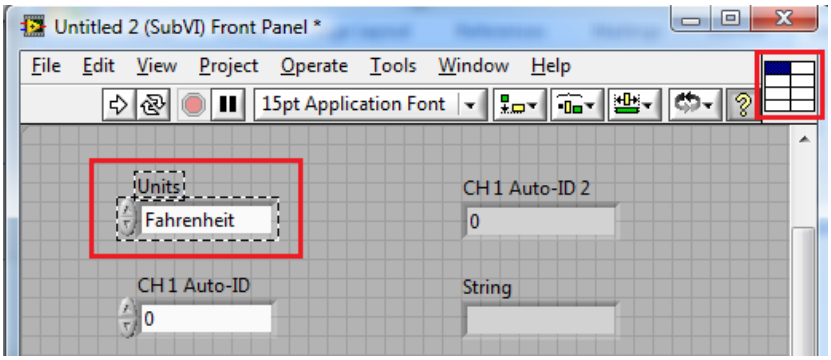


After you select a pattern to use for the connector pane, you assign a front panel control or indicator to a connector pane terminal (one of the boxes). It is generally good programming practice to organize the inputs to a subVI on the left and the outputs on the right.

To assign a terminal to a front panel control or indicator:

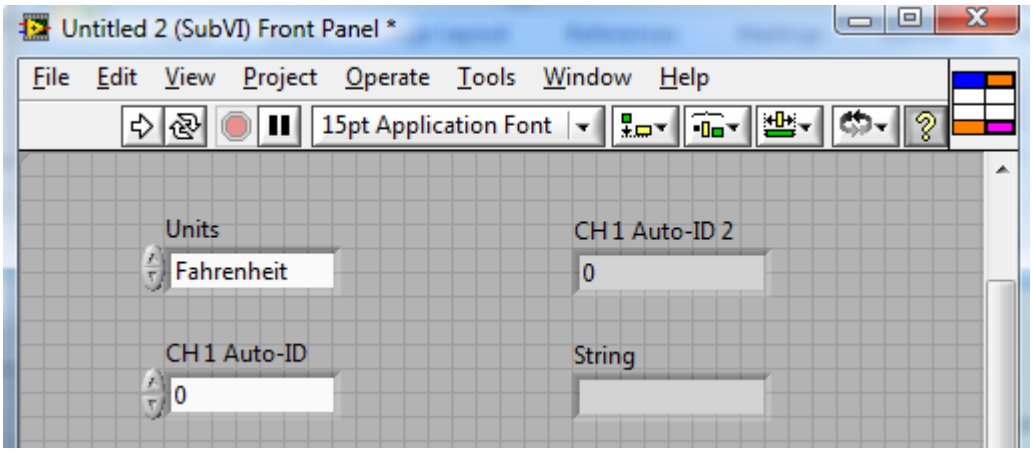
1. Click a terminal box of the connector pane.

2. Click the front panel control or indicator you want to assign to that terminal.

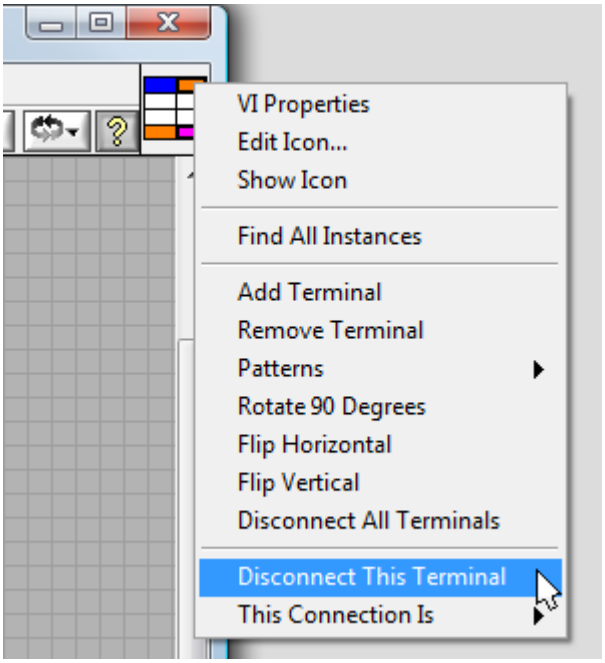


Note, that in this example, the original 4-terminal pattern would have been sufficient, but we chose the 8-terminal pattern to allow for greater separation between the terminal nodes on the subVI. Note, too, that the terminal color changes to that of the data type to which you have connected it. You can also select the control or indicator first and then select the terminal.

3. Repeat these steps for all the controls and indicators that you will use to pass data to and from the calling VI.

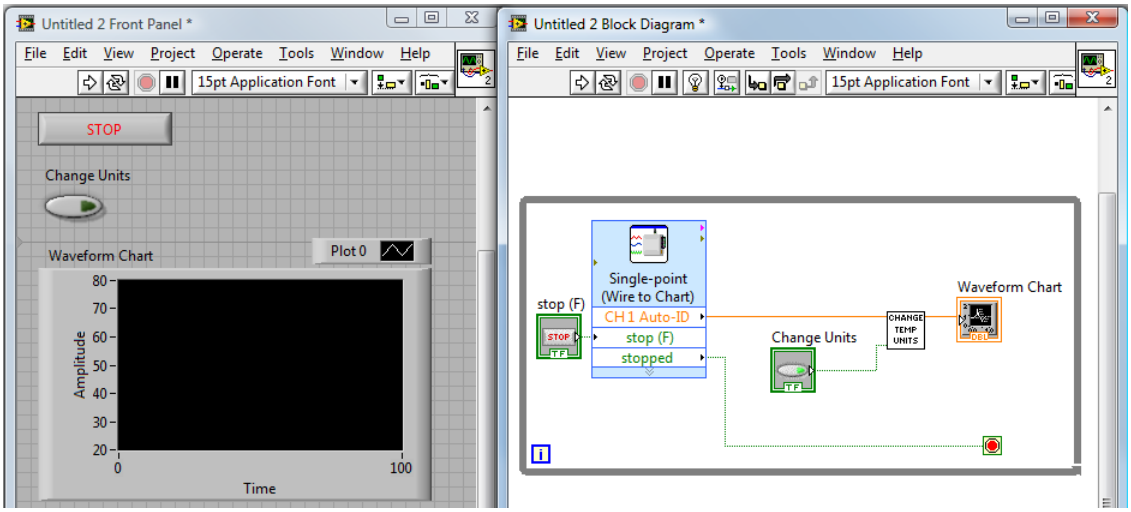


If you make a mistake, and need to modify the connector pane, simply right-click on the terminal that was mistakenly created and choose Disconnect This Terminal. In addition, you can choose the Disconnect All Terminals selection to start over.



Exercise 7

Create a Temperature Conversion SubVI



Completed front panel and block diagram

In this exercise, you will create a program using the Analog Express VI that measures temperature data. The program will allow you to display the data in Celsius or Fahrenheit units. The code that performs the selection of the units will be a separate VI, called by your program as a subVI.

OBJECTIVES

In this exercise, you will

- Create a LabVIEW VI to collect temperature data.
- Incorporate code that allows the user to view the data in Celsius or Fahrenheit units.
- Create a subVI to perform the temperature selection and conversion.

MATERIALS

SensorDAQ or LabQuest interface
Vernier Stainless Steel Temperature Probe
computer
LabVIEW
USB cable

PROCEDURE

Part I Connect Equipment

1. Connect the USB cable to the SensorDAQ or LabQuest interface.
2. Connect the other end of the USB cable to any available USB port on your computer. If you are using a LabQuest interface with a power button, turn it on.

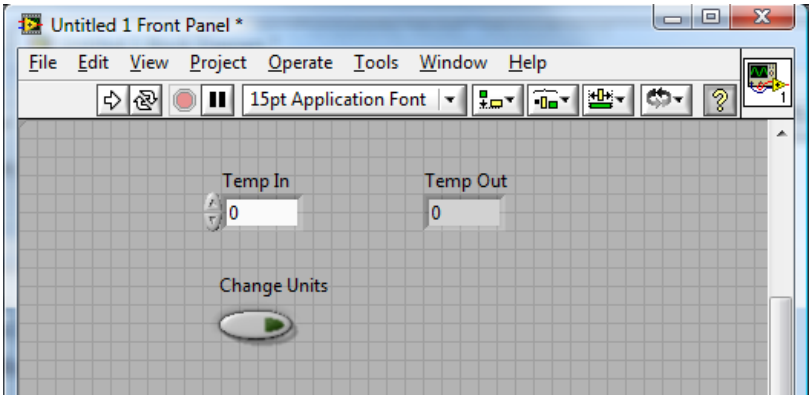
3. Connect the Temperature Probe to Ch. 1.

Part II Start LabVIEW and Create a VI to Convert Temperature Units

4. Start LabVIEW.

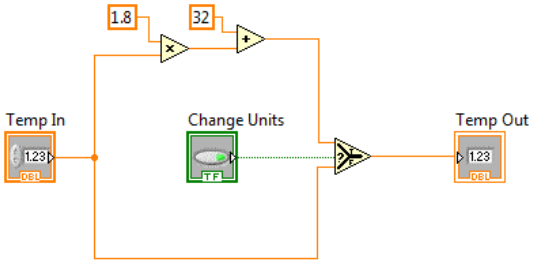
5. From the Getting Started window, click the "Blank VI" link under the New category.

6. On the front panel, add a Numeric Control and label it "Temp In", a Boolean Push Button labeled "Change Units", and a Numeric Indicator labeled "Temp Out". Organize your front panel with the controls on the left and indicators on the right.



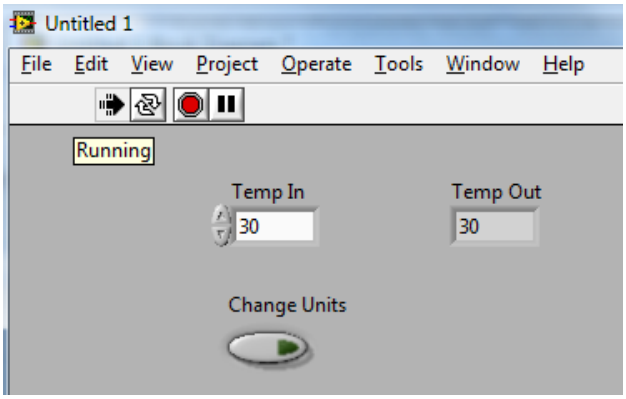
7. Go to the block diagram using <Ctrl-E>.

8. Build the block diagram as shown in the figure below. You will need to add the Select Function from the Comparison palette and the Add and Multiply Functions from the Numeric palette.

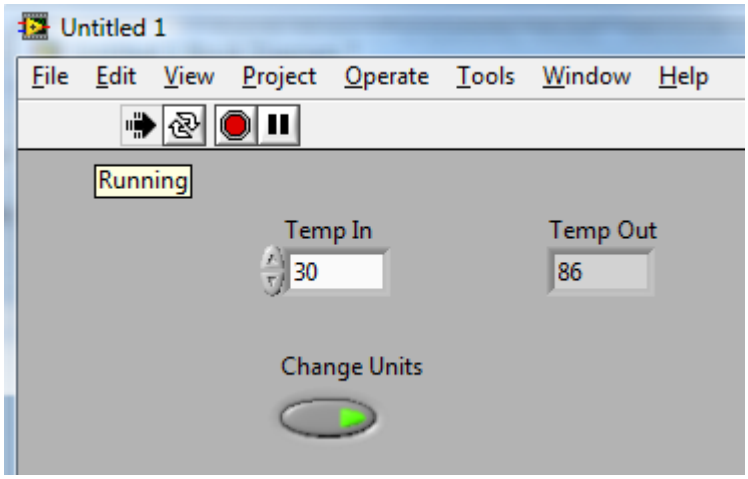


9. View the front panel using <Ctrl-E>.

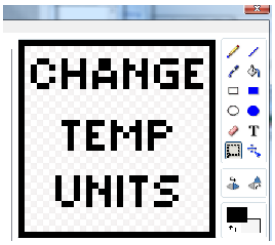
10. Type a value of 30 in the Temp In control and run the program.



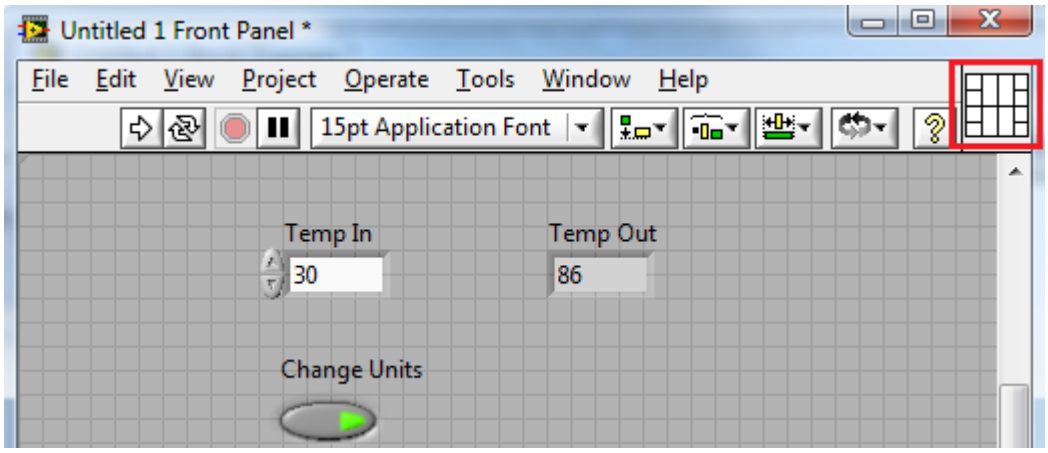
11. Click the Change Units control, and run the program again to verify that the VI is performing properly.



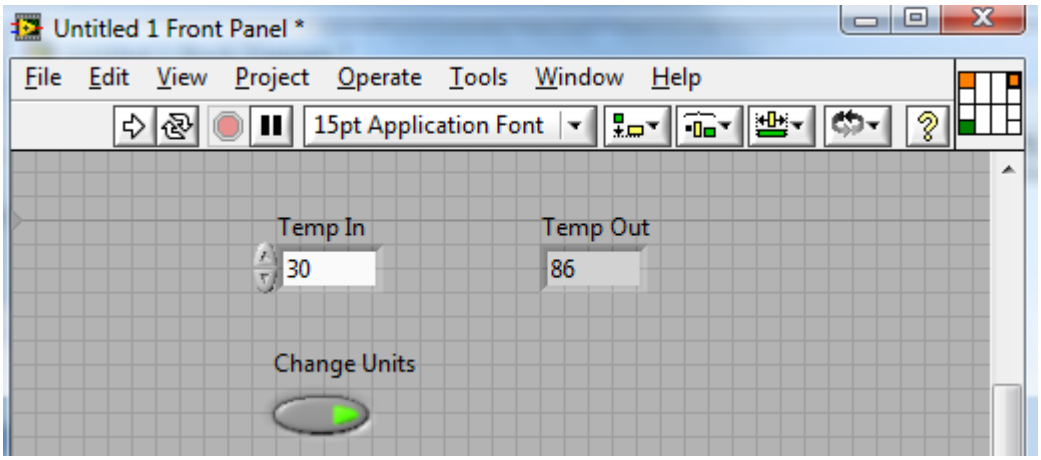
12. Open the icon editor by right-clicking the icon in the upper-right corner of the front panel and selecting Edit Icon. Create an appropriate icon and press the OK button to close the icon editor.



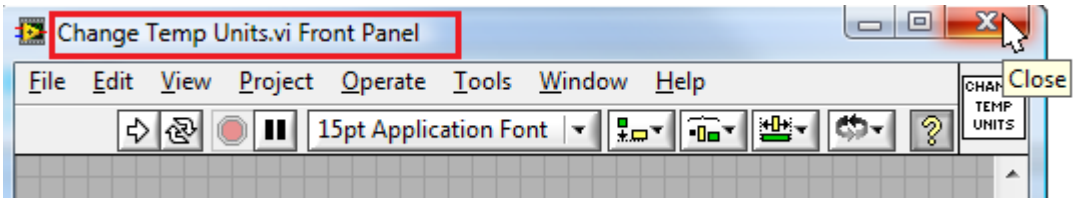
13. Show the connector pane by right-clicking the icon in the upper-right corner of the front panel and selecting Show Connector.


Untitled 1 Front Panel *
File Edit View Project Operate Tools Window Help
15pt Application Font
Temp In
30
Temp Out
86
Change Units

14. Create terminals for the two front panel controls and the indicator.
    a. Click the upper-left terminal box and click the Temp In control.
    b. Click the bottom-left terminal box and click the Change Units control.
    c. Click the upper-right terminal box and click the Temp Out indicator.



15. Save the VI with the name Change Temp Units and close this VI.

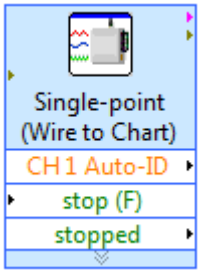


Part III Create a VI to Measure Temperature in Degrees Celsius or Fahrenheit

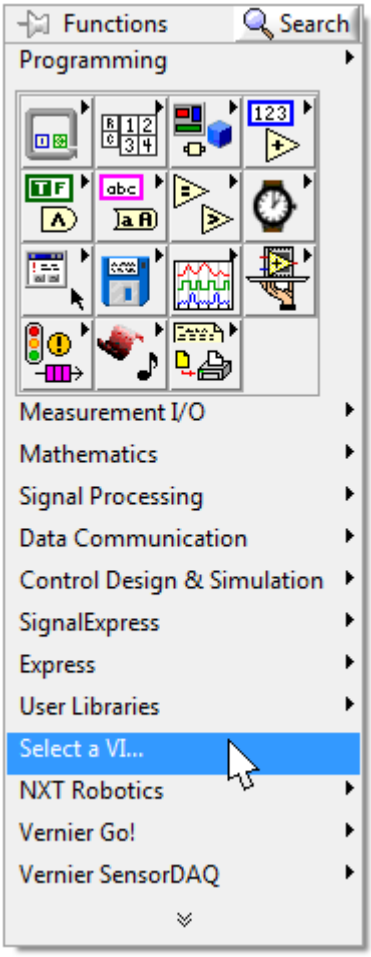
16. Open a New Blank VI.

17. View the block diagram workspace using <Ctrl-E>.

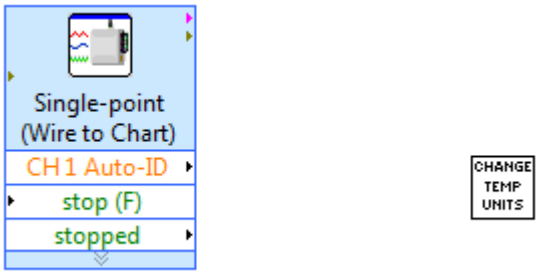
18. Place the Analog Express VI in the block diagram with a timing length of 20 seconds and sample rate of 2 samples/second.



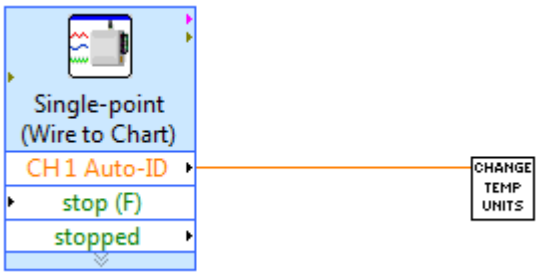
19. Place the Change Temp Units VI into your block diagram workspace. This VI can be found by going to the Functions palette and choosing Select a VI.



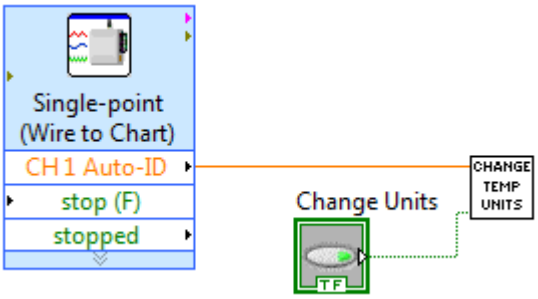
20. Navigate to the Change Temp Units VI and double-click to place it on the block diagram.



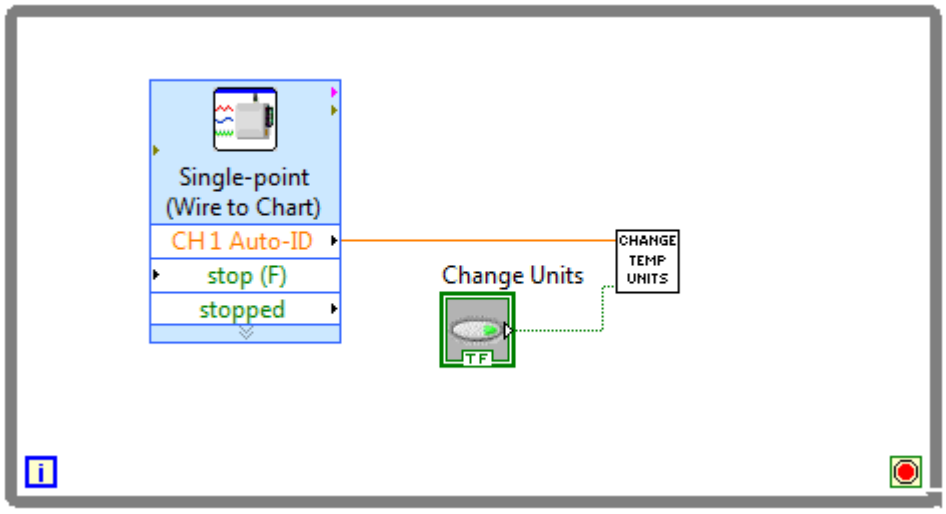
21. Wire the Express VI's CH 1 Auto-ID terminal to the subVI's Temp In terminal.



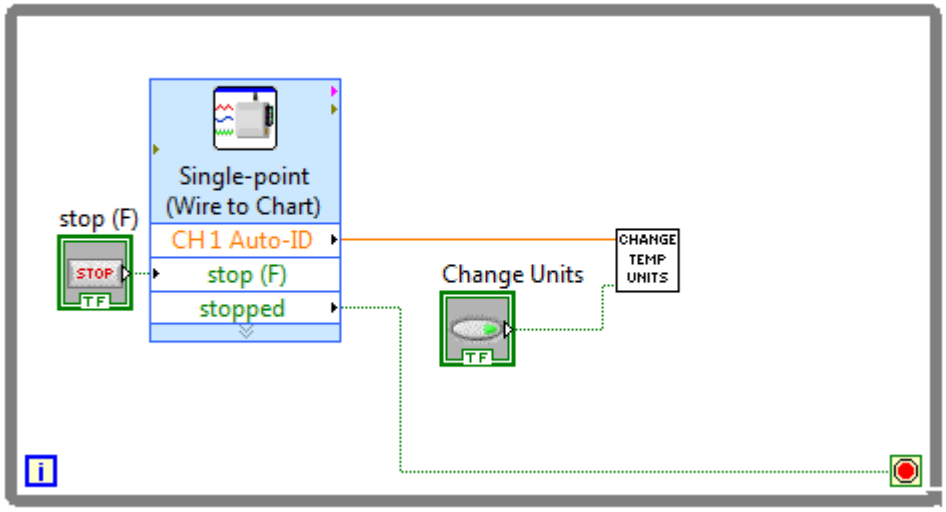
22. Right-click the subVI's Change Units input terminal and select Create ► Control from the menu.



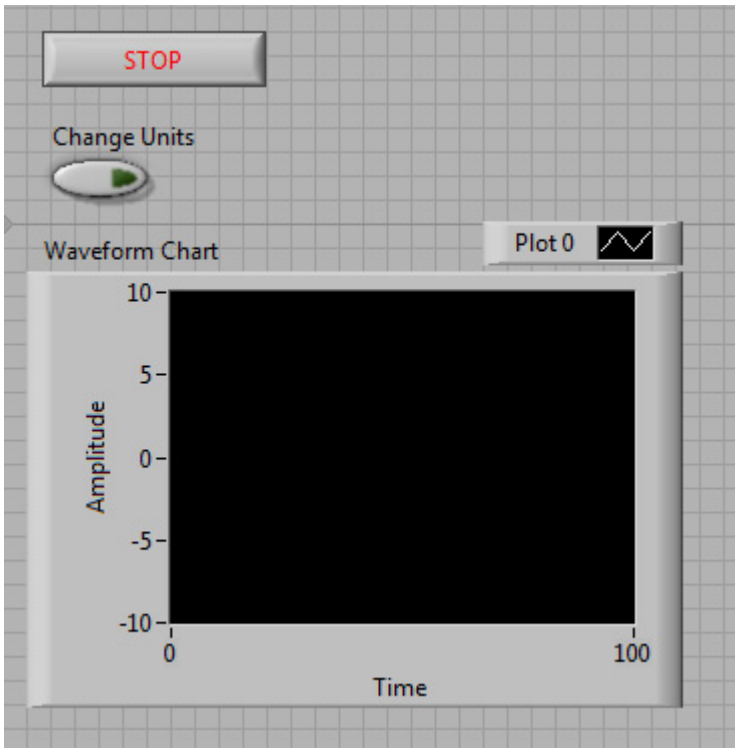
23. Place the code within a While Loop.



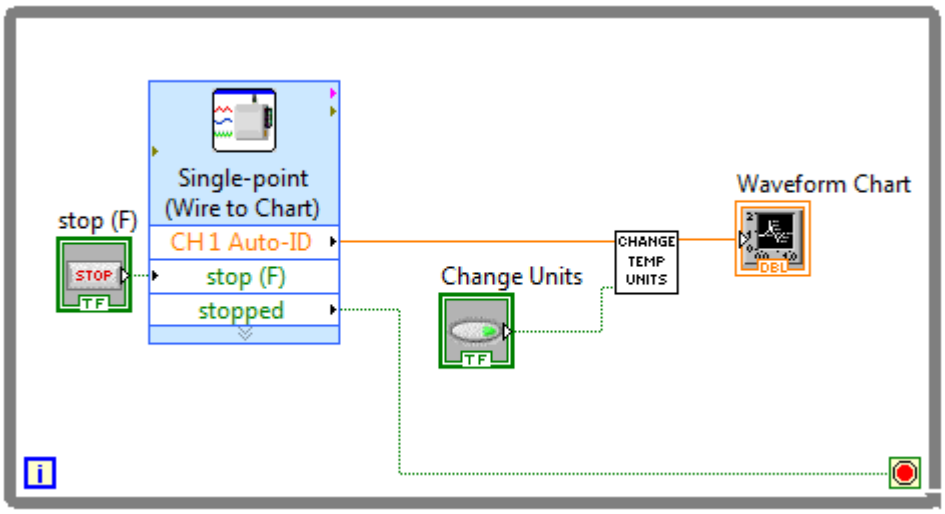
24. Wire the Analog Express VI's stopped output terminal to the While Loop's conditional terminal. In addition, create a STOP button control by right-clicking on the Express VI's "stop (F)" input terminal and selecting Create ► Control.



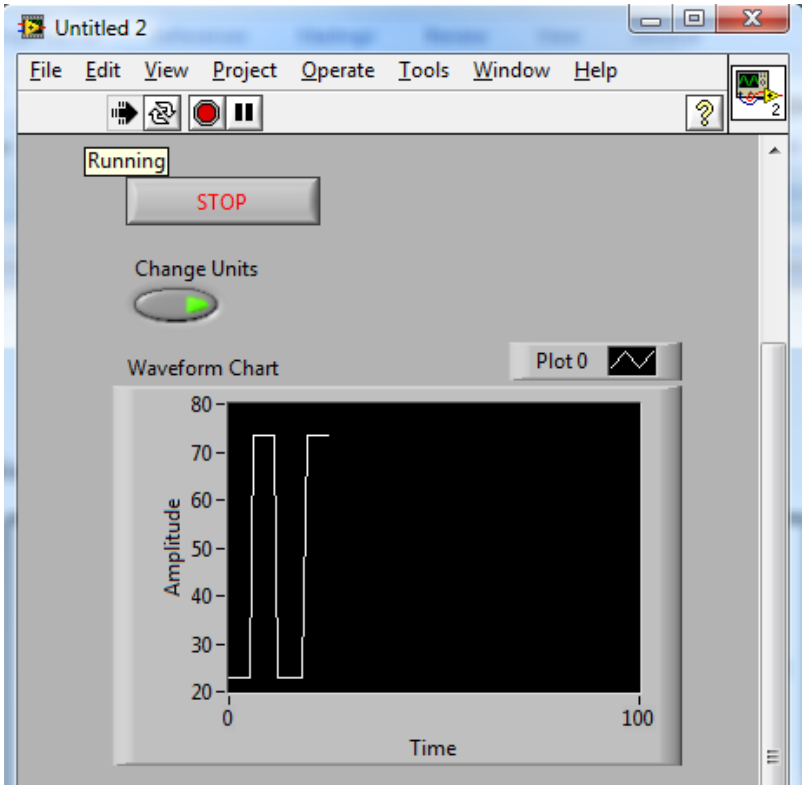
25. Go to the front panel and add a Chart.



26. Go to the block diagram and wire the subVI's Temp Out terminal to the Chart terminal.



27. Go to the front panel and run the VI. Click the Change Units control to change the units from Celsius to Fahrenheit.



Tip: If you are sharing this LabVIEW program to be opened, viewed, and run on a different computer, you must send both the program file and the subVI file.

EXTENSIONS

1. A chart's plot keeps a history of previous data points; therefore, when the units are changed, the previous data points do not change. Replace the chart with a thermometer. A thermometer does not keep a history, so the user interface will only display the most recent data point.

2. Modify the temperature conversion subVI to provide an option of selecting Kelvin units. Use an Enum control wired to a Case Structure. The Enum control should have three items with names that correspond to the three units. The Case Stucture should have a case for every value.

3. Use the second method to create a subVI inside your temperature conversion subVI. Highlight the Add and Multiply functions and their corresponding constants that are found inside the Fahrenheit case of the Case Structure (refer to Step 8 above), and then choose Create SubVI from the Edit menu. Name this new subVI "C to F" and create an appropriate icon.

Chapter 8

SensorDAQ Automation

SENSORDAQ TERMINAL[1]

SensorDAQ has three connectors for Vernier analog sensors and one connector for Vernier digital sensors. These connectors provide an easy way to use many sensors. For extra versatility, the SensorDAQ is equipped with a screw terminal connector. The connector has terminals for analog input, analog output, digital input/output, and a general purpose counter/timer. A 5-volt terminal and several ground terminals are also included.

By combining LabVIEW programming with the SensorDAQ input/outputs, the concept of automation may be introduced. This can include creating programs that turn on and off electronic devices, sensor feedback and control projects, or using PID control to manage motor speed.

The SensorDAQ is not designed to directly power external devices (the outputs only provide a few milliamps of current), but instead should be used as the controller. This usually means controlling relays, using an amplifier like the Vernier Power Amplifier, or designing a proper circuit.

A brief overview of the output terminals follows:

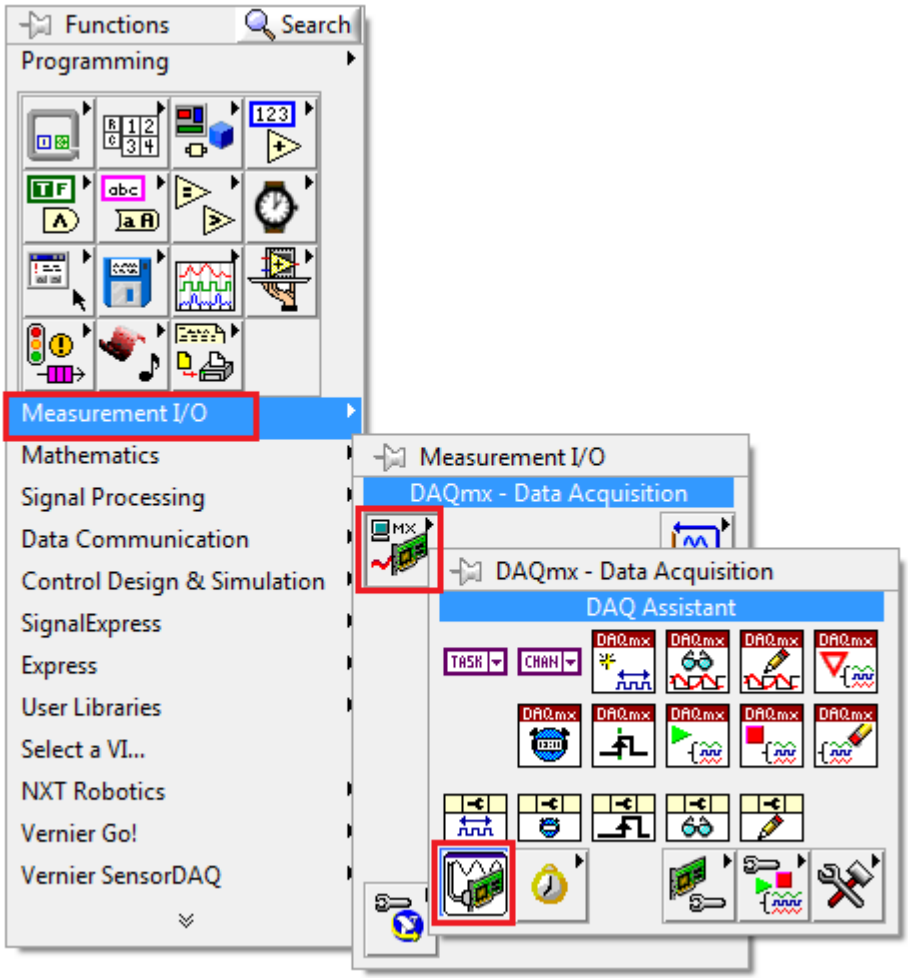
Terminal	Signal Name	Reference	Direction	Description
5,8,10	GND	—	—	**Ground**: Reference point for single-ended AI measurements, bias current return point for differential mode measurements, AO voltages, digital signals at the I/O connector, +5 VDC supply, and the +2.5 VDC reference.
11,12	AI <0..1>	Varies	Input	**Analog Input Channels 0 and 1**: For single-ended measurements, each signal is an analog input voltage channel. For differential measurements, AI 0 and AI 1 are the positive and negative inputs respectfully, of differential analog input channel 0.
9	AO 0	GND	Output	**Analog Output Channel 0**: Supplies the voltage output of AO channel 0 from 0–5 V with an output current drive value of 5 mA. The maximum update rate is 150 Hz, software timed.
1–4	P0.<0.3	GND	Input or Output	**Digital I/O Signals**: You can individually configure each signal as an input or output.
6	+5 V	GND	Output	**+5 V Power Source**: Provides +5 V power.
7	PFI 0	GND	Input	**PFI 0**: This pin is configurable as either a digital trigger, an event counter input, pulse generation output, or as a period, semi-period, two edge separation timer.

[1] This chapter applies only to the Vernier SensorDAQ interface. If you are using a LabQuest or LabQuest Mini, skip this chapter.

DAQ ASSISTANT

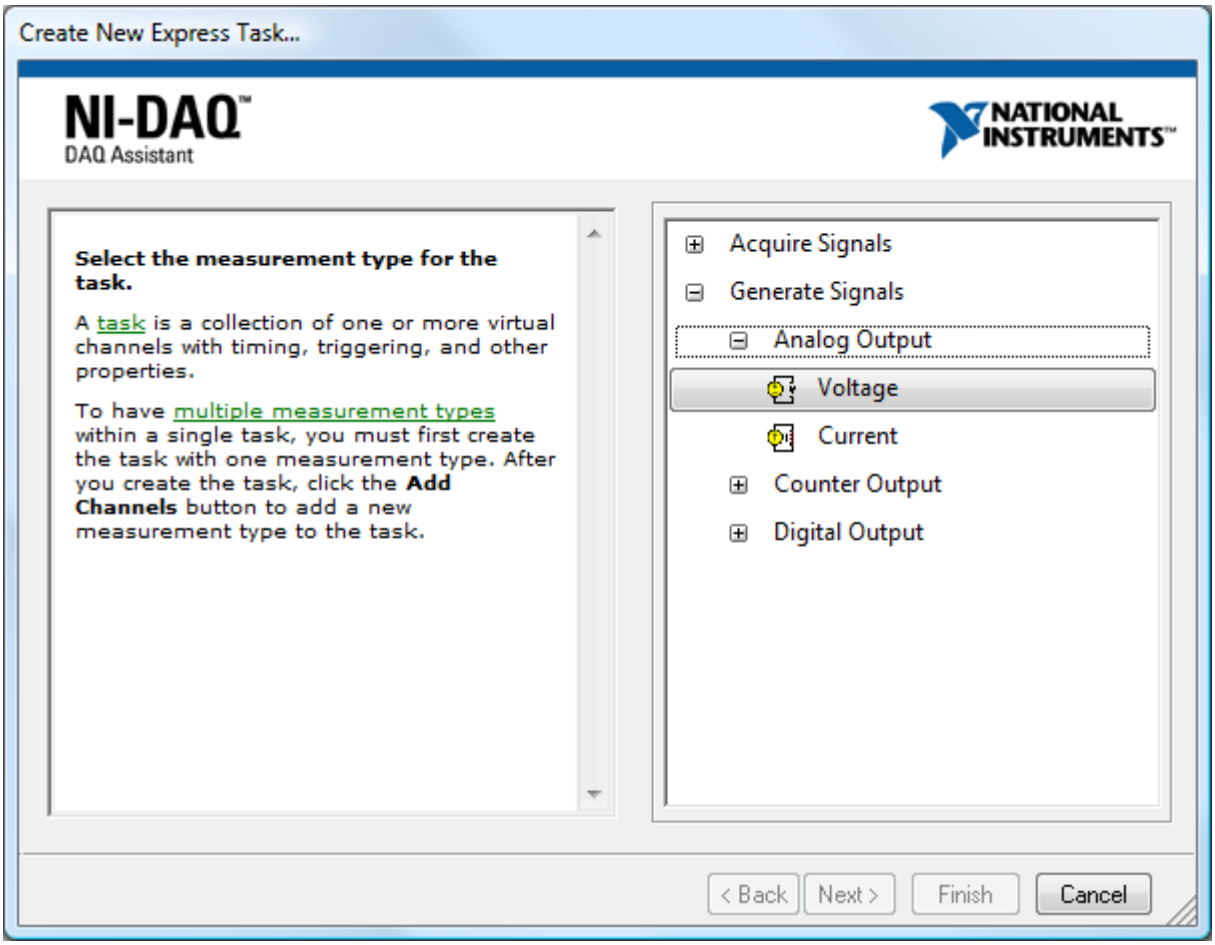
The Analog Express VI was used in the previous exercises to communicate and program the SensorDAQ analog channels. The Express VI for the screw terminal is called DAQ Assistant. This Express VI is used to configure and program National Instruments hardware. The DAQ Assistant Express VI can be used to configure and program the SensorDAQ screw terminals, but it cannot configure the Vernier analog or digital sensor channels.

The DAQ Assistant is located in the Measurement I/O ► DAQmx – Data Acquisition palette.

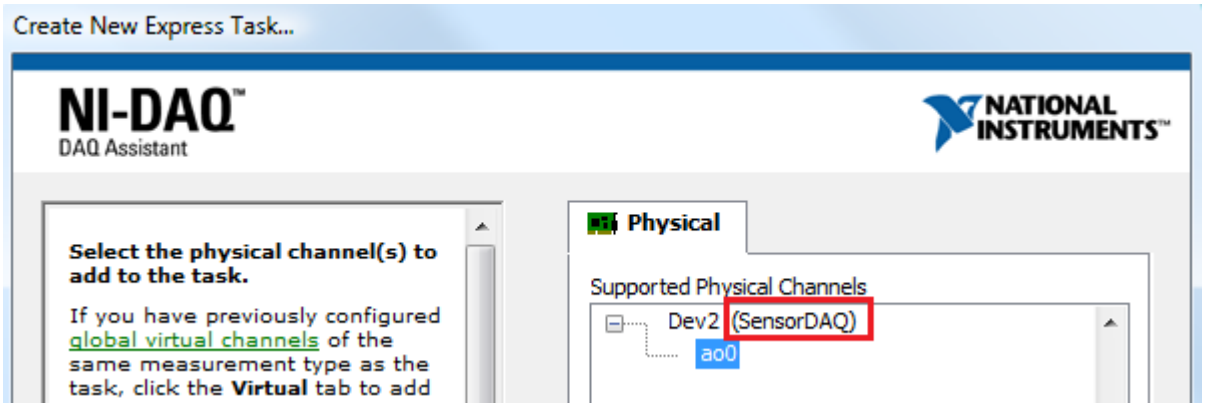


When you place the DAQ Assistant in the block diagram workspace, a configuration dialog box appears. Use the tree control in this popup to select the type of tasks you want to create. The DAQ Assistant is an Express VI designed for many pieces of hardware. Realize that some selections will not work with the SensorDAQ. For example, in the dialog window there is a task

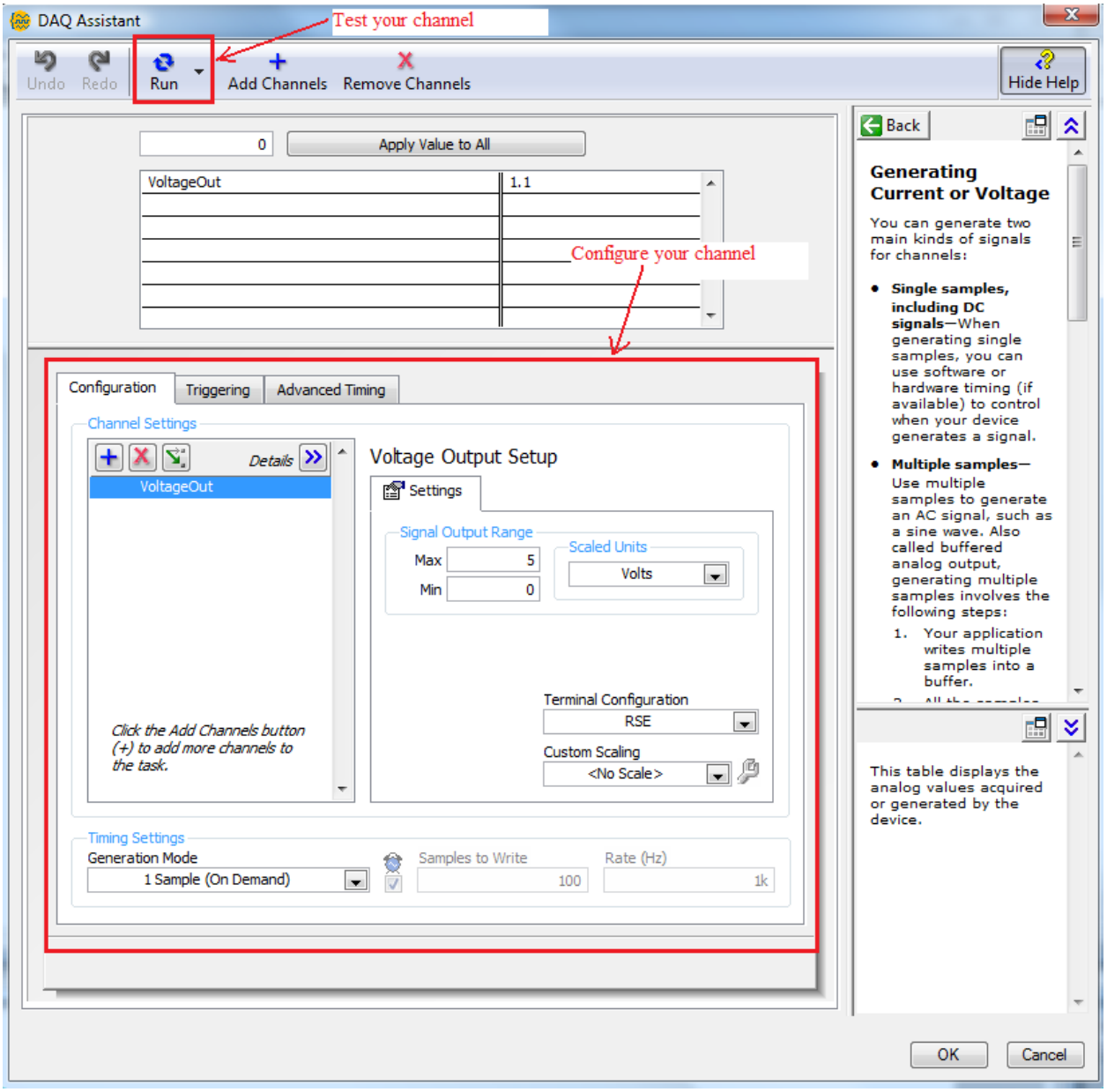
for Analog Output ► Current. The SensorDAQ analog output line does not support this and it is not a valid selection. If you choose this, you will receive an error message. You must choose Analog Output ► Voltage.



After selecting the type of task, the next step is to choose the channel. Make sure that the device is SensorDAQ, and then click the channel name, such as ao0, to highlight it and continue.

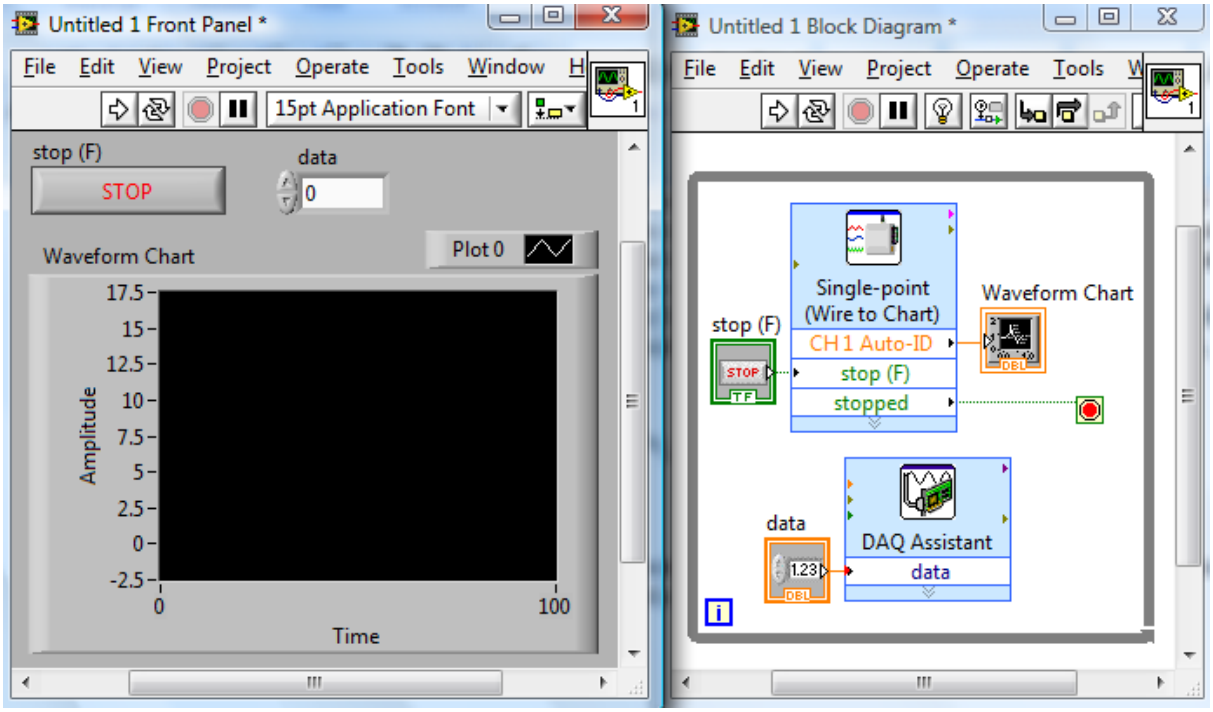


The DAQ Assistant channel configuration and test dialog window now appears. Configure your channel, as needed, in the lower half of the dialog box. You can test your channel by running the test panel in the upper portion of the dialog box. Again, there may be configuration options that are not valid for the SensorDAQ screw terminal. If you choose an invalid option, you will get a warning message.



Exercise
8

Control Analog Out, Digital Out, and Pulse Out



Completed front panel and block diagram

In this exercise, you will create a program to control the SensorDAQ's analog out terminal (pin 9 of the screw terminal connector).[1] In addition, a Voltage Probe will be used to read the voltage of the analog out terminal. The Analog Express VI is a simple way to program data collection from a Vernier sensor. The DAQ Assistant Express VI is a simple way to control the SensorDAQ's screw terminal channels. Steps will be taken to modify the VI to control the digital-out line and the counter line.

OBJECTIVES

In this exercise, you will

- Create a LabVIEW VI to control and measure the screw terminal analog-out line.
- Modify the VI to control and measure one of the four digital-out lines.
- Modify the VI to control and measure a pulse train output from the counter line.
- Learn to use DAQ Assistant.

[1] This exercise applies specifically to the Vernier SensorDAQ interface. If you are using a LabQuest interface, skip this exercise.

MATERIALS

SensorDAQ
USB cable
computer
LabVIEW
Vernier Voltage Probe
small slotted screwdriver
short wire stripped of electrical insulation on both ends

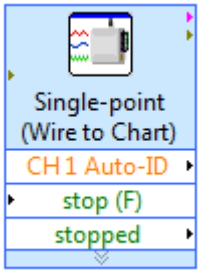
PROCEDURE

Part I Connect Equipment to Perform Analog Output

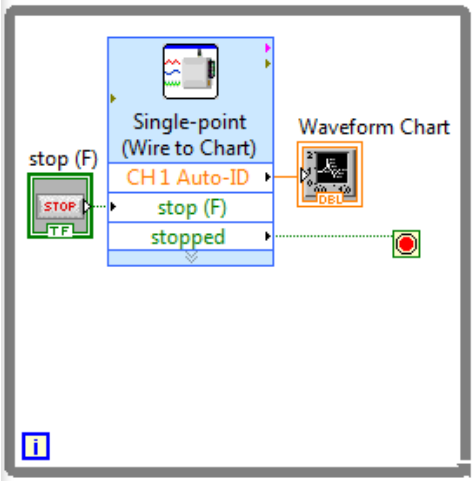
1. Connect the USB cable to the SensorDAQ.
2. Connect the other end of the USB cable to any available USB port on your computer. The green LED on the SensorDAQ (next to the USB cable port) should be blinking.
3. Connect the Voltage Probe to Ch. 1.
4. Connect two short wire segments to SensorDAQ screw terminal AO (pin 9) and GND (pin 10).
5. Clip the red lead of the Voltage Probe to the pin 9 wire and the black lead to the pin 10 wire.

Part II Start LabVIEW and Create a VI to Control and Measure Analog Output

6. Start LabVIEW.
7. In the Getting Started window, click the Blank VI link in the New category.
8. View the block diagram using <Ctrl-E>.
9. Place the Analog Express VI in the block diagram. Leave the sample rate and experiment length at the defaults of 10 samples/second and 10 seconds, respectively.

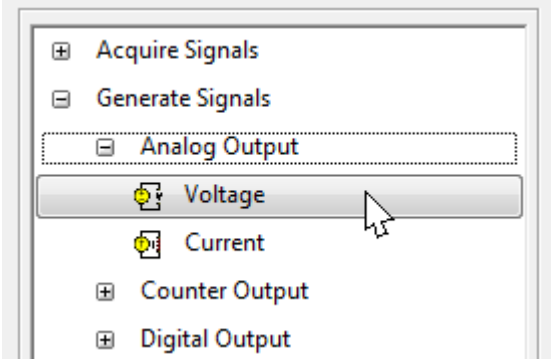


10. Place the Express VI within a While Loop. Create a STOP button control and a Waveform Chart indicator, and wire them appropriately to the Analog Express VI. Wire the Express VI's "stopped" terminal to the While Loop's conditional terminal.

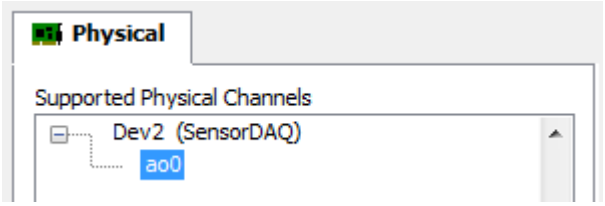


11. Place the DAQ Assistant in the While Loop. The DAQ Assistant is found in the Measurement I/O ► DAQmx – Data Acquisition palette.

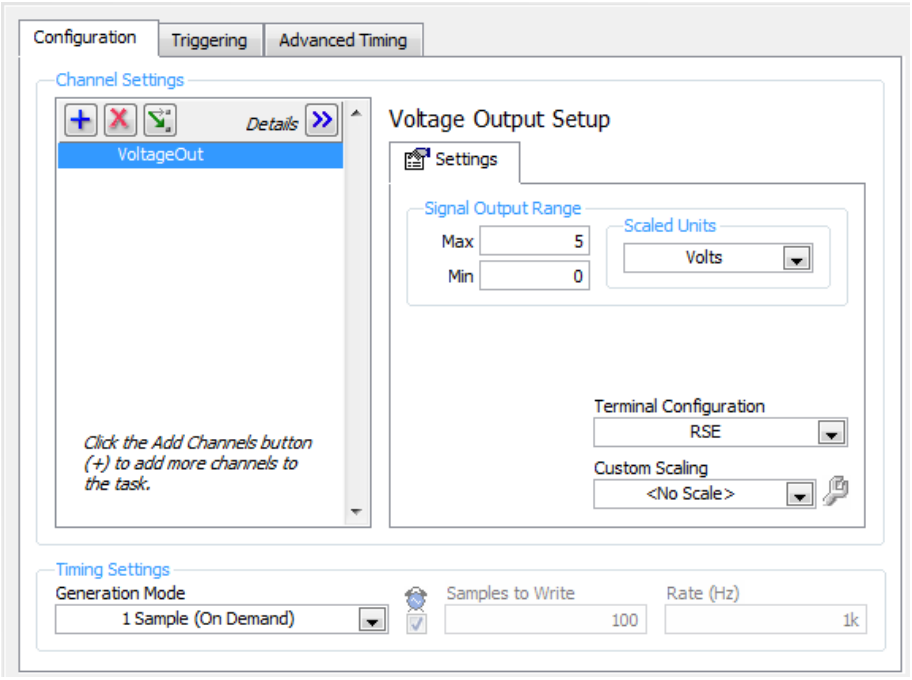
12. Configure the DAQ Assistant for Generate Signal ► Analog Output ► Voltage.



13. SensorDAQ has one analog-output channel on the screw terminal called ao0. Select this channel by clicking on it. Click Finish to close this dialog box.

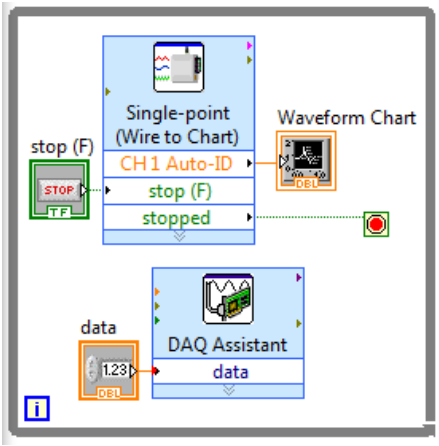


14. Keep the default configuration values: Signal Output Range of 0 to 5 Volts, Terminal Configuration of RSE, Custom Scaling of <No Scale>, and Generation Mode of 1 sample (On Demand).



15. Click OK to close the DAQ Assistant's Configuration window.

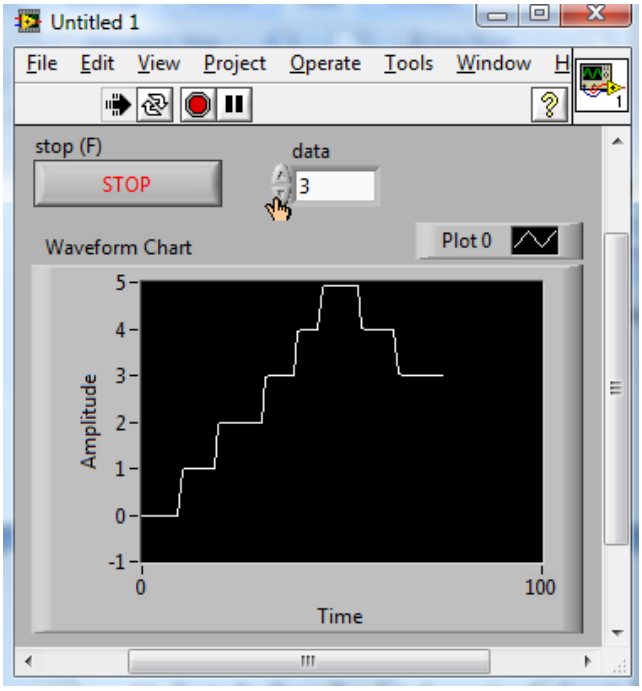
16. Create a control to control the analog out value (right-click the DAQ Assistant's data input terminal and select Create ► Control).



17. Go to the front panel and organize the location of the "data" control, the STOP button, and the Chart.

Tip: To avoid an error message during operation, set the minimum and maximum values for the "data" control by right-clicking the control, selecting Properties, and clicking the Data Entry tab. The minimum should be set at 0, the maximum at 5.

18. Run the VI and change the "data" control's value to a value between 0 and 5. The Chart is reading the Voltage Probe and should verify the voltage output from the analog output line.



Part III Modify the Equipment Connections to Perform Digital Output

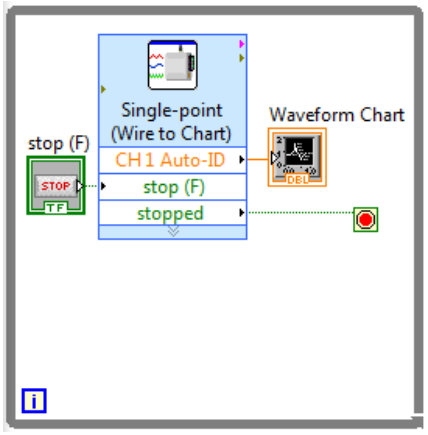
19. Connect two wires to SensorDAQ's digital I/O screw terminal PO0 (pin 1) and GND (pin 10).

20. Clip the Voltage Probe red lead to the pin 1 wire and the black lead to the pin 10 wire.

Part IV Modify the VI to Control and Measure Digital Output

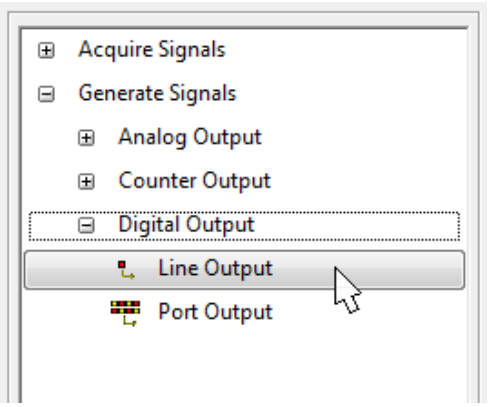
21. View the block diagram using <Ctrl-E>.

22. Highlight and delete the DAQ Assistant and the data control.

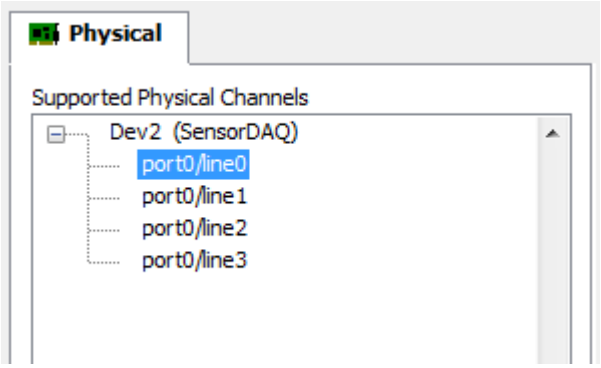


23. Place a new DAQ Assistant in the While Loop (the DAQ Assistant is found in the Measurement I/O ► DAQmx – Data Acquisition palette).

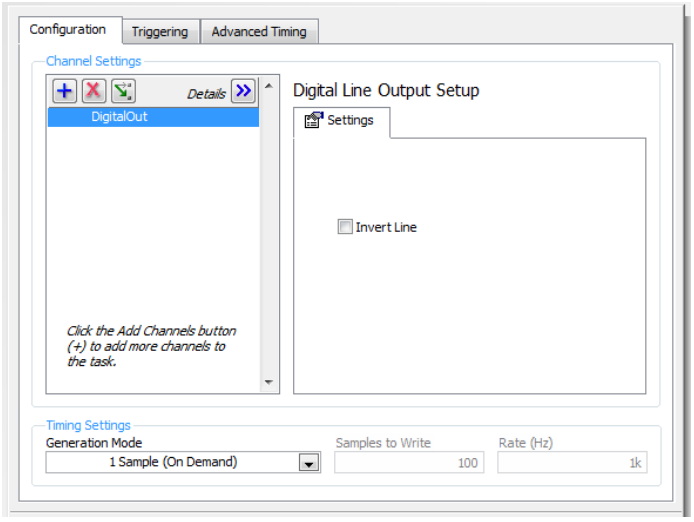
24. Configure the DAQ Assistant for Generate Signals ► Digital Output ► Line Output.



25. SensorDAQ has four digital output channels on the screw terminal. Select port0/line0 from the list by clicking it. Click Finish to close this dialog box.

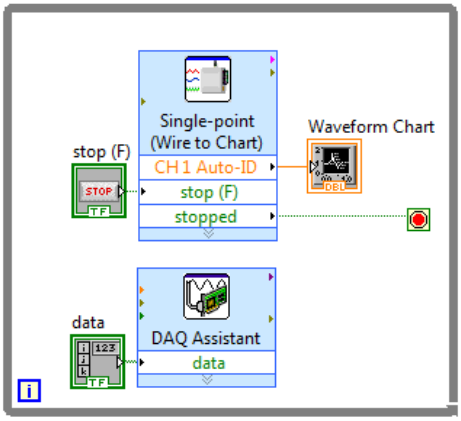


26. Keep the default configuration values.

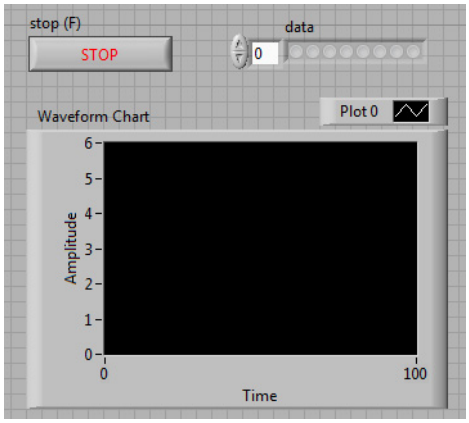


27. Click OK to close the DAQ Assistant's Configuration Popup.

28. Create a control to control the digital out value by right-clicking on the DAQ Assistant's data input terminal and selecting Create ► Control.



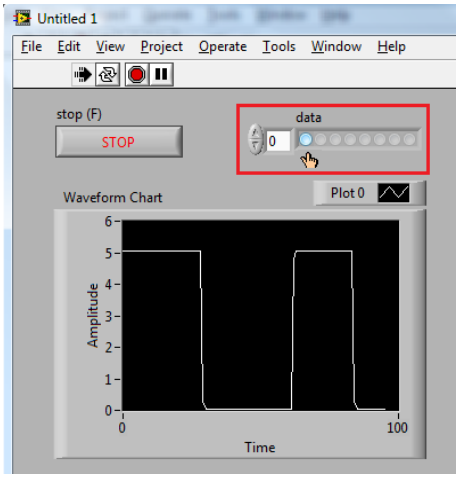
29. View the front panel using <Ctrl-E>. Arrange the controls and clear the chart. The chart can be cleared by right-clicking on the chart and selecting Data Operations ► Clear Chart.



30. The data control for controlling the digital output line is an array of Boolean values. The Boolean elements are dimmed, indicating that they are uninitialized. Initialize the first element by clicking the element; it should turn blue. The Boolean value is now initialized with a True value. All of the other elements are not necessary for controlling the first digital line. The other elements are left uninitialized, making this an array of one element.

Tip: An array includes elements that are values or data (in this case, the data are Boolean) and an index display. The index display (the number and the increment/decrement arrow button) is to the left of the elements. If you have an array with many elements, the index display allows you to view the elements that are not readily visible. For example, to view element 56 you would enter that number in the index display. Do not confuse the index display as a way to change the values of the elements.

31. Run the VI and change the first Boolean element in the data array from True to False a few times. The digital line is controlled by this element, and the voltage measurement on the chart should reflect the line turning on (5 volts) and off (0 volts).



Part V Modify the Connections to Perform Pulse Output

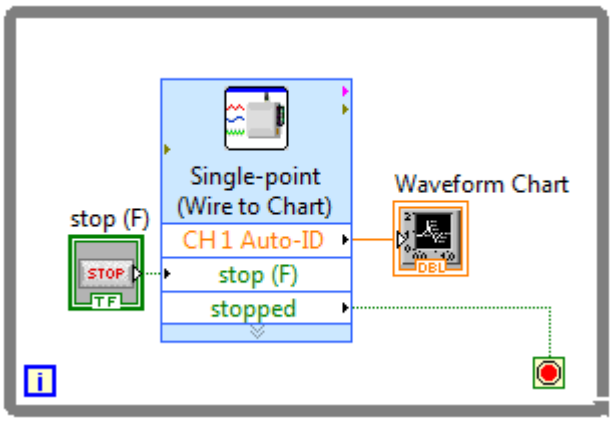
32. Connect wires to SensorDAQ's counter screw terminal PFI 0 (pin 7) and GND (pin 10).

33. Clip the Voltage Probe red lead to the pin 7 wire and the black lead to the pin 10 wire.

Part VI Modify the VI to Control and Measure Pulse Train Output

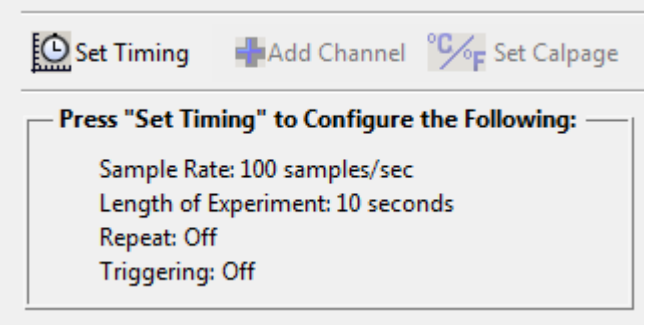
34. Go to the block diagram using <Ctrl-E>.

35. Highlight and delete the DAQ Assistant and the data control, and resize the While Loop to contain the STOP control, the Analog Express VI, and the Chart.



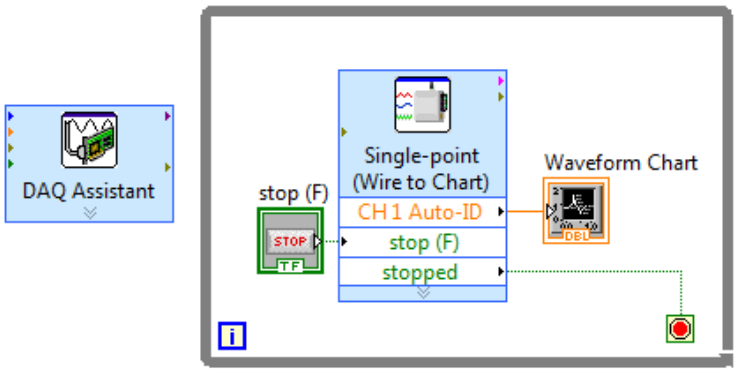
36. Open the configuration dialog of the Analog Express VI by double-clicking on it.

37. Modify the timing for a sample rate of 100 samples/second for 10 seconds.

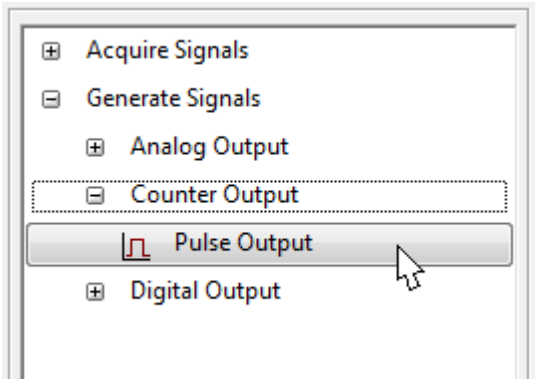


38. Place a new DAQ Assistant to the left of the While Loop (the DAQ Assistant is found in the Measurement I/O » DAQmx – Data Acquisition palette).

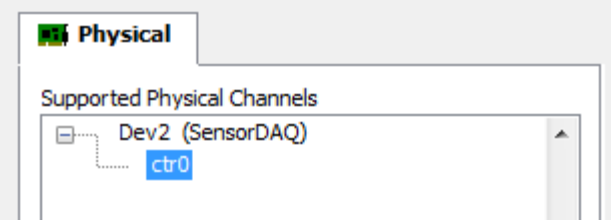
Tip: The DAQ Assistant is not wired to anything; it is floating outside the While loop. This is because we want to start the pulse output, and start monitoring the voltage probe reading at the same time. The DAQ Assistant is not placed inside the While loop because we only want to start pulse output one time. In addition, the design of the Analog Express VI does not allow you to start data collection and then start pulse output.



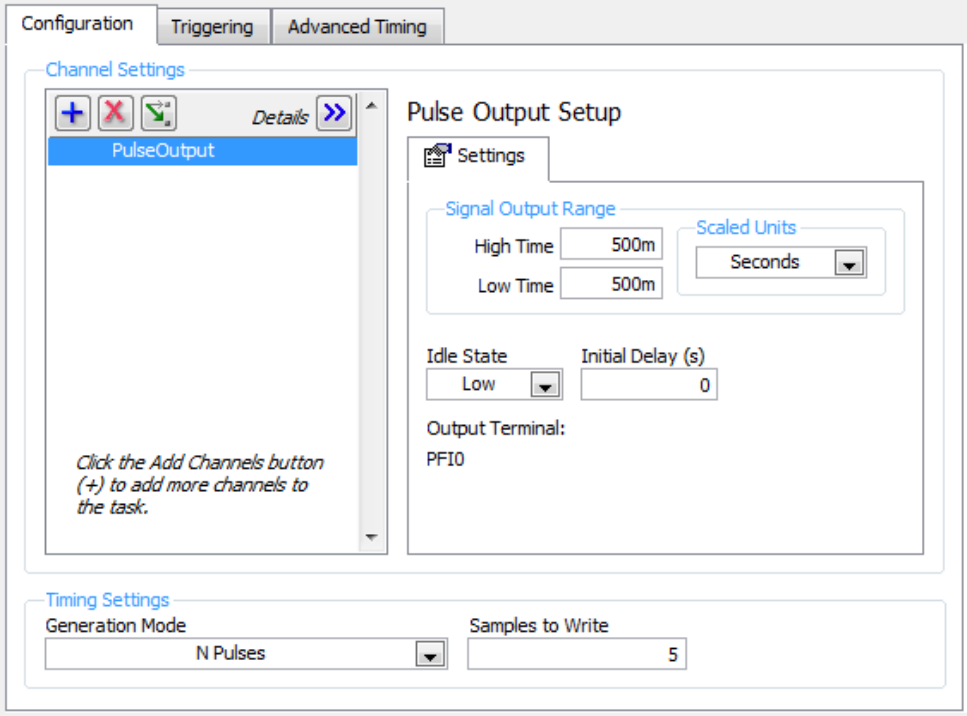
39. Configure the DAQ Assistant for Generate Signals ► Counter Output ► Pulse Output.



40. SensorDAQ has one counter/timer channel on the screw terminal, called ctr0. Select this channel by clicking it. Click Finish to close this dialog box.



41. Modify the pulse output configuration with a high time of 500 milliseconds, a low time of 500 milliseconds, generation mode of N Pulses, and with 5 Samples to Write. Click OK to close the DAQ Assistant's Configuration window.

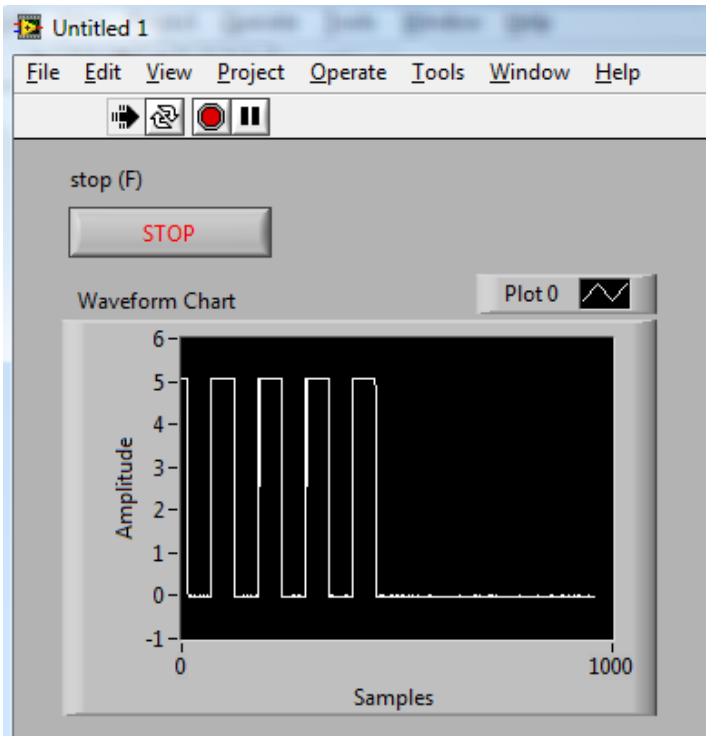


42. Go to the front panel.

43. Change the x-axis on the chart to "Samples" and a maximum x scale value of 1000.

44. Clear the chart's previous plot (right-click on the chart and select Data Operations ► Clear Chart).

45. Run the VI. The voltage measurement of the 5 pulses should show on the chart's plot line.



EXTENSIONS

1. Write a program to automate analog output. The voltage should increase by 0.1 volts, every second, for 50 seconds.

2. Write a program to produce a sine wave voltage output pattern with a frequency of 1 Hz and an amplitude of 5 volts. Consider using the Simulate Signal Express VI to generate the output values. This Express VI will create an array of all of the values, but you will need to feed them into the DAQ Assistant one value at a time.

3. Write a program to turn on digital output line 1 if a sensor reading is above a user-defined threshold, and turn on digital output line 2 if the sensor reading is below the threshold.

Projects: Build a Sensor Control Program

Learning how to use LabVIEW is best done by actually using LabVIEW. Reading or hearing lectures can be helpful at first, but actually starting a LabVIEW project from scratch, designing a front panel and block diagram, and figuring out the steps and logic required to perform a task is how you will really learn LabVIEW.

Below are two projects. Read through the description and design requirements of each project, and then choose one project to work on. If you have time, do both.

PROJECT 1 – AUDIO FEEDBACK

The LabVIEW front panel is used to design a user interface that provides visual feedback in the form of charts and graphs, meters, thermometers, flashing LEDs, and other indicators. Audio feedback is an alternative method that may, in some circumstances, be preferred. Examples of audio feedback include smoke alarms that emit an audible tone when smoke is detected, stoves that beep when a pre-defined temperature is reached, and manufacturing stations that provide an emergency sound when corrective action is required. Often, it is not important to know the exact value of a measurement, but only an approximation or a change in the value.

DESIGN REQUIREMENTS

In this project, you will create a program that provides an audio signal as feedback of temperature. You will use a Temperature Probe to measure the temperature in degrees Celsius. The sound should be created by your computer, with the frequency of the tone directly proportional to the temperature reading.

MATERIALS

SensorDAQ or LabQuest interface
Vernier Stainless Steel Temperature Probe
computer
LabVIEW
USB cable
beaker of warm tap water

TIPS

1. The LabVIEW functions palette includes a Graphics and Sound subpalette. Inside this subpalette is a subVI called Beep.vi. Use the Beep subVI as the means to making sound with your computer.

2. A good way to start is to build a simple program to get to know how this subVI works. Place it within a While Loop and control the frequency. Create constants for the “duration” and “use system alert?” input terminals. The “use system alert?” terminal must be set to False so that the sound is based on your frequency value, rather than the system alert sound. Use this VI to determine what upper and lower level frequency values are appropriate in terms of audible feedback to the user.

PROJECT 2 – CELL PHONE DECODER

Cell phones continue to add features that increase their importance to people throughout the world. Communicating with family, friends, and business associates are common uses, and providing a means for emergency contact may be the most important feature of all. Cell phones are also capable of wireless internet access, sending and receiving photos and files, GPS receiver, MP3 player, and even sensor feedback. But the original feature of a cell phone, as well as a land-line phone, was the ability to transmit numbers, symbols, and letters to the telephone company with the push of a button.

Dual Tone Multi-Frequency, or DTMF, is a communication method developed by engineers with the trademark name "touch-tone," and allows the telephone company to know what numbers you dial, issue commands to a switching system, or enable the caller to communicate with a computer system using the numbered keypad.

The DTMF system uses eight different frequency signals transmitted in pairs. Each time you press a key, two tones are produced. For example, when you press "5," your phone produces a 770 Hz sinusoidal signal and a 1336 Hz sinusoidal signal. The following table shows how the DTMF keypad is laid out in a 4×4 matrix, with each row representing a low frequency, and each column representing a high frequency.

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

In this project, your challenge is to create a program that is able to decode a DTMF signal. If you accomplish this goal, your program should be able to tell you what button was pressed on a cell phone out of your sight by decoding the waveform, as the telephone company does.

DESIGN REQUIREMENTS

In this project, you will create a Dual Tone Multi-Frequency (DTMF) decoder for a cell phone or tone dialer. If these are not available, a third option is to run the Cell Phone.vi program (found on the accompanying CD), and use the speakers of your computer to generate the tones. You will use a Microphone to measure the waveform of sound as a user presses the buttons from 1 to 9. Your program should display the value of the selected number for as long as the phone, tone dialer, or computer is producing a noise. The number should no longer be displayed when the noise from the button press ends.

MATERIALS

SensorDAQ or LabQuest interface
USB cable
computer
LabVIEW
Vernier Microphone
cell phone, tone dialer, or computer running Cell Phone.vi

TIPS

1. Consider creating your program in steps. The first step is to continuously read the microphone data at a fast sampling rate. The next step is to provide waveform analysis to determine the low frequency and the high frequency within the signal. The final step is to provide decision making within your program to display a number that corresponds to the low and high frequency.

2. Consider the Tone Measurements Express VI as a way to search a specified frequency range to find the single tone with the highest amplitude.

3. Express VIs typically provide results that are passed as Dynamic Data. You may have to convert this to a numeric data type. To do this, you will need to use the Convert From Dynamic Data Express VI. Choose the Single Scalar option from the configuration dialog box.

4. Your frequency measurements are not likely to match exactly the numbers shown in the table above. Case structures allow you to specify a range of values for comparisons (for example, rather than specifying a value of 697, use 685..715 instead).

Projects: Build a Sensor Control Program

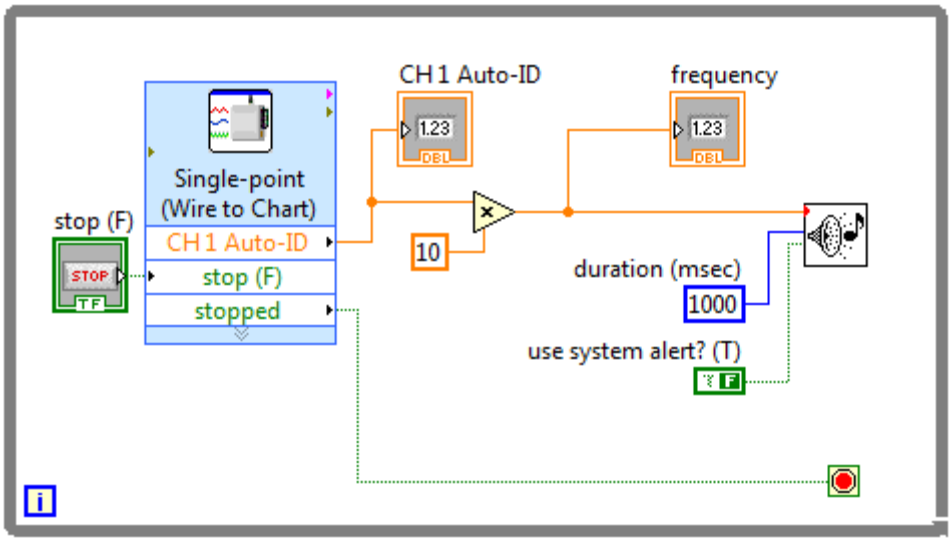
The project section of this book includes two separate challenges. These are fun for students, and they should get a feeling of accomplishment when they get them working.

We consider the first challenge, Audio Feedback, the easier of the two projects. The second project may be the better option for the more advanced students or if you have extra time to allow for an additional project.

Encourage your students to work towards creating a simple and concise program. In addition, the code should be kept visibly clean and organized. Can you tell what the program is supposed to do with a quick look at the code? The front panel should provide the user with the necessary controls and indicators, and should also be clean and organized.

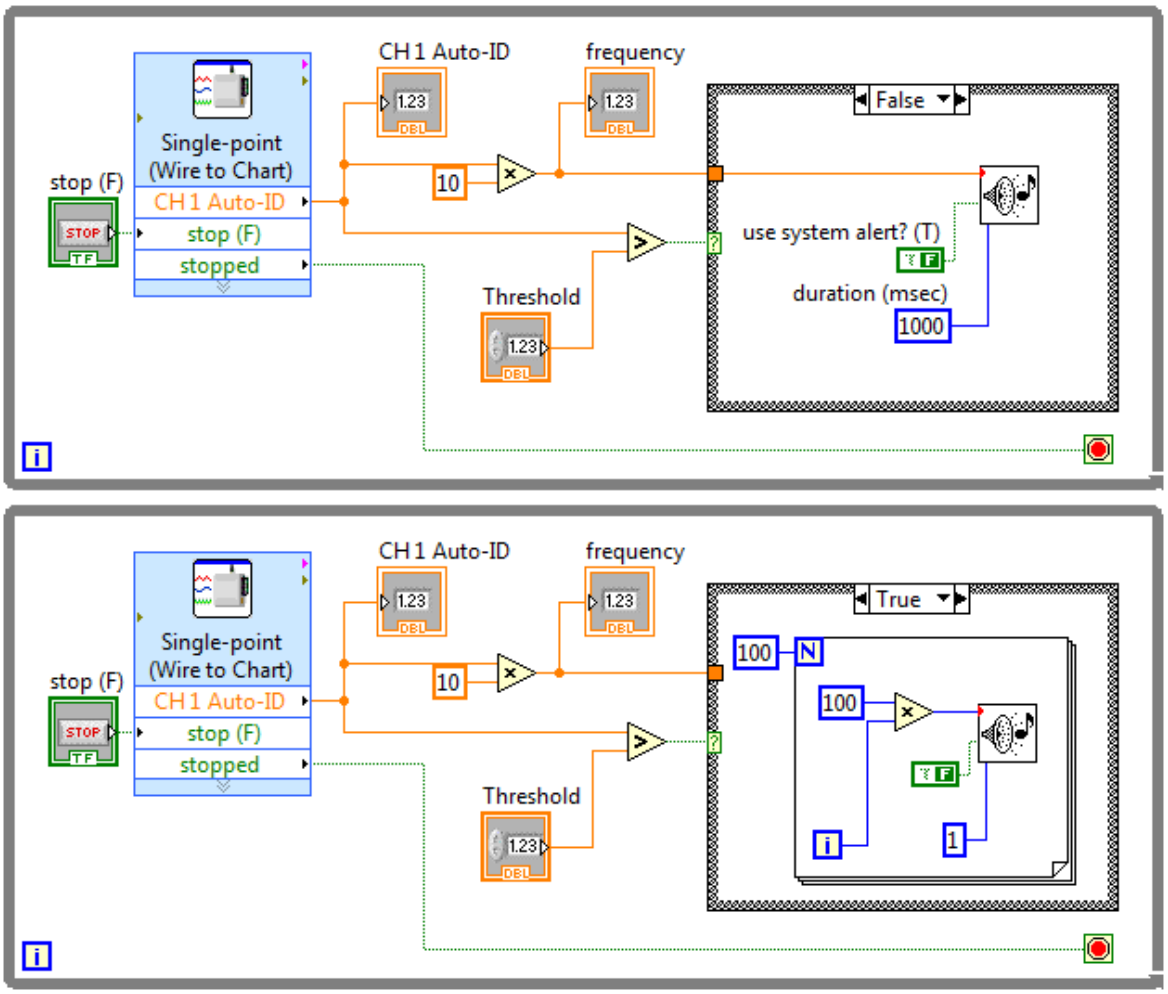
PROJECT 1 – AUDIO FEEDBACK

The Project1 Audio Feedback.vi file can be found on the accompanying CD. This program reads the Vernier Stainless Steel Temperature Probe in a While Loop, with the timing configured in the Analog Express VI set at a rate of 1 sample/second and a length of 120 seconds. The temperature reading is multiplied by 10 in order to create an appropriate frequency level. The result of the multiplication is input into the Beep.vi as the frequency value. The tone is held for a duration of 1000 milliseconds (1 second). After the tone has stayed on for a duration of 1 second, the While Loop iterates and the Express VI outputs another data point.



EXTENSIONS

Extensions to this program could include adding an upper level threshold value. If the threshold is met, a unique sound could signal a warning. An example program that creates a unique pulsing sound when the temperature is above a threshold is shown below. Two screenshots are provided in order to see both cases of the Case Structure.



Top: Audio Feedback program showing the Case Structure's False Case.

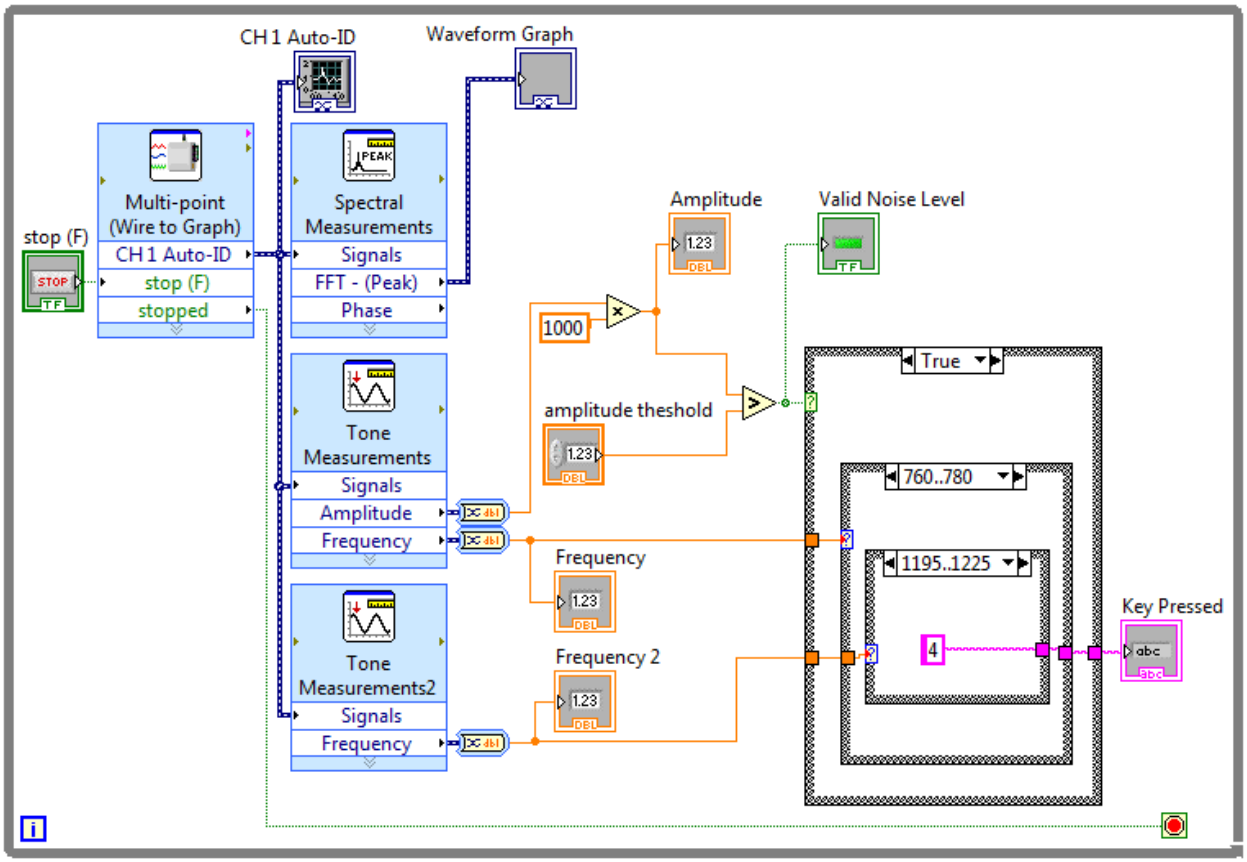
Bottom: Feedback program showing the Case Structure's True Case.

PROJECT 2 – CELL PHONE DECODER

This project is designed to decode the tone of a cell phone or tone dialer. In some classrooms, cell phones are not allowed; therefore, a LabVIEW program called Cell Phone.vi is included on the accompanying CD. This VI uses the computer's speakers to generate DTMF tones for buttons 1–9. And even if you will be using cell phones for this project, the Cell Phone VI might be of interest to the students.

As is the case for many programming challenges, there are different ways to determine the correct number that is being pressed on the phone or tone dialer. One method is to create an FFT waveform, and then perform an analysis on this waveform. This waveform peaks at the location of fundamental frequencies. An analysis to determine where these peaks occur provides the low and high frequency values in the DTMF.

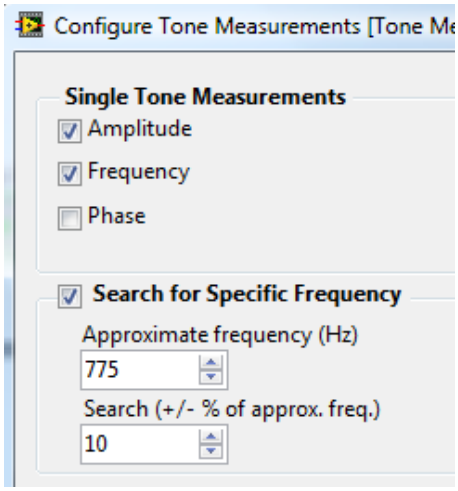
A second method is to use the Tone Measurements Express VI. This VI is described as being able to "find the single tone with the highest amplitude or to search a specified frequency range to find the single tone with the highest amplitude." A program that uses this Express VI to perform the analysis is described below.



The Project2 DTMF Decoder.vi file can be found on the accompanying CD. This program reads the Vernier Microphone in a While Loop, with the timing configured in the Analog Express VI set to "Repeat" at a rate of 20,000 samples/second and a length of 0.2 seconds. The microphone signal from the Analog Express VI is sent to the Spectral Measurement Express VI. The resulting FFT is displayed on a front panel graph to provide the user with a visual representation of the frequency pairs. The microphone signal is also sent to two Tone Measurements Express VIs. These Express VIs analyze the waveform and determine the low and high frequency. These

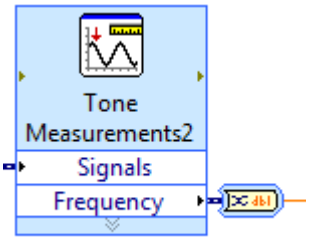
frequency values are sent to a nested Case Structure that determines the number that corresponds to the frequency pair and displays it to the user.

The analysis in this example program uses the Tone Measurements Express VI to locate the DTMF frequency pair (low frequency and high frequency). When the Tone Measurements Express VI's configuration dialog box is opened, there is an option to "Search for Specific Frequency".

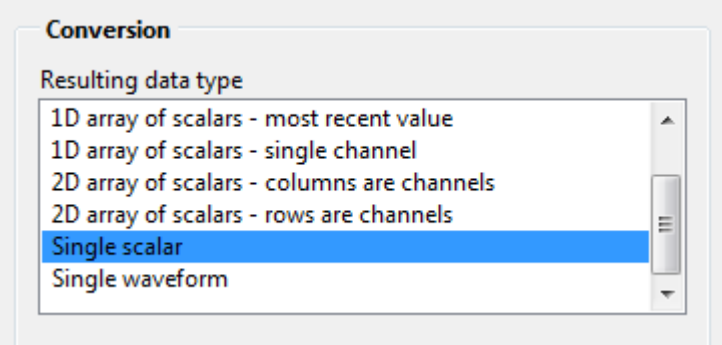


Checking this option allows you to place two instances of the Express VI in your program; one instance is configured to locate the low frequency (in the range of 697 to 852 Hz) and the other instance is configured to locate the high frequency (in the range of 1209 to 1477 Hz).

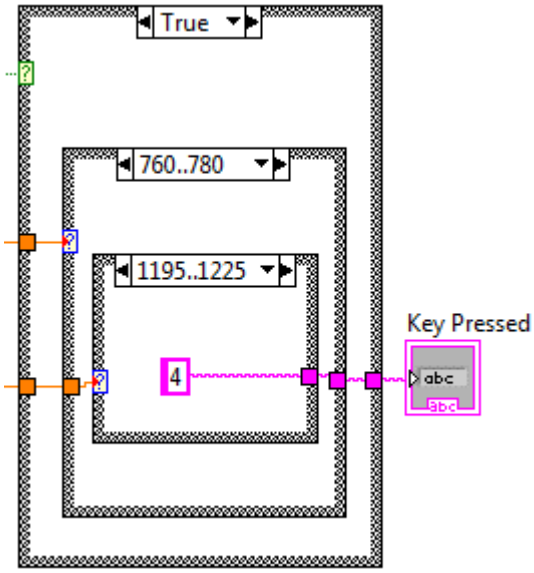
The next step of the program is to use the two frequency values to display the corresponding number 1–9. However, the frequency value sent from the Express VI must first be converted from a dynamic data type to a numeric data type. Express VIs commonly send results as dynamic data. This data type is very flexible; however the Case Structure does not accept this data type. Use the Convert From Dynamic Data Express VI to change the dynamic data to numeric data.



Be sure to double-click the Convert From Dynamic Data Express VI and select "Single scalar" from the "Resulting data type" section of the configuration dialog box.

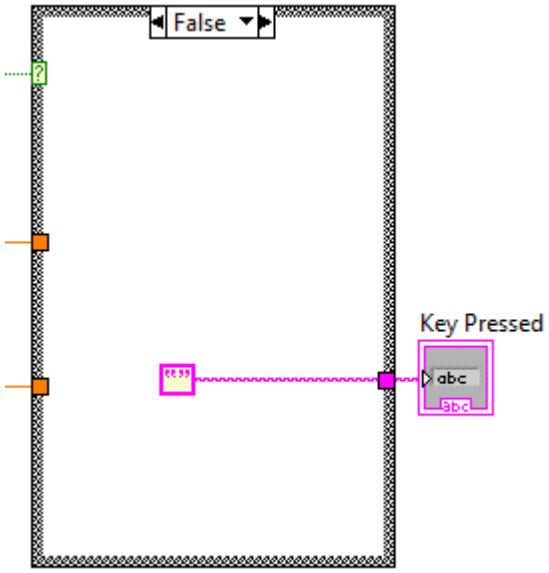


The frequency values are now numeric data types and can be wired directly to a Case Structure to perform the decision making portion of the program. There are three nested Case Structures in this example, as shown in the figure below. Another option is to sum the high and low frequency values and use that value to determine the button number.



The outside Case Structure is used to determine if the tone is coming from a button press or if it is just white noise picked up by the Microphone (an advanced feature that is not stated as a design requirement). The Tone Measurements Express VI that is configured to find the low frequency is also configured to get the amplitude. This amplitude is compared to a user-defined threshold value using the Greater? Function. If the amplitude is not greater than the threshold

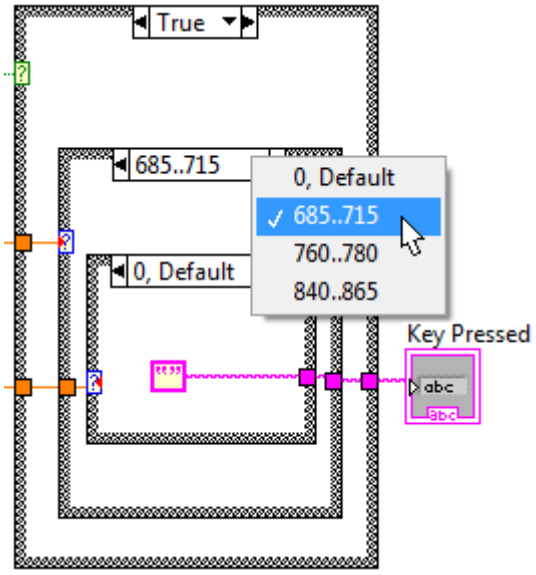
value, the Greater? Function sends a Boolean False value to the outside Case Structure. If this is the case, an empty string value should be displayed to the user, and no code should be performed. This case is shown below.



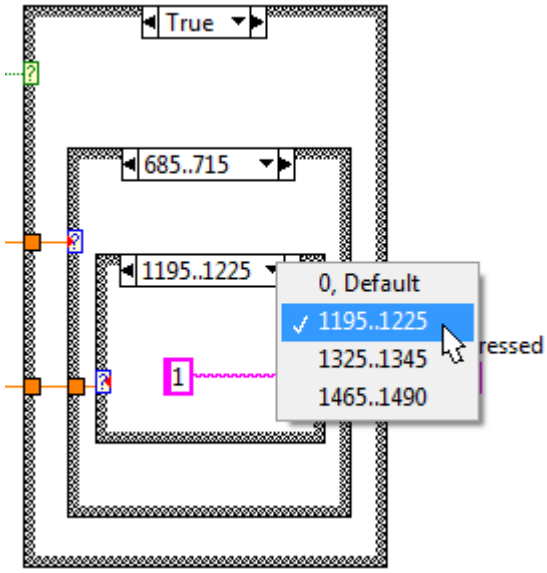
If the amplitude is greater than the threshold value, the Greater? Function sends a Boolean True value to the outside Case Structure. If this is the case, it means the signal is loud enough to be a button press, and the number 1–9 must be determined. The code inside the True Case includes two nested Case Structures to perform the decision making. The outside of these two Case Structures uses the low frequency value as the input value for the case selector; the inside Case Structure uses the high frequency values as the input value.

Remember that the low frequency value for numbers 1, 2, and 3 is 697 Hz. The value for numbers 4, 5, and 6 is 770 Hz. And the value for 7, 8, and 9 is 852 Hz. We cannot expect the result to exactly match these numbers, so the Case Structure is given a range. All three of these

low frequency values have a case, and a fourth default case is included for values outside of the three frequencies. If you click on the down arrow of the Case Structure, these four cases are shown in a shortcut menu, as shown below.



The low frequency is used to narrow the list to three numbers and the high frequency is used as a case selector value to determine the exact number. For example, if the user pressed a "1," the low frequency is 697 Hz, and the outside Case Structure is sent to the "685..715" case. The high frequency for "1" is 1209 Hz, and the inside Case Structure is sent to the "1195..1225" case.



Inside this case is a string value of "1". This value is sent to the "Key Pressed" indicator to display the number on the front panel to the user.

EXTENSIONS

1. Can the program be modified to decode what vowel is spoken by a user?

2. Shift registers are used within While Loops to transfer values from one loop iteration to the next. Use a shift register to store the previous button presses. When you press a new button, add that value to the string that was stored in the shift register in order to display all of the buttons that have been pressed.

3. Import an image of a cell phone onto your front panel and display the button press value on that image, as it would appear on a real cell phone.

Appendix
A

About the Electronic Resources

The electronic resources, accessed through your account at www.vernier.com, contain the following:

- **Chapters and Exercises** – Contains the chapters and exercises for the student activities and projects in this book.
- **Supplemental Materials**
  - **LabVIEW VIs** – Contains the .vi files for each of the student activities (both experiments and projects) in this book. These VIs can be used with any version of LabVIEW 2009 and newer. There are two different versions, depending on the interface used:
    - **LabQuest VIs** – The folder LabQuest VIs contains the VIs for use with any Vernier LabQuest interface and versions of LabVIEW.
    - **SensorDAQ VIs** – The folder SensorDAQ VIs contains the VIs for use with the Vernier SensorDAQ interface and versions of LabVIEW.

USING THE WORD-PROCESSING FILES

All file names begin with the chapter number, followed by an abbreviation of the title. This provides a way for you to edit the content and activities to match your lab situation, your equipment, or your style of teaching. The files contain all figures, text, and tables in the same format as printed in *Hands-On Introduction to NI LabVIEW with Vernier*.

Appendix
B

Equipment and Supplies

This book was written to allow the student activities (exercises and projects) to be completed with little or no supplies. Activities using the Temperature Probe generally use air temperature or hand temperature. Activities that use the Microphone generally use the voice. Exercise 4 uses a tuning fork, but this could be modified to allow the students to hum instead.

A list of equipment and supplies for all the activities is given below. The amounts listed are for individual workstations.

Exercise 4	tuning fork
Exercise 8	2 short wires with the insulation stripped off the ends, precision slotted screwdriver
Project 1	warm tap water, cup or beaker
Project 2	cell phone or tone dialer (optional)

All activities require the use of NI LabVIEW. We recommend and distribute LabVIEW Education Edition. The advantage of the Education Edition is that it automatically installs files that are needed for use with Vernier interfaces. For example, LabVIEW Education Edition 2009 automatically installs the SensorDAQ files and the required software driver that you need. NI LabVIEW version 8.2 or newer can also be used. See *Appendix D* for more information.

Appendix
C

Vernier Products for Engineering Education

NI LabVIEW Education Edition software and all of the hardware required for the activities in this book are available from Vernier Software & Technology. A table of order codes follows, as well as product descriptions and other Vernier products for engineering education.

Vernier Products Used in this Lab Manual

Item	Activity	Order Code
SensorDAQ	All	SDAQ
LabQuest	All (except Ch 8)	LABQ
LabQuest 2	All (except Ch 8)	LABQ2
LabQuest Mini	All (except Ch 8)	LQ-MINI
LabVIEW Education Edition	All	LVEE-1
Temperature Probe	1, 5, 6, 7, 9	TMP-BTA or STS-BTA
Microphone	2, 3, 4, 9	MCA-BTA
Voltage Probe	8 (for SensorDAQ only)	Included with SensorDAQ
Hands-On Introduction to NI LabVIEW with Vernier SensorDAQ	All	LWV

Product Descriptions

SensorDAQ

Designed by National Instruments and Vernier, the SensorDAQ offers convenience and power to engineering students. This small USB interface is perfect for data acquisition with Vernier sensors, sensor-control projects, and introducing LabVIEW programming. It includes screw terminals specifically for engineering projects.

LabQuest Mini

LabQuest Mini is much like a SensorDAQ, but without the screw terminals. It connects via USB and requires no power supply. It has three analog and two digital sensor ports. All the chapters in this book, except Chapter 8, can be done using LabQuest Mini.

LabQuest 2

LabQuest 2 is the deluxe standalone interface from Vernier. It connects via USB and is powered by a built-in rechargeable, high-capacity battery. When not using it with a computer and LabVIEW software, its large, high-resolution touch screen and Wi-Fi connectivity allow students to collect, analyze, and share sensor data wirelessly on any device with a web browser. All the chapters in this book, except Chapter 8, can be done using LabQuest 2.

LabQuest

LabQuest is the original standalone interface from Vernier. It can be used for data collection in the field, or it can be used connected to a computer (as it would be using the examples in this book). All the chapters in this book, except Chapter 8, can be done using LabQuest.

LabVIEW Education Edition

The new National Instruments LabVIEW Education Edition software helps teachers bring STEM concepts to life through hands-on learning. This education edition is industry-standard NI LabVIEW (used throughout the engineering disciplines) with modules for educational hardware, such as the Vernier SensorDAQ, Go! Link, and LEGO NXT Sensor Adapter.

Stainless Steel Temperature Probe

This rugged and durable temperature probe has a sealed stainless steel shaft and tip that can be used in organic liquids, salt solutions, acids, and bases in a range of –40 to 135°C.

Microphone

The Vernier Microphone can be used to display and study the waveforms of sounds from voices and musical instruments.

Voltage Probe

The Voltage Probe has a range of +/-10 V, and can be used to measure the potential in direct-current or alternating-current circuits.

Optional Vernier Products

In addition to the products used in this book, Vernier Software & Technology offers a wide variety of sensors and products designed for engineering education.

Vernier Sensors for Engineering Education

Over 50 sensors can be used with SensorDAQ, including those listed below.

Item	Order Code
3-Axis Accelerometer	3D-BTA
Blood Pressure	BPS-BTA
CO_2 Gas Sensor	CO2-BTA
Current Probe	DCP-BTA
EKG Sensor	EKG-BTA
Hand Dynamometer	HD-BTA
Instrumentation Amplifier	INA-BTA
Light Sensor	LS-BTA
Motion Detector	MD-BTD
Photogate	VPG-BTD
Rotary Motion Sensor	RMV-BTD
Spirometer	SPR-BTA
Thermocouple	TCA-BTA

Vernier Products for Engineering Education

Engineering Projects with NI LabVIEW and Vernier

This book is meant to follow the *Hand-on Introduction to NI LabVIEW with Vernier* book. It is a collection of 12 open-ended, challenging projects. Each project involves some construction and some LabVIEW programming.

Digital Control Unit (DCU)

The Digital Control Unit gives you an easy way to use SensorDAQ, LabPro, LabQuest 2, the original LabQuest, or LabQuest Mini digital sensor ports for exciting, do-it-yourself projects. It provides useful current (up to 600 mA) for controlling electrical devices.

Breadboard Cable

The Breadboard Cable is an easy way for students to build their own sensor circuitry and then input the signal into a Vernier interface. One end of the cable is a standard British Telecom Analog (BTA) connector that plugs into the interface's sensor port. The other end of the cable is a prototyping board connector that exposes the six sensor lines.

Power Amplifier

The Vernier Power Amplifier allows you to control useful output circuits with SensorDAQ and other Vernier lab interfaces. You can drive loads with +/-10 V and currents up to 1 A. It has a built-in current sensor so you can monitor the current output at the same time you control the voltage. Engineering projects, with feedback loops, motor speed control, PID control, and many others are naturals with this hardware.

NXT Sensor Adapter

The NXT Sensor Adapter can be used with LEGO MINDSTORMS NXT robots and over 30 Vernier sensors for sensor-based control systems. The adapter is slightly larger than the LEGO Sound Sensor, and includes a LEGO NXT cable socket on one end and a Vernier BTA sensor socket on the other end. Incorporate science, control, and engineering into your NXT projects using this low-cost adapter.

STEM with Vernier and LEGO MINDSTORMS NXT Lab Manual

This book contains 14 experiments and four robotics projects for data collection using the LEGO NXT Intelligent Brick. This book covers topics in environmental science and engineering, including soil, water, acidity, and UV radiation. Full construction and programming instructions are given for the robotics projects, such as an aquarium monitor or a plant waterer.

STEM 2 with Vernier and LEGO MINDSTORMS NXT Lab Manual

This book contains 12 experiments and eight robotics projects for data collection using the LEGO NXT Intelligent Brick. It covers topics in physical science and engineering, including mechanics, pressure, electricity, and magnetism. Full construction and programming instructions are given for robotics projects, such as a battery tester, string tension tester, Cartesian diver, and mine sweeper.

Appendix D

Software Requirements and Installation

There are three pieces of software required to do all the activities in this book. These three pieces include LabVIEW, the Vernier Toolkit (the driver software required to communicate with the SensorDAQ or any LabQuest interface), and the examples.

SOFTWARE INSTALLATION

1. Install LabVIEW. If LabVIEW is already installed on your computer, check the version of LabVIEW you are using. You must use LabVIEW version 2009 or newer.

2. Run the appropriate installer for the Vernier Toolkit. You can download the installers from the Vernier website, www.vernier.com/labview/downloads

3. (SensorDAQ and Windows only) If you are using a version of LabVIEW that is not the Education Edition, you will need to install the SensorDAQ driver (NI-DAQmx) separately. In order to work correctly, you will need to install NI-DAQmx 9.4 or newer.

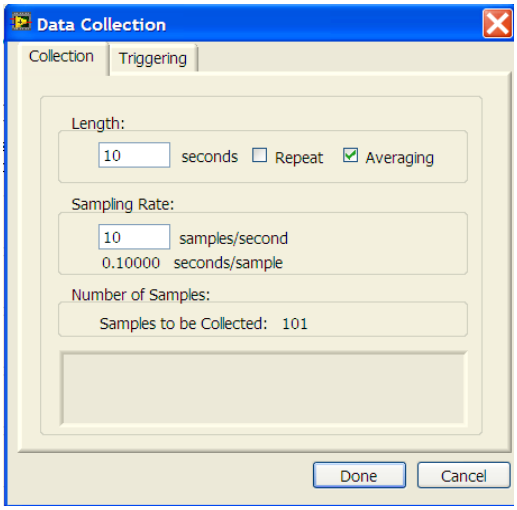
4. The example VIs for the book are found on the CD at the back of the book. Once LabVIEW and the Vernier Toolkit are installed, you will be able to run the examples.

Appendix
E

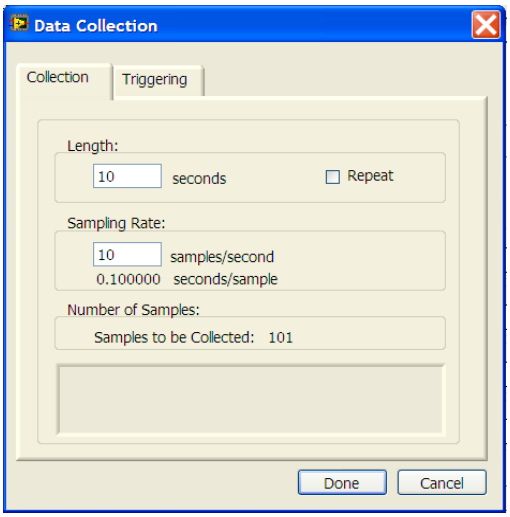
Notes on Screenshots

You may find that the screenshots in this book do not exactly match what you see on your screen. There are several possible reasons:

- There are minor differences in the look of things depending on the version of LabVIEW you are using.
- The Macintosh version of LabVIEW has some differences when compared to the Windows version. Most of the screenshots in this book were done with Windows computers.
- There are even minor differences in the look of things depending on the version of Windows you are using.
- There are minor differences depending on whether you are using a Vernier SensorDAQ interface or a Vernier LabQuest interface. In particular, the dialog that allows you to set the timing of the data collection is slightly different (see below).



Data-collection settings, SensorDAQ Analog Express VI



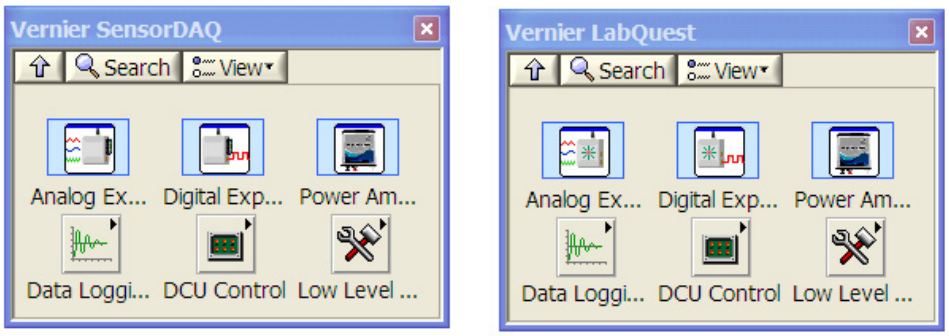
Data-collection settings, LabQuest Analog Express VI

Notice that the SensorDAQ Analog Express VI has an extra check box labeled Averaging. On the LabQuest dialog there is no such check box. On any LabQuest interface, data is always averaged. This means that during slow data collection, the LabQuest interface actually takes data at a rapid rate and reports the average of all the readings made since the last reading.

Finally, LabVIEW includes a feature to customize the palette view. This feature may be helpful if you want to change how many of the palettes are exposed to your students, but it also can make things look slightly different.

Be flexible with the minor differences; however, make sure that you have access to the SensorDAQ or LabQuest palettes.

If the SensorDAQ or LabQuest palette that you need is not there, or does not include Express VIs and subpalettes shown below, then you have not properly installed the necessary files. See *Appendix D* for information.

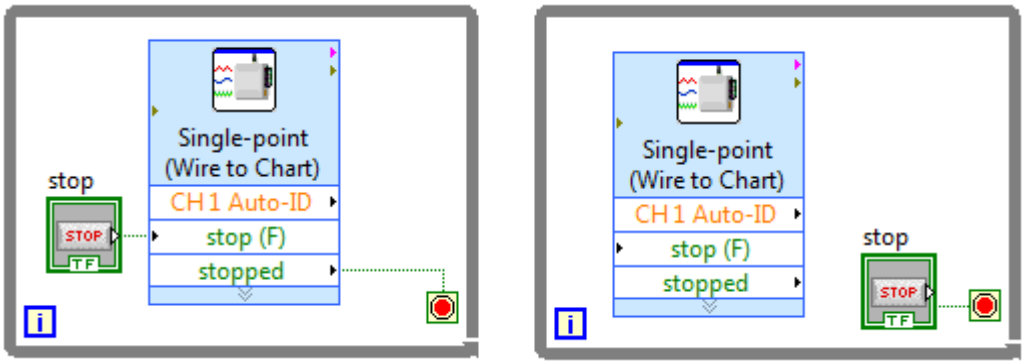


Appendix
F

Common Programming Errors

The exercises in this book mostly use the use the Analog Express VI. Below are some common errors to watch for when using this Express VI.

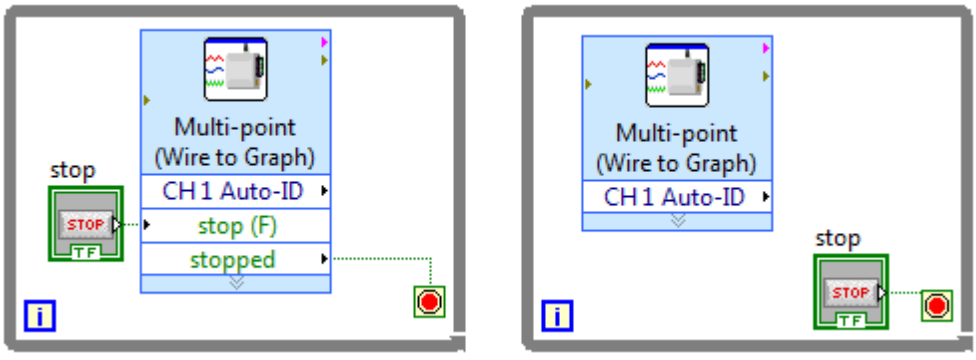
1. The Analog Express VI must be placed within a While Loop if it is configured for point-by-point data collection (sample rates <200 samples/second) or if it is configured for continuous multi-point data collection (sample rates >200 samples/second and Repeat is Active). You will know that the Express VI is configured for placement within a While Loop if the "stop (F)" input terminal and the "stopped" output terminal are visible. The important programming method to follow is to stop the Express VI first, and then the While Loop. The following diagrams show the correct and incorrect wiring diagrams.



Correct (left): Stop the Express VI and then the loop.

Incorrect (right): Stop the loop without stopping the Express VI

2. Continuous, multi-point data collection requires Repeat to be Active. If the Express VI is configured for data-collection rates greater than 200 samples/second, and Repeat is not Active, the Express VI will cause an error if it is placed within a While Loop. The following diagrams show the correct and incorrect wiring diagrams.



Correct (left): Repeat is Active and the stop terminals are visible.

Incorrect (right): Repeat is not Active and therefore this is not configured to be continuous, and the stop terminals are not visible.

3. If you are using a SensorDAQ interface in Exercise 8, you will be using the DAQ Assistant Express VI. Here is a common error made using it.

   The DAQ Assistant Express VI is a great way to quickly program the SensorDAQ screw terminal channels. However, be aware that this Express VI, when configured, is specifically set to run with the SensorDAQ connected to your computer. If you attach a new SensorDAQ, the Express VI will not work. A common error message that occurs for this problem will say something similar to the following:

   "Error -201003 occurred at DAQmx Create Channel (AO-Voltage-Basic).vi:2
   Device cannot be accessed. Possible causes:

   Device is no longer present in the system.
   Device is not powered.
   Device is powered, but was temporarily without power.
   Device is damaged.

   Ensure the device is properly connected and powered. Turn the computer off and on again. If you suspect that the device is damaged, contact National Instruments at ni.com/support.

   Device Specified: Dev4

   Task Name: _unnamedTask<0>"

   This error occurs because your computer gives each SensorDAQ a unique device number. The DAQ Assistant is configured with this device number. If you save a DAQ Assistant VI created with SensorDAQ device 1 (for example), and then attach a second SensorDAQ (device 2), and run the saved VI, an error will occur. There is a way to manage the device number assigned to the SensorDAQ. This is done with Measurement and Automation Explorer (MAX), an application installed in the LabVIEW directory. It may be easier to recreate and reconfigure a new DAQ Assistant. The best option is to keep a single SensorDAQ connected to each computer. Every computer will then have a SensorDAQ that is named device 1, allowing all VIs to be shared between computers.

Appendix
G

Using the Book in the Classroom

One of the authors of the book, Steve Decker, teaches at Oregon Episcopal School in Portland, Oregon. He incorporated the material in this book into his Electronics course before it went to press. The following is an account of his classroom experience. He is happy to collaborate with anyone using this book. You may contact him at sdecker@vernier.com.

INTRODUCTION

In late 2009, I taught LabVIEW programming using *Hands-On Introduction to LabVIEW with Vernier* " to a group of 15 seniors and one junior at the Oregon Episcopal School, in Oregon. OES is a private Episcopal school, with both day students from the local community and boarding students from other countries and other parts of the US. I used the Vernier SensorDAQ.

LabVIEW was taught as part of a semester-long Electronics course that is a senior elective, so to some degree the students are self-selected based on interest in science and engineering. The students worked in teams of two on the LabVIEW exercises. The course is lab and project centered and starts with basic conceptual circuit concepts and moves through detailed AC circuit analysis. A centerpiece of the course is a semester-long independent design project, assigned in early September and completed by the end of the semester, that absorbs about 25% of instructional and homework time. The scope of the independent-design project and its implementation techniques are left to the students. For many of the students' projects, LabVIEW and the SensorDAQ could be used to advantage.

TIMING

The LabVIEW unit was placed immediately after an extensive series of conceptual circuits labs, just after the midpoint of the semester, and as the students were moving from system level to detailed design on their independent projects. A copy of this book was distributed to each student, and a heavy emphasis was placed on reading the appropriate chapters before the class session. The unit was set up to take four 50-minute periods to accomplish all eight exercises, and two periods for the project. Groups that were moving faster than the average were encouraged to pursue as many extensions as possible. Classes were organized with 10 minutes of initial lecture about tips and techniques, followed by hands-on work with the exercises, sensors, and SensorDAQ. Students were allowed access to the lab after school to complete work not finished in that day's session. The sessions were divided up as follows:

Session	Allotted Time for Activities	Activity
1	50 minutes	Exercises 1, 2, and 3
2	50 minutes	Exercises 4 and 5
3	50 minutes	Exercises 6 and 7
4	50 minutes	Exercise 8
5	50 minutes	Project day 1
6	60 minutes	Project day 2

The session with Exercises 6 and 7 had a bit too little time, while the one with Exercise 8 was too generous, which surprised me. The high-performance teams completed all exercises in the allotted time. The average team required about an extra hour outside of class, and the slowest team required about two hours outside of class, due in part to some struggles with English. The design challenge project was put at the end of the course. The students were given one 50-minute period and one 60-minute period to complete their chosen design challenge. The 60 minute period included a brief presentation/demo by each team. I added a third design challenge[1] choice that I judged to be intermediate in difficulty between the two in the text—the design of a musical instrument. Again, some teams elected to put in time outside of class, largely because they got absorbed in the project.

EVALUATION OF THE STUDENTS' WORK

The completed exercises and the design challenge were e-mailed to me for evaluation. They were judged on the basis of functionality, appearance, and quality of comments included in the code. My comments were inserted directly into the LabVIEW virtual instrument (VI) and mailed back to the students. The teams were required to make a brief presentation of their design challenge results to the class at the end of the second work period.

Two minor difficulties arose in this process. In Exercise 7, the students need to be reminded to mail in the subVI as well as the VI. In Exercise 8, the DAQ Assistant Express VI did not run on my computer because my SensorDAQ had a device number that was different than the student's[2]. It was easier to evaluate this exercise by looking over the students' shoulders.

DISCUSSION

During both the exercises and the design challenge, the students' engagement level was high, even in the face of some frustration for those who had no programming experience. (Only three of the students had had any programming experience prior to the course. Some teams have been learning microprocessor (Basic) programming on their own as part of their design project.)

The following information came from written anonymous evaluations by students at the end of the unit:

Most students reported a good correlation between the circuit and communications concepts in the earlier weeks of the class and the information flow concepts in LabVIEW.

Some students thought the information would help their independent design project, depending on its nature. I think more students would have benefitted in this area if the unit had been earlier in the semester, before their system level designs were complete.

In terms of timing, most students found the pace a little fast, both on the exercises and on the design challenges. In general, they felt that one more period on the exercises and one more on the design challenges would have been beneficial. The pacing is a tradeoff between keeping the fast groups engaged, and allowing enough to time to let the slower groups have enough success to motivate themselves. If the students could get a version of the software on their personal computers, then some of the material could be pushed into homework time without requiring the students to use out of class time at school on the school computers.

[1] An outline of this project to create a musical instrument is included on the CD.
[2] An explanation of the device number issue can be found in *Appendix F*.

In the design challenge project, to induce students to challenge themselves, I offered an extra-credit incentive of 30% if they worked on the most difficult challenge (the cell phone DTMF decoder). I offered a variable incentive (up to 30%) for the musical instrument, depending on how far the student took the idea beyond the basic audio instrument of the first challenge. One group went as far as to build external circuitry to make an electric piano. This aspect of the unit needs a bit of tuning; I think the incentive was a bit too high.

Overall, over half the class liked the material in this unit. With one exception, the remainder of the class was neutral towards it. I will definitely include this material again in this course next year, with a few adjustments.