

Data Acquisition and Signal Conditioning Course Manual

Course Software Version 2009
February 2010 Edition
Part Number 320733N-01

Copyright

© 1995–2010 National Instruments Corporation. All rights reserved.

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

National Instruments respects the intellectual property of others, and we ask our users to do the same. NI software is protected by copyright and other intellectual property laws. Where NI software may be used to reproduce software or other materials belonging to others, you may use NI software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

Trademarks

CVI, National Instruments, NI, ni.com, and LabVIEW are trademarks of National Instruments Corporation. Refer to the *Terms of Use* section on ni.com/legal for more information about National Instruments trademarks.

Tektronix® and Tek are registered trademarks of Tektronix, Inc. The mark LabWindows is used under a license from Microsoft Corporation. Windows is a registered trademark of Microsoft Corporation in the United States. Other product and company names mentioned herein are trademarks or trade names of their respective companies.

Members of the National Instruments Alliance Partner Program are business entities independent from National Instruments and have no agency, partnership, or joint-venture relationship with National Instruments.

Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the **patents.txt** file on your media, or ni.com/legal/patents.

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

Worldwide Offices

Australia 1800 300 800, Austria 43 662 457990-0, Belgium 32 (0) 2 757 0020, Brazil 55 11 3262 3599, Canada 800 433 3488,
China 86 21 5050 9800, Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 358 (0) 9 725 72511,
France 01 57 66 24 24, Germany 49 89 7413130, India 91 80 41190000, Israel 972 3 6393737, Italy 39 02 41309277,
Japan 0120-527196, Korea 82 02 3451 3400, Lebanon 961 (0) 1 33 28 28, Malaysia 1800 887710, Mexico 01 800 010 0793,
Netherlands 31 (0) 348 433 466, New Zealand 0800 553 322, Norway 47 (0) 66 90 76 60, Poland 48 22 328 90 10,
Portugal 351 210 311 210, Russia 7 495 783 6851, Singapore 1800 226 5886, Slovenia 386 3 425 42 00,
South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 (0) 8 587 895 00, Switzerland 41 56 2005151,
Taiwan 886 02 2377 2222, Thailand 662 278 6777, Turkey 90 212 279 3031, United Kingdom 44 (0) 1635 523545

For further support information, refer to the *Additional Information and Resources* appendix. To comment on National Instruments documentation, refer to the National Instruments Web site at ni.com/info and enter the info code feedback.

Contents

Student Guide

A. NI Certification	vii
B. Course Description	viii
C. What You Need to Get Started	viii
D. Course Goals.....	x
E. Course Conventions	xi

Lesson 1

Overview of a DAQ System

A. DAQ System Overview	1-2
B. Sensors	1-2
C. Signals.....	1-4
D. DAQ Hardware	1-10
E. Signal Conditioning	1-10
F. DAQ Software	1-11
Summary	1-12

Lesson 2

Data Acquisition Hardware and Software

A. DAQ Hardware Overview	2-2
B. Components of a DAQ Device	2-7
C. Choosing Appropriate DAQ Hardware	2-13
D. DAQ Software Overview.....	2-20
E. Overview of NI-DAQmx VIs	2-27
Summary	2-31

Lesson 3

Analog Input

A. Grounding Issues	3-2
B. Sampling Considerations	3-13
C. Single Sample Software-Timed Acquisition	3-20
D. DAQ Device Architectures	3-23
E. Finite Buffered Acquisition	3-26
F. Continuous Buffered Acquisition	3-30
G. Triggering	3-35
Summary	3-40

Lesson 4

Analog Output

A. Analog Output Architecture.....	4-2
B. Single Sample Generation	4-3
C. Finite Buffered Generation	4-5
D. Continuous Buffered Generation	4-10
E. Triggered Generation	4-14
Summary	4-15

Lesson 5

Digital I/O

A. Digital Overview.....	5-2
B. Digital I/O	5-5
C. Hardware-Timed Digital I/O	5-7
Summary	5-9

Lesson 6

Counters

A. Counter Signals.....	6-2
B. Edge Counting	6-7
C. Pulse Generation	6-10
D. Pulse Measurement	6-15
E. Frequency Measurement.....	6-18
F. Position Measurement.....	6-23
Summary	6-27

Lesson 7

Signal Conditioning

A. Overview of Signal Conditioning.....	7-2
B. Signal Conditioning Systems	7-2
C. Signal Conditioning for Voltage Measurements	7-3
D. Temperature Measurements.....	7-13
E. Strain, Pressure, Load, and Torque Measurements	7-20
F. Sound and Vibration Measurements	7-32
Summary	7-38

Lesson 8

Synchronization

A. Synchronizing Measurements	8-2
B. Single Device Synchronization.....	8-6
C. Multiple Device Synchronization	8-9
D. Counters and Synchronization	8-13
Summary	8-17

Appendix A**DAQ Theory**

A. Theory of Common Sensors	A-2
B. Analog I/O Circuitry	A-7

Appendix B**Signal Processing**

A. Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT).....	B-2
B. Magnitude and Phase Information	B-4
C. Frequency Spacing and Symmetry of the DFT/FFT	B-5
D. Power Spectrum.....	B-10
E. About Spectral Leakage and Smoothing Windows	B-14
F. Characteristics of Different Types of Window Functions	B-18
G. Determining Which Type of Window to Use	B-24
H. Filtering.....	B-27
I. Ideal Filters	B-27
J. Practical (Nonideal) Filters	B-30
K. Advantages of Digital Filters over Analog Filters	B-31
L. IIR and FIR Filters	B-32
M. Infinite Impulse Response Filters	B-34
N. IIR Filter Comparison	B-39
O. Transient Response of IIR Filters	B-41
P. Finite Impulse Response Filters.....	B-43
Summary	B-46

Appendix C**Course Slides****Appendix D****Additional Information and Resources****Course Evaluation**

National Instruments
Not for Distribution

Student Guide

Thank you for purchasing the *Data Acquisition and Signal Conditioning* course kit. This course manual and accompanying software are used in the two-day, hands-on *Data Acquisition and Signal Conditioning* course.

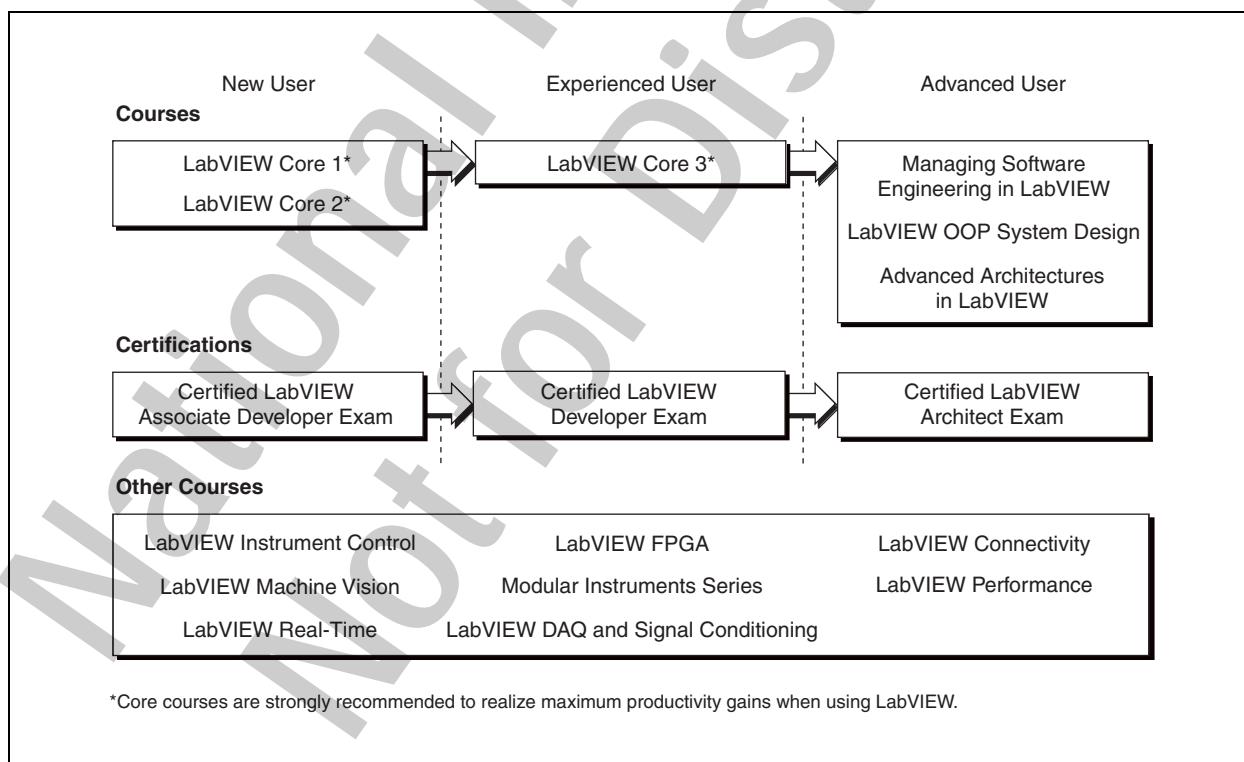
You can apply the full purchase of this course kit toward the corresponding course registration fee if you register within 90 days of purchasing the kit. Visit ni.com/training for online course schedules, syllabi, training centers, and class registration.



Note For course manual updates and corrections, refer to ni.com/info and enter the info code daqnscc.

A. NI Certification

The *Data Acquisition and Signal Conditioning* course is one of many courses offered by National Instruments. The following illustration shows the courses that are part of the LabVIEW training series. If you want to build your proficiency with LabVIEW and prepare for exams to become an NI Certified LabVIEW Developer and NI Certified LabVIEW Architect, refer to ni.com/training.



B. Course Description

This course teaches you the fundamentals of data acquisition (DAQ) and signal conditioning and introduces techniques you can implement in real-world applications.

This course also expands on recommended LabVIEW programming styles from the *LabVIEW Core 1* and *LabVIEW Core 2* courses and discusses several programming guidelines you can use in your data acquisition and signal conditioning applications. This course manual assumes that you are familiar with Windows, that you have experience writing algorithms in the form of flowcharts or block diagrams, and that you have taken the *LabVIEW Core 1* course or you have familiarity with all the concepts contained therein. The course and exercises manuals are divided into lessons, described as follows.

In the course manual, each lesson consists of the following:

- An introduction that describes the purpose of the lesson and what you will learn
- A description of the topics in the lesson
- A summary that outlines important concepts and skills taught in the lesson
- A summary quiz that tests and reinforces important concepts and skills taught in the lesson

In the exercise manual, each lesson consists of the following:

- A set of exercises to reinforce topics
- (Optional) Self-study and challenge exercise sections or additional exercises

Several exercises in this manual use a plug-in multifunction I/O data acquisition (DAQ) device connected to an NI BNC-2120. Some exercises also use a cDAQ chassis, NI 9219 universal analog input module, and sensors as described in Lesson 7, *Signal Conditioning*.

C. What You Need to Get Started

Before you use this manual, make sure you have all the following items:

- Windows XP or later installed on your computer
- LabVIEW Full or Professional Development System 2009 or later
- NI-DAQmx driver software version 9.0.2 or later

- Multifunction DAQ device
- NI BNC-2120, wires, and cable
- cDAQ chassis, NI 9219 universal analog input module, and sensors as described in Lesson 7, *Signal Conditioning*
- USB Cable
- Data Acquisition and Signal Conditioning* course CD, containing the following files:

Directory	Description
Exercises	Contains all the VIs and support files needed to complete the exercises in this course
Solutions	Contains completed versions of the VIs you build in the exercises for this course



Note This course assumes you are using the default installation of LabVIEW. If you have changed the palette views from the default settings, some palette paths described in the course may not match your settings. To reset palette views to LabVIEW defaults, select **Tools»Options** from the top pull-down menu and select **Controls/Functions Palettes** under Category. Set **Palette** to **Category (Standard)**, set **Navigation Buttons** to **Label Selected Icons**, set **Loading** to **Load palette in background**, and enable the **Use window titles in Functions palette** check box. Click **OK** to apply the changes and close the dialog box.

Installing the Course Software

To install the software used for this course, insert the course CD and install the required files in the following directory structure:

- All exercise VIs and associated support files are in the <Exercises>\DAQ and Signal Conditioning directory.
- All exercise solutions are in the <Exercises>\DAQ and Signal Conditioning directory.



Tip Folder names in angle brackets, such as <Exercises>, refer to folders on the root directory of your computer.

D. Course Goals

The purpose of this course is to teach you the components of a DAQ system and to teach you how to use that system. By the end of this course, you will understand the components of a DAQ system:

- Sensors
- Signals
- Signal conditioning
- DAQ hardware
- DAQ software

You also will know how to use LabVIEW with a DAQ device to:

- Acquire analog signals
- Generate analog signals
- Perform digital input and output
- Use counters for event counting, pulse generation, pulse measurement, and frequency measurement
- Perform signal conditioning on acquired signals
- Synchronize multiple tasks and multiple devices

This course does *not* describe any of the following:

- Basic principles of LabVIEW covered in the *LabVIEW Core 1* course
- Every built-in VI, function, or object; refer to the *LabVIEW Help* for more information about LabVIEW features not described in this course
- Developing a complete application for any student in the class; refer to the NI Example Finder, available by selecting **Help»Find Examples**, for example VIs you can use and incorporate into VIs you create

Refer to the *LabVIEW Help* for more information about a particular DAQmx VI.

If you are seeking help for developing your application, please discuss this need with your instructor outside of class time or contact National Instruments Technical Support. For further information on technical support, refer to the National Instruments Web site at ni.com.

E. Course Conventions

The following conventions are used in this course manual:

- » The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.



This icon denotes a tip, which alerts you to advisory information.



This icon denotes a note, which alerts you to important information.



This icon denotes a caution, which advises you of precautions to take to avoid injury, data loss, or a system crash.

bold

Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names, controls and indicators on the front panel, dialog boxes and pages, and sections and components of dialog boxes. Window names and palette names are also denoted in bold.

italic

Italic text denotes variables, emphasis, a cross-reference, or an introduction to a key concept. Italic text also denotes text that is a placeholder for a word or value that you must supply.

monospace

Text in this font denotes text or characters that you enter from the keyboard, sections of code, programming examples, and syntax examples. This font also is used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.

monospace

italic

Italic text in this font denotes text that is a placeholder for a word or value that you must supply.

National Instruments
Not for Distribution

Overview of a DAQ System

This lesson introduces the basics of data acquisition (DAQ).

Topics

- A. DAQ System Overview
- B. Sensors
- C. Signals
- D. DAQ Hardware
- E. Signal Conditioning
- F. DAQ Software

National Instruments
Not for Distribution

A. DAQ System Overview

Data acquisition is the automatic collection of data from sensors, instruments, and devices: in a factory, laboratory, or in the field. The purpose of a DAQ system is to measure a physical phenomenon, such as light, temperature, pressure, or sound. A DAQ system includes the following building blocks:

- Sensor/Signal
- DAQ Hardware
- Signal Conditioning
- DAQ Software

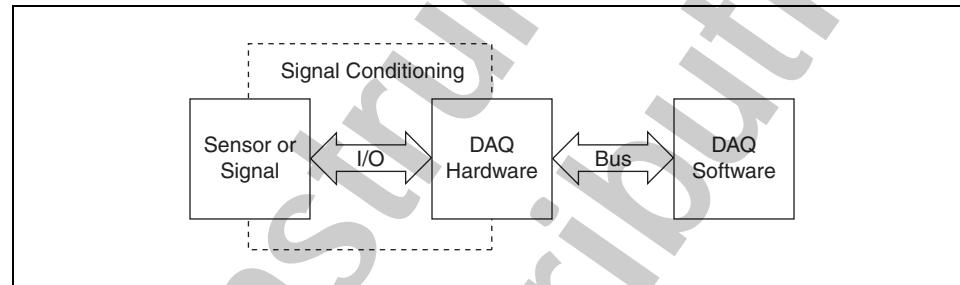


Figure 1-1. DAQ System Overview

With these building blocks, you can bring the physical phenomenon you want to measure into the computer for analysis and presentation.

B. Sensors

Signal acquisition is the process of converting physical phenomena into data the computer can use. A measurement starts with using a sensor, also called a transducer, to convert a physical phenomenon into an electrical signal that a DAQ system measures. Sensors can generate electrical signals to measure parameters, such as temperature, force, position, sound, or light. Table 1-1 lists some common sensors.

Table 1-1. Common Measurements and Sensors

Phenomena	Sensors
Temperature	Thermocouples Resistive temperature detectors (RTDs) Thermistors Integrated circuit sensors
Light	Vacuum tube photosensors Photoconductive cells

Table 1-1. Common Measurements and Sensors (Continued)

Phenomena	Sensors
Sound	Microphones
Force and pressure	Strain gage Piezoelectric transducers Load cells
Position (displacement)	Potentiometers Linear voltage differential transformers (LVDTs) Optical encoders
Fluid flow	Head meters Rotational flowmeters Ultrasonic flowmeters
pH	pH electrodes

Types of Sensors

Sensors are used for a variety of needs, such as measuring temperature, pressure, and fluid flow. Refer to ni.com/sensors for more information about sensors and where to obtain them.

Different sensors have different requirements for converting a physical phenomenon into a measurable signal. For example, a resistance temperature detector (RTD) needs an excitation current to measure the temperature. A thermocouple does not need excitation current, but it does need cold-junction compensation. Strain gages use a configuration of resistors called a Wheatstone bridge to measure strain. Before you set up the system, you should know if the sensor has any special requirements. Contact the vendor of the sensor for more information about how to properly use the sensor.

C. Signals

A sensor converts a physical phenomenon into a signal that you can acquire. However, there are also times that you want to acquire voltage signals that are not produced by a sensor.

Not all signals are measured in the same manner. You first need to categorize the signal as digital or analog.

After you categorize the signal, decide which type of information you want from that signal. The possible types of information you can obtain from a signal are state, rate, level, shape, and frequency, as shown in Figure 1-2.

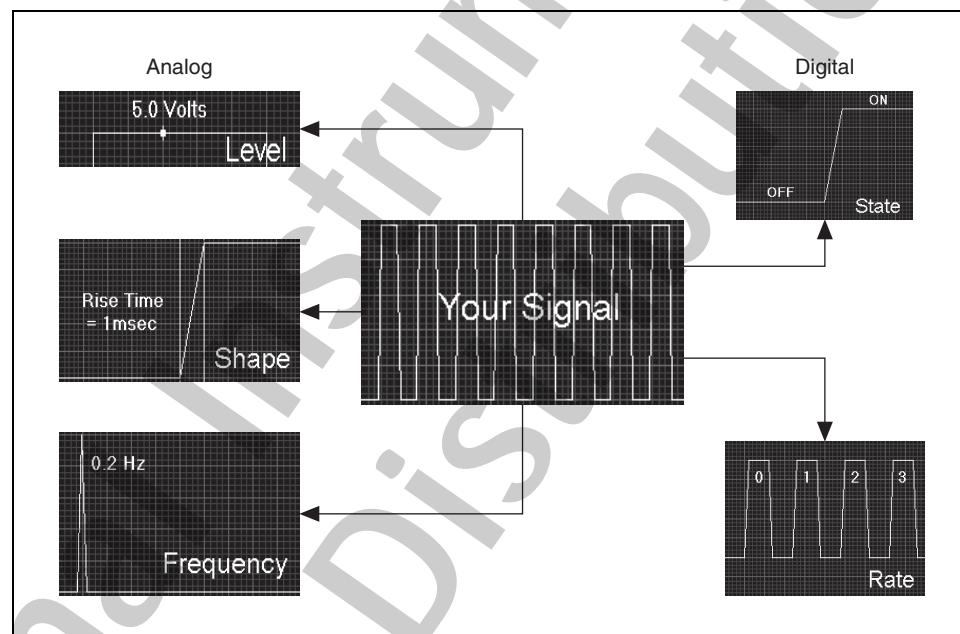


Figure 1-2. Signal Information



Note This discussion of signals assumes that you are acquiring the signal. However, most of the points also apply to generating a signal. The difference between acquiring and generating a signal is that you do not need analysis when you generate a signal with a specific frequency.

Analog Signals

Unlike digital signals, an analog signal can be at any voltage level with respect to time. Because an analog signal can be at any state at any time, the physical aspects you want to measure differ from those of a digital signal.

Analog Signal Information

You can measure the level, shape, and frequency of an analog signal, as shown in Figure 1-3.

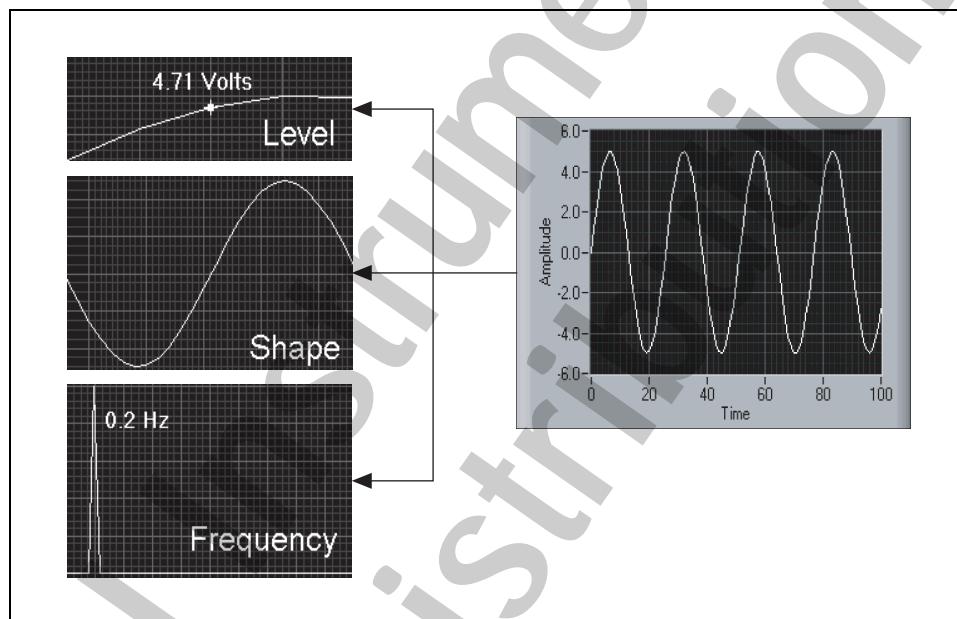


Figure 1-3. Analog Signal Information

- **Level**—Measuring the level of an analog signal is similar to measuring the state of a digital signal. The difference is that an analog signal can be at any voltage level, but a digital signal is either a low or high voltage signal.
- **Shape**—Measuring the shape of the signal is often important because analog signals can be at any state with respect to time. For example, a sine wave has a different shape than a sawtooth wave. Measuring the shape of a signal can lead to analysis of other aspects of the signal, such as peak values, slope, or integration.
- **Frequency**—Measuring the frequency of an analog signal is similar to measuring the rate of a digital signal. However, you cannot directly measure the frequency of an analog signal. You need to perform software analysis on the signal to extract the frequency information, usually with a Fourier Transform.

The level of most signals does not change much with respect to time. However, you usually need to measure the signal with a high level of accuracy. Therefore, you need a DAQ device with a high resolution, but not a high sample rate.

Using a variety of sensors, you could measure the voltage of a power supply, the temperature of a mixing tank, the pressure inside a hose, or the load on a piece of machinery, as shown in Figure 1-4.

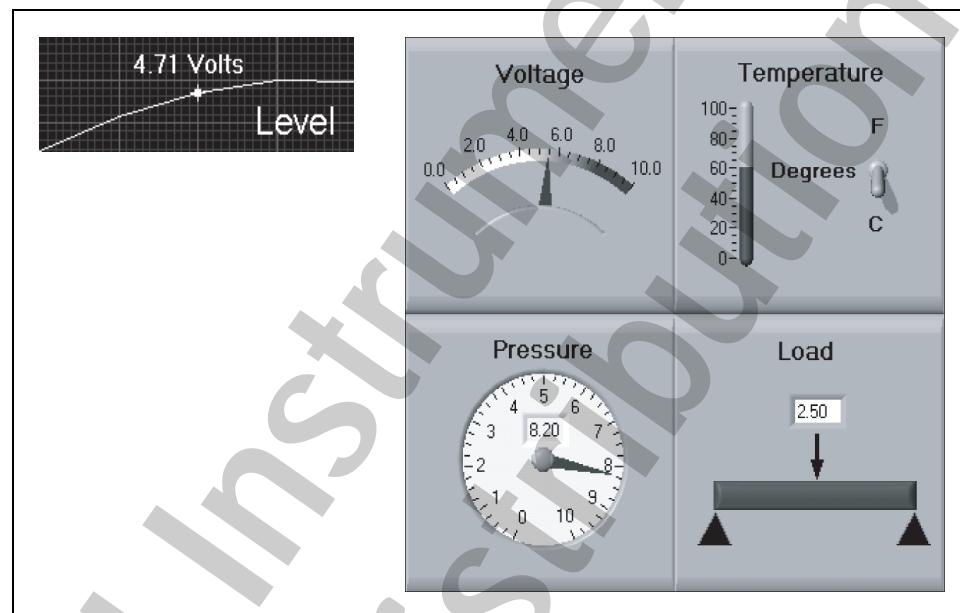


Figure 1-4. Level Examples

When you measure the shape of a signal, you need to know the relationship of that signal with respect to time. Some signals change rapidly with respect to time. Most applications in which you measure shape also need a high level of accuracy. Therefore, you need a DAQ device with a high resolution and a high sample rate.

Examples of measuring shape are abundant in the medical, electronic, and automotive industries, and they range from measuring a heartbeat, to measuring a video signal, to measuring the vibration of a spring. After acquiring the signal, you can analyze it to extract the specific information you need about the shape.

For example, when you measure blood pressure, you are concerned with the peak value. However, with resistor-capacitor (RC) circuit response, you are more concerned with how the amplitude varies over time, as shown in Figure 1-5.

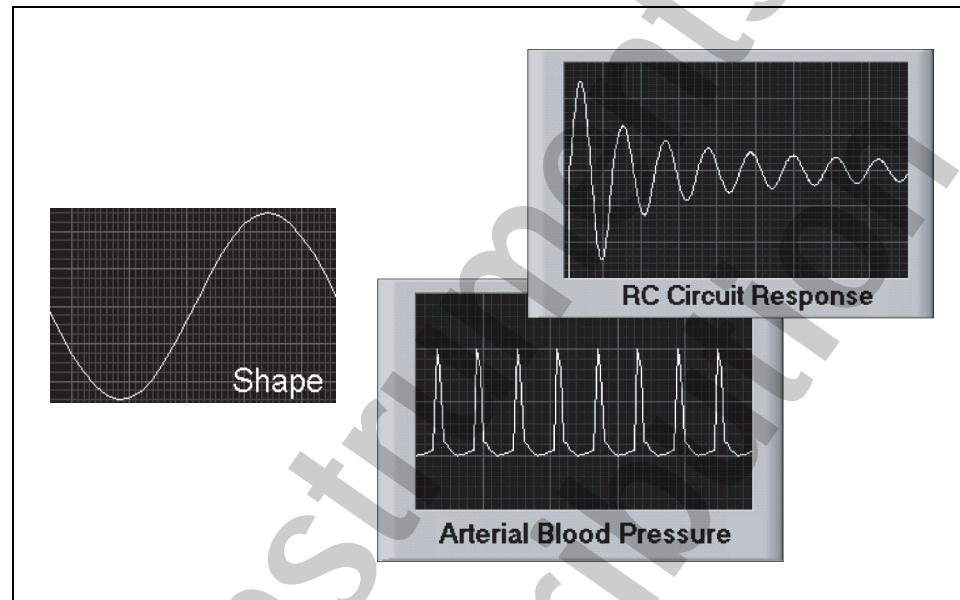


Figure 1-5. Shape Examples

When you acquire the signal with respect to time (time plot), you use software analysis to convert the time plot signal into the frequency plot. LabVIEW provides the necessary software analysis, as shown in Figure 1-6.

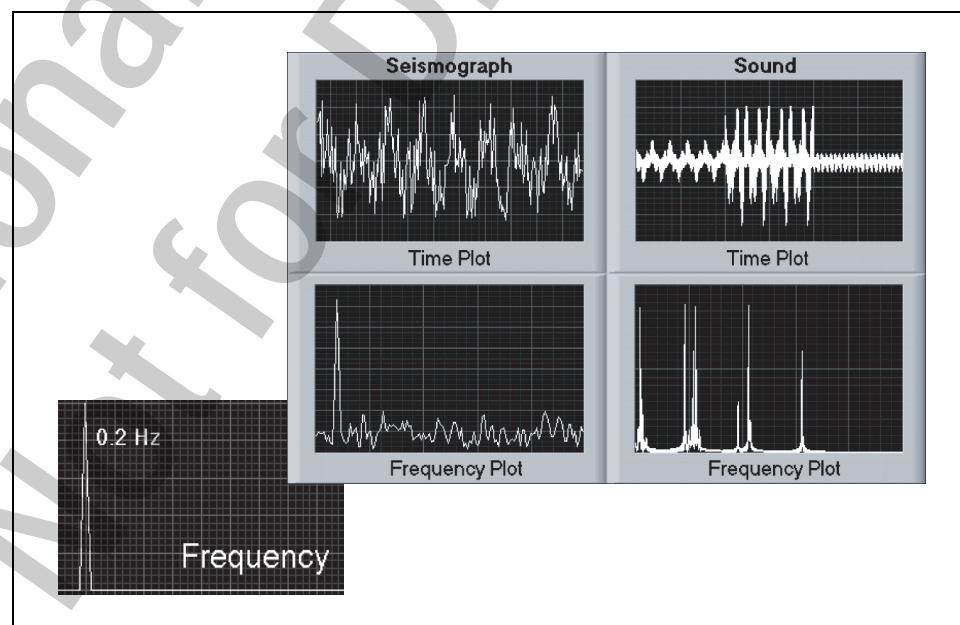


Figure 1-6. Frequency Examples

Digital Signals

A digital signal has only two possible states—ON (logic high) or OFF (logic low). A common type of digital signal is a Transistor-to-Transistor Logic (TTL) signal. The specifications for a TTL signal define a voltage level between 0 and 0.8 V as logic low and a voltage level between 2 and 5 V as logic high. Most digital devices accept a TTL-compatible signal.

Digital Signal Information

You can only measure two aspects of a digital signal: state and rate.

- **State**—A digital signal has only two possible states: ON or OFF. One of the aspects of a digital signal that you can measure whether the state is ON or OFF.
- **Rate**—A digital signal changes state with respect to time. You can measure the rate, or how the digital signal changes states over time.

Consider an example of measuring the state of a digital signal, as shown in Figure 1-7.

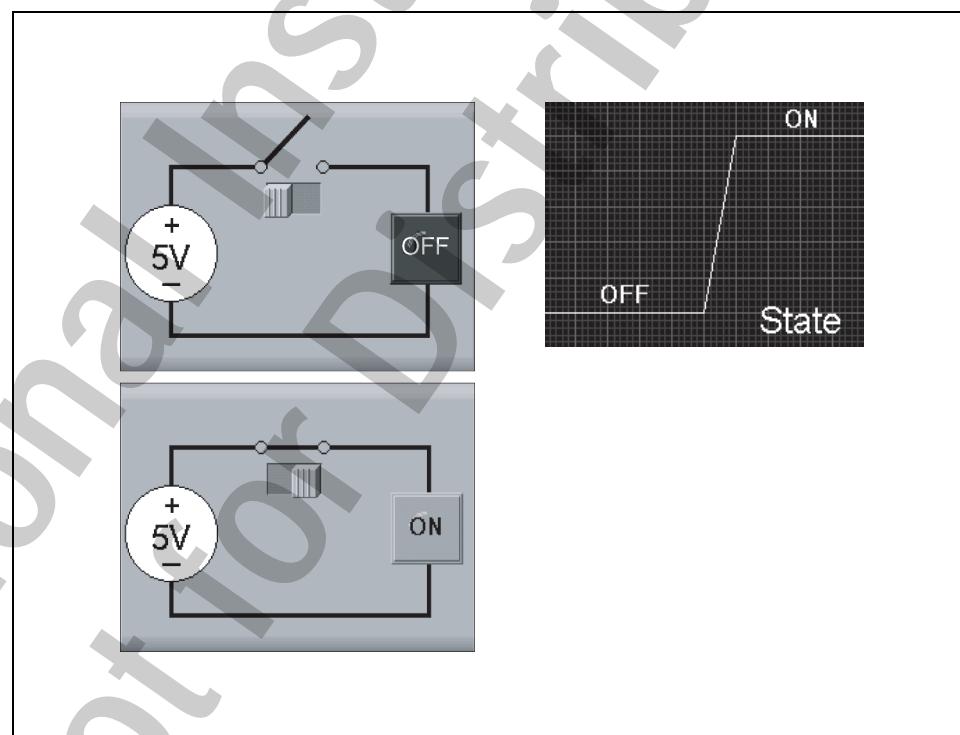


Figure 1-7. State Example

Assume you have a switch that you want to monitor. This switch turns a light on or off. In the example in Figure 1-7, when the switch is open, you measure 0 V (OFF). When the switch is closed, you measure 5 V (ON). By measuring the state of the digital signal, you can determine if the light is on or off.

See Figure 1-8 for an example for measuring the rate of a digital signal.

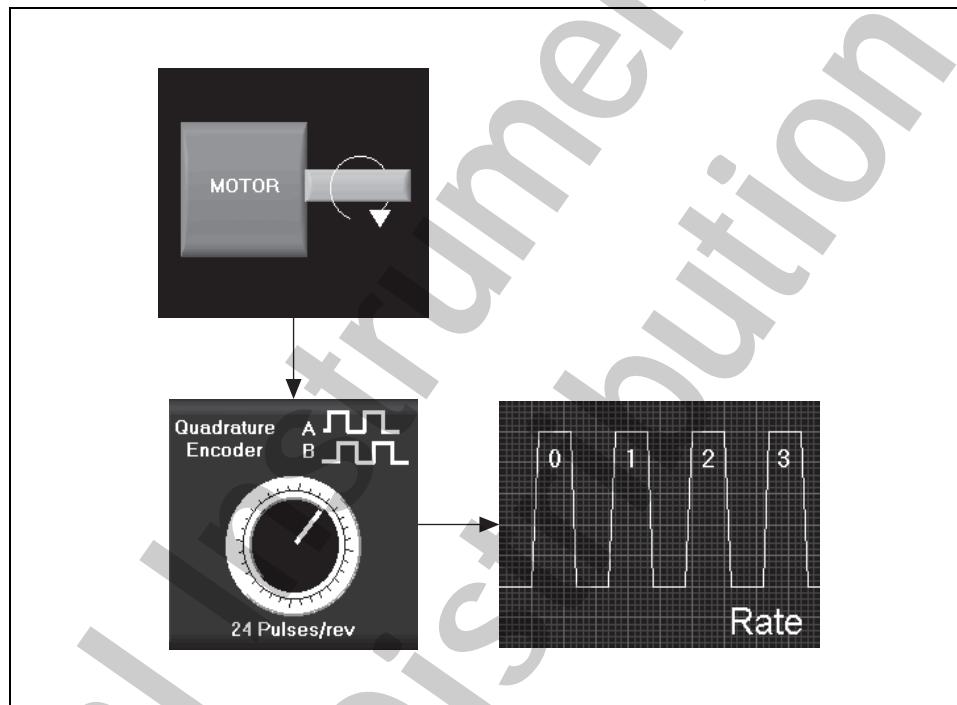


Figure 1-8. Rate Example

Assume you have a motor, and you want to determine how fast the shaft of the motor is spinning. An encoder is a transducer that can convert the rotary motion of the motor shaft into a digital signal. When an encoder rotates, it produces two digital signals. Each digital signal is a series of alternating on or off states, called a pulse train. For each increment of rotation, you get a pulse. The increment of rotation depends on the encoder. For example, the BNC-2120 terminal block you use in this course has an encoder that gives 96 pulses per revolution. You can measure the rate of one of the pulse trains to determine how fast the shaft rotates. You can measure both pulse trains to determine not only how fast the shaft rotates, but also the direction in which it rotates.

D. DAQ Hardware

The purpose of DAQ hardware is to transfer the data between your software and your sensor or signal. DAQ hardware can both acquire and generate analog and digital signals. The DAQ hardware transfers signals to and from the DAQ software through a bus. A few examples of available buses include PCI, PCI Express, PXI, PXI Express, and USB.

E. Signal Conditioning

You cannot always connect the signal directly to a DAQ device. You might need to alter the signal to make it suitable for a DAQ device to measure. Signal conditioning is the process of measuring and manipulating signals to improve accuracy, isolation, filtering, and so on. Signal conditioning is not always required.

The purpose of signal conditioning is to take a signal that is difficult for your DAQ device to measure and alter the signal to make it easier to measure. Many sensors need some sort of external hardware to perform their job. For example, resistance temperature detectors need an excitation current, and strain gages need a configuration of resistors called a Wheatstone bridge. Also, to measure signals from sensors, you must convert them into a form which a DAQ device can accept. For example, the output voltage of most thermocouples is very small and susceptible to noise. Therefore, you might need to amplify the thermocouple output before you digitize it. This amplification is a form of signal conditioning.

Sometimes, signal conditioning can occur in the sensor itself. For example, some microphones and accelerometers contain a built-in amplifier to output a more readable signal. Signal conditioning can also occur along the path between the sensor or signal and DAQ hardware. For example, you can place external amplifiers and filters along the path to help reduce noise. Signal conditioning can also occur in the DAQ hardware. There are many National Instruments DAQ hardware products that have built-in signal conditioning, such as amplification, Wheatstone bridge completion for strain gages, cold-junction compensation for thermocouples, lowpass filtering, and more.

Refer to ni.com/signalconditioning for more information about National Instruments signal conditioning hardware.

Figure 1-9 shows some common types of sensors and signals and the signal conditioning each requires.

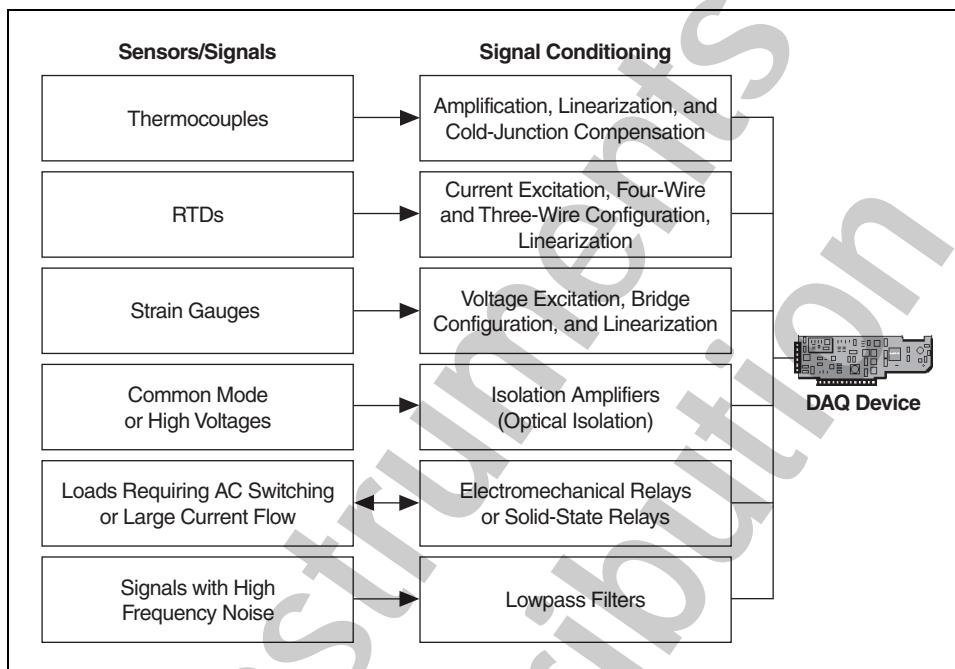


Figure 1-9. Common Types of Sensors/Signals and Signal Conditioning

F. DAQ Software

After you acquire your data from the DAQ hardware, you often still need to do something with the data. You can use DAQ software to generate a report, interact with the data, manipulate signals, analyze signals, log the data to file or a database, and much more.

Summary

A DAQ system consists of the following components:

- Sensors that convert a physical phenomenon into a measurable signal
- Signals, either digital or analog. Depending on the signal, you can measure the state, rate, level, shape, or frequency.
- Signal conditioning, which makes signals easier to measure with a DAQ device
- DAQ hardware
- DAQ software

National Instruments
Not for Distribution

Self-Review: Quiz

1. Match each DAQ term to the definition that best fits:

Sensor	a. Takes a signal that is difficult for your DAQ device to measure and makes it easier to measure
Signal Conditioning	b. Transfers signals to and from software through a bus
DAQ Hardware	c. Operates on data after it has been acquired
DAQ Software	d. Converts physical phenomena into measurable electrical signals

2. List the 3 types of measurements that can be made of analog signals.
3. List the 2 types of measurements that can be made from digital signals.

National Instruments
Not for Distribution

Self-Review: Quiz Answers

1. Match each DAQ term to the definition that best fits:

Sensor

d. Converts physical phenomena into measurable electrical signals

Signal Conditioning

a. Takes a signal that is difficult for your DAQ device to measure and makes it easier to measure

DAQ Hardware

b. Transfers signals to and from software through a bus

DAQ Software

c. Operates on data after it has been acquired

2. List the 3 types of measurements that can be made of analog signals.

- **Level**
- **Shape**
- **Frequency**

3. List the 2 types of measurements that can be made from digital signals.

- **State**
- **Rate**

Notes

National Instruments
Not for Distribution

Data Acquisition Hardware and Software

This lesson describes DAQ hardware and DAQ software.

Topics

- A. DAQ Hardware Overview
- B. Components of a DAQ Device
- C. Choosing Appropriate DAQ Hardware
- D. DAQ Software Overview
- E. Overview of NI-DAQmx VIs

National Instruments
Not for Distribution

A. DAQ Hardware Overview

A typical DAQ system has three basic types of hardware—a terminal block, a cable, and a DAQ device, as shown in Figure 2-1. This section describes each type of hardware, then focuses on the components of a DAQ device and what each component does. You will also learn important considerations to keep in mind when you configure a DAQ device.

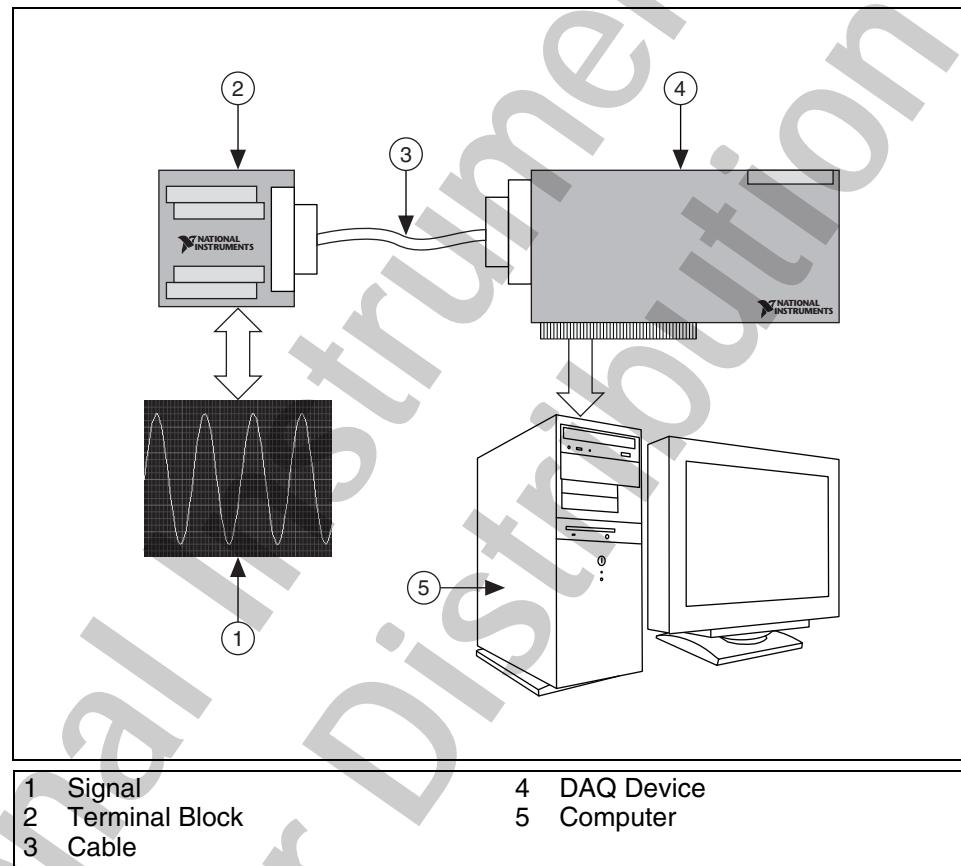


Figure 2-1. Typical DAQ System

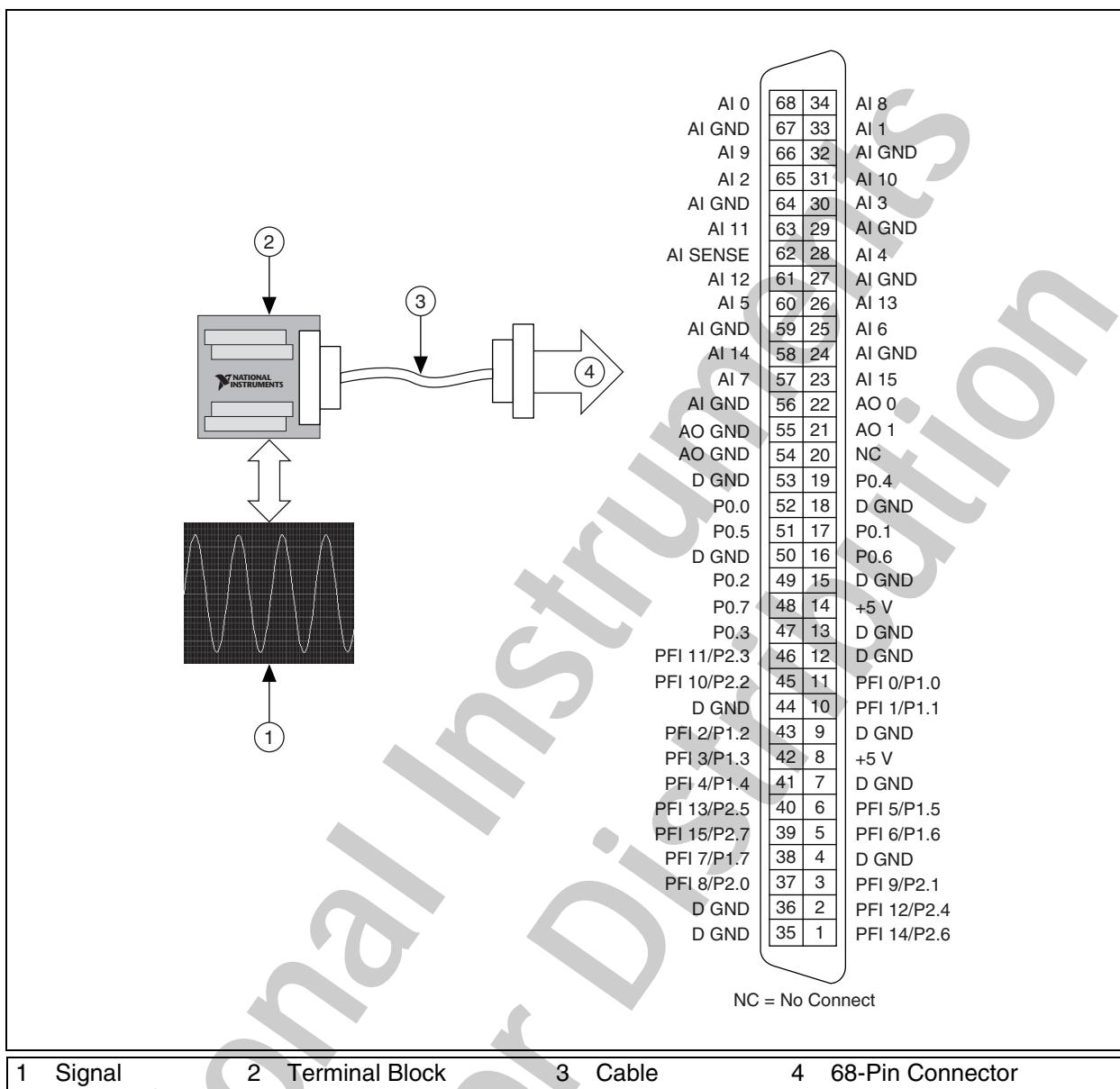
After converting a physical phenomena into a measurable signal, you need to acquire that signal. To acquire a signal, you need a terminal block, a cable, a DAQ device, and a computer. This hardware combination can transform a standard computer into a measurement and automation system.

Terminal Block and Cable

A terminal block provides a place to connect signals. It consists of screw terminals, spring terminals, or BNC connectors for connecting signals and a connector for attaching a cable to connect the terminal block to a DAQ device.

The type of terminal block you should choose depends on two factors—the device and the number of signals you are measuring. Figure 2-2 shows an example layout of the screw terminals.

National Instruments
Not for Distribution

**Figure 2-2.** Terminal Block and Cable with Pinout

Terminal blocks can be shielded or non-shielded. Shielded terminal blocks offer better protection against noise. Some terminal blocks contain extra features, such as cold-junction compensation, that are necessary to measure a thermocouple properly.

A cable transports the signal from the terminal block to the DAQ device. Cables come in 100-, 68-, and 50-pin configurations. Choose a configuration depending on the terminal block and the DAQ device you are using. Cables, like terminal blocks, are shielded or non-shielded.

Refer to the DAQ section of ni.com/products for more information about specific types of terminal blocks and cables.

BNC-2120 Signal Accessory

Figure 2-3 shows the terminal block you are using for this course, the BNC-2120.

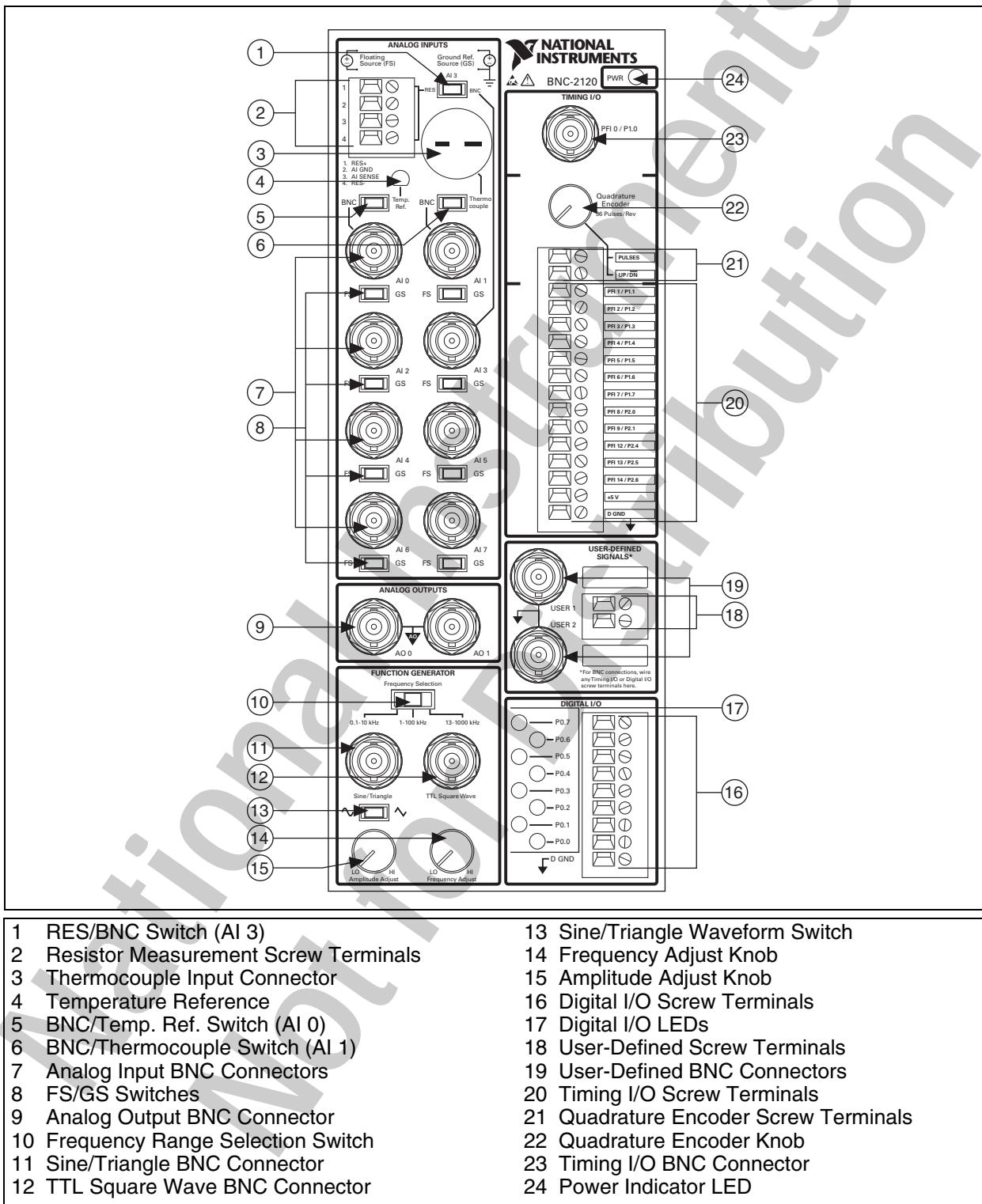


Figure 2-3. BNC-2120 Terminal Block

The NI BNC-2120 is a shielded connector block with signal-labeled BNC connectors. You can use this connector block with NI X Series, M Series, E Series, and S Series multifunction I/O DAQ devices, as well as analog output devices. The BNC-2120 simplifies the connection of analog signals, some digital signals, and two user-defined connections to the DAQ device while maintaining the integrity of your measurements with a shielded enclosure.

The BNC-2120 includes a function generator with a switch to select the frequency range of the signal, a frequency knob, and an amplitude knob. The function generator can produce sine or triangle waveforms and TTL-compatible square waveforms.

The BNC-2120 contains a mechanical quadrature encoder circuit that produces 96 pulses per encoder revolution. Two outputs, PULSES and UP/DN, are at the screw terminals located below the quadrature encoder knob. PULSES outputs a pulse train generated by rotating the encoder shaft. It provides four pulses per one mechanical click of the encoder. UP/DN outputs a high or a low signal indicating rotation direction. When the direction is counterclockwise, UP/DN is low. When the direction is clockwise, UP/DN is high.

DAQ Devices

Typical NI multifunction I/O data acquisition (DAQ) devices have four standard elements: analog input, analog output, digital I/O, and counters.

National Instruments also makes high-speed digital devices that offer timed digital I/O, high-speed analog output devices for advanced waveform generation, and dynamic signal acquisition (DSA) devices for analyzing rapidly changing signals, such as vibration or sonar. Refer to ni.com/modularinstruments for more information.

You can transfer the signal you measure with the DAQ device to the computer through a variety of different bus structures. For example, you can use a DAQ device that plugs into the PCI bus of a computer, a DAQ device connected to the PCMCIA socket of a laptop, or a DAQ device connected to the USB port of a computer.

If you do not have a DAQ device, you can simulate one in Measurement and Automation Explorer to complete your software testing. You learn to simulate a device in the *DAQ Software Overview* section of this lesson.

You can also use PXI/CompactPCI to create a portable, versatile, and rugged measurement system. Refer to ni.com/products for more information about specific types of DAQ devices.

B. Components of a DAQ Device

Figure 2-4 shows the components of a PCI, PCI Express, PXI, or PXI Express multifunction I/O DAQ device.

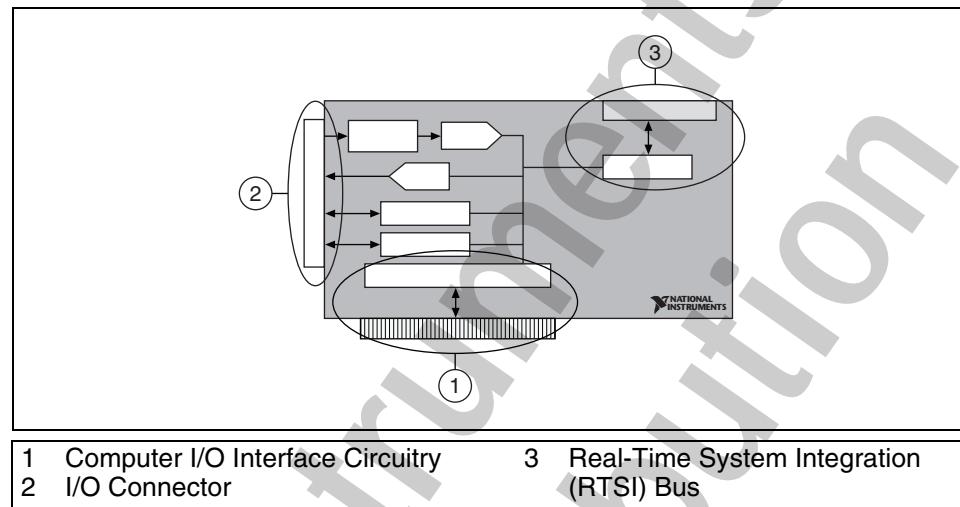


Figure 2-4. Components of a Multifunction I/O DAQ Device

Interfaces

Typical DAQ devices using a PCI, PCI Express, PXI, or PXI Express bus have three interfaces for receiving and sending signals: the I/O connector, the computer I/O interface circuitry, and the Real-Time System Integration (RTSI) Bus.

- **I/O Connector**—The I/O connector is the means by which the signal enters or leaves the DAQ device. The I/O connector has 100-, 68-, or 50-pins, depending on the device. One end of the cable connects to the I/O connector and the other end connects to the terminal block. You can find the specific pinout in the DAQ device documentation.
- **Computer I/O Interface Circuitry**—The computer I/O interface circuitry transfers information between the DAQ device and the computer. The computer I/O interface circuitry can differ depending on the bus protocol you use. For example, the PCI bus has connection leads that plug into a PCI slot, but the USB connection requires a cable.
- **RTSI Bus**—The RTSI bus shares and synchronizes signals between multiple DAQ devices in the same computer. For example, if you want two devices to perform analog input at the same rate, you can share a clock signal over the RTSI bus so both devices use the same clock signal. Use a RTSI cable to connect the devices together unless you are using the PXI platform. The PXI chassis has a backplane that the devices plug into that acts as a built-in RTSI cable to share signals among any modules in the chassis.

Analog Input Circuitry

After entering the I/O connector, the analog input signal passes through the analog input circuitry before it passes to the analog-to-digital converter (ADC). The analog input circuitry consists of a multiplexer and an instrumentation amplifier. Figure 2-5 shows details of the analog input circuitry.

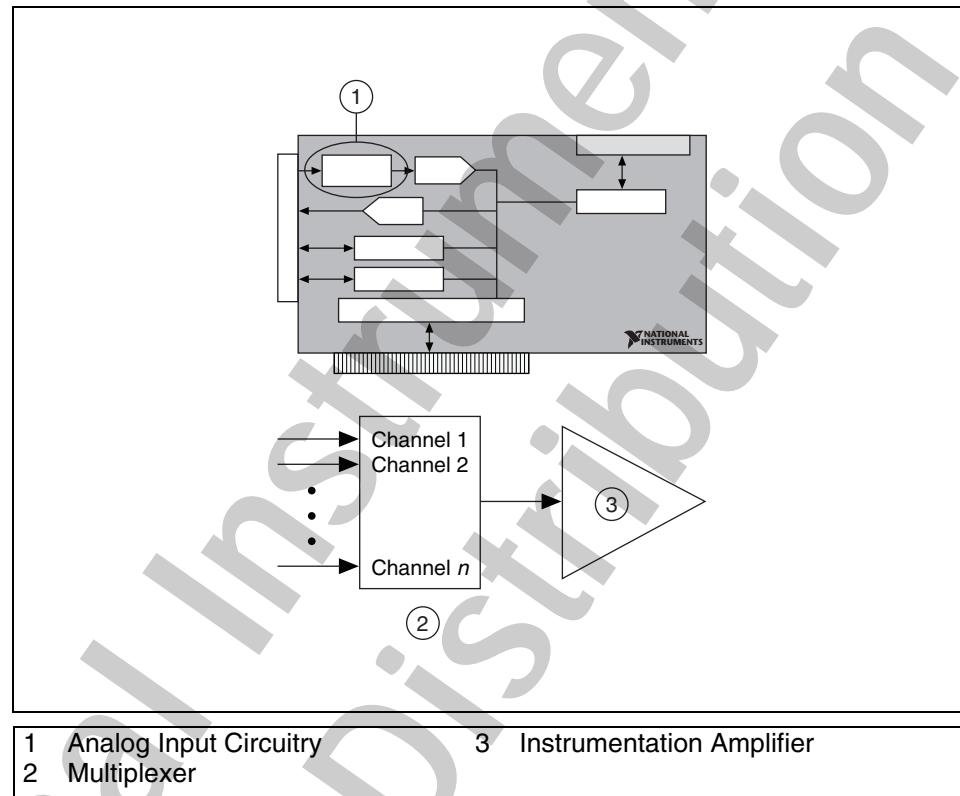


Figure 2-5. Analog Input Circuitry

- **Multiplexer**—The multiplexer, or mux, is a switch that connects only one of several input channels to the instrumentation amplifier at a time. When you acquire data from multiple channels, the mux rotates through the channels, connecting them one at a time to the amplifier. LabVIEW controls the order in which the mux connects the incoming signals to the amplifier.
- **Instrumentation Amplifier**—The instrumentation amplifier can amplify or attenuate the signal that it receives. The purpose of the amplifier is to make the signal fill the range of the ADC as much as possible. When an amplifier amplifies or attenuates the signal, it is referred to as applying a gain. The gain is the amount of amplification that is applied. For example, a gain of two amplifies the signal by two. A gain of 0.5 makes the signal two times smaller, thus attenuating the signal.

Analog-to-Digital Converter (ADC)

The ADC is an electronic device that converts an analog voltage into a digital number that you can send to the computer for interpretation using the computer I/O interface circuitry. The analog input circuitry combines with the ADC to acquire an analog signal so you can measure the level, shape, or frequency of that signal. You can use the analog input functionality of the DAQ device in applications ranging from testing a power supply to measuring a heartbeat to analyzing speech. Figure 2-6 shows an ADC.

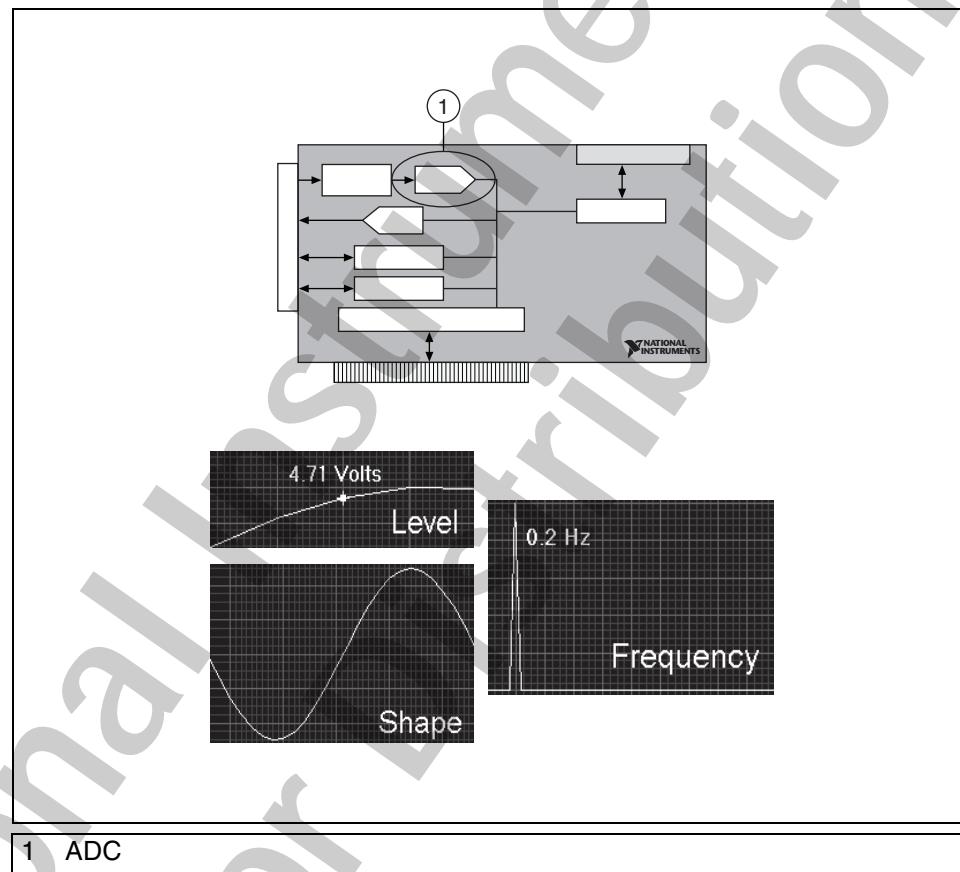


Figure 2-6. Analog-to-Digital Converter (ADC)

Analog Output Circuitry

The analog output circuitry uses a digital-to-analog converter (DAC). A DAC performs the opposite task of an ADC. It takes a digital number that was sent from the computer through the computer I/O interface circuitry and converts it into an analog signal that is output through the I/O connector.

A DAC is useful for generating DC signals (level), specific tones (frequencies), and waveforms (shapes). You can use the analog output functionality of a DAQ device in applications ranging from control systems using a proportional integral derivative (PID) control, to controlling servo motors, to generating a series of specific tones for a siren or alarm.

Figure 2-7 shows a DAC.

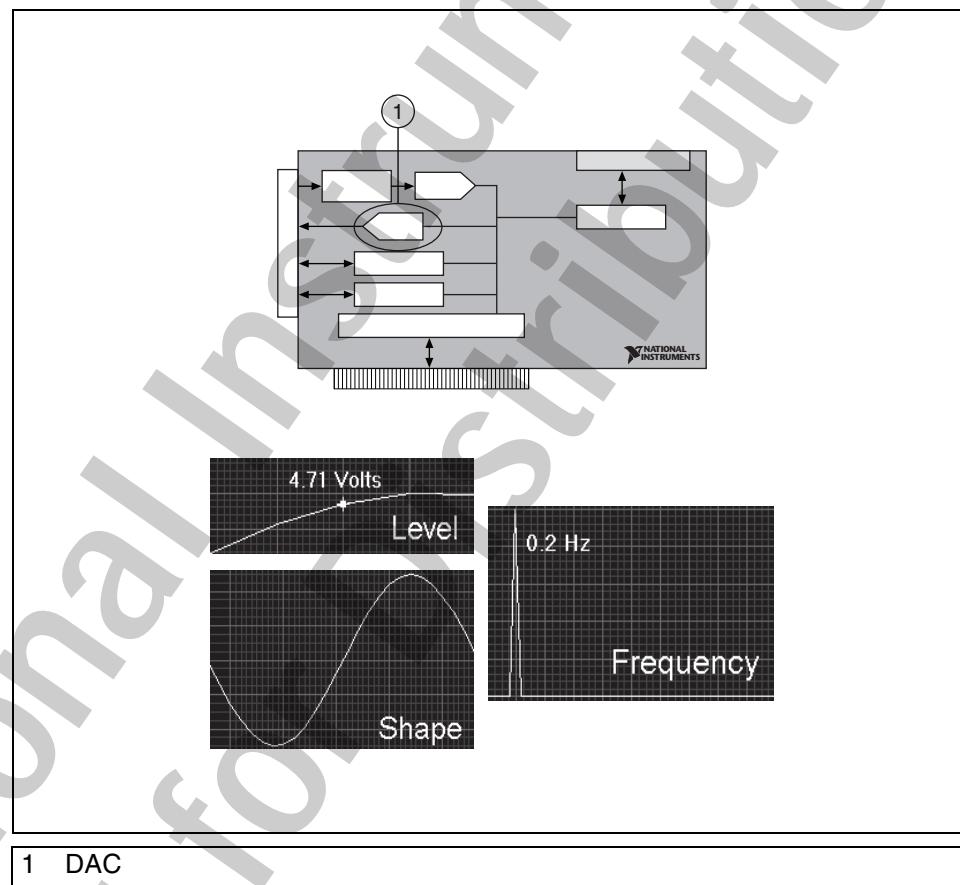


Figure 2-7. Digital-to-Analog Converter (DAC)

Digital I/O Circuitry

The digital I/O circuitry can perform both input and output functions. A typical DAQ device has multiple digital lines that can acquire or generate digital signals using software timing or hardware timing. If the DAQ device does not have a dedicated digital timing clock, it must use correlated digital input/output to achieve hardware timing. Multiple digital lines on the same DAQ device can be synchronized and clocked with an onboard or external clock.

You can use the digital I/O functionality of the DAQ device in applications ranging from monitoring a switch to see if it has changed states to controlling a relay. Figure 2-8 shows the details of digital I/O circuitry.

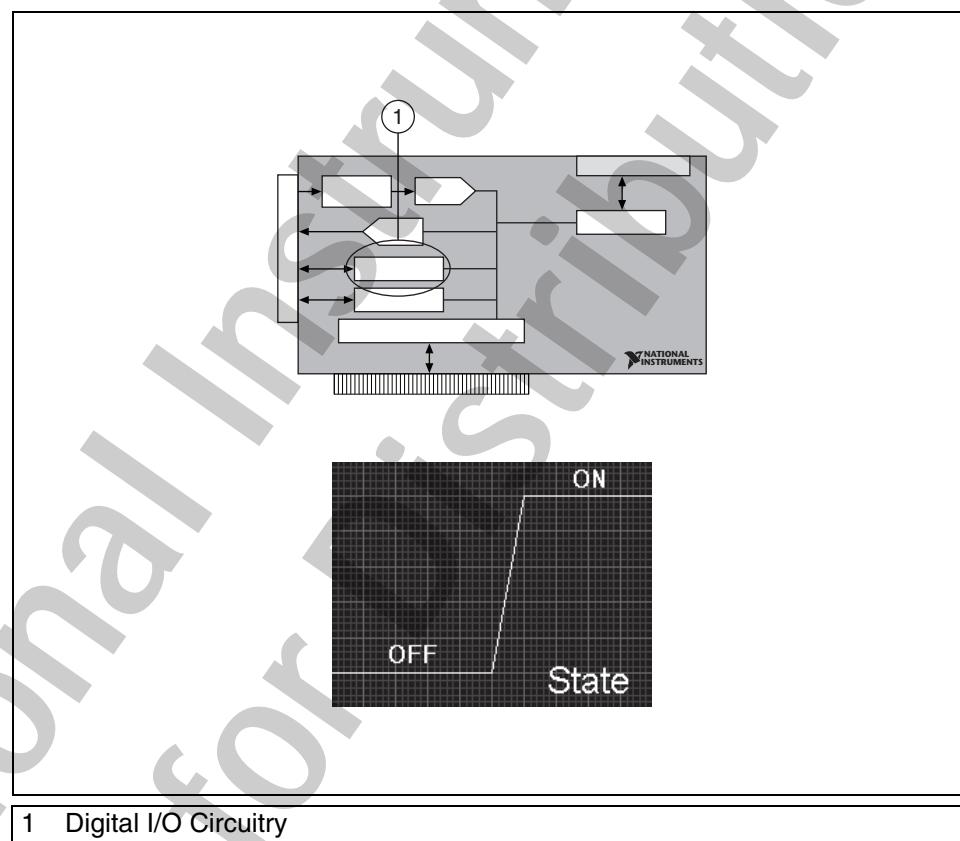


Figure 2-8. Digital I/O Circuitry

Counter Circuitry

Counters acquire and generate digital signals. Their built-in timing signals called timebases make them ideal for measuring the rate of a digital signal. You use the counter functionality of a DAQ device in applications ranging from measuring the frequency of a motor shaft to controlling stepper motors by generating a specific frequency pulse train. Figure 2-9 shows details of counter circuitry.

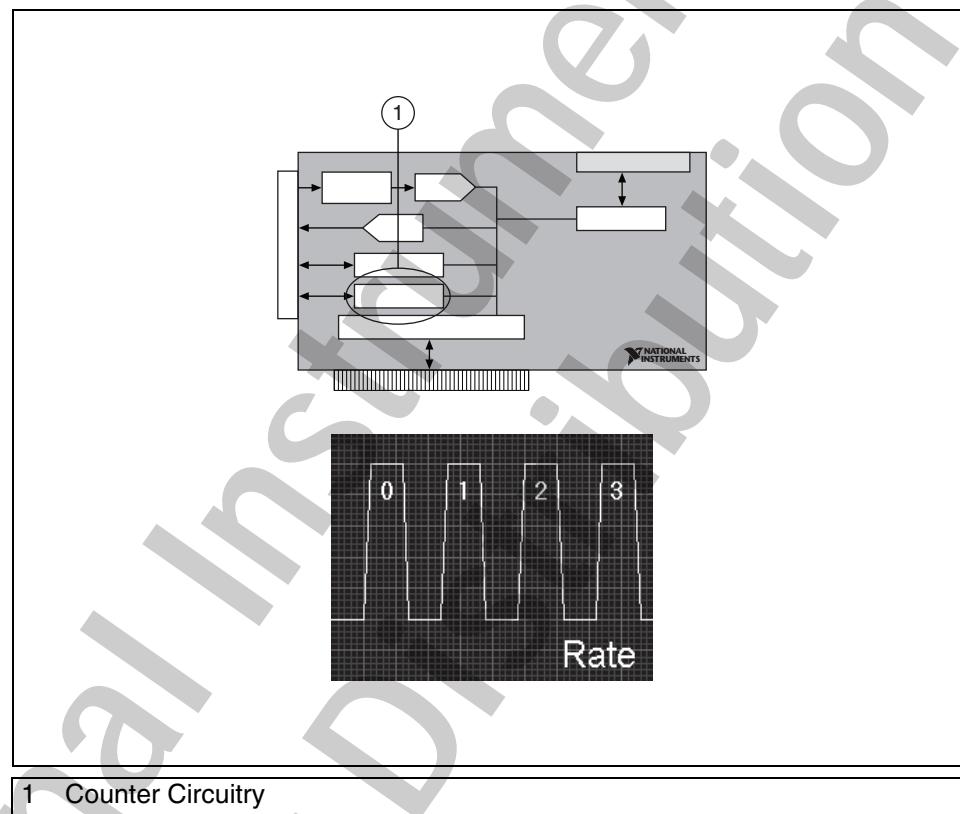


Figure 2-9. Counter Circuitry

C. Choosing Appropriate DAQ Hardware

When choosing DAQ hardware, you must determine application needs as they relate to:

- Bus of DAQ device
- Signals to measure
- Accuracy of measurements

Bus Considerations

This section examines the bus options available to you, and outlines considerations to keep in mind when choosing the right bus for your measurement application.

There are five questions to consider when trying to choose the right bus for a data acquisition application.

How much data will I be streaming across this bus?

All PC buses have a limit to the amount of data that can be transferred in a certain period of time. This is known as the bus bandwidth and is often specified in megabytes per second (MB/s). If dynamic waveform measurements are important in your application, be sure to consider a bus with enough bandwidth.

Depending on the bus that you choose, the total bandwidth can be shared among several devices or dedicated to certain devices. The PCI bus, for example, has a theoretical bandwidth of 132 MB/s that is shared among all PCI devices in the computer. Buses that offer dedicated bandwidth, such as PCI Express and PXI Express, provide the maximum data throughput per device.

When taking waveform measurements, you have a certain sampling rate and resolution that needs to be achieved based on how fast your signal is changing. You can calculate the minimum required bandwidth by taking the number of bytes per sample (rounded up to the next byte), multiplied by the sampling speed, and then multiplied by the number of channels.

Your bus bandwidth needs to be able to support the speed at which data is being acquired, and it is important to note that the actual system bandwidth will be lower than the theoretical bus limits. Actual observed bandwidth depends on the number of devices in a system and any additional bus traffic from overhead. If you need to stream a lot of data on a large number of channels, bandwidth may be the most important consideration when choosing your data acquisition bus.

What are my single-point I/O requirements?

Applications that require single-point reads and writes often depend on I/O values being updated immediately and consistently. Based on how bus architectures are implemented in both hardware and software, single-point I/O requirements could be the determining factor for the bus that you choose.

Bus latency is the responsiveness of I/O. It is the time delay between when a driver software function is called and the actual hardware value of the I/O is updated. Depending on the bus you choose, this delay could range from less than a microsecond to a few milliseconds.

In a PID control system, for example, this bus latency can directly impact the maximum speed of the control loop.

Another important factor in single-point I/O applications is determinism, which is a measure of how consistently I/O can execute on time. Buses that always have the same latency when communicating with I/O are more deterministic than buses that can vary their responsiveness. Determinism is important for control applications because it directly impacts the reliability of the control loop, and many control algorithms are designed with the expectation that the control loop will always execute at a constant rate. Any deviation from the expected rate will make the overall control system less effective and less reliable. Therefore, buses that are high in latency with poor determinism, such as USB, should be avoided when implementing closed-loop control applications.

The software-side communications bus implementation plays a big part in bus latency and determinism. Buses and software drivers that have support for real-time operating systems will provide the best determinism and therefore give you the highest performance. In general, internal buses such as PCI Express and PXI Express are better for low-latency, single-point I/O applications than external buses such as USB or wireless.

Do I need to synchronize multiple devices?

Many measurement systems have complex synchronization needs, whether it is synchronizing hundreds of input channels or multiple types of instruments. A stimulus-response system, for example, might require the output channels to share the same sample clocks and start triggers as the input channels to correlate the I/O and better analyze the results. Data acquisition devices on different buses provide different ways of accomplishing this. Almost all NI data acquisition (DAQ) devices provide access to programmable function input (PFI) lines that can be used to route clocks and triggers between different devices, and software support in NI-DAQmx to easily configure these lines. Certain buses, however, have additional timing and triggering lines built in to make multidevice synchronization as easy as possible. PCI and PCI Express devices offer the real-time system integration (RTSI) bus, on which multiple boards in a desktop system can be cabled directly together inside the case. This removes the need for additional wiring through the front connector and simplifies I/O connectivity.

The best bus option for synchronizing multiple devices is the PXI platform, including PXI and PXI Express. This open standard was specifically for high-performance synchronization and triggering, with a number of different options for synchronizing I/O modules within the same chassis, as well as synchronizing multiple chassis.

How portable should this system be?

The dramatic adoption of portable computing is undeniable, and it offers engineers and scientists new ways to innovate with PC-based data acquisition. Portability is an important factor for many applications and could easily be the primary reason to choose one bus over another.

In-vehicle data acquisition applications, for example, benefit from hardware that is compact and easily transported. External buses, like USB and Ethernet, are particularly good for portable data acquisition systems because of quick hardware installation and compatibility with laptop computers. Bus-powered USB devices provide additional convenience because they do not require a separate power supply to be present and they are powered off the USB port. Using wireless data transfer buses is another good option for portability because the measurement hardware itself becomes portable, while the computer can remain stationary.

How far will my measurements be from my computer?

The distance between the measurements you need and where the computer is located can vary drastically from application to application. To achieve the best signal integrity and measurement accuracy, you should place your data acquisition hardware as close to the signal source as possible. This can be a challenge for large distributed measurements, like those used for structural health monitoring or environmental monitoring. Running long cables across a bridge or factory floor is costly and can result in noisy signals.

One solution to this problem is to use a portable computing platform to move the entire system closer to the signal source. With wireless technology, the physical connection between the computer and the measurement hardware is removed all together, and you can take distributed measurements and send the data back to a central location.

Table 2-1. Choosing the Right Bus for Your Measurement Application

Bus	Waveform Streaming*	Single-Point I/O	Multi-Device Synchronization	Portability	Distributed I/O
PCI	132 MB/s (shared)	Best	Better	Good	Good
PCI Express	250 MB/s (per lane)	Best	Better	Good	Good
PXI	132 MB/s (shared)	Best	Best	Better	Better
PXI Express	250 MB/s (per lane)	Best	Best	Better	Better
USB	60 MB/s	Better	Good	Best	Better
Ethernet	12.5 MB/s	Good	Good	Best	Best
Wireless	6.75 MB/s	Good	Good	Best	Best

* Maximum theoretical data streaming rates based on following bus specifications: PCI, PCI Express 1.0, PXI, PXI Express 1.0, USB 2.0, 100Mbps Ethernet and Wi-Fi 802.11g.

Signal Considerations

Choose a DAQ device with an adequate number of channels, sampling rate, and input range for your applications.

Code width is the smallest change in the signal that the DAQ device can detect. Choose a DAQ device that has a small enough code width for your application. To calculate code width, you must know the resolution and device input range of the DAQ device.

Resolution

The number of bits used to represent an analog signal determines the resolution of the ADC. The resolution on a DAQ device is similar to the marks on a ruler. The more marks a ruler has, the more precise the measurements are. The higher the resolution is on a DAQ device, the higher the number of divisions into which a system can break down the ADC range, and therefore, the smaller the detectable change. A 3-bit ADC divides the range into 2^3 or eight divisions. A binary or digital code between 000 and 111 represents each division. The ADC translates each measurement of the analog signal to one of the digital divisions.

Figure 2-10 shows a 5 kHz sine wave digital image obtained by a 3-bit ADC. The digital signal does not represent the original signal adequately because the converter has too few digital divisions to represent the varying voltages of the analog signal. However, increasing the resolution to 16 bits to increase the ADC number of divisions from eight (2^3) to 65,536 (2^{16}) allows the 16-bit ADC to obtain an extremely accurate representation of the analog signal.

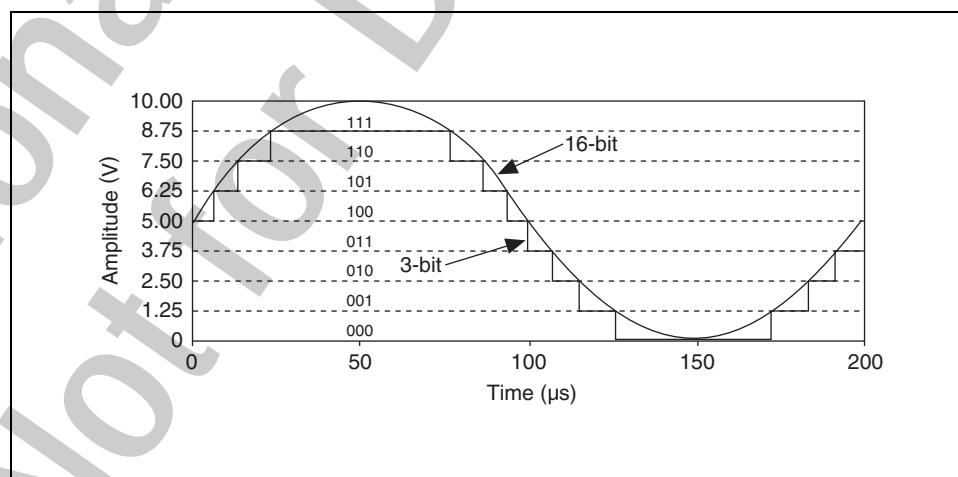


Figure 2-10. Signal Resolution

Amplification and Device Input Range

Amplification or attenuation of a signal might occur before the signal digitizes to improve the representation of the signal. By amplifying or attenuating a signal, you can effectively decrease the input range of an ADC and thus allow the ADC to use as many of the available digital divisions as possible to represent the signal.

For example, Figure 2-11 shows the effects of applying amplification to a signal that fluctuates between 0 and 5 V using a 3-bit ADC and a range of 0 to 10 V. With no amplification, or gain = 1, the ADC uses only four of the eight divisions in the conversion. By amplifying the signal by two before digitizing, the ADC uses all eight digital divisions, and the digital representation is much more accurate. Effectively, the device has an allowable input range of 0 to 5 V because any signal above 5 V when amplified by a factor of two makes the input to the ADC greater than 10 V.

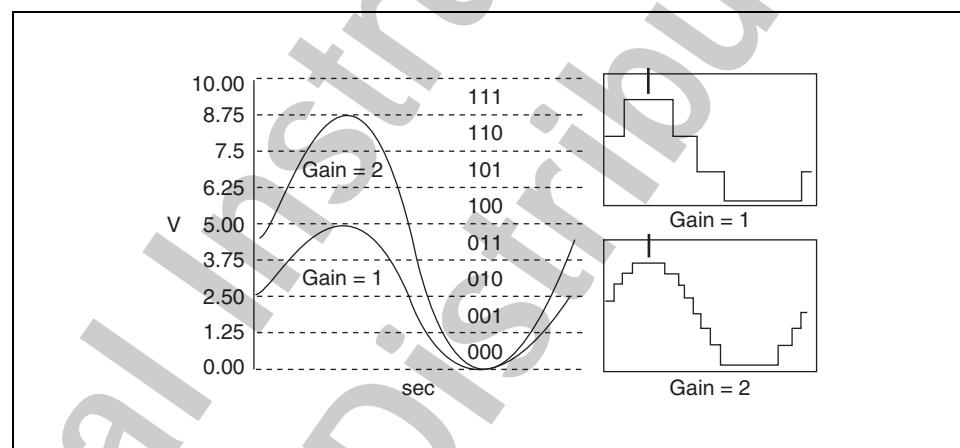


Figure 2-11. Signal with Amplification

The amplification gains available on the DAQ device determine the device input ranges available. Common device input ranges on DAQ devices include ± 10 , ± 5 , ± 2 , ± 1 , ± 0.5 , ± 0.2 , and ± 0.1 V.

You do not set the amplification gains or device input ranges directly. Instead, set your minimum and maximum expected values in software when configuring a virtual channel. The DAQ device automatically chooses which device input range to use based on your minimum and maximum settings.

The resolution and device input ranges available on a DAQ device determine the smallest detectable change in the input voltage. This change in voltage represents one least significant bit (LSB) of the digital value and is also called the code width.

Code Width

Code width is the smallest change in a signal that a system can detect. Code width is calculated using the following formula:

$$\text{code width} = \frac{\text{device input range}}{2^{\text{resolution in bits}}}$$

The smaller the code width, the more accurately a device can represent the signal. The formula confirms what you have already learned in the discussion on resolution, range, and gain:

- Larger resolution = smaller code width = more accurate representation of the signal
- Smaller device input range = smaller code width = more accurate representation of the signal

Determining the code width is important in selecting a DAQ device. For example, a 16-bit DAQ device using a -10 to 10 V device input range detects a 305 μ V change, while the same device using a -5 to 5 V device input range would detect a change of 153 μ V.

$$\frac{\text{device input range}}{2^{\text{resolution}}} = \frac{10 - (-10)}{2^{16}} = 305 \mu\text{V} \quad \frac{5 - (-5)}{2^{16}} = 153 \mu\text{V}$$

Accuracy Considerations

The result of any measurement is only an approximate estimate of the actual value being measured. In the real world, the actual value can never be perfectly measured. There is always some physical limit to how well we can measure a number. This limit is referred to as the accuracy of a measurement.

A measurement is accurate if the interval formed by the measurement and range of its uncertainty contains the true value of the quantity being measured. An ideal instrument always reports the true value; the uncertainty is zero. An instrument in the real world reports the true value with an uncertainty specified by the instrument manufacturer. This uncertainty can be viewed as the maximum error permitted by the specification. Sources of error affecting the accuracy of DAQ device include gain errors and offset errors from the amplifier and ADC, as well as noise in the system.

Use the specifications manual of the DAQ device to determine its accuracy. In the specifications manual, the Analog Input Absolute Accuracy table lists the absolute accuracy for each device input range and lists the absolute accuracy equation used to make the calculations.

D. DAQ Software Overview

The final component of a complete DAQ system is the software, as shown in Figure 1-1, *DAQ System Overview*. The computer receives raw data through the DAQ device. The application you write presents and manipulates the raw data in a form you can understand. The software also controls the DAQ system by controlling the DAQ device and specifying when and from which channels to acquire data. The DAQ software thus acts as a single programming interface for programming the analog input, analog output, digital I/O, and counter/timers on hundreds of multifunction DAQ hardware devices.

Typically, DAQ software includes drivers and application software. Drivers are unique to the device or type of device and include the set of commands the device accepts. Application software, such as LabVIEW, sends the drivers commands, such as acquire, and returns a reading. The application software also displays and analyzes the acquired data.

NI measurement devices include NI-DAQmx driver software—a collection of VIs you use to configure, acquire data from, and send data to the measurement devices.

A measurement system consists of the following software applications:

- NI-DAQmx—Software that controls the DAQ device.
- Measurement & Automation Explorer (MAX)—Software tool for testing and configuring hardware.
- LabVIEW—Software that you use to create an application to send commands to the driver and acquire, analyze, and present data.

NI-DAQmx

You use NI-DAQmx software to communicate with National Instruments DAQ devices. NI-DAQmx provides a user interface and set of tools for programming and configuring your DAQ device. NI-DAQmx includes the following advantages over previous versions of NI-DAQ:

- The DAQ Assistant is an interactive wizard for configuring NI-DAQmx measurement tasks, channels, and scales for use in LabVIEW. Use the DAQ Assistant to generate NI-DAQmx code to run tasks and channels, or to deploy NI-DAQmx code to another DAQ system. Use LabVIEW or MAX to launch the DAQ Assistant.
- Increased performance, including faster single-point analog I/O and multithreading.
- A simpler API for creating DAQ applications using fewer functions and VIs than earlier versions of NI-DAQ.

- Expanded functionality in LabVIEW, including Property Nodes for data acquisition and improved waveform data type support for both analog and digital I/O.
- Similar API and functionality for ANSI C, LabWindows™/CVI™, and Measurement Studio, including native .NET and C++ interfaces.



Note You cannot use NI-DAQmx with third-party DAQ devices. Contact the vendor of the third-party device to obtain a driver specific to that device.

NI-DAQmx is compatible with the following software applications and programming languages:

- National Instruments LabVIEW
- National Instruments LabWindows/CVI
- National Instruments Measurement Studio
- Microsoft Visual C/C++
- Microsoft C# .NET
- Microsoft Visual Basic .NET
- Microsoft Visual Basic 6.0
- ANSI C

Refer to the *NI-DAQmx Readme* for more information about the compatibility of NI-DAQmx with specific versions of these languages.

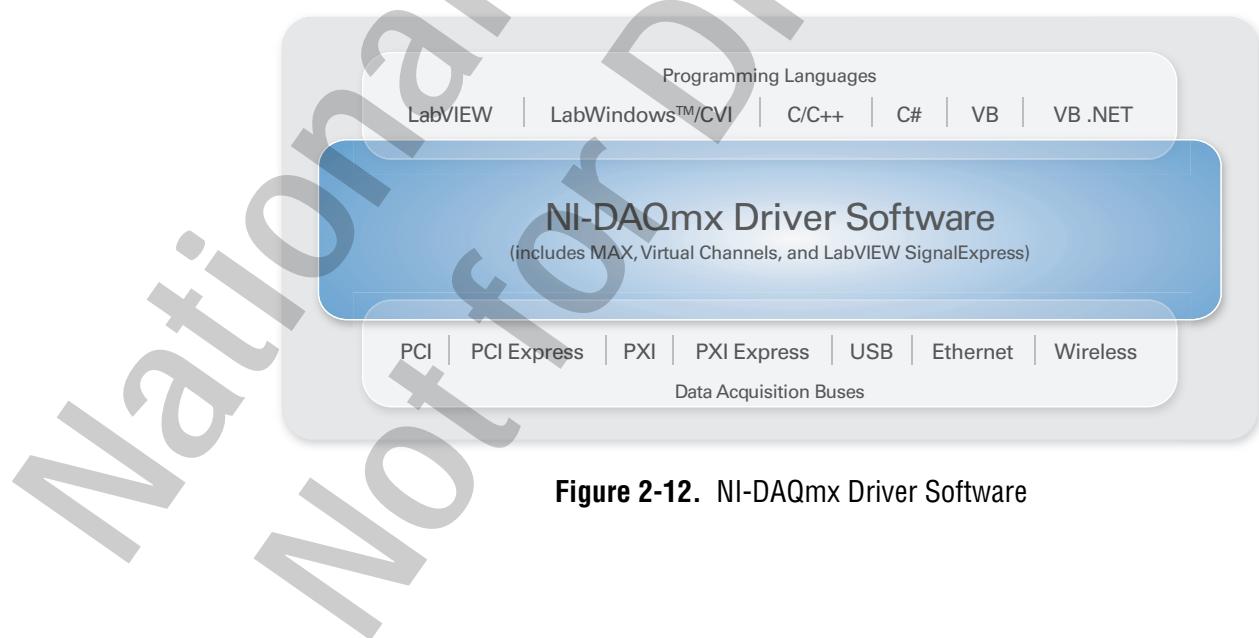


Figure 2-12. NI-DAQmx Driver Software

Measurement & Automation Explorer

Measurement & Automation Explorer (MAX) is a Windows-based application installed during NI-DAQmx installation. Use MAX to configure and test NI software and hardware, add new channels and interfaces, execute system diagnostics, and view the devices and instruments you have connected to your system. You must use MAX to configure devices when programming with NI-DAQmx. Double-click the Measurement & Automation icon on the desktop to launch MAX. MAX includes the following function categories:

- Data Neighborhood
- Devices and Interfaces
- Historical Data
- Scales
- Software
- VI Logger Tasks
- IVI Drivers
- Remote Systems

Data Neighborhood

Data Neighborhood provides access to descriptively named shortcuts to configure physical channels in your system, including DAQ virtual channels and tasks. The Data Neighborhood category also provides utilities for testing and reconfiguring those virtual channels. You can also access the DAQ Assistant from Data Neighborhood to create and configure settings for virtual channels and tasks.

The DAQ Assistant is an interactive wizard for building and configuring your measurement channels and tasks.

- **Channel**—An NI-DAQmx channel maps configuration information such as scaling and input limits to a specified physical channel. You can set up the configuration information for the channel and give the channel a descriptive name at the same time. Later, you can use the descriptive name to access that channel and its configuration in a task or LabVIEW. You can give the channel a description, decide the type of transducer the channel uses, set the range (determines gain), choose the grounding mode, assign custom scaling for the virtual channel, and give the channel a descriptive name to replace the channel number all at the same time.

For example, assume channel 0 is wired to a temperature sensor. You can create a virtual channel for channel 0 and call it Temperature Sensor. You can create virtual channels for analog I/O, counter I/O, and digital I/O. In this case, referring to a channel by a name (Temperature Sensor) instead of a number (0) helps you remember what the channel does.

- **Task**—An NI-DAQmx task is a collection of one or more virtual channels with the same timing and triggering. Conceptually, a task represents a measurement or a generation that you want to perform. The channels that compose the task can be used in multiple tasks (global channel) or assigned to just one specific task (local channel). You can also create new channels while creating a task or you can compose a task with channels that have been created using the DAQ Assistant.

Devices and Interfaces

The Devices and Interfaces category lists installed and detected NI hardware. The Devices and Interfaces category also includes the Self-Test, Test Panels, Reset, Properties, and Self-Calibrate utilities for configuring and testing devices.

The Self-Test utility runs an internal test on a DAQ device to ensure that all resources are properly assigned and that the device is configured correctly.

The Test Panel utility tests the analog I/O, digital I/O, and counter I/O functionality of a DAQ device. Use the Test Panel utility to troubleshoot the functionality of the device and system configuration directly from NI-DAQmx. If the device does not work in the Test Panel, it will not work in LabVIEW. If you experience problems with data acquisition in a LabVIEW program, run the Self-Test and Test Panel utilities to begin troubleshooting.

The Reset utility resets the DAQ device to its default state.

The Properties utility allows you to configure and view the RTSI configuration and accessory devices that are used with your DAQ device. The system resources for the device, such as the memory range and IRQ level, are listed in the Attributes tab in the window to the right of the Configuration window in MAX.

The Self-Calibrate utility performs an internal calibration of the DAQ device.

Scales

The Scales category lists all the currently configured custom scales and provides utilities for testing and reconfiguring those custom scales. Scales also provides access to the DAQ Assistant, which allows you to create new custom scales.

Use the DAQ Assistant to create custom scales you can use to determine scaling information for existing virtual channels. Each custom scale can have its own name and description to help you identify it. A custom scale can be one of the following four types:

- **Linear**—Scales that use the formula $y = mx + b$.
- **Map Ranges**—Scales in which values are scaled proportionally from a range of raw values to a range of scaled values.
- **Polynomial**—Scales that use the formula $y = a_0 + (a_1 \times x) + (a_2 \times x^2) + \dots + (a_n \times x^n)$.
- **Table**—Scales in which you enter the raw value and the corresponding scaled value in a table format.

Software

The Software category shows all the currently installed NI software. The icon for each software package is also a shortcut that you can use to launch the software. The Software category also includes a Software Update Agent. The purpose of the Software Update Agent is to check if the installed National Instruments software is the latest version. If the software is not the latest version, the Software Update Agent opens the Web page on ni.com to download the latest version of the software.

Software Architecture for Windows

The main component of NI-DAQmx, the `nidaq32.dll`, makes function calls directly to a DAQ device. The function that the `nidaq32.dll` performs depends on where you access it from. Both MAX and LabVIEW can communicate with NI-DAQmx. MAX primarily configures and tests the DAQ device. MAX not only helps configure devices, but it also lists the devices that are present in the system. To do this, MAX must communicate with the Windows Device Manager and the Windows Registry.

Simulating a DAQ Device

You can create NI-DAQmx simulated devices in NI-DAQmx 7.4 or later. Using NI-DAQmx simulated devices, you can try NI products in your application without the hardware. A simulated NI-DAQmx device is a replica of a real device created using the **NI-DAQmx Simulated Device** option in the Create New menu of MAX for the purpose of operating a function or program without hardware. Its driver is loaded, and programs using it are fully verified. When you later acquire the hardware, you can import the NI-DAQmx simulated device configuration to the physical device using the MAX Portable Configuration Wizard. With NI-DAQmx simulated devices, you also can export a physical device configuration onto a system that does not have the physical device installed. Then, using the NI-DAQmx simulated device, you can work on your applications on a portable system and upon returning to the original system, you can easily import your application work. Additional information about NI-DAQmx simulated devices is available in the *NI-DAQmx Help* and the *DAQ Getting Started Guide*.

Creating NI-DAQmx Simulated Devices

To create an NI-DAQmx simulated device, complete the following steps:

1. Right-click **Devices and Interfaces** and select **Create»New**.
2. A dialog box prompts you to select a device to add. Select **NI-DAQmx Simulated Device** and click **Finish**.
3. In the **Choose Device** dialog box, select the family of devices for the device you want to simulate.
4. Select the device and click **OK**. In the configuration tree in MAX, the icons for NI-DAQmx simulated devices are yellow. The icons for physical devices are green.
5. If you select a PXI device, you are prompted to select a chassis number and PXI slot number.
6. If you select an SCXI chassis, the SCXI configuration panels open.

Removing NI-DAQmx Simulated Devices

To remove an NI-DAQmx simulated device, complete the following steps:

1. Expand **Devices and Interfaces»NI-DAQmx Devices**.
2. Right-click the NI-DAQmx simulated device you want to delete.
3. Click **Delete**.



Note In the MAX configuration tree diagram shown in Figure 2-13, the icons for NI-DAQmx simulated devices are yellow. The icons for physical devices are green.

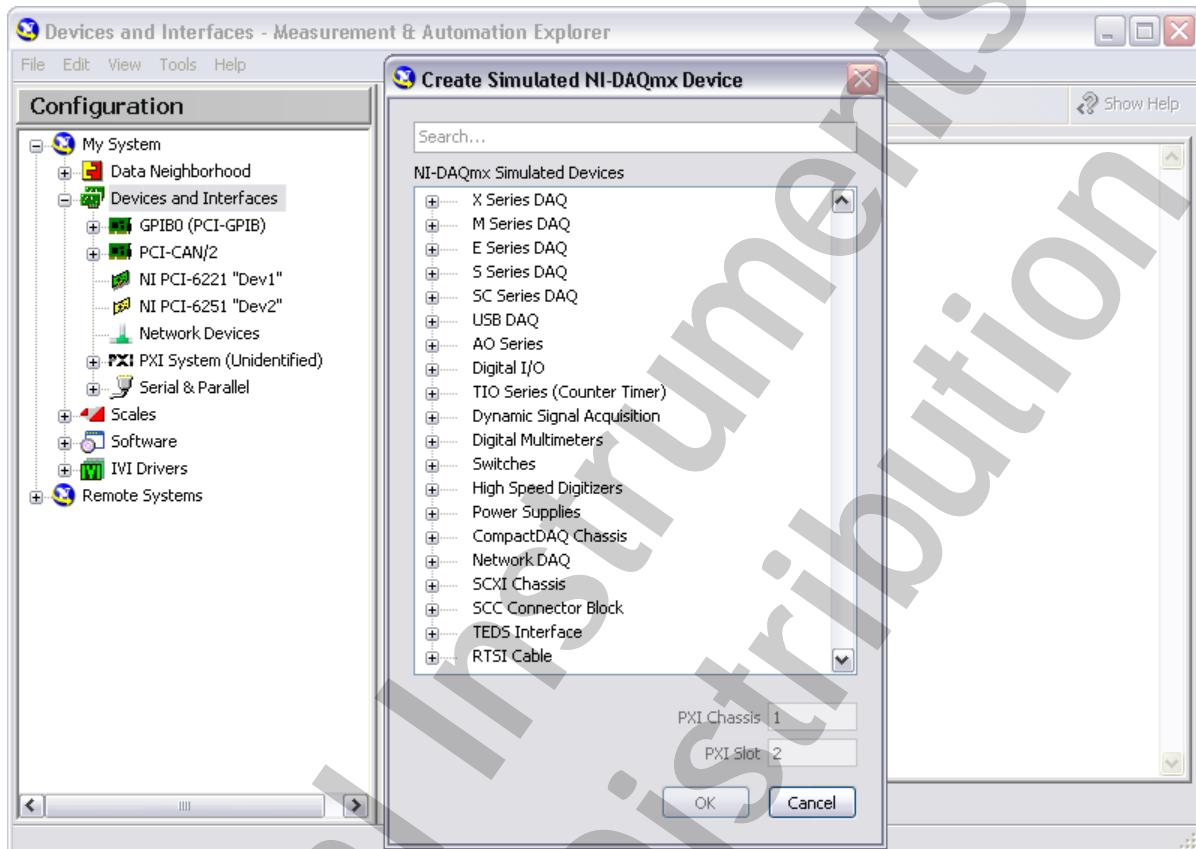


Figure 2-13. Creating NI-DAQmx Simulated Devices

E. Overview of NI-DAQmx VIs

Now that you have an understanding of sensors, signals, DAQ device configuration, and MAX, you can begin learning to use LabVIEW to develop a data acquisition application. NI-DAQmx supports software packages other than LabVIEW, but this course describes only LabVIEW development of DAQ applications.

DAQmx Name Controls

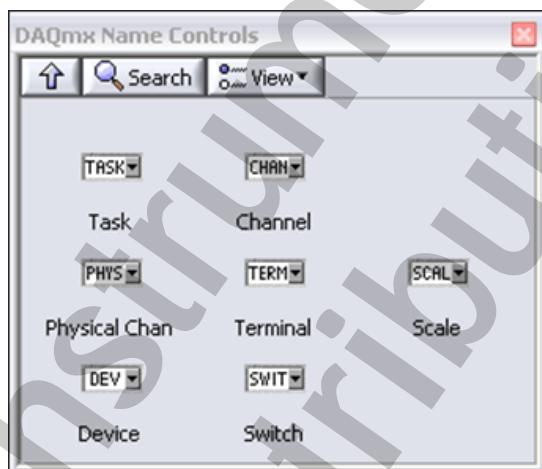


Figure 2-14. DAQmx Control Palette

The **DAQmx Name Controls** palette includes controls for the task name, channel name, physical channel, terminal, scale name, device number, and switch. Because you typically configure tasks and channels in MAX prior to LabVIEW programming, you typically use only the Task Name and Channel Name controls. The values for the physical channel, terminal, scale name, and device number are configured in MAX when you create and configure a task or channel. The currently configured tasks and channels automatically populate the menu rings for the DAQmx Task Name and DAQmx Channel Name controls.

DAQmx - Data Acquisition VIs

Use the NI-DAQmx VIs with NI-DAQ hardware devices to develop instrumentation, acquisition, and control applications. Refer to the *DAQ Getting Started Guide* or the *NI-DAQmx Readme* for a complete listing of devices that NI-DAQmx supports.

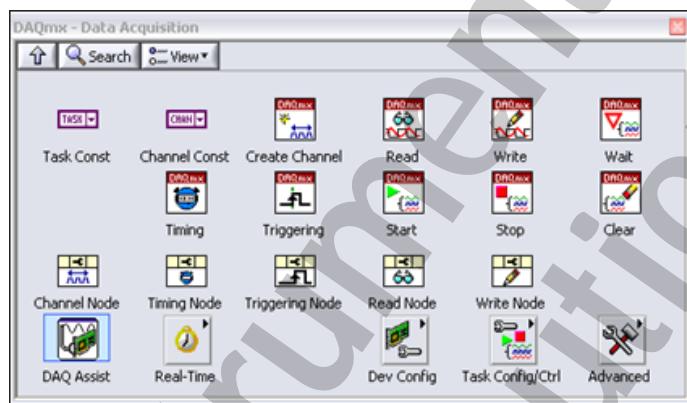


Figure 2-15. DAQmx Palette

The **DAQmx - Data Acquisition** palette includes the following constants, VIs, Property Nodes, and subpalettes:

Constants

- DAQmx Task Name Constant—Lists all tasks you create and save using the DAQ Assistant. Right-click the constant and select **I/O Name Filtering** from the shortcut menu to limit the tasks the constant displays and to limit what you can enter in the constant.
- DAQmx Global Channel Constant—Lists all virtual channels you create and save by using the DAQ Assistant. Click and select **Browse** to select multiple channels. Right-click the constant and select **I/O Name Filtering** from the shortcut menu to limit the channels the constant displays and to limit what you can enter in the constant.

VIs



- DAQmx Create Virtual Channel VI—Creates a virtual channel or set of virtual channels and adds them to a task. The instances of this polymorphic VI correspond to the I/O type of the channel, such as analog input, digital output, or counter output; the measurement or generation to perform, such as temperature measurement, voltage generation, or event counting; and in some cases, the sensor to use, such as a thermocouple or RTD for temperature measurements.

This function sets the same settings that you configure in MAX when creating a virtual channel. Use this function if the operator of your program might periodically change the physical channel connection, but

not other important settings, such as terminal configuration mode or the custom scale applied. Use the physical channel pull-down menu to specify the device number of the DAQ device and the actual physical channel your signal is connected to.



- DAQmx Read VI—Reads samples from the task or channels you specify. The instances of this polymorphic VI specify what format of samples to return, whether to read a single sample or multiple samples at once, and whether to read from one or multiple channels. You can select an instance by right clicking on the DAQmx Read VI and choose **Select Type** to include additional configuration options for read operations.



- DAQmx Write VI—Writes samples to task or channels you specify. The instances of this polymorphic VI specify the format of the samples to write, whether to write one or multiple samples, and whether to write to one or multiple channels. You can select an instance by right clicking on the DAQmx Write VI and choosing **Select Type** to include additional configuration options for write operations



- DAQmx Wait Until Done VI—Waits for the measurement or generation to complete. Use this VI to ensure that the specified operation is complete before you stop the task.



- DAQmx Timing VI—Configures number of samples to acquire or generate and creates a buffer when needed. The instances of this polymorphic VI correspond to the type of timing to use on the task.



- DAQmx Trigger VI—Configures triggering for the task. The instances of this polymorphic VI correspond to the trigger and trigger type to configure.



- DAQmx Start Task VI—Transitions the task to the running state to begin the measurement or generation. Using this VI is required for some applications and is optional for others.



- DAQmx Stop Task VI—Stops the task and returns it to the state the task was in before you used the DAQmx Start Task VI or used the DAQmx Write VI with the autostart input set to TRUE.



- DAQmx Clear Task VI—Clears the task. Before clearing, this VI stops the task, if necessary, and releases any resources reserved by the task. You cannot use a task after you clear it unless you recreate the task.

Property Nodes



- DAQmx Channel Property Node—A Property Node with the DAQmx Channel class preselected. Right-click the Property Node and choose **Select Filter** from the shortcut menu to make the Property Node show only the properties supported by a particular device installed in the system or supported by all the devices installed in the system.



- DAQmx Timing Property Node—A Property Node with the DAQmx Timing class preselected. Right-click the Property Node and choose **Select Filter** from the shortcut menu to make the Property Node show only the properties supported by a particular device installed in the system or supported by all the devices installed in the system.



- DAQmx Trigger Property Node—A Property Node with the DAQmx Trigger class preselected. Right-click the Property Node and choose **Select Filter** from the shortcut menu to make the Property Node show only the properties supported by a particular device installed in the system or supported by all the devices installed in the system.



- DAQmx Read Property Node—A Property Node with the DAQmx Read class preselected. Right-click the Property Node and choose **Select Filter** from the shortcut menu to make the Property Node show only the properties supported by a particular device installed in the system or supported by all the devices installed in the system.



- DAQmx Write Property Node—A Property Node with the DAQmx Write class preselected. Right-click the Property Node and choose **Select Filter** from the shortcut menu to make the Property Node show only the properties supported by a particular device installed in the system or supported by all the devices installed in the system.



- DAQ Assistant Express VI—Creates, edits, and runs tasks using NI-DAQmx. Refer to the *DAQ Getting Started Guide* for a complete listing of devices NI-DAQmx supports.

Subpalettes

- Use the DAQmx Real-Time VIs and functions located on the **DAQmx Real-Time** palette to configure and perform real-time operations.
- Use the DAQmx Device Configuration VIs and functions located on the **DAQmx Device Configuration** palette for hardware specific configuration and control.
- Use the DAQmx Advanced Task Options VIs and functions located on the **DAQmx Advanced Task Options** palette for advanced configuration and control of tasks.
- Use the DAQmx Advanced VIs and Functions located on the **DAQmx Advanced** palette to access advanced and miscellaneous features of NI-DAQmx.

Summary

- Typical DAQ hardware consists of a terminal block, a cable, and a DAQ device.
- Typical DAQ devices have connectors, ADC, DAC, digital I/O lines, and counters.
- Code width is the smallest voltage change a DAQ device can detect based on resolution and device input range.
- LabVIEW communicates with DAQ devices using NI-DAQmx software.
- NI-DAQmx simulation simulates the NI-DAQmx hardware.
- You can use MAX to access the DAQ Assistant, view test panels, and obtain software updates to help configure and test the system.
- You can use the DAQ Assistant from MAX or LabVIEW to create and edit virtual channels, tasks, and custom scales.
- The **DAQmx - Data Acquisition** palette contains all the VIs necessary to perform customized data acquisition and generation.

National Instruments
Not for Distribution

Self-Review: Quiz

1. Which of the following are components of a DAQ device?
 - a. Analog Input Circuitry
 - b. Data Transfer Bus
 - c. RAM
 - d. Counter Circuitry
 - e. Onboard FIFOs

2. All DAQmx VIs and property nodes are accessible through the NI-DAQmx palette.
 - a. True
 - b. False

3. Code width defines how close your measurement is to its true value.
 - a. True
 - b. False

National Instruments
Not for Distribution

Self-Review: Quiz Answers

1. Which of the following are components of a DAQ device?
 - a. **Analog Input Circuitry**
 - b. **Data Transfer Bus**
 - c. RAM
 - d. Counter Circuitry
 - e. Onboard FIFOs

2. All DAQmx VIs and property nodes are accessible through the NI-DAQmx palette.
 - a. **True**
 - b. False

3. Code width defines how close your measurement is to its true value.
 - a. True
 - b. **False**

Notes

National Instruments
Not for Distribution

Analog Input

This lesson describes decisions you must make when sampling an analog input signal and the LabVIEW features that you use specifically with NI-DAQmx VIs. This lesson also describes the theory and concepts of analog and digital triggering.

Topics

- A. Grounding Issues
- B. Sampling Considerations
- C. Single Sample Software-Timed Acquisition
- D. DAQ Device Architectures
- E. Finite Buffered Acquisition
- F. Continuous Buffered Acquisition
- G. Triggering

A. Grounding Issues

To produce accurate and noise-free measurements, knowledge of the nature of the signal source, a suitable configuration of the DAQ device, and an appropriate cabling scheme might be required. Figure 3-1 shows a typical data acquisition system. The integrity of the acquired data depends on the entire analog signal path.

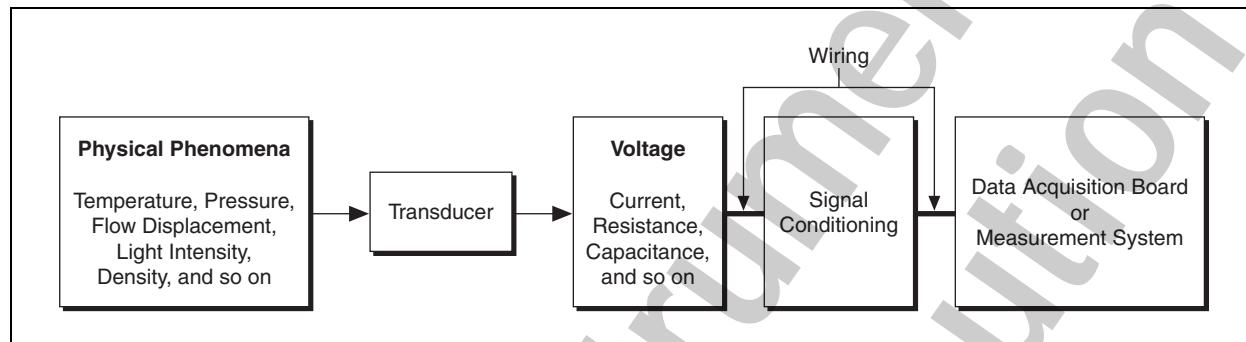


Figure 3-1. Typical DAQ System

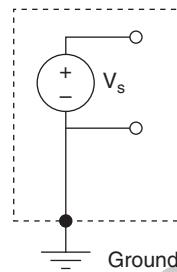
In order to cover a wide variety of applications, most DAQ devices provide some flexibility in their analog input stage configuration. This flexibility can lead to confusion about the proper applications of the various input configurations and their relative merits. The purpose of this part of the course is to help clarify the types of input configurations available on DAQ devices and to explain how to choose and use the configuration best suited for the application. An understanding of the types of signal sources and measurement systems is a prerequisite to apply good measurement techniques.

Types of Signal Sources

Analog input acquisitions use grounded and floating signal sources.

Grounded Signal Sources

A grounded source is one in which the voltage signals are referenced to a system ground, such as the earth or a building ground, as shown in Figure 3-2.

**Figure 3-2.** Grounded Signal Source

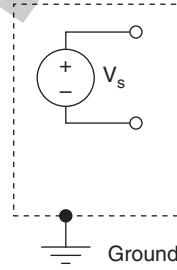
Grounded sources use the system ground, therefore, they share a common ground with the measurement device. The most common examples of grounded sources are devices that plug into a building ground through wall outlets, such as signal generators and power supplies.



Note The grounds of two independently grounded signal sources generally are not at the same potential. The difference in ground potential between two instruments connected to the same building ground system is typically 10 mV to 200 mV. The difference can be higher if power distribution circuits are not properly connected. This causes a phenomenon known as a ground loop.

Floating Signal Sources

In a floating signal source, the voltage signal is not referenced to any common ground, such as the earth or a building ground, as shown in Figure 3-3.

**Figure 3-3.** Floating Signal Source

Some common examples of floating signal sources are batteries, thermocouples, transformers, and isolation amplifiers. Notice in the illustration of the floating signal source in Figure 3-3 that neither terminal of the source is connected to the electrical outlet ground as in Figure 3-2, which depicts a grounded signal source. Each terminal is independent of the system ground.

Types of Measurement Systems

The most common electrical equivalent produced by signal conditioning circuitry associated with transducers is in the form of voltage.

Transformation to other electrical phenomena, such as current and frequency, may be encountered in cases where the signal is to be carried over long cable distances in harsh environments. Virtually in all cases, the transformed signal is ultimately converted back into a voltage signal before measurement. Therefore, it is important to understand the voltage signal source.

A voltage signal is measured as the potential difference across two points, as shown in Figure 3-4.

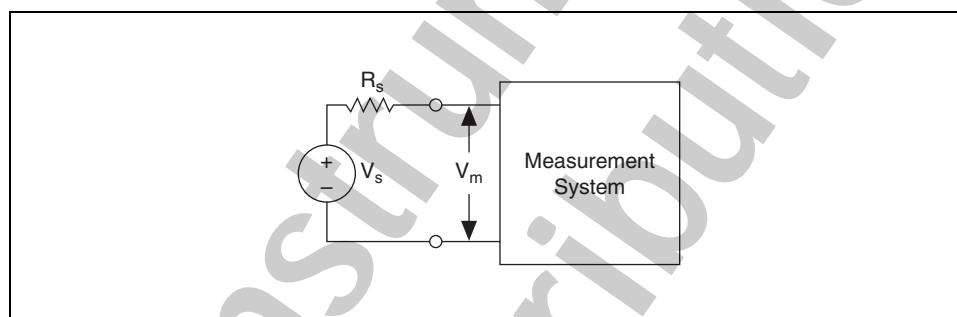


Figure 3-4. Measurement System

You configure a measurement system based on the hardware you use and the measurement you take.

Differential Measurement System

Differential measurement systems are similar to floating signal sources in which you make the measurement with respect to a floating ground that is different from the measurement system ground. Neither of the inputs of a differential measurement system are tied to a fixed reference, such as the earth or a building ground. Handheld, battery-powered instruments and DAQ devices with instrumentation amplifiers are examples of differential measurement systems.

A typical National Instruments device uses an implementation of an eight-channel differential measurement system as shown in Figure 3-5. Using analog multiplexers in the signal path increases the number of measurement channels when only one instrumentation amplifier exists. In Figure 3-5, the analog input ground (AIGND) pin is the measurement system ground.

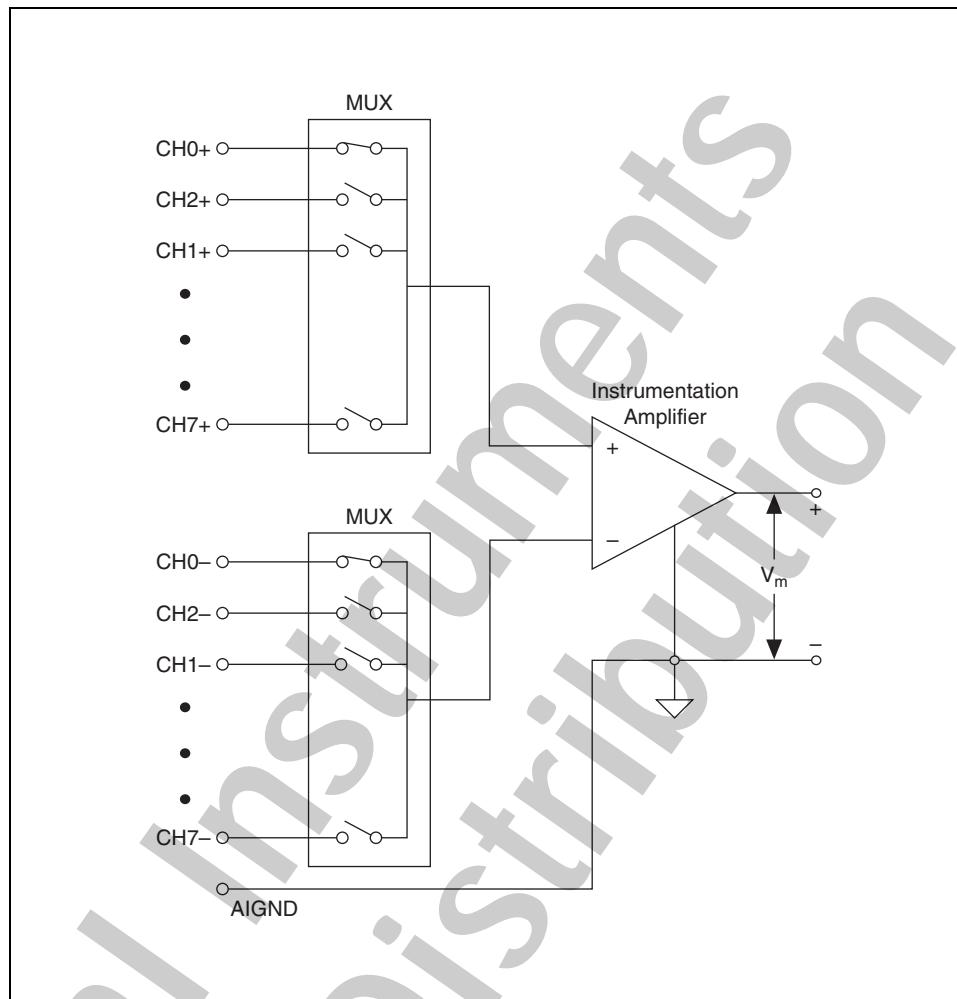


Figure 3-5. Eight-channel Differential Measurement System

An ideal differential measurement system responds only to the potential difference between its two terminals—the positive (+) and negative (−) inputs. A common-mode voltage is any voltage which you measure with respect to the instrumentation amplifier ground present at both amplifier inputs. An ideal differential measurement system completely rejects, or does not measure, common-mode voltage. Rejecting common-mode voltage is useful because unwanted noise often is introduced as common-mode voltage in the circuit that makes up the cabling system of a measurement system.

However, several factors, such as the common-mode voltage range and common-mode rejection ratio (CMRR) parameters, limit the ability of practical, real-world differential measurement systems to reject the common-mode voltage.

Common-Mode Voltage

The common-mode voltage range limits the allowable voltage range on each input with respect to the measurement system ground. Violating this constraint results not only in measurement error, but also in possible damage to components on the device. The following equation defines common-mode voltage (V_{cm}):

$$V_{cm} = (V_+ + V_-) / 2$$

where V_+ is the voltage at the noninverting terminal of the measurement system with respect to the measurement system ground, and V_- is the voltage at the inverting terminal of the measurement system with respect to the measurement system ground.

Common-Mode Rejection Ratio

CMRR measures the ability of a differential measurement system to reject the common-mode voltage signal. The CMRR is a function of frequency and typically reduces with frequency. The higher the CMRR, the better the amplifier can extract differential signals in the presence of common-mode noise. Using a balanced circuit can optimize the CMRR. Most DAQ devices specify the CMRR up to the power line frequency, which is 60 Hz. The following equation defines CMRR in decibels (dB):

$$\text{CMRR(dB)} = 20 \log \left(\frac{\text{Differential Gain}}{\text{Common-Mode Gain?}} \right)$$

Figure 3-6 shows a simple circuit in which CMRR in dB is measured as:

$$20 \log \frac{V_{cm}}{V_{out}}$$

where $V_+ + V_- = V_{cm}$

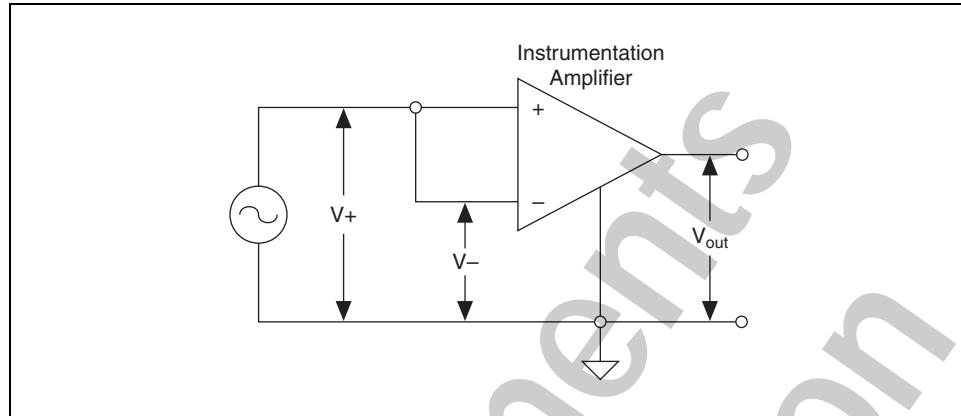


Figure 3-6. Common-mode Voltage Illustration

Referenced and Non-Referenced Single-Ended Measurement Systems

Referenced and non-referenced single-ended measurement systems are similar to grounded sources in that you make the measurement with respect to a ground. A ground referenced single-ended (GRSE) or referenced single-ended (RSE) measurement system measures voltage with respect to the ground, AIGND, which is directly connected to the measurement system ground. Figure 3-7 shows an eight-channel referenced single-ended measurement system.

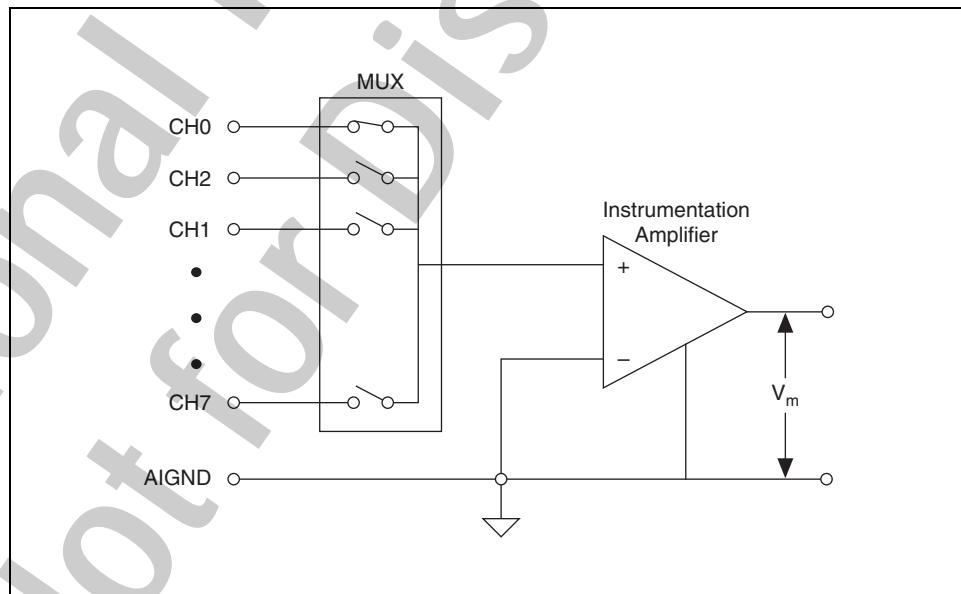


Figure 3-7. Eight-channel Referenced Single-Ended Measurement System

DAQ devices often use a non-referenced single-ended (NRSE) measurement technique, or pseudodifferential measurement, which is a variant of the referenced single-ended measurement technique. Figure 3-8 shows a NRSE measurement system.

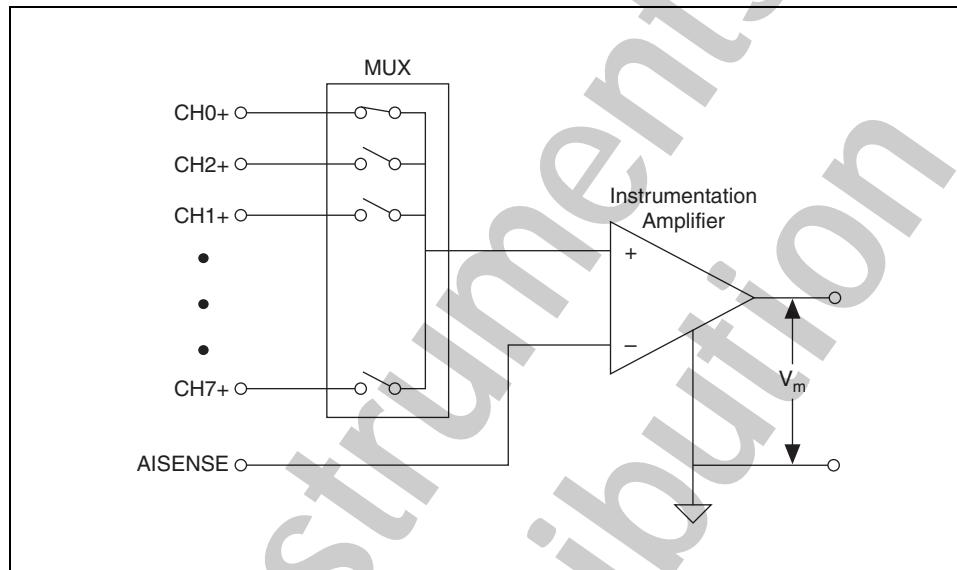


Figure 3-8. NRSE Measurement System

In a NRSE measurement system, all measurements are still made with respect to a single-node analog input sense (AISENSE on M Series devices), but the potential at this node can vary with respect to the measurement system ground (AIGND). A single-channel NRSE measurement system is the same as a single-channel differential measurement system.

Measuring Signal Sources

There are two types of signal sources—grounded signal sources and floating non-referenced signal sources.

Measuring Grounded Signal Sources

A grounded signal source is best measured with a differential measurement system. Figure 3-9 shows the pitfall of using a ground-referenced measurement system to measure a grounded signal source. In this case, the measured voltage, V_m , is the sum of the signal voltage, V_s , and the potential difference, ΔV_g , that exists between the signal source ground and the measurement system ground. This potential difference is generally not a DC level. A noisy measurement system results, often showing power-line frequency (60 Hz) components in the readings. Ground-loop introduced noise may have both AC and DC components, introducing offset errors and noise in the measurements. The potential difference between the two grounds causes a ground-loop current to flow in the interconnection.

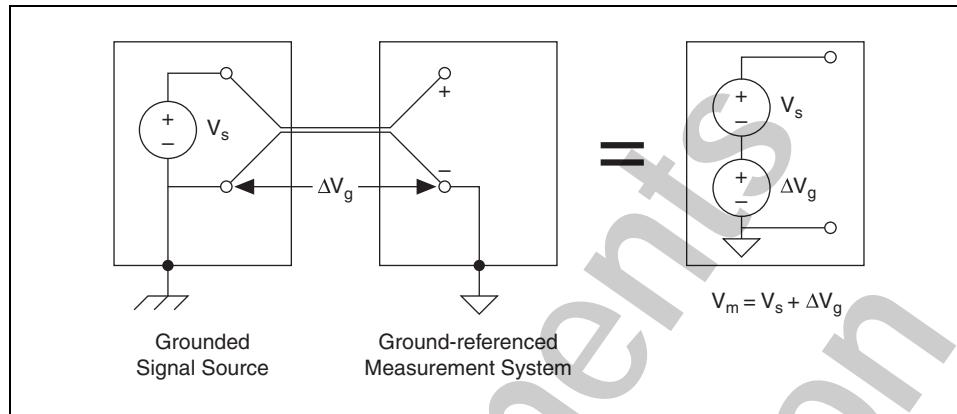


Figure 3-9. Pitfall of Using Ground-Referenced Measurement System with Grounded Signal Source

You can still use a ground-referenced system if the signal voltage levels are high and the interconnection wiring between the source and the measurement device has a low impedance. In this case, the signal voltage measurement is degraded by ground loop, but the degradation may be tolerable. You must observe the polarity of a grounded signal source before connecting it to a ground-referenced measurement system because the signal source might be shorted to ground, which can damage the signal source.

Both the differential (DIFF) and the NRSE input configurations provide non-referenced measurements on a typical DAQ device. With either configuration, differences between references of the source and the measuring device appear as common-mode voltage to the measurement system, and are subtracted from the measured signal. Figure 3-10 illustrates this concept.

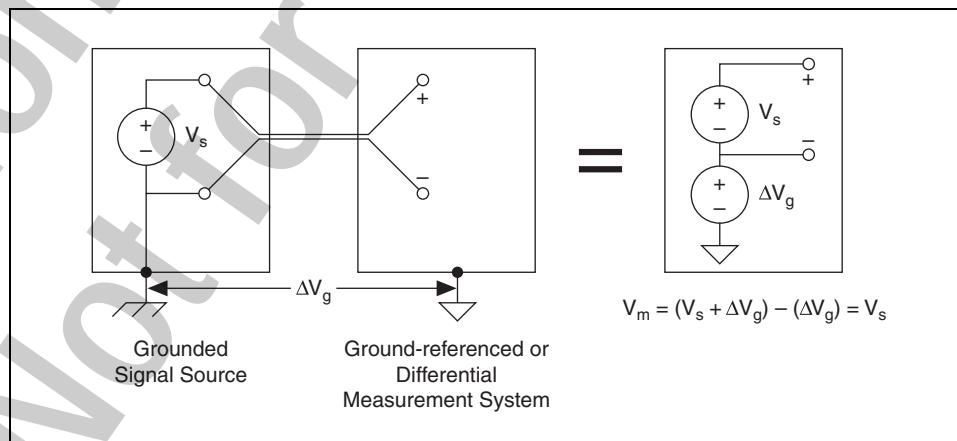


Figure 3-10. Using Non-referenced Measurement System with a Grounded Signal Source

Measuring Floating Sources

You can measure floating signal sources with both differential and single-ended measurement systems. However, in case of the differential measurement system, make sure the common-mode voltage level of the signal with respect to the measurement system ground remains in the common-mode input range of the measurement device.

A variety of phenomena—for example, the instrumentation amplifier input bias currents—can move the voltage level of the floating source out of the valid range of the input stage of a DAQ device. To anchor this voltage level to some reference, use resistors as shown in Figure 3-11.

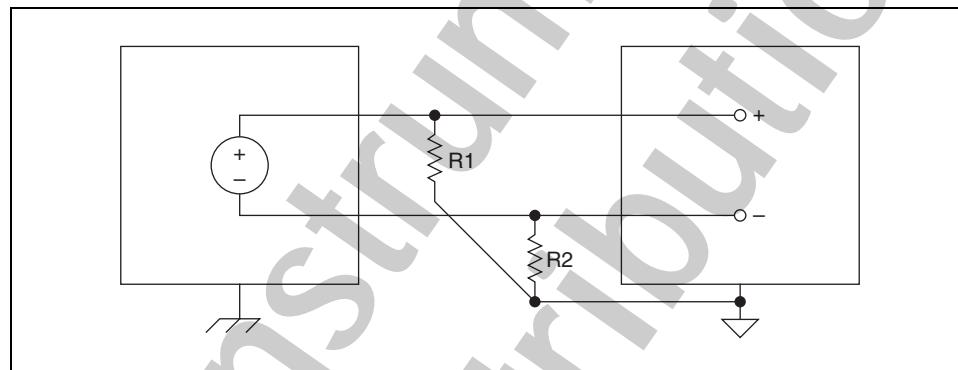


Figure 3-11. Using Bias Resistors with a Floating Signal Source

These bias resistors provide a DC path from the instrumentation amplifier inputs to the instrumentation amplifier ground. If you do not use resistors and the source is truly floating, the source is not likely to remain within the common-mode signal range of the instrumentation amplifier, and the amplifier saturates. You must reference the source to AIGND. The easiest way to connect the signal source to the measurement system is to connect the positive side of the signal to the positive input of the instrumentation amplifier and connect the negative side of the signal to AIGND and to the negative input of the instrumentation amplifier without bias resistors. This connection works well for DC-coupled sources with a source impedance less than $100\ \Omega$. For larger source impedances, this connection leaves the signal path significantly out of balance. Noise that couples electrostatically onto the positive line does not couple onto the negative line because it is connected to AIGND. Therefore, this noise appears as a differential-mode signal instead of a common-mode signal, and the instrumentation amplifier does not reject it. In this case, instead of directly connecting the negative line to AIGND, connect it to AIGND through a resistor that is about 100 times the equivalent source impedance. The resistor puts the signal path nearly in balance, so that about the same amount of noise couples onto both connections, yielding better rejection of electrostatically coupled noise.

You can fully balance the signal path by connecting another resistor of the same value between the positive input and AIGND, as shown Figure 3-11. This fully balanced configuration offers slightly better noise rejection but has the disadvantage of loading down the source with the series combination (sum) of the two resistors. If, for example, the source impedance is $2\text{ k}\Omega$ and each of the two resistors is $100\text{ k}\Omega$, the resistors load down the source with $200\text{ k}\Omega$ and produce a -1% gain error.

Both inputs of the instrumentation amplifier require a DC path to ground for the instrumentation amplifier to work. If the source is AC coupled (capacitively coupled), the instrumentation amplifier needs a resistor between the positive input and AIGND. If the source has low impedance, choose a resistor that is large enough not to significantly load the source but small enough not to produce significant input offset voltage as a result of input bias current (typically $100\text{ k}\Omega$ to $1\text{ M}\Omega$). In this case, you can tie the negative input directly to AIGND. If the source has high output impedance, you should balance the signal path as previously described using the same value resistor on both the positive and negative inputs. Some gain error results from loading down the source.



Caution Failure to use the following resistors results in erratic or saturated (positive full-scale or negative full-scale) readings.

Resistors provide a return path to ground for instrumentation amplifier input bias currents. Only R2 is required for DC-coupled signal sources. For AC-coupled sources, $R1 = R2$.

If you use single-ended input mode, you can use an RSE input system for a floating signal source. No ground loop is created in this case. You also can use the NRSE input system, which is preferable from a noise pickup point of view. Floating sources require bias resistor(s) between the AISENSE input and the measurement system ground (AIGND) in the NRSE input configuration.

Not for distribution

Summary of Signal Sources and Measurement Systems

Figure 3-12 summarizes ways to connect a signal source to a measurement system.

Input	Signal Source Type	
	Floating Signal Source (Not Connected to Building Ground)	Grounded Signal Source
	<p>Examples</p> <ul style="list-style-type: none"> • Ungrounded Thermocouples • Signal Conditioning with Isolated Outputs • Battery Devices 	<p>Examples</p> <ul style="list-style-type: none"> • Plug-in Instruments with Nonisolated Outputs
Differential (DIFF)		
Single-Ended — Ground Referenced (RSE)		<p>NOT RECOMMENDED</p> <p>Ground-loop losses, V_g, are added to measured signal.</p>
Single-Ended — Nonreferenced (NRSE)		

Figure 3-12. Summary of Signal Sources and Measurement Systems

B. Sampling Considerations

As humans, we perceive the world in analog. Everything you see and hear is a continuous transmission of information to your senses. This continuous stream is what defines analog data. Digital information, on the other hand, estimates analog data using only ones and zeros. Because of the many advantages of digital signal processing, analog signals are converted to digital form before they are processed with a computer. A digital signal is one that can assume only a finite set of values in both the dependent and independent variables. The independent variable is usually time or space, and the dependent variable is usually amplitude.

Digital signals are everywhere in the world around us. Telephone companies use digital signals to represent the human voice. Radio, TV, video players, phones, and hi-fi sound systems are all gradually converting to the digital domain because of its superior fidelity, noise reduction, and signal processing flexibility. Data is transmitted from satellites to earth ground stations in digital form. NASA's pictures of distant planets and outer space often are processed digitally to remove noise and to extract useful information. Economic data, census results, and stock market prices are all available in digital form.

Sampling Signals

To acquire an analog signal, you must first convert an analog signal into its digital representation. In practice, this is implemented by using an analog-to-digital (A/D) converter. Consider an analog signal $x(t)$ that is sampled every Δt seconds. The time interval Δt is known as the sampling interval or sampling period. Its reciprocal, $1/\Delta t$, is known as the sampling frequency, with units of samples/second. Each of the discrete values of $x(t)$ at $t = 0, \Delta t, 2\Delta t, 3\Delta t$, etc., is known as a sample. Thus, $x(0), x(\Delta t), x(2\Delta t), \dots$, are all samples. The signal $x(t)$ can be represented by the discrete set of samples as shown in the following equation.

$$\{x(0), x(\Delta t), x(2\Delta t), x(3\Delta t), \dots, x(k\Delta t), \dots\}$$

Figure 3-13 shows an analog signal and its corresponding sampled version. The sampling interval is Δt . The samples are defined at discrete intervals of time.

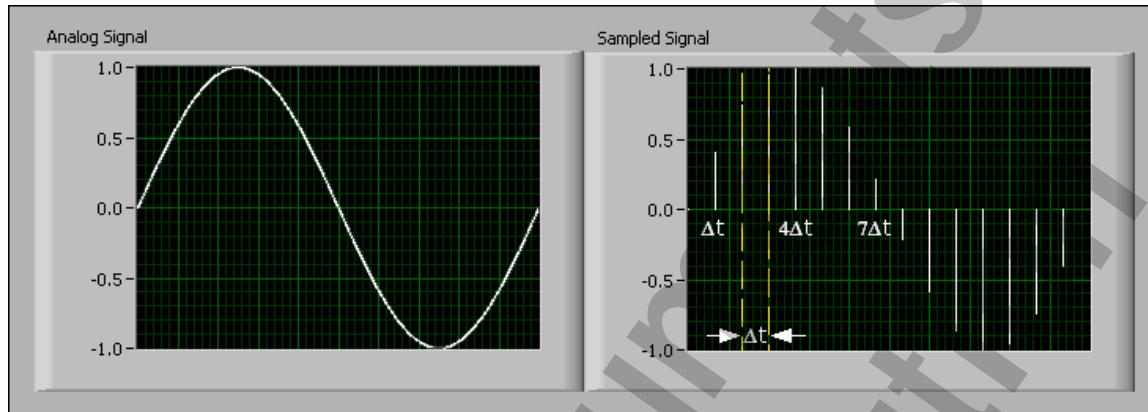


Figure 3-13. Analog Signal and Sampled Signal

In this course, the following notation represents the individual samples:

$$x[i] = x(i\Delta t), \quad \text{for } i = 0, 1, 2, \dots$$

If N samples are obtained from the signal $x(t)$, $x(t)$ can be represented by the sequence:

$$X = \{x[0], x[1], x[2], x[3], \dots, x[N-1]\}$$

This is known as the digital representation or the sampled version of $x(t)$. Notice that the sequence $X = \{x[i]\}$ is indexed on the integer variable i and does not contain any information about the sampling rate. By knowing just the values of the samples contained in X , you will not know what the sample rate is.

Sampling Rate

One of the most important elements of an analog input or analog output measurement system is the rate at which the measurement device samples an incoming signal or generates the output signal. The scan rate, or sampling rate in NI-DAQmx, determines how often an analog-to-digital (A/D) or digital-to-analog (D/A) conversion takes place. A fast input sampling rate acquires more points in a given time and can form a better representation of the original signal than a slow sampling rate can. Generating a 1 Hz signal using 1,000 points per cycle at 1,000 S/s produces a much finer representation than using 10 points per cycle at a sample rate of 10 S/s.

Aliasing

Sampling too slowly results in aliasing, which is a misrepresentation of the analog signal. Undersampling causes the signal to appear as if it has a different frequency than it actually does. To avoid aliasing, sample several times faster than the frequency of the signal.

Figure 3-14 shows an adequately sampled signal and the aliasing effects of undersampling.

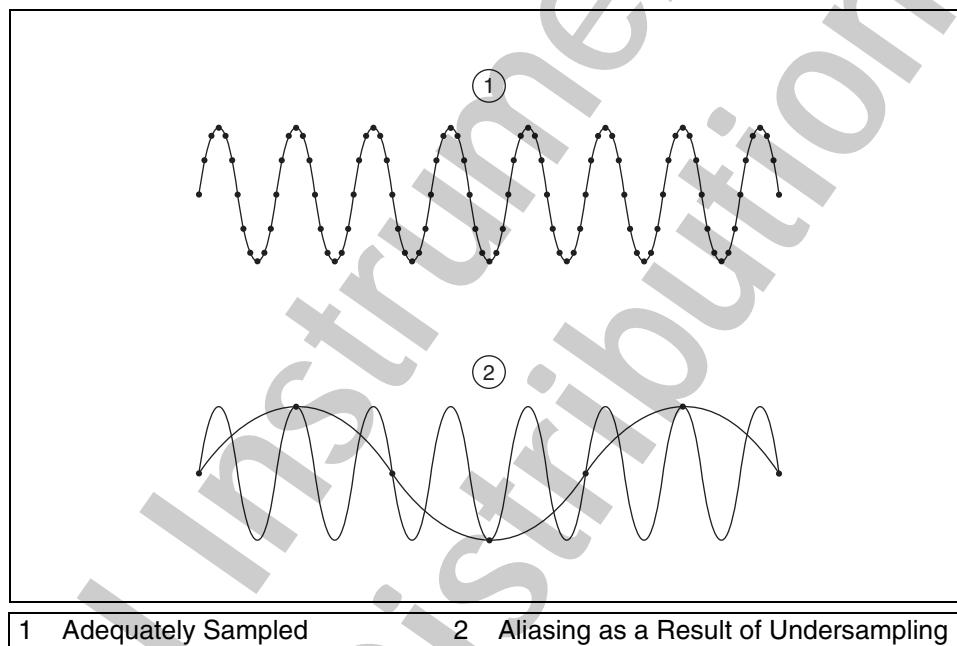


Figure 3-14. Adequately Sampled Signal and Undersampled Signal

For frequency measurements, according to the Nyquist theorem, you must sample at a rate greater than twice the maximum frequency component in the signal you are acquiring to accurately represent the signal. The Nyquist frequency is the maximum frequency you can represent accurately without aliasing for a given sampling rate. The Nyquist frequency is one half the sampling frequency. Signals with frequency components above the Nyquist frequency appear aliased between DC and the Nyquist frequency. The alias frequency is the absolute value of the difference between the frequency of the input signal and the closest integer multiple of the sampling rate.

For example, assume the sampling frequency, f_s , is 100 Hz. Also assume that the input signal contains the following frequencies: 25 Hz, 70 Hz, 160 Hz, and 510 Hz, as shown in Figure 3-15.

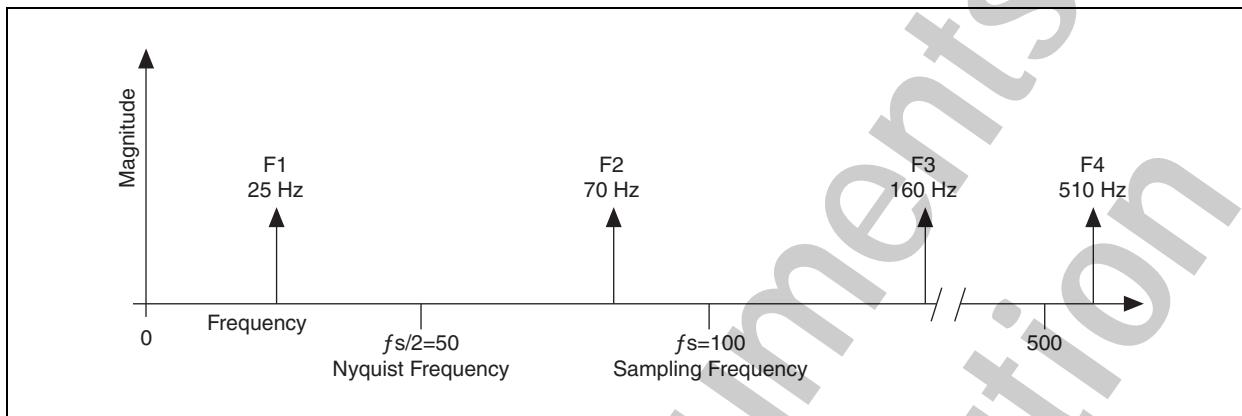


Figure 3-15. Frequencies in Example Signal

Frequencies below the Nyquist frequency ($f_s/2 = 50$ Hz) are sampled correctly, as shown Figure 3-16. Frequencies above the Nyquist frequency appear as aliases. For example, F_1 (25 Hz) appears at the correct frequency, but F_2 (70 Hz), F_3 (160 Hz), and F_4 (510 Hz) have aliases at 30 Hz, 40 Hz, and 10 Hz, respectively.

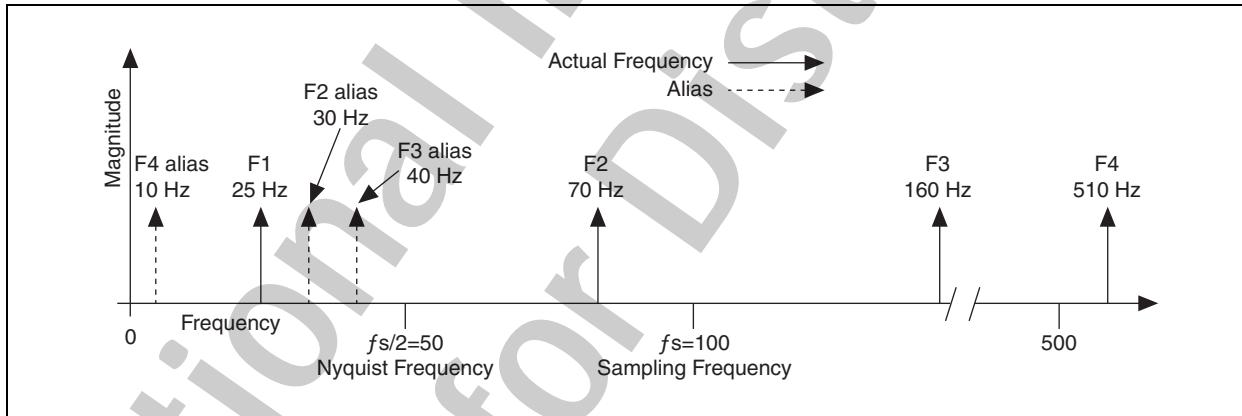


Figure 3-16. Alias Frequencies

Use the following equation to calculate the alias frequency:

$$\text{Alias Freq.} = |(\text{Closest Int. Mult. of Sampling Freq.} - \text{Input Freq})|$$

For example,

$$\text{Alias } F_2 = |100 - 70| = 30 \text{ Hz}$$

$$\text{Alias } F_3 = |(2)100 - 160| = 40 \text{ Hz}$$

$$\text{Alias } F_4 = |(5)100 - 510| = 10 \text{ Hz}$$

Determining How Fast to Sample

You might want to sample at the maximum rate available on the measurement device. However, if you sample very fast over long periods of time, you might not have enough memory or hard disk space to hold the data. Figure 3-17 shows the effects of various sampling rates.

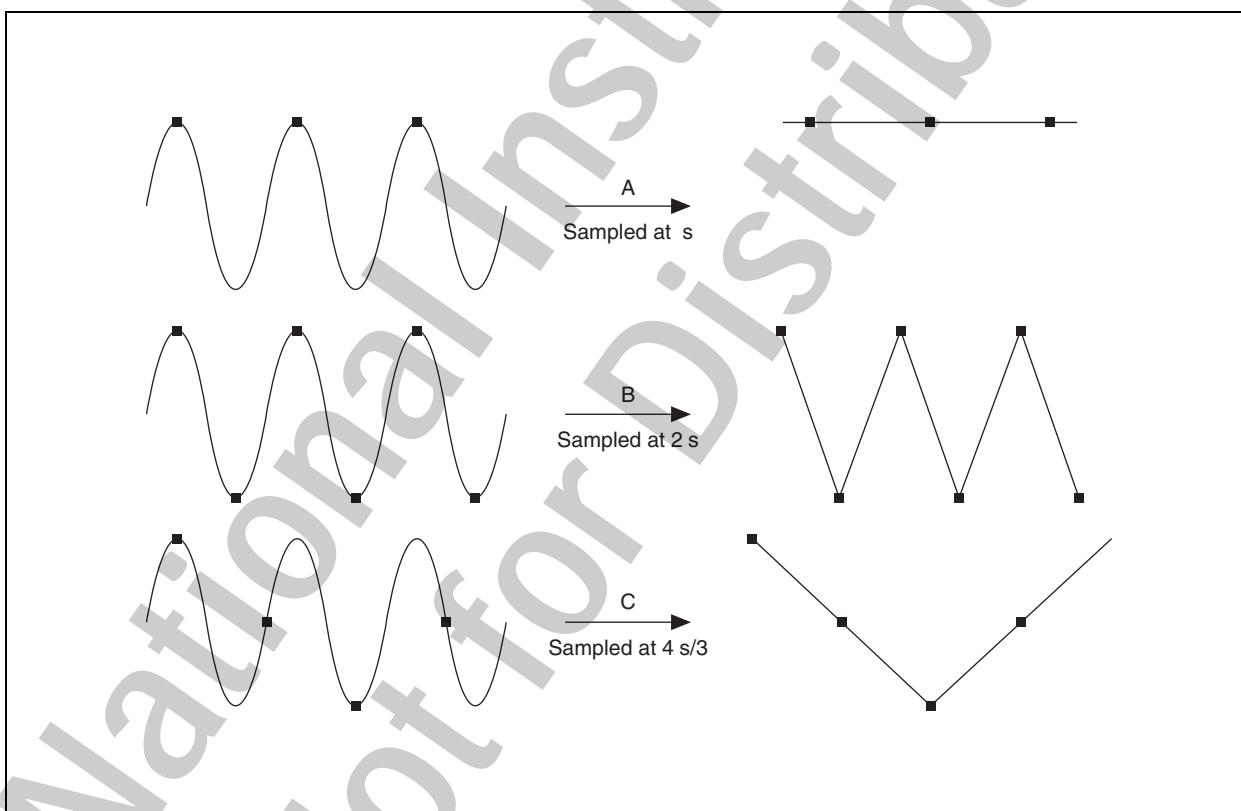


Figure 3-17. Results of Various Sampling Rates

Example A samples the sine wave of frequency f at the same frequency f_s . The acquired samples result in an alias at DC. However, if you increase the sampling rate to $2f_s$, the digitized waveform has the correct frequency or the same number of cycles as the original waveform but appears as a triangle waveform as shown in Example B. By increasing the sampling rate to well above f_s , you can more accurately reproduce the waveform. In Example C, the sampling rate is at $4f_s/3$. Because in this case the Nyquist frequency is below f_s , $(4f_s/3 \times 1)/2 = 2f_s/3$, this sampling rate reproduces an alias waveform of incorrect frequency and shape.

The Nyquist theorem provides a starting point for the adequate sampling rate—greater than two times the highest frequency component in the signal. Unfortunately, this rate is often inadequate for practical purposes. Real-world signals often contain frequency components that lie above the Nyquist frequency and are erroneously aliased and added to the components of the signal that are sampled accurately, producing distorted sampled data. Therefore, for practical purposes, sampling is usually done at several times the maximum frequency—5 to 10 times is typical in industry.

Anti-aliasing Filters

You have seen that the sampling rate should be at least twice the maximum frequency of the signal that you are sampling. In other words, the maximum frequency of the input signal should be less than or equal to half of the sampling rate. But how do you ensure that this is definitely the case in practice? Even if you are sure that the signal being measured has an upper limit on its frequency, pickup from stray signals (such as the power line frequency or from local radio stations) could contain frequencies higher than the Nyquist frequency. These frequencies may then alias into the desired frequency range and thus give you erroneous results.

To be completely sure that the frequency content of the input signal is limited, a lowpass filter (a filter that passes low frequencies but attenuates the high frequencies) is added before the ADC. This filter is called an anti-alias filter because it prevents the aliasing components from being sampled by attenuating the higher frequencies (greater than Nyquist). Anti-aliasing filters are analog filters. Figure 3-18 shows an ideal anti-alias filter.

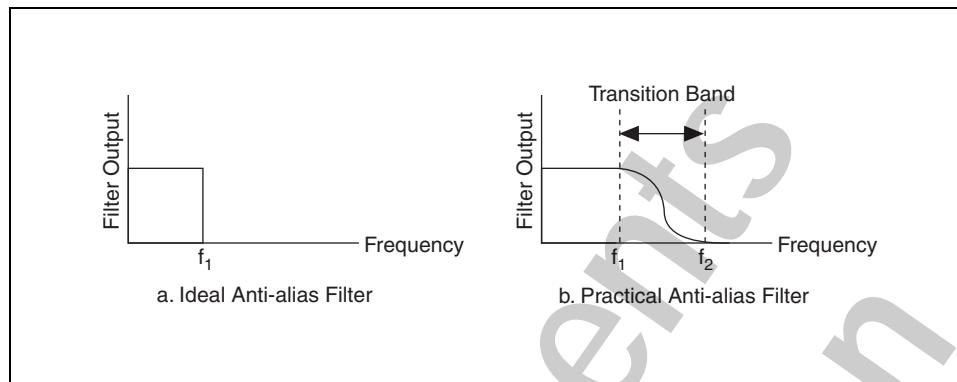


Figure 3-18. Ideal and Practical Anti-alias Filters

An ideal anti-aliasing filter passes all the desired input frequencies (below f_1) and cuts off all the undesired frequencies (above f_1). However, an ideal anti-aliasing filter is not physically possible. In practice, filters look as shown in illustration (b) in Figure 3-18. Practical anti-aliasing filters pass all frequencies $< f_1$ and cut off all frequencies $> f_2$. The region between f_1 and f_2 is known as the transition band, which contains a gradual attenuation of the input frequencies. Although you want to pass only signals with frequencies $< f_1$, the signals in the transition band could still cause aliasing. Therefore, in practice, you should use a sampling frequency greater than two times the highest frequency in the transition band. Because this sampling frequency turns out to be more than two times the maximum input frequency (f_1), you might see that the sampling rate is more than twice the maximum input frequency.

C. Single Sample Software-Timed Acquisition

You can perform a single sample, software-timed acquisition using the DAQmx Read VI.

Using the DAQmx Read VI

The DAQmx Read VI, located on the **DAQmx - Data Acquisition** palette, reads samples from the task or channels you specify. The instances of this polymorphic VI specify what format of samples to return, whether to read a single sample or multiple samples at once, and whether to read from one or multiple channels. Use the pull-down menu to select an instance of the VI, as shown in Figure 3-19.

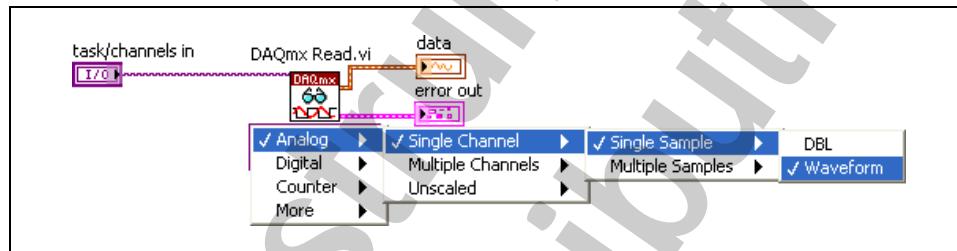


Figure 3-19. DAQmx Read VI

In the first selection menu, you can choose from the following types of input:

- Analog
- Digital
- Counter
- More (Raw Data)

Use the second selection menu to determine the number of channels to read from or if the data is unscaled. Use the third selection menu to choose to read a single sample or multiple samples. If you select a single sample, use the fourth selection menu to select whether to return the data as a waveform or a double-precision floating-point value. If you select multiple samples, use the fourth selection menu to select whether to return the data as a waveform or an array of double-precision floating point values.

When addressing analog input or analog output channels, you might want to address more than one channel at a time. If these channels have the same type of timing and triggering, group the channels into a task. Otherwise, use the I/O Name Filtering tool on the shortcut menu of the NI-DAQmx global or physical channel control/constant and select **Allow Multiple Names** (selected by default). Separate the channel names with a comma. You cannot address multiple tasks at a time.

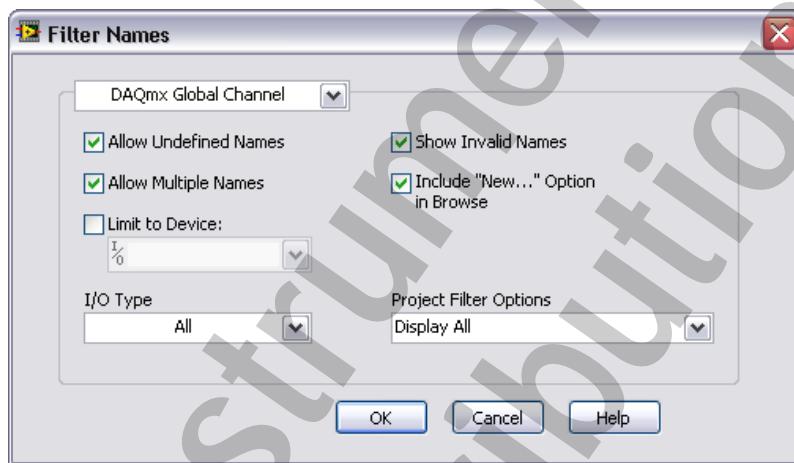


Figure 3-20. I/O Name Filtering Window

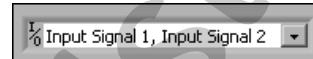


Figure 3-21. DAQmx Name Control

Waveform Data Type

The waveform data type is a cluster that consists of the following elements:

- **Y**—A 1D array of numeric data points, which can be a single point or a waveform depending on the operation. The representation of the 1D array is DBL.
- **t_0** —A scalar value that represents the time, according to the system clock, when the first point in the Y array was acquired. This element is also called the initial time or the timestamp.
- **Δt** —A scalar value that represents the time between data points in the Y array.
- **Attributes**—A string that allows you to bundle other information along with the waveform, such as the device number or channel number.

The waveform data type has many benefits over the conventional scaled array.

- **The presence of t_0** —Before the waveform data type existed, you could not determine when data was acquired. The waveform data type automatically returns the time of day and the date in the t_0 element, which gives you a real-world acquisition time for the data.
- **Easier Graphing**—The waveform data type simplifies graphing the data. Previous versions of LabVIEW requires bundling the value of the initial point (x_0) and the time between points (Δx) with the data (Y array). The waveform data type contains these elements, so all you have to do is wire the waveform data type to the graph.
- **Easier Multiple Plot Graphing**—The waveform data type simplifies multiple plot graphs. Previous versions of LabVIEW requires bundling the x_0 , Δx , and Y array for each plot, then build an array to create a multiple plot graph. Using a waveform data type, you wire a 1D array of waveforms to the graph for a multiple plot. If you acquire data on multiple channels with an analog input VI, the VI returns a 1D array, which you can wire directly to the graph.

Single Sample Software-Timed Acquisition

For a single sample software-timed acquisition, use the DAQmx Read inside a While Loop to acquire one sample each iteration. You one of the wait functions inside the While Loop to implement a software-timed rate of acquisition.

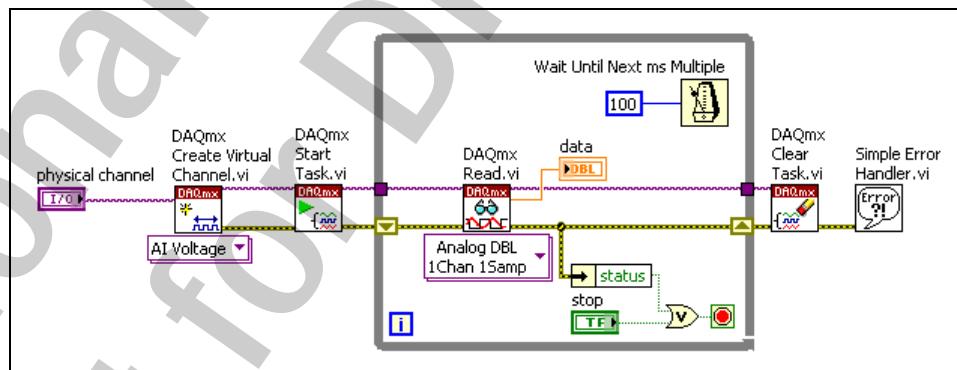


Figure 3-22. Single Sample Software-Timed Acquisition

D. DAQ Device Architectures

The number and arrangement of the components on the device depends on the DAQ device you use. The architecture of the device affects how you sample a signal. National Instruments DAQ devices that perform analog input can have one of two main architectures, as shown in Figure 3-23.

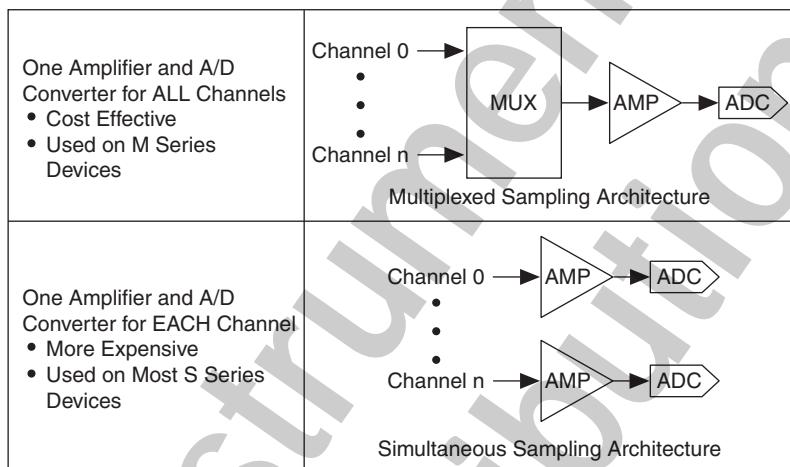


Figure 3-23. Analog Input Architectures

The multiplexed sampling architecture consists of one multiplexer (MUX), one instrumentation amplifier, and one ADC. In this layout, all the input channels must share one ADC. Using only one ADC makes this architecture very cost effective, so it is used on M Series and some X Series devices.

The simultaneous sampling architecture consists of an instrumentation amplifier and an ADC for each channel. This architecture is used on most S Series devices. Although this architecture is more expensive than using one ADC for all channels, it allows you to perform simultaneous sampling and delivers much higher sample rates per channel.

Sampling Terminology

- **Samples per Channel per Second**—The number of samples to acquire for each second.
- **Sample Clock**—A pulse train used to start sample acquisition. Each time the sample clock produces a pulse, one sample per channel is acquired.
- **AI Convert Clock**—A pulse train used to trigger an A/D conversion.
- **Sample Duration**—The time it takes to complete one set of samples. Use the following formula to calculate the sample duration:

$$\text{Sample Duration} = (\# \text{ of channels} - 1) \times \text{AI Convert Clock}$$

Multiplexed Sampling

In sampling a signal, you can choose from multiplexed sampling or simultaneous sampling. Figure 3-24 shows an example of multiplexed sampling.

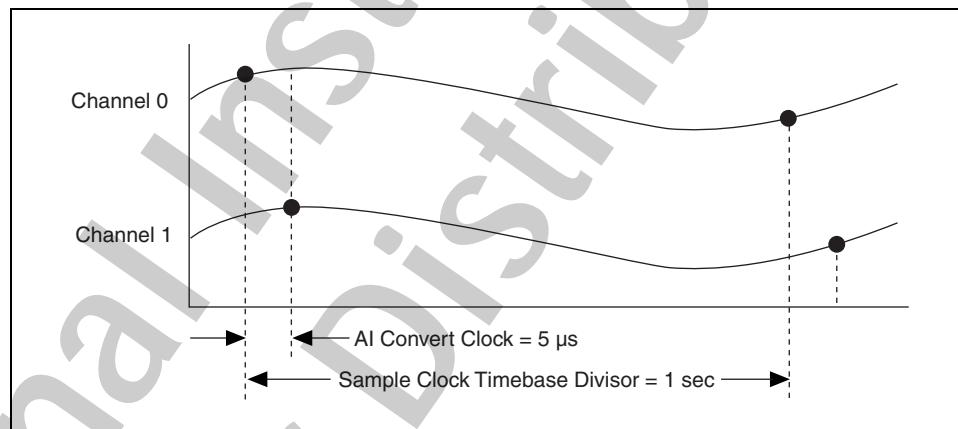


Figure 3-24. Multiplexed Sampling

The most common method, multiplexed sampling, shares one ADC between all the channels on the device. This architecture is found on many DAQ devices. Multiplexed sampling uses a sample clock and the AI Convert Clock to control the MUX. To understand how these two clocks interact, assume you are acquiring data on two channels. When the sample clock signals the start of an acquisition, the MUX connects the first channel to the ADC, and the AI Convert Clock pulses once. When the AI Convert Clock pulses, the ADC acquires one point from the first channel. Before the AI Convert Clock pulses again, the MUX connects the second channel to the ADC. When the AI Convert Clock pulses again, the ADC takes one point from the second channel. When the sample duration has elapsed, the sample clock pulses again, and the cycle repeats. The sample clock determines how often the device takes a sample of all the channels. The AI Convert Clock

actually takes the samples. Because interval sampling uses a sample clock and the AI Convert Clock, the device can sample the channels in a short period of time.

In Figure 3-24, the device takes a sample from each channel every second, but the lag between samples is only 5 μ s, as determined by the period of the AI Convert Clock. For the cost-effectiveness of having only one ADC, you can achieve near-simultaneous sampling.

By default, NI-DAQmx chooses the fastest AI Convert Clock rate possible that still allows for adequate settling time. If necessary, you can manually set the AI Convert Clock rate using the DAQmx Timing Property Node.

Simultaneous Sampling

If the time relationship between the signals is important, you could use interval sampling, but sometimes interval scanning does not preserve the time relationship between signals to a narrow enough tolerance. In this case, you should use simultaneous sampling, as shown in Figure 3-25.

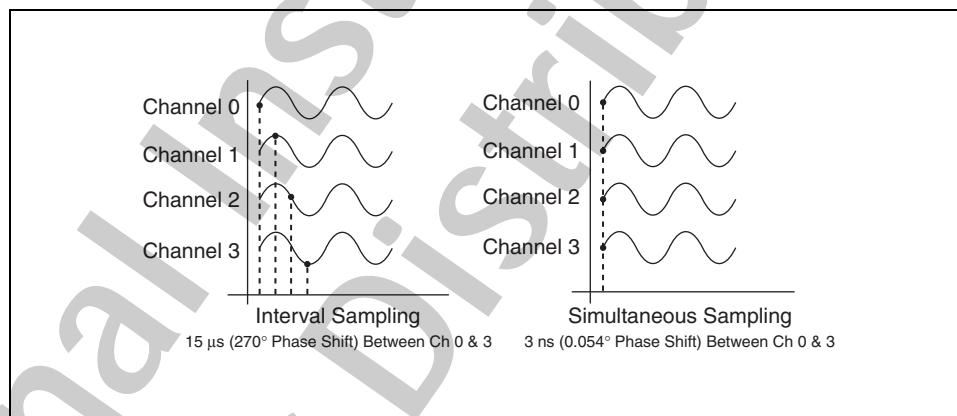


Figure 3-25. Simultaneous Sampling

Simultaneous sampling uses one ADC for each channel so you can sample all channels at the same time. While this requires a more expensive architecture than interval scanning, it eliminates the lag between channels caused by having to share the ADC between all channels. Because simultaneous sampling samples every channel at the same time, you only need a sample clock to determine the sampling rate.

Compare all three types of sampling by determining the phase shift that occurs when you sample four 50 kHz signals at a rate of 200 kHz. With round-robin sampling, all the samples must be evenly spaced, which causes a 15 μ s delay between the time of the sample on channel 0 to the time of the sample on channel 3. This corresponds to a 270° phase shift. With interval scanning, assume you have a 5 μ s interchannel delay. Again, you experience a 15 μ s delay between channel 0 and channel 3. With simultaneous

sampling, you experience only a three nanosecond delay between channel 0 and channel 3. This delay corresponds to a 0.054° phase shift. Simultaneous sampling provides a great advantage in preserving the time relationship between signals, albeit at a higher cost. The S Series family and some X Series devices can perform simultaneous sampling.

E. Finite Buffered Acquisition

To acquire multiple points at the same time, select an instance of the DAQmx Read VI that reads multiple samples. Use the DAQmx Read VI in combination with the DAQmx Timing VI, the DAQmx Start Task VI, and the DAQmx Clear Task VI to create a hardware-timed buffered acquisition VI.

- **Hardware-Timed Acquisition**—A hardware signal, such as a sample clock or AI Convert Clock, controls the rate of acquisition. A hardware clock is much faster than a software loop, so you can sample a higher range of frequencies without aliasing the signal. A hardware clock is also more accurate than a software loop. A software loop rate can be affected by a variety of actions, such as the opening of another program on the computer, but a hardware clock stays consistent.
- **Buffered Acquisition**—Acquires multiple points with one call to the device. The points are transferred from the device to an intermediate memory buffer before LabVIEW reads them.

DAQmx Timing VI

The DAQmx Timing VI configures the sample rate, the number of samples to acquire or generate and creates a buffer when needed. The instances of this polymorphic VI correspond to the type of timing to use on the task. The available timing options are Sample Clock, Handshaking, Implicit, Use Waveform, and Change Detection.

For analog input, select the Sample Clock instance in the pull-down menu of the DAQmx Timing VI. This instance of the VI includes the following parameters:

- **Sample mode**—Specifies if the task executes continuously or for a finite duration.
- **Samples per channel**—Specifies the number of samples to input or output if sample mode is **Finite Samples**. This value specifies the size of the intermediate memory buffer that stores the data as it is transferred from the DAQ device to LabVIEW.
- **Rate**—Specifies the sampling rate in samples per channel per second. If you use an external source for the Sample Clock, set this input to the maximum expected rate of that clock.

- **Source**—Specifies the source terminal of the Sample Clock. Leave this input unwired to use the default onboard clock of the device.
- **Active edge**—Specifies on which edge of the clock to measure or generate samples. Select the rising or falling edge of the sample clock.
- **Task/channels in**—Specifies the name of the task or a list of virtual channels the operation applies to. If you provide a list of channels, NI-DAQmx creates a task automatically.

The Handshaking instance of the DAQmx Timing VI determines the number of digital samples to acquire or generate using digital handshaking between the device and a peripheral device. Refer to Lesson 5, *Digital I/O*, of this manual for more information about the Handshaking instance of the DAQmx Timing VI.

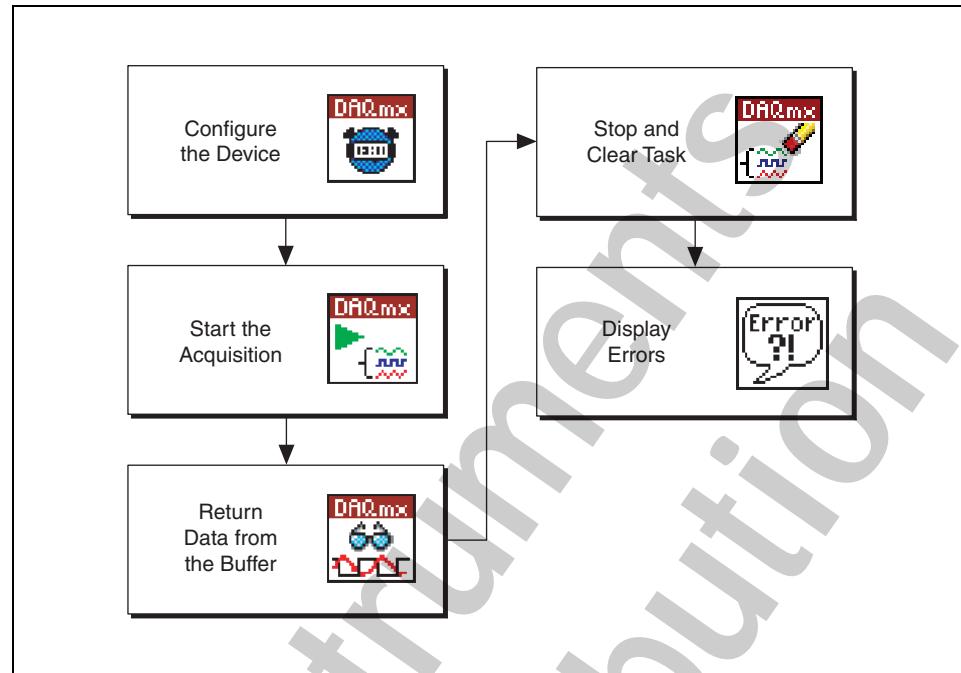
The Implicit instance of the DAQmx Timing VI sets only the number of samples to acquire or generate without specifying the timing. Typically, you should use this instance when the task does not require sample timing, such as tasks that utilize counters for buffered frequency measurement, buffered period measurement, or pulse train generation.

The Use Waveform instance of the DAQmx Timing VI uses the **dt** component of the **waveform** input to determine the sample clock rate. **dt** is the time in seconds between samples. If **sample mode** is **Finite Samples**, NI-DAQmx generates the number of samples in the waveform. This VI does not actually output any samples. You must wire the same waveform to the DAQmx Write VI to produce the samples. Refer to Lesson 4, *Analog Output*, of this manual for more information about the Use Waveform instance of the DAQmx Timing VI.

The Change Detection instance of the DAQmx Timing VI configures the task to acquire samples on the rising and / or falling edges of the lines or ports you specify. To detect both rising and falling edges on a line or port, wire the name of that line or port to both rising edge physical channels and falling edge physical channels.

Finite Buffered Acquisition Flowchart

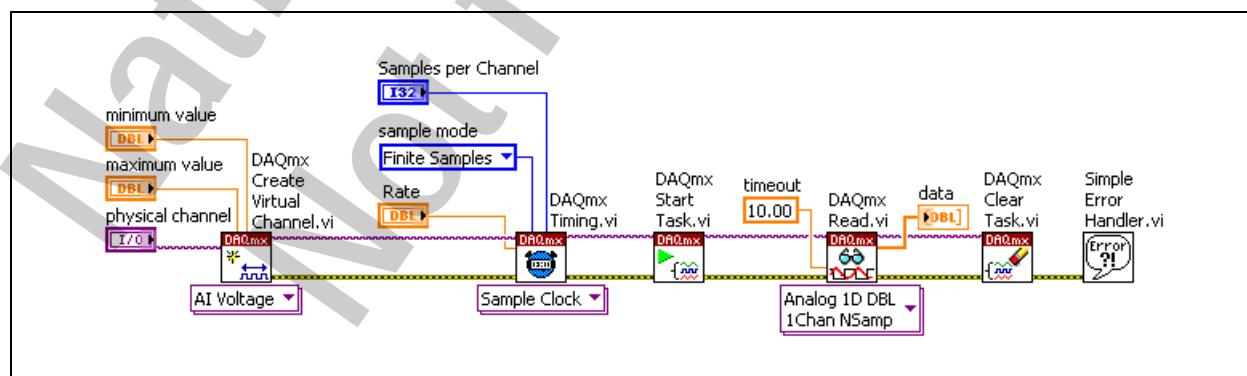
The flowchart in Figure 3-26 shows a basic finite buffered acquisition. A finite buffered acquisition acquires a set number of points at a specified rate. Use the DAQmx Timing VI to configure the timing and buffer for the device. The DAQmx Read VI waits until all the samples on each channel are available before returning the data and moving on. The DAQmx Clear Task VI stops acquisition and clears the task. The Error Handler VI displays any errors that occurred during the process.

**Figure 3-26.** Finite Buffered Acquisition Flowchart

Finite Buffered Acquisition Example

Figure 3-27 shows how to create a finite buffered acquisition VI. Start a finite buffered acquisition by configuring the sample mode and acquisition rate with the DAQmx Timing VI. To acquire a finite number of samples, set the sample mode to Finite Samples. To set the acquisition rate, use the Rate input. To set the number of samples to acquire, use the Samples per Channel input.

Next, the DAQmx Start Task VI starts the acquisition. The program then waits at the DAQmx Read VI until the buffer is full. When the buffer is full, the DAQmx Read VI returns the data from the buffer, the DAQmx Clear Task VI stops the acquisition and clears the task, and the Simple Error Handler VI displays any errors.

**Figure 3-27.** Finite Buffered Acquisition VI Block Diagram

Because the **number of samples per channel** input for the DAQmx Read VI is unwired, NI-DAQmx automatically determines how many samples to read based on the configuration you set in the DAQmx Timing VI. NI-DAQmx automatically determines this value and the **number of samples per channel** input is set to -1. The DAQmx Read VI returns a 2D array that can be directly wired to a waveform graph. The array does not include timing information, unlike the waveform data type.

Always wire the error clusters from one VI to another. If **error in** detects an error in the DAQmx Start Task VI, DAQmx Read VI, or DAQmx Clear Task VI, the VI returns the error information in **error out** and does not continue to run. For example, assume an error occurs in the DAQmx Start Timing VI. The DAQmx Start Timing VI stops executing and passes the error information to the DAQmx Start Task VI. The DAQmx Start Task VI does not execute—it passes the error to the next VI. The error information passes through each VI to the Error Handler VI for display.

What's Really Happening?

To understand what happens when you perform a buffered acquisition, examine a buffered acquisition on a lower level, as shown in Figure 3-28.

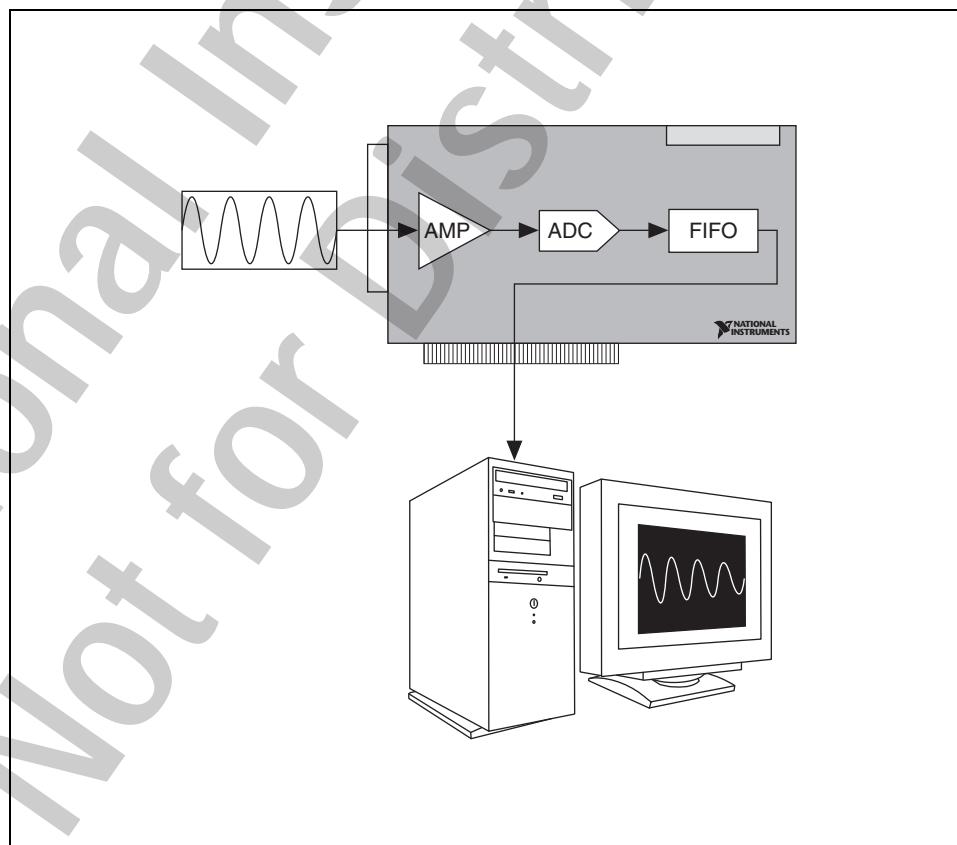


Figure 3-28. Buffered Acquisition Flow

You already know that when you acquire an analog signal, it passes through the instrumentation amplifier to the ADC. However, you might not know what happens to the signal after that. The signal passes to an onboard First In First Out (FIFO) buffer which stores the data until it can be transferred from the device to the computer. The data is then transferred from the device to a PC buffer through Direct Memory Access (DMA) or Interrupt Request (IRQ).

The PC buffer is a memory location that stores the data after it has left the device. The **number of samples per channel** input of the DAQmx Timing VI (or the **buffer size** input of the DAQmx Configure Input Buffer VI) configures the PC buffer, which stores the data until DAQmx Read VI is ready to retrieve it. The DAQmx Read VI then transfers the data to a LabVIEW buffer, which can then be displayed on the front panel. The LabVIEW buffer can place the data in a waveform graph, an array, or a waveform data type, depending on the DAQmx Read VI instance and how you wire the output of the DAQmx Read VI.

Buffer Transfer

The transfer of data between the PC buffer and the LabVIEW buffer is important in analog input operations. The **number of samples per channel** input of the DAQmx Timing VI allocates the PC buffer. When you perform a buffered acquisition, the acquisition begins when you call the DAQmx Start Task VI. After the acquisition begins, the PC buffer starts filling with data. A buffered acquisition fills the PC buffer until it is full. The rate at which the buffer fills is determined by the rate that you set in the DAQmx Timing VI. When the buffer is full, the DAQmx Read VI transfers the data from the PC buffer to the LabVIEW buffer. The DAQmx Read VI removes all the data at one time in a buffered acquisition.

F. Continuous Buffered Acquisition

The main difference between a finite buffered acquisition and continuous buffered acquisition is the number of points that you acquire. With a finite buffered acquisition, you acquire a set number of points. With a continuous buffered acquisition, you can acquire data indefinitely. The flowchart in Figure 3-29 shows a continuous buffered acquisition.

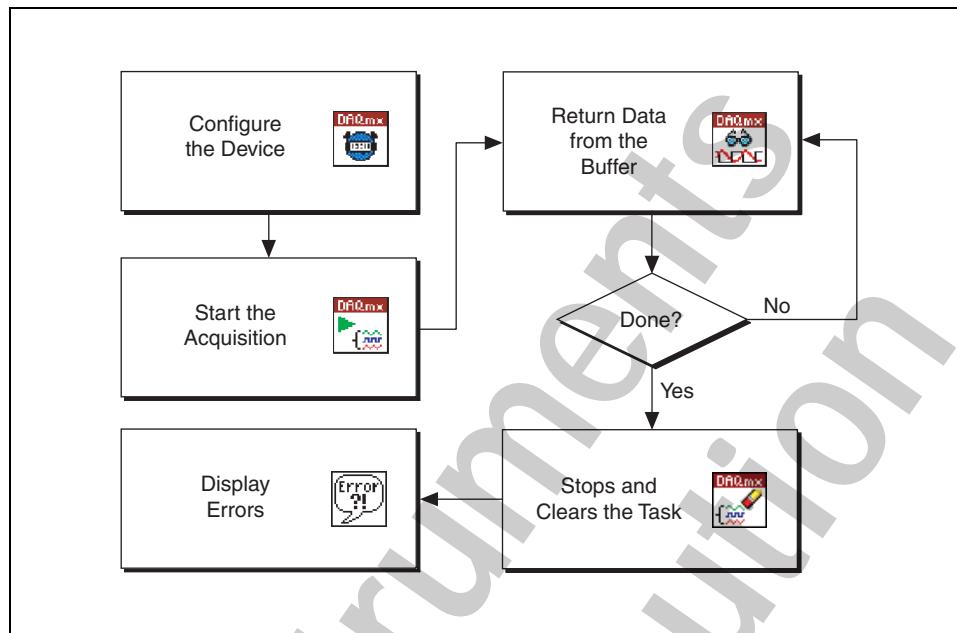


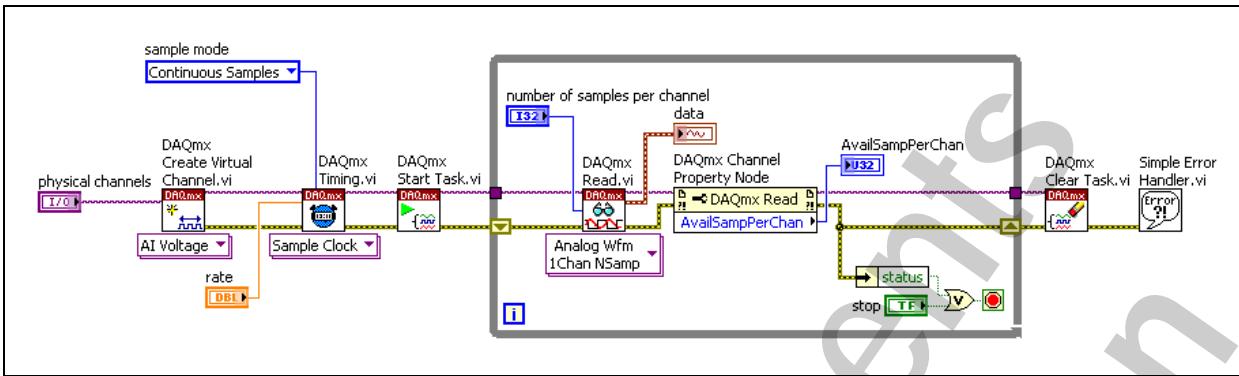
Figure 3-29. Continuous Buffered Acquisition Flowchart

The first three steps of the flowchart are identical to the first three steps of the buffered acquisition flowchart. Configure a device with the DAQmx Timing VI, start the acquisition with the DAQmx Start Task VI, and prepare to read the data with the DAQmx Read VI. Because you acquire data continuously, you need to read data continuously. Place the DAQmx Read VI in a loop. The loop completes when an error occurs or when you stop the loop from the front panel. Every time the loop runs, the DAQmx Read VI returns data. When the loop ends, the DAQmx Clear Task VI stops acquisition and clears the task. The Simple Error Handler VI displays any errors that occurred during the process.

Continuous Buffered Acquisition

Figure 3-30 shows the block diagram of an example continuous buffered acquisition VI that is similar to a buffered acquisition with the following changes:

- The DAQmx Read VI is inside a While Loop.
- The **number of samples per channel** input is user-specified. With finite acquisition, NI-DAQmx automatically determines how many samples to read. If you leave the **number of samples per channel** input unwired or set to -1 , NI-DAQmx reads the total number of samples available in the buffer.
- Monitor the available samples per channel (backlog).

**Figure 3-30.** Continuous Buffered Acquisition VI Block Diagram

Start a continuous buffered acquisition by configuring the sample mode and acquisition rate with the DAQmx Timing VI. If the samples per channel input is left unwired, NI-DAQmx automatically determines the buffer size based on the rate input, as shown in Table 3-1.

Table 3-1. NI DAQmx Default Buffer Sizes

Sample Rate	Buffer Size
No Rate Specified	10 kS
0–100 S/s	1 kS
100–10,000 S/s	10 kS
10,000–1,000,000 S/s	100 kS
>1,000,000 S/s	1 MS

You can also manually set the buffer size using the DAQmx Configure Input Buffer VI.

The DAQmx Start VI starts the acquisition.

The DAQmx Read VI, which is located inside a While Loop, reads the data from the continuous buffered acquisition. To prevent buffer overflow, the **number of samples per channel** should not be equal to the buffer size. It is good practice to set the **number of samples per channel to read** to one-fourth or one-half the buffer size for a continuous buffered acquisition. Because LabVIEW continually sends data into the buffer, it is important to monitor the number of available samples per channel in the buffer to make sure the buffer is emptying fast enough.

If the number of available samples per channel (backlog) increases steadily, the buffer might overflow and generate an error. The While Loop containing the DAQmx Read VI can be stopped when the user clicks a button on the front panel or when an error occurs in the DAQmx Read VI, such as a buffer overflow. After the While Loop stops, the DAQmx Clear Task VI stops acquisition and clears the task, and the Simple Error Handler VI displays any errors.

Circular Buffer

A continuous buffered operation is difficult because the computer is using a single buffer, but you are acquiring more data than the buffer can hold. To acquire more data than the buffer can hold, use a circular buffer. Figure 3-31 shows how a circular buffer works.

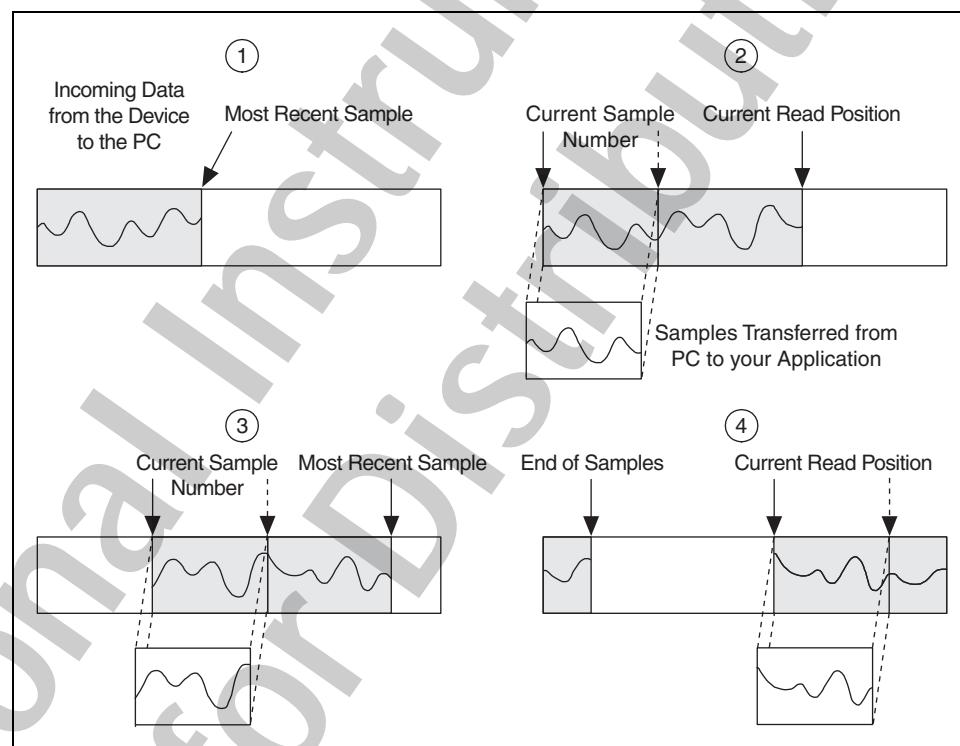


Figure 3-31. Circular Buffer Operation

A circular buffer is similar to a regular buffer, but when you get to the end of a circular buffer, instead of stopping, you start over at the beginning. Start with the PC buffer that was allocated by the **samples per channel** input of the DAQmx Timing VI. When the DAQmx Start Task VI starts the acquisition, the PC buffer starts to fill with data. The acquisition occurs inside the While Loop.

Assume that you set the **number of samples per channel to read** to between one-fourth and one-half of the PC buffer size. When the number of samples per channel in the PC buffer is equal to the **number of samples per channel to read**, the DAQmx Read VI transfers that number of samples per channel from the PC buffer to the LabVIEW buffer. The DAQmx Read VI sets a flag called the current sample position so it can continue reading where it left off.

Meanwhile, the PC buffer continues to fill with data. The DAQmx Read VI continues transferring data from the PC buffer to the LabVIEW buffer while the PC buffer fills. When the end of data mark reaches the end of the PC buffer, the new data is written at the beginning of the buffer. The difference between the end of samples mark and the current sample position is equal to the number of available samples per channel (backlog). LabVIEW must read the data from the buffer fast enough to prevent the end of samples mark from catching up to the current sample position. Otherwise the new data overwrites the old data, and LabVIEW generates an error.

Overwrite Error

The most common error you might encounter when performing a circular buffered acquisition is the overwrite error. The overwrite error occurs when the end of samples mark catches up to the current sample position and you overwrite data. The problem occurs when LabVIEW does not read data from the PC buffer quickly enough. Several options exist to help avoid the error, but not all options might apply to the situation, and some work better than others.

- Increase the number of samples per channel (buffer size) with the DAQmx Timing VI. Increasing the buffer size does not solve the problem if you are not emptying the buffer fast enough. Remember the guideline for setting the samples per channel to read at one-fourth to one-half of the buffer size. Increasing the buffer size only works if it makes this guideline true.
- Empty the buffer more quickly by increasing the number of samples per channel to read. Do not set the number of samples per channel to read too high because you will wait in the DAQmx Read VI for the number of samples per channel in the buffer to equal the number of samples per channel to read. The time spent waiting for samples to fill the buffer could be spent emptying the buffer.
- Decrease the samples per channel rate with the DAQmx Timing VI. This setting slows down the rate that data is being sent to the buffer, but it might not be an option if you want a certain sample rate.
- Avoid slowing down the loop with unnecessary analysis.

Overflow Error

Another error you might encounter with a continuous buffered acquisition involves overflowing the FIFO buffer on the device. Overflow error is not as common as overwriting the PC buffer, and it is not as easy to correct. The problem occurs when the FIFO buffer does not empty fast enough. The FIFO buffer relies on either DMA or IRQ to transfer the data from the FIFO buffer to the PC buffer. When the FIFO buffer does not empty fast enough, then the options to prevent the error are few.

- Make sure you are using DMA to transfer the data, if DMA is available. DMA is faster than IRQ and can improve performance significantly. For more information about using DMA, refer to the *NI-DAQmx Help* and the DAQmx Channel Property Node, Data Transfer Mechanism property.
- Decrease the samples per channel rate in the DAQmx Timing VI.
- Purchase a device with a larger FIFO buffer. However, this option might delay the problem instead of solving it.
- Purchase a computer with a faster bus to expedite the data transfer from the FIFO buffer to the PC buffer. Overflow is caused by the system not transferring the data off the device fast enough. A computer with a faster bus can transfer the data from the FIFO buffer faster.

G. Triggering

A trigger is a signal that causes an action, such as starting the acquisition of data. Use a trigger if you need to set a measurement to start at a certain time. For example, imagine that you want to test the response of a circuit board to a pulse input. You can use that pulse input as a trigger to communicate to the measurement device to start acquiring samples. If you do not use this trigger, you have to start acquiring data before you apply the test pulse.

When you configure a trigger, you must make two main decisions—what action you want the trigger to cause and how to produce the trigger.

If you want the trigger to begin the measurement, use a start trigger. If you want to acquire data before the trigger occurs, use a reference trigger, also known as a stop trigger, that captures samples before and after a trigger point, which becomes the reference position in the samples.

In addition to specifying the action you want a trigger to cause, you need to determine the source of the trigger. If you need to trigger off an analog signal, use an analog edge trigger or an analog window trigger. If the trigger signal is digital, you can use a digital edge trigger with a PFI pin as the source.

Types of Triggers

There are different types of triggers based on analog and digital signals. Examples are analog edge triggers, analog window triggers, and digital edge triggers.

Analog Edge Triggering

An analog edge trigger occurs when an analog signal meets a condition you specify, such as the signal level or the rising or falling edge of the slope. When the measurement device identifies the trigger condition, it performs the action you associated with the trigger, such as starting the measurement or marking which sample was acquired when the trigger occurred. For example, consider an application that monitors a temperature system. If you want to begin data acquisition only after the temperature rises to 50 °C, configure an analog trigger to occur when the temperature signal has a rising slope and voltage level corresponding to 50 °C. Figure 3-32 shows triggering on a rising slope at a level of 3.2 V.

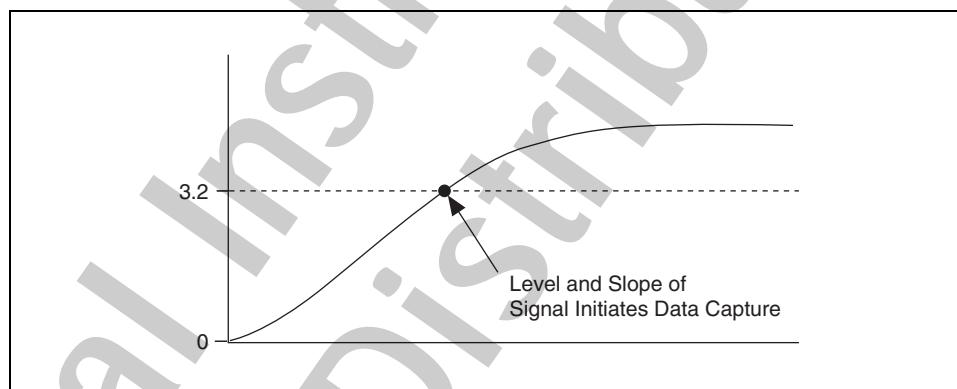


Figure 3-32. Triggering on a Rising Slope

Hysteresis

Hysteresis adds a window above or below the trigger level and often reduces false triggering due to noise or jitter in the signal. When using hysteresis with a rising slope, the trigger asserts when the signal starts below level (or threshold level) and then crosses above level. The trigger deasserts when the signal crosses below level minus hysteresis.

When using hysteresis with a falling slope, the trigger asserts when the signal starts above level (or threshold level) and then crosses below level. The trigger deasserts when the signal crosses above level plus hysteresis.

Figure 3-33 demonstrates the data captured when using hysteresis with a rising and falling edge slope at a level of 2.7 V.

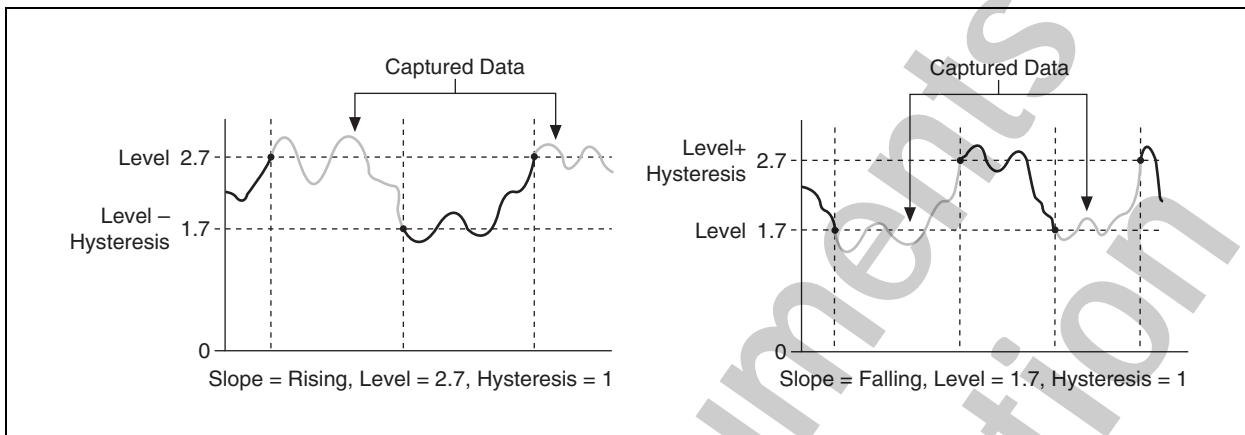


Figure 3-33. Hysteresis with a Rising and Falling Edge Slope

Analog Window Triggering

An analog window trigger occurs when an analog signal passes into (enters) or passes out of (leaves) a window as defined by the two voltage levels: Window Top and Window Bottom. Specify the voltage levels by setting the window top value and the window bottom value. In Figure 3-34, the trigger acquires data when the signal enters the window.

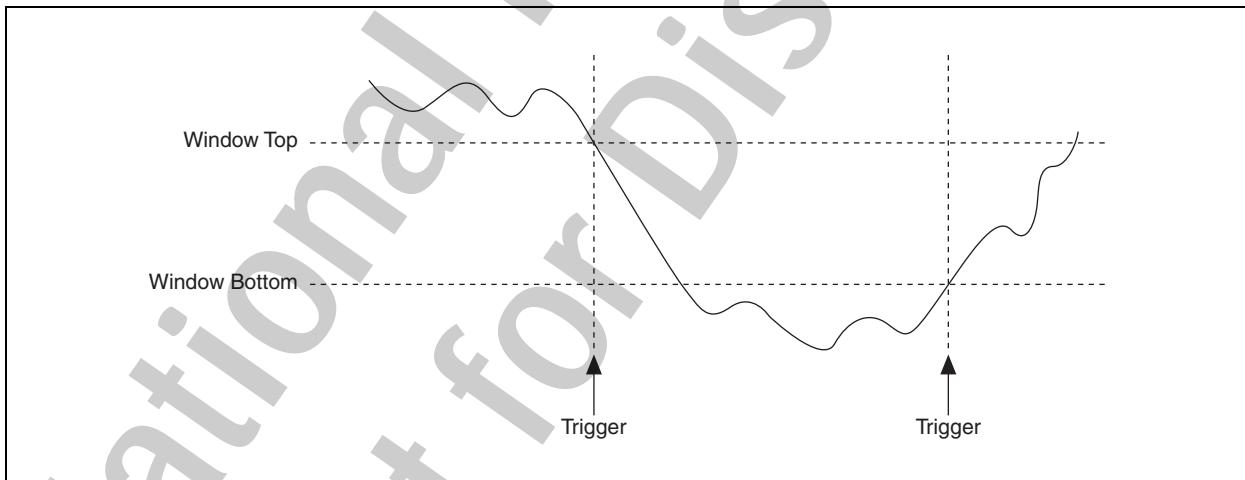


Figure 3-34. Analog Window Triggering—Entering

In Figure 3-35, the trigger acquires data when the signal leaves the window.

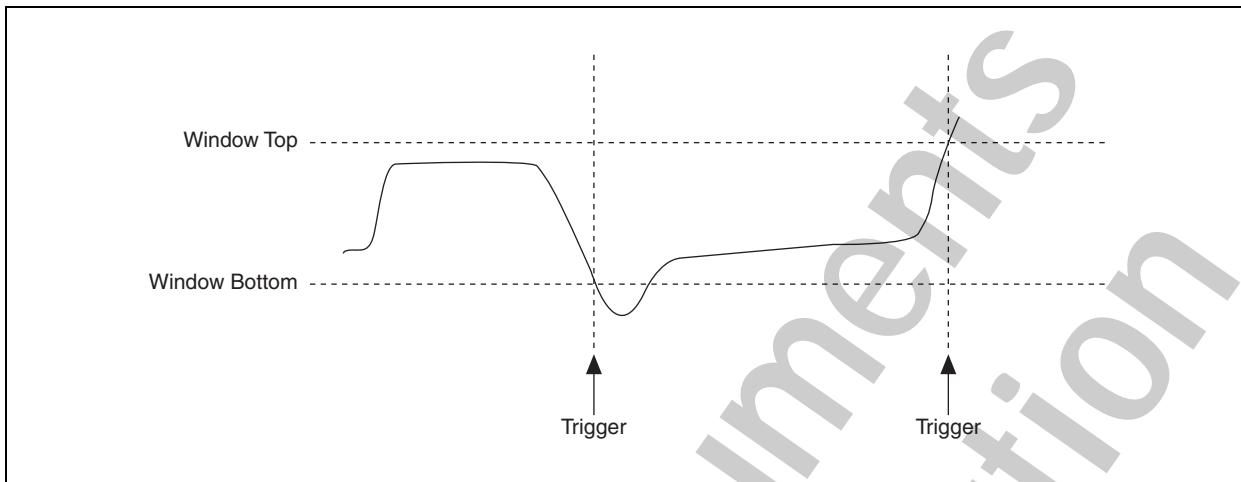


Figure 3-35. Analog Window Triggering—Leaving

Digital Edge Triggering

A digital edge trigger is usually a TTL signal that has two discrete levels: a high level and a low level. A digital signal creates a falling edge when it moves from a high level to a low level. The signal creates a rising edge when it moves from a low level to a high level. You can produce start or reference triggers based on the falling or rising edge of a digital signal, as shown in Figure 3-36. You usually connect digital trigger signals to PFI pins in a National Instruments measurement device.

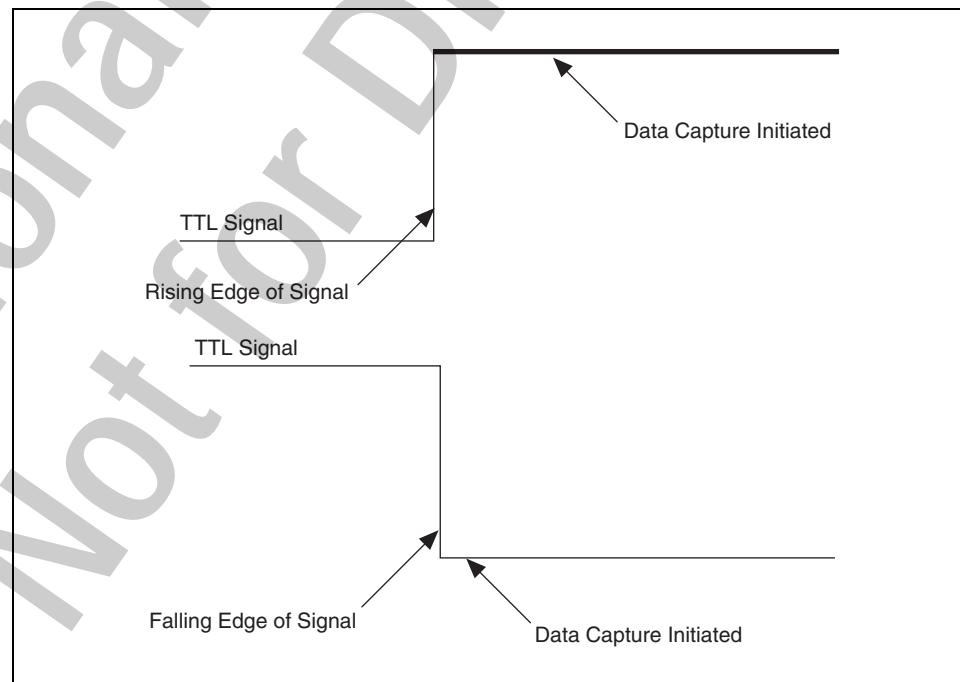


Figure 3-36. Digital Edge Triggering

Actions Caused by Triggering

There are four actions that a trigger can cause. Triggers are named after the actions they cause:

- **Advance Trigger**—Causes a switch device to execute the next entry in its instruction (scan) list.
- **Pause Trigger**—Pauses an acquisition. Deasserting pause trigger resumes an acquisition.
- **Reference Trigger**—Establishes the reference point in a set of input samples. Data acquired up to the reference point is pretrigger data. Data acquired after this reference point is posttrigger data.

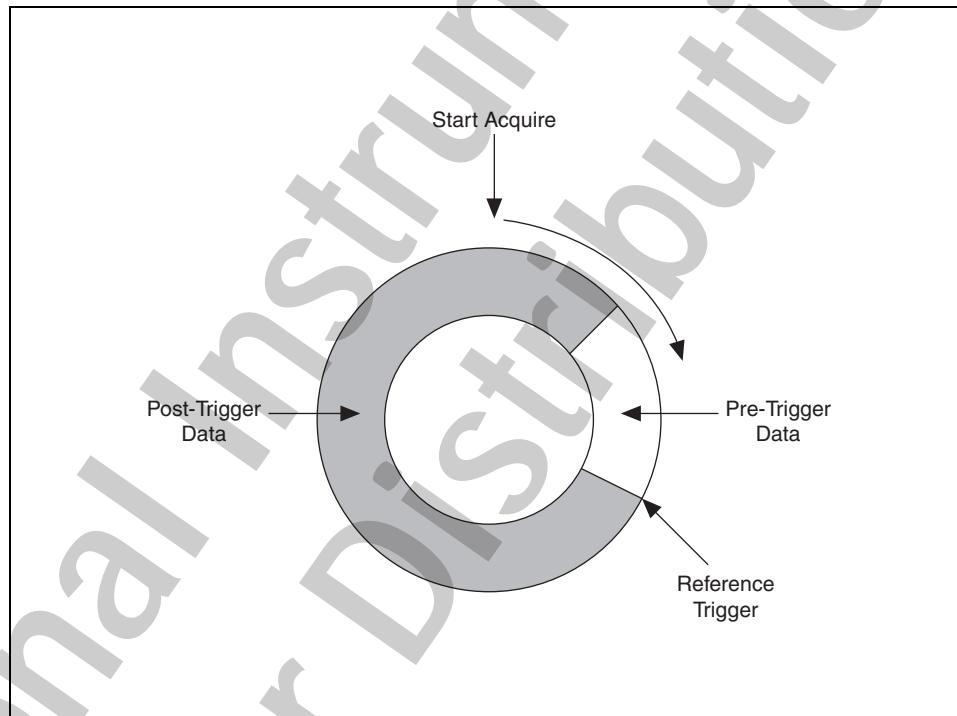


Figure 3-37. Reference Trigger

- **Start Trigger**—Begins an acquisition or generation.

This course describes only start, reference, and pause triggers.

Note Not all DAQ devices support analog triggering. Refer to the device documentation to determine if your device supports analog triggering.

Summary

- Use the Nyquist theorem to determine how fast you need to sample.
- DAQ Channel Names and the waveform data type make DAQ programming easy and flexible.
- Single-point acquisition is software timed, non-buffered, and useful for slow changing signals.
- Multiplexed sampling and simultaneous sampling affect phase relationships of signals differently.
- Buffered acquisition is hardware timed with a regular buffer.
- Continuous acquisition is hardware timed with a circular buffer.
- Buffered acquisitions can be triggered with a digital or analog signal.
- You can stream data to disk for later analysis and presentation.
- Use the NI-DAQmx code generation feature for rapid prototyping.
- Triggers can cause start, reference, pause, or advance actions.
- You can trigger off a digital edge.
- You can trigger off an analog edge or window.
- Many DAQ devices support analog triggering.
- Analog triggering is not a computationally intensive operation.
- Use the DAQ Assistant to test and configure triggering for NI-DAQmx tasks and channels. You also can use the DAQ Assistant to generate code in LabVIEW.

Self-Review: Quiz

1. Which of the following grounding modes should you not use with a grounded signal source?
 - a. Differential
 - b. Referenced single-ended
 - c. Non-referenced single-ended

2. The Nyquist Theorem helps to determine the sampling rate. What problem does this help with?
 - a. Spying
 - b. Noise
 - c. Aliasing
 - d. Isolation

3. Which of the following operations can the DAQmx Read VI be used for?
 - a. Single-point
 - b. Multi-sample
 - c. Multi-channel
 - d. All of the above

4. Software-timed, single-point reads are good for getting information on the shape of a waveform
 - a. True
 - b. False

5. Buffered acquisitions require the use of a clock signal.
 - a. True
 - b. False

National Instruments
Not for Distribution

Self-Review: Quiz Answers

1. Which of the following grounding modes should you not use with a grounded signal source?
 - a. Differential
 - b. Referenced single-ended**
 - c. Non-referenced single-ended

2. The Nyquist Theorem helps to determine the sampling rate. What problem does this help with?
 - a. Spying
 - b. Noise
 - c. Aliasing**
 - d. Isolation

3. Which of the following operations can the DAQmx Read VI be used for?
 - a. Single-point
 - b. Multi-sample
 - c. Multi-channel
 - d. All of the above**

4. Software-timed, single-point reads are good for getting information on the shape of a waveform
 - a. True
 - b. False**

5. Buffered acquisitions require the use of a clock signal.
 - a. True**
 - b. False

Notes

National Instruments
Not for Distribution

Analog Output

In this lesson, you will learn how to generate an analog signal. Much of what you learned about analog input has a parallel in analog output.

Topics

- A. Analog Output Architecture
- B. Single Sample Generation
- C. Finite Buffered Generation
- D. Continuous Buffered Generation
- E. Triggered Generation

National Instruments
Not for Distribution

A. Analog Output Architecture

Typical DAQ devices have a digital-to-analog converter (DAC) for each analog output channel. All DACs are updated at the same time, so only one clock signal is needed. The analog output clock is the update clock. The output of the analog output channels is synchronized, similar to the way analog input channels are synchronized during simultaneous sampling.

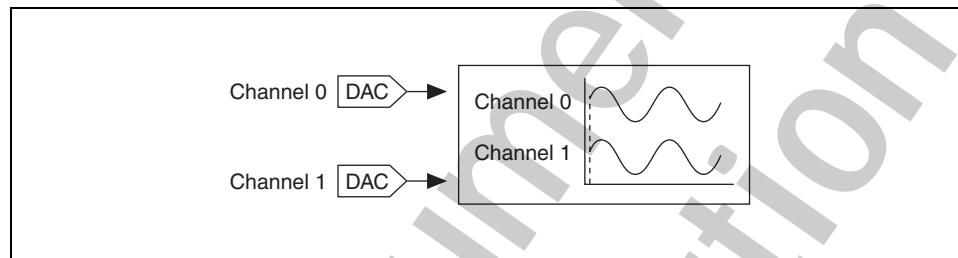


Figure 4-1. Analog Output Channels

Analog Output Considerations

The DAC has a range that is determined by a reference voltage. The reference voltage can be an internal signal or an external signal. The internal reference voltage is a +10 V signal. You can set the range of the DAC as bipolar or unipolar.

Bipolar

A bipolar signal has a range that includes positive and negative values. If you set the device in bipolar mode, the range of the DAC is determined as follows:

- Maximum voltage = $+V_{ref}$
- Minimum voltage = $-V_{ref}$

For example, if you are using the +10 V internal reference voltage, the range of the DAC would be set at -10 to +10 V. However, if the signal only goes from -5 to +5 V, you are not maximizing the resolution of the DAC. To maximize the resolution, you could provide an external reference voltage with a value of +5 V. Now the range of the DAC is -5 to +5 V, the same as the signal, and you are using the full resolution of the DAC to generate the signal.

Unipolar

A unipolar signal has a range that includes only positive values. If you set the device in unipolar mode, the range of the DAC is determined as follows:

- Maximum voltage = $+V_{ref}$
- Minimum voltage = 0 V

For example, if you use the +10 V internal reference voltage, the range of the DAC is set at 0 V to +10 V. If the signal is only 0 V to +5 V, you are not maximizing the resolution of the DAC. To maximize the resolution, you can provide an external reference voltage with a value of +5 V. The range of the DAC is 0 V to +5 V, the same range as the signal, so you will be using the full resolution of the DAC to generate the signal.

B. Single Sample Generation

You can perform a single sample generation using the DAQmx Write VI.

Using the DAQmx Write VI

The DAQmx Write VI located on the **DAQmx - Data Acquisition** palette writes samples to the task or channels you specify. The instances of this polymorphic VI specify the format of the samples to write, whether to write one or multiple samples, and whether to write to one or multiple channels. This lesson describes the analog output instance of the DAQmx Write VI. Use the pull-down menu to select the instance of this VI.

There are four selection windows used to determine the instance of the Write VI. The first selection window allows you to choose the type of output:

- Analog
- Digital
- Counter
- Raw data

The second selection window determines the number of channels to write to or if the data is of the unscaled type. The third selection window allows you to choose to output either a single sample or multiple samples. For a single sample output, the fourth selection window allows you to select the data to be written as either a waveform or double value. For a multiple sample output, the fourth selection window allows you to select the data to be written as a waveform or an array of double values.

For a single sample output, the **auto start** terminal is, by default, set to true. This is because the task state model can be entirely implicitly controlled when outputting a single output. However, for a multiple sample output, the **auto start** terminal is by default set to false. This is because additional timing must be configured when outputting multiple samples with the DAQmx Timing VI, and the DAQmx Start Task VI and DAQmx Clear Task VI must also be used.

Single Sample Generation

If the signal level is more important than the generation rate, output just a single sample. Generate one sample at a time when you need to generate a constant, or DC, signal. To control when your device generates a signal, you can use either software or hardware timing.

- **Software-timed**—The rate at which samples are generated is determined by the software application and the operating system, not the DAQ device. Because the generation is entirely dependent on your operating system's resources, any interruption in the system can possibly affect the generation.
- **Hardware-timed**—A TTL signal, such as a clock on the DAQ device, controls the rate of generation. A hardware-time generation can run at a much faster rate than a software-time generation and maintain a higher value of accuracy. Not all devices support hardware timing. Consult your device's documentation to verify if it does or does not support hardware timing.

Setting the Timing for an Analog Output Generation

To inform DAQmx to use either hardware or software timing, use the DAQmx Timing VI and/or the DAQmx Sample Timing Type property node. By selecting the **Sample Clock** instance of the Timing VI or setting the Sample Timing Type property node to Sample Clock, you are telling DAQmx to use the Sample Clock from your DAQ device to control the generation. For a software-time generation, set the Sample Timing Type property node to **On Demand**. If you do not specify the timing type with the DAQmx Timing VI or the DAQmx Sample Timing Type property node, you will be using a software-timed generation.

In addition, the DAQmx Timing VI contains a **Use Waveform** instance. This instance uses the **dt** component of the waveform input to determine the sample clock rate. **dt** is the time in seconds between samples. This establishes hardware-timing for the analog generation. The Use Waveform instance of the DAQmx Timing VI does not actually output the waveform values, it only uses the waveform to configure the timing. Wire the waveform to the DAQmx Write VI to produce the samples.

C. Finite Buffered Generation

To generate multiple analog output samples, configure the pull-down menu of the DAQmx Write VI for multiple samples. Use a multiple sample generation if you want to generate a time-varying signal, such as an AC sine wave. Multiple point generation also is known as buffered analog output. Buffered analog output can be either finite or continuous. However, both buffering methods involve the following two main steps:

1. Write samples into the buffer. The points are taken from LabVIEW and placed in an intermediate memory buffer before they are sent to the device. A buffered generation is similar to sending an entire email at once instead of sending it one word at a time.
2. Transfer samples from the buffer to the DAQ device. The rate at which the samples are transferred depends on the timing that you specify. As with single sample generation, you can use either a software or hardware-timed generation.

In a hardware-timed generation, a hardware signal called the update clock controls the rate of generation. A hardware clock can run much faster than a software loop, so you can generate a wider range of signal frequencies and shapes. A hardware clock also is more accurate than a software loop. A software loop rate can be interfered with by a variety of actions, such as the opening of another program on the computer.

Finite Buffered Generation Implementation

Figure 4-2 shows a flowchart of a buffered generation.

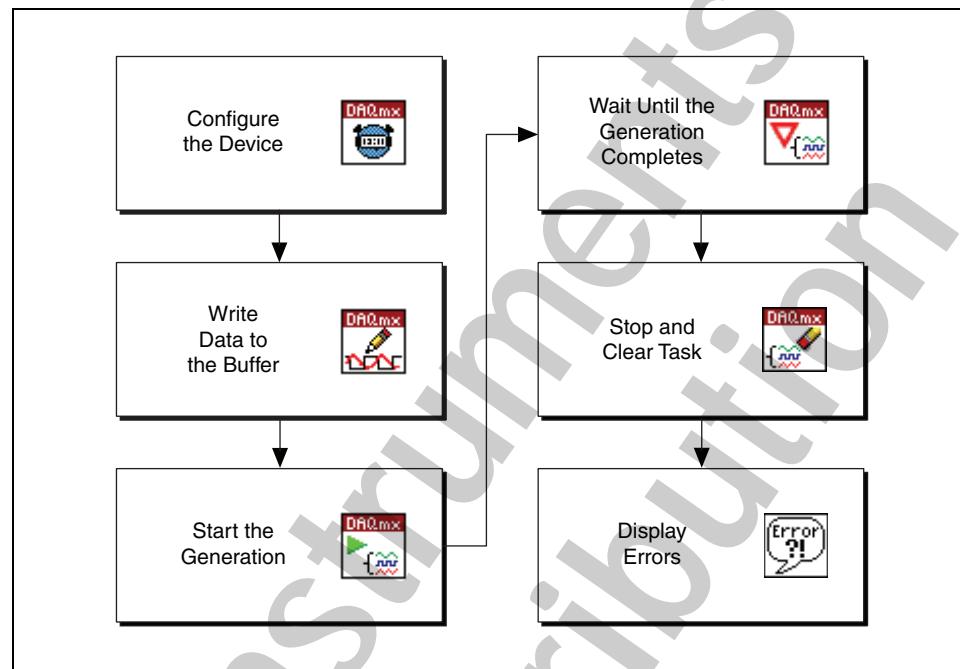


Figure 4-2. Finite Buffered Generation Flowchart

Figure 4-3 shows an example of a typical finite buffered generation using the Sample Clock and an array of doubles for the output data.

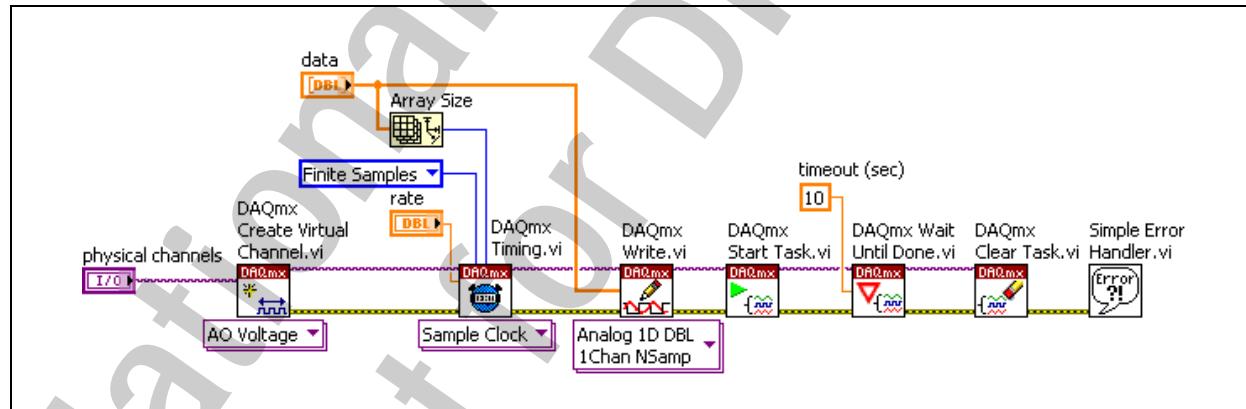


Figure 4-3. Finite Buffered Generation VI Block Diagram

You also can use the waveform data type to determine the timing and sample data, as seen in Figure 4-4.

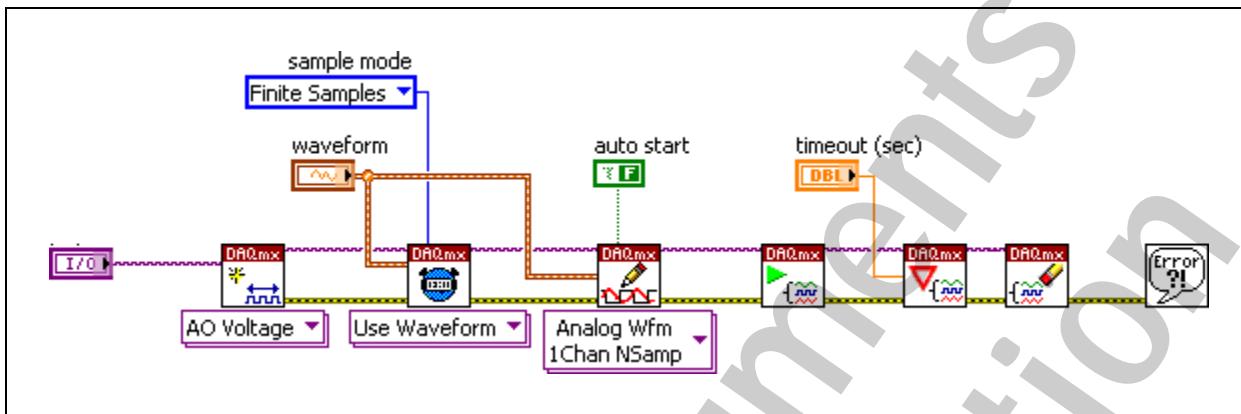


Figure 4-4. Using Waveform Timing

There are few differences between the two types of finite buffered generation—using an array of doubles with the Sample Clock and using a waveform data type to set the Sample Clock and samples. The DAQmx Timing VI instance changes, as well as the data that is wired to the **data** terminal of the DAQmx Write VI. Both types follow the same general structure that we will describe now.

The DAQmx Create Virtual Channel VI can be used to programmatically create an analog output virtual channel. If you have already created your virtual channel or task using the DAQ Assistant in MAX, you can skip this VI and wire the channel/task to the next VI, the DAQmx Timing VI.

The DAQmx Timing VI has two instances that we can use for analog output—Sample Clock and Use Waveform. Since we are generating a finite number of samples, set the **sample mode** to Finite Samples for both instances. When using the Sample Clock, we also specify the generation **rate** and **number of samples**. The number of samples value determines the buffer size. For the Use Waveform instance, simply wire the waveform data type to the **waveform** terminal. This instance of the VI will determine the sample clock rate and number of samples (buffer size) based on the data contained in the waveform.

The DAQmx Write VI actually sends the data to the PC buffer. You can select to output either a waveform data type or an array of doubles. For the Use Waveform instance of the DAQmx Timing VI, select to output a waveform type from the pull-down menu of the Write VI. Wire the same waveform you used to set the timing to the **data** terminal of the Write VI. When using the built-in Sample Clock for timing, select to output an array of doubles from the pull-down menu of the Write VI. Then, wire the array of doubles you want to generate to the **data** terminal of the Write VI.

For multiple samples, the **auto start** parameter defaults to a False value. Since we explicitly start the task, wait for it to finish, and then stop the task, we want to leave the auto start value to False.

The DAQmx Start VI begins the generation. The DAQmx Wait Until Done VI waits until the task has completed, or a timeout occurs. When either of these occur, control is then passed to the DAQmx Clear Task VI and the task is stopped and cleared. As is usual with the LabVIEW programming in this course, the error cluster is passed to each VI and if necessary, an error message is displayed.

DAQmx Reset VI

In analog output, when you write a value to an analog output channel, the channel continues to output that voltage until a new value is written to the analog output channel or the device is reset by the DAQmx Reset VI (Located on the **DAQmx - Data Acquisition»DAQmx Device Configuration** palette) or the device is powered off.

Assume you are writing a sine wave to an analog output channel and the last value in the buffer is seven. You generate the entire sine wave, and after the generation is complete, the analog output channel continues to generate a voltage of seven. Rather than resetting the device every time, it is easier to write a value of zero to the channel after the generation is complete. In Traditional NI-DAQ, you can use the AO Write One Update VI located on the **Utility** palette to perform just such an operation.

Output Waveform Frequency

The frequency of the output waveform depends on the update rate and the number of cycles of the waveform present in the buffer, as illustrated in Figure 4-5.

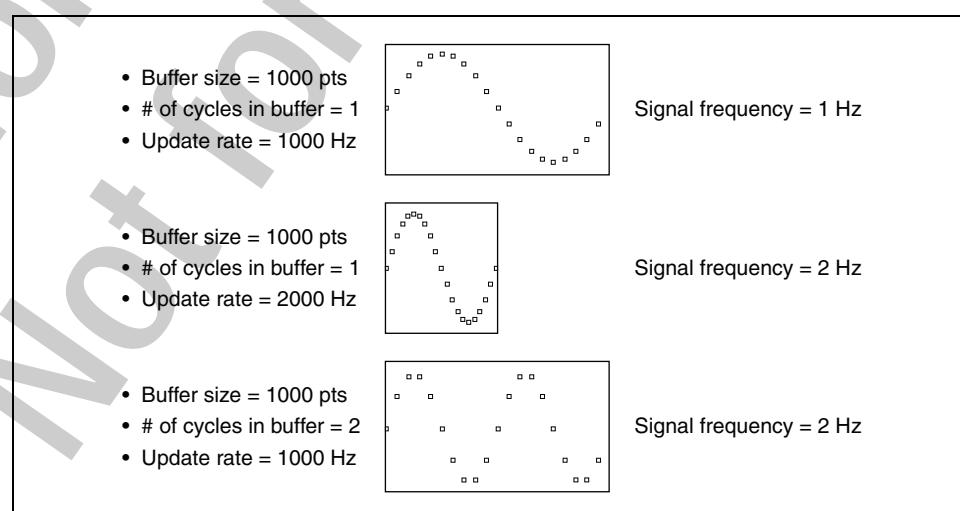


Figure 4-5. Output Waveform Frequency Examples

The formula for calculating the frequency of the output signal is as follows:

$$\text{signal frequency} = \text{cycles} \times [(\text{update rate})/(\text{points in the buffer})]$$

The following example illustrates how the update rate and the number of cycles of the waveform in the buffer affect the signal frequency. Assume you have a 1,000 point buffer that holds one cycle of the waveform. If you generate the signal with an update rate of 1 kHz, the signal frequency is as follows:

$$(1 \text{ cycle}) \times [(1,000 \text{ points per second})/(1,000 \text{ points})] = 1 \text{ Hz}$$

If you double the update rate and leave everything else the same, the signal frequency is as follows:

$$(1 \text{ cycle}) \times [(2,000 \text{ points per second})/(1,000 \text{ points})] = 2 \text{ Hz}$$

If you double the number of cycles in the buffer and leave everything else the same, the signal frequency is as follows:

$$(2 \text{ cycles}) \times [(1,000 \text{ points per second})/(1,000 \text{ points})] = 2 \text{ Hz}$$

Therefore, if you double the update rate or the number of cycles, you double the frequency of the output waveform.

D. Continuous Buffered Generation

The main difference between a finite buffered generation and continuous buffered generation is the number of points that are generated. In a finite buffered generation, you generate the data in the buffer a finite number of times. In a continuous buffered generation, you can generate data indefinitely.

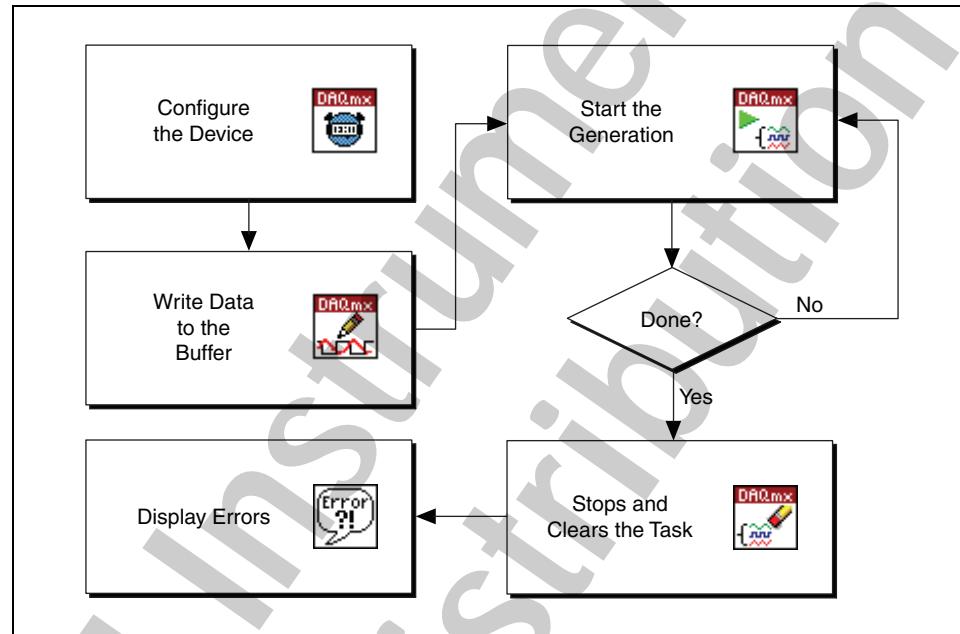


Figure 4-6. Continuous Buffered Generation Flowchart

Figure 4-7 is similar to a buffered generation with the following differences:

- The DAQmx Timing VI is set to Continuous Samples sample mode.
- The DAQmx Is Task Done VI is used instead of the DAQmx Wait Until Done VI in a While Loop.

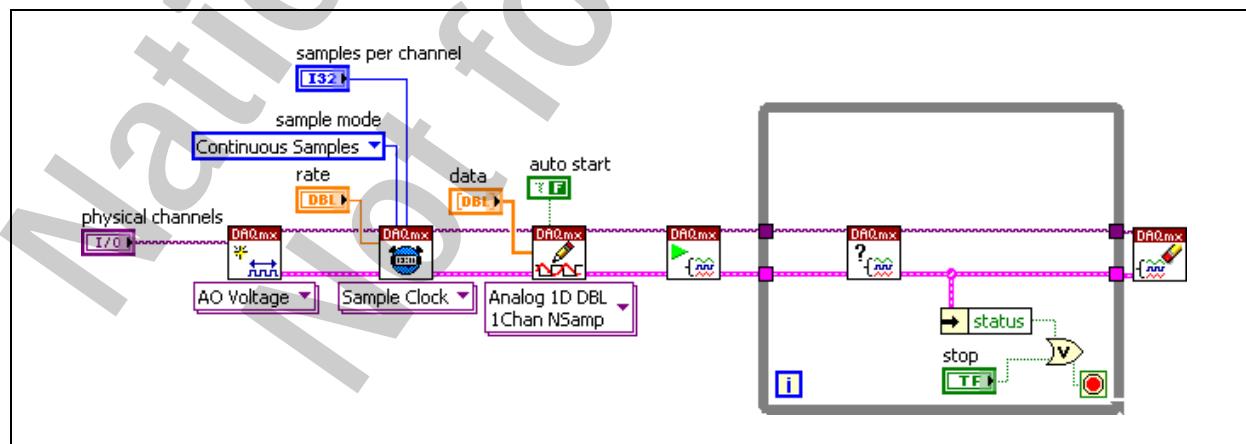


Figure 4-7. Continuous Buffered Generation VI Block Diagram

You start by configuring the virtual channel and timing settings with the Create Virtual Channel VI and Timing VI. Next, you write to the buffer with the DAQmx Write VI and start the task with DAQmx Start Task VI.

The While Loop polls the task to see if it has completed by using the DAQmx Is Task Done VI. The generation stops when either the user clicks the stop button or an error occurs. The data in the buffer is being regenerated after it has all been processed. After the While Loop stops, the DAQmx Clear Task VI stops acquisition and clears the task, and then errors are reported.



Note You can write new data to the buffer with each loop iteration. You must place a Write VI in the while loop and wire the new data to the data terminal. Using a Write property node, select the Regeneration Mode property and set it to not allow regeneration. You need to make sure you generate new data fast enough to prevent the buffer from regenerating old data. This situation is similar to performing a continuous buffered acquisition. You must make sure you read the data out of the buffer fast enough to prevent the data from being overwritten.

Regeneration

Generating the same data more than once is called regeneration. You can configure NI-DAQmx to allow or disallow regeneration by setting the Regeneration Mode attribute/property. By default, NI-DAQmx allows regeneration for sample clock timing and disallows it for handshaking or burst handshaking timing. When regeneration is disallowed, new data must be continuously written to the device.

Allowing Regeneration and Using Onboard Memory

When the **Use Only Onboard Memory** attribute/property is true, NI-DAQmx transfers data only once to the device and that data is continually regenerated from there. Attempting to write new data to the device after starting the task returns an error. In addition, the amount of data written to the device before starting the task must fit in the onboard memory of the device.

When the **Use Only Onboard Memory** attribute/property is false, NI-DAQmx continuously transfers data from the host memory buffer to the device even though this data is not changing. Thus, if you write new data to the device after starting the task, that new data is generated and regenerated until you write more new data. This type of regeneration is sometimes called PC memory or user buffer regeneration.

When this attribute/property is false, you can also set the **Data Transfer Request Condition** attribute/property to specify when to transfer data from the host buffer to the device.

Glitching

Glitching refers to the generation of a waveform in which, when transitioning from old samples in the buffer to new samples, a mixture of old and new samples is generated rather than just the new samples. This situation may occur when continuously generating samples if the Regeneration Mode write attribute/property is set to Allow Regeneration. Glitching occurs when, while you write new samples, a subset of these new samples are generated and then, since you have not finished writing all of the new samples, a subset of the old samples is generated. After your write operation completes, only the new samples are generated.

NI-DAQmx reduces the likelihood of glitching by ensuring that the writing of new samples does not overtake the generation. This glitching protection works by pausing the write until the total samples generated is more than one buffer ahead of the current write position. However, NI-DAQmx does not ensure that the generation does not overtake the new samples being written. If this occurs, a glitch results, and NI-DAQmx reports the `kWarningPotentialGlitchDuringWrite` warning (error 200015). The following suggestions can help you to avoid generating glitches:

- Write new samples that are almost one buffer ahead of the total samples generated. By writing the new samples almost one buffer ahead of the total samples generated, there is less of a chance that the generation overtakes the new samples that are being written. If you are updating the entire buffer at one time, wait to write the new samples until the total samples generated attribute/property is one sample greater than an integral number of buffer sizes. For example, if the buffer size is 1000 samples, wait to write new samples until the total samples generated is either 1001, 2001, 3001, and so on.
- Increase the buffer size. If the buffer size is larger, there is less of a chance that the generation overtakes the new samples that are being written.
- Decrease the sample clock rate. If the sample clock rate is slower, there is less of a chance that the generation overtakes the new samples that are being written.

In Figures 4-8 and 4-9, the sine wave is generated from old samples and the square wave is generated from the new samples. Figure 4-8 depicts glitching.

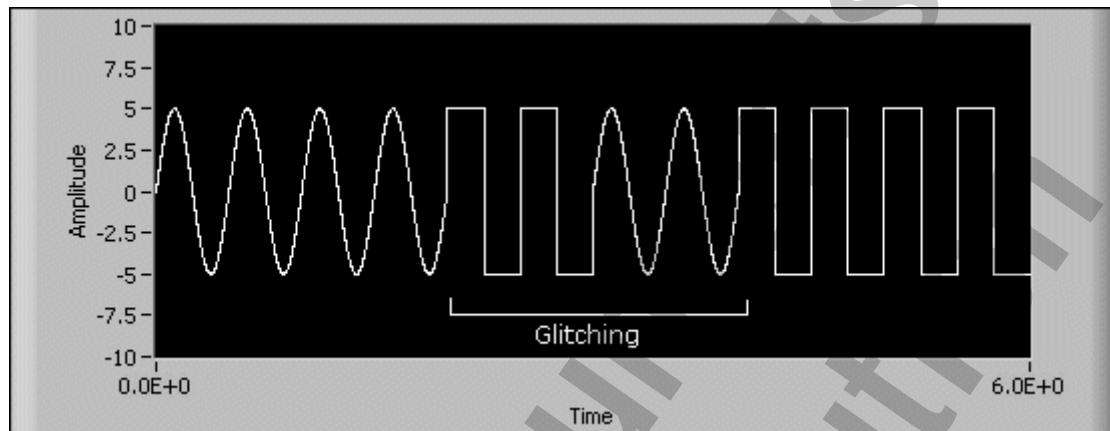


Figure 4-8. Waveforms With Glitching

Figure 4-9 depicts the same waveforms without glitching.

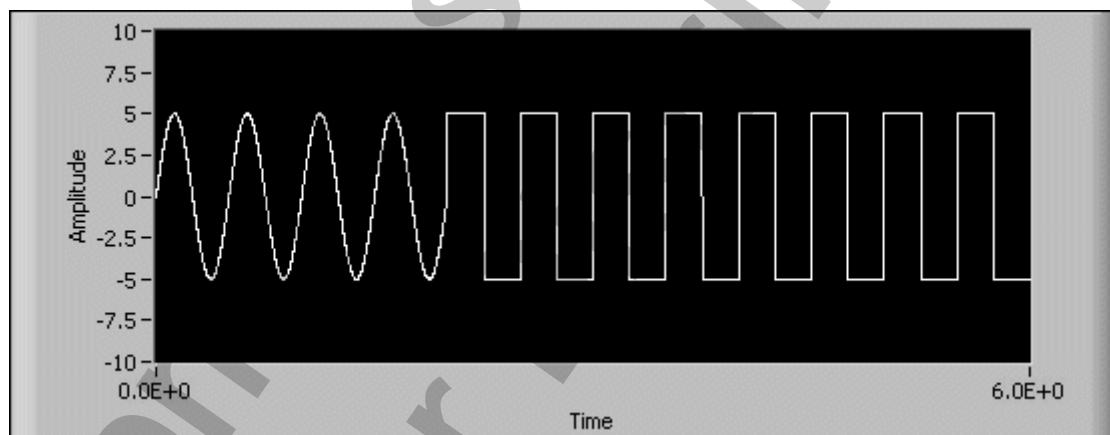


Figure 4-9. Waveforms Without Glitching

E. Triggered Generation

To perform a triggered generation, use the DAQmx Trigger VI, as shown in Figure 4-10.

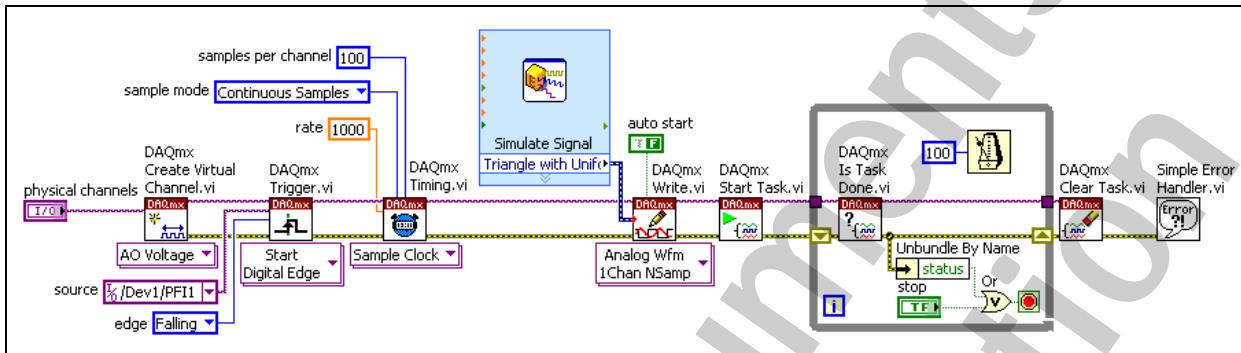


Figure 4-10. Triggered Generation Example

Summary

- Typical DAQ devices have a DAC for each analog output channel.
- DAQmx VIs allow single-point, buffered, continuous, or triggered generation.
- The output waveform frequency depends on the update rate and the number of cycles of the waveform in the buffer.
- To output waveform data including the waveform timing properties, use the Use Waveform instance of the DAQmx Timing VI.

National Instruments
Not for Distribution

National Instruments
Not for Distribution

Self-Review: Quiz

1. In a typical DAQ device how many channels are there per DAC?
 - a. 1
 - b. 8
 - c. 16
 - d. 32

2. If you generate a sinusoidal waveform with 200 samples and 10 cycles at an output rate of 1 kHz, what is the apparent rate of the sine wave?
 - a. 1000 Hz
 - b. 500 Hz
 - c. 50 Hz
 - d. 20 Hz

National Instruments
Not for Distribution

National Instruments
Not for Distribution

Self-Review: Quiz Answers

1. In a typical DAQ device how many channels are there per DAC?
 - a. 1
 - b. 8
 - c. 16
 - d. 32

2. If you generate a sinusoidal waveform with 200 samples and 10 cycles at an output rate of 1 kHz, what is the apparent rate of the sine wave?
 - a. 1000 Hz
 - b. 500 Hz
 - c. **50 Hz**
 - d. 20 Hz

National Instruments
Not for Distribution

Notes

National Instruments
Not for Distribution

Digital I/O

This lesson describes the digital functionality of a DAQ device, which can perform digital input and digital output.

Topics

- A. Digital Overview
- B. Digital I/O
- C. Hardware-Timed Digital I/O

National Instruments
Not for Distribution

A. Digital Overview

The digital lines on a DAQ device accept and generate TTL-compatible signals. A TTL signal has two states—logic low and logic high, as shown in Figure 5-1. Logic low signals are signals between 0 to +0.8 V. Logic high signals are signals between +2 to +5 V. Signals between +0.8 and +2.0 V are indeterminate.

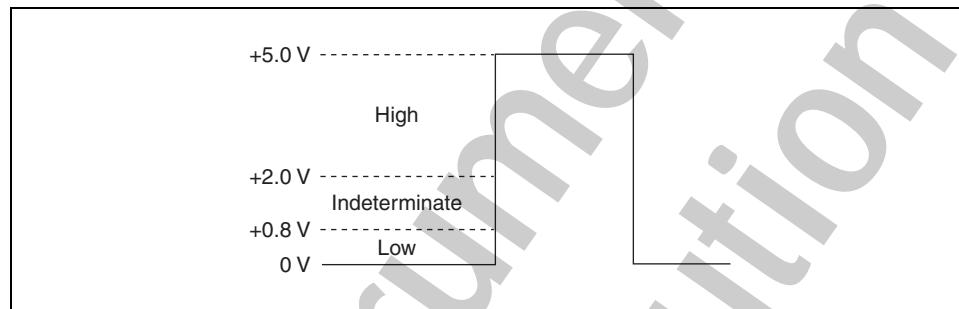


Figure 5-1. TTL Signal States

To ensure that digital lines measure the signal correctly, make sure the voltage level of the signal is never between +0.8 and +2 V.

Digital Terminology

Following are some of the common terms used with digital I/O operations.

- **Bit**—The smallest unit of data used in a digital operation. Bits are binary, so they can be a 1 or a 0.
- **Byte**—A binary number consisting of 8 related bits of data. Bytes are also used to denote the amount of memory required to store one byte of data.
- **Line**—An individual signal in a digital port. The difference between a bit and a line is that the bit refers to the actual data transferred, and the line refers to the hardware the bit is transferred on. However, the terms line and bit are fairly interchangeable. For example, an 8-bit port is the same as a port with eight lines.
- **Port**—A collection of digital lines. Usually the lines are grouped into an 8-bit port. Typical multifunction I/O DAQ devices have multiple 8-bit ports, each containing eight lines of bidirectional DIO signals. LabVIEW VIs often refer to a port as a digital channel.

- **Port Width**—The number of lines in a port. For example, some DAQ devices have three ports each containing 8 digital lines. Therefore, the port width is eight.
- **Mask**—Determines if a digital line is ignored. For example, if you write to a port, but you do not want to write to all the lines, you can set a mask to ignore the lines you do not want to write to.

NI-DAQmx Digital Notation

In NI-DAQmx, use the following conventions to describe digital lines and ports. In each case, X corresponds to the device number of your DAQ device, Y corresponds to a digital port, and A and B correspond to digital lines on your DAQ device.

- **Port**— $\text{Dev}X/\text{Port}Y$
- **Single Line**— $\text{Dev}X/\text{Port}Y/\text{Line}A$
- **Multiple Lines**— $\text{Dev}X/\text{Port}Y/\text{Line}A:\text{B}$. The lines are read (or written) in ascending order, starting at Line A and continuing to Line B. To explicitly control the order in which the digital lines are read (or written), use the notation $\text{Dev}X/\text{Port}Y/\text{Line}A$, $\text{Dev}X/\text{Port}Y/\text{Line}B$. The comma separates each digital line.

Digital Channel Data Formats

Digital channel data format deals with the type of the digital data that is read or written. You can choose which data format to use when using the DAQmx Read VI and DAQmx Write VI.

Line Format (Boolean)

The line format represents each line within a channel as a single Boolean value (a single byte). The states of the data are limited to 1s (true) and 0s (false). Line formats are only provided for single sample reads and writes.

Use the line format when it is convenient for manipulating or displaying the digital data. A typical application is controlling or reading back relay states. For high-speed digital applications, you should generally not use the line format.

Port Format (Integer)

The port format matches the native format of digital devices that can represent only two digital states and organize individual lines into collections known as ports. For more information, refer to the *Digital Data (Integer Format)* topic in the *DAQmx Help*.

The port format is the most efficient in terms of space, as it requires only a bit of memory per line. In addition, the port format is often the most efficient in time as it matches the native format of many devices.

The largest integer supported is 32 bits; therefore, you can read and write digital channels with no more than 32 lines when using the port format.

Waveform

The waveform data format includes the channel name and timing information with the actual data represented in a dedicated digital format. Your ADE provides a mechanism for extracting and setting individual parts of the waveform.

The dedicated digital format represents digital data similar to logic analyzers and digital simulation tools. Each channel has no limits on the number of lines. In addition, the digital format allows for additional states beyond basic 1s and 0s. The ADE can take advantage of this format by tailoring data and graph displays for the digital data.

For input tasks, you can use the additional information for a variety of purposes. For example, you can update graphs to show the timing information and include labels with the channel names. Because there is overhead associated with including this additional information, NI-DAQmx allows you to configure the information you want to include.

For output tasks, the timing information is the primary field that is useful. A waveform generated by a library may include timing information that you can use to set up the timing for your output task.

When reading data, the waveform data includes the time when the first sample in the waveform was acquired, t_0 , and the amount of time that elapsed between each sample, dt . However, there are limitations on these two values.

B. Digital I/O

To perform digital I/O in NI-DAQmx, select a digital instance of the DAQmx Read VI or DAQmx Write VI. In addition to these VIs, use the DAQmx Timing VI and DAQmx Triggering VI to configure digital I/O tasks. To programmatically create a digital channel, use the DAQmx Create Virtual Channel VI.

DAQmx Create Virtual Channel VI

To programmatically create a digital input or output channel, select the Digital Input or Digital Output instance of the DAQmx Create Virtual Channel VI. These instances of the VI allows you to create a channel composed of a digital port, a digital line, or a collection of digital lines. In addition, use the **line grouping** input to determine the manner in which the channel organizes the digital lines. You can select to create one channel for each line or to create one channel for all lines.

DAQmx Read VI

The DAQmx Read VI reads samples from the task or channels you specify. The instances of this polymorphic VI specify what format of samples to return, whether to read a single sample or multiple samples at once, and whether to read from one or multiple channels. Select a digital instance from the pull-down menu to perform digital input.

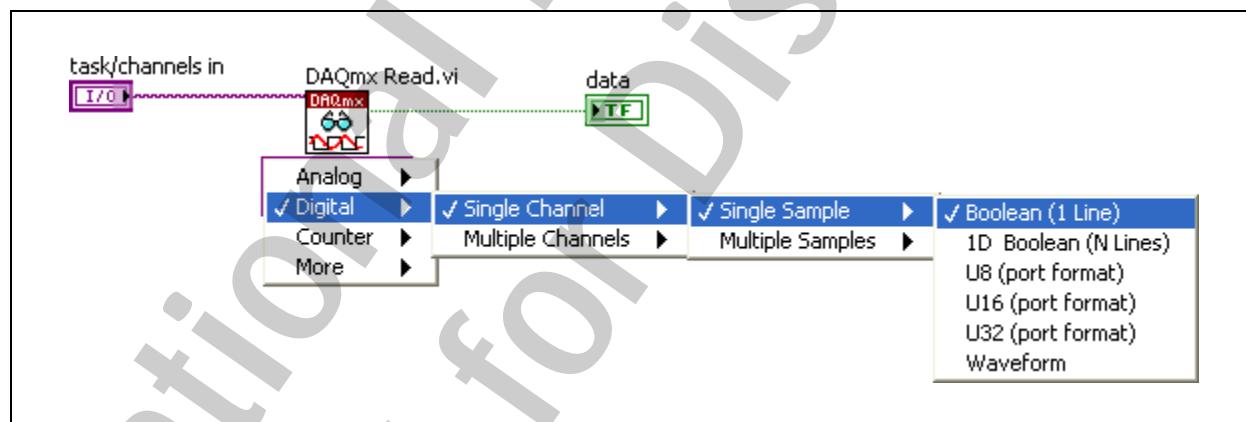


Figure 5-2. DAQmx Read VI—Digital Instance

Select to read either a single channel or multiple channels. If the channel line grouping is set to one channel for all lines, reading a single channel returns all the values on each of the lines in the channel. If the channel line grouping is set to one channel for each line, you must read multiple channels in order to read the values on each of the digital lines specified.

Select to read either a single sample or multiple samples at once. The data type options for the return value(s) allow you to return the value in line or port format. Line format consists of a single Boolean value or an array of Boolean values (for multiple lines). An unsigned 8-bit integer (U8) or unsigned 16-bit integer (U16) or unsigned 32-bit integer (U32) are the options for returning the value in port format. When reading multiple channels, the data type options are the same, with an array dimension added to each type to account for the multiple channel reading.

If you select the U8 or U16 or U32 port format to return data, use the Number to Boolean Array function located on the **Numeric»Conversion** palette, to convert the number into an array of Boolean values.

DAQmx Write VI

The DAQmx Write VI writes samples to the task or channels you specify. The instances of this polymorphic VI specify the format of the samples to write, whether to write one or multiple samples, and whether to write to one or multiple channels. Select a digital instance of the DAQmx Write VI to perform digital output.

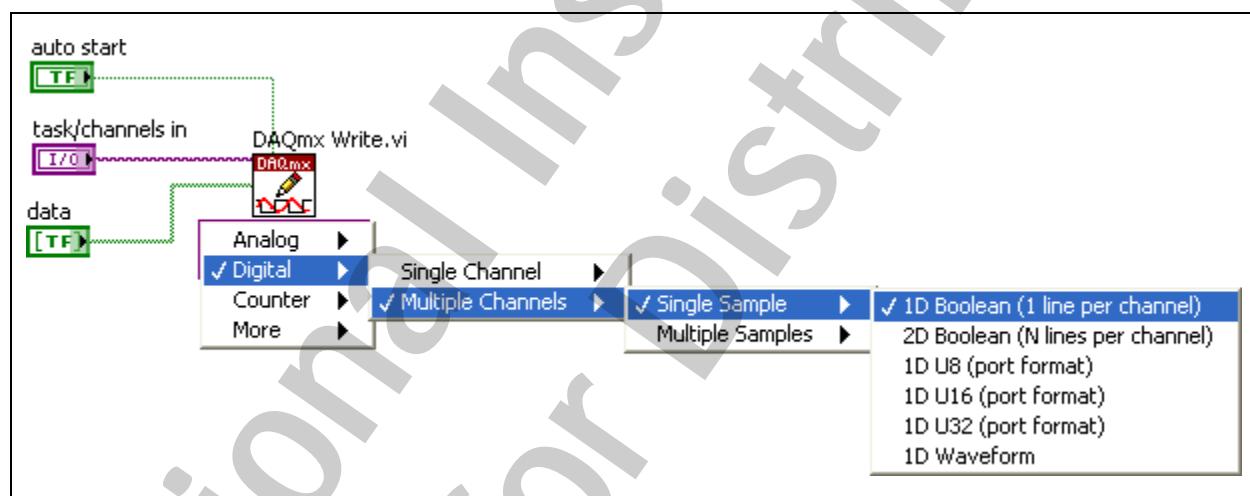


Figure 5-3. DAQmx Write VI—Digital Instance

The settings for the digital instance are configured in the same manner as the digital instance of the DAQmx Read VI.

By default, the **auto start** input of the DAQmx Write VI is True when writing single samples and False when writing multiple samples. If you use the DAQmx Start VI and DAQmx Stop Task VI, always set the **auto start** input to False. This allows you greater control of the task state model and improves the speed of your program.

DAQmx Timing VI

The handshaking instance of the DAQmx Timing VI determines the number of digital samples to acquire or generate using digital handshaking between the device and a peripheral device. Instead of specifying a sampling rate, specify the number of digital samples to acquire or generate using digital handshaking.



Note Not all devices support digital handshaking. Consult your device documentation to see if handshaking is supported on your device.

DAQmx Trigger VI

Use the DAQmx Trigger VI to configure triggering for the task. The instances of this polymorphic VI correspond to the trigger and trigger type to configure. Configure digital triggers settings the same way you configure analog input and analog output triggers. Refer to Lesson 3, *Analog Input*, of this manual for more information about configuring triggers.

C. Hardware-Timed Digital I/O

If your DAQ device supports hardware-timed digital I/O, you can program hardware-timed digital acquisitions and generations similar to the way you program hardware-timed analog acquisitions and generations.

Onboard Digital I/O Sample Clock

Some DAQ devices, such as X Series devices, have a dedicated onboard sample clock for digital I/O. You can program hardware-timed acquisitions and generations the same way you program hardware-timed analog acquisitions and generations using the DAQmx Timing VI.

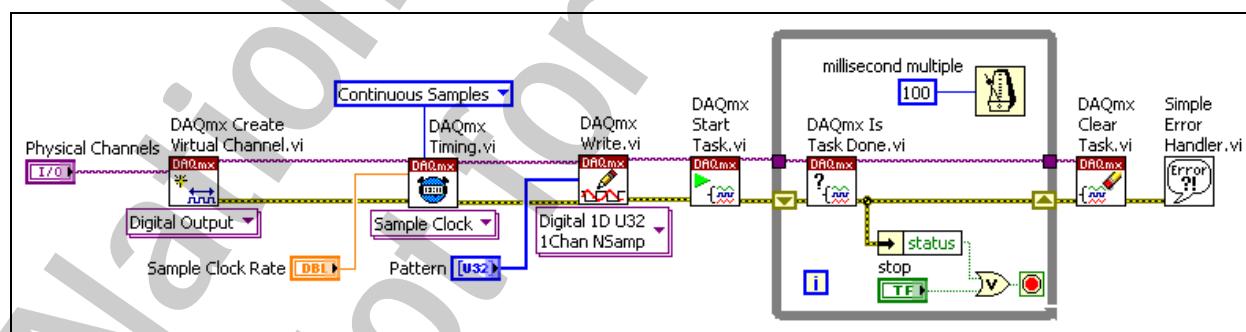


Figure 5-4. Hardware-timed Digital Generation Using DIO Sample Clock

Correlated Digital I/O

Some DAQ devices, such as M Series devices, do not have a dedicated onboard sample clock for digital I/O, but do support correlated digital I/O. These devices can use a different clock, typically the AI or AO sample clock or an external clock, for hardware-timed acquisitions and generations.

Specify which clock to use with the source input of the DAQmx Timing VI.

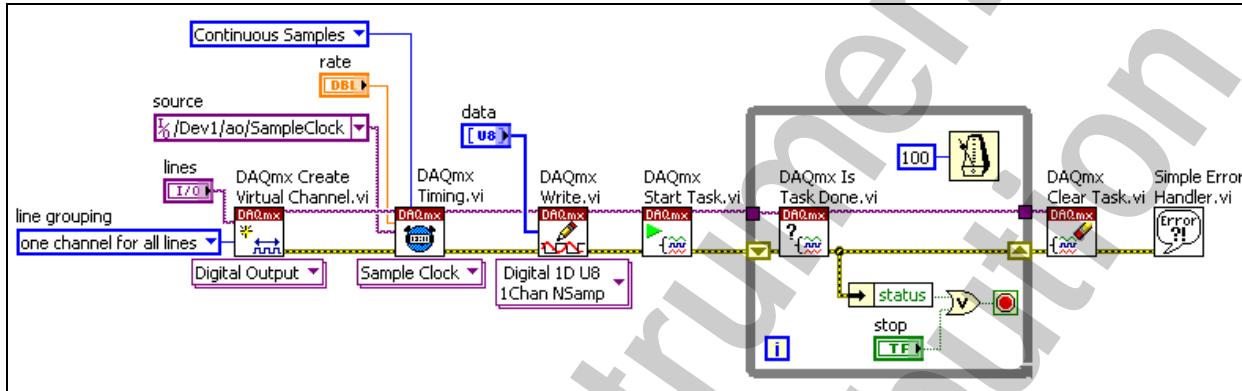


Figure 5-5. Correlated Digital Output Example VI Block Diagram

Hardware-Timed Digital I/O Caveat

A few DAQ devices do not support hardware-timed digital I/O because they do not have a dedicated onboard digital I/O sample clock and do not support correlated digital I/O.

Summary

- Digital lines on a typical DAQ device can:
 - Read and write TTL-compatible signals.
 - Perform only non-timed digital I/O.
 - Be configured individually for input or output.
- Typical DAQ devices have digital lines that can read and write TTL-compatible signals.
- The specifications for a TTL signal define 0 to +0.8 V as logic low and +2 to +5 V as logic high. Between +0.8 and +2 V can be interpreted as high or low.
- Use the DAQmx Read VI and DAQmx Write VI to easily configure digital input and output operations.
- For hardware-timed digital I/O, use an onboard digital I/O sample clock or correlated digital I/O.

National Instruments
Not for Distribution

National Instruments
Not for Distribution

Self-Review: Quiz

1. Digital signals are always between 0 and 5 Volts.
 - a. True
 - b. False

2. All DAQ devices have a dedicated onboard sample clock for digital I/O.
 - a. True
 - b. False

National Instruments
Not for Distribution

National Instruments
Not for Distribution

Self-Review: Quiz Answers

1. Digital signals are always between 0 and 5 Volts.
 - a. True
 - b. **False**

2. All DAQ devices have a dedicated onboard sample clock for digital I/O.
 - a. True
 - b. **False**

National Instruments
Not for Distribution

Notes

National Instruments
Not for Distribution

Counters

This lesson focuses on the counter functionality of a DAQ device. This lesson contains an overview of counters including counter signals, the parts of a counter, the pins you connect a counter signal to, basic counter terminology, and different types of counter chips. The lesson also describes the DAQmx VIs used for counter operations.

Topics

- A. Counter Signals
- B. Edge Counting
- C. Pulse Generation
- D. Pulse Measurement
- E. Frequency Measurement
- F. Position Measurement

A. Counter Signals

Counters operate with TTL-compatible signals. A TTL-compatible signal has the following specifications:

$$0 \text{ V} - 0.8 \text{ V} = \text{logic low}$$

$$2 \text{ V} - 5 \text{ V} = \text{logic high}$$

$$\text{Maximum Rise/Fall Time} = 50 \text{ ns}$$

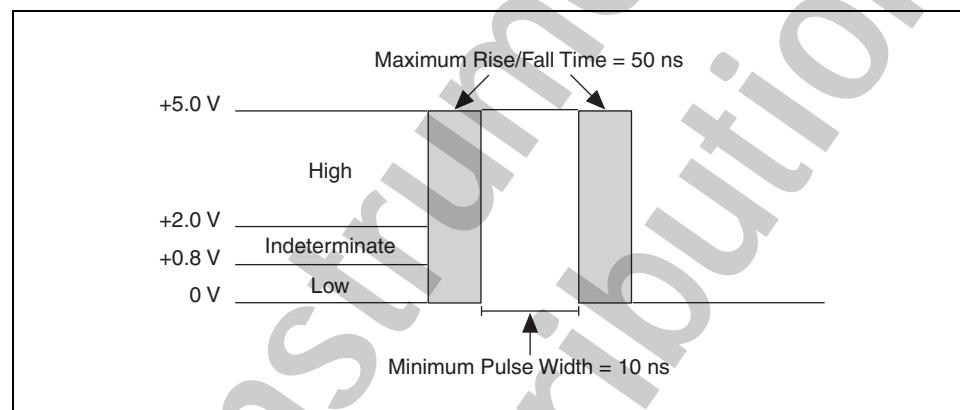


Figure 6-1. TTL-Compatible Signal Specifications

Digital I/O devices can set or monitor the state of a digital line. Counters, however, are not only concerned with the state of the signal but also with the transition from one state to another. A counter can detect rising edges (transition from logic low to logic high) and falling edges (transition from logic high to logic low). Two important parameters related to sensing rising and falling edges are the rise/fall time and the minimum pulse width. The rise/fall time is a measure of how quickly the signal transitions from low to high or high to low. For a counter to detect the edge, the transition must occur within 50 ns or less, as defined by the specifications for a TTL-compatible signal.

In addition to this time restraint, there must be a minimum delay from the time a counter detects a rising or falling edge until it can detect another rising or falling edge. This delay is known as the minimum pulse width. The minimum pulse width depends on the counter chip that is used. E Series devices have a DAQ-STC chip, which has minimum pulse width of 10 ns for both the source and the gate. Consult the hardware documentation for the specific DAQ device to determine the minimum pulse width required for the counters.

M Series devices have a custom designed NI-STC 2 chip. The NI-STC 2 is an application-specific integrated circuit that controls interboard and intraboard synchronization and timing for multifunction DAQ operations. The NI-STC 2, used on all M Series devices, was also designed to be compatible with the latest ADC technology, including the 18-bit ADCs used on the high-accuracy devices.

There are five different types of counter measurements—edge counting, pulse generation, pulse measurement, frequency measurement, and position measurement.

Parts of a Counter

A counter has the following main components:

- **Count Register**—Stores the current count of the counter. You can query the count register with software.
- **Source**—An input signal that can change the current count stored in the count register. The counter looks for rising or falling edges on the source signal. Whether a rising or falling edge changes, the count is software-selectable. The type of edge selected is called the active edge of the signal. When an active edge is received on the source signal, the count changes. Whether an active edge increments or decrements the current count is also software-selectable. The source signal must be TTL-compatible.
- **Gate**—An input signal that determines if an active edge on the source changes the count. Counting can occur when the gate is high, low, or between various combinations of rising and falling edges. Gate settings are made in software. The gate is similar to a line mask in digital I/O because it allows you to acknowledge or ignore active edges on the source.
- **Out**—An output signal that generates pulses or a series of pulses, otherwise known as a pulse train. The output signal is TTL-compatible.

Counter Pins

Analog input, analog output, and digital I/O all have dedicated pins for input or output operations. Counters use a combination of Programmable Function Input (PFI) pins and dedicated pins for their operations. The output pins for counters are used solely for generating pulses on the out of a counter. The source and gate pins for counters are PFI pins and can be used for applications other than the source or gate of a counter. For example, pin 3 on the 68-pin connector can be used as PFI9, the gate of counter 0, or both. The ability to use one pin for multiple applications offers a great deal of flexibility. For example, you could wire an external TTL signal into pin 3 and use it to trigger an analog input operation and be a gate for a counter operation.

Counter Terminology

The following terms are important to understand when using counters.

- **Terminal Count**—The last count before a counter reaches 0. For example, when a counter that increments the count reaches its maximum count, it has reached the terminal count. The next increment of the count forces the counter to roll over and start counting at 0.
- **Resolution**—How high the counter can count before reaching the terminal count, specified in bits. The following formula calculates the maximum count based on the resolution:

$$\text{max count} = 2^{(\text{resolution})} - 1$$

Common counter resolutions are 16-, 24-, or 32-bits.

- **Timebase**—A signal of known frequency that is provided by the DAQ device. Typical frequencies for timebases range from 100 Hz to 80 MHz. The timebase can be routed internally to the source of a counter to provide a signal of known frequency.

Counter Chips

Depending on the DAQ device, you could be using the NI-STC2, NI-STC3 or NI-TIO counter chip.

NI-TIO

The NI-TIO is a counter chip offered on NI devices. It is a 32-bit counter. The NI-TIO increments and decrements the count, supports encoders and a hardware trigger signal, has digital filters to remove glitches, changes the frequency of a pulse train instantly, and offers 100 kHz, 20 MHz, and 80 MHz timebases. The NI-TIO is used on the 660x family of devices. The M Series DAQ cards also support the NI-TIO counter features, such as two-edge separation and quadrature encoding, and can be used for counter synchronization.

NI-STC2

The NI-STC2 chip used on the M Series devices is a custom designed application-specific integrated circuit (ASIC) that controls interboard and intraboard synchronization and timing for multifunction DAQ operations. The NI-STC2 manages all digital signals on the device, including inputs from the ADC, outputs to the DAC, digital I/O lines, and counter/timers. In addition to providing more digital I/O lines, the NI-STC2 also has six DMA channels dedicated to I/O operations. With six DMA channels, all six operations on the device (analog input, analog output, digital input, digital output, counter/timer 0, and counter/timer 1) can execute simultaneously on their own dedicated DMA channel. This feature dramatically improves execution speed and data throughput when multiple

operations are performed simultaneously. The NI-STC2 incorporates two 80 MHz, 32-bit counter/timers with built-in encoder support.

NI-STC3

The NI-STC3 chip used on the X Series devices is a custom designed ASIC that controls interboard and intraboard synchronization and timing for multifunction DAQ operations. The NI-STC3 manages all digital signals on the device, including inputs from the ADC, outputs to the DAC, digital I/O lines, and counter/timers. NI-STC3 includes eight DMA channels to stream data directly between the device and PC memory without CPU interaction or additional programming effort. These eight channels provide parallel data streams for the analog I/O, digital I/O, and all four counter/timers. With an expanded 127 sample FIFO for each of the four 32-bit counters, the NI-STC3 can perform buffered counter operations, such as event counting or generating a PWM output, at faster rates than prior devices. The NI-STC3 incorporates four 100 MHz, 32-bit counter/timers with built-in encoder support. Also, operations that previously required two counters on the NI-STC2 and NI-TIO chips, can now be done with a single counter channel on the NI-STC3 chip. NI-STC3 technology offers completely independent sample clocks and triggers for each different group of I/O on a multifunction device.

Counter I/O

Like analog input, analog output, and digital I/O, counter operations use the DAQmx Read VI. For counter operations, select a counter instance of the DAQmx Read VI. The DAQmx Write VI is not used with counters. The DAQmx Create Virtual Channel VI, DAQmx Timing VI, and DAQmx Triggering VI are also used to configure counter measurements or generations.

DAQmx Create Virtual Channel VI

To programmatically create a counter input or output channel, select the Counter Input or Counter Output instance of the DAQmx Create Virtual Channel VI.

A Counter Input channel allows you to measure either frequency, period, count edges, pulse width, or semi period.

The configuration options for a Counter Output channel allow you to generate a pulse in terms of frequency, time, or ticks.

DAQmx Read VI

To read a sample or samples from a counter task, select a counter instance from the DAQmx Read VI pull-down menu. For counters, you can read only a single channel at a time, so the single or multiple channel selection window is no longer available.

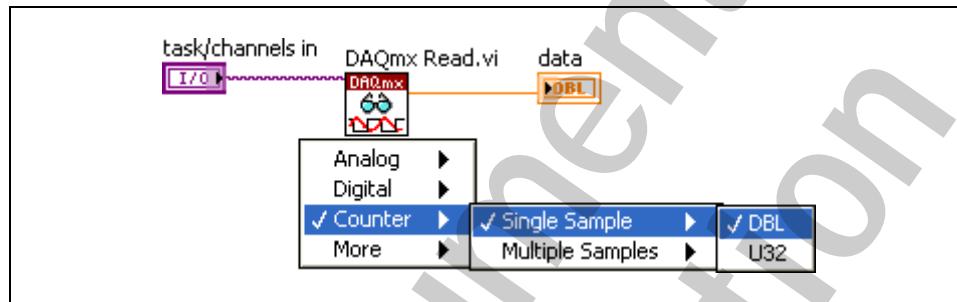


Figure 6-2. DAQmx Read VI—Counter Instance

Select to read either a single sample or multiple samples at once. When reading single samples, select to return the data as either a double-precision, floating-point numeric or unsigned 32-bit integer (U32). Multiple samples return as a 1D array of double-precision, floating-point numerics or a 1D array of unsigned 32-bit integers.

DAQmx Timing VI

For counter operations, select the Sample Clock or Implicit instances of the DAQmx Timing VI. The Sample Clock instance allows you to configure the actual timing rates. The Implicit instance sets only the number of samples to acquire or generate without specifying timing. You can use the Implicit instance of the DAQmx Timing VI later in this lesson when generating pulse trains.

DAQmx Trigger VI

Use the DAQmx Trigger VI to configure triggering for the task. The instances of this polymorphic VI correspond to the trigger and trigger type to configure. Configure counter triggers settings the same way you configure analog input and analog output triggers. In addition, use the DAQmx Trigger Property Node to configure settings for a pause trigger. Refer to Lesson 4, *Analog Output*, of this manual for more information about configuring triggers.

B. Edge Counting

Edge counting is the most basic counter operation. In edge counting, the focus is on measuring the source signal. This section describes simple edge counting and time measurement.

Simple Edge Counting

Simple edge counting fits into the basic definition of a counter. The active edges of the source signal increment the value of the count register. An active edge can be software-selected to be a falling or rising edge. The gate and out are not used for simple edge counting.

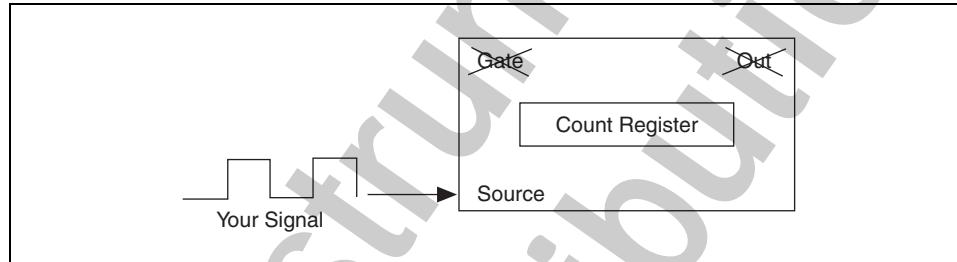


Figure 6-3. Simple Edge Counting

Time Measurement

Time measurement is a variation on simple edge counting. When you perform simple edge counting, the source is the unknown. You use the counter to help you measure the source. When you perform time measurement, the source is a timebase of known frequency. You can use your knowledge of the timebase frequency to help you measure elapsed time. Time measurement uses a timebase for the source instead of using the signal you are trying to measure.

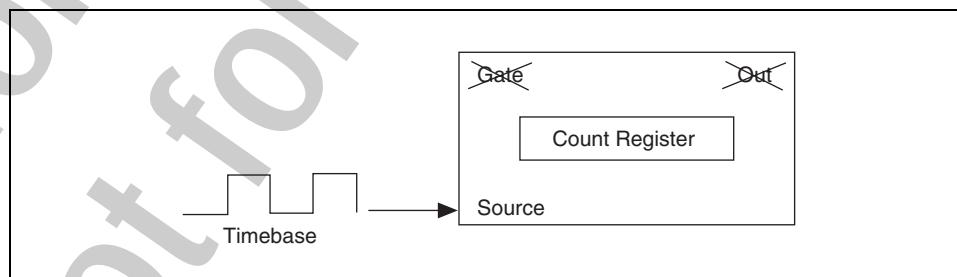


Figure 6-4. Simple Edge Counting—Time Measurement

The following formula calculates the elapsed time:

$$\text{elapsed time} = (\text{count register value}) \times (\text{timebase period})$$

where timebase period = 1/timebase frequency.

The only difference between time measurement and simple edge counting is the signal that is used for the source.

Active Edge

When a counter is configured for simple edge counting or time measurement, the count increments when an active edge is received on the source. You can use LabVIEW to specify if the active edge is rising or falling.

In Figure 6-5, the rising edge was selected as the active edge. The count increments by one every time a rising edge is reached.

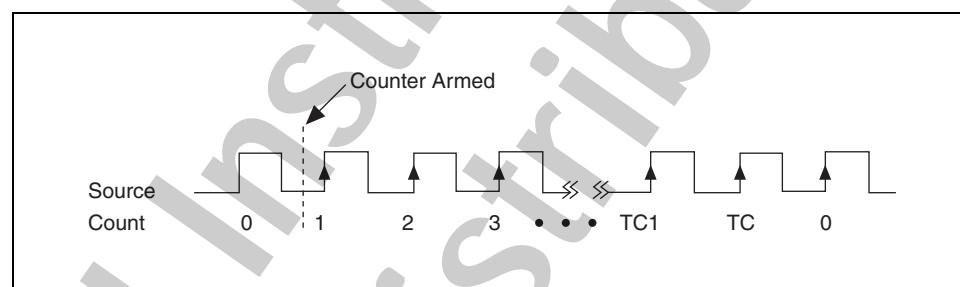


Figure 6-5. Rising Edge Counting

Notice that the count cannot increment until the counter has been armed (started). A counter has a fixed number it can count to as determined by the resolution of the counter. For example, a 24-bit counter can count to:

$$2(\text{Counter resolution}) - 1 = 2^{(24)} - 1 = 16,777,215$$

When a 24-bit counter reaches the value of 16,777,215, it has reached the terminal count. The next active edge forces the counter to roll over and start at 0.

Advanced Edge Counting

In addition to performing simple edge counting, NI-DAQmx can be easily configured to perform more advanced edge counting methods. These methods include pause trigger (gated) counting, and continuous and finite buffered counting.

Pause Trigger (Gated) Counting

In pause triggering, also known as gated triggering, an additional TTL signal enables/disables the count register. The counter value increases when the gate level is either high or low, depending on the configuration settings you choose with the DAQmx Trigger property node.

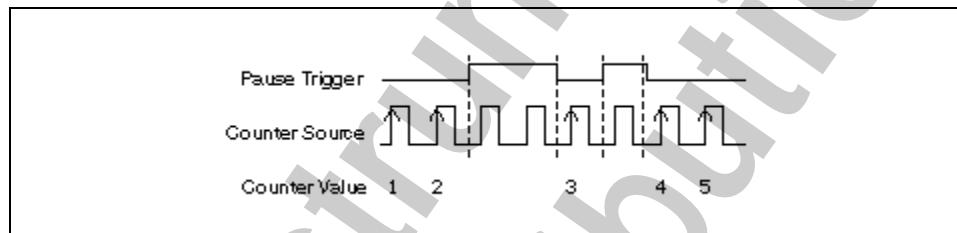


Figure 6-6. Pause Trigger Counting

Continuous Buffered Edge Counting

In continuous buffered edge counting, an additional TTL signal latches the current count register value into a buffer. Thus, the value in the buffer is only updated on the signal's active edge. Figure 6-7 demonstrates this transfer of the count register into the buffer.

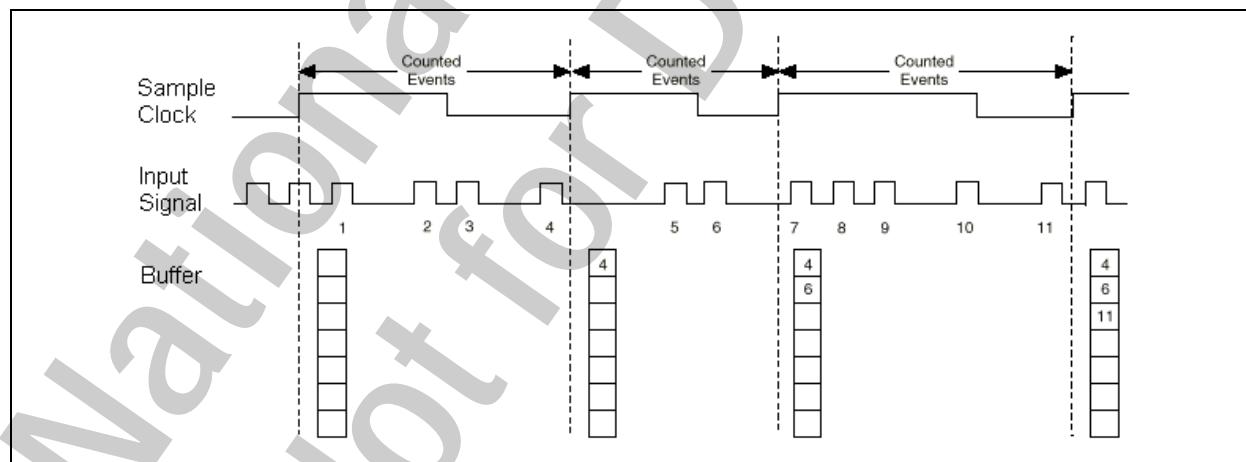


Figure 6-7. Continuous Buffered Edge Counting

With buffered edge counting, the device latches the number of edges counted onto each active edge of the sample clock and stores the number in the buffer. There is no built-in clock for buffered edge counting, so you must supply an external sample clock.

Finite Buffered Edge Counting

Finite buffered edge counting follows the same method for data transfer as continuous buffered edge counting, except that only a finite number of counts are acquired. Use the DAQmx Timing VI to set the number of samples to acquire.

C. Pulse Generation

A counter not only measures TTL signals, but it also generates TTL signals. Using a counter to generate a TTL signal is known as pulse generation.

The output signal shown in Figure 6-8 is generated on the out component of the counter. The generated signal can be a single pulse or a continuous set of pulses known as a pulse train. The counter uses a timebase as the source to help generate the pulse. For now, you only need to understand that a timebase helps generate the pulse. Knowing how the timebase generates a pulse is not necessary to program the counter for pulse generation and is beyond the scope of this course. You do not have to physically connect the timebase to counter source because NI-DAQmx already handles this for you.

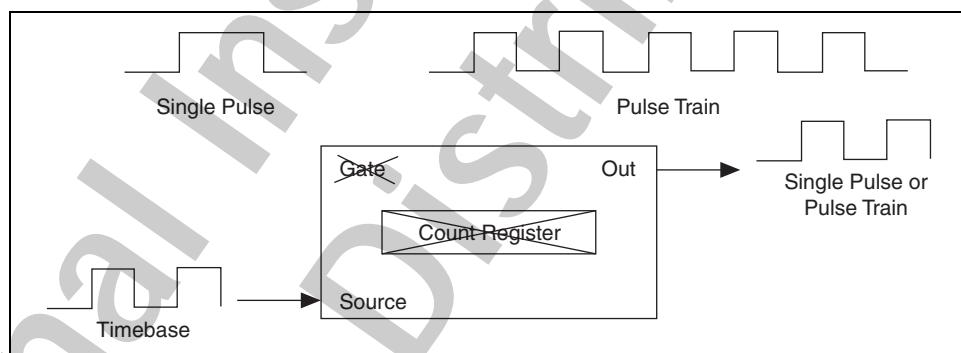


Figure 6-8. Pulse Generation

Pulse Characteristics

To generate a pulse, you must understand certain characteristics of a pulse.

A pulse is a rapid change in the amplitude of a signal from its idle value to an active value for a short period of time. Pulses can have high or low idle states. A pulse with a low idle state starts at the low value (typically zero), pulses high, and returns to low. A pulse with a high idle state starts high, pulses to low, and returns to high.

A pulse train is more than one pulse. You can use a pulse or pulse train as a clock signal, a gate, or a trigger for a measurement or a pulse generation. You can use a single pulse of known duration to determine an unknown signal frequency or to trigger an analog acquisition. You can use a pulse train of known frequency to determine an unknown pulse width.

Each pulse or pulse train consists of three parts:

- **High Time**—The amount of time the pulse is at a high level.
- **Low Time**—The amount of time the pulse is at a low level.
- **Initial Delay**—The amount of time the output remains at the idle state before generating the pulse. The idle state always replaces high time or low time for the first pulse of a generation, depending on the idle state.

The pairing of high time and low time pair is a pulse specification.

Figure 6-9 shows the parts of a pulse.

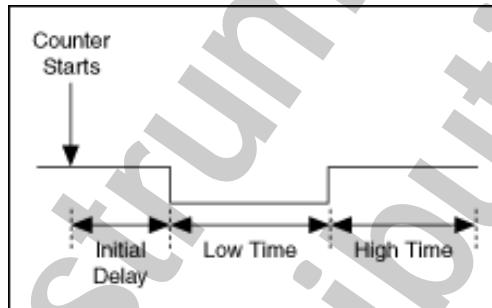


Figure 6-9. Parts of a Pulse

Figure 6-10 shows the parts of a pulse train.

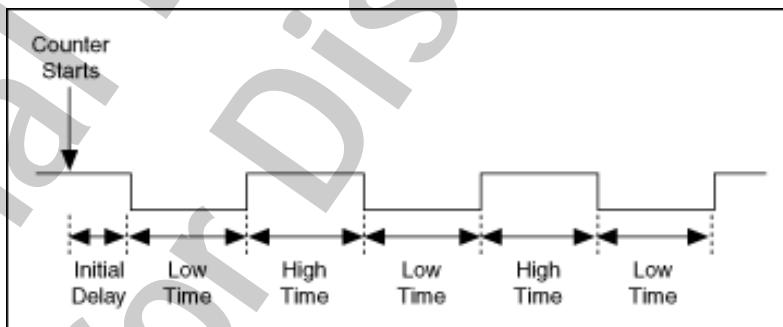


Figure 6-10. Parts of a Pulse Train

Before you generate a pulse, you need to determine if you want to output the pulse or pulse train in terms of frequency, time, or number of ticks of the counter timebase. For frequency, you need to determine the duty cycle. For time, you specify the high time and the low time. Use the number of ticks if you are using a counter timebase with an unknown rate. When you configure a pulse generation, the output appears at the counter output terminal.

The period of a pulse is the time taken by the pulse to complete one cycle, so by adding the high time to the low time, you can obtain the pulse period. After determining the period of the pulse, take the inverse to obtain the frequency of the pulse.

The high time and the low time of a pulse are not always equal, so you need a property of a pulse that helps you determine if the high time is larger than the low time or vice versa. The parameter you use is called the duty cycle. The duty cycle gives you a number between 0 and 1. This number is often converted into a percentage. A pulse where the high time is equal to the low time has a duty cycle of 0.5, or 50%. A duty cycle greater than 50% means the high time is larger than the low time, and a duty cycle less than 50% means the low time is larger than the high time.

$$\text{Pulse Period} = \text{High Time} + \text{Low Time}$$

$$\text{Pulse Frequency} = 1 / \text{Pulse Period}$$

$$\text{Duty Cycle} = \text{High Time} / \text{Pulse Period}$$

Generating Single Pulses, Finite Pulse Trains, and Continuous Pulse Trains

When generating pulses, you can generate either a single pulse, a finite pulse train, or a continuous pulse train. By default, single pulses are generated unless you use the DAQmx Timing VI with the Implicit or Sample Clock timing types.

In Implicit mode, the **Samples Per Channel** input to the DAQmx Timing VI determines the number of pulses to generate for finite pulse trains. In Figure 6-11, three pulses are generated:

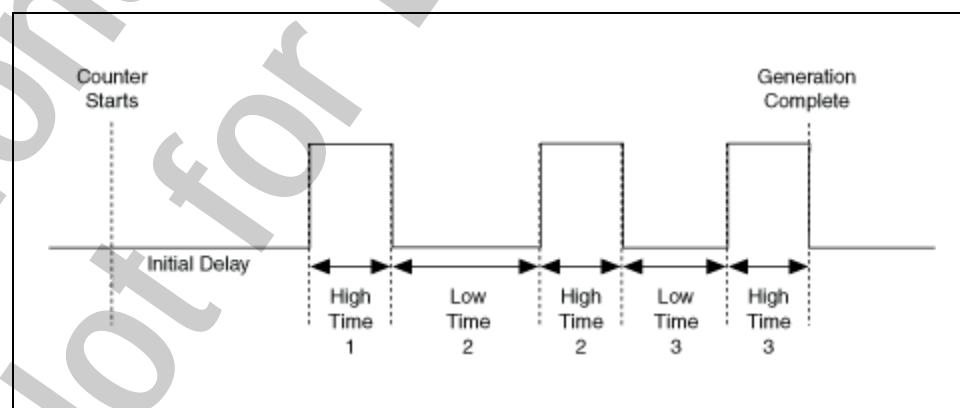


Figure 6-11. Implicit Mode for Finite Pulse Train Generation

In Sample Clock mode, which uses a software buffer, the **Samples per Channel** input determines the number of distinct pulse specification (High Time/Low Time) transitions to generate. In Figure 6-12, **Samples per Channel** is set to three. Notice that the counter increments after the sample clock pulses and the counter signal transitions from high to low. The device generates the high and low values specified in the DAQmx Create Virtual Channel VI until the first sample clock arrives.

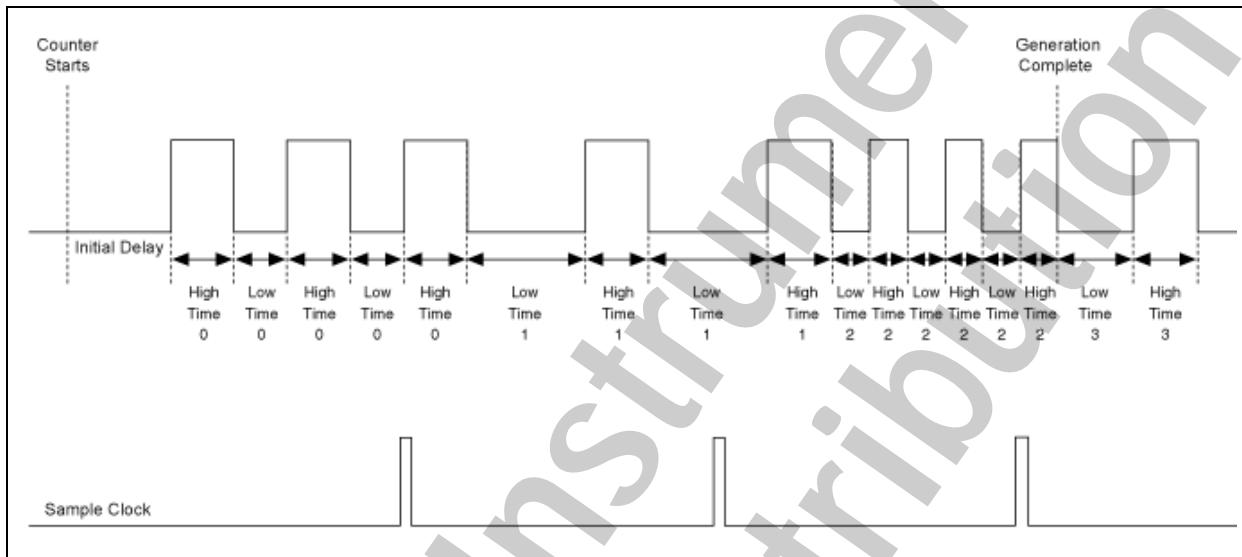


Figure 6-12. Sample Clock Mode for Pulse Train Generation

As illustrated previously, to output pulse trains in which pulse specifications are hardware timed and change deterministically, you must use a software buffer, if supported by your device.

You can specify the size of the buffer by calling the DAQmx Configure Output Buffer VI, by specifying the buffer size property in the buffer property node, or by writing a number of pulse specifications using the DAQmx Write Counter MultiPoint VI before starting the task. This is ideal for applications that require pulse-width modulation, such as proportional integral derivative (PID) loop control applications. The high and low times provided in the DAQmx Create Virtual Channel VI are ignored in this case.

If you do not use a software buffer, all pulses generated will be the same, unless you update the high time and low time while the application is running. This will cause the pulse specifications to be software timed and change on-demand.

You can use the same properties that create the channel to update the pulse specifications of the pulse train generation. Because you need two properties to specify the pulse specifications of the pulse train, the specifications only update when you set one of the two.

For example, if you specify the pulse generation in terms of frequency, the frequency and duty cycle control the specifications of the generation. However, the pulse specifications only update when you set the frequency property. The same is true when you specify pulse generation in terms of time or ticks; the low time and low ticks control when the pulse specifications update. When updating the pulse specifications of the pulse generation, a complete period of the current specification generates before the new pulse specification takes effect. Updating the pulse specifications while running is not supported on buffered pulse train generation.

In some devices, such as M Series and S Series devices, generating finite pulse trains requires the use of paired counters. In devices that require paired counters on a finite pulse train generation, the first counter (for example, Counter 0) generates a pulse of desired width. The second counter (Counter 1) generates the pulse train, which is gated by the pulse of the first counter. The routing is done internally. Figure 6-13 shows a two counter finite pulse train timing diagram.

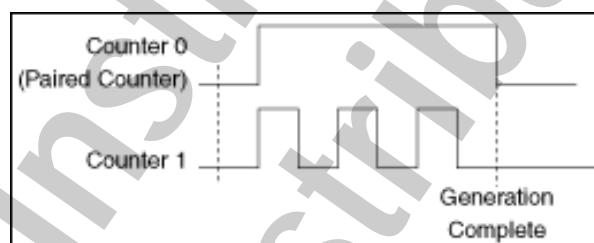


Figure 6-13. Finite Continuous Pulse Train Using Paired Counters

STC3-based devices, such as X Series devices, do not require paired counters.

D. Pulse Measurement

When you measure a pulse, the signal you are measuring is used as the gate and the source is a timebase of known frequency, as shown in Figure 6-14. You can use the known frequency of the timebase and the value of the count register to determine characteristics of the gate pulse, such as pulse period and pulse width.

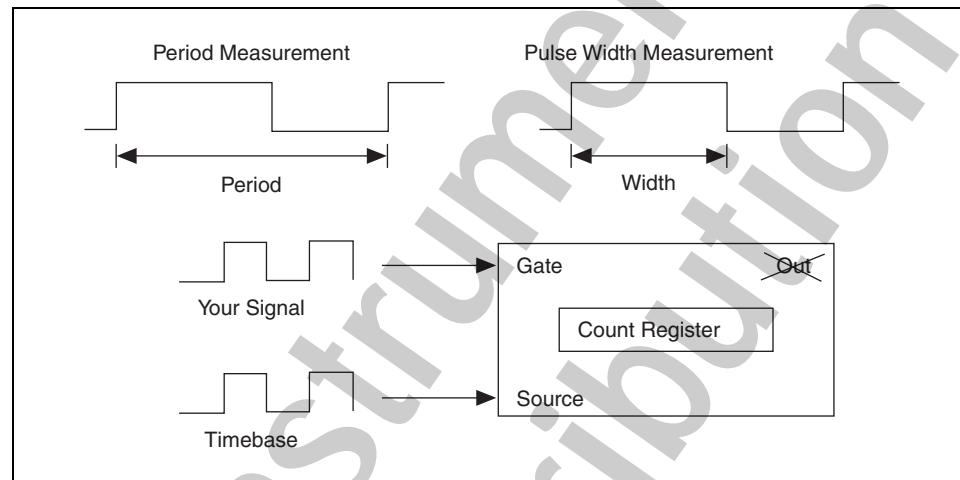


Figure 6-14. Pulse Measurement

Period Measurement

Period measurement is one type of pulse measurement. With period measurement, you still count the active edges on the source signal. However, unlike simple edge counting, you only increment the count during the period of the gate signal. Figure 6-15 shows period measurement that is started and stopped by a rising edge on the gate signal.

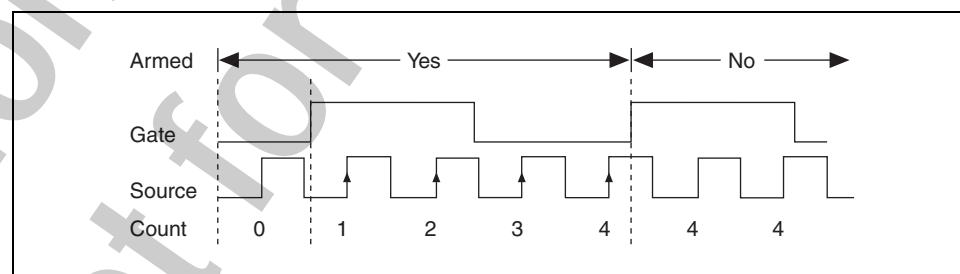


Figure 6-15. Period Measurement

You also can start and stop counting between falling edges. The count reflects the number of rising edges on the source between the two rising edges on the gate. Therefore, to perform period measurement, you need a signal with two rising edges or two falling edges. A single pulse has only one rising edge and one falling edge, so you would not be able to measure its period.

In the previous example, one period of the gate signal has four counts. Remember that the source is a timebase of known frequency. Assume that you are using a 100 kHz timebase. The formula for calculating the period of the gate is as follows:

$$\text{pulse period} = \text{count} \times (1/\text{source frequency})$$

For the previous example, with the 100 kHz timebase, the formula yields:

$$\text{pulse period} = 4 \times (1/100,000) = 0.04 \text{ milliseconds}$$

Semi Period Measurement

Semi period measurement is very similar to period measurement, but you are only measuring the time between consecutive edges.

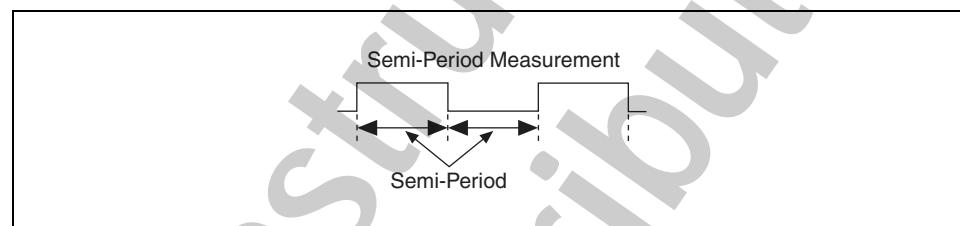


Figure 6-16. Semi-Period Measurement

A buffered semi-period measurement is equivalent to a buffered pulse width measurement for both the low and high segments of the gate. Like the other buffered measurements, this task allows you to continuously measure consecutive pulses without resetting the counter. The first edge of a user-defined polarity initiates the measurement, and each subsequent edge (both falling and rising) latches the current count register value into memory. This type of measurement allows one to measure the period and duty cycle of a pulse train on the gate because we have a time measurement between all edges.

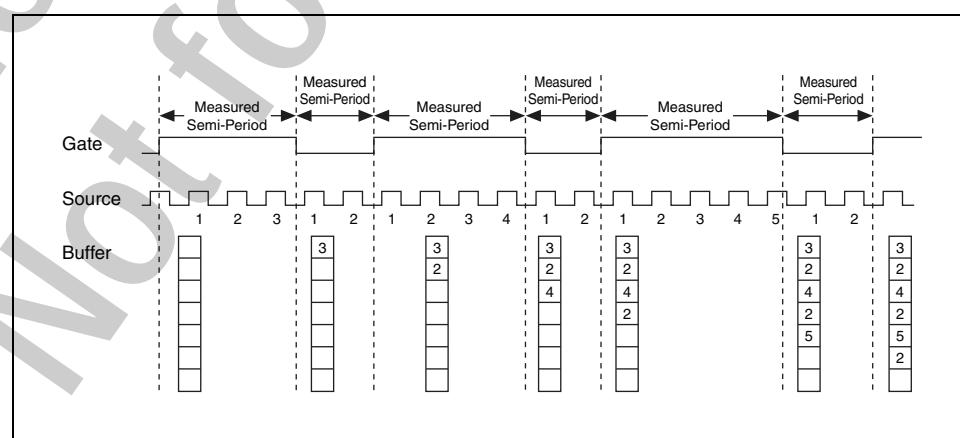


Figure 6-17. Buffered Semi-Period Measurement

Pulse Width Measurement

Pulse width measurement is very similar to period measurement. The difference is where you stop counting. With period measurement, you started and stopped counting with two rising edges on the gate signal. With pulse width measurement, you count only during the pulse width, so you start counting on one edge and end on the opposite edge. The count value increments only between two opposite edges, as shown in Figure 6-18.

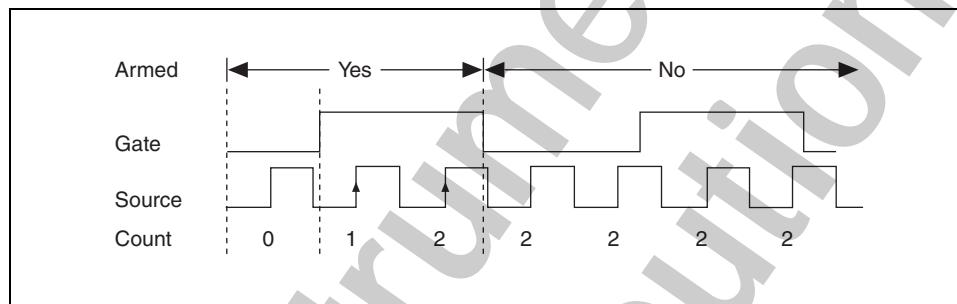


Figure 6-18. Pulse Width Measurement

The formula for calculating the pulse width is the same as the formula for the period:

$$\text{pulse width} = \text{count} \times (1/\text{source frequency})$$

For the previous example, a 100 kHz source yields:

$$\text{pulse width} = 2 \times (1/100,000) = 0.02 \text{ milliseconds}$$

0.02 milliseconds is half the value that you received for period measurement, so you must have a gate signal with a duty cycle of 50%.

Pulse width measurements can return the width of the pulse in units of seconds, ticks, or from a custom scale.

Pulse Measurements

Some devices support measuring the frequency, time, and ticks of individual pulses. These devices can return the specifications of the measured pulse as tuple of frequency/duty cycle, high/low time, or high/low ticks.

E. Frequency Measurement

This section describes three ways to measure the frequency of a TTL pulse train using one or more counters. The frequency of a waveform is simply the inverse of its period at any given time. Thus, the easiest type of frequency measurement is merely the inverse of a period measurement. The two other types of frequency measurements become necessary because the first type of frequency measurement becomes increasingly inaccurate as the gate frequency approaches the timebase frequency of the counter.

Frequency Measurements For STC2-Based DAQ Devices

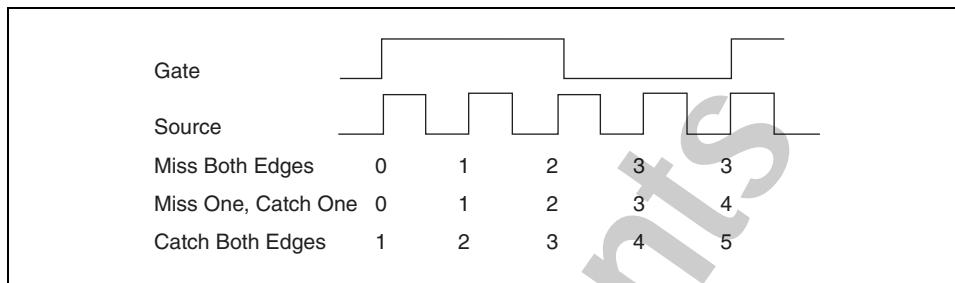
This section describes three ways to measure the frequency of a TTL pulse train using one or more counters when using a STC2-based DAQ device. The frequency of a waveform is simply the inverse of its period at any given time. Thus, the easiest type of frequency measurement is merely the inverse of a period measurement. The two other types of frequency measurements become necessary because the first type of frequency measurement becomes increasingly inaccurate as the gate frequency approaches the timebase frequency of the counter.

Low Frequency with 1 Counter Method

The first frequency measurement is really a period measurement. When the period is acquired, take the inverse to compute frequency. The advantage of this method is that it uses only one counter and is easy to perform. However, this method relies on a relatively slow gate signal because the accuracy of a period measurement depends on the number of source edges that occur within one gate period. In NI-DAQmx, this method is called **Low Frequency with 1 Counter**.

Quantization Error

As the gate frequency approaches the source frequency, period measurements suffer from quantization error. For example, consider a period measurement that uses a 20 MHz internal timebase on the source. Now, suppose the gate signal is roughly 5 MHz, or 1/4 of the source frequency. Figure 6-19 shows three possible scenarios in which the first and last source edges may or may not be included in the period measurement.

**Figure 6-19.** Quantization Error

In the first scenario, the measurement misses the first and last source edges, counting a total of only three edges. The second scenario catches the first four edges and misses the last edge. The third scenario shows all five source edges being counted. The second scenario is obviously more accurate, but because the source edges are so closely synchronized with the gate edges, the counter is equally likely to pick any one of the three scenarios shown.

Pulse measurements always have an error of ± 1 source cycle, which is generally negligible when one source cycle accounts for only 1% (or less) of the pulse measurement. However, one source cycle accounts for between 33% to 20% of the measurement. This is known as quantization error and can be avoided by choosing a different measurement scheme.

High Frequency with 2 Counters Method

The second method of measuring frequency uses two counters—one to generate a pulse train with a known frequency and one to perform a period measurement. Counter 1 performs a period measurement, using the external signal as a source instead of the internal timebase. The gate signal of counter 1 comes from the output of counter 0, which is generating a pulse train. Because you know the frequency of the output of counter 0, you know exactly the length of the gate cycle on counter 1. Based on the number of source edges that arrive on the source of counter 1, you can deduce frequency, dividing the period measurement of counter 1 by the gate period. For example, if counter 0 outputs a pulse train of 10 Hz, the gate period is 0.1 s. If during that time, you count 100 source edges, you know the source frequency on counter 1 is $(100 \pm 1)/0.1$ or 1000 ± 10 Hz. In NI-DAQmx, this method is called **High Frequency with 2 Counters**.

Use this high-frequency measurement method if you measure a digital frequency or period of a signal with a high frequency component. To perform measurements using this method in NI-DAQmx, a paired counter generates a pulse train with a period specified using the measurement time input of the Counter Input»Frequency instance of the DAQmx Create Virtual Channel VI. The measurement time is generally much larger than the period of the input signal being measured to reduce quantization error. However, the measurement time must be small enough to keep the counter

from rolling over. The measurement counter counts the number of periods of the input signal that occur during the measurement time, averages the results, and returns the averaged value in the DAQmx Read VI. The value returned is calculated as follows:

$$\text{Period (in seconds)} = \text{Measurement Time} / \text{Number of Periods Counted}$$

$$\text{Frequency (in Hz)} = \text{Number of Periods Counted} / \text{Measurement Time}$$

To determine if you should use the high-frequency measurement method, refer to the quantization error tables in the *NI-DAQmx Help*. If the quantization error listed for the one-counter method is too high, use the high-frequency measurement method instead.

Large Range with 2 Counters Method

The third method of measuring frequency also uses two counters, except the counter that generates the pulse train (counter 0) uses the external signal as the source, and the counter that performs the period measurement (counter 1) uses the internal timebase as its source. Like the Low Frequency with 1 Counter method, this method uses the pulse train from the output of counter 0 to gate the period measurement on counter 1.

The advantage of the divide down method is that it introduces less error than period measurements or averaging.

If you measure the digital frequency or the period of a counter signal, you can use this two-counter method to measure signals with large ranges. This method is useful when you have a widely varying signal to measure and would like increased accuracy throughout the entire range. Refer to the quantization error tables in the *NI-DAQmx Help* for more information on increasing measurement accuracy with the large-range measurement method. You can also use this method to measure signal frequencies that are faster than your counter timebase rate as long as the input signal does not exceed the maximum input frequency supported by the counter. In NI-DAQmx, this method is called **Large Range with 2 Counters**.

To perform measurements using this method in NI-DAQmx, a paired counter is used to divide the input signal by a value specified using the Divisor input of the Counter Input»Frequency of the DAQmx Create Virtual Channel VI. However, you need to be careful the Divisor you choose does not cause the counter to roll over. This divisor has the effect of shifting the measurable frequency range upward.

The Divisor scales the measured period and returns data according to the following equations:

$$\text{Period} = \text{Measured Period} / \text{Divisor}$$

$$\text{Frequency} = \text{Divisor} \times \text{Measured Period}$$

For example, if you use a 24-bit counter and the Counter Timebase Rate is 100 kHz, the measurable frequency range is approximately 0.006 Hz to 50 kHz because:

$$\text{Frequency} = (\text{Counter Timebase} / \text{Count}) \times \text{Divisor}$$

$$\text{Frequency} = (100 \text{ kHz} / 2^{24}) \times 1 = .006 \text{ Hz} \text{ and } (100 \text{ kHz} / 2) \times 1 = 50 \text{ kHz}$$

However, with a divisor of 4, the measurement frequency range is 0.024 Hz to 200 kHz because:

$$\text{Frequency} = (100 \text{ kHz} / 2^{24}) \times 4 = .024 \text{ Hz} \text{ and } (100 \text{ kHz} / 2) \times 4 = 200 \text{ kHz}$$

To determine if you should use the large-range measurement method, refer to the quantization error tables in the *NI-DAQmx Help*. If the quantization error listed for the one-counter method in that section is too high, use the large-range measurement method instead.

Frequency Measurements For STC3-Based DAQ Devices

If you are using a STC3-based DAQ device, you can use Sample Clock timing and you can choose to enable or disable averaging.

STC3-based devices have a sample clock for counters, which enables you to use sample clock timing for counter measurements. Similar to the sample clock timing in an Analog Input acquisition, the sample clock determines when a measurement is stored into the on board memory. This way the data rate and the read latency are decoupled from the frequency of the signal being measured.

Sample Clock Timing (Averaging Disabled) Method

This method is good for low frequencies and uses 1 counter channel.

Note that many or very few periods may happen between sample clocks, and only the last period measured before each sample clock edge will be stored into memory when using the sample clock timing (averaging disabled) method for frequency measurements.

To implement this method, use the Counter Input»Frequency instance of the DAQmx Create Virtual Channel VI, the Sample Clock instance of the DAQmx Timing VI, and set the EnableAveraging property to False using a DAQmx Channel Property Node.

Sample Clock Timing (Averaging Enabled) Method

This method is good when measuring high frequencies and varying range of frequencies. This method uses 1 counter channel.

When using this method, each sample clock pulse will store into memory both the number of completed periods between sample clock pulses and the period measured during those completed periods. DAQmx will calculate the average frequency using these measurements.

To implement this method, use the Counter Input»Frequency instance of the DAQmx Create Virtual Channel VI, the Sample Clock instance of the DAQmx Timing VI, and set the EnableAveraging property to True using a DAQmx Channel Property Node.

F. Position Measurement

A quadrature encoder is a popular sensor used in counter applications. A quadrature encoder allows you to measure position, and it converts rotary motion into a measurable signal. DAQ devices using the NI-TIO, NI-STC2, or NI-STC3 chips directly support quadrature encoders.

How Encoders Work

An encoder is a sensor that allows you to measure position or distance. To understand how an encoder works, examine the quadrature encoder shown in Figure 6-20.

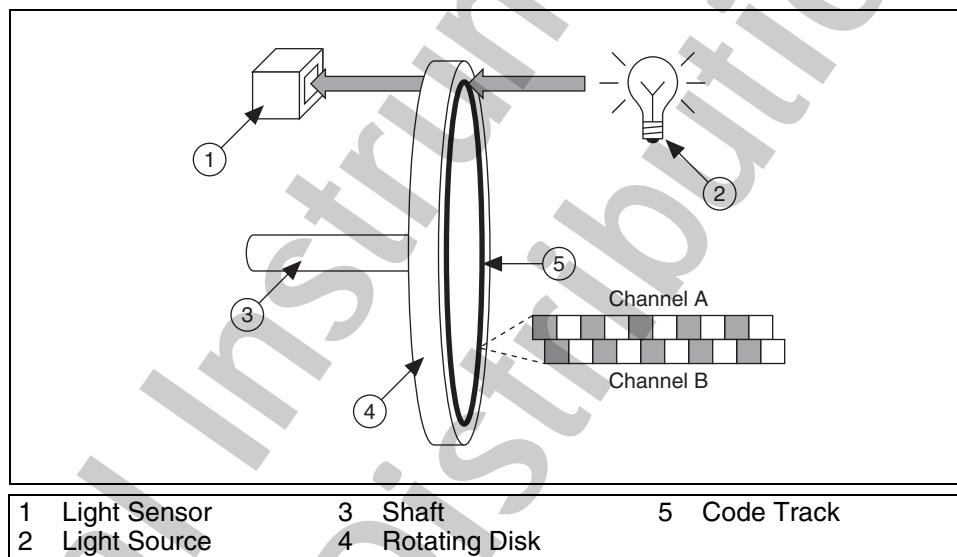


Figure 6-20. Quadrature Encoder

A quadrature encoder helps you convert rotary motion into a measurable signal. The rotary motion you are interested in is the spinning of the shaft. You could be interested in the direction the shaft is rotating, the speed the shaft is rotating, or both. A disk is attached to the shaft so that it rotates in the same direction and at the same speed as the shaft. The rotating disk is placed between a light source and a light sensor. The disk has a section of alternating opaque and transparent sections called the code track. An opaque section blocks the light from the source to the sensor, and a transparent section allows the light to pass from the source to the sensor. The code track consists of two rings of alternating opaque and transparent sections. Each ring produces a pulse train. The two rings are offset so that, depending on the direction the disk rotates, one pulse train leads the other. The number of opaque and transparent sections determines how many pulses are produced per revolution. This is an important specification to know if you are keeping track of how many revolutions the shaft has completed.

Quadrature Encoder

Most encoders produce a TTL-compatible signal that can be used with a counter. As you saw earlier, a quadrature encoder produces two pulse trains. The two pulse trains are referred to as Channel A and Channel B. Channel A and Channel B are always out of phase by 90° , as shown in Figure 6-21. The leading channel is determined by the direction of rotation. If the encoder is rotated in a clockwise direction, Channel A leads Channel B. If the encoder is rotated in a counter-clockwise direction, Channel B leads Channel A.

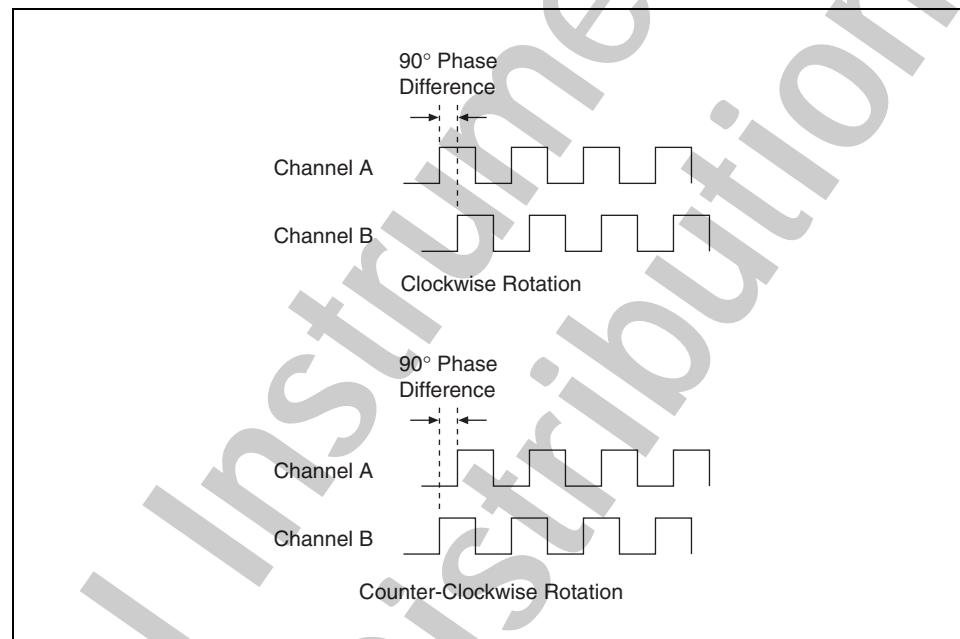


Figure 6-21. Quadrature Encoder Signals



Note Some encoders might not produce a signal that is suitable for a counter. For example, some encoders produce a differential signal, and none of the NI counter chips support differential inputs. If the signal produced by the encoder is not compatible with the counter chip you are using, you need to use signal conditioning before you send the signal to the counter.

Decoding Quadrature Encoders

Counters on most DAQ devices support three types of decoding for quadrature encoders: X1, X2, and X4. With X1 decoding, when signal A leads signal B, the counter increments on the rising edge of signal A. When signal B leads signal A, the counter decrements on the falling edge of signal A.

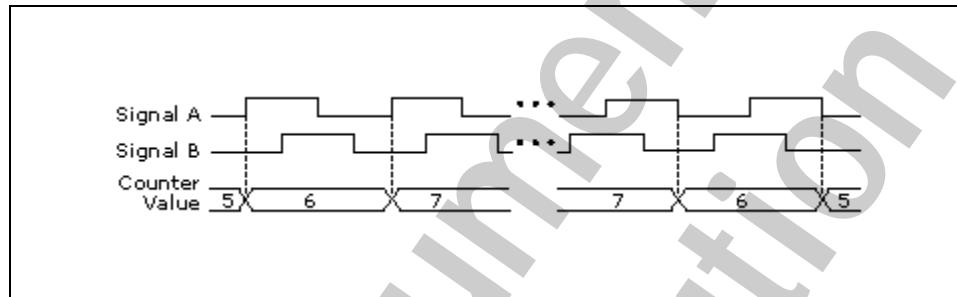


Figure 6-22. X1 Decoding

With X2 decoding, the same behavior holds as with X1, except the counter increments and decrements on both rising and falling edges of signal A.

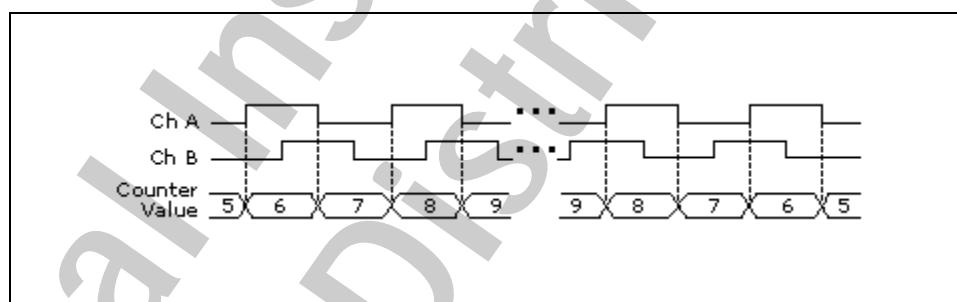


Figure 6-23. X2 Decoding

Similarly, with X4 decoding, the counter increments and decrements on both rising and falling edges of both signal A and signal B. X4 decoding is more sensitive to position, but is also more likely to provide an incorrect measurement if there is vibration in the encoder.

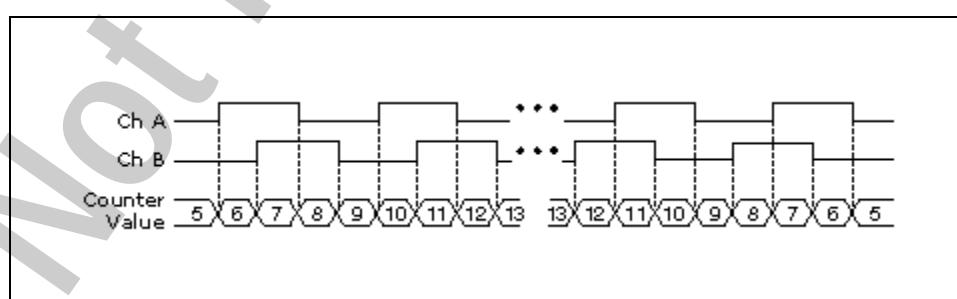


Figure 6-24. X4 Decoding

To select the decoding type for the quadrature encoder with NI-DAQmx in LabVIEW, use the decoding type input of the Counter Input»Position»Angular Encoder or Counter Input»Position»Linear Encoder instance of the DAQmx Create Virtual Channel VI.

Z Indexing

Many encoders use a third signal for Z indexing, which produces a pulse at fixed positions that you can use for precise determination of a reference position. For instance, if the Z index is 45° for an angular encoder, the encoder sends a pulse on the Z input terminal every time the encoder is turned to the 45° mark.

The behavior of signal Z differs with designs. You must refer to the documentation for an encoder to obtain the timing of signal Z in relation to the A and B signals. In NI-DAQmx, you can configure Z indexing with the Z Index Phase property.

To implement Z indexing for the quadrature encoder with NI-DAQmx in LabVIEW, set the z index enable input to true on the Counter Input»Position»Angular Encoder or Counter Input»Position»Linear Encoder instance of the DAQmx Create Virtual Channel VI. Use the z index phase input to specify the states at which signal A and signal B must be while signal Z is high for NI-DAQmx to reset the measurement. Use the z index value input to specify in units the value to which to reset the measurement when signal Z is high and signal A and signal B are at the states you specify with z index phase.

Summary

- Counters accept and generate TTL signals.
- The main components of a counter are the source, gate, out, and count register.
- NI devices can have one of three different counter chips:
 - NI-STC 2 (32-bit)—M Series devices
 - NI-TIO (32-bit)—660x devices
- You can use the Easy Counter VIs to perform edge counting, pulse generation, pulse measurement, and frequency measurement.
- A quadrature encoder is a transducer that converts rotary motion into two pulse trains that are out of phase by 90°.

National Instruments
Not for Distribution

National Instruments
Not for Distribution

Self-Review: Quiz

1. Which of the following are components of a counter?
 - a. Source
 - b. Gate
 - c. Multiplexer
 - d. Register
 - e. Output
2. What is the terminal count of a 24-bit counter?
3. What error occurs when the frequency being measured by the Low Frequency with 1 Counter method approaches the timebase of the DAQ device?

National Instruments
Not for Distribution

National Instruments
Not for Distribution

Self-Review: Quiz Answers

1. Which of the following are components of a counter?

- a. Source
- b. Gate
- c. Multiplexer
- d. Register
- e. Output

2. What is the terminal count of a 24-bit counter?

$$2^{24} - 1 = 16,777,216 - 1 = 16,777,215$$

3. What error occurs when the frequency being measured by the Low Frequency with 1 Counter method approaches the timebase of the DAQ device?

Quantization Error

Notes

National Instruments
Not for Distribution

Signal Conditioning

This lesson introduces using signal conditioning with a data acquisition system to accurately measure a wide variety of physical phenomena. This lesson also describes signal conditioning methods for improving signal quality.

Topics

- A. Overview of Signal Conditioning
- B. Signal Conditioning Systems
- C. Signal Conditioning for Voltage Measurements
- D. Temperature Measurements
- E. Strain, Pressure, Load, and Torque Measurements
- F. Sound and Vibration Measurements

National Instruments
Not for Distribution

A. Overview of Signal Conditioning

A typical data acquisition system consists of a physical phenomenon, sensors, signal conditioning, data acquisition hardware, and data acquisition software on a computer.

The signal source is typically a sensor that measures a physical phenomenon. Different sensor manufacturers and sensor types measure a wide variety of signal levels. The signal usually travels through two or more wires varying in length from inches to miles, perhaps traversing through high voltage or other electromagnetic hazards. Many signal quality problems can occur before the signal is acquired into the DAQ software, so the data acquisition system typically requires signal conditioning. Signal conditioning can occur in the sensor, along the path between the sensor and DAQ hardware, and in the DAQ hardware.

Most real-world sensors and transducers generate signals that you must condition before a DAQ device can reliably and accurately acquire the signal. Signal conditioning cleans up the signal using techniques such as amplification, attenuation, filtering, and isolation. Signal conditioning systems can be found integrated into DAQ devices such as CompactDAQ and PXI SC DAQ devices, in separate front-end devices such as Signal Conditioning eXtensions for Instrumentation (SCXI) or Signal Conditioning Components (SCC), and integrated into sensors such as IEPE accelerometers. The goal is to prepare the signal before it is measured by the analog-to-digital converter (ADC) of the DAQ device.

The data acquisition device provides the capability to digitize the conditioned analog signal. The computer can then analyze and present the digitized and conditioned signal.

B. Signal Conditioning Systems

National Instruments offers a wide variety of signal conditioning products that can improve your measurement system accuracy and performance. For more information about NI signal conditioning products, refer to ni.com/signalconditioning.

CompactDAQ

NI CompactDAQ hardware provides the plug-and-play simplicity of USB to sensor and electrical measurements on the benchtop, in the field, and on the production line. It delivers fast and accurate measurements in a small, simple, and affordable system by combining the ease of use and low cost of a data logger with the performance and flexibility of modular instrumentation.

PXI Express

PXI Express Signal Conditioning modules are capable of measuring high-speed measurements and contain built-in signal conditioning, so they can measure sensors and high-voltage signals directly. They ensure measurement accuracy with programmable gain and filter settings. These modules can also use the PXI trigger bus for multimodule synchronization and use the PXI platform for high channel count applications.

SCXI

NI SCXI is a high channel count, front-end signal conditioning and switching platform for use with multifunction DAQ devices. An SCXI system consists of multichannel signal conditioning modules installed in one or more rugged chassis that connect to a multifunction DAQ device. You can choose from a wide selection of analog input, analog output, digital I/O, and switching modules to fit your application needs.

C. Signal Conditioning for Voltage Measurements

In addition to handling specific sensors, signal conditioning devices perform a variety of general-purpose conditioning functions to improve the quality, flexibility, and reliability of a measurement system.

Amplification

Because real-world signals are often very small in magnitude, signal conditioning can improve the accuracy of data. Amplifiers boost the level of the input signal to better match the range of the ADC, increasing the resolution and sensitivity of the measurement. Though many DAQ devices include onboard amplifiers, many sensors, such as thermocouples, require additional amplification.

Many sensors produce voltage output signals of millivolts or even microvolts. Amplifying these low-level signals directly on a DAQ device also amplifies any noise picked up from the signal connections. When the signal is small, even a small amount of noise can drown out the signal itself, leading to erroneous data. A simple method for reducing the signal-to-noise ratio is to amplify the signal as close to the source as possible. This boosts the signal above the noise level before noise in the connections can corrupt the signal and improves the signal-to-noise ratio of the measurement. For example, Figure 7-1 shows a J-type thermocouple outputting a low-level voltage signal that varies by about $50 \mu\text{V}/^\circ\text{C}$.

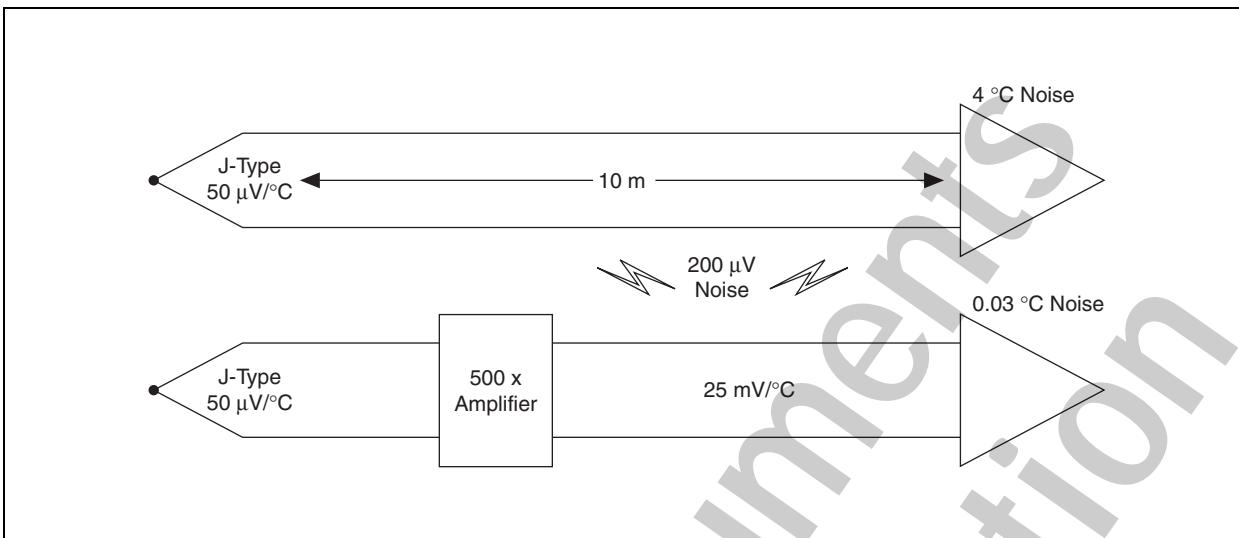


Figure 7-1. J-type Thermocouple Outputting Low-level Voltage Signal

Suppose the thermocouple leads travel 10 m through an electrically noisy environment to the DAQ system. If the noise sources in the environment couple $200 \mu\text{V}$ onto the thermocouple leads, you get a temperature reading with about $4 \text{ }^\circ\text{C}$ of noise. Amplify the signal close to the thermocouple, before noise corrupts the signal to reduce the effect on the final measurement. Amplifying the signal with a gain of 500 near the thermocouple produces a thermocouple signal that varies by about $25 \text{ mV}/^\circ\text{C}$. Because this high-level signal travels the same 10 m, the $200 \mu\text{V}$ of noise coupled onto this signal after the amplification has less effect on the final measurement, adding only $0.03 \text{ }^\circ\text{C}$ of noise.



Tip There are several ways to reduce noise:

- Use shielded cables or a twisted pair of cables
- Minimize wire length to minimize noise, which the lead wires pick up
- Keep signal wires away from AC power cables and monitors to reduce 50 or 60 Hz noise

Signal-to-Noise Ratio

The SNR is a measure of how much noise exists in a signal compared to the signal itself. SNR defines the voltage level of the signal divided by the voltage level of the noise. The larger the SNR, the better it is.

Attenuation

Attenuation, the opposite of amplification, is necessary when voltages to be digitized are beyond the digitizer input range. This form of signal conditioning decreases the input signal amplitude so that the conditioned signal is within ADC range. Attenuation is necessary for measuring high voltages.

Filtering

Signal conditioning systems can include filters to reject unwanted noise within a certain frequency range. Almost all DAQ applications are subject to some degree of 50 or 60 Hz noise picked up from the power lines or machinery. Therefore, most signal conditioning systems include lowpass filters designed specifically to provide maximum rejection of 50 or 60 Hz noise. For example, the SCXI-1125 module includes a lowpass filter with a cutoff bandwidth of 4 Hz so that rejection of 50 or 60 Hz noise is maximized (90 dB).

Filters are generally grouped into one of five classifications—lowpass, highpass, bandpass, bandstop, and all-pass. These classifications refer to the frequency range (the passband) of signals that the filter is intended to pass from the input to the output without attenuation. Because most National Instruments signal conditioning modules use a lowpass filter, this section focuses on lowpass filters.

An ideal lowpass filter does not attenuate any input signal frequency components in the passband, which is defined as all frequencies below the cutoff frequency. An ideal lowpass filter completely attenuates all signal components in the stopband, which includes all frequencies above the cutoff frequency. The ideal lowpass filter also has a phase shift that is linear with respect to frequency. This linear phase property means that signal components of all frequencies are delayed by a constant time, independent of frequency, thereby preserving the overall shape of the signal. Real filters subject input signals to mathematical transfer functions that approximate the characteristics of an ideal filter. Figure 7-2 compares the attenuation of transfer functions of an ideal filter and a real filter.

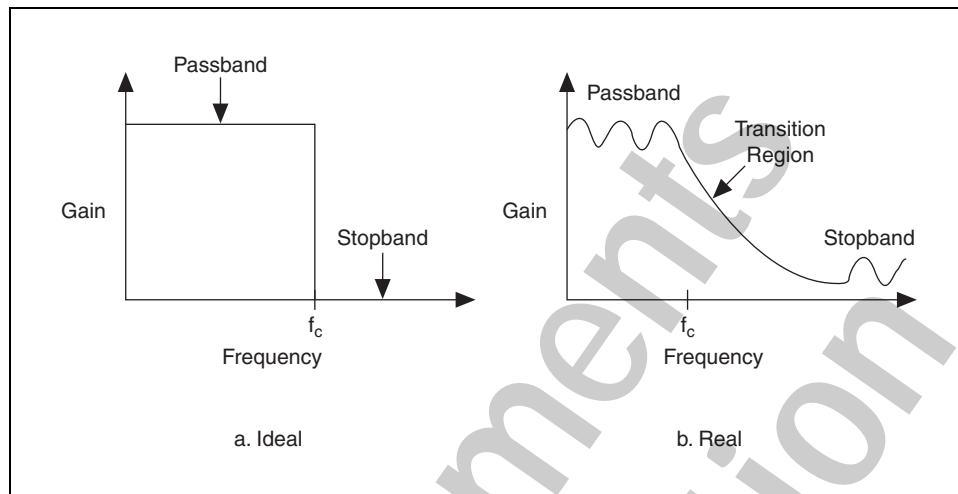
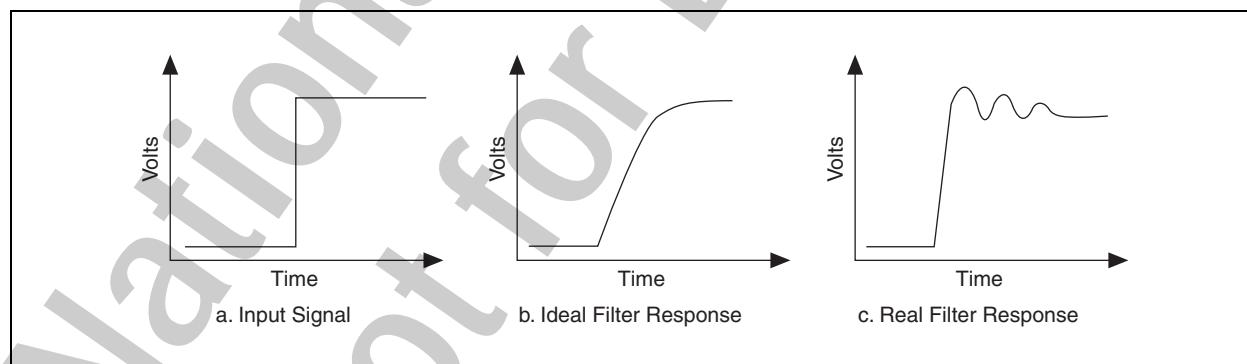
**Figure 7-2.** Transfer Functions of Ideal Filter and Real Filter

Figure 7-2 shows a real filter having a ripple (an uneven variation in attenuation versus frequency) in the passband, a transition region between the passband and the stopband, and a stopband with finite attenuation and ripple.

In addition, real filters have some nonlinearity in their phase response, which causes signal components at higher frequencies to be delayed by longer times than signal components at lower frequencies, resulting in an overall shape distortion of the signal. This can be observed when a square wave or step input is sent through a lowpass filter. An ideal filter smooths the edges of the input signal. A real filter causes some ringing in the total signal because the higher-frequency components of the signal are delayed. Figure 7-3 shows examples of these responses to a step input.

**Figure 7-3.** Filter Responses

Anti-aliasing Filters

Another common use of filters is to prevent signal aliasing—a phenomenon that arises when a signal is undersampled (sampled too slowly). The Nyquist theorem states that when you sample an analog signal, any signal components at frequencies greater than one-half the sampling frequency appear in the sampled data as a lower frequency signal. You can avoid this signal distortion only by removing any signal components above one-half the sampling frequency with lowpass filters before the signal is sampled.

Figure 7-4 shows a sine wave signal sampled at the indicated points. When these sampled data points are used to reconstruct the waveform, as shown by the dotted line, the signal appears to have a lower frequency than the original sine wave.

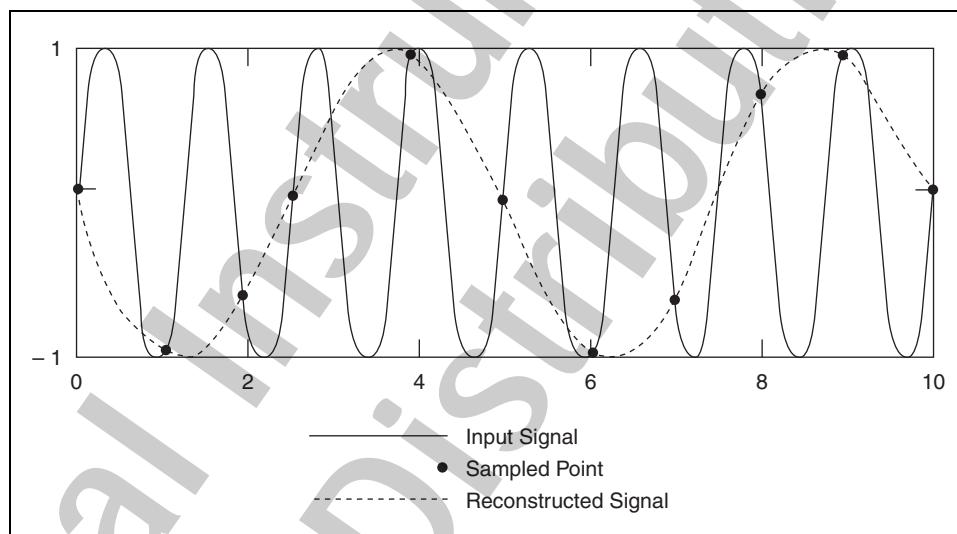


Figure 7-4. Undersampled Sine Wave Signal

You can increase the sampling rate or pass the signal through a lowpass filter to remove high-frequency components. Increasing the sampling rate can be expensive and often impractical, especially when the upper limit bandwidth of high-frequency noise can be much greater than the bandwidth of the signal of interest. Therefore, a common practice is to use lowpass filters to remove any frequency elements above the Nyquist frequency.

Only analog filters can prevent aliasing. Digital filters cannot remove aliased signals because it is impossible to remove aliasing after the signal has been sampled.

Typically, a programmable analog filter is used as a front-end signal conditioner for digitizing instruments, such as DAQ devices or modules. Analog filters are available with standard transfer functions that provide trade-offs in real filter characteristics, such as rolloff, passband ripple, and phase linearity. Standard transfer functions include Butterworth,

Chebyshev, Bessel, and elliptic filters. For example, Butterworth filters exhibit very flat frequency response in the passband, while Chebyshev filters provide steeper attenuation at the expense of some passband ripple. The Bessel filter provides a linear phase response over the entire passband, minimizing distortion of waveshapes but exhibits a less sharp rolloff and, therefore, less attenuation in the stopband. The Cauer elliptic filter, with its extremely sharp rolloff, is especially useful as an anti-aliasing filter for multichannel digitizing DAQ systems. However, the large phase nonlinearity makes it more appropriate for applications involving analysis of the frequency content of signals as opposed to phase content or waveform shape.

Figure 7-5 shows the transfer function of three signal conditioning modules used by National Instruments. The elliptic filter, with its extremely sharp rolloff, is especially useful as an anti-aliasing filter for multichannel digitizing DAQ systems.

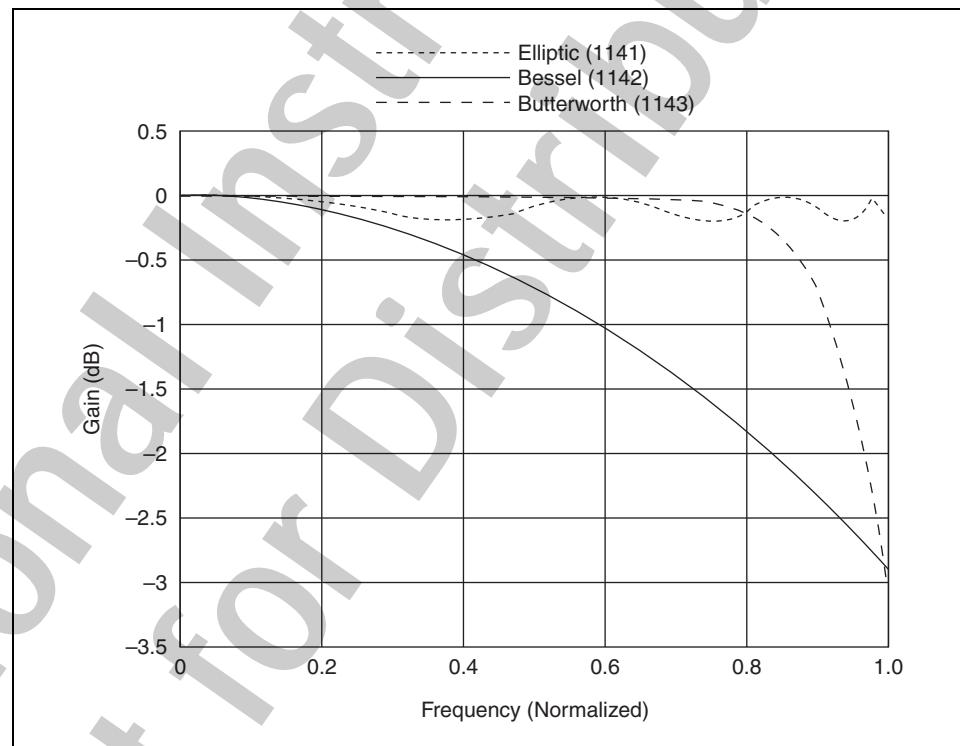


Figure 7-5. Transfer Function of Three Signal Conditioning Modules

The excellent attenuation performance of the elliptic filter above the cutoff frequency proves beneficial as an anti-aliasing filter. You can sample at a lower rate because of the sharper rolloff of the filter. For example, consider a 16-channel application with the signals of interest limited to a 10 kHz bandwidth. However, anti-aliasing filters are required to prevent distortion by unwanted noise signals above 10 kHz. With the SCXI-1141 eighth-order elliptic filters, for example, you can program the filter cutoff frequency for

the 16 channels for 10 kHz. With the cutoff frequency set to 10 kHz, the attenuation of the SCXI-1141 filters reaches 80 dB at about 15 kHz. Therefore, the sampling rate of the digitizing DAQ device can safely be set to twice the 15 kHz frequency, or 30 kS/s. Because 16 channels are sampled by the one DAQ device, the device is programmed for an aggregate sampling rate of 480 kS/s.

If you use a different type of filter with a less sharp rolloff, the DAQ device would need a much higher sampling rate. For example, a typical eighth-order Butterworth filter programmed for a cutoff frequency of 10 kHz reaches an attenuation of 80 dB at about 30 kHz. The sampling rate of the DAQ device used with this filter must be set at 60 kS/s per channel, or 960 kS/s aggregate. This higher sampling rate requires a more expensive DAQ device (or fewer channels per DAQ device), and greater data storage capacity. Therefore, the sharper attenuation of the elliptic filter reduces the requirements of the digitizer, allowing lower sampling rates and lower data storage requirements.

However, the elliptic filter has a large phase nonlinearity as shown in Figure 7-6. This nonlinear phase response of the filter makes the filter more appropriate for applications that involve analysis of frequency content as opposed to phase content or waveform shape.

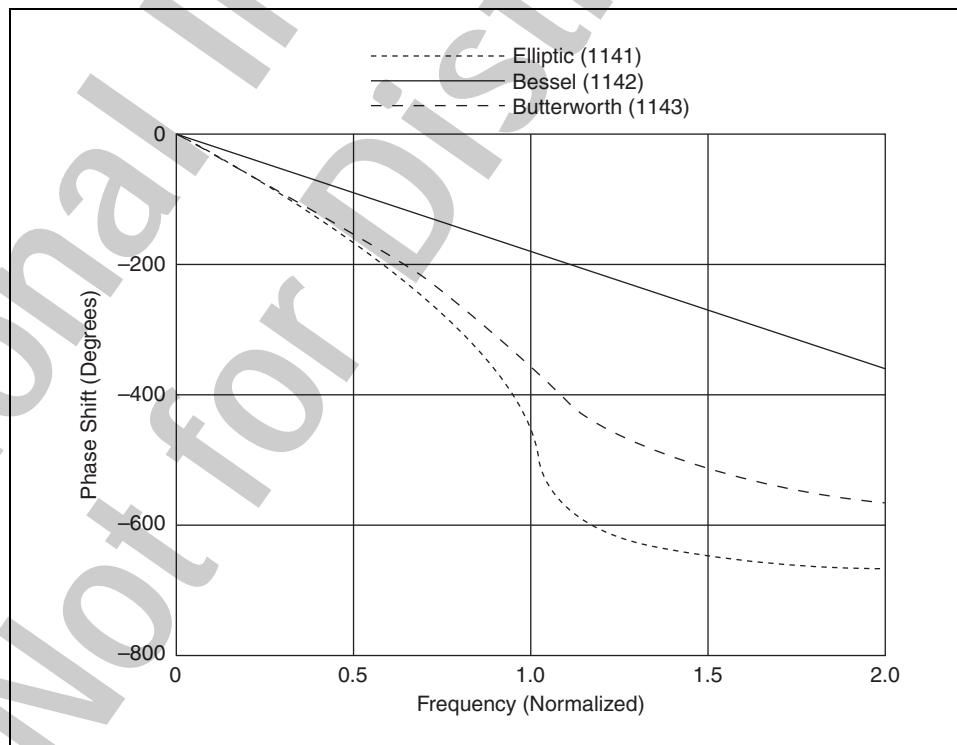


Figure 7-6. Phase Shift Versus Frequency For Filters

When you choose a data acquisition filter, consider both the magnitude and phase response of the filter for a particular application to ensure no loss of signal information occurs as a result of filtering.

Traditional analog filters are typically built with op-amps, resistors, and capacitors. However, switched-capacitor technology is commonly used in the implementation of adjustable anti-aliasing filters. Essentially, a switched capacitor replaces the resistor in the more traditional analog filter designs. Because the impedance of the switched capacitor is a function of the switching frequency, you can vary the cutoff frequency of the switched capacitor filter by varying the frequency of the clock signal that controls switching. Also, it is easier to accurately fabricate filters composed of op-amps and matched capacitors.

However, switched-capacitor filters have potential drawbacks. Used alone, switched-capacitor filters tend to be too noisy for analog conditioning applications. Therefore, the NI signal conditioning modules use a unique hybrid design of switched-capacitor filters and continuous-time filters to combine the technological and performance benefits of both. The main components of an NI signal conditioning filter stage consist of the switched-capacitor discrete-time filter, the programmable analog continuous-time prefilter, and the programmable analog continuous-time postfilter.

- **Switched-capacitor, discrete-time filter**—The cutoff frequency is easily controlled by varying the frequency of the input clock signal.
- **Programmable analog continuous-time prefilter**—Switched-capacitor filters are sampling devices and are subject to aliasing. Therefore, the analog prefilter attenuates any frequencies outside the Nyquist limit of the switching frequency.
- **Programmable analog continuous-time postfilter**—The output of the switched-capacitor filter, a staircase representation of the analog signal, is reconstructed with the analog postfilter. This postfilter also removes any feedthrough noise from the high-frequency clock that drives the switched-capacitor filter.

Isolation

Improper grounding of the system is one of the most common causes of measurement problems, noise, and damaged DAQ devices. Signal conditioning systems with isolation can prevent most of these problems. Such devices pass the signal from its source to the measurement device without a physical connection by using a transformer, optical, or capacitive coupling techniques. Besides breaking ground loops, isolation blocks high-voltage surges and rejects common-mode voltage, thereby, protecting operators and expensive measurement equipment.

Suppose you need to monitor temperature using thermocouples soldered to a high-voltage machine that radiates large electro-magnetic fields. Although the thermocouples output a differential voltage of less than 50 mV, this output voltage can be at a high output potential with respect to ground due to the capacitive coupling that the machine has with the thermocouple. This potential between both leads of a differential signal and ground is called the common-mode voltage. Ideally, it should be completely ignored by the measurement system. Connecting the thermocouple leads directly to a nonisolated device (which can typically handle 12 V of common-mode voltage) would probably damage the device. However, you can connect the thermocouple leads to an isolated signal conditioner, which rejects the high common-mode voltage, safely passing the 50 mV differential signal on to the measurement device for an accurate measurement.

Isolation Specifications

Manufacturers specify isolation differently. Some manufacturers just provide an isolation number without describing if that number is a signal level or a transient level. Without this information and an acute understanding of the I/O signals, you can damage the measurement system and possibly create a shock hazard for the system operators. Agencies such as Underwriters Laboratories (UL) and the International Electrotechnical Commission (IEC) designate compliance requirements for safe design of high voltage instrumentation. Products that display these symbols are tested, in some cases by the agency themselves, to ensure they meet their specifications.

In addition to looking for the seal of approval from one of these agencies, the safest way to determine the isolation rating of a signal conditioning system is to look for two key specifications—working voltage rating and installation category rating.

Working Voltage Rating

This specification describes the maximum continuous voltage that you can apply to inputs under normal working conditions. This specification is described with respect to the unit ground reference and includes both the signal level and any common-mode voltage associated with the signal.

Installation Category Rating

By definition, an installation category rating describes the locations where you can use a particular measurement device based on the possible transient signals at that location. In a more general sense, this specification describes the possible transient signals that the device can withstand.

Installation category ratings are described as Category I through Category IV, as shown in Figure 7-7.

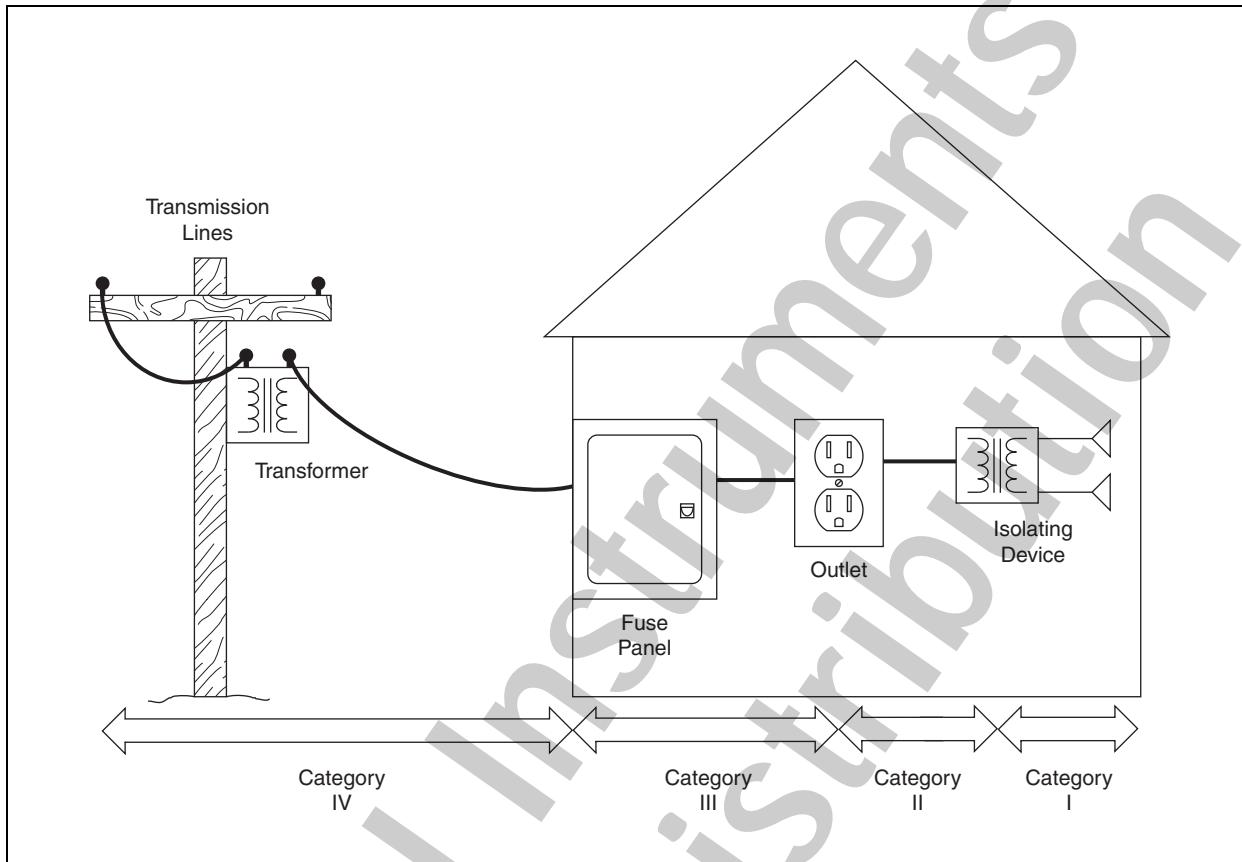


Figure 7-7. Installation Category Ratings

Depending on the location of the electrical distribution system, there is a certain amount of damping in the system. Damping occurs less toward the generating plant and increases as the transmission system spreads out. This damping reduces the overvoltages that are present in the system. The closer you are to the source, the higher the anticipated voltage transients. Depending on the location of the measurement system, certain precautions need to be made to protect the measurement system from potentially dangerous overvoltages that are present in the distribution system. The IEC created the following four categories to partition circuits with different levels of overvoltages.

- Installation Category IV—Distribution level. Equipment, such as generators, substations, and transformers.
- Installation Category III—Fixed installation. Equipment permanently connected to the distribution network, such as air conditioners and furnaces.

- Installation Category II—Equipment consuming energy from a fixed installation system. This includes equipment, such as drills, televisions, radios, and computers.
- Installation Category I—Equipment for connection to circuits where transient overvoltages are limited to a sufficiently low level by design. Category I equipment includes equipment, such as low-voltage power supplies.

Safety isolation provides a safety barrier between the user and their equipment and high voltages and overvoltage transients. Equipment that provides safety isolation is a very desirable feature.

All isolated National Instruments SCXI modules are double insulated for a continuous working voltage of 250 V_{rms}. In addition, they are tested by applying a 2,300 V source to their inputs for a minute to make sure that the module does not fail with overvoltage transients. The isolated SCXI modules adhere to the IEC-1010 specification for a Category II installation.

Refer to the following IEC publications for more information about installation categories.

- IEC 664-1—Installation coordination for equipment within low-voltage systems.
- IEC 1010-1—Safety requirements for electrical equipment for measurement, control, and laboratory use.

D. Temperature Measurements

Sensors convert physical phenomena, such as temperature, into electrical properties, such as voltage or resistance. Sensor characteristics determine many of the signal conditioning requirements of a data acquisition system.

Thermocouples

One of the most frequently used temperature transducers is the thermocouple. Thermocouples are very rugged and inexpensive and can operate over a wide temperature range. A thermocouple is created whenever two dissimilar metals touch, and a small open-circuit voltage is produced as a function of temperature differences across the metals. This thermoelectric voltage is known as the Seebeck voltage, named after Thomas Seebeck, who discovered it in 1821. The voltage is nonlinear with respect to temperature. However, for small changes in temperature, the voltage is approximately linear, or

$$\Delta V \approx S\Delta T$$

where ΔV is the change in voltage, S is the Seebeck coefficient, and ΔT is the change in temperature.

S varies with changes in temperature, which causes the output voltages of thermocouples to be nonlinear over their operating ranges. Several types of thermocouples are available and are designated by capital letters that indicate their composition according to American National Standards Institute (ANSI) conventions. For example, a J-type thermocouple has one iron conductor and one constantan (a copper-nickel alloy) conductor.

You can monitor thermocouples with versatile PC-based data acquisition systems. Thermocouples have some special signal conditioning requirements.

Thermocouple Circuits

To measure a thermocouple Seebeck voltage, you cannot simply connect the thermocouple to a voltmeter or other measurement system because connecting the thermocouple wires to the measurement system creates additional thermoelectric circuits.

Consider the circuit illustrated in Figure 7-8, in which a J-type thermocouple is in a candle flame that has a temperature you want to measure. The two thermocouple wires are connected to the copper leads of a DAQ device.

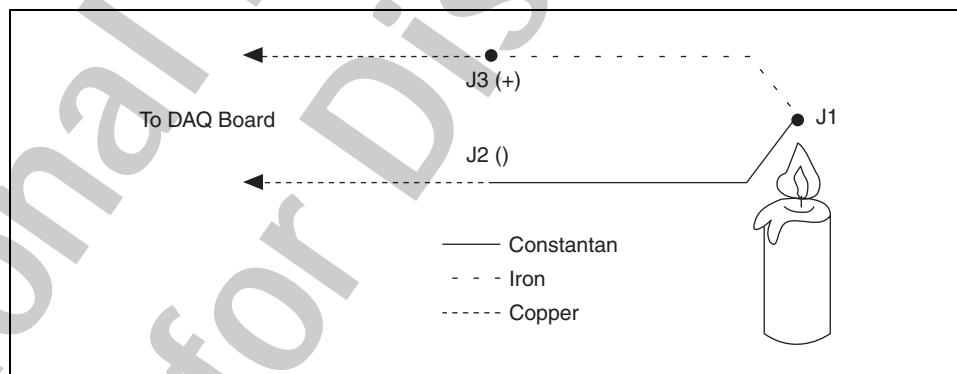


Figure 7-8. J-Type Thermocouple

Notice that the circuit contains three dissimilar metal junctions—J1, J2, and J3. J1 is the thermocouple junction that generates a Seebeck voltage proportional to the temperature of the candle flame. J2 and J3 each have their own Seebeck coefficient and generate their own thermoelectric voltage proportional to the temperature at the DAQ terminals. To determine the voltage contribution from J1, you need to know the temperatures of junctions J2 and J3 and the voltage-to-temperature relationships for these junctions. You can then subtract the contributions of the parasitic thermocouples at J2 and J3 from the measured voltage.

Cold-Junction Compensation

Thermocouples require some form of temperature reference to compensate for parasitic thermocouples. The term cold junction comes from the traditional practice of holding this reference junction at $0\text{ }^{\circ}\text{C}$ in an ice bath. Thermocouple reference tables created by the National Institute of Standards and Technology (NIST) are illustrated in Figure 7-9.

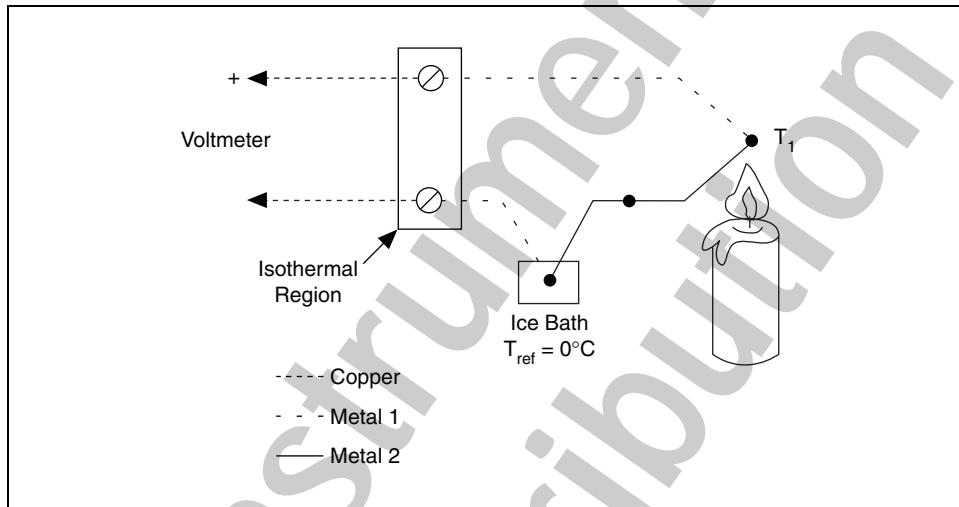


Figure 7-9. Traditional Temperature Measurement with Reference Junction Held at $0\text{ }^{\circ}\text{C}$

In Figure 7-9, the measured voltage depends on the difference in temperatures T_1 and T_{ref} . In this case, T_{ref} is $0\text{ }^{\circ}\text{C}$. Notice that because the voltmeter lead connections are the same temperature, or isothermal, the voltages generated at these two points are equal and opposing. Therefore, the net voltage error added by these connections is zero.

Under these conditions, if the measurement temperature is above $0\text{ }^{\circ}\text{C}$, a thermocouple has a positive output. If the measurement temperature is below $0\text{ }^{\circ}\text{C}$, the output is negative. When the reference junction and the measurement junction are the same temperature, the net voltage is zero.

Although an ice bath reference is accurate, it is not always practical. A more practical approach is to measure the temperature of the reference junction with a direct-reading temperature sensor and subtract the parasitic thermocouple thermoelectric voltage contributions. This process is called cold-junction compensation (CJC). You can simplify computing CJC by taking advantage of some thermocouple characteristics.

By using the Thermocouple Law of Intermediate Metals and making some simple assumptions, you can see that the voltage the DAQ device measures in Figure 7-8 depends only on the thermocouple type, the thermocouple voltage, and the cold-junction temperature. The measured voltage is in fact

independent of the composition of the measurement leads and the cold junctions, J2 and J3.

According to the Thermocouple Law of Intermediate Metals, illustrated in Figure 7-10, inserting any type of wire into a thermocouple circuit has no effect on the output as long as both ends of that wire are the same temperature, or isothermal.

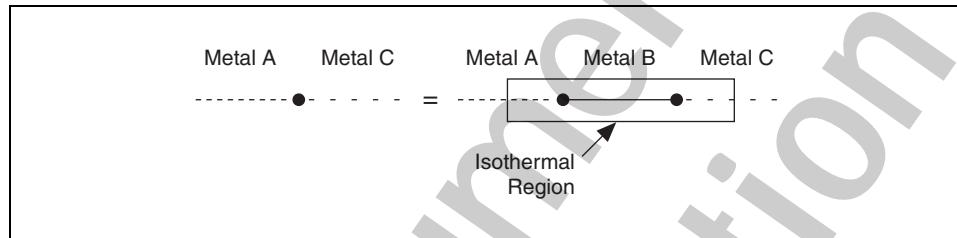


Figure 7-10. Thermocouple Law of Intermediate Metals

Consider the circuit in Figure 7-11. This circuit is similar to the previously described circuit in Figure 7-8, but a short length of constantan wire has been inserted just before junction J3, and the junctions are assumed to be held at identical temperatures. Assuming that junctions J3 and J4 are the same temperature, the Thermocouple Law of Intermediate Metals indicates that the circuit in Figure 7-11 is electrically equivalent to the circuit in Figure 7-8. Consequently, any result taken from the circuit in Figure 7-11 also applies to the circuit illustrated in Figure 7-8.

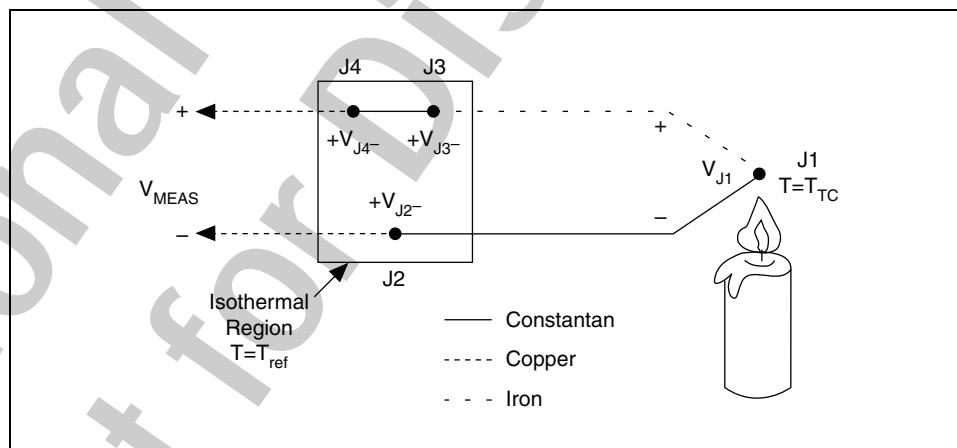


Figure 7-11. Inserting an Extra Lead in the Isothermal Region

In Figure 7-11, junctions J2 and J4 are the same type (copper-constantan). Because both are in the isothermal region, J2 and J4 are also the same temperature. The junctions occur in opposite directions, however, so their total contribution to the measured voltage is zero. Junctions J1 and J3 are both iron-constantan junctions and also point in opposite directions but may

be different temperatures. Therefore, junctions J1 and J3 are the only two junctions with outputs that have any effect on the total voltage measured.

Using the notation $V_{Jx}(T_y)$ to indicate the voltage generated by the junction J_x at temperature T_y , the general thermocouple problem is reduced to the following equation:

$$V_{MEAS} = V_{J1}(T_{TC}) + V_{J3}(T_{ref}) \quad (7-1)$$

where V_{MEAS} is the voltage the DAQ device measures, T_{TC} is the temperature of the thermocouple at J1, and T_{ref} is the temperature of the reference junction.

Notice that in Equation 7-1, $V_{Jx}(T_y)$ is a voltage generated at temperature T_y with respect to some reference temperature. Because long as both V_{J1} and V_{J3} are functions of temperature relative to the same reference temperature, Equation 7-1 is valid. As stated earlier, for example, NIST thermocouple reference tables are generated with the reference junction held at 0 °C.

Because junction J3 is the same type as J1 but in the opposite direction, $V_{J3}(T_{ref}) = -V_{J1}(T_{ref})$. Because V_{J1} is the voltage that the thermocouple type undergoing testing generates, this voltage can be renamed V_{TC} . Therefore, Equation 7-1 is rewritten as follows:

$$V_{MEAS} = V_{TC}(T_{TC}) - V_{TC}(T_{ref}) \quad (7-2)$$

Therefore, by measuring V_{MEAS} and T_{ref} and knowing the voltage-to-temperature relationship of the thermocouple, you can determine the temperature of the thermocouple.

The techniques for implementing cold-junction compensation require that the temperature at the reference junction be sensed with a direct-reading sensor. A direct-reading sensor has an output that depends only on the temperature of the measurement point. Semiconductor sensors, thermistors, or RTDs are commonly used to measure the reference-junction temperature. For example, several SCXI terminal blocks include thermistors that are located near the screw terminals to which thermocouple wires are connected.



Note NI-DAQmx driver software includes built-in routines that perform the required software compensation.

Linearizing the Data

Thermocouple output voltages are highly nonlinear. The Seebeck coefficient can vary by a factor of three or more over the operating temperature range of some thermocouples. For this reason, you must either approximate the thermocouple voltage-versus-temperature curve using polynomials or use the following look-up table. The polynomials are in the following form:

$$T = a_0 + a_1 v + a_2 v^2 + \dots + a_n v^n \quad (7-3)$$

where v is the thermocouple voltage in volts, T is the temperature in degrees Celsius, and a_0 through a_n are coefficients that are specific to each thermocouple type. NI software can linearize the thermocouple output voltages for various thermocouples.

Table 7-1. Thermocouple Voltage Output Extremes (mV)

Thermocouple Type	Conductor		Temperature Range (°C)	Voltage Range (mV)	Seebeck Coefficient ($\mu\text{V}/^\circ\text{C}$)
	Positive	Negative			
E	Chromel	Constantan	-270 to 1,000	-9.835 to 76.358	58.70 at 0 °C
J	Iron	Constantan	-210 to 1,200	-8.096 to 69.536	50.37 at 0 °C
K	Chromel	Alumel	-270 to 1,372	-6.548 to 54.874	39.48 at 0 °C
T	Copper	Constantan	-270 to 400	-6.258 to 20.869	38.74 at 0 °C
S	Platinum-10% Rhodium	Platinum	-50 to 1,768	-0.236 to 18.698	10.19 at 600 °C
R	Platinum-13% Rhodium	Platinum	-50 to 1,768	-0.226 to 21.108	11.35 at 600 °C

Resistance Temperature Detectors (RTD)

Resistance temperature detectors (RTDs) operate on the principle of changes in electrical resistance of pure metals and are characterized by a linear positive change in resistance with temperature. Typical elements used for RTDs include nickel (Ni) and copper (Cu), but platinum (Pt) is by far the most common because of its wide temperature range, accuracy, and stability.

RTDs are constructed by one of two different manufacturing configurations. Wire-wound RTDs are constructed by winding a thin wire into a coil. A more common configuration is the thin-film element, which consists of a very thin layer of metal laid out on a plastic or ceramic substrate. Thin-film elements are cheaper and more widely available because they can achieve

higher nominal resistances with less platinum. To protect the RTD, a metal sheath encloses the RTD element and the lead wires connected to it.

RTDs are popular because of their excellent stability, and exhibit the most linear signal with respect to temperature of any electronic temperature sensor. They are generally more expensive than alternatives, however, because of the careful construction and use of platinum. RTDs are also characterized by a slow response time and low sensitivity; and because they require current excitation, they can be prone to self-heating.

RTDs are commonly categorized by their nominal resistance at 0 °C. Typical nominal resistance values for platinum thin-film RTDs include 100 Ω and 1000 Ω. The relationship between resistance and temperature is very nearly linear and follows the equation below.

$$\text{For } < 0 \text{ }^{\circ}\text{C}, R_T = R_0 [1 + aT + bT^2 + cT^3 (T - 100)]$$

$$\text{For } > 0 \text{ }^{\circ}\text{C}, R_T = R_0 [1 + aT + bT^2]$$

where R_T = resistance at temperature T, R_0 = nominal resistance, and a, b, and c are constants used to scale the RTD.

Thermistors

Thermistors (thermally sensitive resistors) are similar to RTDs in that they are electrical resistors whose resistance changes with temperature.

Thermistors are manufactured from metal oxide semiconductor material which is encapsulated in a glass or epoxy bead.

Thermistors have a very high sensitivity, making them extremely responsive to changes in temperature. For example, a 2252 Ω thermistor has a sensitivity of $-100 \text{ } \Omega/\text{ }^{\circ}\text{C}$ at room temperature. In comparison, a 100 Ω RTD has a sensitivity of $0.4 \text{ } \Omega/\text{ }^{\circ}\text{C}$. Thermistors also have a low thermal mass that results in fast response times, but are limited by a small temperature range.

Thermistors have either a negative temperature coefficient (NTC) or a positive temperature coefficient (PTC). The first has a resistance which decreases with increasing temperature and the latter exhibits increased resistance with increasing temperature.

Signal Conditioning for RTDs and Thermistors

Because RTDs and thermistors are resistive devices, you must supply them with an excitation current and then read the voltage across their terminals. If extra heat cannot be dissipated, I²R heating caused by the excitation current can raise the temperature of the sensing element above that of the ambient temperature. Self-heating will actually change the resistance of the

RTD or thermistor, causing error in the measurement. The effects of self-heating can be minimized by supplying lower excitation current.

The easiest way to connect an RTD or thermistor to a measurement device is with a 2-wire connection.

With this method, the two wires that provide the RTD or thermistor with its excitation current are also used to measure the voltage across the sensor. Because of the low nominal resistance of RTDs, measurement accuracy can be drastically affected by lead wire resistance. For example, lead wires with a resistance of $1\ \Omega$ connected to a $100\ \Omega$ platinum RTD cause a 1% measurement error.

A 3-wire or 4-wire connection method can eliminate the effects of lead wire resistance. The connection places leads on a high impedance path through the measurement device, effectively eliminating error caused by lead wire resistance. It is not necessary to use a 3 or 4-wire connection method for thermistors because they typically have much higher nominal resistance values than RTDs.

RTD and thermistor output signals are typically in the millivolt range, making them susceptible to noise. Lowpass filters are commonly used in RTD and thermistor data acquisition systems to effectively eliminate high frequency noise in RTD and thermistor measurements. For instance, lowpass filters are useful for removing the 60 Hz power line noise that is prevalent in most laboratory and plant settings.

E. Strain, Pressure, Load, and Torque Measurements

Strain (ϵ) is the amount of deformation of a body due to an applied force.

$$\epsilon = \frac{\Delta L}{L}$$

More specifically, strain is defined as the fractional change in length, as shown in Figure 7-12.

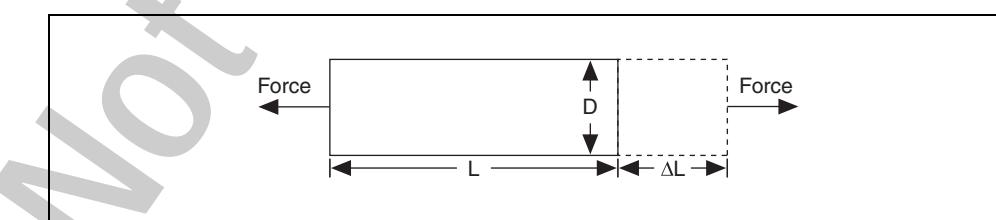


Figure 7-12. Strain Components

Strain can be positive (tensile) or negative (compressive). Although dimensionless, strain is sometimes expressed in units, such as in./in. or mm/mm. In practice, the magnitude of measured strain is very small. Therefore, strain is often expressed as microstrain ($\mu\epsilon$), which is $\epsilon \times 10^{-6}$.

When a bar is strained with a uniaxial force, as Figure 7-12, a phenomenon known as Poisson Strain causes the girth of the bar, D, to contract in the transverse, or perpendicular, direction. Poisson Strain expresses both the thinning and elongation that occurs in a strained bar. The magnitude of this transverse contraction is a material property indicated by its Poisson's Ratio. The Poisson's Ratio ν of a material is defined as the negative ratio of the strain in the transverse direction (perpendicular to the force) to the strain in the axial direction (parallel to the force), or $\nu = -\epsilon_T/\epsilon$. Poisson's Ratio for steel, for example, ranges from 0.25 to 0.3.

Strain Gage

Although there are several methods of measuring strain, the most common is with a strain gage, a device whose electrical resistance varies in proportion to the amount of strain in the device. For example, the piezoresistive strain gage is a semiconductor device whose resistance varies nonlinearly with strain. The most widely used gage is the bonded metallic strain gage.

The metallic strain gage consists of a very fine wire or, more commonly, metallic foil arranged in a grid pattern. The grid pattern maximizes the amount of metallic wire or foil subject to strain in the parallel direction as shown in Figure 7-13. The cross sectional area of the grid is minimized to reduce the effect of shear strain and Poisson Strain. The grid is bonded to a thin backing, called the carrier, which is attached directly to the test specimen. Therefore, the strain experienced by the test specimen is transferred directly to the strain gage, which responds with a linear change in electrical resistance. Strain gages are available commercially with nominal resistance values from 30 to 3,000 Ω , with 120 Ω , 350 Ω , and 1,000 Ω being the most common values.

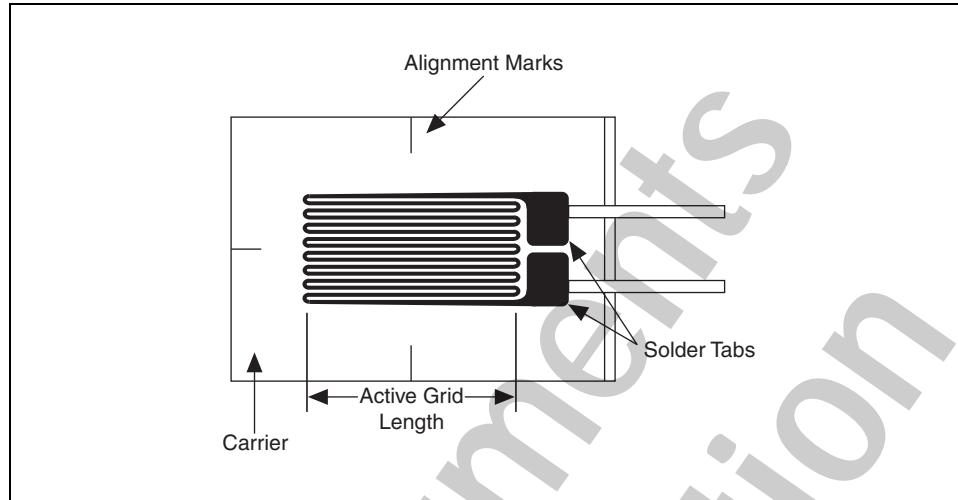


Figure 7-13. Strain Gage Grid Pattern

It is very important that the strain gage is properly mounted onto the test specimen so that the strain is accurately transferred from the test specimen through the adhesive and strain gage backing to the foil itself. Manufacturers of strain gages are the best source of information on proper mounting of strain gages.

A fundamental parameter of the strain gage is its sensitivity to strain, expressed quantitatively as the gage factor (GF). Gage factor is defined as the ratio of fractional change in electrical resistance to the fractional change in length (strain):

$$GF = \frac{\frac{\Delta R}{R}}{\frac{\Delta L}{L}} = \frac{\Delta R}{R} \cdot \frac{L}{\Delta L}$$

The gage factor for metallic strain gages is typically around 2.

Ideally, the resistance of the strain gage would change only in response to applied strain. However, strain gage material, and the specimen material to which the gage is applied, also responds to changes in temperature. Strain gage manufacturers attempt to minimize sensitivity to temperature by processing the gage material to compensate for the thermal expansion of the specimen material for which the gage is intended. While compensated gages reduce the thermal sensitivity, they do not totally remove it. For example, consider a gage compensated for aluminum that has a temperature coefficient of 23 ppm/°C. With a nominal resistance of 1,000 Ω, GF = 2, the equivalent strain error is still 11.5 με/°C. Therefore, additional temperature compensation is important.

Strain Gage Measurement

Strain measurements rarely involve quantities larger than a few millistrain ($\epsilon \times 10^{-3}$). Therefore, measuring the strain requires accurate measurement of very small changes in resistance. For example, suppose a test specimen undergoes a substantial strain of $500 \mu\epsilon$. A strain gage with a gage factor GF = 2 exhibits a change in electrical resistance of only $2 \times (500 \times 10^{-6}) = 0.1\%$. For a 120Ω gage, this is a change of only 0.12Ω .

To measure such small changes in resistance and to compensate for the temperature sensitivity discussed in the previous section, strain gages are almost always used in a bridge configuration with a voltage or current excitation source. The general Wheatstone bridge, shown in Figure 7-14, consists of four resistive arms with an excitation voltage, V_{EX} , that is applied across the bridge.

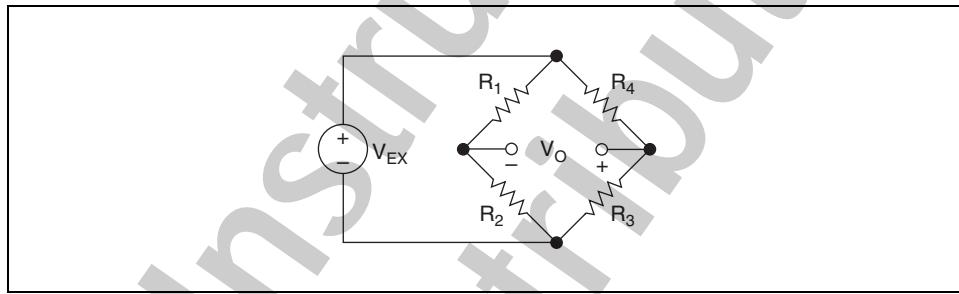
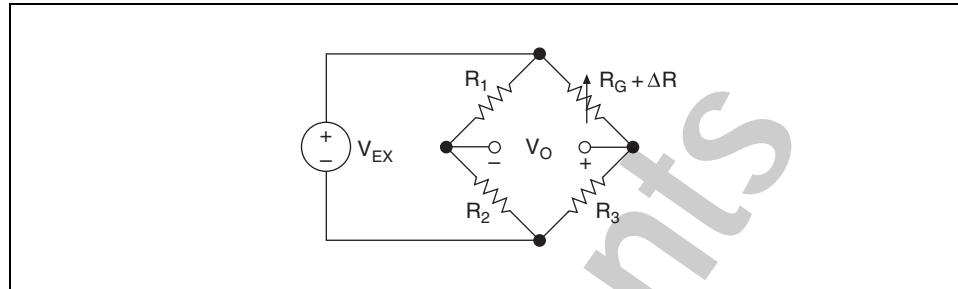


Figure 7-14. General Wheatstone Bridge

The output voltage of the bridge, V_O , equals

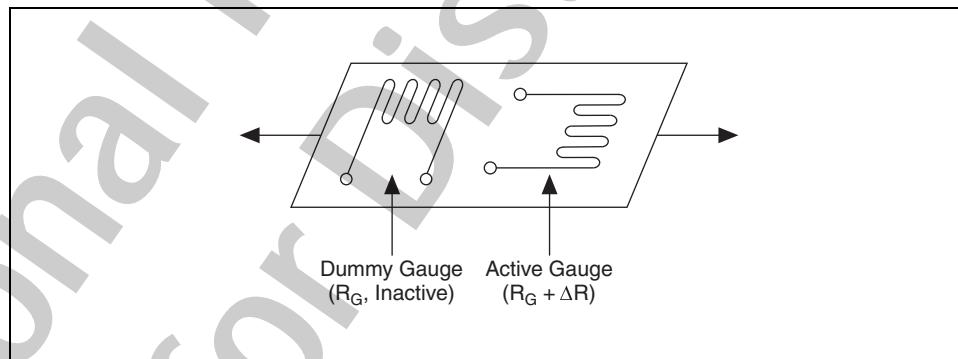
$$V_O = \left[\frac{R_3}{R_3 + R_4} - \frac{R_2}{R_1 + R_2} \right] \times V_{EX}$$

From this equation, it is apparent that when $R_1/R_2 = R_3/R_4$, the voltage output V_O is zero. Under these conditions, the bridge is said to be balanced. Any change in resistance in any arm of the bridge results in a nonzero output voltage. Therefore, if you replace R_4 in Figure 7-15 with an active strain gage, any changes in the strain gage resistance unbalances the bridge and produce a nonzero output voltage. If the nominal resistance of the strain gage is designated as R_G , the strain-induced change in resistance, ΔR , can be expressed as $\Delta R = R_G \times GF \times \epsilon$. Assuming that $R_1 = R_2$ and $R_3 = R_G$, the bridge equation above can be rewritten to express V_O/V_{EX} as a function of strain. Notice the presence of the $1/(1 + GF \times \epsilon/2)$ term that indicates the nonlinearity of the quarter-bridge output with respect to strain.

**Figure 7-15.** Replace Resistor with Active Strain Gage

$$\frac{V_O}{V_{EX}} = -\frac{GF \times \varepsilon}{4} \left(\frac{1}{1 + GF \times \frac{\varepsilon}{2}} \right)$$

By using two strain gages in the bridge, the effect of temperature can be avoided. For example, Figure 7-16 shows a strain gage configuration where one gage is active ($R_G + \Delta R$), and a second gage is placed transverse to the applied strain. Therefore, the strain has little effect on the second gage, called the dummy gage. However, any changes in temperature affects both gages in the same way. Because the temperature changes are identical in the two gages, the ratio of their resistance does not change, the voltage V_O does not change, and the effects of the temperature change are minimized.

**Figure 7-16.** Active Strain Gage and Dummy Strain Gage

Alternatively, you can double the sensitivity of the bridge to strain by making both gages active, although in different directions. For example, Figure 7-17 shows a bending beam application with one bridge mounted in tension ($R_G + \Delta R$) and the other mounted in compression ($R_G - \Delta R$). This half-bridge configuration, whose circuit diagram is also shown in Figure 7-17, yields an output voltage that is linear and approximately doubles the output of the quarter-bridge circuit.

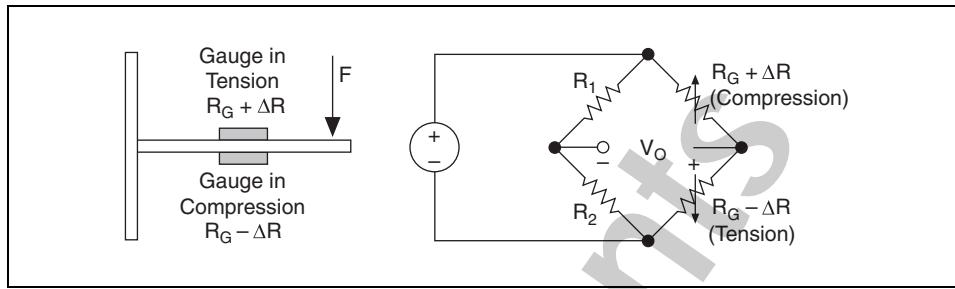


Figure 7-17. Half-bridge Circuit

$$\frac{V_O}{V_{EX}} = \frac{GF \times \epsilon}{2}$$

Finally, you can further increase the sensitivity of the circuit by making all four of the arms of the bridge active strain gages and mounting two gages in tension and two gages in compression. Figure 7-18 shows the full-bridge circuit.

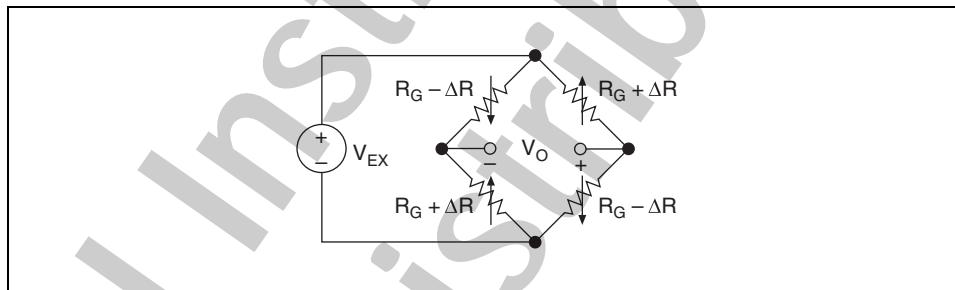


Figure 7-18. Full-bridge Circuit

$$\frac{V_O}{V_{EX}} = -G \times \epsilon$$

The equations given here for the Wheatstone bridge circuits assume an initially balanced bridge that generates zero output when no strain is applied. In practice, however, resistance tolerances and strain induced by gage application generates some initial offset voltage. This initial offset voltage is typically handled in two ways. First, you can use a special offset-nulling, or balancing, circuit to adjust the resistance in the bridge to rebalance the bridge to zero output. Alternatively, you can measure the initial unstrained output of the circuit and compensate in software. Refer to the *Strain Gage Equations* section of this lesson for equations for quarter-, half-, and full-bridge circuits that express strain and that take initial output voltages into account. These equations also include the effect of resistance in the lead wires connected to the gages.

Lead Wire Resistance

The figures and equations in the previous section ignore the resistance in the lead wires of the strain gage. While ignoring the lead resistances can be beneficial to understanding the basics of strain gage measurements, doing so in practice can be very dangerous. For example, consider the two-wire connection of a strain gage shown in the top half of Figure 7-19. Suppose each lead wire connected to the strain gage is 15 m long with lead resistance R_L equal to 1 Ω. Therefore, the lead resistance adds 2 Ω of resistance to that arm of the bridge. Besides adding an offset error, the lead resistance also desensitizes the output of the bridge. From the strain equations in the *Strain Gage Equations* section of this lesson you can see that the amount of desensitization is quantified by the term $(1 + R_L/R_G)$. You can compensate for this error by measuring the lead resistance R_L and using the measured value in the strain equations. However, a more difficult problem arises from changes in the lead resistance due to temperature changes. Given typical temperature coefficients for copper wire, a slight change in temperature can generate a measurement error of several μe .

Therefore, the preferred connection scheme for quarter-bridge strain gages is the three-wire connection, shown in the bottom half of Figure 7-19. In this configuration, R_{L1} and R_{L3} appear in adjacent arms of the bridge. Any changes in resistance due to temperature cancel each other. The lead resistance in the third lead, R_{L2} , is connected to the measurement input. This lead carries very little current, and the effect of its lead resistance is negligible.

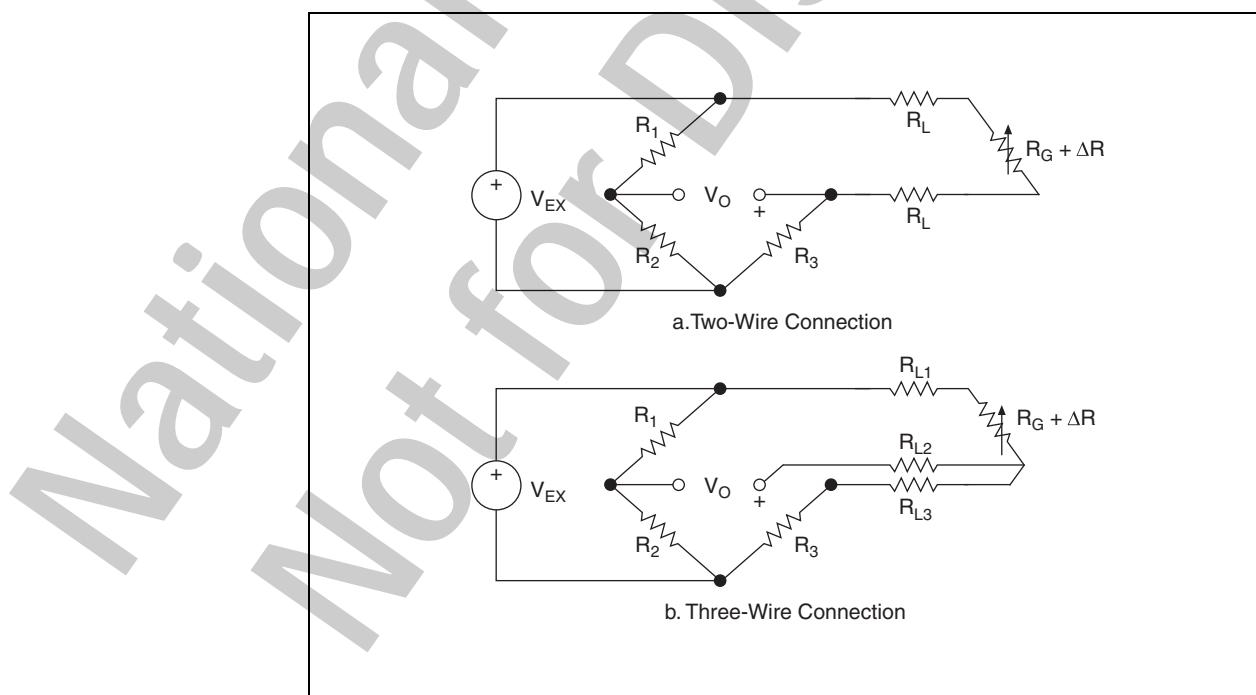


Figure 7-19. Quarter-bridge Strain Gage Connections

Signal Conditioning for Strain Gages

Strain gage measurement involves sensing extremely small changes in resistance. Therefore, proper selection and use of the bridge, signal conditioning, wiring, and data acquisition components are required for reliable measurements.

Bridge Completion

Unless you are using a full-bridge strain gage sensor with four active gages, you need to complete the bridge with reference resistors. Therefore, strain gage signal conditioners typically provide half-bridge completion networks consisting of two high-precision reference resistors. Figure 7-20 shows the wiring of a half-bridge strain gage circuit to a conditioner with completion resistors R_1 and R_2 . The nominal resistance of the completion resistors is less important than how well the two resistors are matched. Ideally, the resistors are well matched and provide a stable reference voltage of $V_{EX}/2$ to the negative input lead of the measurement channel. The high resistance of the completion resistors helps minimize the current draw from the excitation voltage.

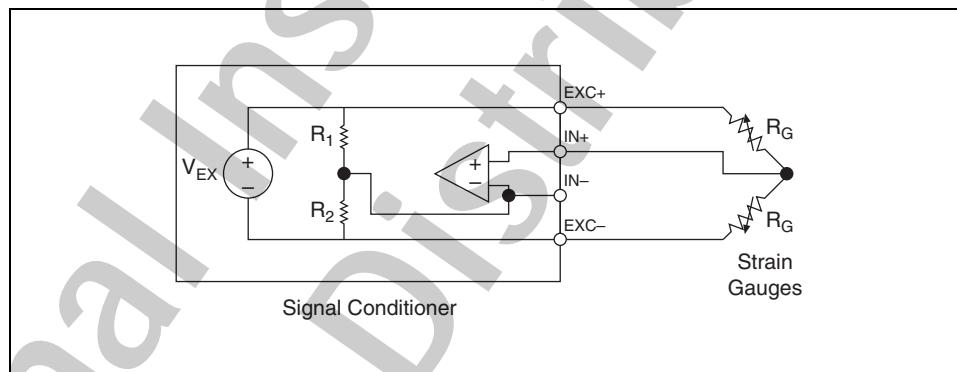


Figure 7-20. Wiring of a Half-Bridge Strain Gage Circuit

For more information on the different bridge configurations, refer to the *Bridge Configurations* topic in the *NI-DAQmx Help*.

Bridge Excitation

Strain gage signal conditioners typically provide a constant voltage source to power the bridge. While there is no standard voltage level that is recognized industry wide, excitation voltage levels of around 3 V and 10 V are common. While a higher excitation voltage generates a proportionately higher output voltage, the higher voltage also can cause larger errors due to self-heating. It is important that the excitation voltage be accurate and stable. Alternatively, you can use a less accurate or stable voltage and accurately measure, or sense, the excitation voltage so the correct strain is calculated.

Remote Sensing

If the strain-gage circuit is located a distance away from the signal conditioner and excitation source, a possible source of error is voltage drops caused by resistance in the wires that connects the excitation voltage to the bridge. Therefore, some signal conditioners include a feature called remote sensing to compensate for this error.

There are two common methods of remote sensing. With feedback remote sensing, you connect extra sense wires to the point where the excitation voltage wires connect to the bridge circuit. The extra sense wires serve to regulate the excitation supply to compensate for lead losses and deliver the needed voltage at the bridge.

An alternative remote sensing scheme uses a separate measurement channel to measure directly the excitation voltage delivered across the bridge. Because the measurement channel leads carry very little current, the lead resistance has negligible effect on the measurement. The measured excitation voltage is then used in the voltage-to-strain conversion to compensate for lead losses.

Signal Amplification

The output of strain gages and bridges is relatively small. In practice, most strain-gage bridges and strain-based transducers output less than $10 \mu\text{V/V}$ ($10 \mu\text{V}$ of output per volt of excitation voltage). With a 10 V excitation voltage, the output signal is $100 \mu\text{V}$. Therefore, strain gage signal conditioners usually include amplifiers to boost the signal level to increase measurement resolution and improve signal-to-noise ratios. SCXI signal conditioning modules, for example, include configurable gain amplifiers with gains up to 2,000.

Bridge Balancing, Offset Nulling

When a bridge is installed, it is very unlikely that the bridge outputs exactly zero volts when no strain is applied. Rather, slight variations in resistance among the bridge arms and lead resistance generate some nonzero initial offset voltage. There are a few different ways that a system can handle this initial offset voltage.

Software Compensation

The first method compensates for the initial voltage in software. With this method, you take an initial measurement before strain input is applied. This initial voltage is then used in the strain equations listed in the *Strain Gage Equations* section of this lesson. This method is simple, fast, and requires no manual adjustments. The disadvantage of the software compensation method is that the offset of the bridge is not removed. If the

offset is large enough, it limits the amplifier gain you can apply to the output voltage, thus limiting the dynamic range of the measurement.

Offset-Nulling Circuit

The second balancing method uses an adjustable resistance, or potentiometer, to physically adjust the output of the bridge to zero. For example, Figure 7-21 shows the offset-nulling circuit of the SCXI-1321 terminal block. By varying the position of the potentiometer (R_{POT}), you can control the level of the bridge output and set the initial output to zero volts. The value of R_{NULL} sets the range that the circuit can balance.

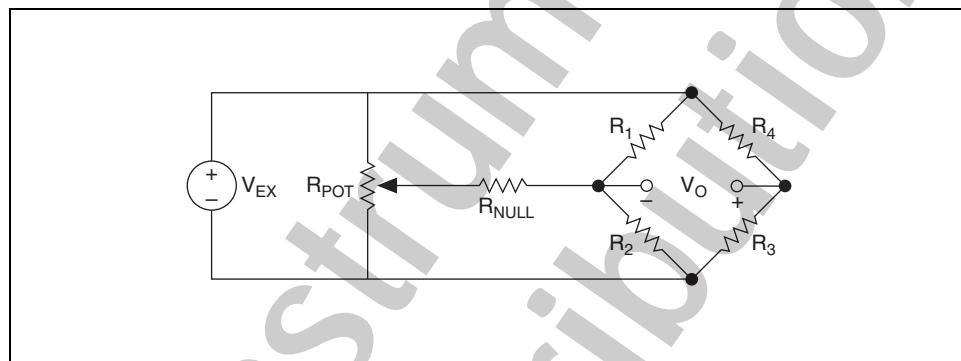


Figure 7-21. Offset-Nulling Circuit of SCXI-1321 Terminal Block

Buffered Offset Nulling

The third method, like the software method, does not affect the bridge directly. With buffered nulling, a nulling circuit adds an adjustable DC voltage to the output of the instrumentation amplifier. For example, the SC-2043-SG strain gage accessory uses this method. The SC-2043-SG includes a user-adjustable potentiometer that can add $\pm 50 \mu\text{V}$ to the output of an instrumentation amplifier that has a fixed gain of 10. Therefore, the nulling range, referred to input, is $\pm 5 \mu\text{V}$.

Shunt Calibration

The normal procedure to verify the output of a strain-gage measurement system relative to some predetermined mechanical input or strain is called shunt calibration. Shunt calibration involves simulating the input of strain by changing the resistance of an arm in the bridge by some known amount. This is accomplished by shunting, or connecting, a large resistor of known value across one arm of the bridge, creating a known ΔR . You can measure the output of the bridge and compare it to the expected voltage value. You can use the results to correct span errors in the entire measurement path or to simply verify general operation to gain confidence in the setup.

Strain Gage Equations

This section includes the complete strain-gage equations for several types of bridge configurations. These equations are included as callable functions (with source code) in NI-DAQ. The function names are `Strain_Convert` and `Strain_Buf_Convert`. In LabVIEW, these equations are included in the Convert Strain Gage Reading VI.

To simplify the equations and account for unbalanced bridges in the nonstrained state, the ratio V_r is

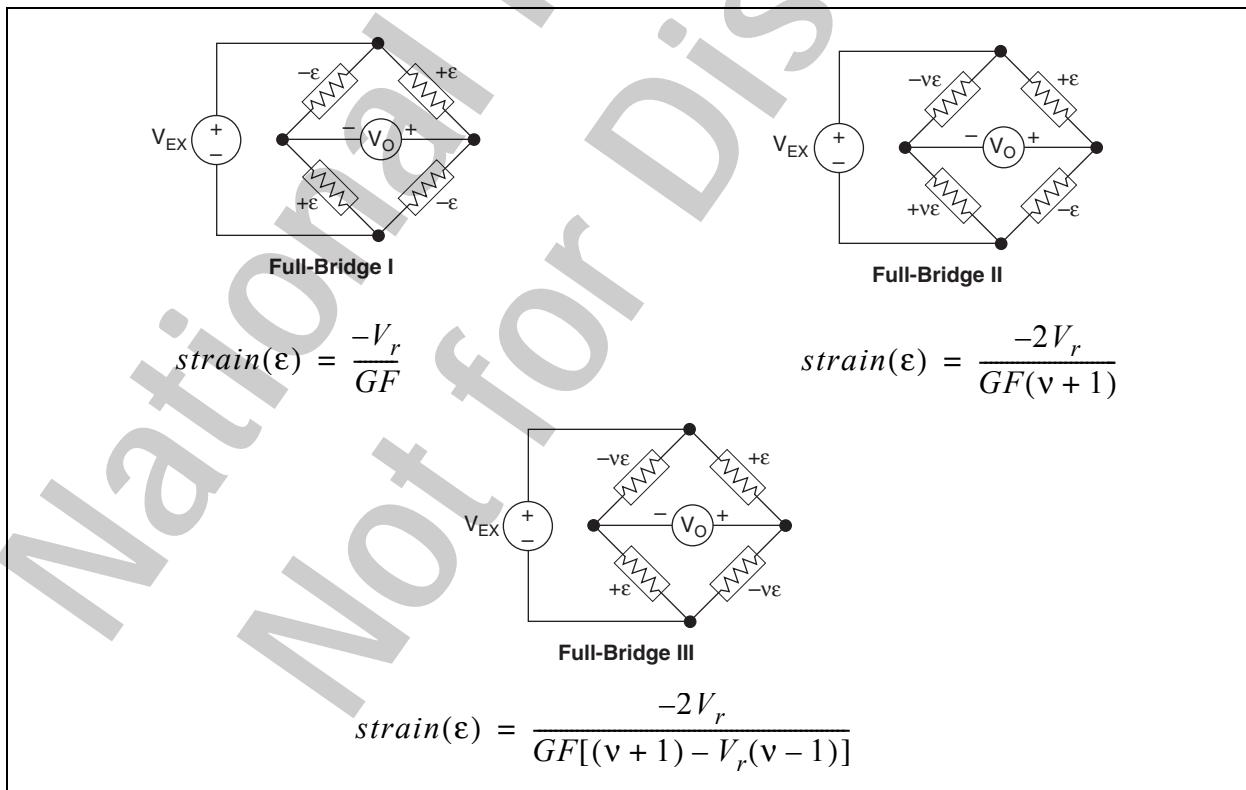
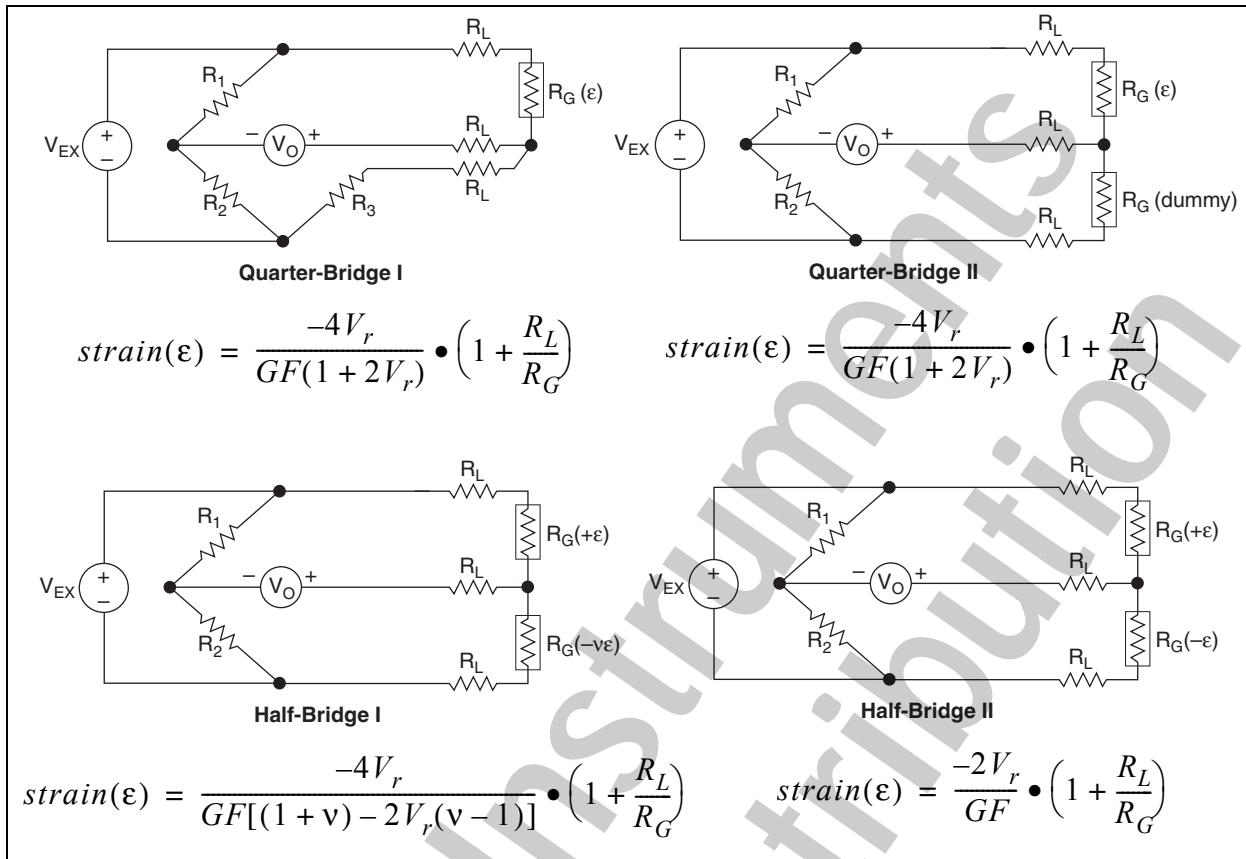
$$V_r = \frac{V_{O(\text{strained})} - V_{O(\text{unstrained})}}{V_{EX}}$$

where $V_{O(\text{strained})}$ is the measured output when strained, and $V_{O(\text{unstrained})}$ is the initial, unstrained output voltage. V_{EX} is the excitation voltage.

Also, the designation $(+\epsilon)$ and $(-\epsilon)$ indicates active strain gages mounted in tension and compression, respectively. The designation $(-\nu\epsilon)$ indicates that the strain gage is mounted in the transversal direction, so that its resistance change is primarily due to the Poisson's Strain, whose magnitude is given as $-\nu\epsilon$.

Other nomenclature used in the equations include:

- R_G = nominal resistance value of strain gage
- GF = gage factor of strain gage
- R_L = lead resistance



Pressure

Pressure is defined as force per unit area that a fluid exerts on its surroundings. Because of the great variety of conditions, ranges, and materials for which pressure must be measured, there are many different types of pressure sensor designs. Often pressure can be converted to some intermediate form, such as displacement. The sensor then converts this displacement into an electrical output such as voltage or current.

Most pressure sensors use strain gages. Strain gages, mounted on a diaphragm where pressure is applied, measure the deformation of the diaphragm that is proportional to the pressure.

Load

Load is the amount of force and weight exerted on a surface or structure. A load cell is a sensor that converts mechanical force into electrical signals. There are many different types of load cells that operate in different ways. One common type of load cell is the strain gage load cell. As the name implies, strain gage load cells use an array of strain gages to measure the deformation of a structural member and convert it into an electrical signal.

Torque

Many torque sensors are composed of strain gages that are affixed to a torsion bar. As the bar turns, the gages respond to the bar's shear stress, which is proportional to the torque.

Signal Conditioning for Pressure, Load, and Torque

Because the most common method for measuring load, pressure, and torque is to employ a full-bridge strain gage-based sensor, the signal conditioning for these sensors is the same as the signal conditioning listed for strain gages.

F. Sound and Vibration Measurements

Vibration occurs when a mass oscillates mechanically about an equilibrium point. A common example of a vibrating mechanical system is a spring-mass-damper system. Vibrations also occur in surfaces, such as the wing of an airplane, or a gong. In many cases, vibrations are unwanted because they waste energy and cause fatigue stress and noise, and systems are usually designed to minimize these types of vibrations. Meanwhile, vibrating structures generate pressure waves, or sound, which can be desirable in the case of musical instruments. A tuning fork is a great example of a vibration that creates sound.

Sound and vibration analysis varies depending on the application. Audio measurements are often concerned with distortion, and particularly the distortion at different octaves. Acoustic testing often is involved in measurements of noise regarding products such as cars, or environmental noise inside of a room. Machine monitoring is concerned with maintaining machine lifetime by measuring the vibration and then performing frequency analysis.

In general, all areas of sound and vibration analysis are concerned with performing some sort of frequency analysis upon the acquired signals. Advanced time domain analysis can be performed as well to further identify characteristics of the signal.

Sound and Vibration Sensors

Accelerometers measure acceleration and vibration, and microphones measure sound pressure level.

Accelerometers

An accelerometer, a sensor that represents acceleration as a voltage, comes in two axial types. The most common accelerometer measures acceleration along only a single axis. This type is often used to measure mechanical vibration levels. The second type is a tri-axial accelerometer. This accelerometer can create a 3D vector of acceleration in the form of orthogonal components. Use this type when you need to determine the type of vibration—lateral, transverse, rotational, and so on—that a component is undergoing or the direction of acceleration of the component.

Both types of accelerometers come with either both leads insulated, or isolated, from the case or with one lead grounded to the case. Some accelerometers rely on the piezoelectric effect to generate voltage. To measure acceleration with this type of sensor, the sensor must be connected to a charge-sensitive amplifier.

Other accelerometers have a charge-sensitive amplifier built inside them. This amplifier accepts a constant current source and varies its impedance with respect to a varying charge on the piezoelectric crystal. You can see this change in impedance as a change in voltage across the inputs of the accelerometer. Thus, the accelerometer uses only two wires per axis for both sensor excitation, or current, and signal output, or voltage. The instrumentation for this type of accelerometer consists of a constant current source and an instrumentation, or differential, amplifier. The current source provides the excitation for the built-in amplifier of the sensor, while the instrumentation amplifier measures the voltage potential across the leads of the sensor.

When choosing an accelerometer, pay attention to the most critical parameters. If the sensor must operate in extreme temperatures, you are limited to a sensor that relies on the piezoelectric effect to generate voltage. If the environment is very noisy, a sensor with a charge-sensitive amplifier built in might be the only usable choice.

To reduce errors when using an accelerometer, consider these factors:

- If the sensor is DC coupled, the DC offset of the accelerometer can drift with both temperature and age. This applies to both types of sensors because charge-sensitive amplifiers are prone to drift. AC coupling the output of the amplifier can minimize the drift in the system.
- Motors, transformers, and other industrial equipment can induce noise currents in the sensor cables. These currents can be an especially large source of noise with sensor systems that rely on the piezoelectric effect to generate voltage. Carefully routing sensor cables can minimize the noise in the cables.
- Accelerometers might have ground loops. Some accelerometers have their cases tied to a sense wire, while others are completely isolated from their cases. If you use a case-grounded sensor in a system with a grounded input amplifier, you set up a large ground loop, creating a source of noise.

Microphones

A microphone is a transducer that converts acoustical waves into electrical signals. The most common instrumentation microphone, a condenser microphone, uses a capacitive sensing element.

A condenser microphone incorporates a stretched metal diaphragm that forms one plate of a capacitor. A metal disk placed close to the diaphragm acts as a backplate. When a sound field excites the diaphragm, the capacitance between the two plates varies according to the variation in the sound pressure. A stable DC voltage is applied to the plates through a high resistance to keep electrical charges on the plate. The change in the capacitance generates an AC output proportional to the acoustic pressure (AP). Figure 7-22 shows a condenser microphone.

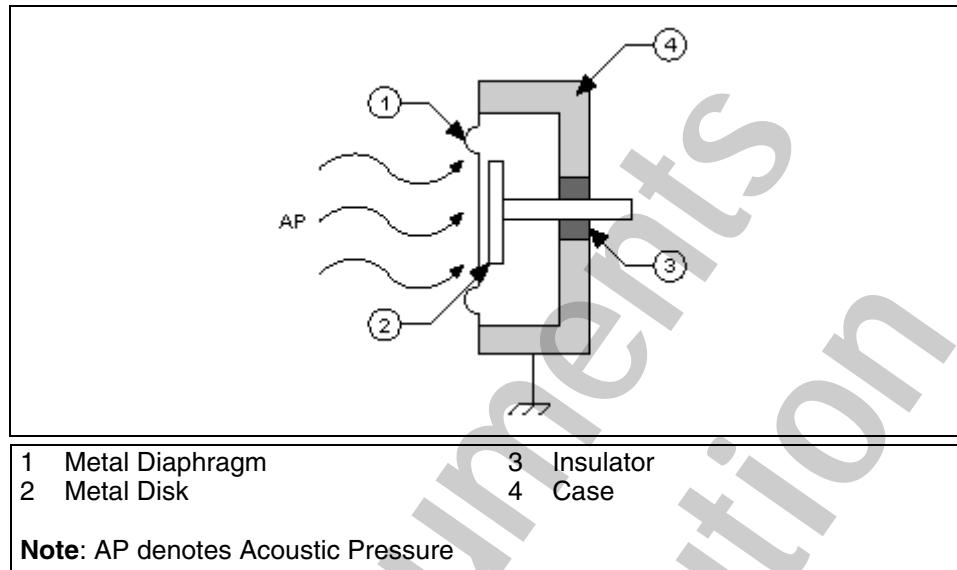


Figure 7-22. Condenser Microphone

An instrumentation microphone usually consists of a microphone cartridge and a pre-amplifier. Sometimes these two components are independent; sometimes the components are combined and cannot be separated.

The major characteristics of a microphone are its sensitivity, usually expressed in mV/Pa, and its frequency response. Microphones are available in different diameters. Common diameters include: 1/8 in., 1/4 in., 1/2 in., and 1 in. Each diameter offers a specific compromise in terms of sensitivity and frequency response.

To reduce errors when using a microphone, keep several factors in mind:

- For measurements in a free field (a sound field with no major nearby reflections), use a free-field microphone pointed at the source of sound.
- For measurements in a diffuse field, such as inside in a highly reverberant room, where sound is coming from all directions, use a random incidence microphone.
- For measurements when the microphone is part of the surface of a room or of the object being measured, use a pressure microphone.
- For outdoor measurements, fit the microphone with suitable protection against the environment. This may include windscreens, rain caps, and built-in heaters to prevent condensation.
- To prevent vibrations from influencing the measurement, you might need to shock mount the microphone. Check the microphone specifications for vibration sensitivity.

- For reproducible measurements, make sure the microphone is mounted firmly and at a precisely reproducible location, both compared to the unit being tested and to the environment.
- Always calibrate the entire measurement chain, including the microphone, before starting the measurement. For highly critical measurements, as an extra precaution, you may want to perform a new calibration immediately after the measurements are completed to make sure the system is still within tolerances.

IEPE Sensors

Many accelerometers and microphones are based on the principle of piezoelectric generation. The piezoelectric effect denotes the ability of ceramic or quartz crystals to generate electric potential upon experiencing compressive stresses. These mechanical stresses are triggered by forces such as acceleration, strain, or pressure. In the case of microphones, acoustic pressure waves cause a diaphragm, or thin membrane, to vibrate and transfer stresses into the surrounding piezoelectric crystals. Accelerometers, on the other hand, contain a seismic mass that directly applies forces to the surrounding crystals in response to shock and vibrations. The voltage generated is proportional to the internal stresses in the crystals.

A particular class of piezoelectric sensors, known by the term integral electronic piezoelectric (IEPE), incorporates an amplifier in its design next to the piezoelectric crystals. Because the charge produced by a piezoelectric transducer is very small, the electrical signal produced by the sensor is susceptible to noise, and you must use sensitive electronics to amplify and condition the signal and reduce the output impedance. IEPE therefore makes the logical step of integrating the sensitive electronics as close as possible to the transducer to ensure better noise immunity and more convenient packaging. A typical IEPE sensor is powered by an external constant current source and modulates its output voltage with respect to the varying charge on the piezoelectric crystal. The IEPE sensor uses only one or two wires for both sensor excitation (current) and signal output (voltage).

Sound and Vibration Signal Conditioning

The signal conditioning circuitry for measuring sound and vibration is fairly straightforward. A typical system for measuring acceleration or sound pressure level includes the following components:

- Sensor
- Current source to excite the sensor
- Proper grounding to eliminate noise pick-up
- AC coupling to remove DC offsets in the system

- An instrumentation amplifier to boost the sensor's signal level
- A lowpass filter to reduce noise and prevent aliasing in the data acquisition system
- Simultaneous sample and hold circuitry to keep multiple signals properly timed with respect to each other

Sound and vibration measurements are highly susceptible to noise. You can reduce this effect, however, by properly grounding the system. You can avoid improper grounding resulting from ground loops or floating nodes by ensuring that either the signal conditioning input or the sensor is grounded, but not both. If the sensor is grounded, you must connect it differentially. If the sensor is floating, you should connect the signal conditioning system's inverting input to ground.

The signal acquired from the sensor consists of both DC and AC components, where the DC portion offsets the AC portion from zero. AC coupling removes the DC offset in the system by means of a capacitor in series with the signal. An AC-coupled sensor system eliminates the long-term DC drift that sensors have due to age and temperature effect, dramatically increasing the resolution and the usable dynamic range of the system.

For accurate measurements, the sampling rate of the system should be at least twice the frequency of the signals being acquired. To be sure that you are sampling the correct range of frequencies, add a lowpass filter before the sampler and the analog-to-digital converter. This ensures that you attenuate higher-frequency noise and that these aliasing components above the sampling rate do not distort the measurement.

Sound and Vibration Analysis Examples

Sound and Vibration Measurement Suite contains functions and graphs for analysis and visualization.

- Sound Quality, Spectral Analysis, Zoom Power Spectrum, Frequency Response
- ANSI and IEC compliant full and fractional octave analysis
- Order analysis tracking and extraction including tachometer processing
- Waterfall, cascade, shaft centerline, orbit, bode, and polar plots
- Universal File Format (UFF58) file I/O support

Summary

- Signal conditioning can be accomplished using cDAQ, PXI SC DAQ devices, SCXI, or SCC.
- Signal conditioning can provide amplification, attenuation, filtering, and isolation for voltage measurements.
- Signal conditioning provides CJC compensation, filtering, isolation, linearization, and amplification for thermocouples.
- Signal conditioning provides excitation, amplification, and filtering for RTDs and thermistors.
- Signal conditioning provides bridge completion, excitation, offset nulling, shunt calibration, remote sensing, and filtering for strain gages.
- Signal conditioning provides excitation, AC coupling, flexible gains, and filtering for accelerometers and microphones.

National Instruments
Not for Distribution

Self-Review: Quiz

1. Name 5 types of signal conditioning.

2. The arrangement of the strain gages is inconsequential.
 - a. True
 - b. False

3. Offset nulling is never necessary because most Wheatstone bridge measurements output exactly 0 V when no strain is applied.
 - a. True
 - b. False

4. Which of the following types of signal conditioning can apply to thermocouple measurements?
 - a. CJC compensation
 - b. Amplification
 - c. Excitation
 - d. Filtering
 - e. Isolation

National Instruments
Not for Distribution

Self-Review: Quiz Answers

1. Name 5 types of signal conditioning.
 - **CJC Compensation**
 - **Bridge Completion**
 - **Offset Nulling**
 - **Amplification**
 - **Filtering**
 - **Isolation, etc.**

2. The arrangement of the strain gages is inconsequential.
 - a. True
 - b. **False**

3. Offset nulling is never necessary because most Wheatstone bridge measurements output exactly 0 V when no strain is applied.
 - a. True
 - b. **False**

4. Which of the following types of signal conditioning can apply to thermocouple measurements?
 - a. **CJC compensation**
 - b. **Amplification**
 - c. Excitation
 - d. **Filtering**
 - e. **Isolation**

Notes

National Instruments
Not for Distribution

Synchronization

This lesson describes synchronization of tasks on a single device, on multiple devices, and with counters.

Topics

- A. Synchronizing Measurements
- B. Single Device Synchronization
- C. Multiple Device Synchronization
- D. Counters and Synchronization

National Instruments
Not for Distribution

A. Synchronizing Measurements

Many applications require making more than one type of measurement at the same time. Simultaneous measurements involve different operations happening at the same time, such as acquiring data on input channels while generating data on output channels. However, these operations are not necessarily correlated to one another. For example, you can start an input operation at the same time you start an output operation but from there, each operation can run independently at its own rate. You must synchronize measurements when measurements must be taken at the same time.

When synchronizing measurements (like making 100 temperature and speed measurements), you need to start all the measurements at the same time as well as share a common clock for latching data among all measurements.

For example, in control-loop applications, you need to make multiple measurements at the beginning of the loop, perform a calculation based on the new measurements, and do outputs based on this calculation.

Control-loop applications require that you start all the measurements at the same time and synchronize them via a common clock signal. Similarly, you would do the same to correlate measurements. For example if you want to make a plot of speed and brake-pad temperature versus time, you will need to first synchronize both the speed and temperature measurements.

Simultaneous Measurements

Sometimes, when attempting to create synchronized measurements, developers unintentionally create simultaneous measurements which are not truly synchronized.

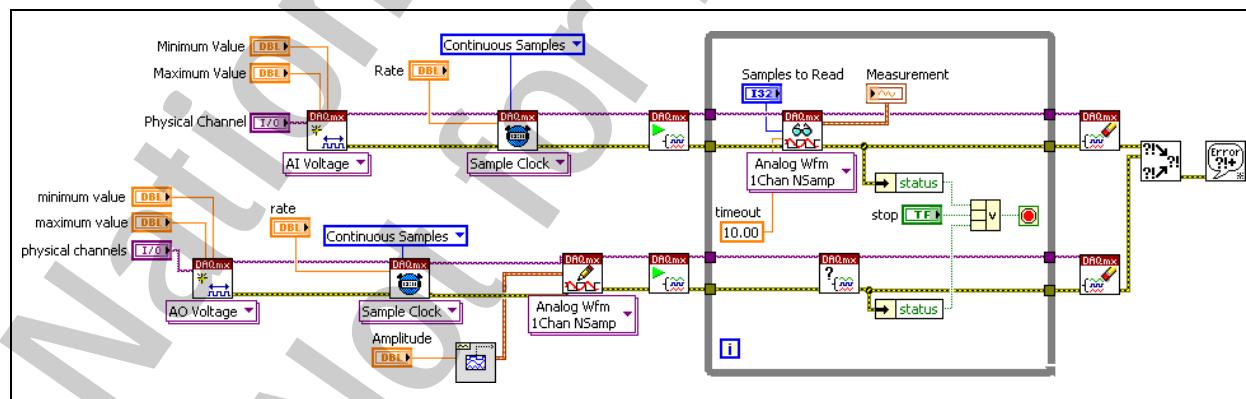


Figure 8-1. Simultaneous AI and AO VI

In Figure 8-1, both tasks in the application are started in parallel but cannot truly be considered to be synchronized. Because the different DAQmx Start Task VIs are called at different times by the software, one measurement could start as much as 50 to 200 milliseconds after the other.

When looking at the timing diagram for this VI it might look something like Figure 8-2.

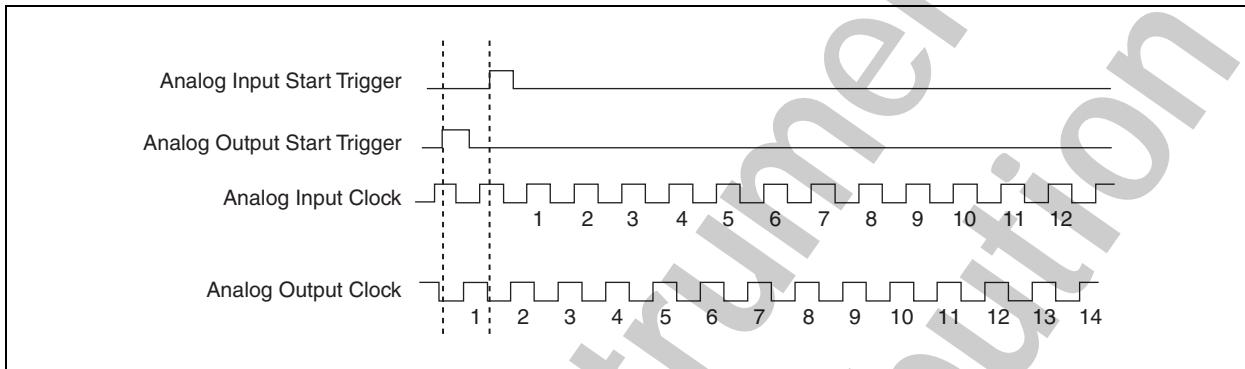


Figure 8-2. Simultaneous Measurements

Because the start trigger is the result of the DAQmx Start Task VI, one of the tasks starts before the other and leads to measurements that are not synchronized. In this example, the analog output leads the analog input. Also in this application, the tasks are created from different timebases, so the measurements are also out of phase.

While simultaneous measurements may have enough accuracy to suffice for some systems, many applications require true synchronization between tasks and boards.

Synchronization Rules

Synchronized operations are created by routing timing and control signals. Synchronization can be within a single device or on multiple devices.

When dealing with Multifunction DAQ devices there are two rules for synchronization.

- Share a Master Timebase (or Reference Clock) and a Start Trigger.

All synchronized devices are programmed to use the same signal (usually the 20MHzTimebase from one of the devices) as their Master Timebase. More generally, one device can be queried for its Master Timebase source and that terminal can be set as the source of the Master Timebase for the other synchronized devices. All synchronized devices are programmed to use the same <task>/StartTrigger terminal as the source of their Start Trigger. You can always share a Start Trigger even if you have not explicitly configured one. There are two advantages of

using this method. The devices need not sample at the same rate, nor acquire the same amount of data. This method also works for synchronizing analog output signals and some counter applications. The disadvantages of using this method are two signals need to be routed using two RTSI or PXI trigger lines. An attribute/property must be set to designate the 20 MHz Timebase of another device as the Master Timebase of a synchronized device.

In Figure 8-3 both sample clocks are based off of the same master timebase. All the tasks have the same start trigger to begin generating their sample clock. The sample clocks are able to run at different rates because their timing is not dependent upon one another.

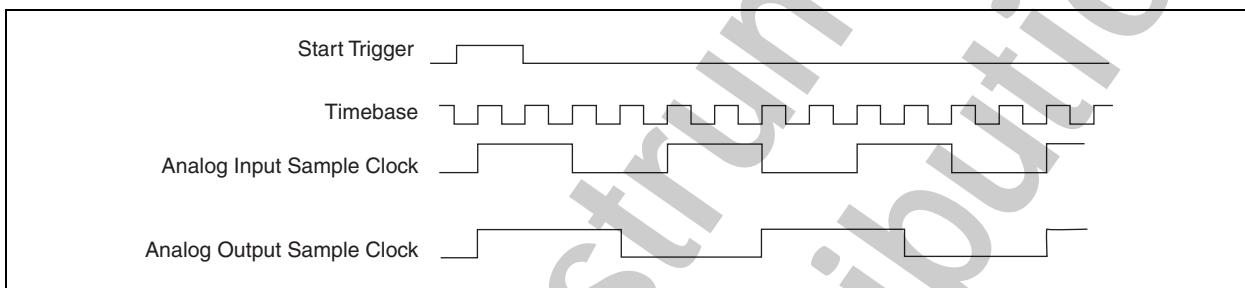


Figure 8-3. Synchronized Measurements

- Share a Sample Clock.

All synchronized devices are programmed to use the <task>/SampleClock terminal from one of the devices as their Sample Clock. The advantage of using this method is only a single signal is routed using but one RTSI line or PXI Trigger line. The disadvantages are that all devices must sample at the same rate and can acquire no more data than the device that is sourcing the Sample Clock.

Synchronization Rules within DAQmx

- Create one master task.
 - Configure the master task but do not use the DAQmx Start Task VI to start the task immediately.
 - Only start the master task after all of the slave tasks have already been configured and started.
- Create as many slave tasks as necessary.
 - Configure slave tasks before starting master task.
 - Verify that the slave is dependent upon the master task for a start trigger or a sample clock.
 - If synchronizing multiple devices, verify that the master timebase for the slave task comes from the same master timebase used by the master task.

Sources of Error

There are several sources of error when synchronizing measurements:

- **Jitter**—is small variations in the period of the clock (from sample to sample). It shows up as noise in the digitized signal and affects higher-frequency signals more. Each component added to the clock's path adds additional jitter. You can control jitter, but not eliminate it, by using an accurate clock source.
- **Stability**—describes how well the clock frequency resists fluctuations. Factors that can cause the frequency to fluctuate include variations in temperature, time (aging), supply voltage, shock, vibration, and capacitive load that the clock must drive. Temperature is often the dominant factor that affects crystal oscillator stability.

Some oscillators are housed inside small ovens with controlled temperature to provide stability that can be orders of magnitude better than with other techniques. These oscillators are known as oven controlled crystal oscillators (OCXOs). For example, the NI 6608 contains an OCXO.

- **Accuracy**—describes how well the actual frequency of the clock matches the specified frequency. An oscillator generates a clock. However, an oscillator never generates a perfect frequency. The accuracy of the oscillator-generated clock is affected by the quality of the crystal and the oscillator's assembly.

You can describe timing errors in several different ways. Some common units of timing error are parts per million (ppm) and parts per billion (ppb). Parts per million gives you a fractional value of error. For example, to find the error in Hertz of an 80 MHz oscillator with 5 ppm error, you multiply the frequency of the oscillator—80,000,000—by 5 divided by 1,000,000 or [80,000,000 Hz (5 Hz/1,000,000 Hz) = 400 Hz].

From this equation, you see that the oscillator can be off by as much as 400 Hz. Therefore, the actual frequency of the oscillator can be anywhere between 79,999,600 Hz and 80,000,400 Hz.

- **Skew**—is a propagation delay that is caused when a signal arrives at two places at different times. For instance, a signal is sent by a controlling device at time T0. A receiving device A acts upon the signal at time T1. A receiving device B acts upon the signal at time T2. If T1 is not equal to T2, the difference between T1 and T2 is the skew. The distance between devices and the cabling between your devices and signal paths within the devices themselves all affect signal arrival times.

B. Single Device Synchronization

Single device synchronization is the most common type of synchronization. Fortunately, single device synchronization also proves to be the easiest because every sample clock on a single device is inherently created from the same master timebase or reference clock. Therefore, the primary concern is that a task either share a trigger if different rates for different tasks are desired. Or, for synchronous measurements, share a sample clock.

Simultaneously Started Measurements

To simultaneously start an analog input and output operation, trigger the operation through a hardware trigger or a software trigger. For a hardware triggered operation, both the analog input and output operation trigger off of the same PFI or RTSI pin. RTSI stands for Real-Time System Integration bus, a dedicated high-speed digital bus designed to facilitate system integration by low-level, high-speed real-time communication between National Instruments devices. Refer to the *Counters and Synchronization* section for more information about RTSI.

Shared Software Trigger

For a software triggered operation, the analog input is triggered off of an external PFI or RTSI pin, and the analog output triggers off the internal AI Start Trigger signal. AI Start Trigger is an internal signal that is directly connected to both the analog input and analog output subsystems. The software triggered method is slightly more accurate than the hardware triggered method because the external signal has only to propagate through one main path to reach both subsystems. However, this delay is almost always insignificant at the speeds at which multifunction I/O DAQ devices operate. The block diagram in Figure 8-4 illustrates this technique.

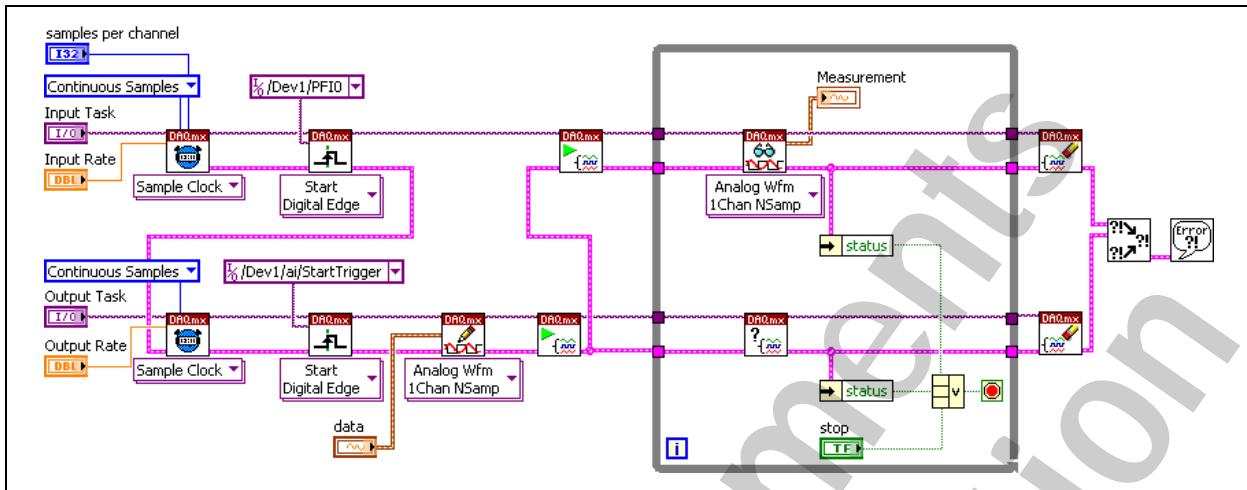


Figure 8-4. Hardware and Software Triggered Start VI

The example in Figure 8-4 is configured so the input operation responds to a digital start trigger on PFI0, configured with the DAQmx Trigger VI. The output operation uses the DAQmx Trigger VI to trigger off of the internal AI Start Trigger. Notice that the output operation must be started before the input operation to ensure that the input does not start and send the internal AI Start Trigger before the output operation is set up to receive the start signal.

Another example of a simultaneous start is based on a software trigger that occurs when the analog input is started with a software call instead of a hardware trigger. The analog output is still called using the internal AI Start Trigger signal. Figure 8-5 illustrates this fully software triggered example. When the operations are simultaneously started, they are not necessarily synchronized. The operations can be set to acquire and generate data independently of another at different rates.

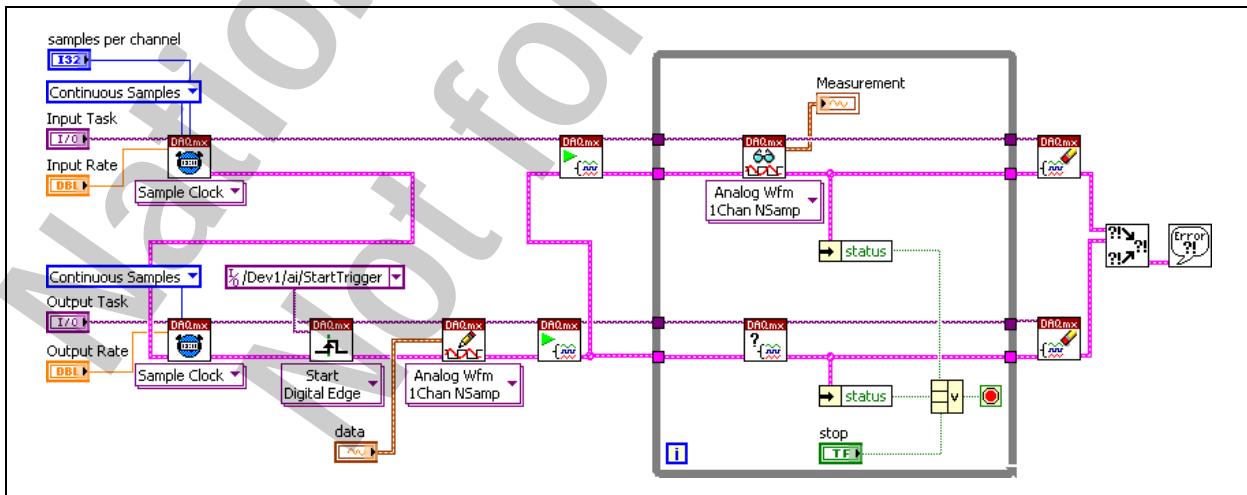


Figure 8-5. Software-Triggered Start VI

Shared Sample Clock

When you want to fully synchronize the analog input and output operations such that each input sample per channel occurs at the same time each output sample per channel is updated, the operations must use a common clock source. Like simultaneous starting operations, there are two main methods to synchronize.

The first method uses the internal AI Start Trigger to trigger the analog output to start at the same time as the input acquisition. Unlike the simultaneous start example, we now set the input and output sample clocks to run at the same rate, causing the operations to be synchronized on a single device. Both the internal input and output sample clocks are derived from the onboard timebase of the device. Since both clocks are derived from the same timebase and started at the same time, the sample clocks will be synchronized. Because the analog input and output subsystems each have their own divide down circuitry to derive their own sample clocks, there could be a small phase difference between the two sample clocks. However, these differences are insignificant with the rates at which many DAQ devices run.

The second method to synchronize analog input and output is to have both operations use either the analog input or analog output sample clock.

Figure 8-6 demonstrates how to synchronize analog input and analog output by sharing the analog output sample clock.

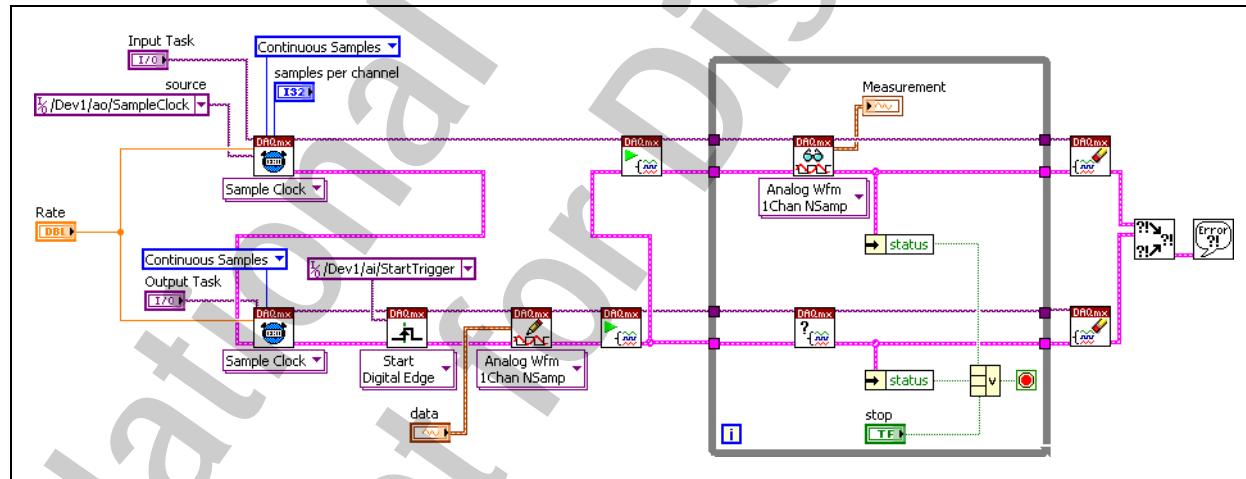


Figure 8-6. Synchronization with Software Trigger VI

To perform a synchronized analog input and output operation with a hardware start trigger, use one of the synchronization methods previously discussed and simply add a hardware start trigger to the master operation, which in all of the previous examples has been the analog input operation.

C. Multiple Device Synchronization

Board synchronization correlates measurements between multiple boards. Numerous applications are aided by a board's ability to synchronize itself. For example, by sharing timing signals, analog measurements can be taken in conjunction with counters by routing the sample clock to the counter to tell it when to latch a value.

Synchronization is especially important in high speed applications when a high channel count is needed. Either the board needs to synchronize to an external device, and/or a master/slave relationship is established where one board is controlling the timing for all the boards in the measurement system.

There are several methods available for synchronization. The methods include using an external clock, using the internal timing bus, or using a Phase-Lock-Loop (PLL) circuit. The PLL circuit method is beyond the scope of this course. The effects of delay and jitter are essential considerations when choosing a method for board synchronization.

External Signal Connections

One method of synchronization is using an external clock that allows the multiple boards to synchronize to this external source. Using this method, the board timing of the boards receiving the external source take on the accuracy and stability of the external clock source.

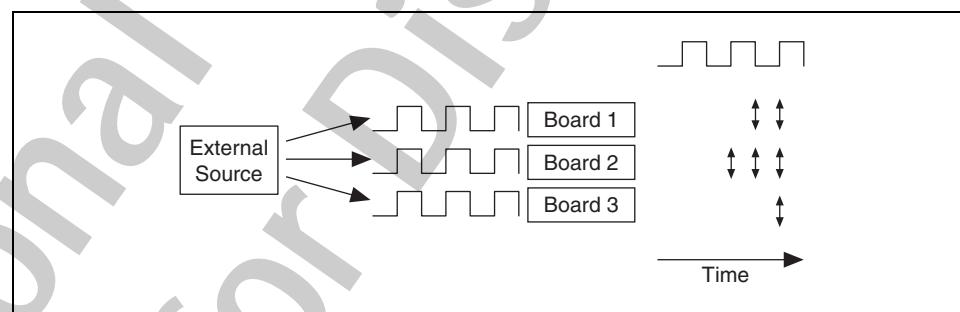


Figure 8-7. External Signal Connections

Using this method of board synchronization results in the synchronization error from three sources. These sources include the length of the signal path, the individual board timing, and jitter. The combination of these factors causes each board to see and respond to the external signal at different times.

For high frequencies, several factors make this method less than ideal. The clock signal begins to deteriorate at frequencies around 5–10 MHz depending on the cabling used to wire the external clock source to the boards. In addition, there is a transmission latency that introduces pronounced phase delays at high speeds depending upon the length of the

signal path. These delays depend upon the capacitive, inductive, and resistive properties of the cabling. All timing sources introduce jitter into the system that could become significant when attempting to synchronize the boards.

Another issue with this method occurs when using a start trigger to simultaneously begin acquisition on all of the boards in a measurement system. Usually, each board triggers within one to two ticks of the board clock upon receiving the start trigger. This can cause phase delays between the boards.

To eliminate phase delays from transmission latencies and triggering, use the RTSI bus to transfer signals. The RTSI bus improves the limitations of using external wiring.

Real-Time System Integration (RTSI) Bus

RTSI is the National Instruments timing bus that connects DAQ boards directly, by means of connectors on top of the boards, for precise synchronization of functions. RTSI is an internal timing bus used to share and exchange timing and control signals between multiple boards through the use of parallel digital lines. The connector is typically located on the top of a DAQ device. The advantage of RTSI is that it allows you to programmatically pass digital signals to use for triggering, clocking, and other tasks between multiple DAQ devices or PXI modules. For synchronization applications, the RTSI bus can be used to allow one board to generate the clock and trigger signal and pass those signals over the bus.

To register your RTSI cable in MAX, right-click **NI-DAQmx Devices**, select **Create New NI-DAQmx Device** from the shortcut menu, and select **RTSI Cable**. A RTSI cable should appear under your NI-DAQmx devices. Right-click the RTSI cable and add the devices it is connected to.

Most NI DAQ devices support RTSI. In plug-in DAQ devices, such as an M Series or X Series PCI device, the actual RTSI connection is accomplished with a special RTSI cable that is manually plugged into the RTSI pins on each device. The RTSI bus interface on a PCI DAQ device is an internal 34-pin connector where signals are shared through a ribbon cable inside the PC enclosure. RTSI cables are available for chaining two, three, four, or five devices together. RTSI functionality varies depending on device type, so always check your device documentation before beginning programming with RTSI.

The RTSI bus has eight lines available to users for sharing timing and triggering signals. Pins 0 to 6 are available for user signals, but pin 7, the RTSI Clock, is reserved for passing device clock signals between devices. Refer to ni.com/products for more information about the RTSI cable.

If using a PXI module, the RTSI bus is built into the PXI chassis as the PXI Trigger Bus on the J2 backplane connector. The PXI Trigger Bus is standardized by the PXI Systems Alliance, now supported by a range of industry leaders. Thus, every PXI DAQ module you insert into a PXI chassis has the same built-in timing connections to every other PXI DAQ module in that chassis. No additional cabling is required. Refer to www.pxisa.org for more information about the PXI Systems Alliance.

Capable of passing timing signals of up to 20 MHz before signal deterioration, RTSI succeeds at high speed applications. However, phase delays due to triggering and transmission latencies becomes more pronounced at high frequencies. To adjust for these issues, the use of a Phase-Locked-Loop (PLL) circuit allows for true multiple board synchronization. PLL is beyond the scope of this course.

Programming with RTSI

NI-DAQmx manages the majority of signal routing through the RTSI bus. However, you are still responsible for telling NI-DAQmx how the devices are connected to one another through internal buses. For PCI systems, you must register the RTSI cable in MAX. For PXI systems, you must identify the type of PXI chassis in use.



Note To bypass NI-DAQmx management of routing signals, you can explicitly route signals to PFI or RTSI lines using the DAQmx Export Signal VI.

If two different applications try to drive the same RTSI line, damage to the board can occur. Since NI-DAQmx manages the routing of RTSI lines for you, it is not always known which RTSI lines are available.

To prevent the problem of double-driving a RTSI line, you can reserve certain RTSI lines in MAX to prevent NI-DAQmx from using that line when automatically routing signals. For PCI devices, right-click the RTSI cable that connects your devices and select Properties. The RTSI Cable Properties window appears, as shown Figure 8-8.



Figure 8-8. RTSI Cable Properties Dialog

Select which RTSI lines should not be used by NI-DAQmx. For PXI devices, highlight the chassis under the PXI system (the chassis must be identified first), and select the **Triggers** tab.

With a multifunction DAQ device, the following signals can be routed over RTSI to be shared between multiple devices.

- AI Start or Reference Trigger
- AI Convert Clock
- AI Sample Clock
- AO Sample Clock
- AO Reference Trigger
- GPCTR0 Source, Gate, or Output

Multi-Device Use Cases

Much like single devices, the common use cases for simultaneous operations involving multiple devices are to simultaneously start operations on multiple devices by sharing a start trigger or to fully synchronize multiple device operations. A third use case is to start operations synchronized across multiple devices with a hardware start trigger. When synchronizing measurements, one analog input sample clock can be shared across all devices involved to synchronize measurements. Alternatively, one of the board clocks of the device can be shared to synchronize the board clocks of all the devices involved.

D. Counters and Synchronization

Counters have better flexibility than the clocks for synchronizing analog or digital inputs and outputs. That increased flexibility lends itself to using the counters in tandem with analog or digital tasks to create more adaptable measurement systems.

By using the counters as the clock sources for analog and digital tasks it allows the developer to create:

- Retriggerable input tasks
- Variable rate input and output tasks
- Event-triggered acquisition

Similarly, the sample clock for the analog or digital measurements can be used with counters, such as using the sample clock as the gate for a buffered counter measurement to correlate all of the different data samples.

Analog Input with the External Clock Generated by a Counter

Use a counter to generate a pulse train, either finite or continuous, to serve as the sample clock for analog input or output operations. The frequency of a pulse train being generated by a counter can be changed instantly. This allows you to correspondingly change the rate of an analog input or output operation. Configure a counter for continuous pulse train generation. Set the sample clock source for the analog input acquisition to the internal output of the counter.



Note The frequency of the AO Sample Clock can be changed instantly. Therefore, you do not need to use a counter to change the rate of an analog output generation at runtime.

Retriggerable Analog Operations

STC3-Based Devices

In certain STC3-based devices, such as X Series, analog input and output operations are hardware retriggerable by using the Retriggerable property in the DAQmx Trigger Property Node.

STC2-Based Devices

However in STC2-based devices, analog input and output operations are not hardware retriggerable, and only counters are hardware retriggerable. To simulate a retriggerable analog operation for STC2-based devices in software, set the task to respond to a trigger and then reconfigure the operation in software after it completes. Reconfiguring the task allows it to respond to the next trigger. However, the time it takes to reconfigure the

operation in software is completely system dependent and could cause the device to miss the next trigger if it does not reconfigure quickly enough.

Since counter operations are retriggerable, they can be used to implement retriggerable analog input and output operations. Two counters are used to create a retriggerable finite pulse train which is used as the analog sample clock. The first counter is configured to create a continuous pulse train and the second counter is set up to perform retriggerable finite pulse generation. This pulse gates or pauses the continuous pulse train generation of the first counter. Therefore, when the trigger occurs, the second counter creates a finite pulse which allows the continuous pulses to be generated on the first counter as long as the finite pulse is still high.

Figure 8-9 demonstrates how to perform a retriggerable analog input acquisition using counters. You do not have to explicitly set one counter for continuous pulse generation and another for a retriggerable finite pulse generation. NI-DAQmx sets this up for you when your VI configures one counter for finite pulse train generation and uses a property node to set it to retriggerable.

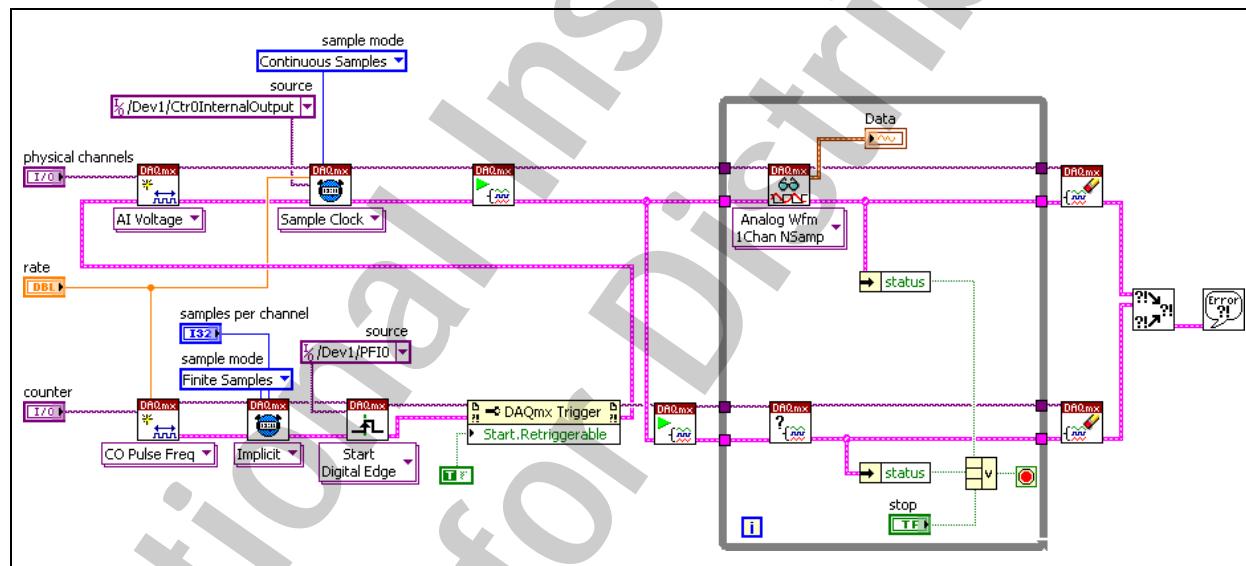


Figure 8-9. Retriggerable Analog Input VI for STC2-Based Devices

Event Triggering

There are cases in which you may want to skip a certain number of trigger pulses before starting an acquisition or generation. This type of trigger is referred to as an “event trigger” since you are triggering on the N th ($2 < N <$ Terminal Count) trigger event. Such an application can be implemented using the counters on the National Instruments multifunction DAQ boards. Figure 8-10 demonstrates how to configure your DAQmx device for event triggering.

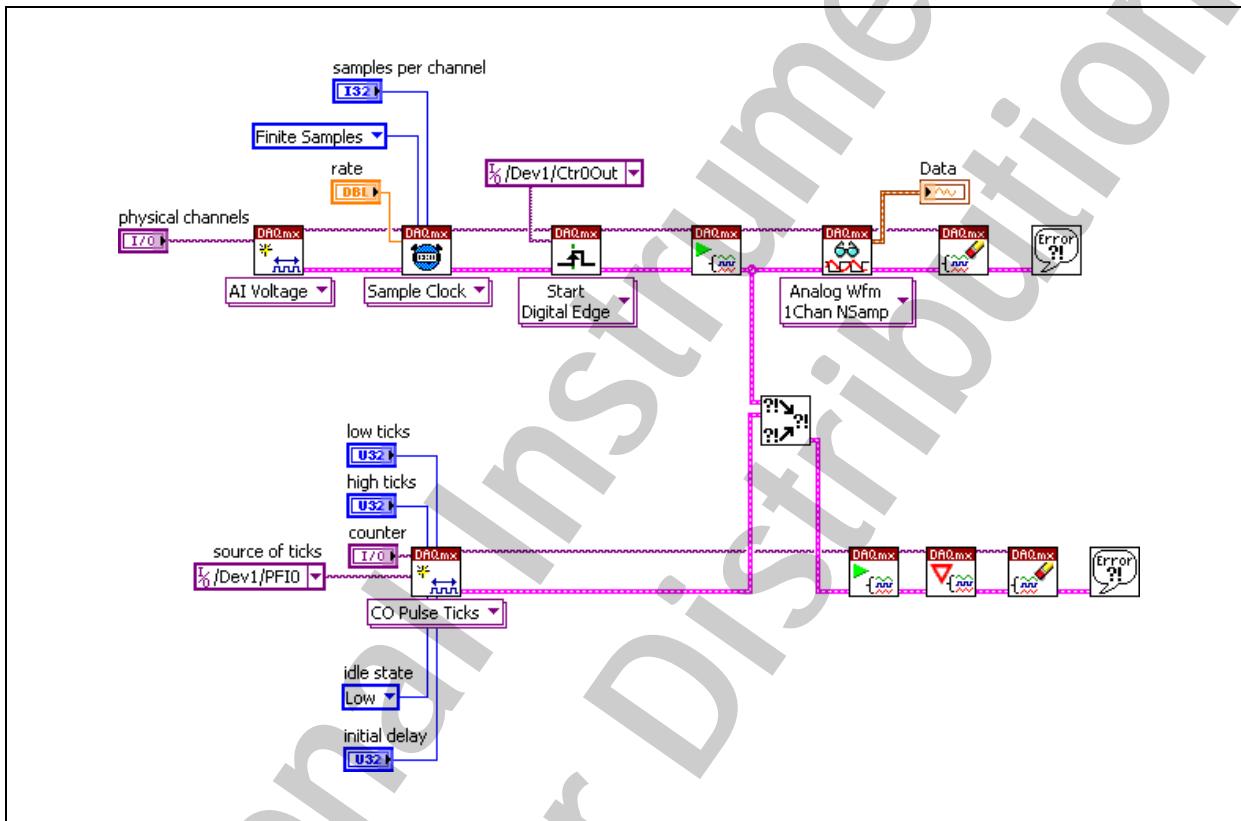


Figure 8-10. Event Triggering VI

The top row configures the sample clock and the trigger. The bottom row configures the counter to divide down the signal so that it will output the “trigger” pulse when it inputs its “Nth” pulse. Notice the instance of the **DAQmx Create Virtual Channel VI - Counter Output» Pulse Generation»Ticks**.

Pulse Generation»Ticks. The high ticks and low ticks inputs are used to determine the “Nth” pulse. The source of the ticks can be a PFI line, an internal timebase, or any other appropriate signal source.

Using Sample Clocks with Counters

STC3-Based Devices

STC3-based devices support Sample Clock timing for buffered counter tasks, so you can synchronize counter input applications performing period, frequency, pulse width or semi-period measurements in the same sense as analog input or output applications. After the counter input task begins, your device measures each consecutive sample of the input signal, but does not store it to the input buffer until an active edge of the Sample Clock occurs.

STC2-Based Devices

When using STC2-based devices, you cannot synchronize counter input applications performing period, frequency, pulse width or semi-period measurements in the same sense as analog input or output applications. These types of counter input applications cannot be programmed to make their measurements at the same time because the signals determine when the measurements are made, and there is no reason to set up multiple devices to measure the same signal. You also cannot use Start Triggers for counter input applications.

You can, however, ensure that all counters are using the same timebase for their input measurements by sharing the CI Counter Timebase signal. Program all devices to use the same signal (usually the 20 MHz Timebase from one of the devices) as their CI Counter Timebase. More generally, one device can be queried for its CI Counter Timebase source and that terminal can be set as the source of the CI Counter Timebase for the other devices.

If you are synchronizing buffered counter input applications performing edge counting, you can do so by sharing the Sample Clock. The Sample Clock must be externally supplied to one of your devices. The other synchronized devices are programmed to use this device's $CtrnGate$ signal as their Sample Clock, where n is the number of the counter.

If you are synchronizing pulse generation counter output applications, you can do so by sharing the CO Counter Timebase and Start Trigger signal. Program all devices to use the same signal (usually the 20MHzTimebase from one of the devices) as their CO Counter Timebase. More generally, one device can be queried for its CO Counter Timebase source and that terminal can be set as the source of the CO Counter Timebase for the other devices. Program all devices to use the same signal as their Digital Edge Start Trigger. This is typically the $CtrnGate$ signal from one of the devices, where n is the number of the counter.

Summary

- Create synchronized measurements by sharing a master timebase and a start trigger or by sharing a sample clock.
- With the NI-DAQmx API, create one master task and multiple slave tasks. Always start all slave tasks before starting the master task.
- Use National Instruments RTSI bus or PXI trigger lines to transfer signals between multiple PCI, PCIe, PXI, or PCIe devices.

National Instruments
Not for Distribution

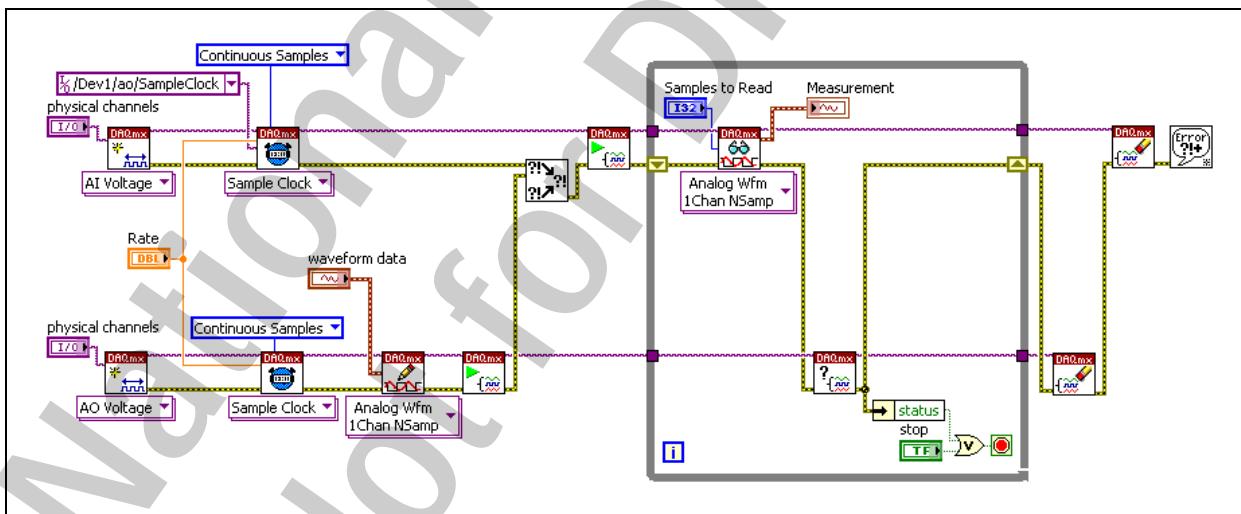
National Instruments
Not for Distribution

Self-Review: Quiz

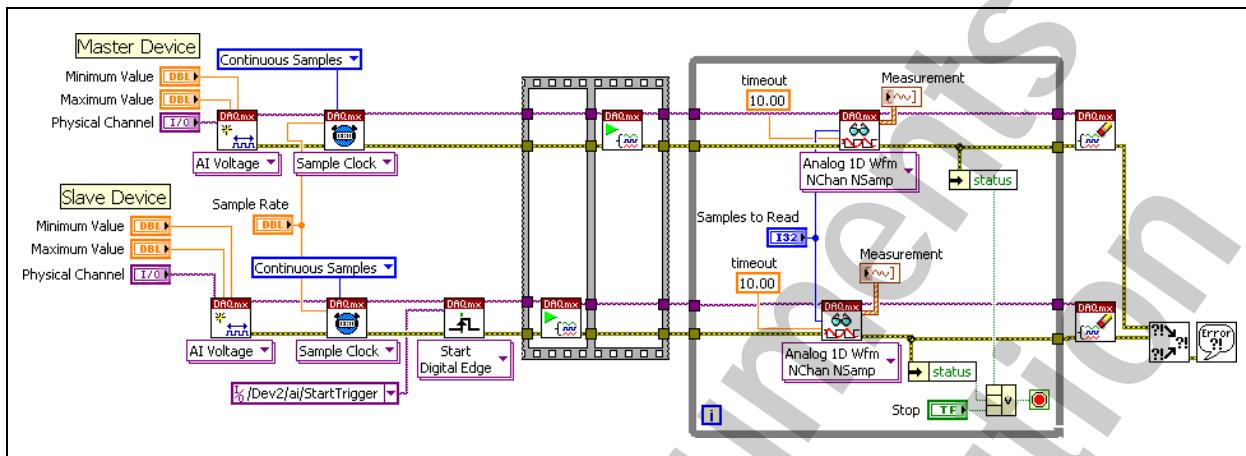
1. To simultaneously start and synchronize multiple tasks, you must share which of the following?
 - a. Master Timebase
 - b. Physical Channel
 - c. Sample Clock
 - d. Indicator
 - e. Trigger

2. To synchronize multiple boards, which of the following could be used?
 - a. GPS
 - b. RTSI Bus
 - c. External Clock
 - d. PXI Trigger Bus

3. Is this measurement synchronized? If so, why? If not, what prevents it from being synchronized?



4. Is this measurement synchronized? If so, why? If not, what prevents it from being synchronized?



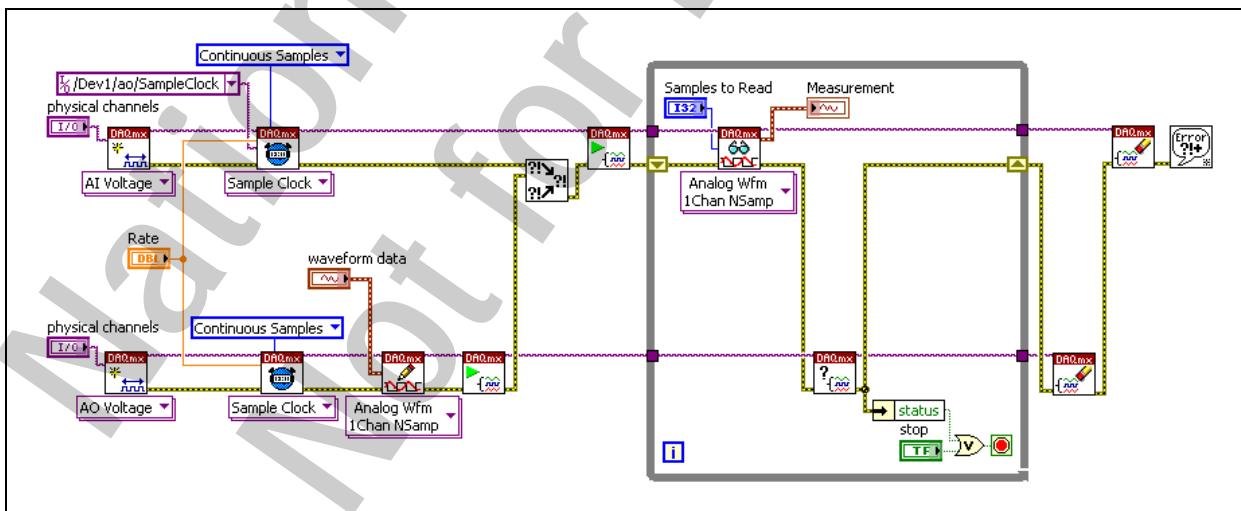
Self-Review: Quiz Answers

1. To simultaneously start and synchronize multiple tasks, you must share which of the following?
 - a. **Master Timebase**
 - b. Physical Channel
 - c. **Sample Clock**
 - d. Indicator
 - e. Trigger

2. To synchronize multiple boards, which of the following could be used?
 - a. **GPS**
 - b. **RTSI Bus**
 - c. **External Clock**
 - d. **PXI Trigger Bus**

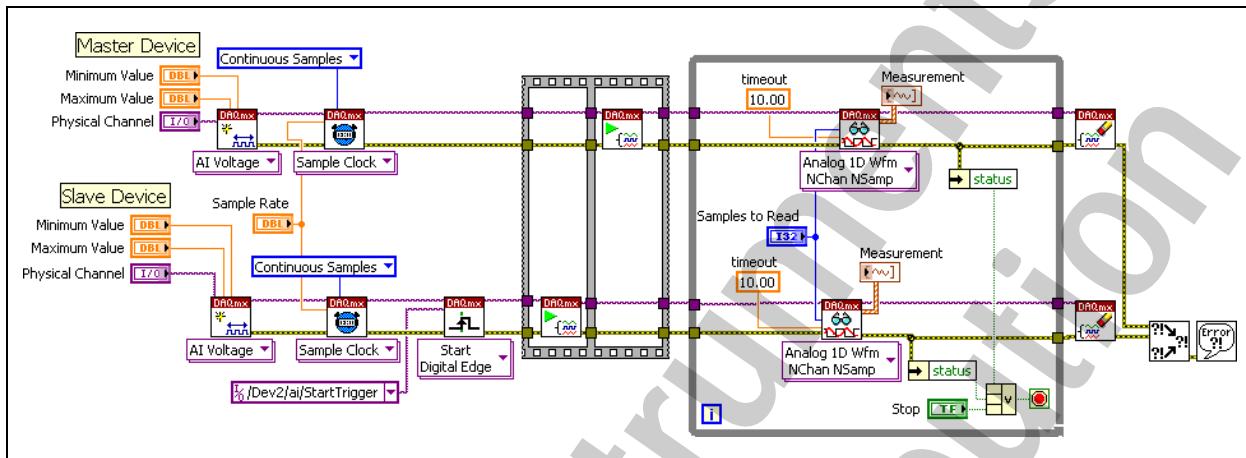
3. Is this measurement synchronized? If so, why? If not, what prevents it from being synchronized?

No, this measurement is not synchronized. The analog output begins creating a sample clock before the analog input task has started.



4. Is this measurement synchronized? If so, why? If not, what prevents it from being synchronized?

No. The measurements share a start trigger, but not a master timebase; therefore, the measurements will be out of phase.



Notes

National Instruments
Not for Distribution

Notes

National Instruments
Not for Distribution

DAQ Theory

This appendix contains the following sections of useful information for DAQ system users.

Topics

- A. Theory of Common Sensors
- B. Analog I/O Circuitry

*National Instruments
Not for Distribution*

A. Theory of Common Sensors

Selecting your sensor is one of the first steps in building a DAQ system. The sensor you choose may affect the other components in your system. For example, some sensors require external signal conditioning for excitation or amplification, and others can make use of the LabVIEW VIs that convert voltage readings into units of temperature or strain. It is important to understand how different sensors operate and to know their advantages and limitations. Below is a discussion of the most common types of sensors: thermocouples, RTDs, thermistors, and strain gages.

Thermocouples

One of the most frequently used temperature sensors is the thermocouple. Thermocouples are very rugged and inexpensive and can operate over a wide temperature range. Thermocouples can tolerate temperatures of several hundred degrees without degradation. Sensors, usually referred to as semiconductor sensors, can seldom function above 70 °C. Also, thermocouples are physically small and can rapidly track temperature changes.

A thermocouple is created whenever two dissimilar metals touch to produce a small open-circuit voltage as a function of temperature. This thermoelectric voltage is known as the Seebeck voltage (after Thomas Seebeck, who discovered it in 1821). The voltage is approximately linear for small changes in temperature, or

$$\Delta V \approx S\Delta T$$

where ΔV is the change in voltage, S is the Seebeck coefficient, and ΔT is the change in temperature. S varies with large changes in temperature, however, causing the thermocouple output voltages to be nonlinear over their operating ranges, as shown in Figure A-1. For this reason, you must use either polynomials or look-up tables to determine the voltage for any given temperature.

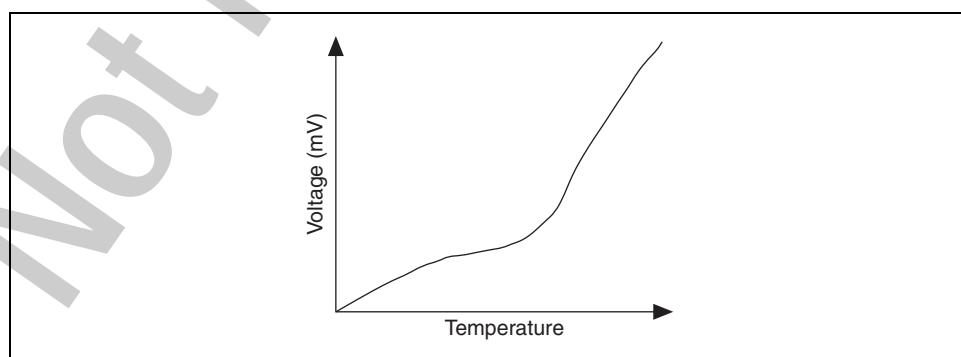


Figure A-1. Thermocouple Temperature versus Voltage Plot

Several types of thermocouples are available. These thermocouples are designated by capital letters that indicate their composition according to American National Standards Institute (ANSI) conventions. For example, a J-type thermocouple has one iron lead and one constantan (a copper-nickel alloy) lead.

Figure A-2 shows a thermocouple. The point at which the thermocouple connects to your measurement system is called the reference junction. When you connect the thermocouple leads to your measurement system leads, you create two additional junctions of dissimilar metals, called cold junctions. These cold junctions induce a thermoelectric voltage in your system. The process of eliminating this voltage is known as cold junction compensation (CJC). You can implement CJC in hardware or software. Both methods require that you measure the temperature at the reference junction with a sensor.

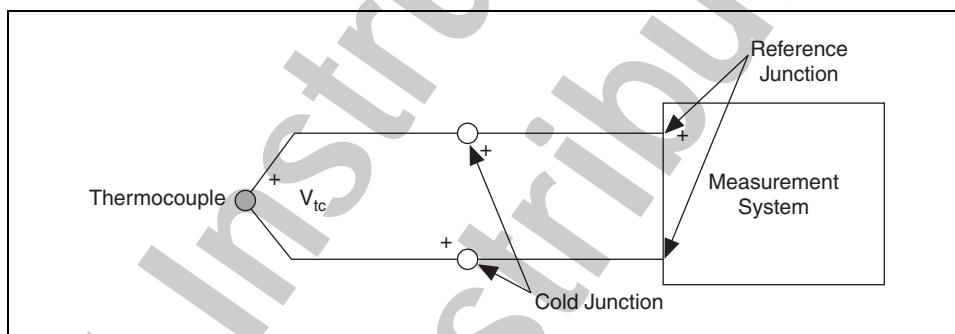


Figure A-2. Thermocouple System

Table A-1 lists some common types of thermocouples. The table identifies the two dissimilar metals that form each thermocouple. The table also shows the valid temperature range for each thermocouple.

Table A-1. Common Types of Thermocouples

Thermocouple Type	Conductor		Temperature Range (°C)	Voltage Range (mV)	Seebeck Coefficient ($\mu\text{V}/(\text{°C})$)
	Positive	Negative			
E	Chromel	Constantan	-270° to 1,000°	-9.835 to 76.358	58.70 at 0 °C
J	Iron	Constantan	-210° to 1,200°	-8.096 to 69.536	50.37 at 0 °C
K	Chromel	Alumel	-270° to 1,372°	-6.548 to 54.874	39.48 at 0 °C
T	Copper	Constantan	-270° to 400°	-6.258 to 20.869	38.74 at 0 °C
S	Platinum-10% Rhodium	Platinum	-50° to 1,768°	-0.236 to 18.698	10.19 at 600 °C
R	Platinum-13% Rhodium	Platinum	-50° to 1,768°	-0.226 to 21.108	11.35 at 600 °C

Thermocouple issues:

- Inexpensive and rugged
- Tolerate high temperatures
- Can track rapid temperature changes
- Requires CJC
- Very small voltages ($7 \mu\text{V} - 40 \mu\text{V}$ per degree C)—needs amplification
- Nonlinear output—measured voltages must be linearized

Resistive Temperature Devices (RTDs)

An RTD is a device whose resistance varies with temperature, as shown in Figure A-3. RTDs are available in different materials and resistance ranges, the most popular being the 100Ω platinum RTD. RTDs are generally more accurate than thermocouples, and do not require cold-junction compensation. However, RTDs are typically more expensive, require linearization for accuracy, and may be affected by lead resistance (for example, long leads).

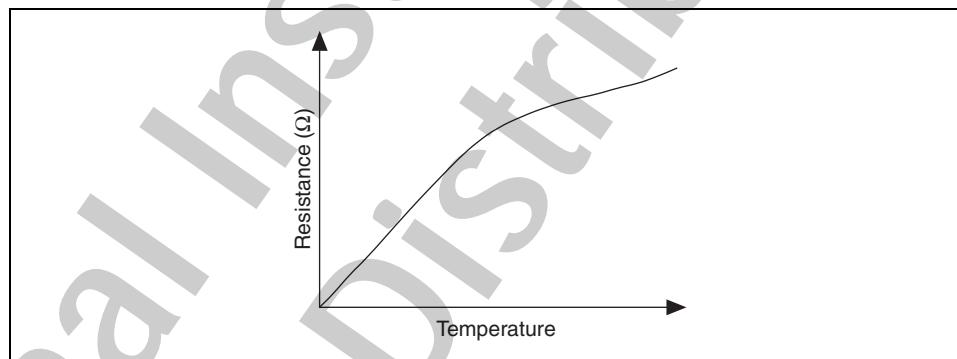


Figure A-3. RTD Temperature versus Resistance Plot

RTDs are available with two, three, or four leads. A two-wire RTD is the simplest, but may be inaccurate due to the wire lead resistance. The three-wire RTD uses a third wire to cancel out this lead resistance. The four-wire RTD (Figure A-4) is the most accurate because it also compensates for any error due to resistance matching of the leads. The RTD is similar to the strain gage in that it requires excitation to induce a voltage across the RTD itself. The form of excitation is usually a current source.

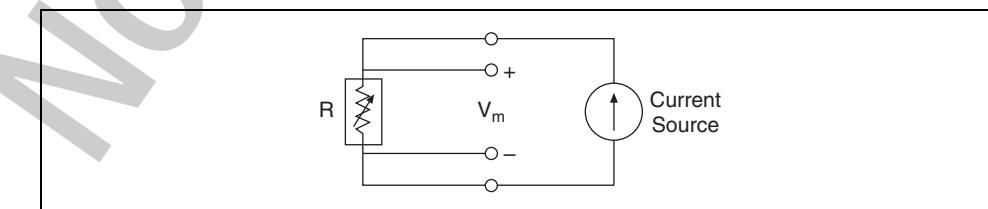


Figure A-4. Four-Wire RTD

In summary, RTDs have the following characteristics:

- More accurate, but more expensive than thermocouples
- Do not require CJC
- Require excitation—from signal conditioning hardware
- Require linearization
- Two-wire configuration is simple but has inaccuracies due to lead resistance
- Three-wire and four-wire RTDs use extra leads to minimize lead resistance

Integrated Circuit Sensors

An integrated circuit (IC) sensor is a temperature sensor made of silicon semiconductor material that acts as a temperature-sensitive resistor. IC sensors require an external power source. Although they are linear and cost efficient, they have slow response times and limited ranges. IC sensors are frequently used as CJC sensors for thermocouple measurements.

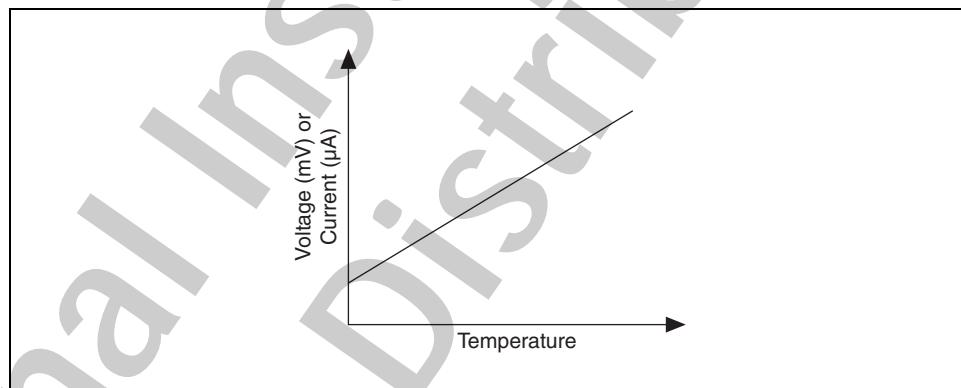


Figure A-5. IC Sensor Temperature versus Resistance Plot

Thermistors

A thermistor is a device whose resistance varies with temperature. This sensor is a piece of semiconductor made from metal oxides pressed into a small bead, disk, wafer or other shape, sintered at high temperatures and finally coated with epoxy or glass. As shown in Figure A-6, thermistors have a nonlinear output and require excitation. Because thermistors are relatively high-resistance devices, they do not require three-wire or four-wire configurations.

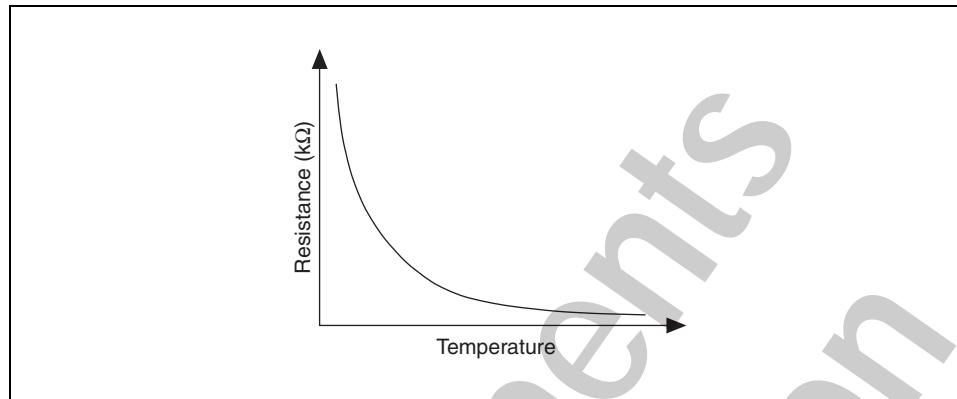


Figure A-6. Thermistor Temperature versus Resistance Plot

In summary, thermistors have the following characteristics:

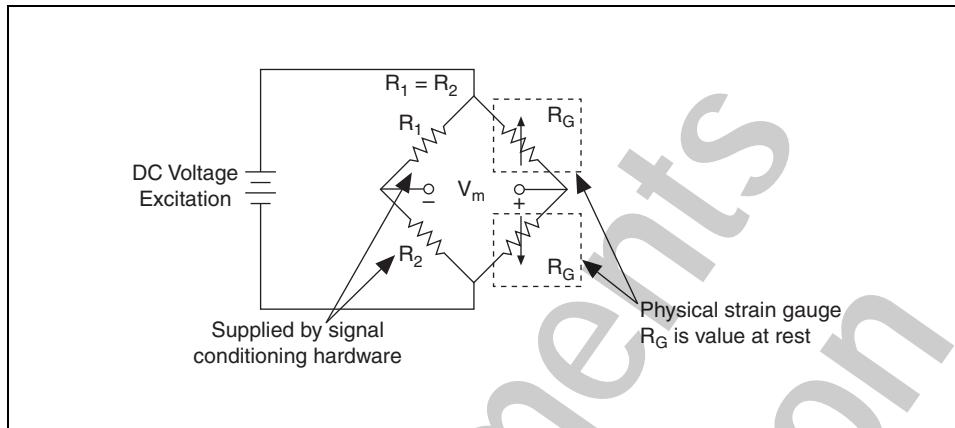
- Require excitation—from signal conditioning hardware
- Require linearization
- High sensitivity
- High resistance—three-wire and four-wire configurations not required

Strain Gages

A strain gage is a device used to sense small movements in materials due to stress or vibrations. Strain gages consist of thin conductors attached to the material to be stressed. Changes in the resistance of the gage indicate deformation of the material. A variety of sensors use strain gage elements mounted on diaphragms or other configurations to sense various physical quantities. For example, load cells are strain gages configured to measure weight.

You usually use strain gages in a configuration of resistors referred to as a Wheatstone bridge. In a bridge, four resistors are placed in a diamond configuration. When you apply a voltage to the bridge, the differential voltage (V_m) at the intermediate nodes varies as the resistor values in the bridge change. The strain gage typically supplies the variable resistors.

Strain gages come in different bridge configurations—full-bridge, half-bridge, and quarter-bridge. For a full-bridge strain gage, the four resistors of the Wheatstone bridge are physically located in the strain gage itself. For a half-bridge strain gage, variable strain gage elements usually occupy two arms of the Wheatstone bridge, while the other two arms are resistors that you or the signal conditioning hardware supply. Figure A-7 shows a half-bridge strain gage. R_1 should equal R_2 , and R_g is the strain gage resistance value at rest. While Figure A-7 shows a DC Voltage source providing excitation, some strain gages require current excitation.

**Figure A-7.** Half-Bridge Strain Gage

B. Analog I/O Circuitry

To help you better understand how a DAQ system converts real-world voltages into digital values that your computer can display and analyze, this section discusses the analog input components of your DAQ device.

Instrumentation Amplifier

To ensure maximum accuracy in the ADC, the instrumentation amplifier applies gain to the input voltage to coerce it to the range of the ADC. Jumpers and/or software configure the range for the ADC and LabVIEW determines the gain to apply according the input signal limits. After amplifying a signal, the amplifier output fluctuates for a period of time, called the settling time, before stabilizing within an acceptable range of the actual voltage. The settling time affects how fast you can accurately sample data depending on the gain that was applied. Figure A-8 shows the settling time characteristics of a standard instrumentation amplifier. Notice the settling time, t_s , that passes before the amplified signal stabilizes within the acceptable voltage range ($\pm\% V$). The amplifier output should be allowed to settle before the ADC begins conversion. If the sampling rate (the rate at which the mux switches channels) exceeds the settling time of the instrumentation amplifier, the acquired data may be incorrect.

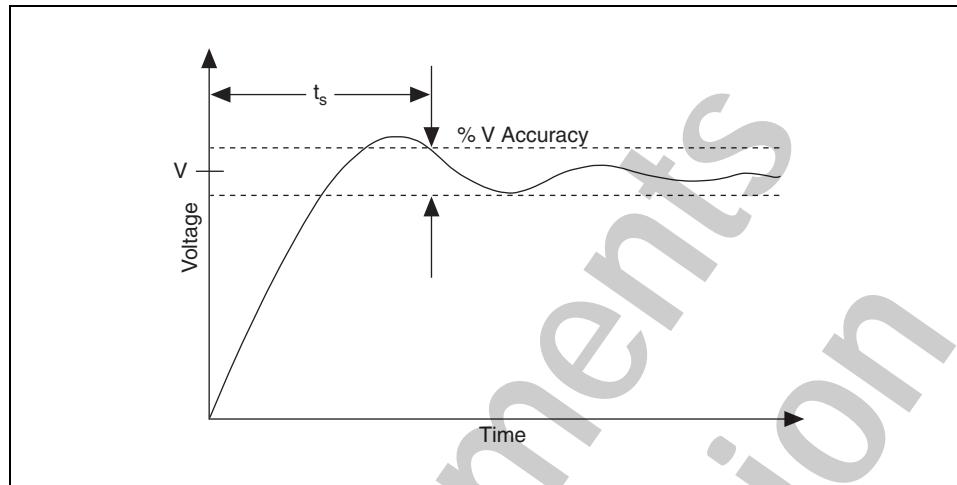


Figure A-8. Settling Time

For example, consider a DAQ device with two input signals as shown in Figure A-9. The signal connected to CH0 is at +5 V and the signal at CH1 is at -5 V. This is a worst-case situation where, when the mux switches, the instrumentation amplifier sees a “step” of 10 V. Assume the settling time for the instrumentation amplifier is 10 μ s and you are sampling at 200 kS/s. The amplifier is not given time to settle before the mux switches for the next reading. Hence, the voltages sent to the ADC for conversion are incorrect.

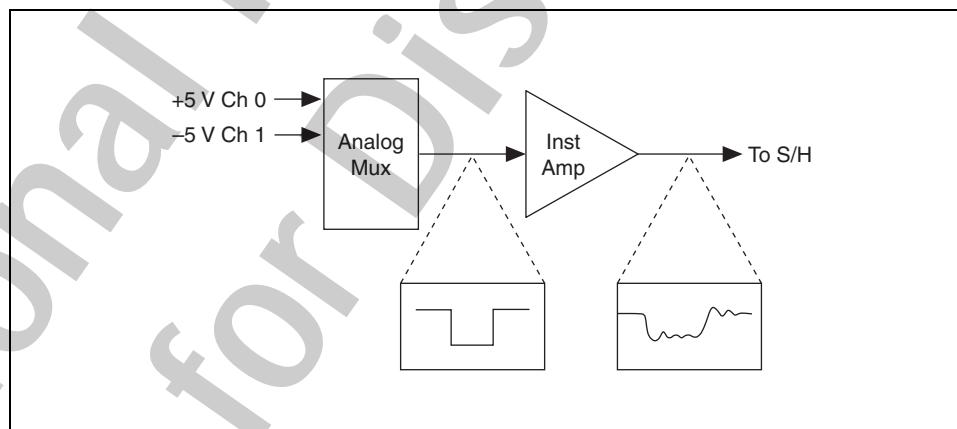


Figure A-9. Effect of Multiplexing Signals into an Instrumentation Amplifier

Figure A-10 shows the signals in Figure A-9 sampled both faster and slower than the settling time. Notice how the ADC digitized the wrong voltage level when the instrumentation amplifier did not have time to settle.

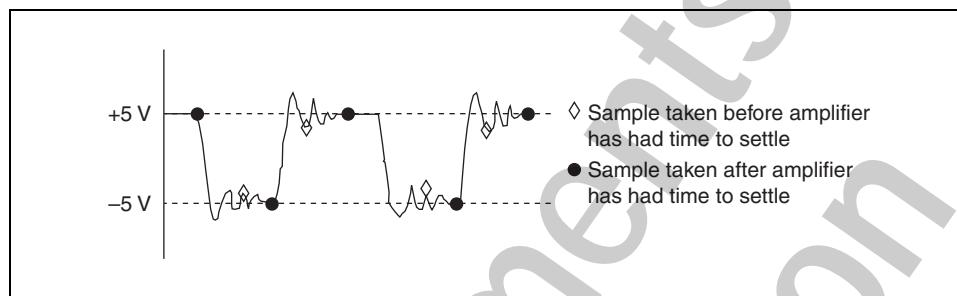


Figure A-10. Effect of Sampling Faster than the Instrumentation Amplifier Settling Time

This example is a worst-case situation—the instrumentation amplifier must swing full-scale. When consecutive input signals are within a certain percentage of each other, the amplifier settling time is much lower. For example, consider the following specifications for an AT-MIO-16X board. The instrumentation amplifier on this board settles to 16-bit accuracy in 40 μ s. You can accurately sample signals that switch between ± 10 V as fast as 25 kHz (worst-case situation). However, if the scanned input channels are within 10% of full-scale range of each other, the amplifier settles much faster and you can sample at the maximum board rate of 100 kHz.

Use the following tips to reduce the effects of settling time:

- Order the signal sampling so that the amplifier experiences the smallest voltage differences possible when switching from one channel to the next.
- Reduce the resistance and capacitance of wires to the DAQ board.
- Obtain a board that has an instrumentation amplifier with superior settling time. Many National Instruments DAQ boards use a custom instrumentation amplifier, the NI-PGIA, that settles equally at all gains. Figure A-11 compares the settling time of the NI-PGIA with an off-the-shelf instrumentation amplifier.

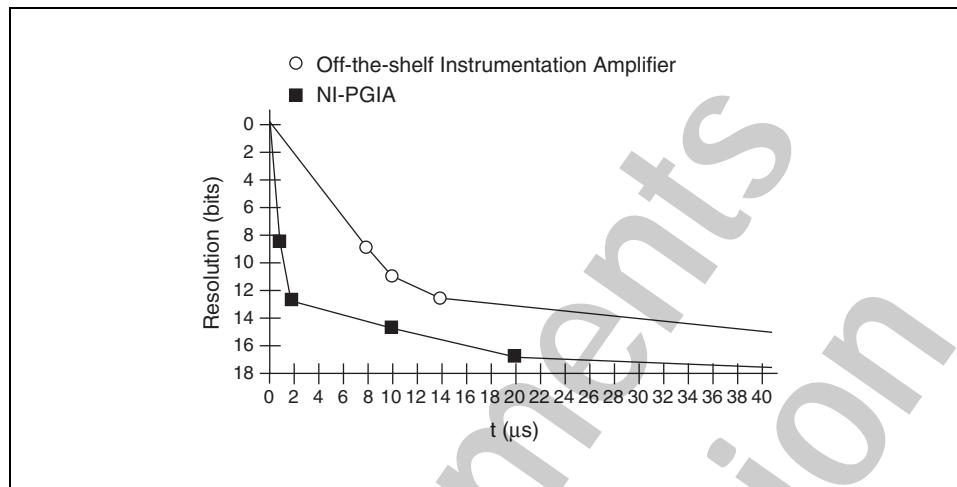


Figure A-11. Settling Time of NI-PGIA versus Off-the-Shelf Instrumentation Amplifier

DAQ device specifications are often in terms of least significant bit (LSB). One LSB is the voltage change increment corresponding to an LSB change in the digital value.

$$1 \text{ LSB} = \frac{\text{Input Range of ADC / Gain}}{2^{\text{number of ADC bits}}}$$

For example, the AT-MIO-16X can settle to ± 0.5 LSB (or $\pm 76.3 \mu\text{V}$) using a gain of 1 and an input range of 10 V.

$$\pm 0.5 \text{ LSB} = \left(\frac{10}{2^{16}} \right) \times 0.5 = \frac{5}{65,536} = \pm 76.3 \mu\text{V}$$

ADC

The fundamental component of the analog input circuitry is the ADC. The ADC digitizes the analog input signal; that is, it converts the analog voltage into a digital value. The ADC stores the digital value in a FIFO buffer until it can be passed to computer memory. During high-speed acquisitions, the FIFO buffer prevents the loss of data due to interrupt latencies that may occur when transferring the data to computer memory.

DAQ devices may use different methods for performing A/D conversions. Some commonly used A/D techniques are successive approximation, flash, subranging (half-flash), integrating, and delta-sigma modulating.

Successive Approximation

The successive approximation ADC is the most popular type of ADC used on DAQ devices because it features high speed and high resolution at a modest cost. Figure A-12 shows an 8-bit successive approximation ADC.

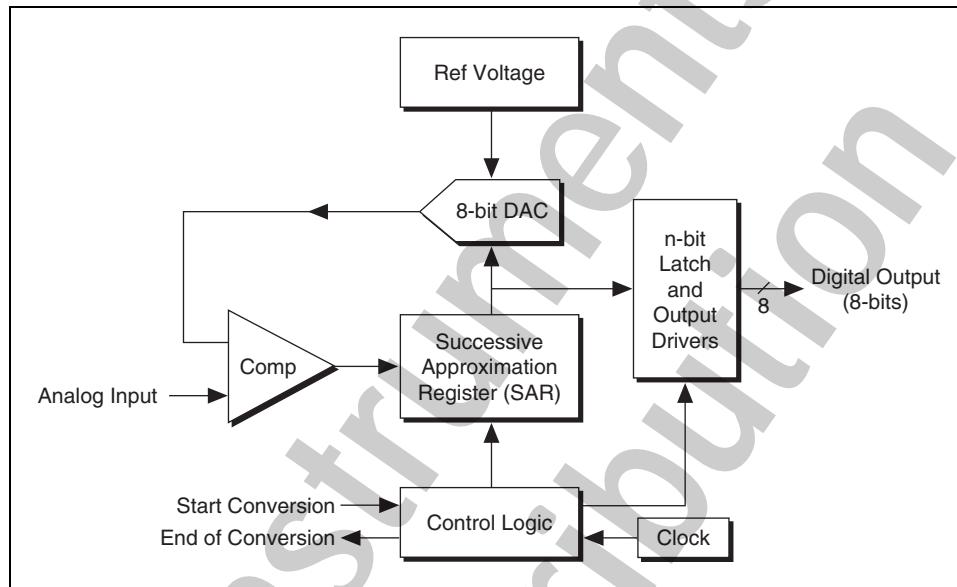


Figure A-12. Successive Approximation ADC

The successive approximation converter uses a technique similar to determining the weight of an object using standard weights. For example, consider that you have four weights—1 g, 2 g, 4 g, and 8 g. You place the unknown weight on one side of a scale and then place the largest known weight on the other side. If the scale does not tip, you add the next largest weight. If the scale tips, you remove the weight and try the next lighter weight. This is done until the scale balances. By totaling the weights put on the scale, you determine the weight of the object.

An 8-bit successive approximation converter works in a similar manner. The SAR initially sets all 8 bits of the DAC to 0. Then, starting with the most significant bit (MSB), each bit is set to 1 and the comparator (Comp) evaluates the output voltage. If the DAC voltage does not exceed the input voltage, the bit is left at 1; otherwise, it is set to 0. A digital code representing the input analog voltage is output after all n bits have been tested. For an 8-bit ADC, this process typically takes less than 2 μ s. Figure A-13 illustrates a conversion sequence for an 8-bit successive approximation ADC.

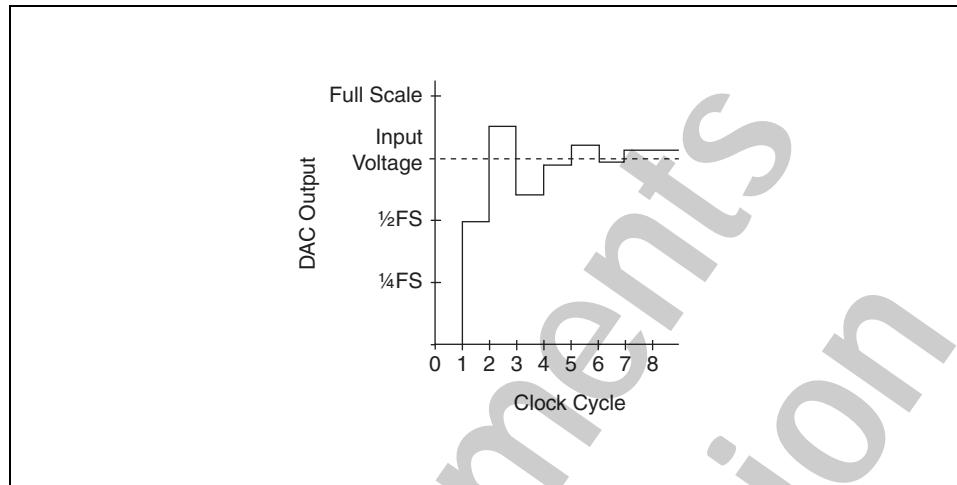


Figure A-13. Conversion Sequence of 8-Bit Successive Approximation ADC

Flash and Half-Flash

The fastest type of ADC is the flash ADC. For an n -bit ADC, the input voltage is applied simultaneously to 2^{n-1} comparators. As shown in Figure A-14, each comparator compares the input voltage to a different reference voltage. The reference voltages of consecutive comparators are one LSB apart. If the input voltage is greater than or equal to the reference voltage, the comparator outputs a 1; otherwise, it outputs a 0. The encoder translates the comparator outputs into a digital code.

Due to component cost and size, flash ADCs are typically available only with 8-bit or less resolution. Half-flash ADCs, a variation of the flash ADCs, use a hybrid technique that results in ADCs with lower cost and higher resolution, but slower conversion times than flash ADCs.

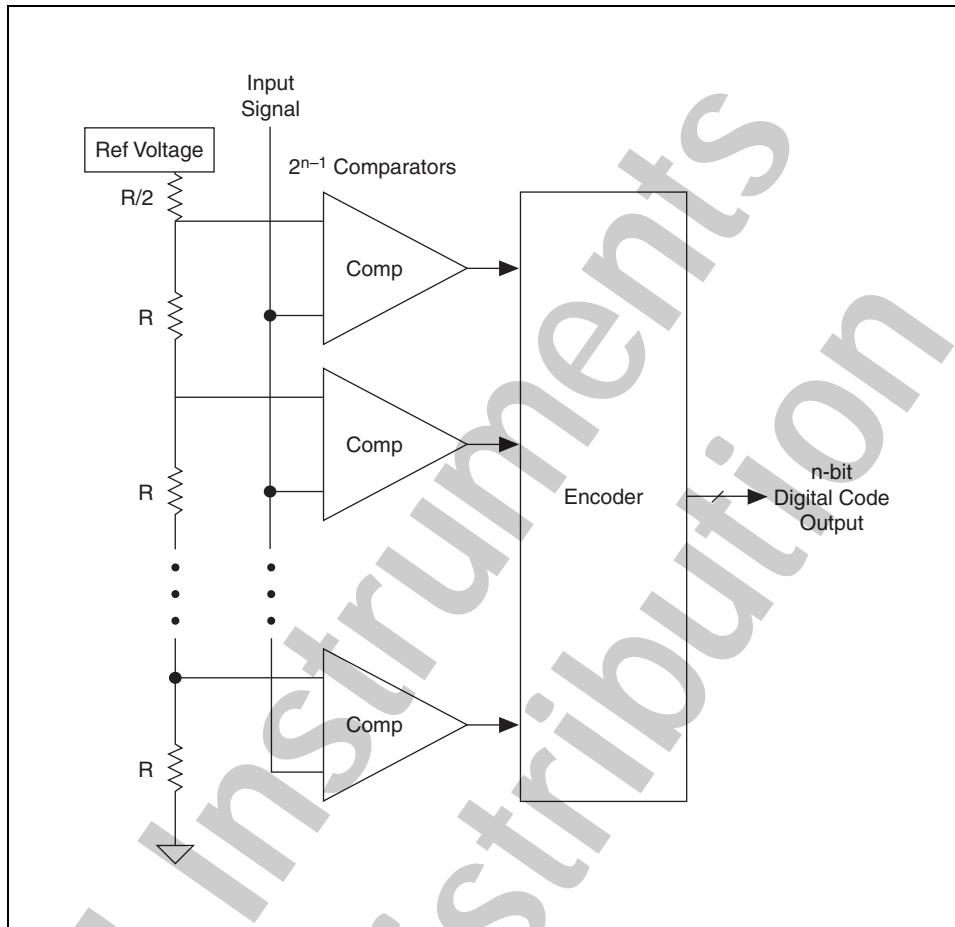


Figure A-14. Flash ADC

Integrating (Dual-Slope)

Another A/D conversion method, integrating, relies on integration to digitize the input signal. This type of ADC has several advantages—high resolution, good linearity, and input noise reduction using averaging. However, its main disadvantage is the slow conversion rate. Therefore, integrating ADCs are primarily used in digital multimeter and other slow measurement devices.

Delta-Sigma Modulating

The state-of-the-art technology in ADCs is the delta-sigma modulating ADC. These ADCs use delta-sigma modulators, combined with oversampling and digital filters to achieve high sampling rates, high resolution, and the best linearity of all ADCs. For example, this type of ADC delivers 16 bits of resolution at 48 kS/s with no differential nonlinearity.

Table A-2. Advantages/Use of Different Types of ADCs

Type of ADC	Advantages/Use
Successive Approximation	High resolution High speed Easily multiplexed Commonly used in DAQ boards DC signals
Flash	Highest speed Mature technology More expensive
Integrating	High resolution Good noise rejection Good linearity Mature technology Slow conversion rate Commonly used in DMMs
Delta Sigma	High resolution Excellent linearity Built-in anti-aliasing AC signals

Notes

National Instruments
Not for Distribution

Notes

National Instruments
Not for Distribution

Signal Processing

This lesson describes the basics of signal processing.

Topics

- A. Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT)
- B. Magnitude and Phase Information
- C. Frequency Spacing and Symmetry of the DFT/FFT
- D. Power Spectrum
- E. About Spectral Leakage and Smoothing Windows
- F. Characteristics of Different Types of Window Functions
- G. Determining Which Type of Window to Use
- H. Filtering
- I. Ideal Filters
- J. Practical (Nonideal) Filters
- K. Advantages of Digital Filters over Analog Filters
- L. IIR and FIR Filters
- M. Infinite Impulse Response Filters
- N. IIR Filter Comparison
- O. Transient Response of IIR Filters
- P. Finite Impulse Response Filters

A. Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT)

The samples of a signal obtained from a DAQ device constitute the time-domain representation of the signal. Time-domain representation gives the amplitudes of the signal at the instant of time during which it is sampled. In many cases you might want to know the frequency content of a signal rather than the amplitudes of the individual samples. The representation of a signal in terms of its individual frequency components is known as the frequency-domain representation of the signal. The frequency-domain representation can give more insight about the signal and the system from which it was generated.

The algorithm used to transform samples of the data from the time domain into the frequency domain is known as the Discrete Fourier Transform or DFT. The DFT establishes the relationship between the samples of a signal in the time domain and their representation in the frequency domain, as shown in Figure B-1. The DFT is used in the fields of spectral analysis, applied mechanics, acoustics, medical imaging, numerical analysis, instrumentation, and telecommunications.

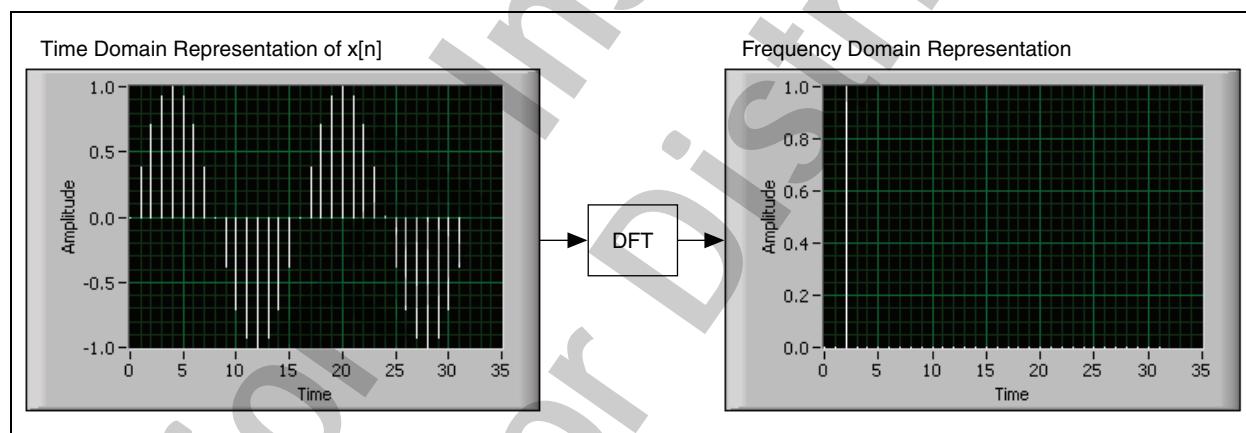


Figure B-1. Discrete Fourier Transform

If you obtain N samples of a signal from a DAQ device and apply the DFT to N samples of the time-domain representation of the signal, the result is also of length N samples, but the information it contains is of the frequency-domain representation.

If the signal is sampled at a sampling rate of f_s Hz, the time interval between the samples (the sampling interval) is Δt , where

$$\Delta t = \frac{1}{f_s}$$

The sample signals are denoted by $x[i]$, $0 \leq i \leq N - 1$ (that is, you have a total of N samples). When the discrete Fourier transform, given by

$$X_k = \sum_{i=0}^{N-1} x_i e^{\frac{-j2\pi ik}{N}} \quad \text{for } k = 0, 1, 2, \dots, N-1 \quad (\text{B-1})$$

is applied to these N samples, the resulting output ($X[k]$, $0 \leq k \leq N - 1$) is the frequency-domain representation of $x[i]$. Notice that both the time-domain x and the frequency-domain X have a total of N samples. Analogous to the time spacing of Δt between the samples of x in the time domain, you have a frequency spacing of

$$\Delta f = \frac{f_s}{N} = \frac{1}{N\Delta t}$$

between the components of X in the frequency domain. Δf also is known as the frequency resolution. To increase the frequency resolution (smaller Δf), either increase the number of samples N with f_s constant or decrease the sampling frequency f_s with N constant.

The Fast Fourier Transform (FFT) is an algorithm that is used to quickly compute the DFT when the number of samples (N) is equal to a power of 2. LabVIEW determines which algorithm to use based on the number of samples obtained. All LabVIEW VIs use the FFT nomenclature.

DFT Calculation Example

Assume that $X[0]$ corresponds to DC, or the average value, of the signal. To see the result of calculating the DFT of a waveform with the use of Equation B-1, consider a DC signal having a constant amplitude of +1 V. Four samples of this signal are taken, as shown in Figure B-2.

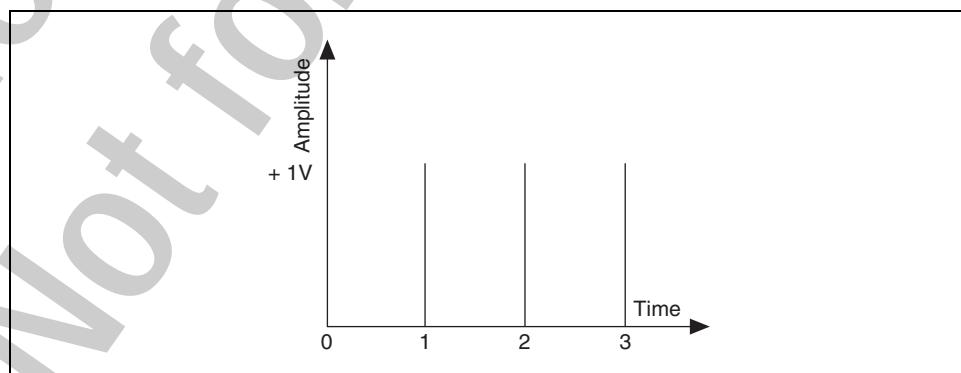


Figure B-2. DC Signal Samples

Each of the samples has a value +1, giving the time sequence

$$x[0] = x[1] = x[2] = x[3] = 1$$

Using Equation B-1 to calculate the DFT of this sequence and making use of Euler's identity,

$$\exp(-j\theta) = \cos(\theta) - j\sin(\theta)$$

results in:

$$X[0] = \sum_{i=0}^{N-1} x_i e^{-j2\pi i 0/N} = x[0] + x[1] + x[3] = 4$$

$$X[1] = x[0] + x[1] \left(\cos\left(\frac{\pi}{2}\right) - j\sin\left(\frac{\pi}{2}\right) \right) + x[2](\cos(\pi) - j\sin(\pi)) + \\ x[3]\left(\cos\left(\frac{3\pi}{2}\right) - j\sin\left(\frac{3\pi}{2}\right)\right) = (1 - j - 1 + j) = 0$$

$$X[2] = x[0] + x[1](\cos(\pi) - j\sin(\pi)) + x[2](\cos(2\pi) - j\sin(2\pi)) + \\ x[3](\cos(3\pi) - j\sin(3\pi)) = (1 - 1 + 1 - 1) = 0$$

$$X[3] = x[0] + x[1]\left(\cos\left(\frac{3\pi}{2}\right) - j\sin\left(\frac{3\pi}{2}\right)\right) + x[2](\cos(3\pi) - j\sin(3\pi)) \\ + x[3]\left(\cos\left(\frac{9\pi}{2}\right) - j\sin\left(\frac{9\pi}{2}\right)\right) = (1 + j - 1 - j) = 0$$

Except for the DC component, $X[0]$, all the other values are zero, as expected. However, the calculated value of $X[0]$ depends on the value of N (the number of samples). Because you had $N = 4$, $X[0] = 4$. If $N = 10$, you would have calculated $X[0] = 10$. This dependency of $X[.]$ on N also occurs for the other frequency components. Usually, you divide the DFT output by N to obtain the correct magnitude of the frequency component.

B. Magnitude and Phase Information

You have seen that N samples of the input signal result in N samples of the DFT. That is, the number of samples in both the time and frequency representations is the same. From Equation B-1, you see that regardless of whether the input signal $x[i]$ is real or complex, $X[k]$ is always complex, although the imaginary part may be zero. Because the DFT is complex, it

contains two pieces of information: the amplitude and the phase. For real signals ($x[i]$ real), such as those obtained from the output of one channel of a DAQ device, the DFT is symmetric about the index $N/2$ with the following properties:

$$|X[k]| = |X[N-k]| \text{ and phase}(X[k]) = -\text{phase}(X[N-k])$$

The magnitude of $X[k]$ is known as even symmetric, and phase($X[k]$) is known as odd symmetric. An even symmetric signal is one that is symmetric about the y-axis, whereas an odd symmetric signal is symmetric about the origin. Figure B-3 demonstrates the symmetry.

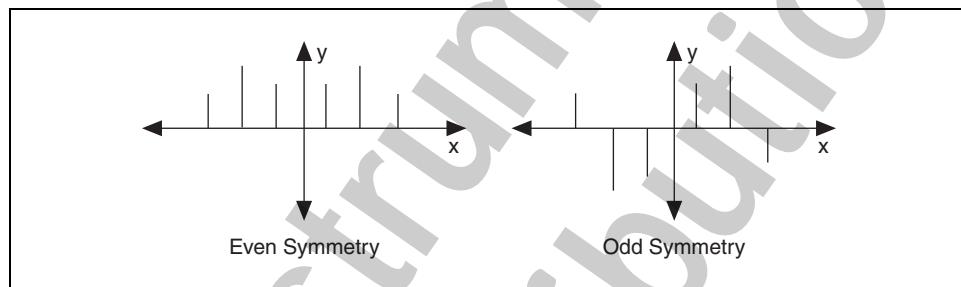


Figure B-3. Even and Odd Symmetry

The net effect of this symmetry is repetition of information contained in the N samples of the DFT. Because of this repetition of information, only half of the samples of the DFT actually need to be computed or displayed. The other half can be obtained from this repetition.



Note If the input signal is complex, the DFT is non-symmetrical, and you cannot use this method.

C. Frequency Spacing and Symmetry of the DFT/FFT

Because the sampling interval is Δt seconds, and if the first ($k = 0$) data sample is assumed to be at 0 seconds, the k^{th} ($k > 0$, k integer) data sample is at $k\Delta t$ seconds. Similarly, the frequency resolution being Δf , where $(\Delta f = \frac{f_s}{N})$, means that the k^{th} sample of the DFT occurs at a frequency of $k\Delta f$ Hz. This is valid for only up to about half the number of samples. The other half represent negative frequency components. Depending on if the number of samples, N , is even or odd, you can have a different interpretation of the frequency that corresponds to the k^{th} sample of the DFT.

Even Number of Samples

Suppose N is even and let $p = \frac{N}{2}$.

Table B-1 shows the frequency to which each element of the complex output sequence X corresponds.

Table B-1. Corresponding Frequency of Array Elements

Array Element	Corresponding Frequency
$X[0]$	DC component
$X[1]$	Δf
$X[2]$	$2\Delta f$
$X[3]$	$3\Delta f$
\vdots	\vdots
$X[p-2]$	$(p-2)\Delta f$
$X[p-1]$	$(p-1)\Delta f$
$X[p]$	$p\Delta f$ (Nyquist frequency)
$X[p+1]$	$-(p-1)\Delta f$
$X[p+2]$	$-(p-2)\Delta f$
\vdots	\vdots
$X[N-3]$	$-3\Delta f$
$X[N-2]$	$-2\Delta f$
$X[N-1]$	$-1\Delta f$

The p^{th} element, $X[p]$, corresponds to the Nyquist frequency. The negative entries in the second column beyond the Nyquist frequency represent negative frequencies.

For example, if $N = 8$, $p = N/2 = 4$, then

$X[0]$	DC
$X[1]$	Δf
$X[2]$	$2\Delta f$
$X[3]$	$3\Delta f$
$X[4]$	$4\Delta f$ (Nyquist freq)
$X[5]$	$-3\Delta f$
$X[6]$	$-2\Delta f$
$X[7]$	$-\Delta f$

$X[1]$ and $X[7]$ have the same magnitude, $X[2]$ and $X[6]$ have the same magnitude, and $X[3]$ and $X[5]$ have the same magnitude. The difference is that whereas $X[1]$, $X[2]$, and $X[3]$ correspond to positive frequency components, $X[5]$, $X[6]$, and $X[7]$ correspond to negative frequency components. Notice that $X[4]$ is at the Nyquist frequency.

Figure B-4 represents this complex sequence for $N = 8$.

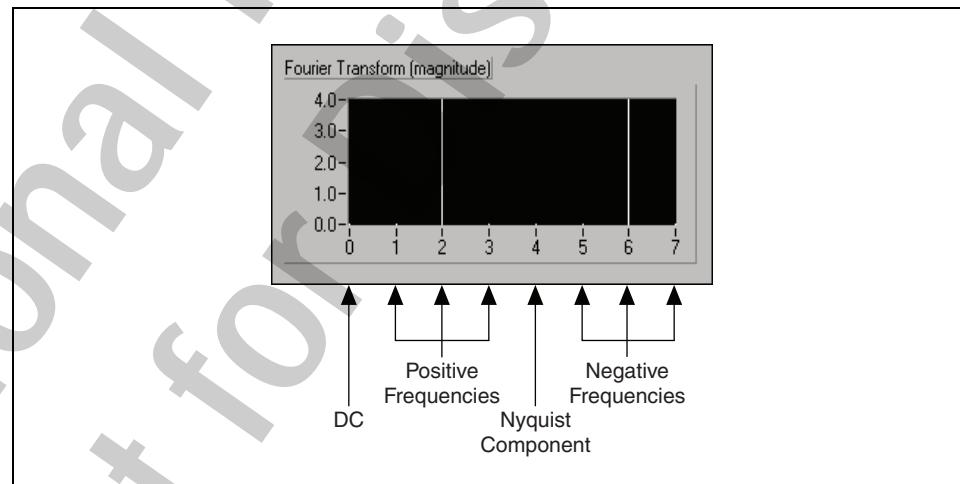


Figure B-4. Two-Sided Transform for $N = 8$

This type of representation, where you see both the positive and negative frequencies, is known as the two-sided transform.

Odd Number of Samples

Suppose that N is odd. Let $p = (N - 1)/2$. Table B-2 shows the frequency to which each element of the complex output sequence X corresponds.

Table B-2. Corresponding Frequency of Array Elements

Array Element	Corresponding Frequency
$X[0]$	DC component
$X[1]$	Δf
$X[2]$	$2\Delta f$
$X[3]$	$3\Delta f$
\vdots	\vdots
$X[p-1]$	$(p-1)\Delta f$
$X[p]$	$p\Delta f$
$X[p+1]$	$-p\Delta f$
$X[p+2]$	$-(p-1)\Delta f$
\vdots	\vdots
$X[N-3]$	$-3\Delta f$
$X[N-2]$	$-2\Delta f$
$X[N-1]$	$-\Delta f$

When N is odd, $N/2$ is not an integer. Therefore, there is no component at the Nyquist frequency.

If $N = 7$, $p = (N-1)/2 = (7-1)/2 = 3$, and you have

$X[0]$	DC
$X[1]$	Δf
$X[2]$	$2\Delta f$
$X[3]$	$3\Delta f$
$X[4]$	$-3\Delta f$
$X[5]$	$-2\Delta f$
$X[6]$	$-\Delta f$

$X[1]$ and $X[6]$ have the same magnitude, $X[2]$ and $X[5]$ have the same magnitude, and $X[3]$ and $X[4]$ have the same magnitude. However, whereas $X[1]$, $X[2]$, and $X[3]$ correspond to positive frequencies, $X[4]$, $X[5]$, and $X[6]$ correspond to negative frequencies. Because N is odd, there is no component at the Nyquist frequency.

Figure B-5 represents the preceding table for $N = 7$.

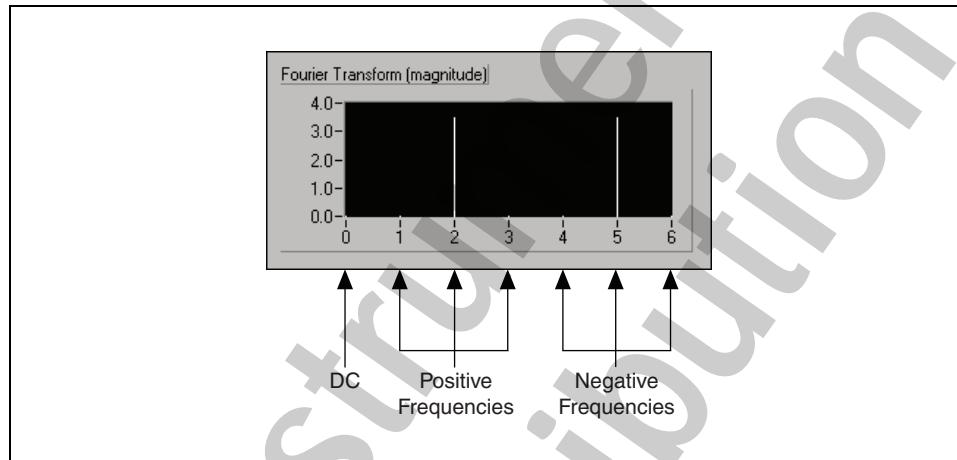


Figure B-5. Two-Sided Transform for $N = 7$

This is also a two-sided transform because you have both positive and negative frequencies.

Fast Fourier Transforms

Direct implementation of the DFT, as in Equation B-1, on N data samples requires approximately N^2 complex operations and is a time-consuming process. However, when the size of the sequence is a power of 2,

$$N = 2^m \text{ for } m = 1, 2, 3, \dots$$

you can implement the computation of the DFT with approximately $N \log_2(N)$ operations. This makes the calculation of the DFT much faster. DSP literature refers to these algorithms as fast Fourier transforms (FFTs). The FFT is a fast algorithm for calculating the DFT when the number of samples (N) is a power of 2.

The advantages of the FFT include speed and memory efficiency because the VI performs the transform in place. The size of the input sequence must be a power of 2. The DFT can process any size sequence efficiently, but the DFT is slower than the FFT and uses more memory because it stores intermediate results during processing.

Zero Padding

A technique employed to make the input sequence size equal to a power of 2 is to add zeros to the end of the sequence so that the total number of samples is equal to the next higher power of 2. For example, if you have 10 samples of a signal, you can add six zeros to make the total number of samples equal to 16 ($= 2^4$, a power of 2). Figure B-6 illustrates this concept.

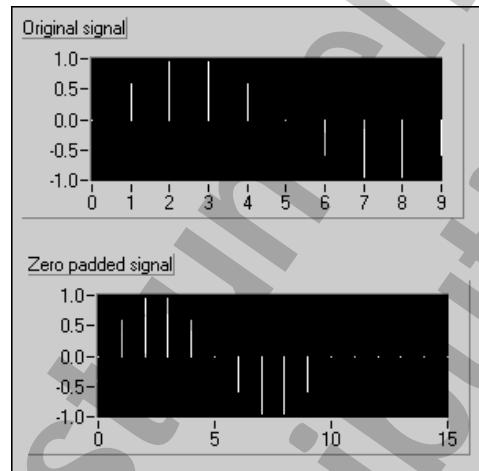


Figure B-6. Zero Padding

In addition to making the total number of samples a power of 2 so that faster computation is made possible by using the FFT, zero padding also helps increase the frequency resolution (recall that $\Delta f = fs/N$) by increasing the number of samples, N .

D. Power Spectrum

The DFT or FFT of a real signal is a complex number, having real and imaginary parts. The power in each frequency component represented by the DFT/FFT can be obtained by squaring the magnitude of that frequency component. Therefore, the power in the k^{th} frequency component (the k^{th} element of the DFT/FFT) is given by $|X[k]|^2$. The plot showing the power in each of the frequency components is known as the power spectrum. Because the DFT/FFT of a real signal is symmetric, the power at a positive frequency of $k\Delta f$ is the same as the power at the corresponding negative frequency of $-k\Delta f$, DC and Nyquist components not included. The total power in the DC and Nyquist components consists of

$$|X[0]|^2 \text{ and } \left|X\left[\frac{N}{2}\right]\right|^2, \text{ respectively.}$$

Loss of Phase Information

Because power is obtained by squaring the magnitude of the DFT/FFT, the power spectrum is always real, and all the phase information is lost. To obtain phase information, use the DFT/FFT, which gives you a complex output.

You can use the power spectrum in applications where phase information is not necessary; for example, to calculate the harmonic power in a signal. You can apply a sinusoidal input to a nonlinear system and see the power in the harmonics at the system output.

Frequency Spacing between Samples

The frequency spacing between the output samples is $\Delta f = fs/N$. In the following table, the power spectrum of a signal $x[n]$ is represented by S_{xx} .

If N is even, let $p = \frac{N}{2}$. Table B-3 shows the format of the output sequence S_{xx} corresponding to the power spectrum.

Table B-3. Interpretations of Array Elements

Array Element	Interpretation
$S_{xx}[0]$	Power in DC component
$S_{xx}[1] = S_{xx}[N-1]$	Power at frequency Δf
$S_{xx}[2] = S_{xx}[N-2]$	Power at frequency $2\Delta f$
$S_{xx}[3] = S_{xx}[N-3]$	Power at frequency $3\Delta f$
\vdots	\vdots
$S_{xx}[p-2] = S_{xx}[N-(p-2)]$	Power at frequency $(p-2)\Delta f$
$S_{xx}[p-1] = S_{xx}[N-(p-1)]$	Power at frequency $(p-1)\Delta f$
$S_{xx}[p]$	Power at Nyquist frequency

Figure B-7 represents the information in the previous table for a sine wave with amplitude = 2 V_{peak} (V_{pk}), and $N = 8$.

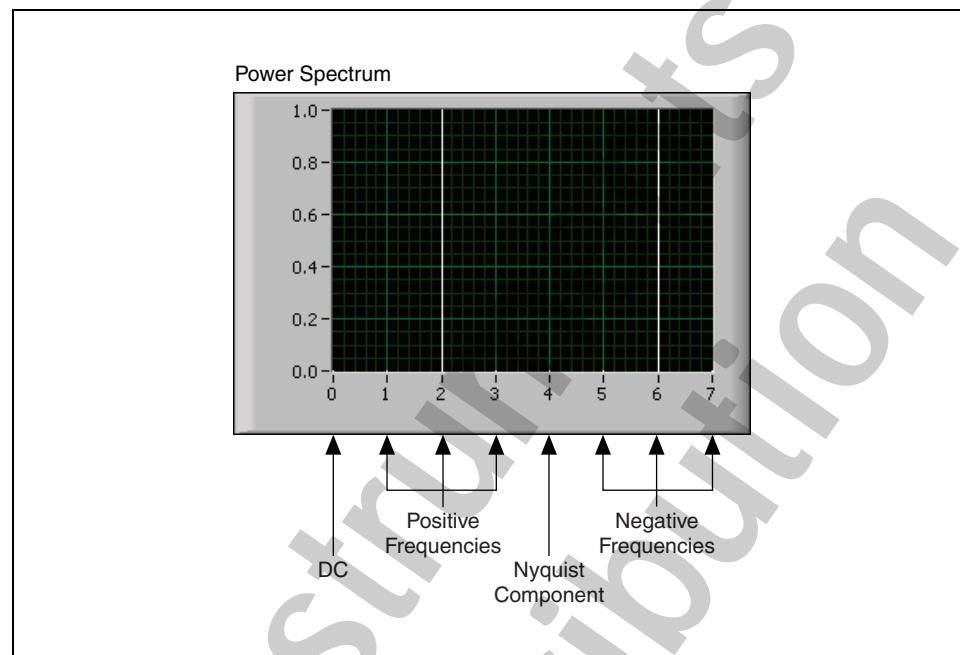


Figure B-7. Power Spectrum for $N = 8$

The output units of the power spectrum calculation are in volts rms squared (V^2_{rms}). So, if the peak amplitude (V_{pk}) of the input signal is 2 V_{pk} , its

rms value is $V_{rms} = \frac{2}{\sqrt{2}} = \sqrt{2}$, so $V^2_{rms} = 2$. This value is divided equally

between the positive and negative frequency components, resulting in the plot shown in Figure B-7.

If N is odd, let $p = (N - 1)/2$. Table B-4 shows the format of the output sequence Sxx corresponding to the power spectrum.

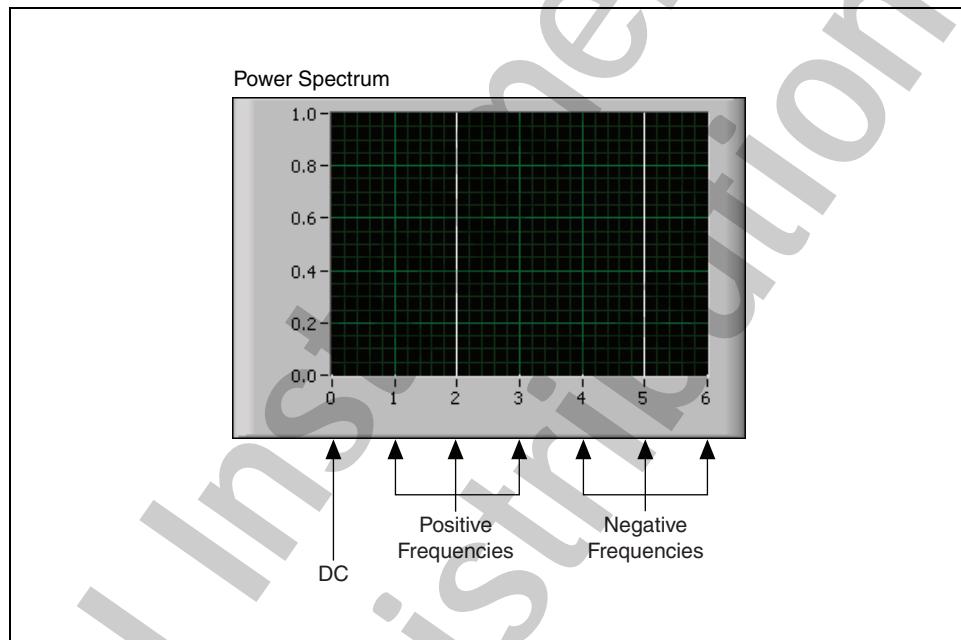
Table B-4. Interpretations of Array Elements

Array Element	Interpretation
$Sxx[0]$	Power in DC component
$Sxx[1] = Sxx[N-1]$	Power at frequency Δf
$Sxx[2] = Sxx[N-2]$	Power at frequency $2\Delta f$
$Sxx[3] = Sxx[N-3]$	Power at frequency $3\Delta f$
\vdots	\vdots
$Sxx[p-2] = Sxx[N-(p-2)]$	Power at frequency $(p-2)\Delta f$

Table B-4. Interpretations of Array Elements (Continued)

Array Element	Interpretation
$S_{xx}[p-1] = S_{xx}[N-(p-1)]$	Power at frequency $(p-1)\Delta f$
$S_{xx}[p] = S_{xx}[p]$	Power at frequency $p\Delta f$

Figure B-8 represents the information in the preceding table for $N = 7$.

**Figure B-8.** Power Spectrum for $N = 8$

Spectral Measurements Express VI

The Spectral Measurements Express VI, located on the Signal Analysis palette, allows you to easily perform various spectral measurements on a signal. These measurements include computing the power spectrum and peak magnitude of a signal. Use the Configure Spectral Measurements property page to configure this Express VI to perform either a magnitude (peak), magnitude (RMS), power spectrum, or power spectral density computation in either linear or dB mode. Additionally, you can specify a windowing mode, averaging parameters, and phase of the spectral measurement. Refer to the *About Spectral Leakage and Smoothing Windows* section of this lesson for more information about windowing modes.

E. About Spectral Leakage and Smoothing Windows

In practical applications, you can obtain only a finite number of samples of the signal. When you use the DFT/FFT to find the frequency content of a signal, the data is assumed to be a single period of a periodically repeating waveform, as shown in Figure B-9. The first period shown is the one sampled. The waveform that corresponds to this period is then repeated in time to produce the periodic waveform.

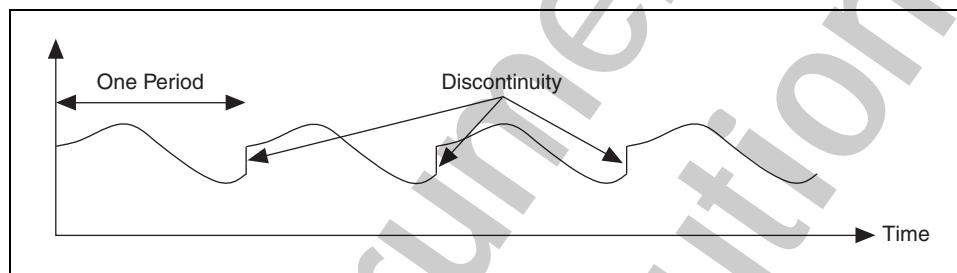


Figure B-9. Discontinuity

Because of the assumption of periodicity of the waveform, discontinuities between successive periods occur. This happens when you sample a noninteger number of cycles. These artificial discontinuities appear as very high frequencies in the spectrum of the signal, frequencies not present in the original signal. These frequencies can be higher than the Nyquist frequency, and as you have seen before, it will be aliased somewhere between 0 and $fs/2$. The spectrum you get by using the DFT/FFT therefore will not be the actual spectrum of the original signal but will be a “smeared” version. It appears as if the energy at one frequency has “leaked out” into all the other frequencies. This phenomenon is known as spectral leakage.

Figure B-10 shows a sine wave and its corresponding Fourier transform. The sampled time-domain waveform is shown in Graph 1. Because the Fourier transform assumes periodicity, you repeat this waveform in time. The periodic time waveform of the sine wave of Graph 1 is shown in Graph 2. The corresponding spectral representation is shown in Graph 3. Because the time record in Graph 2 is periodic, with no discontinuities, its spectrum is a single line that shows the frequency of the sine wave. The reason that the waveform in Graph 2 does not have any discontinuities is because you sampled an integer number of cycles—in this case, one cycle—of the time waveform.

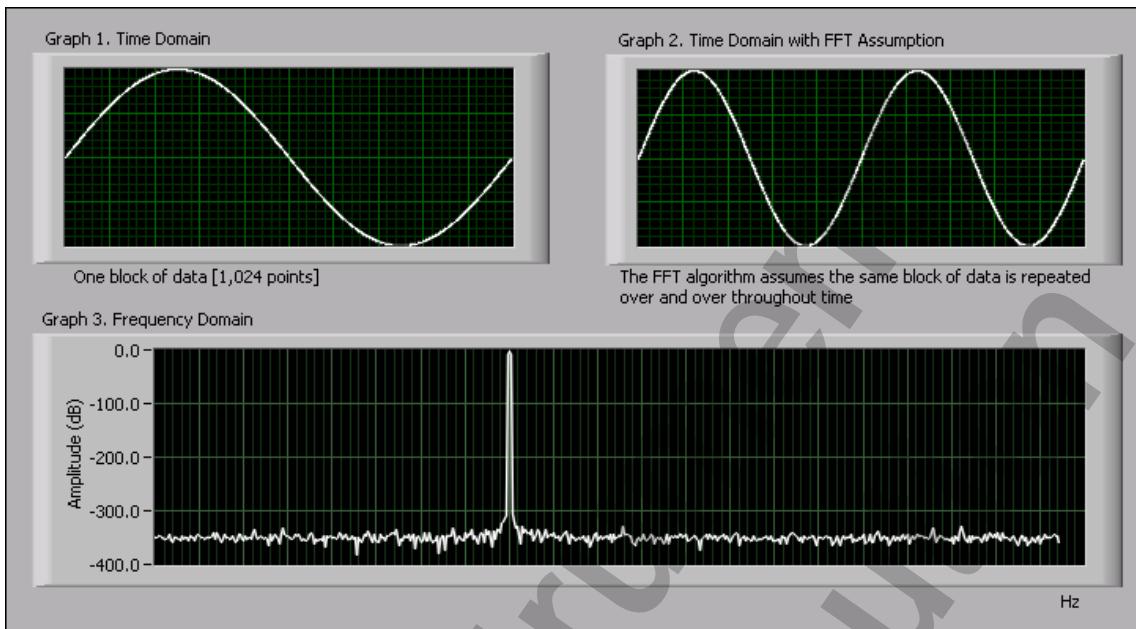


Figure B-10. Sine Wave and Corresponding Fourier Transform

In Figure B-11, you see the spectral representation when you sample a noninteger number of cycles of the time waveform, namely 1.25. Graph 1 now consists of 1.25 cycles of the sine wave. When you repeat this periodically, the resulting waveform as shown in Graph 2 consists of discontinuities. The corresponding spectrum is shown in Graph 3, in which the energy spreads over a wide range of frequencies. This smearing of the energy is spectral leakage. The energy leaked out of one of the FFT lines and smeared itself into all the other lines.

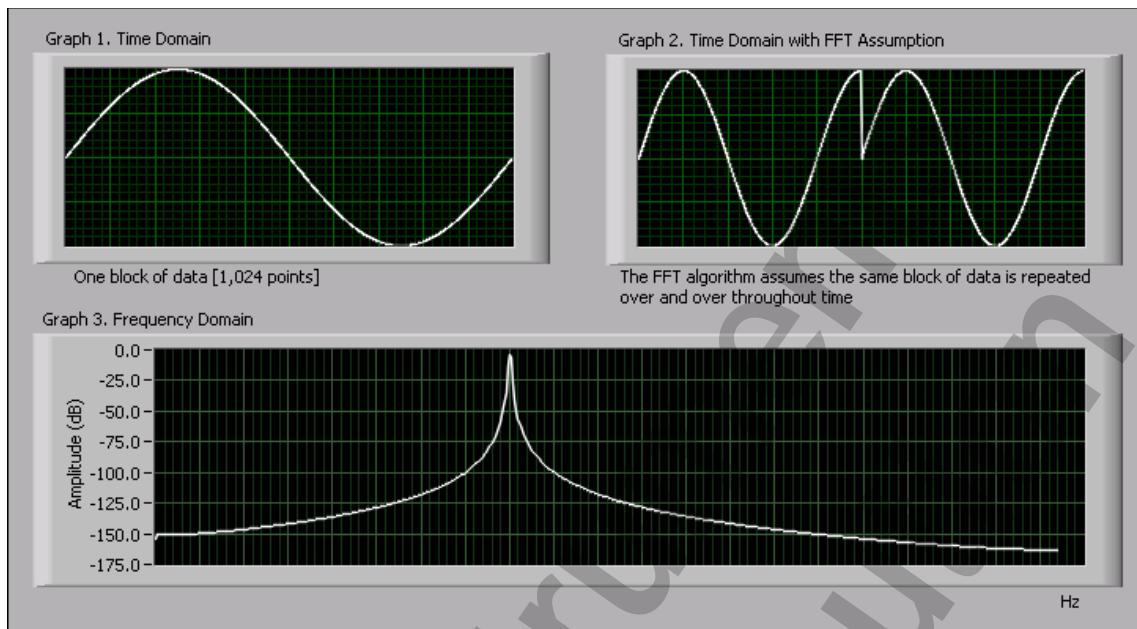


Figure B-11. Spectral Representation When Sampling a Nonintegral Number of Samples

Leakage exists because of the finite time record of the input signal. One solution for overcoming leakage is to take an infinite time record, from $-x$ to $+x$, then have the FFT calculate one single line at the correct frequency. Because you are limited to having a finite time record, another technique known as windowing reduces the spectral leakage.

The amount of spectral leakage depends on the amplitude of the discontinuity. The larger the discontinuity, the more leakage, and vice versa. You can use windowing to reduce the amplitude of the discontinuities at the boundaries of each period. Windowing consists of multiplying the time record by a finite-length window whose amplitude varies smoothly and gradually toward zero at the edges. This is shown in Figure B-12, where the original time signal is windowed using a Hamming window. The time waveform of the windowed signal gradually tapers to zero at the ends. When performing Fourier or spectral analysis on finite-length data, you can use windows to minimize the transition edges of the sampled waveform. A smoothing window function applied to the data before it is transformed into the frequency domain minimizes spectral leakage.

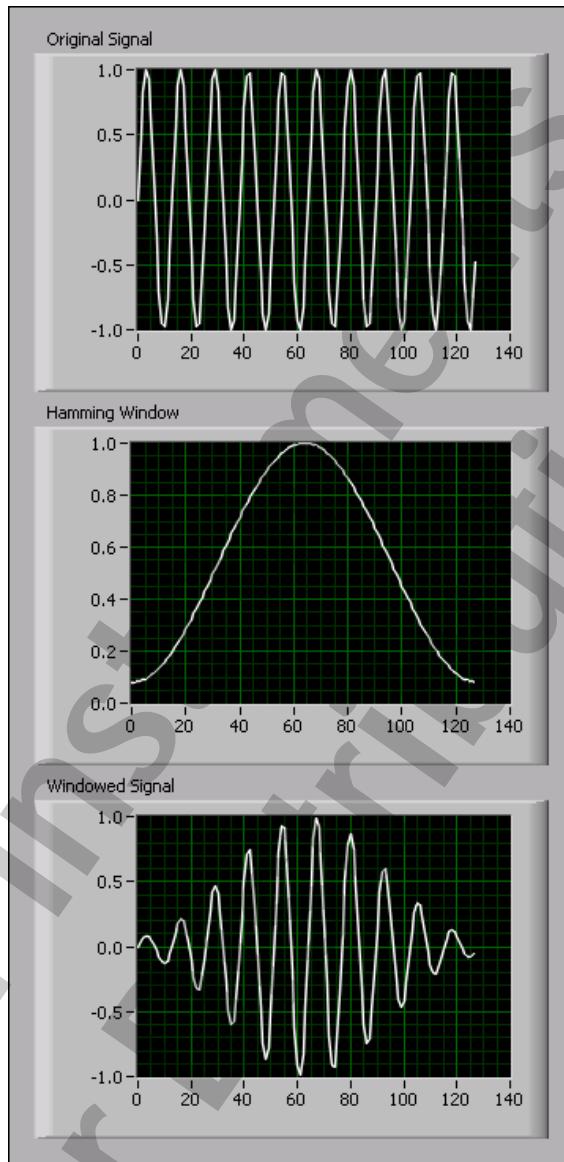


Figure B-12. Time Signal Windowed Using a Hamming Window

If the time record contains an integral number of cycles, as shown in Figure B-10, the assumption of periodicity does not result in any discontinuities, and therefore no spectral leakage occurs. The problem arises only when you have a nonintegral number of cycles.

There are several reasons to use windowing:

- Defining the duration of the observation
- Reducing spectral leakage
- Separating a small amplitude signal from a larger amplitude signal with frequencies very close to each other

F. Characteristics of Different Types of Window Functions

Applying a window to, or windowing, a signal in the time domain is equivalent to multiplying the signal by the window function. Because multiplication in the time domain is equivalent to convolution in the frequency domain, the spectrum of the windowed signal is a convolution of the spectrum of the original signal with the spectrum of the window. Windowing changes the shape of the signal in the time domain and affects the spectrum that you see.

Depending on the application, one window may be more useful than the others. Using the Spectral Measurements Express VI, you can choose from rectangular (none), Hanning, Hamming, Blackman-Harris, Exact Blackman, Blackman, Flat Top, 4 Term B-Harris, 7 Term B-Harris, and Low Sidelobe windows. The Exponential, General Cosine, Cosine Tapered, Force, Kaiser-Bessel and Triangle windowing functions are also available in the **Analyze»Signal Processing»Windows** palette.

Rectangular (None)

The rectangular window has a value of one over its time interval. Mathematically, it can be written as

$$w[n] = 1.0 \text{ for } n = 0, 1, 2, \dots, N - 1$$

where N is the length of the window. Applying a rectangular window is equivalent to not using any window because the rectangular function truncates the signal to within a finite time interval. The rectangular window has the highest amount of spectral leakage. Figure B-13 shows the rectangular window for $N = 32$.

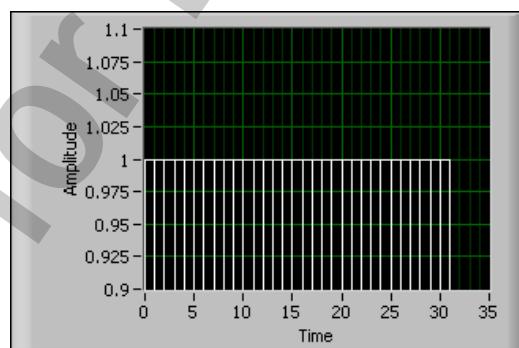


Figure B-13. Rectangular Window

The rectangular window is useful for analyzing transients that have a duration shorter than that of the window. It also is used in order tracking, where the sampling frequency is adjusted depending on the speed of the shaft of a machine. In this application, it detects the main mode of vibration of the machine and its harmonics.

Hanning

The Hanning window has a shape similar to that of a half-cycle of a cosine wave. Its defining equation is

$$w[n] = 0.5 - 0.5 \cos(2\pi n/N) \text{ for } n = 0, 1, 2, \dots, N - 1$$

Figure B-14 shows a Hanning window with $N = 32$.

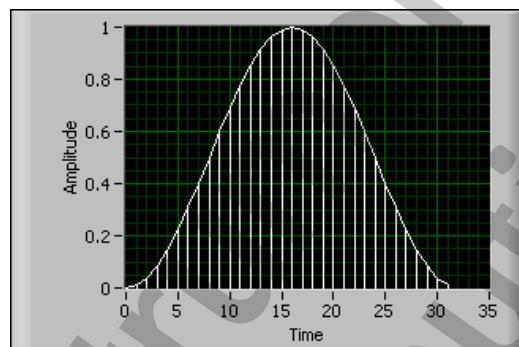


Figure B-14. Hanning Window

The Hanning window is useful for analyzing transients longer than the time duration of the window and for general purpose applications.

Hamming

The Hamming window is a modified version of the Hanning window. Its shape also is similar to that of a cosine wave. It can be defined as:

$$w[n] = 0.54 - 0.46 \cos(2\pi n/N) \text{ for } n = 0, 1, 2, \dots, N - 1$$

Figure B-15 shows a Hamming window with $N = 32$.

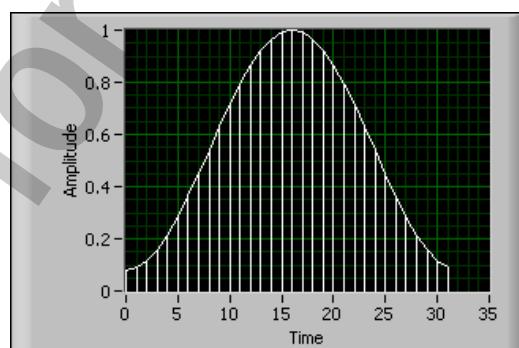


Figure B-15. Hamming Window

The Hanning and Hamming windows are somewhat similar. In the time domain, the Hamming window does not get as close to 0 near the edges as the Hanning window.

Blackman-Harris

The Blackman-Harris window, as well as the Hanning window, is useful for measuring very low-level components in the presence of a large input signal, such as a distortion measurement.

The Blackman-Harris window applies a three-term window to the input signal. It can be defined as:

$$w[n] = 0.422323 - 0.49755 \cos(2\pi n/N) + 0.07922 \cos(4\pi n/N)$$

for $n = 0, 1, 2, \dots, N - 1$

Figure B-16 shows a Blackman-Harris window with $N = 32$.

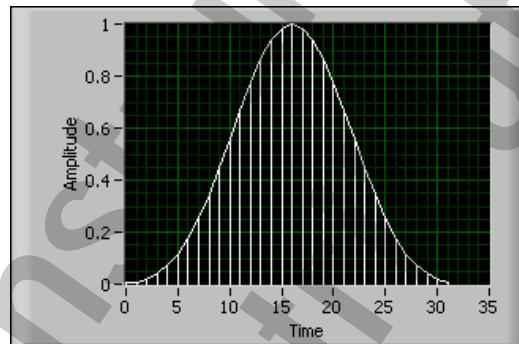


Figure B-16. Blackman-Harris Window

Exact Blackman

The Exact Blackman window is similar to the Blackman-Harris window, but with less taper at the end.

It can be defined as:

$$w[n] = [a_0 - a_1 \cos(2\pi n/N) + a_2 \cos(4\pi n/N)]$$

for $n = 0, 1, 2, \dots, N - 1$

$$a_0 = 7938/18608$$

$$a_1 = 9240/18608$$

$$a_2 = 1430/18608$$

Figure B-17 shows an Exact Blackman window with $N = 32$.

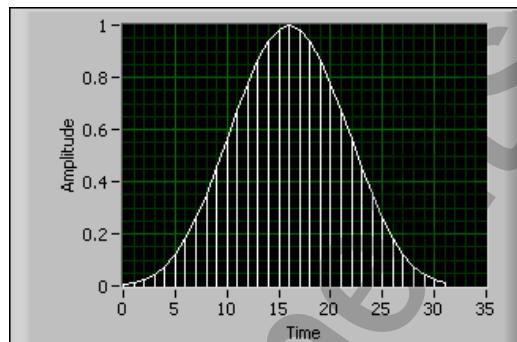


Figure B-17. Exact Blackman Window

Blackman

The Blackman window is similar to the Hanning and Hamming window, but has an additional cosine term that further reduces the ripple effect.

It can be defined as:

$$w[n] = 0.42 - 0.5 \cos(2\pi n/N) + 0.08 \cos(4\pi n/N)$$

for $n = 0, 1, 2, \dots, N - 1$

Figure B-18 shows a Blackman window with $N = 32$.

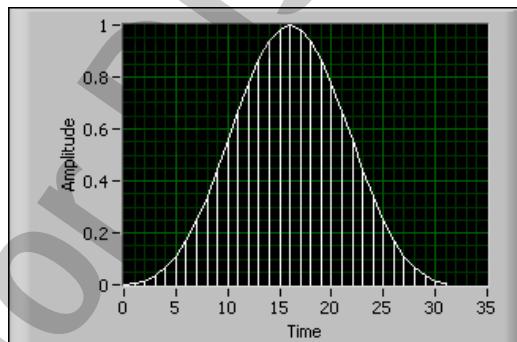


Figure B-18. Blackman Window

Flat Top

The Flat Top window consists of more cosine terms than any other window we have studied thus far. The second harmonic term causes the window to drop below zero.

It can be defined as:

$$w[n] = 0.21557895 - 0.41663158 \cos(2\pi n/N) + 0.277263158 \cos(4\pi n/N) \\ - 0.083578947 \cos(6\pi n/N) + 0.006947368 \cos(8\pi n/N)$$

for $n = 0, 1, 2, \dots, N - 1$

Figure B-19 shows a Flat Top window with $N = 32$.

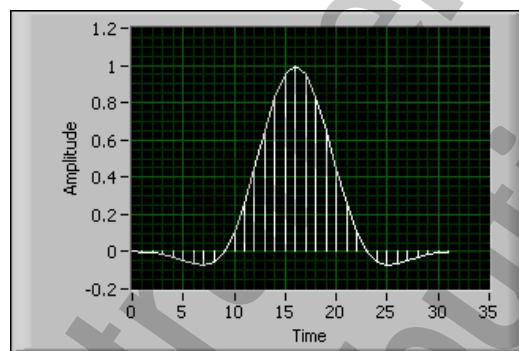


Figure B-19. Flat Top Window

4 Term B-Harris

The 4 Term B-Harris window is an extension of the Blackman-Harris window, adding an additional cosine term.

It can be defined as:

$$w[n] = 0.35875 - 0.48829 \cos(2\pi n/N) \\ + 0.14128 \cos(4\pi n/N) - 0.01168 \cos(6\pi n/N)$$

for $n = 0, 1, 2, \dots, N - 1$

Figure B-20 shows a 4 Term B-Harris window with $N = 32$.

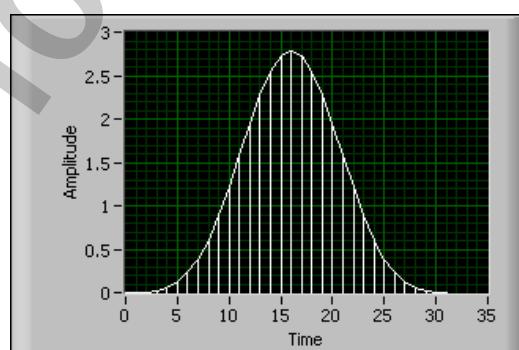


Figure B-20. 4 Term B-Harris Window

7 Term B-Harris

The 7 Term B-Harris window is an extension of the Blackman-Harris window, adding an additional four cosine terms.

It can be defined as:

$$\begin{aligned} w[n] = & 0.27105 - 0.43329 \cos(2\pi n/N) + 0.21812 \cos(4\pi n/N) \\ & - 0.06593 \cos(6\pi n/N) + 0.01081 \cos(8\pi n/N) \\ & - 7.7658E-4 \cos(10\pi n/N) + 1.3887E-5 \cos(12\pi n/N) \end{aligned}$$

for $n = 0, 1, 2, \dots, N - 1$

Figure B-21 shows a 7 Term B-Harris window with $N = 32$.

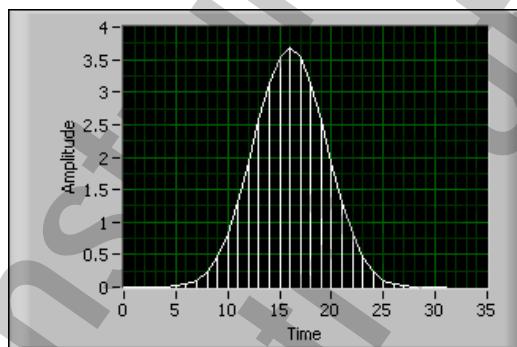


Figure B-21. 7 Term B-Harris Window

Low Sidelobe

The Low Sidelobe window further reduces the size of the mainlobe.

It can be defined as:

$$w[n] = 0.323215218 - 0.471492057 \cos(2\pi n/N) + 0.17553428 \cos(4\pi n/N)$$

$$- 0.028497078 \cos(6\pi n/N) + 0.001261367 \cos(8\pi n/N)$$

for $n = 0, 1, 2, \dots, N - 1$

Figure B-22 shows a Low Sidelobe window with $N = 32$.

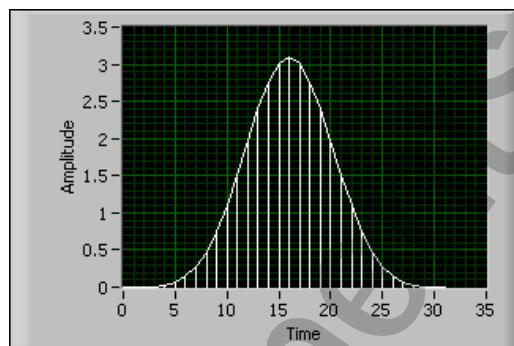


Figure B-22. Low Sidelobe Window

G. Determining Which Type of Window to Use

The type of window you choose depends on the type of signal you have and what you are looking for. Selecting the correct window requires knowledge of the signal that you are analyzing. Table B-5 shows the different types of signals and the windows you can use with them.

Table B-5. Window for Signal Type

Type of signal	Window
Transients whose duration is shorter than the length of the window	Rectangular
Transients whose duration is longer than the length of the window	Hanning
General purpose applications	Hanning
Order tracking	Rectangular
System analysis (frequency response measurements)	Hanning (for random excitation), rectangular (for pseudorandom excitation)
Separation of two tones with frequencies very close to each other but with widely differing amplitudes	Kaiser-Bessel
Separation of two tones with frequencies very close to each other but with almost equal amplitudes	Rectangular

If you do not have sufficient prior knowledge of the signal, experiment with different windows to find the best one.

Table B-6. Window Descriptions

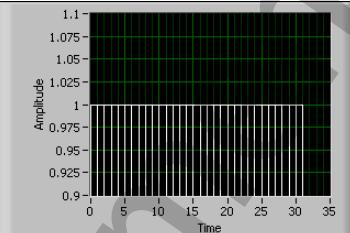
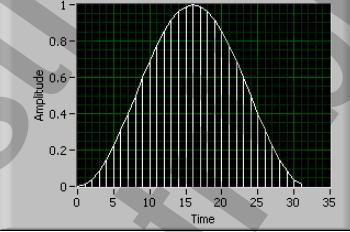
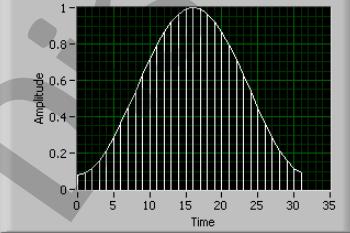
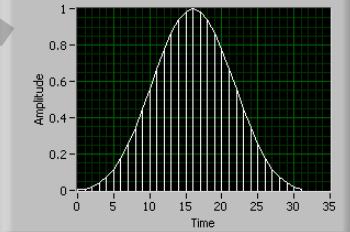
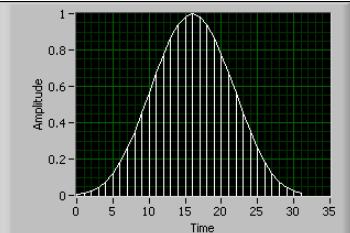
Window	Equation	Shape	Applications
Rectangular (None)	$w[n] = 1.0$		Detecting transients whose duration is shorter than the length of the window; order tracking; separating two tones with frequencies and amplitudes very close to each other; system response
Hanning	$w[n] = 0.5 - 0.5 \cos\left(\frac{2\pi n}{N}\right)$		General purpose applications; system analysis; transients whose duration is longer than the length of the window
Hamming	$w[n] = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N}\right)$		—
Blackman- Harris	$w[n] = 0.422322 - 0.49755 \cos\left(\frac{2\pi n}{N}\right) + 0.07922 \cos\left(\frac{4\pi n}{N}\right)$		Similar to Blackman
Exact Blackman	$w[n] = \frac{7938}{18608} - \frac{9240}{18608} \cos\left(\frac{2\pi n}{N}\right) + \frac{1430}{18608} \cos\left(\frac{4\pi n}{N}\right)$		Similar to Blackman

Table B-6. Window Descriptions (Continued)

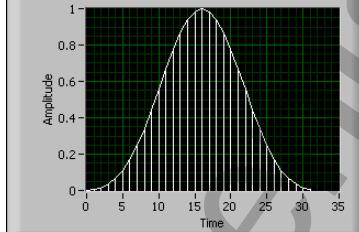
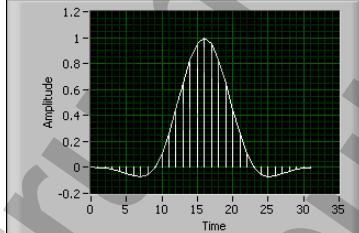
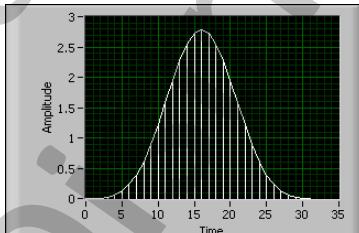
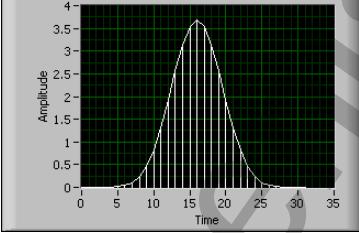
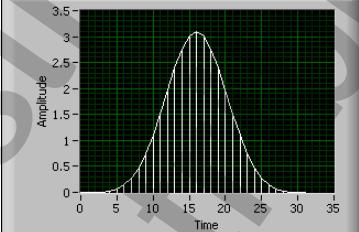
Window	Equation	Shape	Applications
Blackman	$w[n] = 0.42 - 0.5 \cos\left(\frac{2\pi n}{N}\right) + 0.8 \cos\left(\frac{4\pi n}{N}\right)$		Transient signals; similar to Hanning and Hamming windows but adds one additional cosine term to reduce ripple
Flat Top	$w[n] = 0.21557895 - 0.41663158 \cos\left(\frac{2\pi n}{N}\right) + 0.277263158 \cos\left(\frac{4\pi n}{N}\right) - 0.083578947 \cos\left(\frac{6\pi n}{N}\right) + 0.006947368 \cos\left(\frac{8\pi n}{N}\right)$		Accurate single tone amplitude measurements when there are no nearby frequency components
4 Term B-Harris	$w[n] = 0.35875 - 0.48829 \cos\left(\frac{2\pi n}{N}\right) + 0.14128 \cos\left(\frac{4\pi n}{N}\right) - 0.01168 \cos\left(\frac{6\pi n}{N}\right)$		Similar to Blackman

Table B-6. Window Descriptions (Continued)

Window	Equation	Shape	Applications
7 Term B-Harris	$w[n] = 0.27105 - 0.43329 \cos\left(\frac{2\pi n}{N}\right) + 0.21812 \cos\left(\frac{4\pi n}{N}\right) - 0.06593 \cos\left(\frac{6\pi n}{N}\right) + 0.01081 \cos\left(\frac{8\pi n}{N}\right) - 7.7658E-4 \cos\left(\frac{10\pi n}{N}\right) + 1.3887E-5 \cos\left(\frac{12\pi n}{N}\right)$		Similar to Blackman
Low Sidelobe	$w[n] = 0.323215218 - 0.471492057 \cos\left(\frac{2\pi n}{N}\right) + 0.17553428 \cos\left(\frac{4\pi n}{N}\right) - 0.028497078 \cos\left(\frac{6\pi n}{N}\right) + 0.001261367 \cos\left(\frac{8\pi n}{N}\right)$		—

H. Filtering

Filtering is the process by which the frequency content of a signal is altered. It is one of the most commonly used signal processing techniques. Common everyday examples of filtering are the bass and treble controls on a stereo system. The bass control alters the low-frequency content of a signal, and the treble control alters the high-frequency content. By varying these controls, you are actually filtering the audio signal. Some other applications where filtering is useful are removing noise and performing decimation (lowpass filtering the signal and reducing the sample rate).

I. Ideal Filters

Filters alter or remove unwanted frequencies. Depending on the frequency range that they either pass or attenuate, they can be classified into the following types:

- A lowpass filter passes low frequencies but attenuates high frequencies.
- A highpass filter passes high frequencies but attenuates low frequencies.

- A bandpass filter passes a certain band of frequencies.
- A bandstop filter attenuates a certain band of frequencies.

Figure B-23 shows the ideal frequency response of these filters.

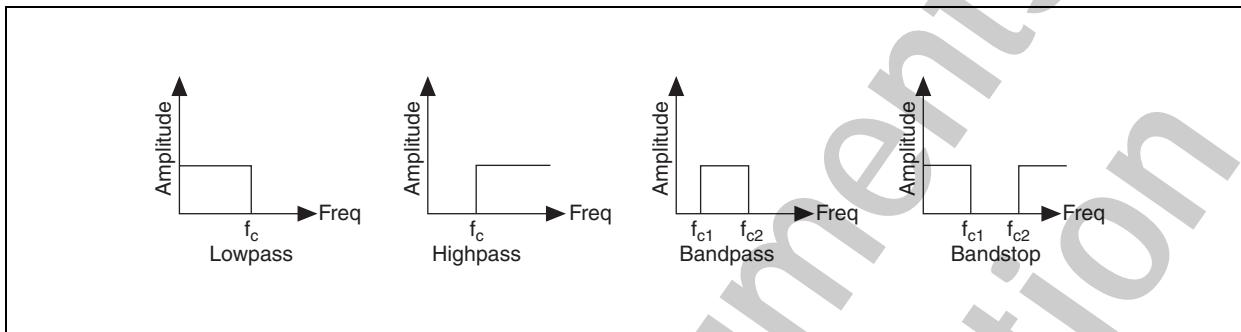


Figure B-23. Ideal Frequency Responses

The lowpass filter passes all frequencies below f_c , and the highpass filter passes all frequencies above f_c . The bandpass filter passes all frequencies between f_{c1} and f_{c2} , and the bandstop filter attenuates all frequencies between f_{c1} and f_{c2} . The frequency points f_c , f_{c1} , and f_{c2} are known as the cutoff frequencies of the filter. When designing filters, you need to specify these cutoff frequencies.

The frequency range that passes through the filter is known as the passband (PB) of the filter. An ideal filter has a gain of one (0 dB) in the passband so that the amplitude of the signal neither increases nor decreases. The stopband (SB) corresponds to that range of frequencies that do not pass through the filter at all and are rejected (attenuated). Figure B-24 shows the passband and the stopband for the different types of filters.

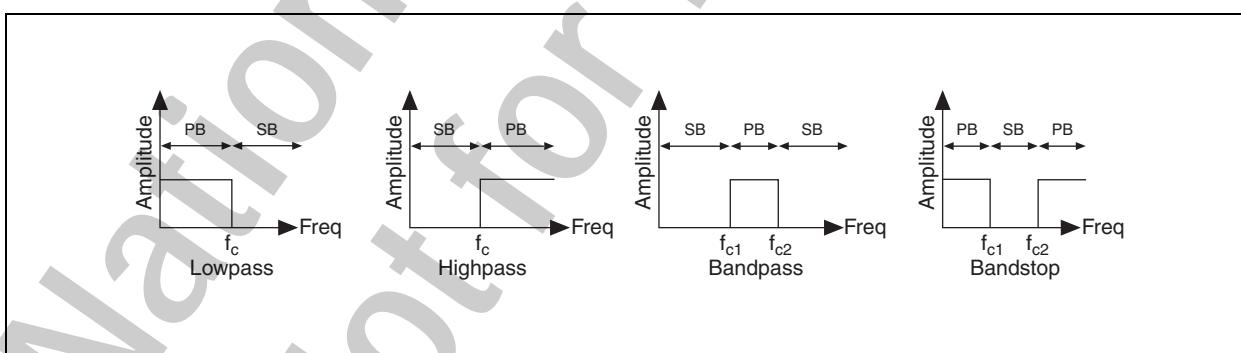


Figure B-24. Filter Passbands and Stopbands

The lowpass and highpass filters have one passband and one stopband, the bandpass filter has one passband but two stopbands, and the bandstop filter has two passbands but one stopband.

How Filters Affect Signal Frequency Content

A signal that contains frequencies of 10 Hz, 30 Hz, and 50 Hz passes through lowpass, highpass, bandpass, and bandstop filters. The lowpass and highpass filters have a cutoff frequency of 20 Hz, and the bandpass and bandstop filters have cutoff frequencies of 20 Hz and 40 Hz. Figure B-25 shows the output of the filter in each case.

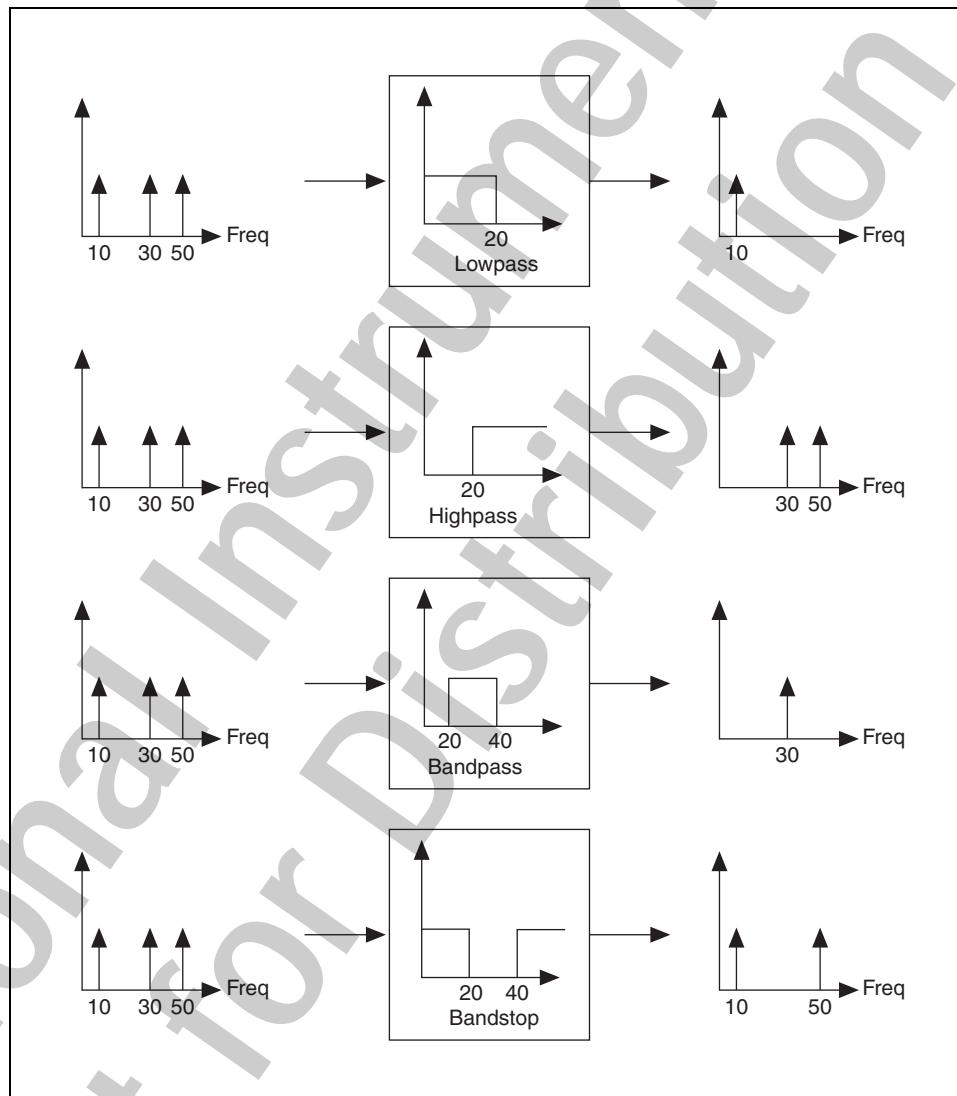


Figure B-25. Filter Outputs

J. Practical (Nonideal) Filters

Transition Band

Ideally, a filter should have a unit gain (0 dB) in the passband and a gain of zero ($-\infty$ dB) in the stopband. However, in a real implementation, not all of these criteria can be fulfilled. In practice, a finite transition region always occurs between the passband and the stopband. In this region, the gain of the filter changes gradually from one (0 dB) in the passband to zero ($-\infty$ dB) in the stopband. Figure B-26 shows the passband, the stopband, and the transition region (TR) for the different types of nonideal filters. The passband becomes the frequency range within which the gain of the filter varies from 0 dB to -3 dB. Although the -3 dB range is most common, depending on the application, other values (-0.5 dB, -1 dB, and so on) also might be used.

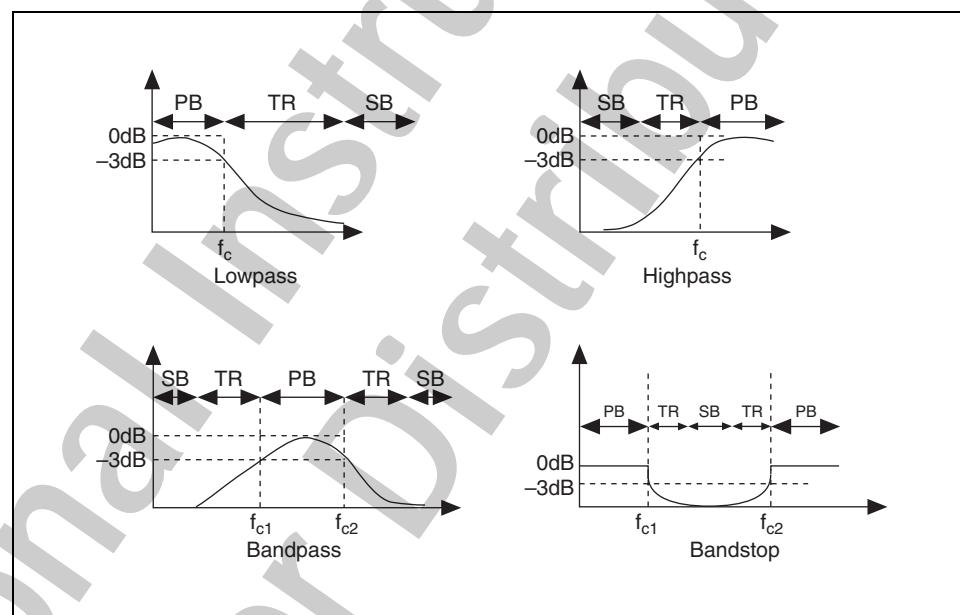


Figure B-26. Transition Bands

Passband Ripple and Stopband Attenuation

In many applications, it is acceptable to allow the gain in the passband to vary slightly from unity. The variation in the passband is called the passband ripple, which is the difference between the actual gain and the desired gain of unity. The stopband attenuation cannot be infinite, and you must specify the value you want. Both the passband ripple and the stopband attenuation are measured in decibels or dB, defined by:

$$\text{dB} = 20 \times \log_{10}(A_o(f)/A_i(f))$$

where \log_{10} denotes the logarithm to the base 10, and $A_i(f)$ and $A_o(f)$ are the amplitudes of a particular frequency f before and after the filtering, respectively.

For example, for -0.02 dB passband ripple, the formula gives

$$-0.02 = 20 \times \log_{10}(A_o(f)/A_i(f))$$

$$A_o(f)/A_i(f) = 10^{-0.001} = 0.9977$$

which shows that the ratio of input and output amplitudes is close to unity.

If you have -60 dB attenuation in the stopband, you have

$$-60 = 20 \times \log_{10}(A_o(f)/A_i(f))$$

$$A_o(f)/A_i(f) = 10^{-3} = 0.001$$

which means the output amplitude is $1/1000$ of the input amplitude. Figure B-27, though not drawn to scale, illustrates this concept.

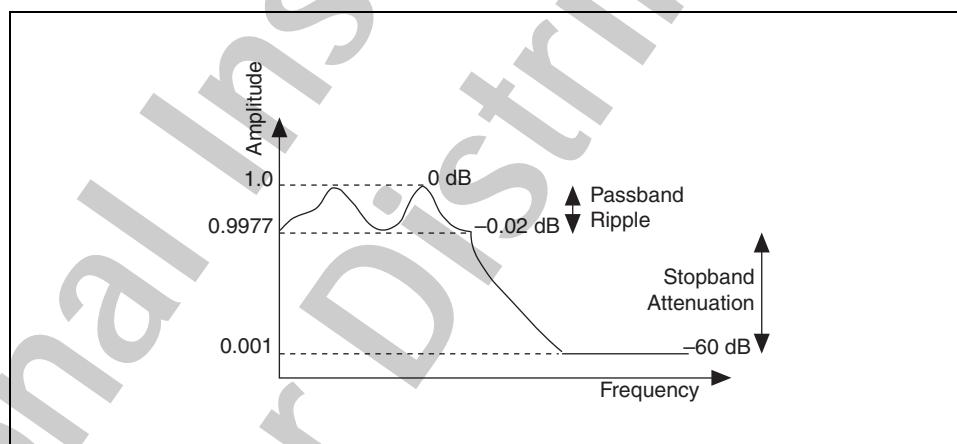


Figure B-27. Passband Ripple



Note Attenuation is usually expressed in decibels without the word minus, but a negative dB value is normally assumed.

K. Advantages of Digital Filters over Analog Filters

An analog filter has an analog signal at both its input and its output. Both the input, $x(t)$, and output, $y(t)$, are functions of a continuous variable t and can take on an infinite number of values. Analog filter design is about 50 years older than digital filter design. Many analog filter design books featuring simple, well-tested filter designs are available. However, this type of filter design is often reserved for specialists because it requires advanced

mathematical knowledge and understanding of the processes involved in the system that affects the filter.

Modern sampling and digital signal processing tools make it possible to replace analog filters with digital filters in applications that require flexibility and programmability. These applications include audio, telecommunications, geophysics, and medical monitoring. The following list explains the advantages of digital filters:

- They are software programmable and therefore easy to build and test.
- They require only the arithmetic operations of multiplication and addition/subtraction, so they are easier to implement.
- They are stable (they do not change with time or temperature) and predictable.
- They do not drift with temperature or humidity or require precision components.
- They have a superior performance-to-cost ratio.
- They do not suffer from manufacturing variations or aging.

L. IIR and FIR Filters

Another method of classification of filters is based on their impulse response. An impulse response is the response of a filter to an input that is an impulse ($x[0] = 1$ and $x[i] = 0$ for all $i \neq 0$), as shown in Figure B-28. The Fourier transform of the impulse response is known as the frequency response of the filter. The frequency response of a filter tells you what the output of the filter is going to be at different frequencies. The frequency response tells you the gain of the filter at different frequencies. For an ideal filter, the gain should be 1 in the passband and 0 in the stopband. All frequencies in the passband are passed without changes to the output, but there is no output for frequencies in the stopband.

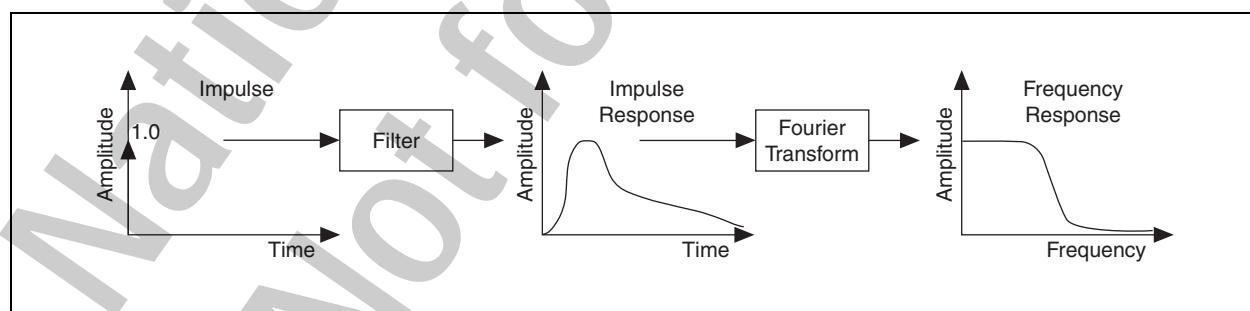


Figure B-28. Impulse Response

If the impulse response of the filter falls to zero after a finite amount of time, it is known as a finite impulse response (FIR) filter. If the impulse response exists indefinitely, it is known as an infinite impulse response (IIR) filter. If the impulse response is FIR or IIR depends on how the output is calculated.

The basic difference between FIR and IIR filters is that for FIR filters, the output depends only on the current and past input values; for IIR filters, the output depends not only on the current and past input values, but also on the past output values.

As an example, consider a cash register at a supermarket. Let $x[k]$ be the cost of the k^{th} item that a customer buys, where $1 \leq k \leq N$, and N is the total number of items. The cash register adds the cost of each item to produce a running total. This running total $y[k]$, up to the k^{th} item, is given by

$$y[k] = x[k] + x[k-1] + x[k-2] + x[k-3] + \dots + x[1] \quad (\text{B-2a})$$

Thus, the total for N items is $y[N]$. Because $y[k]$ is the total up to the k^{th} item, and $y[k-1]$ is the total up to the $(k-1)^{\text{st}}$ item, you can rewrite Equation B-2a as

$$y[k] = y[k - 1] + x[k] \quad (\text{B-2b})$$

If you add a sales tax of 8%, Equations B-2a and B-2b yield

$$\begin{aligned} y[k] &= 1.08x[k] + 1.08x[k - 1] \\ &+ 1.08x[k - 2] + 1.08x[k - 3] + \dots + 1.08x[1] \end{aligned} \quad (\text{B-3a})$$

$$y[k] = y[k - 1] + 1.08x[k] \quad (\text{B-3b})$$

Both Equations B-3a and B-3b are identical in describing the behavior of the cash register. The difference is that Equation B-3a is implemented only in terms of the inputs, but Equation B-3b is implemented in terms of both the input and the output. Equation B-3a is known as the nonrecursive, or FIR, implementation. Equation B-3b is known as the recursive, or IIR, implementation.

Filter Coefficients

In Equation B-3a, the multiplying constant for each term is 1.08. In Equation B-3b, the multiplying constants are 1 (for $y[k - 1]$) and 1.08 (for $x[k]$). These multiplying constants are known as the coefficients of the filter. For an IIR filter, the coefficients that multiply the inputs are forward coefficients, and those that multiply the outputs reverse coefficients.

Equations of the form B-2a, B-2b, B-3a, or B-3b that describe the operation of the filter are known as difference equations.

Advantages and Disadvantages of FIR and IIR Filters

The advantage of digital IIR filters over finite impulse response (FIR) filters is that IIR filters usually require fewer coefficients to perform similar filtering operations. Thus, IIR filters execute much faster and do not require extra memory because they execute in place.

The disadvantage of IIR filters is that the phase response is nonlinear. If the application does not require phase information, such as simple signal monitoring, IIR filters may be appropriate. Use FIR filters for applications that require linear phase responses. The recursive nature of IIR filters makes them difficult to design and implement.

M. Infinite Impulse Response Filters

IIR filters are digital filters whose output is calculated by adding a weighted sum of past output values with a weighted sum of past and current input values. Denoting the input values by $x[.]$ and the output values by $y[.]$, the general difference equation characterizing IIR filters is

$$\begin{aligned} a_0y[i] + a_1y[i-1] + a_2y[i-2] + \dots + a_{N_y-1}y[i-(N_y-1)] &= \\ b_0x[i] + b_1x[i-1] + b_2x[i-2] + \dots + b_{N_x-1}x[i-(N_x-1)] \\ a_0y[i] = -a_1y[i-1] - a_2y[i-2] + \dots - a_{N_y-1}y[i-(N_y-1)] &+ \\ b_0x[i] + b_1x[i-1] + b_2x[i-2] + \dots + b_{N_x-1}x[i-(N_x-1)] \\ y[i] = \frac{1}{a_0} \left(-\sum_{j=1}^{N_y-1} a[j]y[i-j] + \sum_{k=0}^{N_x-1} b[k]x[i-k] \right) \end{aligned} \quad (\text{B-4})$$

where N_x is the number of forward coefficients ($b[k]$), and N_y is the number of reverse coefficients ($a[j]$). The output sample at the present sample index i is the sum of scaled present and past inputs ($x[i]$ and $x[i-k]$ when $j \neq 0$) and scaled past outputs ($y[i-j]$). Usually, N_x is equal to N_y , and this value is known as the order of the filter.



Note In all of the IIR filters implemented in LabVIEW, the coefficient a_0 is 1.

Practical IIR Filters

A lower order reduces arithmetic operations and therefore reduces computation error. A problem with higher order filtering is that you quickly run into precision errors with orders much greater than 20–30. This is the main reason for the cascade implementations over the direct form. Refer to the *LabVIEW Help* for more information about cascade

form implementations. The orders of 1–20 are reasonable, with 30 being an upper limit. A higher order also means more filter coefficients and longer processing time.

The impulse response of the filter described by Equation B-4 is of infinite length for nonzero coefficients. In practical filter applications, however, the impulse response of stable IIR filters decays to near zero in a finite number of samples.

In practice, the frequency response of filters differs from that of ideal filters. Depending on the shape of the frequency response, the IIR filters can be further classified into the following categories:

- Butterworth filters
- Chebyshev filters
- Chebyshev II or inverse Chebyshev filters
- Elliptic filters
- Bessel filters

In the following sections about each of these filters, the input signal is an impulse response. The signal is then filtered using the Filter Express VI and then the frequency response taken by using the Frequency Response Function (Mag-Phase) VI with no windowing applied.

Butterworth Filters

A Butterworth filter has no ripples in the passband or the stopband. Due to the lack of ripples, it is also known as the maximally flat filter. Its frequency response is characterized by a smooth response at all frequencies.

Figure B-29 shows the response of a lowpass Butterworth filter of different orders.

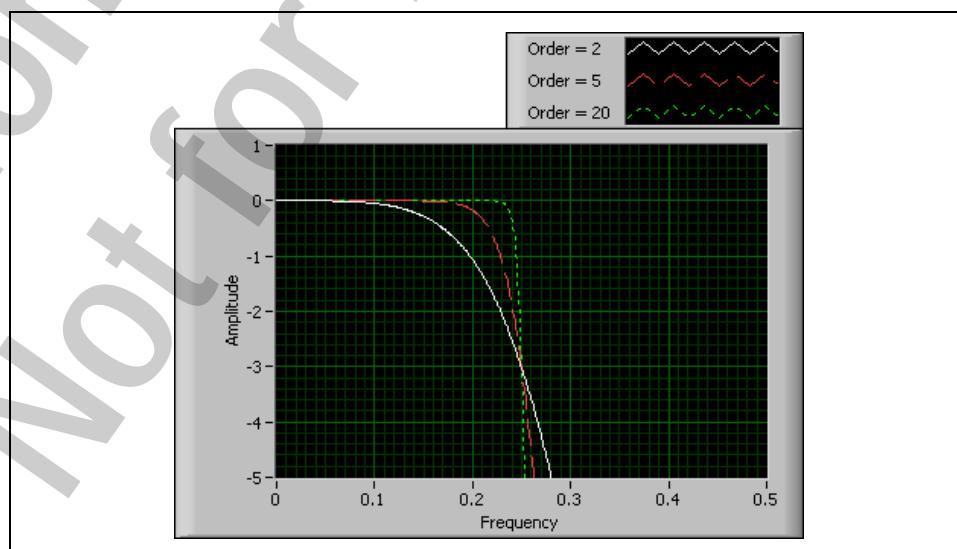


Figure B-29. Butterworth Response

The region where the output of the filter is equal to 0 or very close to 0 is the passband of the filter. The region where the output approaches the negative amplitudes is the stopband. The region in between the passband and the stopband where the output gradually changes from 0 to negative amplitudes is the transition region.

The advantage of Butterworth filters is a smooth, monotonically decreasing frequency response in the transition region. As seen in Figure B-29, the higher the filter order, the steeper the transition region.

Chebyshev Filters

The frequency response of Butterworth filters is not always an accurate approximation of the ideal filter response because of the slow rolloff between the passband, also known as the portion of interest in the spectrum, and the stopband, also known as the unwanted portion of the spectrum.

Chebyshev filters have a smaller transition region than a Butterworth filter of the same order. However, this is achieved at the expense of ripples in the passband. You can use LabVIEW to specify the maximum amount of ripple (in dB) in the passband for a Chebyshev filter. The frequency response characteristics of Chebyshev filters have an equiripple (ripples all have the same magnitude) magnitude response in the passband, monotonically decreasing magnitude response in the stopband, and a sharper rolloff in the transition region as compared to Butterworth filters of the same order.

Figure B-30 shows the response of a lowpass Chebyshev filter of different orders. In this graph, the y-axis scaling is in decibels. The steepness of the transition region increases with increasing order. The number of ripples in the passband also increases with increasing order.

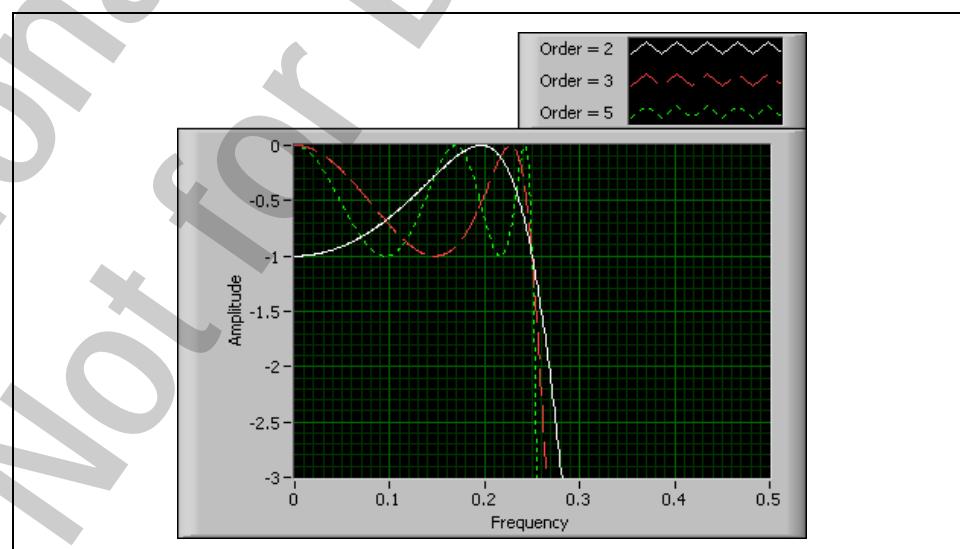


Figure B-30. Chebyshev Response

The advantage of Chebyshev filters over Butterworth filters is the sharper transition between the passband and the stopband with a lower order filter. This produces smaller absolute errors and higher execution speeds.

Chebyshev II or Inverse Chebyshev Filters

Chebyshev II, also known as inverse Chebyshev or Type II Chebyshev filters, are similar to Chebyshev filters except that Chebyshev II filters have ripples in the stopband as opposed to the passband and are maximally flat in the passband as opposed to the stopband. For Chebyshev II filters, you can specify the amount of attenuation (in dB) in the stopband. The frequency response characteristics of Chebyshev II filters are equiripple magnitude response in the stopband, monotonically decreasing magnitude response in the passband, and a rolloff sharper than Butterworth filters of the same order. Figure B-31 plots the response of a lowpass Chebyshev II filter of different orders.

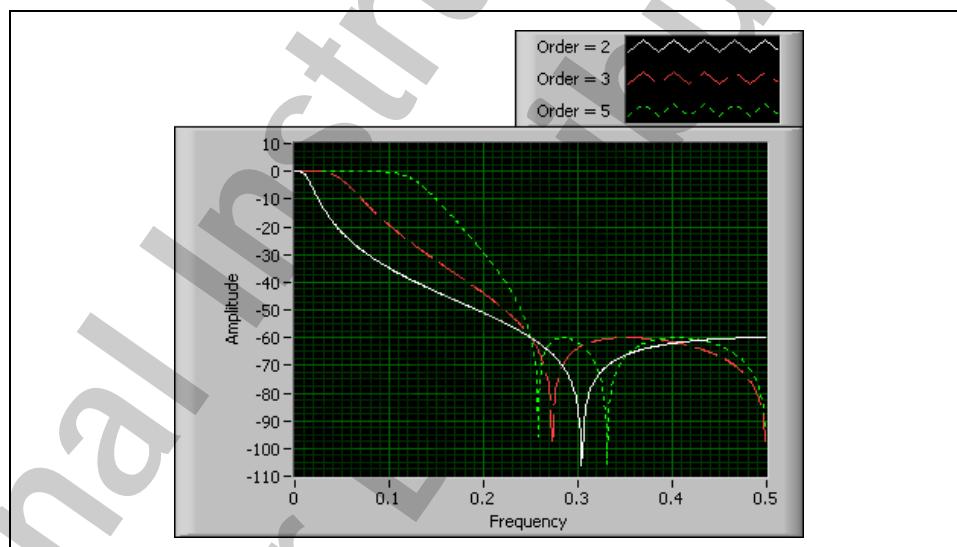


Figure B-31. Chebyshev II Response

The advantage of Chebyshev II filters over Butterworth filters is that Chebyshev II filters give a sharper transition between the passband and the stopband with a lower order filter. This difference corresponds to a smaller absolute error and higher execution speed. One advantage of Chebyshev II filters over regular Chebyshev filters is that Chebyshev II filters have the ripples in the stopband instead of the passband.

Elliptic Filters

Chebyshev type I and II filters have a sharper transition region than a Butterworth filter of the same order. This is because they allowed ripples in the passband (type I) or the stopband (type II). Elliptic filters distribute the ripples over the passband and the stopband. Equiripples in the passband and the stopband characterize the magnitude response of elliptic filters. Compared with the same order Butterworth or Chebyshev filters, the elliptic design provides the sharpest transition between the passband and the stopband. For this reason, elliptic filters are quite popular in applications in which short transition bands are required and in which ripples can be tolerated. Figure B-32 plots the response of a lowpass elliptic filter of different orders.

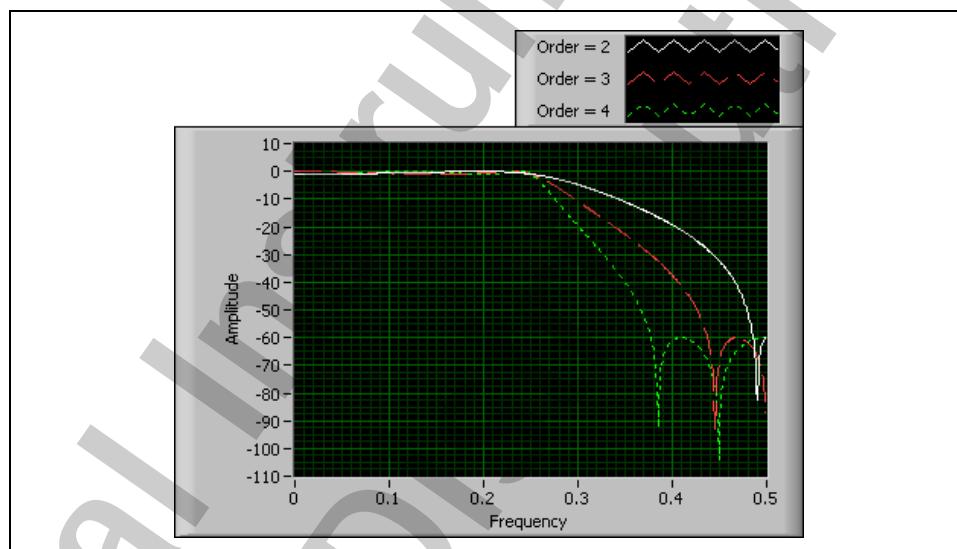


Figure B-32. Elliptic Response

Notice the sharp transition edge for even low-order elliptic filters. For elliptic filters, you can specify the amount of ripple (in dB) in the passband and the attenuation (in dB) in the stopband.

Bessel Filters

The Bessel filter was designed with a square wave in mind and is thus ideal for digital filtering. Much like the Butterworth filter, the Bessel filter has a smooth passband and stopband response. Using the same filter order, the stopband attenuation of the Bessel filter is much lower than that of the Butterworth filter. Of all the filter types, the Bessel filter has the widest transition region for a fixed order. The main advantage of the Bessel filter is that the phase response is nearly linear throughout the passband.

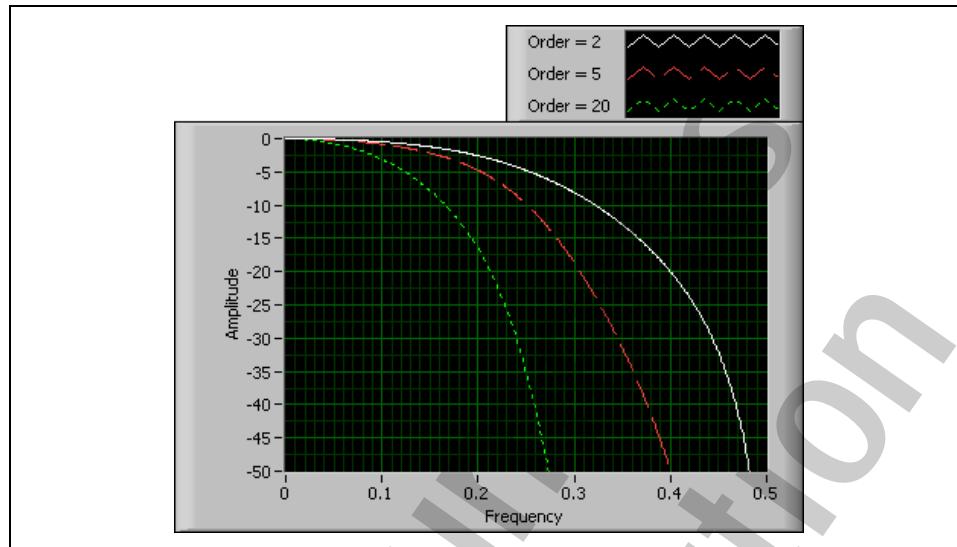


Figure B-33. Bessel Response

N. IIR Filter Comparison

Figure B-34 shows a comparison of the lowpass frequency responses for the five different IIR filter designs, all having the same order (five). The elliptic filter has the narrowest transition region, and the Bessel filter has the widest.



Figure B-34. IIR Filter Comparison

Table B-7 compares the filter types.

Table B-7. IIR Filter Comparison

IIR Filter Design	Response Characteristics	Width of Transition Region for a Fixed Order	Order Required for Given Filter Specifications
Butterworth	No ripples	—	—
Chebychev	Ripple in PB	—	—
Inverse Chebyshev	Ripple in SB	—	—
Elliptic	Ripples in PB and SB	Narrowest	Lowest
Bessel	No ripples	Widest	Highest

The LabVIEW digital filter VIs handle all the design issues, computations, memory management, and actual data filtering internally and are transparent to the user. You do not need to be an expert in digital filters or digital filter theory to process the data. You need only to specify the control parameters such as the filter order, cutoff frequencies, amount of ripple, and stopband attenuation.

Deciding Which Filter to Use

Now that you have seen the different types of filters and their characteristics, the question arises as to which filter design best suits for the application. Some of the factors affecting the choice of a suitable filter are if you require linear phase, if you can tolerate ripples, and if a narrow transition band is required. The flowchart in Figure B-35 provides a guideline for selecting the correct filter. You might need to experiment with several different options before finding the best one.

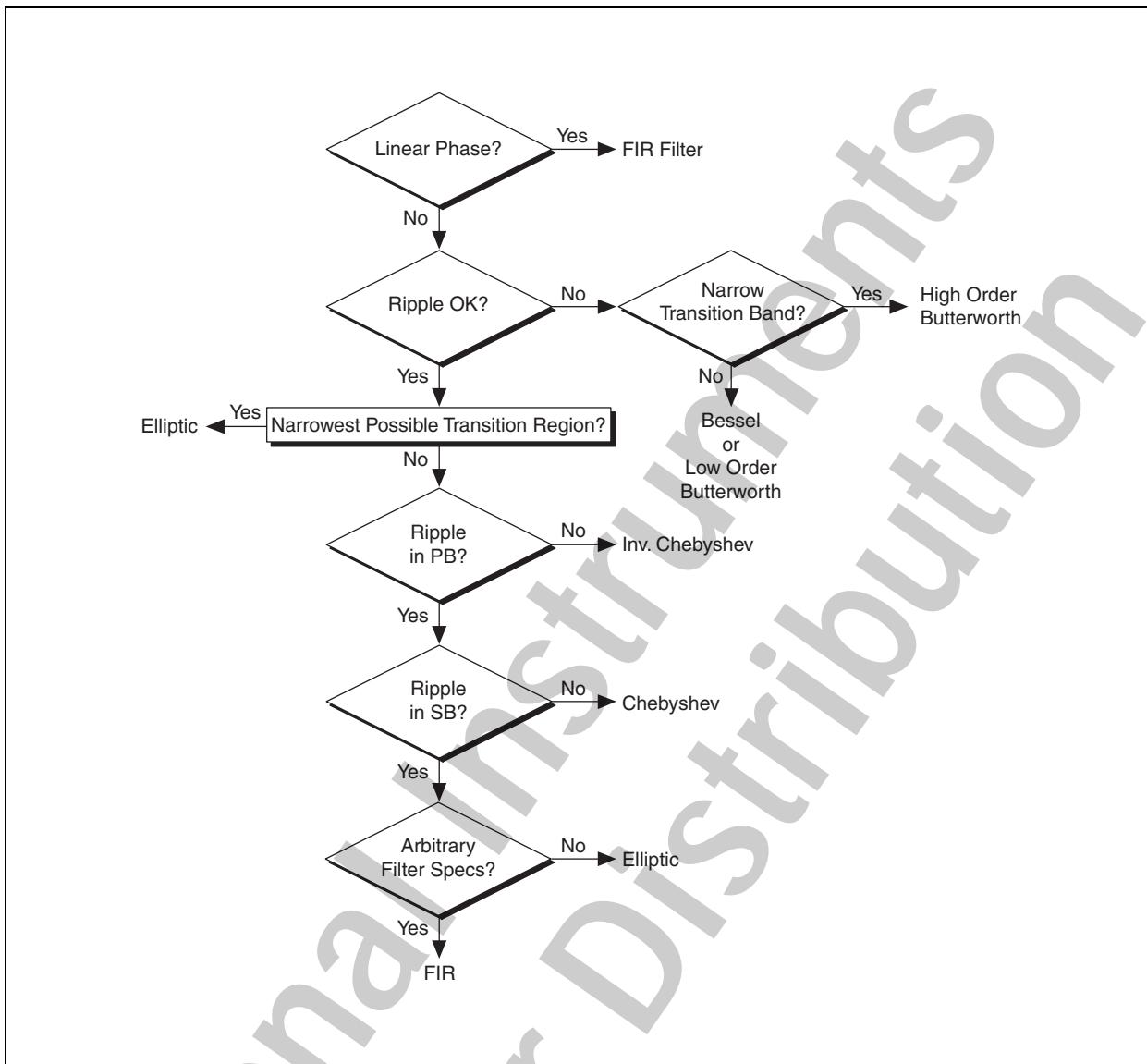


Figure B-35. Filter Flowchart

O. Transient Response of IIR Filters

The output of a general IIR filter is given by

$$y[i] = -a_1y[i-1] - a_2y[i-2] - \dots - a_{N_y-1}y[i-(N_y-1)] + b_0x[i] \quad (B-5) \\ + b_1x[i-1] + b_2x[i-2] + \dots + b_{N_x-1}x[i-(N_x-1)]$$

where N_x is the number of forward coefficients, N_y is the number of reverse coefficients, and a_0 is assumed to be equal to 1. With a second-order filter where $N_x = N_y = 2$, the corresponding difference equation is:

$$y[i] = -a_1y[i-1] - a_2y[i-2] + b_0x[i] + b_1x[i-1] + b_2x[i-2] \quad (B-6)$$

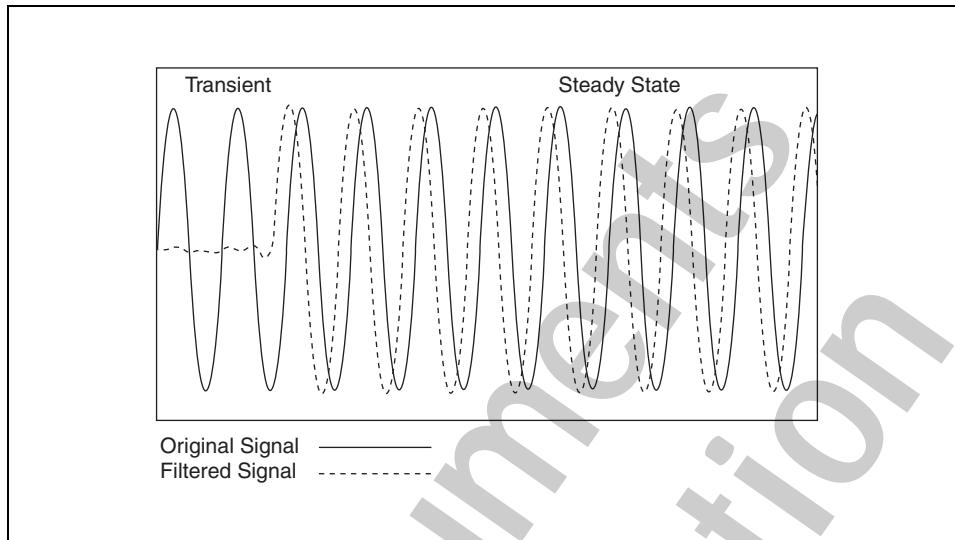
To calculate the current output (at the i^{th} instant) of the filter, you need to know the past two outputs (at the $(i-1)^{\text{st}}$ and the $(i-2)^{\text{nd}}$ time instants) and the current input (at the i^{th} time instant) and past two inputs (at the $(i-1)^{\text{st}}$ and $(i-2)^{\text{nd}}$ time instants).

Suppose you start the filtering process by taking the first sample of the input data. At this time instant, you do not have the previous inputs ($x[i-1]$ and $x[i-2]$) or previous outputs ($y[i-1]$ and $y[i-2]$). By default, these values are assumed to be zero. When you get the second data sample, you have the previous input ($x[i-1]$) and the previous output ($y[i-1]$) that you calculated from the first sample but not $x[i-2]$ and $y[i-2]$. Again, by default, these are assumed to be zero. After you start processing the third input data sample, all the terms on the right hand side of Equation B-6 have the previously calculated values. A certain amount of delay occurs before which calculated values are available for all the terms on the right hand side (RHS) of the difference equation that describes the filter. The output of the filter during this time interval is a transient and is known as the transient response. For lowpass and highpass filters, the duration of the transient response, or delay, is equal to the order of the filter. For bandpass and bandstop filters, this delay is $2 \times \text{order}$.

IIR filters contain the following properties.

- Negative indices that result from Equation B-4 are assumed to be zero the first time you call the VI.
- Because we assume the initial filter state to be zero (negative indices), a transient proportional to the filter order occurs before the filter reaches a steady state. The duration of the transient response, or delay, for lowpass and highpass filters is equal to the order of the filter.
- The duration of the transient response for bandpass and bandstop filters is twice the filter order.

Each time a Filter VI is called, this transient appears at the output, as illustrated in Figure B-36. You can eliminate this transient response on successive calls by enabling the state memory of the VI. To enable state memory, set the **init/cont** parameter of the VI to TRUE (continuous filtering).

**Figure B-36.** Filtered Signal

- The number of elements in the filtered sequence equals the number of elements in the input sequence.
- The filter retains the internal filter state values when the filtering completes.

P. Finite Impulse Response Filters

Because the output of an IIR filter depends on the previous outputs and the current and previous inputs, the IIR filter has infinite memory, resulting in an impulse response of infinite length.

Finite Impulse Response (FIR) filters are digital filters with a finite impulse response. FIR filters are also known as nonrecursive filters, convolution filters, or moving-average (MA) filters because you can express the output of a FIR filter as a finite convolution

$$y_i = \sum_{k=0}^{n-1} h_k x_{i-k}$$

where x represents the input sequence to be filtered, y represents the output filtered sequence, and h represents the FIR filter coefficients.

The output of an FIR filter depends only on the current and past inputs. Because it does not depend on the past outputs, its impulse response decays to zero in a finite amount of time. The output of a general FIR filter is given by

$$y[i] = b_0 x[i] + b_1 x[i-1] + b_2 x[i-2] + \dots + b_{M-1} x[i-(M-1)]$$

where M is the number of taps of the filter, and b_0, b_1, \dots, b_{M-1} are its coefficients.

FIR filters have some important characteristics:

- They can achieve linear phase response and pass a signal without phase distortion.
- They are always stable. During filter design or development, you do not need to worry about stability concerns.
- FIR filters are simpler and easier to implement than IIR filters.
- Figure B-37 plots a typical magnitude and phase response of FIR filters versus normalized frequency. The discontinuities in the phase response arise from the discontinuities introduced when you compute the magnitude response using the absolute value. The discontinuities in phase are on the order of π . The phase, however, is clearly linear.

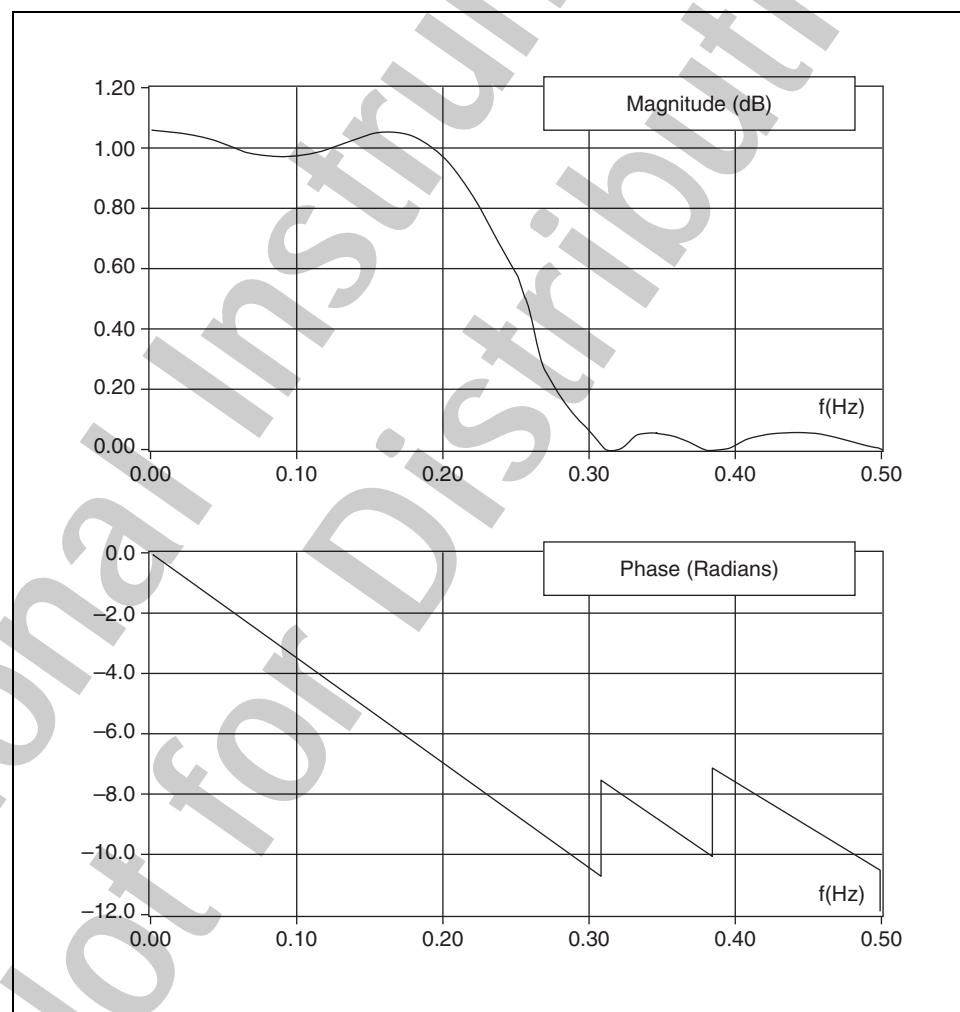


Figure B-37. Typical Magnitude and Phase Response of FIR Filters

The simplest method for designing linear-phase FIR filters is the window design method. To design a FIR filter by windowing, start with an ideal frequency response, calculate its impulse response, and then truncate the impulse response to produce a finite number of coefficients. The truncation of the ideal impulse response results in the effect known as the Gibbs phenomenon—oscillatory behavior near abrupt transitions (cutoff frequencies) in the FIR filter frequency response.

You can reduce the effects of the Gibbs phenomenon by smoothing the truncation of the ideal impulse response using a smoothing window. By tapering the FIR coefficients at each end, you can diminish the height of the side lobes in the frequency response. The disadvantage of this method is that the main lobe widens, resulting in a wider transition region at the cutoff frequencies.

The selection of a window function is similar to the choice between Chebyshev and Butterworth IIR filters in that it is a trade off between side lobes near the cutoff frequencies and the width of the transition region.

*National Instruments
Not for Distribution*

Summary

- Fast calculation of the Discrete Fourier Transform (DFT) is possible by using an algorithm known as the fast Fourier transform (FFT). You can use FFT algorithm when the number of signal samples is a power of two.
- The power spectrum can be calculated from the DFT/FFT by squaring the magnitude of the individual frequency components. The Power Spectrum VI in the advanced analysis library does this automatically for you. The Power Spectrum VI units of the output are V^2_{rms} . The power spectrum does not provide any phase information.
- The DFT, FFT, and power spectrum are useful for measuring the frequency content of stationary or transient signals. The FFT provides the average frequency content of the signal over the entire time that the signal was acquired. Use the FFT mostly for stationary signal analysis when the signal is not significantly changing in frequency content over the time that the signal is acquired or when you want only the average energy at each frequency line.
- To reduce the spectral leakage, the finite time waveform is multiplied by a window function.
- Windows can be used to separate two sine waves that have widely different amplitudes but are very close in frequency.
- For practical filters, the gain in the passband may not always be equal to 1, the attenuation in the stopband may not always be $-x$, and there exists a transition region of finite width.
- The width of the transition region depends on the filter order, and the width decreases with increasing order.
- The output of FIR filters depends only on the current and past input values, but the output of IIR filters depends on the current and past input values and the past output values.
- IIR filters can be classified according to the presence of ripples in the passband and/or the stopband.
- Because of the dependence of its output on past outputs, a transient appears at the output of an IIR filter each time the VI is called. This transient can be eliminated after the first call to the VI by setting its **init/cont** control to TRUE.
- Use the Spectral Measurements Express VI and Filter Express VI to easily configure spectral calculations and filtering options for a given input signal.

Notes

National Instruments
Not for Distribution

Notes

National Instruments
Not for Distribution

Course Slides

This appendix contains the Course Slides.

Topics

- A. Overview of a DAQ System
- B. Data Acquisition Hardware and Software
- C. Analog Input
- D. Analog Output
- E. Digital I/O
- F. Counters
- G. Signal Conditioning
- H. Synchronization

National Instruments
Not for Distribution

Lesson 1 Overview of a Data Acquisition System

TOPICS

- A. DAQ System Overview
- B. Sensors
- C. Signals
- D. DAQ Hardware
- E. Signal Conditioning
- F. DAQ Software



ni.com/training

A. DAQ System Overview

Data Acquisition (DAQ)—the automatic collection of data from sensors, instruments, and devices in a factory, laboratory, or in the field.

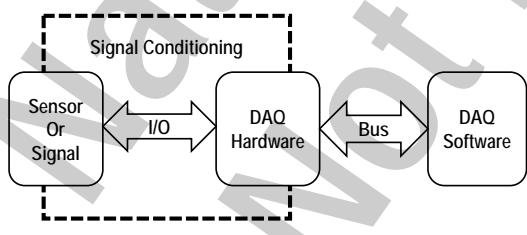
Purpose

To measure an electrical or physical phenomenon such as voltage, current, temperature, pressure, or sound



ni.com/training

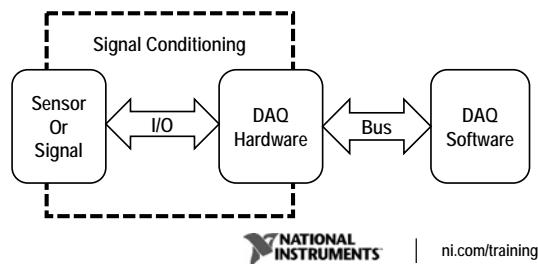
DAQ System Overview



ni.com/training

B. Sensor Overview

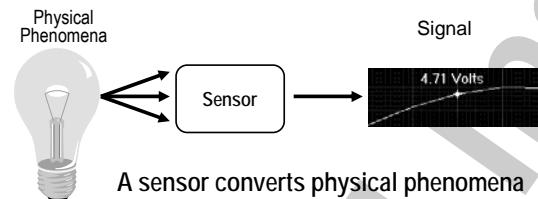
- What is a sensor?
- Types of sensors



NATIONAL INSTRUMENTS

ni.com/training

What is a Sensor?



A sensor converts physical phenomena into measurable electrical signals

NATIONAL INSTRUMENTS

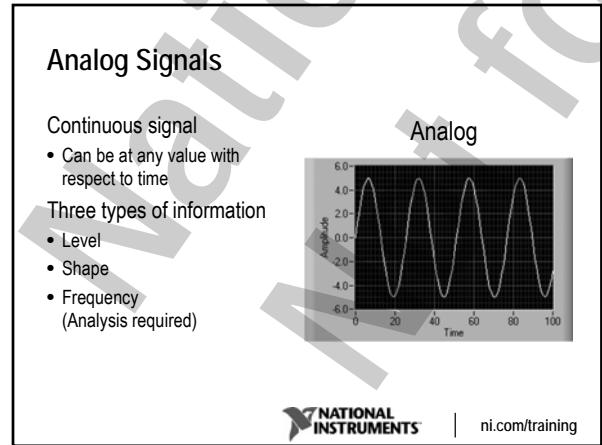
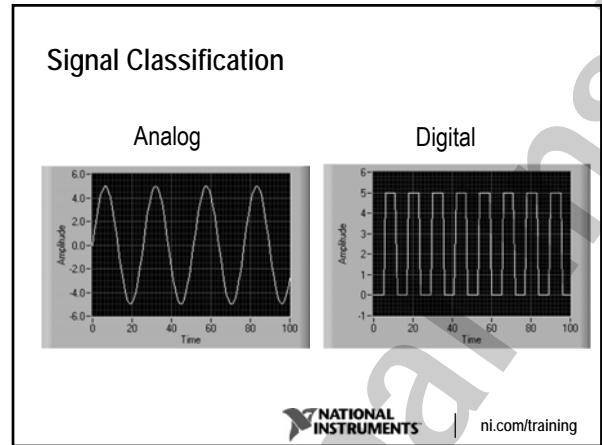
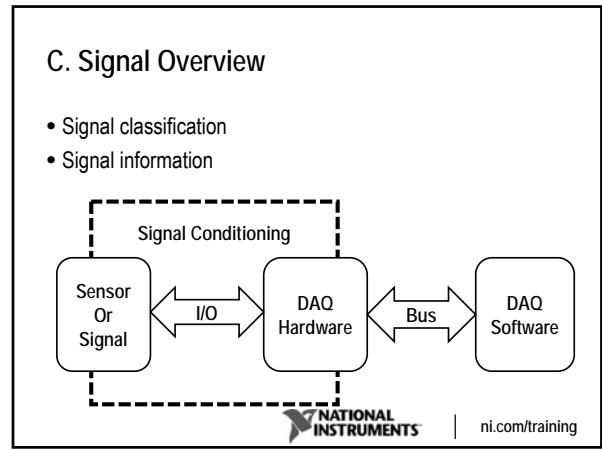
ni.com/training

Types of Sensors

Phenomena	Sensors
Temperature	Thermocouples Resistive Temperature Devices (RTDs) Thermistors
Strain and Pressure	Strain gages Piezoelectric transducers
Sound	Microphone
Vibration	Accelerometer
Position and Displacement	Potentiometers Linear voltage differential transformer Optical encoder
Fluid	Flow meters Rotameters
pH	pH electrodes
Light	Vacuum tube Photo sensors

NATIONAL INSTRUMENTS

ni.com/training



Analog Signal Information

4.71 Volts
Level

Shape

0.2 Hz
Frequency

Analog

ni.com/training

Analog Signal – Level Examples

4.71 Volts
Level

Common examples of level measurements

Voltage

Temperature

Pressure

Load

ni.com/training

Analog Signal – Shape Examples

Common examples of shape measurements

Shape

RC Circuit Response

Arterial Blood Pressure

ni.com/training

Analog Signal – Frequency Examples

Common examples of frequency measurements

Analysis Required

NATIONAL INSTRUMENTS | ni.com/training

Digital Signals

Two possible levels:
• High/On
• Low/Off

Two types of information:
• State
• Rate

Digital Signal Example

NATIONAL INSTRUMENTS | ni.com/training

Digital Signal Information

Digital

Amplitude

Time

OFF

ON

State

Rate

NATIONAL INSTRUMENTS | ni.com/training

Digital Signal – State Example

Position of the switch determines the state of the signal

NATIONAL INSTRUMENTS | ni.com/training

Digital Signal – Rate Example

- Shaft spins
- Encoder converts rotation into two digital pulse trains
- Measure the rate of the pulse train

24 Pulses/rev

NATIONAL INSTRUMENTS | ni.com/training

5 Ways to Measure the Same Signal

5.0 Volts
Level

Rise Time = 1msec
Shape

0.2 Hz
Frequency

Your Signal

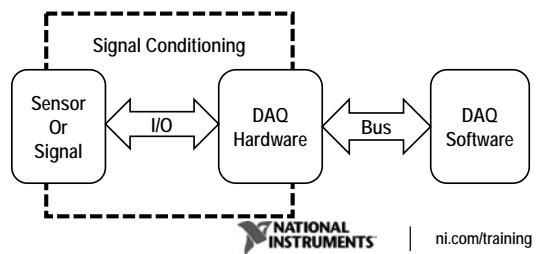
ON
OFF
State

Rate

NATIONAL INSTRUMENTS | ni.com/training

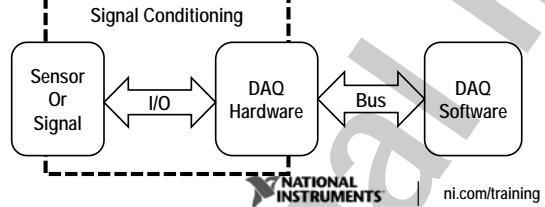
D. DAQ Hardware Overview

- Purpose of DAQ hardware
 - Transfer data between your sensor/signal and your software



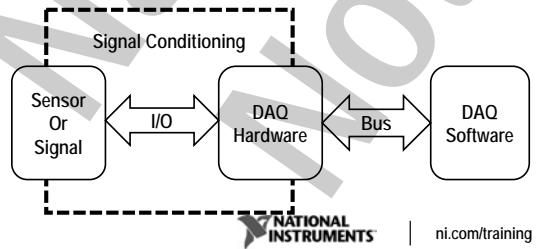
DAQ Hardware Overview

- DAQ hardware
 - Can both acquire and generate analog and digital signals
 - Transfers signals to and from DAQ software through a bus (PCI, PCIe, PXI, PXIe, USB, etc)



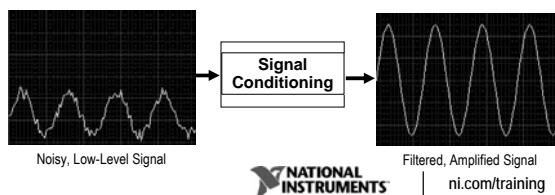
E. Signal Conditioning Overview

- Purpose of signal conditioning
- Where signal conditioning occurs
- Signal conditioning examples



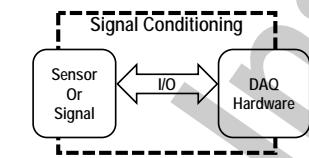
Purpose of Signal Conditioning

- Signal conditioning takes a signal that is difficult for your DAQ device to measure and makes it easier to measure
- Signal conditioning is not always required
 - Depends on the sensor or signal being measured



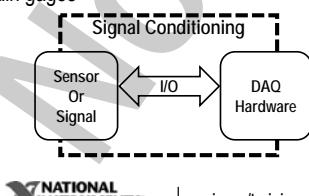
Where Signal Conditioning Occurs

- Can occur in:
 - Sensor
 - Path between the sensor and DAQ hardware
 - DAQ hardware



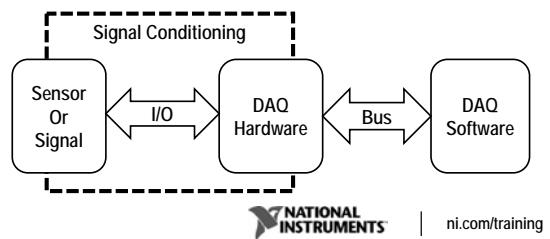
Signal Conditioning Example

- Strain gage
 - Needs to receive excitation voltage
 - Outputs a low voltage
- Signal conditioning for strain gages
 - Provide excitation voltage
 - Complete bridge circuit
 - Amplify the signal
 - Filter out noise



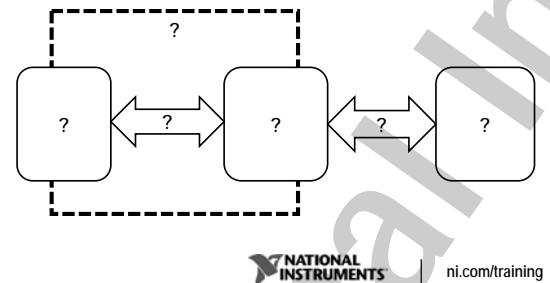
F. DAQ Software Overview

- After acquiring data, you usually still need to do more
 - Signal processing, generate a report, interact with data, etc.



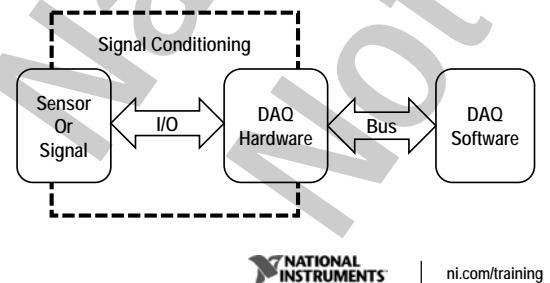
Summary—Quiz

1. List the components of a Data Acquisition System.



Summary—Quiz

1. List the components of a Data Acquisition System.



Summary-Matching Quiz

- | | |
|------------------------|---|
| 1. Sensor | a) Takes a signal that is difficult for your DAQ device to measure and makes it easier to measure |
| 2. Signal Conditioning | b) Transfers signals to and from software through a bus |
| 3. DAQ Hardware | c) Operates on data after it has been acquired |
| 4. DAQ Software | d) Converts physical phenomena into measurable electrical signals |



ni.com/training

Summary-Matching Quiz Answer

- | | |
|------------------------|---|
| 1. Sensor | a) Takes a signal that is difficult for your DAQ device to measure and makes it easier to measure |
| 2. Signal Conditioning | b) Transfers signals to and from software through a bus |
| 3. DAQ Hardware | c) Operates on data after it has been acquired |
| 4. DAQ Software | d) Converts physical phenomena into measurable electrical signals |



ni.com/training

Summary-Quiz

2. Name the 3 types of measurements that can be made of analog signals.



ni.com/training

Summary-Quiz Answer

2. Name the 3 types of measurements that can be made of analog signals.
- Level
 - Shape
 - Frequency



ni.com/training

Summary-Quiz

3. Name the 2 types of measurements that can be made from digital signals.



ni.com/training

Summary-Quiz Answer

3. Name the 2 types of measurements that can be made from digital signals.
- State
 - Rate



ni.com/training

Lesson 2
Data Acquisition Hardware and Software

TOPICS

A. DAQ Hardware Overview	D. DAQ Software Overview
B. Components of a DAQ Device	E. Overview of NI-DAQmx VIs
C. Choosing Appropriate DAQ Hardware	

NATIONAL INSTRUMENTS | ni.com/training

A. DAQ Hardware Overview

- Hardware setup
- Components of a DAQ device
- Connection types

NATIONAL INSTRUMENTS | ni.com/training

Why Use NI Hardware?

- Tight integration between hardware and software
- High-speed Application-Specific Integrated Circuits (ASIC)
- Extensive price/performance offerings on a wide variety of platforms
- Widest range of possible measurement types
- NIST-traceable calibration for accurate repeatable measurements

NATIONAL INSTRUMENTS | ni.com/training

Data Acquisition Hardware

DAQ Hardware turns your PC into a measurement and automation system

The diagram illustrates the DAQ hardware architecture. It starts with a 'Sensors' block containing 'Any type' of sensors. An arrow points to a 'Signal' block, which then leads to a 'Signal Connector' block. This connector can be connected directly or through a 'Terminal Block'. Another arrow points to a 'DAQ Device' block, which supports 'PCI/PCIe', 'PCIePXI', and 'USB'. A final arrow points to a 'Computer' block, which runs 'Windows', 'Linux', or 'Mac (DAGmx, Usb)'. The National Instruments logo is at the bottom left, and the URL 'ni.com/training' is at the bottom right.

NATIONAL INSTRUMENTS | ni.com/training

Signal Connector

Route your signal to specific lines on your DAQ device

PCI/PCIe

- Requires
 - Terminal block for connections to the sensors
 - Cable to connect the DAQ device and terminal block

USB

- Signals connect directly to the DAQ device
- Often has sensor-specific connector
 - BNC
 - RJ-50
 - Dsub

The image shows two types of DAQ hardware: a large black PCI/PCIe module with multiple connectors and a smaller white USB device. The National Instruments logo is at the bottom left, and the URL 'ni.com/training' is at the bottom right.

NATIONAL INSTRUMENTS | ni.com/training

BNC-2120 Shielded Connector Block

A photograph of the BNC-2120, a vertical shielded connector block with numerous ports for various signal types, including BNC, SMA, and analog inputs/outputs.

- Quadrature Encoder
- 8 LEDs for Digital I/O
- Counter I/O
- Function Generator
- Function Generator Frequency and Amplitude Control
- Temperature Sensor
- Analog Input
- Analog Output

NATIONAL INSTRUMENTS | ni.com/training

DAQ Device

DAQ devices connect to the bus of your computer

Most DAQ devices have:

- Analog Input
- Analog Output
- Digital I/O
- Counters

Specialty devices exist for specific applications

- High speed digital I/O
- High speed waveform generation
- Dynamic Signal Acquisition (vibration, sonar)

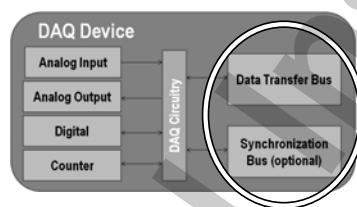


ni.com/training

B. Components of a DAQ Device

Data Transfer Bus

- Connects the DAQ device to the computer
- Can be a variety of bus structures
 - USB, PCI, PCI Express, PXI, PXI Express



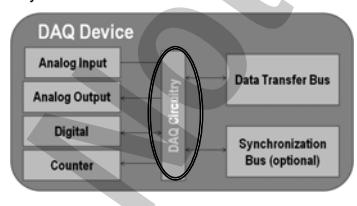
ni.com/training

Components of a DAQ Device

DAQ Circuitry

Contains all circuitry necessary for completing a DAQ task

- Clock and timing circuitry
- On-board FIFOs
- Signal routing
- Precision rails for calibration



ni.com/training

Components of a DAQ Device

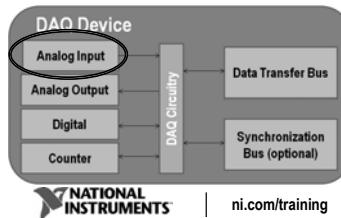
Analog Input Circuitry

Some signal conditioning

- Multiplexing of signals
 - Switch that has multiple input channels but only lets one channel at a time through to the instrumentation amplifier
- Instrumentation Amplifier
 - Either amplifies or attenuates your signal

Analog-to-Digital Converter (ADC)

- Converts an analog signal to a digital number

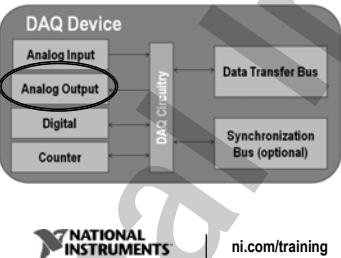


Components of a DAQ Device

Analog Output Circuitry

Digital-to-Analog Converter (DAC)

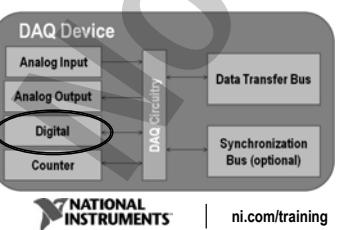
- Converts digital number to analog signal
- Usually one DAC per channel



Components of a DAQ Device

Digital I/O Circuitry

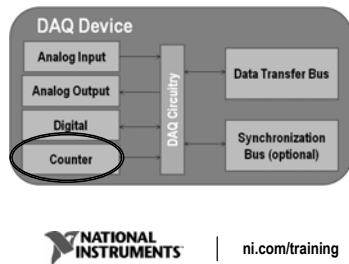
- Can input or output digital signals



Components of a DAQ Device

Counter Circuitry

- Can input or output digital signals
- Suitable for measuring rate
 - Built in timing signals
 - Counting functionality



ni.com/training

C. Choosing Appropriate DAQ Hardware

- Bus Considerations
- Signal Considerations
- Accuracy Considerations



ni.com/training

Bus Considerations

- How much data will I be streaming across this bus?
 - Bus bandwidth
- What are my single-point I/O requirements?
 - Bus latency and determinism
- Do I need to synchronize multiple devices?
 - Bus synchronization options
- How portable should this system be?
- How far will my measurements be from my computer?



ni.com/training

Bus Considerations

Bus	Waveform Streaming	Single-Point I/O	Multi-Device Synchronization	Portability	Distributed I/O
PCI	132 MB/s (shared)	Best	Better	Good	Good
PCI Express	250 MB/s (per lane)	Best	Better	Good	Good
PXI	132 MB/s (shared)	Best	Best	Better	Better
PXI Express	250 MB/s (per lane)	Best	Best	Better	Better
USB	60 MB/s	Better	Good	Best	Better
Ethernet	12.5 MB/s	Good	Good	Best	Best
Wireless	6.75 MB/s	Good	Good	Best	Best

Signal Considerations

- How many channels?
 - Choose DAQ device(s) with enough channels
- How quickly do you need to acquire/generate samples of the signal?
 - Choose DAQ device with fast enough sampling rate
- What are the expected minimum and maximum measurements?
 - Choose DAQ device with appropriate range



ni.com/training

Signal Considerations

- What is the smallest change in your signal that you need to detect?
 - Choose DAQ device with a small enough code width
 - To calculate the code width, you must know:
 - Resolution
 - Device input range



ni.com/training

Calculating Code Width – Resolution

- Resolution
 - Number of bits the ADC uses to represent a signal
- Resolution determines how many different voltage changes can be measured
- Example: 16-bit resolution

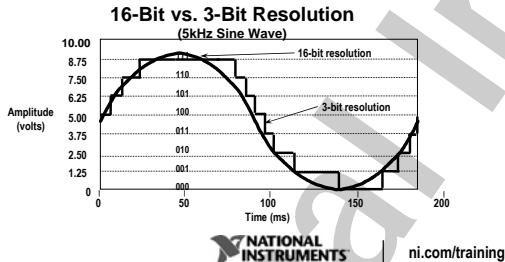
$$\text{# of levels} = 2^{\text{resolution}} = 2^{16} = 65,536 \text{ levels}$$
- Larger resolution = more precise representation of your signal



ni.com/training

Calculating Code Width – Resolution Example

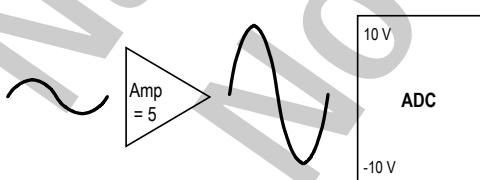
- 3-bit resolution can represent 8 voltage levels
- 16-bit resolution can represent 65,536 voltage levels



ni.com/training

Calculating Code Width – Amplification and Device Input Range

- DAQ devices have a built-in amplifier
 - Amplifies the signal to better fit the range of the ADC
 - Better utilizes the ADC resolution



ni.com/training

Calculating Code Width – Amplification and Device Input Range

- The amplification gains available on the DAQ device determine the device input ranges available
 - Example device input ranges include ± 10 , ± 5 , ± 2 , ± 1 , ± 0.5 , ± 0.2 , ± 0.1 V

NATIONAL INSTRUMENTS | ni.com/training

Calculating Code Width – Amplification and Device Input Range

- You do not set the amplification gains or device input ranges directly
- You set your minimum and maximum expected values in software
 - DAQ device automatically chooses which device input range to use based on your min/max settings
 - Min=-3V, Max=3.5V $\rightarrow \pm 5$ V device input range
 - Min=-9V, Max=8V $\rightarrow \pm 10$ V device input range
- Setting proper min/max
 - More precise representation of your signal
 - Utilizes all of your available resolution

NATIONAL INSTRUMENTS | ni.com/training

Calculating Code Width

Code width is the smallest change in the signal that your system can detect (determined by resolution and device input range)

$$\text{code width} = \frac{\text{Device input range}}{2^{\text{resolution}}}$$

Smaller Code Width = more precise representation of your signal

Example: 16-bit device, device input range ± 10 V

$$\text{Device input range} = \frac{10 - (-10)}{2^{16}} = 305 \mu\text{V}$$

Use smaller device input range: $\frac{5 - (-5)}{2^{16}} = 153 \mu\text{V}$

Use smaller device input range and use device with higher resolution: $\frac{5 - (-5)}{2^{18}} = 38 \mu\text{V}$

NATIONAL INSTRUMENTS | ni.com/training

Accuracy Considerations

- How close to the true value does your measurement need to be?
 - Make sure your DAQ device has an acceptable absolute accuracy
 - Absolute accuracy defines the overall uncertainty of your measurement
- Accuracy considerations
 - Code Width \neq Accuracy
 - Sources of error affecting accuracy
 - Gain errors and offset errors from amplifier and ADC
 - Noise in the system



ni.com/training

Accuracy Considerations

- Use the specifications manual of the DAQ device
 - Lists the absolute accuracy for each device input range
 - Lists absolute accuracy equation and numbers for each component if you want to calculate it yourself
- Example AI Absolute Accuracy table
 - When this DAQ device is using a ± 2 V device input range, the absolute accuracy of the measurement will be within $\pm 410 \mu\text{V}$ of the true value

Nominal Range		Absolute Accuracy at Full Scale ^a (μV)
Positive Full Scale	Negative Full Scale	
10	-10	1,020
5	-5	1,010
2	-2	410
1	-1	220
0.5	-0.5	130
0.2	-0.2	74
0.1	-0.1	52



ni.com/training

Exercise 2-1: Device Input Range, Resolution, Code Width, and Accuracy

To determine the optimal configuration for a data acquisition measurement system.

GOAL

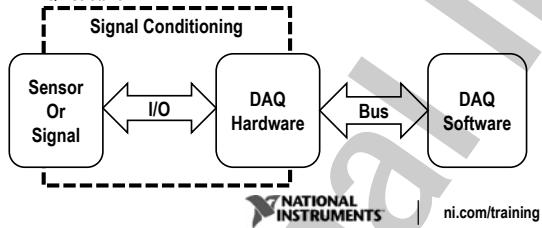
Exercise 2-1: Device Input Range, Resolution, Code Width, and Accuracy

- If you increase the resolution of your DAQ device, what happens to the code width?
- If you increase the device input range, what happens to code width?

DISCUSSION

D. DAQ Software Overview

- NI-DAQmx Software Architecture
- NI-DAQmx Overview
- Measurement & Automation Explorer (MAX) Overview
- DAO Assistant



NI-DAQmx Software Architecture



What is NI-DAQmx?

- Driver-level software
 - DLL that makes direct calls to your DAQ device
- NI-DAQmx does not support 3rd-party data acquisition devices
- Supports the following National Instruments software:
 - LabVIEW
 - Measurement Studio
 - Signal Express
 - LabWindows/CVI
 - LabVIEW Real-Time Module



ni.com/training

NI-DAQmx Platform Support

Also supports the following 3rd party languages:

- Microsoft Visual Basic .NET
- Microsoft Visual Basic 6.0
- Microsoft Visual C/C++
- Microsoft C# .NET
- ANSI C



ni.com/training

Benefits of NI-DAQmx

- DAQ Assistant
- Increased performance – faster single point I/O and multithreading
- Simple and intuitive API
- DAQ property nodes and waveform support
- Similar API for all programming languages
- Run NI-DAQmx programs and Assistant without the hardware!!
- Reduces development time with its interactive features



ni.com/training

What is MAX?

- MAX stands for Measurement & Automation Explorer
- MAX provides access to all your National Instruments DAQ, GPIB, IMAQ, IVI, Motion, VISA, CAN, Modular Instruments, PXI, and VXI devices
- Used for configuring and testing devices
 - Data Neighborhood
 - Devices and Interfaces
 - Historical Data
 - Scales
 - Software
 - VI Logger Tasks
 - IVI Drivers
 - Remote Systems

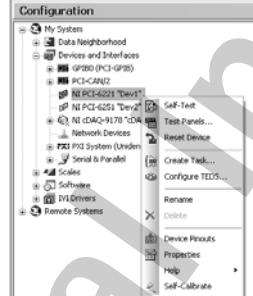


Measurement & Automation

NATIONAL INSTRUMENTS | ni.com/training

Devices and Interfaces

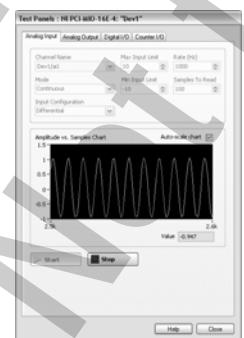
- Shows currently installed and detected National Instruments hardware
- Includes utilities for configuring and testing your DAQ devices
 - Self-Test
 - Test Panels
 - Reset
 - Properties
 - Self-Calibrate



NATIONAL INSTRUMENTS | ni.com/training

Test Panels

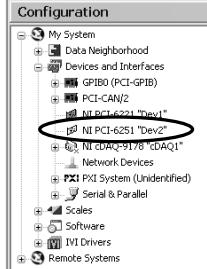
- Utility for testing
 - Analog Input
 - Analog Output
 - Digital I/O
 - Counter I/O
- Great tool for troubleshooting



NATIONAL INSTRUMENTS | ni.com/training

NI-DAQmx Simulated Devices

- Run NI-DAQmx programs and Assistant without the hardware!
- Assistant and programs run just like on a real device with some exceptions:
 - Timing and triggering are instantaneous
 - Reads return simulated data (for AI, data is a sine wave with some noise)
- Most DAQmx devices are supported (DAQmx plug-in devices, cDAQ, and more)



NATIONAL INSTRUMENTS | ni.com/training

Exercise 2-2: Using Measurement & Automation Explorer

To become familiar with the Devices and Interfaces section of MAX and to explore the test panel functionality.

GOAL

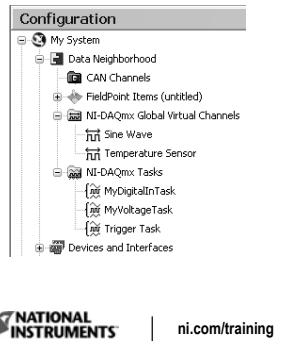
Exercise 2-2: Using Measurement & Automation Explorer

- How can you verify if the analog output channels on your DAQ device are outputting the correct voltages?

DISCUSSION

Data Neighborhood

- Provides access to DAQ Assistant
- Shows configured tasks and channels
- Includes utilities for testing and reconfiguring tasks and channels



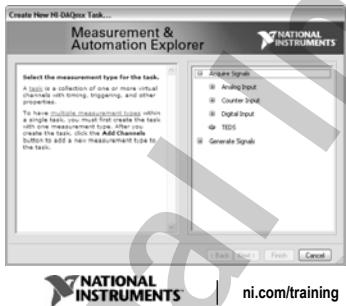
DAQ Assistant – Channels

Interface to create channels for:

- Analog Input
- Analog Output
- Counter Input
- Counter Output
- Digital I/O
- TEDS

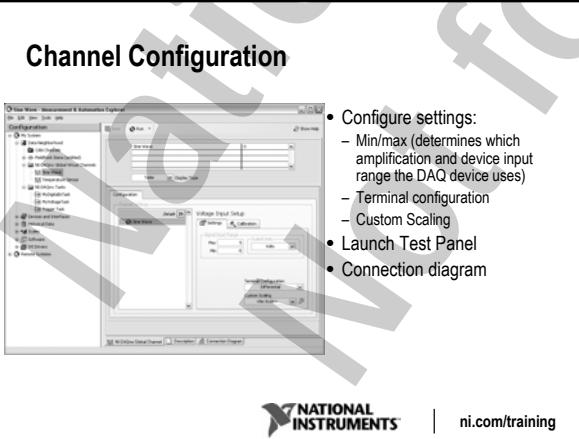
Each channel has:

- Measurement type
- Sensor/signal type
- Name



Channel Configuration

- Configure settings:
 - Min/max (determines which amplification and device input range the DAQ device uses)
 - Terminal configuration
 - Custom Scaling
- Launch Test Panel
- Connection diagram



DAQ Assistant – Tasks

- Task: A collection of channels with homogeneous timing and triggering
- Use new or existing channels

NATIONAL INSTRUMENTS | ni.com/training

Scope of Channels within a Task

- Local Channels: Can only be used in that particular task
- Global Channels: Can be used in multiple tasks and referenced outside the context of a task

NATIONAL INSTRUMENTS | ni.com/training

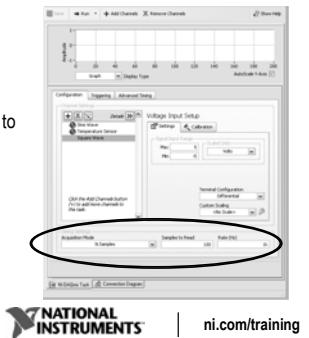
Task Configuration

- Configure settings for physical channels in the task
- Configure settings for global channels in Data Neighborhood>NI-DAQmx Global Channels

NATIONAL INSTRUMENTS | ni.com/training

Task Timing

- Configure task timing – single sample (On Demand), single sample (HW Timed), finite, or continuous
- Configure number of samples to read and sampling rate
- Select clock settings

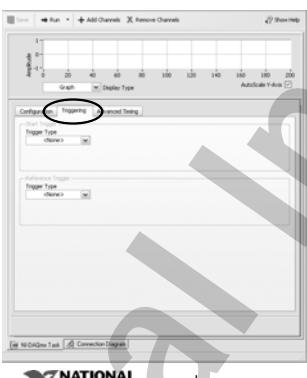


NATIONAL INSTRUMENTS

ni.com/training

Task Triggering

Configure start and reference triggers

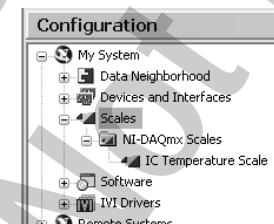


NATIONAL INSTRUMENTS

ni.com/training

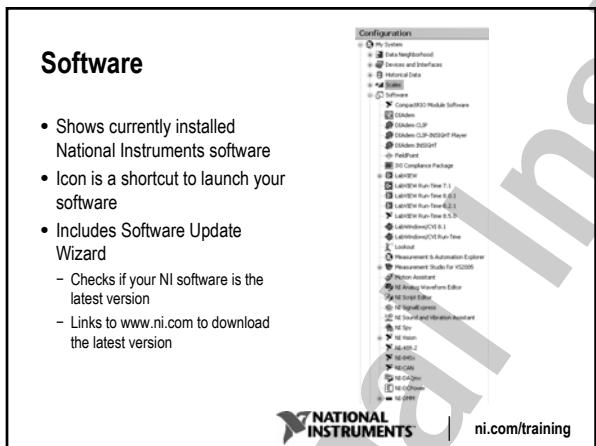
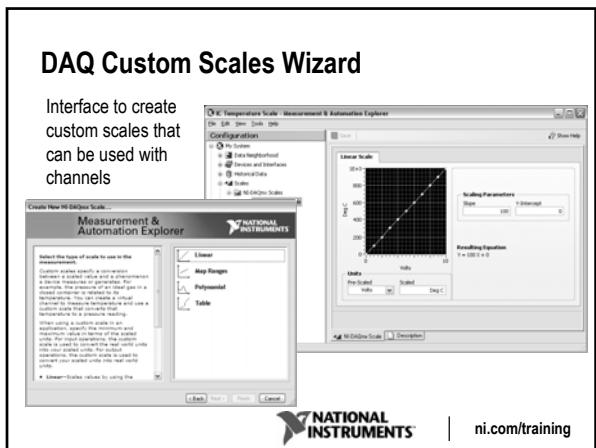
Scales

- Provides access to DAQ Custom Scales Wizard
- Shows configured scales
- Includes utility for viewing and reconfiguring your custom scales



NATIONAL INSTRUMENTS

ni.com/training



Exercise 2-3: DAQ Assistant and Custom Scales Wizard

To create NI-DAQmx channels using the DAQ Assistant and then to create an NI-DAQmx task from these three channels. Also, you will create a custom scale to convert the temperature sensor's voltage to degrees Celsius.

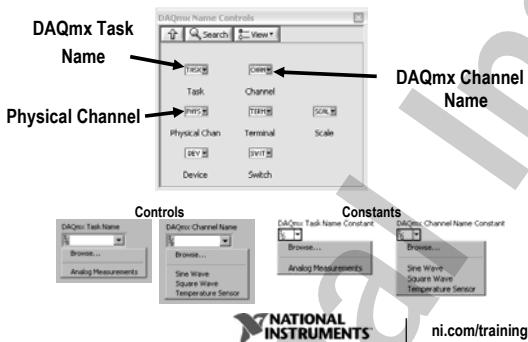
GOAL

Exercise 2-3: DAQ Assistant and Custom Scales Wizard

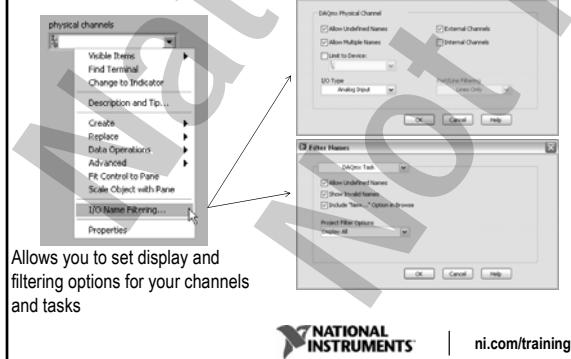
- Can you reuse the local Square Wave virtual channel you created in a different task?

DISCUSSION

DAQmx Name Controls

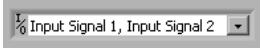


I/O Name Filtering



Addressing Multiple Channels

Dissimilar Scales, Ranges, and Terminal Configurations? Use Multiple Channels



- MAX Channels
 - Separate channel names with a comma when creating task
- Dynamically Created Channels
 - Create each channel separately and add to task

Note: You can only reference multiple channels, not multiple tasks!

NATIONAL INSTRUMENTS | ni.com/training

E. Overview of NI-DAQmx VIs – Primary Functions



- Create Virtual Channel
- Read
- Write
- Timing
- Trigger
- Task Functions

NATIONAL INSTRUMENTS | ni.com/training

Overview of NI-DAQmx VIs — Property Nodes



- Property Node – Used to read or write VI and object properties
- Specific property node for
 - Channel
 - Timing
 - Triggering
 - Reading
 - Writing

NATIONAL INSTRUMENTS | ni.com/training

Create Virtual Channel VI & Channel Property Node

Properties include

- Channel Type
- Physical Channel Name
- Description
- Analog I/O Custom Scale Name
- Digital I/O Number of Lines
- Counter I/O Pulse Duty Cycle
- ...And many more!

task in ────────── DAQmx Channel ────────── task out

- DAQmx Channel
- ActiveChan
- PhysicalChanName
- AI.Strainage.PoissonRatio
- AI.Strain.Units
- Cl.Freq.StartingEdge

- Create Virtual Channel VI
 - Programmatic creation of virtual channel(s)
 - Adds the created channel(s) to a specified task

Timing VI & Timing Property Node

- Properties include
 - Sample Mode
 - Samples per Channel
 - Sample Timing Type
 - Sample Clock Source
 - Master Timebase Source
 - ...And many more!

Timing VI

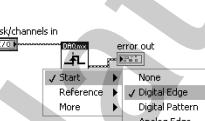
- Configures sample timing and task duration
- Creates buffer when needed

task in → DAQmx Timing → task out

Sample Clock (Analog/Digital)
Handshaking (Digital)
Implicit (Counter)
Use Waveform (Analog Output)
Change Detection (Digital Input)
Pipelined Sample Clock (Digital)

NATIONAL INSTRUMENTS | ni.com/training

Trigger VI & Trigger Property Node

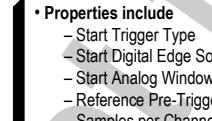


Properties include

- Start Trigger Type
- Start Digital Edge Source
- Start Analog Window Top
- Reference Pre-Trigger Samples per Channel
- Reference Analog Edge Slope
- ...And many more!

Trigger VI

- Configures the task to start or stop on a rising or falling digital edge, analog edge, or analog windows



Properties include

- Channel Type
 - Physical Channel Name
 - Description
 - Analog I/O Custom Scale Name
 - Digital I/O Number of Lines
 - Counter I/O Pulse Duty Cycle

...And many more!

A screenshot of a Windows-style configuration dialog box. The title bar says "task in..." and the bottom right corner says "task out". The main area contains a tree view with the following items:

- DAQmx Channel
- ActiveChan
- PhysicalChanName
- AI,StrainGage,PoissonRatio
- AI,Strain,Units
- CI,Freq,StartingEdge

The item "DAQmx Channel" is currently selected, indicated by a blue border around its icon.

ni.com/training

NATIONAL INSTRUMENTS

Timing VI & Timing Property Node

- **Properties include**
 - Sample Mode
 - Samples per Channel
 - Sample Timing Type
 - Sample Clock Source
 - Master Timebase Source
 - ...And many more!

The DAQmx Timing node is a central component in the DAQmx Timing subgraph. It receives input from the task in terminal and provides output to the task out terminal. The node has several internal parameters: SampQuant., SampPerChan, SampTimingType, SampClk.Rate, SampClk.Src, and AltSamp.Src.

ni.com/training

NATIONAL INSTRUMENTS

Trigger VI & Trigger Property Node

- **Properties include**
 - Start Trigger Type
 - Start Digital Edge Source
 - Start Analog Window Top
 - Reference Pre-Trigger Samples per Channel
 - Reference Analog Edge Slope
 - And many more!

NATIONAL
INSTRUMENTS

ni.com/training

Read VI & Read Property Node

• Read VI

- Used for analog, digital, counter, and raw data
- Single or multiple channels and samples

• Properties include

- Offset
- Channels to Read
- Waveform Attributes
- Status – Current Sample #
- Advanced – Raw Data Width
- ...And many more!

NATIONAL INSTRUMENTS | ni.com/training

Write VI & Write Property Node

• Write VI

- Used for analog, digital, and unscaled data
- Single or multiple channels and samples

• Properties include

- Position
- Offset
- Regeneration Mode
- Status – Space Available in Buffer
- Advanced – Raw Data Width
- ...And many more!

NATIONAL INSTRUMENTS | ni.com/training

Starting and Stopping Tasks

DAQmx Start Task

- Begins measurement or generation
- Increases user control
 - Get a task ready, but do not start until desired

DAQmx Stop Task

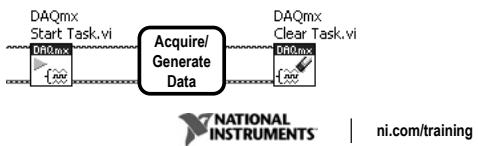
- Stops the measurement or generation
- Can restart task if stopped
- If no restart needed, then use the DAQmx Clear VI to stop and clear the task

NATIONAL INSTRUMENTS | ni.com/training

Clearing Tasks

DAQmx Clear Task

- Stops the task if necessary
- Releases any resources the task reserved
- Use this VI to clear the task when you are finished with the task



ni.com/training

Summary—Quiz

1. Which of the following are components of a DAQ device?
 - a) Analog Input Circuitry
 - b) Data Transfer Bus
 - c) RAM
 - d) Counter Circuitry
 - e) Onboard FIFOs



ni.com/training

Summary—Quiz Answer

1. Which of the following are components of a DAQ device?
 - a) Analog Input Circuitry
 - b) Data Transfer Bus
 - c) RAM
 - d) Counter Circuitry
 - e) Onboard FIFOs



ni.com/training

Summary—Quiz

2. All DAQmx VIs and property nodes are accessible through the NI-DAQmx palette.
- a) True
 - b) False



ni.com/training

Summary—Quiz Answer

2. All DAQmx VIs and property nodes are accessible through the NI-DAQmx palette.
- a) True
 - b) False



ni.com/training

Summary—Quiz

3. Code width defines how close your measurement is to its true value.
- a) True
 - b) False



ni.com/training

Summary—Quiz Answer

3. Code width defines how close your measurement is to its true value.
- a) True
 - b) False



ni.com/training

National Instruments
Not for Distribution

Lesson 3
Analog Input

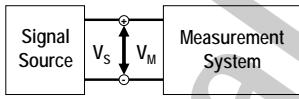
TOPICS

A. Grounding Issues	E. Finite Buffered Acquisition
B. Sampling Considerations	F. Continuous Buffered Acquisition
C. Single Sample Acquisition	G. Triggering
D. DAQ Device Architectures	

 | ni.com/training

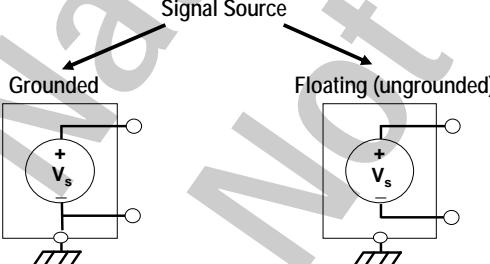
A. Grounding Issues

- To get correct analog input measurements, you must properly ground your system
- How the signal is grounded affects how you should ground the instrumentation amplifier on the DAQ device
- Steps to proper grounding of your system:
 1. Determine how your signal is grounded
 2. Choose a grounding mode for your Measurement System



 | ni.com/training

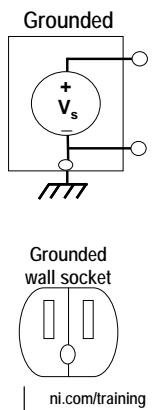
Signal Source Categories



 | ni.com/training

Grounded Signal Source

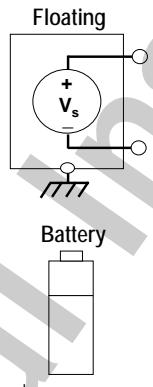
- Signal is referenced to a system ground
 - Earth ground
 - Building ground
- Examples:
 - Power supplies
 - Signal Generators
 - Anything that plugs into a grounded, electrical wall socket



NATIONAL INSTRUMENTS

Floating Signal Source

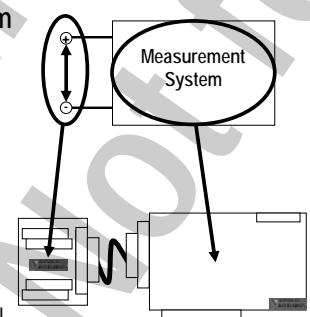
- Signal is NOT referenced to a system ground
 - Earth ground
 - Building ground
- Examples:
 - Batteries
 - Thermocouples
 - Transformers
 - Isolation Amplifiers



NATIONAL INSTRUMENTS

Measurement System

- Three modes of grounding for your Measurement System
 - Differential
 - Referenced Single-Ended (RSE)
 - Non-Referenced Single-Ended (NRSE)
- Mode you choose will depend on how your signal is grounded

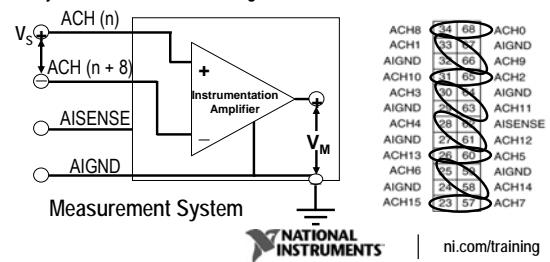


NATIONAL INSTRUMENTS

ni.com/training

Differential Mode

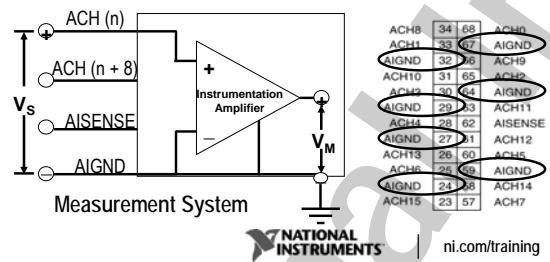
- Two channels used for each signal
 - ACH 0 is paired with ACH 8, ACH 1 is paired with ACH 9, and so on.
- Rejects common-mode voltage and common-mode noise



ACH8	34	68	ACH0
ACH1	33	67	AIGND
AIGND	32	66	ACH9
ACH10	31	65	ACH2
ACH3	30	64	AIGND
AIGND	29	63	ACH11
ACH4	28	62	AISENSE
AIGND	27	61	ACH12
ACH13	26	60	ACH5
ACH6	25	59	AIGND
AIGND	24	58	ACH14
ACH15	23	57	ACH7

Referenced Single-Ended (RSE) Mode

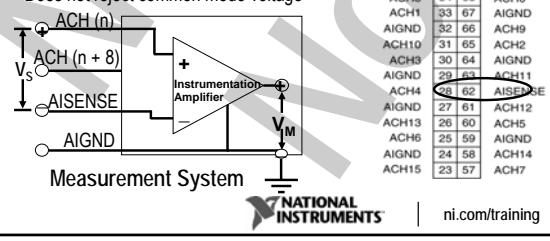
- Measurement made with respect to system ground
- One channel used for each signal
- Does not reject common mode voltage



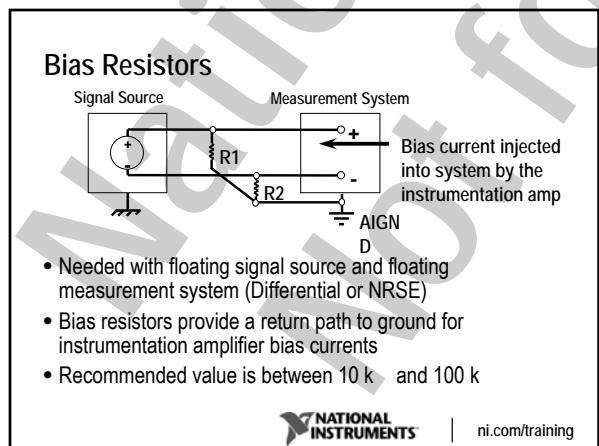
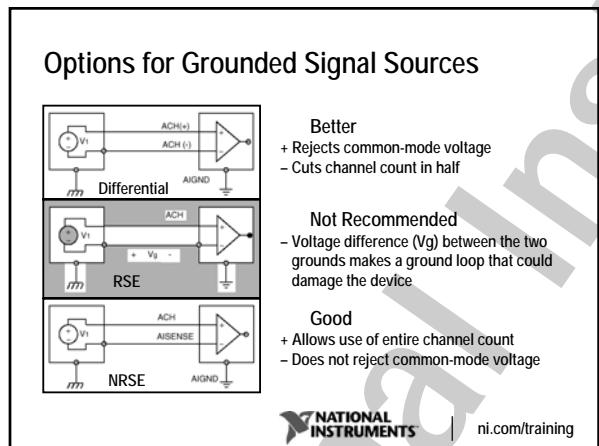
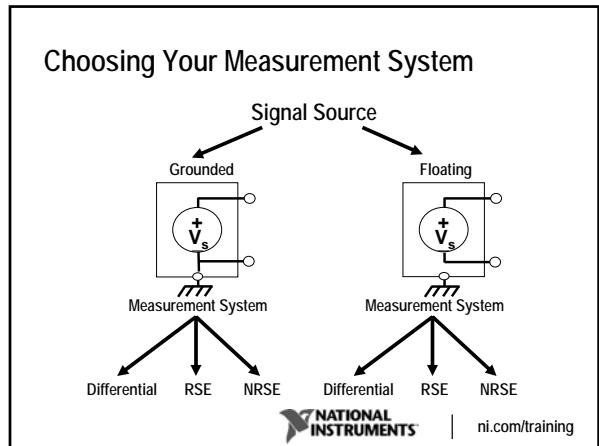
ACH8	34	68	ACH0
ACH1	33	67	AIGND
AIGND	32	66	ACH9
ACH10	31	65	ACH2
ACH3	30	64	AIGND
AIGND	29	63	ACH11
ACH4	28	62	AISENSE
AIGND	27	61	ACH12
ACH13	26	60	ACH5
ACH6	25	59	AIGND
AIGND	24	58	ACH14
ACH15	23	57	ACH7

Non-Referenced Single-Ended (NRSE) Mode

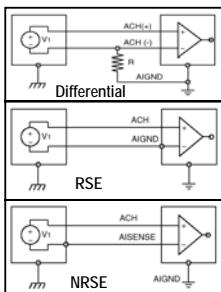
- Variation on RSE
- One channel used for each signal
- Measurement made with respect to AISENSE not system ground
- AISENSE is floating
- Does not reject common mode voltage



ACH8	34	68	ACH0
ACH1	33	67	AIGND
AIGND	32	66	ACH9
ACH10	31	65	ACH2
ACH3	30	64	AIGND
AIGND	29	63	ACH11
ACH4	28	62	AISENSE
AIGND	27	61	ACH12
ACH13	26	60	ACH5
ACH6	25	59	AIGND
AIGND	24	58	ACH14
ACH15	23	57	ACH7



Options for Floating Signal Sources

**Best**

- + Rejects common-mode voltage
- Cuts channel count in half
- Needs bias resistors

Better

- + Allows use of entire channel count
- Do not need bias resistors
- Does not reject common-mode voltage

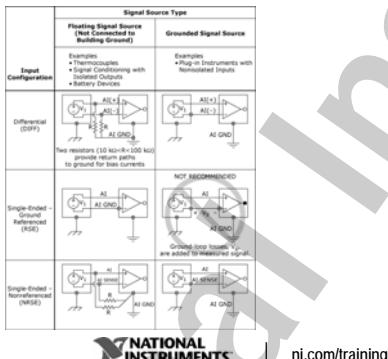
Good

- + Allows use of entire channel count
- Needs bias resistors
- Does not reject common-mode voltage



ni.com/training

Signal Source Connections Overview



ni.com/training

Exercise 3-1: Differential, RSE, and NRSE

To learn how to choose a grounding mode for a measurement system and how to connect signals properly to that measurement system.

GOAL

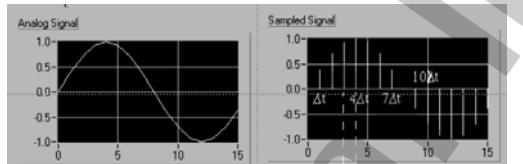
Exercise 3-1: Differential, RSE, and NRSE

- What are the pros and cons of choosing RSE for the floating signal source?
- What are the pros and cons of choosing differential for the floating signal source?

DISCUSSION

Sampling Signals

- Individual samples are represented by: $x[i] = x(i \Delta t)$, for $i = 0, 1, 2, \dots$
- If N samples are obtained from signal $x(t)$:
 $X = \{x[0], x[1], x[2], \dots x[N-1]\}$
- The sequence $X = \{x[i]\}$ is indexed on i and does not contain sampling rate information

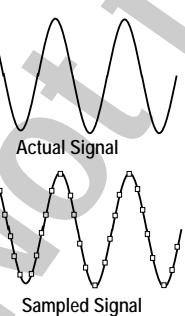


NATIONAL INSTRUMENTS

ni.com/training

B. Sampling Considerations

- Actual analog input signal is continuous with respect to time
- Sampled signal is series of discrete samples acquired at a specified sampling rate
- The faster we sample, the more our sampled signal will look like our actual signal
- If not sampled fast enough, a problem known as aliasing will occur

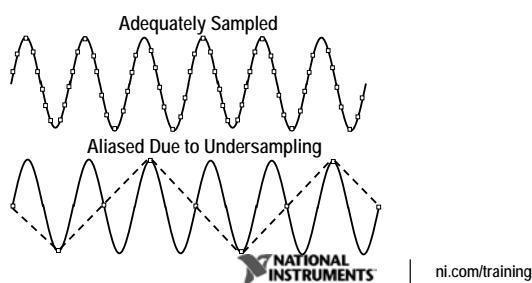


NATIONAL INSTRUMENTS

ni.com/training

Aliasing

- Sample rate – how often an A/D conversion takes place
- Alias – misrepresentation of a signal



Nyquist Theorem

You must sample at greater than 2 times the maximum frequency component of your signal to accurately represent the FREQUENCY of your signal.

NOTE: You must sample between 5–10 times greater than the maximum frequency component of your signal to accurately represent the SHAPE of your signal.



ni.com/training

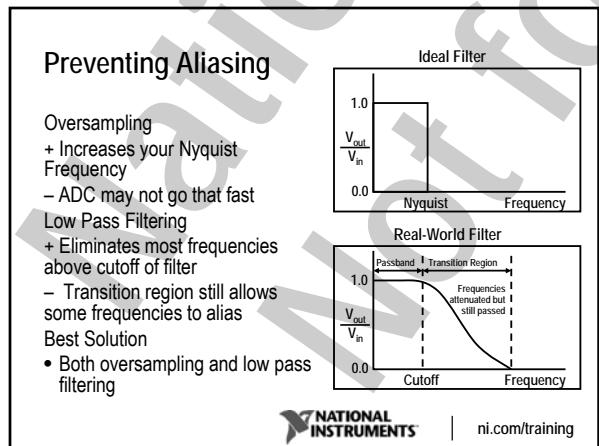
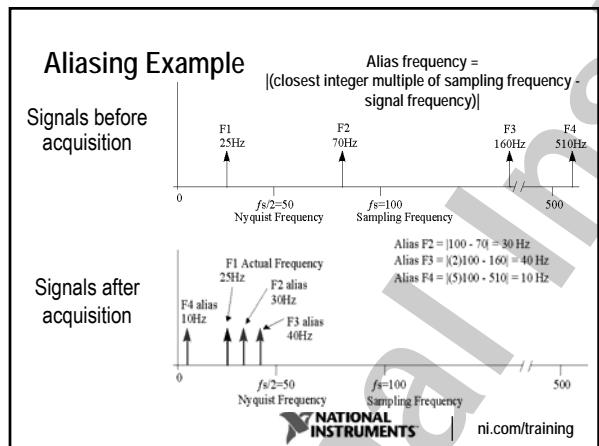
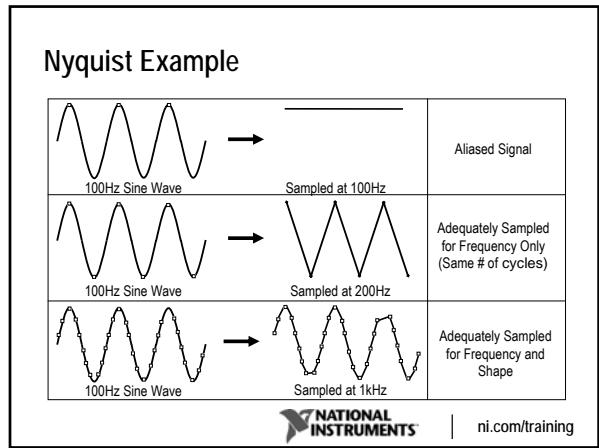
Nyquist Frequency

- Half the sampling frequency
- You will only get a proper representation of signals that are equal to or less than your Nyquist Frequency
- Signals above Nyquist Frequency will alias according to the following formula:

$$\text{Alias frequency} = |(\text{closest integer multiple of sampling frequency} - \text{signal frequency})|$$



ni.com/training



Exercise 3-2: Sampling Rate and Aliasing

To demonstrate aliasing and the effects of sampling rate on an input signal.

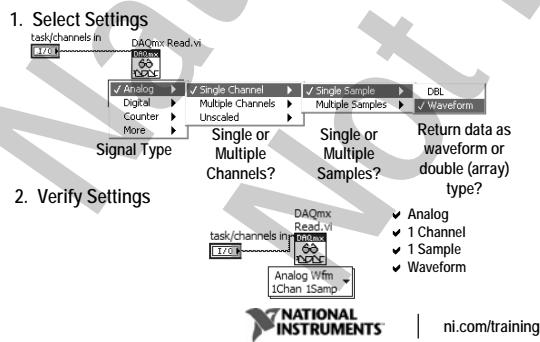
GOAL

Exercise 3-2: Sampling Rate and Aliasing

- If the analog output signal is set to 5000 Hz, what should the analog input sampling rate be in order to correctly detect the analog output frequency?

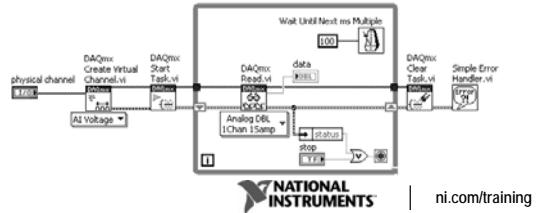
DISCUSSION

DAQmx Read Function



C. Single Sample Software-Timed Acquisition

- DAQmx Read VI
 - Acquires one sample each iteration
- Wait Until Next ms Multiple function
 - Determines rate of acquisition using software timing



ni.com/training

Exercise 3-3: Voltmeter VI

To acquire an analog signal using a DAQ device.

GOAL

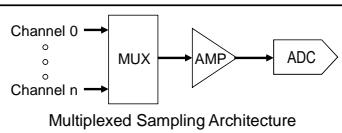
Exercise 3-3: Voltmeter VI

- What determines the rate at which this VI acquires samples?

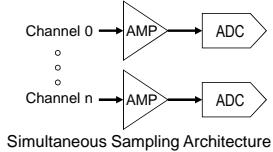
DISCUSSION

D. DAQ Device Architectures

- One amplifier and ADC for ALL channels
- Cost effective
 - Used on M Series and some X Series devices



- One amplifier and A/D Converter for EACH channel
- More expensive
 - Used on most S Series and cDAQ devices and some X Series devices



ni.com/training

Sampling Terminology

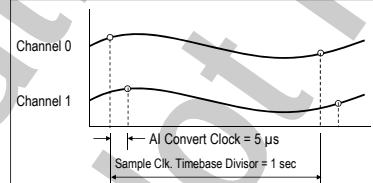
- Sample
 - A single measurement from a single channel
- Sample Rate
 - Samples per Channel per Second
- Sample Clock
 - Clock that controls time interval between samples
 - During each cycle of the sample clock, one sample for each channel is acquired
- AI Convert Clock
 - Clock that directly causes analog-to-digital conversions
 - Interchannel delay



ni.com/training

Multiplexed Sampling

- Uses both Sample Clock and AI Convert Clock
 - Gives the effect of simultaneous sampling for less money
- By default, NI-DAQmx chooses the fastest AI Convert Clock rate possible that allows for adequate settling time
 - Can manually set AI Convert Clock rate using property node



Two Channel Example

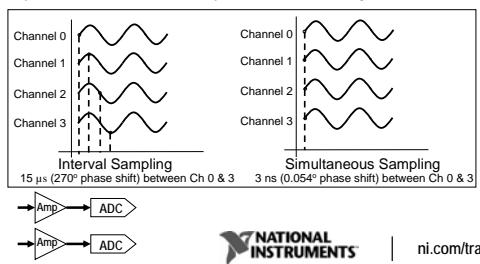
- Samples per Channel per Second = 1 sample/chan/sec
- Sample Clock Timebase Divisor = 1/Samples per Channel per Second = 1sec/chan/sample
- Sample duration = (# of channels - 1) * AI Convert Clock = 5 μs



ni.com/training

Simultaneous Sampling

- Used when time relationship between signals is important
- Available on all S-Series devices and some X-Series devices
- Only uses a sample clock to synchronize the taking of samples



NATIONAL
INSTRUMENTS

ni.com/training

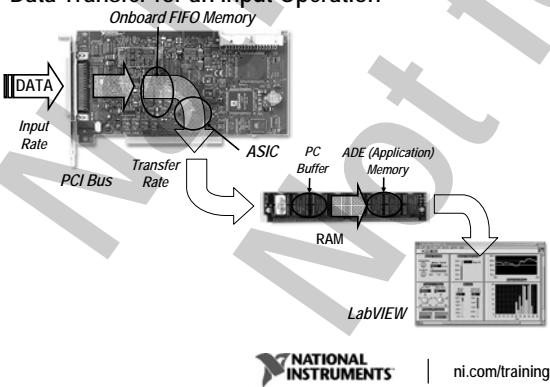
Buffered Analog Input

- Buffer – temporary storage in computer memory for acquired or generated data
- Data transfer mechanism transfers samples from your device into the buffer where they await your call to the DAQmx Read VI to copy them to your application
 - Finite transfer
 - Continuous transfer

NATIONAL
INSTRUMENTS

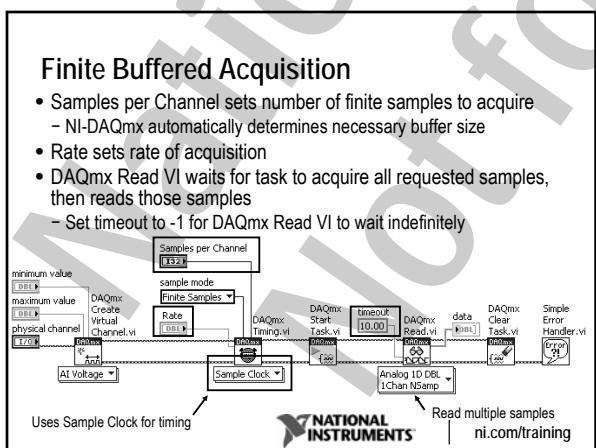
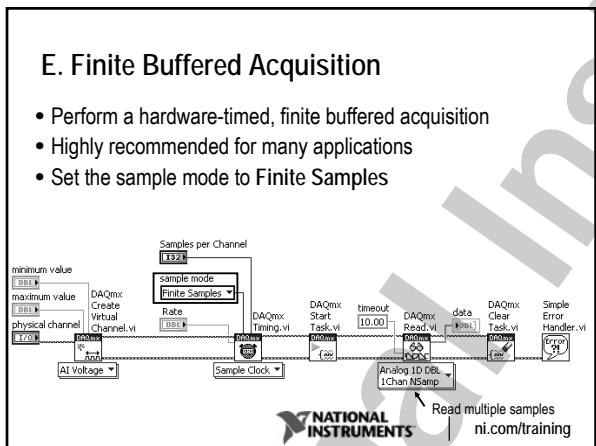
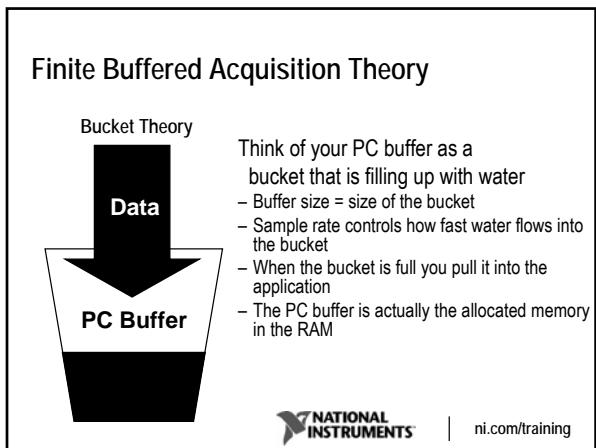
ni.com/training

Data Transfer for an Input Operation



NATIONAL
INSTRUMENTS

ni.com/training



Exercise 3-4: Finite Acquisition with Analysis

To acquire an array of data using a finite buffered configuration.

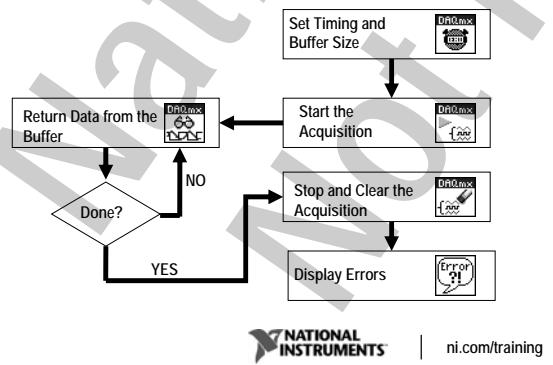
GOAL

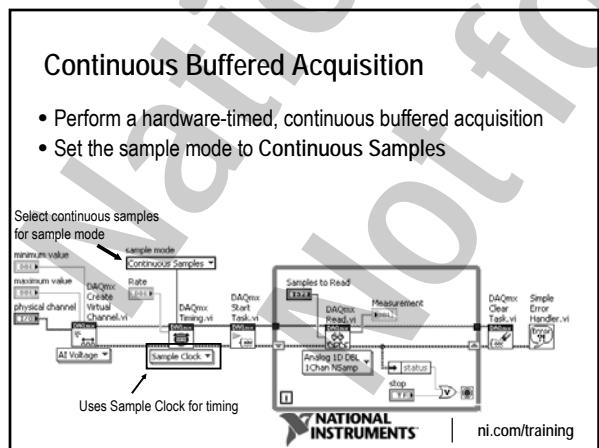
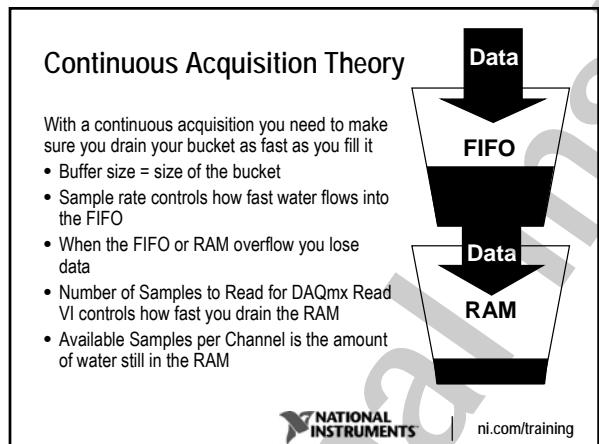
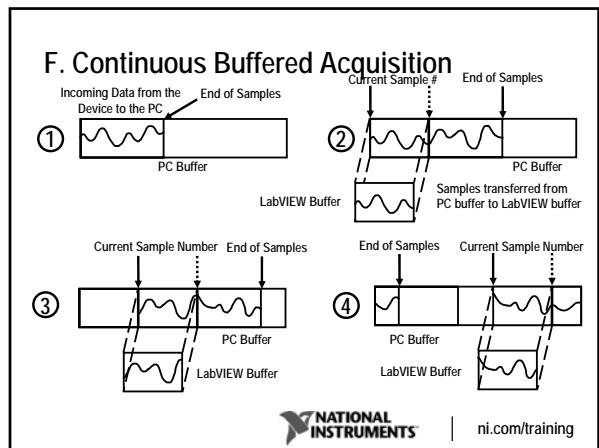
Exercise 3-4: Finite Acquisition with Analysis

- What determines the rate at which this VI acquires samples?

DISCUSSION

Continuous Buffered Acquisition Flowchart





Continuous Buffered Acquisition

- Rate sets rate of acquisition
 - NI-DAQmx automatically determines buffer size based on rate
- Rate sets rate of acquisition
- Samples to Read determines the number of samples to read each iteration of the While Loop

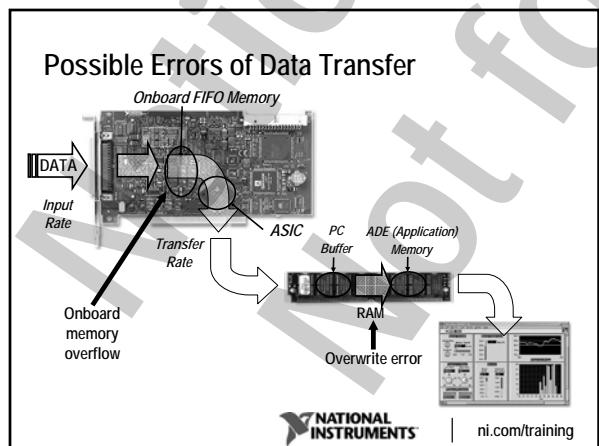
ni.com/training

Configuring the Number of Samples to Read for Continuous Acquisition

- Rate sets rate of acquisition (sampling rate)
 - NI-DAQmx automatically determines buffer size based on rate

Sample Rate	Buffer Size
No rate specified	10 kS
0-100 S/s	1 kS
100-10,000 S/s	10 kS
10,000-1,000,000 S/s	100 kS
> 1,000,000 S/s	1 MS

ni.com/training



Data Loss with Continuous Acquisition

Overflow Error

- Indicates that NI-DAQmx was unable to retrieve data from the FIFO fast enough
- Consequence:
 - The data in the FIFO will be overwritten
- How to avoid an overflow error:
 - Decrease the Samples per Channel per Second rate
 - Purchase a device with a larger FIFO
 - Purchase a faster computer with a faster bus



ni.com/training

Data Loss with Continuous Acquisition

Overwrite Error

- Indicates that you are not reading data from the PC buffer fast enough
- Consequence:
 - Your unread data will be overwritten by newer data
- How to avoid an overwrite error:
 - Increase the buffer size with the DAQmx Timing VI
 - Increase the Number of Samples per Channel to read with DAQmx Read VI
 - Decrease the Samples per Channel per Second acquisition rate with the DAQmx Timing VI
 - Don't do any extra processing in your loop with the DAQmx Read VI



ni.com/training

Exercise 3-5: Continuous Acquisition and Logging

To continuously acquire data from a DAQ device and log this data to a file.

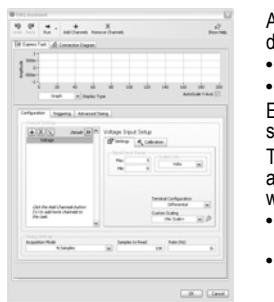
GOAL

Exercise 3-5: Continuous Acquisition and Logging

- What will happen if you add too many processing functions inside the While Loop?

DISCUSSION

G. Triggering – DAQmx Actions and Causes



Action – when a DAQ device does something

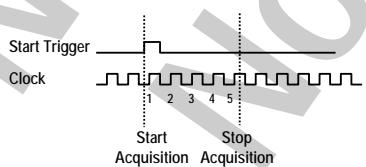
- Generating waveform output
 - Acquiring data
- Each action needs a cause or stimulus – a trigger
- Trigger is named after the action it causes and the way it was produced
- Action caused: Start, Reference, Pause, Advance (Switch devices)
 - Production method: Analog or Digital

NATIONAL INSTRUMENTS

ni.com/training

Trigger Actions – Start Triggering

- Valid for input or output operations
- Valid for finite or continuous operations
- Example – Input 5 samples on start trigger:

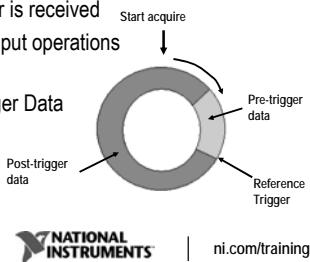


NATIONAL INSTRUMENTS

ni.com/training

Trigger Actions – Reference Triggering

- Acquisition starts as soon as software is started
- Data in the buffer keeps getting overwritten (first in first out) until the reference trigger is received
- Typically only for finite input operations
- Post-trigger Data = Size of Buffer – Pre-trigger Data

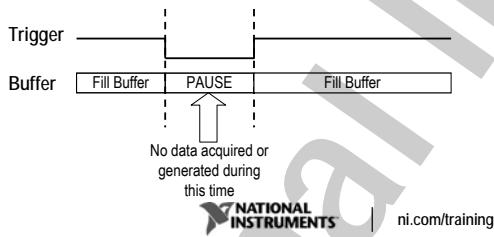


NATIONAL INSTRUMENTS

ni.com/training

Trigger Actions – Pause Triggering

- Allows you to pause an acquisition/generation
- Operation stops when trigger signal is low and resumes when trigger signal is high



NATIONAL INSTRUMENTS

ni.com/training

Trigger Types – Digital Edge Triggering

- Accepts TTL/CMOS compatible signals
 - 0 to 0.8V = logic low
 - 2 to 5V = logic high
- Trigger on rising or falling edge of signal



NATIONAL INSTRUMENTS

ni.com/training

Trigger Types – Analog Edge Triggering

Edge – Trigger off signal level and slope (rising or falling)

Slope = Rising Level = 2.7

Slope = Falling Level = 2.7

NIATIONAL INSTRUMENTS | ni.com/training

Trigger Types – Analog Edge Triggering

Hysteresis – adds a window above or below the trigger level to reduce false triggering due to noise or jitter

Slope = Rising, Level = 2.7, Hysteresis = -1

Slope = Falling, Level = 1.7, Hysteresis = 1

NIATIONAL INSTRUMENTS | ni.com/training

Trigger Types – Analog Triggering Support

Not all DAQ devices support analog triggering

- Most M Series and X Series devices support analog triggering
- Some C Series Modules support analog triggering

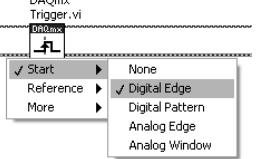
Check ni.com to see if your device supports analog triggering

NIATIONAL INSTRUMENTS | ni.com/training

Triggering Analog Input

Begin data acquisition based on:

- Digital Edge (rising or falling)
- Digital Pattern (digital pattern of 0's and 1's)
- Analog Edge (slope and level)
- Analog Window (specify upper and lower window limits)



NATIONAL INSTRUMENTS | ni.com/training

Exercise 3-6: Triggered Continuous Acquisition

Write a VI to trigger analog input on a digital edge.

GOAL

Exercise 3-6: Triggered Continuous Acquisition

- How would you change the VI if you also needed to acquire 100 samples of pre-trigger data?

DISCUSSION

Summary—Quiz

1. Which of the following grounding modes should you not use with a grounded signal source?
 - a) Differential
 - b) Referenced single-ended
 - c) Non-referenced single-ended



ni.com/training

Summary—Quiz Answer

1. Which of the following grounding modes should you not use with a grounded signal source?
 - a) Differential
 - b) Referenced single-ended
 - c) Non-referenced single-ended



ni.com/training

Summary—Quiz

2. The Nyquist Theorem helps to determine the sampling rate. What problem does this help with?
 - a) Spying
 - b) Noise
 - c) Aliasing
 - d) Isolation



ni.com/training

Summary—Quiz Answer

2. The Nyquist Theorem helps to determine the sampling rate. What problem does this help with?
- a) Spying
 - b) Noise
 - c) Aliasing
 - d) Isolation



ni.com/training

Summary—Quiz

3. Which of the following operations can the DAQmx Read VI be used for?
- a) Single-point
 - b) Multi-sample
 - c) Multi-channel
 - d) All of the above



ni.com/training

Summary—Quiz Answer

3. Which of the following operations can the DAQmx Read VI be used for?
- a) Single-point
 - b) Multi-sample
 - c) Multi-channel
 - d) All of the above



ni.com/training

Summary—Quiz

4. Software-timed, single-point reads are good for getting information on the shape of a waveform
- a) True
 - b) False



ni.com/training

Summary—Quiz Answer

4. Software-timed, single-point reads are good for getting information on the shape of a waveform
- a) True
 - b) False



ni.com/training

Summary—Quiz

5. Buffered acquisitions require the use of a clock signal.
- a) True
 - b) False



ni.com/training

Summary—Quiz Answer

5. Buffered acquisitions require the use of a clock signal.
- a) True
 - b) False



ni.com/training

National Instruments
Not for Distribution

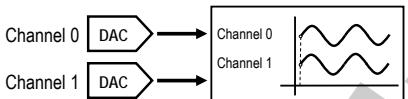
Lesson 4
Analog Output

TOPICS

A. Analog Output Architecture	D. Continuous Buffered Generation
B. Single Sample Generation	E. Triggered Generation
C. Finite Buffered Generation	

NATIONAL INSTRUMENTS | ni.com/training

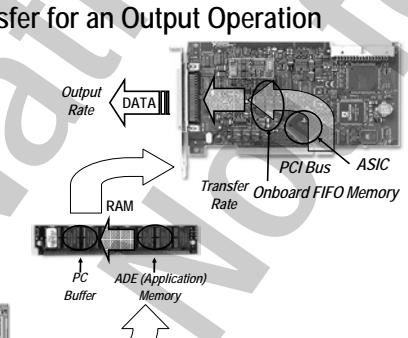
A. Analog Output Architecture



- Most multifunction DAQ devices have a Digital-to-Analog Converter (DAC) for each analog output channel
- DACs are updated at the same time
- Similar to simultaneous sampling for analog input

NATIONAL INSTRUMENTS | ni.com/training

Data Transfer for an Output Operation



LabVIEW

NATIONAL INSTRUMENTS | ni.com/training

DAQmx Write VI

1. Select Settings

Signal type: Analog DBL
task/channels in: 1/0
Analog DBL
1Chan 1Samp
Single Channel
Multiple Channels
Unscaled

Single or multiple channels?

Single or multiple samples?

Write data as waveform or double (array) type?
1D DBL
Waveform

2. Verify Settings

DAQmx Write VI
task/channels in: 1/0
Analog Wfm
1Chan NSamp

Analog
1 Channel
N Samples
Waveform

NATIONAL INSTRUMENTS | ni.com/training

Auto Start Parameter to Write VI

Controls whether the Write VI starts the generation

- For single samples, auto start is true by default
- For multiple samples, auto start is false by default

When using Start/Clear Task VI, always set auto start to false

auto start
Single Sample DAQmx
Analog DBL 1Chan 1Samp

auto start
Multiple Samples DAQmx
Analog 1D DBL 1Chan NSamp

NATIONAL INSTRUMENTS | ni.com/training

B. Single Sample Generation

Use when signal level is more important than generation rate

- Example: Outputting a constant DC voltage

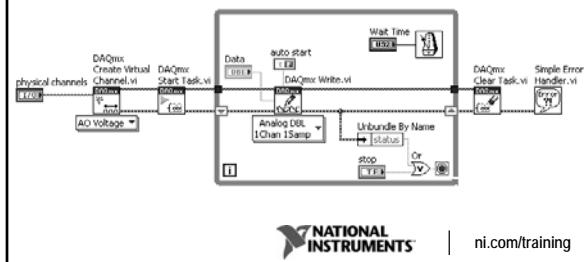
Set timing with DAQmx Timing VI

- Software-timed
 - Rate is determined by the OS or the program (by adding a time delay in the generation loop)
 - Sample Timing Type property is set to On Demand
- Hardware-timed
 - Clock on your device controls the timing. Much faster and more accurate than a software loop.
 - Sample Timing Type property is set to Sample Clock

NATIONAL INSTRUMENTS | ni.com/training

Software-Timed Analog Output Loop

Update the voltage on the analog output channel until user hits stop button



Exercise 4-1: Continuous Single-Point Generation

To create a VI that produces a variable voltage signal.

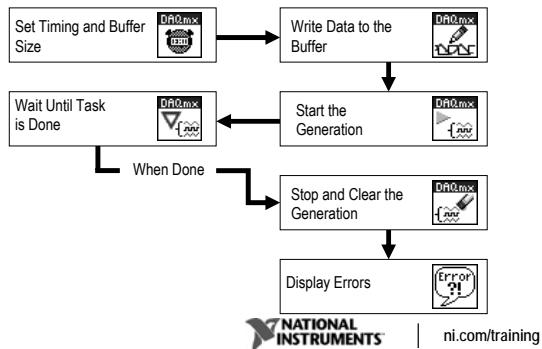
GOAL

Exercise 4-1: Continuous Single-Point Generation

- Should you use this application to output a 10 Hz sine wave? Why or why not?

DISCUSSION

C. Finite Buffered Generation Flowchart



Output Waveform Frequency

Output waveform frequency depends on three factors:

- Update rate
- Points in the buffer
- Number of cycles in the buffer



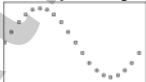
$$\text{Signal Frequency} = \frac{\text{# of cycles in buffer}}{\text{points in the buffer}} \times \text{update rate}$$



ni.com/training

Output Waveform Frequency

- Buffer size = 1000 pts
- # of cycles in buffer = 1
- Update rate = 1000 Hz
- Buffer size = 1000 pts
- # of cycles in buffer = 1
- Update rate = 2000 Hz
- Buffer size = 1000 pts
- # of cycles in buffer = 2
- Update rate = 1000 Hz



Signal frequency = 1 Hz



Signal frequency = 2 Hz



Signal frequency = 2 Hz

$$\text{Signal Frequency} = \frac{\text{# of cycles in buffer}}{\text{points in the buffer}} \times \text{update rate}$$



ni.com/training

Wait Until Done vs. Is Task Done



Wait Until Done VI

- Used for finite generations
- User can set timeout
- Blocks task until finished executing



Is Task Done VI

- Used for error checking in continuous generations
- Polls to determine state of the generation



ni.com/training

Timing for Finite Generation

Set timing with DAQmx Timing VI

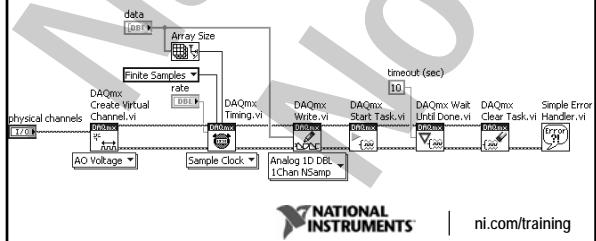
- Software-timed
 - Rate is determined by the OS or the program (by adding a time delay in the generation loop)
 - Sample Timing Type property is set to On Demand
- Hardware-timed
 - Clock on your device controls the timing. Much faster and more accurate than a software loop
 - Sample Timing Type property is set to Sample Clock



ni.com/training

Finite Buffered Generation Example

- Set the sample mode to Finite Samples
- Write data to buffer with DAQmx Write VI
- Use DAQmx Wait Until Done VI



Exercise 4-2: Finite Buffered Generation

To create a VI that generates a finite waveform of sound data.

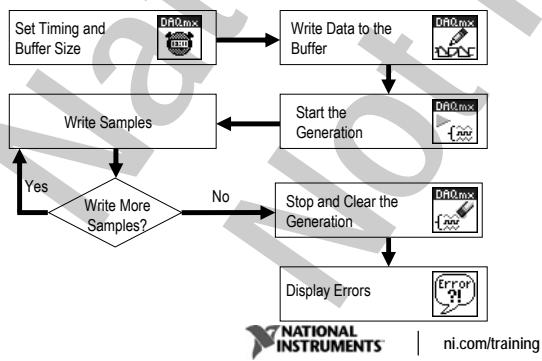
GOAL

Exercise 4-2: Finite Buffered Generation

- What would happen if you removed the DAQmx Wait Until Done VI from the block diagram?

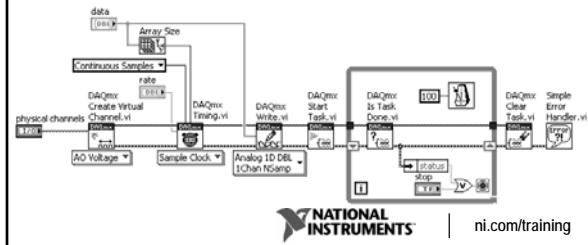
DISCUSSION

D. Continuous Buffered Generation Flowchart



Continuous Waveform Generation Using the Sample Clock

- Set the sample mode to Continuous Samples
- Write data to buffer with DAQmx Write VI
- Use DAQmx Is Task Done VI



NATIONAL INSTRUMENTS | ni.com/training

Waveform Generation Using dt for Timing

- Use Waveform instance of DAQmx Timing VI to use dt for timing

NATIONAL INSTRUMENTS | ni.com/training

Regeneration

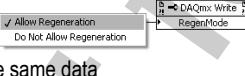
Use Regeneration Mode property

- Allow Regeneration generates the same data multiple times

Use On Board Memory property

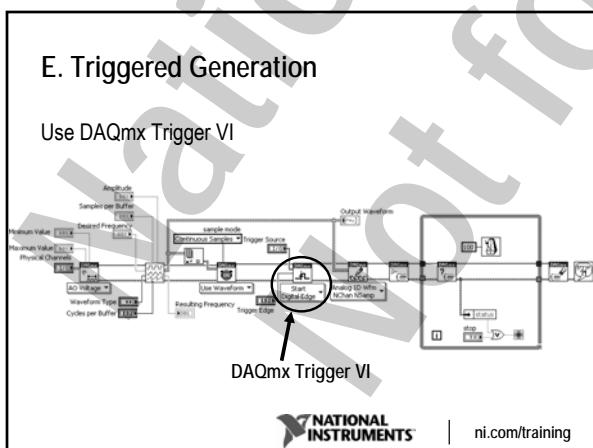
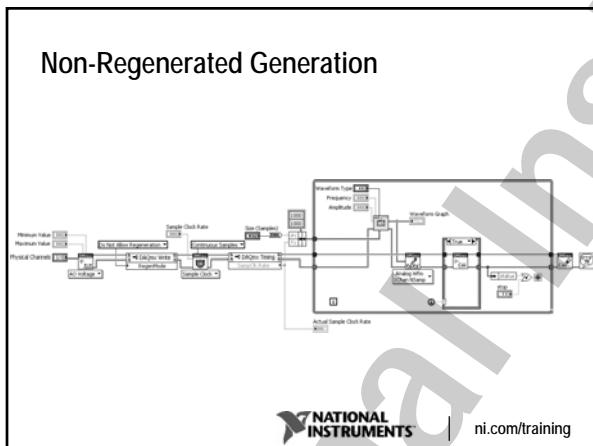
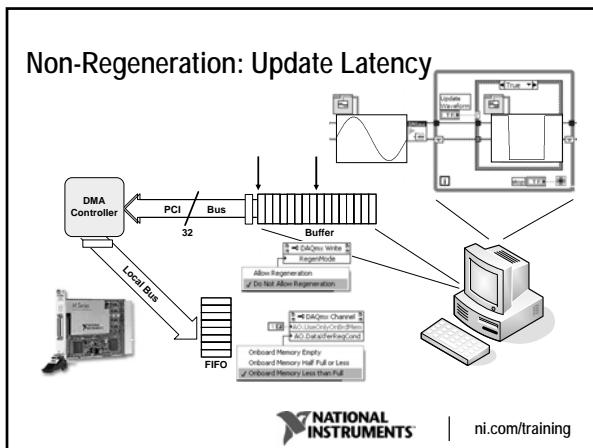
- If true, regenerate data from onboard memory of device
- If false (default), regenerate data from PC buffer

If regeneration is enabled and you write new data to the buffer, glitching can occur during the transition



NATIONAL INSTRUMENTS

| ni.com/training



Exercise 4-3: Triggered Continuous Buffered Generation

To build a VI to trigger a continuous buffered generation on an analog input channel.

GOAL

Exercise 4-3: Triggered Continuous Buffered Generation

- How would you modify the block diagram to output your own custom analog signal?

DISCUSSION

Summary—Quiz

1. In a typical DAQ device how many channels are there per DAC?
 - 1
 - 8
 - 16
 - 32



ni.com/training

Summary—Quiz Answer

1. In a typical DAQ device how many channels are there per DAC?
 - a) 1
 - b) 8
 - c) 16
 - d) 32



ni.com/training

Summary—Quiz

2. If you generate a sinusoidal waveform with 200 samples and 10 cycles at an output rate of 1 kHz, what is the apparent rate of the sine wave?
 - a) 1000 Hz
 - b) 500 Hz
 - c) 50 Hz
 - d) 20 Hz



ni.com/training

Summary—Quiz Answer

2. If you generate a sinusoidal waveform with 200 samples and 10 cycles at an output rate of 1 kHz, what is the apparent rate of the sine wave?
 - a) 1000 Hz
 - b) 500 Hz
 - c) 50 Hz
 - d) 20 Hz



ni.com/training

Lesson 5
Digital I/O

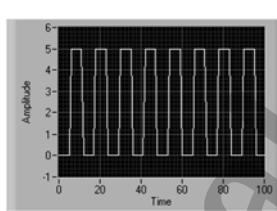
TOPICS

- A. Digital Overview
- B. Digital I/O
- C. Hardware-Timed Digital I/O

NATIONAL INSTRUMENTS | ni.com/training

A. Digital Overview

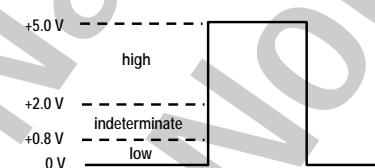
- A digital signal has two discrete levels
 - High (5 V, 24 V, etc)
 - Low (0 V)



NATIONAL INSTRUMENTS | ni.com/training

Digital Signals

Digital lines on a DAQ device accept and generate TTL compatible signals



+5.0 V
high

+2.0 V
indeterminate

+0.8 V
0 V
low

Definition of a TTL Signal

NATIONAL INSTRUMENTS | ni.com/training

Digital Terminology

- Bit – The smallest unit of data. Each bit is either a 1 or a 0
- Byte – A binary number consisting of 8 related bits of data
- Line – One individual signal in a port. Bit refers to the data transferred. Line refers to the hardware
- Port – A collection of digital lines (usually four or eight)
- Port Width – Number of lines per port (usually four or eight)
- Mask – Determines which lines are read from or written to



ni.com/training

NI-DAQmx Digital Terminology

Port Notation (specifying a single port)

- Dev x / Port y
- cDAQ a Mod b / Port c

Line Notation (specifying single or multiple lines)

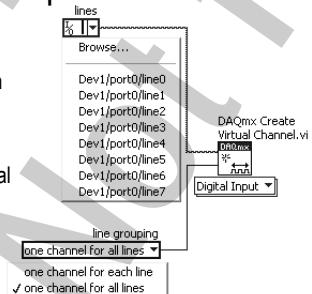
- Dev x / Port y / Line a
- cDAQ a Mod b / Port c / Line m : n
- Dev x / Port y / Line a, Dev x / Port y / Line b



ni.com/training

Digital Virtual Channel Options

- Create a digital channel to be a port, line, or collection of lines
- Choose how to group digital lines into one or more virtual channels
 - Affects how to configure DAQmx Read VI



ni.com/training

Digital Constant I/O Filtering Options

- By default, only lines are displayed as options in digital channel constants
- I/O Filtering gives the option to selectively display lines and ports
- Right-click on channel constant/control and choose I/O Filtering

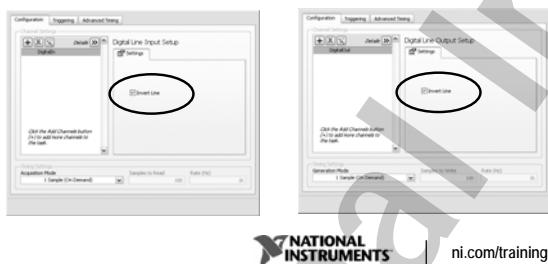


NATIONAL INSTRUMENTS

ni.com/training

DAQ Assistant – Digital I/O

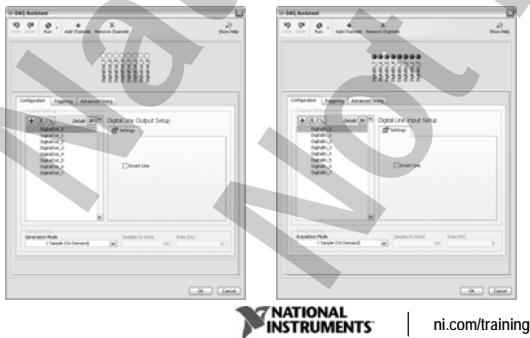
- Default setting is Active High
- Can invert the lines to Active Low



NATIONAL INSTRUMENTS

ni.com/training

DAQ Assistant – Digital Test Panels

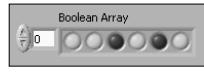


NATIONAL INSTRUMENTS

ni.com/training

Digital Channel Data Formats

Line Formats

- One line per channel
 - Represented by a single Boolean value
- Multiple lines per channel
 - Represented by a 1D array of Booleans with each element corresponding to a line in the channel in the order in which they are specified

NATIONAL INSTRUMENTS | ni.com/training

Digital Channel Data Formats

Port Formats

- A port organizes individual lines into collection
- Represented by a U8, U16, or U32 integer where each line requires one bit
 - Most efficient in terms of space

Waveform Format

- Couples channel and timing information to data
- Not used for static digital I/O

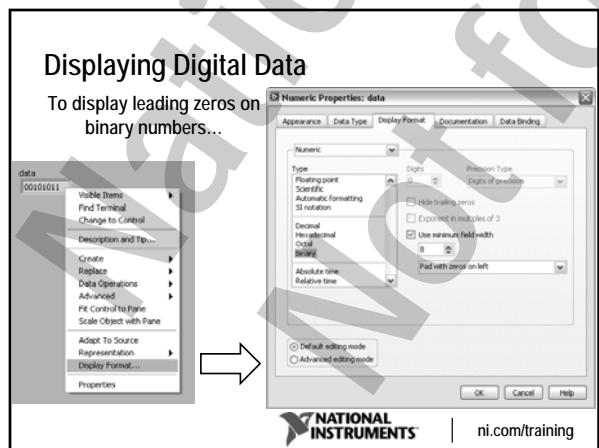
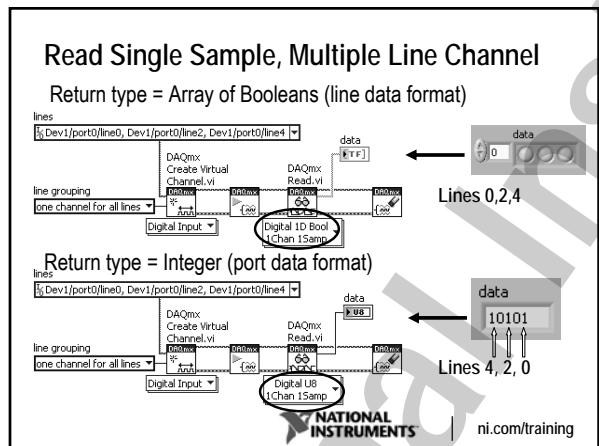
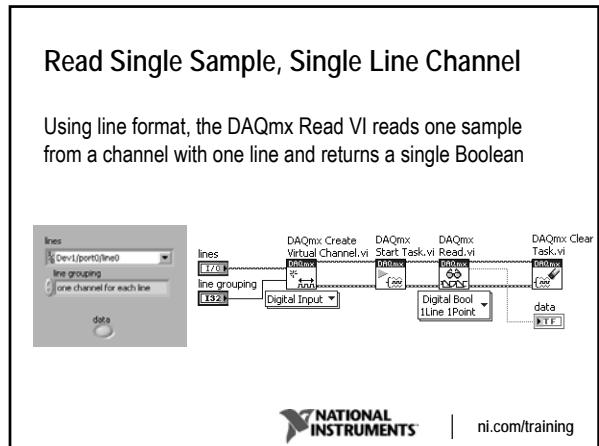
NATIONAL INSTRUMENTS | ni.com/training

B. Digital I/O – DAQmx Read VI



- Can choose line, port, or waveform data format
 - Line data types are only available for single sample read and writes
- Reading and writing multiple samples for static I/O is not very practical – Use handshaking or clocked operations instead

NATIONAL INSTRUMENTS | ni.com/training



Exercise 5-1: Digital Read

To acquire digital data using the DAQ device, and display the digital data on the front panel of a LabVIEW VI.

GOAL

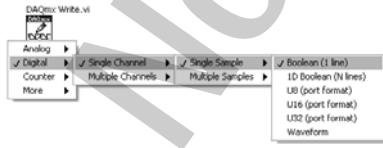
Exercise 5-1: Digital Read

- How would you modify the VI if you wanted to read from both port0/line0 and port0/line7?

DISCUSSION

Digital I/O – DAQmx Write VI

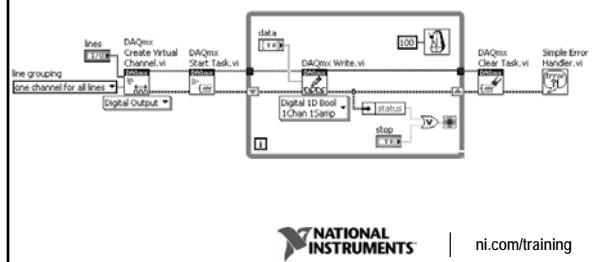
- Use with Digital Output task to generate digital data on digital output channel
- Same data formats as Digital Input task
 - Line
 - Port
 - Waveform



ni.com/training

Digital I/O – DAQmx Write VI Example

- Software-timed digital output



Exercise 5-2: Digital Write

To output digital data from the DAQ device and display updates on LEDs of the BNC-2120.

GOAL

Exercise 5-2: Digital Write

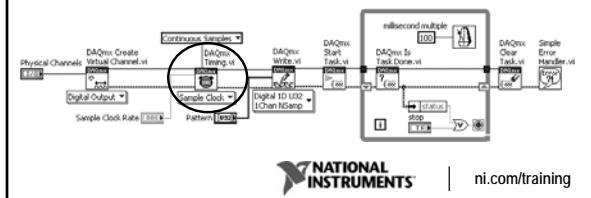
- What happens if you run this VI when the Boolean array is not initialized to 8 elements of data?

DISCUSSION

C. HW-timed DIO – Onboard DIO Sample Clock

Some DAQ devices (X Series) have a dedicated onboard sample clock for digital I/O

- Subject to same rules as buffered analog input and output
- Use sample clock for hardware-timed digital I/O



HW-timed DIO – Correlated Digital I/O

Some DAQ devices (i.e. M Series) do not have a dedicated onboard sample clock for digital I/O timing but do support correlated digital I/O

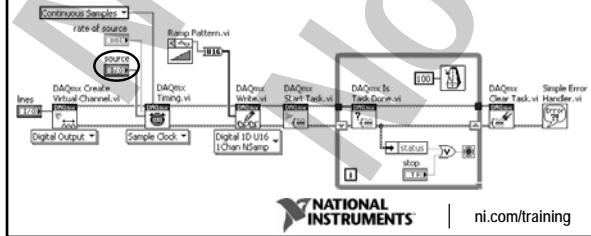
- Correlated digital I/O is digital input or output that is correlated to another clock signal (i.e. AI or AO sample clock, external clock)
- Subject to same rules as buffered analog input and output



HW-timed DIO – Correlated Digital I/O Example

Can use correlated digital I/O to create a digital waveform

- Waveform can be used to transfer digital data
- Digital input can read information from a port



Exercise 5-3: Correlated Digital Output

To output digital data from the DAQ device using the Analog Output Sample Clock as the sample clock and view the digital data on the LEDs of the BNC-2120.

GOAL

Exercise 5-3: Correlated Digital Output

- What clock is the digital output task in this VI using?
- What other clocks could this task use?

DISCUSSION

Summary—Quiz

1. Digital signals are always between 0 and 5 Volts.
 - a) True
 - b) False



| ni.com/training

Summary—Quiz Answer

1. Digital signals are always between 0 and 5 Volts.
 - a) True
 - b) False



ni.com/training

Summary—Quiz

2. All DAQ devices have a dedicated onboard sample clock for digital I/O.
 - a) True
 - b) False



ni.com/training

Summary—Quiz Answer

2. All DAQ devices have a dedicated onboard sample clock for digital I/O.
 - a) True
 - b) False



ni.com/training

Lesson 6 Counters

TOPICS

- A. Counter Overview
- B. Edge Counting
- C. Pulse Generation
- D. Pulse Measurement
- E. Frequency Measurement
- F. Position Measurement

NATIONAL INSTRUMENTS | ni.com/training

A. What is a Counter?

Two basic functions

- To count based on the comparison of input signals (gate, source)
- To generate pulses based upon inputs and register values

Many applications are derived from basic counting

- Edge counting, such as simple edge counting and time measurement
- Pulse, semi-period, and period width measurement
- Frequency measurement
- Single pulse and pulse train generation
- Position and velocity measurement

Gate → [Count Reg] → Out

NATIONAL INSTRUMENTS | ni.com/training

Counter Signals

Counters accept and generate TTL signals

Maximum Rise/Fall Time = 50 ns

+5.0 V
+2.0 V
+0.8 V
0 V

high
indeterminate
low

Minimum Pulse Width = 10 ns

NATIONAL INSTRUMENTS | ni.com/training

Parts of a Counter

- Count Register
 - Stores the current count
- Source
 - Input signal that changes the current count
 - Active edge (rising or falling) of input signal changes the count
 - Choose if count increments or decrements on an active edge
- Gate
 - Input signal that controls when counting occurs
 - Counting can occur when gate is high, low, or between various combinations of rising and falling edges
- Out
 - Output signal used to generate pulses

NATIONAL INSTRUMENTS | ni.com/training

Counter Pins

Counter gate and source are PFI pins

- PFI stands for Programmable Function Input
- Allows use of one pin for multiple applications

Example: Use pin 3 as digital trigger for analog input and counter gate

PFI0/TRIG1	11	45	EXTSTROBE*
PFI1/TRIG2	10	44	DGND
DGND	9	3	PFI2/CONVERT*
+5 V	8	4	PFI3/GPCTR1_SOURCE
DGND	7	1	PFI4/GPCTR1_GATE
PFI5/UPDATE	6	40	GPCTR1_OUT
PFI6/WFTRIG	5	39	DGND
DGND	3	38	PFI7/STARTSCAN
PFI8/GPCTR0_GATE	3	37	PFI8/GPCTR0_SOURCE
GPCTR0_OUT	2	36	DGND

NATIONAL INSTRUMENTS | ni.com/training

Source and Gate Selectors in DAQmx

- Source and gate selection offers a great deal of flexibility
 - The source and gate pins for counters can be used for multiple applications
 - You only need to specify the input terminal for your measurement and DAQmx will connect it to the appropriate source or gate depending on the application
- Counter signals can be input on any PFI pins
- DAQ Assistant will give you the default PFI pin for the application

NATIONAL INSTRUMENTS | ni.com/training

Counter Terminology

- Terminal Count
 - Value of the last count before a counter rolls over to 0
- Resolution
 - The size of the counter register specified in bits
 - Counter register size = $2^{(\text{resolution})} - 1$
 - Typical resolutions - 16, 24, 32 bit
- Timebase
 - Internal signal that can be routed to the source
 - Common timebases - 100 kHz, 20 MHz, 80 MHz, 100 MHz



ni.com/training

DAQ Assistant

Correct PFI line for each type of application is automatically chosen by NI-DAQmx



ni.com/training

B. Edge Counting

Types of Edge Counting

- Simple
- Pause trigger (gated)
- Continuous buffered
- Finite buffered



ni.com/training

Edge Counting

Active edges on source signal increment the count

- Active edge can be either rising or falling

Known frequency timebase

- Time elapsed = (count) x (timebase period)

NATIONAL INSTRUMENTS | ni.com/training

Edge Counting – Simple

- Count increments for the specified edge on the source
- You can change active edge to falling
- Counter will roll over when it reaches terminal count
 - Terminal count = $2^{(\text{Counter resolution})} - 1$

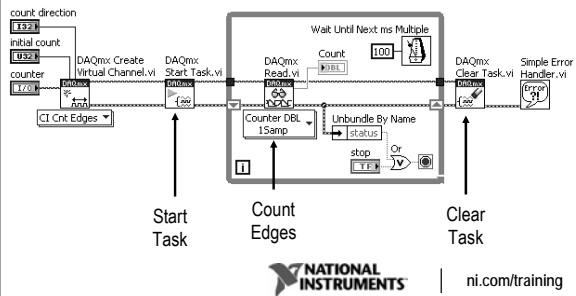
NATIONAL INSTRUMENTS | ni.com/training

Edge Counting – DAQ Assistant

NATIONAL INSTRUMENTS | ni.com/training

Edge Counting – Simple

Use the *Counter* instance of the DAQmx Read VI



Exercise 6-1: Simple Edge Counting

To create a VI to count the number of edges produced by rotating the knob of the Quadrature Encoder.

GOAL

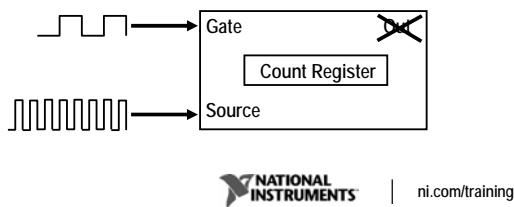
Exercise 6-1: Simple Edge Counting

- What is the highest number that will be output by the count indicator?

DISCUSSION

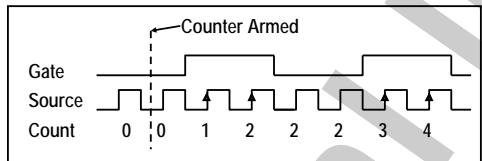
Edge Counting – Pause Trigger (Gated)

- Non-buffered
- Gate pauses increment/decrement of the count register
- Active edges on source increment register only when gate is enabled



Edge Counting – Pause Trigger (Gated)

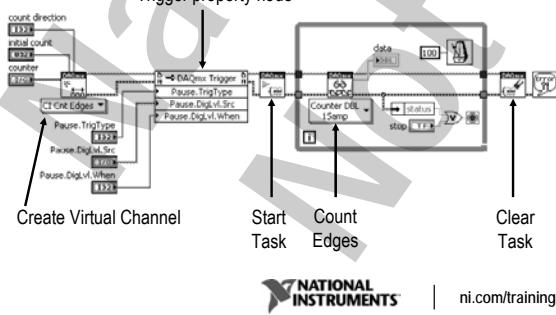
- Counter increments or decrements when gate is high or low (parameters are software selectable)
- Counter will pause counting while in the non-active state



NATIONAL INSTRUMENTS | ni.com/training

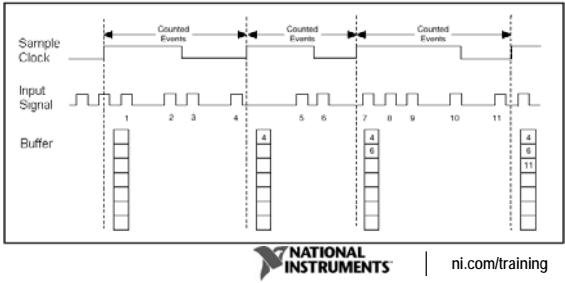
Edge Counting – Pause Trigger Example

Pause trigger is set with DAQmx
Trigger property node



Edge Counting – Continuous Buffered

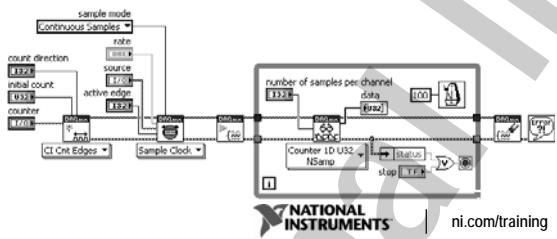
Device latches the number of edges counted onto each active edge of the sample clock and stores the number in the buffer



Edge Counting – Continuous Buffered

Use the DAQmx Timing VI

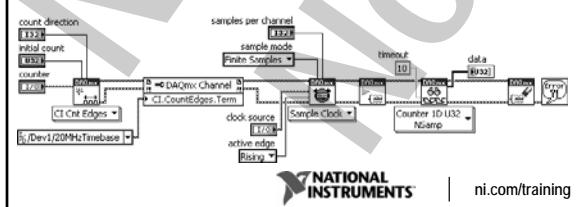
- Source determines when a sample is inserted into the buffer



Edge Counting – Finite Buffered

Use DAQmx Timing VI

- Source determines when a sample is inserted into the buffer
- Counter stops reading when the Samples per Channel value is reached



Exercise 6-2: Advanced Edge Counting

To use a pause trigger and finite buffered methods to perform edge counting.

GOAL

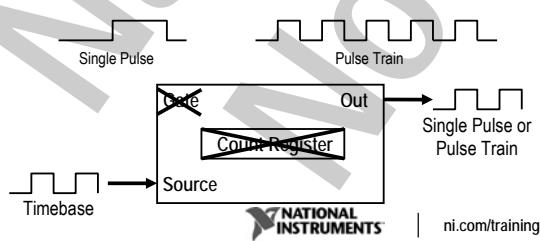
Exercise 6-2: Advanced Edge Counting

- When would finite buffered counting be helpful?

DISCUSSION

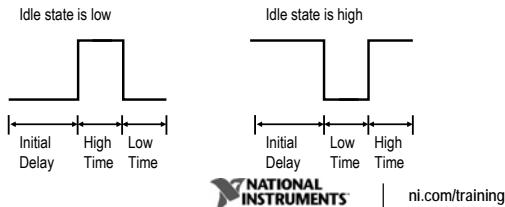
C. Pulse Generation

- A pulse is rapid change in amplitude of a signal from its idle value to an active value for a short period of time
- Generate TTL pulse or pulse train on the counter out pin



Pulse Characteristics

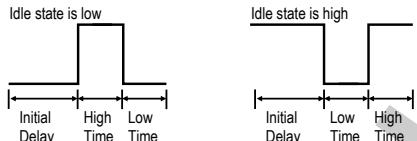
- Can have a low idle state or high idle state
 - Consists of three parts
 - High time, low time, initial delay



NATIONAL INSTRUMENTS

ni.com/training

Pulse Characteristics



$$\text{Pulse Period} = \text{High Time} + \text{Low Time}$$

$$\text{Pulse Frequency} = \frac{1}{\text{Pulse Period}}$$

$$\text{Duty Cycle} = \frac{\text{High Time}}{\text{Pulse Period}}$$

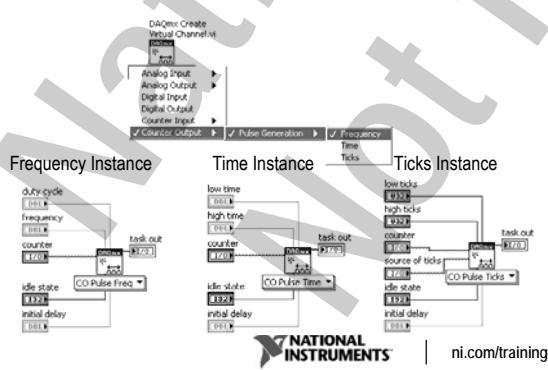
NATIONAL INSTRUMENTS

ni.com/training

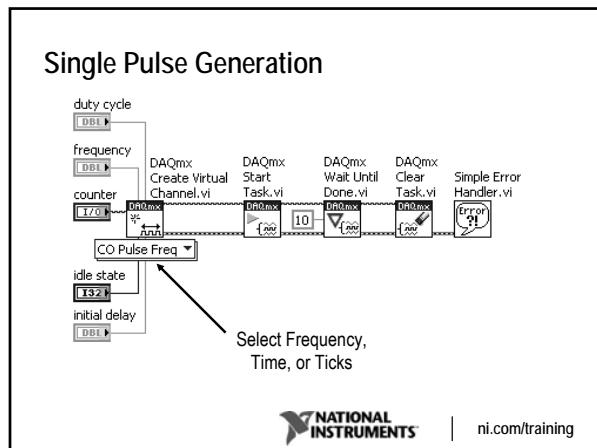
1

— 1 —

DAQmx Create Virtual Channel



ni.com/training



Exercise 6-3: Pulse Generation

To build a VI that generates a single pulse using a counter.

GOAL

Exercise 6-3: Pulse Generation

- This VI defines the pulse by setting the duty cycle and frequency.

How would you modify the VI to define the pulse by setting the high time and low time instead?

DISCUSSION

Pulse Train Generation

- Pulse Train – Generates more than one pulse
- Initial Delay – How long the output remains in the idle state before generation
- High Time – Amount of time the pulse is at a high level (5V)
- Low Time – Amount of time the pulse is at a low level (0V)

Idle state is low

Initial Delay High Time Low Time High Time Low Time High Time Low Time High Time Low Time

NATIONAL INSTRUMENTS | ni.com/training

Pulse Train Generation

duty cycle
frequency
sample mode
counter
idle state
initial delay

Timing VI:
Implicit and continuous

NATIONAL INSTRUMENTS | ni.com/training

Exercise 6-4: Pulse Train Generation

To create a VI to generate a pulse train.

GOAL

Exercise 6-4: Pulse Train Generation

- How would you modify this VI to only output 10 pulses?

DISCUSSION

Finite Pulse Train

- STC2-based devices require the use of 2 counter channels
- First counter generates continuous pulse train
 - Second counter generates pulse that gates first counter
 - Pulse width determines length of pulse train
 - DAQmx uses pulse train settings to determine gating pulse width
 - Everything is managed by DAQmx
 - User simply specifies the counter output channel, pulse train characteristics, and number of pulses to generate

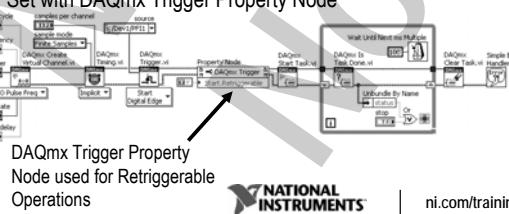
STC3-based devices only use 1 counter channel



ni.com/training

Retriggerable Finite Pulse Train

- Similar to a Finite Pulse Train
- The counter pulse used for gating is retriggerable
 - Can use this property to create a retriggerable finite pulse train
- Set with DAQmx Trigger Property Node



Exercise 6-5: Retriggerable Pulse Train Generation

To build a VI that generates a retriggerable finite pulse train.

GOAL

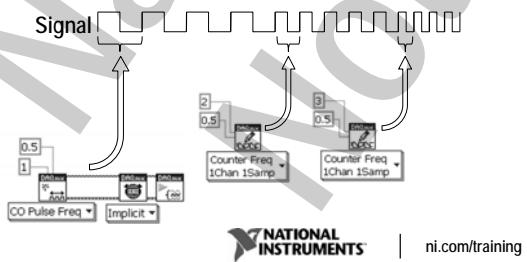
Exercise 6-5: Retriggerable Pulse Train Generation

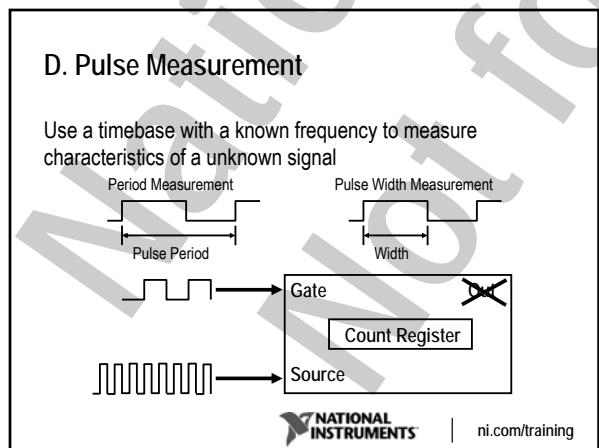
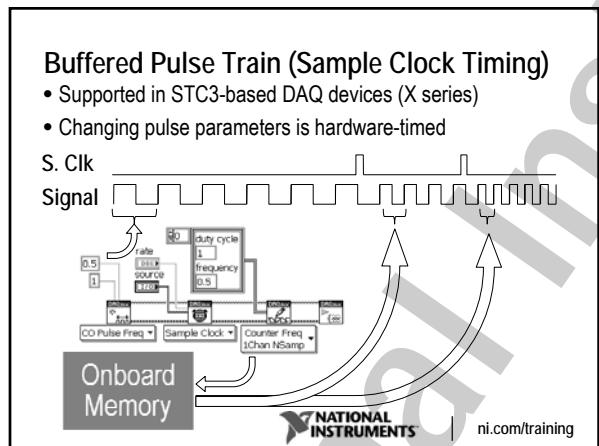
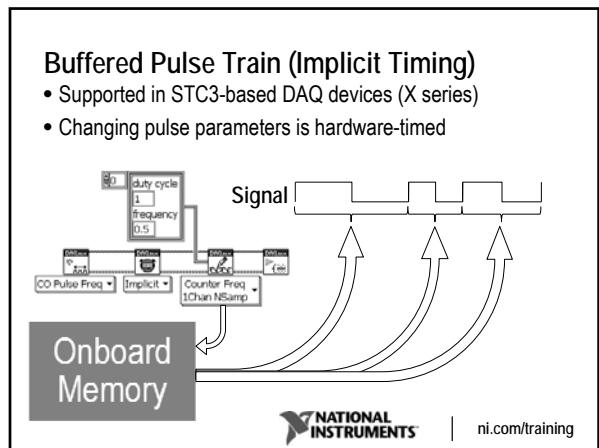
- If the PFI1 line receives another trigger pulse while the VI is outputting 5 pulses, will the VI output another 5 pulses afterwards?

DISCUSSION

Non-Buffered Pulse Train (Implicit Timing)

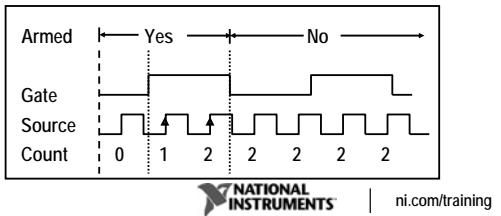
- Supported in STC2-based and STC3-based DAQ devices
- Changing pulse parameters is software-timed





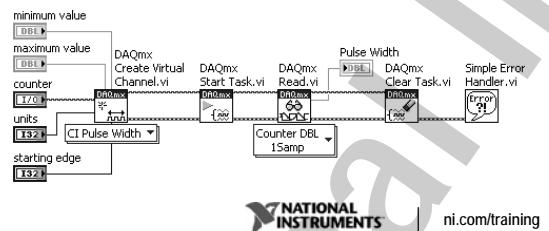
Pulse Width Measurement

- Count will increment for each rising edge on source
 - Counting can start on either rising or falling edge
- Width of Gate = (Count) x (1/source frequency)



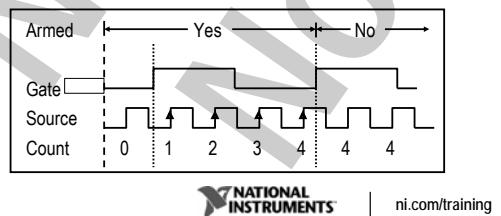
Single Pulse Width Measurement

Set maximum and minimum values of the unknown pulse as accurately as possible. This allows NI-DAQmx to choose the best internal timebase.



Period Measurement

- Count will increment for each rising edge on source
 - Counting can either start and end on rising or falling edges
- Period of Gate = (Count) x (1/source frequency)



Single Period Measurement

Semi Period Measurement

Exercise 6-6: Pulse Width and Period Measurement

To build a VI to measure the pulse width and period of a pulse train.

GOAL

Exercise 6-6: Pulse Width and Period Measurement

- What measurement is the inverse of period?

GOAL

E. Frequency Measurements – STC2 Chip

DAQ devices using NI-STC2 chip have three ways to measure frequency

- Low Frequency with 1 Counter
 - Good for low frequencies (depends on allowable error)
 - Uses 1 counter
- High Frequency with 2 Counters
 - Good for high frequencies (depends on allowable error)
 - Utilizes 2 counters, DAQmx automatically reserves the second
- Large Range with 2 Counters
 - Good for varying frequencies
 - Utilizes 2 counters, DAQmx automatically reserves the second

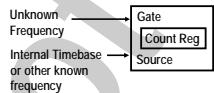


ni.com/training

Frequency Measurement – Low Frequency with 1 Counter Method

Use period measurement and take the inverse

- Frequency = 1 / Period



Pros	Cons
<ul style="list-style-type: none">• Only uses 1 counter• Good at low frequencies ($f < \text{timebase} / 100$)	<ul style="list-style-type: none">• Can see large error at high frequencies due to phenomenon called quantization error



ni.com/training

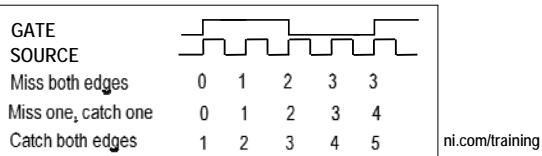
Quantization Error

Quantization error

- Inherent uncertainty in digitizing an analog value as a result of the finite resolution of the conversion process

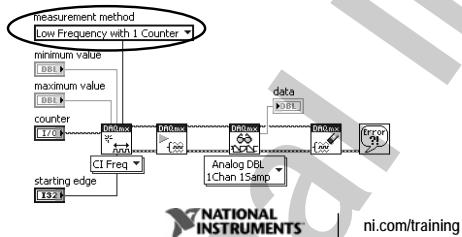
Example

- Gate period is exactly four source cycles
- Measurement could be off by +/- 1 source cycles



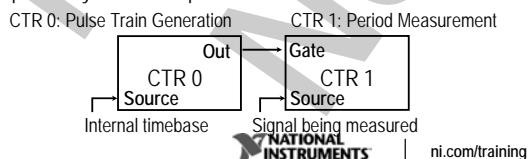
Frequency Measurement – Low Frequency with 1 Counter Method

- Specify the counter input frequency measurement as the application with "Low Frequency with 1 Counter" selected.
- This is an example of a non-buffered, single measurement



Frequency Measurement – High Frequency with 2 Counters Method

- Count the number of edges of the unknown signal during a known period of time (measurement time)
- The longer the period of the known pulse, the less significant the quantization error
- Unknown frequency is calculated by dividing the number of pulses by the known period



Frequency Measurement – High Frequency with 2 Counters Method

- Specify the counter input frequency measurement as the application with “High Frequency with 2 Counters” selected
- Measurement Time specifies length of time for the measurement

NATIONAL INSTRUMENTS | ni.com/training

Frequency Measurement – Large Range with 2 Counters Method

- Divide the unknown signal down to a slower frequency
- Perform period measurement on that slower frequency
- Multiply by divisor value to obtain the correct frequency
- The larger the divisor, the slower it takes to perform period measurement and the more accurate the measurement

CTR 0: Pulse Train Generation CTR 1: Period Measurement

NATIONAL INSTRUMENTS | ni.com/training

Frequency Measurement – Large Range with 2 Counters Method

- Specify the counter input frequency measurement as the application with “Large Range with 2 Counters” selected
- Divisor specifies value to divide the input signal by an integer

NATIONAL INSTRUMENTS | ni.com/training

Frequency Measurements – STC3 Chip

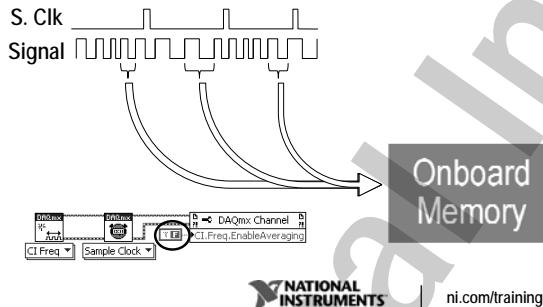
DAQ devices using NI-STC3 chip have two additional ways to measure frequency

Sample Clock timing (averaging disabled)	Sample Clock timing (averaging enabled)
Good for low frequencies	Good for high frequencies and varying range of frequencies
Uses 1 counter	Uses 1 counter



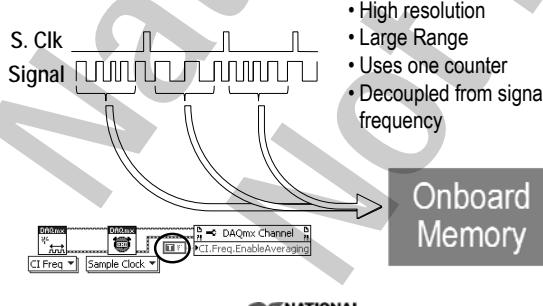
ni.com/training

Sample Clock Timing (Averaging Disabled) with STC3



ni.com/training

Sample Clock Timing (Averaging Enabled) with STC3



ni.com/training

Exercise 6-7: Frequency Measurement

To build a VI to measures frequency using a counter.

GOAL

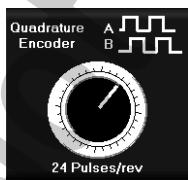
Exercise 6-7: Frequency Measurement

- Can you use the frequency measurement methods discussed for period measurements? (i.e. Low Frequency with 1 Counter, High Frequency with 2 Counters, etc)

DISCUSSION

F. Position Measurement

- You can measure position with a quadrature encoder
- BNC-2120 has a quadrature encoder
- NI-TIO, NI-STC2, and NI-STC3 chips directly support quadrature encoders



ni.com/training

How Does an Encoder Work?

Quadrature Encoder

- Shaft and disk rotate
- Code track either passes or blocks light to sensor
- Light sensor creates two pulse trains

NATIONAL INSTRUMENTS | ni.com/training

Quadrature Encoder

Clockwise Rotation
90° phase difference

Quadrature encoders produce two pulse trains 90 degrees out of phase

Counter-Clockwise Rotation
90° phase difference

Clockwise rotation

- Channel A leads Channel B

 Counter-Clockwise rotation

- Channel B leads Channel A

NATIONAL INSTRUMENTS | ni.com/training

Quadrature Encoder – Decoding

Types of decoding

- X1 – Counter increments on rising edge of A if A leads and decrements on falling edge of A if B leads
- X2 – Counter increments/decrements on rising and falling edges of A depending if A or B leads
- X4 – Counter increments/decrements on rising and falling edges of both A and B depending if A or B leads

ni.com/training

Quadrature Encoder – Z Indexing

Many encoders use a third signal for Z indexing

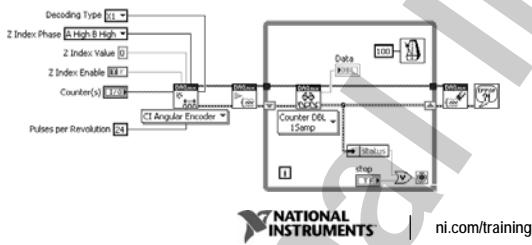
- Produces a pulse at fixed positions
 - For example, at 0 degrees or 45 degrees on an angular encoder
 - Refer to the documentation for an encoder to obtain the timing of signal Z in relation to the A and B signals
- Use for precise determination of a reference position



ni.com/training

Quadrature Encoder Example

Use the Counter Input»Angular Encoder or Counter Input»Linear Encoder instance of the DAQmx Create Virtual Channel VI



Sample Clock Timing with NI-STC3

DAQ devices with NI-STC3 chip can use sample clock timing with the following measurements

- Pulse width measurement
- Pulse measurement
- Frequency and period measurements
- Position measurement
- Two-signal separation measurement



ni.com/training

Summary—Quiz

1. Which of the following are components of a counter?
 - a) Source
 - b) Gate
 - c) Multiplexer
 - d) Register
 - e) Output



ni.com/training

Summary—Quiz Answer

1. Which of the following are components of a counter?
 - a) Source
 - b) Gate
 - c) Multiplexer
 - d) Register
 - e) Output



ni.com/training

Summary—Quiz

2. What is the terminal count of a 24-bit counter?



ni.com/training

Summary—Quiz Answer

2. What is the terminal count of a 24-bit counter?

$$2^{24} - 1 = 16777216 - 1 = 16,777,215$$



ni.com/training

Summary—Quiz

3. What error occurs when the frequency being measured by the Low Frequency with 1 Counter method approaches the timebase of the DAQ device?



ni.com/training

Summary—Quiz Answer

3. What error occurs when the frequency being measured by the Low Frequency with 1 Counter method approaches the timebase of the DAQ device?

Quantization Error



ni.com/training

Lesson 7 Signal Conditioning

TOPICS

- A. Overview of Signal Conditioning
- B. Signal Conditioning Systems
- C. Signal Conditioning for Voltage Measurements
- D. Temperature Measurements
- E. Strain, Pressure, Load, and Torque Measurements
- F. Sound and Vibration Measurements

NATIONAL INSTRUMENTS | ni.com/training

Signal Conditioning Overview

Sensors and signals sometimes require signal conditioning before being measured.

- Different sensors require different signal conditioning

```

graph LR
    Sensor[Sensor Or Signal] -- "I/O" --> DAQHardware[DAQ Hardware]
    DAQHardware -- "Bus" --> DAQSoftware[DAQ Software]
    subgraph SignalConditioning [Signal Conditioning]
        Sensor
        DAQHardware
    end
    
```

NATIONAL INSTRUMENTS | ni.com/training

National Instruments Signal Conditioning Systems

	Low to Medium Channel Count	High Channel Count
Integrated Signal Conditioning		
Front End Signal Conditioning		

NATIONAL INSTRUMENTS | ni.com/training

CompactDAQ – Integrated, Low to Medium Channel Count

- Connectors, signal conditioning, and DAQ integrated into chassis and modules
- Ideal for portable measurement systems
- Hot-swappable modules
- Up to 24-bit accuracy
- Input rates up to 500kS/s for analog
- Cross-platform devices
 - CompactRIO
 - CompactDAQ
 - USB sleeve

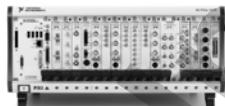


NATIONAL INSTRUMENTS

ni.com/training

PXI Express – Integrated, High Channel Count

- Integrated signal conditioning for bridge, thermocouple, vibration, etc
- Dedicated bandwidth per device
- Up to 24-bit resolution
- Programmable gain and filter settings
- NIST-traceable calibration certificates
- >500 channels per chassis
- Device and multi-chassis synchronization using PXI platform



NATIONAL INSTRUMENTS

ni.com/training

SCXI – Front-end, High Channel Count

- Front-end signal conditioning platform for multifunction DAQ devices
- Multiplexer, matrix, and general-purpose switching
- Wide range of analog and digital conditioning options
- Reconfigurable desktop, rack-mount and portable solution
- Several chassis options; ideal for high channel-count systems



NATIONAL INSTRUMENTS

ni.com/training

Signal Conditioning for Voltage Measurements

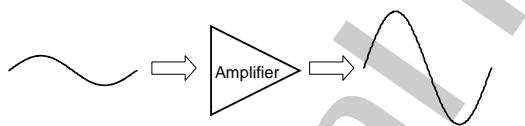
- Amplification
- Attenuation
- Isolation
- Filtering



ni.com/training

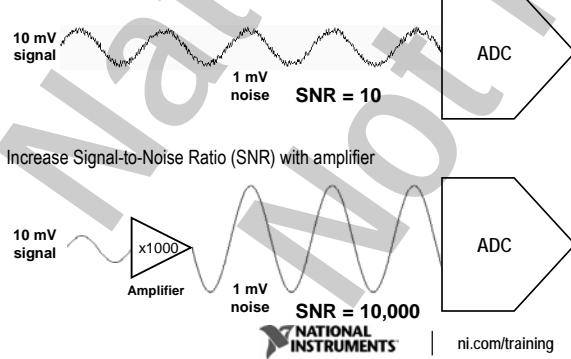
Amplification

- Used on low-level signals
- Maximizes use of Analog-to-Digital Converter (ADC) range and increases accuracy
- Increases Signal-to-Noise Ratio (SNR)



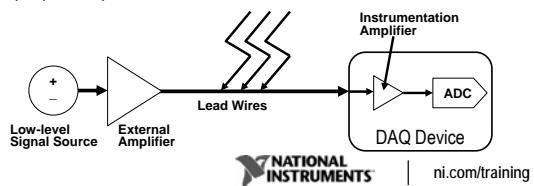
ni.com/training

Amplification



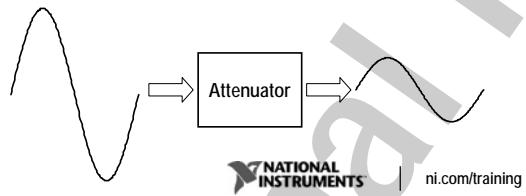
Amplification

- Amplify the low-level voltages near the signal source to reduce the effects of noise
- Configure the minimum and maximum in DAQmx programming so DAQ device will automatically select the proper amplification to best utilize DAQ device resolution

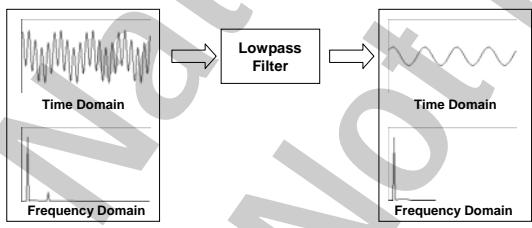


Attenuation

- Decreases the input signal amplitude to fit within the range of the DAQ device
- Necessary when input signal voltages are beyond the range of the DAQ device

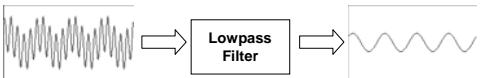


Filtering

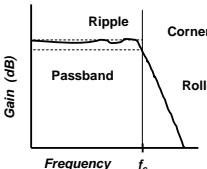


Removes noise
Blocks unwanted frequencies

Filtering



Bode Plot



- Passband – frequencies the filter lets pass
- Ripple - filter's effect on the signal's amplitude
- Corner – frequency where the filter begins blocking the signal
- Rolloff – how sharply the filter cuts off unwanted frequencies

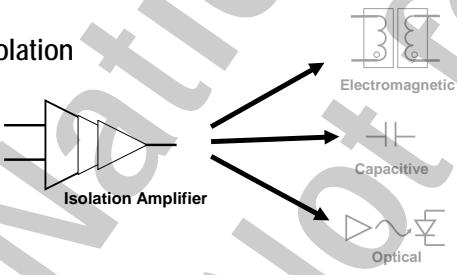
NATIONAL INSTRUMENTS | ni.com/training

Filtering – Examples

- Remove/reject unwanted noise within certain frequency range
 - 50/60 Hz noise rejected by lowpass 4Hz filter
 - Some DAQ devices have built-in programmable filters
- Prevent noise caused by aliasing from higher frequencies
 - Implement an anti-aliasing filter with a lowpass filter to remove frequency components greater than one half the sampling frequency (Nyquist Theorem)

NATIONAL INSTRUMENTS | ni.com/training

Isolation



Pass signal from its source to measurement device without a physical connection

- Blocks high common-mode signals
- Breaks ground loops
- Protects your instrumentation

NATIONAL INSTRUMENTS | ni.com/training

Isolation – Referencing / Common-Mode Voltage

- Floating signals – reference to your instrumentation
- Common-mode voltage – can damage your amplifier
- Ground loops – caused by multiple reference points



ni.com/training

Isolation Specifications

Isolation Committees – UL, IEC

Component requirements
Spacing requirements
Testing procedures

Safe Working Voltage

Defines acceptable normal voltage you can apply

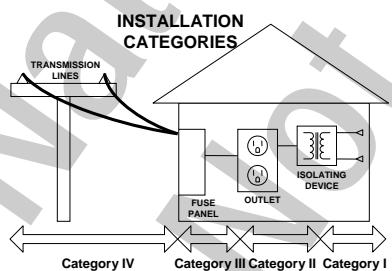
Installation Rating

Defines the possible transients your system can accept



ni.com/training

Installation Ratings



ni.com/training

Most Common Sensor Measurements

- Temperature
- Strain, pressure, load, and torque
 - Bridge-based measurements
- Sound and vibration
 - Closely related measurements



ni.com/training

Temperature Sensor Measurements

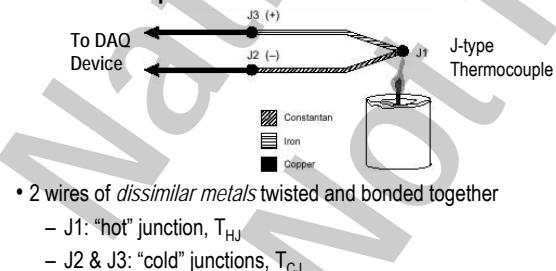
Temperature Sensors

- Thermocouple
- Resistance Temperature Detectors (RTD)
- Thermistor (thermally sensitive resistors)



ni.com/training

Thermocouple Construction



ni.com/training

Thermocouple Output

Temp Diff (F)	E (mV)	J (mV)	K (mV)	R (mV)
0	0	0	0	0
1000	~40	~20	~10	~5
2000	~60	~30	~15	~8
3000	~70	~40	~20	~10

- Voltage signal of a thermocouple is proportional to temperature at the hot junction
- The voltage vs. temperature relationship is nonlinear over large temperature ranges

NATIONAL INSTRUMENTS | ni.com/training

Thermocouple Color Codes

Type	Material		Color Code		Overall Jacket		Range (°C)	
Thermocouple Grade	Positive Wire	Negative Wire	Positive Wire	Negative Wire	Extension	Overall Jacket	min	max
J	Iron	Constantan	White	Red	Black	Brown	0	750
K	Chromel	Alumel	Yellow	Red	Yellow	Brown	-200	1250
T	Copper	Constantan	Blue	Red	Blue	Brown	-200	350
E	Chromel	Constantan	Purple	Red	Purple	Brown	-200	900

Note: Unlike most leads, the red lead for thermocouples is negative

NATIONAL INSTRUMENTS | ni.com/training

Thermocouple Operation

- Due to the "Seebeck Effect," dissimilar metals in contact produce a voltage (mV) proportional to the temperature difference between the hot (J1) and cold junctions (J2, J3)

To DAQ Device

J3 (+)
J2 (-)
J1

J-type Thermocouple

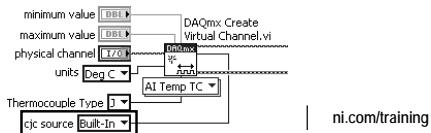
Legend:

- Constantan
- Iron
- Copper

NATIONAL INSTRUMENTS | ni.com/training

Cold Junction Compensation Signal Conditioning

- CJC signal conditioning accounts and corrects for the voltage added at cold junctions
- Some NI DAQ devices use a direct-reading sensor at the reference junction
 - Direct-reading sensor measures the reference-junction temperature
 - Software adds the appropriate voltage value to the measured voltage to cancel out the cold junction voltage



ni.com/training

Additional Signal Conditioning for Thermocouples

- Filtering
 - Thermocouples output low voltage (mV) signals, making them susceptible to noise
 - Use lowpass filter to remove 60 Hz power line noise, etc
- Amplification
 - Improve noise performance by amplifying the low voltage signals near the signal source
- Isolation
 - Thermocouples commonly mounted/soldered directly to a conductive material (i.e. steel, water) making them susceptible to common-mode noise and ground loops



ni.com/training

Exercise 7-1: Thermocouple Measurement

To read the temperature from a thermocouple with the NI 9219.

GOAL

Exercise 7-1: Thermocouple Measurement

- How does the VI and NI 9219 compensate for voltages added at cold junctions?

DISCUSSION

Active and Passive Temperature Sensors

Thermocouples are passive temperature sensors

- Do not require excitation
- RTDs and Thermistors are active temperature sensors
- Require current or voltage excitation



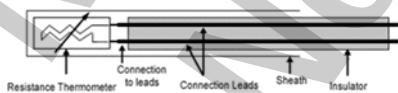
Excitation – A sensor that requires excitation must receive a voltage or current from an external source



ni.com/training

Resistance Temperature Detectors (RTDs)

- Operate on the principle of changes in electrical resistance of pure metals
- Characterized by a linear positive change in resistance with temperature



Physical Architecture of an RTD



ni.com/training

Thermistors

Thermistors (thermally sensitive resistors)

- Similar to RTDs in that they are electrical resistors whose resistance changes with temperature
- Manufactured from metal oxide semiconductor material which is encapsulated in a glass or epoxy bead



ni.com/training

Wire Varieties

RTDs and thermistors come in the following wire varieties

- Two-wire
- Three-wire
- Four-wire

Each requires a different wiring to DAQ device



ni.com/training

Additional Signal Conditioning for RTDs and Thermistors

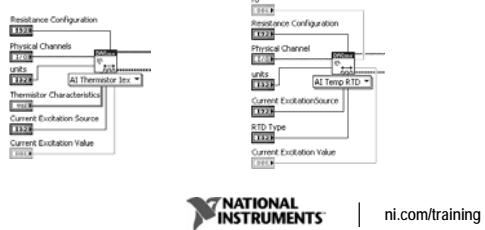
- Lowpass filter
 - Both sensors output low voltage (mV) signals
 - Use lowpass filter to remove 60 Hz power line noise, etc
- Amplification
 - Improve noise performance by amplifying the low-voltage signals near the signal source



ni.com/training

RTD and Thermistor Example

Use the DAQmx Create Virtual Channel to configure RTD and thermistor parameters



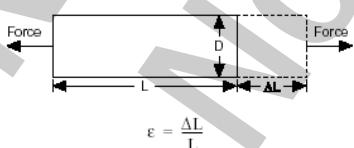
Temperature Sensor Comparison

	Thermocouple	RTD	Thermistor
Temperature Range	-267°C to 2316°C	-260°C to 850°C	-100°C to 500°C
Accuracy	Good	Best	Good
Linearity	Better	Best	Good
Sensitivity	Good	Better	Best



E. Strain, Pressure, Load, and Torque Measurements

- Strain is the amount of deformation of a body due to an applied force
- Strain (ϵ) is defined as the fractional change in length as shown below



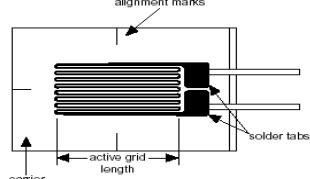
$$\epsilon = \frac{\Delta L}{L}$$



Strain Gage Construction



- Strain gages measure strain
- Thin wire or metal foil in a zigzag pattern is fastened to a "carrier"



NATIONAL INSTRUMENTS | ni.com/training

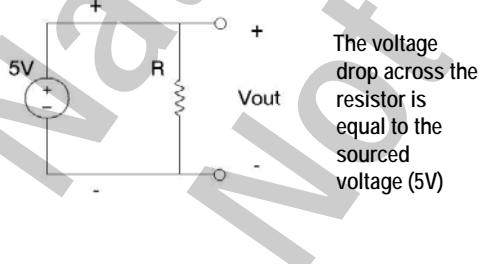
Strain – Gage Factor

- Fundamental parameter of the strain gage is its sensitivity to strain
- Expressed as the gage factor (GF)
- Gage factor is defined as the ratio of fractional change in electrical resistance to the fractional change in length (strain)

$$GF = \frac{\Delta R / R}{\Delta L / L} = \frac{\Delta R / R}{\varepsilon}$$

NATIONAL INSTRUMENTS | ni.com/training

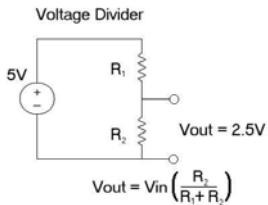
Strain – Circuit Basics



The voltage drop across the resistor is equal to the sourced voltage (5V)

NATIONAL INSTRUMENTS | ni.com/training

Strain – Circuit Basics

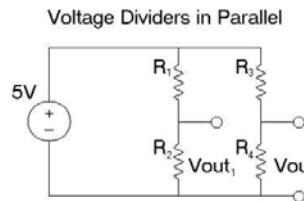


If $R_1=R_2$ then voltage is dropped equally across the resistors (2.5V across each)



ni.com/training

Strain – Circuit Basics

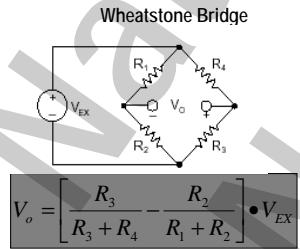


If all resistors are equal then voltage is dropped equally across them (2.50 V across each)



ni.com/training

Strain – Circuit Basics



- If all resistors are equal then no voltage is read.
- If one or more resistors change, voltage is returned.

Wheatstone Bridge measures small changes in resistance

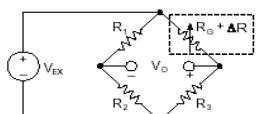


ni.com/training

Strain – Circuit Basics

If you replace a resistor with an active strain gage in the Wheatstone Bridge, any changes in the strain gage resistance will unbalance the Bridge

Quarter Bridge Strain Gage



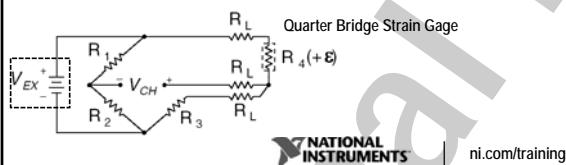
$$\frac{V_O}{V_{EX}} = -\frac{GF \cdot \varepsilon}{4} \left(\frac{1}{1 + GF \cdot \frac{\varepsilon}{2}} \right)$$



ni.com/training

Required Signal Conditioning for Strain Gages

- Excitation
 - Strain gages require voltage excitation levels of between 2.5 V and 10 V
- Bridge Completion
 - Must complete the Wheatstone Bridge in the DAQ device



ni.com/training

Bridge Terminology

Bridge Configuration	# of Active Elements	# of Active Elements outside Terminal Block
Quarter-bridge	1	1
Half-bridge	2	2
Full-bridge	4	4

Alignment and wiring of active elements can enhance or minimize certain strains



ni.com/training

Bridge Completion

- Quarter Bridge I
- Quarter Bridge II
- Half Bridge I
- Half Bridge II

 ni.com/training

Bridge Completion (cont.)

- Full Bridge I
- Full Bridge II
- Full Bridge III

 ni.com/training

Optional Signal Conditioning for Strain Gages

- Remote Sensing
- Offset Nulling
- Shunt Calibration
- Amplification
- Filtering

 ni.com/training

Remote Sensing

- If strain gage circuit is far from the signal conditioner and excitation source, it might introduce a source of error caused by a voltage drop from the resistance in the connecting wires
- Use remote sensing to compensate for this error



NATIONAL INSTRUMENTS

ni.com/training

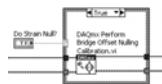
Offset Nulling

Purpose

- Ensures that ~zero volts are produced at rest (no strain)

Offset Nulling

- Essentially an offset calibration
- Can be performed in HW or SW
- Compensates for inherent bridge imbalance
- Coarse and fine potentiometers are used to perform nulling in hardware



NATIONAL INSTRUMENTS

ni.com/training

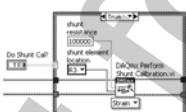
Shunt Calibration

Purpose

- Verify the output of a strain gage measurement system relative to some predetermined mechanical input or strain

Shunt Calibration

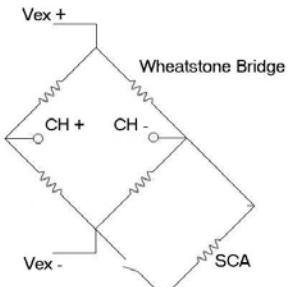
- Essentially a gain calibration
- Known resistance is introduced to the circuit and the measured strain is compared to the expected value
- Correction factor is applied to all subsequent readings



NATIONAL INSTRUMENTS

ni.com/training

Shunt Calibration Circuit



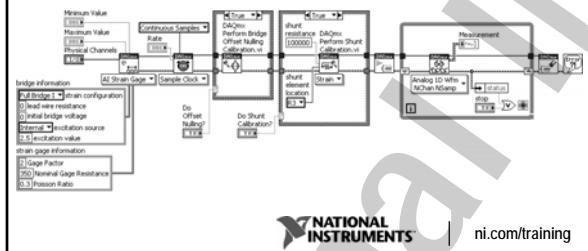
- Shunt resistor is in parallel to one of the bridge resistors
- The shunt circuit is engaged when the switch is closed (done programmatically)



ni.com/training

Strain Gage Example

- Configure bridge information and strain gage information
- Includes options for offset nulling and shunt calibration



ni.com/training

Load, Pressure, Torque

Strain gages also used in common load cells, pressure sensors, and torque sensors

- Load cell
 - Uses array of strain gages to measure deformation, in proportion to force, of a structural member
- Pressure sensor
 - Uses strain gages mounted to a diaphragm to measure the deformation, in proportion to pressure, of the diaphragm
- Torque sensor
 - Uses strain gages affixed to a torsion bar to measure the shear stress, in proportion to torque, of the bar



ni.com/training

Exercise 7-2: Strain Measurement

To acquire data from a strain gage connected to the NI 9219 and apply offset nulling for more accurate strain measurements.

GOAL

Exercise 7-2: Strain Measurement

- What are possible consequences of not performing offset nulling?

DISCUSSION

Sound and Vibration Measurements

- Sound and vibration are closely related
- Vibration occurs when a mass oscillates mechanically about an equilibrium point
 - Sound consists of pressure waves generated by vibrating structures. Pressure waves can also induce the vibration of structures

Examples

- Electrical power turbine
- Airplane wing surface
- Musical instruments



ni.com/training

Sound and Vibration Measurements

Wide range of applications ,common examples

- Acoustic and vibration testing
 - i.e. noise, vibration, and harshness
- Audio testing
 - i.e. measuring the sound of an audio speaker
- Machine monitoring
 - i.e. predicting failure on power turbines



ni.com/training

Sound and Vibration Measurements

Sound sensor

- Microphones measure sound pressure level

Vibration sensor

- Accelerometers measure acceleration typically based on piezoelectric theory

Sound and vibration sensors both measure oscillations

- Similar theory, measurement, and signal conditioning
- Typically both microphones and accelerometers have an integrated amplifier to boost the signal before acquisition



ni.com/training

Sound and Vibration Signal Conditioning

- Excitation (current)
- AC coupling to remove DC offset
- Flexible gain settings for different signal ranges
- Lowpass filter to reduce noise and prevent aliasing

Benefits of NI sound & vibration DAQ hardware

- 24-bit resolution for improved dynamic range
- Each channel has its own ADC for synchronized measurements



ni.com/training

Sound and Vibration Analysis Examples

Sound and Vibration Measurement Suite contains functions and graphs for analysis and visualization

- Sound Quality, Spectral Analysis, Zoom Power Spectrum, and Frequency Response
- ANSI and IEC-compliant full and fractional octave analysis
- Order analysis tracking and extraction including tachometer processing
- Waterfall, cascade, shaft centerline, orbit, bode, and polar plots
- Universal File Format (UFF58) file I/O support



ni.com/training

Summary—Quiz

1. Name 5 types of signal conditioning.



ni.com/training

Summary—Quiz Answers

1. Name 5 types of signal conditioning.

CJC Compensation
Bridge Completion
Offset Nulling
Amplification
Filtering
Isolation
...



ni.com/training

Summary—Quiz

2. The arrangement of the strain gages is inconsequential.
- a) True
 - b) False



ni.com/training

Summary—Quiz Answer

2. The arrangement of the strain gages is inconsequential.
- a) True
 - b) False



ni.com/training

Summary—Quiz

3. Offset nulling is never necessary because most Wheatstone bridge measurements output exactly 0 V when no strain is applied.
- a) True
 - b) False



ni.com/training

Summary—Quiz Answer

3. Offset nulling is never necessary because most Wheatstone bridge measurements output exactly 0 V when no strain is applied.
- a) True
 - b) False



ni.com/training

Summary—Quiz

4. Which of the following types of signal conditioning can apply to thermocouple measurements?
- a) CJC compensation
 - b) Amplification
 - c) Excitation
 - d) Filtering
 - e) Isolation



ni.com/training

Summary—Quiz Answers

4. Which of the following types of signal conditioning can apply to thermocouple measurements?
- a) CJC compensation
 - b) Amplification
 - c) Excitation
 - d) Filtering
 - e) Isolation



ni.com/training

Lesson 8 Synchronization

TOPICS

- A. Synchronizing Measurements
- B. Single Device Synchronization
- C. Multiple Device Synchronization
- D. Counters and Synchronization

NATIONAL INSTRUMENTS | ni.com/training

A. Synchronizing Measurements

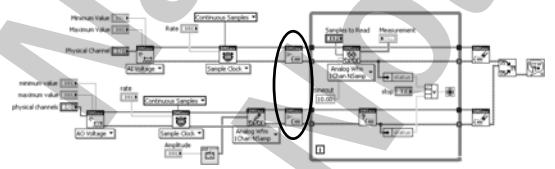
Many applications require more than one type of measurement at a time

- Simultaneous Measurements
 - Operations are happening at the same time but are not necessarily synchronized
 - Cannot prove that measurements occurred at the same instant
- Synchronous Measurements
 - Measurements are correlated

NATIONAL INSTRUMENTS | ni.com/training

Example: Simultaneous Analog Input/Output

Problem: Simultaneous measurements, but not synchronous



NATIONAL INSTRUMENTS | ni.com/training

Example: Simultaneous Analog Input/Output

Timing diagram for simultaneous measurements

- AI and AO clocks are based on different timebases
 - Because the timebases are different, the clock signals are out of phase
- Start Triggers for this VI come from the DAQmx Start Task VIs
 - Due to software timing, start triggers can be 100s of milliseconds off from one another

NATIONAL INSTRUMENTS | ni.com/training

Synchronization Rules

- Share a Master Timebase* and a Start Trigger
 - Sharing a Master Timebase
 - Prevents phase discrepancies
 - Allows for different rates derived from the timebase
 - Sharing a Start Trigger
 - Verifies that tasks start synchronously
- Share a Sample Clock
 - Clocked analog and digital measurements only update on a new rising edge from a sample clock
 - Sample Clock derived from master timebase or externally sourced

*Also known as a Reference Clock

NATIONAL INSTRUMENTS | ni.com/training

Synchronization Rules

Share a Master Timebase and Trigger

- Both sample clocks are derived from the same timebase
 - For a single board, the timebase is always assumed to be the same
 - For multiple boards, must share the timebase to avoid phase errors
- Different sample clocks can be set to different rates
- All tasks have start triggers
 - If not explicitly created, comes from when the software starts the task.

NATIONAL INSTRUMENTS | ni.com/training

Synchronization Rules within DAQmx

1. Create one master task
 - Configure master task, but do not start immediately
 - Only start master task after all slave tasks have started
2. Create as many slave tasks as necessary
 - Start slave tasks before starting master task



ni.com/training

Sources of Error

There are several sources of error when synchronizing measurements:

- Jitter
 - Small variations in the period of the clock (from sample to sample)
 - Each component added to the clock's path adds additional jitter
- Stability
 - Can be subject to variations due to temperature, aging, etc.
- Accuracy
 - An oscillator never generates a perfect frequency
 - Error expressed in parts per million (ppm) and parts per billion (ppb)
- Skew
 - Propagation delay that is caused when a signal arrives at two places at different times
 - Affected by distance and impedance of signal paths



ni.com/training

B. Single Device Synchronization

- All tasks require a start trigger of some sort
 - But not all measurements require an external hardware trigger
- If a trigger is not explicitly configured for a task, the software creates an implicitly called start trigger.
 - The internal start trigger is called <Device>/<Type>/StartTrigger
 - For example: Dev1/ai/StartTrigger
- Can use the internal trigger to guarantee that one task does not start before the other



ni.com/training

Example: Simultaneous Analog Input/Output

Solution: Trigger slave task start on the start of the master task

*The analog output task starts, but must wait for the analog input task to receive a trigger

NATIONAL INSTRUMENTS | ni.com/training

Exercise 8-1: Simultaneously Started Analog Input and Output

To become familiar with the two different ways to simultaneously start an analog input and analog output operation.

GOAL

Exercise 8-1: Simultaneously Started Analog Input and Output

- After the analog input and output are simultaneously started, is the acquisition and generation synchronized?

GOAL

Synchronous Measurements

Share a Sample Clock

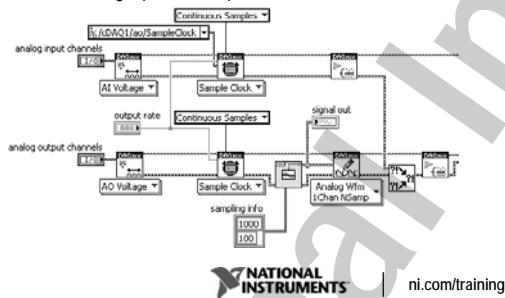
- Synchronous Measurements always run at the same rate
- Create synchronous measurements by sharing a sample clock between different tasks
 - Input tasks latch the value on a clock edge
 - Output tasks update a value on a clock edge
- If no clock edges are seen, no values are updated or latched



ni.com/training

Example: Synchronous Measurements

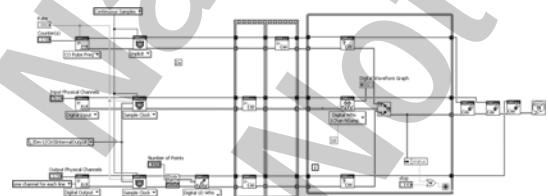
Synchronous Analog Input and Output



ni.com/training

Example: Synchronize Multiple Tasks

Use the counter's internal output to synchronize multiple tasks



Note: This VI can only work on a device that supports clocked digital I/O



ni.com/training

C. Multiple Device Synchronization

Same rules apply for multiple devices as apply for a single device:

1. Share a Master Timebase and a Start Trigger
2. Share a Sample Clock

NI-DAQmx Rules

1. Create one master task

- Configure master task, but do not start immediately
- Only start master task after all slave tasks have started

2. Create as many slave tasks as necessary

- Start slave tasks before starting master task



ni.com/training

Multi-Device Synchronization

Synchronization Bus

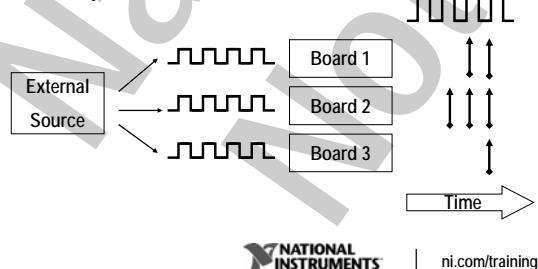
- Data Transfer Bus cannot pass timing and synchronization signals
- If sharing internal signals between devices, must setup a synchronization bus
 - Must physically connect all boards to be synchronized with a synchronization bus cable
- USB Devices have no synchronization bus
 - Must synchronize USB devices with external sample clock



ni.com/training

External Signal Connections

Different boards may start at different clock edges due to skew and jitter

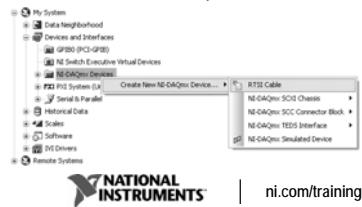


ni.com/training

Multi-Device Routing in DAQmx

Real-Time System Integration (RTSI) Bus

- RTSI Cables are used for sharing timing and synchronization lines with other devices
- Setup RTSI Cable in MAX
- Must physically connect the RTSI Cable to all boards to be synchronized
 - Connector is on top back of board
- Once configured DAQmx automatically handles routing across RTSI bus



NATIONAL
INSTRUMENTS

ni.com/training

RTSI Bus



- RTSI Connector
- System Integration
 - Allows you to share timing signals between multiple devices

NATIONAL
INSTRUMENTS

ni.com/training

RTSI

- Most NI DAQ Devices support RTSI
 - PCI Devices: need a RTSI cable
 - PXI Devices: built in to the PXI chassis, referred to as the PXI Trigger Bus
- Distinguishes PXI devices from CPCI devices
- Can pass timing signals up to 20 MHz
- As with other signal connections, trigger latencies and transmission line effects become pronounced at high frequencies

NATIONAL
INSTRUMENTS

ni.com/training

Programming with RTSI

- Management of the RTSI bus is hidden from user
- PCI Systems: Must register RTSI cable in MAX
- PXI Systems: Must register PXI system in MAX
- USB Systems: Must explicitly route lines to a PFI and physically connect PFI lines between multiple devices
- Use Export Signal.vi to explicitly route signals to RTSI



ni.com/training

Multiple Device Simultaneous Operations

Much like with single devices, the common use cases for multiple device simultaneous operations are:

- Start multiple board operations simultaneously
 - Can use a hardware start trigger
- Synchronize multiple board operations
 - Can synchronize inputs and outputs on multiple devices
- Synchronize multiple board operations and start simultaneously



ni.com/training

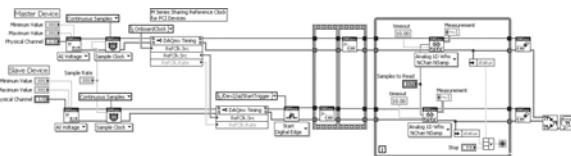
Create a RTSI Cable in MAX

Create a RTSI Cable in MAX for the synchronization of multiple devices

DEMONSTRATION

Example: Multiple Device Analog Input

Share a Timebase and a Start Trigger – M Series (PCI)



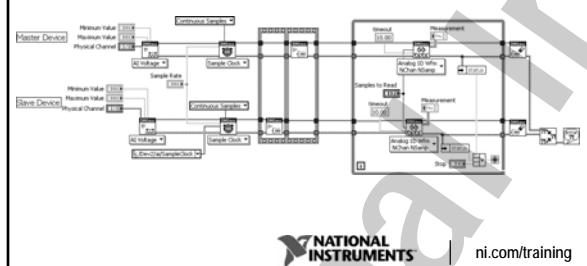
Note: For CompactDAQ the Reference Clock and Start Trigger must be routed out of a PFI line on the master device and into a PFI line on the slave device(s).



ni.com/training

Example: Multiple Device Analog Input

Share a Sample Clock



Multi-Device Synchronization Resources

- Find Examples > Hardware Input and Output > DAQmx > Synchronization > Multi-Device
- Examples help to show caveats of using different devices with one another



ni.com/training

Exercise 8-2: Synchronous Analog Input and Output

To perform a synchronized analog input and output operation by sharing the AO Sample Clock.

GOAL

Exercise 8-2: Synchronous Analog Input and Output

- How would you modify the VI to add a digital output task to generate data at the same time as the thermocouple and analog output tasks?

DISCUSSION

D. Counters and Synchronization

Counters have better flexibility than the clocks for analog or digital inputs and outputs

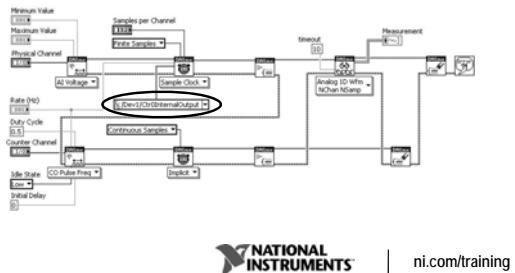
- Use the counters as clock sources for analog & digital tasks
 - Allows for the creation of
 - Retriggerable input tasks (Retriggerable Finite Pulse Train)
 - Variable rate input/output (Continuous Pulse Train)
 - Trigger after N pulses (Single Pulse with Initial Delay)
- Use AI/AO Sample Clock as the gate for counter measurements
 - Synchronize counter measurements with analog measurements



| ni.com/training

Example: Counters and Analog Tasks

Internal Output of counter pulse train used for analog input clock



Retriggerable AI/AO

- For STC2-based devices, AI/AO operations are not retriggerable but counters are
 - Use counters to create retriggerable finite pulse train
 - AI/AO performs continuous operation using the retriggerable finite counter output as the sample clock
- Note: Certain STC3-based devices such as X Series can use the Start.Retriggerable property in the DAQmx Trigger property node for AI/AO tasks instead of using counters

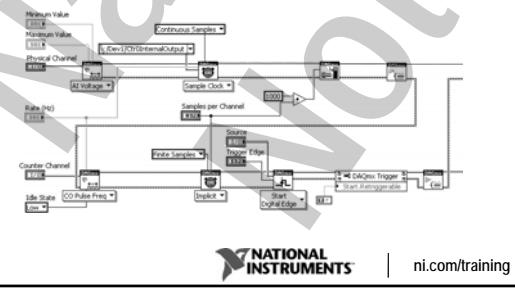


NATIONAL INSTRUMENTS

| ni.com/training

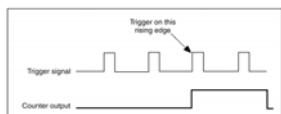
Retriggerable Analog Input Example (STC2-based Devices)

Use a retriggerable counter as the clock source for a continuous analog input.



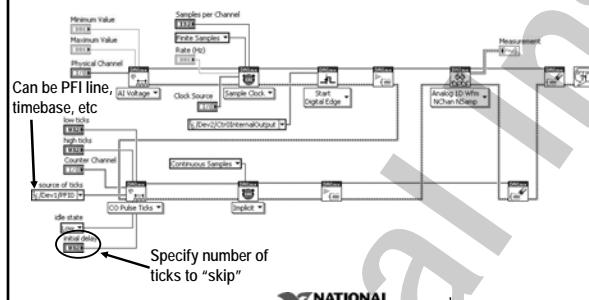
Event Triggering

- Allows you to trigger on the Nth trigger
- Uses a counter to monitor the pulses of your signal
- Actual trigger is generated on counter's output pin



ni.com/training

Event Triggering



ni.com/training

Use Sample Clocks with Counters

Counter Input:

- Can use an analog sample clock as the gate for a buffered counter measurement.
- Allows for counter measurements to be correlated to analog/digital measurements

Counter Output:

- Generate a pulse for every N ticks of the sample clock
 - For Example: Want to use counter as a trigger for a frame grabber.
 - Grab 1 frame for every 1000 analog input samples
 - Counter outputs a pulse every 1000th tick of the sample clock



ni.com/training

Summary—Quiz

1. To simultaneously start and synchronize multiple tasks, you must share which of the following?
 - a) Master Timebase
 - b) Physical Channel
 - c) Sample Clock
 - d) Indicator
 - e) Trigger



ni.com/training

Summary—Quiz Answer

1. To simultaneously start and synchronize multiple tasks, you must share which of the following?
 - a) Master Timebase
 - b) Physical Channel
 - c) Sample Clock
 - d) Indicator
 - e) Trigger



ni.com/training

Summary—Quiz

2. To synchronize multiple boards, which of the following could be used?
 - a) GPS
 - b) RTSI Bus
 - c) External Clock
 - d) PXI Trigger Bus



ni.com/training

Summary—Quiz Answer

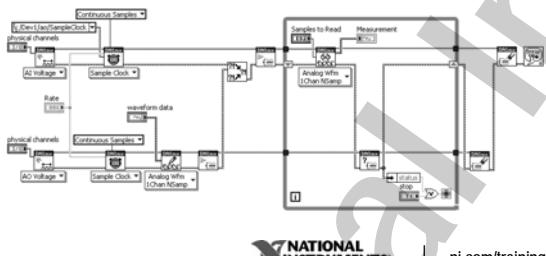
2. To synchronize multiple boards, which of the following could be used?
- GPS
 - RTSI Bus
 - External Clock
 - PXI Trigger Bus



ni.com/training

Summary—Quiz

3. Is this measurement synchronized? If so, why? If not, what prevents it from being synchronized?



ni.com/training

Summary—Quiz Answer

3. Is this measurement synchronized? If so, why? If not, what prevents it from being synchronized?

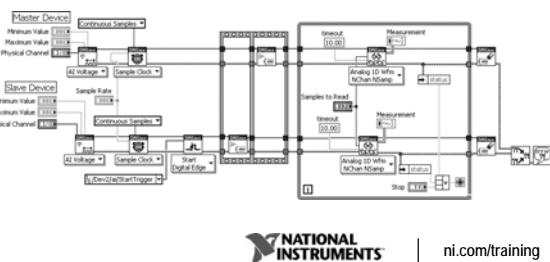
No, this measurement is not synchronized. The analog output begins creating a sample clock before the analog input task has started.



ni.com/training

Summary—Quiz

4. Is this measurement synchronized? If so, why? If not, what prevents it from being synchronized?



ni.com/training

Summary—Quiz Answer

4. Is this measurement synchronized? If so, why? If not, what prevents it from being synchronized?

No. The measurements share a start trigger, but not a master timebase; therefore, the measurements will be out of phase.



ni.com/training

Continuing Your LabVIEW Education

- Instructor Led Training
 - LabVIEW Core 2: Learn about multiple loop design patterns, property nodes and building an executable
 - Hardware courses such as LabVIEW Instrument Control and LabVIEW FPGA
 - Online courses such as Machine Vision and LabVIEW Real-Time
- Self-Paced: a variety of instructional packages and tools designed to educate you at your own pace



ni.com/training

Additional Information and Resources

This appendix contains additional information about National Instruments technical support options and LabVIEW resources.

National Instruments Technical Support Options

Visit the following sections of the award-winning National Instruments Web site at ni.com for technical support and professional services:

- **Support**—Technical support at ni.com/support includes the following resources:
 - **Self-Help Technical Resources**—For answers and solutions, visit ni.com/support for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on. Registered users also receive access to the NI Discussion Forums at ni.com/forums. NI Applications Engineers make sure every question submitted online receives an answer.
 - **Standard Service Program Membership**—This program entitles members to direct access to NI Applications Engineers via phone and email for one-to-one technical support as well as exclusive access to on demand training modules via the Services Resource Center. NI offers complementary membership for a full year after purchase, after which you may renew to continue your benefits.
For information about other technical support options in your area, visit ni.com/services or contact your local office at ni.com/contact.
 - **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. The NI Alliance Partners joins system integrators, consultants, and hardware vendors to provide comprehensive service and expertise to customers. The program ensures qualified, specialized assistance for application and system development. To learn more, call your local NI office or visit ni.com/alliance.

If you searched ni.com and could not find the answers you need, contact your local office or NI corporate headquarters. Phone numbers for our worldwide offices are listed at the front of this manual. You also can visit the

Worldwide Offices section of ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

Other National Instruments Training Courses

National Instruments offers several training courses for LabVIEW users. These courses continue the training you received here and expand it to other areas. Visit ni.com/training to purchase course materials or sign up for instructor-led, hands-on courses at locations around the world.

National Instruments Certification

Earning an NI certification acknowledges your expertise in working with NI products and technologies. The measurement and automation industry, your employer, clients, and peers recognize your NI certification credential as a symbol of the skills and knowledge you have gained through experience. Visit ni.com/training for more information about the NI certification program.

LabVIEW Resources

The following publications offer more information about LabVIEW.

LabVIEW Publications

Many books have been written about LabVIEW programming and applications. The National Instruments Web site contains a list of all the LabVIEW books and links to places to purchase these books. Publisher information is also included so you can directly contact the publisher for more information on the contents and ordering information for LabVIEW and related computer-based measurement and automation books.

Web Sites

ni.com/products

ni.com/labview

ni.com/dataacquisition

ni.com/sensors

ni.com/signalconditioning

Course Evaluation

Course _____

Location _____

Instructor _____ Date _____

Student Information (optional)

Name _____

Company _____ Phone _____

Instructor

Please evaluate the instructor by checking the appropriate circle. Unsatisfactory Poor Satisfactory Good Excellent

Instructor's ability to communicate course concepts

<input type="radio"/>				
-----------------------	-----------------------	-----------------------	-----------------------	-----------------------

Instructor's knowledge of the subject matter

<input type="radio"/>				
-----------------------	-----------------------	-----------------------	-----------------------	-----------------------

Instructor's presentation skills

<input type="radio"/>				
-----------------------	-----------------------	-----------------------	-----------------------	-----------------------

Instructor's sensitivity to class needs

<input type="radio"/>				
-----------------------	-----------------------	-----------------------	-----------------------	-----------------------

Instructor's preparation for the class

<input type="radio"/>				
-----------------------	-----------------------	-----------------------	-----------------------	-----------------------

Course

Training facility quality

<input type="radio"/>				
-----------------------	-----------------------	-----------------------	-----------------------	-----------------------

Training equipment quality

<input type="radio"/>				
-----------------------	-----------------------	-----------------------	-----------------------	-----------------------

Was the hardware set up correctly? Yes No

The course length was Too long Just right Too short

The detail of topics covered in the course was Too much Just right Not enough

The course material was clear and easy to follow. Yes No Sometimes

Did the course cover material as advertised? Yes No

I had the skills or knowledge I needed to attend this course. Yes No If no, how could you have been better prepared for the course? _____

What were the strong points of the course? _____

What topics would you add to the course? _____

What part(s) of the course need to be condensed or removed? _____

What needs to be added to the course to make it better? _____

How did you benefit from taking this course? _____

Are there others at your company who have training needs? Please list. _____

Do you have other training needs that we could assist you with? _____

How did you hear about this course? NI Web site NI Sales Representative Mailing Co-worker

Other _____

National Instruments
Not for Distribution