USB-6421 User Manual

2024-09-16

Contents

USB-6421 User Manual

The USB-6421 User Manual provides detailed descriptions of the product functionality and the step by step processes for use.

Looking for Something Else?

For information not found in the User Manual for your product, such as specifications and API reference, browse ***Related Information***.

Related information:

- USB-6421 Specifications
- NI-DAQmx User Manual
- Software and Driver Downloads
- Release Notes
- License Setup and Activation
- Dimensional Drawings
- Product Certifications
- Letter of Volatility
- Discussion Forums
- NI Learning Center

USB-6421 Overview

The USB-6421 is a multifunction I/O Data Acquisition (DAQ) device featuring a combination of analog input, analog output, digital I/O, and counter/timers. Use the USB-6421 for a wide range of electromechanical test and measurement applications.

USB-6421 Key Features

- 16 single-ended and 8 differential multiplexer-scanning analog input channels
- 2 analog output channels
- 16 digital input/output channels
- 250 kS/s sampling rate on analog input and analog output channels
- 16-bit resolution for analog input and analog output channels
- 4 counters
- Compact and rugged enclosure with removable spring terminals for direct signal connection
- Multiple mounting options, including on a desktop, rack, DIN rail, or wall

USB-6421 Driver Support

Determine the earliest driver version supported for your product.

Tip To optimize product performance, update to the most recent driver version.

Table 1. Earliest Driver Version Support

Product	Driver Name	Earliest Version Support
USB-6421	NI-DAQmx	2024 Q3.1

Related tasks:

- Installing Software

Components of a USB-6421 System

The USB-6421 is designed for use in a system that may require hardware, drivers, and software to optimize USB-6421 for your application. Use the minimum required USB-6421 system components as a starting point for building your system.

Table 2. Minimum Required USB-6421 System Components

Component	Description and Recommendations
Computer	The USB-6421 connects to a computer through USB for power and communication.
USB-6421	Your USB DAQ device.
USB Cable	A USB cable connect your USB DAQ device to your computer.
Mounting Accessories	The USB-6421 supports a number of mounting configurations including: • Resting on a flat surface, unmounted • Mounting to a DIN rail, wall, or panel • Mounting in a rack • Securing in place with zip ties
Backshells for Connector Blocks	You can install backshells over the spring-terminal connector blocks to protect the wiring and provide strain relief. The USB-6421 shipping kit includes backshells, but you can order more separately.
NI-DAQmx	Instrument driver software that provides functions to interact with the USB-6421 and execute measurements using the USB-6421. **Note** For optimal performance, use the most current version of NI-DAQmx with the USB-6421.
NI Applications	NI-DAQmx offers driver support for the following applications: • LabVIEW • LabVIEW Real-Time Module • FlexLogger • LabWindows/CVI • LabWindows/CVI Real-Time Module • NI-DAQmx Measurement Studio Integration (Measurement Studio 2019)

Component	Description and Recommendations
	• C/C++ • .NET • Python

Part Numbers for Recommended Accessories

Use part numbers to purchase the cables and accessories NI recommends to optimize the performance of the USB-6421.

Table 3. Part Numbers for Recommended Accessories

Accessory	Description	Part Number
USB-64xx Mounting Kit for DIN Rail	Use to mount the USB-6421 on a DIN rail horizontally.	789986-01
USB-64xx Mounting Kit for DIN Rail, Wall, or Panel Mount	Use to mount the USB-6421 on a DIN rail vertically or to mount it on a wall or a panel. Includes a right-angle USB Type-C cable for mounting close to a wall.	789955-01
USB-64xx Rack Mount Shelf	Use to mount up to two USB-6421 in a 19-in. rack. Uses 1U of rack height. Includes two right-angle, 2 m USB Type-C cables.	789953-01
Spring-terminal connector	Use to wire the USB-6421.	785502-01
Backshell for spring-terminal connector	Use to protect the wiring and provide strain relief.	785080-01
USB Cable, Type-C to Type-C with Screw, USB 3.2 Gen 1, 2m	Use to connect the USB-6421 to a computer. One end of the cable has a locking thumb screw for securing the cable to the USB-6421.	789956-02

Additional Cabling and Accessory Guidance

NI recommends the following:

- Using individually shielded, twisted-pair wires that are 2 m or less to connect AI signals to the USB-6421.

Related tasks:

- Mounting the USB-6421 on a DIN Rail
- Mounting the USB-6421 on a Wall or Panel
- Mounting the USB-6421 in a Rack

USB-6421 Theory of Operation

The USB-6421 is a multifunction I/O Data Acquisition (DAQ) device that connects to a computer through a USB Type-C interface. This single USB Type-C cable provides power as well as USB 3.0 SuperSpeed communication.

NI Signal Streaming

NI signal streaming technology performs the USB communication. NI signal streaming technology allows for efficient, high-speed, bidirectional data streaming to and from the host computer. NI signal streaming uses USB bandwidth efficiently to support data transfer from the different analog input, analog output, and digital I/O subsystems on the USB-6421.

Analog Input

The USB-6421 analog input provides 16 single-ended or 8 differential input channels. The maximum input range of ±10 V is suitable for connection to a wide variety of electronics or sensors with conditioned outputs. A programmable gain instrumentation amplifier with four different ranges delivers a scaled signal to the 16-bit ADC to maximize resolution for lower voltage signals.

An accurate, low-drift onboard reference plus integrated self-calibration circuitry ensures accurate performance across time and temperature.

Analog Output

The analog output circuitry consists of two independent 16-bit, ±10 V analog outputs. You can use these outputs for stimulus or simple control. You can also make single-point updates or generate waveforms.

Digital I/O, Counters, and Timers

The USB-6421 has 16 digital I/O lines. You can configure each line individually as input or output. You can use the digital I/O lines for either single-point updates or high-speed waveform input/output.

The USB-6421 also has four 32-bit counters that you can route to the digital I/O lines. These counters can count events, measure frequency, measure using an incremental encoder, generate frequency, generate a pulse train, and more.

Flexible routing allows you to route any counter to any of the digital I/O lines.

Timing, Triggering, and Synchronization

Independent timing engines control how each device subsystem operates. This functionality allows the subsystems to either run independently or synchronously.

You can route timing and trigger signals to or from the digital I/O lines to synchronize with other equipment or DAQ devices. The timing and trigger signals include signals such as a Start Trigger to control when an operation starts or a Sample Clock, which toggles each time a sample is acquired.

Enclosure and Connectivity

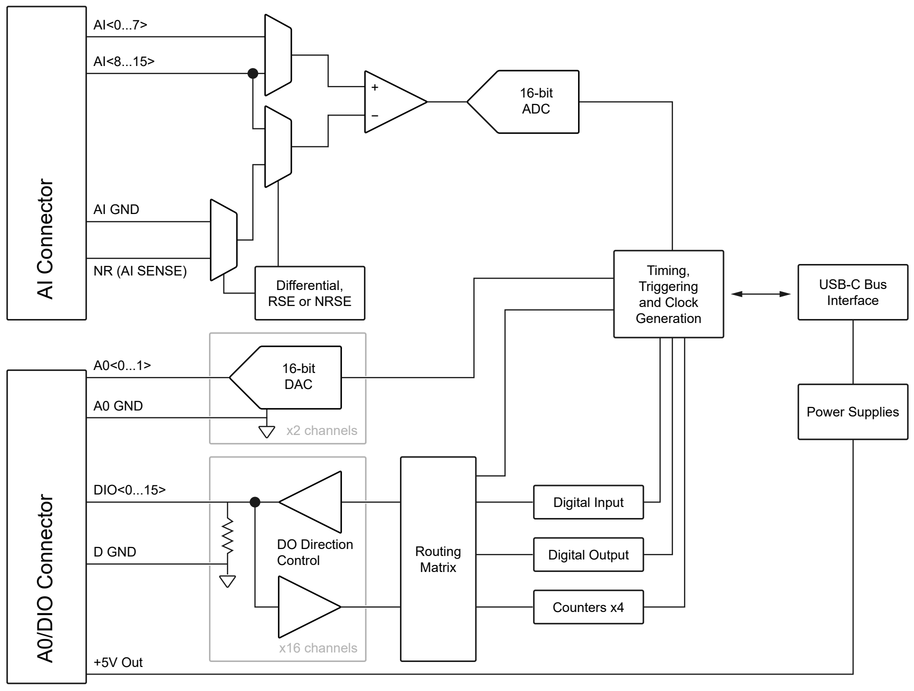
All this functionality is housed in a rugged metal enclosure. You can use the enclosure as-is on a desk or mount it to equipment with built-in zip-tie mounting slots. Optional DIN rail, wall-mount, and rack-mount kits are available for mounting in other environments.

I/O connections are made through removable spring-terminal connectors. These connector blocks use push-in technology that enables you to wire of input signals rapidly. You can use the included backshells to protect the wiring and provide strain relief.

USB-6421 Block Diagram

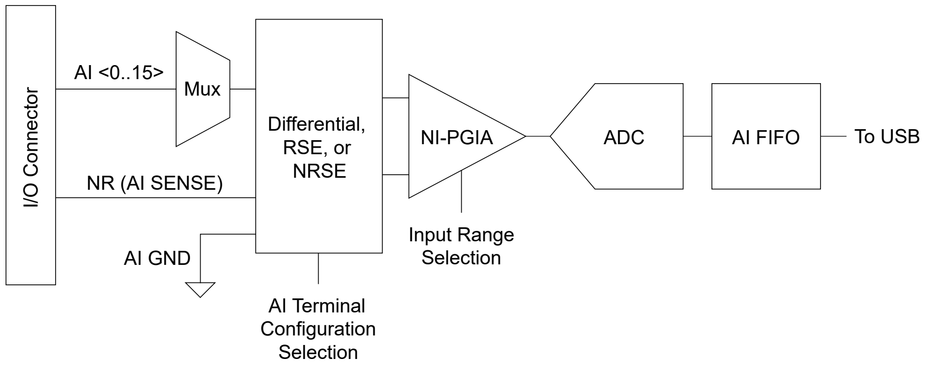
Use the USB-6421 block diagram to learn more about the different analog input, analog output, and digital I/O subsystems of the USB-6421 work together.

Figure 78. USB-6421 Block Diagram



Analog Input Block Diagram

Figure 2. USB-6421 Analog Input Block Diagram



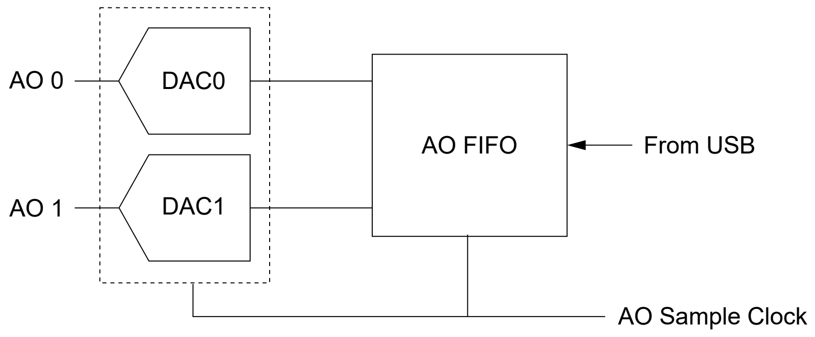
The main blocks featured in the USB-6421 analog input circuitry are as follows:

- **I/O Connector**—Provides connectivity for analog input signals to the USB-6421. The proper way to connect analog input signals depends on the analog input ground-reference settings.
- **Multiplexers (Mux)**—Route one AI channel at a time to the ADC through the NI-PGIA. Each USB-6421 has one ADC.
- **Ground-Reference Settings**—Selects between differential, referenced single-ended, and non-referenced single-ended input modes. Each AI channel can use a different mode.
- **NI Programmable Gain Instrumentation Amplifier (NI-PGIA)**—Can amplify or attenuate an AI signal to ensure that you use the maximum resolution of the ADC. The NI-PGIA is a measurement- and instrument-class amplifier that minimizes settling times for all input ranges. The USB-6421 uses the NI-PGIA to deliver high accuracy even when sampling multiple channels with small input ranges at fast rates. The USB-6421 can sample channels in any order. You can individually program each channel in a sample with a different input range.
- **Analog-to-Digital Converter (ADC)** —Digitizes the AI signal by converting the analog voltage into a digital number.
- **AI FIFO**—A large first-in-first-out (FIFO) buffer that holds data during AI acquisitions to ensure that no data is lost. The USB-6421 can perform both single and multiple A/D conversions of a fixed or infinite number of samples. The USB-6421 can handle multiple A/D conversion operations with direct memory access (DMA) or programmed I/O. You can stream the AI data through USB at full rate.

Analog Output Block Diagram

The USB-6421 has two voltage output channels capable of either software-timed single-point updates or hardware-timed waveform generation.

Figure 81. USB-6421 Analog Output Block Diagram



The main blocks featured in the USB-6421 analog output circuitry are as follows:

- **Digital-to-analog converters (DACs)**—Convert digital codes to analog voltages.
- **AO FIFO**—Enables analog output waveform generation. It is a first-in-first-out (FIFO) memory buffer between the computer and the DACs. It allows you to either stream a waveform with new data continuously provided from USB or download an entire waveform to the USB-6421 where it can be regenerated entirely from the onboard buffer.
- **AO Sample Clock**—Reads a sample from the AO FIFO and generates the AO voltage.

Digital I/O Block Diagram

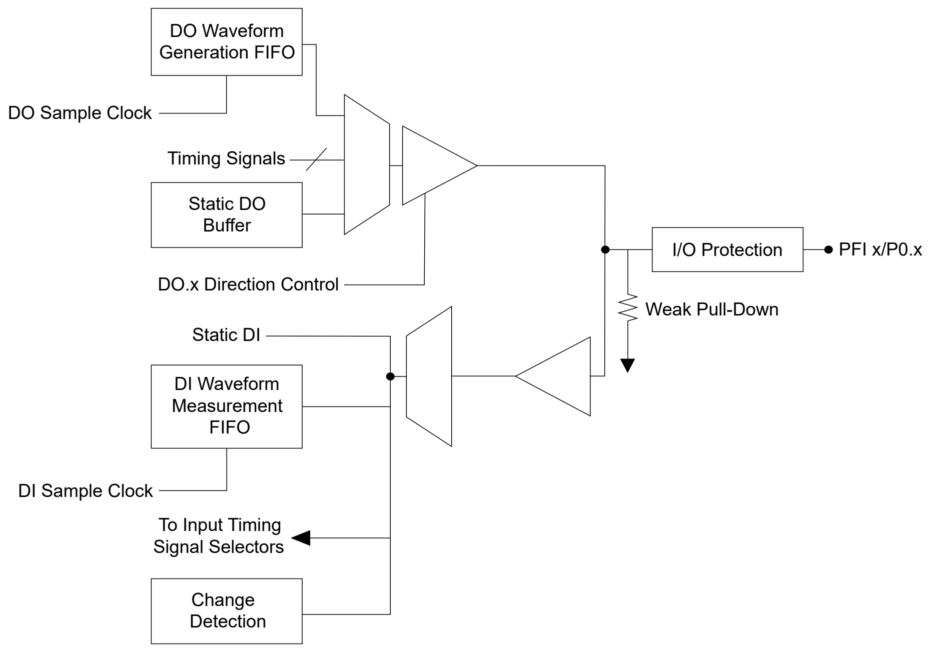
The USB-6421 has 16 bidirectional digital I/O signals that are grouped together in software as a single port referred to as Port 0.

These signals can function as digital I/O as well as counter, timer, or triggering I/O. When used as counter, timer, or triggering I/O, the lines are referred to as ***Programmable Function Interface (PFI)*** lines. The digital I/O lines on the USB-6421 support the following features:

- Direction and function of each terminal individually controllable
- Static digital input and output
- High-speed digital waveform generation
- High-speed digital waveform acquisition
- Digital input change detection trigger/interrupt
- Timing input signal for analog input, analog output, digital input, digital output, or counter/timer functions
- Timing output signal from analog input, analog output, digital input, digital output, or counter/timer functions

The following figure shows the circuitry of one digital I/O line. Each digital I/O line is similar.

Figure 4. USB-6421 Digital I/O Block Diagram



In software, these channels are referred to as port0/line0:15 when used as digital I/O and PFI 0:15 when used for other purposes, such as timing I/O.

USB-6421 Examples

NI installs example code with your software or driver that demonstrates the functionality of USB-6421. Use these examples to learn about the product or accelerate your own application development.

Most NI products install examples that you can access directly or from within NI software. The example experience can differ slightly across products and versions.

LabVIEW Examples

USB-6421 LabVIEW examples are located in the `%ProgramFiles%\NI\LVAddons\nidaqmx\1\examples\NI-DAQmx` directory.

Table 4. Common USB-6421 LabVIEW Examples

Example Name	Description
Voltage (with Events) - Continuous Input	Demonstrates how to continuously acquire buffered voltage measurements. It uses software events to ensure that the program does not become unresponsive while waiting for samples to become available.
Voltage - Continuous Input	Demonstrates how to continuously acquire voltage measurement.
Voltage - Finite Input	Demonstrates how to acquire a single buffer of voltage measurements.
Voltage - SW-Timed Input	Demonstrates how to acquire a voltage based off of software timing.
Voltage - Continuous Output	Demonstrates how to continuously re-generate analog output data from a buffer in your computer's memory.
Voltage - Finite Output	Demonstrates how to output a voltage based on a finite amount of data from a buffer in your computer's memory.
Voltage - On Demand Output	Demonstrates how to generate a user-defined voltage using the analog output.

Example Name	Description
Counter - Count Edges	Demonstrates how to continuously read back the number of digital edges that have been counted by a counter input.
Counter - Read Encoder	Demonstrates how to use a counter to continually monitor the angular position of an encoder.
Counter - Read Pulse Duty Cycle and Frequency (Continuous)	Demonstrates how to configure a pulse measurement to acquire frequency and duty cycle based off of an external clock.
Counter - Continuous Output	Demonstrates how to continuously generate digital pulses using a counter output.
Counter - Finite Output	Demonstrates how to generate a finite number of digital pulses using a counter output. On some hardware, this requires the use of a second counter to generate a gate signal.
Counter - Single Pulse Output	Demonstrates how to generate a single pulse using a counter output. The single pulse may be triggered once or repeatedly from an external source.
Digital - Change Detection	Demonstrates how to acquire a continuous amount of digital data based off of the changes of an external signal across one or many lines on either the rising or falling edges.
Digital - Continuous Input	Demonstrates how to acquire a continuous amount of digital data based off of a sample clock.
Digital - SW-Timed Input	Demonstrates how to acquire the state of digital lines based off of software timing.
Digital - Continuous Output	This example Demonstrates how to generate a continuous amount of digital data based off of a sample clock.
Digital - Finite Output	Demonstrates how to generate a finite amount of digital data based off of a sample clock.
Digital - SW-Timed Output	Demonstrates how to generate digital output data using software timing.

Python Examples in

USB-6421 Python examples are located in GitHub at `github.com/ni/nidaqmx-python/tree/master/examples`.

Table 5. Common USB-6421 Python Examples

Example Name	**Description**
`analog_in/cont_voltage_acq_int_clk.py`	Demonstrates how to acquire a continuous amount of data using the USB-6421's internal clock.
`analog_in/cont_voltage_acq_int_clk_every_n_samples_event.py`	Demonstrates how to use Every N Samples events to acquire a continuous amount of data using the USB-6421's internal clock. The Every N Samples events indicate when data is available from NI-DAQmx.
`analog_in/voltage_acq_int_clk.py`	Demonstrates how to acquire a finite amount of DATA using the USB-6421's internal clock.
`analog_in/voltage_sample.py`	Demonstrates how to acquire a voltage measurement using software timing.
`analog_out/cont_gen_voltage_wfm_int_clk.py`	Demonstrates how to output a continuous periodic waveform using an internal sample clock.

Example Name	Description
`analog_out/ cont_gen_voltage_wfm_int_clk_every_n_samples_event.py`	Demonstrates how to use a Every N Samples events to output a continuous periodic waveform to an Analog Output Channel using an internal sample clock. The Every N Samples events indicate when the specified number of samples generation is complete.
`analog_out/gen_voltage_wfm_int_clk.py`	Demonstrates how to output a finite number of voltage samples to an Analog Output Channel using an internal sample clock.
`analog_out/voltage_update.py`	Demonstrates how to output a single Voltage Update (Sample) to an Analog Output Channel.
`counter_in/cnt_dig_event.py`	Demonstrates how to count digital events on a Counter Input Channel. You can configure the Initial Count, Count Direction, and Edge.
`counter_in/read_pulse_freq.py`	Demonstrates how to configure a pulse measurement to acquire frequency

Example Name	Description
	and duty cycle.
`counter_out/cont_gen_dig_pulse_train.py`	Demonstrates how to generate a continuous digital pulse train from a Counter Output Channel. You can configure the Frequency, Duty Cycle, and Idle State.
`counter_out/write_single_dig_pulse.py`	Demonstrates how to generate a single digital pulse from a Counter Output Channel. You can configure the Initial Delay, High Time, Low Time, and Idle State.
`digital_in/acq_dig_port_int_clk.py`	Demonstrates how to input a finite digital pattern using the USB-6421's internal clock.
`digital_in/cont_acq_dig_lines_int_clk.py`	Demonstrates how to acquire a continuous digital waveform using the USB-6421's internal clock.
`digital_in/read_dig_lines.py`	Demonstrates how to read values from one or more digital input channels.
`digital_in/read_dig_port.py`	Demonstrates how to read values from a digital input port.
`digital_out/cont_gen_dig_port_int_clk.py`	Demonstrates how

Example Name	Description
	to output a continuous digital pattern using the USB-6421's clock.
`digital_out/gen_dig_line_int_clk.py`	Demonstrates how to output a finite digital waveform using the USB-6421's internal clock.
`digital_out/write_dig_lines.py`	Demonstrates how to write values to a digital output channel.
`digital_out/write_dig_port.py`	Demonstrates how to write values to a digital output port.
`system_properties.py`	Demonstrates how to use system properties in NI-DAQmx.

.NET Examples

USB-6421 examples in .NET are located in the `Users\Public\Public Documents\National Instruments\NI-DAQ\Examples\...` directory.

Table 6. Common USB-6421 .NET Examples

Example Name	Description
`Analog In\Measure Voltage\AcqMultVoltageSamples_SWTimed\CS\AcqMultVoltageSamples_SWTimed.sln`	Demonstrates how to acquire a finite amount of data using a software timer.
`Analog In\Measure Voltage\AcqOneVoltageSample\CS\AcqOneVoltageSample.sln`	Demonstrates how to acquire a single reading from a constant or slowly varying signal.
`Analog In\Measure Voltage\`	Demonstrates how to acquire a finite

Example Name	Description
`AcqVoltageSamples_IntClk\CS\ AcqVoltageSamples_IntClk.sln`	amount of data using an internal clock.
`Analog In\Measure Voltage\ ContAcqVoltageSamples_IntClk\CS\ ContAcqVoltageSamples_IntClk.sln`	Demonstrates how to acquire a continuous amount of data using the USB-6421's internal clock.
`Analog In\Measure Voltage\ ContAcqVoltageSamples_SWTimed\CS\ ContAcqVoltageSamples_SWTimed.sln`	Demonstrates how to acquire a continuous amount of data using a software timer.
`Analog Out\Generate Voltage\ ContGenVoltageWfm_IntClk\CS\ ContGenVoltageWfm_IntClk.sln`	Demonstrates how to continuously output a periodic waveform using an internal sample clock.
`Analog Out\Generate Voltage\ GenMultVoltUpdates_IntClk\CS\ GenMultVoltUpdates_IntClk.sln`	Demonstrates how to output multiple voltage updates (samples) to an analog output channel.
`Analog Out\Generate Voltage\ GenMultVoltUpdates_SWTimed\CS\ GenMultVoltUpdates_SWTimed.sln`	Demonstrates how to output multiple voltage updates (samples) to an analog output channel in a software timed loop.
`Analog Out\Generate Voltage\ GenVoltageUpdate\CS\ GenVoltageUpdate.sln`	Demonstrates how to output a single voltage update (sample) to an analog output channel.
`Counter\Count Digital Events\ CountDigEvents\CS\CountDigEvents.sln`	Demonstrates how to count digital events on a Counter Input Channel. You can configure the Initial Count, Count Direction, and Edge. This example shows how to count edges on the counter's default source pin, but you could easily expand it to count edges on any PFI, RTSI, or internal signal.
`Counter\Count Digital Events\ CountDigEventsBuffContinuous_ExtClk\CS\ CountDigEventsBuffContinuous_ExtClk.sln`	Demonstrates how to count buffered digital events on a Counter Input channel. You can configure the initial count, count direction, edge, and sample clock source. Edges are counted on the counter's default input terminal, but you could easily modify it to count edges on a PFI or RTSI line. Note: For buffered event counting, an external sample clock is

Example Name	Description
	necessary to signal when a sample should be inserted into the buffer. Specify the source terminal of the external clock in the clock source text box when you run the example.
`Counter\Generate Pulse\GenDigPulse\CS\GenDigPulse.sln`	Demonstrates how to generate a single digital pulse from a counter output channel. You can configure the initial delay, high time, low time, and idle state in software. This example shows how to configure the pulse in terms of time, but you can easily modify it to generate a pulse in terms of frequency and duty cycle or ticks.
`Counter\Generate Pulse\GenDigPulseTrain_Continuous\CS\GenDigPulseTrain_Continuous.sln`	Demonstrates how to generate a continuous digital pulse train from a counter output channel. You can configure the frequency, duty cycle, and idle state. This example shows how to configure the pulse in terms of frequency and duty cycle, but you can easily modify it to generate a pulse in terms of time or ticks.
`Digital\Generate Values\WriteDigChan\CS\WriteDigChan.sln`	Demonstrates how to write values to a digital output channel.
`Digital\Generate Values\WriteDigPort\CS\WriteDigPort.sln`	Demonstrates how to write values to a digital output port.
`Digital\Read Values\ReadDigChan\CS\ReadDigChan.sln`	Demonstrates how to read values from one or more digital input channels.
`Digital\Read Values\ReadDigPort\CS\ReadDigPort.sln`	Demonstrates how to read a single value from a digital port.

Browsing and Searching for Examples in NI Example Finder

Use NI Example Finder to browse and to search for examples.

You can use NI Example Finder to find examples for the following products.

- LabVIEW
- LabWindows/CVI
- NI drivers accessible from LabVIEW
- NI drivers accessible from LabWindows/CVI

1. Launch LabVIEW or LabWindows/CVI.
2. Open NI Example Finder.

Option	Description
LabVIEW	Select **Help » Find Examples.** from the menu bar.
LabWindows/CVI	Click **Find Examples...** from the Examples section of the Welcome Page.

NI Example Finder launches.

3. **Optional:** Configure NI Example Finder for LabWindows/CVI.
   a. Click **Setup**. Configure Example Finder opens.
   b. In Configure Example Finder, click **Software**, then select LabWindows/CVI, and click **OK**.

NI Example Finder updates with all the examples for LabWindows/CVI.

4. Search the example VIs for your product.

Option	Description
Click the Browse tab.	Choose **Browse** when you want to drill down through folders to find examples organized by task category. **Tip** Examples installed with NI drivers or third-party drivers are often found within the **Hardware Input and Output** folder. Examples installed with toolkits or modules are often found within the **Toolkits and Modules** folder.
Click the Search tab.	Choose **Search** when you want to find examples by searching for topics, products, or modules relevant to your application.

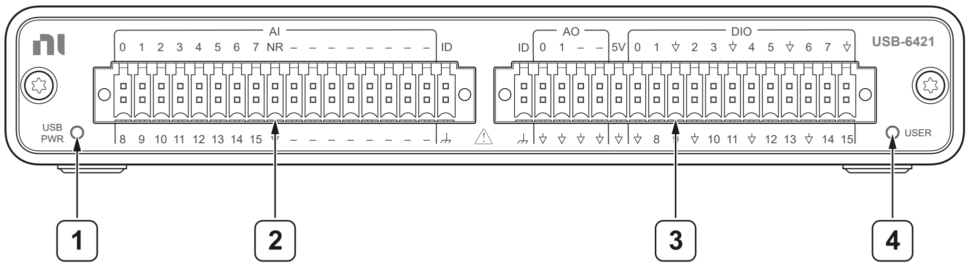
5. To open an example, double-click the folder or the example.

> **Tip** You can modify an example VI to fit your application. You can also copy and paste from one or more examples into a VI that you create.

USB-6421 Front Panel

Use the front panel to identify the connectors and LEDs of the USB-6421.

Figure 78. USB-6421 Front Panel



1. USB PWR LED
2. AI connector
3. AO/DIO connector
4. USER LED

USB PWR LED

The USB PWR LED indicates the power status and communication activity.

Table 7. USB-6421 USB PWR LED Indicator Status

Color	Pattern	Meaning
Off	—	The USB-6421 is not powered or not ready
Green	Solid	The USB-6421 is powered and ready but not actively communicating with the host PC
	Blinking	USB-6421 is powered on and actively communicating with the host PC

Why Is the USB PWR LED Off?

The USB PWR LED remains off when the USB-6421 is not powered or ready for operation.

This could be caused by the following reasons:

- The USB cable is not connected.
- The upstream PC or hub is not connected.
- The PC has put the USB-6421 into a low-power suspend or sleep state.
- The PC does not have a version of NI-DAQmx installed that supports the USB-6421.
- The USB-6421 is busy updating firmware.

Note When you connect the USB-6421 to a computer that has a different version of NI-DAQmx installed than the computer that the USB-6421 was connected to previously, NI-DAQmx checks to see if the USB-6421 requires a firmware update. If it does, the USB-6421 automatically updates the firmware. This update may take several minutes. During this time, the LED will remain off until the USB-6421 is ready. Leave the USB-6421 connected to the computer during this time.

AI Connector

Provides pins for Analog Input signal connections.

Related reference:

- USB-6421 AI Connector Pinout

AO/DIO Connector

Provides pins for analog output and digital I/O signal connections.

Related reference:

- USB-6421 AO/DIO Connector Pinout

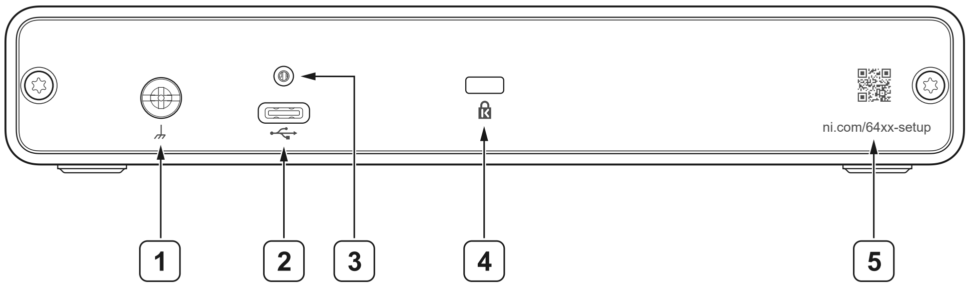
USER LED

This feature is not supported yet, but support will be added in the future.

USB-6421 Rear Panel

Use the rear panel to identify the features and connectors of the USB-6421.

Figure 78. USB-6421



1. Chassis ground lug
2. USB Type-C connector
3. USB Type-C jack socket
4. Security lock slot
5. Setup QR code

Chassis Ground Lug

Use the chassis ground lug to connect the USB-6421 to earth ground. Ensure this screw is always attached during operation because it also secures the USB connector.

USB Type-C Connector

Use the USB Type-C connector to connect the USB-6421 to a host computer through a USB Type-C cable.

USB Type-C Jack Socket

Use to secure your USB Type-C cable to the USB-6421.

Security Lock

Insert a security cable to secure the USB-6421.

Setup QR Code

Use the QR code on the USB-6421 to access the setup page. You can also access the setup page by going to `ni.com/64xx-setup`.

USB-6421 Pinouts

USB-6421 AI Connector Pinout

Use the pinout to connect to analog input terminals on the USB-6421.

Figure 58. USB-6421 AI Connector Pinout

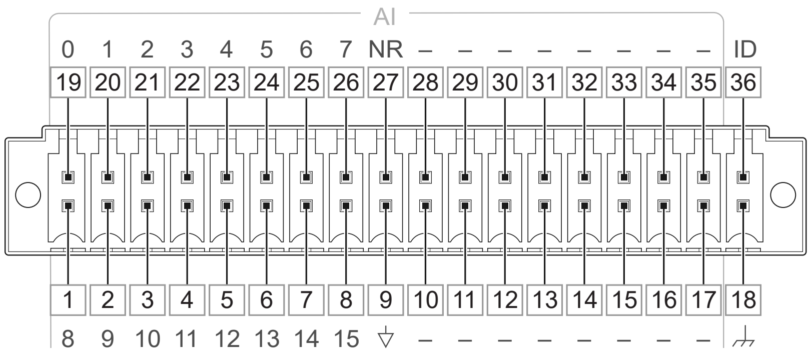


Table 8. USB-6421 AI Connector Pin Assignments

Pin	Signal
1	AI 8
2	AI 9
3	AI 10
4	AI 11
5	AI 12
6	AI 13
7	AI 14
8	AI 15
9	AI GND
10	No connect
11	No connect
12	No connect
13	No connect
14	No connect

Pin	Signal
15	No connect
16	No connect
17	No connect
18	CHSGND
19	AI 0
20	AI 1
21	AI 2
22	AI 3
23	AI 4
24	AI 5
25	AI 6
26	AI 7
27	NR (AI SENSE)
28	No connect
29	No connect
30	No connect
31	No connect
32	No connect
33	No connect
34	No connect
35	No connect
36	ID 0

Table 9. USB-6421 AI Connector Signal Descriptions

Signal	Function	Reference	Direction	Description
AI <0..7>	Analog input channels	Varies	Input	Supports differential or

Signal	Function	Reference	Direction	Description
				single-ended measurement modes. The default configuration is differential mode. In differential mode, these channels are the positive input for the differential pair. The negative input of the differential pair is located directly beneath the positive input. In single-ended mode, each signal is a separate analog input voltage channel. The ground reference in single-ended mode is configurable. In referenced single-ended (RSE) mode, AI GND is the reference for the voltage measurement. In non-referenced single-ended (NRSE) mode, the NR pin is the reference. **Note** You can

Signal	Function	Reference	Direction	Description
				configure the input mode per channel.
AI <8..15>	Analog input channels	Varies	Input	Supports single-ended measurements only. The default configuration is RSE mode. In RSE mode, AI GND is the reference for the voltage measurement. In NRSE mode, the NR pin is the reference. For differential measurements, refer to the descriptions for AI <0..7>.
AI GND	Analog input ground	—	—	The reference point for single-ended measurements in RSE mode and the bias current return point for differential measurements. AI GND, AO GND, D GND, and CHSGND are all connected internally.

Signal	Function	Reference	Direction	Description
NR (AI SENSE)	AI SENSE for NRSE mode	—	Input	The AI SENSE pin is labeled "NR" because it is used when the input terminal is configured to NRSE mode. In NRSE mode, AI SENSE acts as a remote sense of a reference voltage that can be at a different voltage potential than AI GND.
CHSGND	Chassis ground	—	—	Connects directly to the chassis ground of the USB-6421 enclosure. It can be used as a termination point for shielded cables to help improve measurement quality.
ID 0	—	—	—	This feature is not supported yet.

Related reference:

- Analog Input Terminal Configurations

USB-6421 AO/DIO Connector Pinout

Use the pinout to connect to analog output and digital input/output terminals on the USB-6421.

Figure 69. USB-6421 AO/DIO Connector Pinout

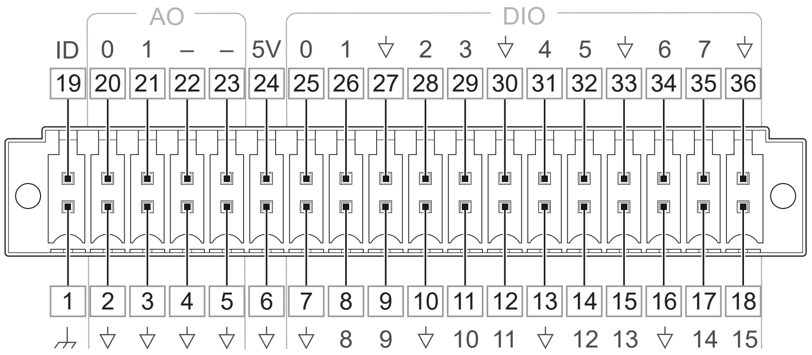


Table 10. USB-6421 AO/DIO Connector Pin Assignments

Pin	Signal
1	CHSGND
2	AO GND
3	AO GND
4	AO GND
5	AO GND
6	D GND
7	D GND
8	PFI 8/P0.8 (port0/line8)
9	PFI 9/P0.9 (port0/line9)
10	D GND
11	PFI 10/P0.10 (port0/line10)
12	PFI 11/P0.11 (port0/line11)
13	D GND
14	PFI 12/P0.12 (port0/line12)
15	PFI 13/P0.13 (port0/line13)
16	D GND
17	PFI 14/P0.14 (port0/line14)
18	PFI 15/P0.15 (port0/line15)
19	ID 1

Pin	Signal
20	AO 0
21	AO 1
22	No connect
23	No connect
24	+5 V
25	PFI 0/P0.0 (port0/line0)
26	PFI 1/P0.1 (port0/line1)
27	D GND
28	PFI 2/P0.2 (port0/line2)
29	PFI 3/P0.3 (port0/line3)
30	D GND
31	PFI 4/P0.4 (port0/line4)
32	PFI 5/P0.5 (port0/line5)
33	D GND
34	PFI 6/P0.6 (port0/line6)
35	PFI 7/P0.7 (port0/line7)
36	D GND

Table 11. USB-6421 AO/DIO Connector Signal Descriptions

Signal	Function	Reference	Direction	Description
AO <0..1>	Analog output channels	AO GND	Output	Supplies the voltage output of the AO channels.
AO GND	Analog output ground	—	—	AO GND is the reference for the AO channels. AI GND, AO GND, D GND, and CHSGND are all

Signal	Function	Reference	Direction	Description
				connected internally.
+5 V	+5 V power source	D GND	Output	Provides current limited +5 V power output that can be used to power external circuitry. Refer to the ***+5 V Power Source*** section for more information. Leave this pin open if you do not use it.
PFI <0..15>/P0.<0..15>	Port 0 digital I/O channels	D GND	Input or output	Digital channels that can be individually configured as input or output. These channels are referred to as port0/line0:15 in software when used as digital I/O. They are referred to as PFI 0:15 when used for other purposes, like timing I/O. Can also be individually configured for the following uses. • Digital I/O

Signal	Function	Reference	Direction	Description
				• Counter/ timer input • Counter/ timer output • External timing or trigger signal input for AI, AO, DI, DO, counter, or timers. • Timing or trigger signal output from AI, AO, DI, DO, counter, or timers.
D GND	Digital ground	—	—	Supplies the reference for the P0.<0..15> pins and +5 V pin. AI GND, AO GND, D GND, and CHSGND are all connected internally.
CHSGND	Chassis ground	—	—	Connects directly to the chassis ground of the USB-6421 enclosure. It can be used as a termination point for shielded cables to help improve measurement

Signal	Function	Reference	Direction	Description
				quality.
ID 1	—	—	—	This feature is not supported yet.

Related reference:

- +5 V Power Source

+5 V Power Source

The +5 V terminals on the I/O connector supply +5 V referenced to D GND. Use these terminals to power external circuitry.

Notice Never connect the +5 V power terminals to analog or digital ground or to any other voltage source on the USB-6421 or any other device. Doing so can damage the device and the computer. NI is not liable for damage resulting from such a connection.

Refer to the ***USB-6421 Specifications*** for more information about the USB-6421 power rating.

Related information:

- USB-6421 Specifications

Analog Input Measurements

Learn about making analog input measurements with the USB-6421.

Analog Input Range

You can individually program the input range of each AI channel on your USB-6421.

Input range refers to the set of input voltages that an analog input channel can digitize with the specified accuracy. The NI programmable gain instrumentation amplifier (NI-PGIA) amplifies or attenuates the AI signal depending on the input range.

The input range affects the resolution of the USB-6421 for an AI channel. ***Resolution*** refers to the voltage of one ADC code. For example, a 16-bit analog-to-digital converter (ADC) converts analog inputs into one of 65,536 (= 2^{16}) codes—that is, one of 65,536 possible digital values. These values are spread fairly evenly across the input range. So, for an input range of -10 V to 10 V, the voltage of each code of a 16-bit ADC is:

$$\frac{10\ V - (-10\ V)}{2^{16}} = 305\ \mu V$$

The USB-6421 uses a calibration method that requires some codes to lie outside of the specified range. This calibration method improves absolute accuracy, but it increases the nominal resolution of input ranges by about 5% over what the formula shown above indicates.

Choose an input range that matches the expected input range of your signal. A large input range can accommodate a large signal variation, but reduces the voltage resolution. Choosing a smaller input range improves the voltage resolution, but may result in the input signal going out of range.

Table 12. USB-6421 Input Range and Nominal Resolution

Input Range (V)	Nominal Resolution (µV)
10	329.14
5	164.24

Input Range (V)	Nominal Resolution (µV)
1.0	32.91
0.2	6.58

Working Voltage Range

The NI programmable gain instrumentation amplifier (NI-PGIA) operates normally by amplifying signals of interest while rejecting common-mode signals under certain conditions.

- **Common-mode voltage (*Vcm*)**—The voltage of measurement reference potential versus AI GND. It depends on the configuration of the input terminal.
- **Signal voltage (*Vs*)**—The value you are trying to measure. It depends on the configuration of the input terminal.
- **Total working voltage of the positive input**—Equivalent to (***Vcm*** + ***Vs***), or subtracting AI GND from AI <0..x>+. Must be less than the maximum working range specified in the ***USB-6421 Specifications***.

If any of these conditions are exceeded, the input voltage is clamped until the fault condition is removed.

Table 13. Common-Mode Voltage, ***Vcm***

Input Terminal Configuration	Condition
Differential Mode	***Vcm*** is equal to the voltage on AI<x>- vs. AI GND, where AI<x>- is the negative half of the differential pair. Refer to the ***Differential Connections for Ground-Referenced Signal Sources*** section.
Referenced Single-Ended (RSE) Mode	***Vcm*** is equal to 0 V since AI GND is the reference for the measurement. Refer to the ***Non-Referenced Single-Ended (NRSE) Connections for Ground-Referenced Signal Sources*** section.
Non-referenced Single-Ended (NRSE) Mode	***Vcm*** is equal to the voltage on the NR pin vs. AI

Input Terminal Configuration	Condition
	GND.

Table 14. Signal Voltage, ***Vs***

Input Terminal Configuration	Condition
Differential Mode	***Vs*** is equal to AI<x>+ minus AI<x>-, where AI<x>- is the negative half of the differential pair.
RSE Mode	***Vs*** is equal to AI<x> minus AI GND.
NRSE Mode	***Vs*** is equal to AI<x> minus the voltage on the NR pin.

Related concepts:

- Differential Connections for Ground-Referenced Signal Sources
- Non-Referenced Single-Ended (NRSE) Connections for Ground-Referenced Signal Sources

Analog Input Terminal Configurations

The USB-6421 supports three analog input terminal configurations: differential mode, referenced single-ended mode, and non-referenced single-ended mode.

- **Differential mode**—The USB-6421 measures the difference in voltage between two AI signals.
- **Referenced single-ended (RSE) mode**—The USB-6421 measures the voltage of an AI signal relative to AI GND.
- **Non-referenced single-ended (NRSE) mode**—The USB-6421 measures the voltage of an AI signal relative to the NR pin (AI SENSE).

The AI terminal configuration determines the ground reference for your measurement and affects how you should connect your AI signals to the USB-6421.

Ground-reference settings are programmed on a per-channel basis. For example, you might configure the USB-6421 to scan 12 channels—four differentially-configured channels and eight single-ended channels.

The following table shows how signals are routed to the NI programmable gain instrumentation amplifier (NI-PGIA) on the USB-6421.

Table 15. Signals Routed to the NI-PGIA on the USB-6421

AI Ground-Reference Settings	Signals Routed to the Positive Input of the NI-PGIA (Vin+)	Signals Routed to the Negative Input of the NI-PGIA (Vin-)
RSE	AI <0..15>	AI GND
NRSE	AI <0..15>	NR (AI SENSE)
Differential	AI <0..7>	AI <8..15>

Note Other mioDAQ products have restrictions on mixing referenced single-ended, non-referenced single-ended, and differential channel types in the same scan. Refer to the ***Designing for Migration*** section for more details.

For differential measurements, AI 0 and AI 8 are the positive and negative inputs of differential analog input channel 0.

Notice The maximum input voltages rating of AI signals with respect to ground (and for signal pairs in differential mode with respect to each other) are listed in the ***USB-6421 Specifications***. Exceeding the maximum input voltage of AI signals distorts the measurement results. Exceeding the maximum input voltage rating can also damage the product and the computer. NI is not liable for any damage resulting from such signal connections. AI ground-reference setting is sometimes referred to as ***AI terminal configuration.***

Related concepts:

- Designing for Migration

Related information:

- USB-6421 Specifications

USB-6421 Differential Mode Pairs

- AI <0, 8>
- AI <1, 9>
- AI <2, 10>
- AI <3, 11>
- AI <4, 12>
- AI <5, 13>
- AI <6, 14>
- AI <7, 15>

Multi-channel Scanning Considerations

The USB-6421 can scan multiple channels at high rates and digitize the signals accurately.

Settling time affects accuracy in dual scanning applications. ***Settling time*** refers to the time it takes the NI-PGIA to amplify the input signal to the desired accuracy before it is sampled by the ADC.

When the USB-6421 switches from one AI channel to another AI channel, the USB-6421 configures the NI programmable gain instrumentation amplifier (NI-PGIA) at the same range only. The NI-PGIA then amplifies the input signal with the same gain.

The USB-6421 is designed to have fast settling times. However, several factors can increase the settling time, which decreases the accuracy of your measurements. To ensure fast settling times, do the following (in order of importance):

1. Use low-impedance sources.
2. Use short, high-quality cabling.
3. Carefully choose the channel scanning order.
4. Avoid scanning faster than necessary.

Use Low-Impedance Sources

Your signal sources should have an impedance of <1 kΩ to ensure fast settling times. Large source impedances increase the settling time of the NI-PGIA, and so decrease the accuracy at fast scanning rates.

Settling times increase when scanning high-impedance signals due to a phenomenon called ***charge injection***. Multiplexers contain switches, usually made of switched capacitors. When one of the channels, for example channel 0, is selected in a multiplexer, those capacitors accumulate charge. When the next channel, for example channel 1, is selected, the accumulated charge leaks backward through channel 1. If the output impedance of the source connected to channel 1 is high enough, the resulting reading of channel 1 can be partially affected by the voltage on channel 0. This effect is referred to as ***ghosting***.

If your source impedance is high, you can decrease the scan rate to allow the NI-PGIA more time to settle. Another option is to use a voltage follower circuit external to the USB-6421 to decrease the impedance seen by the USB-6421.

Related information:

- Eliminate Ghosting on Adjacent Input Channels by Decreasing Source Impedance

Use Short, High-Quality Cabling

Using short, high-quality cables can minimize several effects that degrade accuracy including crosstalk, transmission line effects, and noise.

The capacitance of the cable can also increase the settling time. NI recommends using individually shielded, twisted-pair wires that are 2 m or less to connect AI signals to the USB-6421.

Carefully Choose the Channel Scanning Order

Avoid Switching from a Large to a Small Input Range

Switching from a channel with a large input range to a channel with a small input range can greatly increase the settling time.

Suppose a 4 V signal is connected to channel 0 and a 1 mV signal is connected to channel 1. The input range for channel 0 is -10 V to 10 V and the input range of channel 1 is -200 mV to 200 mV.

When the multiplexer switches from channel 0 to channel 1, the input to the NI-PGIA switches from 4 V to 1 mV. The approximately 4 V step from 4 V to 1 mV is 1,000% of the new full-scale range. For a 16-bit device to settle within 0.0015% (15 ppm or 1 LSB) of the ±200 mV full-scale range on channel 1, the input circuitry must settle to within 0.000031% (0.31 ppm or 1/50 LSB) of the ±10 V range. Some devices can take many microseconds for the circuitry to settle this much.

To avoid this effect, you should arrange your channel scanning order so that transitions from large to small input ranges are infrequent.

In general, you do not need this extra settling time when the NI-PGIA is switching from a small input range to a larger input range.

Insert Grounded Channel between Signal Channels.

Another technique to improve settling time is to connect an input channel to ground. Then insert this channel in the scan list between two of your signal channels. The input range of the grounded channel should match the input range of the signal after the grounded channel in the scan list.

Consider again the example above where a 4 V signal is connected to channel 0 and a 1 mV signal is connected to channel 1. Suppose the input range for channel 0 is -10 V to 10 V and the input range of channel 1 is -200 mV to 200 mV.

You can connect channel 2 to AI GND (or you can use the internal ground; refer to ***Internal Channels*** below). Set the input range of channel 2 to -200 mV to 200 mV to match channel 1. Then scan channels in the order: 0, 2, 1.

Inserting a grounded channel between signal channels improves settling time because the NI-PGIA adjusts to the new input range setting faster when the input is grounded.

Minimize Voltage Step between Adjacent Channels

When scanning between channels that have the same input range, the settling time increases with the voltage step between the channels. If you know the expected input range of your signals, you can group signals with similar expected ranges together in your scan list.

For example, suppose all channels in a system use a -5 V to 5 V input range. The signals on channels 0, 2, and 4 vary between 4.3 V and 5 V. The signals on channels 1, 3, and 5 vary between -4 V and 0 V. Scanning channels in the order 0, 2, 4, 1, 3, 5 produces more accurate results than scanning channels in the order 0, 1, 2, 3, 4, 5.

Related information:

- Internal Channels

Avoid Scanning Faster Than Necessary

Designing your system to scan at slower speeds gives the NI programmable gain instrumentation amplifier (NI-PGIA) more time to settle to a more accurate level.

Example 1

Averaging many AI samples can increase the accuracy of the reading by decreasing noise effects. In general, the more points you average, the more accurate the final result. However, you may choose to decrease the number of points you average and slow down the scanning rate.

Suppose you want to sample 10 channels over a period of 20 ms and average the results. You could acquire 250 points from each channel at a scan rate of 125 kS/s. Another method would be to acquire 500 points from each channel at a scan rate of 250 kS/s. Both methods take the same amount of time. Doubling the number of samples averaged (from 250 to 500) decreases the effect of noise by a factor of 1.4 (the square root of 2). However, doubling the number of samples (in this example) decreases the time the NI-PGIA has to settle from 4 µs to 2 µs. In some cases, the slower scan rate system returns more accurate results.

Example 2

If the time relationship between channels is not critical, you can sample from the same channel multiple times and scan less frequently. For example, suppose an application requires averaging 100 points from channel 0 and averaging 100 points from channel 1. You could alternate reading between channels—that is, read one point from channel 0, then one point from channel 1, and so on. You also could read all 100 points from channel 0 then read 100 points from channel 1. The second method switches between channels much less often and is affected much less by settling time.

Analog Input Data Acquisition Methods

When performing analog input measurements, you either can perform software-timed or hardware-timed acquisitions.

Software-Timed Acquisitions

With a software-timed acquisition, software controls the rate of the acquisition.

Software sends a separate command to the hardware to initiate each ADC conversion. In NI-DAQmx, software-timed acquisitions are referred to as having ***on-demand timing***. Software-timed acquisitions are also referred to as ***immediate*** or ***static*** acquisitions and are typically used for reading a single sample of data.

Hardware-Timed Acquisitions

With hardware-timed acquisitions, a digital hardware signal controls the rate of the acquisition. This signal can be generated internally on the USB-6421 or provided externally.

Hardware-timed acquisitions have several advantages over software-timed acquisitions:

- The time between samples can be much shorter.
- The timing between samples is deterministic.
- Hardware-timed acquisitions can use hardware triggering.

Hardware-timed operations are buffered. In a buffered acquisition, data is moved efficiently from the onboard FIFO memory of the USB-6421 to a PC buffer using USB signal streaming before it is transferred to application memory. Buffered acquisitions typically allow for much faster transfer rates than non-buffered acquisitions because data is moved in large blocks, rather than one point at a time.

One property of buffered I/O operations is the sample mode. The sample mode can be either ***finite*** or ***continuous***.

- **Finite sample mode**—Acquires a specific, predetermined number of data samples. After the specified number of samples has been read in, the acquisition stops.
- **Continuous sample mode**—Acquires an unspecified number of samples. Instead of acquiring a set number of data samples and stopping, a continuous acquisition continues until you stop the operation.

If data cannot be transferred across the bus fast enough, or if the user program does not read data out of the PC buffer fast enough to keep up with the data transfer, the buffer could reach an overflow condition, causing an error to be generated.

Note The USB-6421 does not support hardware-timed single-point mode.

Analog Input Signal Connections

You can connect floating signal sources and ground-referenced signal sources to the USB-6421.

Floating Signal Sources (Not Connected to Building Ground)

Examples of floating signal sources include ungrounded thermocouples, signal conditioning with isolated outputs, and battery devices.

Figure 65. Differential Mode Floating Signal Sources Connections

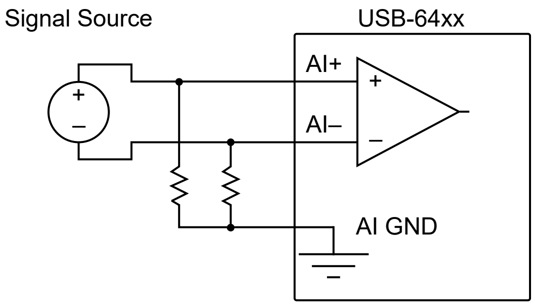


Figure 10. Non-Referenced Single-Ended (NRSE) Mode Floating Signal Sources Connections

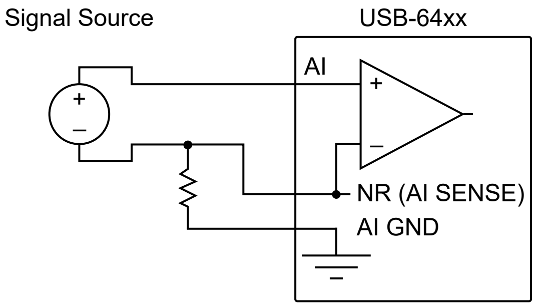
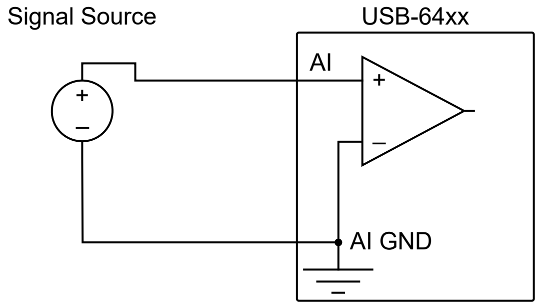


Figure 11. Referenced Single-Ended (RSE) Mode Floating Signal Sources Connections



Ground-Referenced Signal Sources

Examples of ground-references signal sources include plug-in instruments with non-isolated outputs.

Figure 12. DIFF Mode Ground-Referenced Signal Sources Connections

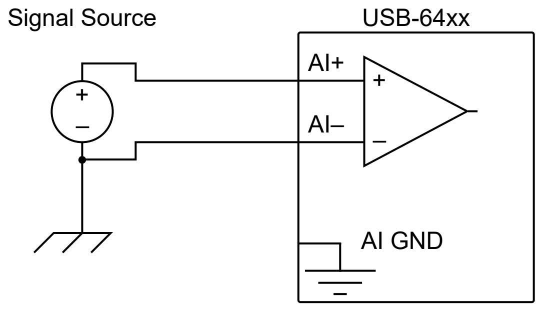
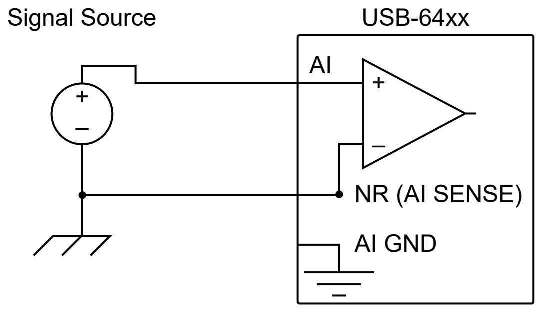
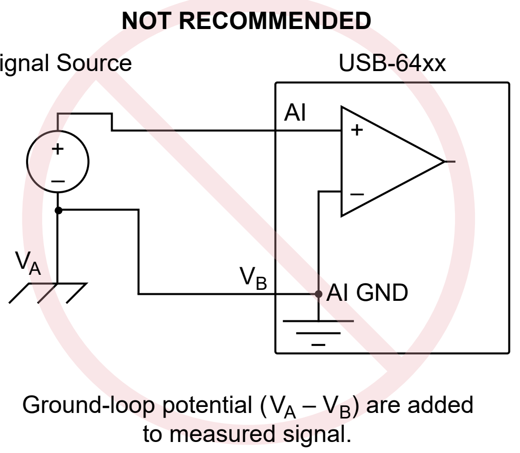


Figure 13. NRSE Mode Ground-Referenced Signal Sources Connections



NI does no recommend connecting ground-referenced signal sources in RSE mode.

Figure 14. RSE Mode Ground-Referenced Signal Sources Connections


NOT RECOMMENDED
Signal Source
USB-64xx
AI
+
–
+
–
V_A
V_B
AI GND
Ground-loop potential ($V_A – V_B$) are added to measured signal.

Refer to the following sections for more information on connecting signals to the USB-6421

Floating Signal Sources Connections

A ***floating signal source*** is a signal source that is not connected to the building ground system, but has an isolated ground-reference point.

Some examples of floating signal sources are outputs of transformers, thermocouples, battery-powered devices, optical isolators, and isolation amplifiers. An instrument or device that has an isolated output is a floating signal source.

Differential Connections for Floating Signal Sources

Differential signal connections reduce noise pickup and increase common-mode noise rejection. Differential signal connections also allow input signals to float within the common-mode limits of the NI programmable gain instrumentation amplifier (NI-PGIA).

Use differential input connections for any channel that meets any of the following conditions:

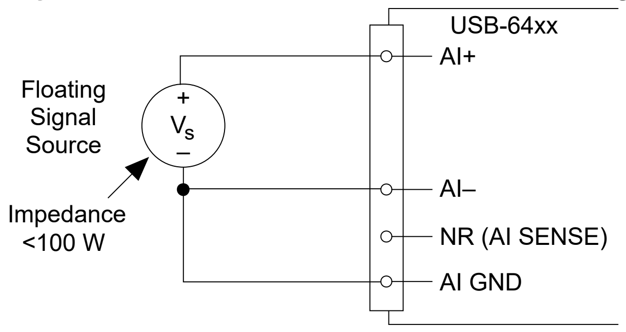
- The input signal is low level (less than 1 V).
- The leads connecting the signal to the USB-6421 are greater than 3 m (10 ft).
- The input signal requires a separate ground-reference point or return signal.
- The signal leads travel through noisy environments.

- Two analog input channels, AI+ and AI-, are available for the signal.

It is important to connect the negative lead of a floating source to AI GND (either directly or through a bias resistor). Otherwise, the source may float out of the maximum working voltage range of the NI-PGIA and the USB-6421 returns erroneous data.

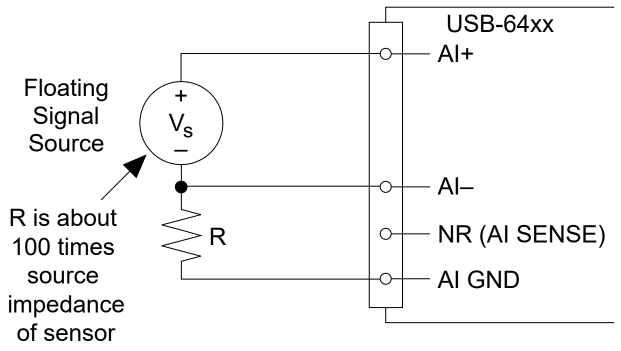
The easiest way to reference the source to AI GND is to connect the positive side of the signal to AI+ and connect the negative side of the signal to AI GND as well as to AI- without using resistors. This connection works well for DC-coupled sources with low source impedance (less than 100 Ω).

Figure 15. Differential Connections for Floating Signal Sources without Bias Resistors



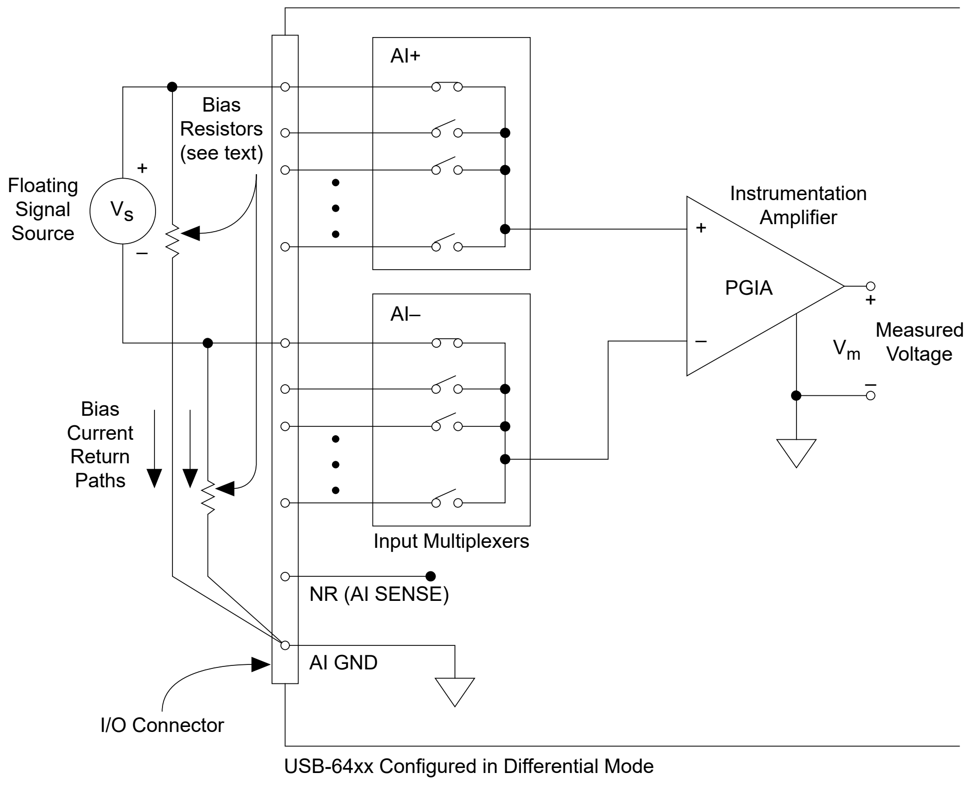
However, for larger source impedances, this connection leaves the differential signal path significantly off balance. Noise that couples electrostatically onto the positive line does not couple onto the negative line because it is connected to ground. This noise appears as a differential mode signal instead of a common-mode signal, and thus appears in your data. In this case, instead of directly connecting the negative line to AI GND, connect the negative line to AI GND through a resistor that is about 100 times the equivalent source impedance. The resistor puts the signal path nearly in balance, so that about the same amount of noise couples onto both connections, yielding better rejection of electrostatically coupled noise. This configuration does not load down the source (other than the very high input impedance of the NI-PGIA).

Figure 16. Differential Connections for Floating Signal Sources with Single Bias Resistor



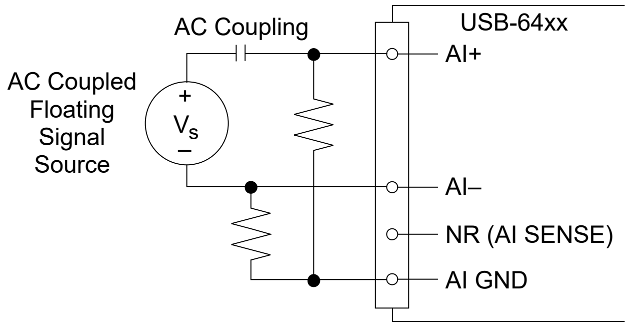
You can fully balance the signal path by connecting another resistor of the same value between the positive input and AI GND, as shown in the following figure. This fully balanced configuration offers slightly better noise rejection, but has the disadvantage of loading the source down with the series combination (sum) of the two resistors. If, for example, the source impedance is 2 kΩ and each of the two resistors is 100 kΩ, the resistors load down the source with 200 kΩ and produce a -1% gain error.

Figure 17. Differential Connections for Floating Signal Sources with Balanced Bias Resistors


AI+
Bias Resistors (see text)
Floating Signal Source
+
V_s
–
Instrumentation Amplifier
+
PGIA
–
+
V_m
Measured Voltage
–
AI–
Bias Current Return Paths
Input Multiplexers
NR (AI SENSE)
AI GND
I/O Connector
USB-64xx Configured in Differential Mode

Both inputs of the NI-PGIA require a DC path to ground in order for the NI-PGIA to work. If the source is AC coupled (capacitively coupled), the NI-PGIA needs a resistor between the positive input and AI GND. If the source has low-impedance, choose a resistor that is large enough not to significantly load the source, but small enough not to produce significant input offset voltage as a result of input bias current (typically 100 kΩ to 1 MΩ). In this case, connect the negative input directly to AI GND. If the source has high output impedance, balance the signal path as previously described using the same value resistor on both the positive and negative inputs; be aware that there is some gain error from loading down the source, as shown in the following figure.

Figure 18. Differential Connections for AC Coupled Floating Sources with Balanced Bias Resistors



Non-Referenced Single-Ended (NRSE) Connections for Floating Signal Sources

Only use NRSE input connections if the input signal meets the following conditions:

- The input signal is high-level (greater than 1 V).
- The leads connecting the signal to the USB-6421 are less than 3 m (10 ft).

Differential input connections are recommended for greater signal integrity for any input signal that does not meet the preceding conditions.

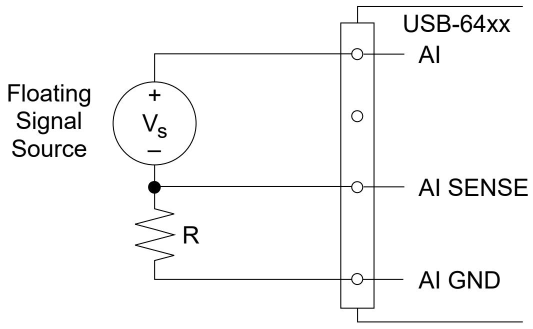
In the single-ended modes, more electrostatic and magnetic noise couples into the signal connections than in differential configurations. The coupling is the result of differences in the signal path. Magnetic coupling is proportional to the area between the two signal conductors. Electrical coupling is a function of how much the electric field differs between the two conductors.

With this type of connection, the NI programmable gain instrumentation amplifier (NI-PGIA) rejects both the common-mode noise in the signal and the ground potential difference between the signal source and the device ground.

It is important to connect the negative lead of a floating signals source to AI GND (either directly or through a resistor). Otherwise the source may float out of the valid input range of the NI-PGIA and the USB-6421 returns erroneous data.

The following figure shows a floating source connected to the USB-6421 in NRSE mode.

Figure 62. NRSE Connections for Floating Signal Sources



All of the bias resistor configurations discussed in the ***Differential Connections for Floating Signal Sources*** section apply to the NRSE bias resistors as well. Replace AI- with AI SENSE in the figures included in that section for configurations with zero to two bias resistors. The noise rejection of NRSE mode is better than RSE mode because the AI SENSE connection is made remotely near the source. However, the noise rejection of NRSE mode is worse than differential mode because the AI SENSE connection is shared with all channels rather than being cabled in a twisted pair with the AI+ signal.

You can use the DAQ Assistant to configure the channels for RSE or NRSE input modes.

Referenced Single-Ended (RSE) Connections for Floating Signal Sources

Only use RSE input connections if the input signal meets the following conditions:

- The input signal can share a common reference point, AI GND, with other signals that use RSE.
- The input signal is high-level (greater than 1 V).
- The leads connecting the signal to the USB-6421 are less than 3 m (10 ft).

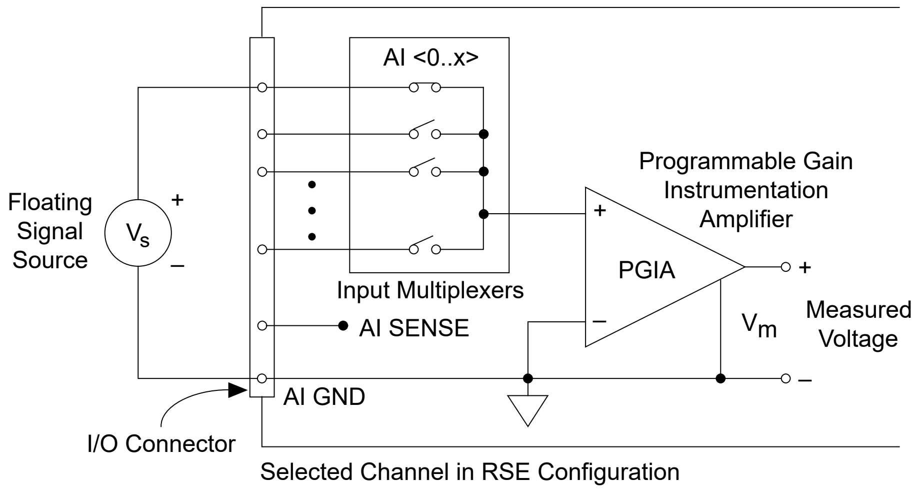
Differential input connections are recommended for greater signal integrity for any input signal that does not meet the preceding conditions.

In the single-ended modes, more electrostatic and magnetic noise couples into the signal connections than in differential configurations. The coupling is the result of differences in the signal path. Magnetic coupling is proportional to the area between the two signal conductors. Electrical coupling is a function of how much the electric field differs between the two conductors.

With this type of connection, the NI programmable gain instrumentation amplifier (NI-PGIA) rejects both the common-mode noise in the signal and the ground potential difference between the signal source and the device ground.

The following figure shows how to connect a floating signal source to the USB-6421 configured for RSE mode.

Figure 22. RSE Connections for Floating Signal Sources



You can use the DAQ Assistant to configure the channels for RSE or non-referenced single-ended (NRSE) input modes.

Ground-Referenced Signal Sources Connections

A ***ground-referenced signal source*** is a signal source connected to the building system ground.

A ground-referenced signal source is already connected to a common ground point with respect to the USB-6421, assuming that the computer is plugged into the same power system as the source. Non-isolated outputs of instruments and devices that plug into the building power system fall into this category.

The difference in ground potential between two instruments connected to the same building power system is typically between 1 and 100 mV, but the difference can be much higher if power distribution circuits are improperly connected. If a grounded signal source is incorrectly measured, this difference can appear as measurement error. Follow the connection instructions for grounded signal sources to eliminate this ground potential difference from the measured signal.

Differential Connections for Ground-Referenced Signal Sources

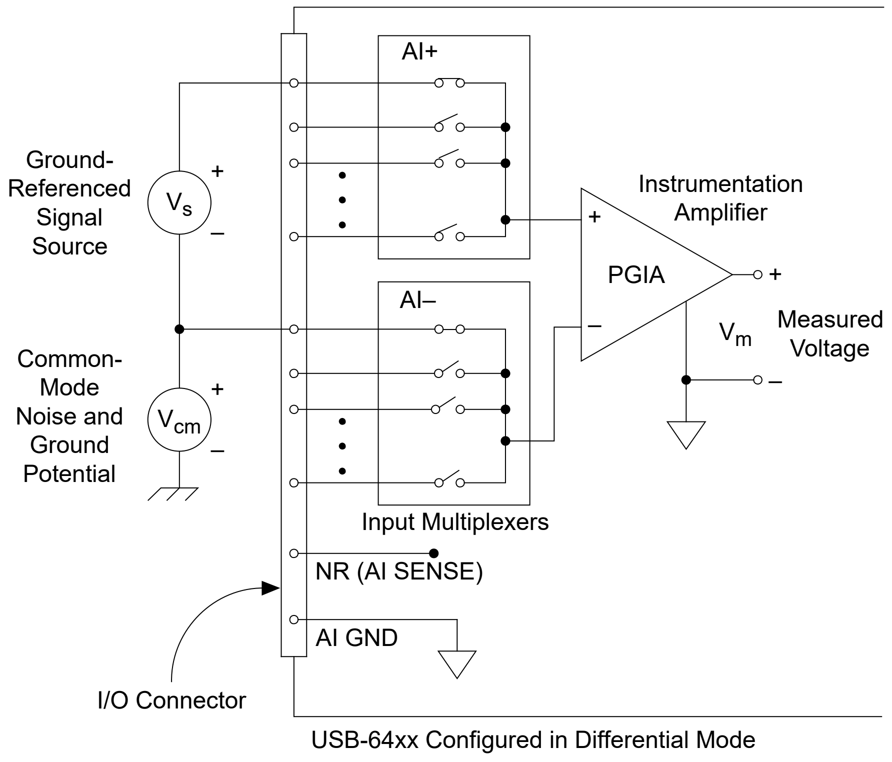
Differential signal connections reduce noise pickup and increase common-mode noise rejection. Differential signal connections also allow input signals to float within the common-mode limits of the NI-PGIA.

Use differential input connections for any channel that meets any of the following conditions.

- The input signal is low level (less than 1 V).
- The leads connecting the signal to the USB-6421 are greater than 3 m (10 ft).
- The input signal requires a separate ground-reference point or return signal.
- The signal leads travel through noisy environments.
- Two analog input channels, AI+ and AI-, are available for the signal.

The following figure shows how to connect a ground-referenced signal source to the USB-6421 configured in differential mode.

Figure 36. Differential Connections for Ground-Referenced Signal Sources


AI+
Ground-Referenced Signal Source
V_s
+
–
Instrumentation Amplifier
PGIA
+
–
V_m
Measured Voltage
AI–
Common-Mode Noise and Ground Potential
V_{cm}
Input Multiplexers
NR (AI SENSE)
AI GND
I/O Connector
USB-64xx Configured in Differential Mode

With this type of connection, the NI-PGIA rejects both the common-mode noise in the signal and the ground potential difference between the signal source and the device ground, shown as Vcm in the figure.

AI+ and AI- must both remain less than the maximum working range specified in the ***USB-6421 Specifications*** .

Non-Referenced Single-Ended (NRSE) Connections for Ground-Referenced Signal Sources

Only use non-referenced signal-ended connections if the input signal meets the following conditions:

- The input signal is high-level (greater than 1 V).
- The leads connecting the signal to the USB-6421 are less than 3 m (10 ft).
- The input signal can share a common reference point with other signals.

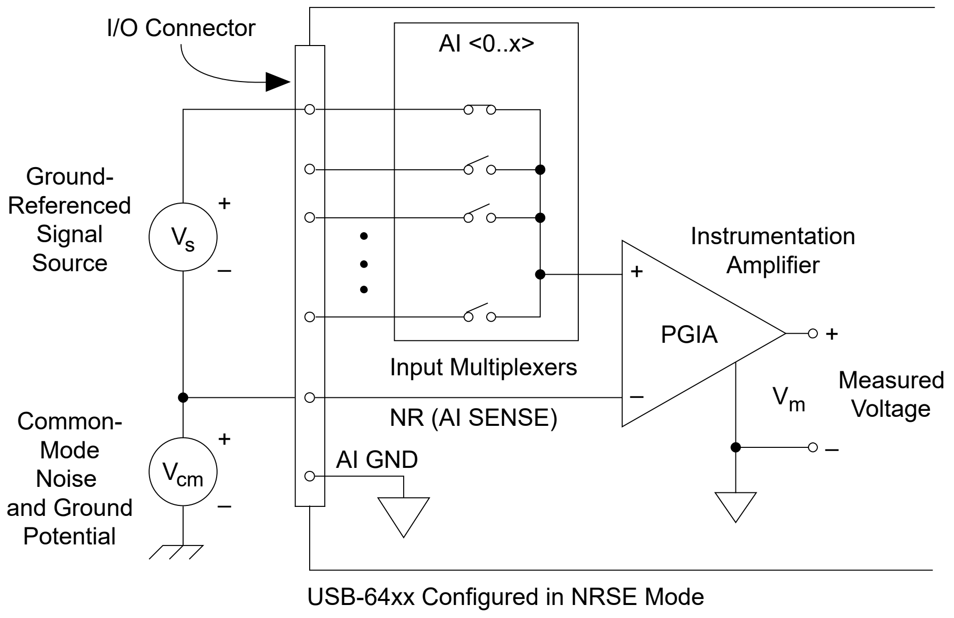
Differential input connections are recommended for greater signal integrity for any input signal that does not meet the preceding conditions.

In the single-ended modes, more electrostatic and magnetic noise couples into the signal connections than in differential configurations. The coupling is the result of differences in the signal path. Magnetic coupling is proportional to the area between the two signal conductors. Electrical coupling is a function of how much the electric field differs between the two conductors.

With this type of connection, the NI Programmable Gain Instrumentation Amplifier (NI-PGIA) rejects both the common-mode noise in the signal and the ground potential difference between the signal source and the device ground.

The following figure shows how to connect ground-reference signal sources in NRSE mode.

Figure 22. Single-Ended Connections for Ground-Referenced Signal Sources (NRSE Configuration)


I/O Connector
AI <0..x>
Ground-Referenced Signal Source
V_s
+
–
Instrumentation Amplifier
+
PGIA
–
+
Input Multiplexers
V_m
Measured Voltage
NR (AI SENSE)
Common-Mode Noise and Ground Potential
V_{cm}
AI GND
USB-64xx Configured in NRSE Mode

AI<0..15> and AI SENSE must both remain less than the maximum working range specified in the ***USB-6421 Specifications*** .

In this mode, NR (AI SENSE) is internally connected to the negative input of the NI-PGIA. Therefore, the ground point of the signal connects to the negative input of the NI-PGIA.

Any potential difference between the device ground and the signal ground appears as a common-mode signal at both the positive and negative inputs of the NI-PGIA, and this difference is rejected by the amplifier. If the input circuitry of a device were referenced to ground, as it is in the RSE ground-reference setting, this difference in ground potentials would appear as an error in the measured voltage.

Using the DAQ Assistant, you can configure the channels for RSE or NRSE input modes. Refer to the ***Configuring AI Ground-Reference Settings in Software*** topic for more information about the DAQ Assistant.

Referenced Single-Ended (RSE) Connections with Ground-Referenced Signal Sources

Do not use RSE connections with ground-referenced signal sources. Use non-referenced single-ended or differential connections instead.

As illustrated in ***Analog Input Signal Connections***, there can be a potential difference between AI GND and the ground of the sensor. In RSE mode, this ground loop causes measurement errors.

Related reference:

- Analog Input Signal Connections

Field Wiring Considerations for Analog Input Signals

Environmental noise can seriously affect the measurement accuracy of the USB-6421 if you do not take proper care when running signal wires between signal sources and the USB-6421.

The following recommendations apply mainly to AI signal routing to the USB-6421, although they also apply to signal routing in general.

Minimize noise pickup and maximize measurement accuracy by taking the following precautions:

- Use differential analog input connections to reject common-mode noise.
- Use individually shielded, twisted-pair wires to connect AI signals to the USB-6421. With this type of wire, the signals attached to the positive and negative input channels are twisted together and then covered with a shield. You then connect

this shield only at one point to the signal source ground. This kind of connection is required for signals traveling through areas with large magnetic fields or high electromagnetic interference.

Related information:

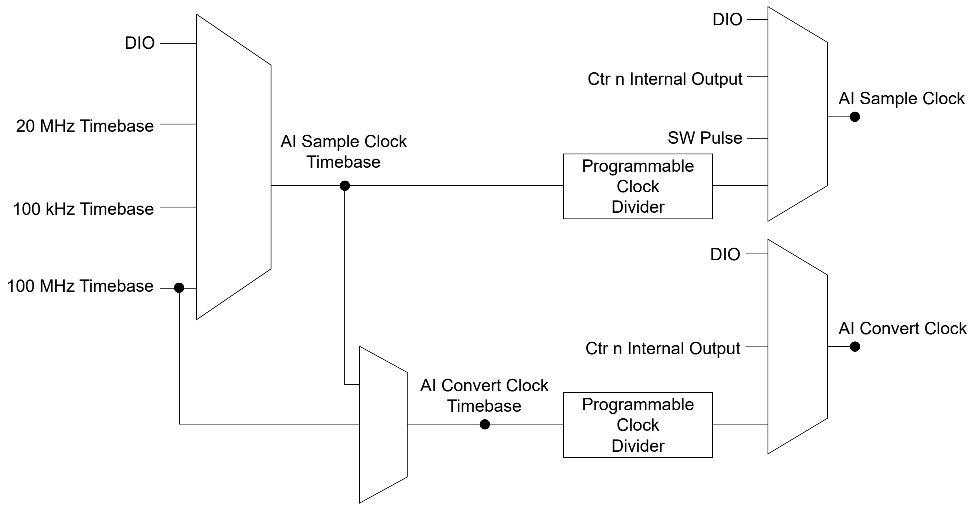
- Field Wiring and Noise Considerations for Analog Signals

Analog Input Timing Engine

The USB-6421 has a flexible timing engine.

The following figure summarizes all of the timing options provided by the analog input timing engine.

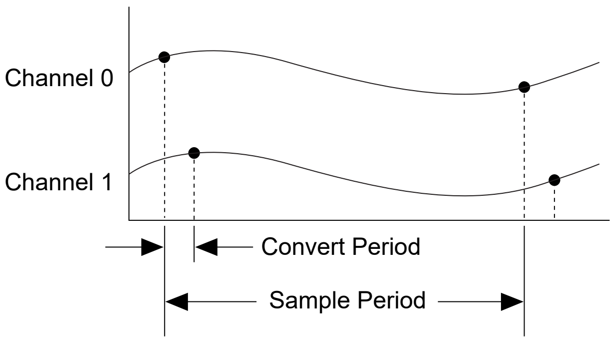
Figure 48. USB-6421 Analog Input Timing Options



The USB-6421 uses AI Sample Clock (ai/SampleClock) and AI Convert Clock (ai/ConvertClock) to perform interval sampling. As the following figure shows, AI Sample Clock controls the sample period, which is determined by the following equation:

$1 / \mathit{SamplePeriod} = \mathit{SampleRate}$

Figure 24. Interval Sampling

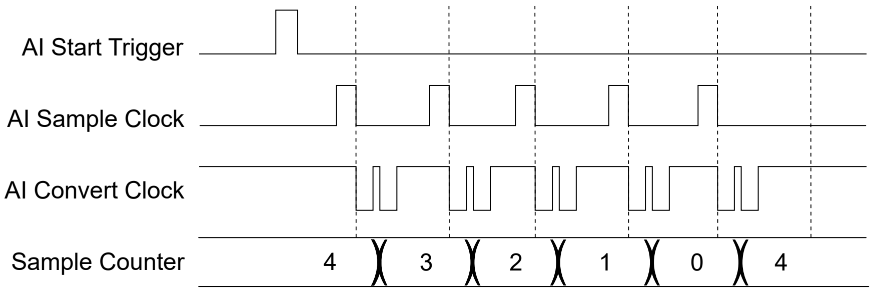


AI Convert Clock controls the Convert Period, which is determined by the following equation:

$1 / \mathit{ConvertPeriod} = \mathit{ConvertRate}$

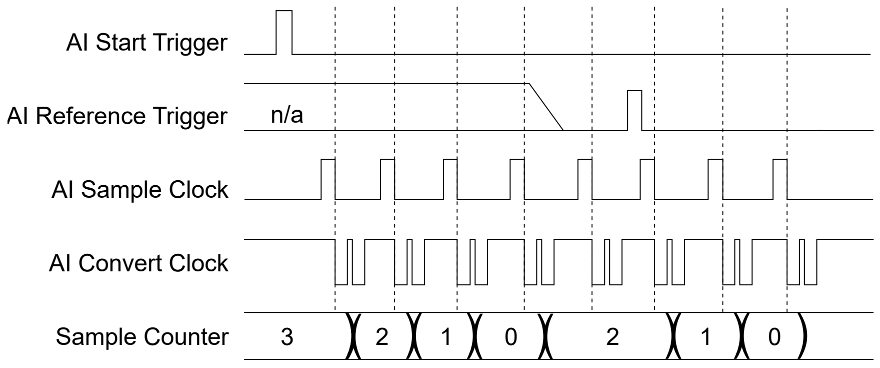
Post-triggered data acquisition allows you to view only data that is acquired after a trigger event is received. A typical post-triggered DAQ sequence is shown in the following figure. The sample counter is loaded with the specified number of post-trigger samples, in this example, five. The value decrements with each pulse on AI Sample Clock until all desired samples have been acquired.

Figure 25. Post-triggered Data Acquisition Example



Pre-triggered data acquisition allows you to view data that is acquired before the trigger of interest, in addition to data acquired after the trigger. The following figure shows a typical pre-triggered DAQ sequence. AI Start Trigger (ai/StartTrigger) can be either a hardware or software signal. If AI Start Trigger is set up to be a software start trigger, an output pulse appears on the ai/StartTrigger line when the acquisition begins. When the AI Start Trigger pulse occurs, the sample counter is loaded with the number of pre-triggered samples, in this example, four. The value decrements with each pulse on AI Sample Clock. The sample counter is then loaded with the number of post-triggered samples, in this example, three.

Figure 26. Pre-triggered Data Acquisition Example



Analog Input Timing Signals

The USB-6421 features six analog input timing signals.

- AI Sample Clock Signal
- AI Sample Clock Timebase Signal
- AI Convert Clock Signal
- AI Convert Clock Timebase Signal
- AI Hold Complete Event Signal
- AI Start Trigger Signal

AI Sample Clock Signal

Use the AI Sample Clock (ai/SampleClock) signal to initiate a set of measurements.

The USB-6421 samples the AI signals of every channel in the task once for every AI Sample Clock. A measurement acquisition consists of one or more samples.

You can specify an internal or external source for AI Sample Clock. You can also specify whether the measurement sample begins on the rising edge or falling edge of AI Sample Clock.

Using an Internal Source

One of the following internal signals can drive AI Sample Clock:

- Counter ***n*** Internal Output
- AI Sample Clock Timebase (divided down)

- A pulse initiated by host software
- Change Detection Event
- Counter ***n*** Sample Clock
- AO Sample Clock (ao/SampleClock)
- DI Sample Clock (di/SampleClock)
- DO Sample Clock (do/SampleClock)

A programmable internal counter divides down the sample clock timebase.

Several other internal signals can be routed to AI Sample Clock through internal routes.

Using an External Source

Use DIO <0..15> as the source of AI Sample Clock.

Routing AI Sample Clock Signal to an Output Terminal

You can route AI Sample Clock out to any DIO <0..15> terminal. This pulse is always active high.

Other Timing Requirements

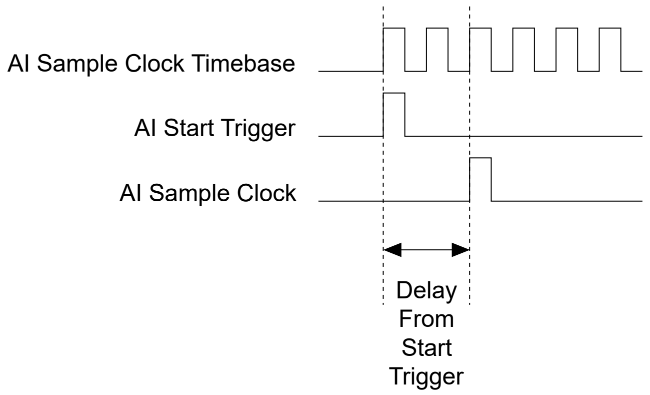
The USB-6421 only acquires data during an acquisition. It ignores AI Sample Clock when a measurement acquisition is not in progress.

A counter/timing engine on the USB-6421 internally generates AI Sample Clock unless you select some external source. AI Start Trigger starts this counter and either software or hardware can stop it once a finite acquisition completes. When using the AI timing engine, you can also specify a configurable delay from AI Start Trigger to the first AI Sample Clock pulse. By default, this delay is set to four ticks of the AI Sample Clock Timebase signal.

When using an externally generated AI Sample Clock, you must ensure the clock signal is consistent with respect to the timing requirements of AI Convert Clock. Failure to do so may result in a scan overrun and will cause an error. Refer to the ***AI Convert Clock Signal*** topic for more information about the timing requirements between AI Convert Clock and AI Sample Clock. The following figure shows the relationship of AI Sample

Clock to AI Start Trigger.

Figure 27. AI Sample Clock and AI Start Trigger



AI Sample Clock Timebase Signal

Use the AI Sample Clock Timebase (ai/SampleClockTimebase) signal to specify a higher frequency timebase that will be divided down to produce the AI Sample Clock.

You can route any of the following signals to be the AI Sample Clock Timebase (ai/ SampleClockTimebase) signal:

- 100 MHz Timebase (default)
- 20 MHz Timebase
- 100 kHz Timebase
- DIO <0..15>

AI Sample Clock Timebase is not available as an output on the I/O connector. AI Sample Clock Timebase is divided down to provide one of the possible sources for AI Sample Clock. You can configure the polarity selection for AI Sample Clock Timebase as either rising or falling edge, except on 100 MHz Timebase or 20 MHz Timebase.

AI Convert Clock Signal

Use the AI Convert Clock (ai/ConvertClock) signal to initiate a single A/D conversion on a single channel.

A sample (controlled by the AI Sample Clock) consists of one or more conversions. You can specify either an internal or external signal as the source of AI Convert Clock. You can also specify whether the measurement sample begins on the rising edge or falling edge of AI Convert Clock.

With NI-DAQmx, the driver chooses the fastest conversion rate possible based on the speed of the A/D converter and adds 10 µs of padding between each channel to allow for adequate settling time. This scheme enables the channels to approximate simultaneous sampling and still allow for adequate settling time. If the AI Sample Clock rate is too fast to allow for this 10 µs of padding, NI-DAQmx chooses the conversion rate so that the AI Convert Clock pulses are evenly spaced throughout the sample.

To explicitly specify the conversion rate, use AI Convert Clock Rate DAQmx Timing property node or function.

Notice Setting the conversion rate higher than the maximum rate specified for your device will result in errors.

Using an Internal Source

One of the following internal signals can drive AI Convert Clock:

- AI Convert Clock Timebase (divided down)
- Counter ***n*** Internal Output
- Change Detection Event
- Counter ***n*** Sample Clock
- AO Sample Clock (ao/SampleClock)
- DI Sample Clock (di/SampleClock)
- DO Sample Clock (do/SampleClock)

A programmable internal counter divides down the AI Convert Clock Timebase to generate AI Convert Clock. The counter is started by AI Sample Clock and continues to count down to zero, produces an AI Convert Clock, reloads itself, and repeats the process until the sample is finished. It then reloads itself in preparation for the next AI Sample Clock pulse.

Several other internal signals can be routed to AI Convert Clock through internal routes. Refer to ***Device Routing in MAX*** for more information.

Using an External Source

Use DIO <0..15> as the source of AI Convert Clock.

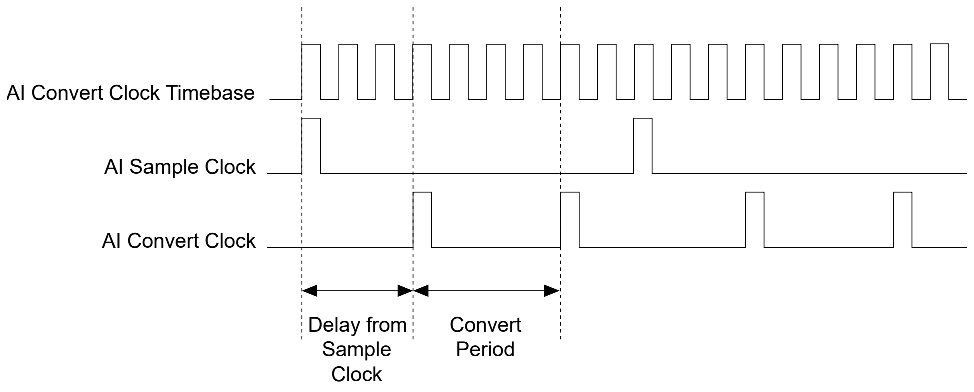
Routing AI Convert Clock Signal to an Output Terminal

You can route AI Convert Clock (as an active low signal) out to any DIO <0..15> terminal.

Using a Delay from Sample Clock to Convert Clock

When using the AI timing engine to generate your Convert Clock, you can also specify a configurable delay from AI Sample Clock to the first AI Convert Clock pulse within the sample. By default, this delay is three ticks of AI Convert Clock Timebase. The following figure shows the relationship of AI Sample Clock to AI Convert Clock.

Figure 28. AI Sample Clock and AI Convert Clock Relationship



Other Timing Requirements

Some clock signals are gated off unless the proper timing requirements are met because of how the sample and conversion level timing of the USB-6421 work.

For example, the USB-6421 ignores both AI Sample Clock and AI Convert Clock until it receives a valid AI Start Trigger signal. Similarly, it ignores all AI Convert Clock pulses until it recognizes an AI Sample Clock pulse. Once the USB-6421 receives the correct number of AI Convert Clock pulses, it ignores subsequent AI Convert Clock pulses until it receives another AI Sample Clock. However, after the USB-6421 recognizes an AI Sample Clock pulse, it causes an error if it receives an AI Sample Clock pulse before the correct number of AI Convert Clock pulses are received.

The following figures show timing sequences for a four-channel acquisition (using AI channels 0, 1, 2, and 3) and demonstrate proper and improper sequencing of AI Sample Clock and AI Convert Clock.

Figure 29. Scan Overrun Condition; AI Sample Clock Too Fast For Convert Clock Causes an Error

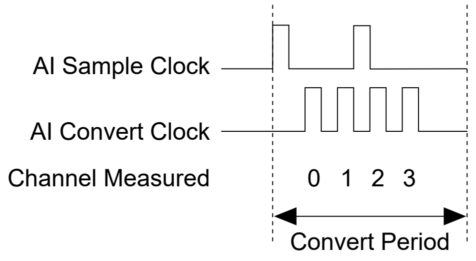


Figure 30. AI Convert Clock Too Fast For AI Sample Clock; AI Convert Clock Pulses Are Ignored

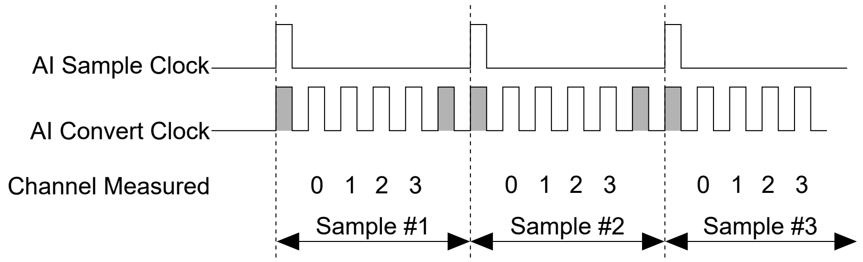


Figure 31. AI Sample Clock and AI Convert Clock Improperly Matched; Leads to Aperiodic Sampling

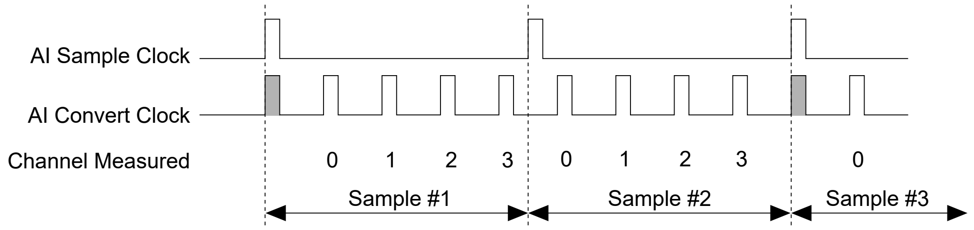
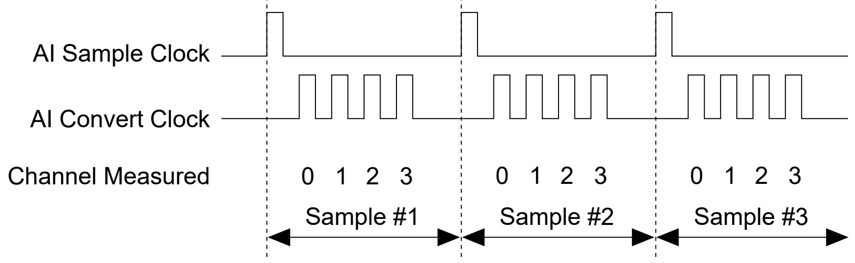


Figure 32. AI Sample Clock and AI Convert Clock Properly Matched



Related information:

- Device Routing in MAX

AI Convert Clock Timebase Signal

Use the AI Convert Clock Timebase (ai/ConvertClockTimebase) signal to specify a higher frequency timebase that will be divided down to produce the AI Convert Clock.

The AI Convert Clock Timebase (ai/ConvertClockTimebase) signal is divided down to provide one of the possible sources for AI Convert Clock. Use one of the following signals as the source of AI Convert Clock Timebase:

- AI Sample Clock Timebase
- 100 MHz Timebase

AI Convert Clock Timebase is not available as an output on the I/O connector.

AI Hold Complete Event Signal

Use the AI Hold Complete Event (ai/HoldCompleteEvent) signal to generate a pulse after each A/D conversion begins.

You can route AI Hold Complete Event out to any DIO <0..15>.

The polarity of AI Hold Complete Event is software-selectable, but it is typically configured so that a low-to-high leading edge can clock external AI multiplexers indicating when the input signal has been sampled and can be removed.

AI Start Trigger Signal

Use the AI Start Trigger (ai/StartTrigger) signal to begin a measurement acquisition.

A measurement acquisition consists of one or more samples. If you do not use triggers, begin a measurement with a software command. Once the acquisition begins, configure the acquisition to stop:

- When a certain number of points are sampled (in finite mode)
- With a software command (in continuous mode)

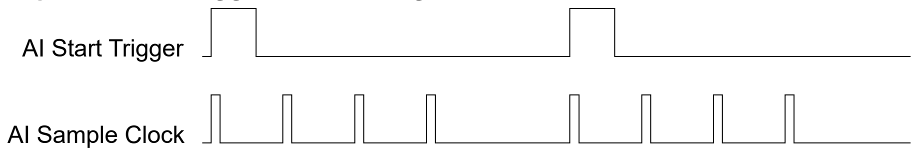
An acquisition that uses a start trigger is sometimes referred to as a ***post-triggered acquisition***.

Retriggerable Analog Input

The AI Start Trigger is configurable as retriggerable. The timing engine generates the sample and convert clocks for the configured acquisition in response to each pulse on an AI Start Trigger signal.

The timing engine ignores the AI Start Trigger signal while the clock generation is in progress. After the clock generation is finished, the counter waits for another Start Trigger to begin another clock generation. The following figure shows a retriggerable analog input with three AI channels and four samples per trigger.

Figure 33. Retriggerable Analog Input



Note Waveform information from LabVIEW does not reflect the delay between triggers. They are treated as a continuous acquisition with constant t0 and dt information.

Using a Digital Source

To use AI Start Trigger with a digital source, specify a source and an edge. The source can be any of the following signals:

- DIO <0..15>
- Counter ***n*** Internal Output
- Change Detection Event
- AO Start Trigger (ao/StartTrigger)
- DI Start Trigger (di/StartTrigger)
- DO Start Trigger (do/StartTrigger)

The source can also be one of several other internal signals on the USB-6421. Refer to ***Device Routing in MAX*** for more information.

You can also specify whether the measurement acquisition begins on the rising edge or falling edge of AI Start Trigger.

Routing AI Start Trigger to an Output Terminal

You can route AI Start Trigger out to any DIO <0..15> terminal. The output is an active high pulse.

The USB-6421 also uses AI Start Trigger to initiate pre-triggered DAQ operations. In most pre-triggered applications, a software trigger generates AI Start Trigger.

Related information:

- Device Routing in MAX

Analog Output Measurements

Learn more about making analog output measurements with the USB-6421.

Minimizing Glitches on the Output Signal

When you use a DAC to generate a waveform, you may observe glitches on the output signal. These glitches are normal; when a DAC switches from one voltage to another, it produces glitches due to released charges. The largest glitches occur when the most significant bit of the DAC code changes. You can build a lowpass deglitching filter to remove some of these glitches, depending on the frequency and nature of the output signal.

Analog Output Data Generation Methods

When performing an analog output operation, you can perform software-timed or hardware-timed generations.

Software-Timed Generations

With a software-timed generation, software controls the rate at which data is generated.

Software sends a separate command to the hardware to initiate each DAC conversion. In NI-DAQmx, software-timed generations are referred to as ***on-demand timing***. Software-timed generations are also referred to as ***immediate*** or ***static*** operations. They are typically used for writing a single value out, such as a constant DC voltage.

Hardware-Timed Generations

With a hardware-timed generation, a digital hardware signal controls the rate of the generation. This signal can be generated internally on the USB-6421 or provided externally.

Hardware-timed generations have several advantages over software-timed generations:

- The time between samples can be much shorter.
- The timing between samples can be deterministic.
- Hardware-timed acquisitions can use hardware triggering.

Hardware-timed operations transfer data to the USB-6421 more efficiently by transferring a larger block of data rather than one point at a time, allowing for much higher update rates. The sample mode can be either ***finite*** or ***continuous***.

- **Finite sample mode**—Generates a specific, predetermined number of data samples. Once the specified number of samples has been written out, the generation stops.
- **Continuous sample mode**—Generates an unspecified number of samples. Instead of generating a set number of data samples and stopping, a continuous generation continues until you stop the operation.

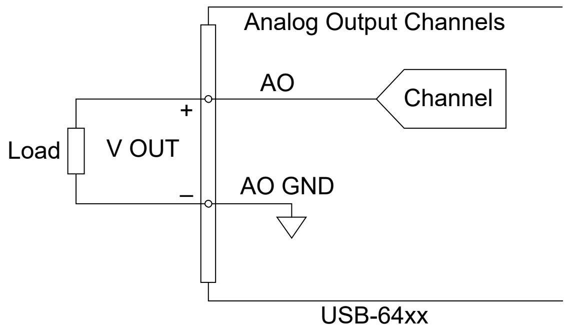
There are three different methods of continuous generation that control what data is written. These methods are ***regeneration, FIFO regeneration***, and ***non-regeneration*** modes:

- **Regeneration**—Repeats data that is already in the buffer. Standard regeneration is when data from the PC buffer is continually downloaded to the FIFO to be written out. New data can be written to the PC buffer at any time without disrupting the output. Use the NI-DAQmx write property RegenMode to allow (or not allow) regeneration. The NI-DAQmx default is to allow regeneration.
- **FIFO regeneration**—Downloads the entire buffer to the FIFO and regenerates it from there. Once the data is downloaded, new data cannot be written to the FIFO. To use FIFO regeneration, the entire buffer must fit within the FIFO size. The advantage of using FIFO regeneration is that it does not require communication with the main host memory once the operation is started, thereby preventing any problems that may occur due to excessive bus traffic. Use the NI-DAQmx DO channel property UseOnlyOnBoardMemory to enable or disable FIFO regeneration.
- **Non-regeneration**—Does not repeat old data. New data must be continually written to the buffer. If the program does not write new data to the buffer at a fast enough rate to keep up with the generation, the buffer underflows and causes an error.

Analog Output Signal Connections

The following figure shows how to make analog output connections to the USB-6421.

Figure 34. USB-6421 Analog Output Signal Connections

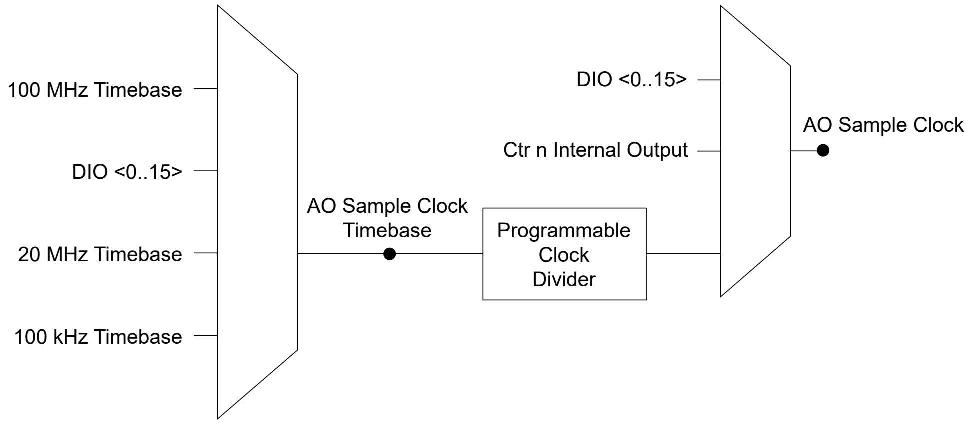


To minimize crosstalk between AO channels, use a separate AO GND return wire for each channel. Connect the return wire for each channel to the AO GND pin directly beneath the AO channel output on the connector.

Analog Output Timing Engine

The following figure summarizes all of the timing options provided by the analog output timing engine.

Figure 35. USB-6421 Analog Output Timing Options



Analog Output Timing Signals

The USB-6421 features three analog output (waveform generation) timing signals.

- AO Start Trigger Signal
- AO Sample Clock Signal
- AO Sample Clock Timebase Signal

AO Start Trigger Signal

Use the AO Start Trigger (ao/StartTrigger) signal to initiate a waveform generation.

If you do not use triggers, you can begin a generation with a software command.

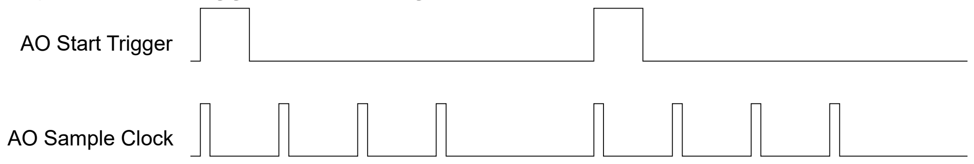
Retriggerable Analog Output

The AO Start Trigger is configurable as retriggerable. The timing engine generates the sample clock for the configured generation in response to each pulse on an AO Start Trigger signal.

The timing engine ignores the AO Start Trigger signal while the clock generation is in progress. After the clock generation is finished, the counter waits for another Start Trigger to begin another clock generation.

The following figure shows a retriggerable AO generation of four samples.

Figure 36. Retriggerable Analog Output



Using a Digital Source

To use AO Start Trigger with a digital source, specify a source and an edge. The source can be one of the following signals:

- A pulse initiated by host software
- DIO <0..15>
- AI Start Trigger (ai/StartTrigger)
- Counter ***n*** Internal Output
- Change Detection Event
- DI Start Trigger (di/StartTrigger)

- DO Start Trigger (do/StartTrigger)

The source can also be one of several other internal signals on the USB-6421. Refer to ***Device Routing in MAX*** for more information.

You can also specify whether the measurement acquisition begins on the rising edge or falling edge of AO Start Trigger.

Routing AO Start Trigger Signal to an Output Terminal

You can route AO Start Trigger out to any DIO <0..15> terminal. The output is an active high pulse.

Related information:

- Device Routing in MAX

AO Sample Clock Signal

Use the AO Sample Clock (ao/SampleClock) signal to initiate AO samples.

Each AO Sample Clock edge simultaneously updates all of the DACs for channels in the task. You can specify an internal or external source for AO Sample Clock. You can also specify whether the DAC update begins on the rising edge or falling edge of AO Sample Clock.

Using an Internal Source

One of the following internal signals can drive AO Sample Clock:

- AO Sample Clock Timebase (divided down)
- Counter ***n*** Internal Output
- Change Detection Event
- Counter ***n*** Sample Clock
- AI Convert Clock (ai/ConvertClock)
- AI Sample Clock (ai/SampleClock)
- DI Sample Clock (di/SampleClock)
- DO Sample Clock (do/SampleClock)

A programmable internal counter divides down the AO Sample Clock Timebase signal.

Several other internal signals can be routed to AO Sample Clock through internal routes. Refer to ***Device Routing in MAX*** for more information.

Using an External Source

Use DIO <0..15> as the source of AO Sample Clock.

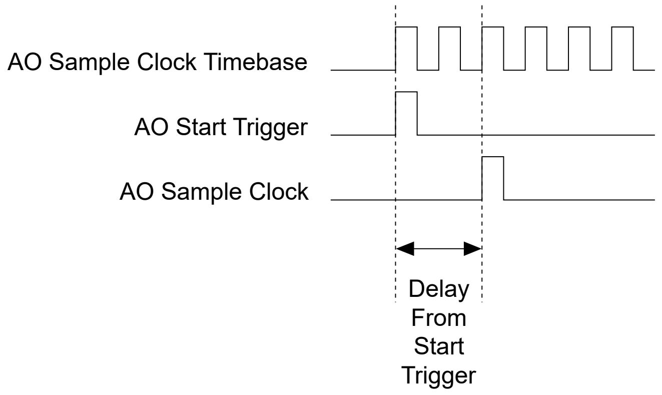
Routing AO Sample Clock Signal to an Output Terminal

You can route AO Sample Clock (as an active low signal) out to any DIO <0..15> terminal.

Other Timing Requirements

The AO timing engine on the USB-6421 internally generates AO Sample Clock unless you select some external source. AO Start Trigger starts the timing engine and either the software or hardware can stop it once a finite generation completes. When using the AO timing engine, you can also specify a configurable delay from AO Start Trigger to the first AO Sample Clock pulse. By default, this delay is two ticks of AO Sample Clock Timebase. The following figure shows the relationship of AO Sample Clock to AO Start Trigger.

Figure 37. AO Sample Clock and AO Start Trigger



Related information:

- Device Routing in MAX

AO Sample Clock Timebase Signal

Use the AO Sample Clock Timebase (ao/SampleClockTimebase) signal to specify a higher frequency clock source to divide down to produce the desired AO Sample Clock rate.

You can route any of the following signals to be the AO Sample Clock Timebase signal:

- 100 MHz Timebase (default)
- 20 MHz Timebase
- 100 kHz Timebase
- DIO <0..15>

AO Sample Clock Timebase is not available as an output on the I/O connector.

You can use an external sample clock signal as AO Sample Clock Timebase signal by dividing the signal down in a DAQ device. You can also use it as AO Sample Clock signal without dividing the signal.

Digital Input/Output Measurements

Learn more about making digital input/output measurements with the USB-6421.

Digital Input Data Acquisition Methods

When performing digital input measurements, you either can perform software-timed or hardware-timed acquisitions.

Software-Timed Acquisitions

With a software-timed acquisition, software controls the rate of the acquisition.

Software sends a separate command to the hardware to initiate each acquisition. In NI-DAQmx, software-timed acquisitions are referred to as having ***on-demand timing***. Software-timed acquisitions are also referred to as ***immediate*** or ***static acquisitions***. They are typically used for reading a single sample of data.

Each of the USB-6421 DIO lines can be used as a static DI or DO line. You can use static DIO lines to monitor or control digital signals. Each DIO can be individually configured as a digital input (DI) or digital output (DO).

All samples of static DI lines and updates of static DO lines are software-timed.

Hardware-Timed Acquisitions

With hardware-timed acquisitions, a digital hardware signal controls the rate of the acquisition.

This signal can be generated internally on the USB-6421 or provided externally.

Hardware-timed acquisitions have several advantages over software-timed acquisitions.

- The time between samples can be much shorter.

- The timing between samples is deterministic.
- Hardware-timed acquisitions can use hardware triggering.

Hardware-timed operations transfer data to the USB-6421 more efficiently by transferring a larger block of data rather than one point at a time, allowing for much higher sample rates. The sample mode can be either ***finite*** or ***continuous***.

- **Finite sample mode**—Acquires a specific, predetermined number of data samples. Once the specified number of samples has been read in, the acquisition stops.
- **Continuous sample mode**—Acquires an unspecified number of samples. Instead of acquiring a set number of data samples and stopping, a continuous acquisition continues until you stop the operation. Continuous acquisition is also referred to as ***double-buffered*** or ***circular-buffered acquisition***.

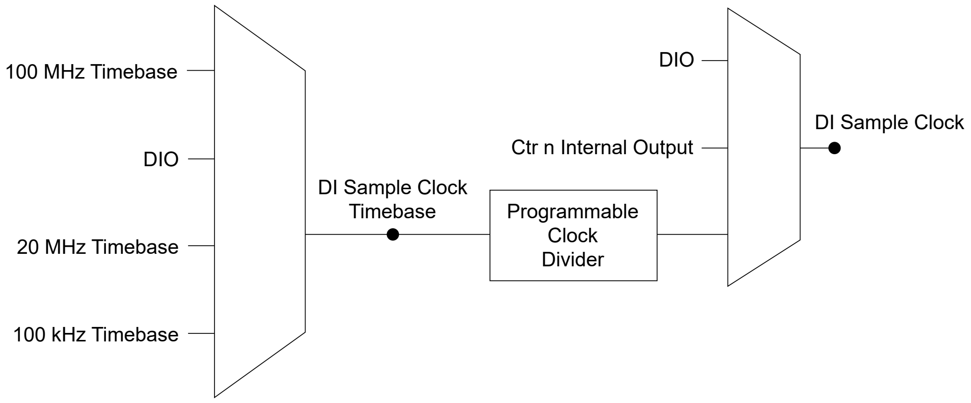
If data cannot be transferred across the bus fast enough, the FIFO becomes full. New acquisitions overwrites data in the FIFO before it can be transferred to host memory, which causes the USB-6421 to generate an error. With continuous operations, if the user program does not read data out of the PC buffer fast enough to keep up with the data transfer, the buffer could reach an overflow condition, causing an error to be generated.

Digital Waveform Acquisition

You can acquire digital waveforms on the DIO lines. You can configure each DIO line to be an output, a static input, or a digital waveform acquisition input.

The DI waveform acquisition FIFO stores the digital samples. The USB-6421 samples the DIO lines on each rising or falling edge of a clock signal, DI Sample Clock.

The following figure summarizes all of the timing options provided by the digital input timing engine.

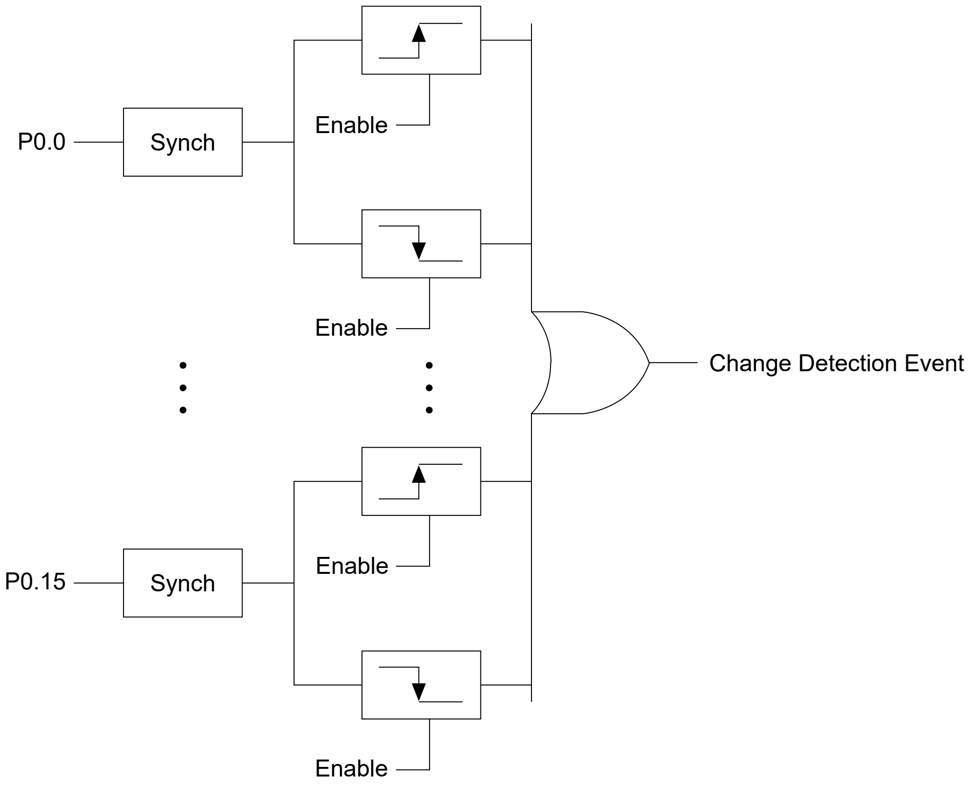


DI Change Detection

You can configure the USB-6421 to detect changes on all 16 digital input lines (P0).

The following figure shows a block diagram of the DIO change detection circuitry.

Figure 78. DI Change Detection



You can enable the DIO change detection circuitry to detect rising edges, falling edges, or either edge individually on each DIO line. The USB-6421 synchronizes each DI signal to the 100 MHz Timebase, and then sends the signal to the change detectors. The circuitry ORs the output of all enabled change detectors from every DI signal. The result of this OR is the Change Detection Event signal.

Change detection performs bus correlation by considering all changes within a 50 ns window one change detection event, which keeps signals on the same bus synchronized in samples and prevents overruns.

The Change Detection Event signal can do the following:

- Drive any DIO <0..15> signal
- Drive the DO Sample Clock or DI Sample Clock
- Generate an interrupt

The Change Detection Event signal can also be used to detect changes on digital output events.

DI Change Detection Applications

The DIO change detection circuitry can interrupt a user program when one of several DIO signals changes state.

You can also use the output of the DIO change detection circuitry to trigger a DI or counter acquisition on the logical OR of several digital signals. By routing the Change Detection Event signal to a counter, you can also capture the relative time between bus changes.

You can also use the Change Detection Event signal to trigger DO or counter generations.

Digital Input Timing Signals

The USB-6421 features five digital input timing signals.

Learn more about each digital input timing signal.

DI Sample Clock Signal

The USB-6421 uses the DI Sample Clock (di/SampleClock) signal to sample the Port 0 terminals and store the result in the DI waveform acquisition FIFO.

You can specify an internal or external source for DI Sample Clock. You can also specify whether the measurement sample begins on the rising edge or falling edge of DI Sample Clock.

If the USB-6421 receives a DI Sample Clock when the FIFO is full, it reports an overflow error to the host software.

Using an Internal Source

To use DI Sample Clock with an internal source, specify the signal source and the polarity of the signal. The source can be any of the following signals:

- DI Sample Clock (di/SampleClock)
- DO Sample Clock (do/SampleClock)
- AI Sample Clock (ai/SampleClock)
- AI Convert Clock (ai/ConvertClock)
- AO Sample Clock (ao/SampleClock)
- Counter ***n*** Sample Clock
- Counter ***n*** Internal Output
- Frequency Output
- DI Change Detection output

Several other internal signals can be routed to DI Sample Clock through internal routes. Refer to ***Device Routing in MAX*** for more information.

Using an External Source

You can route any DIO <0..15> signals as DI Sample Clock. You can sample data on the rising or falling edge of DI Sample Clock.

Routing DI Sample Clock to an Output Terminal

You can route DI Sample Clock out to any DIO <0..15> terminal. The DIO circuitry inverts the polarity of DI Sample Clock before driving the DIO terminal.

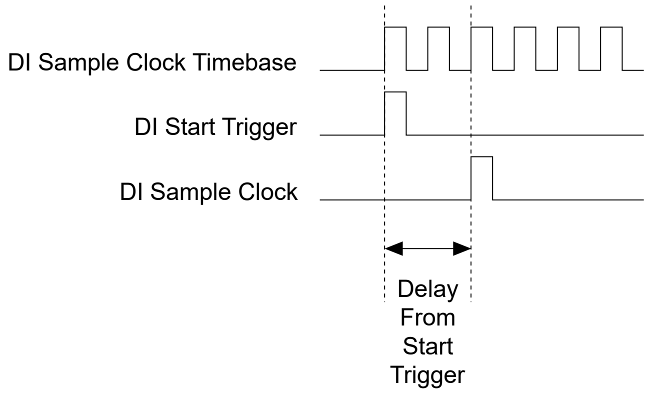
Other Timing Requirements

The USB-6421 only acquires data during an acquisition. It ignores DI Sample Clock when a measurement acquisition is not in progress.

The DI timing engine on the USB-6421 internally generates DI Sample Clock unless you select an external source. DI Start Trigger starts the timing engine and either software or hardware can stop it once a finite acquisition completes. When using the DI timing engine, you can also specify a configurable delay from DI Start Trigger to the first DI Sample Clock pulse.

By default, this delay is set to two ticks of the DI Sample Clock Timebase signal.

Figure 39. DI Sample Clock and DI Start Trigger



Related information:

- Device Routing in MAX

DI Sample Clock Timebase Signal

You can route any of the following signals to be the DI Sample Clock Timebase (di/SampleClockTimebase) signal:

- 100 MHz Timebase (default)
- 20 MHz Timebase
- 100 kHz Timebase
- DIO <0..15>

Refer to the device routing table in MAX for all additional routable signals. To find the device routing table for your device, launch MAX and select **Devices and Interfaces** »

NI-DAQmx Devices. Click a device to open a tabbed window in the middle pane. Click the **Device Routes** tab at the bottom of the pane to display the device routing table.

DI Sample Clock Timebase is not available as an output on the I/O connector. DI Sample Clock Timebase is divided down to provide one of the possible sources for DI Sample Clock. You can configure the polarity selection for DI Sample Clock Timebase as either rising or falling edge except for the 100 MHz Timebase or 20 MHz Timebase.

You might use DI Sample Clock Timebase if you want to use an external sample clock signal, but need to divide the signal down. If you want to use an external sample clock signal, but do not need to divide the signal, then you should use DI Sample Clock rather than DI Sample Clock Timebase.

DI Start Trigger Signal

Use the DI Start Trigger (di/StartTrigger) signal to begin a measurement acquisition.

A measurement acquisition consists of one or more samples. If you do not use triggers, begin a measurement with a software command. Once the acquisition begins, configure the acquisition to stop:

- When a certain number of points are sampled (in finite mode)
- With a software command (in continuous mode)

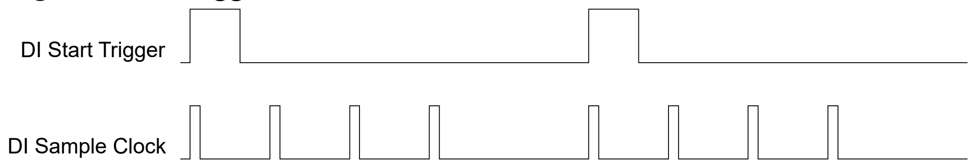
An acquisition that uses a start trigger is sometimes referred to as a ***post-triggered acquisition***.

Retriggerable DI

The DI Start Trigger is configurable as retriggerable. When the DI Start Trigger is configured as retriggerable, the timing engine generates the sample and convert clocks for the configured acquisition in response to each pulse on a DI Start Trigger signal.

The timing engine ignores the DI Start Trigger signal while the clock generation is in progress. After the clock generation is finished, the timing engine waits for another Start Trigger to begin another clock generation. The following figure shows a retriggerable DI of four samples.

Figure 40. Retriggerable DI



Note Waveform information from LabVIEW does not reflect the delay between triggers. They are treated as a continuous acquisition with constant t0 and dt information.

Using a Digital Source

To use DI Start Trigger with a digital source, specify a source and an edge. The source can be any of the following signals:

- DIO <0..15>
- Counter ***n*** Internal Output
- Change Detection Event
- AI Start Trigger (ai/StartTrigger)
- AO Start Trigger (ao/StartTrigger)
- DO Start Trigger (do/StartTrigger)

The source can also be one of several other internal signals on the USB-6421. Refer to ***Device Routing in MAX*** for more information.

You can also specify whether the measurement acquisition begins on the rising edge or falling edge of DI Start Trigger.

Routing DI Start Trigger to an Output Terminal

You can route DI Start Trigger out to any DIO <0..15> terminal. The output is an active high pulse. All DIO terminals are configured as inputs by default.

Related information:

- Device Routing in MAX

Digital Output Data Generation Methods

When performing a digital waveform operation, you either can perform software-timed or hardware-timed generations.

Software-Timed Generations

With a software-timed generation, software controls the rate at which data is generated.

Software sends a separate command to the hardware to initiate each update. In NI-DAQmx, software-timed generations are referred to as ***on-demand timing***. Software-timed generations are also referred to as ***immediate*** or ***static*** operations. They are typically used for writing a single value out, such as a constant digital value.

Hardware-Timed Generations

With a hardware-timed generation, a digital hardware signal controls the rate of the generation. This signal can be generated internally on the USB-6421 or provided externally.

Hardware-timed generations have several advantages over software-timed generations:

- The time between samples can be much shorter.
- The timing between samples can be deterministic.
- Hardware-timed acquisitions can use hardware triggering.

Hardware-timed operations transfer data to the USB-6421 more efficiently by transferring a larger block of data rather than one point at a time, allowing for much higher update rates. The sample mode can be either ***finite*** or ***continuous***.

- **Finite sample mode**—Generates a specific, predetermined number of data samples. Once the specified number of samples has been written out, the generation stops.
- **Continuous sample mode**—Generates an unspecified number of samples. Instead of generating a set number of data samples and stopping, a continuous generation

continues until you stop the operation.

There are three different methods of continuous generation that control what data is written. These methods are ***regeneration, FIFO regeneration***, and ***non-regeneration*** modes:

- **Regeneration**—Repeats data that is already in the buffer. Standard regeneration is when data from the PC buffer is continually downloaded to the FIFO to be written out. New data can be written to the PC buffer at any time without disrupting the output. Use the NI-DAQmx write property RegenMode to allow (or not allow) regeneration. The NI-DAQmx default is to allow regeneration.
- **FIFO regeneration**—Downloads the entire buffer to the FIFO and regenerates it from there. Once the data is downloaded, new data cannot be written to the FIFO. To use FIFO regeneration, the entire buffer must fit within the FIFO size. The advantage of using FIFO regeneration is that it does not require communication with the main host memory once the operation is started, thereby preventing any problems that may occur due to excessive bus traffic. Use the NI-DAQmx DO channel property UseOnlyOnBoardMemory to enable or disable FIFO regeneration.
- **Non-regeneration**—Does not repeat old data. New data must be continually written to the buffer. If the program does not write new data to the buffer at a fast enough rate to keep up with the generation, the buffer underflows and causes an error.

Digital Waveform Generation

You can generate digital waveforms on the DIO lines. The DO waveform generation FIFO stores the digital samples.

The USB-6421 has a DMA controller dedicated to moving data from the system memory to the DO waveform generation FIFO. The USB-6421 moves samples from the FIFO to the DIO terminals on each rising or falling edge of a clock signal, DO Sample Clock. You can configure each DIO signal to be an input, a static output, or a digital waveform generation output.

The FIFO supports a retransmit mode. In the retransmit mode, after all the samples in the FIFO have been clocked out, the FIFO begins outputting all of the samples again in the same order. For example, if the FIFO contains five samples, the pattern generated consists of sample #1, #2, #3, #4, #5, #1, #2, #3, #4, #5, #1, and so on.

Digital Output Timing Signals

The USB-6421 features four digital output (waveform generation) timing signals.

Learn more about each digital output timing signal.

DO Sample Clock Signal

The USB-6421 uses the DO Sample Clock (do/SampleClock) signal to update the DO terminals with the next sample from the DO waveform generation FIFO.

You can specify an internal or external source for DO Sample Clock. You can also specify whether the DAC update begins on the rising edge or falling edge of DO Sample Clock. If the USB-6421 receives a DO Sample Clock when the FIFO is empty, the USB-6421 reports an underflow error to the host software.

Using an Internal Source

One of the following internal signals can drive DO Sample Clock:

- DI Sample Clock (di/SampleClock)
- DO Sample Clock (do/SampleClock)
- AI Sample Clock (ai/SampleClock)
- AI Convert Clock (ai/ConvertClock)
- AO Sample Clock (ao/SampleClock)
- Counter ***n*** Sample Clock
- Counter ***n*** Internal Output
- Frequency Output
- DI Change Detection output

Several other internal signals can be routed to DO Sample Clock through internal routes. Refer to ***Device Routing in MAX*** for more information.

Using an External Source

Use DIO <0..15> as the source of DO Sample Clock.

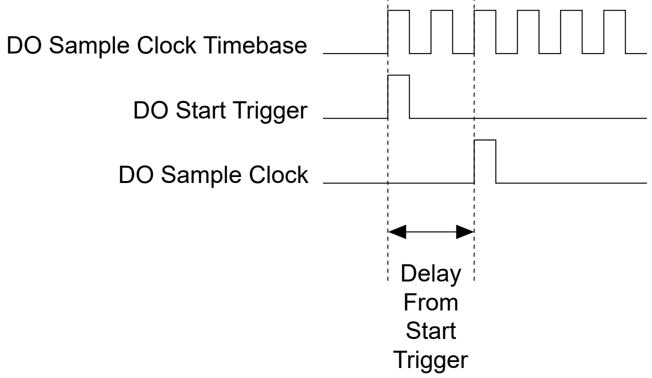
Routing DO Sample Clock to an Output Terminal

You can route DO Sample Clock (as an active low signal) out to any DIO <0..15> terminal.

Other Timing Requirements

The DO timing engine on the USB-6421 internally generates DO Sample Clock unless you select some external source. DO Start Trigger starts the timing engine and either the software or hardware can stop it once a finite generation completes. When using the DO timing engine, you can also specify a configurable delay from DO Start Trigger to the first DO Sample Clock pulse. By default, this delay is two ticks of DO Sample Clock Timebase. The following figure shows the relationship of DO Sample Clock to DO Start Trigger

Figure 41. DO Sample Clock and DO Start Trigger



Related information:

- Device Routing in MAX

DO Sample Clock Timebase Signal

The DO Sample Clock Timebase (do/SampleClockTimebase) signal is divided down to provide a source for DO Sample Clock.

You can route any of the following signals to be the DO Sample Clock Timebase signal:

- 100 MHz Timebase (default)
- 20 MHz Timebase
- 100 kHz Timebase
- DIO <0..15>

DO Sample Clock Timebase is not available as an output on the I/O connector.

You might use DO Sample Clock Timebase if you want to use an external sample clock signal, but need to divide the signal down. If you want to use an external sample clock signal, but do not need to divide the signal, then you should use DO Sample Clock rather than DO Sample Clock Timebase.

DO Start Trigger Signal

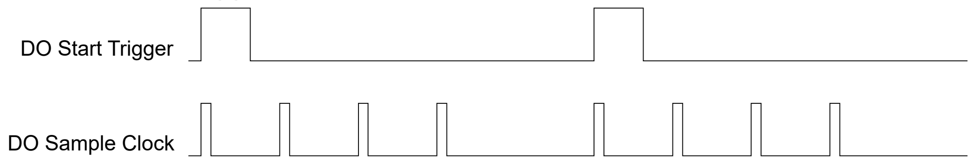
Use the DO Start Trigger (do/StartTrigger) signal to initiate a waveform generation. If you do not use triggers, you can begin a generation with a software command.

Retriggerable DI

The DO Start Trigger is configurable as retriggerable. When DO Start Trigger is configured as retriggerable, the timing engine generates the sample clocks for the configured generation in response to each pulse on a DO Start Trigger signal.

The timing engine ignores the DO Start Trigger signal while the clock generation is in progress. After the clock generation is finished, the timing engine waits for another start trigger to begin another clock generation. The following figure shows a retriggerable DO of four samples.

Figure 59. Retriggerable DO



Using a Digital Source

To use DO Start Trigger with a digital source, specify a source and an edge. The source can be any of the following signals:

- A pulse initiated by host software
- DIO <0..15>
- AI Start Trigger (ai/StartTrigger)
- AO Start Trigger (ao/StartTrigger)
- Counter ***n*** Internal Output
- DI Start Trigger (do/StartTrigger)
- Change Detection Event

The source can also be one of several other internal signals on the USB-6421. Refer to ***Device Routing in MAX*** for more information.

You can also specify whether the measurement acquisition begins on the rising edge or falling edge of DI Start Trigger.

Routing DO Start Trigger to an Output Terminal

You can route DO Start Trigger out to any DIO <0..15> terminal. The output is an active high pulse. All DIO terminals are configured as inputs by default.

I/O Protection

Each DIO signal is protected against overvoltage, undervoltage, and overcurrent conditions as well as ESD events.

Follow these guidelines to avoid these fault conditions:

- If you configure a DIO line as an output, do not connect it to any external signal source, ground, or power supply.
- If you configure a DIO line as an output, understand the current requirements of the load connected to these signals. Do not exceed the specified current output limits of the USB-6421. NI has several signal conditioning solutions for digital applications requiring high current drive.
- If you configure a DIO line as an input, do not drive the line with voltages outside of its normal operating range. The DIO lines have a smaller operating range than the AI signals.

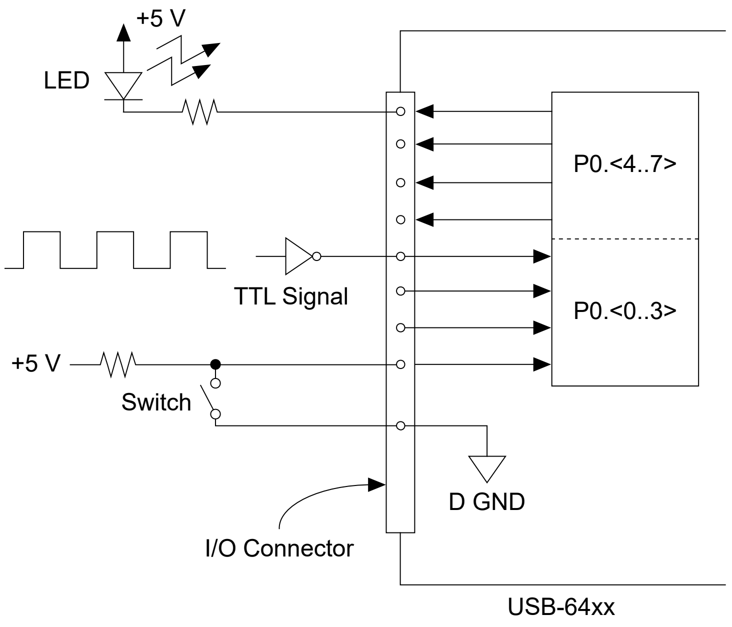
- Treat the USB-6421 as you would treat any static sensitive device. Always properly ground yourself and the equipment when handling the USB-6421 or connecting to it.

Digital I/O Signal Connections

The DIO signals are referenced to D GND. You can individually program each line as an input or output.

The following figure shows P0.<4..7> configured for digital input and P0.<0..3> configured for digital output, the switch receiving TTL signals and sensing external device states, and the LED sending TTL signals and driving external devices.

Figure 65. Digital I/O Connections



Notice Exceeding the maximum input voltage ratings, which are listed in the ***USB-6421 Specifications***, can damage the USB-6421 and the computer. NI is not liable for any damage resulting from such signal connections.

For best signal integrity on high speed digital signals, use twisted pair wiring and twist each signal wire with a D GND wire. Connect the D GND wire to the D GND pin closest to the signal pin. Note that this requires sharing one D GND pin for every two signals.

Using DIO Terminals as Timing Input Signals

Use DIO terminals to route external timing signals to many different functions.

Each DIO terminal can be routed to any of the following signals:

- AI Convert Clock (ai/ConvertClock)
- AI Sample Clock (ai/SampleClock)
- AI Sample Clock Timebase (ai/SampleClockTimebase)
- AI Start Trigger (ai/StartTrigger)
- AO Start Trigger (ao/StartTrigger)
- AO Sample Clock (ao/SampleClock)
- AO Sample Clock Timebase (ao/SampleClockTimebase)
- Counter input signals for all counters—Source, Gate, Aux, HW_Arm, A, B, Z
- Counter ***n*** Sample Clock
- DI Sample Clock (di/SampleClock)
- DI Sample Clock Timebase (di/SampleClockTimebase)
- DO Sample Clock (do/SampleClock)

Most functions allow you to configure the polarity of DIO inputs and whether the input is edge or level sensitive.

Using DIO Terminals to Export Timing Output Signals

You can route any of the following timing signals to any DIO terminal configured as an output:

- AI Convert Clock* (ai/ConvertClock)
- AI Hold Complete Event (ai/HoldCompleteEvent)
- AI Sample Clock (ai/SampleClock)
- AI Start Trigger (ai/StartTrigger)
- AO Sample Clock* (ao/SampleClock)
- AO Start Trigger (ao/StartTrigger)
- DI Sample Clock (di/SampleClock)
- DI Start Trigger (di/StartTrigger)
- DO Sample Clock* (do/SampleClock)
- DO Start Trigger (do/StartTrigger)

- Counter ***n*** Source
- Counter ***n*** Gate
- Counter ***n*** Internal Output
- Counter ***n*** Sample Clock
- Counter ***n*** Counter ***n*** HW Arm
- Frequency Output
- Change Detection Event

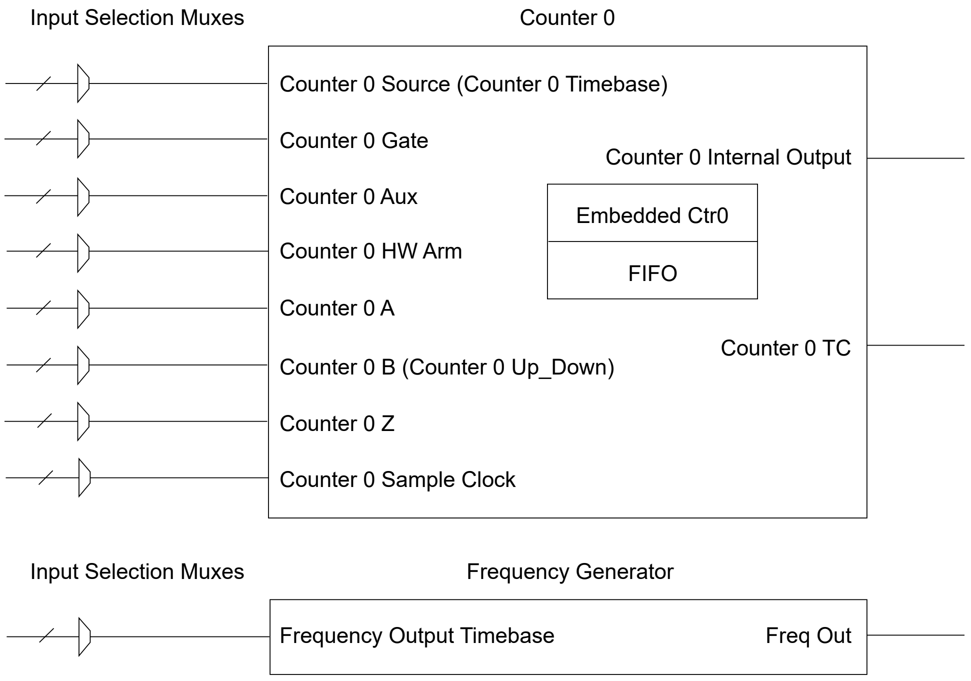
Note Signals with an * are inverted before being driven to a terminal; that is, these signals are active low.

Counters

The USB-6421 has four general-purpose 32-bit counter/timers and one frequency generator. The general-purpose counter/timers can be used for many measurement and pulse generation applications.

The following figure shows the USB-6421 Counter 0 and the frequency generator. All four counters on the USB-6421 are identical.

Figure 44. USB-6421 Counter 0 and Frequency Generator



Counters have eight input signals, although in most applications only a few inputs are used.

Each counter also contains an embedded counter (Embedded Ctr***n***) for use in what are traditionally two-counter measurements and generations. The embedded counters cannot be programmed independent of the main counter; signals from the embedded counters are not routable.

Counter Timing Engine

USB-6421 counters do not have the ability to divide down a timebase to produce an internal counter sample clock. For sample clocked operations, an external signal must be provided to supply a clock source.

The source can be any of the following signals:

- AI Sample Clock
- AI Start Trigger
- AO Sample Clock
- DI Sample Clock
- DI Start Trigger
- DO Sample Clock
- CTR ***n*** Internal Output
- Freq Out
- DIO <0..15>
- Change Detection Event

Not all timed counter operations require a sample clock. For example, a simple buffered pulse width measurement latches in data on each edge of a pulse. For this measurement, the measured signal determines when data is latched in. These operations are referred to as implicit timed operations. However, many of the same measurements can be clocked at an interval with a sample clock. These are referred to as sample clocked operations. The following table shows the different options for the different measurements.

Table 17. Counter Timing Measurements

Measurement	Implicit Timing Support
Buffered Edge Count	X
Buffered Pulse Width	✓
Buffered Pulse	✓
Buffered Semi-Period	✓
Buffered Frequency	✓
Buffered Period	✓

Measurement	Implicit Timing Support
Buffered Position	X
Buffered Two-Signal Edge Separation	✓

Counter Input Applications

Refer to the following sections for more information on the various counter input applications available on the USB-6421.

Counting Edges

In edge counting applications, the counter counts edges on its Source after the counter is armed. You can configure the counter to count rising or falling edges on its Source input.

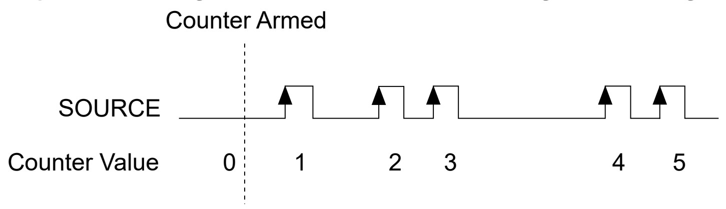
You can also control the direction of counting (up or down), as described in the ***Controlling the Direction of Counting*** section. The counter values can be read on demand or with a sample clock.

Single Point (On-Demand) Edge Counting

With single point (on-demand) edge counting, the counter counts the number of edges on the Source input after the counter is armed.

On-demand refers to the fact that software can read the counter contents at any time without disturbing the counting process. The following figure shows an example of single point edge counting.

Figure 45. Single Point (On-Demand) Edge Counting



Controlling the Direction of Counting

In edge counting applications, the counter can count up or down. You can configure the counter to do the following:

- Always count up
- Always count down
- Count up when the Counter 0 B input is high; count down when it is low

Pulse-Width Measurement

In pulse-width measurements, the counter measures the width of a pulse on its Gate input signal. You can configure the counter to measure the width of high pulses or low pulses on the Gate signal.

You can route an internal or external periodic clock signal (with a known period) to the Source input of the counter. The counter counts the number of rising (or falling) edges on the Source signal while the pulse on the Gate signal is active.

You can calculate the pulse width by multiplying the period of the Source signal by the number of edges returned by the counter.

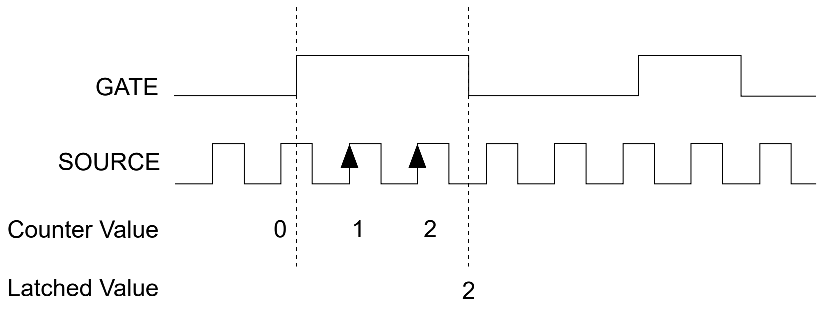
A pulse-width measurement is accurate even if the counter is armed while a pulse train is in progress. If a counter is armed while the pulse is in the active state, it waits for the next transition to the active state to begin the measurement.

Single Pulse-Width Measurement

With single pulse-width measurement, the counter counts the number of edges on the Source input while the Gate input remains active.

When the Gate input goes inactive, the counter stores the count in the FIFO and ignores other edges on the Gate and Source inputs. Software then reads the stored count. The following figure shows an example of a single pulse-width measurement.

Figure 47. Single Pulse-Width Measurement



Pulse versus Semi-Period Measurements

In hardware, pulse measurement and semi-period are the same measurement. Both measure the high and low times of a pulse. The functional difference between the two measurements is how the data is returned.

In a semi-period measurement, each high or low time is considered one point of data and returned in units of seconds or ticks. In a pulse measurement, each pair of high and low times is considered one point of data and returned as a paired sample in units of frequency and duty cycle, high and low time or high and low ticks. When reading data, 10 points in a semi-period measurement gets an array of five high times and five low times. When you read 10 points in a pulse measurement, you get an array of 10 pairs of high and low times.

Also, pulse measurements support sample clock timing while semi-period measurements do not.

Pulse Measurement

In pulse measurements, the counter measures the high and low time of a pulse on its Gate input signal after the counter is armed.

A pulse is defined in terms of its high and low time, high and low ticks or frequency and duty cycle, which is similar to the pulse-width measurement, except that the inactive pulse is measured as well.

You can route an internal or external periodic clock signal (with a known period) to the Source input of the counter. The counter counts the number of rising (or falling) edges occurring on the Source input between two edges of the Gate signal.

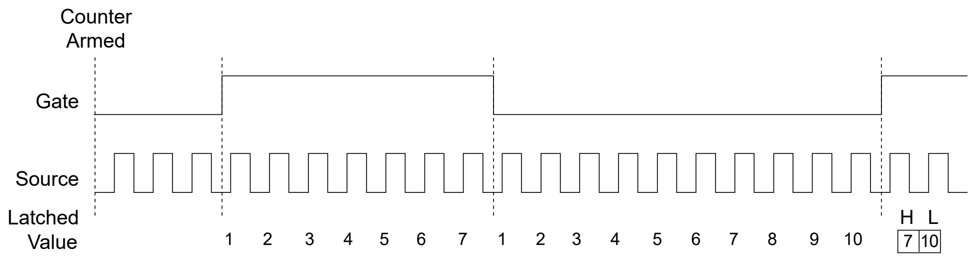
You can calculate the high and low time of the Gate input by multiplying the period of the Source signal by the number of edges returned by the counter.

Single Pulse Measurement

Single (on-demand) pulse measurement is equivalent to two single pulse-width measurements on the high (H) and low (L) ticks of a pulse.

The following figure shows an example of a single (on-demand) pulse measurement.

Figure 48. Single (On-Demand) Pulse Measurement



Semi-Period Measurement

In semi-period measurements, the counter measures a semi-period on its Gate input signal after the counter is armed. A semi-period is the time between any two consecutive edges on the Gate input.

You can route an internal or external periodic clock signal (with a known period) to the Source input of the counter. The counter counts the number of rising (or falling) edges occurring on the Source input between two edges of the Gate signal.

You can calculate the semi-period of the Gate input by multiplying the period of the Source signal by the number of edges returned by the counter.

Single Semi-Period Measurement

Single semi-period measurement is equivalent to single pulse-width measurement.

Frequency Measurement

You can use the counters to measure frequency in several different ways.

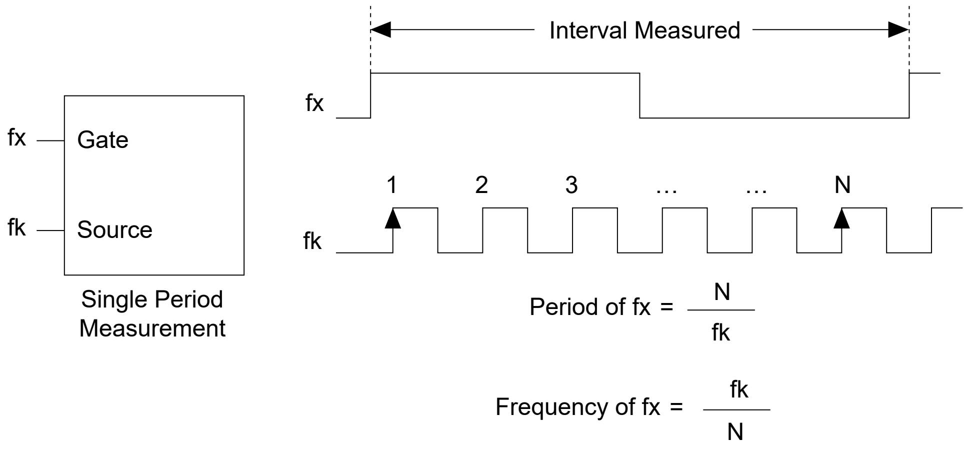
Low Frequency with One Counter

For low frequency measurements with one counter, you measure one period of your signal using a known timebase. You can route the signal to measure (***fx***) to the Gate of a counter.

You can route a known timebase (fk) to the Source of the counter. The known timebase can be an onboard timebase, such as 100 MHz Timebase, 20 MHz Timebase, or 100 kHz Timebase, or any other signal with a known rate.

You can configure the counter to measure one period of the gate signal. The frequency of fx is the inverse of the period. The following figure illustrates this method.

Figure 68. Low Frequency with One Counter



High Frequency with Two Counters

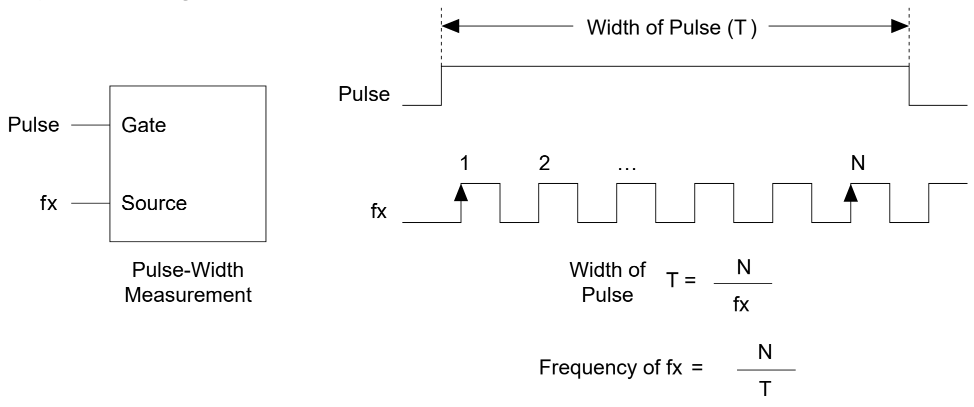
For high frequency measurements with two counters, you measure one pulse of a known width using your signal and derive the frequency of your signal from the result.

Note Counter 0 is always paired with Counter 1. Counter 2 is always paired with Counter 3.

In this method, you route a pulse of known duration (T) to the Gate of a counter. You can generate the pulse using a second counter. You can also generate the pulse externally and connect it to a DIO terminal. You only need to use one counter if you generate the pulse externally.

Route the signal to measure (fx) to the Source of the counter. Configure the counter for a single pulse-width measurement. If you measure the width of pulse T to be N periods of fx, the frequency of fx is N/T. The following figure illustrates this method.

Figure 52. High Frequency with Two Counters



Another option is to measure the width of a known period instead of a known pulse.

Large Range of Frequencies with Two Counters

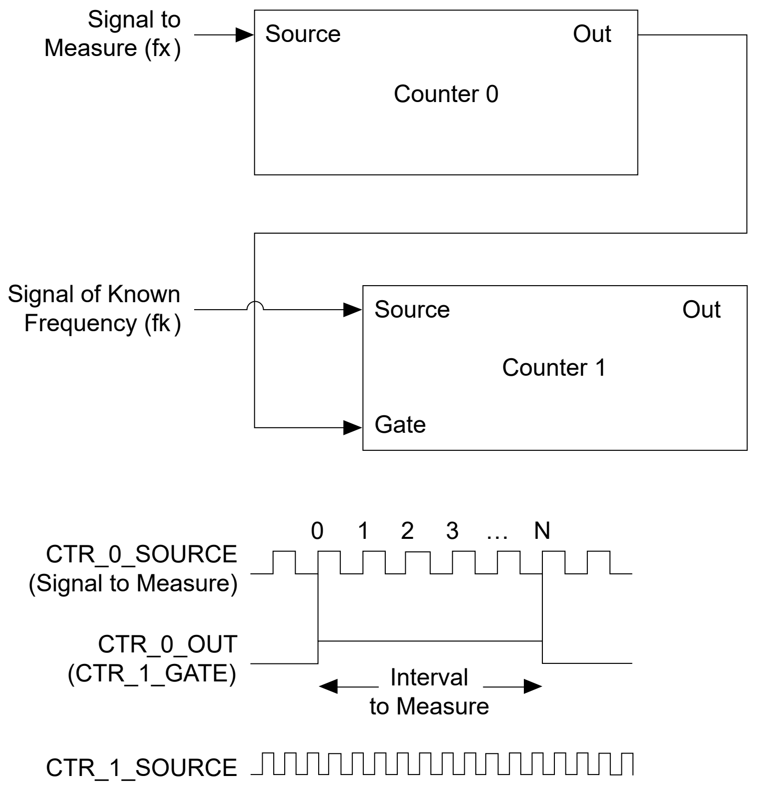
By using two counters, you can accurately measure a signal that might be high or low frequency. This technique is called reciprocal frequency measurement.

When measuring a large range of frequencies with two counters, you generate a long pulse using the signal to measure. You then measure the long pulse with a known timebase. The USB-6421 can measure this long pulse more accurately than the faster input signal.

Note Counter 0 is always paired with Counter 1. Counter 2 is always paired with Counter 3.

You can route the signal to measure to the Source input of Counter 0, as shown in the following figure. Assume this signal to measure has frequency fx. NI-DAQmx automatically configures Counter 0 to generate a single pulse that is the width of N periods of the source input signal.

Figure 51. Large Range of Frequencies with Two Counters



NI-DAQmx then routes the Counter 0 Internal Output signal to the gate of Counter 1. You can then route a signal of known frequency (fk) as a counter timebase to the Counter 1 Source input. NI-DAQmx configures Counter 1 to perform a single pulse-width measurement. Suppose the result is that the pulse width is J periods of the fk clock.

From Counter 0, the length of the pulse is N/fx. From Counter 1, the length of the same pulse is J/fk. Therefore, the frequency of fx is given by fk = fk * (N/J).

Choosing a Method for Measuring Frequency

The best method to measure frequency depends on several factors including the expected frequency of the signal to measure, the desired accuracy, how many counters are available, and how long the measurement can take.

Which Method is Best?

This depends on the frequency to be measured, the rate at which you want to monitor the frequency and the accuracy you desire.

- Low frequency measurements with one counter is a good method for many applications. However, the accuracy of the measurement decreases as the frequency increases.
- High frequency measurements with two counters is accurate for high frequency signals. However, the accuracy decreases as the frequency of the signal to measure decreases. At very low frequencies, this method may be too inaccurate for your application. Another disadvantage of this method is that it requires two counters (if you cannot provide an external signal of known width). An advantage of high frequency measurements with two counters is that the measurement completes in a known amount of time.
- Measuring a large range of frequencies with two counters measures high and low frequency signals accurately. However, it requires two counters, and it has a variable sample time and variable error % dependent on the input signal.

The following table summarizes some of the differences in methods of measuring frequency.

Table 18. Frequency Measurement Method Comparison

Method	**Number of Counters Used**	**Number of Measurements Returned**	**Measures High Frequency Signals Accurately**	**Measures Low Frequency Signals Accurately**
Low frequency with one counter	1	1	Poor	Good
High frequency with two counters	1 or 2	1	Good	Poor
Large range of frequencies with two counters	2	1	Good	Good
Sample clocked (averaged)	1	1	Good	Good

The remainder of this topic provides more detail.

Considerations for Choosing Frequency Measurement Method

For all frequency measurement methods, assume the following:

fx	is the frequency to be measured if no error
fk	is the known source or gate frequency
measurement time (T)	is the time it takes to measure a single sample
Divide down (N)	is the integer to divide down measured frequency, only used in large range two counters
fs	is the sample clock rate, only used in sample clocked frequency measurements

Here is how these variables apply to each method, with a summary in the following table:

- **One counter**—With one counter measurements, a known timebase is used for the source frequency (fk). The measurement time is the period of the frequency to be measured, or 1/fk.
- **Two counter high frequency**—With the two counter high frequency method, the second counter provides a known measurement time. The gate frequency equals 1/measurement time.
- **Two counter large range**—The two counter larger range measurement is the same as a one counter measurement, but now the user has an integer divide down of the signal. An internal timebase is still used for the source frequency (fk), but the divide down means that the measurement time is the period of the divided down signal, or N/fx where N is the divide down.

Table 19. Frequency measurement methods

Variable	**One Counter**	**Two Counter**	
		High Frequency	**Large Range**
f k	Known timebase	$\frac{1}{\text{gating period}}$	Known timebase
Measurement time	$\frac{1}{fx}$	gating period	$\frac{N}{fx}$
Max. frequency error	$fx \times \frac{fx}{fk - fx}$	***f k***	$fx \times \frac{fx}{N \times fk - fx}$
Max. error %	$\frac{fx}{fk - fx}$	$\frac{fk}{fx}$	$\frac{fx}{N \times fk - fx}$

Note Accuracy equations do not take clock stability into account. Refer to your device specifications for clock stability.

For a practical example, consider measuring a 50 kHz signal. Assuming that the measurement times for the sample clocked (with averaging) and two counter frequency measurements are configured the same, the following table summarizes the results.

Table 20. 50 kHz Frequency Measurement Methods

Variable	One Counter	Two Counter	
		High Frequency	Large Range
fx	50,000	50,000	50,000
f k	100 M	1,000	100 M
Measurement time (***ms***)	.02	1	1
N	—	—	50
Max. frequency error (Hz)	25	1,000	.5
Max. error %	.05	2	.001

From these results, you can see that while the measurement time for one counter is shorter, the accuracy is best in the sample clocked and two counter large range measurements. For another example, the following table shows the results for 5 MHz.

Table 21. 5 MHz Frequency Measurement Methods

Variable	One Counter	Two Counter	
		High Frequency	Large Range
fx	5 M	5 M	5 M
f k	100 M	1,000	100 M
Measurement time (***ms***)	.0002	1	1
N	—	—	5,000
Max. Frequency error (Hz)	263 k	1,000	50

Variable	One Counter	Two Counter	
		High Frequency	Large Range
Max. Error %	5.26	.02	.001

The following table summarizes some of the differences in methods of measuring frequency.

Table 22. Frequency Measurement Method Comparison

Method	Number of Counters Used	Number of Measurements Returned	Measures High Frequency Signals Accurately	Measures Low Frequency Signals Accurately
Low frequency with one counter	1	1	Poor	Good
High frequency with two counters	1 or 2	1	Good	Poor
Large range of frequencies with two counters	2	1	Good	Good

Period Measurement

In period measurements, the counter measures a period on its Gate input signal after the counter is armed. You can configure the counter to measure the period between two rising edges or two falling edges of the Gate input signal.

You can route an internal or external periodic clock signal (with a known period) to the Source input of the counter. The counter counts the number of rising (or falling) edges occurring on the Source input between the two active edges of the Gate signal.

You can calculate the period of the Gate input by multiplying the period of the Source signal by the number of edges returned by the counter.

Period measurements return the inverse results of frequency measurements.

Position Measurement

You can use the counters to perform position measurements with quadrature encoders or two-pulse encoders. You can measure angular position with X1, X2, and X4 angular encoders. Linear position can be measured with two-pulse encoders.

You can choose to do either a single point (on-demand) position measurement or a buffered (sample clock) position measurement. You must arm a counter to begin position measurements.

Measurements Using Quadrature Encoders with X1, X2, or X3 encoding

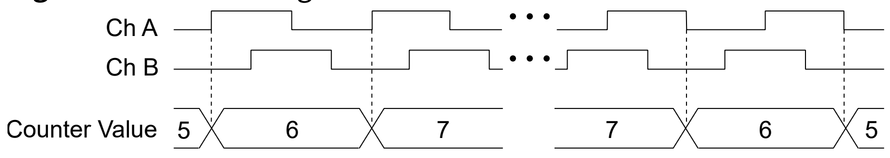
The counters can perform measurements of quadrature encoders that use X1, X2, or X4 encoding. A quadrature encoder can have up to three channels—channels A, B, and Z.

X1 Encoding

When channel A leads channel B in a quadrature cycle, the counter increments. When channel B leads channel A in a quadrature cycle, the counter decrements. The amount of increments and decrements per cycle depends on the type of encoding—X1, X2, or X4.

The following figure shows a quadrature cycle and the resulting increments and decrements for X1 encoding. When channel A leads channel B, the increment occurs on the rising edge of channel A. When channel B leads channel A, the decrement occurs on the falling edge of channel A.

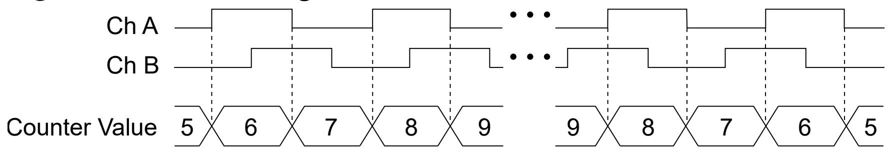
Figure 52. X1 Encoding



X2 Encoding

The same behavior holds for X2 encoding except the counter increments or decrements on each edge of channel A, depending on which channel leads the other. Each cycle results in two increments or decrements, as shown in the following figure.

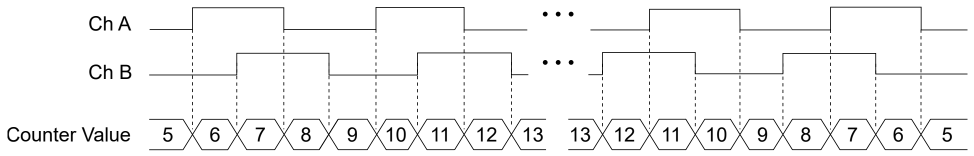
Figure 53. X2 Encoding



X4 Encoding

Similarly, the counter increments or decrements on each edge of channels A and B for X4 encoding. Whether the counter increments or decrements depends on which channel leads the other. Each cycle results in four increments or decrements, as shown in the following figure.

Figure 54. X4 Encoding



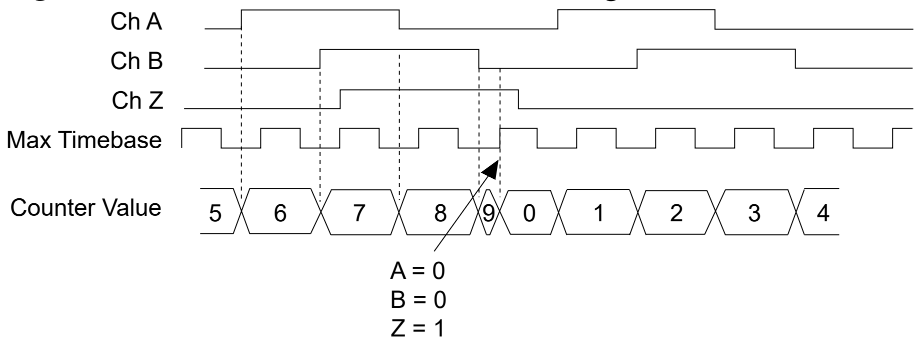
Channel Z Behavior

Some quadrature encoders have a third channel, channel Z, which is also referred to as the index channel. A high level on channel Z causes the counter to be reloaded with a specified value in a specified phase of the quadrature cycle. You can program the counter reload to occur in any one of the four phases in a quadrature cycle.

Channel Z behavior—when it goes high and how long it stays high—differs with quadrature encoder designs. You must refer to the documentation for your quadrature encoder to obtain timing of channel Z with respect to channels A and B. You must then ensure that channel Z is high during at least a portion of the phase you specify for reload. For instance, in Figure 7-21, channel Z is never high when channel A is high and channel B is low. Thus, the reload must occur in some other phase.

In the following figure, the reload phase is when both channel A and channel B are low. The reload occurs when the phase is true and channel Z is high. Incrementing and decrementing takes priority over reloading. Thus, when the channel B goes low to enter the reload phase, the increment occurs first. The reload occurs within one maximum timebase period after the reload phase becomes true. After the reload occurs, the counter continues to count as before. The following figure illustrates channel Z reload with X4 decoding.

Figure 55. Channel Z Reload with X4 Decoding

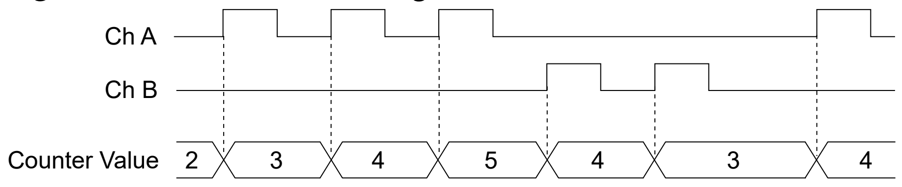


Measurements Using Two Pulse Encoders

The counter supports two pulse encoders that have two channels—channels A and B.

The counter increments on each rising edge of channel A. The counter decrements on each rising edge of channel B, as shown in the following figure.

Figure 56. Measurements Using Two Pulse Encoders



Two-Signal Edge-Separation Measurement

Two-signal edge-separation measurement is similar to pulse-width measurement, except that there are two measurement signals—Aux and Gate.

An active edge on the Aux input starts the counting and an active edge on the Gate input stops the counting. You must arm a counter to begin a two edge separation measurement.

After the counter has been armed and an active edge occurs on the Aux input, the counter counts the number of rising (or falling) edges on the Source. The counter ignores additional edges on the Aux input.

The counter stops counting upon receiving an active edge on the Gate input. The counter stores the count in the FIFO.

You can configure the rising or falling edge of the Aux input to be the active edge. You can configure the rising or falling edge of the Gate input to be the active edge.

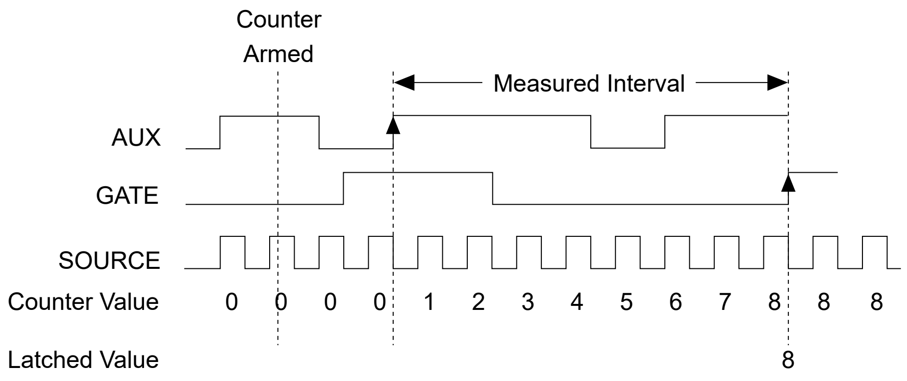
Use this measurement type to count events or measure the time that occurs between edges on two signals. This type of measurement is sometimes referred to as start/stop trigger measurement, second gate measurement, or A-to-B measurement.

Single Two-Signal Edge-Separation Measurement

With single two-signal edge-separation measurement, the counter counts the number of rising (or falling) edges on the Source input occurring between an active edge of the Gate signal and an active edge of the Aux signal.

The counter then stores the count in the FIFO and ignores other edges on its inputs. Software then reads the stored count. The following figure shows an example of a single two-signal edge-separation measurement.

Figure 57. Single Two-Signal Edge-Separation Measurement



Counter Output Applications

Refer to the following sections for more information on the various counter output applications available on the USB-6421.

Simple Pulse Generation

The USB-6421 supports the following methods of simple pulse generation.

Single Pulse Generation

The counter can output a single pulse. The pulse appears on the Counter ***n*** Internal

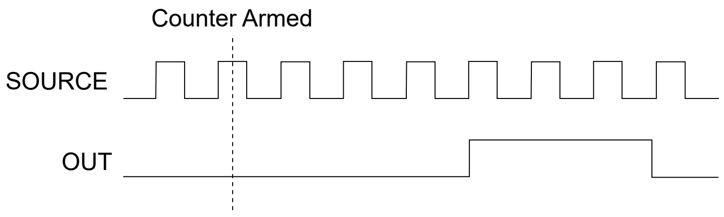
Output signal of the counter.

You can specify a delay from when the counter is armed to the beginning of the pulse. The delay is measured in terms of a number of active edges of the Source input.

You can specify a pulse width. The pulse width is also measured in terms of a number of active edges of the Source input. You can also specify the active edge of the Source input (rising or falling).

The following figure shows a generation of a pulse with a pulse delay of four and a pulse width of three (using the rising edge of Source).

Figure 58. Single Pulse Generation



Single Pulse Generation with Start Trigger

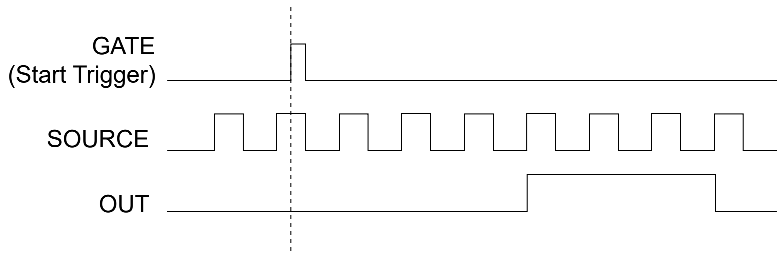
The counter can output a single pulse in response to one pulse on a hardware Start Trigger signal. The pulse appears on the Counter ***n*** Internal Output signal of the counter.

You can route the Start Trigger signal to the Gate input of the counter. You can specify a delay from the Start Trigger to the beginning of the pulse. You can also specify the pulse width. The delay and pulse width are measured in terms of a number of active edges of the Source input.

After the Start Trigger signal pulses once, the counter ignores the Gate input.

The following figure shows a generation of a pulse with a pulse delay of four and a pulse width of three (using the rising edge of Source).

Figure 59. Single Pulse Generation with Start Trigger



Pulse Train Generation

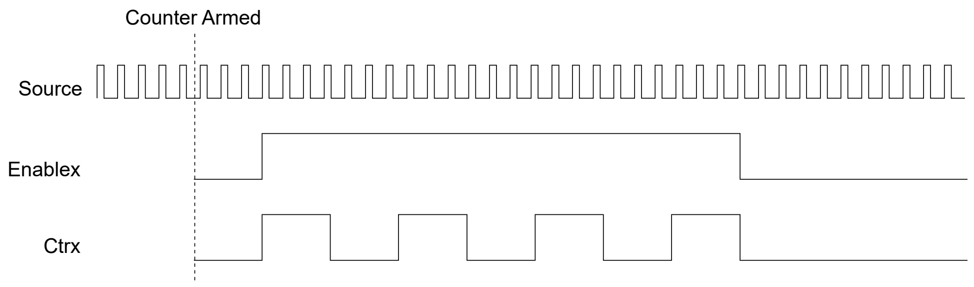
The USB-6421 supports the following methods of pulse train generation.

Finite Pulse Train Generation

Finite pulse train generation creates a train of pulses with programmable frequency and duty cycle for a predetermined number of pulses

With USB-6421 counters, the primary counter generates the specified pulse train and the embedded counter counts the pulses generated by the primary counter. When the embedded counter reaches the specified tick count, it generates a trigger that stops the primary counter generation. The following figure shows an example of finite pulse train generation.

Figure 69. Finite Pulse Train Generation: Four Ticks Initial Delay, Four Pulses

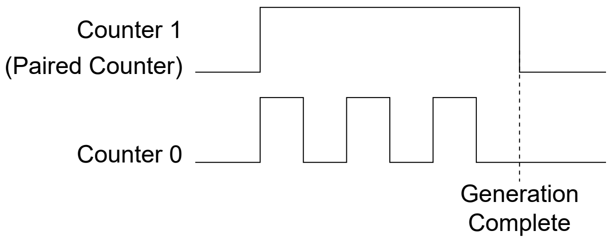


In Legacy Mode, the counter operation requires two counters and does not use the embedded counter. For example, to generate four pulses on Counter 0, Counter 0 generates the pulse train, which is gated by the paired second counter. The paired counter, Counter 1, generates a pulse of desired width.

Note Counter 0 is always paired with Counter 1. Counter 2 is always paired with Counter 3.

The routing is done internally. The following figure shows an example finite pulse train timing diagram.

Figure 61. Finite Pulse Train Timing in Legacy Mode



Retriggerable Pulse or Pulse Train Generation

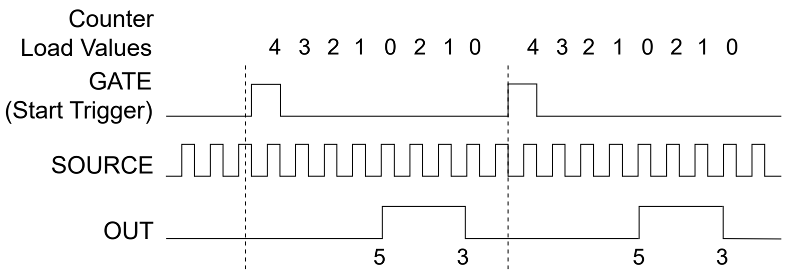
The counter can output a single pulse or multiple pulses in response to each pulse on a hardware Start Trigger signal. The generated pulses appear on the Counter ***n*** Internal Output signal of the counter.

You can route the Start Trigger signal to the Gate input of the counter. You can specify a delay from the Start Trigger to the beginning of each pulse. You can also specify the pulse width. The delay and pulse width are measured in terms of a number of active edges of the Source input. The initial delay can be applied to only the first trigger or to all triggers using the CO.EnableInitalDelayOnRetrigger property. The default for a single pulse is True, while the default for finite pulse trains is False.

The counter ignores the Gate input while a pulse generation is in progress. After the pulse generation is finished, the counter waits for another Start Trigger signal to begin another pulse generation.

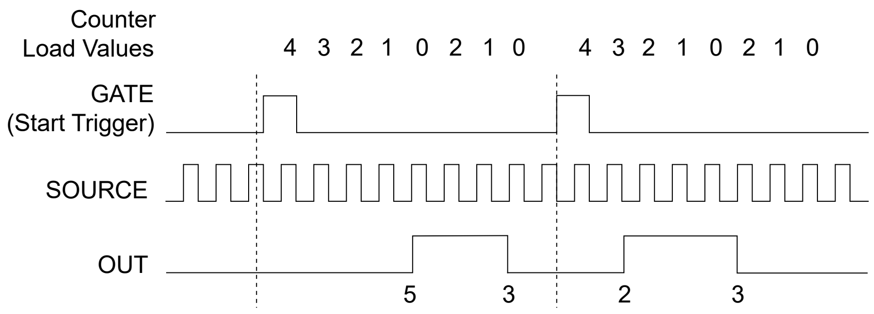
The following figure shows a generation of two pulses with a pulse delay of five and a pulse width of three (using the rising edge of Source) with CO.EnableInitalDelayOnRetrigger set to the default True.

Figure 62. Retriggerable Single Pulse Generation with Initial Delay on Retrigger



The following figure shows the same pulse train with CO.EnableInitalDelayOnRetrigger set to the default False.

Figure 63. Retriggerable Single Pulse Generation with Initial Delay on Retrigger Set to False



The minimum time between the trigger and the first active edge is two ticks of the source.

Continuous Pulse Train Generation

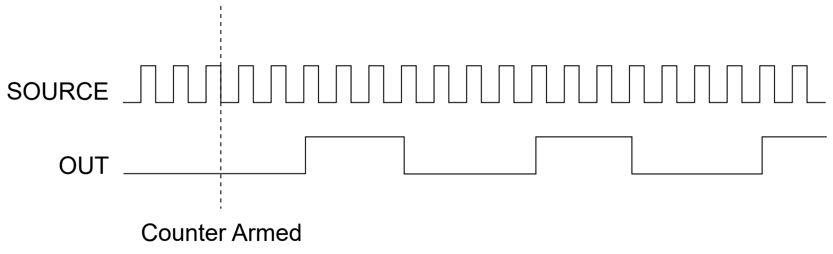
Continuous pulse train generation creates a train of pulses with programmable frequency and duty cycle. The pulses appear on the Counter ***n*** Internal Output signal of the counter.

You can specify a delay from when the counter is armed to the beginning of the pulse train. The delay is measured in terms of a number of active edges of the Source input.

You specify the high and low pulse widths of the output signal. The pulse widths are also measured in terms of a number of active edges of the Source input. You can also specify the active edge of the Source input (rising or falling).

The counter can begin the pulse train generation as soon as the counter is armed, or in response to a hardware Start Trigger. You can route the Start Trigger to the Gate input of the counter.

Figure 64. Continuous Pulse Train Generation



Continuous pulse train generation is sometimes called frequency division. If the high and low pulse widths of the output signal are M and N periods, then the frequency of the Counter ***n*** Internal Output signal is equal to the frequency of the Source input divided by M + N.

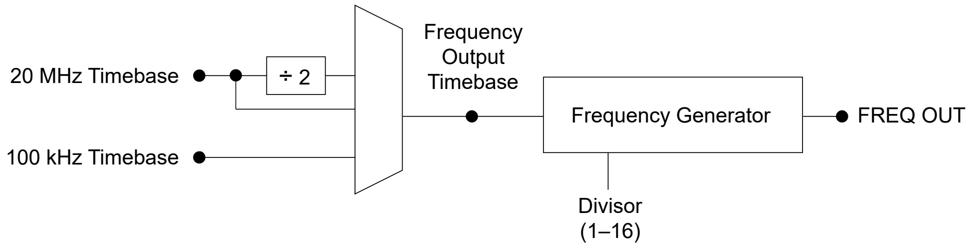
Frequency Generation

You can generate a frequency by using a counter in pulse train generation mode or by using the frequency generator circuit.

The frequency generator can output a square wave at many different frequencies. The frequency generator is independent of the four general-purpose 32-bit counter/timer modules on the USB-6421.

The following figure shows a block diagram of the frequency generator.

Figure 65. Frequency Generator Block Diagram

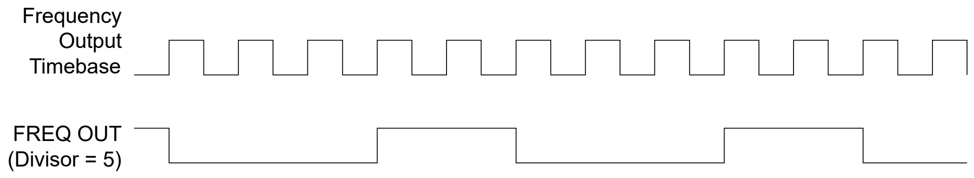


The frequency generator generates the Frequency Output signal. The Frequency Output signal is the Frequency Output Timebase divided by a number you select from 1 to 16. The Frequency Output Timebase can be either the 20 MHz Timebase, the 20 MHz Timebase divided by 2, or the 100 kHz Timebase.

The duty cycle of Frequency Output is 50% if the divider is either 1 or an even number. For an odd divider, suppose the divider is set to D. In this case, Frequency Output is low for (D + 1)/2 cycles and high for (D - 1)/2 cycles of the Frequency Output Timebase.

The following figure shows the output waveform of the frequency generator when the divider is set to 5.

Figure 66. Frequency Generator Output Waveform



Frequency Output can be routed out to any DIO <0..15> or RTSI <0..7> terminal. All DIO terminals are set to high-impedance at startup. The FREQ OUT signal can also be routed to many internal timing signals.

In software, program the frequency generator as you would program one of the counters for pulse train generation.

Frequency Division

The counters can generate a signal with a frequency that is a fraction of an input signal. This function is equivalent to continuous pulse train generation.

Pulse Generation for ETS

In the equivalent time sampling (ETS) application, the counter produces a pulse on the output a specified delay after an active edge on Gate.

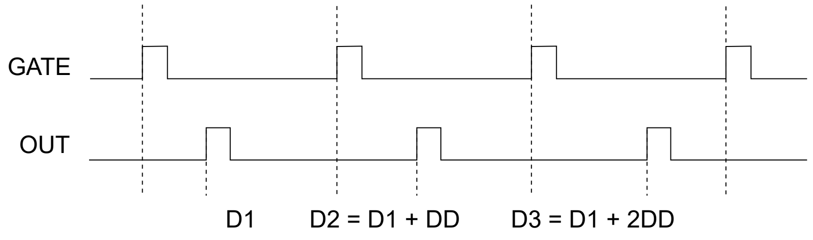
After each active edge on Gate, the counter cumulatively increments the delay between the Gate and the pulse on the output by a specified amount. Thus, the delay between the Gate and the pulse produced successively increases.

The increase in the delay value can be between 0 and 255. For instance, if you specify the increment to be 10, the delay between the active Gate edge and the pulse on the output increases by 10 every time a new pulse is generated.

Suppose you program your counter to generate pulses with a delay of 100 and pulse width of 200 each time it receives a trigger. Furthermore, suppose you specify the delay increment to be 10. On the first trigger, your pulse delay is 100, on the second it is 110, on the third it is 120; the process repeats until the counter is disarmed. The counter ignores any Gate edge that is received while the pulse triggered by the previous Gate edge is in progress.

The waveform thus produced at the counter's output can be used to provide timing for under sampling applications where a digitizing system can sample repetitive waveforms that are higher in frequency than the Nyquist frequency of the system. The following figure shows an example of pulse generation for ETS; the delay from the trigger to the pulse increases after each subsequent Gate active edge.

Figure 67. Pulse Generation for ETS



Counter Timing Signals

The USB-6421 features the following counter timing signals.

In this section, ***n*** refers to the USB-6421 Counter 0, 1, 2, or 3. For example, Counter ***n*** Source refers to four signals—Counter 0 Source (the source input to Counter 0), Counter 1 Source (the source input to Counter 1), Counter 2 Source (the source input to Counter 2), or Counter 3 Source (the source input to Counter 3).

Counter *n* Source Signal

The selected edge of the Counter ***n*** Source signal increments and decrements the counter value depending on the application the counter is performing.

The following table lists how the terminal is used in various applications.

Table 23. Counter Applications and Counter ***n*** Source

Application	**Purpose of Source Terminal**
Pulse Generation	Counter Timebase
One Counter Time Measurements	Counter Timebase
Two Counter Time Measurements	Input Terminal
Non-Buffered Edge Counting	Input Terminal

Application	Purpose of Source Terminal
Two-Edge Separation	Counter Timebase

Routing a Signal to Counter *n* Source

Each counter has independent input selectors for the Counter ***n*** Source signal. Any of the following signals can be routed to the Counter ***n*** Source input:

- 100 MHz Timebase
- 20 MHz Timebase
- 100 kHz Timebase
- DIO <0..15>
- Change Detection Event

In addition, TC or Gate from a counter can be routed to a different counter source.

Some of these options may not be available in some driver software.

Routing Counter *n* Source to an Output Terminal

You can route Counter ***n*** Source out to any DIO <0..15> terminal. All PFIs are set to high-impedance at startup.

Counter *n* Gate Signal

The Counter ***n*** Gate signal can perform many different operations depending on the application including starting and stopping the counter, and saving the counter contents.

Routing a Signal to Counter *n* Gate

Each counter has independent input selectors for the Counter ***n*** Gate signal. Any of the following signals can be routed to the Counter ***n*** Gate input:

- DIO <0..15>
- AI Start Trigger (ai/StartTrigger)

- AO Sample Clock (ao/SampleClock)
- DI Sample Clock (di/SampleClock)
- DO Sample Clock (do/SampleClock)
- Change Detection Event

In addition, a counter's Internal Output or Source can be routed to a different counter's gate.

Some of these options may not be available in some driver software.

Routing Counter *n* Gate to an Output Terminal

You can route Counter *n* Gate out to any DIO <0..15> terminal. All PFIs are set to high-impedance at startup.

Counter *n* Aux Signal

The Counter *n* Aux signal indicates the first edge in a two-signal edge-separation measurement.

Routing a Signal to Counter *n* Aux

Each counter has independent input selectors for the Counter *n* Aux signal. Any of the following signals can be routed to the Counter *n* Aux input:

- DIO <0..15>
- AI Start Trigger (ai/StartTrigger)
- Change Detection Event

In addition, a counter's Internal Output, Gate or Source can be routed to a different counter's Aux. A counter's own gate can also be routed to its Aux input.

Some of these options may not be available in some driver software.

Counter *n* A, Counter *n* B, and Counter *n* Z Signals

Counter *n* B can control the direction of counting in edge counting applications. Use the A, B, and Z inputs to each counter when measuring quadrature encoders or measuring two pulse encoders.

Routing Signals to A, B, and Z Counter Inputs

Each counter has independent input selectors for each of the A, B, and Z inputs. DIO <0..15> signals can be routed to each input:

Routing Counter ***n*** Z Signal to an Output Terminal

You can route Counter ***n*** Z out to any DIO <0..15> terminal.

Counter ***n*** Up_Down Signal

Counter ***n*** Up_Down is another name for the Counter ***n*** B signal.

Counter ***n*** HW Arm Signal

The Counter ***n*** HW Arm signal enables a counter to begin an input or output function.

To begin any counter input or output function, you must first enable, or arm, the counter. In some applications, such as a buffered edge count, the counter begins counting when it is armed. In other applications, such as single pulse-width measurement, the counter begins waiting for the Gate signal when it is armed. Counter output operations can use the arm signal in addition to a start trigger.

Software can arm a counter or configure counters to be armed on a hardware signal. Software calls this hardware signal the Arm Start Trigger. Internally, software routes the Arm Start Trigger to the Counter ***n*** HW Arm input of the counter.

Routing Signals to Counter ***n*** HW Arm Input

Any of the following signals can be routed to the Counter ***n*** HW Arm input:

- DIO <0..15>
- AI Start Trigger (ai/StartTrigger)
- Change Detection Event

A counter's Internal Output can be routed to a different counter's HW Arm.

Some of these options may not be available in some driver software.

Counter *n* Sample Clock Signal

Use the Counter ***n*** Sample Clock (Ctr***n***SampleClock) signal to perform sample clocked acquisitions and generations.

You can specify an internal or external source for Counter ***n*** Sample Clock. You can also specify whether the measurement sample begins on the rising edge or falling edge of Counter ***n*** Sample Clock.

If the USB-6421 receives a Counter ***n*** Sample Clock when the FIFO is full, it reports an overflow error to the host software.

Using an Internal Source

To use Counter ***n*** Sample Clock with an internal source, specify the signal source and the polarity of the signal. The source can be any of the following signals:

- DI Sample Clock (di/SampleClock)
- DO Sample Clock (do/SampleClock)
- AI Sample Clock (ai/SampleClock)
- AI Convert Clock (ai/ConvertClock)
- AO Sample Clock (ao/SampleClock)
- DI Change Detection output

Several other internal signals can be routed to Counter ***n*** Sample Clock through internal routes. Refer to ***Device Routing in MAX*** for more information.

Using an External Source

You can route DIO <0..15> as Counter ***n*** Sample Clock.

You can sample data on the rising or falling edge of Counter ***n*** Sample Clock.

Routing Counter ***n*** Sample Clock to an Output Terminal

You can route Counter ***n*** Sample Clock out to any DIO <0..15> terminal. The DIO circuitry inverts the polarity of Counter ***n*** Sample Clock before driving the DIO terminal.

Counter ***n*** Internal Output and Counter ***n*** TC Signals

The Counter ***n*** Internal Output signal changes in response to Counter ***n*** TC.

The two software-selectable output options are pulse output on TC and toggle output on TC. The output polarity is software-selectable for both options.

With pulse or pulse train generation tasks, the counter drives the pulse(s) on the Counter ***n*** Internal Output signal. The Counter ***n*** Internal Output signal can be internally routed to be a counter/timer input or an “external” source for AI, AO, DI, or DO timing signals.

Routing Counter ***n*** Internal Output to an Output Terminal

You can route Counter ***n*** Internal Output to any DIO <0..15> terminal. All PFIs are set to high-impedance at startup.

Frequency Output Signal

The Frequency Output (FREQ OUT) signal is the output of the frequency output generator.

Counter Triggering

Counters supports the following triggering action.

Start Trigger

For counter output operations, a start trigger can be configured to begin a finite or continuous pulse generation. Once a continuous generation has triggered, the pulses continue to generate until you stop the operation in software. For finite generations, the specified number of pulses is generated and the generation stops unless you use the retriggerable attribute. When you use this attribute, subsequent start triggers cause the generation to restart.

When using a start trigger, the start trigger source is routed to the Counter n Gate signal input of the counter.

Counter input operations can use the arm start trigger to have start trigger-like behavior.

Cascading Counters

You can internally route the Counter ***n*** Internal Output and Counter ***n*** TC signals of each counter to the Gate inputs of the other counter. By cascading two counters together, you can effectively create a 64-bit counter.

By cascading counters, you can also enable other applications. For example, to improve the accuracy of frequency measurements, use reciprocal frequency measurement, as described in the frequency generation ***Large Range of Frequencies with Two Counters*** section.

Related concepts:

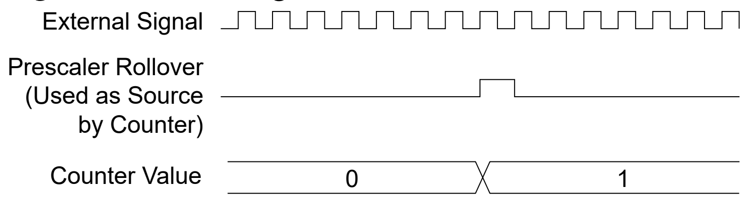
- Large Range of Frequencies with Two Counters

Counter Signal Prescaling

Prescaling allows the counter to count a signal that is faster than the maximum timebase of the counter. The USB-6421 offers 8X and 2X prescaling on each counter (prescaling can be disabled).Each prescaler consists of a small, simple counter that counts to eight (or two) and rolls over. This counter can run faster than the larger counters, which simply count the rollovers of this smaller counter. Thus, the prescaler acts as a frequency divider on the Source and puts out a frequency that is one-eighth (or one-half) of what it is accepting.

The following figure illustrates prescaling.

Figure 68. Prescaling



Prescaling is intended to be used for frequency measurement where the measurement is made on a continuous, repetitive signal. The prescaling counter cannot be read; therefore, you cannot determine how many edges have occurred since the previous rollover. Prescaling can be used for event counting provided it is acceptable to have an error of up to seven (or one) ticks. Prescaling can be used when the counter Source is an external signal. Prescaling is not available if the counter Source is one of the internal timebases (100 MHz timebase, 20 MHz timebase, or 100k Hz timebase).

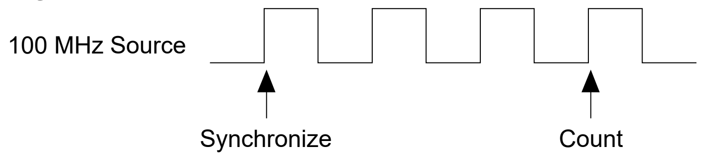
Counter Signal Synchronization Modes

The 32-bit counter counts up or down synchronously with the Source signal. The Gate signal and other counter inputs are asynchronous to the Source signal, so the USB-6421 synchronizes these signals before presenting them to the internal counter.

100 MHz Source Mode

In 100 MHz source mode, the USB-6421 synchronizes signals on the rising edge of the source, and counts on the third rising edge of the source. Edges are pipelined so no counts are lost, as shown in the following figure.

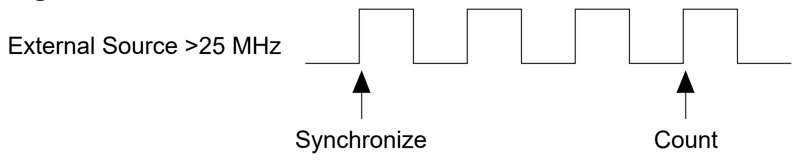
Figure 69. 100 MHz Source Mode



External Source Greater than 25 MHz

With an external source greater than 25 MHz, the USB-6421 synchronizes signals on the rising edge of the source, and counts on the third rising edge of the source. Edges are pipelined so no counts are lost, as shown in the following figure.

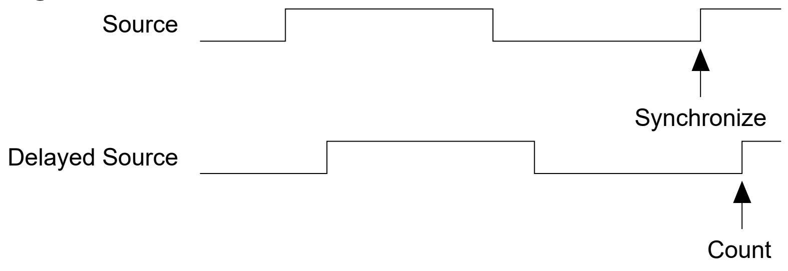
Figure 70. External Source Greater than 25 MHz



External or Internal Source Less than 25 MHz

With an external or internal source less than 25 MHz, the USB-6421 generates a delayed Source signal by delaying the Source signal by several nanoseconds. The USB-6421 synchronizes signals on the rising edge of the delayed Source signal, and counts on the following rising edge of the source, as shown in the following figure.

Figure 71. External or Internal Source Less than 25 MHz



Installing the USB-6421

Complete the following steps to install the USB-6421.

1. Unpacking the Kit
2. Installing Software
   Before installing the USB-6421, you must install the software you plan to use with it.
3. Wiring the USB-6421
4. Verifying the Installation
   Before using the USB-6421, verify that it is installed correctly through Hardware Configuration Utility or MAX.

Unpacking the Kit

1. Remove the device from the package and inspect the device for loose components or any other sign of damage.

   Notice Never touch the exposed pins of connectors.

   Note Do not install a device if it appears damaged in any way.

2. Unpack any other items and documentation from the kit.

Installing Software

Before installing the USB-6421, you must install the software you plan to use with it.

1. Install your application software, such as LabVIEW or FlexLogger.
2. Install NI-DAQmx.

   Note FlexLogger installs NI-DAQmx by default if you install it with the recommended settings. Therefore, you might not need to install NI-DAQmx separately if you installed FlexLogger as your application software.

Related reference:

- USB-6421 Driver Support

Related information:

- Download LabVIEW
- Download FlexLogger
- Download NI-DAQmx

Wiring the USB-6421

What to Use

- 0.14 mm^2 to 1.5 mm^2 (26 AWG to 16 AWG) copper conductor wire with 10 mm(0.394 in.) of insulation stripped from the end

What to Do

Refer to the following table for how to insert a wire into a terminal depending on what type of wire you are using or if you are using a ferrule.

Option	Description
When using a solid wire or stranded wire with a ferrule	Push the wire into the terminal when using a solid wire or stranded wire with a ferrule
When using a stranded wire without a ferrule	Press the push button and then push the wire into the terminal

Note You must use 2-wire ferrules to create a secure connection when connecting more than one wire to a single terminal.

Verifying the Installation

Before using the USB-6421, verify that it is installed correctly through Hardware Configuration Utility or MAX.

Verifying the Installation in Hardware Configuration Utility

NI recommends using Hardware Configuration Utility to perform and to validate initial hardware configuration.

1. Open Hardware Configuration Utility.
   The USB-6421 appears in the system pane automatically.
2. Record the name that Hardware Configuration Utility assigns to the USB-6421 or provide a custom name.
   Use this name when programming the USB-6421.
3. Validate that your instrument is installed correctly: select the USB-6421 module in the system pane, expand the **Troubleshooting** area of the configuration pane, and click **Self-test**.
   Hardware Configuration Utility reports when it has validated the hardware setup.

Verifying the Installation in MAX

To configure your NI hardware, use Measurement & Automation Explorer (MAX). MAX informs other programs about the NI hardware products in the system and their hardware configuration. MAX is automatically installed with NI-DAQmx.

Note MAX is not available on Linux.

1. Launch MAX.
2. In the configuration tree, expand **Devices and Interfaces** to see the list of installed NI hardware.

   Note If you do not see the device in the list, press **<F5>** to refresh the list of installed devices. If the device is still not listed, power off the system, ensure that the device is correctly installed, and restart.

3. Record the name MAX assigns to the hardware. Use this identifier when programming the USB-6421.
4. Self-test the hardware by selecting the item in the configuration tree and clicking **Self-Test** in the MAX toolbar.
   MAX self-test performs a basic verification of hardware resources.

What Do I Do If the USB-6421 Does Not Appear in Hardware Configuration Utility or MAX?

1. Check if you must refresh the connection between the hardware and the software.

Software	Description
Hardware Configuration Utility	Click the refresh button (↻).
MAX	a. In the MAX configuration tree, expand **Devices and Interfaces**. b. To see the list of installed hardware, expand the **Chassis** tree and press **F5** to refresh the list.

2. If the USB-6421 is still not listed, complete the following steps.
   a. Power off the system.
   b. Ensure that all hardware is correctly installed.
   c. Check that the USB cable is intact and fully inserted. If using a USB hub, validate connections and power to the hub.
   d. Restart the system.

What Should I Do If the USB-6421 Fails the Self-Test?

1. Reset the USB-6421 through Hardware Configuration Utility or MAX and then perform the self-test again.
2. Restart the system, and then perform the self-test again.
3. Unplug and re-plug the USB Type-C cable from the computer.
4. Perform the self-test again.

Note If the module fails the self-test again, contact NI or visit ni.com/support for further troubleshooting information.

Mounting the USB-6421

You can mount the USB-6421 in several ways.

Refer to the following sections for more information on mounting options for the USB-6421.

Mounting the USB-6421 on a DIN Rail

You can mount the USB-6421 on a DIN rail vertically or horizontally.

The mounting kit you need depends on which orientation you plan to mount the USB-6421.

Table 24. USB-6421 Mounting Kit Options

Mounting Orientation	Mounting Kit	Part Number
Horizontal, connectors facing upward or downward	USB-64xx Mounting Kit for DIN Rail	789986-01
Vertical, connectors facing outward	USB-64xx Mounting Kit for DIN Rail, Wall, or Panel Mount	789955-01

Related reference:

- Part Numbers for Recommended Accessories

Mounting the USB-6421 on a DIN Rail Horizontally

You can mount the USB-6421 on a DIN rail with the USB-64***xx*** Mounting Kit for DIN Rail.

What to Use

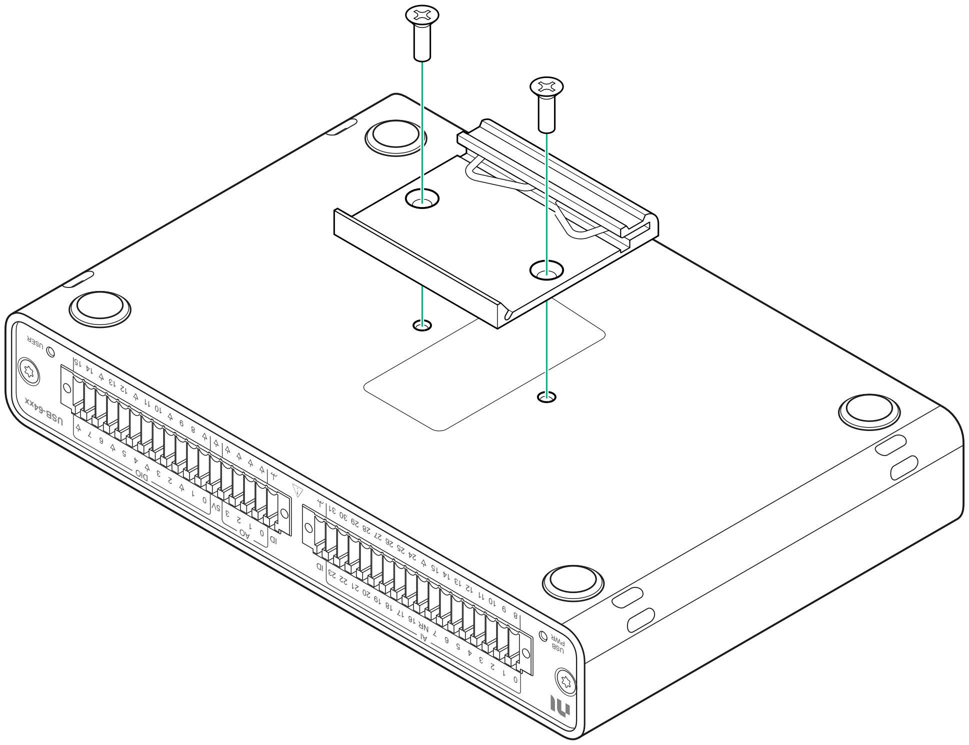
- USB-6421
- USB-64xx Mounting Kit for DIN Rail (P/N 789986-01)
  - DIN rail clip
  - #6-32 × 5/16 in. screws (x2)

- #2 Phillips screwdriver

What to Do

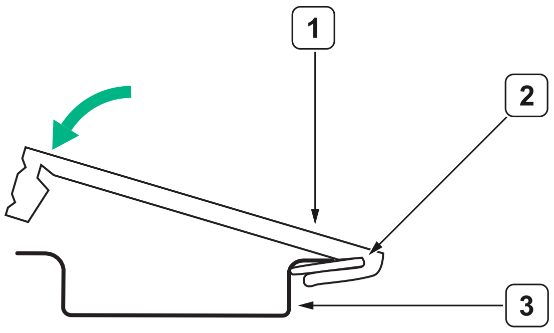
1. Fasten the DIN rail clip to the bottom of the USB-6421 using two #6-32 × 5/16 in. screws with a number #2 Phillips screwdriver.

   Figure 72. USB-6421 DIN Rail Clip Installation



2. Clip the USB-6421 onto the DIN rail with the larger lip of the DIN rail clip positioned up.

   Figure 73. DIN Rail Clip Parts



1. DIN rail clip
2. DIN rail spring
3. DIN rail

Mounting the USB-6421 on a DIN Rail Vertically

You can mount the USB-6421 on a DIN rail vertically with the USB-64***xx*** Mounting Kit for DIN Rail, Wall, or Panel Mount.

What to Use

- USB-6421
- USB-64xx Mounting Kit for DIN Rail, Wall, or Panel Mount (P/N 789955-01)
  - Mounting bracket
  - DIN rail clip
  - #6-32 × 5/16 in. Screws (x4)
  - Right-angle USB Type-C cable
- #2 Phillips screwdriver

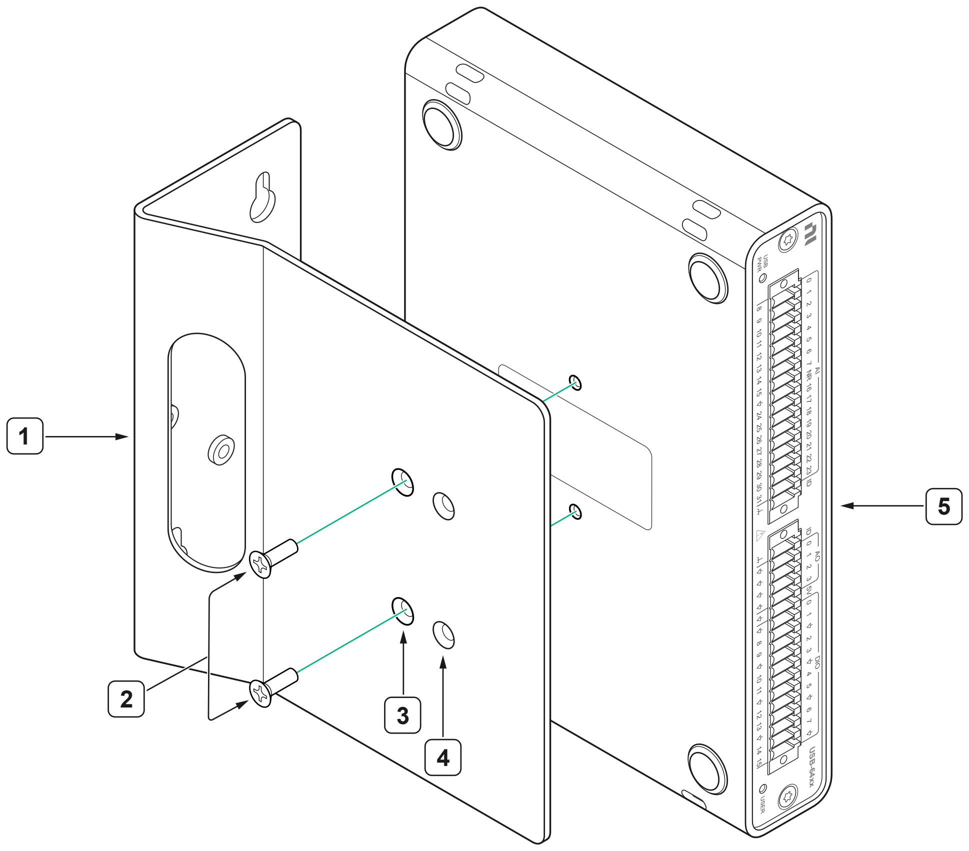
What to Do

1. Choose which set of mounting bracket screw holes to use.

Option	Description
Set A	Use these screw holes to position the USB-6421 as close to the wall as possible. **Note** The USB cable cannot be installed or removed after mounting when using these mounting holes.
Set B	Use these screw holes to position the USB-6421 slightly farther from the wall to allow clearance access to the USB cable after mounting.

2. Install the right-angle USB Type-C cable now if you are using the closest set of mounting screw holes (Set A). Connect the right-angle end of the USB Type-C cable to the USB-6421.
3. Fasten the mounting bracket to the USB-6421 using a #2 Phillips screwdriver and screws.

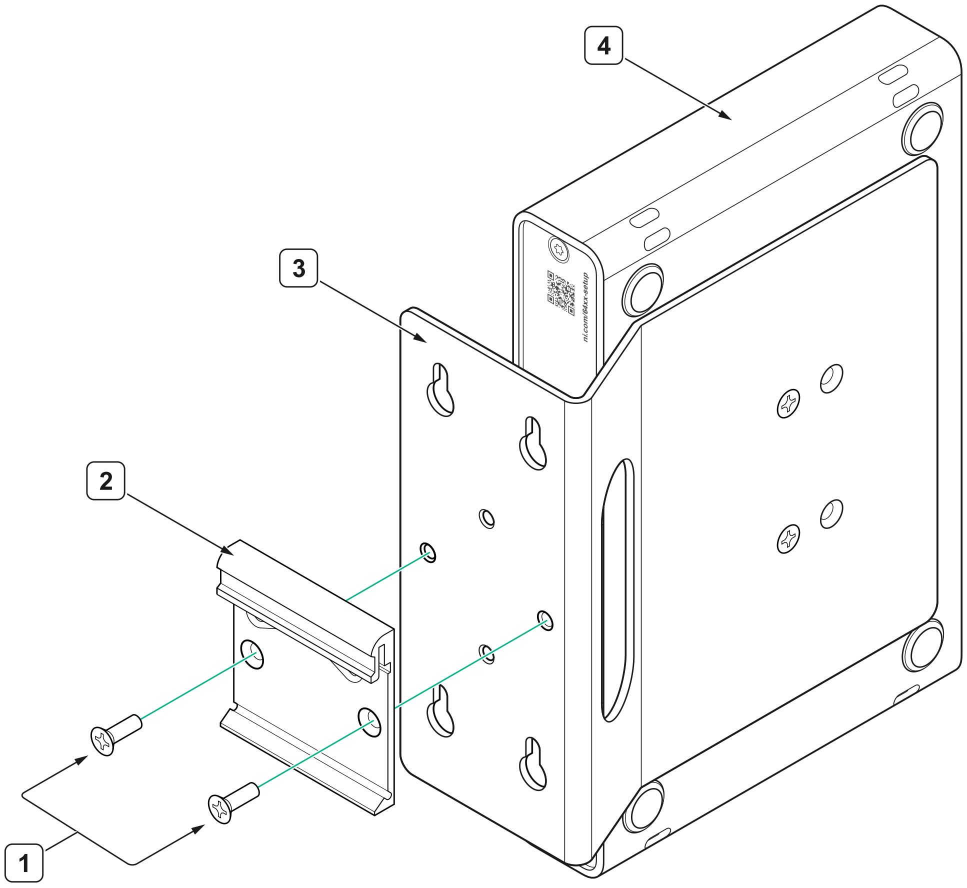
Figure 77. USB-6421 Mounting Bracket Installation



1. Mounting bracket
2. Screw
3. Mounting bracket screw holes set A
4. Mounting bracket screw holes set B
5. USB-6421

4. Fasten the DIN rail clip to the wall bracket using two #6-32 × 5/16 in. screws with a number #2 Phillips screwdriver.

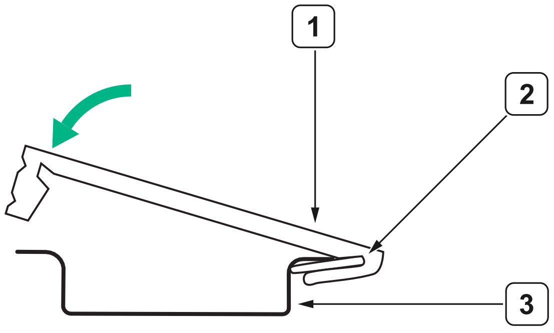
Figure 75. USB-6421 DIN Rail Clip Installation



1. Screws
2. DIN rail clip
3. Mounting bracket
4. USB-6421

5. Clip the USB-6421 onto the DIN rail with the larger lip of the DIN rail clip positioned up.

Figure 76. DIN Rail Clip Parts



1. DIN rail clip
2. DIN rail spring
3. DIN rail

Mounting the USB-6421 on a Wall or Panel

You can mount the USB-6421 on a wall or panel with the USB-64***xx*** Mounting Kit for DIN Rail, Wall, or Panel Mount.

What to Use

- USB-6421
- USB-64xx Mounting Kit for DIN Rail, Wall, or Panel Mount (P/N 789955-01)
  - Mounting bracket
  - #6-32 × 5/16 in. screws (x2)
  - Right-angle USB Type-C cable
- #2 Phillips screwdriver

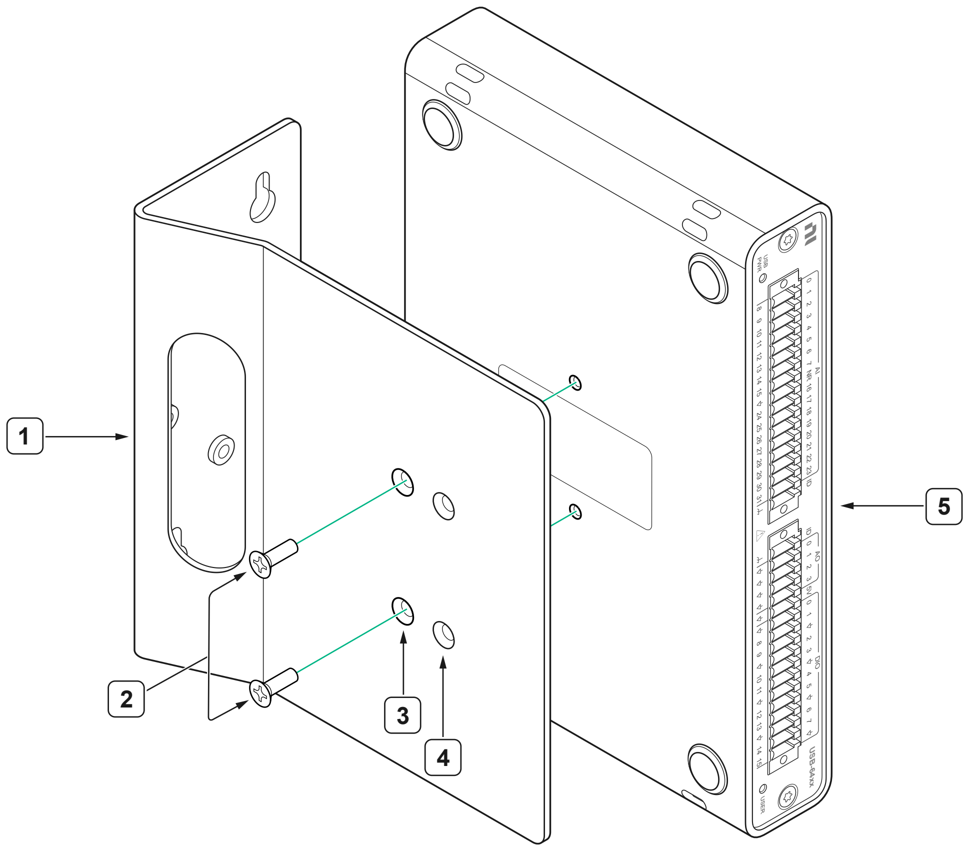
What to Do

1. Choose which set of mounting bracket screw to use.

Option	Description
Set A	Use these screw holes to position the USB-6421 as close to the wall as possible. **Note** The USB cable cannot be installed or removed after mounting when using these mounting holes.
Set B	Use these screw holes to position the USB-6421 slightly farther from the wall to allow clearance access to the USB cable after mounting.

2. Install the right-angle USB Type-C cable now if you are using the closest set of mounting screw holes (Set A). Connect the right-angle end of the USB Type-C cable to the USB-6421.
3. Fasten the mounting bracket to the USB-6421 using a #2 Phillips screwdriver and screws.

Figure 77. USB-6421 Mounting Bracket Installation



1. Mounting bracket
2. Screw
3. Mounting bracket screw holes set A
4. Mounting bracket screw holes set B
5. USB-6421

4. Fasten the mounting bracket to a wall or panel using the keyhole slots with M4 or 8-32 pan head screws. Position the screws on the wall using the dimensions below as a guide.

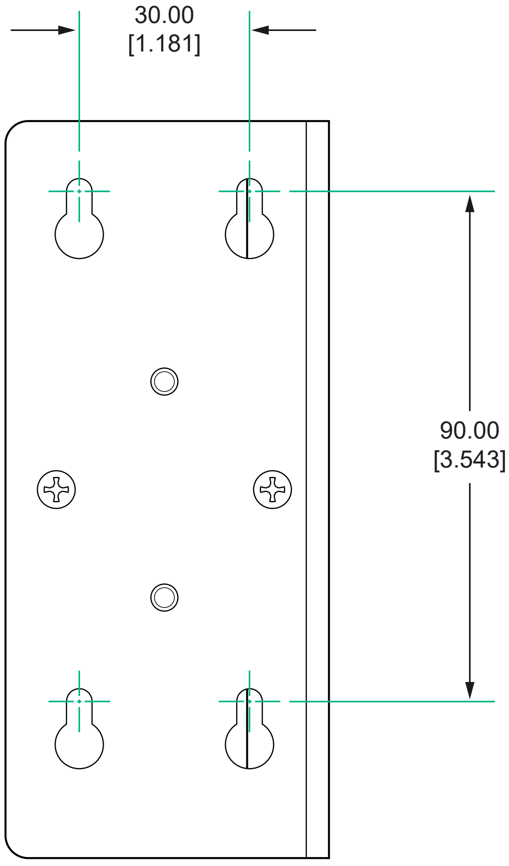
Related reference:

- Part Numbers for Recommended Accessories

USB-6421 Mounting Bracket Dimensions

The mounting bracket is included in the USB-64xx Mounting Kit for DIN Rail, Wall, or Panel Mount (P/N 789955-01).

Figure 78. USB-6421 Mounting Bracket Dimensions



Mounting the USB-6421 in a Rack

You can mount up to two USB-6421 devices in a 19-in. rack with the USB-64***xx*** Rack Mount Shelf.

What to Use

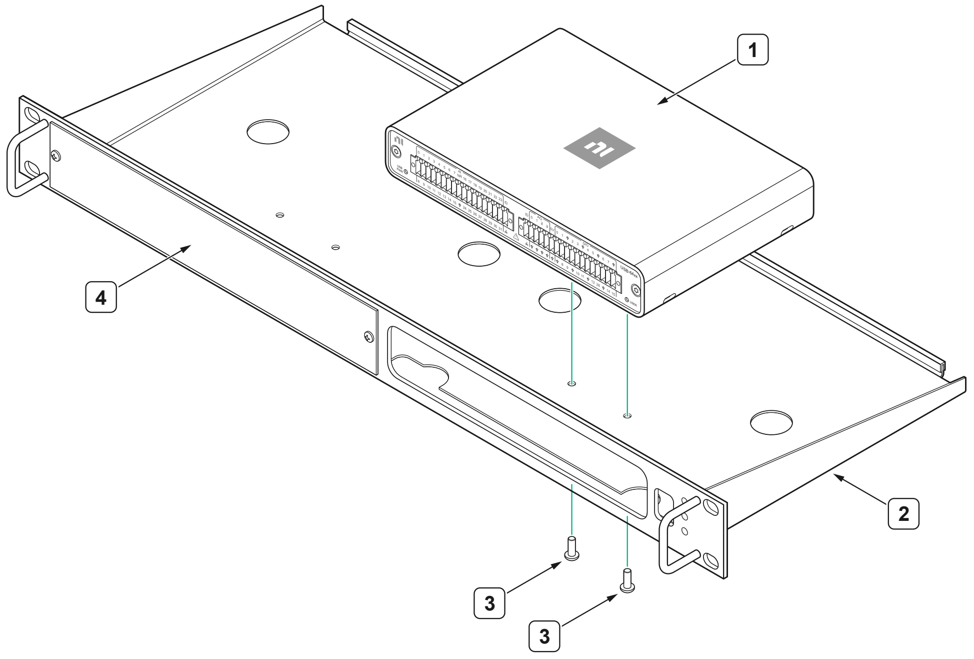
- USB-6421
- USB-64xx Rack Mount Shelf (P/N 789953-01)
  - Rack mount shelf
  - Screws (x2)

Note Two screws are required per device. You can mount up to two devices per shelf.

What to Do

1. Place the USB-6421 on the rack mount shelf.

   Figure 79. USB-6421 Rack Mount Shelf Installation

   1. USB-6421
   2. Rack mount shelf
   3. Screws
   4. Removable filler panel
2. Insert two screws through the bottom of the rack mount shelf and tighten with a screwdriver to fasten the USB-6421 to the rack mount shelf.
3. If you are mounting more than one USB-6421, remove the filler panel by loosening the screws with a screwdriver.
4. If your USB connection is on the front of the rack, feed the USB Type-C cable through the holes on the front panel.

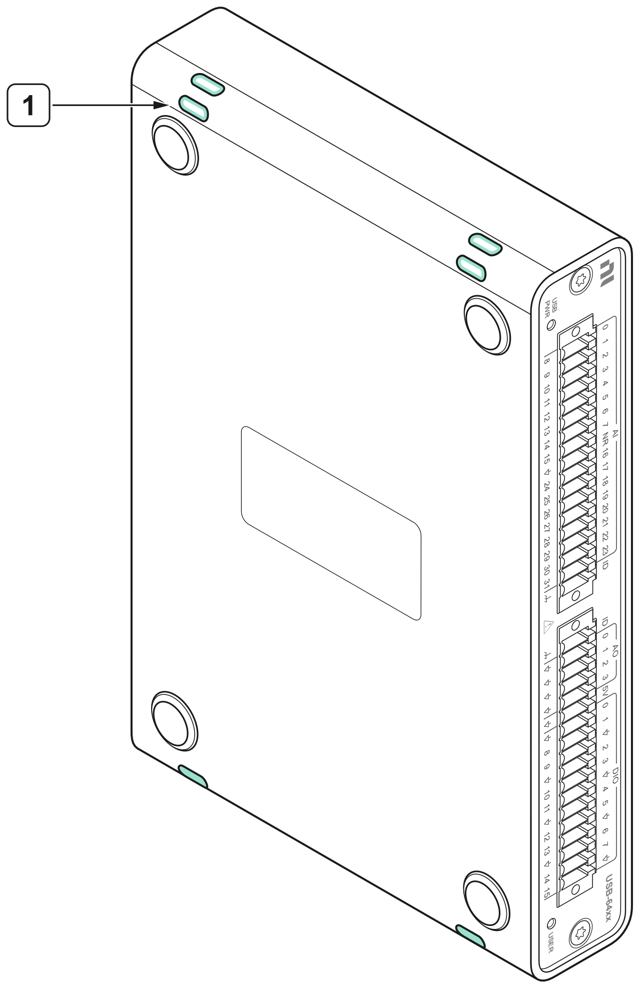
Related reference:

- Part Numbers for Recommended Accessories

Mounting the USB-6421 with Zip Ties

The USB-6421 has four holes that allow you to use zip ties to mount it.

Figure 80. USB-6421 Zip Tie Holes Location



1. Holes for zip ties

What to Use

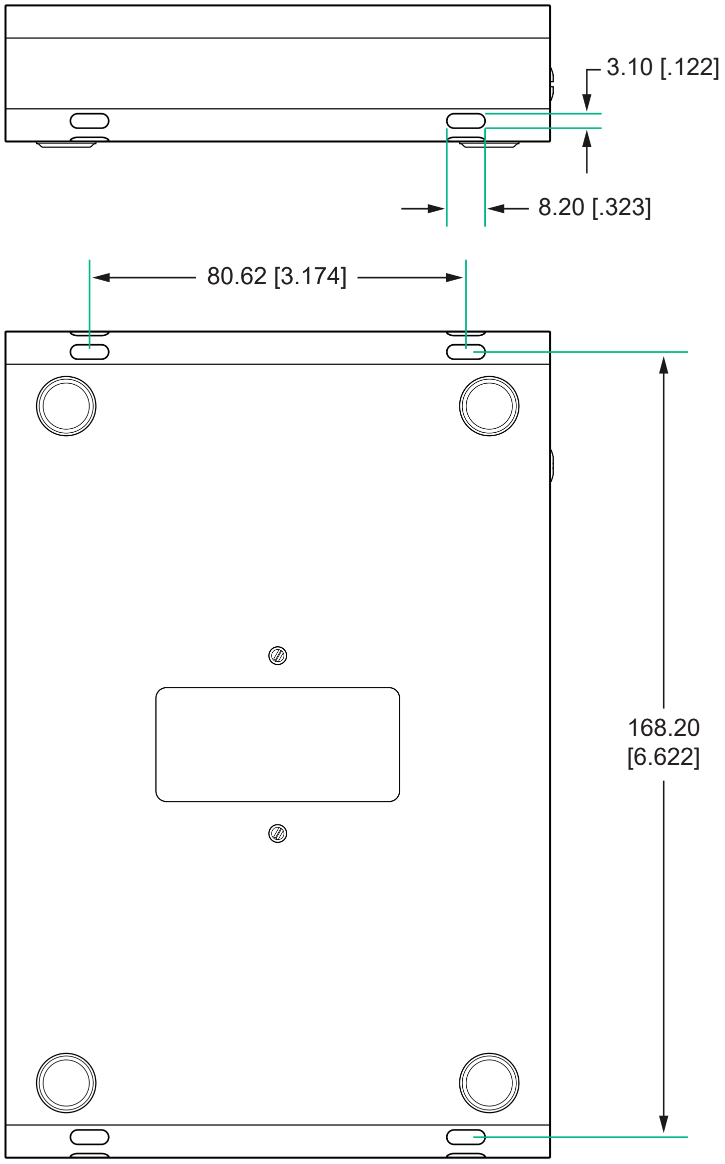
- USB-6421
- Zip ties

What to Do

1. Place the USB-6421.
2. Secure the USB-6421 by threading zip ties through the holes and fastening them.

USB-6421 Zip Tie Holes Dimensions

Figure 81. USB-6421 Zip Tie Holes Dimensions



Migrating from Previous Products

The USB-6421 shares core I/O capabilities and NI-DAQmx programming API with previous generations of NI multifunction I/O products such as M Series and X Series. In many cases, you can run existing application on the USB-6421 with no or minimal modifications.

The following steps explain the general process for porting an application.

1. Review specifications and channel count to ensure that the USB-6421 provides enough analog input, analog output, or digital input/output for your application.
2. Re-wire the system to connect the USB-6421 to your I/O.
3. Upgrade NI-DAQmx on your system to a version that supports the USB-6421.
4. Plug in the USB-6421 and run your application using the USB-6421.
5. Identify and fix any issues.

Designing for Migration

The USB-6421 input multiplexers provide a lot of flexibility when configuring AI terminals. However, simultaneously-sampled mioDAQ devices, such as the USB-6451, have more restrictions on the AI Terminal Configuration and channel scanlist.

Follow these rules to enable migration to simultaneously-sampled mioDAQ devices:

- Only include a channel a single time in a given scan.
- When a specific channel is configured for differential (DIFF) mode, do not use the positive or negative inputs from that same channel for referenced single-ended (RSE) or non-referenced single-ended (NRSE) measurements.
- When performing single-ended measurements, pick a common measurement reference for all channels, either RSE or NRSE.