

NI-DAQ[™]mx Help

January 2008, 370466L-01

This help file contains information about using NI-DAQmx to program your National Instruments device. NI-DAQmx is the software you use to communicate with and control your NI data acquisition (DAQ) device. Refer to *Support in NI-DAQ 8.7* in the *NI-DAQ 8.7 Readme* for a list of devices supported in NI-DAQmx.

This document describes only NI-DAQmx. For information on Traditional NI-DAQ (Legacy), refer to the *Traditional NI-DAQ (Legacy) User Manual*.

For more information about this help file, refer to the following topics:

Using Help

Related Documentation

Important Information

Technical Support and Professional Services

To comment on National Instruments documentation, refer to the <u>National</u> <u>Instruments Web site</u>.

© 2003–2008 National Instruments Corporation. All rights reserved.

Related Documentation

Many manuals also are available as PDFs. You must have Adobe Acrobat Reader with Search and Accessibility 5.0.5 or later installed to view the PDFs. Refer to the <u>Adobe Systems Incorporated Web site</u> to download Acrobat Reader. Refer to the <u>National Instruments Product</u> <u>Manuals Library</u> for updated documentation resources.

The following documents contain information that you may find helpful as you use this help file. For additional details on these documents, along with their default installation locations, refer to ni.com/kb.

- DAQ Assistant Help
- DAQ Getting Started Guide
- Getting Started with LabVIEW
- Getting Started with LabVIEW SignalExpress
- LabVIEW Help
- LabVIEW Real-Time User Manual
- LabVIEW SignalExpress Help
- LabWindows™/CVI™ Help
- Measurement & Automation Explorer Help for NI-DAQmx
- NI Measurement Studio Help
- NI-DAQmx C Reference Help
- NI-DAQmx Data Acquisition VIs
- SCXI Quick Start Guide
- PID Control Toolset Manual
- Taking an NI-DAQmx Measurement in LabVIEW
- Taking an NI-DAQmx Measurement in LabVIEW SignalExpress
- Taking an NI-DAQmx Measurement in LabWindows/CVI
- Using NI-DAQmx with LabVIEW Project
- Device Documentation

For a descriptions of NI-DAQmx documents along with default installation locations, refer to <u>NI-DAQmx for Windows Documentation</u> on ni.com.

Using Help

<u>Conventions</u> <u>Navigating Help</u> <u>Searching Help</u> <u>Printing Help File Topics</u>

Conventions

This help file uses the following formatting and typographical conventions:

<>	Angle brackets that contain numbers separated by an ellipsis represent a range of values associated with a bit or signal name—for example, AO <03>.
»	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.
P	This icon denotes a tip, which alerts you to advisory information.
	This icon denotes a note, which alerts you to important information.
attribute/property	This term is used to represent properties for LabVIEW, Visual C++, Visual Basic .NET, and Visual C#; and Get and Set Attribute functions for ANSI C and LabWindows™/CVI™.
bold	Bold text denotes items that you must select or click on in the software, such as menu items and dialog box options. Bold text also denotes parameter names, emphasis, or an introduction to a key concept.
green	Underlined text in this color denotes a link to a help topic, help file, or Web address.
italic	Italic text denotes variables, emphasis, cross references, 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.
function/VI	This term is used to generically represent functions, VIs, and methods, depending on the programming language you use. A function/VI might not exactly match the term used in your programming language. Consult the appropriate reference documentation,

such as the *NI-DAQmx C Reference Help*, for the specific terms.

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

Navigating Help (Windows Only)

To navigate this help file, use the **Contents**, **Index**, and **Search** tabs to the left of this window or use the following toolbar buttons located above the tabs:

- Hide—Hides the navigation pane from view.
- Locate—Locates the currently displayed topic in the Contents tab, allowing you to view related topics.
- **Back**—Displays the previously viewed topic.
- **Forward**—Displays the topic you viewed before clicking the **Back** button.
- **Options**—Displays a list of commands and viewing options for the help file.

Searching Help (Windows Only)

Use the **Search** tab to the left of this window to locate content in this help file. If you want to search for words in a certain order, such as "related documentation," add quotation marks around the search words as shown in the example. Searching for terms on the **Search** tab allows you to quickly locate specific information and information in topics that are not included on the **Contents** tab.

Wildcards

You also can search using asterisk (*) or question mark (?) wildcards. Use the asterisk wildcard to return topics that contain a certain string. For example, a search for "prog*" lists topics that contain the words "program," "programmatically," "progress," and so on.

Use the question mark wildcard as a substitute for a single character in a search term. For example, "?ext" lists topics that contain the words "next," "text," and so on.



Note Wildcards do not work when searching with Simplified Chinese, Traditional Chinese, Japanese, and Korean characters.

Nested Expressions

Use nested expressions to combine searches to further refine a search. You can use Boolean expressions and wildcards in a nested expression. For example, "example AND (program OR VI)" lists topics that contain "example program" or "example VI." You cannot nest expressions more than five levels.

Boolean Expressions

Click the **•** button to add Boolean expressions to a search. The following Boolean operators are available:

- **AND** (default)—Returns topics that contain both search terms. You do not need to specify this operator unless you are using nested expressions.
- **OR**—Returns topics that contain either the first or second term.
- **NOT**—Returns topics that contain the first term without the second term.
- **NEAR**—Returns topics that contain both terms within eight words of each other.

Search Options

Use the following checkboxes on the **Search** tab to customize a search:

- Search previous results—Narrows the results from a search that returned too many topics. You must remove the checkmark from this checkbox to search all topics.
- Match similar words—Broadens a search to return topics that contain words similar to the search terms. For example, a search for "program" lists topics that include the words "programs," "programming," and so on.
- Search titles only—Searches only in the titles of topics.

Printing Help File Topics (Windows Only)

Complete the following steps to print an entire book from the **Contents** tab:

- 1. Right-click the book.
- 2. Select **Print** from the shortcut menu to display the **Print Topics** dialog box.
- 3. Select the **Print the selected heading and all subtopics** option.
 - Note Select Print the selected topic if you want to print the single topic you have selected in the **Contents** tab.
- 4. Click the **OK** button.

Printing PDF Documents

This help file may contain links to PDF documents. To print PDF documents, click the print button located on the Adobe Acrobat Viewer toolbar.

Getting Started with NI-DAQmx

This section provides an <u>overview of NI-DAQ</u>, including a discussion of the differences between NI-DAQmx and Traditional NI-DAQ (Legacy). It also includes information on <u>configuring tasks</u> and pointers on how to create NI-DAQmx applications in different <u>ADEs</u>, such as LabWindows/CVI and LabVIEW.

NI-DAQ 8.x Overview

National Instruments measurement devices are packaged with NI-DAQ driver software, an extensive library of functions and VIs you can call from your application software, such as LabVIEW or LabWindows/CVI, to program your NI measurement devices. Measurement devices include DAQ devices such as the M Series multifunction I/O (MIO) devices, signal conditioning modules, and switch modules. Driver software has an application programming interface (API), which is a library of VIs, functions, classes, attributes, and properties for creating applications for your device.

NI-DAQ 8.*x* comes with the latest version of the software driver, which is called NI-DAQmx. NI-DAQmx replaces Traditional NI-DAQ (Legacy). NI-DAQmx and Traditional NI-DAQ (Legacy) have their own APIs, hardware configurations, and software configurations. NI-DAQmx CDs also include LabVIEW SignalExpress LE, an easy-to-use configuration-based tool specifically designed for data logging applications.

NI-DAQmx

NI-DAQmx has the following advantages over Traditional NI-DAQ (Legacy):

- <u>DAQ Assistant</u>—a graphical way to configure virtual channels and measurement tasks for your device, and to generate NI-DAQmx code based on your virtual channels and tasks, for use in LabVIEW, LabVIEW SignalExpress, LabWindows/CVI, and Measurement Studio.
- Increased performance, including faster single-point analog I/O and multithreading.
- NI-DAQmx simulated devices for testing and modifying applications without plugging in hardware.
- Simpler, more intuitive APIs for creating DAQ applications using fewer functions and VIs than earlier versions of NI-DAQ.
- Expanded functionality for LabVIEW, including property nodes and waveform data type support.
- Similar APIs and functionality for ANSI C, LabWindows/CVI, and Measurement Studio, including native .NET and C++ interfaces.
- Improved support and performance for the LabVIEW Real-Time Module.

Traditional NI-DAQ (Legacy)

Traditional NI-DAQ (Legacy) is an upgrade of the earlier version of NI-DAQ. Traditional NI-DAQ (Legacy) has the same VIs and functions and works the same way as NI-DAQ 6.9.3, except you can use Traditional NI-DAQ (Legacy) and NI-DAQmx on the same computer, and some hardware is no longer supported.

Who Can Use NI-DAQmx

You should install and use NI-DAQmx if the following situations apply:

- You are a new NI-DAQ user.
- You are using devices supported by NI-DAQmx; refer to the *NI-DAQ Readme* for a list of supported devices.
- You are using Windows Vista/2000/NT/XP.

If you are using NI application software with NI-DAQmx, you must use LabVIEW, LabWindows/CVI, or Measurement Studio version 7.*x* or later, LabVIEW SignalExpress 2.*x* or later (including SignalExpress LE), or the LabVIEW Real-Time Module 7.1 or later.

If you use one of the Microsoft .NET languages, Visual C# and/or Visual Basic .NET, or a device supported only by NI-DAQmx, such as an M Series device, you must use NI-DAQmx.

You also can use NI-DAQmx with a supported compiler, such as an ANSI C compiler.

Who Must Use Traditional NI-DAQ (Legacy)

Install and use Traditional NI-DAQ (Legacy) if one of the following situations apply:

- You have a device that is not supported by NI-DAQmx, such as the AT E Series multifunction DAQ devices.
- You are using a version of LabVIEW, LabWindows/CVI, or Measurement Studio earlier than version 7.0.
- You are upgrading from NI-DAQ 6.9.*x* and have existing applications that you do not want to port to NI-DAQmx now.
- Note The earliest version of NI application software supported by Traditional NI-DAQ (Legacy) is version 6.0. LabVIEW, LabWindows/CVI, or Measurement Studio versions 6.*x* can use Traditional NI-DAQ (Legacy) from the NI-DAQ 8.*x* distribution.

Configuring a Task

DAQ Assistant Introduction to MAX Capabilities of MAX

DAQ Assistant

The DAQ Assistant is a graphical interface for configuring measurement tasks, channels, and scales. Using the DAQ Assistant, you can interactively build a measurement channel or task for use in LabVIEW 7.*x* or later, LabVIEW SignalExpress 2.*x* or later, LabWindows/CVI 7.*x* or later, and Measurement Studio 7.*x* or later. With these NI application software packages, you also can use the DAQ Assistant to generate code for use in your applications. Refer to the DAQ Assistant Help for additional information.

See Also

Creating Channels and Tasks with the DAQ Assistant

Introduction to Measurement & Automation Explorer (MAX)

You configure your NI measurement and signal conditioning devices with MAX. MAX informs other programs which devices you have in your system and how they are configured. Use MAX to add, configure, test, and remove a measurement device or signal conditioning device.

To check the system resources used by a DAQ device and to select attached accessories, expand **Devices and Interfaces** in the configuration tree, and right-click the device for options. For more information, refer to *Measurement & Automation Explorer Help for NI-DAQmx*.

Capabilities of MAX for NI-DAQmx

You can use MAX for the following measurement configuration actions:

- Configuring resources and other device-specific settings for DAQ devices in your system
- Testing the resources and the functionality of DAQ devices in your system
- Configuring channels, scales, and tasks using the DAQ Assistant
- Creating and configuring NI-DAQmx simulated devices

When you run an application using NI-DAQ, the software reads the configuration to determine the devices you configured. Therefore, you must configure DAQ devices first with MAX. Refer to *Measurement & Automation Explorer Help for NI-DAQmx* for more information about configuring and testing DAQ devices.

Exporting and Importing a Configuration in MAX

You can save virtual channels, tasks, devices, and their relationships for reuse in other systems also running MAX. To reuse a configuration, you must first export a channel, task, or device configuration. Exporting the configuration creates an .nce configuration file that you can then import into another system with MAX. Using the Import and Export features in MAX, you can create an NI-DAQmx simulated version of a physical device or import an NI-DAQmx simulated device configuration onto a physical device. For detailed instructions on how to export and import configurations for deployment, refer to *Measurement & Automation Explorer Help for NI-DAQmx*.



Note When you use LabWindows/CVI to create a distribution, you can specify to invoke the MAX Configuration Export Wizard to include the hardware configurations in your deployed application.

You can also programmatically import and export configuration files. For more information, refer to the MAX Configuration VI Reference for LabVIEW or the MAX Configuration Function Reference for LabWindows/CVI.

Distributed Applications

You can use the DAQmx I/O Server to bind to a global virtual channel created in MAX or to a DAQ channel created in LabVIEW Project. You can use the network variable in LabVIEW Project to bind to the channel, or you can use a third-party OPC. Refer to the NI-DAQmx topics in the LabVIEW documentation for instructions on binding to a DAQ channel using LabVIEW Project.

Getting Started in your ADE

LabVIEW LabVIEW SignalExpress LabWindows/CVI Measurement Studio .NET without Measurement Studio ANSI C without LabWindows/CVI

Creating an Application with LabVIEW

If you program your NI-DAQmx-supported device in LabVIEW, you can interactively create virtual channels—both global and local—and tasks by launching the DAQ Assistant from MAX or from within LabVIEW. Refer to the DAQ Assistant Help for additional information. You also can create local virtual channels and tasks, and write your own applications using the NI-DAQmx API. To get started in LabVIEW, follow these general steps:

- 1. Open an existing or new LabVIEW VI.
- 2. Build your VI, using the NI-DAQmx VIs and properties.

For help with NI-DAQmx VIs, refer to *LabVIEW NI-DAQmx VI Reference Help*. For general help with programming in LabVIEW, refer to *LabVIEW Help*.

Creating an Application in LabVIEW SignalExpress

If you use your NI-DAQmx-supported device in LabVIEW SignalExpress, you can create a project that includes NI-DAQmx steps. With LabVIEW SignalExpress, you can log and analyze data. You can also add global virtual channels that you created in MAX to your NI-DAQmx steps in LabVIEW SignalExpress. Refer to the DAQ Assistant Help for additional information. To get started in LabVIEW SignalExpress, follow these general steps:

- 1. Click Add Step and select Acquire Signals»Acquire DAQmx»Analog Input»Voltage to drop the DAQmx Acquire Step.
- 2. Click the + button to add a channel to the NI-DAQmx step.

For help with using the DAQ Assistant with LabVIEW SignalExpress, refer to *Taking an NI-DAQmx Measurement in LabVIEW SignalExpress*. For general help with programming in LabVIEW SignalExpress, refer to *LabVIEW SignalExpress Help*.

Creating an Application with LabWindows/CVI

If you program your NI-DAQmx-supported device in LabWindows/CVI, you can interactively create global or local virtual channels and tasks by launching the DAQ Assistant from MAX or from within LabWindows/CVI. You can generate the configuration code based on your task or channel in LabWindows/CVI. Refer to the DAQ Assistant Help for additional information about generating code. You also can create local virtual channels and tasks, and write your own applications using the NI-DAQmx API. To create an application, follow these general steps:

- 1. Create a new project file (.prj).
- 2. Open an existing or new source file (.c).
- 3. Add your source file to the project.
- 4. Select **NI-DAQmx** from the Library Tree, and choose the function panel you want to use.
- 5. To view examples of NI-DAQmx applications in LabWindows/CVI, launch the NI Example Finder.
- 6. Build your application.

For help with NI-DAQmx functions, refer to *NI-DAQmx C Function Reference Help*. For general help with programming in LabWindows/CVI, refer to *LabWindows/CVI Help*, accessible through **Start**»All **Programs**»National Instruments»LabWindows CVI»LabWindows CVI Help.

Creating an Application in Measurement Studio with Visual C++, Visual C#, or Visual Basic .NET

If you program your NI-DAQmx-supported device in Measurement Studio using Visual C++, Visual C#, or Visual Basic .NET, you can interactively create channels and tasks by launching the DAQ Assistant from MAX or from within Visual Studio .NET. You can generate the configuration code based on your task or channel in Measurement Studio. Refer to the DAQ Assistant Help for additional information about generating code. You also can create channels and tasks, and write your own applications in your ADE using the NI-DAQmx API.

For help with NI-DAQmx methods and properties, refer to the NI-DAQmx .NET Class Library or the NI-DAQmx Visual C++ Class Library included in the *NI Measurement Studio Help*. For general help with programming in Measurement Studio, refer to the *NI Measurement Studio Help*, which is fully integrated with the Microsoft Visual Studio .NET help. To view this help file in Visual Studio. NET, select **Measurement Studio**»NI **Measurement Studio Help**.

To create an application in Visual C++, Visual C#, or Visual Basic .NET, follow these general steps:

- 1. In Visual Studio .NET, select **File»New»Project** to launch the New Project dialog box.
- 2. Find the Measurement Studio folder for the language you want to create a program in.
- 3. Choose a project type. You add DAQ tasks as a part of this step.

Creating a .NET Application without Measurement Studio

With the Microsoft .NET Framework version 1.1 or later, you can use NI-DAQmx to create applications using Visual C# and Visual Basic .NET without Measurement Studio. You need Microsoft Visual Studio .NET 2003 or Microsoft Visual Studio 2005 for the API documentation to be installed.

The installed documentation contains the NI-DAQmx API overview, measurement tasks and concepts, and function reference. This help is fully integrated into the Visual Studio .NET documentation. To view the NI-DAQmx .NET documentation, go to **Start»All Programs»National Instruments»NI-DAQ»NI-DAQmx .NET Reference Help**. Expand **NI Measurement Studio Help»NI Measurement Studio .NET Class Library»Reference** to view the function reference. Expand **NI Measurement Studio Help»NI Measurement Studio .NET Class Library»Using the Measurement Studio .NET Class Library»Using the Measurement Studio .NET Class** to view conceptual topics for using NI-DAQmx with Visual C# and Visual Basic .NET.

To get to the same help topics from within Visual Studio, go to **Help»Contents**. Select **Measurement Studio** from the **Filtered By** dropdown list and follow the previous instructions.

Creating an ANSI C Application without LabWindows/CVI

NI-DAQmx has a C API that you can use to create applications. To create an application, follow these general steps:

- 1. Create a new project.
- 2. Open existing or new source files (.c), and add them to the project. Make sure you include the NI-DAQmx header file, nidaqmx.h, in your source code files. You can find this header file at NI-DAQ\DAQmx ANSI C Dev\include.
- 3. Add the NI-DAQmx import library, nidaqmx.lib, to the project. The import library files are located under NI-DAQ\DAQmx ANSI C Dev\lib\.
- 4. To view examples of NI-DAQmx applications, go to the NI-DAQ\Examples\DAQmx ANSI C directory.
- 5. Build your application.

For help with NI-DAQmx functions, refer to the *NI-DAQmx C Reference Help*, which is installed by default at **Start»All Programs»National Instruments»NI-DAQ»NI-DAQmx C Reference Help**.

Examples

Each API includes a collection of programming examples to help you get started developing an application. You can modify example code and save it in an application. You can use examples to develop a new application or add example code to an existing application.

To run examples without hardware installed, you can use an NI-DAQmx simulated device. In MAX, refer to the *Measurement & Automation Explorer Help for NI-DAQmx* by selecting **Help»Help Topics» NI-DAQmx** for information on NI-DAQmx simulated devices.

To find the locations of examples for your software application, refer to the following table.

Software Application	Example Location	
LabVIEW or LabWindows/CVI	Help»Find Examples	
LabVIEW SignalExpress	Program Files\National Instruments\SignalExpress\Examples	
ANSI C	*NI-DAQ\Examples\DAQmx ANSI C	
MFC 7.0 C++	*NI-DAQ\Examples\MStudioVC2003	
Visual Basic .NET and C# for Visual Studio 2003	*NI-DAQ\Examples\DotNET1.1	
MFC 8.0 C++	*NI-DAQ\Examples\MStudioVC2005	
Visual Basic .NET and C# for Visual Studio 2005	*NI-DAQ\Examples\DotNET2.0	
* For Windows XP, the default path is <drive>:\Documents and Settings\All</drive>		

Users\Documents\National Instruments\NI-DAQ\Examples\..... For Windows Vista, the default path is <drive>:\Users\Public\Documents\ National Instruments\NI-DAQ\Examples\.....

Note Visual Studio 2003 and Visual Studio 2005 do not require Measurement Studio.

Troubleshooting

Installation and Configuration

Refer to the *DAQ Getting Started Guide* and the *SCXI Quick Start Guide* for general installation and configuration instructions.

Use the following resources if you have problems installing your DAQ hardware and/or software:

- For troubleshooting instructions, refer to the Hardware Installation/ Configuration Troubleshooter at ni.com/support/install.
- Refer to ni.com/kb for documents on troubleshooting common installation and programming problems and for answering frequently asked questions about NI products.
- If you think you have damaged your device and need to return your National Instruments hardware for repair or calibration, refer to ni.com/support and search on Sending a Board for Repair or Calibration to learn how to begin the Return Merchandise Authorization (RMA) process.

For LabWindows/CVI users, if the Data Acquisition function panel is disabled, you may need to uninstall NI-DAQ and reinstall it, making sure that you add support for LabWindows/CVI. If you have installed LabWindows/CVI support and Data Acquisition is still dimmed, select **Library**»Customize. In the Customize Library Menu dialog box, check Data Acquisition, and restart LabWindows/CVI. You might also need to verify that the dataacq.lib is in the bin directory.

Programming

To help you get started programming, you can use the <u>shipping examples</u> for your ADE.

You can also visit NI's extensive library of <u>technical support resources</u> at ni.com/support.

You can interactively configure global virtual channels and tasks with the <u>DAQ Assistant</u>. For NI application software such as LabVIEW, you can use the DAQ Assistant to generate code.

Finally, the *NI-DAQmx Help* contains programming flowcharts for common applications such as measuring temperature, current, strain, position, and acceleration.

External Connections

In addition to the information on making signal connections in this help file, the Connection Diagram tab in the DAQ Assistant within MAX shows you how to connect signals.

Calibration

- For information on externally calibrating your device, including step-by-step calibration procedures, refer to <u>ni.com/calibration</u>.
- For an overview of calibration, including the difference between self-calibration and external calibration, refer to <u>Calibration</u>.
- For device-specific information required for calibration with NI-DAQmx, refer to <u>Device-Specific Calibration</u>.
- For information on channel calibration, refer to <u>What Is Channel</u> <u>Calibration?</u>

CPU Usage

NI-DAQmx tasks use 100% of the CPU if no other processes are running. However, as soon as another process requires the CPU, the NI-DAQmx task yields to that process.

Troubleshooting an SCXI System

The following are some tips to help you troubleshoot problems with an SCXI system:

- Can MAX establish communication with the chassis? If not, try one or all of the following: Connect the DAQ device to a different module in the chassis. Try a different cable assembly. Try a different chassis. Try a different DAQ device. If you have multiple chassis, disconnect them and reconnect them one at a time to isolate the problem.
- Make sure that each SCXI chassis connected to a single DAQ device has a unique address.
- If you have multiple SCXI modules, remove all the modules and test each module individually.
- If a particular chassis does not work, try another one.
- If you are getting erroneous readings from your signal source, disconnect the signal source and short the input channel to ground. You should get a 0 V reading.
- Alternately, connect a battery or other known signal source to the input channel.
- Run an example program to see if you still get erroneous results.

Frequently Asked Questions (FAQ)

For answers to frequently asked questions, visit the <u>NI-DAQmx FAQ</u>.

Generic Programming Flowcharts

This section contains general programming flowcharts that you can use when creating an application. You also can find programming flowcharts for typical applications—such as measuring temperature, measuring current, and measuring strain—in the *Common Applications* section of this help file.

In the programming flowcharts, many applications also include explicit control functions to start, stop, and clear the task. For instance, for applications that use your counter/timer, such as finite counter input, you need to call the Start function/VI to arm the counter. In LabVIEW, clearing occurs automatically. For other ADEs, you must include these functions in your application.

Functions and VIs produce the core functionality of the NI-DAQmx API. For instance, NI-DAQmx includes functions for timing, triggering, reading, and writing samples. However, for advanced functionality, Visual C++, Visual C#, Visual Basic .NET, and LabVIEW require properties. ANSI C and LabWindows/CVI employ the Get and Set Attribute functions. For more information, refer to the programming reference help for your ADE.

Analog Input Programming Flowcharts

Single Sample Analog Input Finite Analog Input Continuous Analog Input

Analog Output Programming Flowcharts

Single Sample Analog Output Finite Analog Output Continuous Analog Output

Digital Input Programming Flowcharts

Single Sample Digital Input Finite Digital Input Continuous Digital Input

Digital Output Programming Flowcharts

Single Sample Digital Output Finite Digital Output Continuous Digital Output

Measuring Counter Values (Counter Input) Programming Flowcharts

Single Point Counter Input

Finite Counter Input

Continuous Counter Input

Analog Input Programming Flowcharts

This section contains general programming flowcharts that you can use when creating an application. You also can find programming flowcharts for typical applications—such as measuring temperature, measuring current, and measuring strain—in the *Common Applications* section of this help file.

Functions and VIs provide the core functionality of the NI-DAQmx API. For instance, NI-DAQmx includes functions for timing, triggering, reading, and writing samples. However, for advanced functionality, Visual C++, Visual C#, Visual Basic .NET, and LabVIEW require properties. ANSI C and LabWindows/CVI employ the Get and Set Attribute functions. For more information, refer to the programming reference help for your ADE.

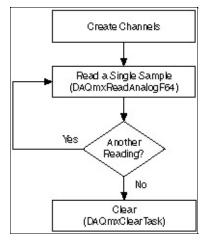
Analog Input Programming Flowcharts

Single Sample Analog Input Finite Analog Input Continuous Analog Input Triggered Acquisition

Single Sample Analog Input Programming Flowchart

Acquiring a single sample is an on-demand operation. In other words, NI-DAQmx acquires one value from an input channel and immediately returns the value. This operation does not require any buffering or hardware timing. For example, if you periodically needed to monitor the fluid level in a tank, you acquire single data points. You can connect the transducer that produces a voltage representing the fluid level to a single channel on your measurement device and initiate a single-channel, single-point acquisition when you want to know the fluid level.

With NI-DAQmx, you also can gather data from multiple channels. For instance, you might want to monitor the fluid level in the tank as well as the temperature. In this case, you need two transducers connected to two channels on your device. The following flowchart depicts the steps to programmatically create a single sample analog input application. If you prefer, you can configure a task for acquiring a single sample using the DAQ Assistant.

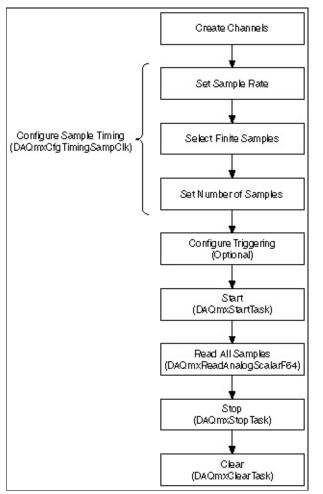


P

Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the previous flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

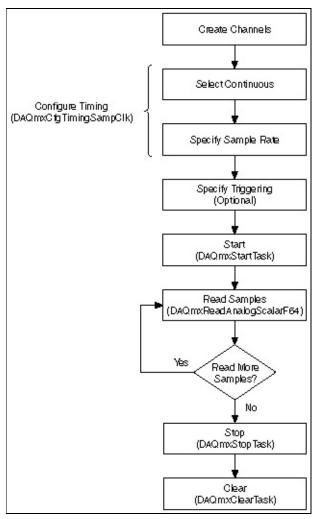
Finite Analog Input Programming Flowchart

One way to acquire multiple samples for one or more channels is to acquire single samples in a repetitive manner. However, acquiring a single sample on one or more channels over and over is inefficient and time consuming. Moreover, you do not have accurate control over the time between each sample or channel. Instead, you can use hardware timing, which uses a buffer in computer memory to acquire data more efficiently. Programmatically, you need to include the timing function, specifying the **sample rate** and the **sample mode** (finite). As with other functions, you can acquire multiple samples for a single channel or multiple channels. You can configure a task for finite analog input using the DAQ Assistant.



Continuous Analog Input Programming Flowchart

If you want to view, process, or log a subset of the samples as they are being acquired, you need to continually acquire samples. For these types of applications, set the **sample mode** to continuous. The following flowchart depicts the main steps required in an NI-DAQmx application for measuring voltage. Instead, you can configure a task for continuous analog input using the <u>DAQ Assistant</u>.



Analog Output Programming Flowcharts

This section contains general programming flowcharts that you can use when creating an application. You also can find programming flowcharts for typical applications—such as <u>generating voltage</u> and <u>generating</u> <u>current</u>—in the *Common Applications* section of this help file.

In the programming flowcharts, many applications also include explicit control functions to start, stop, and clear the task. In LabVIEW, clearing occurs automatically. For other ADEs, you must include these functions in your application.

Functions and VIs provide the core functionality of the NI-DAQmx API. For instance, NI-DAQmx includes functions for timing, triggering, reading, and writing samples. However, for advanced functionality, Visual C++, Visual C#, Visual Basic .NET, and LabVIEW require properties. ANSI C and LabWindows/CVI employ the Get and Set Attribute functions. For more information, refer to the programming reference help for your ADE.

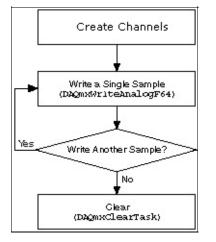
Analog Output Programming Flowcharts

Single Sample Analog Output Finite Analog Output Continuous Analog Output

Single Sample Analog Output Programming Flowchart

Generating a single sample is an on-demand operation. In other words, NI-DAQmx generates one value from an input channel and immediately returns the value. This operation does not require any buffering or hardware timing.

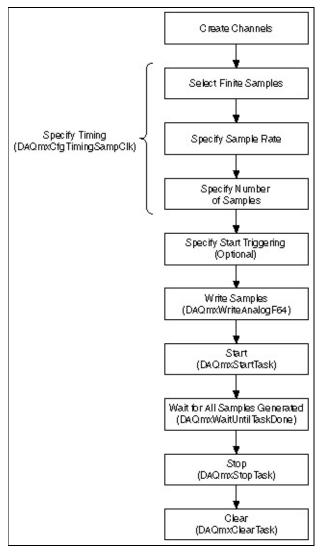
With NI-DAQmx, you also can generate samples from multiple channels. If you prefer, you can configure a task for generating a single sample using the <u>DAQ Assistant</u>.



Tip To increase performance, especially when multiple samples are written, include the Start function/VI and Stop function/VI in your application. In the preceding flowchart, the Start function/VI would come just before you write samples, and Stop would come just before you clear the task.

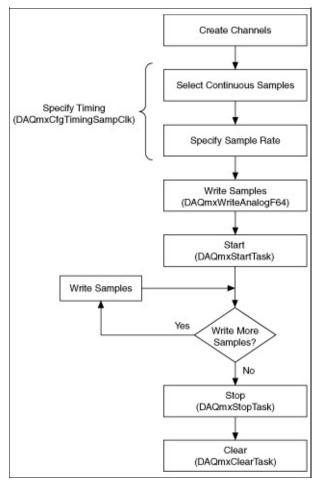
Finite Analog Output Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to generate a finite number of voltage samples in a buffered generation. If you prefer, you can configure this task using the <u>DAQ</u> <u>Assistant</u>.



Continuous Analog Output Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to continuously generate voltage samples. If you prefer, you can configure this task using the <u>DAQ Assistant</u>.



Counter Programming Flowcharts

This section contains general programming flowcharts that you can use when creating an application. You also can find programming flowcharts for typical applications—such as <u>counting edges</u> and <u>generating pulses</u> in the *Common Applications* section of this help file.

In the programming flowcharts, many applications also include explicit control functions to start, stop, and clear the task. For instance, for applications that use your counter, such as counting edges or measuring period, you need to call the Start function/VI to arm the counter. In LabVIEW, clearing occurs automatically. For other ADEs, you must include these functions in your application.

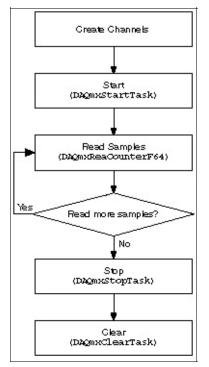
Functions and VIs provide the core functionality of the NI-DAQmx API. For instance, NI-DAQmx includes functions for timing, triggering, reading, and writing samples. However, for advanced functionality, Visual C++, Visual C#, Visual Basic .NET, and LabVIEW require properties. ANSI C and LabWindows/CVI employ the Get and Set Attribute functions. For more information, refer to the programming reference help for your ADE.

Counter Input Programming Flowcharts

Single Point Counter Input Finite Counter Input Continuous Counter Input

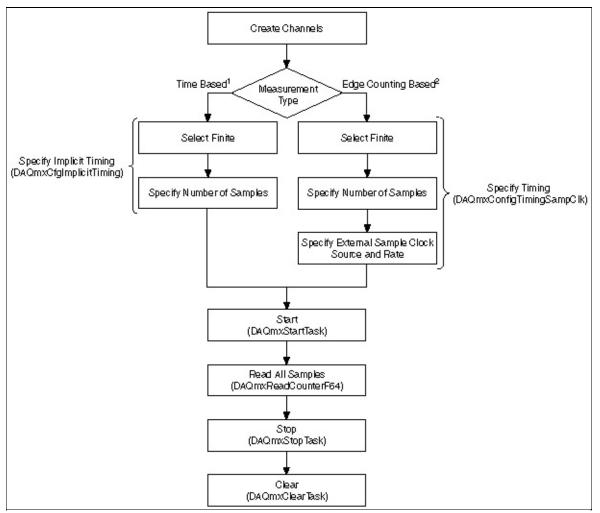
Single Point Counter Input Programming Flowchart

The following flowchart depicts the main steps you must complete for an on-demand counting application. If you prefer, you can configure this task using the <u>DAQ Assistant</u>.



Finite Counter Input Programming Flowchart

The following flowchart depicts the main steps you must complete for finite counter input in an NI-DAQmx application. If you prefer, you can configure this task using the <u>DAQ Assistant</u>.

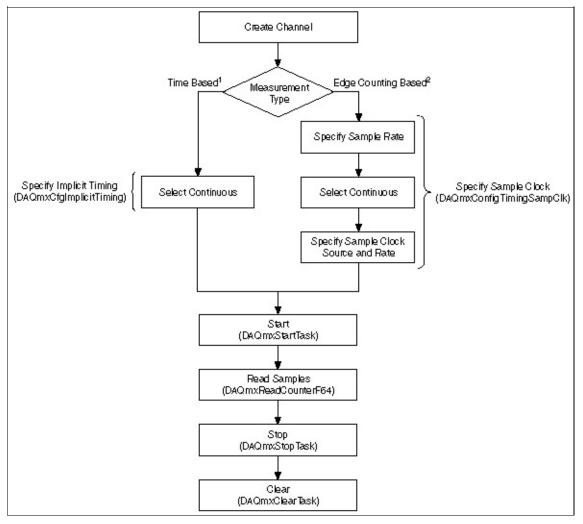


¹Time-based measurements include period, semi-period, pulse width, two-edge separation, and digital frequency.

²Edge counting-based measurements include edge counting, encoderbased position measurements, and GPS timestamp measurements.

Continuous Counter Input Programming Flowchart

The following flowchart depicts the main steps you must complete for continuous counting in an NI-DAQmx application. If you prefer, you can configure this task using the <u>DAQ Assistant</u>.



¹Time-based measurements include period, semi-period, pulse width, two-edge separation, and digital frequency.

²Edge counting-based measurements include edge counting, encoderbased position measurements, and GPS timestamp measurements.

Digital Input Programming Flowcharts

This section contains general programming flowcharts that you can use when creating an application. You also can find programming flowcharts for typical applications—such as <u>measuring a digital value</u>—in the *Common Applications* section of this help file.

In the programming flowcharts, many applications also include explicit control functions to start, stop, and clear the task. In LabVIEW, clearing occurs automatically. For other ADEs, you must include these functions in your application.

Functions and VIs provide the core functionality of the NI-DAQmx API. For instance, NI-DAQmx includes functions for timing, triggering, reading, and writing samples. However, for advanced functionality, Visual C++, Visual C#, Visual Basic .NET, and LabVIEW require properties. ANSI C and LabWindows/CVI employ the Get and Set Attribute functions. For more information, refer to the programming reference help for your ADE.

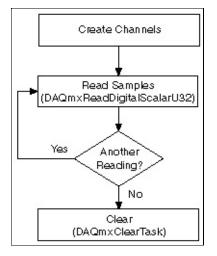
Digital Input Programming Flowcharts

Single Sample Digital Input Finite Digital Input Continuous Digital Input

Single Sample Digital Input Programming Flowchart

Acquiring a single sample is an on-demand operation. In other words, NI-DAQmx acquires one value from an input channel and immediately returns the value. This operation does not require any buffering or hardware timing. For example, if you periodically needed to monitor the fluid level in a tank, you acquire single data points. You can connect the transducer that produces a voltage representing the fluid level to a single channel on your measurement device and initiate a single-channel, single-point acquisition when you want to know the fluid level.

With NI-DAQmx, you also can gather data from multiple channels. For instance, you might want to monitor the fluid level in the tank as well as the temperature. In this case, you need two transducers connected to two channels on your device. The following flowchart depicts the steps to programmatically create an application to measure digital values. If you prefer, you can configure a task for acquiring a single sample using the DAQ Assistant.

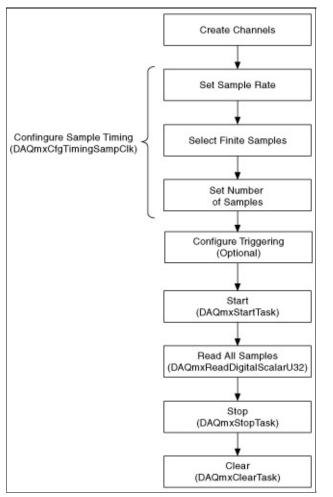


P

Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the previous flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

Finite Digital Input Programming Flowchart

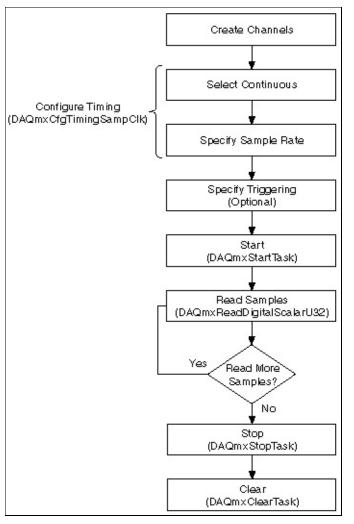
One way to acquire multiple samples for one or more channels is to acquire single samples in a repetitive manner. However, acquiring a single data sample on one or more channels over and over is inefficient and time consuming. Moreover, you do not have accurate control over the time between each sample or channel. Instead, you can use hardware timing, which uses a buffer in computer memory to acquire data more efficiently. Programmatically, you need to include the timing function, specifying the **sample rate** and the **sample mode** (finite). As with other functions, you can acquire multiple samples for a single channel or multiple channels. You can configure a task for measuring digital values using the DAQ Assistant.



Note Triggering and Sample Clock timing for Digital I/O are not supported on all devices.

Continuous Digital Input Programming Flowchart

If you want to view, process, or log a subset of the samples as they are being acquired, you need to continually acquire samples. For these types of applications, set the **sample mode** to continuous. The following flowchart depicts the main steps required in an NI-DAQmx application for acquiring digital signals. You can configure a task for continuously acquiring digital values using the <u>DAQ Assistant</u>.



Note Sample clock timing for Digital I/O is <u>not supported</u> on all devices.

Digital Output Programming Flowcharts

This section contains general programming flowcharts that you can use when creating an application. You also can find programming flowcharts for typical applications in the *Common Applications* section of this help file.

In the programming flowcharts, many applications also include explicit control functions to start, stop, and clear the task. For instance, for applications that use your counter/timer, such as counting edges or measuring period, you need to call the Start function/VI to arm the counter. In LabVIEW, clearing occurs automatically. For other ADEs, you must include these functions in your application.

Functions and VIs provide the core functionality of the NI-DAQmx API. For instance, NI-DAQmx includes functions for timing, triggering, reading, and writing samples. However, for advanced functionality, Visual C++, Visual C#, Visual Basic .NET, and LabVIEW require properties. ANSI C and LabWindows/CVI employ the Get and Set Attribute functions. For more information, refer to the programming reference help for your ADE.

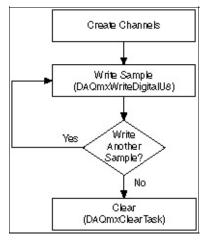
Digital Output Programming Flowcharts

Single Sample Digital Output Finite Digital Output Continuous Digital Output

Single Sample Digital Output Programming Flowchart

Generating a single sample is an on-demand operation. In other words, NI-DAQmx generates one value on an output channel immediately after the Write function/VI is called. This operation does not require any buffering or hardware timing.

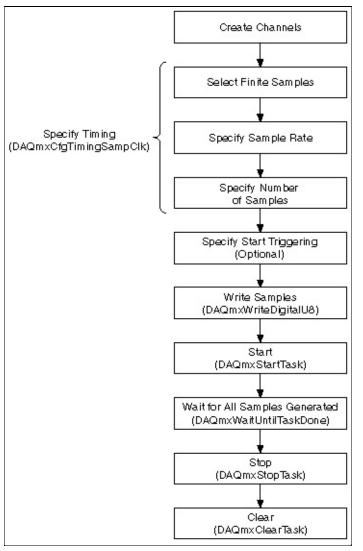
With NI-DAQmx, you also can generate samples from multiple channels. If you prefer, you can configure a task for generating a single sample using the <u>DAQ Assistant</u>.



Tip To increase performance, especially when multiple samples are written, include the Start function/VI and Stop function/VI in your application. In the preceding flowchart, the Start function/VI would come just before you write samples, and Stop would come just before you clear the task.

Finite Digital Output Programming Flowchart

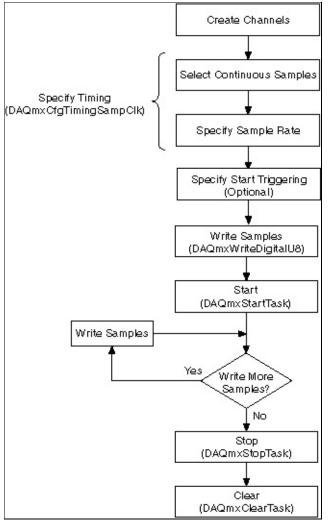
The following flowchart depicts the main steps required in an NI-DAQmx application to generate a finite number of digital values in a buffered generation. If you prefer, you can configure a task for generating digital values using the <u>DAQ Assistant</u>.



Note Sample clock timing for Digital I/O is <u>not supported</u> on all devices.

Continuous Digital Output Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to continuously generate digital values. If you prefer, you can configure a task for generating digital values using the <u>DAQ Assistant</u>.

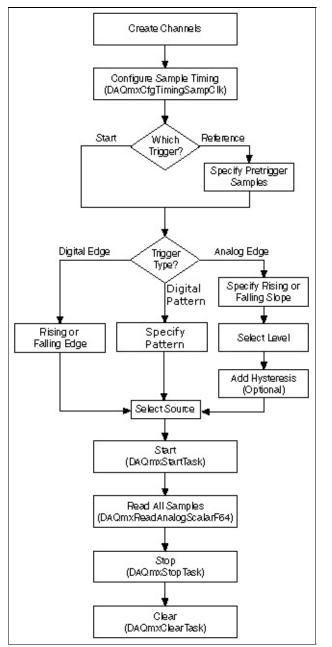


 $\overline{\mathbb{N}}$

Note Sample clock timing for Digital I/O is <u>not supported</u> on all devices.

Triggered Acquisition Programming Flowchart

The following flowchart depicts the main steps you follow for adding triggering to an acquisition. If you prefer, you can configure triggering with the <u>DAQ Assistant</u>.

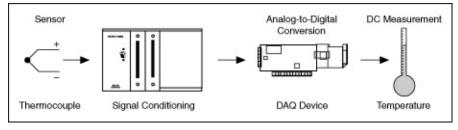


Measuring Temperature

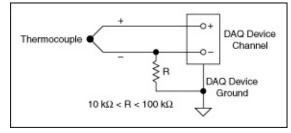
Note Temperature measurements may require you to condition the signal. The conditioning requirements depend on your sensor. Refer to the <u>Overview of Temperature Sensor Types</u> for an explanation of sensor types and conditioning requirements.

Using a Thermocouple to Measure Temperature

A popular way to measure temperature with a DAQ device is to use a <u>thermocouple</u>, as shown in the following figure, because thermocouples are inexpensive, easy to use, and easy to obtain. Thermocouples produce a voltage that varies based on temperature. Using a thermocouple, you can measure a voltage and use a formula to convert the voltage measurement to temperature.



The typical wiring for a thermocouple, as shown in the following figure, uses a resistor, R, only if the thermocouple is not grounded at any other point. If, for example, the thermocouple tip were already grounded, using a resistor would cause a ground loop and result in erroneous readings.



You also can measure temperature using <u>Resistance Temperature</u> <u>Detectors (RTD)</u> and <u>Thermistors</u>.

Making Signal Connections

Creating a Program

<u>RTD</u>

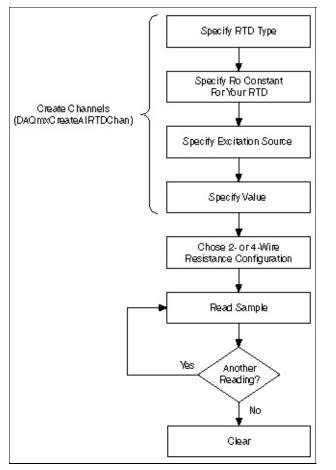
<u>Thermistor</u>

Thermocouple

Examples

Measuring Temperature with an RTD Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to measure temperature with an RTD. Alternatively, you can configure a task for measuring temperature using <u>the DAQ Assistant</u>.

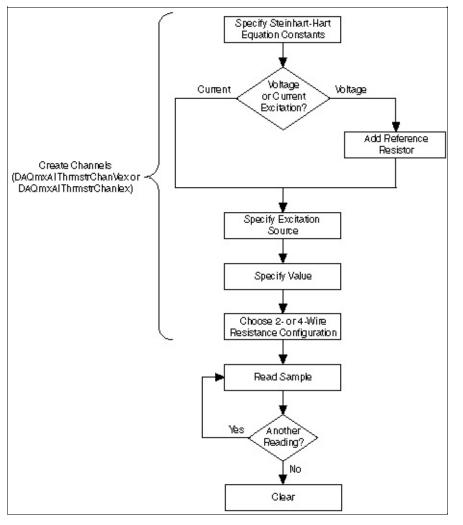


Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the previous flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

Measuring temperature is an example of analog input measurement. Refer to <u>Analog Input Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Measuring Temperature with a Thermistor Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to measure temperature with a thermistor. Alternatively, you can configure a task for measuring temperature using the DAQ Assistant.

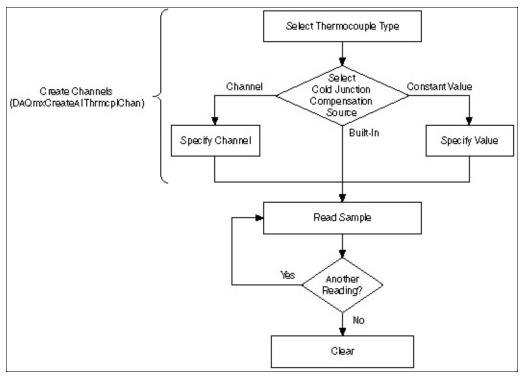


Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the previous flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

Measuring temperature is an example of analog input measurement. Refer to <u>Analog Input Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Measuring Temperature with a Thermocouple Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to measure temperature with a thermocouple. Alternatively, you can configure a task for measuring temperature using the DAQ Assistant.



Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the previous flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

Measuring temperature is an example of analog input measurement. Refer to <u>Analog Input Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Measuring and Generating Current

Many measurement devices can measure and generate current. To measure or generate current with a DAQ device, you need a resistor. Current then can be measured through an analog input connector or generated through an analog output connector. The resistance must be placed in parallel with the connector and the current source. To measure voltage dropped across the resistor and convert it to current, use Ohm's Law.

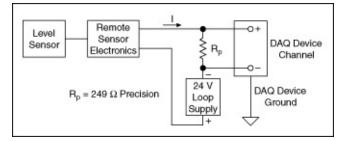
$$I_{(A)} = V_{(V)} / R_{(\Omega)}$$

where I is the current, V is the voltage, and R is the resistance.

4 to 20 mA Loops

4 to 20 milliamp (4-20 mA) loops are commonly used in measurement systems. 4-20 mA loops couple a dynamic range with a live zero of 4 mA for open circuit detection in a system that does not produce sparks. Other advantages include a variety of compatible hardware, a long operating range, and low cost. 4-20 mA loops have a variety of uses, including digital communications, control applications, and reading remote sensors.

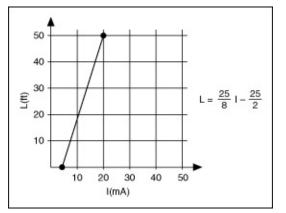
The purpose of the 4-20 mA current loop is for the sensor to transmit a signal in the form of a current. In the following figure, the Level Sensor and Remote Sensor Electronics are typically built into a single unit. An external 24 VDC supply powers the sensor. The sensor regulates the current, which represents the value of what the sensor measures, in this case, the fluid level in a tank.



Current Loop Wiring

The DAQ device reads the voltage drop across the 249 Ω resistor $R_p,$ using Ohm's Law.

Because the current is 4-20 mA and R_p is 249 Ω , *V* ranges from 0.996 V to 4.98 V, which is within the range that DAQ devices can read. Although the equation is useful for calculating the current, the current typically represents a physical quantity you want to measure. In the following figure, the tank level measures 0 to 50 feet. 4 mA represents 0 feet, and 20 mA represents 50 feet. *L* is the tank level, and *I* is the current.



Linear Relationship between Tank Level and Current

Using the Ohm's Law equation and substituting 0.249 for the value of R_p , you can derive *L* in terms of measured voltage:

 $L = \frac{25 \times l'}{8 \times 0.249} - \frac{25}{2}$

Making Signal Connections

Creating a Program

Measuring Current

Generating Current

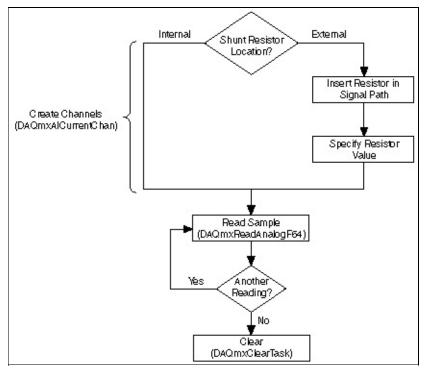
Examples

See Also

Tips on Measuring AC Current

Measuring Current Programming Flowchart

The following flowchart illustrates the main steps required in an NI-DAQmx application to measure current. Alternatively, you can configure a task for measuring current using <u>the DAQ Assistant</u>.



Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the previous flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

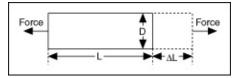
Measuring current is an example of an analog input measurement. Refer to <u>Analog Input Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Tips on Measuring AC Current

To measure AC current, insert a precisely calibrated, low-value resistor into the signal path and measure the voltage drop across the resistor. You must then perform high-pass filtering on the resulting signal to remove the DC component. You can perform this filtering using an analog filter or digital signal processing techniques, such as the filtering tools in the analysis library of LabVIEW.

Measuring Strain

Strain (ϵ) is the amount of deformation of a body due to an applied force. Specifically, strain is the fractional change in length, as shown in the following figure.



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 ($\mu\epsilon$).

When a uniaxial force strains a bar, as in the preceding figure, a phenomenon known as Poisson Strain causes the girth of the bar, D, to contract in the transverse direction, which is perpendicular to the force. The magnitude of this transverse contraction is a material property indicated by its Poisson's Ratio. The Poisson's Ratio of a material is the negative ratio of the strain in the transverse direction to the strain in the axial direction, which is parallel to the force. Poisson's Ratio for steel, for example, ranges from 0.25 to 0.3.

To measure strain, you can use one or more <u>strain gages</u> in a Wheatstone bridge in one of several <u>bridge configurations</u>. Refer to <u>Signal Conditioning Requirements for Strain Gages</u> for more information about strain gages and bridge configurations.

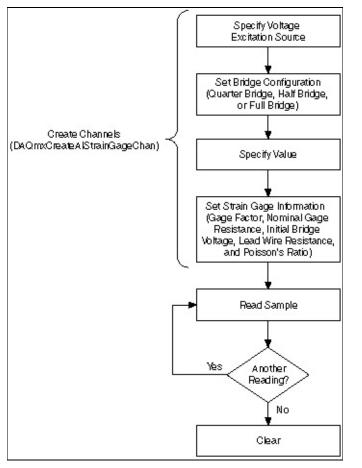
Making Signal Connections

Creating a Program

Examples

Measuring Strain Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to measure strain. Alternatively, you can configure a task to measure strain with a strain gage using <u>the DAQ Assistant</u>.



Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the previous flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

Measuring strain is an example of an analog input measurement. Refer to <u>Analog Input Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Measuring Resistance

Resistance is the opposition to passage of an electric current. One Ohm (Ω) is the resistance through which one volt (V) of electric force causes one ampere (A) to flow. Two common methods for measuring resistance are the 2-wire method and the 4-wire method. Both methods send a current through a resistor with a measurement device measuring the voltage drop from the signal before and after it crosses the resistor. The 2-wire method is easier to implement, but this method is less accurate than the 4-wire method for resistances below 100 Ω . To calculate resistance, use the following equation.

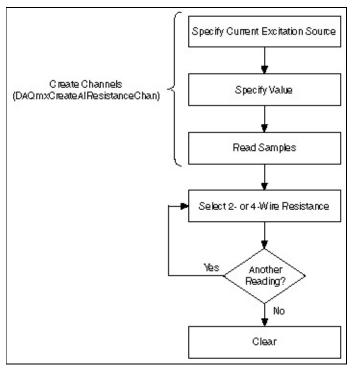
 $R_{(\Omega)} = V_{(V)} / I_{(A)}$

where R is the resistance, V is the voltage, and I is the current.

Making Signal Connections Creating a Program Examples

Measuring Resistance Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to measure resistance. Alternatively, you can configure a task for measuring resistance using <u>the DAQ Assistant</u>.

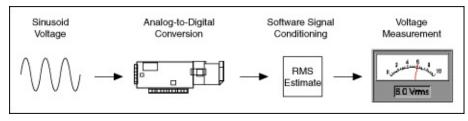


Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the preceding flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

Measuring resistance is an example of an analog input measurement. Refer to <u>Analog Input Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Measuring Voltage

Most measurement devices can measure, or read, voltage. Two common voltage measurements are direct current (DC) and alternating current (AC).



Measuring DC Voltage

DC voltage is useful for measuring phenomena that change slowly with time, such as temperature, pressure, or strain. With DC signals, you want to accurately measure the amplitude of a signal at a given point in time.

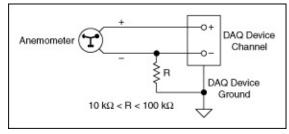
Wind Speed Example

The following figure shows a typical wiring diagram for an anemometer with an output range of 0 to 10 V, which corresponds to wind speed of 0 to 200 mph. Use the following equation to scale the data:

anemometer reading(
$$V$$
)×20 $\left(\frac{mph}{V}\right)$ = wind speed (mph)

Using this equation, a measurement of 3 V would correspond to a wind speed of 60 mph (3 V \times 20 mph/V = 60 mph).

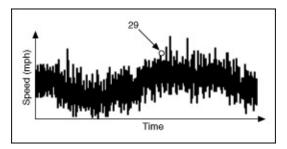
Notice that the wiring diagram in the following figure uses a resistor, R, because an anemometer is usually not a grounded signal source. If the anemometer transducer were already grounded, using a resistor would cause a ground loop and result in erroneous readings.



Averaging

Averaging can improve measurement accuracy for noisy and rapidly changing signals.

The following figure shows what an actual wind speed might look like over time. Due to gusting winds, the speed values look noisy. Notice that the wind speed reading of 29 mph is a peak speed that might give the impression that the wind is holding at 29 mph. A better representation might be to take the average speed over a short period of time.



One common reason for averaging is to eliminate 50 or 60 Hz power line noise. The oscillating magnetic field around power lines can introduce noise voltages on unshielded transducer wiring. Because power line noise is sinusoidal, or shaped like a sine wave, the average over one period is zero. If you use a scan rate that is an integer multiple of the noise and average data for an integer multiple of periods, you can eliminate the line noise. One example that works for both 50 and 60 Hz is to sample at 300 samples per second and average 30 points. Notice that 300 is an integer multiple of both 50 and 60. One period of the 50 Hz noise is 300/50 = 6 points. One period of the 60 Hz noise is 300/60 = 5points. Averaging 30 points is an integer multiple of both periods, so you can ensure that you average whole periods.

Measuring AC Voltage

AC voltage is a waveform that constantly increases, decreases, and reverses polarity. AC voltage is common in household, lab, and industrial devices because most power lines deliver AC voltage. You can measure AC voltages to determine the maximum, minimum, and peak-to-peak values of a signal. The peak-to-peak value of a signal is the maximum voltage swing, from maximum to minimum.

AC Voltage and Root Mean Square (RMS)

Voltage, current, and power are not constant values because AC signals alternate. However, you can use $V_{\rm rms}$ (root mean square) to measure voltage, current, and power such that a load connected to a 120 volts AC (VAC) source develops the same amount of power as that same load connected to a 120 volts DC (VDC) source. With RMS, the power formula for DC also works for AC. For sinusoidal waveforms, $V_{\rm rms}$ = Vpeak/square root of 2. Because voltmeters read $V_{\rm rms}$, the 120 VAC of a typical U.S. wall outlet actually has a peak value of about 170 V.

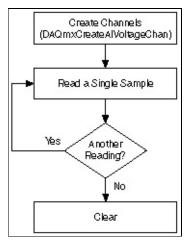
Making Signal Connections

Creating a Program

Examples

Measuring Voltage Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to measure voltage. Alternatively, you can configure a task for measuring voltage using <u>the DAQ Assistant</u>.



Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the previous flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

Measuring voltage is an example of an analog input measurement. Refer to <u>Analog Input Programming Flowcharts</u> for additional flowcharts that can help you create an application.

To measure AC voltages, you generally use a hardware timed acquisition, such as the ones shown in <u>Finite Voltage Measurements</u> and <u>Continuous Voltage Measurements</u>. To measure DC voltages, you often do not need a buffer or hardware timing, so you can use a simple application such as the one shown in <u>Acquiring a Single Sample</u>.

Generating Voltage

You can generate single sample DC signals or time-varying multiple sample signals.

Single Samples—Including Steady Signals

Use single samples if the signal level is more important than the generation rate. For instance, generate one sample at a time if you need to generate a constant, or DC, signal. You can use <u>software or hardware</u> <u>timing</u> if the device supports hardware timing to control when the device generates a signal.

Time-Varying Multiple Samples

Use multiple samples if the generation rate is just as important as the signal level, as in an AC sine wave. Function generators are a common type of device that you can program to produce certain types of waveforms, such as sine, triangle, and square waves. You also can use a DAQ device as a function generator. You do this by generating one cycle of a sine wave, such as with the Sine Generation VI in LabVIEW, storing one cycle of sine wave data in a waveform, and programming the device to generate the values continuously from the waveform one point at a time at a specified rate.

Also called buffered analog output, generating multiple samples involves the following steps:

- 1. Your application writes multiple samples into a buffer.
- 2. All the samples in the buffer are then sent to your device according to the timing you specify. You can use software or hardware timing (if your device supports hardware timing) to control when your device generates a signal.

Making Signal Connections

Creating a Program

Generating Voltage General Programming Flow

Generating Multiple Samples Programming Flow

Examples

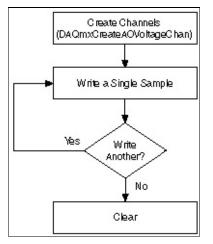
See Also

External Reference Sources for Generating Voltage

What is a Buffer?

Generating Voltage Programming Flowchart

The following flowchart illustrates the main steps required in an NI-DAQmx application to generate voltage. Alternatively, you can configure a task for generating voltage using <u>the DAQ Assistant</u>.



Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the previous flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

Generating voltage is an example of an analog output measurement. Refer to <u>Analog Output Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Measuring and Generating Digital Values

Signals that are read or measured are called input signals. Those signals that are generated are called output, or standard output. Some specialized devices also support a Wired-OR output. Refer to the device documentation for more information about the types of input and output the device supports.

This section covers software-timed digital input/output operationsor unstrobed operations. These signals are controlled by <u>software timing</u>.

Measuring and generating digital values are used in a number of applications, including controlling relays and monitoring alarm states. Generally, measuring and generating digital values is used in laboratory testing, production testing, and industrial process monitoring and control.

Making Signal Connections

Creating a Program

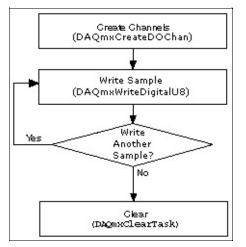
Measuring a Digital Value

Generating a Digital Value

Examples

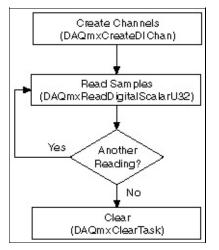
Generating a Digital Value Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to generate digital values. If you prefer, you can configure a task for generating digital values using the <u>DAQ Assistant</u>.



Measuring a Digital Value Programming Flowchart

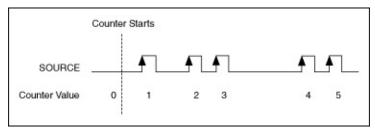
The following flowchart depicts the main steps required in an NI-DAQmx application to measure digital values. If you prefer, you can configure a task for acquiring digital values using the <u>DAQ Assistant</u>.



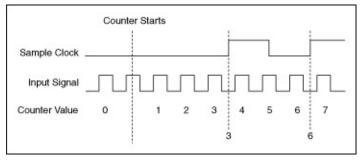
Edge Counting

Edge counting is when a device counts rising or falling edges using a counter channel. You can choose to do either single point or buffered sample clock edge counting.

The following figure shows an example of edge counting in which the <u>counter</u> in a device counts five edges on the input terminal.



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.



In NI-DAQmx, when doing on-demand edge counting, you first arm the counter by calling the Start function/VI. Each subsequent read returns the number of edges counted since the counter was started. If you perform multiple reads without first starting the counter, the counter implicitly starts and stops with each Read function/VI call, and the number of counted edges is not cumulative between read calls.

You also can pause counting with on-demand edge counting in NI-DAQmx by configuring a pause trigger. To configure a pause trigger, use the trigger attributes/properties to set the source terminal of the digital trigger as well as the level on which to pause.

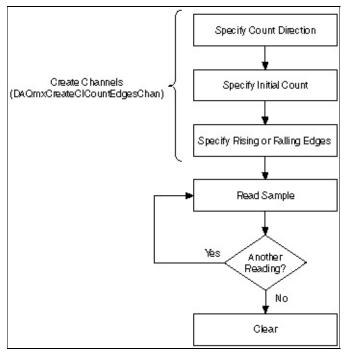
Making Signal Connections

Creating a Program

Examples

Edge Counting Programming Flowchart

The following flowchart depicts the main steps you must complete for counting edges in an NI-DAQmx application. If you prefer, you can configure a task for counting edges using the <u>DAQ Assistant</u>.



Edge counting is an example of a counter measurement. Refer to <u>Counter Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Generating Pulses

Some measurement devices can generate a pulse from the <u>counter</u> of the device. A pulse is a signal whose amplitude deviates from zero for a short period of time. The pulse is either high or low. A high pulse starts low, pulses high, and returns low, and a low pulse starts high, pulses low, and returns high.



Note You can use Butterworth filters only in the LabVIEW Full and Professional Development Systems.

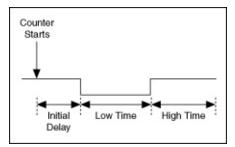
A pulse train is more than one pulse. You can use a pulse or pulse train as a clock signal, a gate, or a trigger for a measurement or a pulse generation. You can use a single pulse of known duration to determine an unknown signal frequency or to trigger an analog acquisition. You can use a pulse train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

Each pulse or pulse train consists of three parts:

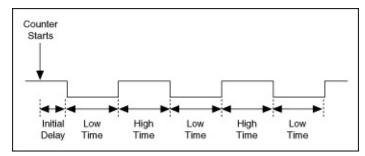
- **Initial Delay**—The amount of time the output remains at the idle state before generating the pulse.
- **High Time**—The amount of time the pulse is at a high level.
- Low Time—The amount of time the pulse is at a low level.

The period of the pulse is the sum of the high time and the low time. The frequency is the reciprocal of the period, 1/period.

The following figure shows the parts of a pulse.

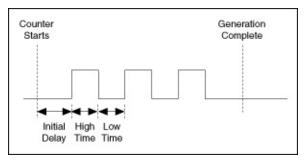


The following figure shows the parts of a pulse train.

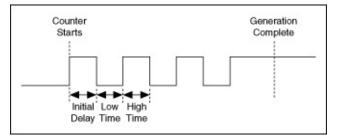


Before you generate a pulse, you need to determine if you want to output the pulse or pulse train in terms of frequency, time, or number of ticks of the counter timebase. For frequency, you need to determine the <u>duty</u> <u>cycle</u>. For time, you specify the high time and the low time. Use the number 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 idle state controls the pulse generation polarity. When you set the idle state to low, the pulse generation starts low for the initial delay, transitions to high for the high time, and transitions to low for the low time, as shown in the following figure. The high time and low time repeat for each pulse.



When you set the idle state to high, the pulse generation starts high for the initial delay, transitions to low for the low time, and transitions to high for the high time, as shown in the following figure. In both cases, the output rests at the idle state after the pulse generation completes.



When generating pulses, you can generate either a single pulse, a finite pulse train, or a continuous pulse train. You can update the high time and

low time of a continuous pulse train generation at any time, including while the application is running. This is useful for applications that require pulse width modulation, such as proportional integral derivative (PID) loop control applications.

By default, single pulses are generated unless you use the Timing function/VI with the implicit timing type. The **Samples Per Channel** input to the Timing function/VI determines the number of pulses to generate for finite pulse trains. Generating a single pulse or a continuous pulse train only requires the use of one counter. However, generating finite pulse trains requires the use of <u>paired counters</u>.

You can configure a variety of triggers with pulse generations. All pulse generations support Start Triggers. Single pulse generation and finite pulse train generation also support the **Retriggerable** property, or attribute, for Start Triggers. To determine if a pulse is complete and the hardware is ready for another Start Trigger, query the **Pulse Done** property. Continuous pulse train generations also support pause triggers. However, you cannot use both the start and the pause trigger at the same time.

You can use the same properties that create the channel to update the rate of the pulse train generation. Because you need two properties to specify the rate of the pulse train, the rate only updates when you set one of the two properties. For example, if you specify the pulse generation in terms of frequency, the **frequency** and **duty cycle** properties control the rate of the generation. However, the rate only updates 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** properties control when the rate updates. When updating the rate of the pulse generation, a complete period of the current rate generates before the new rate takes effect.

Making Signal Connections

Creating a Program

<u>Generating a Pulse</u>

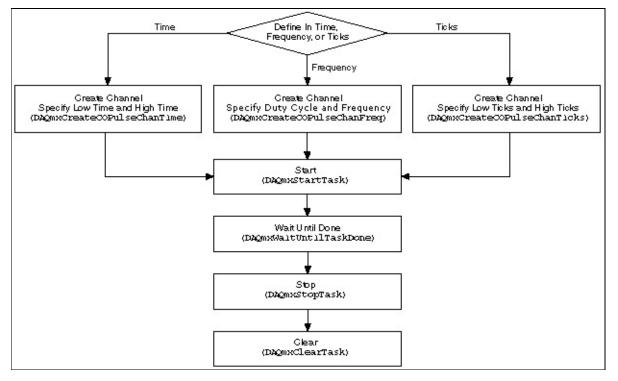
Generating a Finite Pulse Train

Generating a Continuous Pulse Train

Examples

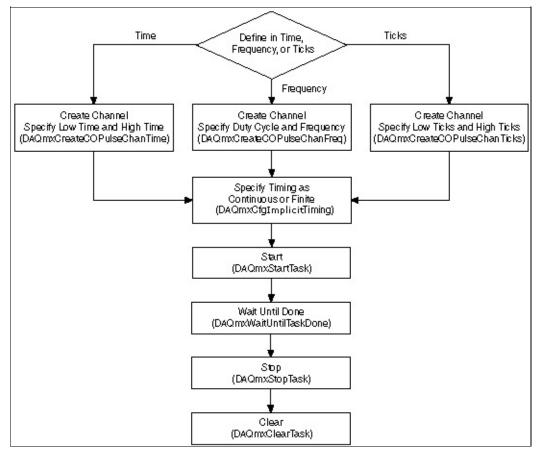
Generating a Pulse Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to generate a pulse. If you prefer, you can configure a task to generate a pulse using the DAQ Assistant.



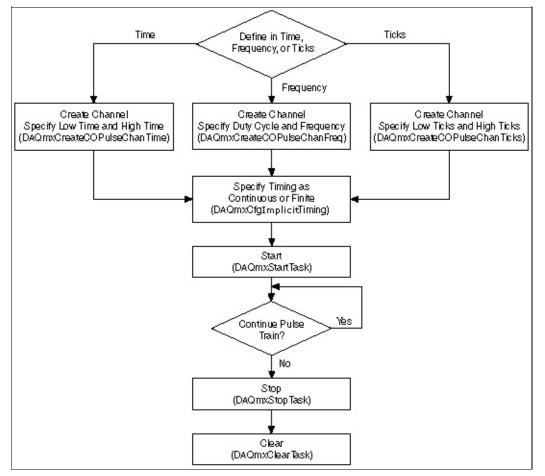
Generating a Finite Pulse Train Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application for generating a finite pulse train. Alternatively, you can configure a task for generating the pulse train using <u>the DAQ Assistant</u>.



Generating a Continuous Pulse Train Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application for generating a continuous pulse train. Alternatively, you can configure a task for generating the pulse train using <u>the DAQ Assistant</u>.



Measuring Acceleration

Acceleration is a change in velocity with respect to time. An <u>accelerometer</u> is a transducer that represents acceleration as a voltage. Accelerometers also can measure vibration and shock. Accelerometers typically convert acceleration measured in g's to voltage. For example, a sensor with a rated output of 10 mV/g should produce 50 mV when subjected to 5 g of acceleration.

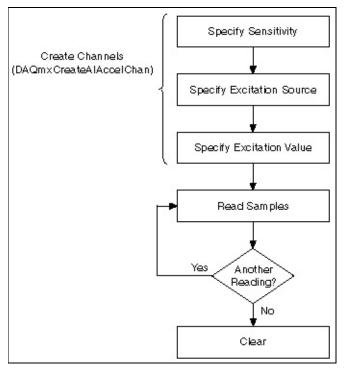
Making Signal Connections

Creating a Program

Examples

Measuring Acceleration Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to measure acceleration. Alternatively, you can configure a task for measuring acceleration using <u>the DAQ Assistant</u>.



Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the previous flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

Measuring acceleration is an example of an analog input measurement. Refer to <u>Analog Input Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Measuring Angular Displacement

Angular displacement is movement around an axis, such as the angular motion of the shaft of a motor. An angular displacement sensor is a device whose output signal represents the rotation of the shaft; it cannot measure the physical displacement of the whole shaft. One type of sensor used to measure angular displacement is a <u>rotary variable</u> <u>differential transformer (RVDT</u>). Another type of sensor used to measure angular displacement is a resolver, which is a rotating transformer that can measure 360° of rotation.

On M Series devices, C Series devices, and NI-TIO-based devices, you can use the counters to <u>perform displacement measurements</u> with <u>quadrature encoders</u>, or angular encoders. You can measure angular position with X1, X2, and X4 angular encoders. You can choose to do either a single-point or a buffered sample clock displacement measurement.

You also can measure velocity with angular encoders, but you need to use a sample clock with a fixed frequency. To measure velocity, use the following formula:

V = D/T

where V is the average velocity, D is the distance, and T is time.

The counter measures the position of the encoder using the A and B signals, which are offset by 90° . The counter also supports the <u>Z index</u>, which provides a precise reference point and is available on some encoders.

Making Signal Connections

Creating a Program

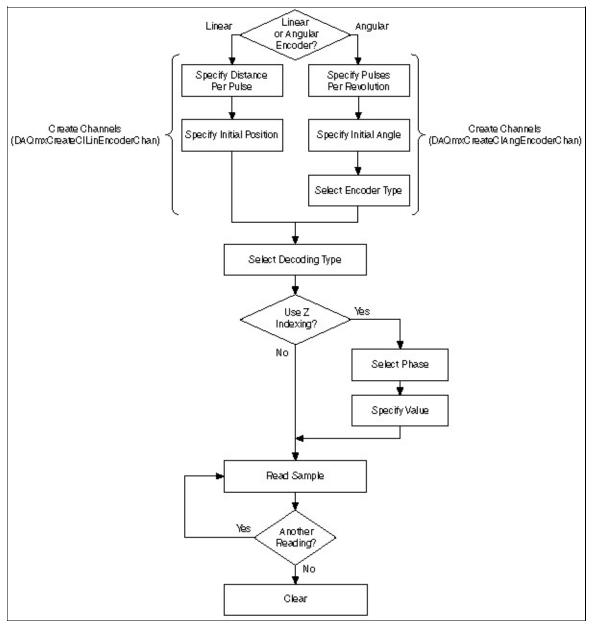
RVDT Programming Flowchart

Encoder Programming Flowchart

Examples

Measuring Position with Encoders Programming Flowchart

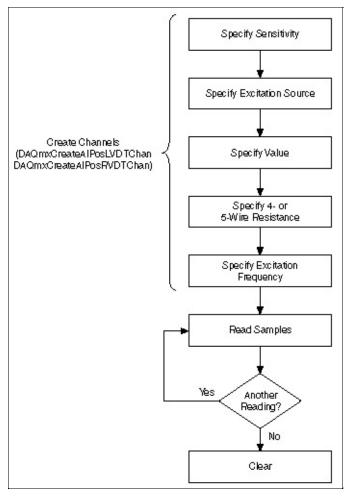
The following flowchart depicts the main steps you must complete for measuring position with an encoder in an NI-DAQmx application. If you prefer, you can configure a task using the <u>DAQ Assistant</u>.



Measuring position with an encoder is an example of a counter measurement. Refer to <u>Counter Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Measuring Position with an RVDT or LVDT Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to measure position with an RVDT or LVDT. Alternatively, you can configure a task for measuring position using <u>the DAQ Assistant</u>.



P

Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the previous flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

Measuring position is an example of an analog input measurement. Refer to <u>Analog Input Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Measuring Linear Displacement

Linear displacement is movement and direction along a single axis. A position or linear displacement sensor is a device whose output signal represents the distance an object has traveled from a reference point. The <u>linear variable differential transformer (LVDT</u>) is a sensor that measures linear displacement.

On M Series devices, C Series devices, and NI-TIO-based devices, you can use the counters to <u>perform displacement measurements</u> with <u>two-pulse encoders</u>. Linear position can be measured with two-pulse encoders. You can choose to do either a single point or a buffered sample clock displacement measurement.

You also can measure velocity with two-pulse encoders, but you need to use a sample clock with a fixed frequency. To measure velocity, use the following formula:

V = D/T

where V is the average velocity, D is the distance, and T is time.

The counter measures the position of the encoder using the A and B signals, which are offset by 90°. The counter also supports the Z index, which provides a precise reference point and is available on some encoders.

Making Signal Connections

Creating a Program

LVDT Programming Flowchart

Encoder Programming Flowchart

Examples

Measuring Analog Frequency

Some devices can measure analog frequency directly using frequencyto-voltage circuitry. Many devices, however, only measure voltage, and you must use software algorithms to convert those measurements to frequency.

Devices that measure analog frequency, such as DSA devices and the SCXI-1126, have circuitry that produces triggers of the same frequency as the measured signal. Every time the signal passes from threshold level minus hysteresis to threshold level, a trigger occurs. A pulse generator uses these triggers and produces a pulse once every frequency cycle. The input frequency range sets the width of this pulse. As the input frequency range increases, the pulse width grows smaller. This pulse train is then converted to a DC signal that has a level proportional to the duty cycle of the pulse train. The duty cycle is the fraction of a period of the pulse train when the pulse is occurring. The DC signal has a voltage that is proportional to the input frequency value.

For devices that cannot measure frequency directly, you need to use software algorithms, such as the Fast Fourier Transform (FFT), to convert voltage to frequency. LabVIEW Full and Professional Development Systems contains advanced analysis VIs that handle these transformations. The LabWindows™/CVI™ full development system also contains advanced analysis functions to help you measure analog frequency. Regardless of whether you use existing VIs or functions or create your own, you need to sample at least twice as fast as the highest frequency component in the signal you are acquiring.

Analog Frequency, Sample Rate, and the Nyquist Theorem

The Nyquist Theorem states that the highest frequency you can accurately represent is half the sampling rate. For instance, to measure the frequency of a 100 Hz signal, you need a sampling rate of at least 200 S/s. In practice, you should use sampling rates of 5 to 10 times the expected frequencies to improve accuracy of measurements.

In addition to sample rate, you need to determine the number of samples to acquire. You must sample a minimum of three cycles of the analog signal. For example, you need to collect at least 15 samples, or points, if you use a sampling rate of 500 S/s to measure the frequency of a 100 Hz signal. Because you sample about five times faster than the signal frequency, you sample about five points per cycle of the signal. You need data from three cycles, so 5 points x 3 cycles = 15 points. In practice, however, you should acquire 10 or more cycles to improve accuracy of measurements, so you should acquire 50 or more samples.

The number of points you collect determines the number of frequency bins that the samples fall into. The size of each bin is the sampling rate divided by the number of points you collect. For example, if you sample at 500 S/s and collect 100 points, you have bins at 5 Hz intervals.

The Nyquist frequency is the bandwidth of the sampled signal and is equal to half the sampling frequency. Frequency components below the Nyquist frequency appear normally. Frequency components above the Nyquist frequency appear aliased between 0 and the Nyquist frequency. The aliased component is the absolute value of the difference between the actual component and the closest integer multiple of the sampling rate. For example, if you have a signal with a component at 800 Hz and you sample at 500 S/s, that component appears aliased at 200 Hz because $|800-(2 \times 500)| = 200(Hz)$.

One way to eliminate aliased components is to use an analog hardware filter before you digitize and analyze the frequency information. If you want to perform all the filtering in software, you must first sample at a rate fast enough to correctly represent the highest frequency component the signal contains. For example, with the highest component at 800 Hz, the minimum sampling rate is 1,600 Hz, but you should sample 5 to 10 times faster than 800 Hz. If the frequency you want to measure is around 100 Hz, you can use a lowpass Butterworth filter with a cutoff frequency (f_c) of

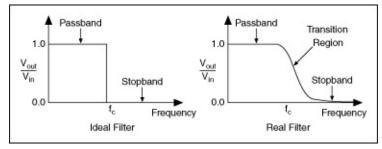
250 Hz to filter out frequencies above 250 Hz and pass frequencies below 250 Hz.



Note LabVIEW includes Butterworth filters with the LabVIEW Full and Professional Development Systems.

Measuring Frequency with Filtering

The following figure shows a lowpass filter.



Lowpass Filter

The Ideal Filter in the figure is optimal. All frequencies above the Nyquist frequency are rejected. The Real Filter in the figure is what you might actually be able to accomplish with a Butterworth filter. The passband is where V_{out}/V_{in} is close to 1. The stopband occurs where V_{out}/V_{in} is close to 0. The frequencies gradually attenuate on the transition region between 1 and 0.

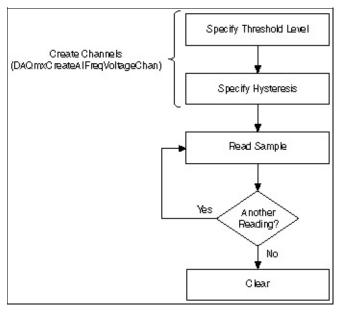
Making Signal Connections

Creating a Program

Examples

Measuring Analog Frequency Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to measure analog frequency. Alternatively, you can configure a task for measuring analog frequency using <u>the DAQ Assistant</u>.

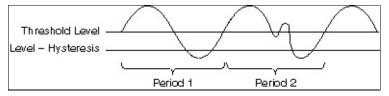


Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the previous flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

Measuring frequency is an example of analog input measurement. Refer to <u>Analog Input Programming Flowcharts</u> for additional flowcharts that can help you create an application.

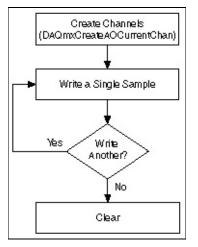
Hysteresis with Analog Frequency Measurements

For waveform repetitions, hysteresis adds a window below the threshold level. Hysteresis is typically used to avoid erroneous measurements due to noise or jitter in the signal. The signal must drop below the threshold level minus the hysteresis before NI-DAQmx recognizes a waveform repetition at the threshold level.



Generating Current Programming Flowchart

The following flowchart illustrates the main steps required in an NI-DAQmx application to generate current. Alternatively, you can configure a task for generating current using <u>the DAQ Assistant</u>.



Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the previous flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

Generating current is an example of an analog output measurement. Refer to <u>Analog Output Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Control

You can create an <u>event response</u> or <u>control loop</u> application in any operating system supported by NI-DAQmx, such as Windows 2000. However, your application can only be <u>deterministic</u> if you have the LabVIEW Real-Time module and use your application on a real-time controller. This section assumes that you are using LabVIEW with NI-DAQmx to create a control application. It does not assume that you have the LabVIEW Real-Time module or the real-time controller.

Event Response

In a <u>control</u> application, an event is the same as an occurrence. This occurrence leads to an action, or a response. An example is monitoring the temperature of an engine. When the temperature rises too high, the engine slows down. The event, in this case, would be the temperature rising above a predetermined level, and the response would be the engine slowing down. Another example comes from manufacturing. In a manufacturing line, a system senses when a part is in front of a station (the event) and takes a reading or manipulates the part (the response). If the system does not sense and respond to the presence of that part in a set amount of time, the manufacturing line creates defective parts.

When creating an event response application, make sure you consider the amount of time needed to respond to the event. For example, if the device controls the temperature of your home, the <u>time</u> to react to events (changes in temperature) is less critical than if the device controls a nuclear reactor. If the application is not time critical, the application does not need to be deterministic, meaning that you do not need the LabVIEW Real-Time Module or a real-time controller.

The relative priority of the task is important as well. Because LabVIEW is multi-threaded, you can separate the application into tasks, each with its own priority. By setting priorities, time-critical tasks can take precedence over non-time-critical tasks. The time-critical task must periodically yield processor resources to the lower-priority tasks so they can execute. By properly separating the time-critical task from lower priority tasks, you can reduce application jitter. Refer to the LabVIEW Real-Time Module Concepts book in the LabVIEW Help for more information about assigning priorities to tasks.

Creating a Program

Examples

See Also

Key Control Concepts

Setting Priorities for Control Applications

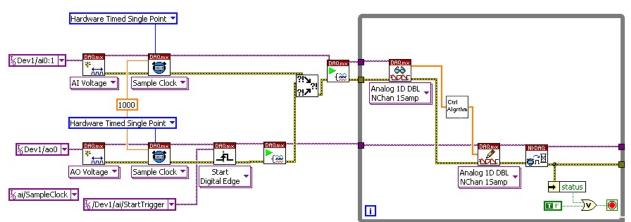
Control Loops

A <u>control application</u> monitors and controls a system. The application continuously loops by reading samples, <u>processing data</u>, and adjusting the output. You can use NI-DAQmx and DAQ devices to create a control application. With the LabVIEW Real-Time Module, you can create deterministic control applications.

Creating a Control Loop Application with NI-DAQmx

The following block diagram shows a typical deterministic control loop application. First, an analog value is read. This value corresponds to the process variable. This value is compared to the set point, which is specified in the Ctrl Algrthm VI in the diagram, and adjusted as necessary within the while loop, possibly using a PID algorithm. The adjusted value is then written. This value corresponds to the actuator output.

In the block diagram, the sampling rates are the same for analog input and output. Because the example shown assumes a single DAQ device, the Start Trigger synchronizes the analog input and analog output tasks. For multiple devices, synchronization works differently. Refer to Synchronization for more information. Notice also that the slave taskthe analog output taskstarts before the analog input task. Finally, within the loop, the Wait for Next Sample Clock VI checks to make sure that the loop executes within the specified sampling rate. If it does not, this VI returns an error.



Examples

See Also

Timing Control Loops

Key Control Concepts

Setting Priorities for Control Applications

Measuring Sound Pressure

Sound pressure is the dynamic variation of the static pressure of air and is measured in force per unit area (Pa). The instantaneous sound pressure is typically averaged over a certain duration to give sound pressure level. Sound pressure level is usually represented on a logarithmic amplitude scale, which is similar to the human perception of hearing. Typical values on this logarithmic scale are a sound level of 0 dB, which is the average threshold of human hearing, 60 to 70 dB for normal conversation, 110 dB at an extremely loud concert, and 150 dB for the noise of a rocket takeoff or a jet engine at close range.

The Sound Pressure Level (SPL or LP) in decibels is defined by the following equation.

SPL = 20 log10 (*p*/*pref*)

where p is the instantaneous sound pressure in Pa and *pref* is 20 μ Pa, the internationally accepted reference for sound pressure measurements, which roughly corresponds to the threshold of human hearing.

You use a <u>microphone</u> to measure sound pressure. The microphone acts as a transducer, creating a voltage signal that is proportional to the instantaneous sound pressure.

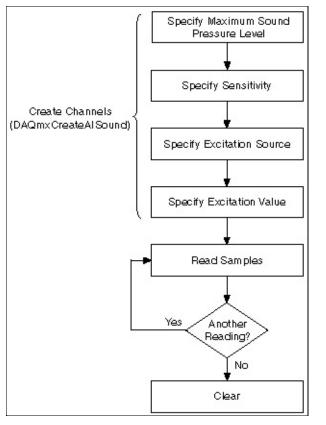
Making Signal Connections

Creating a Program

Examples

Measuring Sound Pressure Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to measure sound pressure. Alternatively, you can configure a task for measuring sound pressure using <u>the DAQ Assistant</u>.



Tip To increase performance, especially when multiple samples are read, include the Start function/VI and Stop function/VI in your application. In the previous flowchart, the Start function/VI would come just before you read samples, and Stop would come just before you clear the task.

Measuring sound pressure is an example of an analog input measurement. Refer to <u>Analog Input Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Measuring Time—Period, Semi-Period, Pulse Width, Two-Edge Separation, and Digital Frequency

You can use the counters on a DAQ device to measure time. Time measurements are <u>period</u>, <u>pulse width</u>, <u>semi-period</u>, <u>frequency</u>, and <u>two-edge separation</u> measurements.

Measuring Digital Frequency

The digital frequency of a signal is the inverse of the period of a signal. To get the frequency of the signal, take the inverse of the period. The formula for frequency is Frequency (in Hz) = Counter Timebase Rate (in Hz) / Count.

The Counter Timebase Rate is a known frequency and is usually a builtin time source. If the counter timebase rate is unknown, you only can make measurements only in terms of ticks of the counter timebase. This may be the case if you are using an external signal for the counter timebase, and the frequency of the external signal is unknown or aperiodic.

Digital frequency is an example of a time measurement. Refer to <u>Configuring a Time Measurement in NI-DAQmx</u> and <u>Two Counter</u> <u>Measurement Method</u> for more information about measuring time.

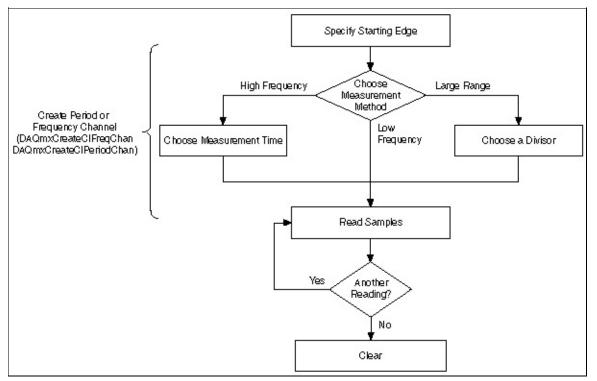
Making Signal Connections

Creating a Program

Examples

Measuring Digital Frequency and Period Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to measure digital frequency or period. Alternatively, you can configure a task for measuring digital frequency using the DAQ Assistant.

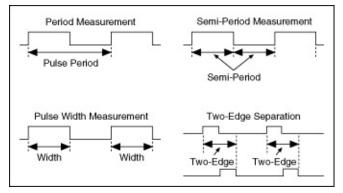


Digital frequency and period are examples of counter measurements. Refer to <u>Counter Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Measuring Period, Semi-Period, Pulse Width, and Two-Edge Separation

You can measure period, semi-period, pulse width, and two-edge separation using <u>counters</u>, such as on a DAQ device, to determine the duration of an event or to determine the interval time between two events.

Period measurements measure the time between consecutive rising or falling edges of a <u>pulse</u>. Semi-period measurements measure the time between consecutive edges. Pulse width measurements measure the time between either a rising and falling edge, or a falling and rising edge. Two-edge separation measurements measure the time between the rising or falling edge of one digital signal and the rising or falling edge of another digital signal.



The formula for period, semi-period, pulse width, and two-edge separation is as follows:

Period, Semi-Period, Pulse Width, or Two-Edge Separation (in seconds) = Count / Counter Timebase Rate (in Hz).

where Count is the number of counter timebase ticks that elapse during one period, semi-period, pulse width, or two-edge separation of the measured input signal or signals.

The Counter Timebase Rate is a known frequency and is usually a builtin time source. If the counter timebase rate is unknown, you only can make measurements only in terms of ticks of the counter timebase. This may be the case if you are using an external signal for the counter timebase, and the frequency of the external signal is unknown or aperiodic.

Period, semi-period, pulse width, and two-edge separation are examples

of time measurements. Refer to <u>Configuring a Time Measurement in NI-DAQmx</u> and <u>Two Counter Measurement Method</u> for more information about measuring time.

Making Signal Connections

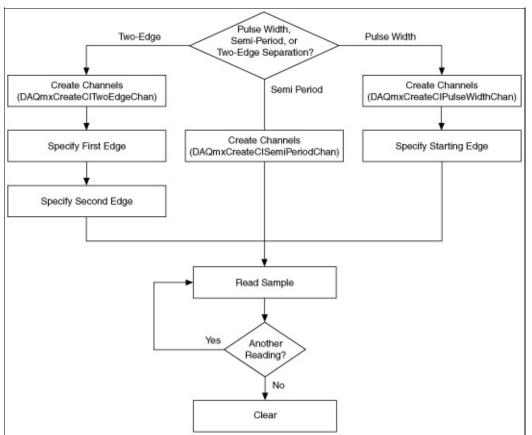
Creating a Program

Measuring Semi-Period, Two-Edge Separation, and Pulse Width Programming Flowchart

<u>Measuring Digital Frequency and Period Programming Flowchart</u> <u>Examples</u>

Measuring Semi-Period, Two-Edge Separation, and Pulse Width Programming Flowchart

The following flowchart demonstrates the main steps required in an NI-DAQmx application to measure semi-period and pulse width. Alternatively, you can configure a task for measuring semi-period and pulse width using the <u>DAQ Assistant</u>.



Period, semi-period, two-edge separation, and pulse width is an example of a counter measurement. Refer to <u>Counter Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Measuring GPS Timestamp

You can take a GPS timestamp measurement with the NI PXI-6608. In a GPS timestamp measurement, the NI PXI-6608 determines the precise time of year using a specialized onboard counter. You can select a single point (on-demand) timestamp or a buffered (sample clock) timestamp.

You can synchronize the GPS timestamp counter to a GPS receiver signal by using a pulse per second (PPS) or an IRIG-B (timecode TTL) synchronization signal from the GPS receiver. PPS does not include any timing information; rather, the PPS accurately reports when the beginning of a second occurs. IRIG-B, on the other hand, has the time encoded in the signal from the beginning of the current year. The GPS counter can latch on the current time upon receiving a hardware gate signal. GPS does not provide year information; however, the time is stored in a 64-bit floating-point number that can be converted to seconds since January 1 of the current year.

When doing an on-demand GPS timestamp measurement, you must first arm the counter by calling the Start function/VI. Each subsequent read returns the number of seconds counted.

When doing a buffered GPS timestamp measurement, the current time is latched on each active edge of the sample clock and stored in the buffer. There is no built-in clock for buffered GPS timestamp measurements, so you must supply an external sample clock.

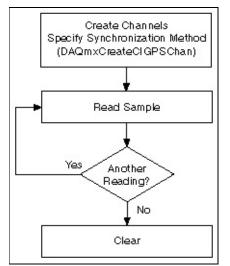
Making Signal Connections

Creating a Program

Examples

GPS Timestamp Programming Flowchart

The following flowchart depicts the main steps required in an NI-DAQmx application to take a GPS timestamp measurement with an NI PXI-6608. Alternatively, you can configure this task using <u>the DAQ Assistant</u>.



GPS timestamp is an example of a counter measurement. Refer to <u>Counter Programming Flowcharts</u> for additional flowcharts that can help you create an application.

Glossary

Prefixes Numbers/Symbols A B C D E F G H I J L M N O P R S T U V W

Symbol	Prefix	Value
n	nano	10 -9
μ	micro	10 -6
m	milli	10 ⁻³
k	kilo	10 ³
М	mega	10 6

Symbol	Meaning
%	percent
+	positive of, or plus
_	negative of, or minus
Ω	ohm
0	degree

Α

acceleration	A change in velocity with respect to time.	
accelerometer	A sensor that represents acceleration as a voltage.	
ADC	Analog-to-digital converter—an electronic device, often an integrated circuit, that converts an analog signal to a digital value.	
address	A character code that identifies a specific location (or series of locations) in memory.	
ADE	Application development environment—some examples include LabVIEW and LabWindows/CVI.	
advanced terminal	A terminal not accessible from the I/O connector or a terminal not commonly used in measurement applications.	
AI	Analog input—acquisition of data.	
amplification	A type of signal conditioning that improves accuracy in the resulting digitized signal by increasing signal amplitude relative to noise.	
analog	Data represented by continuously variable physical quantities.	
AO	Analog output—generation of data.	
angular displacement	Movement about an axis, such as the angular motion of the shaft of a motor.	
angular displacement sensor	A device whose output signal represents the rotation of the shaft, such as a rotary variable differential transformer (RVDT).	
API	Application programming interface—A library of functions, classes or VIs, attributes, and properties for creating applications for your device.	
asynchronous	 Hardware—a signal that occurs or is acted upon at an arbitrary time, without synchronization to another signal, such as a reference clock. Software—a VI or function that begins an 	

operation and returns prior to the completion or termination of the operation.

attenuation The reduction of a voltage or acoustical pressure. Measured referenced to the original voltage.

В

bandwidth	The range of frequencies present in a signal, or the range of frequencies to which a measuring device can respond.
base address	A memory address that serves as the starting address for programmable registers. All other addresses are located by adding to the base address.
bipolar	A signal range that includes both positive and negative values (for example, 5 V to +5 V).
BIOS	Basic Input/Output System—BIOS functions are the fundamental level of any PC or compatible computer. BIOS functions embody the basic operations needed for successful use of the computer hardware resources.
bit	The smallest unit of data used in a digital operation. Bits are binary, so they can be either a 1 or a 0.
buffer	In software, temporary storage for acquired or to-be- generated samples.
bus	The group of conductors that interconnect individual circuitry in a computer. Typically, a bus is the expansion vehicle to which I/O or other devices are connected. Examples of PC buses are the ISA bus and PCI bus.

С

- C Series A family of devices or modules used for analog input, analog output, digital input/output, and counter/timer applications. C Series devices work with chassis based on the CompactDAQ, CompactRIO, and other architectures, and are components of the NI USB-9XXX devices.
- cDAQ The prefix of the product model name of a CompactDAQ device, such as NI cDAQ-9172.
- CH Channel.
- channel
 Physical—a terminal or pin at which you can measure or generate an analog or digital signal. A single physical channel can include more than one terminal, as in the case of a differential analog input channel or a digital port of eight lines. The name used for a counter physical channel is an exception because that physical channel name is not the name of the terminal where the counter measures or generates the digital signal.
 - 2. Virtual—a collection of property settings that can include a name, a physical channel, input terminal connections, the type of measurement or generation, and scaling information. You can define NI-DAQmx virtual channels outside a task (global) or inside a task (local). Configuring virtual channels is optional in Traditional NI-DAQ (Legacy) and earlier versions, but is integral to every measurement you take in NI-DAQmx. In Traditional NI-DAQ (Legacy), you configure virtual channels in MAX. In NI-DAQmx, you can configure virtual channels either in MAX or in a program, and you can configure channels as part of a task or separately.
 - 3. Switch—a switch channel represents any connection point on a switch. It may be made up

of one or more signal wires (commonly one, two, or four), depending on the switch topology. A virtual channel cannot be created with a switch channel. Switch channels may be used only in the NI-DAQmx Switch functions and VIs.

clock A periodic digital signal.

CMRR Common-mode rejection ratio—a measure of the ability of an instrument to reject interference from a commonmode signal, usually expressed in decibels (dB).

code width The smallest detectable change in an input voltage of a DAQ device.

cold-junction A method of compensating for inaccuracies in compensation thermocouple circuits.

CompactDAQ An architecture or chassis for C Series devices.

configuration Refers to the left window in MAX, which contains items such as Data Neighborhood and Devices and Interfaces.

counter/timer A circuit that counts digital edges. Counters and timers usually have from 16 bits to 48 bits (sometimes more) counting capability. The total number of counts possible equals 2^N , where N is the number of bits in the counter. When the edges counted are produced by a clock, elapsed time can be computed from the number of edges counted if the clock frequency is known.

- convert clock The clock on a multiplexed device that directly causes ADC conversions.
- custom scale A method of instructing NI-DAQmx to apply additional scaling to your data. Refer to the Create Scale function/VI in your reference help.

D

- DAC Digital-to-analog converter—an electronic device, often an integrated circuit, that converts a digital value into a corresponding analog voltage or current.
- DAQ Refer to <u>data acquisition</u>.

DAQ A graphical interface for configuring measurement tasks, virtual channels, and scales.

DAQ device A device that acquires or generates data and can contain multiple channels and conversion devices. DAQ devices include plug-in devices, PCMCIA cards, and DAQPad devices, which connect to a computer USB or 1394 (FireWire) port. SCXI modules are considered DAQ devices.

data Samples.

data

acquisition (DAQ)

- 1. Acquiring and measuring analog or digital electrical signals from sensors, transducers, and test probes or fixtures.
- 2. Generating analog or digital electrical signals.
- dB Decibel—the unit for expressing a logarithmic measure of the ratio of two signal levels: dB=20log10 V1/V2, for signals in volts.

DC direct current

delay fromThe amount of time to wait after receiving a sample clocksampleedge before beginning the acquisition of a sample.

delay fromThe amount of time to wait after receiving a Start Triggerstartbefore beginning the operation.

- determinism Characteristic of a system that describes how consistently it can respond to external events or perform operations within a given time limit.
- device1. An instrument or controller you can access as a
single entity that controls or monitors real-world I/O
points. A device often is connected to a host
computer through some type of communication

network.

2. See also <u>DAQ device</u> and <u>measurement device</u>.

digital DIO	A TTL signal. Refer to <u>edge</u> . digital input/output
DMA	direct memory access—A method of transferring data between a buffer and a device that is used most often for high-speed operations.
driver	Software unique to the device or type of device, and includes the set of commands the device accepts.
drop-down listbox	A graphical box with a down arrow button that lets you select values or options from a list. To select a value or option in the selection box, click the down arrow for a complete list of values or options, then use your arrow keys or mouse to select a value or option from the list.
DSUB	D-subminiature connector
DUT	device under test—a device used for testing purposes.

Ε

- E Series A standard architecture for instrumentation-class, multichannel data acquisition devices.
- edge A digital edge is a single rising or falling TTL transition. An analog edge is defined by the slope, level, and hysteresis settings.
- event A digital signal produced from a device or circuit. For an advanced discussion of events, refer to <u>Events</u>.
- excitation Supplying a voltage or current source to energize a sensor or circuit.

F

- fall time The time for a signal to transition from 90% to 10% of the maximum signal amplitude.
- filtering A type of signal conditioning that you can use to remove unwanted frequency components from the signal you are measuring.
- FIFO A type of memory that implements a First In First Out strategy in which samples are removed in the order they were written. FIFOs are typically used as intermediate buffers between an ADC or DAC and the memory buffer.

floating Signal sources with voltage signals that are not connected to signal an absolute reference or system ground.

G

gain The factor by which a signal is amplified, often expressed in decibels (dB). Gain as a function of frequency is commonly referred to as the magnitude of the frequency response function.

groundedSignal sources with voltage signals that are referenced to asignalsystem ground, such as the earth or a building ground.sourcesGrounded signal sources are also called referenced signal

sources.

Н

- hardware The physical components of a computer system, such as the circuit boards, plug-in boards, chassis, enclosures, peripherals, and cables. hardware A means of controlling signal generation. A digital signal, such as a clock on a DAO device, controls the rate of timing generation. hardware A form of triggering in which the source of the trigger is an triggering analog or digital signal. Refer to <u>Software Trigger</u>. hex Hexadecimal—a base-16 numbering system. hysteresis A window around a trigger level that is often used to reduce false triggering due to noise or jitter in the signal.
- Hz Hertz—cycles per second of a periodic signal.

I

IEEE Family of IEEE standards defining a variety of smart P1451 transducer interfaces. All of the standards within this family support the concept of a TEDS that provides selfidentification and plug and play operation to transducers.

IEEE An IEEE standard that defines the concept of plug and play

P1451.4 sensors with analog signals. This is accomplished with the addition of a TEDS in memory, typically an EEPROM, embedded within the sensor and communicated through a simple, low-cost serial connection.

instrument Refer to driver.

driver

- internal A physical channel not accessible from an I/O connector.
- channel Internal channels are often used for calibration and are intended for advanced applications.
- interrupt A method whereby a device notifies the computer of some condition on the device that requires the computer's attention. When this condition is a request for data or a notification of available data, interrupts are used as a data transfer mechanism.

interrupt The relative priority at which a device can interrupt.

- level
- I/O Input/Output—the transfer of data to/from a computer system involving communications channels, operator interface devices, and/or data acquisition and control interfaces.
- IRQ Interrupt ReQuest.
- ISA Industry Standard Architecture—Also refers to a common PC expansion bus.
- isolation A type of signal conditioning in which you isolate the transducer signals from the computer. Isolation makes sure the measurements from the measurement device are not affected by differences in ground potentials.

J

jitter The amount of time that the loop cycle time varies from the desired time.

LED	light-emitting diode—a semiconductor light source.
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.
linear displacement	Movement in one direction along a single axis.
linear displacement sensor	A device that measures linear displacement.
linearization	A type of signal conditioning in which software linearizes the voltage levels from transducers, so the voltages can be scaled to measure physical phenomena.
LSB	least significant bit—often used to refer to the smallest voltage change detectable by an A/D converter or the smallest voltage change that can be generated by a D/A converter.
LVDT	Linear-voltage differential transformer—A sensor used to measure linear displacement. An LVDT consists of a passive transform with one primary and two secondary windings. The primary winding is excited by an audio frequency range AC voltage, whose imbalance between the secondary windings, is proportional to the displacement. The secondary windings are identical, but are normally connected with opposite polarity, so the transducer at resting position will have zero output voltage.

Μ

M Series	A standard architecture for instrumentation-class, multichannel data acquisition devices.
MAX	Measurement & Automation Explorer—A centralized configuration environment that allows you to configure all of your National Instruments devices.
measurement device	DAQ devices such as the M Series multifunction I/O (MIO) devices, SCXI signal conditioning modules, and switch modules.
memory buffer	Refer to <u>buffer</u> .
memory mapping	A technique for reading and writing to a device directly from your program, which avoids the overhead of delegating the reads and writes to kernel-level software. Delegation to the kernel is safer, but slower. Memory mapping is less safe because an entire 4 KB page of memory must be exposed to your program for this to work, but it is faster.
microphone	A transducer that converts acoustical waves into electrical signals.
MIO	multifunction I/O—Designates a category of data acquisition devices that have multiple analog input channels, digital I/O channels, timing, and optionally, analog output channels. An MIO product can be considered a miniature mixed signal tester, due to its broad range of signal types and flexibility. It is also known as multifunction DAQ. An E Series device is an example of an MIO device.
module	A board assembly and its associated mechanical parts, front panel, optional shields, and so on. A module contains everything required to occupy one or more slots in a mainframe. SCXI and PXI devices are modules.
multiplexed mode	An SCXI operating mode in which analog input channels are multiplexed into one module output so that the

cabled DAQ device has access to the multiplexed output as well as the outputs on all other multiplexed modules in the chassis through the SCXI bus. Also called serial mode.

- multiplexer A switching device with multiple terminals that sequentially connects each of its terminals to a single terminal, typically at high speeds. Often used to measure several signals with a single analog input channel.
- multithreading Running tasks of an application for a short amount of time to give the impression of multiple tasks running simultaneously.

Ν

- NI-DAQ Driver software included with all NI measurement devices. NI-DAQ is an extensive library of VIs and functions you can call from an application development environment (ADE), such as LabVIEW, to program all the features of an NI measurement device, such as configuring, acquiring and generating data from, and sending data to the device.
- NI-DAQ 7.x Includes two NI-DAQ drivers—NI-DAQmx and Traditional NI-DAQ (Legacy)—each with its own API, hardware configuration, and software configuration.
- NI-DAQmx The latest NI-DAQ driver with new VIs, functions, and development tools for controlling measurement devices. The advantages of NI-DAQmx over earlier versions of NI-DAQ include the DAQ Assistant for configuring channels and measurement tasks for your device for use in LabVIEW, LabWindows/CVI, and Measurement Studio; increased performance such as faster single-point analog I/O; and a simpler API for creating DAQ applications using fewer functions and VIs than earlier versions of NI-DAQ.
- NI-DAQmx A replica of a 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. An NI-DAQmx simulated device behaves similarly to a physical device. Its driver is loaded, and programs using it are fully verified.
- nonlinearity A measure in percentage of full-scale range (FSR) of the worst-case deviation from the ideal transfer function—a straight line.

This specification is included only for DAQ products, such as signal conditioning products, that do not have an ADC. Because a product with this specification can also be used with a DAQ product with an ADC, this nonlinearity specification must be added to the relative accuracy specification of the DAQ product with the ADC. NRSE Nonreferenced single-ended mode—all measurements are made with respect to a common (NRSE) measurement system reference, but the voltage at this reference can vary with respect to the measurement system ground.

0

onboard Provided by the data acquisition device.

onboard Channels provided by the plug-in data acquisition device. channels

onboard The default source for a particular clock. Usually, the device has dedicated a circuit for producing this signal and its only purpose is to act as the source for a certain clock.

onboard Memory provided by a device for temporary storage of input memory or output data. Typically, onboard memory is a FIFO, which is distinct from computer memory.

operating Base-level software that controls a computer, runs programs,

system interacts with users, and communicates with installed hardware or peripheral devices. Also referred to as OS.

Ρ

parallel mode	A type of SCXI operating mode in which the module sends each of its input channels directly to a separate analog input channel of the device connected to the module.
pattern I/O	pattern input and output—a digital I/O operation on which a clock signal initiates a digital transfer. Because the clock signal is a constant frequency, you can generate and receive patterns at a constant rate.
PCI	peripheral component interconnect—a high-performance expansion bus architecture originally developed by Intel to replace ISA and EISA. PCI has achieved widespread acceptance as a standard for PCs and work stations, and it offers a theoretical maximum transfer rate of 132 Mbytes/s.
PCMCIA	An expansion bus architecture that has found widespread acceptance as a de facto standard in notebook-size computers. PCMCIA originated as a specification for add- on memory cards written by the Personal Computer Memory Card International Association.
PFI	programmable function interface—general purpose input terminals, fixed purpose output terminals. The name of the fixed output signal is often placed on the I/O connector next to the terminal as a hint.
physical channel	Refer to <u>channel</u> .
PID	proportional integral derivative—Combination of proportional, integral, and derivative control actions. Refers to a control method in which the controller output is proportional to the error, its time history, and the rate at which it is changing. The error is the difference between the observed and desired values of a variable that is under control action.
pin	Refer to <u>terminal</u> .
Poisson's	The negative ratio of the strain in the transverse direction

Ratio	(perpendicular to the force) to the strain in the axial direction (parallel to the force).
port	A collection of digital lines. Usually the lines are grouped into either a 8-bit or 32-bit port. Most E Series devices have one 8-bit port.
port width	The number of lines in a port. For example, most E Series devices have one port with eight lines; therefore, the port width is eight.
position sensor	Refer to linear displacement sensor.
posttrigger samples	If there is no Reference Trigger, posttrigger samples are the data acquired after the task is started. If there is a Reference Trigger, this is the data acquired after the Reference Trigger.
plug and play devices	Devices that do not require DIP switches or jumpers to configure resources on the devices. Also called switchless devices.
plug and play sensors	A transducer with an associated TEDS—includes both Virtual TEDS and smart TEDS sensors.
pretrigger samples	Data acquired before the occurrence of the Reference Trigger.
pretriggering	The technique used on a measurement device to keep a circular buffer filled with samples, so that when the Reference Trigger conditions are met, the buffer includes samples leading up to the trigger condition as well as samples acquired immediately after the trigger.
programmed I/O	A data transfer mechanism in which a buffer is not used and instead, the computer reads and writes directly to the device.
propagation delay	The amount of time required for a signal to pass through a circuit.
pulsed output	A form of counter signal generation by which a pulse is generated when a counter reaches a certain value.
PWM	pulse-width modulation

- PXI PCI eXtensions for Instrumentation—a rugged, open system for modular instrumentation based on CompactPCI, with special mechanical, electrical, and software features. The PXI standard was originally developed by National Instruments in 1997 and is now managed by the PXI Systems Alliance.
- PXI trigger The timing bus that connects PXI DAQ devices directly, by means of connectors built into the backplane of the PXI chassis, for precise synchronization of functions. This bus is functionally equivalent to the RTSI bus for PCI DAQ devices.

R

- range The minimum and maximum analog signal levels that the ADC can digitize.
- raw Data that has not been changed in any way. For input, data is returned exactly as received from the device. For output, data is written as is to the device. Refer to <u>unscaled</u> and <u>scaled</u>.
- real time A property of an event or system in which samples are processed as they are acquired instead of being accumulated and processed at a later time.

referenced Signal sources with voltage signals that are referenced to a signal system ground, such as the earth or a building ground. Also called grounded signal sources.

- resolution The smallest amount of input signal change that a device or sensor can detect. The term *discrimination* is also used for resolution.
- rise time The time for a signal to transition from 10% to 90% of the maximum signal amplitude.
- route A connection between a pair of terminals. Any time the source or destination terminal of a signal is specified, a route is created.
- RSE Referenced single-ended mode—all measurements are made with respect to a common reference measurement system or a ground. Also called a grounded measurement system.
- RTD Resistance temperature detector—a metallic probe that measures temperature based on its coefficient of resistivity.
- RTSI bus Real-time system integration bus—the NI timing bus that connects DAQ devices directly, by means of connectors on top of the devices, for precise synchronization of functions. This bus is functionally equivalent to the PXI Trigger bus for PXI DAQ devices.
- RVDT rotary variable differential transformer—a sensor whose output signal represents the rotation of the shaft.

S		
S	seconds	
S	samples. Refer to <u>sample</u> .	
S/s	samples per second—used to express the rate at which a measurement device samples an analog signal.	
sample	A single measurement from a single channel or, for output, a single generation to a single channel.	
sample clock	The clock that initiates an acquisition of one sample from each channel in the scan list. For example, with each sample clock pulse, M Series devices acquire a sample on each analog input channel in a task by multiplexing each channel through a single ADC. On simultaneous sampling devices, the sample clock initiates the simultaneous acquisition of one sample from each channel in the task through a dedicated, per-channel ADC. No multiplexing (and therefore no convert clock) is necessary for S Series devices.	
sample clock rate	Refer to <u>sample rate</u> .	
sample rate	The number of samples per channel per second. For example, a sample rate of 10 S/s means sampling each channel 10 times per second.	
scale	Data that has been mathematically transformed into engineering units. Other manipulations also can be done such as reordering to match the channel order.	
scanning	Method of sequentially connecting channels.	
SCC	Signal conditioning component—low channel count analog or digital I/O modules for conditioning DAQ systems.	
SCXI	Signal Conditioning eXtensions for Instrumentation—the NI product line for conditioning low-level signals within an external chassis near sensors so that only high-level signals are sent to measurement devices in the noisy PC environment. SCXI is an open standard available for all vendors.	

sensor	A device that responds to a physical stimulus (heat, light, sound, pressure, motion, flow, and so on) and produces a corresponding electrical signal.	
signal	A means of conveying information. An analog waveform, a clock, and a single digital (TTL) edge are all examples of signals.	
signal conditioning	The manipulation of signals to prepare them for digitizing.	
smart TEDS sensor	A transducer with a built-in self-identification EEPROM that provides the TEDS.	
software timing	A means of controlling signal generation. The software, such as NI-DAQmx, and the operating system control the rate of generation.	
software trigger	A VI or function that, when it executes, triggers an action such as starting an acquisition.	
source impedance	A parameter of signal sources that reflects current-driving ability of voltage sources (lower is better) and the voltage- driving ability of current sources (higher is better).	
static AO	Analog output operations that use software timing.	
static digital I/O	Software-timed digital I/O operations that do not involve the use of control signals in data transfers. Also known as software-timed I/O or unstrobed I/O.	
strain	The amount of deformation of a body due to an applied force.	
strobed I/O	Any operation in which every data transfer is timed by hardware signals. In the case of sample clock timing, this hardware signal is a clock edge. In the case of handshaking I/O, hardware signals involve two or three handshaking lines.	
STC	system timing controller	
synchronous	 Hardware—a signal that occurs or is acted upon in synchrony with another signal, such as a reference clock. 	
	2. Software—a VI or function that begins an operation	

and returns only when the operation is complete.

Т

- task In NI-DAQmx, a collection of one or more channels, timing, and triggering and other properties that apply to the task itself. Conceptually, a task represents a measurement or generation you want to perform.
- task buffer Refer to <u>buffer</u>.
- TCR temperature coefficient of resistance—the average resistance change per one degree at temperatures between 0 °C and 100 °C.
- TEDS transducer electronic data sheet—standardized data structure, defined by IEEE 1451.4, for describing sensors, typically stored in nonvolatile memory within a sensor. The manufacturer of the sensor stores, into this memory, initial information such as manufacturer name, sensor type, model number, serial number, and calibration data. The TEDS data structure also includes space for custom information such as channel ID, location, position, direction, tag number, etc. Alternatively, the TEDS data may be stored in a file or database record as a Virtual TEDS. For information on IEEE 1451.4compliant TEDS sensors, refer to www.ni.com/pnp.
- TEDS Class I A smart TEDS sensor with a constant-current powered Sensor transducer with a two-wire interface, such as an accelerometer. Class I transducers also include diodes or analog switches with which the multiplexing of the analog signal with the digital TEDS information on the single pair of wires is possible. The digital portion of the mixed-mode interface (Class 1 or Class 2) is based on the 1-Wire protocol from Maxim/Dallas Semiconductor.
- TEDS Class A smart TEDS sensor with separate wires for the analog II Sensor and digital portions of the TEDS mixed-mode interface. The analog input/output of the transducer is left unmodified, and the digital TEDS circuit is added in parallel, such as thermocouples, RTDs, and bridge-based sensors. The digital portion of the mixed-mode interface

	(Class 1 or Class 2) is based on the 1-Wire protocol from
	Maxim/Dallas Semiconductor.
terminal	A named location on a DAQ device where a signal is either generated (output or produced) or acquired (input or consumed).
terminal count	When counting up, an N bit counter reaches its terminal count at 2^N -1. An N bit counter counting down reaches its terminal count at 0.
thermistor	A semiconductor sensor that produces a repeatable change in electrical resistance as a function of temperature. Most thermistors have a negative temperature coefficient.
thermocouple	A temperature sensor created by joining two dissimilar metals. The junction produces a small voltage as a function of the temperature.
threshold	The voltage level a signal must reach for a trigger to occur.
tick	A digital edge of a clock.
timebase	A clock that is divided down to produce another clock or a clock provided to a counter for measuring elapsed time.
Traditional NI-DAQ (Legacy)	An upgrade of the earlier version of NI-DAQ. Traditional NI-DAQ (Legacy) has the same VIs and functions and works the same way as NI-DAQ 6.9. <i>x</i> , except you can use both Traditional NI-DAQ (Legacy) and NI-DAQmx on the same computer, and some hardware is no longer supported.
transducer	Refer to <u>sensor</u> .
transducer excitation	A type of signal conditioning that uses external voltages and currents to excite the circuitry of a signal conditioning system into measuring physical phenomena.
trigger	Any signal that causes a device to perform an action, such as starting an acquisition.
TTL	Transistor-transistor logic—a signal having two discrete levels, a high and a low level.

- U
- unipolar A signal range that is always positive (for example, 0 to +10 V).
- unscaled Samples in the integer form that the hardware produces or requires. Although no mathematical transformations are applied to unscaled data, other manipulations may be done such as reordering to match the channel order.

unstrobed Refer to <u>static digital I/O</u>. I/O

- USB A USB-based family of devices used for analog input, analog
- DAQ output, digital input/output, and counter/timer applications. Some example devices include the NI USB-9201, NI USB-9211, NI USB-9215, NI USB-9221, NI USB-9233, and NI USB-9237 devices. These devices are also referred to as USB DAQ with Integrated Signal Conditioning.

V

V volts

VI Refer to <u>virtual instrument</u>.

virtual Refer to <u>channel</u>.

channel

virtual A program in LabVIEW that models the appearance and instrument function of a physical instrument.

VISA Virtual Instrumentation Software Architecture.

W

waveform data A LabVIEW data type that bundles timing information along with the data.

WDT Refer to <u>waveform data type</u>.

Important Information

Warranty Copyright Trademarks Patents Warning Regarding Use of NI Products

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

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

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

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 <u>National Instruments trademarks</u>.

FireWire® is the registered trademark of Apple Computer, Inc.

Handle Graphics®, MATLAB®, Real-Time Workshop®, Simulink®, Stateflow®, and xPC TargetBox® are registered trademarks, and TargetBox™ and Target Language Compiler™ are trademarks of The MathWorks, Inc.

Tektronix® and Tek are registered trademarks of Tektronix, Inc.

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 CD, or <u>ni.com/patents</u>.

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS. SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE. INJURY. OR DEATH. THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES. INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR

APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at ni.com for technical support and professional services:

- <u>Support</u>—Online technical support resources at ni.com/support include the following:
 - Self-Help Resources—For answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable <u>KnowledgeBase</u>, product <u>manuals</u>, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
 - Free Technical Support—All registered users receive free Basic Service, which includes access to hundreds of Applications Engineers worldwide in the <u>NI Developer</u> <u>Exchange</u> at ni.com/exchange. National Instruments Applications Engineers make sure every question receives an answer.

For information about other <u>technical support options</u> in your area, visit ni.com/services or <u>contact</u> your local office at ni.com/contact.

- Training and Certification—Visit ni.com/training for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- <u>System Integration</u>—If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit ni.com/alliance.
- Declaration of Conformity (DoC)—A DoC is our claim of compliance with the Council of the European Communities using the manufacturers declaration of conformity. This system affords the user protection for electronic compatibility (EMC) and product safety. You can obtain the DoC for your product by visiting ni.com/certification.
- <u>Calibration Certificate</u>—If your product supports calibration, you can obtain the calibration certificate for your product at

ni.com/calibration.

If you searched ni.com and could not find the answers you need, contact your <u>local office</u> or NI corporate headquarters. You also can visit the <u>Worldwide Offices</u> 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.

Branch Offices

Office	Telephone Number
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	33 (0) 1 57 66 24 24
Germany	49 89 7413130
India	91 80 41190000
Israel	972 0 3 6393737
Italy	39 02 41309277
Japan	0120-527196 / 81 3 5472 2970
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 3390150
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
United States (Corporate)	512 683 0100