# NI-DAQmx Python API Documentation

Release 0.5.0

**National Instruments** 

# API Reference:

1	Abou	ut	3
2	Feat	ures	5
3	Insta	allation	7
4	Usag	ge	9
5	Supp	oort / Feedback	11
	D.,	/ Footone Domosto	12
6	6.1	S/Feature Requests Information to Include When Asking for Help	13 13
7	Addi	itional Documentation	15
8	Lice		17
	8.1	nidaqmx.constants	17
	8.2	nidaqmx.errors	51
	8.3	nidaqmx.scale	52
	8.4	nidaqmx.stream_readers	56
	8.5	nidaqmx.stream_writers	77 91
	8.6	nidaqmx.system	91
		± •	98
		1 7	04
			07
		1	09
	8.7		12
			18
		*	56
		8.7.3 nidaqmx.task.export_signals	216
		8.7.4 nidaqmx.task.in_stream	220
		8.7.5 nidaqmx.task.out_stream	226
		8.7.6 nidaqmx.task.timing	229
		8.7.7 nidaqmx.task.triggers	236
	8.8	nidaqmx.types	
	8.9	nidaqmx.utils	:49

9	Indices and Tables	251
Pyt	thon Module Index	253

Info	Contains a Python API for interacting with NI-DAQmx. See GitHub for the latest source.
Author	National Instruments

API Reference: 1

2 API Reference:

**About** 

The **nidaqmx** package contains an API (Application Programming Interface) for interacting with the NI-DAQmx driver. The package is implemented in Python. This package was created and is supported by NI. The package is implemented as a complex, highly object-oriented wrapper around the NI-DAQmx C API using the ctypes Python library.

**nidaqmx** 0.5 supports all versions of the NI-DAQmx driver that ships with the C API. The C API is included in any version of the driver that supports it. The **nidaqmx** package does not require installation of the C header files.

Some functions in the **nidaqmx** package may be unavailable with earlier versions of the NI-DAQmx driver. Visit the ni.com/downloads to upgrade your version of NI-DAQmx.

nidaqmx supports only the Windows operating system.

nidaqmx supports CPython 2.7, 3.4+, PyPy2, and PyPy3.

4 Chapter 1. About

# **Features**

The following represents a non-exhaustive list of supported features for **nidaqmx**:

- Fully-object oriented
- Fully-featured Task class
- Fully-featured Scale class
- Fully-featured System sub-package with System, Device, PhysicalChannel, WatchdogTask, etc. classes
- NI-DAQmx Events
- NI-DAQmx Streams
- Enums support in both Python 2 and 3
- · Exceptions support
- Warnings support
- Collections that emulate Python container types
- Single, dynamic read and write methods (see *Usage*)
- Performant, NumPy-based reader and writer classes
- Optional parameters
- Implicitly verified properties
- Context managers

The following features are not yet supported by the **nidaqmx** package:

- · Calibration methods
- Real-time methods

6 Chapter 2. Features

# Installation

Running **nidaqmx** requires NI-DAQmx or NI-DAQmx Runtime. Visit the ni.com/downloads to download the latest version of NI-DAQmx.

**nidaqmx** can be installed with pip:

\$ python -m pip install nidaqmx

Or easy\_install from setuptools:

\$ python -m easy\_install nidaqmx

You also can download the project source and run:

\$ python setup.py install

Usage

The following is a basic example of using an nidaqmx.task.Task object. This example illustrates how the single, dynamic nidaqmx.task.Task.read() method returns the appropriate data type.

```
>>> import nidaqmx
>>> with nidaqmx.Task() as task:
       task.ai_channels.add_ai_voltage_chan("Dev1/ai0")
. . .
        task.read()
-0.07476920729381246
>>> with nidaqmx.Task() as task:
        task.ai_channels.add_ai_voltage_chan("Dev1/ai0")
        task.read(number_of_samples_per_channel=2)
[0.26001373311970705, 0.37796597238117036]
>>> from nidaqmx.constants import LineGrouping
>>> with nidagmx.Task() as task:
       task.di_channels.add_di_chan(
            "cDAQ2Mod4/port0/line0:1", line_grouping=LineGrouping.CHAN_PER_LINE)
        task.read(number_of_samples_per_channel=2)
. . .
[[False, True], [True, True]]
```

A single, dynamic nidaqmx.task.Task.write() method also exists.

```
5
```

Consider using the nidaqmx.stream\_readers and nidaqmx.stream\_writers classes to increase the performance of your application, which accept pre-allocated NumPy arrays.

Following is an example of using an nidagmx.system.System object.

```
>>> import nidaqmx.system
>>> system = nidaqmx.system.System.local()
>>> system.driver_version
DriverVersion(major_version=16L, minor_version=0L, update_version=0L)
>>> for device in system.devices:
       print (device)
Device (name=Dev1)
Device(name=Dev2)
Device (name=cDAQ1)
>>> import collections
>>> isinstance(system.devices, collections.Sequence)
True
>>> device = system.devices['Dev1']
>>> device == nidaqmx.system.Device('Dev1')
>>> isinstance(device.ai_physical_chans, collections.Sequence)
>>> phys_chan = device.ai_physical_chans['ai0']
>>> phys_chan
PhysicalChannel(name=Dev1/ai0)
>>> phys_chan == nidaqmx.system.PhysicalChannel('Dev1/ai0')
True
>>> phys_chan.ai_term_cfgs
[<TerminalConfiguration.RSE: 10083>, <TerminalConfiguration.NRSE: 10078>,
→<TerminalConfiguration.DIFFERENTIAL: 10106>]
>>> from enum import Enum
>>> isinstance(phys_chan.ai_term_cfgs[0], Enum)
True
```

10 Chapter 4. Usage

CF	łΔ	РΊ	ΓF	R	5
OI.	-		_	ıı	$\mathbf{\mathcal{U}}$

# Support / Feedback

The **nidaqmx** package is supported by NI. For support for **nidaqmx**, open a request through the NI support portal at ni.com.

# Bugs / Feature Requests

To report a bug or submit a feature request, please use the GitHub issues page.

# Information to Include When Asking for Help

Please include all of the following information when opening an issue:

- Detailed steps on how to reproduce the problem and full traceback, if applicable.
- The python version used:

```
$ python -c "import sys; print(sys.version)"
```

• The versions of the **nidaqmx**, numpy, six and enum34 packages used:

```
$ python -m pip list
```

- The version of the NI-DAQmx driver used. Follow this KB article to determine the version of NI-DAQmx you have installed.
- The operating system and version, for example Windows 7, CentOS 7.2, ...

				$\overline{}$
$\cap$	ΛГ	TC	$\Box$	
(, $\square$	Αı	<b>-</b> 1	П.	•

# **Additional Documentation**

Refer to the NI-DAQmx Help for API-agnostic information about NI-DAQmx or measurement concepts. NI-DAQmx Help installs only with the full version of NI-DAQmx.

License

**nidaqmx** is licensed under an MIT-style license (see LICENSE). Other incorporated projects may be licensed under different licenses. All licenses allow for non-commercial and commercial use.

# nidaqmx.constants

```
class nidaqmx.constants.ACExcitWireMode
    Bases: enum.Enum

FIVE_WIRE = 5
    5-wire.

FOUR_WIRE = 4
    4-wire.

SIX_WIRE = 6
    6-wire.

class nidaqmx.constants.ADCTimingMode
    Bases: enum.Enum
```

# AUTOMATIC = 16097

Uses the most appropriate supported timing mode based on the Sample Clock Rate.

#### BEST\_50\_HZ\_REJECTION = 14713

Improves 50 Hz noise rejection while decreasing noise rejection at other frequencies.

### BEST\_60\_HZ\_REJECTION = 14714

Improves 60 Hz noise rejection while decreasing noise rejection at other frequencies.

# CUSTOM = 10137

Use **ai\_adc\_custom\_timing\_mode** to specify a custom value controlling the tradeoff between speed and resolution.

#### HIGH RESOLUTION = 10195

Increases resolution and noise rejection while decreasing conversion rate.

#### HIGH SPEED = 14712

Increases conversion rate while decreasing resolution.

# class nidaqmx.constants.AOIdleOutputBehavior

Bases: enum. Enum

# HIGH IMPEDANCE = 12527

Set the channel to high-impedance, effectively disconnecting the analog output circuitry from the I/O connector.

# MAINTAIN\_EXISTING\_VALUE = 12528

Continue generating the current value.

#### ZERO VOLTS = 12526

Generate 0 V.

# class nidaqmx.constants.AOPowerUpOutputBehavior

Bases: enum. Enum

#### CURRENT = 10134

Current output.

#### HIGH IMPEDANCE = 12527

High-impedance state.

#### VOLTAGE = 10322

Voltage output.

# class nidaqmx.constants.AccelChargeSensitivityUnits

Bases: enum. Enum

# PICO\_COULOMBS\_PER\_G = 16099

PicoCoulombs per g.

# PICO\_COULOMBS\_PER\_INCHES\_PER\_SECOND\_SQUARED = 16101

PicoCoulombs per in/s^2.

#### PICO\_COULOMBS\_PER\_METERS\_PER\_SECOND\_SQUARED = 16100

PicoCoulombs per m/s^2.

#### class nidaqmx.constants.AccelSensitivityUnits

Bases: enum.Enum

# $M_VOLTS_PER_G = 12509$

mVolts/g.

# $VOLTS_PER_G = 12510$

Volts/g.

#### class nidaqmx.constants.AccelUnits

Bases: enum. Enum

#### FROM CUSTOM SCALE = 10065

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

#### G = 10186

1 g is approximately equal to 9.81 m/s/s.

# INCHES\_PER\_SECOND\_SQUARED = 12471

Inches per second per second.

# METERS\_PER\_SECOND\_SQUARED = 12470

Meters per second per second.

# class nidaqmx.constants.AcquisitionType

Bases: enum. Enum

#### CONTINUOUS = 10123

Acquire or generate samples until you stop the task.

#### FINITE = 10178

Acquire or generate a finite number of samples.

#### HW TIMED SINGLE POINT = 12522

Acquire or generate samples continuously using hardware timing without a buffer. Hardware timed single point sample mode is supported only for the sample clock and change detection timing types.

# class nidagmx.constants.Action

Bases: enum. Enum

#### CANCEL = 1

Cancel

#### COMMIT = 0

Commit

#### class nidaqmx.constants.ActiveLevel

Bases: enum. Enum

#### **ABOVE = 10093**

Pause the measurement or generation while the signal is above the threshold.

#### BELOW = 10107

Pause the measurement or generation while the signal is below the threshold.

# ${\bf class} \; {\tt nidaqmx.constants.} \\ {\bf ActiveOrInactiveEdgeSelection}$

Bases: enum. Enum

#### **ACTIVE = 14617**

Active edges.

#### INACTIVE = 14618

Inactive edges.

#### class nidaqmx.constants.AngleUnits

Bases: enum. Enum

#### DEGREES = 10146

Degrees.

#### FROM CUSTOM SCALE = 10065

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

# RADIANS = 10273

Radians.

### TICKS = 10304

Ticks.

#### class nidaqmx.constants.AngularVelocityUnits

Bases: enum. Enum

# DEGREES\_PER\_SECOND = 16082

Degrees per second.

#### FROM CUSTOM SCALE = 10065

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

#### RADIANS PER SECOND = 16081

Radians per second.

#### RPM = 16080

Revolutions per minute.

#### class nidagmx.constants.AutoZeroType

Bases: enum. Enum

#### EVERY SAMPLE = 10164

Perform an auto zero at every sample of the acquisition.

#### NONE = 10230

Do not perform an autozero.

#### ONCE = 10244

Perform an auto zero at the beginning of the acquisition. This auto zero task might not run if you have used DAQmx Control Task previously in your task.

#### class nidaqmx.constants.BreakMode

Bases: enum. Enum

#### BREAK BEFORE MAKE = 10110

When advancing to the next entry in the scan list, disconnect all previous connections before making any new connections.

#### NO ACTION = 10227

When advancing to the next entry in the scan list, leave all previous connections intact.

#### class nidagmx.constants.BridgeConfiguration

Bases: enum. Enum

# FULL BRIDGE = 10182

Sensor is a full bridge. If you set **ai\_excit\_use\_for\_scaling** to True, NI-DAQmx divides the measurement by the excitation value. Many sensors scale data to native units using scaling of volts per excitation.

#### HALF BRIDGE = 10187

Sensor is a half bridge. If you set **ai\_excit\_use\_for\_scaling** to True, NI-DAQmx divides the measurement by the excitation value. Many sensors scale data to native units using scaling of volts per excitation.

#### $NO_BRIDGE = 10228$

Sensor is not a Wheatstone bridge.

# QUARTER\_BRIDGE = 10270

Sensor is a quarter bridge. If you set **ai\_excit\_use\_for\_scaling** to True, NI-DAQmx divides the measurement by the excitation value. Many sensors scale data to native units using scaling of volts per excitation.

#### class nidagmx.constants.BridgeElectricalUnits

Bases: enum. Enum

# M\_VOLTS\_PER\_VOLT = 15897

Millivolts per volt.

#### VOLTS\_PER\_VOLT = 15896

Volts per volt.

#### class nidaqmx.constants.BridgePhysicalUnits

Bases: enum. Enum

# BAR = 15880

Bar.

#### FOOT POUNDS = 15884

Pound-feet.

#### INCH OUNCES = 15882

Ounce-inches.

# INCH POUNDS = 15883

Pound-inches.

#### KILOGRAM FORCE = 15877

kilograms-force.

#### NEWTONS = 15875

Newtons.

#### $NEWTON\_METERS = 15881$

Newton metres.

# PASCALS = 10081

Pascals.

#### POUNDS = 15876

Pounds.

# POUNDS\_PER\_SQ\_INCH = 15879

Pounds per square inch.

#### class nidagmx.constants.BridgeShuntCalSource

Bases: enum. Enum

#### BUILT IN = 10200

Use the internal shunt.

# USER PROVIDED = 10167

Use an external shunt.

### class nidagmx.constants.BridgeUnits

Bases: enum. Enum

# $FROM\_CUSTOM\_SCALE = 10065$

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

# $FROM\_TEDS = 12516$

Units defined by TEDS information associated with the channel.

# M\_VOLTS\_PER\_VOLT = 15897

Millivolts per volt.

#### VOLTS PER VOLTS = 15896

Volts per volt.

# class nidaqmx.constants.BusType

Bases: enum.Enum

#### $COMPACT_DAQ = 14637$

CompactDAQ.

#### PCI = 12582

PCI.

# PCIE = 13612

PCI Express.

PC CARD = 12585

```
PC Card/PCMCIA.
     PXI = 12583
         PXI.
     PXIE = 14706
         PXI Express.
     SCC = 14707
         SCC.
     SCXI = 12584
         SCXI.
     SWITCH_BLOCK = 15870
         SwitchBlock.
     TCPIP = 14828
         TCP/IP.
     UNKNOWN = 12588
         Unknown bus type.
     USB = 12586
         USB.
class nidagmx.constants.CJCSource
     Bases: enum. Enum
     BUILT IN = 10200
          Use a cold-junction compensation channel built into the terminal block.
     CONSTANT_USER_VALUE = 10116
          You must specify the cold-junction temperature.
     SCANNABLE_CHANNEL = 10113
          Use a channel for cold-junction compensation.
class nidaqmx.constants.CalibrationMode2
     Bases: enum. Enum
     CHARGE = 16105
         Charge
     VOLTAGE = 10322
          Voltage
{\bf class} \; {\tt nidaqmx.constants.CalibrationTerminalConfig}
     Bases: enum. Enum
     DIFF = 10106
         Differential
     PSEUDO_DIFF = 12529
         Pseudodifferential
class nidaqmx.constants.ChannelType
     Bases: enum.Enum
     ANALOG_INPUT = 10100
          Analog input channel.
```

#### ANALOG OUTPUT = 10102

Analog output channel.

#### COUNTER INPUT = 10131

Counter input channel.

#### COUNTER OUTPUT = 10132

Counter output channel.

#### DIGITAL INPUT = 10151

Digital input channel.

#### DIGITAL OUTPUT = 10153

Digital output channel.

#### class nidaqmx.constants.ChargeUnits

Bases: enum. Enum

#### COULOMBS = 16102

Coulombs.

#### FROM CUSTOM SCALE = 10065

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

#### PICO COULOMBS = 16103

PicoCoulombs.

#### class nidagmx.constants.ConstrainedGenMode

Bases: enum. Enum

#### FIXED 50 PERCENT DUTY CYCLE = 14711

Pulse duty cycle must be 50 percent. The frequency can change while the task runs.

# $FIXED_HIGH_FREQ = 14709$

Pulse frequency must be above 7.63 Hz and cannot change while the task runs. In this mode, the duty cycle has 8 bits of resolution.

#### $FIXED_LOW_FREQ = 14710$

Pulse frequency must be below 366.21 Hz and cannot change while the task runs. In this mode, the duty cycle has 16 bits of resolution.

# UNCONSTRAINED = 14708

Counter has no restrictions on pulse generation.

# class nidaqmx.constants.CountDirection

Bases: enum.Enum

#### COUNT DOWN = 10124

Decrement counter.

# $COUNT_UP = 10128$

Increment counter.

### EXTERNAL\_SOURCE = 10326

The state of a digital line controls the count direction. Each counter has a default count direction terminal.

#### class nidagmx.constants.CounterFrequencyMethod

Bases: enum. Enum

# DYNAMIC AVERAGING = 16065

Uses one counter and automatically configures the counter settings based on the range of frequencies to be measured. During the acquisition, the counter dynamically adjusts the number of periods that are averaged to balance measurement accuracy and latency.

#### HIGH FREQUENCY 2 COUNTERS = 10157

Use two counters, one of which counts pulses of the signal to measure during the specified measurement time.

# LARGE\_RANGE\_2\_COUNTERS = 10205

Use one counter to divide the frequency of the input signal to create a lower-frequency signal that the second counter can more easily measure.

#### LOW FREQUENCY 1 COUNTER = 10105

Use one counter that uses a constant timebase to measure the input signal.

#### class nidaqmx.constants.Coupling

Bases: enum. Enum

#### AC = 10045

Remove the DC offset from the signal.

#### DC = 10050

Allow NI-DAQmx to measure all of the signal.

#### GND = 10066

Remove the signal from the measurement and measure only ground.

# class nidaqmx.constants.CurrentShuntResistorLocation

Bases: enum. Enum

#### EXTERNAL = 10167

Use a shunt resistor external to the device. You must specify the value of the shunt resistor by using ai\_current\_shunt\_resistance.

#### INTERNAL = 10200

Use the built-in shunt resistor of the device.

# LET\_DRIVER\_CHOOSE = -1

# class nidaqmx.constants.CurrentUnits

Bases: enum. Enum

# AMPS = 10342

Amperes.

# $FROM_CUSTOM_SCALE = 10065$

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

# $FROM\_TEDS = 12516$

Units defined by TEDS information associated with the channel.

#### class nidagmx.constants.DataJustification

Bases: enum. Enum

# $\mathtt{LEFT} = 10209$

Samples occupy the higher bits of the integer.

#### RIGHT = 10279

Samples occupy the lower bits of the integer.

#### class nidagmx.constants.DataTransferActiveTransferMode

Bases: enum. Enum

# DMA = 10054

Direct Memory Access. Data transfers take place independently from the application.

#### INTERRUPT = 10204

Data transfers take place independently from the application. Using interrupts increases CPU usage because the CPU must service interrupt requests. Typically, you should use interrupts if the device is out of DMA channels.

#### POLLED = 10264

Data transfers take place when you call DAQmx Read or DAQmx Write.

#### USB BULK = 12590

Data transfers take place independently from the application using a USB bulk pipe.

#### class nidagmx.constants.DeassertCondition

Bases: enum. Enum

#### ONBOARD MEMORY CUSTOM THRESHOLD = 12577

Deassert the signal when the amount of space available in the onboard memory is below the value specified with **rdy\_for\_xfer\_event\_deassert\_cond\_custom\_threshold**.

#### ON BOARD MEMORY FULL = 10236

Deassert the signal when the onboard memory fills.

# ON\_BOARD\_MEMORY\_MORE\_THAN\_HALF\_FULL = 10237

Deassert the signal when more than half of the onboard memory of the device fills.

# class nidaqmx.constants.DigitalDriveType

Bases: enum. Enum

#### ACTIVE DRIVE = 12573

Drive the output pin to approximately 0 V for logic low and +3.3 V or +5 V, depending on the device, for logic high.

#### OPEN\_COLLECTOR = 12574

Drive the output pin to 0 V for logic low. For logic high, the output driver assumes a high-impedance state and does not drive a voltage.

### class nidagmx.constants.DigitalPatternCondition

Bases: enum. Enum

#### PATTERN DOES NOT MATCH = 10253

Trigger when the physical channels do not match the specified pattern.

#### PATTERN MATCHES = 10254

Trigger when the physical channels match the specified pattern.

# $class \verb| nidaqmx.constants.DigitalWidthUnits|\\$

Bases: enum. Enum

#### SAMPLE CLOCK PERIODS = 10286

Complete periods of the Sample Clock.

#### SECONDS = 10364

Seconds.

#### TICKS = 10304

Timebase ticks.

#### class nidagmx.constants.EddyCurrentProxProbeSensitivityUnits

Bases: enum. Enum

# IL = 14837

Volts/mil.

#### ILLIMETER = 14839

Volts/mMeter.

#### MICRON = 14840

mVolts/micron.

#### MIL = 14836

mVolts/mil.

#### MILLIMETER = 14838

mVolts/mMeter.

# class nidaqmx.constants.Edge

Bases: enum. Enum

#### FALLING = 10171

Falling edge(s).

#### RISING = 10280

Rising edge(s).

#### class nidagmx.constants.EncoderType

Bases: enum. Enum

#### TWO\_PULSE\_COUNTING = 10313

Two pulse counting.

#### x 1 = 10090

If signal A leads signal B, count the rising edges of signal A. If signal B leads signal A, count the falling edges of signal A.

#### $X_2 = 10091$

Count the rising and falling edges of signal A.

#### X 4 = 10092

Count the rising and falling edges of signal A and signal B.

#### class nidaqmx.constants.EncoderZIndexPhase

Bases: enum. Enum

### $AHIGH\_BHIGH = 10040$

Reset the measurement when signal A and signal B are high.

# AHIGH BLOW = 10041

Reset the measurement when signal A is high and signal B is low.

#### ALOW BHIGH = 10042

Reset the measurement when signal A is low and signal B high.

#### $ALOW_BLOW = 10043$

Reset the measurement when signal A and signal B are low.

# ${\bf class} \; {\tt nidaqmx.constants.EveryNSamplesEventType}$

 $Bases \colon \texttt{enum.Enum}$ 

#### ACQUIRED\_INTO\_BUFFER = 1

Acquired Into Buffer

# TRANSFERRED\_FROM\_BUFFER = 2

Transferred From Buffer

# class nidagmx.constants.ExcitationDCorAC

Bases: enum. Enum

#### USE AC = 10045

AC excitation.

#### USE DC = 10050

DC excitation.

# class nidaqmx.constants.ExcitationIdleOutputBehavior

Bases: enum. Enum

#### MAINTAIN EXISTING VALUE = 12528

Continue generating the current value.

#### ZERO\_VOLTS\_OR\_AMPERES = 12526

Drive excitation output to zero.

#### class nidagmx.constants.ExcitationSource

Bases: enum. Enum

#### EXTERNAL = 10167

Use an excitation source other than the built-in excitation source of the device. If you select this value, you must specify the amount of excitation.

#### INTERNAL = 10200

Use the built-in excitation source of the device. If you select this value, you must specify the amount of excitation.

#### NONE = 10230

Supply no excitation to the channel.

#### class nidaqmx.constants.ExcitationVoltageOrCurrent

Bases: enum. Enum

# USE CURRENT = 10134

Current excitation.

#### USE VOLTAGE = 10322

Voltage excitation.

### class nidaqmx.constants.ExportAction

Bases: enum. Enum

# INTERLOCKED = 12549

Handshake Event deasserts after the Handshake Trigger asserts, plus the amount of time specified with hshk\_event\_interlocked\_deassert\_delay.

#### LEVEL = 10210

The exported Sample Clock goes high at the beginning of the sample and goes low when the last AI Convert begins.

#### PULSE = 10265

Send a pulse to the terminal.

#### TOGGLE = 10307

Toggle the state of the terminal from low to high or from high to low.

#### class nidaqmx.constants.FillMode

Bases: enum. Enum

# $GROUP_BY_CHANNEL = 0$

Group by Channel

#### GROUP BY SCAN NUMBER = 1

Group by Scan Number

#### class nidaqmx.constants.FilterResponse

Bases: enum. Enum

#### BUTTERWORTH = 16076

Butterworth filter response.

#### CONSTANT GROUP DELAY = 16075

Constant group delay filter response.

#### ELLIPTICAL = 16077

Elliptical filter response.

#### $HARDWARE\_DEFINED = 10191$

Use the hardware-defined filter response.

#### class nidaqmx.constants.FilterType

Bases: enum. Enum

#### BANDPASS = 16073

Bandpass filter.

#### CUSTOM = 10137

Custom filter.

#### HIGHPASS = 16072

Highpass filter.

#### LOWPASS = 16071

Lowpass filter.

#### NOTCH = 16074

Notch filter.

# ${\bf class} \; {\tt nidaqmx.constants.} \\ {\bf ForceIEPES ensor Sensitivity Units}$

Bases: enum. Enum

### M\_VOLTS\_PER\_NEWTON = 15891

Millivolts per newton.

#### M\_VOLTS\_PER\_POUND = 15892

Millivolts per pound.

# class nidaqmx.constants.ForceUnits

Bases: enum. Enum

#### FROM CUSTOM SCALE = 10065

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

#### KILOGRAM FORCE = 15877

Kilograms-force.

#### NEWTONS = 15875

Newtons.

# POUNDS = 15876

Pounds.

#### class nidaqmx.constants.FrequencyUnits

Bases: enum. Enum

# $FROM\_CUSTOM\_SCALE = 10065$

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

#### HZ = 10373

Hertz.

#### TICKS = 10304

Timebase ticks.

# class nidaqmx.constants.FuncGenType

Bases: enum. Enum

#### SAWTOOTH = 14754

Sawtooth wave.

#### SINE = 14751

Sine wave.

#### SQUARE = 14753

Square wave.

#### TRIANGLE = 14752

Triangle wave.

#### class nidaqmx.constants.GpsSignalType

Bases: enum. Enum

#### IRIGB = 10070

Use the IRIG-B synchronization method. The GPS receiver sends one synchronization pulse per second, as well as information about the number of days, hours, minutes, and seconds that elapsed since the beginning of the current year.

#### NONE = 10230

Do not synchronize the counter to a GPS receiver. The timestamp measurement returns the number of seconds that elapsed since the device powered up unless you set **ci\_timestamp\_initial\_seconds**.

### PPS = 10080

Use the PPS synchronization method. The GPS receiver sends one synchronization pulse per second, but does not send any timing information. The timestamp measurement returns the number of seconds that elapsed since the device powered up unless you set **ci\_timestamp\_initial\_seconds**.

#### class nidaqmx.constants.HandshakeStartCondition

 $Bases \colon \texttt{enum.Enum}$ 

# ${\tt IMMEDIATE} = 10198$

Device is waiting for space in the FIFO (for acquisition) or waiting for samples (for generation).

#### WAIT FOR HANDSHAKE TRIGGER ASSERT = 12550

Device is waiting for the Handshake Trigger to assert.

#### WAIT FOR HANDSHAKE TRIGGER DEASSERT = 12551

Device is waiting for the Handshake Trigger to deassert.

# class nidaqmx.constants.Impedance1

Bases: enum.Enum

# $FIFTY\_OHMS = 50$

50 Ohms.

#### ONE M OHM = 1000000

1 M Ohm.

# SEVENTY\_FIVE\_OHMS = 75

75 Ohms.

```
TEN G OHMS = 100000000000
         10 G Ohm.
class nidaqmx.constants.InputCalSource
     Bases: enum. Enum
     GROUND = 2
         Ground
     LOOPBACK 0 = 0
         Loopback 0 degree shift
     LOOPBACK_180 = 1
         Loopback 180 degree shift
class nidaqmx.constants.InputDataTransferCondition
     Bases: enum. Enum
     ONBOARD_MEMORY_CUSTOM_THRESHOLD = 12577
         Transfer
                   data
                         from
                               the
                                      device
                                                                                             with
                                              when
                                                      the
                                                           number
                                                                         samples
                                                                                   specified
         ai data xfer custom threshold are in the device FIFO.
     ON_BOARD_MEMORY_MORE_THAN_HALF_FULL = 10237
         Transfer data from the device when more than half of the onboard memory of the device fills.
     ON BOARD MEMORY NOT EMPTY = 10241
         Transfer data from the device when there is data in the onboard memory.
     WHEN ACQUISITION COMPLETE = 12546
         Transfer data when the acquisition is complete.
class nidaqmx.constants.LVDTSensitivityUnits
     Bases: enum. Enum
     M VOLTS PER VOLT PER MILLIMETER = 12506
         mVolts/Volt/mMeter.
     M_VOLTS_PER_VOLT_PER_MILLI_INCH = 12505
         mVolts/Volt/0.001 Inch.
class nidaqmx.constants.Language
     Bases: enum. Enum
     CHS = 5
     DEU = 2
     ENG = 0
     FRA = 1
     JPN = 3
     KOR = 4
     RAW = -1
class nidaqmx.constants.LengthUnits
     Bases: enum. Enum
     FROM_CUSTOM_SCALE = 10065
         Units a custom scale specifies. If you select this value, you must specify a custom scale name.
```

30 Chapter 8. License

INCHES = 10379 Inches.

#### METERS = 10219

Meters.

#### TICKS = 10304

Ticks.

#### class nidagmx.constants.Level

Bases: enum. Enum

#### HIGH = 10192

Logic high.

#### LOW = 10214

Logic low.

#### NO CHANGE = 10160

Do not change the state of the lines. On some devices, you can select this value only for entire ports.

#### TRISTATE = 10310

High-impedance state. You can select this state only on devices with bidirectional lines. You cannot select this state for dedicated digital output lines. On some devices, you can select this value only for entire ports.

#### class nidaqmx.constants.LineGrouping

Bases: enum. Enum

#### CHAN FOR ALL LINES = 1

One Channel For All Lines

#### $CHAN_PER_LINE = 0$

One Channel For Each Line

#### class nidaqmx.constants.LoggingMode

Bases: enum. Enum

#### LOG = 15844

Enable logging for the task. You cannot read data using DAQmx Read when using this mode. If you require access to the data, read from the TDMS file.

#### $LOG_AND_READ = 15842$

Enable both logging and reading data for the task. You must use DAQmx Read to read samples for NI-DAQmx to stream them to disk.

#### OFF = 10231

Disable logging for the task.

# class nidaqmx.constants.LoggingOperation

Bases: enum. Enum

#### CREATE = 15848

Create a new TDMS file. If the file already exists, NI-DAQmx returns an error.

# CREATE\_OR\_REPLACE = 15847

Create a new TDMS file, or replace an existing TDMS file.

#### OPEN = 10437

Open an existing TDMS file, and append data to that file. If the file does not exist, NI-DAQmx returns an error.

# $OPEN_OR_CREATE = 15846$

Open an existing TDMS file, and append data to that file. If the file does not exist, NI-DAQmx creates a new TDMS file.

#### class nidaqmx.constants.LogicFamily

Bases: enum. Enum

#### FIVE V = 14619

Compatible with TTL and 5 V CMOS signals.

# THREE POINT THREE V = 14621

Compatible with LVTTL signals.

#### TWO POINT FIVE V = 14620

Compatible with 2.5 V CMOS signals.

# class nidaqmx.constants.LogicLvlBehavior

Bases: enum. Enum

#### NONE = 10230

Supply no excitation to the channel.

#### $PULL_UP = 16064$

High logic.

#### class nidagmx.constants.MIOAIConvertTimebaseSource

Bases: enum. Enum

#### $EIGHTY_M_HZ_TIMEBASE = 14636$

Use the onboard 80 MHz timebase.

# EIGHT M HZ TIMEBASE = 16023

Use the onboard 8 MHz timebase.

#### MASTER TIMEBASE = 10282

Use the same source as the Master Timebase.

# ONE\_HUNDRED\_M\_HZ\_TIMEBASE = 15857

Use the onboard 100 MHz timebase.

#### SAMPLE TIMEBASE = 10284

Use the same source as Sample Clock timebase.

#### TWENTY\_M\_HZ\_TIMEBASE = 12537

Use the onboard 20 MHz timebase.

# class nidaqmx.constants.ModulationType

Bases: enum.Enum

#### AM = 14756

Amplitude modulation.

#### FM = 14757

Frequency modulation.

#### NONE = 10230

No modulation.

# class nidagmx.constants.OutputDataTransferCondition

Bases: enum. Enum

#### ON BOARD MEMORY EMPTY = 10235

Transfer data to the device only when there is no data in the onboard memory of the device.

# ON\_BOARD\_MEMORY\_HALF\_FULL\_OR\_LESS = 10239

Transfer data to the device any time the onboard memory is less than half full.

#### ON BOARD MEMORY LESS THAN FULL = 10242

Transfer data to the device any time the onboard memory of the device is not full.

## class nidaqmx.constants.OverflowBehavior

Bases: enum. Enum

## GNORE OVERRUNS = 15863

NI-DAQmx ignores Sample Clock overruns, and the task continues to run.

#### TOP TASK AND ERROR = 15862

Stop task and return an error.

## class nidaqmx.constants.OverwriteMode

Bases: enum. Enum

## DO\_NOT\_OVERWRITE\_UNREAD\_SAMPLES = 10159

The acquisition stops when it encounters a sample in the buffer that you have not read.

## OVERWRITE\_UNREAD\_SAMPLES = 10252

When an acquisition encounters unread data in the buffer, the acquisition continues and overwrites the unread samples with new ones. You can read the new samples by setting **relative\_to** to **ReadRelativeTo.MOST\_RECENT\_SAMPLE** and setting **offset** to the appropriate number of samples.

## class nidaqmx.constants.PathCapability

Bases: enum. Enum

 $CHANNEL_IN_USE = 10434$ 

CHANNEL\_RESERVED\_FOR\_ROUTING = 10436

CHANNEL SOURCE CONFLICT = 10435

PATH\_ALREADY\_EXISTS = 10432

PATH\_AVAILABLE = 10431

PATH UNSUPPORTED = 10433

## class nidaqmx.constants.Polarity

Bases: enum. Enum

## $ACTIVE_HIGH = 10095$

High state is the active state.

## ACTIVE LOW = 10096

Low state is the active state.

## ${\bf class} \; {\tt nidaqmx.constants.PowerUpChannelType}$

Bases: enum. Enum

## $CHANNEL\_CURRENT = 1$

Current Channel

## CHANNEL\_HIGH\_IMPEDANCE = 2

High-Impedance Channel

#### CHANNEL VOLTAGE = 0

Voltage Channel

## class nidaqmx.constants.PowerUpStates

Bases: enum. Enum

## HIGH = 10192

Logic high.

#### LOW = 10214

Logic low.

#### TRISTATE = 10310

High-impedance state. You can select this state only on devices with bidirectional lines. You cannot select this state for dedicated digital output lines. On some devices, you can select this value only for entire ports.

#### class nidagmx.constants.PressureUnits

Bases: enum. Enum

#### BAR = 15880

Bar.

## $FROM_CUSTOM_SCALE = 10065$

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

#### PASCALS = 10081

Pascals.

## POUNDS\_PER\_SQ\_INCH = 15879

Pounds per square inch.

## class nidaqmx.constants.ProductCategory

Bases: enum. Enum

## $AO\_SERIES = 14647$

AO Series.

## $B\_SERIES\_DAQ = 14662$

B Series DAQ.

## $COMPACT_DAQ_CHASSIS = 14658$

CompactDAQ chassis.

### C SERIES MODULE = 14659

C Series I/O module.

## $DIGITAL_IO = 14648$

Digital I/O.

## DSA = 14649

Dynamic Signal Acquisition.

## E SERIES DAQ = 14642

E Series DAQ.

## $M_SERIES_DAQ = 14643$

M Series DAQ.

### $NETWORK_DAQ = 14829$

Network DAQ.

## NIELVIS = 14755

NI ELVIS.

#### SCC CONNECTOR BLOCK = 14704

SCC Connector Block.

## $SCC_MODULE = 14705$

SCC Module.

## SCXI MODULE = 14660

SCXI module.

#### SC EXPRESS = 15886

SC Express.

## $SC\_SERIES\_DAQ = 14645$

SC Series DAQ.

## SWITCHES = 14650

Switches.

#### S SERIES DAQ = 14644

S Series DAQ.

## TIO SERIES = 14661

TIO Series.

## UNKNOWN = 12588

Unknown category.

## USBDAQ = 14646

USB DAQ.

## X SERIES DAQ = 15858

X Series DAQ.

## class nidaqmx.constants.RTDType

Bases: enum. Enum

## CUSTOM = 10137

You must use ai\_rtd\_a, ai\_rtd\_b, and ai\_rtd\_c to supply the coefficients for the Callendar-Van Dusen equation.

## $PT_3750 = 12481$

Pt3750.

### PT 3851 = 10071

Pt3851.

#### PT 3911 = 12482

Pt3911.

## $PT_3916 = 10069$

Pt3916.

## PT 3920 = 10053

Pt3920.

## PT 3928 = 12483

Pt3928.

## class nidaqmx.constants.RVDTSensitivityUnits

Bases: enum. Enum

## M\_VPER\_VPER\_DEGREE = 12507

mVolts/Volt/Degree.

## M\_VPER\_VPER\_RADIAN = 12508

mVolts/Volt/Radian.

## class nidaqmx.constants.RawDataCompressionType

Bases: enum. Enum

## LOSSLESS PACKING = 12555

Remove unused bits from samples. No resolution is lost.

#### LOSSY LSB REMOVAL = 12556

Remove unused bits from samples. Then, if necessary, remove bits from samples until the samples are the size specified with **ai\_lossy\_lsb\_removal\_compressed\_samp\_size**. This compression type limits resolution to the specified sample size.

### NONE = 10230

Do not compress samples.

#### class nidaqmx.constants.ReadRelativeTo

Bases: enum. Enum

## CURRENT\_READ\_POSITION = 10425

Start reading samples relative to the last sample returned by the previous read. For the first read operation, this position is the first sample acquired or the first pretrigger sample if you configured a reference trigger for the task.

#### FIRST PRETRIGGER SAMPLE = 10427

Start reading samples relative to the first pretrigger sample. You specify the number of pretrigger samples to acquire when you configure a reference trigger.

#### FIRST SAMPLE = 10424

Start reading samples relative to the first sample acquired.

#### MOST RECENT SAMPLE = 10428

Start reading samples relative to the next sample acquired. For example, use this value and set **offset** to -1 to read the last sample acquired.

#### REFERENCE TRIGGER = 10426

Start reading samples relative to the first sample after the reference trigger occurred.

### class nidagmx.constants.RegenerationMode

Bases: enum. Enum

## ALLOW\_REGENERATION = 10097

Allow NI-DAQmx to regenerate samples that the device previously generated. When you choose this value, the write marker returns to the beginning of the buffer after the device generates all samples currently in the buffer.

### DONT ALLOW REGENERATION = 10158

Do not allow NI-DAQmx to regenerate samples the device previously generated. When you choose this value, NI-DAQmx waits for you to write more samples to the buffer or until the timeout expires.

## ${\bf class} \; {\tt nidaqmx.constants.RelayPosition}$

Bases: enum.Enum

CLOSED = 10438

OPEN = 10437

#### class nidagmx.constants.ResistanceConfiguration

Bases: enum.Enum

#### FOUR WIRE = 4

4-wire mode.

#### THREE WIRE = 3

3-wire mode.

## $TWO_WIRE = 2$

2-wire mode.

### class nidagmx.constants.ResistanceUnits

Bases: enum.Enum

#### FROM CUSTOM SCALE = 10065

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

#### FROM TEDS = 12516

Units defined by TEDS information associated with the channel.

### OHMS = 10384

Ohms.

## class nidaqmx.constants.ResistorState

Bases: enum. Enum

#### PULL DOWN = 15951

pull down state for pull up pull down resistors

#### PULL UP = 15950

pull up state for pull up/pull down resistors

## class nidaqmx.constants.ResolutionType

Bases: enum. Enum

#### BITS = 10109

Bits.

## class nidaqmx.constants.SCXI1124Range

Bases: enum. Enum

 $NEG_{10}TO_{10}V = 14634$ 

 $NEG_1_{TO_1_V} = 14632$ 

 $NEG_5_{TO_5_V} = 14633$ 

 $ZERO_TO_FIVE_V = 14630$ 

ZERO TO ONE V = 14629

 $ZERO_TO_TEN_V = 14631$ 

 $ZERO_TO_TWENTY_M_A = 14635$ 

## class nidaqmx.constants.SampClkOverrunBehavior

Bases: enum. Enum

## ${\tt REPEAT\_LAST\_SAMPLE} = 16062$

Repeat the last sample.

## RETURN SENTINEL VALUE = 16063

Return the sentinel value.

# ${\bf class} \; {\tt nidaqmx.constants.SampleInputDataWhen} \\$

Bases: enum. Enum

## HANDSHAKE\_TRIGGER\_ASSERTS = 12552

Latch data when the Handshake Trigger asserts.

## HANDSHAKE\_TRIGGER\_DEASSERTS = 12553

Latch data when the Handshake Trigger deasserts.

## class nidaqmx.constants.SampleTimingType

Bases: enum. Enum

## BURST\_HANDSHAKE = 12548

Determine sample timing using burst handshaking between the device and a peripheral device.

#### CHANGE DETECTION = 12504

Acquire samples when a change occurs in the state of one or more digital input lines. The lines must be contained within a digital input channel.

#### HANDSHAKE = 10389

Determine sample timing by using digital handshaking between the device and a peripheral device.

#### IMPLICIT = 10451

Configure only the duration of the task.

## $ON_DEMAND = 10390$

Acquire or generate a sample on each read or write operation. This timing type is also referred to as static or software-timed.

#### PIPELINED SAMPLE CLOCK = 14668

Device acquires or generates samples on each sample clock edge, but does not respond to certain triggers until a few sample clock edges later. Pipelining allows higher data transfer rates at the cost of increased trigger response latency. Refer to the device documentation for information about which triggers pipelining affects. This timing type allows handshaking with some devices using the Pause trigger, the Ready for Transfer event, or the Data Active event. Refer to the device documentation for more information.

#### SAMPLE CLOCK = 10388

Acquire or generate samples on the specified edge of the sample clock.

#### class nidagmx.constants.ScaleType

Bases: enum. Enum

#### LINEAR = 10447

Scale values by using the equation y=mx+b, where x is a prescaled value and y is a scaled value.

#### MAP RANGES = 10448

Scale values proportionally from a range of pre-scaled values to a range of scaled values.

### NONE = 10230

Do not scale electrical values to physical units.

#### POLYNOMIAL = 10449

Scale values by using an Nth order polynomial equation.

## $\mathtt{TABLE} = 10450$

Map a list of pre-scaled values to a list of corresponding scaled values, with all other values scaled proportionally.

### TWO POINT LINEAR = 15898

You provide two pairs of electrical values and their corresponding physical values. NI-DAQmx uses those values to calculate the slope and y-intercept of a linear equation and uses that equation to scale electrical values to physical values.

#### class nidaqmx.constants.ScanRepeatMode

Bases: enum.Enum

#### CONTINUOUS = 10117

The task returns to the beginning of the scan list when it reaches the end of the scan list.

#### FINITE = 10172

The task advances through the scan list one time only. NI-DAQmx ignores any Advance Triggers after completing the scan list.

## class nidaqmx.constants.Sense

 $Bases \colon \texttt{enum.Enum}$ 

```
LOCAL = 16095
         Local.
     REMOTE = 16096
         Remote.
class nidagmx.constants.ShuntCalSelect
     Bases: enum. Enum
     A = 12513
         Switch A.
     AAND B = 12515
         Switches A and B.
     B = 12514
         Switch B.
{\bf class} {\bf nidaqmx.constants.ShuntElementLocation}
     Bases: enum. Enum
     NONE = 10230
     R 1 = 12465
     R_2 = 12466
     R_3 = 12467
    R 4 = 14813
class nidaqmx.constants.ShuntResistorSelect
     Bases: enum. Enum
     A = 12513
         Α
    B = 12514
         В
class nidaqmx.constants.Signal
     Bases: enum. Enum
     ADVANCE TRIGGER = 12488
     ADV CMPLT EVENT = 12492
     AI_CONVERT_CLOCK = 12484
     AI HOLD CMPLT EVENT = 12493
     CHANGE DETECTION EVENT = 12511
         Timed Loop executes each time the Change Detection Event occurs.
     COUNTER_OUTPUT_EVENT = 12494
         Timed Loop executes each time the Counter Output Event occurs.
     REFERENCE TRIGGER = 12490
     SAMPLE CLOCK = 12487
         Timed Loop executes on each active edge of the Sample Clock.
     SAMPLE\_COMPLETE = 12530
```

Timed Loop executes each time the Sample Complete Event occurs.

START TRIGGER = 12491

#### TEN M HZ REF CLOCK = 12536

#### TWENTY M HZ TIMEBASE CLOCK = 12486

## WATCHDOG TIMER EXPIRED EVENT = 12512

## class nidaqmx.constants.SignalModifiers

Bases: enum. Enum

#### DO NOT INVERT POLARITY = 0

Do not invert polarity

## $INVERT_POLARITY = 1$

Invert polarity

### class nidaqmx.constants.Slope

Bases: enum. Enum

#### FALLING = 10171

Trigger on the falling slope of the signal.

## RISING = 10280

Trigger on the rising slope of the signal.

## class nidaqmx.constants.SoftwareTrigger

Bases: enum. Enum

## ADVANCE TRIGGER = 12488

Place holder enum to make editting internal enum easier.

#### class nidagmx.constants.SoundPressureUnits

Bases: enum. Enum

## $FROM\_CUSTOM\_SCALE = 10065$

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

## PA = 10081

Pascals.

#### class nidaqmx.constants.SourceSelection

Bases: enum. Enum

## EXTERNAL = 10167

External to the device.

#### INTERNAL = 10200

Internal to the device.

#### class nidagmx.constants.StrainGageBridgeType

Bases: enum. Enum

## FULL\_BRIDGE\_I = 10183

Four active gages with two pairs subjected to equal and opposite strains.

## FULL\_BRIDGE\_II = 10184

Four active gages with two aligned with maximum principal strain and two Poisson gages in adjacent arms.

#### FULL\_BRIDGE\_III = 10185

Four active gages with two aligned with maximum principal strain and two Poisson gages in opposite arms.

## $HALF_BRIDGE_I = 10188$

Two active gages with one aligned with maximum principal strain and one Poisson gage.

## HALF BRIDGE II = 10189

Two active gages with equal and opposite strains.

#### QUARTER BRIDGE I = 10271

Single active gage.

#### QUARTER BRIDGE II = 10272

Single active gage and one dummy gage.

## class nidaqmx.constants.StrainGageRosetteMeasurementType

Bases: enum. Enum

#### CARTESIAN SHEAR STRAIN XY = 15976

The tensile strain coplanar to the surface of the material under stress in the XY coordinate direction.

## CARTESIAN\_STRAIN\_X = 15974

The tensile strain coplanar to the surface of the material under stress in the X coordinate direction.

#### CARTESIAN\_STRAIN\_Y = 15975

The tensile strain coplanar to the surface of the material under stress in the Y coordinate direction.

## $MAX\_SHEAR\_STRAIN = 15977$

The maximum strain coplanar to the cross section of the material under stress.

#### MAX SHEAR STRAIN ANGLE = 15978

The angle at which the maximum shear strain of the rosette occurs.

#### PRINCIPAL STRAIN 1 = 15971

The maximum tensile strain coplanar to the surface of the material under stress.

## PRINCIPAL\_STRAIN\_2 = 15972

The minimum tensile strain coplanar to the surface of the material under stress.

#### PRINCIPAL STRAIN ANGLE = 15973

The angle at which the principal strains of the rosette occur.

## class nidaqmx.constants.StrainGageRosetteType

Bases: enum.Enum

#### DELTA = 15969

A delta rosette consists of three strain gages, each separated by a 60 degree angle.

### RECTANGULAR = 15968

A rectangular rosette consists of three strain gages, each separated by a 45 degree angle.

### TEE = 15970

A tee rosette consists of two gages oriented at 90 degrees with respect to each other.

## class nidaqmx.constants.StrainUnits

Bases: enum. Enum

#### FROM CUSTOM SCALE = 10065

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

#### STRAIN = 10299

Strain.

### class nidaqmx.constants.SwitchChannelUsage

Bases: enum. Enum

#### LOAD CHANNEL = 10440

You can use the channel only as the output for a signal passing through the switch.

## RESERVED\_FOR\_ROUTING\_CHANNEL = 10441

You can use the channel only to complete routes within a switch.

```
SOURCE CHANNEL = 10439
          You can use the channel only as an input for a signal.
class nidaqmx.constants.SyncType
     Bases: enum. Enum
     MASTER = 15888
         Device is the source for shared clocks and triggers.
     NONE = 10230
         Disables trigger skew correction.
     SLAVE = 15889
          Device uses clocks and triggers from the master device.
class nidaqmx.constants.TEDSUnits
     Bases: enum. Enum
     FROM_CUSTOM_SCALE = 10065
          Units a custom scale specifies. If you select this value, you must specify a custom scale name.
          Units defined by TEDS information associated with the channel.
class nidaqmx.constants.TaskMode
     Bases: enum. Enum
     TASK ABORT = 6
         Abort
     TASK COMMIT = 3
         Commit
     TASK RESERVE = 4
         Reserve
     TASK START = 0
         Start
     TASK STOP = 1
         Stop
     TASK UNRESERVE = 5
         Unreserve
     TASK VERIFY = 2
         Verify
class nidagmx.constants.TaskStringFormat
     Bases: enum. Enum
     INI = 0
     JSON = 2
     TAB_DELIMITED = 1
class nidagmx.constants.TemperatureUnits
     Bases: enum. Enum
     DEG C = 10143
         Degrees Celsius.
```

DEG F = 10144

Degrees Fahrenheit.

#### DEG R = 10145

Degrees Rankine.

#### FROM CUSTOM SCALE = 10065

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

## K = 10325

Kelvins.

## class nidaqmx.constants.TerminalConfiguration

Bases: enum. Enum

## DEFAULT = -1

Default.

#### DIFFERENTIAL = 10106

Differential.

## NRSE = 10078

Non-Referenced Single-Ended.

#### PSEUDODIFFERENTIAL = 12529

Pseudodifferential.

#### RSE = 10083

Referenced Single-Ended.

### class nidagmx.constants.ThermocoupleType

Bases: enum. Enum

### B = 10047

B-type thermocouple.

## $\mathtt{E}=10055$

E-type thermocouple.

## J = 10072

J-type thermocouple.

## K = 10073

K-type thermocouple.

### N = 10077

N-type thermocouple.

## R = 10082

R-type thermocouple.

## S = 10085

S-type thermocouple.

## T = 10086

T-type thermocouple.

## $class \; \verb| nidaqmx.constants.TimeUnits| \\$

Bases: enum. Enum

### FROM CUSTOM SCALE = 10065

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

## SECONDS = 10364

Seconds.

#### TICKS = 10304

Timebase ticks.

## class nidaqmx.constants.TorqueUnits

Bases: enum. Enum

## FOOT POUNDS = 15884

Pound-feet.

#### FROM CUSTOM SCALE = 10065

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

#### INCH OUNCES = 15882

Ounce-inches.

## INCH POUNDS = 15883

Pound-inches.

## $NEWTON\_METERS = 15881$

Newton meters.

## class nidaqmx.constants.TriggerType

Bases: enum. Enum

#### ANALOG EDGE = 10099

Trigger when an analog signal crosses a threshold.

#### ANALOG LEVEL = 10101

Pause the measurement or generation while an analog signal is above or below a level.

#### ANALOG WINDOW = 10103

Trigger when an analog signal enters or leaves a range of values.

## $DIGITAL\_EDGE = 10150$

Trigger on a rising or falling edge of a digital pulse.

#### DIGITAL LEVEL = 10152

Pause the measurement or generation while a digital signal is at either a high or low state.

## DIGITAL\_PATTERN = 10398

Pause the measurement or generation while digital physical channels either match or do not match a digital pattern.

#### INTERLOCKED = 12549

Use the Handshake Trigger as a control signal for asynchronous handshaking, such as 8255 handshaking.

#### NONE = 10230

Disable reference triggering for the task.

### SOFTWARE = 10292

Advance to the next entry in a scan list when you call DAQmx Send Software Trigger.

## class nidaqmx.constants.TriggerUsage

 $Bases \colon \texttt{enum.Enum}$ 

#### ADVANCE = 12488

Advance trigger.

## ARM\_START = 14641

Arm Start trigger.

## HANDSHAKE = 10389

Handshake trigger.

#### PAUSE = 12489

Pause trigger.

## REFERENCE = 12490

Reference trigger.

## **START**= 12491

Start trigger.

## class nidaqmx.constants.UnderflowBehavior

Bases: enum. Enum

## AUSE\_UNTIL\_DATA\_AVAILABLE = 14616

Pause the task until samples are available in the FIFO.

## HALT\_OUTPUT\_AND\_ERROR = 14615

Stop generating samples and return an error.

## class nidaqmx.constants.UnitsPreScaled

Bases: enum. Enum

#### AMPS = 10342

Amperes.

#### BAR = 15880

Bar.

## COULOMBS = 16102

Coulombs.

### DEGREES = 10146

Degrees.

## $DEGREES_PER_SECOND = 16082$

Degrees per second.

## $DEG_C = 10143$

Degrees Celsius.

## $DEG_F = 10144$

Degrees Fahrenheit.

## DEG R = 10145

Degrees Rankine.

## FOOT POUNDS = 15884

Pound-feet.

## FROM TEDS = 12516

Units defined by TEDS information associated with the channel.

#### G = 10186

1 g is approximately equal to 9.81 m/s/s.

## HERTZ = 10373

Hertz.

## INCHES = 10379

Inches.

## $INCHES\_PER\_SECOND = 15960$

Inches per second.

## INCHES\_PER\_SECOND\_SQUARED = 12471

Inches per second per second.

## INCH OUNCES = 15882

Ounce-inches.

## INCH POUNDS = 15883

Pound-inches.

#### K = 10325

Kelvins.

## KILOGRAM\_FORCE = 15877

Kilograms-force.

## METERS = 10219

Meters.

## METERS\_PER\_SECOND = 15959

Meters per second.

## METERS PER SECOND SQUARED = 12470

Meters per second per second.

## $M_VOLTS_PER_VOLT = 15897$

Millivolts per volt.

## NEWTONS = 15875

Newtons.

## NEWTON METERS = 15881

Newton meters.

## OHMS = 10384

Ohms.

## PA = 10081

Pascals.

## $PICO_COULOMBS = 16103$

PicoCoulombs.

## POUNDS = 15876

Pounds.

## POUNDS\_PER\_SQ\_INCH = 15879

Pounds per square inch.

## RADIANS = 10273

Radians.

## RADIANS\_PER\_SECOND = 16081

Radians per second.

## RPM = 16080

Revolutions per minute.

### SECONDS = 10364

Seconds.

## STRAIN = 10299

Strain.

#### TICKS = 10304

Ticks.

#### VOLTS = 10348

Volts.

## VOLTS\_PER\_VOLT = 15896

Volts per volt.

#### class nidaqmx.constants.UsageTypeAI

Bases: enum. Enum

## ACCELERATION\_4\_WIRE\_DC\_VOLTAGE = 16106

Acceleration measurement using a 4 wire DC voltage based sensor.

## ACCELERATION\_ACCELEROMETER\_CURRENT\_INPUT = 10356

Acceleration measurement using an accelerometer.

## ACCELERATION\_CHARGE = 16104

Acceleration measurement using a charge-based sensor.

#### BRIDGE = 15908

Measure voltage ratios from a Wheatstone bridge.

#### CHARGE = 16105

Charge measurement.

#### CURRENT = 10134

Current measurement.

#### CURRENT ACRMS = 10351

Current RMS measurement.

## FORCE\_BRIDGE = 15899

Force measurement using a bridge-based sensor.

## $FORCE\_IEPE\_SENSOR = 15895$

Force measurement using an IEPE Sensor.

## FREQUENCY\_VOLTAGE = 10181

Frequency measurement using a frequency to voltage converter.

## POSITION ANGULAR RVDT = 10353

Position measurement using an RVDT.

## POSITION EDDY CURRENT PROX PROBE = 14835

Position measurement using an eddy current proximity probe.

#### POSITION LINEAR LVDT = 10352

Position measurement using an LVDT.

## PRESSURE\_BRIDGE = 15902

Pressure measurement using a bridge-based sensor.

## RESISTANCE = 10278

Resistance measurement.

#### ROSETTE STRAIN GAGE = 15980

Strain measurement using a rosette strain gage.

## SOUND\_PRESSURE\_MICROPHONE = 10354

Sound pressure measurement using a microphone.

#### STRAIN STRAIN GAGE = 10300

Strain measurement.

#### TEDS = 12531

Measurement type defined by TEDS.

## TEMPERATURE BUILT IN SENSOR = 10311

Temperature measurement using a built-in sensor on a terminal block or device. On SCXI modules, for example, this could be the CJC sensor.

## TEMPERATURE RTD = 10301

Temperature measurement using an RTD.

## TEMPERATURE\_THERMISTOR = 10302

Temperature measurement using a thermistor.

### TEMPERATURE THERMOCOUPLE = 10303

Temperature measurement using a thermocouple.

## TORQUE\_BRIDGE = 15905

Torque measurement using a bridge-based sensor.

#### **VELOCITY IEPE SENSOR = 15966**

Velocity measurement using an IEPE Sensor.

## VOLTAGE = 10322

Voltage measurement.

#### VOLTAGE ACRMS = 10350

Voltage RMS measurement.

## VOLTAGE\_CUSTOM\_WITH\_EXCITATION = 10323

Voltage measurement with an excitation source. You can use this measurement type for custom sensors that require excitation, but you must use a custom scale to scale the measured voltage.

## class nidaqmx.constants.UsageTypeAO

Bases: enum.Enum

#### CURRENT = 10134

Current generation.

## FUNCTION\_GENERATION = 14750

Function generation.

### VOLTAGE = 10322

Voltage generation.

#### class nidagmx.constants.UsageTypeCI

Bases: enum. Enum

## $COUNT\_EDGES = 10125$

Count edges of a digital signal.

## ${\tt DUTY\_CYCLE} = 16070$

Measure the duty cycle of a digital signal.

#### FREQUENCY = 10179

Measure the frequency of a digital signal.

## PERIOD = 10256

Measure the period of a digital signal.

### POSITION ANGULAR ENCODER = 10360

Angular position measurement using an angular encoder.

#### POSITION LINEAR ENCODER = 10361

Linear position measurement using a linear encoder.

## $PULSE\_FREQ = 15864$

Pulse measurement, returning the result as frequency and duty cycle.

#### PULSE TICKS = 15866

Pulse measurement, returning the result as high ticks and low ticks.

#### PULSE TIME = 15865

Pulse measurement, returning the result as high time and low time.

#### PULSE WIDTH DIGITAL = 10359

Measure the width of a pulse of a digital signal.

## PULSE\_WIDTH\_DIGITAL\_SEMI\_PERIOD = 10289

Measure the time between state transitions of a digital signal.

## PULSE\_WIDTH\_DIGITAL\_TWO\_EDGE\_SEPARATION = 10267

Measure time between edges of two digital signals.

#### TIME GPS = 10362

Timestamp measurement, synchronizing the counter to a GPS receiver.

#### VELOCITY ANGULAR ENCODER = 16078

Angular velocity measurement using an angular encoder.

#### VELOCITY LINEAR ENCODER = 16079

Linear velocity measurement using a linear encoder.

## class nidaqmx.constants.UsageTypeCO

Bases: enum. Enum

## PULSE\_FREQUENCY = 10119

Generate digital pulses defined by frequency and duty cycle.

#### PULSE TICKS = 10268

Generate digital pulses defined by the number of timebase ticks that the pulse is at a low state and the number of timebase ticks that the pulse is at a high state.

## $PULSE\_TIME = 10269$

Generate pulses defined by the time the pulse is at a low state and the time the pulse is at a high state.

## class nidaqmx.constants.VelocityIEPESensorSensitivityUnits

Bases: enum.Enum

## M\_VOLTS\_PER\_INCH\_PER\_SECOND = 15964

Millivolts per inch per second.

### M\_VOLTS\_PER\_MILLIMETER\_PER\_SECOND = 15963

Millivolts per millimeter per second.

## class nidaqmx.constants.VelocityUnits

Bases: enum.Enum

#### FROM CUSTOM SCALE = 10065

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

## INCHES\_PER\_SECOND = 15960

Inches per second.

## METERS PER SECOND = 15959

Meters per second.

## class nidaqmx.constants.VoltageUnits

Bases: enum. Enum

#### FROM CUSTOM SCALE = 10065

Units a custom scale specifies. If you select this value, you must specify a custom scale name.

## FROM TEDS = 12516

Units defined by TEDS information associated with the channel.

#### VOLTS = 10348

Volts.

## class nidagmx.constants.WDTTaskAction

Bases: enum. Enum

#### $CLEAR_EXPIRATION = 1$

Clear Expiration

## $RESET_TIMER = 0$

Reset Timer

#### class nidagmx.constants.WaitMode

Bases: enum. Enum

#### POLL = 12524

Repeatedly check for available samples as fast as possible. This mode allows for the highest sampling rates at the expense of CPU efficiency.

#### SLEEP = 12547

Check for available samples once per the amount of time specified in **sleep time**.

## WAIT\_FOR\_INTERRUPT = 12523

Check for available samples when the system receives an interrupt service request. This mode is the most CPU efficient, but results in lower possible sampling rates.

#### YIELD = 12525

Repeatedly check for available samples, but yield control to other threads after each check. This mode offers a balance between sampling rate and CPU efficiency.

## class nidaqmx.constants.WatchdogAOExpirState

Bases: enum. Enum

## CURRENT = 10134

Current output.

#### NO CHANGE = 10160

Expiration does not affect the port. Do not change the state of any lines in the port, and do not lock the port.

## VOLTAGE = 10322

Voltage output.

## ${\bf class} \; {\tt nidaqmx.constants.WatchdogCOExpirState}$

Bases: enum.Enum

#### HIGH = 10192

High logic.

## LOW = 10214

Low logic.

#### NO CHANGE = 10160

Expiration does not affect the state of the counter output. The channels retain their states at the time of the watchdog timer expiration, and no further counter generation runs.

## class nidagmx.constants.WaveformAttributes

Bases: enum. Enum

#### SAMPLES AND TIMING = 10140

Return the samples and timing information.

## SAMPLES ONLY = 10287

Return only samples.

## SAMPLES\_TIMING\_AND\_ATTRIBUTES = 10141

Return the samples, timing information, and other attributes, such as the name of the channel.

#### class nidagmx.constants.WindowTriggerCondition1

Bases: enum. Enum

## $ENTERING_WINDOW = 10163$

Trigger when the signal enters the window.

#### LEAVING WINDOW = 10208

Trigger when the signal leaves the window.

## ${\bf class} \; {\tt nidaqmx.constants.WindowTriggerCondition2}$

Bases: enum. Enum

#### INSIDE WINDOW = 10199

Pause the measurement or generation while the trigger is inside the window.

## $OUTSIDE_WINDOW = 10251$

Pause the measurement or generation while the signal is outside the window.

## class nidaqmx.constants.WriteBasicTEDSOptions

 $Bases \colon \texttt{enum.Enum}$ 

## ${\tt DO\_NOT\_WRITE} = 12540$

blah

## $WRITE_TO_EEPROM = 12538$

blah

## $WRITE\_TO\_PROM = 12539$

blah

## class nidagmx.constants.WriteRelativeTo

Bases: enum. Enum

## CURRENT\_WRITE\_POSITION = 10430

Write samples relative to the current position in the buffer.

### FIRST SAMPLE = 10424

Write samples relative to the first sample.

# nidaqmx.errors

```
exception nidaqmx.errors.DagError (message, error_code, task_name=u'')
```

Bases: nidaqmx.errors.Error

Error raised by any DAQmx method.

8.2. nidagmx.errors 51

```
error_code
    int - Specifies the NI-DAQmx error code.

error_type
    nidaqmx.error_codes.DAQmxErrors - Specifies the NI-DAQmx error type.

exception nidaqmx.errors.DaqWarning (message, error_code)
    Bases: exceptions.Warning

Warning raised by any NI-DAQmx method.

error_code
    int - Specifies the NI-DAQmx error code.

error_type
    nidaqmx.error_codes.DAQmxWarnings - Specifies the NI-DAQmx error type.

nidaqmx.errors.DaqResourceWarning
    alias of _ResourceWarning
```

# nidaqmx.scale

```
class nidaqmx.scale.Scale (name)
    Bases: object

Represents a DAQmx scale.
__init__ (name)

    Parameters name (str) - Specifies the name of the scale to create.
__weakref__
    list of weak references to the object (if defined)

static calculate_reverse_poly_coeff (forward_coeffs, min_val_x=-5.0, max_val_x=5.0, num_points_to_compute=1000, reverse_poly_order=-1)
```

Computes a set of coefficients for a polynomial that approximates the inverse of the polynomial with the coefficients you specify with the "forward\_coeffs" input. This function generates a table of x versus y values over the range of x. This function then finds a polynomial fit, using the least squares method to compute a polynomial that computes x when given a value for y.

#### **Parameters**

- **forward\_coeffs** (List[float]) Is the list of coefficients for the polynomial that computes y given a value of x. Each element of the list corresponds to a term of the equation.
- min\_val\_x (Optional[float]) Is the minimum value of x for which you use the polynomial. This is the smallest value of x for which the function generates a y value in the table.
- max\_val\_x (Optional[float]) Is the maximum value of x for which you use the polynomial. This is the largest value of x for which the function generates a y value in the table.
- num\_points\_to\_compute (Optional[int]) Is the number of points in the table of x versus y values. The function spaces the values evenly between "min\_val\_x" and "max\_val\_x".

• **reverse\_poly\_order** (Optional[int]) – Is the order of the reverse polynomial to compute. For example, an input of 3 indicates a 3rd order polynomial. A value of -1 indicates a reverse polynomial of the same order as the forward polynomial.

**Returns** Specifies the list of coefficients for the reverse polynomial. Each element of the list corresponds to a term of the equation. For example, if index three of the list is 9, the fourth term of the equation is 9y^3.

### **Return type** List[float]

```
static create_lin_scale (scale_name, slope, y_intercept=0.0, pre_scaled_units=<UnitsPreScaled.VOLTS: 10348>, scaled_units=None)
```

Creates a custom scale that uses the equation y=mx+b, where x is a pre-scaled value, and y is a scaled value. The equation is identical for input and output. If the equation is in the form x=my+b, you must first solve for y in terms of x.

#### **Parameters**

- **scale\_name** (str) Specifies the name of the scale to create.
- **slope** (*float*) Is the slope, m, in the equation.
- y\_intercept (Optional[float]) Is the y-intercept, b, in the equation.
- pre\_scaled\_units (Optional[nidaqmx.constants.UnitsPreScaled])

   Is the units of the values to scale.
- **scaled\_units** (Optional[str]) Is the units to use for the scaled value. You can use an arbitrary string. NI-DAQmx uses the units to label a graph or chart.

**Returns** Indicates an object that represents the created custom scale.

Return type nidagmx.scale.Scale

Creates a custom scale that scales values proportionally from a range of pre-scaled values to a range of scaled values.

### **Parameters**

- **scale\_name** (str) Specifies the name of the scale to create.
- **prescaled\_min** (float) Is the smallest value in the range of pre-scaled values. NI-DAQmx maps this value to "scaled\_min".
- prescaled\_max (float) Is the largest value in the range of pre-scaled values. NI-DAOmx maps this value to "scaled max".
- **scaled\_min** (float) Is the smallest value in the range of scaled values. NI-DAQmx maps this value to "prescaled\_min". Read operations clip samples that are smaller than this value. Write operations generate errors for samples that are smaller than this value.
- **scaled\_max** (float) Is the largest value in the range of scaled values. NI-DAQmx maps this value to "prescaled\_max". Read operations clip samples that are larger than this value. Write operations generate errors for samples that are larger than this value.
- pre\_scaled\_units (Optional[nidaqmx.constants.UnitsPreScaled])
   Is the units of the values to scale.
- **scaled\_units** (Optional[str]) Is the units to use for the scaled value. You can use an arbitrary string. NI-DAQmx uses the units to label a graph or chart.

**Returns** Indicates an object that represents the created custom scale.

8.3. nidagmx.scale 53

Return type nidagmx.scale.Scale

Creates a custom scale that uses an nth order polynomial equation. NI-DAQmx requires both a polynomial to convert pre- scaled values to scaled values (forward) and a polynomial to convert scaled values to pre-scaled values (reverse). If you only know one set of coefficients, use the DAQmx Compute Reverse Polynomial Coefficients function to generate the other set.

#### **Parameters**

- $scale_name(str)$  Specifies the name of the scale to create.
- **forward\_coeffs** (List[float]) Is an list of coefficients for the polynomial that converts pre-scaled values to scaled values. Each element of the list corresponds to a term of the equation.
- reverse\_coeffs (List[float]) Is an list of coefficients for the polynomial that converts scaled values to pre-scaled values. Each element of the list corresponds to a term of the equation.
- pre\_scaled\_units (Optional[nidaqmx.constants.UnitsPreScaled])

   Is the units of the values to scale.
- **scaled\_units** (Optional[str]) Is the units to use for the scaled value. You can use an arbitrary string. NI-DAQmx uses the units to label a graph or chart.

**Returns** Indicates an object that represents the created custom scale.

Return type nidaqmx.scale.Scale

```
static create_table_scale (scale_name, prescaled_vals, pre_scaled_units=<UnitsPreScaled.VOLTS: 10348>, scaled_units=None)
```

Creates a custom scale that maps an list of pre-scaled values to an list of corresponding scaled values. NI-DAQmx applies linear interpolation to values that fall between the values in the table. Read operations clip scaled samples that are outside the maximum and minimum scaled values found in the table. Write operations generate errors for samples that are outside the minimum and maximum scaled values found in the table.

## **Parameters**

- $scale_name(str)$  Specifies the name of the scale to create.
- **prescaled\_vals** (*List[float]*) Is the list of pre-scaled values that map to the values in "scaled\_vals".
- **scaled\_vals** (List[float]) Is the list of scaled values that map to the values in "prescaled\_vals".
- pre\_scaled\_units (Optional[nidaqmx.constants.UnitsPreScaled])

   Is the units of the values to scale.
- **scaled\_units** (Optional[str]) Is the units to use for the scaled value. You can use an arbitrary string. NI-DAQmx uses the units to label a graph or chart.

**Returns** Indicates an object that represents the created custom scale.

Return type nidaqmx.scale.Scale

## description

str – Specifies a description for the scale.

## lin\_slope

float – Specifies the slope, m, in the equation y=mx+b.

## lin\_y\_intercept

*float* – Specifies the y-intercept, b, in the equation y=mx+b.

#### map\_pre\_scaled\_max

float – Specifies the largest value in the range of pre-scaled values. NI-DAQmx maps this value to map\_scaled\_max.

#### map\_pre\_scaled\_min

float – Specifies the smallest value in the range of pre-scaled values. NI-DAQmx maps this value to map\_scaled\_min.

### map\_scaled\_max

float – Specifies the largest value in the range of scaled values. NI-DAQmx maps this value to map\_pre\_scaled\_max. Reads coerce samples that are larger than this value to match this value. Writes generate errors for samples that are larger than this value.

#### map\_scaled\_min

float – Specifies the smallest value in the range of scaled values. NI-DAQmx maps this value to map\_pre\_scaled\_min. Reads coerce samples that are smaller than this value to match this value. Writes generate errors for samples that are smaller than this value.

#### name

str – Specifies the name of this scale.

#### poly\_forward\_coeff

List[float] – Specifies a list of coefficients for the polynomial that converts pre-scaled values to scaled values. Each element of the list corresponds to a term of the equation. For example, if index three of the list is 9, the fourth term of the equation is 9x<sup>3</sup>.

### poly\_reverse\_coeff

List[float] – Specifies a list of coefficients for the polynomial that converts scaled values to pre-scaled values. Each element of the list corresponds to a term of the equation. For example, if index three of the list is 9, the fourth term of the equation is 9y^3.

#### pre\_scaled\_units

nidagmx.constants.UnitsPreScaled - Specifies the units of the values that you want to scale.

save (save\_as=u'', author=u'', overwrite\_existing\_scale=False, allow\_interactive\_editing=True, allow\_interactive\_deletion=True)
Saves this custom scale to MAX.

#### **Parameters**

- **save\_as** (Optional[str]) Is the name to save the task, global channel, or custom scale as. If you do not specify a value for this input, NI-DAQmx uses the name currently assigned to the task, global channel, or custom scale.
- author (Optional[str]) Is a name to store with the task, global channel, or custom scale.
- **options** (Optional[int]) Specifies whether to allow the task, global channel, or custom scale to be deleted through MAX.
- **overwrite\_existing\_scale** (Optional[bool]) Specifies whether to overwrite a custom scale of the same name if one is already saved in MAX. If this input is False and a custom scale of the same name is already saved in MAX, this function returns an error.

8.3. nidagmx.scale 55

- allow\_interactive\_editing (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be edited in the DAQ Assistant. If allow\_interactive\_editing is True, the DAQ Assistant must support all task or global channel settings.
- allow\_interactive\_deletion (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be deleted through MAX.

#### scale\_type

nidaqmx.constants.ScaleType - Indicates the method or equation form that the custom scale uses.

## scaled\_units

str – Specifies the units to use for scaled values. You can use an arbitrary string.

#### table\_pre\_scaled\_vals

List[float] - Specifies a list of pre-scaled values. These values map directly to the values in table\_scaled\_vals.

#### table\_scaled\_vals

*List[float]* – Specifies a list of scaled values. These values map directly to the values in **table\_pre\_scaled\_vals**.

# nidaqmx.stream\_readers

class nidaqmx.stream\_readers.AnalogSingleChannelReader(task\_in\_stream)

Bases: nidaqmx.stream\_readers.ChannelReaderBase

Reads samples from an analog input channel in an NI-DAQmx task.

 $\verb"read_many_sample" (\textit{data}, number\_of\_samples\_per\_channel = -1, timeout = 10.0)$ 

Reads one or more floating-point samples from a single analog input channel in a task.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 1D NumPy array of floating-point values to hold the samples requested.

Each element in the array corresponds to a sample from the channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

## Return type int

## read\_one\_sample (timeout=10)

Reads a single floating-point sample from a single analog input channel in a task.

Parameters timeout (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates a single floating-point sample from the task.

Return type float

## verify\_array\_shape

bool – Indicates whether the size and shape of the user-defined NumPy arrays passed to read methods are verified. Defaults to True when this object is instantiated.

Setting this property to True may marginally adversely impact the performance of read methods.

## class nidaqmx.stream\_readers.AnalogMultiChannelReader(task\_in\_stream)

Bases: nidagmx.stream\_readers.ChannelReaderBase

Reads samples from one or more analog input channels in an NI-DAQmx task.

read\_many\_sample (data, number\_of\_samples\_per\_channel=-1, timeout=10.0)

Reads one or more floating-point samples from one or more analog input channels in a task.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 2D NumPy array of floating-point values to hold the samples requested. The size of the array must be large enough to hold all requested samples from all channels in the task; otherwise, an error is thrown.

Each row corresponds to a channel in the task. Each column corresponds to a sample from each channel. The order of the channels in the array corresponds to the order in which you add the channels to the task or to the order of the channels you specify with the "channels\_to\_read" property.

If the size of the array is too large or the array is shaped incorrectly, the previous statement may not hold true as the samples read may not be separated into rows and columns properly. Set the "verify\_array\_shape" property on this channel reader object to True to validate that the NumPy array object is shaped properly. Setting this property to True may marginally adversely impact the performance of the method.

 number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) – Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

## Return type int

## read\_one\_sample (data, timeout=10)

Reads a single floating-point sample from one or more analog input channels in a task.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 1D NumPy array of floating-point values to hold the samples requested.

Each element in the array corresponds to a sample from each channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• **timeout** (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

#### verify array shape

*bool* – Indicates whether the size and shape of the user-defined NumPy arrays passed to read methods are verified. Defaults to True when this object is instantiated.

Setting this property to True may marginally adversely impact the performance of read methods.

```
class nidaqmx.stream_readers.AnalogUnscaledReader(task_in_stream)
```

Bases: nidagmx.stream readers.ChannelReaderBase

Reads unscaled samples from one or more analog input channels in an NI-DAQmx task.

```
read_int16 (data, number_of_samples_per_channel=-1, timeout=10.0)
```

Reads one or more unscaled 16-bit integer samples from one or more analog input channels in a task.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 2D NumPy array of unscaled 16-bit integer values to hold the samples requested. The size of the array must be large enough to hold all requested samples from all channels in the task; otherwise, an error is thrown.

Each row corresponds to a channel in the task. Each column corresponds to a sample from each channel. The order of the channels in the array corresponds to the order in which you add the channels to the task or to the order of the channels you specify with the "channels\_to\_read" property.

If the size of the array is too large or the array is shaped incorrectly, the previous statement may not hold true as the samples read may not be separated into rows and columns properly. Set the "verify\_array\_shape" property on this channel reader object to True to validate that the NumPy array object is shaped properly. Setting this property may marginally adversely impact the performance of the method.

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

## Return type int

read\_int32 (data, number\_of\_samples\_per\_channel=-1, timeout=10.0)

Reads one or more unscaled 32-bit integer samples from one or more analog input channels in a task.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 2D NumPy array of unscaled 32-bit integer values to hold the samples requested. The size of the array must be large enough to hold all requested samples from all channels in the task; otherwise, an error is thrown.

Each row corresponds to a channel in the task. Each column corresponds to a sample from each channel. The order of the channels in the array corresponds to the order in which you add the channels to the task or to the order of the channels you specify with the "channels\_to\_read" property.

If the size of the array is too large or the array is shaped incorrectly, the previous statement may not hold true as the samples read may not be separated into rows and columns properly. Set the "verify\_array\_shape" property on this channel reader object to True to validate that the NumPy array object is shaped properly. Setting this property may marginally adversely impact the performance of the method.

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) – Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

## Return type int

read uint16 (data, number of samples per channel=-1, timeout=10.0)

Reads one or more unscaled 16-bit unsigned integer samples from one or more analog input channels in a task.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 2D NumPy array of unscaled 16-bit unsigned integer values to hold the samples requested. The size of the array must be large enough to hold all requested samples from all channels in the task; otherwise, an error is thrown.

Each row corresponds to a channel in the task. Each column corresponds to a sample from each channel. The order of the channels in the array corresponds to the order in which you add the channels to the task or to the order of the channels you specify with the "channels to read" property.

If the size of the array is too large or the array is shaped incorrectly, the previous statement may not hold true as the samples read may not be separated into rows and columns properly. Set the "verify\_array\_shape" property on this channel reader object to True to validate that the NumPy array object is shaped properly. Setting this property may marginally adversely impact the performance of the method.

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

## Return type int

read\_uint32 (data, number\_of\_samples\_per\_channel=-1, timeout=10.0)

Reads one or more unscaled unsigned 32-bit integer samples from one or more analog input channels in a task.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 2D NumPy array of unscaled 32-bit unsigned integer values to hold the samples requested. The size of the array must be large enough to hold all requested samples from all channels in the task; otherwise, an error is thrown.

Each row corresponds to a channel in the task. Each column corresponds to a sample from each channel. The order of the channels in the array corresponds to the order in which you add the channels to the task or to the order of the channels you specify with the "channels\_to\_read" property.

If the size of the array is too large or the array is shaped incorrectly, the previous statement may not hold true as the samples read may not be separated into rows and columns properly. Set the "verify\_array\_shape" property on this channel reader object to True to validate that the NumPy array object is shaped properly. Setting this property may marginally adversely impact the performance of the method.

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

# Return type int

#### verify\_array\_shape

*bool* – Indicates whether the size and shape of the user-defined NumPy arrays passed to read methods are verified. Defaults to True when this object is instantiated.

Setting this property to True may marginally adversely impact the performance of read methods.

class nidaqmx.stream\_readers.CounterReader(task\_in\_stream)
 Bases: nidaqmx.stream\_readers.ChannelReaderBase

Reads samples from a counter input channel in an NI-DAQmx task.

read\_many\_sample\_double (data, number\_of\_samples\_per\_channel=-1, timeout=10.0)

Reads one or more floating-point samples from a single counter input channel in a task.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 1D NumPy array of floating-point values to hold the samples requested.

Each element in the array corresponds to a sample from the channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

## Return type int

#### read\_many\_sample\_pulse\_frequency (frequencies,

duty\_cycles,

number\_of\_samples\_per\_channel=-1, timeout=10.0)

Reads one or more pulse samples in terms of frequency from a single counter input channel in a task.

This read method accepts preallocated NumPy arrays to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in preallocated arrays is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• **frequencies** (numpy.ndarray) – Specifies a preallocated 1D NumPy array of floating-point values to hold the frequency portion of the pulse samples requested.

Each element in the array corresponds to a sample from the channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• duty\_cycles (numpy.ndarray) - Specifies a preallocated 1D NumPy array of floating-point values to hold the duty cycle portion of the pulse samples requested.

Each element in the array corresponds to a sample from the channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

#### Return type int

Reads one or more pulse samples in terms of ticks from a single counter input channel in a task.

This read method accepts preallocated NumPy arrays to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in preallocated arrays is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

### **Parameters**

• high\_ticks (numpy.ndarray) – Specifies a preallocated 1D NumPy array of 32-bit unsigned integer values to hold the high ticks portion of the pulse samples requested.

Each element in the array corresponds to a sample from the channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• **low\_ticks** (*numpy.ndarray*) – Specifies a preallocated 1D NumPy array of 32-bit unsigned integer values to hold the low ticks portion of the pulse samples requested.

Each element in the array corresponds to a sample from the channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

 number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

## Return type int

Reads one or more pulse samples in terms of time from a single counter input channel in a task.

This read method accepts preallocated NumPy arrays to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in preallocated arrays is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• high\_times (numpy.ndarray) - Specifies a preallocated 1D NumPy array of floating-point values to hold the high time portion of the pulse samples requested.

Each element in the array corresponds to a sample from the channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• low\_times (numpy.ndarray) - Specifies a preallocated 1D NumPy array of floating-point values to hold the low time portion of the pulse samples requested.

Each element in the array corresponds to a sample from the channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

## Return type int

read\_many\_sample\_uint32 (data, number\_of\_samples\_per\_channel=-1, timeout=10.0)

Reads one or more 32-bit unsigned integer samples from a single counter input channel in a task.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 1D NumPy array of 32-bit unsigned integer values to hold the samples requested.

Each element in the array corresponds to a sample from the channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) – Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

### Return type int

#### read\_one\_sample\_double(timeout=10)

Reads a single floating-point sample from a single counter input channel in a task.

Parameters timeout (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

#### Returns

**Indicates a single floating-point sample from the task.** 

**Return type** float

## read\_one\_sample\_pulse\_frequency (timeout=10)

Reads a pulse sample in terms of frequency from a single counter input channel in a task.

Parameters timeout (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates a pulse sample in terms of frequency from the task.

Return type nidaqmx.types.CtrFreq

## read one sample pulse ticks(timeout=10)

Reads a pulse sample in terms of ticks from a single counter input channel in a task.

Parameters timeout (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates a pulse sample in terms of ticks from the task.

**Return type** *nidagmx.types.CtrTick* 

## read\_one\_sample\_pulse\_time (timeout=10)

Reads a pulse sample in terms of time from a single counter input channel in a task.

**Parameters timeout** (Optional[float]) – Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set

timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates a pulse sample in terms of time from the task.

**Return type** *nidaqmx.types.CtrTime* 

## read\_one\_sample\_uint32 (timeout=10)

Reads a single 32-bit unsigned integer sample from a single counter input channel in a task.

**Parameters timeout** (Optional[float]) – Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates a single 32-bit unsigned integer sample from the task.

Return type int

### verify\_array\_shape

bool – Indicates whether the size and shape of the user-defined NumPy arrays passed to read methods are verified. Defaults to True when this object is instantiated.

Setting this property to True may marginally adversely impact the performance of read methods.

class nidaqmx.stream\_readers.DigitalSingleChannelReader (task\_in\_stream)

Bases: nidaqmx.stream\_readers.ChannelReaderBase

Reads samples from a digital input channel in an NI-DAQmx task.

read\_many\_sample\_port\_byte (data, number\_of\_samples\_per\_channel=-1, timeout=10.0)

Reads one or more 8-bit unsigned integer samples from a single digital input channel in a task.

Use this method for devices with up to 8 lines per port.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 1D NumPy array of 8-bit unsigned integer values to hold the samples requested.

Each element in the array corresponds to a sample from the channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

## Return type int

read\_many\_sample\_port\_uint16 (data, number\_of\_samples\_per\_channel=-1, timeout=10.0) Reads one or more 16-bit unsigned integer samples from a single digital input channel in a task.

Use this method for devices with up to 16 lines per port.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 1D NumPy array of 16-bit unsigned integer values to hold the samples requested.

Each element in the array corresponds to a sample from the channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

## Return type int

read\_many\_sample\_port\_uint32 (*data*, *number\_of\_samples\_per\_channel=-1*, *timeout=10.0*)
Reads one or more 32-bit unsigned integer samples from a single digital input channel in a task.

Use this method for devices with up to 32 lines per port.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 1D NumPy array of 32-bit unsigned integer values to hold the samples requested.

Each element in the array corresponds to a sample from the channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

## Return type int

### read\_one\_sample\_multi\_line (data, timeout=10)

Reads a single boolean sample from a single digital input channel in a task. The channel can contain multiple digital lines.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) - Specifies a preallocated 1D NumPy array of boolean values to hold the samples requested.

Each element in the array corresponds to a sample from a line in the channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

## read\_one\_sample\_one\_line (timeout=10)

Reads a single boolean sample from a single digital input channel in a task. The channel can contain only one digital line.

Parameters timeout (Optional[float]) – Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates a single boolean sample from the task.

Return type bool

## read\_one\_sample\_port\_byte (timeout=10)

Reads a single 8-bit unsigned integer sample from a single digital input channel in a task.

Use this method for devices with up to 8 lines per port.

Parameters timeout (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates a single 8-bit unsigned integer sample from the task.

Return type int

## read\_one\_sample\_port\_uint16(timeout=10)

Reads a single 16-bit unsigned integer sample from a single digital input channel in a task.

Use this method for devices with up to 16 lines per port.

Parameters timeout (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates a single 16-bit unsigned integer sample from the task.

Return type int

## read\_one\_sample\_port\_uint32 (timeout=10)

Reads a single 32-bit unsigned integer sample from a single digital input channel in a task.

Use this method for devices with up to 32 lines per port.

Parameters timeout (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates a single 32-bit unsigned integer sample from the task.

Return type int

## verify\_array\_shape

*bool* – Indicates whether the size and shape of the user-defined NumPy arrays passed to read methods are verified. Defaults to True when this object is instantiated.

Setting this property to True may marginally adversely impact the performance of read methods.

```
class nidaqmx.stream_readers.DigitalMultiChannelReader(task_in_stream)
```

Bases: nidaqmx.stream\_readers.ChannelReaderBase

Reads samples from one or more digital input channels in an NI-DAQmx task.

```
read_many_sample_port_byte (data, number_of_samples_per_channel=-1, timeout=10.0)
```

Reads one or more 8-bit unsigned integer samples from one or more digital input channel in a task.

Use this method for devices with up to 8 lines per port.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 2D NumPy array of 8-bit unsigned integer values to hold the samples requested. The size of the array must be large enough to hold all requested samples from all channels in the task; otherwise, an error is thrown.

Each row corresponds to a channel in the task. Each column corresponds to a sample from each channel. The order of the channels in the array corresponds to the order in which you add the channels to the task or to the order of the channels you specify with the "channels to read" property.

If the size of the array is too large or the array is shaped incorrectly, the previous statement may not hold true as the samples read may not be separated into rows and columns properly. Set the "verify\_array\_shape" property on this channel reader object to True to validate that the NumPy array object is shaped properly. Setting this property may marginally adversely impact the performance of the method.

 number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) – Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

## Return type int

read\_many\_sample\_port\_uint16 (data, number\_of\_samples\_per\_channel=-1, timeout=10.0)

Reads one or more 16-bit unsigned integer samples from one or more digital input channels in a task.

Use this method for devices with up to 16 lines per port.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

## **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 2D NumPy array of 16-bit unsigned integer values to hold the samples requested. The size of the array must be large enough to hold all requested samples from all channels in the task; otherwise, an error is thrown.

Each row corresponds to a channel in the task. Each column corresponds to a sample from each channel. The order of the channels in the array corresponds to the order in which you add the channels to the task or to the order of the channels you specify with the "channels\_to\_read" property.

If the size of the array is too large or the array is shaped incorrectly, the previous statement may not hold true as the samples read may not be separated into rows and columns properly. Set the "verify\_array\_shape" property on this channel reader object to True to validate that the NumPy array object is shaped properly. Setting this property may marginally adversely impact the performance of the method.

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

### Return type int

read\_many\_sample\_port\_uint32 (data, number\_of\_samples\_per\_channel=-1, timeout=10.0)

Reads one or more 32-bit unsigned integer samples from one or more digital input channels in a task.

Use this method for devices with up to 32 lines per port.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 2D NumPy array of 32-bit unsigned integer values to hold the samples requested. The size of the array must be large enough to hold all requested samples from all channels in the task; otherwise, an error is thrown.

Each row corresponds to a channel in the task. Each column corresponds to a sample from each channel. The order of the channels in the array corresponds to the order in which you add the channels to the task or to the order of the channels you specify with the "channels\_to\_read" property.

If the size of the array is too large or the array is shaped incorrectly, the previous statement may not hold true as the samples read may not be separated into rows and columns properly. Set the "verify\_array\_shape" property on this channel reader object to True to validate that the NumPy array object is shaped properly. Setting this property may marginally adversely impact the performance of the method.

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp"

property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** Indicates the number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels.

## Return type int

## read\_one\_sample\_multi\_line(data, timeout=10)

Reads a single boolean sample from one or more digital input channels in a task. The channels can contain multiple digital lines.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 2D NumPy array of boolean values to hold the samples requested. The size of the array must be large enough to hold all requested samples from all channels in the task; otherwise, an error is thrown.

Each row corresponds to a channel in the task. Each column corresponds to a line from each channel. The order of the channels in the array corresponds to the order in which you add the channels to the task or to the order of the channels you specify with the "channels\_to\_read" property.

If the size of the array is too large or the array is shaped incorrectly, the previous statement may not hold true as the samples read may not be separated into rows and columns properly. Set the "verify\_array\_shape" property on this channel reader object to True to validate that the NumPy array object is shaped properly. Setting this property may marginally adversely impact the performance of the method.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

## read\_one\_sample\_one\_line (data, timeout=10)

Reads a single boolean sample from one or more digital input channels in a task. The channel can contain only one digital line.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) - Specifies a preallocated 1D NumPy array of boolean values to hold the samples requested.

Each element in the array corresponds to a sample from each channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• **timeout** (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

## read\_one\_sample\_port\_byte (data, timeout=10)

Reads a single 8-bit unsigned integer sample from one or more digital input channels in a task.

Use this method for devices with up to 8 lines per port.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 1D NumPy array of 8-bit unsigned integer values to hold the samples requested.

Each element in the array corresponds to a sample from each channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

## read\_one\_sample\_port\_uint16 (data, timeout=10)

Reads a single 16-bit unsigned integer sample from one or more digital input channels in a task.

Use this method for devices with up to 16 lines per port.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 1D NumPy array of 16-bit unsigned integer values to hold the samples requested.

Each element in the array corresponds to a sample from each channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any

samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

## read\_one\_sample\_port\_uint32 (data, timeout=10)

Reads a single 32-bit unsigned integer sample from one or more digital input channels in a task.

Use this method for devices with up to 32 lines per port.

This read method accepts a preallocated NumPy array to hold the samples requested, which can be advantageous for performance and interoperability with NumPy and SciPy.

Passing in a preallocated array is valuable in continuous acquisition scenarios, where the same array can be used repeatedly in each call to the method.

#### **Parameters**

• data (numpy.ndarray) – Specifies a preallocated 1D NumPy array of 32-bit unsigned integer values to hold the samples requested.

Each element in the array corresponds to a sample from each channel. The size of the array must be large enough to hold all requested samples from the channel in the task; otherwise, an error is thrown.

• **timeout** (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

## verify\_array\_shape

*bool* – Indicates whether the size and shape of the user-defined NumPy arrays passed to read methods are verified. Defaults to True when this object is instantiated.

Setting this property to True may marginally adversely impact the performance of read methods.

# nidagmx.stream writers

class nidaqmx.stream\_writers.AnalogSingleChannelWriter(task\_out\_stream,

auto\_start=<nidaqmx.stream\_writers.UnsetAutoStartSe
object>)

Bases: nidaqmx.stream\_writers.ChannelWriterBase

Writes samples to an analog output channel in an NI-DAQmx task.

## auto start

bool – Specifies if the write method automatically starts the task if you did not explicitly start it with the DAQmx Start Task method.

If you do not specify a value for this parameter, NI-DAQmx determines its value based on the type of write method used. If you use a one sample write method, its value is True; conversely, if you use a many sample write method, its value is False.

## verify\_array\_shape

bool – Indicates whether the size and shape of the user-defined NumPy arrays passed to read methods are verified. Defaults to True when this object is instantiated.

Setting this property to True may marginally adversely impact the performance of read methods.

### write\_many\_sample (data, timeout=10.0)

Writes one or more floating-point samples to a single analog output channel in a task.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

#### **Parameters**

- data (numpy.ndarray) Contains a 1D NumPy array of floating-point samples to write to the task. Each element of the array corresponds to a sample to write.
- timeout (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote.

## Return type int

## write\_one\_sample (data, timeout=10)

Writes a single floating-point sample to a single analog output channel in a task.

#### **Parameters**

- data (float) Specifies the floating-point sample to write to the task.
- auto\_start (Optional[bool]) Specifies if this method automatically starts the task if you did not explicitly start it with the DAQmx Start Task method.
- timeout (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

class nidaqmx.stream\_writers.AnalogMultiChannelWriter(task\_out\_stream,

auto\_start=<nidaqmx.stream\_writers.UnsetAutoStartSen.
object>)

Bases: nidaqmx.stream\_writers.ChannelWriterBase

Writes samples to one or more analog output channels in an NI-DAQmx task.

## auto\_start

bool – Specifies if the write method automatically starts the task if you did not explicitly start it with the DAQmx Start Task method.

If you do not specify a value for this parameter, NI-DAQmx determines its value based on the type of write method used. If you use a one sample write method, its value is True; conversely, if you use a many sample write method, its value is False.

## verify\_array\_shape

bool – Indicates whether the size and shape of the user-defined NumPy arrays passed to read methods are verified. Defaults to True when this object is instantiated.

Setting this property to True may marginally adversely impact the performance of read methods.

## write\_many\_sample (data, timeout=10.0)

Writes one or more floating-point samples to one or more analog output channels in a task.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

#### **Parameters**

• data (numpy.ndarray) - Contains a 2D NumPy array of floating-point samples to write to the task.

Each row corresponds to a channel in the task. Each column corresponds to a sample to write to each channel. The order of the channels in the array corresponds to the order in which you add the channels to the task.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote to each channel in the task.

## Return type int

## write\_one\_sample (data, timeout=10)

Writes a single floating-point sample to one or more analog output channels in a task.

#### **Parameters**

• data (numpy.ndarray) - Contains a 1D NumPy array of floating-point samples to write to the task.

Each element of the array corresponds to a channel in the task. The order of the channels in the array corresponds to the order in which you add the channels to the task.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

class nidaqmx.stream\_writers.AnalogUnscaledWriter(task\_out\_stream,

auto\_start=<nidaqmx.stream\_writers.UnsetAutoStartSentinel
object>)

Bases: nidagmx.stream writers.ChannelWriterBase

Writes unscaled samples to one or more analog output channels in an NI-DAQmx task.

## auto start

bool – Specifies if the write method automatically starts the task if you did not explicitly start it with the DAQmx Start Task method.

If you do not specify a value for this parameter, NI-DAQmx determines its value based on the type of write method used. If you use a one sample write method, its value is True; conversely, if you use a many sample write method, its value is False.

## verify\_array\_shape

*bool* – Indicates whether the size and shape of the user-defined NumPy arrays passed to read methods are verified. Defaults to True when this object is instantiated.

Setting this property to True may marginally adversely impact the performance of read methods.

## write\_int16 (data, timeout=10.0)

Writes one or more unscaled 16-bit integer samples to one or more analog output channels in a task.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

#### **Parameters**

• data (numpy.ndarray) – Contains a 2D NumPy array of unscaled 16-bit integer samples to write to the task.

Each row corresponds to a channel in the task. Each column corresponds to a sample to write to each channel.

• **timeout** (Optional[float]) – Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote to each channel in the task.

## Return type int

## write\_int32 (data, timeout=10.0)

Writes one or more unscaled 32-bit integer samples to one or more analog output channels in a task.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

### **Parameters**

• data (numpy.ndarray) - Contains a 2D NumPy array of unscaled 32-bit integer samples to write to the task.

Each row corresponds to a channel in the task. Each column corresponds to a sample to write to each channel.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the

submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote to each channel in the task.

## Return type int

## write\_uint16 (data, timeout=10.0)

Writes one or more unscaled 16-bit unsigned integer samples to one or more analog output channels in a task.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

### **Parameters**

 data (numpy.ndarray) - Contains a 2D NumPy array of unscaled 16-bit unsigned integer samples to write to the task.

Each row corresponds to a channel in the task. Each column corresponds to a sample to write to each channel.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote to each channel in the task.

## Return type int

### write\_uint32 (data, timeout=10.0)

Writes one or more unscaled 32-bit unsigned integer samples to one or more analog output channels in a task.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

### **Parameters**

• data (numpy.ndarray) - Contains a 2D NumPy array of unscaled 32-bit unsigned integer samples to write to the task.

Each row corresponds to a channel in the task. Each column corresponds to a sample to write to each channel.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the

submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote to each channel in the task.

### Return type int

Bases: nidaqmx.stream\_writers.ChannelWriterBase

Writes samples to a counter output channel in an NI-DAQmx task.

### auto\_start

bool – Specifies if the write method automatically starts the task if you did not explicitly start it with the DAQmx Start Task method.

If you do not specify a value for this parameter, NI-DAQmx determines its value based on the type of write method used. If you use a one sample write method, its value is True; conversely, if you use a many sample write method, its value is False.

## verify\_array\_shape

*bool* – Indicates whether the size and shape of the user-defined NumPy arrays passed to read methods are verified. Defaults to True when this object is instantiated.

Setting this property to True may marginally adversely impact the performance of read methods.

## write\_many\_sample\_pulse\_frequency (frequencies, duty\_cycles, timeout=10.0)

Writes one or more pulse samples in terms of frequency to a single counter output channel in a task.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

#### **Parameters**

- **frequencies** (numpy.ndarray) Contains a 1D NumPy array of floating-point values that holds the frequency portion of the pulse samples to write to the task. Each element of the array corresponds to a sample to write.
- **duty\_cycles** (numpy.ndarray) Contains a 1D NumPy array of floating-point values that holds the duty cycle portion of the pulse samples to write to the task. Each element of the array corresponds to a sample to write.
- timeout (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote.

**Return type** int

### write\_many\_sample\_pulse\_ticks (high\_ticks, low\_ticks, timeout=10.0)

Writes one or more pulse samples in terms of ticks to a single counter output channel in a task.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

### **Parameters**

- high\_ticks (numpy.ndarray) Contains a 1D NumPy array of 32-bit unsigned integer values that holds the high ticks portion of the pulse samples to write to the task. Each element of the array corresponds to a sample to write.
- **low\_ticks** (numpy.ndarray) Contains a 1D NumPy array of 32-bit unsigned integer values that holds the low ticks portion of the pulse samples to write to the task. Each element of the array corresponds to a sample to write.
- timeout (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote.

## Return type int

write many sample pulse time (high times, low times, timeout=10.0)

Writes one or more pulse samples in terms of time to a single counter output channel in a task.

If the task uses on-demand timing, this method returns only after the device generates all samples. On-demand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

## **Parameters**

- high\_times (numpy.ndarray) Contains a 1D NumPy array of floating-point values that holds the high time portion of the pulse samples to write to the task. Each element of the array corresponds to a sample to write.
- **low\_times** (numpy.ndarray) Contains a 1D NumPy array of floating-point values that holds the low time portion of the pulse samples to write to the task. Each element of the array corresponds to a sample to write.
- timeout (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote.

## Return type int

write\_one\_sample\_pulse\_frequency (frequency, duty\_cycle, timeout=10)

Writes a new pulse frequency and duty cycle to a single counter output channel in a task.

### **Parameters**

- **frequency** (float) Specifies at what frequency to generate pulses.
- **duty\_cycle** (float) Specifies the width of the pulse divided by the pulse period. NI-DAQmx uses this ratio combined with frequency to determine pulse width and the interval between pulses.
- auto\_start (Optional[bool]) Specifies if this method automatically starts the task if you did not explicitly start it with the DAQmx Start Task method.
- timeout (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

## write\_one\_sample\_pulse\_ticks (high\_ticks, low\_ticks, timeout=10)

Writes a new pulse high tick count and low tick count to a single counter output channel in a task.

#### **Parameters**

- high\_ticks (float) Specifies the number of ticks the pulse is high.
- low\_ticks (float) Specifies the number of ticks the pulse is low.
- timeout (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

# $\verb|write_one_sample_pulse_time| (high\_time, low\_time, timeout=10)$

Writes a new pulse high time and low time to a single counter output channel in a task.

### **Parameters**

- high\_time (float) Specifies the amount of time the pulse is high.
- low\_time (float) Specifies the amount of time the pulse is low.
- timeout (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

class nidaqmx.stream\_writers.DigitalSingleChannelWriter(task\_out\_stream,

auto\_start=<nidaqmx.stream\_writers.UnsetAutoStartS
object>)

Bases: nidagmx.stream writers.ChannelWriterBase

Writes samples to a single digital output channel in an NI-DAQmx task.

## auto\_start

bool – Specifies if the write method automatically starts the task if you did not explicitly start it with the DAQmx Start Task method.

If you do not specify a value for this parameter, NI-DAQmx determines its value based on the type of write method used. If you use a one sample write method, its value is True; conversely, if you use a many sample write method, its value is False.

## verify\_array\_shape

bool – Indicates whether the size and shape of the user-defined NumPy arrays passed to read methods are verified. Defaults to True when this object is instantiated.

Setting this property to True may marginally adversely impact the performance of read methods.

## write\_many\_sample\_port\_byte (data, timeout=10.0)

Writes one or more 8-bit unsigned integer samples to a single digital output channel in a task.

Use this method for devices with up to 8 lines per port.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

#### **Parameters**

- data (numpy.ndarray) Contains a 1D NumPy array of 8-bit unsigned integer samples to write to the task. Each element of the array corresponds to a sample to write.
- timeout (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote.

## Return type int

## write\_many\_sample\_port\_uint16 (data, timeout=10.0)

Writes one or more 16-bit unsigned integer samples to a single digital output channel in a task.

Use this method for devices with up to 16 lines per port.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

## **Parameters**

- data (numpy.ndarray) Contains a 1D NumPy array of 16-bit unsigned integer samples to write to the task. Each element of the array corresponds to a sample to write.
- timeout (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote.

Return type int

## write\_many\_sample\_port\_uint32 (data, timeout=10.0)

Writes one or more 32-bit unsigned integer samples to a single digital output channel in a task.

Use this method for devices with up to 32 lines per port.

If the task uses on-demand timing, this method returns only after the device generates all samples. On-demand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

#### **Parameters**

- data (numpy.ndarray) Contains a 1D NumPy array of 32-bit unsigned integer samples to write to the task. Each element of the array corresponds to a sample to write.
- **timeout** (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote.

## Return type int

## write\_one\_sample\_multi\_line (data, timeout=10)

Writes a single boolean sample to a single digital output channel in a task. The channel can contain multiple digital lines.

#### **Parameters**

- data (numpy.ndarray) Contains a 1D NumPy array of boolean samples to write to the task. Each element of the array corresponds to a line in the channel.
- **timeout** (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

## write\_one\_sample\_one\_line (data, timeout=10)

Writes a single boolean sample to a single digital output channel in a task. The channel can contain only one digital line.

## **Parameters**

- data (int) Specifies the boolean sample to write to the task.
- **timeout** (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the

submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

## write\_one\_sample\_port\_byte (data, timeout=10)

Writes a single 8-bit unsigned integer sample to a single digital output channel in a task.

Use this method for devices with up to 8 lines per port.

#### **Parameters**

- data (int) Specifies the 8-bit unsigned integer sample to write to the task.
- timeout (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

## write\_one\_sample\_port\_uint16 (data, timeout=10)

Writes a single 16-bit unsigned integer sample to a single digital output channel in a task.

Use this method for devices with up to 16 lines per port.

#### **Parameters**

- data (int) Specifies the 16-bit unsigned integer sample to write to the task.
- timeout (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

## write\_one\_sample\_port\_uint32 (data, timeout=10)

Writes a single 32-bit unsigned integer sample to a single digital output channel in a task.

Use this method for devices with up to 32 lines per port.

#### **Parameters**

- data (int) Specifies the 32-bit unsigned integer sample to write to the task.
- timeout (Optional[float]) Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

class nidagmx.stream\_writers.DigitalMultiChannelWriter(task\_out\_stream,

auto\_start=<nidaqmx.stream\_writers.UnsetAutoStartSe
object>)

Bases: nidagmx.stream writers.ChannelWriterBase

Writes samples to one or more digital output channels in an NI-DAQmx task.

#### auto start

*bool* – Specifies if the write method automatically starts the task if you did not explicitly start it with the DAQmx Start Task method.

If you do not specify a value for this parameter, NI-DAQmx determines its value based on the type of write method used. If you use a one sample write method, its value is True; conversely, if you use a many sample write method, its value is False.

## verify\_array\_shape

bool – Indicates whether the size and shape of the user-defined NumPy arrays passed to read methods are verified. Defaults to True when this object is instantiated.

Setting this property to True may marginally adversely impact the performance of read methods.

## write\_many\_sample\_port\_byte (data, timeout=10.0)

Writes one or more 8-bit unsigned integer samples to one or more digital output channels in a task.

Use this method for devices with up to 8 lines per port.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

#### **Parameters**

• data (numpy.ndarray) – Contains a 2D NumPy array of 8-bit unsigned integer samples to write to the task.

Each row corresponds to a channel in the task. Each column corresponds to a sample to write to each channel. The order of the channels in the array corresponds to the order in which you add the channels to the task.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote to each channel in the task.

## Return type int

## write\_many\_sample\_port\_uint16 (data, timeout=10.0)

Writes one or more 16-bit unsigned integer samples to one or more digital output channels in a task.

Use this method for devices with up to 16 lines per port.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

## **Parameters**

• data (numpy.ndarray) – Contains a 2D NumPy array of 16-bit unsigned integer samples to write to the task.

Each row corresponds to a channel in the task. Each column corresponds to a sample to write to each channel. The order of the channels in the array corresponds to the order in which you add the channels to the task.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote to each channel in the task.

### Return type int

## write\_many\_sample\_port\_uint32 (data, timeout=10.0)

Writes one or more 32-bit unsigned integer samples to one or more digital output channels in a task.

Use this method for devices with up to 32 lines per port.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

#### **Parameters**

• data (numpy.ndarray) – Contains a 2D NumPy array of 32-bit unsigned integer samples to write to the task.

Each row corresponds to a channel in the task. Each column corresponds to a sample to write to each channel. The order of the channels in the array corresponds to the order in which you add the channels to the task.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote to each channel in the task.

## Return type int

## write\_one\_sample\_multi\_line (data, timeout=10)

Writes a single boolean sample to one or more digital output channels in a task. The channel can contain multiple digital lines.

#### **Parameters**

• data (numpy.ndarray) – Contains a 2D NumPy array of boolean samples to write to the task.

Each row corresponds to a channel in the task. Each column corresponds to a line from each channel. The order of the channels in the array corresponds to the order in which you add the channels to the task.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

### write\_one\_sample\_one\_line (data, timeout=10)

Writes a single boolean sample to one or more digital output channels in a task. The channel can contain only one digital line.

### **Parameters**

• data (numpy.ndarray) - Contains a 1D NumPy array of boolean samples to write to the task.

Each element in the array corresponds to a channel in the task. The order of the channels in the array corresponds to the order in which you add the channels to the task.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

## write one sample port byte (data, timeout=10)

Writes a single 8-bit unsigned integer sample to one or more digital output channels in a task.

Use this method for devices with up to 8 lines per port.

## **Parameters**

• data (numpy.ndarray) – Contains a 1D NumPy array of 8-bit unsigned integer samples to write to the task.

Each element in the array corresponds to a channel in the task. The order of the channels in the array corresponds to the order in which you add the channels to the task.

• **timeout** (Optional[float]) – Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

## write\_one\_sample\_port\_uint16(data, timeout=10)

Writes a single 16-bit unsigned integer sample to one or more digital output channels in a task.

Use this method for devices with up to 16 lines per port.

#### **Parameters**

• data (numpy.ndarray) – Contains a 1D NumPy array of 16-bit unsigned integer samples to write to the task.

Each element in the array corresponds to a channel in the task. The order of the channels in the array corresponds to the order in which you add the channels to the task.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

### write\_one\_sample\_port\_uint32 (data, timeout=10)

Writes a single 32-bit unsigned integer sample to one or more digital output channels in a task.

Use this method for devices with up to 32 lines per port.

#### **Parameters**

• data (numpy.ndarray) – Contains a 1D NumPy array of 32-bit unsigned integer samples to write to the task.

Each element in the array corresponds to a channel in the task. The order of the channels in the array corresponds to the order in which you add the channels to the task.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

# nidaqmx.system

class nidaqmx.system.system.System
 Bases: object

Represents a DAQmx system.

Contains static properties that access tasks, scales, and global channels stored in Measurement Automation Explorer (MAX), performs immediate operations on DAQ hardware, and creates classes from which you can get information about the hardware.

## add\_cdaq\_sync\_connection(ports\_to\_connect)

Adds a cDAQ Sync connection between devices. The connection is not verified.

**Parameters** ports\_to\_connect (nidaqmx.types.CDAQSyncConnection) - Specifies the cDAQ Sync ports to connect.

are\_configured\_cdaq\_sync\_ports\_disconnected(chassis\_devices\_ports=u'', timeout=-1.0)

Verifies configured cDAQ Sync connections between devices. Failures generally indicate a wiring issue or that a device has been powered off or removed. Stop all NI-DAQmx tasks running on the devices prior to running this function because any running tasks cause the verification process to fail.

## **Parameters**

• chassis\_devices\_ports (Optional[str]) - Specifies the names of the CompactDAQ chassis, C Series modules, or cDAQ Sync ports in comma separated form to search. If no names are specified, all cDAQ Sync ports on connected, non-simulated devices are scanned.

• **timeout** (Optional[float]) - Specifies the time in seconds to wait for the device to respond before timing out.

**Returns** Returns the port-to-port connections that failed verification.

**Return type** List[nidaqmx.types.CDAQSyncConnection]

auto\_configure\_cdaq\_sync\_connections (chassis\_devices\_ports=u'', timeout=-1.0)

Detects and configures cDAQ Sync connections between devices. Stop all NI-DAQmx tasks running on the devices prior to running this function because any running tasks cause auto-configuration to fail.

#### **Parameters**

- chassis\_devices\_ports (Optional[str]) Specifies the names of the CompactDAQ chassis, C Series modules, or cDAQ Sync ports in comma separated form to search. If no names are specified, all cDAQ Sync ports on connected, non-simulated devices are scanned.
- **timeout** (Optional[float]) Specifies the time in seconds to wait for the device to respond before timing out. If a timeout occurs, no configuration is changed.

**Returns** Returns the configured port-to-port connections.

**Return type** List[nidagmx.types.CDAQSyncConnection]

connect\_terms (source\_terminal, destination\_terminal, signal\_modifiers=<SignalModifiers.DO\_NOT\_INVERT\_POLARITY.

Creates a route between a source and destination terminal. The route can carry a variety of digital signals, such as triggers, clocks, and hardware events.

### **Parameters**

- **source\_terminal** (str) Specifies the originating terminal of the route. A DAQmx terminal constant lists all terminals available on devices installed in the system. You also can specify a source terminal by specifying a string that contains a terminal name.
- **destination\_terminal** (str) Specifies the receiving terminal of the route. A DAQmx terminal constant provides a list of all terminals available on devices installed in the system. You also can specify a destination terminal by specifying a string that contains a terminal name.
- **signal\_modifiers** (Optional[nidaqmx.constants. SignalModifiers]) Specifies whether to invert the signal this function routes from the source terminal to the destination terminal.

#### devices

*nidaqmx.system.\_collections.DeviceCollection* – Indicates the collection of devices for this DAQmx system.

disconnect\_terms (source\_terminal, destination\_terminal)

Removes signal routes you created by using the DAQmx Connect Terminals function. The DAQmx Disconnect Terminals function cannot remove task-based routes, such as those you create through timing and triggering configuration.

### **Parameters**

- **source\_terminal** (*str*) Specifies the originating terminal of the route. A DAQmx terminal constant lists all terminals available on devices installed in the system. You also can specify a source terminal by specifying a string that contains a terminal name.
- **destination\_terminal** (str) Specifies the receiving terminal of the route. A DAQmx terminal constant provides a list of all terminals available on devices installed in

the system. You also can specify a destination terminal by specifying a string that contains a terminal name.

## driver\_version

collections.namedtuple - Indicates the major, minor and update portions of the installed version of NI-DAQmx.

- •major\_version (int): Indicates the major portion of the installed version of NI-DAQmx, such as 7 for version 7.0.
- •minor\_version (int): Indicates the minor portion of the installed version of NI-DAQmx, such as 0 for version 7.0.
- •update\_version (int): Indicates the update portion of the installed version of NI-DAQmx, such as 1 for version 9.0.1.

## get\_analog\_power\_up\_states (device\_name)

Gets the power up states for analog physical channels.

**Parameters device\_name** (str) – Specifies the name as configured in MAX of the device to which this operation applies.

#### Returns

Contains the physical channels and power up states set. Each element of the list contains a physical channel and the power up state set for that physical channel.

- physical\_channel (str): Specifies the physical channel that was modified.
- power\_up\_state (float): Specifies the power up state set for the physical channel specified with the **physical\_channel** input.
- channel\_type (nidaqmx.constants.AOPowerUpOutputBehavior): Specifies the output type for the physical channel specified with the physical\_channel input.

**Return type** power\_up\_states (List[nidaqmx.types.AOPowerUpState])

# ${\tt get\_analog\_power\_up\_states\_with\_output\_type} \ (physical\_channels)$

Gets the power up states for analog physical channels.

**Parameters physical\_channels** (List[str]) – Indicates the physical channels that were modified.

## Returns

Contains the physical channels and power up states set. Each element of the list contains a physical channel and the power up state set for that physical channel.

- physical channel (str): Specifies the physical channel that was modified.
- power\_up\_state (float): Specifies the power up state set for the physical channel specified with the **physical\_channel** input.
- channel\_type (nidaqmx.constants.AOPowerUpOutputBehavior): Specifies the output type for the physical channel specified with the physical\_channel input.

**Return type** power\_up\_states (List[nidagmx.types.AOPowerUpState])

### get\_digital\_logic\_family\_power\_up\_state(device\_name)

Gets the digital logic family for a device.

**Parameters** device\_name (str) – Specifies the name as configured in MAX of the device to which this operation applies.

**Returns** Specifies the logic family to set the device to when it powers up. A logic family corresponds to voltage thresholds that are compatible with a group of voltage standards. Refer to device documentation for information on the logic high and logic low voltages for these logic families.

**Return type** *nidaqmx.constants.LogicFamily* 

### get\_digital\_power\_up\_states (device\_name)

Gets the power up states for digital physical lines.

**Parameters device\_name** (str) – Specifies the name as configured in MAX of the device to which this operation applies.

## Returns

Contains the physical channels and power up states set. Each element of the list contains a physical channel and the power up state set for that physical channel.

- physical\_channel (str): Indicates the physical channel that was modified.
- power\_up\_state (nidaqmx.constants.PowerUpStates): Indicates the power up state set for the physical channel specified with the **physical channel** output.

**Return type** List[nidaqmx.types.DOPowerUpState]

## get\_digital\_pull\_up\_pull\_down\_states (device\_name)

Gets the resistor level for lines when they are in tristate logic.

**Parameters** device\_name (str) – Specifies the name as configured in MAX of the device to which this operation applies.

#### Returns

Contains the physical channels and power up states set. Each element of the list contains a physical channel and the power up state set for that physical channel.

- physical\_channel (str): Indicates the physical channel that was modified.
- power\_up\_state (nidaqmx.constants.ResistorState): Indicates the power up state set for the physical channel specified with the physical\_channel output.

**Return type** List[nidagmx.types.DOResistorPowerUpState]

## global\_channels

*nidaqmx.system.\_collections.PersistedChannelCollection* – Indicates the collection of global channels for this DAQmx system.

### static local()

nidagmx.system.system.System: Represents the local DAQmx system.

### remove\_cdaq\_sync\_connection (ports\_to\_disconnect)

Removes a cDAQ Sync connection between devices. The connection is not verified.

Parameters ports\_to\_disconnect (nidaqmx.types.CDAQSyncConnection) - Specifies the cDAQ Sync ports to disconnect.

#### scales

*nidaqmx.system.\_collections.PersistedScaleCollection* – Indicates the collection of custom scales for this DAQmx system.

## set\_analog\_power\_up\_states (device\_name, power\_up\_states)

Updates power up states for analog physical channels.

#### **Parameters**

- **device\_name** (str) Specifies the name as configured in MAX of the device to which this operation applies.
- power\_up\_states (List[nidaqmx.types.AOPowerUpState]) Contains the physical channels and power up states to set. Each element of the list contains a physical channel and the power up state to set for that physical channel.
  - physical channel (str): Specifies the physical channel to modify.
  - power\_up\_state (float): Specifies the power up state to set for the physical channel specified with the **physical\_channel** input.
  - channel\_type (nidaqmx.constants.AOPowerUpOutputBehavior): Specifies the output type for the physical channel specified with the **physical\_channel** input.

# $\verb|set_analog_power_up_states_with_output_type| (power_up\_states)|$

Updates power up states for analog physical channels.

**Parameters power\_up\_states** (List[nidaqmx.types.AOPowerUpState]) — Contains the physical channels and power up states to set. Each element of the list contains a physical channel and the power up state to set for that physical channel.

- physical\_channel (str): Specifies the physical channel to modify.
- power\_up\_state (float): Specifies the power up state to set for the physical channel specified with the **physical\_channel** input.
- channel\_type (nidaqmx.constants.AOPowerUpOutputBehavior): Specifies the output type for the physical channel specified with the physical\_channel input.

# $\verb|set_digital_logic_family_power_up_state| (\textit{device}\_name, logic\_family)|$

Sets the digital logic family to use when the device powers up.

### **Parameters**

- **device\_name** (str) Specifies the name as configured in MAX of the device to which this operation applies.
- logic\_family (nidaqmx.constants.LogicFamily) Specifies the logic family set to the device to when it powers up. A logic family corresponds to voltage thresholds that are compatible with a group of voltage standards. Refer to device documentation for information on the logic high and logic low voltages for these logic families.

## set\_digital\_power\_up\_states (device\_name, power\_up\_states)

Updates power up states for digital physical channels.

## **Parameters**

- **device\_name** (str) Specifies the name as configured in MAX of the device to which this operation applies.
- power\_up\_states (List[nidaqmx.types.DOPowerUpState]) Contains the physical channels and power up states to set. Each element of the list contains a physical channel and the power up state to set for that physical channel.
  - physical\_channel (str): Specifies the digital line or port to modify. You cannot modify dedicated digital input lines.
  - power\_up\_state (nidaqmx.constants.PowerUpStates): Specifies the power up state to set for the physical channel specified with the **physical\_channel** input.

## set\_digital\_pull\_up\_pull\_down\_states (device\_name, power\_up\_states)

Sets the resistor level to pull up or pull down for lines when they are in tristate logic.

### **Parameters**

- **device\_name** (str) Specifies the name as configured in MAX of the device to which this operation applies.
- power\_up\_states (List[nidaqmx.types.DOResistorPowerUpState]) Contains the physical channels and power up states to set. Each element of the list contains a physical channel and the power up state to set for that physical channel.
  - physical\_channel (str): Specifies the digital line or port to modify. You cannot modify dedicated digital input lines.
  - power\_up\_state (nidaqmx.constants.ResistorState): Specifies the power up state to set for the physical channel specified with the **physical\_channel** input.

#### tasks

nidaqmx.system.\_collections.PersistedTaskCollection - Indicates the collection of saved tasks for this DAQmx system.

## tristate\_output\_term (output\_terminal)

Sets a terminal to high-impedance state. If you connect an external signal to a terminal on the I/O connector, the terminal must be in high-impedance state. Otherwise, the device could double-drive the terminal and damage the hardware. If you use this function on a terminal in an active route, the function fails and returns an error.

**Parameters output\_terminal** (str) – Specifies the terminal on the I/O connector to set to high-impedance state. A DAQmx terminal constant lists all available terminals on installed devices. You also can specify an output terminal by using a string that contains a terminal name.

# nidaqmx.system.collections

## nidagmx.system.device collection

```
class nidaqmx.system._collections.device_collection.DeviceCollection
    Bases: _abcoll.Sequence
```

Contains the collection of devices for a DAQmx system.

This class defines methods that implements a container object.

## device\_names

*List[str]* – Indicates the names of all devices on this device collection.

## nidaqmx.system.persisted\_channel\_collection

```
{\bf class} \ {\tt nidaqmx.system.\_collections.persisted\_channel\_collection.PersistedChannelCollection} \\ {\bf Bases: \_abcoll.Sequence}
```

Contains the collection of global channels for a DAQmx system.

This class defines methods that implements a container object.

### global\_channel\_names

*List[str]* – The names of all the global channels on this collection.

## nidagmx.system.persisted scale collection

Contains the collection of custom scales on a DAQmx system.

This class defines methods that implements a container object.

#### scale names

List[str] – Indicates the names of all the custom scales on this collection.

## nidaqmx.system.persisted\_task\_collection

```
{\bf class} \ {\tt nidaqmx.system.\_collections.persisted\_task\_collection.} {\bf PersistedTaskCollection} \\ {\bf Bases:\_abcoll.Sequence}
```

Contains the collection of task saved on a DAQmx system.

This class defines methods that implements a container object.

#### task names

*List[str]* – Indicates the names of all the tasks on this collection.

## nidaqmx.system.physical\_channel\_collection

Contains the collection of analog input physical channels for a DAQmx device.

This class defines methods that implements a container object.

Contains the collection of analog output physical channels for a DAQmx device.

This class defines methods that implements a container object.

Contains the collection of counter input physical channels for a DAQmx device.

This class defines methods that implements a container object.

Contains the collection of counter output physical channels for a DAQmx device.

This class defines methods that implements a container object.

Contains the collection of digital input lines for a DAQmx device.

This class defines methods that implements a container object.

Contains the collection of digital input ports for a DAQmx device.

This class defines methods that implements a container object.

Contains the collection of digital output lines for a DAQmx device.

This class defines methods that implements a container object.

 $\begin{array}{ll} \textbf{class} \; \texttt{nidaqmx.system.\_collections.physical\_channel\_collection.DOPortsCollection} \; (\textit{device\_name}) \\ \textbf{Bases:} \qquad \qquad \textit{nidaqmx.system.\_collections.physical\_channel\_collection.} \\ \textit{PhysicalChannelCollection} \end{array}$ 

Contains the collection of digital output ports for a DAQmx device.

This class defines methods that implements a container object.

class nidaqmx.system.\_collections.physical\_channel\_collection.PhysicalChannelCollection(device\_ Bases: \_abcoll.Sequence

Contains the collection of physical channels for a DAQmx device.

This class defines methods that implements a container object.

all

*nidaqmx.system.physical\_channel.PhysicalChannel* – Specifies a physical channel object that represents the entire list of physical channels on this channel collection.

### channel\_names

List[str] – Specifies the entire list of physical channels on this collection.

# nidaqmx.system.device

```
class nidaqmx.system.device.Device (name)
    Bases: object

Represents a DAQmx device.
__init__ (name)

    Parameters name (str) - Specifies the name of the device.
__weakref__
    list of weak references to the object (if defined)
```

## accessory\_product\_nums

*List[int]* – Indicates the unique hardware identification number for accessories connected to the device. Each list element corresponds to a connector. For example, index 0 corresponds to connector 0. The list contains 0 for each connector with no accessory connected.

### accessory\_product\_types

List[str] - Indicates the model names of accessories connected to the device. Each list element corresponds

to a connector. For example, index 0 corresponds to connector 0. The list contains an empty string for each connector with no accessory connected.

## accessory\_serial\_nums

List[int] – Indicates the serial number for accessories connected to the device. Each list element corresponds to a connector. For example, index 0 corresponds to connector 0. The list contains 0 for each connector with no accessory connected.

static add\_network\_device (ip\_address, device\_name=u'', attempt\_reservation=False, timeout=10.0)

Adds a Network cDAQ device to the system and, if specified, attempts to reserve it.

### **Parameters**

- **ip\_address** (*str*) Specifies the string containing the IP address (in dotted decimal notation) or hostname of the device to add to the system.
- **device\_name** (Optional[str]) Indicates the name to assign to the device. If unspecified, NI-DAQmx chooses the device name.
- attempt\_reservation (Optional[bool]) Indicates if a reservation should be attempted after the device is successfully added. By default, this parameter is set to False.
- **timeout** (Optional[float]) Specifies the time in seconds to wait for the device to respond before timing out.

**Returns** Specifies the object that represents the device this operation applied to.

**Return type** *nidagmx.system.device.Device* 

### ai\_bridge\_rngs

*List[float]* – Indicates pairs of input voltage ratio ranges, in volts per volt, supported by devices that acquire using ratiometric measurements. Each pair consists of the low value followed by the high value.

#### ai\_charge\_rngs

*List[float]* – Indicates in coulombs pairs of input charge ranges for the device. Each pair consists of the low value followed by the high value.

#### ai couplings

List[nidagmx.constants.Coupling] – Indicates the coupling types supported by this device.

## ai\_current\_int\_excit\_discrete\_vals

List[float] – Indicates the set of discrete internal current excitation values supported by this device.

## ai current rngs

*List[float]* – Indicates the pairs of current input ranges supported by this device. Each pair consists of the low value, followed by the high value.

## ai\_dig\_fltr\_lowpass\_cutoff\_freq\_discrete\_vals

List[float] – Indicates the set of discrete lowpass cutoff frequencies supported by this device. If the device supports ranges of lowpass cutoff frequencies, use AI.DigFltr.Lowpass.CutoffFreq.RangeVals to determine supported frequencies.

## $\verb"ai_dig_fltr_lowpass_cutoff_freq_range_vals"$

List[float] – Indicates pairs of lowpass cutoff frequency ranges supported by this device. Each pair consists of the low value, followed by the high value. If the device supports a set of discrete lowpass cutoff frequencies, use AI.DigFltr.Lowpass.CutoffFreq.DiscreteVals to determine the supported frequencies.

## ai\_dig\_fltr\_types

List[nidaqmx.constants.FilterType] - Indicates the AI digital filter types supported by the device.

### ai freq rnqs

*List[float]* – Indicates the pairs of frequency input ranges supported by this device. Each pair consists of the low value, followed by the high value.

## ai\_gains

*List[float]* – Indicates the input gain settings supported by this device.

### ai lowpass cutoff freq discrete vals

*List[float]* – Indicates the set of discrete lowpass cutoff frequencies supported by this device. If the device supports ranges of lowpass cutoff frequencies, use **ai\_lowpass\_cutoff\_freq\_range\_vals** to determine supported frequencies.

## ai\_lowpass\_cutoff\_freq\_range\_vals

List[float] – Indicates pairs of lowpass cutoff frequency ranges supported by this device. Each pair consists of the low value, followed by the high value. If the device supports a set of discrete lowpass cutoff frequencies, use ai\_lowpass\_cutoff\_freq\_discrete\_vals to determine the supported frequencies.

### ai\_max\_multi\_chan\_rate

float – Indicates the maximum sampling rate for an analog input task from this device. To find the maximum rate for the task, take the minimum of **ai\_max\_single\_chan\_rate** or the indicated sampling rate of this device divided by the number of channels to acquire data from (including cold-junction compensation and autozero channels).

## ai\_max\_single\_chan\_rate

float – Indicates the maximum rate for an analog input task if the task contains only a single channel from this device.

## ai\_meas\_types

List[nidaqmx.constants.UsageTypeAI] – Indicates the measurement types supported by the physical channels of the device. Refer to ai\_meas\_types for information on specific channels.

## ai\_min\_rate

*float* – Indicates the minimum rate for an analog input task on this device. NI-DAQmx returns a warning or error if you attempt to sample at a slower rate.

## ai\_physical\_chans

*List[nidaqmx.system.\_collections.PhysicalChannelCollection]* – Indicates a collection that contains all the analog input physical channels available on the device.

## ai\_resistance\_rngs

*List[float]* – Indicates pairs of input resistance ranges, in ohms, supported by devices that have the necessary signal conditioning to measure resistances. Each pair consists of the low value followed by the high value.

## ai\_samp\_modes

List[nidaqmx.constants.AcquisitionType] – Indicates sample modes supported by devices that support sample clocked analog input.

## ai\_simultaneous\_sampling\_supported

bool – Indicates if the device supports simultaneous sampling.

## ai\_trig\_usage

List[nidaqmx.constants.TriggerUsage] – Indicates the triggers supported by this device for an analog input task.

## ai\_voltage\_int\_excit\_discrete\_vals

List[float] – Indicates the set of discrete internal voltage excitation values supported by this device. If the device supports ranges of internal excitation values, use ai\_voltage\_int\_excit\_range\_vals to determine supported excitation values.

## ai\_voltage\_int\_excit\_range\_vals

List[float] – Indicates pairs of internal voltage excitation ranges supported by this device. Each pair consists of the low value, followed by the high value. If the device supports a set of discrete internal excitation values, use ai\_voltage\_int\_excit\_discrete\_vals to determine the supported excitation values.

## ai\_voltage\_rngs

*List[float]* – Indicates pairs of input voltage ranges supported by this device. Each pair consists of the low value, followed by the high value.

## anlg\_trig\_supported

bool - Indicates if the device supports analog triggering.

#### ao\_current\_rngs

*List[float]* – Indicates pairs of output current ranges supported by this device. Each pair consists of the low value, followed by the high value.

### ao\_gains

List[float] – Indicates the output gain settings supported by this device.

#### ao\_max\_rate

float – Indicates the maximum analog output rate of the device.

#### ao min rate

float - Indicates the minimum analog output rate of the device.

## ao\_output\_types

List[nidaqmx.constants.UsageTypeAO] – Indicates the generation types supported by the physical channels of the device. Refer to **ao\_output\_types** for information on specific channels.

## ao\_physical\_chans

*List[nidaqmx.system.\_collections.PhysicalChannelCollection]* – Indicates a collection that contains all the analog output physical channels available on the device.

## ao\_samp\_clk\_supported

bool – Indicates if the device supports the sample clock timing type for analog output tasks.

## ao\_samp\_modes

List[nidaqmx.constants.AcquisitionType] – Indicates sample modes supported by devices that support sample clocked analog output.

## ao\_trig\_usage

List[nidaqmx.constants.TriggerUsage] - Indicates the triggers supported by this device for analog output tasks.

## ao\_voltage\_rngs

List[float] – Indicates pairs of output voltage ranges supported by this device. Each pair consists of the low value, followed by the high value.

### bus\_type

nidaqmx.constants.BusType - Indicates the bus type of the device.

### carrier serial num

*int* – Indicates the serial number of the device carrier. This value is zero if the carrier does not have a serial number.

#### chassis module devices

List[nidaqmx.system.device.Device] - Indicates a list containing the names of the modules in the chassis.

## ci\_max\_size

int – Indicates in bits the size of the counters on the device.

#### ci max timebase

*float* – Indicates in hertz the maximum counter timebase frequency.

## ci\_meas\_types

List[nidaqmx.constants.UsageTypeCI] – Indicates the measurement types supported by the physical channels of the device. Refer to **ci\_meas\_types** for information on specific channels.

#### ci physical chans

*List[nidaqmx.system.\_collections.PhysicalChannelCollection]* – Indicates a collection that contains all the counter input physical channels available on the device.

## ci\_samp\_clk\_supported

bool – Indicates if the device supports the sample clock timing type for counter input tasks.

## ci\_samp\_modes

List[nidaqmx.constants.AcquisitionType] – Indicates sample modes supported by devices that support sample clocked counter input.

## ci\_trig\_usage

List[nidaqmx.constants.TriggerUsage] - Indicates the triggers supported by this device for counter input tasks.

#### co max size

int – Indicates in bits the size of the counters on the device.

### co\_max\_timebase

*float* – Indicates in hertz the maximum counter timebase frequency.

## co\_output\_types

List[nidaqmx.constants.UsageTypeCO] – Indicates the generation types supported by the physical channels of the device. Refer to **co\_output\_types** for information on specific channels.

## co\_physical\_chans

*List[nidaqmx.system.\_collections.PhysicalChannelCollection]* – Indicates a collection that contains all the counter output physical channels available on the device.

## co\_samp\_clk\_supported

bool – Indicates if the device supports Sample Clock timing for counter output tasks.

#### co\_samp\_modes

 $List[\textit{nidaqmx.constants.AcquisitionType}] - Indicates \ sample \ modes \ supported \ by \ devices \ that \ support \ sample \ clocked \ counter \ output.$ 

## co\_trig\_usage

List[nidaqmx.constants.TriggerUsage] - Indicates the triggers supported by this device for counter output tasks.

## compact dag chassis device

nidaqmx.system.device.Device – Indicates the name of the CompactDAQ chassis that contains this module.

## ${\tt compact\_daq\_slot\_num}$

int – Indicates the slot number in which this module is located in the CompactDAQ chassis.

### delete\_network\_device()

Deletes a Network DAQ device previously added to the host. If the device is reserved, it is unreserved before it is removed.

## dev\_is\_simulated

bool - Indicates if the device is a simulated device.

#### dev serial num

*int* – Indicates the serial number of the device. This value is zero if the device does not have a serial number.

### di lines

*List[nidaqmx.system.\_collections.PhysicalChannelCollection]* – Indicates a collection that contains all the digital input lines available on the device.

#### di max rate

float – Indicates the maximum digital input rate of the device.

### di\_ports

*List[nidaqmx.system.\_collections.PhysicalChannelCollection]* – Indicates a collection that contains all the digital input ports available on the device.

### di\_trig\_usage

List[nidaqmx.constants.TriggerUsage] - Indicates the triggers supported by this device for digital input tasks.

### dig\_trig\_supported

bool – Indicates if the device supports digital triggering.

### do lines

List[nidaqmx.system.\_collections.PhysicalChannelCollection] – Indicates a collection that contains all the digital output lines available on the device.

#### do max rate

float – Indicates the maximum digital output rate of the device.

## do\_ports

*List[nidaqmx.system.\_collections.PhysicalChannelCollection]* – Indicates a collection that contains all the digital output ports available on the device.

## do\_trig\_usage

List[nidaqmx.constants.TriggerUsage] - Indicates the triggers supported by this device for digital output tasks.

#### name

*str* – Specifies the name of this device.

## num\_dma\_chans

int – Indicates the number of DMA channels on the device.

## pci bus num

int - Indicates the PCI bus number of the device.

#### pci dev num

int – Indicates the PCI slot number of the device.

### product\_category

nidaqmx.constants.ProductCategory - Indicates the product category of the device. This category corresponds to the category displayed in MAX when creating NI-DAQmx simulated devices.

## product\_num

int – Indicates the unique hardware identification number for the device.

### product\_type

str – Indicates the product name of the device.

### pxi\_chassis\_num

int – Indicates the PXI chassis number of the device, as identified in MAX.

#### pxi slot num

int – Indicates the PXI slot number of the device.

## reserve\_network\_device (override\_reservation=None)

Reserves the Network DAQ device for the current host. Reservation is required to run NI-DAQmx tasks, and the device must be added in MAX before it can be reserved.

**Parameters override\_reservation** (Optional[bool]) - Indicates if an existing reservation on the device should be overridden by this reservation. By default, this parameter is set to false.

### reset\_device()

Immediately aborts all active tasks associated with a device, disconnects any routes, and returns the device to an initialized state. Aborting a task immediately terminates the currently active operation, such as a read or a write. Aborting a task puts the task into an unstable but recoverable state. To recover the task, use DAQmx Start to restart the task or use DAQmx Stop to reset the task without starting it.

## self\_test\_device()

Performs a brief test of device resources. If a failure occurs, refer to your device documentation for more information.

### tcpip\_ethernet\_ip

str – Indicates the IPv4 address of the Ethernet interface in dotted decimal format. This property returns 0.0.0.0 if the Ethernet interface cannot acquire an address.

### tcpip\_hostname

str – Indicates the IPv4 hostname of the device.

### tcpip wireless ip

*str* – Indicates the IPv4 address of the 802.11 wireless interface in dotted decimal format. This property returns 0.0.0.0 if the wireless interface cannot acquire an address.

## tedshwteds\_supported

bool - Indicates whether the device supports hardware TEDS.

#### terminals

List[str] – Indicates a list of all terminals on the device.

## unreserve\_network\_device()

Unreserves or releases a Network DAQ device previously reserved by the host.

# nidaqmx.system.physical\_channel

```
class nidaqmx.system.physical_channel.PhysicalChannel (name)
    Bases: object

Represents a DAQmx physical channel.
    __init__ (name)

    Parameters name (str) - Specifies the name of the physical channel.
    __weakref__
```

list of weak references to the object (if defined)

### ai input srcs

*List[str]* – Indicates the list of input sources supported by the channel. Channels may support using the signal from the I/O connector or one of several calibration signals.

#### ai\_meas\_types

List[nidaqmx.constants.UsageTypeAI] - Indicates the measurement types supported by the channel.

#### ai\_term\_cfgs

List[nidaqmx.constants.TerminalConfiguration] – Indicates the list of terminal configurations supported by the channel.

# ao\_manual\_control\_amplitude

float – Indicates the current value of the front panel amplitude control for the physical channel in volts.

#### ao\_manual\_control\_enable

*bool* – Specifies if you can control the physical channel externally via a manual control located on the device. You cannot simultaneously control a channel manually and with NI-DAOmx.

#### ao\_manual\_control\_freq

float - Indicates the current value of the front panel frequency control for the physical channel in hertz.

# ao\_manual\_control\_short\_detected

bool – Indicates whether the physical channel is currently disabled due to a short detected on the channel.

#### ao\_output\_types

List[nidagmx.constants.UsageTypeAO] – Indicates the output types supported by the channel.

# ao\_power\_amp\_channel\_enable

*bool* – Specifies whether to enable or disable a channel for amplification. This property can also be used to check if a channel is enabled.

# ao\_power\_amp\_gain

*float* – Indicates the calibrated gain of the channel.

## ao\_power\_amp\_offset

*float* – Indicates the calibrated offset of the channel in volts.

# ao\_power\_amp\_overcurrent

bool - Indicates if the channel detected an overcurrent condition.

# ao\_power\_amp\_scaling\_coeff

List[float] - Indicates the coefficients of a polynomial equation used to scale from pre-amplified values.

# ao\_power\_up\_output\_types

List[nidaqmx.constants.AOPowerUpOutputBehavior] - Indicates the power up output types supported by the channel.

# ao\_term\_cfgs

List[nidaqmx.constants.TerminalConfiguration] – Indicates the list of terminal configurations supported by the channel.

#### ci\_meas\_types

 $List[\textit{nidaqmx.constants.UsageTypeCI}] - Indicates \ the \ measurement \ types \ supported \ by \ the \ channel.$ 

# clear\_teds()

Removes TEDS information from the physical channel you specify. This function temporarily overrides any TEDS configuration for the physical channel that you performed in MAX.

# co\_output\_types

List[nidaqmx.constants.UsageTypeCO] - Indicates the output types supported by the channel.

# configure\_teds (file\_path=u'')

Associates TEDS information with the physical channel you specify. If you do not specify the filename of a data sheet in the **file path** input, this function attempts to find a TEDS sensor connected to the physical

channel. This function temporarily overrides any TEDS configuration for the physical channel that you performed in MAX.

**Parameters file\_path** (Optional[str]) – Is the path to a Virtual TEDS data sheet that you want to associate with the physical channel. If you do not specify anything for this input, this function attempts to find a TEDS sensor connected to the physical channel.

#### di change detect supported

bool – Indicates if the change detection timing type is supported for the digital input physical channel.

# di\_port\_width

*int* – Indicates in bits the width of digital input port.

#### di\_samp\_clk\_supported

bool – Indicates if the sample clock timing type is supported for the digital input physical channel.

# di\_samp\_modes

List[nidaqmx.constants.AcquisitionType] – Indicates the sample modes supported by devices that support sample clocked digital input.

#### do port width

int – Indicates in bits the width of digital output port.

# do\_samp\_clk\_supported

bool - Indicates if the sample clock timing type is supported for the digital output physical channel.

#### do\_samp\_modes

List[nidaqmx.constants.AcquisitionType] – Indicates the sample modes supported by devices that support sample clocked digital output.

#### name

str – Specifies the name of this physical channel.

# teds\_bit\_stream

*List[int]* – Indicates the TEDS binary bitstream without checksums.

#### teds mfg id

int – Indicates the manufacturer ID of the sensor.

#### teds\_model\_num

int – Indicates the model number of the sensor.

# teds serial num

int – Indicates the serial number of the sensor.

#### teds template ids

*List[int]* – Indicates the IDs of the templates in the bitstream in **teds\_bit\_stream**.

#### teds version letter

*str* – Indicates the version letter of the sensor.

# teds\_version\_num

int – Indicates the version number of the sensor.

# 

Writes data from a 1D list of 8-bit unsigned integers to the TEDS sensor.

# **Parameters**

• **bit\_stream** (Optional[List[int]]) – Is the TEDS bitstream to write to the sensor. This bitstream must be constructed according to the IEEE 1451.4 specification.

• basic\_teds\_options (Optional[nidaqmx.constants. WriteBasicTEDSOptions]) - Specifies how to handle basic TEDS data in the hitstream

Writes data from a virtual TEDS file to the TEDS sensor.

#### **Parameters**

- **file\_path** (Optional[str]) Specifies the filename of a virtual TEDS file that contains the bitstream to write.
- basic\_teds\_options (Optional[nidaqmx.constants. WriteBasicTEDSOptions]) Specifies how to handle basic TEDS data in the bitstream.

# nidagmx.system.storage

# nidaqmx.system.persisted\_channel

```
class nidaqmx.system.storage.persisted_channel.PersistedChannel(name)
    Bases; object
```

Represents a saved DAQmx global channel.

Use the DAQmx Persisted Channel properties to query information about programmatically saved global channels.

```
___init___(name)
```

**Parameters** name – Specifies the name of the global channel.

```
weakref
```

list of weak references to the object (if defined)

#### allow interactive deletion

bool – Indicates whether the global channel can be deleted through MAX.

# allow\_interactive\_editing

bool - Indicates whether the global channel can be edited in the DAQ Assistant.

# author

str – Indicates the author of the global channel.

#### delete()

Deletes this global channel from MAX.

This function does not remove the global channel from tasks that use it.

# nidaqmx.system.persisted\_scale

```
class nidaqmx.system.storage.persisted_scale.PersistedScale(name)
    Bases: object
    Represents a saved DAQmx custom scale.
    Use the DAQmx Persisted Scale properties to query information about programmatically saved custom scales.
```

\_\_\_init\_\_\_(name)

**Parameters** name – Specifies the name of the saved scale.

#### \_\_weakref\_

list of weak references to the object (if defined)

#### allow interactive deletion

bool – Indicates whether the custom scale can be deleted through MAX.

#### allow\_interactive\_editing

bool – Indicates whether the custom scale can be edited in the DAQ Assistant.

#### author

str – Indicates the author of the custom scale.

#### delete()

Deletes this custom scale from MAX.

This function does not remove the custom scale from virtual channels that use it.

#### load()

Loads this custom scale.

**Returns** Indicates the loaded Scale object.

Return type nidaqmx.scale.Scale

# nidagmx.system.persisted task

```
class nidaqmx.system.storage.persisted_task.PersistedTask(name)
```

Bases: object

Represents a saved DAQmx task.

Use the DAQmx Persisted Task properties to query information about programmatically saved tasks.

```
___init___(name)
```

**Parameters** name – Specifies the name of the saved task.

## \_\_weakref\_

list of weak references to the object (if defined)

# allow\_interactive\_deletion

bool – Indicates whether the task can be deleted through MAX.

#### allow interactive editing

bool - Indicates whether the task can be edited in the DAQ Assistant.

# author

str – Indicates the author of the task.

# ${\tt delete}()$

Deletes this task from MAX.

This function does not clear the copy of the task stored in memory. Use the DAQmx Clear Task function to clear that copy of the task.

# load()

Loads this saved task.

If you use this function to load a task, you must use DAQmx Clear Task to destroy it.

**Returns** Indicates the loaded Task object.

**Return type** *nidagmx.task.Task* 

# nidagmx.system.watchdog

class nidaqmx.system.watchdog.WatchdogTask(device\_name, task\_name=u'', timeout=10)
Bases: object

Represents the watchdog configurations for a DAQmx task.

\_\_init\_\_ (device\_name, task\_name=u'', timeout=10)

Creates and configures a task that controls the watchdog timer of a device. The timer activates when you start the task.

Use the DAQmx Configure Watchdog Expiration States functions to configure channel expiration states. This class does not program the watchdog timer on a real-time controller.

#### **Parameters**

- **device\_name** (str) Specifies is the name as configured in MAX of the device to which this operation applies.
- task\_name (str) Specifies the name to assign to the task. If you use this constructor in a loop and specify a name for the task, you must use the DAQmx Clear Task method within the loop after you are finished with the task. Otherwise, NI-DAQmx attempts to create multiple tasks with the same name, which results in an error.
- **timeout** (float) Specifies the amount of time in seconds until the watchdog timer expires. A value of -1 means the internal timer never expires. Set this input to -1 if you use an Expiration Trigger to expire the watchdog task. If this time elapses, the device sets the physical channels to the states you specify with the digital physical channel expiration states input.

# \_\_weakref\_

list of weak references to the object (if defined)

## cfg\_watchdog\_ao\_expir\_states (expiration\_states)

Configures the expiration states for an analog watchdog timer task.

Parameters expiration\_states - (List[nidaqmx.system.watchdog.AOExpirationState]):
Contains the states to which to set analog physical channels when the watchdog timer expires. Each element of the list contains an analog physical channel name, the corresponding expiration state, and the output type for that analog physical channel. The units of "expiration state" must be specified in volts for an analog output voltage expiration state, or amps for an analog output current expiration state.

physical\_channel (str): Specifies the analog output channel to modify. You cannot modify dedicated analog input lines.

**expiration\_state (float): Specifies the value to set the** channel to upon expiration.

**output\_type** (**nidaqmx.constants.WatchdogAOExpirState**): Specifies the output type of the physical channel.

**Returns** Indicates the list of objects representing the configured expiration states.

**Return type** List[nidaqmx.system.\_watchdog\_modules.expiration\_state.ExpirationState]

#### cfg watchdog co expir states(expiration states)

Configures the expiration states for a counter watchdog timer task.

**Parameters expiration\_states** – (List[nidaqmx.system.watchdog.COExpirationState]): Contains the states to which to set counter physical channels when the watchdog timer expires. Each element of the list contains a counter physical channel name and the corresponding state for that counter physical channel.

**physical\_channel (str): Specifies the counter output channel to** modify. You cannot modify dedicated counter input lines.

**expiration\_state** (**nidaqmx.constants.WatchdogCOExpirState**): Specifies the value to set the channel to upon expiration.

**Returns** Indicates the list of objects representing the configured expiration states.

**Return type** List[nidaqmx.system.\_watchdog\_modules.expiration\_state.ExpirationState]

# cfg\_watchdog\_do\_expir\_states(expiration\_states)

Configures the expiration states for a digital watchdog timer task.

**Parameters expiration\_states** – (List[nidaqmx.system.watchdog.DOExpirationState]): Contains the states to which to set digital physical channels when the watchdog timer expires. Each element of the list contains a digital physical channel name and the corresponding state for that digital physical channel.

**physical\_channel (str): Specifies the digital output channel to** modify. You cannot modify dedicated digital input lines.

**expiration\_state** (**nidaqmx.constants.Level**): **Specifies the** value to set the channel to upon expiration.

**Returns** Indicates the list of objects representing the configured expiration states.

**Return type** List[nidaqmx.system.\_watchdog\_modules.expiration\_state.ExpirationState]

## clear\_expiration()

Unlock a device whose watchdog timer expired.

This function does not program the watchdog timer on a real-time controller. Use the Real-Time Watchdog VIs to program the watchdog timer on a real-time controller.

# close()

Clears the task.

Before clearing, this method aborts the task, if necessary, and releases any resources the task reserved. You cannot use a task after you clear it unless you recreate the task.

If you create a DAQmx Task object within a loop, use this method within the loop after you are finished with the task to avoid allocating unnecessary memory.

# control (action)

Alters the state of a task according to the action you specify.

Parameters action (nidaqmx.constants.TaskMode) - Specifies how to alter the task state.

#### expir\_trig\_dig\_edge\_edge

nidaqmx.constants.Edge - Specifies on which edge of a digital signal to expire the watchdog task.

# expir\_trig\_dig\_edge\_src

str – Specifies the name of a terminal where a digital signal exists to use as the source of the Expiration Trigger.

# expir\_trig\_trig\_on\_network\_conn\_loss

bool – Specifies the watchdog timer behavior when the network connection is lost between the host and the chassis. If set to true, the watchdog timer expires when the chassis detects the loss of network connection.

# expir\_trig\_trig\_type

nidaqmx.constants.TriggerType-Specifies the type of trigger to use to expire a watchdog task.

# expiration\_states

# **nidaqmx.system.\_watchdog\_modules.expiration\_states\_collection.** ExpirationStatesCollection:

Gets the collection of expiration states for this watchdog task.

## expired

bool – Indicates if the watchdog timer expired. You can read this property only while the task is running.

#### name

str – Indicates the name of the task.

# reset\_timer()

Reset the internal timer. You must continually reset the internal timer to prevent it from timing out and locking the device.

This function does not program the watchdog timer on a real-time controller. Use the Real-Time Watchdog VIs to program the watchdog timer on a real-time controller.

#### start()

Transitions the task to the running state to begin the measurement or generation. Using this method is required for some applications and is optional for others.

#### stop()

Stops the task and returns it to the state the task was in before the DAQmx Start Task method ran.

#### timeout

*float* – Specifies in seconds the amount of time until the watchdog timer expires. A value of -1 means the internal timer never expires. Set this input to -1 if you use an Expiration Trigger to expire the watchdog task

# nidagmx.system.expiration state

 $\begin{array}{c} \textbf{class} \; \texttt{nidaqmx.system.\_watchdog\_modules.expiration\_state.ExpirationState} \; (\textit{task\_handle}, \\ physical\_channel) \end{array}$ 

Bases: object

Represents a DAQmx Watchdog expiration state.

#### expir\_states\_ao\_state

float – Specifies the state to set the analog output physical channels when the watchdog task expires.

# expir\_states\_ao\_type

nidaqmx.constants.WatchdogAOExpirState - Specifies the output type of the analog output physical channels when the watchdog task expires.

#### expir\_states\_co\_state

nidaqmx.constants.WatchdogCOExpirState - Specifies the state to set the counter output channel terminal when the watchdog task expires.

# expir\_states\_do\_state

nidaqmx.constants.Level – Specifies the state to which to set the digital physical channels when the watchdog task expires. You cannot modify the expiration state of dedicated digital input physical channels.

#### nidagmx.system.expiration states collection

class nidaqmx.system.\_watchdog\_modules.expiration\_states\_collection.ExpirationStatesCollection
Bases: object

Contains the collection of expiration states for a DAQmx Watchdog Task.

This class defines methods that implements a container object.

# nidaqmx.task

**Parameters** new\_task\_name (Optional[str]) - Specifies the name to assign to the task.

If you use this method in a loop and specify a name for the task, you must use the DAQmx Clear Task method within the loop after you are finished with the task. Otherwise, NI-DAQmx attempts to create multiple tasks with the same name, which results in an error.

#### \_\_weakref\_

list of weak references to the object (if defined)

# add\_global\_channels(global\_channels)

Adds global virtual channels from MAX to the given task.

# Parameters global\_channels (List[nidaqmx.system.storage. persisted\_channel.PersistedChannel]) - Specifies the channels to add to the task.

These channels must be valid channels available from MAX. If you pass an invalid channel, NI-DAQmx returns an error. This value is ignored if it is empty.

#### ai channels

nidaqmx.\_task\_modules.ai\_channel\_collection.AIChannelCollection - Gets the collection of analog input channels for this task.

#### ao\_channels

nidaqmx.\_task\_modules.ao\_channel\_collection.AOChannelCollection - Gets the collection of analog output channels for this task.

## channel names

List[str] – Indicates the names of all virtual channels in the task.

#### channels

nidaqmx.\_task\_modules.channels.channel.Channel - Specifies a channel object that represents the entire list of virtual channels in this task.

# ci\_channels

 $\label{lem:collection} \emph{nidaqmx.\_task\_modules.ci\_channel\_collection.CIChannelCollection} - \textbf{Gets the collection of counter input channels for this task.}$ 

# close()

Clears the task.

Before clearing, this method aborts the task, if necessary, and releases any resources the task reserved. You cannot use a task after you clear it unless you recreate the task.

If you create a DAQmx Task object within a loop, use this method within the loop after you are finished with the task to avoid allocating unnecessary memory.

#### co channels

nidaqmx.\_task\_modules.co\_channel\_collection.COChannelCollection - Gets the collection of counter output channels for this task.

#### control (action)

Alters the state of a task according to the action you specify.

Parameters action (nidaqmx.constants.TaskMode) - Specifies how to alter the task state.

# devices

List[nidaqmx.system.device.Device] - Indicates a list of Device objects representing all the devices in the task.

#### di channels

nidaqmx.\_task\_modules.di\_channel\_collection.DIChannelCollection - Gets the collection of digital input channels for this task.

#### do channels

nidaqmx.\_task\_modules.do\_channel\_collection.DOChannelCollection - Gets the collection of digital output channels for this task.

#### export\_signals

 $nidaqmx.\_task\_modules.export\_signals.ExportSignals$  — Gets the exported signal configurations for the task.

#### in stream

nidaqmx.\_task\_modules.in\_stream.InStream - Gets the read configurations for the task.

# is\_task\_done()

Queries the status of the task and indicates if it completed execution. Use this function to ensure that the specified operation is complete before you stop the task.

**Returns** Indicates if the measurement or generation completed.

**Return type** bool

#### name

str – Indicates the name of the task.

#### number\_of\_channels

*int* – Indicates the number of virtual channels in the task.

#### number of devices

int – Indicates the number of devices in the task.

#### out stream

 $nidaqmx.\_task\_modules.out\_stream.OutStream$  — Gets the write configurations for the task.

read (number\_of\_samples\_per\_channel=<nidaqmx.task.UnsetNumSamplesSentinel object>, timeout=10.0)

Reads samples from the task or virtual channels you specify.

This read method is dynamic, and is capable of inferring an appropriate return type based on these factors:
- The channel type of the task. - The number of channels to read. - The number of samples per channel.

The data type of the samples returned is independently determined by the channel type of the task.

For digital input measurements, the data type of the samples returned is determined by the line grouping format of the digital lines. If the line grouping format is set to "one channel for all lines", the data type of the samples returned is int. If the line grouping format is set to "one channel per line", the data type of the samples returned is boolean.

If you do not set the number of samples per channel, this method assumes one sample was requested. This method then returns either a scalar (1 channel to read) or a list (N channels to read).

If you set the number of samples per channel to ANY value (even 1), this method assumes multiple samples were requested. This method then returns either a list (1 channel to read) or a list of lists (N channels to read).

#### **Parameters**

• number\_of\_samples\_per\_channel (Optional[int]) - Specifies the number of samples to read. If this input is not set, assumes samples to read is 1. Conversely, if this input is set, assumes there are multiple samples to read.

If you set this input to nidaqmx.constants. READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.constants.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to True, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

• **timeout** (Optional[float]) - Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

#### Returns

The samples requested in the form of a scalar, a list, or a list of lists. See method docstring for more info.

NI-DAQmx scales the data to the units of the measurement, including any custom scaling you apply to the channels. Use a DAQmx Create Channel method to specify these units.

# Return type dynamic

# **Example**

```
>>> task = Task()
>>> task.ai_channels.add_voltage_channel('Dev1/ai0:3')
>>> data = task.read()
>>> type(data)
<type 'list'>
>>> type(data[0])
<type 'float'>
```

## register\_done\_event (callback\_method)

Registers a callback function to receive an event when a task stops due to an error or when a finite acquisition task or finite generation task completes execution. A Done event does not occur when a task is stopped explicitly, such as by calling DAQmx Stop Task.

**Parameters callback\_method** (function) - Specifies the function that you want DAQmx to call when the event occurs. The function you pass in this parameter must have the following prototype:

```
>>> def callback(task_handle, status, callback_data):
>>> return 0
```

Upon entry to the callback, the task\_handle parameter contains the handle to the task on which the event occurred. The status parameter contains the status of the task when the event occurred. If the status value is negative, it indicates an error. If the status value is zero, it indicates no error. If the status value is positive, it indicates a warning. The callbackData parameter contains the value you passed in the callbackData parameter of this function.

Passing None for this parameter unregisters the event callback function.

```
register_every_n_samples_acquired_into_buffer_event (sample_interval, back method) call-
```

Registers a callback function to receive an event when the specified number  $o\bar{f}$  samples is written from the device to the buffer. This function only works with devices that support buffered tasks.

When you stop a task explicitly any pending events are discarded. For example, if you call DAQmx Stop Task then you do not receive any pending events.

#### **Parameters**

- **sample\_interval** (*int*) Specifies the number of samples after which each event should occur.
- **callback\_method** (function) Specifies the function that you want DAQmx to call when the event occurs. The function you pass in this parameter must have the following prototype:

```
>>> def callback(task_handle, every_n_samples_event_type,
>>> number_of_samples, callback_data):
>>> return 0
```

Upon entry to the callback, the task\_handle parameter contains the handle to the task on which the event occurred. The every\_n\_samples\_event\_type parameter contains the EveryNSamplesEventType.ACQUIRED\_INTO\_BUFFER value. The number\_of\_samples parameter contains the value you passed in the sample\_interval parameter of this function. The callback\_data parameter contains the value you passed in the callback\_data parameter of this function.

Passing None for this parameter unregisters the event callback function.

```
register_every_n_samples_transferred_from_buffer_event (sample_interval, call-
back method)
```

Registers a callback function to receive an event when the specified number of samples is written from the buffer to the device. This function only works with devices that support buffered tasks.

When you stop a task explicitly any pending events are discarded. For example, if you call DAQmx Stop Task then you do not receive any pending events.

# **Parameters**

- **sample\_interval** (*int*) Specifies the number of samples after which each event should occur.
- callback\_method (function) Specifies the function that you want DAQmx to call when the event occurs. The function you pass in this parameter must have the following prototype:

```
>>> def callback(task_handle, every_n_samples_event_type,
>>> number_of_samples, callback_data):
>>> return 0
```

Upon entry to the callback, the task\_handle parameter contains the handle to the task on which the event occurred. The every\_n\_samples\_event\_type parameter contains the EveryNSamplesEventType.TRANSFERRED\_FROM\_BUFFER value. The number\_of\_samples parameter contains the value you passed in the sample\_interval parameter of this function. The callback\_data parameter contains the value you passed in the callback\_data parameter of this function.

Passing None for this parameter unregisters the event callback function.

```
register_signal_event (signal_type, callback_method)
```

Registers a callback function to receive an event when the specified hardware event occurs.

When you stop a task explicitly any pending events are discarded. For example, if you call DAQmx Stop Task then you do not receive any pending events.

#### **Parameters**

- signal\_type (nidaqmx.constants.Signal) Specifies the type of signal for which you want to receive results.
- **callback\_method** (function) Specifies the function that you want DAQmx to call when the event occurs. The function you pass in this parameter must have the following prototype:

```
>>> def callback(task_handle, signal_type, callback_data):
>>> return 0
```

Upon entry to the callback, the task\_handle parameter contains the handle to the task on which the event occurred. The signal\_type parameter contains the integer value you passed in the signal\_type parameter of this function. The callback\_data parameter contains the value you passed in the callback\_data parameter of this function.

Passing None for this parameter unregisters the event callback function.

save (save\_as=u'', author=u'', overwrite\_existing\_task=False, allow\_interactive\_editing=True, allow\_interactive\_deletion=True)
Saves this task and any local channels it contains to MAX.

This function does not save global channels. Use the DAQmx Save Global Channel function to save global channels.

#### **Parameters**

- **save\_as** (Optional[str]) Is the name to save the task, global channel, or custom scale as. If you do not specify a value for this input, NI-DAQmx uses the name currently assigned to the task, global channel, or custom scale.
- author (Optional[str]) Is a name to store with the task, global channel, or custom scale.
- overwrite\_existing\_task (Optional[bool]) Specifies whether to overwrite a task of the same name if one is already saved in MAX. If this input is False and a task of the same name is already saved in MAX, this function returns an error.
- allow\_interactive\_editing (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be edited in the DAQ Assistant. If al-

low\_interactive\_editing is True, the DAQ Assistant must support all task or global channel settings.

• allow\_interactive\_deletion (Optional[bool]) - Specifies whether to allow the task, global channel, or custom scale to be deleted through MAX.

# start()

Transitions the task to the running state to begin the measurement or generation. Using this method is required for some applications and is optional for others.

If you do not use this method, a measurement task starts automatically when the DAQmx Read method runs. The autostart input of the DAQmx Write method determines if a generation task starts automatically when the DAQmx Write method runs.

If you do not use the DAQmx Start Task method and the DAQmx Stop Task method when you use the DAQmx Read method or the DAQmx Write method multiple times, such as in a loop, the task starts and stops repeatedly. Starting and stopping a task repeatedly reduces the performance of the application.

#### stop()

Stops the task and returns it to the state the task was in before the DAQmx Start Task method ran or the DAQmx Write method ran with the autostart input set to TRUE.

If you do not use the DAQmx Start Task method and the DAQmx Stop Task method when you use the DAQmx Read method or the DAQmx Write method multiple times, such as in a loop, the task starts and stops repeatedly. Starting and stopping a task repeatedly reduces the performance of the application.

#### timing

nidaqmx.\_task\_modules.timing.Timing - Gets the timing configurations for the task.

# triggers

nidagmx.\_task\_modules.triggers.Triggers - Gets the trigger configurations for the task.

# wait\_until\_done (timeout=10.0)

Waits for the measurement or generation to complete.

Use this method to ensure that the specified operation is complete before you stop the task.

**Parameters timeout** (Optional[float]) — Specifies the maximum amount of time in seconds to wait for the measurement or generation to complete. This method returns an error if the time elapses. The default is 10. If you set timeout (sec) to nidaqmx.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout (sec) to 0, the method checks once and returns an error if the measurement or generation is not done.

write (data, auto\_start=<nidaqmx.task.UnsetAutoStartSentinel object>, timeout=10.0)

Writes samples to the task or virtual channels you specify.

This write method is dynamic, and is capable of accepting the samples to write in the various forms for most operations:

- •Scalar: Single sample for 1 channel.
- •List/1D numpy.ndarray: Multiple samples for 1 channel or 1 sample for multiple channels.
- •List of lists/2D numpy.ndarray: Multiple samples for multiple channels.

The data type of the samples passed in must be appropriate for the channel type of the task.

For counter output pulse operations, this write method only accepts samples in these forms:

- •Scalar CtrFreq, CtrTime, CtrTick (from nidaqmx.types): Single sample for 1 channel.
- •List of CtrFreq, CtrTime, CtrTick (from nidaqmx.types): Multiple samples for 1 channel or 1 sample for multiple channels.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

#### **Parameters**

• data (dynamic) – Contains the samples to write to the task.

The data you write must be in the units of the generation, including any custom scales. Use the DAQmx Create Channel methods to specify these units.

• **auto\_start** (Optional[bool]) - Specifies if this method automatically starts the task if you did not explicitly start it with the DAQmx Start Task method.

The default value of this parameter depends on whether you specify one sample or many samples to write to each channel. If one sample per channel was specified, the default value is True. If multiple samples per channel were specified, the default value is False.

• timeout (Optional[float]) - Specifies the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.constants.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Returns** Specifies the actual number of samples this method successfully wrote.

Return type int

# nidagmx.task.channel

Bases: object

Represents virtual channel or a list of virtual channels.

#### chan\_type

nidaqmx.constants.ChannelType - Indicates the type of the virtual channel.

#### channel\_names

*List[str]* – Specifies the unflattened list of the virtual channels.

# description

str – Specifies a user-defined description for the channel.

#### is\_global

bool – Indicates whether the channel is a global channel.

## name

*str* – Specifies the name of the virtual channel this object represents.

# physical\_channel

nidaqmx.system.physical\_channel.PhysicalChannel - Specifies the name of the physical channel upon which this virtual channel is based.

```
save (save_as=u'', author=u'', overwrite_existing_channel=False, allow_interactive_editing=True, allow_interactive_deletion=True)

Saves this local or global channel to MAX as a global channel.
```

#### **Parameters**

- **save\_as** (Optional[str]) Is the name to save the task, global channel, or custom scale as. If you do not specify a value for this input, NI-DAQmx uses the name currently assigned to the task, global channel, or custom scale.
- author (Optional[str]) Is a name to store with the task, global channel, or custom scale.
- overwrite\_existing\_channel (Optional[bool]) Specifies whether to overwrite a global channel of the same name if one is already saved in MAX. If this input is False and a global channel of the same name is already saved in MAX, this function returns an error.
- allow\_interactive\_editing (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be edited in the DAQ Assistant. If allow\_interactive\_editing is True, the DAQ Assistant must support all task or global channel settings.
- allow\_interactive\_deletion (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be deleted through MAX.

# nidaqmx.task.ai\_channel

class nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel(task\_handle, virtual\_or\_physical\_name)
Bases: nidaqmx. task modules.channels.channel

Represents one or more analog input virtual channels and their properties.

# ai\_ac\_excit\_freq

*float* – Specifies the AC excitation frequency in Hertz.

# ai\_ac\_excit\_sync\_enable

*bool* – Specifies whether to synchronize the AC excitation source of the channel to that of another channel. Synchronize the excitation sources of multiple channels to use multichannel sensors. Set this property to False for the master channel and to True for the slave channels.

#### ai\_ac\_excit\_wire\_mode

nidaqmx.constants.ACExcitWireMode – Specifies the number of leads on the LVDT or RVDT. Some sensors require you to tie leads together to create a four- or five- wire sensor. Refer to the sensor documentation for more information.

# ai\_accel\_4\_wire\_dc\_voltage\_sensitivity

float – Specifies the sensitivity of the 4 wire DC voltage acceleration sensor connected to the channel. This value is the units you specify with AI.Accel.4WireDCVoltage.SensitivityUnits. Refer to the sensor documentation to determine this value.

# ai\_accel\_4\_wire\_dc\_voltage\_sensitivity\_units

 $\label{lem:nidagmx.constants.AccelSensitivityUnits} - Specifies \quad the \quad units \quad of \\ AI.Accel.4WireDCVoltage.Sensitivity.$ 

#### ai\_accel\_charge\_sensitivity

*float* – Specifies the sensitivity of the charge acceleration sensor connected to the channel. This value is the units you specify with AI.Accel.Charge.SensitivityUnits. Refer to the sensor documentation to determine this value.

# ai accel charge sensitivity units

nidaqmx.constants.AccelChargeSensitivityUnits - Specifies the units of AI.Accel.Charge.Sensitivity.

# ai\_accel\_sensitivity

float – Specifies the sensitivity of the accelerometer. This value is in the units you specify with ai\_accel\_sensitivity\_units. Refer to the sensor documentation to determine this value.

#### ai\_accel\_sensitivity\_units

nidaqmx.constants.AccelSensitivityUnits - Specifies the units of ai\_accel\_sensitivity.

#### ai accel units

nidaqmx.constants.AccelUnits - Specifies the units to use to return acceleration measurements from the channel.

# ai\_acceld\_b\_ref

float – Specifies the decibel reference level in the units of the channel. When you read samples as a waveform, the decibel reference level is included in the waveform attributes.

#### ai\_adc\_custom\_timing\_mode

int – Specifies the timing mode of the ADC when ai\_adc\_timing\_mode is ADCTimingMode.CUSTOM.

# ai\_adc\_timing\_mode

nidaqmx.constants.ADCTimingMode – Specifies the ADC timing mode, controlling the tradeoff between speed and effective resolution. Some ADC timing modes provide increased powerline noise rejection. On devices that have an AI Convert clock, this setting affects both the maximum and default values for ai\_conv\_rate. You must use the same ADC timing mode for all channels on a device, but you can use different ADC timing modes for different devices in the same task.

#### ai\_atten

*float* – Specifies the amount of attenuation to use.

#### ai auto zero mode

*nidaqmx.constants.AutoZeroType* – Specifies how often to measure ground. NI-DAQmx subtracts the measured ground voltage from every sample.

# ai\_averaging\_win\_size

*int* – Specifies the number of samples to average while acquiring data. Increasing the number of samples to average reduces noise in your measurement.

#### ai\_bridge\_balance\_coarse\_pot

int – Specifies by how much to compensate for offset in the signal. This value can be between 0 and 127.

#### ai\_bridge\_balance\_fine\_pot

int – Specifies by how much to compensate for offset in the signal. This value can be between 0 and 4095.

# ai\_bridge\_cfg

nidaqmx.constants.BridgeConfiguration - Specifies the type of Wheatstone bridge connected to the channel.

#### ai bridge electrical units

nidaqmx.constants.BridgeElectricalUnits – Specifies from which electrical unit to scale data. Select the same unit that the sensor data sheet or calibration certificate uses for electrical values.

# ai\_bridge\_initial\_ratio

float – Specifies in volts per volt the ratio of output voltage from the bridge to excitation voltage supplied to the bridge while not under load. NI-DAQmx subtracts this value from any measurements before applying scaling equations. If you set ai\_bridge\_initial\_voltage, NI-DAQmx coerces this property to ai\_bridge\_initial\_voltage divided by ai\_excit\_actual\_val. If you set this property, NI-DAQmx coerces ai\_bridge\_initial\_voltage to the value of this property times ai\_excit\_actual\_val. If you set both this property and ai\_bridge\_initial\_voltage, and their values conflict, NI-DAQmx returns an error. To avoid this error, reset one property to its default value before setting the other.

# ai\_bridge\_initial\_voltage

float - Specifies in volts the output voltage of the bridge while not under load. NI-DAQmx subtracts

this value from any measurements before applying scaling equations. If you set ai\_bridge\_initial\_ratio, NI-DAQmx coerces this property to ai\_bridge\_initial\_ratio times ai\_excit\_actual\_val. This property is set by DAQmx Perform Bridge Offset Nulling Calibration. If you set this property, NI-DAQmx coerces ai\_bridge\_initial\_ratio to the value of this property divided by ai\_excit\_actual\_val. If you set both this property and ai\_bridge\_initial\_ratio, and their values conflict, NI-DAQmx returns an error. To avoid this error, reset one property to its default value before setting the other.

#### ai\_bridge\_nom\_resistance

float – Specifies in ohms the resistance of the bridge while not under load.

#### ai\_bridge\_physical\_units

nidaqmx.constants.BridgePhysicalUnits – Specifies to which physical unit to scale electrical data. Select the same unit that the sensor data sheet or calibration certificate uses for physical values.

# ai\_bridge\_poly\_forward\_coeff

List[float] – Specifies an list of coefficients for the polynomial that converts electrical values to physical values. Each element of the list corresponds to a term of the equation. For example, if index three of the list is 9, the fourth term of the equation is 9x<sup>3</sup>.

# ai\_bridge\_poly\_reverse\_coeff

List[float] – Specifies an list of coefficients for the polynomial that converts physical values to electrical values. Each element of the list corresponds to a term of the equation. For example, if index three of the list is 9, the fourth term of the equation is 9x<sup>3</sup>.

#### ai\_bridge\_scale\_type

nidaqmx.constants.ScaleType - Specifies the scaling type to use when scaling electrical values from the sensor to physical units.

#### ai bridge shunt cal enable

bool – Specifies whether to enable a shunt calibration switch. Use **ai\_bridge\_shunt\_cal\_select** to select the switch(es) to enable.

# ai\_bridge\_shunt\_cal\_gain\_adjust

*float* – Specifies the result of a shunt calibration. This property is set by DAQmx Perform Shunt Calibration. NI-DAQmx multiplies data read from the channel by the value of this property. This value should be close to 1.0.

# ai\_bridge\_shunt\_cal\_select

nidaqmx.constants.ShuntCalSelect - Specifies which shunt calibration switch(es) to enable. Use ai\_bridge\_shunt\_cal\_enable to enable the switch(es) you specify with this property.

# ai\_bridge\_shunt\_cal\_shunt\_cal\_a\_actual\_resistance

float – Specifies in ohms the actual value of the internal shunt calibration A resistor.

## ai\_bridge\_shunt\_cal\_shunt\_cal\_a\_resistance

float – Specifies in ohms the desired value of the internal shunt calibration A resistor.

#### ai\_bridge\_shunt\_cal\_shunt\_cal\_a\_src

nidaqmx.constants.BridgeShuntCalSource - Specifies whether to use internal or external shunt when Shunt Cal A is selected.

# ai\_bridge\_shunt\_cal\_shunt\_cal\_b\_actual\_resistance

float – Specifies in ohms the actual value of the internal shunt calibration B resistor.

#### ai\_bridge\_shunt\_cal\_shunt\_cal\_b\_resistance

float – Specifies in ohms the desired value of the internal shunt calibration B resistor.

# ai\_bridge\_table\_electrical\_vals

*List[float]* – Specifies the list of electrical values that map to the values in **ai\_bridge\_table\_physical\_vals**. Specify this value in the unit indicated by **ai\_bridge\_electrical\_units**.

# ai\_bridge\_table\_physical\_vals

*List[float]* – Specifies the list of physical values that map to the values in **ai\_bridge\_table\_electrical\_vals**. Specify this value in the unit indicated by **ai\_bridge\_physical\_units**.

# ai\_bridge\_two\_point\_lin\_first\_electrical\_val

*float* – Specifies the first electrical value, corresponding to **ai\_bridge\_two\_point\_lin\_first\_physical\_val**. Specify this value in the unit indicated by **ai\_bridge\_electrical\_units**.

# ai\_bridge\_two\_point\_lin\_first\_physical\_val

*float* – Specifies the first physical value, corresponding to **ai\_bridge\_two\_point\_lin\_first\_electrical\_val**. Specify this value in the unit indicated by **ai\_bridge\_physical\_units**.

# ai\_bridge\_two\_point\_lin\_second\_electrical\_val

float – Specifies the second electrical value, corresponding to ai\_bridge\_two\_point\_lin\_second\_physical\_val. Specify this value in the unit indicated by ai\_bridge\_electrical\_units.

# ai\_bridge\_two\_point\_lin\_second\_physical\_val

float – Specifies the second physical value, corresponding to ai\_bridge\_two\_point\_lin\_second\_electrical\_val. Specify this value in the unit indicated by ai\_bridge\_physical\_units.

# ai\_bridge\_units

nidaqmx.constants.BridgeUnits-Specifies in which unit to return voltage ratios from the channel.

#### ai\_charge\_units

nidaqmx.constants.ChargeUnits - Specifies the units to use to return charge measurements from the channel.

## ai\_coupling

nidaqmx.constants.Coupling - Specifies the coupling for the channel.

# ai\_current\_acrms\_units

nidaqmx.constants.CurrentUnits - Specifies the units to use to return current RMS measurements from the channel.

#### ai\_current\_shunt\_loc

nidaqmx.constants.CurrentShuntResistorLocation — Specifies the shunt resistor location for current measurements.

# ai\_current\_shunt\_resistance

float – Specifies in ohms the external shunt resistance for current measurements.

#### ai current units

nidaqmx.constants.CurrentUnits - Specifies the units to use to return current measurements from the channel.

#### ai\_custom\_scale

nidaqmx.system.scale.Scale - Specifies the name of a custom scale for the channel.

# ai\_data\_xfer\_custom\_threshold

int – Specifies the number of samples that must be in the FIFO to transfer data from the device if ai\_data\_xfer\_req\_cond is InputDataTransferCondition.ONBOARD\_MEMORY\_CUSTOM\_THRESHOLD.

## ai\_data\_xfer\_mech

nidaqmx. constants. DataTransferActiveTransferMode - Specifies the data transfer mode for the device.

# ai\_data\_xfer\_req\_cond

nidaqmx.constants.InputDataTransferCondition - Specifies under what condition to transfer data from the onboard memory of the device to the buffer.

#### ai dc offset

float – Specifies the DC value to add to the input range of the device. Use **ai\_rng\_high** and **ai\_rng\_low** to specify the input range. This offset is in the native units of the device.

#### ai dev scaling coeff

List[float] – Indicates the coefficients of a polynomial equation that NI-DAQmx uses to scale values from the native format of the device to volts. Each element of the list corresponds to a term of the equation. For example, if index two of the list is 4, the third term of the equation is  $4x^2$ . Scaling coefficients do not account for any custom scales or sensors contained by the channel.

#### ai\_dig\_fltr\_bandpass\_center\_freq

float – Specifies the center frequency of the passband for the digital filter.

#### ai\_dig\_fltr\_bandpass\_width

float – Specifies the width of the passband centered around the center frequency for the digital filter.

## ai\_dig\_fltr\_coeff

*List[float]* – Specifies the digital filter coefficients.

# ai\_dig\_fltr\_enable

bool – Specifies whether the digital filter is enabled or disabled.

#### ai\_dig\_fltr\_highpass\_cutoff\_freq

float – Specifies the highpass cutoff frequency of the digital filter.

# ai\_dig\_fltr\_lowpass\_cutoff\_freq

float – Specifies the lowpass cutoff frequency of the digital filter.

# ai\_dig\_fltr\_notch\_center\_freq

float – Specifies the center frequency of the stopband for the digital filter.

# ai\_dig\_fltr\_notch\_width

float – Specifies the width of the stopband centered around the center frequency for the digital filter.

#### ai\_dig\_fltr\_order

int – Specifies the order of the digital filter.

# ai\_dig\_fltr\_response

nidagmx.constants.FilterResponse - Specifies the digital filter response.

# ai\_dig\_fltr\_type

nidaqmx.constants.FilterType - Specifies the digital filter type.

#### ai dither enable

*bool* – Specifies whether to enable dithering. Dithering adds Gaussian noise to the input signal. You can use dithering to achieve higher resolution measurements by over sampling the input signal and averaging the results.

# ai\_eddy\_current\_prox\_sensitivity

float – Specifies the sensitivity of the eddy current proximity probe. This value is in the units you specify with ai\_eddy\_current\_prox\_sensitivity\_units. Refer to the sensor documentation to determine this value.

# ai\_eddy\_current\_prox\_sensitivity\_units

nidaqmx.constants.EddyCurrentProxProbeSensitivityUnits - Specifies the units of ai\_eddy\_current\_prox\_sensitivity.

# ai\_eddy\_current\_prox\_units

nidaqmx.constants.LengthUnits - Specifies the units to use to return proximity measurements from the channel.

# ai\_enhanced\_alias\_rejection\_enable

bool – Specifies whether to enable enhanced alias rejection. Leave this property set to the default value for most applications.

#### ai excit actual val

float – Specifies the actual amount of excitation supplied by an internal excitation source. If you read an internal excitation source more precisely with an external device, set this property to the value you read. NI-DAQmx ignores this value for external excitation. When performing shunt calibration, some devices set this property automatically.

#### ai\_excit\_d\_cor\_ac

nidaqmx.constants.ExcitationDCorAC - Specifies if the excitation supply is DC or AC.

#### ai\_excit\_idle\_output\_behavior

nidaqmx.constants.ExcitationIdleOutputBehavior-Specifies whether this channel will disable excitation after the task is uncommitted. Setting this to Zero Volts or Amps disables excitation after task uncommit. Setting this attribute to Maintain Existing Value leaves the excitation on after task uncommit.

#### ai\_excit\_sense

nidaqmx.constants.Sense - Specifies whether to use local or remote sense to sense excitation.

#### ai excit src

nidagmx.constants.ExcitationSource - Specifies the source of excitation.

# ai\_excit\_use\_for\_scaling

bool – Specifies if NI-DAQmx divides the measurement by the excitation. You should typically set this property to True for ratiometric transducers. If you set this property to True, set **ai\_max** and **ai\_min** to reflect the scaling.

# ai\_excit\_use\_multiplexed

*bool* – Specifies if the SCXI-1122 multiplexes the excitation to the upper half of the channels as it advances through the scan list.

#### ai\_excit\_val

float – Specifies the amount of excitation that the sensor requires. If ai\_excit\_voltage\_or\_current is ExcitationVoltageOrCurrent.USE\_VOLTAGE, this value is in volts. If ai\_excit\_voltage\_or\_current is ExcitationVoltageOrCurrent.USE\_CURRENT, this value is in amperes.

# ai\_excit\_voltage\_or\_current

nidaqmx.constants.ExcitationVoltageOrCurrent - Specifies if the channel uses current or voltage excitation.

# ai\_filter\_delay

float – Indicates the amount of time between when the ADC samples data and when the sample is read by the host device. This value is in the units you specify with ai\_filter\_delay\_units. You can adjust this amount of time using ai\_filter\_delay\_adjustment.

#### ai\_filter\_delay\_adjustment

float – Specifies the amount of filter delay that gets removed if ai\_remove\_filter\_delay is enabled. This delay adjustment is in addition to the value indicated by ai\_filter\_delay. This delay adjustment is in the units you specify with ai\_filter\_delay\_units.

# ai\_filter\_delay\_units

 $\label{lem:constants_def} \textit{nidaqmx.constants.DigitalWidthUnits} - Specifies \ \ \textit{the units of ai\_filter\_delay} \ \ \textit{and ai\_filter\_delay\_adjustment}.$ 

# ai\_force\_iepe\_sensor\_sensitivity

*float* – Specifies the sensitivity of the IEPE force sensor connected to the channel. Specify this value in the unit indicated by **ai force iepe sensor sensitivity units**.

# ai\_force\_iepe\_sensor\_sensitivity\_units

nidaqmx.constants.ForceIEPESensorSensitivityUnits - Specifies the units for ai\_force\_iepe\_sensor\_sensitivity.

#### ai force read from chan

*bool* – Specifies whether to read from the channel if it is a cold-junction compensation channel. By default, DAQmx Read does not return data from cold-junction compensation channels. Setting this property to True forces read operations to return the cold-junction compensation channel data with the other channels in the task.

# ai\_force\_units

nidaqmx.constants.ForceUnits - Specifies in which unit to return force or load measurements from the channel.

# ai\_freq\_hyst

float – Specifies in volts a window below **ai\_freq\_thresh\_voltage**. The input voltage must pass below **ai\_freq\_thresh\_voltage** minus this value before NI- DAQmx recognizes a waveform repetition at **ai\_freq\_thresh\_voltage**. Hysteresis can improve the measurement accuracy when the signal contains noise or jitter.

# ai\_freq\_thresh\_voltage

*float* – Specifies the voltage level at which to recognize waveform repetitions. You should select a voltage level that occurs only once within the entire period of a waveform. You also can select a voltage that occurs only once while the voltage rises or falls.

## ai\_freq\_units

nidaqmx.constants.FrequencyUnits - Specifies the units to use to return frequency measurements from the channel.

#### ai gain

float – Specifies a gain factor to apply to the channel.

#### ai impedance

nidaqmx.constants.Impedance1 - Specifies the input impedance of the channel.

## ai\_input\_src

str – Specifies the source of the channel. You can use the signal from the I/O connector or one of several calibration signals. Certain devices have a single calibration signal bus. For these devices, you must specify the same calibration signal for all channels you connect to a calibration signal.

## ai\_lead\_wire\_resistance

float – Specifies in ohms the resistance of the wires that lead to the sensor.

#### ai\_lossy\_lsb\_removal\_compressed\_samp\_size

*int* – Specifies the number of bits to return in a raw sample when **ai\_raw\_data\_compression\_type** is set to **RawDataCompressionType.LOSSY\_LSB\_REMOVAL**.

# ai\_lowpass\_cutoff\_freq

float – Specifies the frequency in Hertz that corresponds to the -3dB cutoff of the filter.

#### ai lowpass enable

bool – Specifies whether to enable the lowpass filter of the channel.

# ai\_lowpass\_switch\_cap\_clk\_src

nidaqmx.constants.SourceSelection – Specifies the source of the filter clock. If you need a higher resolution for the filter, you can supply an external clock to increase the resolution. Refer to the SCXI-1141/1142/1143 User Manual for more information.

## ai\_lowpass\_switch\_cap\_ext\_clk\_div

int – Specifies the divisor for the external clock when you set ai\_lowpass\_switch\_cap\_clk\_src to Source-Selection.EXTERNAL. On the SCXI-1141, SCXI-1142, and SCXI-1143, NI-DAQmx determines the filter cutoff by using the equation f/(100\*n), where f is the external frequency, and n is the external clock divisor. Refer to the SCXI-1141/1142/1143 User Manual for more information.

## ai\_lowpass\_switch\_cap\_ext\_clk\_freq

float – Specifies the frequency of the external clock when you set ai\_lowpass\_switch\_cap\_clk\_src to SourceSelection.EXTERNAL. NI-DAQmx uses this frequency to set the pre- and post- filters on the SCXI-1141, SCXI-1142, and SCXI-1143. On those devices, NI-DAQmx determines the filter cutoff by using the equation f/(100\*n), where f is the external frequency, and n is the external clock divisor. Refer to the SCXI-1141/1142/1143 User Manual for more information.

# ai\_lowpass\_switch\_cap\_out\_clk\_div

*int* – Specifies the divisor for the output clock. NI-DAQmx uses the cutoff frequency to determine the output clock frequency. Refer to the SCXI-1141/1142/1143 User Manual for more information.

# ai\_lvdt\_sensitivity

float – Specifies the sensitivity of the LVDT. This value is in the units you specify with ai lvdt sensitivity units. Refer to the sensor documentation to determine this value.

## ai\_lvdt\_sensitivity\_units

nidagmx.constants.LVDTSensitivityUnits - Specifies the units of ai\_lvdt\_sensitivity.

# ai\_lvdt\_units

nidaqmx.constants.LengthUnits - Specifies the units to use to return linear position measurements from the channel.

#### ai max

float – Specifies the maximum value you expect to measure. This value is in the units you specify with a units property. When you query this property, it returns the coerced maximum value that the device can measure with the current settings.

# ai\_meas\_type

nidaqmx.constants.UsageTypeAI – Indicates the measurement to take with the analog input channel and in some cases, such as for temperature measurements, the sensor to use.

# ai\_mem\_map\_enable

bool – Specifies for NI-DAQmx to map hardware registers to the memory space of the application, if possible. Normally, NI- DAQmx maps hardware registers to memory accessible only to the kernel. Mapping the registers to the memory space of the application increases performance. However, if the application accesses the memory space mapped to the registers, it can adversely affect the operation of the device and possibly result in a system crash.

#### ai microphone sensitivity

*float* – Specifies the sensitivity of the microphone. This value is in mV/Pa. Refer to the sensor documentation to determine this value.

# ai\_min

*float* – Specifies the minimum value you expect to measure. This value is in the units you specify with a units property. When you query this property, it returns the coerced minimum value that the device can measure with the current settings.

#### ai open chan detect enable

bool – Specifies whether to enable open channel detection.

# ai\_open\_thrmcpl\_detect\_enable

bool – Specifies whether to apply the open thermocouple detection bias voltage to the channel. Changing the value of this property on a channel may require settling time before the data returned is valid. To compensate for this settling time, discard unsettled data or add a delay between committing and starting

the task. Refer to your device specifications for the required settling time. When open thermocouple detection is enabled, use **open\_thrmcpl\_chans\_exist** to determine if any channels were open.

# ai\_pressure\_units

nidaqmx.constants.PressureUnits - Specifies in which unit to return pressure measurements from the channel.

#### ai probe atten

float – Specifies the amount of attenuation provided by the probe connected to the channel. Specify this attenuation as a ratio.

#### ai\_raw\_data\_compression\_type

nidaqmx.constants.RawDataCompressionType - Specifies the type of compression to apply to raw samples returned from the device.

#### ai\_raw\_samp\_justification

nidaqmx.constants.DataJustification – Indicates the justification of a raw sample from the device.

#### ai\_raw\_samp\_size

int – Indicates in bits the size of a raw sample from the device.

#### ai\_remove\_filter\_delay

bool - Specifies if filter delay removal is enabled on the device.

#### ai\_resistance\_cfg

nidaqmx.constants.ResistanceConfiguration – Specifies the resistance configuration for the channel. NI-DAQmx uses this value for any resistance-based measurements, including temperature measurement using a thermistor or RTD.

## ai\_resistance\_units

nidaqmx. constants. ResistanceUnits — Specifies the units to use to return resistance measurements.

# ai resolution

*float* – Indicates the resolution of the analog-to-digital converter of the channel. This value is in the units you specify with **ai\_resolution\_units**.

#### ai\_resolution\_units

nidagmx.constants.ResolutionType - Indicates the units of ai\_resolution.

#### ai\_rng\_high

float – Specifies the upper limit of the input range of the device. This value is in the native units of the device. On E Series devices, for example, the native units is volts.

## ai\_rng\_low

*float* – Specifies the lower limit of the input range of the device. This value is in the native units of the device. On E Series devices, for example, the native units is volts.

# ai\_rosette\_strain\_gage\_gage\_orientation

*float* – Specifies gage orientation in degrees with respect to the X axis.

# ai\_rosette\_strain\_gage\_rosette\_meas\_type

nidaqmx.constants.StrainGageRosetteMeasurementType-Specifies the type of rosette measurement.

#### ai\_rosette\_strain\_gage\_rosette\_type

nidaqmx.constants.StrainGageRosetteType - Indicates the type of rosette gage.

# ai\_rosette\_strain\_gage\_strain\_chans

*List[str]* – Indicates the raw strain channels that comprise the strain rosette.

#### ai rtd a

*float* – Specifies the 'A' constant of the Callendar-Van Dusen equation. NI-DAQmx requires this value when you use a custom RTD.

#### ai rtd b

float – Specifies the 'B' constant of the Callendar-Van Dusen equation. NI-DAQmx requires this value when you use a custom RTD.

#### ai rtd c

float – Specifies the 'C' constant of the Callendar-Van Dusen equation. NI-DAQmx requires this value when you use a custom RTD.

# ai\_rtd\_r\_0

float – Specifies in ohms the sensor resistance at 0 deg C. The Callendar-Van Dusen equation requires this value. Refer to the sensor documentation to determine this value.

#### ai\_rtd\_type

nidagmx.constants.RTDType - Specifies the type of RTD connected to the channel.

# ai\_rvdt\_sensitivity

float – Specifies the sensitivity of the RVDT. This value is in the units you specify with ai\_rvdt\_sensitivity\_units. Refer to the sensor documentation to determine this value.

# ai\_rvdt\_sensitivity\_units

nidagmx.constants.RVDTSensitivityUnits - Specifies the units of ai\_rvdt\_sensitivity.

#### ai\_rvdt\_units

nidaqmx.constants.AngleUnits - Specifies the units to use to return angular position measurements from the channel.

# ai\_samp\_and\_hold\_enable

bool – Specifies whether to enable the sample and hold circuitry of the device. When you disable sample and hold circuitry, a small voltage offset might be introduced into the signal. You can eliminate this offset by using **ai\_auto\_zero\_mode** to perform an auto zero on the channel.

# ai\_sound\_pressure\_max\_sound\_pressure\_lvl

float – Specifies the maximum instantaneous sound pressure level you expect to measure. This value is in decibels, referenced to 20 micropascals. NI-DAQmx uses the maximum sound pressure level to calculate values in pascals for **ai\_max** and **ai\_min** for the channel.

# ai\_sound\_pressure\_units

nidaqmx.constants.SoundPressureUnits - Specifies the units to use to return sound pressure measurements from the channel.

# ai\_sound\_pressured\_b\_ref

float – Specifies the decibel reference level in the units of the channel. When you read samples as a waveform, the decibel reference level is included in the waveform attributes. NI- DAQmx also uses the decibel reference level when converting ai\_sound\_pressure\_max\_sound\_pressure\_lvl to a voltage level.

# ai\_strain\_force\_read\_from\_chan

bool – Specifies whether the data is returned by DAQmx Read when set on a raw strain channel that is part of a rosette configuration.

#### ai\_strain\_gage\_cfg

nidaqmx.constants.StrainGageBridgeType - Specifies the bridge configuration of the strain gages.

# ai\_strain\_gage\_gage\_factor

*float* – Specifies the sensitivity of the strain gage. Gage factor relates the change in electrical resistance to the change in strain. Refer to the sensor documentation for this value.

# ai\_strain\_gage\_poisson\_ratio

float – Specifies the ratio of lateral strain to axial strain in the material you are measuring.

#### ai strain units

nidaqmx.constants.StrainUnits - Specifies the units to use to return strain measurements from the channel.

#### ai teds is teds

bool – Indicates if the virtual channel was initialized using a TEDS bitstream from the corresponding physical channel.

#### ai\_teds\_units

str – Indicates the units defined by TEDS information associated with the channel.

## ai\_temp\_units

nidaqmx.constants.TemperatureUnits - Specifies the units to use to return temperature measurements from the channel.

## ai\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the terminal configuration for the channel.

## ai\_thrmcpl\_cjc\_chan

nidaqmx.\_task\_modules.channels.channel.Channel-Indicates the channel that acquires the temperature of the cold junction if ai\_thrmcpl\_cjc\_src is CJCSource1.SCANNABLE\_CHANNEL. If the channel is a temperature channel, NI-DAQmx acquires the temperature in the correct units. Other channel types, such as a resistance channel with a custom sensor, must use a custom scale to scale values to degrees Celsius.

#### ai\_thrmcpl\_cjc\_src

nidagmx.constants.CJCSource - Indicates the source of cold-junction compensation.

## ai\_thrmcpl\_cjc\_val

*float* – Specifies the temperature of the cold junction if **ai\_thrmcpl\_cjc\_src** is **CJC-Source1.CONSTANT\_USER\_VALUE**. Specify this value in the units of the measurement.

#### ai\_thrmcpl\_lead\_offset\_voltage

*float* – Specifies the lead offset nulling voltage to subtract from measurements on a device. This property is ignored if open thermocouple detection is disabled.

# ai\_thrmcpl\_scale\_type

nidaqmx.constants.ScaleType - Specifies the method or equation form that the thermocouple scale uses.

# ai thrmcpl\_type

nidaqmx.constants.ThermocoupleType - Specifies the type of thermocouple connected to the channel. Thermocouple types differ in composition and measurement range.

#### ai\_thrmstr\_a

float – Specifies the 'A' constant of the Steinhart-Hart thermistor equation.

# ai\_thrmstr\_b

float – Specifies the 'B' constant of the Steinhart-Hart thermistor equation.

#### ai thrmstr c

float – Specifies the 'C' constant of the Steinhart-Hart thermistor equation.

# ai\_thrmstr\_r\_1

*float* – Specifies in ohms the value of the reference resistor for the thermistor if you use voltage excitation. NI-DAQmx ignores this value for current excitation.

#### ai torque units

nidaqmx.constants.TorqueUnits - Specifies in which unit to return torque measurements from the channel.

#### ai\_usb\_xfer\_req\_count

*int* – Specifies the maximum number of simultaneous USB transfers used to stream data. Modify this value to affect performance under different combinations of operating system and device.

#### ai\_usb\_xfer\_req\_size

*int* – Specifies the maximum size of a USB transfer request in bytes. Modify this value to affect performance under different combinations of operating system and device.

#### ai\_velocity\_iepe\_sensor\_sensitivity

*float* – Specifies the sensitivity of the IEPE velocity sensor connected to the channel. Specify this value in the unit indicated by **ai\_velocity\_iepe\_sensor\_sensitivity\_units**.

# ai\_velocity\_iepe\_sensor\_sensitivity\_units

nidaqmx.constants.VelocityIEPESensorSensitivityUnits - Specifies the units for ai\_velocity\_iepe\_sensor\_sensitivity.

#### ai velocity iepe sensord b ref

float – Specifies the decibel reference level in the units of the channel. When you read samples as a waveform, the decibel reference level is included in the waveform attributes.

#### ai velocity units

nidaqmx.constants.VelocityUnits - Specifies in which unit to return velocity measurements from the channel.

#### ai voltage acrms units

nidaqmx.constants.VoltageUnits - Specifies the units to use to return voltage RMS measurements from the channel.

# ai\_voltage\_units

nidaqmx.constants.VoltageUnits - Specifies the units to use to return voltage measurements from the channel.

# ai\_voltaged\_b\_ref

*float* – Specifies the decibel reference level in the units of the channel. When you read samples as a waveform, the decibel reference level is included in the waveform attributes.

# chan\_type

nidaqmx.constants.ChannelType - Indicates the type of the virtual channel.

## channel names

*List[str]* – Specifies the unflattened list of the virtual channels.

#### description

str – Specifies a user-defined description for the channel.

# is\_global

bool – Indicates whether the channel is a global channel.

# name

str – Specifies the name of the virtual channel this object represents.

#### physical\_channel

nidaqmx.system.physical\_channel.PhysicalChannel - Specifies the name of the physical channel upon which this virtual channel is based.

**save** (save\_as=u'', author=u'', overwrite\_existing\_channel=False, allow\_interactive\_editing=True, allow\_interactive\_deletion=True)

Saves this local or global channel to MAX as a global channel.

#### **Parameters**

- **save\_as** (Optional[str]) Is the name to save the task, global channel, or custom scale as. If you do not specify a value for this input, NI-DAQmx uses the name currently assigned to the task, global channel, or custom scale.
- author (Optional[str]) Is a name to store with the task, global channel, or custom scale.
- **overwrite\_existing\_channel** (Optional[bool]) Specifies whether to overwrite a global channel of the same name if one is already saved in MAX. If this input is False and a global channel of the same name is already saved in MAX, this function returns an error.
- allow\_interactive\_editing (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be edited in the DAQ Assistant. If allow\_interactive\_editing is True, the DAQ Assistant must support all task or global channel settings.
- allow\_interactive\_deletion (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be deleted through MAX.

# nidaqmx.task.ao\_channel

class nidaqmx.\_task\_modules.channels.ao\_channel.AOChannel(task\_handle, virtual\_or\_physical\_name)
Bases: nidaqmx. task modules.channels.channel

Represents one or more analog output virtual channels and their properties.

# ao\_current\_units

nidaqmx.constants.CurrentUnits - Specifies in what units to generate current on the channel. Write data to the channel in the units you select.

#### ao custom scale

nidaqmx.system.scale.Scale - Specifies the name of a custom scale for the channel.

# ao\_dac\_offset\_ext\_src

str – Specifies the source of the DAC offset voltage if **ao\_dac\_offset\_src** is **SourceSelection.EXTERNAL**. The valid sources for this signal vary by device.

# ao\_dac\_offset\_src

nidaqmx.constants.SourceSelection - Specifies the source of the DAC offset voltage. The value of this voltage source determines the full-scale value of the DAC.

#### ao dac offset val

*float* – Specifies in volts the value of the DAC offset voltage. To achieve best accuracy, the DAC offset value should be hand calibrated.

# ao\_dac\_ref\_allow\_conn\_to\_gnd

bool – Specifies whether to allow grounding the internal DAC reference at run time. You must set this property to True and set **ao\_dac\_ref\_src** to **SourceSelection.INTERNAL** before you can set **ao\_dac\_ref\_conn\_to\_gnd** to True.

# ao\_dac\_ref\_conn\_to\_gnd

bool – Specifies whether to ground the internal DAC reference. Grounding the internal DAC reference has the effect of grounding all analog output channels and stopping waveform generation across all analog output channels regardless of whether the channels belong to the current task. You can ground the internal DAC reference only when ao\_dac\_ref\_src is SourceSelection.INTERNAL and ao\_dac\_ref\_allow\_conn\_to\_gnd is True.

#### ao dac ref ext src

str – Specifies the source of the DAC reference voltage if **ao\_dac\_ref\_src** is **SourceSelection.EXTERNAL**. The valid sources for this signal vary by device.

## ao\_dac\_ref\_src

nidaqmx.constants.SourceSelection - Specifies the source of the DAC reference voltage. The value of this voltage source determines the full-scale value of the DAC.

#### ao dac ref val

*float* – Specifies in volts the value of the DAC reference voltage. This voltage determines the full-scale range of the DAC. Smaller reference voltages result in smaller ranges, but increased resolution.

#### ao\_dac\_rng\_high

*float* – Specifies the upper limit of the output range of the device. This value is in the native units of the device. On E Series devices, for example, the native units is volts.

# ao\_dac\_rng\_low

float – Specifies the lower limit of the output range of the device. This value is in the native units of the device. On E Series devices, for example, the native units is volts.

#### ao data xfer mech

nidaqmx.constants.DataTransferActiveTransferMode-Specifies the data transfer mode for the device.

#### ao data xfer req cond

nidaqmx.constants.OutputDataTransferCondition - Specifies under what condition to transfer data from the buffer to the onboard memory of the device.

#### ao dev scaling coeff

List[float] – Indicates the coefficients of a linear equation that NI-DAQmx uses to scale values from a voltage to the native format of the device. Each element of the list corresponds to a term of the equation. The first element of the list corresponds to the y-intercept, and the second element corresponds to the slope. Scaling coefficients do not account for any custom scales that may be applied to the channel.

# ao\_enhanced\_image\_rejection\_enable

*bool* – Specifies whether to enable the DAC interpolation filter. Disable the interpolation filter to improve DAC signal-to- noise ratio at the expense of degraded image rejection.

#### ao\_filter\_delay

*float* – Specifies the amount of time between when the sample is written by the host device and when the sample is output by the DAC. This value is in the units you specify with **ao\_filter\_delay\_units**.

# ao\_filter\_delay\_adjustment

float – Specifies an additional amount of time to wait between when the sample is written by the host device and when the sample is output by the DAC. This delay adjustment is in addition to the value indicated by **ao filter delay**. This delay adjustment is in the units you specify with **ao filter delay units**.

#### ao\_filter\_delay\_units

 ${\it nidaqmx.constants.DigitalWidthUnits-Specifies~the~units~of~ao\_filter\_delay~and~ao\_filter\_delay\_adjustment.}$ 

# ao\_func\_gen\_amplitude

float – Specifies the zero-to-peak amplitude of the waveform to generate in volts. Zero and negative values are valid.

# ao\_func\_gen\_fm\_deviation

*float* – Specifies the FM deviation in hertz per volt when **ao\_func\_gen\_modulation\_type** is **Modulation-Type.FM**.

# ao\_func\_gen\_freq

float – Specifies the frequency of the waveform to generate in hertz.

#### ao\_func\_gen\_modulation\_type

nidaqmx.constants.ModulationType – Specifies if the device generates a modulated version of the waveform using the original waveform as a carrier and input from an external terminal as the signal.

#### ao\_func\_gen\_offset

*float* – Specifies the voltage offset of the waveform to generate.

#### ao func gen square duty cycle

*float* – Specifies the square wave duty cycle of the waveform to generate.

# ao\_func\_gen\_type

nidagmx.constants.FuncGenType - Specifies the kind of the waveform to generate.

#### ao gain

float – Specifies in decibels the gain factor to apply to the channel.

## ao\_idle\_output\_behavior

nidaqmx.constants.A0IdleOutputBehavior - Specifies the state of the channel when no generation is in progress.

# ao\_load\_impedance

float – Specifies in ohms the load impedance connected to the analog output channel.

#### ao max

float – Specifies the maximum value you expect to generate. The value is in the units you specify with a units property. If you try to write a value larger than the maximum value, NI- DAQmx generates an error. NI-DAQmx might coerce this value to a smaller value if other task settings restrict the device from generating the desired maximum.

# ao\_mem\_map\_enable

bool – Specifies for NI-DAQmx to map hardware registers to the memory space of the application, if possible. Normally, NI- DAQmx maps hardware registers to memory accessible only to the kernel. Mapping the registers to the memory space of the application increases performance. However, if the application accesses the memory space mapped to the registers, it can adversely affect the operation of the device and possibly result in a system crash.

#### ao\_min

float – Specifies the minimum value you expect to generate. The value is in the units you specify with a units property. If you try to write a value smaller than the minimum value, NI- DAQmx generates an error. NI-DAQmx might coerce this value to a larger value if other task settings restrict the device from generating the desired minimum.

# ao\_output\_impedance

float – Specifies in ohms the impedance of the analog output stage of the device.

# ao\_output\_type

nidaqmx.constants.UsageTypeAO - Indicates whether the channel generates voltage, current, or a waveform.

# ao\_reglitch\_enable

bool – Specifies whether to enable reglitching. The output of a DAC normally glitches whenever the DAC is updated with a new value. The amount of glitching differs from code to code and is generally largest at major code transitions. Reglitching generates uniform glitch energy at each code transition and provides for more uniform glitches. Uniform glitch energy makes it easier to filter out the noise introduced from glitching during spectrum analysis.

# ao\_resolution

*float* – Indicates the resolution of the digital-to-analog converter of the channel. This value is in the units you specify with **ao\_resolution\_units**.

#### ao resolution units

nidagmx.constants.ResolutionType - Specifies the units of ao\_resolution.

## ao\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the terminal configuration of the channel.

#### ao usb xfer req count

*int* – Specifies the maximum number of simultaneous USB transfers used to stream data. Modify this value to affect performance under different combinations of operating system and device.

## ao\_usb\_xfer\_req\_size

*int* – Specifies the maximum size of a USB transfer request in bytes. Modify this value to affect performance under different combinations of operating system and device.

#### ao\_use\_only\_on\_brd\_mem

*bool* – Specifies whether to write samples directly to the onboard memory of the device, bypassing the memory buffer. Generally, you cannot update onboard memory directly after you start the task. Onboard memory includes data FIFOs.

#### ao voltage current limit

*float* – Specifies the current limit, in amperes, for the voltage channel.

## ao\_voltage\_units

nidaqmx.constants.VoltageUnits – Specifies in what units to generate voltage on the channel. Write data to the channel in the units you select.

#### chan\_type

nidagmx.constants.ChannelType - Indicates the type of the virtual channel.

# channel\_names

*List[str]* – Specifies the unflattened list of the virtual channels.

# description

str – Specifies a user-defined description for the channel.

## is\_global

bool – Indicates whether the channel is a global channel.

#### name

*str* – Specifies the name of the virtual channel this object represents.

# physical channel

nidaqmx.system.physical\_channel.PhysicalChannel - Specifies the name of the physical channel upon which this virtual channel is based.

**save** (save\_as=u'', author=u'', overwrite\_existing\_channel=False, allow\_interactive\_editing=True, allow\_interactive\_deletion=True)

Saves this local or global channel to MAX as a global channel.

## **Parameters**

- **save\_as** (Optional[str]) Is the name to save the task, global channel, or custom scale as. If you do not specify a value for this input, NI-DAQmx uses the name currently assigned to the task, global channel, or custom scale.
- author (Optional[str]) Is a name to store with the task, global channel, or custom scale.
- overwrite\_existing\_channel (Optional[bool]) Specifies whether to overwrite a global channel of the same name if one is already saved in MAX. If this input is False and a global channel of the same name is already saved in MAX, this function returns an error.

- allow\_interactive\_editing (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be edited in the DAQ Assistant. If allow\_interactive\_editing is True, the DAQ Assistant must support all task or global channel settings.
- allow\_interactive\_deletion (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be deleted through MAX.

# nidagmx.task.ci channel

Represents one or more counter input virtual channels and their properties.

#### chan\_type

nidaqmx.constants.ChannelType - Indicates the type of the virtual channel.

#### channel names

*List[str]* – Specifies the unflattened list of the virtual channels.

# ci\_ang\_encoder\_initial\_angle

float – Specifies the starting angle of the encoder. This value is in the units you specify with ci\_ang\_encoder\_units.

# ci\_ang\_encoder\_pulses\_per\_rev

*int* – Specifies the number of pulses the encoder generates per revolution. This value is the number of pulses on either signal A or signal B, not the total number of pulses on both signal A and signal B.

# ci\_ang\_encoder\_units

nidaqmx.constants.AngleUnits - Specifies the units to use to return angular position measurements from the channel.

#### ci count

int – Indicates the current value of the count register.

#### ci\_count\_edges\_active\_edge

nidagmx.constants.Edge - Specifies on which edges to increment or decrement the counter.

# ci\_count\_edges\_count\_dir\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

# ci\_count\_edges\_count\_dir\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the filter recognizes.

# ci\_count\_edges\_count\_dir\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# ci\_count\_edges\_count\_dir\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

#### ci\_count\_edges\_count\_dir\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

# ci\_count\_edges\_count\_dir\_logic\_lvl\_behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior on the count reset line.

#### ci\_count\_edges\_count\_dir\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

# ci\_count\_edges\_count\_reset\_active\_edge

nidaqmx.constants.Edge - Specifies on which edge of the signal to reset the count.

# ci\_count\_edges\_count\_reset\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

#### ci\_count\_edges\_count\_reset\_dig\_fltr\_min\_pulse\_width

float – Specifies the minimum pulse width the filter recognizes.

# ci\_count\_edges\_count\_reset\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# ci\_count\_edges\_count\_reset\_dig\_fltr\_timebase\_src

str – Specifies the input of the signal to use as the timebase of the pulse width filter.

# ci\_count\_edges\_count\_reset\_dig\_sync\_enable

bool – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

# ci\_count\_edges\_count\_reset\_enable

bool – Specifies whether to reset the count on the active edge specified with ci\_count\_edges\_count\_reset\_term.

#### ci\_count\_edges\_count\_reset\_logic\_lvl\_behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior on the count reset line.

## ci\_count\_edges\_count\_reset\_reset\_cnt

int – Specifies the value to reset the count to.

# ci\_count\_edges\_count\_reset\_term

str – Specifies the input terminal of the signal to reset the count.

# ci\_count\_edges\_count\_reset\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

#### ci\_count\_edges\_dig\_fltr\_enable

*bool* – Specifies whether to apply the pulse width filter to the signal.

# ci\_count\_edges\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the filter recognizes.

## ci\_count\_edges\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# ci\_count\_edges\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

# ci\_count\_edges\_dig\_sync\_enable

bool – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

#### ci\_count\_edges\_dir

nidaqmx.constants.CountDirection - Specifies whether to increment or decrement the counter on each edge.

#### ci\_count\_edges\_dir\_term

str – Specifies the source terminal of the digital signal that controls the count direction if ci count edges dir is CountDirection1.EXTERNAL SOURCE.

## ci\_count\_edges\_gate\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the gate input signal.

# ci\_count\_edges\_gate\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the digital filter recognizes.

# ci\_count\_edges\_gate\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# ci\_count\_edges\_gate\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

# ci\_count\_edges\_gate\_enable

bool - Specifies whether to enable the functionality to gate the counter input signal for a count edges measurement.

#### ci\_count\_edges\_gate\_logic\_lvl\_behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior on the gate input line.

# ci\_count\_edges\_gate\_term

str – Specifies the gate terminal.

## ci\_count\_edges\_gate\_term\_cfg

nidagmx.constants.TerminalConfiguration - Specifies the gate terminal configuration.

## ci\_count\_edges\_gate\_when

nidaqmx.constants.Level - Specifies whether the counter gates input pulses while the signal is high or low.

# ci\_count\_edges\_initial\_cnt

int – Specifies the starting value from which to count.

# ci\_count\_edges\_logic\_lvl\_behavior

nidaqmx.constants.LoqicLvlBehavior - Specifies the logic level behavior on the input line.

# ci\_count\_edges\_term

str – Specifies the input terminal of the signal to measure.

# ci\_count\_edges\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

#### ci\_ctr\_timebase\_active\_edge

nidaqmx.constants.Edge - Specifies whether a timebase cycle is from rising edge to rising edge or from falling edge to falling edge.

# ci\_ctr\_timebase\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

#### ci\_ctr\_timebase\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the filter recognizes.

# ci\_ctr\_timebase\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# $\verb"ci_ctr_timebase_dig_fltr_timebase\_src"$

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

#### ci ctr timebase dig sync enable

bool – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

# ci\_ctr\_timebase\_master\_timebase\_div

*int* – Specifies the divisor for an external counter timebase. You can divide the counter timebase in order to measure slower signals without causing the count register to roll over.

#### ci\_ctr\_timebase\_rate

*float* – Specifies in Hertz the frequency of the counter timebase. Specifying the rate of a counter timebase allows you to take measurements in terms of time or frequency rather than in ticks of the timebase. If you use an external timebase and do not specify the rate, you can take measurements only in terms of ticks of the timebase.

#### ci\_ctr\_timebase\_src

str – Specifies the terminal of the timebase to use for the counter.

#### ci\_custom\_scale

nidagmx.system.scale.Scale - Specifies the name of a custom scale for the channel.

#### ci data xfer mech

nidaqmx. constants. DataTransferActiveTransferMode – Specifies the data transfer mode for the channel.

#### ci data xfer req cond

nidaqmx.constants.InputDataTransferCondition - Specifies under what condition to transfer data from the onboard memory of the device to the buffer.

#### ci dup count prevention

*bool* – Specifies whether to enable duplicate count prevention for the channel. Duplicate count prevention is enabled by default. Setting **ci\_prescaler** disables duplicate count prevention unless you explicitly enable it.

# ci\_duty\_cycle\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

# ci\_duty\_cycle\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the digital filter recognizes.

#### ci\_duty\_cycle\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# ci\_duty\_cycle\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

#### ci\_duty\_cycle\_logic\_lvl\_behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior on the input line.

# ci\_duty\_cycle\_starting\_edge

nidaqmx.constants.Edge - Specifies which edge of the input signal to begin the duty cycle measurement.

#### ci\_duty\_cycle\_term

str – Specifies the input terminal of the signal to measure.

# ci\_duty\_cycle\_term\_cfg

 ${\it nidaqmx.constants.Terminal Configuration-Specifies~the~input~terminal~configuration.}$ 

# ci\_encoder\_a\_input\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

#### ci encoder a input dig fltr min pulse width

float – Specifies in seconds the minimum pulse width the filter recognizes.

# ci\_encoder\_a\_input\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

#### ci encoder a input dig fltr timebase src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

# ci\_encoder\_a\_input\_dig\_sync\_enable

bool – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

## ci\_encoder\_a\_input\_logic\_lvl\_behavior

nidaqmx.constants.LoqicLvlBehavior - Specifies the logic level behavior on the input line.

# ci\_encoder\_a\_input\_term

str – Specifies the terminal to which signal A is connected.

# ci\_encoder\_a\_input\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

# ci\_encoder\_b\_input\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

#### ci\_encoder\_b\_input\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the filter recognizes.

# ci\_encoder\_b\_input\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

## ci\_encoder\_b\_input\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

# ci\_encoder\_b\_input\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

#### ci\_encoder\_b\_input\_logic\_lvl\_behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior on the input line.

# ci\_encoder\_b\_input\_term

str – Specifies the terminal to which signal B is connected.

## ci\_encoder\_b\_input\_term\_cfg

nidagmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

#### ci\_encoder\_decoding\_type

nidaqmx.constants.EncoderType – Specifies how to count and interpret the pulses the encoder generates on signal A and signal B. EncoderType2.X\_1, EncoderType2.X\_2, and EncoderType2.X\_4 are valid for quadrature encoders only. EncoderType2.TWO\_PULSE\_COUNTING is valid for two-pulse encoders only.

#### ci\_encoder\_z\_index\_enable

*bool* – Specifies whether to use Z indexing for the channel.

# ci\_encoder\_z\_index\_phase

nidaqmx.constants.EncoderZIndexPhase - Specifies the states at which signal A and signal B must be while signal Z is high for NI-DAQmx to reset the measurement. If signal Z is never high

while signal A and signal B are high, for example, you must choose a phase other than **EncoderZIndex-Phase1.AHIGH BHIGH**.

# ci\_encoder\_z\_index\_val

float – Specifies the value to which to reset the measurement when signal Z is high and signal A and signal B are at the states you specify with **ci\_encoder\_z\_index\_phase**. Specify this value in the units of the measurement.

#### ci\_encoder\_z\_input\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

## ci\_encoder\_z\_input\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the filter recognizes.

## ci\_encoder\_z\_input\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# ci\_encoder\_z\_input\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

#### ci\_encoder\_z\_input\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

# ci\_encoder\_z\_input\_logic\_lvl\_behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior on the input line.

# ci\_encoder\_z\_input\_term

str – Specifies the terminal to which signal Z is connected.

## ci\_encoder\_z\_input\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

# ci\_freq\_dig\_fltr\_enable

*bool* – Specifies whether to apply the pulse width filter to the signal.

# $\verb"ci_freq_dig_fltr_min_pulse_width"$

float – Specifies in seconds the minimum pulse width the filter recognizes.

#### ci\_freq\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# ${\tt ci\_freq\_dig\_fltr\_timebase\_src}$

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

#### ci\_freq\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

# ci\_freq\_div

*int* – Specifies the value by which to divide the input signal if **ci\_freq\_meas\_meth** is **CounterFrequencyMethod.LARGE\_RANGE\_2\_COUNTERS**. The larger the divisor, the more accurate the measurement. However, too large a value could cause the count register to roll over, which results in an incorrect measurement.

## ci\_freq\_enable\_averaging

bool – Specifies whether to enable averaging mode for Sample Clock-timed frequency measurements.

# ci\_freq\_logic\_lvl\_behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior on the input line.

### ci freq meas meth

nidaqmx.constants.CounterFrequencyMethod - Specifies the method to use to measure the frequency of the signal.

# ci\_freq\_meas\_time

float – Specifies in seconds the length of time to measure the frequency of the signal if ci\_freq\_meas\_meth is CounterFrequencyMethod.HIGH\_FREQUENCY\_2\_COUNTERS. Measurement accuracy increases with increased measurement time and with increased signal frequency. If you measure a high-frequency signal for too long, however, the count register could roll over, which results in an incorrect measurement.

# ci\_freq\_starting\_edge

nidaqmx.constants.Edge - Specifies between which edges to measure the frequency of the signal.

### ci\_freq\_term

str – Specifies the input terminal of the signal to measure.

### ci\_freq\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

#### ci freq units

nidaqmx.constants.FrequencyUnits - Specifies the units to use to return frequency measurements.

### ci qps sync method

nidaqmx.constants.GpsSignalType - Specifies the method to use to synchronize the counter to a GPS receiver.

### ci qps sync src

str – Specifies the terminal to which the GPS synchronization signal is connected.

# ci\_lin\_encoder\_dist\_per\_pulse

*float* – Specifies the distance to measure for each pulse the encoder generates on signal A or signal B. This value is in the units you specify with **ci\_lin\_encoder\_units**.

## ci\_lin\_encoder\_initial\_pos

*float* – Specifies the position of the encoder when the measurement begins. This value is in the units you specify with **ci\_lin\_encoder\_units**.

### ci\_lin\_encoder\_units

nidaqmx.constants.LengthUnits - Specifies the units to use to return linear encoder measurements from the channel.

## ci max

*float* – Specifies the maximum value you expect to measure. This value is in the units you specify with a units property. When you query this property, it returns the coerced maximum value that the hardware can measure with the current settings.

### ci\_max\_meas\_period

float – Specifies the maximum period (in seconds) in which the device will recognize signals. For frequency measurements, a signal with a higher period than the one set in this property will return 0 Hz. For duty cycle, the device will return 0 or 1 depending on the state of the line during the max defined period of time. Period measurements will return NaN. Pulse width measurement will return zero.

#### ci meas type

nidagmx.constants.UsageTypeCI - Indicates the measurement to take with the channel.

# ci\_mem\_map\_enable

*bool* – Specifies for NI-DAQmx to map hardware registers to the memory space of the application, if possible. Normally, NI- DAQmx maps hardware registers to memory accessible only to the kernel. Mapping the registers to the memory space of the application increases performance. However, if the application

accesses the memory space mapped to the registers, it can adversely affect the operation of the device and possibly result in a system crash.

### ci min

*float* – Specifies the minimum value you expect to measure. This value is in the units you specify with a units property. When you query this property, it returns the coerced minimum value that the hardware can measure with the current settings.

### ci\_num\_possibly\_invalid\_samps

*int* – Indicates the number of samples that the device might have overwritten before it could transfer them to the buffer.

### ci\_output\_state

nidagmx.constants.Level - Indicates the current state of the out terminal of the counter.

### ci\_period\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

# ci\_period\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the filter recognizes.

### ci\_period\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# ci\_period\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

# ci\_period\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

# ci\_period\_div

*int* – Specifies the value by which to divide the input signal if **ci\_period\_meas\_meth** is **CounterFrequencyMethod.LARGE\_RANGE\_2\_COUNTERS**. The larger the divisor, the more accurate the measurement. However, too large a value could cause the count register to roll over, which results in an incorrect measurement.

# ci\_period\_enable\_averaging

bool – Specifies whether to enable averaging mode for Sample Clock-timed period measurements.

# ci\_period\_logic\_lvl\_behavior

nidaqmx.constants.LogicLv1Behavior - Specifies the logic level behavior on the input line.

# $\verb"ci_period_meas_meth"$

nidaqmx. constants. CounterFrequencyMethod — Specifies the method to use to measure the period of the signal.

### ci\_period\_meas\_time

float – Specifies in seconds the length of time to measure the period of the signal if ci\_period\_meas\_meth is CounterFrequencyMethod.HIGH\_FREQUENCY\_2\_COUNTERS. Measurement accuracy increases with increased measurement time and with increased signal frequency. If you measure a high-frequency signal for too long, however, the count register could roll over, which results in an incorrect measurement.

### ci\_period\_starting\_edge

nidaqmx.constants.Edge - Specifies between which edges to measure the period of the signal.

# ci\_period\_term

str – Specifies the input terminal of the signal to measure.

### ci\_period\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

### ci\_period\_units

nidaqmx.constants.TimeUnits - Specifies the unit to use to return period measurements.

#### ci prescaler

*int* – Specifies the divisor to apply to the signal you connect to the counter source terminal. Scaled data that you read takes this setting into account. You should use a prescaler only when you connect an external signal to the counter source terminal and when that signal has a higher frequency than the fastest onboard timebase. Setting this value disables duplicate count prevention unless you explicitly set **ci\_dup\_count\_prevention** to True.

## ci\_pulse\_freq\_dig\_fltr\_enable

bool – Specifies whether to apply a digital filter to the signal to measure.

# ci\_pulse\_freq\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the filter recognizes.

## ci\_pulse\_freq\_dig\_fltr\_timebase\_rate

float – Specifies in hertz the rate of the digital filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# ci\_pulse\_freq\_dig\_fltr\_timebase\_src

str – Specifies the terminal of the signal to use as the timebase of the digital filter.

### ci\_pulse\_freq\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

## ci\_pulse\_freq\_logic\_lvl\_behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior on the count reset line.

# ci\_pulse\_freq\_starting\_edge

nidaqmx.constants.Edge - Specifies on which edge of the input signal to begin pulse measurement.

# ci\_pulse\_freq\_term

str – Specifies the input terminal of the signal to measure.

# ci\_pulse\_freq\_term\_cfg

 ${\it nidaqmx.constants.TerminalConfiguration}$  - Specifies the input terminal configuration.

# ci\_pulse\_freq\_units

nidaqmx.constants.FrequencyUnits - Specifies the units to use to return pulse specifications in terms of frequency.

### ci\_pulse\_ticks\_dig\_fltr\_enable

bool – Specifies whether to apply a digital filter to the signal to measure.

# ci\_pulse\_ticks\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the filter recognizes.

# ${\tt ci\_pulse\_ticks\_dig\_fltr\_timebase\_rate}$

float – Specifies in hertz the rate of the digital filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# ci\_pulse\_ticks\_dig\_fltr\_timebase\_src

str – Specifies the terminal of the signal to use as the timebase of the digital filter.

# ci\_pulse\_ticks\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

### ci\_pulse\_ticks\_logic\_lvl\_behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior on the count reset line.

### ci\_pulse\_ticks\_starting\_edge

nidaqmx.constants.Edge - Specifies on which edge of the input signal to begin pulse measurement.

### ci\_pulse\_ticks\_term

str – Specifies the input terminal of the signal to measure.

## ci\_pulse\_ticks\_term\_cfg

 ${\it nidaqmx.constants.Terminal Configuration} - {\bf Specifies\ the\ input\ terminal\ configuration}.$ 

# ci\_pulse\_time\_dig\_fltr\_enable

bool – Specifies whether to apply a digital filter to the signal to measure.

### ci\_pulse\_time\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the filter recognizes.

## ci\_pulse\_time\_dig\_fltr\_timebase\_rate

float – Specifies in hertz the rate of the digital filter timebase. NI-DAQmx uses this value to compute settings for the filter.

## ci\_pulse\_time\_dig\_fltr\_timebase\_src

str – Specifies the terminal of the signal to use as the timebase of the digital filter.

## ci\_pulse\_time\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

# ci\_pulse\_time\_logic\_lvl\_behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior on the count reset line.

#### ci\_pulse\_time\_starting\_edge

nidaqmx.constants.Edge - Specifies on which edge of the input signal to begin pulse measurement.

# ci\_pulse\_time\_term

str – Specifies the input terminal of the signal to measure.

#### ci pulse time term cfq

nidaqmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

### ci\_pulse\_time\_units

nidaqmx.constants.TimeUnits - Specifies the units to use to return pulse specifications in terms of high time and low time.

## ci\_pulse\_width\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

# ci\_pulse\_width\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the filter recognizes.

# ci\_pulse\_width\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

### ci pulse width dig fltr timebase src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

# ci\_pulse\_width\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

#### ci pulse width logic lvl behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior on the input line.

### ci\_pulse\_width\_starting\_edge

nidaqmx.constants.Edge - Specifies on which edge of the input signal to begin each pulse width measurement.

## ci\_pulse\_width\_term

str – Specifies the input terminal of the signal to measure.

# ci\_pulse\_width\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

# ci\_pulse\_width\_units

nidaqmx.constants.TimeUnits - Specifies the units to use to return pulse width measurements.

# ci\_samp\_clk\_overrun\_behavior

nidaqmx.constants.SampClkOverrunBehavior – Specifies the counter behavior when data is read but a new value was not detected during a sample clock.

### ci\_samp\_clk\_overrun\_sentinel\_val

int – Specifies the sentinel value returned when the No New Sample Behavior is set to Sentinel Value.

# ci\_semi\_period\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

# ci\_semi\_period\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the filter recognizes.

# $\verb"ci_semi_period_dig_fltr_timebase_rate"$

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

### ci\_semi\_period\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

# ci\_semi\_period\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

### ci semi period logic lvl behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior on the count reset line.

# ci\_semi\_period\_starting\_edge

nidaqmx.constants.Edge – Specifies on which edge of the input signal to begin semi-period measurement. Semi-period measurements alternate between high time and low time, starting on this edge.

### ci\_semi\_period\_term

str – Specifies the input terminal of the signal to measure.

# ci\_semi\_period\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

# ci\_semi\_period\_units

nidaqmx.constants.TimeUnits - Specifies the units to use to return semi-period measurements.

#### ci tc reached

bool – Indicates whether the counter rolled over. When you query this property, NI-DAQmx resets it to False.

## ci\_thresh\_voltage

*float* – Specifies the digital threshold value in Volts for high and low input transitions. Some devices do not support this for differential channels.

### ci timestamp initial seconds

*int* – Specifies the number of seconds that elapsed since the beginning of the current year. This value is ignored if **ci\_gps\_sync\_method** is **GpsSignalType1.IRIGB**.

### ci\_timestamp\_units

nidagmx.constants.TimeUnits - Specifies the units to use to return timestamp measurements.

### ci\_two\_edge\_sep\_first\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

# ci\_two\_edge\_sep\_first\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the filter recognizes.

# ci\_two\_edge\_sep\_first\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# ci\_two\_edge\_sep\_first\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

## ci\_two\_edge\_sep\_first\_dig\_sync\_enable

bool – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

# ci\_two\_edge\_sep\_first\_edge

nidaqmx.constants.Edge - Specifies on which edge of the first signal to start each measurement.

# ci\_two\_edge\_sep\_first\_logic\_lvl\_behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior on the input line.

# ci\_two\_edge\_sep\_first\_term

str – Specifies the source terminal of the digital signal that starts each measurement.

## ci\_two\_edge\_sep\_first\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

# ci\_two\_edge\_sep\_second\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

# ci\_two\_edge\_sep\_second\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the filter recognizes.

# ci\_two\_edge\_sep\_second\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# $\verb"ci_two_edge_sep_second_dig_fltr_timebase\_src"$

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

# ci\_two\_edge\_sep\_second\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

# ci\_two\_edge\_sep\_second\_edge

nidaqmx.constants.Edge - Specifies on which edge of the second signal to stop each measurement.

### ci two edge sep second logic lvl behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior on the count reset line.

## ci\_two\_edge\_sep\_second\_term

str – Specifies the source terminal of the digital signal that stops each measurement.

### ci\_two\_edge\_sep\_second\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

### ci\_two\_edge\_sep\_units

nidaqmx.constants.TimeUnits - Specifies the units to use to return two-edge separation measurements from the channel.

## ci\_usb\_xfer\_req\_count

*int* – Specifies the maximum number of simultaneous USB transfers used to stream data. Modify this value to affect performance under different combinations of operating system and device.

### ci\_usb\_xfer\_req\_size

*int* – Specifies the maximum size of a USB transfer request in bytes. Modify this value to affect performance under different combinations of operating system and device.

### ci\_velocity\_a\_input\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

# ci\_velocity\_a\_input\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the digital filter recognizes.

## ci\_velocity\_a\_input\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# ci\_velocity\_a\_input\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

# ci\_velocity\_a\_input\_logic\_lvl\_behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior of the input terminal.

### ci\_velocity\_a\_input\_term

str – Specifies the terminal to which signal A is connected.

# ci\_velocity\_a\_input\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

# $\verb|ci_velocity_ang_encoder_pulses_per_rev|\\$

*int* – Specifies the number of pulses the encoder generates per revolution. This value is the number of pulses on either signal A or signal B, not the total number of pulses on both signal A and signal B.

### ci\_velocity\_ang\_encoder\_units

nidaqmx.constants.Angular Velocity Units - Specifies the units to use to return angular velocity counter measurements.

## ci\_velocity\_b\_input\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

# ci\_velocity\_b\_input\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the digital filter recognizes.

# ci\_velocity\_b\_input\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

### ci\_velocity\_b\_input\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

# ci\_velocity\_b\_input\_logic\_lvl\_behavior

nidaqmx.constants.LogicLvlBehavior - Specifies the logic level behavior of the input terminal.

### ci\_velocity\_b\_input\_term

*str* – Specifies the terminal to which signal B is connected.

### ci\_velocity\_b\_input\_term\_cfg

nidaqmx.constants.TerminalConfiguration - Specifies the input terminal configuration.

#### ci\_velocity\_div

int – Specifies the value by which to divide the input signal.

# ci\_velocity\_encoder\_decoding\_type

nidaqmx.constants.EncoderType – Specifies how to count and interpret the pulses the encoder generates on signal A and signal B. X1, X2, and X4 are valid for quadrature encoders only. Two Pulse Counting is valid for two-pulse encoders only.

### ci\_velocity\_lin\_encoder\_dist\_per\_pulse

*float* – Specifies the distance to measure for each pulse the encoder generates on signal A or signal B. This value is in the units you specify in CI.Velocity.LinEncoder.DistUnits.

# ci\_velocity\_lin\_encoder\_units

nidaqmx.constants.VelocityUnits - Specifies the units to use to return linear encoder velocity measurements from the channel.

# ci\_velocity\_meas\_time

*float* – Specifies in seconds the length of time to measure the velocity of the signal.

# description

*str* – Specifies a user-defined description for the channel.

### is global

bool – Indicates whether the channel is a global channel.

#### name

str – Specifies the name of the virtual channel this object represents.

# physical\_channel

nidaqmx.system.physical\_channel.PhysicalChannel - Specifies the name of the physical channel upon which this virtual channel is based.

**save** (save\_as=u'', author=u'', overwrite\_existing\_channel=False, allow\_interactive\_editing=True, allow\_interactive\_deletion=True)

Saves this local or global channel to MAX as a global channel.

# **Parameters**

- **save\_as** (Optional[str]) Is the name to save the task, global channel, or custom scale as. If you do not specify a value for this input, NI-DAQmx uses the name currently assigned to the task, global channel, or custom scale.
- author (Optional[str]) Is a name to store with the task, global channel, or custom scale.
- overwrite\_existing\_channel (Optional[bool]) Specifies whether to overwrite a global channel of the same name if one is already saved in MAX. If this input is False and a global channel of the same name is already saved in MAX, this function returns an error.

- allow\_interactive\_editing (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be edited in the DAQ Assistant. If allow\_interactive\_editing is True, the DAQ Assistant must support all task or global channel settings.
- allow\_interactive\_deletion (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be deleted through MAX.

# nidagmx.task.co channel

Represents one or more counter output virtual channels and their properties.

### chan\_type

nidaqmx.constants.ChannelType - Indicates the type of the virtual channel.

#### channel names

*List[str]* – Specifies the unflattened list of the virtual channels.

### co\_auto\_incr\_cnt

*int* – Specifies a number of timebase ticks by which to increase the time spent in the idle state for each successive pulse.

### co\_constrained\_gen\_mode

nidaqmx.constants.ConstrainedGenMode – Specifies constraints to apply when the counter generates pulses. Constraining the counter reduces the device resources required for counter operation. Constraining the counter can also allow additional analog or counter tasks on the device to run concurrently. For continuous counter tasks, NI-DAQmx consumes no device resources when the counter is constrained. For finite counter tasks, resource use increases with the frequency regardless of the constraint mode. However, fixed frequency constraints significantly reduce resource usage, and fixed duty cycle constraint marginally reduces it.

#### co count

int – Indicates the current value of the count register.

## co\_ctr\_timebase\_active\_edge

nidaqmx.constants.Edge - Specifies whether a timebase cycle is from rising edge to rising edge or from falling edge to falling edge.

# co\_ctr\_timebase\_dig\_fltr\_enable

*bool* – Specifies whether to apply the pulse width filter to the signal.

### co\_ctr\_timebase\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the filter recognizes.

# co\_ctr\_timebase\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

### co\_ctr\_timebase\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

## co\_ctr\_timebase\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

#### co ctr timebase master timebase div

*int* – Specifies the divisor for an external counter timebase. You can divide the counter timebase in order to generate slower signals without causing the count register to roll over.

#### co ctr timebase rate

*float* – Specifies in Hertz the frequency of the counter timebase. Specifying the rate of a counter timebase allows you to define output pulses in seconds rather than in ticks of the timebase. If you use an external timebase and do not specify the rate, you can define output pulses only in ticks of the timebase.

# co\_ctr\_timebase\_src

*str* – Specifies the terminal of the timebase to use for the counter. Typically, NI-DAQmx uses one of the internal counter timebases when generating pulses. Use this property to specify an external timebase and produce custom pulse widths that are not possible using the internal timebases.

#### co\_data\_xfer\_mech

nidaqmx.constants.DataTransferActiveTransferMode - Specifies the data transfer mode for the device. For buffered operations, use DMA or USB Bulk. For non-buffered operations, use Polled.

## co\_data\_xfer\_req\_cond

nidaqmx.constants.OutputDataTransferCondition - Specifies under what condition to transfer data from the buffer to the onboard memory of the device.

# co\_enable\_initial\_delay\_on\_retrigger

bool – Specifies whether to apply the initial delay to retriggered pulse trains.

### co\_mem\_map\_enable

bool – Specifies for NI-DAQmx to map hardware registers to the memory space of the application, if possible. Normally, NI- DAQmx maps hardware registers to memory accessible only to the kernel. Mapping the registers to the memory space of the application increases performance. However, if the application accesses the memory space mapped to the registers, it can adversely affect the operation of the device and possibly result in a system crash.

## co output state

nidaqmx.constants.Level - Indicates the current state of the output terminal of the counter.

# co\_output\_type

nidagmx.constants.UsageTypeCO - Indicates how to define pulses generated on the channel.

#### co\_prescaler

*int* – Specifies the divisor to apply to the signal you connect to the counter source terminal. Pulse generations defined by frequency or time take this setting into account, but pulse generations defined by ticks do not. You should use a prescaler only when you connect an external signal to the counter source terminal and when that signal has a higher frequency than the fastest onboard timebase.

## co\_pulse\_done

bool – Indicates if the task completed pulse generation. Use this value for retriggerable pulse generation when you need to determine if the device generated the current pulse. For retriggerable tasks, when you query this property, NI-DAQmx resets it to False.

# co\_pulse\_duty\_cyc

*float* – Specifies the duty cycle of the pulses. The duty cycle of a signal is the width of the pulse divided by period. NI- DAQmx uses this ratio and the pulse frequency to determine the width of the pulses and the delay between pulses.

# co\_pulse\_freq

float – Specifies the frequency of the pulses to generate. This value is in the units you specify with **co\_pulse\_freq\_units** or when you create the channel.

# co\_pulse\_freq\_initial\_delay

*float* – Specifies in seconds the amount of time to wait before generating the first pulse.

#### co pulse freq units

nidaqmx.constants.FrequencyUnits - Specifies the units in which to define pulse frequency.

# co\_pulse\_high\_ticks

*int* – Specifies the number of ticks the pulse is high.

# co\_pulse\_high\_time

*float* – Specifies the amount of time that the pulse is at a high voltage. This value is in the units you specify with **co\_pulse\_time\_units** or when you create the channel.

# co\_pulse\_idle\_state

nidagmx.constants.Level - Specifies the resting state of the output terminal.

#### co\_pulse\_low\_ticks

*int* – Specifies the number of ticks the pulse is low.

# co\_pulse\_low\_time

*float* – Specifies the amount of time that the pulse is at a low voltage. This value is in the units you specify with **co\_pulse\_time\_units** or when you create the channel.

#### co pulse term

str – Specifies on which terminal to generate pulses.

# co\_pulse\_ticks\_initial\_delay

int – Specifies the number of ticks to wait before generating the first pulse.

### co\_pulse\_time\_initial\_delay

float – Specifies in seconds the amount of time to wait before generating the first pulse.

### co pulse time units

nidaqmx.constants.TimeUnits - Specifies the units in which to define high and low pulse time.

# co\_rdy\_for\_new\_val

bool – Indicates whether the counter is ready for new continuous pulse train values.

# co\_usb\_xfer\_req\_count

*int* – Specifies the maximum number of simultaneous USB transfers used to stream data. Modify this value to affect performance under different combinations of operating system and device.

### co\_usb\_xfer\_req\_size

*int* – Specifies the maximum size of a USB transfer request in bytes. Modify this value to affect performance under different combinations of operating system and device.

# co\_use\_only\_on\_brd\_mem

*bool* – Specifies whether to write samples directly to the onboard memory of the device, bypassing the memory buffer. Generally, you cannot update onboard memory directly after you start the task. Onboard memory includes data FIFOs.

#### description

str – Specifies a user-defined description for the channel.

# is\_global

*bool* – Indicates whether the channel is a global channel.

### name

str – Specifies the name of the virtual channel this object represents.

# physical\_channel

nidaqmx.system.physical\_channel.PhysicalChannel - Specifies the name of the physical channel upon which this virtual channel is based.

**save** (save\_as=u'', author=u'', overwrite\_existing\_channel=False, allow\_interactive\_editing=True, allow\_interactive\_deletion=True)

Saves this local or global channel to MAX as a global channel.

#### **Parameters**

- **save\_as** (Optional[str]) Is the name to save the task, global channel, or custom scale as. If you do not specify a value for this input, NI-DAQmx uses the name currently assigned to the task, global channel, or custom scale.
- author (Optional[str]) Is a name to store with the task, global channel, or custom scale.
- overwrite\_existing\_channel (Optional[bool]) Specifies whether to overwrite a global channel of the same name if one is already saved in MAX. If this input is False and a global channel of the same name is already saved in MAX, this function returns an error.
- allow\_interactive\_editing (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be edited in the DAQ Assistant. If allow\_interactive\_editing is True, the DAQ Assistant must support all task or global channel settings.
- allow\_interactive\_deletion (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be deleted through MAX.

## nidagmx.task.di channel

class nidaqmx.\_task\_modules.channels.di\_channel.DIChannel(task\_handle, virtual\_or\_physical\_name)

Bases: nidaqmx.\_task\_modules.channels.channel

Represents one or more digital input virtual channels and their properties.

# chan\_type

nidagmx.constants.ChannelType - Indicates the type of the virtual channel.

### channel\_names

*List[str]* – Specifies the unflattened list of the virtual channels.

# description

*str* – Specifies a user-defined description for the channel.

### di acquire on

nidaqmx.constants.ActiveOrInactiveEdgeSelection - Specifies on which edge of the sample clock to acquire samples.

### di\_data\_xfer\_mech

nidaqmx.constants.DataTransferActiveTransferMode - Specifies the data transfer mode for the device.

# ${\tt di\_data\_xfer\_req\_cond}$

nidaqmx.constants.InputDataTransferCondition - Specifies under what condition to transfer data from the onboard memory of the device to the buffer.

# di\_dig\_fltr\_enable

*bool* – Specifies whether to enable the digital filter for the line(s) or port(s). You can enable the filter on a line-by- line basis. You do not have to enable the filter for all lines in a channel.

# di\_dig\_fltr\_enable\_bus\_mode

bool – Specifies whether to enable bus mode for digital filtering. If you set this property to True, NI-DAQmx treats all lines that use common filtering settings as a bus. If any line in the bus has jitter, all lines in the bus hold state until the entire bus stabilizes, or until 2 times the minimum pulse width elapses. If you set this property to False, NI-DAQmx filters all lines individually. Jitter in one line does not affect other lines.

### di\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the filter recognizes as a valid high or low state transition.

# di\_dig\_fltr\_timebase\_rate

float – Specifies in hertz the rate of the digital filter timebase. NI-DAQmx uses this value to compute settings for the filter.

### di\_dig\_fltr\_timebase\_src

str – Specifies the terminal of the signal to use as the timebase of the digital filter.

## di dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

### di invert lines

*bool* – Specifies whether to invert the lines in the channel. If you set this property to True, the lines are at high logic when off and at low logic when on.

## di\_logic\_family

nidaqmx.constants.LogicFamily - Specifies the logic family to use for acquisition. A logic family corresponds to voltage thresholds that are compatible with a group of voltage standards. Refer to the device documentation for information on the logic high and logic low voltages for these logic families.

# di\_mem\_map\_enable

bool – Specifies for NI-DAQmx to map hardware registers to the memory space of the application, if possible. Normally, NI- DAQmx maps hardware registers to memory accessible only to the kernel. Mapping the registers to the memory space of the application increases performance. However, if the application accesses the memory space mapped to the registers, it can adversely affect the operation of the device and possibly result in a system crash.

### di\_num\_lines

*int* – Indicates the number of digital lines in the channel.

### di tristate

bool – Specifies whether to tristate the lines in the channel. If you set this property to True, NI-DAQmx tristates the lines in the channel. If you set this property to False, NI-DAQmx does not modify the configuration of the lines even if the lines were previously tristated. Set this property to False to read lines in other tasks or to read output-only lines.

### di\_usb\_xfer\_req\_count

*int* – Specifies the maximum number of simultaneous USB transfers used to stream data. Modify this value to affect performance under different combinations of operating system and device.

## di\_usb\_xfer\_req\_size

*int* – Specifies the maximum size of a USB transfer request in bytes. Modify this value to affect performance under different combinations of operating system and device.

## is\_global

bool - Indicates whether the channel is a global channel.

# name

*str* – Specifies the name of the virtual channel this object represents.

### physical channel

nidaqmx.system.physical\_channel.PhysicalChannel - Specifies the name of the physical channel upon which this virtual channel is based.

save (save\_as=u'', author=u'', overwrite\_existing\_channel=False, allow\_interactive\_editing=True, allow\_interactive\_deletion=True)
Saves this local or global channel to MAX as a global channel.

### **Parameters**

- **save\_as** (Optional[str]) Is the name to save the task, global channel, or custom scale as. If you do not specify a value for this input, NI-DAQmx uses the name currently assigned to the task, global channel, or custom scale.
- **author** (Optional[str]) Is a name to store with the task, global channel, or custom scale.
- overwrite\_existing\_channel (Optional[bool]) Specifies whether to overwrite a global channel of the same name if one is already saved in MAX. If this input is False and a global channel of the same name is already saved in MAX, this function returns an error.
- allow\_interactive\_editing (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be edited in the DAQ Assistant. If allow\_interactive\_editing is True, the DAQ Assistant must support all task or global channel settings.
- allow\_interactive\_deletion (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be deleted through MAX.

# nidagmx.task.do channel

class nidaqmx.\_task\_modules.channels.do\_channel.DOChannel(task\_handle, virtual\_or\_physical\_name)
Bases: nidaqmx.\_task\_modules.channels.channel.Channel

Represents one or more digital output virtual channels and their properties.

### chan\_type

nidaqmx.constants.ChannelType - Indicates the type of the virtual channel.

# channel names

*List[str]* – Specifies the unflattened list of the virtual channels.

### description

str – Specifies a user-defined description for the channel.

### do\_data\_xfer\_mech

nidaqmx.constants.DataTransferActiveTransferMode-Specifies the data transfer mode for the device.

# ${\tt do\_data\_xfer\_req\_cond}$

nidaqmx.constants.OutputDataTransferCondition - Specifies under what condition to transfer data from the buffer to the onboard memory of the device.

### do\_generate\_on

nidaqmx.constants.ActiveOrInactiveEdgeSelection - Specifies on which edge of the sample clock to generate samples.

### do invert lines

*bool* – Specifies whether to invert the lines in the channel. If you set this property to True, the lines are at high logic when off and at low logic when on.

### do\_line\_states\_done\_state

nidaqmx.constants.Level - Specifies the state of the lines in a digital output task when the task completes execution.

### do line states paused state

nidaqmx.constants.Level - Specifies the state of the lines in a digital output task when the task pauses.

# do\_line\_states\_start\_state

nidaqmx.constants.Level - Specifies the state of the lines in a digital output task when the task starts.

# do\_logic\_family

nidaqmx.constants.LogicFamily – Specifies the logic family to use for generation. A logic family corresponds to voltage thresholds that are compatible with a group of voltage standards. Refer to the device documentation for information on the logic high and logic low voltages for these logic families.

#### do\_mem\_map\_enable

*bool* – Specifies for NI-DAQmx to map hardware registers to the memory space of the application, if possible. Normally, NI- DAQmx maps hardware registers to memory accessible only to the kernel. Mapping the registers to the memory space of the application increases performance. However, if the application accesses the memory space mapped to the registers, it can adversely affect the operation of the device and possibly result in a system crash.

#### do num lines

*int* – Indicates the number of digital lines in the channel.

### do\_output\_drive\_type

nidaqmx.constants.DigitalDriveType - Specifies the drive type for digital output channels.

# do\_overcurrent\_auto\_reenable

*bool* – Specifies whether to automatically reenable channels after they no longer exceed the current limit specified by **do\_overcurrent\_limit**.

### do\_overcurrent\_limit

*float* – Specifies the current threshold in Amperes for the channel. A value of 0 means the channel observes no limit. Devices can monitor only a finite number of current thresholds simultaneously. If you attempt to monitor additional thresholds, NI-DAQmx returns an error.

# do\_overcurrent\_reenable\_period

float – Specifies the delay in seconds between the time a channel no longer exceeds the current limit and the reactivation of that channel, if **do overcurrent auto reenable** is True.

### do\_tristate

*bool* – Specifies whether to stop driving the channel and set it to a high-impedance state. You must commit the task for this setting to take effect.

# do\_usb\_xfer\_req\_count

*int* – Specifies the maximum number of simultaneous USB transfers used to stream data. Modify this value to affect performance under different combinations of operating system and device.

### do\_usb\_xfer\_req\_size

*int* – Specifies the maximum size of a USB transfer request in bytes. Modify this value to affect performance under different combinations of operating system and device.

# do\_use\_only\_on\_brd\_mem

bool - Specifies whether to write samples directly to the onboard memory of the device, bypassing the

memory buffer. Generally, you cannot update onboard memory after you start the task. Onboard memory includes data FIFOs.

# is\_global

bool – Indicates whether the channel is a global channel.

#### name

str – Specifies the name of the virtual channel this object represents.

### physical\_channel

nidaqmx.system.physical\_channel.PhysicalChannel - Specifies the name of the physical channel upon which this virtual channel is based.

**save** (save\_as=u'', author=u'', overwrite\_existing\_channel=False, allow\_interactive\_editing=True, allow\_interactive\_deletion=True)

Saves this local or global channel to MAX as a global channel.

### **Parameters**

- **save\_as** (Optional[str]) Is the name to save the task, global channel, or custom scale as. If you do not specify a value for this input, NI-DAQmx uses the name currently assigned to the task, global channel, or custom scale.
- author (Optional[str]) Is a name to store with the task, global channel, or custom scale.
- overwrite\_existing\_channel (Optional[bool]) Specifies whether to overwrite a global channel of the same name if one is already saved in MAX. If this input is False and a global channel of the same name is already saved in MAX, this function returns an error.
- allow\_interactive\_editing (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be edited in the DAQ Assistant. If allow\_interactive\_editing is True, the DAQ Assistant must support all task or global channel settings.
- allow\_interactive\_deletion (Optional[bool]) Specifies whether to allow the task, global channel, or custom scale to be deleted through MAX.

# nidaqmx.task.channel\_collection

Contains the collection of channels for a DAQmx Task.

This class defines methods that implements a container object.

#### all

nidaqmx.\_task\_modules.channels.channel.Channel - Specifies a channel object that represents the entire list of virtual channels on this channel collection.

# channel\_names

*List[str]* – Specifies the entire list of virtual channels on this channel collection.

## nidagmx.task.ai channel collection

```
class nidaqmx._task_modules.ai_channel_collection.AIChannelCollection (task_handle)
    Bases: nidaqmx._task_modules.channel_collection.ChannelCollection
```

Contains the collection of analog input channels for a DAQmx Task.

```
add_ai_accel_4_wire_dc_voltage_chan (physical_channel, name_to_assign_to_channel=u'',
                                                terminal config=<TerminalConfiguration.DEFAULT:
                                                -1>,
                                                            min\ val=-5.0,
                                                                                 max val=5.0,
                                                units=<AccelUnits.G:
                                                                                      10186>,
                                                sensitivity=1000.0,
                                                                                      sensitiv-
                                                ity units=<AccelSensitivityUnits.M VOLTS PER G:
                                                12509>, voltage excit source=<ExcitationSource.INTERNAL:
                                                                         voltage_excit_val=0.0,
                                                10200>.
                                                use_excit_for_scaling=False,
                                                                                          cus-
                                                tom_scale name=u'')
```

Creates channel(s) to measure acceleration. Use this instance for custom sensors that require excitation. You can use the excitation to scale the measurement.

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config
   TerminalConfiguration] Specifies the input terminal configuration for
  the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional [nidaqmx.constants.AccelUnits]) Specifies the units to use to return acceleration measurements from the channel.
- **sensitivity** (Optional[float]) Is the sensitivity of the sensor. This value is in the units you specify with the **sensitivity\_units** input. Refer to the sensor documentation to determine this value.
- **sensitivity\_units** (Optional[nidaqmx.constants. AccelSensitivityUnits]) Specifies the units of the **sensitivity** input.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **voltage\_excit\_val** (Optional[float]) Specifies in volts the amount of excitation supplied to the sensor. Refer to the sensor documentation to determine appropriate excitation values.
- use\_excit\_for\_scaling (Optional[bool]) Specifies if NI- DAQmx divides the measurement by the excitation. You should typically set use\_excit\_for\_scaling to True for ratiometric transducers. If you set use\_excit\_for\_scaling to True, set max\_val and min\_val to reflect the scaling.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_ai_accel_chan (physical_channel, name_to_assign_to_channel=u'', termi-
nal_config=<TerminalConfiguration.DEFAULT: -1>, min_val=-5.0,
max_val=5.0, units=<AccelUnits.G: 10186>, sensitivity=1000.0, sen-
sitivity_units=<AccelSensitivityUnits.M_VOLTS_PER_G: 12509>,
current_excit_source=<ExcitationSource.INTERNAL: 10200>, cur-
rent_excit_val=0.004, custom_scale_name=u'')
Creates channel(s) that use an accelerometer to measure acceleration.
```

#### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config
   TerminalConfiguration]

   Specifies the input terminal configuration for the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional [nidaqmx.constants.AccelUnits]) Specifies the units to use to return acceleration measurements from the channel.
- **sensitivity** (Optional[float]) Is the sensitivity of the sensor. This value is in the units you specify with the **sensitivity\_units** input. Refer to the sensor documentation to determine this value.
- sensitivity\_units (Optional[nidaqmx.constants. AccelSensitivityUnits]) Specifies the units of the sensitivity input.
- current\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **current\_excit\_val** (Optional[float]) Specifies in amperes the amount of excitation to supply to the sensor. Refer to the sensor documentation to determine this value.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

Return type nidagmx. task modules.channels.ai channel.AIChannel

Creates channel(s) that use a charge-based sensor to measure acceleration.

#### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config (Optional[nidaqmx.constants. TerminalConfiguration]) Specifies the input terminal configuration for the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional [nidaqmx.constants.AccelUnits]) Specifies the units to use to return acceleration measurements from the channel.
- **sensitivity** (Optional[float]) Is the sensitivity of the sensor. This value is in the units you specify with the **sensitivity\_units** input. Refer to the sensor documentation to determine this value.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** nidagmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_ai_bridge_chan (physical_channel, name_to_assign_to_channel=u'', min_val=-0.002, max_val=0.002, units=<BridgeUnits.VOLTS_PER_VOLTS: 15896>, bridge_config=<BridgeConfiguration.FULL_BRIDGE: 10182>, voltage_excit_source=<ExcitationSource.INTERNAL: 10200>, voltage_excit_val=2.5, nominal_bridge_resistance=350.0, custom scale name=u'')
```

Creates channel(s) that measure voltage ratios from a Wheatstone bridge. Use this instance with bridge-based sensors that measure phenomena other than strain, force, pressure, or torque, or that scale data to physical units NI-DAQmx does not support.

# **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.

- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.BridgeUnits]) Specifies in which unit to return voltage ratios from the channel.
- bridge\_config (Optional[nidaqmx.constants. BridgeConfiguration]) Specifies information about the bridge configuration and measurement.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies information about the bridge configuration and measurement.
- **voltage\_excit\_val** (Optional[float]) Specifies information about the bridge configuration and measurement.
- nominal\_bridge\_resistance (Optional[float]) Specifies information about the bridge configuration and measurement.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM CUSTOM SCALE**.

**Return type** nidagmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_ai_charge_chan (physical_channel, name_to_assign_to_channel=u'', termi-
nal_config=<TerminalConfiguration.DEFAULT: -1>, min_val=-1e-09,
max_val=1e-09, units=<ChargeUnits.COULOMBS: 16102>, cus-
tom_scale_name=u'')
```

Creates channel(s) that use a sensor with charge output.

## **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config
   TerminalConfiguration]

   Specifies the input terminal configuration for the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.ChargeUnits]) Specifies the units to use to return charge measurements from the channel.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

Returns Indicates the newly created channel object.

**Return type** nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_ai_current_chan (physical_channel, name_to_assign_to_channel=u'', termi-nal_config=<TerminalConfiguration.DEFAULT: -1>, min_val=-0.01, max_val=0.01, units=<CurrentUnits.AMPS: 10342>, shunt_resistor_loc=<CurrentShuntResistorLocation.LET_DRIVER_CHOOSE: -1>, ext_shunt_resistor_val=249.0, custom_scale_name=u'')

Creates channel(s) to measure current.
```

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config
   TerminalConfiguration]

   Specifies the input terminal configuration for the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.CurrentUnits]) Specifies the units to use to return current measurements.
- shunt\_resistor\_loc (Optional[nidaqmx.constants. CurrentShuntResistorLocation]) Specifies the location of the shunt resistor. For devices with built-in shunt resistors, specify the location as INTERNAL. For devices that do not have built-in shunt resistors, you must attach an external one, set this input to EXTERNAL and use the ext\_shunt\_resistor\_val input to specify the value of the resistor.
- ext\_shunt\_resistor\_val (Optional[float]) Specifies in ohms the resistance of an external shunt resistor.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** nidagmx. task modules.channels.ai channel.AIChannel

```
add_ai_current_rms_chan (physical_channel, name_to_assign_to_channel=u'', termi-nal_config=<TerminalConfiguration.DEFAULT: -1>, min_val=-0.01, max_val=0.01, units=<CurrentUnits.AMPS: 10342>, shunt_resistor_loc=<CurrentShuntResistorLocation.LET_DRIVER_CHOOSE: -1>, ext_shunt_resistor_val=249.0, custom_scale_name=u'')

Creates a channel to measure current RMS, the average (mean) power of the acquired current.
```

#### **Parameters**

• **physical\_channel** (str) – Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.

- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config (Optional[nidaqmx.constants. TerminalConfiguration]) Specifies the input terminal configuration for the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.CurrentUnits]) Specifies the units to use to return current measurements.
- shunt\_resistor\_loc (Optional[nidaqmx.constants. CurrentShuntResistorLocation]) Specifies the location of the shunt resistor. For devices with built-in shunt resistors, specify the location as INTERNAL. For devices that do not have built-in shunt resistors, you must attach an external one, set this input to EXTERNAL and use the ext\_shunt\_resistor\_val input to specify the value of the resistor.
- ext\_shunt\_resistor\_val (Optional[float]) Specifies in ohms the resistance of an external shunt resistor.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

Return type nidagmx. task modules.channels.ai channel.AIChannel

```
add_ai_force_bridge_polynomial_chan (physical_channel, name_to_assign_to_channel=u'',
                                                min\ val = -100.0,
                                                                               max_val = 100.0,
                                                units=<ForceUnits.POUNDS:
                                                                                       15876>,
                                                bridge_config=<BridgeConfiguration.FULL_BRIDGE:</pre>
                                                10182>, voltage excit source=<ExcitationSource.INTERNAL:
                                                10200>.
                                                              voltage \ excit \ val=2.5,
                                                                                         nom-
                                                inal bridge resistance=350.0,
                                                                                           for-
                                                ward_coeffs=None, reverse_coeffs=None, electri-
                                                cal units=<BridgeElectricalUnits.M VOLTS PER VOLT:
                                                15897>, physical_units=<BridgePhysicalUnits.POUNDS:
                                                15876>, custom scale name=u'')
```

Creates channel(s) that use a Wheatstone bridge to measure force or load. Use this instance with sensors whose specifications provide a polynomial to convert electrical values to physical values. When you use this scaling type, NI-DAQmx requires coefficients for a polynomial that converts electrical values to physical values (forward), as well as coefficients for a polynomial that converts physical values to electrical values (reverse). If you only know one set of coefficients, use the DAQmx Compute Reverse Polynomial Coefficients function to generate the other set.

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input,

NI-DAQmx uses the physical channel name as the virtual channel name.

- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional [nidaqmx.constants.ForceUnits]) Specifies in which unit to return force measurements from the channel.
- bridge\_config (Optional[nidaqmx.constants. BridgeConfiguration]) Specifies information about the bridge configuration and measurement.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies information about the bridge configuration and measurement.
- voltage\_excit\_val (Optional[float]) Specifies information about the bridge configuration and measurement.
- nominal\_bridge\_resistance (Optional[float]) Specifies information about the bridge configuration and measurement.
- **forward\_coeffs** (Optional [List[float]]) Specifies how to scale electrical values from the sensor to physical units.
- reverse\_coeffs (Optional[List[float]]) Specifies how to scale electrical values from the sensor to physical units.
- electrical\_units (Optional[nidaqmx.constants. BridgeElectricalUnits]) - Specifies how to scale electrical values from the sensor to physical units.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx*. *task modules.channels.ai channel.AIChannel* 

```
add ai force bridge table chan (physical channel,
                                                               name to assign to channel=u'',
                                         min\ val = -100.0,
                                                                               max \ val = 100.0,
                                         units=<ForceUnits.POUNDS:
                                                                                      15876>,
                                         bridge_config=<BridgeConfiguration.FULL_BRIDGE:
                                         10182>, voltage_excit_source=<ExcitationSource.INTERNAL:
                                         10200>.
                                                                         voltage_excit_val=2.5,
                                         nominal_bridge_resistance=350.0,
                                         electrical_vals=None,
                                                                                       electri-
                                         cal_units=<BridgeElectricalUnits.M_VOLTS_PER_VOLT:
                                         15897>,
                                                          physical_vals=None,
                                                                                        physi-
                                         cal_units=<BridgePhysicalUnits.POUNDS:
                                                                                      15876>,
                                         custom_scale_name=u'')
```

Creates channel(s) that use a Wheatstone bridge to measure force or load. Use this instance with sensors whose specifications provide a table of electrical values and the corresponding physical values. When you use this scaling type, NI-DAQmx performs linear scaling between each pair of electrical and physical

values. The input limits specified with **min\_val** and **max\_val** must fall within the smallest and largest physical values. For any data outside those endpoints, NI-DAQmx coerces that data to the endpoints.

#### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional [nidaqmx.constants.ForceUnits]) Specifies in which unit to return force measurements from the channel.
- bridge\_config (Optional[nidaqmx.constants. BridgeConfiguration]) Specifies information about the bridge configuration and measurement.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies information about the bridge configuration and measurement.
- **voltage\_excit\_val** (Optional[float]) Specifies information about the bridge configuration and measurement.
- nominal\_bridge\_resistance (Optional[float]) Specifies information about the bridge configuration and measurement.
- **electrical\_vals** (Optional[List[float]]) Specifies how to scale electrical values from the sensor to physical units.
- electrical\_units (Optional[nidaqmx.constants. BridgeElectricalUnits]) Specifies how to scale electrical values from the sensor to physical units.
- **physical\_vals** (Optional[List[float]]) Specifies how to scale electrical values from the sensor to physical units.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx*.\_*task*\_*modules.channels.ai*\_*channel.AIChannel* 

```
add_ai_force_bridge_two_point_lin_chan (physical_channel,
```

```
name_to_assign_to_channel=u'',
min\ val = -100.0,
                            max val=100.0,
units=<ForceUnits.POUNDS:
                                   15876>,
bridge config=<BridgeConfiguration.FULL BRIDGE:
10182>,
age excit source=<ExcitationSource.INTERNAL:
10200>.
                      voltage excit val=2.5,
nominal bridge resistance=350.0,
first_electrical_val=0.0,
                                       sec-
ond_electrical_val=2.0,
                                    electri-
cal units=<BridgeElectricalUnits.M_VOLTS_PER_VOLT:
            first_physical_val=0.0,
15897>.
                                       sec-
ond\_physical\_val = 100.0,
                                     physi-
cal_units=<BridgePhysicalUnits.POUNDS:
15876>, custom_scale_name=u'')
```

Creates channel(s) that use a Wheatstone bridge to measure force or load. Use this instance with sensors whose specifications do not provide a polynomial for scaling or a table of electrical and physical values. When you use this scaling type, NI-DAQmx uses two points of electrical and physical values to calculate the slope and y-intercept of a linear equation and uses that equation to scale electrical values to physical values.

#### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.ForceUnits]) Specifies in which unit to return force measurements from the channel.
- bridge\_config (Optional[nidaqmx.constants. BridgeConfiguration]) Specifies information about the bridge configuration and measurement.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies information about the bridge configuration and measurement.
- **voltage\_excit\_val** (Optional[float]) Specifies information about the bridge configuration and measurement.
- nominal\_bridge\_resistance (Optional[float]) Specifies information about the bridge configuration and measurement.
- **first\_electrical\_val** (Optional[float]) Specifies how to scale electrical values from the sensor to physical units.
- **second\_electrical\_val** (Optional[float]) Specifies how to scale electrical values from the sensor to physical units.

- electrical\_units (Optional[nidaqmx.constants. BridgeElectricalUnits]) Specifies how to scale electrical values from the sensor to physical units.
- **first\_physical\_val** (Optional[float]) Specifies how to scale electrical values from the sensor to physical units.
- **second\_physical\_val** (Optional[float]) Specifies how to scale electrical values from the sensor to physical units.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Return type** *nidagmx*. *task modules.channels.ai channel.AIChannel* 

```
add_ai_force_iepe_chan (physical_channel, name_to_assign_to_channel=u'', terminal_config=<TerminalConfiguration.DEFAULT:
-1>, min_val=-2000.0, max_val=2000.0, units=<ForceUnits.NEWTONS: 15875>, sensitivity=2.25, sensitivity_units=<ForceIEPESensorSensitivityUnits.M_VOLTS_PER_NEWTON: 15891>, current_excit_source=<ExcitationSource.INTERNAL: 10200>, current_excit_val=0.004, custom_scale_name=u'') Creates channel(s) that use an IEPE force sensor to measure force or load.
```

## **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config (Optional[nidaqmx.constants. TerminalConfiguration]) Specifies the input terminal configuration for the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional [nidaqmx.constants.ForceUnits]) Specifies in which unit to return force measurements from the channel.
- **sensitivity** (Optional[float]) Is the sensitivity of the sensor. This value is in the units you specify with the **sensitivity\_units** input. Refer to the sensor documentation to determine this value.
- sensitivity\_units (Optional[nidaqmx.constants. ForceIEPESensorSensitivityUnits]) Specifies the units of the sensitivity input.

- current\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **current\_excit\_val** (Optional[float]) Specifies in amperes the amount of excitation to supply to the sensor. Refer to the sensor documentation to determine this value.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM CUSTOM SCALE**.

**Return type** *nidagmx*.\_*task*\_*modules.channels.ai*\_*channel.AIChannel* 

```
\begin{tabular}{ll} {\bf add\_ai\_freq\_voltage\_chan} & (physical\_channel, & name\_to\_assign\_to\_channel=u'', & min\_val=1, \\ & max\_val=100, & units=<&FrequencyUnits.HZ: & 10373>, & threshold\_level=0.0, & hysteresis=0.0, & custom\_scale\_name=u'') \\ \end{tabular}
```

Creates channel(s) that use a frequency-to-voltage converter to measure frequency.

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.FrequencyUnits]) Specifies the units to use to return frequency measurements.
- threshold\_level (Optional[float]) Specifies in volts the level at which to recognize waveform repetitions. You should select a voltage level that occurs only once within the entire period of a waveform. You also can select a voltage that occurs only once while the voltage rises or falls.
- hysteresis (Optional[float]) Specifies in volts a window below level. The input voltage must pass below threshold\_level minus hysteresis before NI-DAQmx recognizes a waveform repetition. Hysteresis can improve measurement accuracy when the signal contains noise or jitter.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx*.\_*task*\_*modules.channels.ai*\_*channel*.*AIChannel* 

```
add_ai_microphone_chan (physical_channel, name_to_assign_to_channel=u'', terminal_config=<TerminalConfiguration.DEFAULT:
-1>, units=<SoundPressureUnits.PA: 10081>, mic_sensitivity=10.0, max_snd_press_level=100.0, current_excit_source=<ExcitationSource.INTERNAL: 10200>, current_excit_val=0.004, custom_scale_name=u'')
```

Creates channel(s) that use a microphone to measure sound pressure.

#### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config (Optional[nidaqmx.constants. TerminalConfiguration]) Specifies the input terminal configuration for the channel.
- units (Optional[nidaqmx.constants.SoundPressureUnits]) Specifies the units to use to return sound pressure measurements.
- mic\_sensitivity (Optional[float]) Is the sensitivity of the microphone. Specify this value in mV/Pa.
- max\_snd\_press\_level (Optional[float]) Is the maximum instantaneous sound pressure level you expect to measure. This value is in decibels, referenced to 20 micropascals.
- current\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **current\_excit\_val** (Optional[float]) Specifies in amperes the amount of excitation to supply to the sensor. Refer to the sensor documentation to determine this value.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** nidagmx.\_task\_modules.channels.ai\_channel.AIChannel

Creates channel(s) that use an eddy current proximity probe to measure position.

# Parameters

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.

- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.LengthUnits]) Specifies the units to use to return position measurements from the channel.
- **sensitivity** (Optional[float]) Is the sensitivity of the sensor. This value is in the units you specify with the **sensitivity\_units** input. Refer to the sensor documentation to determine this value.
- sensitivity\_units (Optional[nidaqmx.constants. EddyCurrentProxProbeSensitivityUnits]) Specifies the units of the sensitivity input.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Return type** *nidagmx*. *task modules.channels.ai channel.AIChannel* 

```
add_ai_pos_lvdt_chan (physical_channel, name_to_assign_to_channel=u'', min_val=-0.1, max_val=0.1, units=<LengthUnits.METERS: 10219>, sensitivity=50.0, sensitivity_units=<LVDTSensitivityUnits.M_VOLTS_PER_VOLT_PER_MILLIMETER: 12506>, voltage_excit_source=<ExcitationSource.INTERNAL: 10200>, voltage_excit_val=1.0, voltage_excit_freq=2500.0, ac_excit_wire_mode=<ACExcitWireMode.FOUR_WIRE: 4>, custom_scale_name=u'')

Creates channel(s) that use an LVDT to measure linear position.
```

## **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.LengthUnits]) Specifies the units to use to return linear position measurements from the channel.
- **sensitivity** (Optional[float]) Is the sensitivity of the sensor. This value is in the units you specify with the **sensitivity\_units** input. Refer to the sensor documentation to determine this value.
- sensitivity\_units (Optional[nidaqmx.constants. LVDTSensitivityUnits]) Specifies the units of the sensitivity input.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **voltage\_excit\_val** (Optional[float]) Specifies in volts the amount of excitation supplied to the sensor. Refer to the sensor documentation to determine appropriate excitation values.

- voltage excit freq (Optional [float]) Specifies in hertz the excitation frequency that the sensor requires. Refer to the sensor documentation to determine this value.
- ac\_excit\_wire\_mode (Optional [nidagmx.constants. ACExcitWireMode]) - Is the number of leads on the sensor. Some sensors require you to tie leads together to create a four- or five- wire sensor. Refer to the sensor documentation for more information.
- custom\_scale\_name (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set units to FROM\_CUSTOM\_SCALE.

**Return type** *nidagmx*. *task modules.channels.ai channel.AIChannel* 

```
add_ai_pos_rvdt_chan (physical_channel, name_to_assign_to_channel=u'',
                                                                                     min\ val = -70.0,
                             max_val=70.0, units=<AngleUnits.DEGREES: 10146>, sensitivity=50.0,
                             sensitivity_units=<RVDTSensitivityUnits.M_VPER_VPER_DEGREE:
                             12507>,
                                                voltage_excit_source=<ExcitationSource.INTERNAL:</pre>
                                                                         voltage_excit_freq=2500.0.
                             10200>.
                                             voltage \ excit \ val=1.0,
                             ac excit wire mode=<ACExcitWireMode.FOUR WIRE:
                                                                                        4>.
                                                                                                CUS-
     tom\_scale\_name=u'') Creates channel(s) that use an RVDT to measure angular position.
```

#### **Parameters**

- physical\_channel (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional [float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional [float]) Specifies in units the maximum value you expect to measure.
- units (Optional [nidaqmx.constants.AngleUnits]) Specifies the units to use to return angular position measurements from the channel.
- sensitivity (Optional [float]) Is the sensitivity of the sensor. This value is in the units you specify with the **sensitivity units** input. Refer to the sensor documentation to determine this value.
- sensitivity\_units (Optional[nidaqmx.constants. RVDTSensitivityUnits [] – Specifies the units of the sensitivity input.
- (Optional [nidagmx.constants. voltage excit source ExcitationSource]) – Specifies the source of excitation.
- voltage\_excit\_val (Optional[float]) Specifies in volts the amount of excitation supplied to the sensor. Refer to the sensor documentation to determine appropriate excitation values.
- voltage\_excit\_freq (Optional[float]) Specifies in hertz the excitation frequency that the sensor requires. Refer to the sensor documentation to determine this value.
- ac\_excit\_wire\_mode (Optional[nidaqmx.constants. ACExcitWireMode 1) - Is the number of leads on the sensor. Some sensors

require you to tie leads together to create a four- or five- wire sensor. Refer to the sensor documentation for more information.

• **custom\_scale\_name** (Optional[str]) – Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx. task modules.channels.ai channel.AIChannel* 

add\_ai\_pressure\_bridge\_polynomial\_chan (physical\_channel,

```
name_to_assign_to_channel=u'',
min_val = -100.0,
                           max_val=100.0,
units=<PressureUnits.POUNDS PER SO INCH:
15879>, bridge_config=<BridgeConfiguration.FULL_BRIDGE:
10182>,
age_excit_source=<ExcitationSource.INTERNAL:
10200>,
                     voltage_excit_val=2.5,
nominal bridge resistance=350.0,
forward coeffs=None,
                                       re-
verse coeffs=None,
                                   electri-
cal_units=<BridgeElectricalUnits.M_VOLTS_PER_VOLT:
15897>.
cal_units=<BridgePhysicalUnits.POUNDS_PER_SQ_INCH:
15879>, custom scale name=u'')
```

Creates channel(s) that use a Wheatstone bridge to measure pressure. Use this instance with sensors whose specifications provide a polynomial to convert electrical values to physical values. When you use this scaling type, NI-DAQmx requires coefficients for a polynomial that converts electrical values to physical values (forward), as well as coefficients for a polynomial that converts physical values to electrical values (reverse). If you only know one set of coefficients, use the DAQmx Compute Reverse Polynomial Coefficients function to generate the other set.

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.PressureUnits]) Specifies in which unit to return pressure measurements from the channel.
- bridge\_config (Optional[nidaqmx.constants. BridgeConfiguration]) Specifies information about the bridge configuration and measurement.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) - Specifies information about the bridge configuration and measurement.

- **voltage\_excit\_val** (Optional[float]) Specifies information about the bridge configuration and measurement.
- nominal\_bridge\_resistance (Optional[float]) Specifies information about the bridge configuration and measurement.
- **forward\_coeffs** (Optional[List[float]]) Specifies how to scale electrical values from the sensor to physical units.
- reverse\_coeffs (Optional[List[float]]) Specifies how to scale electrical values from the sensor to physical units.
- electrical\_units (Optional[nidaqmx.constants. BridgeElectricalUnits]) Specifies how to scale electrical values from the sensor to physical units.
- physical\_units (Optional[nidaqmx.constants. BridgePhysicalUnits]) Specifies how to scale electrical values from the sensor to physical units.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM CUSTOM SCALE**.

Return type nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_ai_pressure_bridge_table_chan (physical_channel, name_to_assign_to_channel=u'',
                                            min\ val = -100.0,
                                                                              max val=100.0,
                                            units=<PressureUnits.POUNDS_PER_SQ_INCH:
                                            15879>, bridge_config=<BridgeConfiguration.FULL_BRIDGE:
                                            10182>, voltage_excit_source=<ExcitationSource.INTERNAL:
                                            10200>,
                                                                        voltage \ excit \ val=2.5,
                                            nominal_bridge_resistance=350.0,
                                            electrical vals=None,
                                                                                      electri-
                                            cal_units=<BridgeElectricalUnits.M_VOLTS_PER_VOLT:
                                            15897>,
                                                            physical_vals=None,
                                                                                       physi-
                                            cal_units=<BridgePhysicalUnits.POUNDS_PER_SQ_INCH:
                                            15879>, custom scale name=u'')
```

Creates channel(s) that use a Wheatstone bridge to measure pressure. Use this instance with sensors whose specifications provide a table of electrical values and the corresponding physical values. When you use this scaling type, NI-DAQmx performs linear scaling between each pair of electrical and physical values. The input limits specified with **min\_val** and **max\_val** must fall within the smallest and largest physical values. For any data outside those endpoints, NI-DAQmx coerces that data to the endpoints.

## **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.

- units (Optional[nidaqmx.constants.PressureUnits]) Specifies in which unit to return pressure measurements from the channel.
- bridge\_config (Optional[nidaqmx.constants. BridgeConfiguration]) Specifies information about the bridge configuration and measurement.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies information about the bridge configuration and measurement.
- **voltage\_excit\_val** (Optional[float]) Specifies information about the bridge configuration and measurement.
- nominal\_bridge\_resistance (Optional[float]) Specifies information about the bridge configuration and measurement.
- **electrical\_vals** (Optional[List[float]]) Specifies how to scale electrical values from the sensor to physical units.
- electrical\_units (Optional[nidaqmx.constants. BridgeElectricalUnits]) Specifies how to scale electrical values from the sensor to physical units.
- physical\_vals (Optional[List[float]]) Specifies how to scale electrical values from the sensor to physical units.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Return type** nidagmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_ai_pressure_bridge_two_point_lin_chan (physical_channel,
```

```
name to assign to channel=u'',
min\ val = -100.0,
                        max \ val = 100.0,
units=<PressureUnits.POUNDS PER SQ INCH:
15879>, bridge_config=<BridgeConfiguration.FULL_BRIDGE:
10182>,
age_excit_source=<ExcitationSource.INTERNAL:</pre>
10200>,
                  voltage excit val=2.5,
nominal bridge resistance=350.0,
first electrical val=0.0,
                                   sec-
ond_electrical_val=2.0,
                                electri-
cal_units=<BridgeElectricalUnits.M_VOLTS_PER_VOLT:
15897>,
                 first_physical_val=0.0,
second_physical_val=100.0,
                                 physi-
cal_units=<BridgePhysicalUnits.POUNDS_PER_SQ_INCH:
15879>, custom_scale_name=u'')
```

Creates channel(s) that use a Wheatstone bridge to measure pressure. Use this instance with sensors whose specifications do not provide a polynomial for scaling or a table of electrical and physical values. When you use this scaling type, NI-DAQmx uses two points of electrical and physical values to calculate the slope and y-intercept of a linear equation and uses that equation to scale electrical values to physical values.

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.PressureUnits]) Specifies in which unit to return pressure measurements from the channel.
- bridge\_config (Optional[nidaqmx.constants. BridgeConfiguration]) Specifies information about the bridge configuration and measurement.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies information about the bridge configuration and measurement.
- **voltage\_excit\_val** (Optional[float]) Specifies information about the bridge configuration and measurement.
- nominal\_bridge\_resistance (Optional[float]) Specifies information about the bridge configuration and measurement.
- **first\_electrical\_val** (Optional[float]) Specifies how to scale electrical values from the sensor to physical units.
- **second\_electrical\_val** (Optional[float]) Specifies how to scale electrical values from the sensor to physical units.
- electrical\_units (Optional[nidaqmx.constants. BridgeElectricalUnits]) Specifies how to scale electrical values from the sensor to physical units.
- **first\_physical\_val** (Optional[float]) Specifies how to scale electrical values from the sensor to physical units.
- **second\_physical\_val** (Optional[float]) Specifies how to scale electrical values from the sensor to physical units.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Return type** nidagmx.\_task\_modules.channels.ai\_channel.AIChannel

Creates channel(s) to measure resistance.

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.ResistanceUnits]) Specifies the units to use to return resistance measurements.
- resistance\_config (Optional[nidaqmx.constants. ResistanceConfiguration]) Specifies the number of wires to use for resistive measurements.
- current\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **current\_excit\_val** (Optional[float]) Specifies in amperes the amount of excitation to supply to the sensor. Refer to the sensor documentation to determine this value.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx.\_task\_modules.channels.ai\_channel.AIChannel* 

```
add_ai_rosette_strain_gage_chan (physical_channel, rosette_type, gage_orientation, rosette_meas_types, name_to_assign_to_channel=u'', min_val=-0.001, max_val=0.001, strain_config=<StrainGageBridgeType.QUARTER_BRIDGE_I: 10271>, voltage_excit_source=<ExcitationSource.INTERNAL: 10200>, voltage_excit_val=2.5, gage_factor=2.0, nominal_gage_resistance=350.0, poisson_ratio=0.3, lead wire resistance=0.0)
```

Creates channels to measure two-dimensional strain using a rosette strain gage.

#### **Parameters**

• **physical\_channel** (str) – Specifies the names of the physical channels to use to create the strain gage virtual channels necessary to calculate the **rosette measurements** channels.

- rosette\_type (nidaqmx.constants.StrainGageRosetteType) Specifies information about the rosette configuration and measurements.
- **gage\_orientation** (*float*) Specifies information about the rosette configuration and measurements.
- **rosette\_meas\_types** (*List[int]*) Specifies information about the rosette configuration and measurements.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx creates a default channel name.
- min\_val (Optional[float]) Specifies the minimum strain you expect to measure. This value applies to each strain gage in the rosette.
- max\_val(Optional[float]) Specifies the maximum strain you expect to measure. This value applies to each strain gage in the rosette.
- **strain\_config** (Optional[nidaqmx.constants. StrainGageBridgeType]) Specifies information about the bridge configuration and measurement.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies information about the bridge configuration and measurement.
- **voltage\_excit\_val** (Optional[float]) Specifies information about the bridge configuration and measurement.
- gage\_factor(Optional[float]) Contains information about the strain gage and measurement.
- nominal\_gage\_resistance (Optional[float]) Contains information about the strain gage and measurement.
- poisson\_ratio (Optional[float]) Contains information about the strain gage and measurement.
- **lead\_wire\_resistance** (Optional[float]) Specifies information about the bridge configuration and measurement.

Return type nidagmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_ai_rtd_chan (physical_channel,
                                             name_to_assign_to_channel=u'',
                                                                                   min_val=0.0,
                     max_val=100.0,
                                                               units=<TemperatureUnits.DEG C:
                     10143>.
                                    rtd type=<RTDType.PT 3750:
                                                                           12481>.
                                                                                          resis-
                     tance config=<ResistanceConfiguration.TWO WIRE:
                                                                               2>.
                                                                                           cur-
                     rent_excit_source=<ExcitationSource.EXTERNAL:</pre>
                                                                           10167>,
                                                                                           cur-
                     rent_excit_val=0.0025, r_0=100.0
     Creates channel(s) that use an RTD to measure temperature.
```

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.

- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TemperatureUnits]) Specifies the units to use to return temperature measurements.
- rtd\_type (Optional[nidaqmx.constants.RTDType]) Specifies the type of RTD connected to the channel.
- resistance\_config
   (Optional[nidaqmx.constants.
   ResistanceConfiguration]) Specifies the number of wires to use for resistive measurements.
- current\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **current\_excit\_val** (Optional[float]) Specifies in amperes the amount of excitation to supply to the sensor. Refer to the sensor documentation to determine this value.
- r\_0 (Optional[float]) Is the sensor resistance in ohms at 0 degrees Celsius. The Callendar-Van Dusen equation requires this value. Refer to the sensor documentation to determine this value.

**Return type** *nidagmx*. *task modules.channels.ai channel.AIChannel* 

```
\label{eq:add_ai_strain_gage_chan} \begin{tabular}{ll} add_ai\_strain\_gage\_chan (physical\_channel, name\_to\_assign\_to\_channel=u'', min\_val=-0.001, max\_val=0.001, units=<StrainUnits.STRAIN: 10299>, strain\_config=<StrainGageBridgeType.FULL\_BRIDGE\_I: 10183>, voltage\_excit\_source=<ExcitationSource.INTERNAL: 10200>, voltage\_excit\_val=2.5, gage\_factor=2.0, initial\_bridge\_voltage=0.0, nominal\_gage\_resistance=350.0, poisson\_ratio=0.3, lead\_wire\_resistance=0.0, custom\_scale\_name=u'') Creates channel(s) to measure strain. \\ \end{tabular}
```

# **Parameters**

- physical\_channel (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.StrainUnits]) Specifies the units to use to return strain measurements.
- strain\_config (Optional[nidaqmx.constants. StrainGageBridgeType]) Specifies information about the bridge configuration and measurement.

- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies information about the bridge configuration and measurement.
- **voltage\_excit\_val** (Optional[float]) Specifies information about the bridge configuration and measurement.
- gage\_factor(Optional[float]) Contains information about the strain gage and measurement.
- initial\_bridge\_voltage (Optional[float]) Specifies information about the bridge configuration and measurement.
- nominal\_gage\_resistance (Optional[float]) Contains information about the strain gage and measurement.
- poisson\_ratio (Optional[float]) Contains information about the strain gage and measurement.
- **lead\_wire\_resistance** (Optional[float]) Specifies information about the bridge configuration and measurement.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Return type** nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

Creates channel(s) that use the built-in sensor of a terminal block or device to measure temperature. On SCXI modules, for example, the built-in sensor could be the CJC sensor.

## **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- units (Optional[nidaqmx.constants.TemperatureUnits]) Specifies the units to use to return temperature measurements.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx. task modules.channels.ai channel.AIChannel* 

Creates channel(s) that use a thermocouple to measure temperature.

### **Parameters**

• **physical\_channel** (str) – Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.

- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TemperatureUnits]) Specifies the units to use to return temperature measurements.
- thermocouple\_type (Optional[nidaqmx.constants. ThermocoupleType]) Specifies the type of thermocouple connected to the channel. Thermocouple types differ in composition and measurement range.
- cjc\_source (Optional[nidaqmx.constants.CJCSource]) Specifies the source of cold-junction compensation.
- cjc\_val (Optional[float]) Specifies in units the temperature of the cold junction if you set cjc\_source to CONSTANT\_VALUE.
- **cjc\_channel** (Optional[str]) Specifies the channel that acquires the temperature of the thermocouple cold-junction if you set **cjc\_source** to **CHANNEL**.

Return type nidagmx. task modules.channels.ai channel.AIChannel

Creates channel(s) that use a thermistor to measure temperature. Use this instance when the thermistor requires current excitation.

## **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TemperatureUnits]) Specifies the units to use to return temperature measurements.
- resistance\_config
   (Optional[nidaqmx.constants.
   ResistanceConfiguration]) Specifies the number of wires to use for resistive measurements.

- current\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **current\_excit\_val** (Optional[float]) Specifies in amperes the amount of excitation to supply to the sensor. Refer to the sensor documentation to determine this value.
- a (Optional[float]) Contains the constants for the Steinhart-Hart thermistor equation. Refer to the sensor documentation to determine values for these constants.
- **b** (Optional[float]) Contains the constants for the Steinhart-Hart thermistor equation. Refer to the sensor documentation to determine values for these constants.
- c (Optional[float]) Contains the constants for the Steinhart-Hart thermistor equation. Refer to the sensor documentation to determine values for these constants.

Return type nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

Creates channel(s) that use a thermistor to measure temperature. Use this instance when the thermistor requires voltage excitation.

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TemperatureUnits]) Specifies the units to use to return temperature measurements.
- resistance\_config (Optional[nidaqmx.constants. ResistanceConfiguration]) Specifies the number of wires to use for resistive measurements.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **voltage\_excit\_val** (Optional[float]) Specifies in volts the amount of excitation supplied to the sensor. Refer to the sensor documentation to determine appropriate excitation values.
- a (Optional[float]) Contains the constants for the Steinhart-Hart thermistor equation. Refer to the sensor documentation to determine values for these constants.

- **b** (Optional [float]) Contains the constants for the Steinhart-Hart thermistor equation. Refer to the sensor documentation to determine values for these constants.
- c (Optional[float]) Contains the constants for the Steinhart-Hart thermistor equation. Refer to the sensor documentation to determine values for these constants.
- **r\_1** (Optional [float]) Specifies in ohms the value of the reference resistor.

**Return type** *nidagmx*. *task modules.channels.ai channel.AIChannel* 

```
add_ai_torque_bridge_polynomial_chan (physical_channel,
```

```
name_to_assign_to_channel=u'',
min_val=-100.0, max_val=100.0,
units=<TorqueUnits.INCH_POUNDS: 15883>,
bridge_config=<BridgeConfiguration.FULL_BRIDGE:
10182>, voltage_excit_source=<ExcitationSource.INTERNAL:
10200>, voltage_excit_val=2.5, nom-
inal_bridge_resistance=350.0, for-
ward_coeffs=None, reverse_coeffs=None, electri-
cal_units=<BridgeElectricalUnits.M_VOLTS_PER_VOLT:
15897>, physical_units=<BridgePhysicalUnits.INCH_POUNDS:
15883>, custom_scale_name=u'')
```

Creates channel(s) that use a Wheatstone bridge to measure torque. Use this instance with sensors whose specifications provide a polynomial to convert electrical values to physical values. When you use this scaling type, NI-DAQmx requires coefficients for a polynomial that converts electrical values to physical values (forward), as well as coefficients for a polynomial that converts physical values to electrical values (reverse). If you only know one set of coefficients, use the DAQmx Compute Reverse Polynomial Coefficients function to generate the other set.

## Parameters

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TorqueUnits]) Specifies in which unit to return torque measurements from the channel.
- bridge\_config (Optional[nidaqmx.constants. BridgeConfiguration]) Specifies information about the bridge configuration and measurement.
- voltage\_excit\_source (Optional[nidagmx.constants. ExcitationSource]) Specifies information about the bridge configuration and measurement
- voltage\_excit\_val (Optional[float]) Specifies information about the bridge configuration and measurement.

- nominal\_bridge\_resistance (Optional[float]) Specifies information about the bridge configuration and measurement.
- **forward\_coeffs** (Optional[List[float]]) Specifies how to scale electrical values from the sensor to physical units.
- reverse\_coeffs (Optional[List[float]]) Specifies how to scale electrical values from the sensor to physical units.
- electrical\_units (Optional[nidaqmx.constants. BridgeElectricalUnits]) Specifies how to scale electrical values from the sensor to physical units.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

Return type nidagmx. task modules.channels.ai channel.AIChannel

```
add ai torque bridge table chan (physical channel,
                                                              name to assign to channel=u'',
                                         min_val = -100.0,
                                                                              max_val=100.0,
                                         units=<TorqueUnits.INCH POUNDS:
                                                                                    15883>.
                                         bridge config=<BridgeConfiguration.FULL BRIDGE:
                                         10182>, voltage excit source=<ExcitationSource.INTERNAL:
                                         10200>,
                                                                        voltage_excit_val=2.5,
                                         nominal_bridge_resistance=350.0,
                                         electrical_vals=None,
                                                                                      electri-
                                         cal_units=<BridgeElectricalUnits.M_VOLTS_PER_VOLT:
                                                          physical_vals=None,
                                          15897>,
                                                                                       physi-
                                         cal units=<BridgePhysicalUnits.INCH POUNDS:
                                          15883>, custom_scale_name=u'')
```

Creates channel(s) that use a Wheatstone bridge to measure torque. Use this instance with sensors whose specifications provide a table of electrical values and the corresponding physical values. When you use this scaling type, NI-DAQmx performs linear scaling between each pair of electrical and physical values. The input limits specified with **min\_val** and **max\_val** must fall within the smallest and largest physical values. For any data outside those endpoints, NI-DAQmx coerces that data to the endpoints.

#### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TorqueUnits]) Specifies in which unit to return torque measurements from the channel.

- bridge\_config (Optional[nidaqmx.constants. BridgeConfiguration]) Specifies information about the bridge configuration and measurement.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies information about the bridge configuration and measurement.
- **voltage\_excit\_val** (Optional[float]) Specifies information about the bridge configuration and measurement.
- nominal\_bridge\_resistance (Optional[float]) Specifies information about the bridge configuration and measurement.
- **electrical\_vals** (Optional[List[float]]) Specifies how to scale electrical values from the sensor to physical units.
- electrical\_units (Optional[nidaqmx.constants. BridgeElectricalUnits]) Specifies how to scale electrical values from the sensor to physical units.
- **physical\_vals** (Optional[List[float]]) Specifies how to scale electrical values from the sensor to physical units.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

Return type nidagmx. task modules.channels.ai channel.AIChannel

## add\_ai\_torque\_bridge\_two\_point\_lin\_chan (physical\_channel,

```
name_to_assign_to_channel=u'',
min\ val = -100.0,
                          max_val = 100.0,
units=<TorqueUnits.INCH_POUNDS:
15883>, bridge config=<BridgeConfiguration.FULL BRIDGE:
10182>.
age excit source=<ExcitationSource.INTERNAL:
                    voltage_excit_val=2.5,
10200>,
nominal_bridge_resistance=350.0,
first electrical val=0.0,
                                     sec-
ond electrical val=2.0,
                                  electri-
cal units=<BridgeElectricalUnits.M VOLTS PER VOLT:
15897>,
            first_physical_val=0.0,
                                     sec-
ond_physical_val=100.0,
                                   physi-
cal_units=<BridgePhysicalUnits.INCH_POUNDS:
15883>, custom_scale_name=u'')
```

Creates channel(s) that use a Wheatstone bridge to measure torque. Use this instance with sensors whose specifications do not provide a polynomial for scaling or a table of electrical and physical values. When you use this scaling type, NI-DAQmx uses two points of electrical and physical values to calculate the slope and y-intercept of a linear equation and uses that equation to scale electrical values to physical values.

## **Parameters**

• **physical\_channel** (str) – Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels

on devices and modules installed in the system.

- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TorqueUnits]) Specifies in which unit to return torque measurements from the channel.
- bridge\_config (Optional[nidaqmx.constants. BridgeConfiguration]) Specifies information about the bridge configuration and measurement.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies information about the bridge configuration and measurement.
- **voltage\_excit\_val** (Optional[float]) Specifies information about the bridge configuration and measurement.
- nominal\_bridge\_resistance (Optional[float]) Specifies information about the bridge configuration and measurement.
- **first\_electrical\_val** (Optional[float]) Specifies how to scale electrical values from the sensor to physical units.
- **second\_electrical\_val** (Optional[float]) Specifies how to scale electrical values from the sensor to physical units.
- electrical\_units (Optional[nidaqmx.constants. BridgeElectricalUnits]) Specifies how to scale electrical values from the sensor to physical units.
- **first\_physical\_val** (Optional[float]) Specifies how to scale electrical values from the sensor to physical units.
- **second\_physical\_val** (Optional[float]) Specifies how to scale electrical values from the sensor to physical units.
- physical\_units (Optional[nidaqmx.constants. BridgePhysicalUnits]) Specifies how to scale electrical values from the sensor to physical units.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx*.\_*task*\_*modules.channels.ai*\_*channel.AIChannel* 

```
add_ai_velocity_iepe_chan (physical_channel, name_to_assign_to_channel=u'', terminal_config=<TerminalConfiguration.DEFAULT:
-1>, min_val=-50.0, max_val=50.0, units=<VelocityUnits.INCHES_PER_SECOND:
15960>, sensitivity=100.0, sensitiv-ity_units=<VelocityIEPESensorSensitivityUnits.M_VOLTS_PER_INCH_PER_SECOND:
15964>, current_excit_source=<ExcitationSource.INTERNAL:
10200>, current_excit_val=0.002, custom_scale_name=u'')
Creates channel(s) that use an IEPE velocity sensor to measure velocity.
```

#### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config (Optional[nidaqmx.constants. TerminalConfiguration]) Specifies the input terminal configuration for the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.VelocityUnits]) Specifies in which unit to return velocity measurements from the channel.
- **sensitivity** (Optional[float]) Is the sensitivity of the sensor. This value is in the units you specify with the **sensitivity\_units** input. Refer to the sensor documentation to determine this value.
- sensitivity\_units (Optional[nidaqmx.constants. VelocityIEPESensorSensitivityUnits]) Specifies the units of the sensitivity input.
- current\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **current\_excit\_val** (Optional[float]) Specifies in amperes the amount of excitation to supply to the sensor. Refer to the sensor documentation to determine this value.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

Creates channel(s) to measure voltage. If the measurement requires the use of internal excitation or you need excitation to scale the voltage, use the AI Custom Voltage with Excitation instance of this function.

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config (Optional[nidaqmx.constants. TerminalConfiguration]) Specifies the input terminal configuration for the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.VoltageUnits]) Specifies the units to use to return voltage measurements.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx*. *task modules.channels.ai channel.AIChannel* 

Creates channel(s) to measure voltage. Use this instance for custom sensors that require excitation. You can use the excitation to scale the measurement.

## **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config (Optional[nidaqmx.constants. TerminalConfiguration]) Specifies the input terminal configuration for the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.

- units (Optional[nidaqmx.constants.VoltageUnits]) Specifies the units to use to return voltage measurements.
- bridge\_config (Optional[nidaqmx.constants. BridgeConfiguration]) Specifies what type of Wheatstone bridge the sensor is.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) - Specifies the source of excitation.
- **voltage\_excit\_val** (Optional[float]) Specifies in volts the amount of excitation supplied to the sensor. Refer to the sensor documentation to determine appropriate excitation values.
- use\_excit\_for\_scaling (Optional[bool]) Specifies if NI- DAQmx divides the measurement by the excitation. You should typically set use\_excit\_for\_scaling to True for ratiometric transducers. If you set use\_excit\_for\_scaling to True, set max\_val and min\_val to reflect the scaling.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Return type** nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_ai_voltage_rms_chan (physical_channel, name_to_assign_to_channel=u'', termi-
nal_config=<TerminalConfiguration.DEFAULT: -1>, min_val=-
5.0, max_val=5.0, units=<VoltageUnits.VOLTS: 10348>, cus-
tom scale name=u'')
```

Creates channel(s) to measure voltage RMS, the average (mean) power of the acquired voltage.

## **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config
   TerminalConfiguration] Specifies the input terminal configuration for
  the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.VoltageUnits]) Specifies the units to use to return voltage measurements.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

Returns Indicates the newly created channel object.

**Return type** *nidaqmx*.\_*task*\_*modules.channels.ai*\_*channel.AIChannel* 

```
add_teds_ai_accel_chan (physical_channel, name_to_assign_to_channel=u'', termi-
nal_config=<TerminalConfiguration.DEFAULT: -1>, min_val=-
5.0, max_val=5.0, units=<AccelUnits.G: 10186>, cur-
rent_excit_source=<ExcitationSource.INTERNAL: 10200>, cur-
rent_excit_val=0.004, custom_scale_name=u'')
```

Creates channel(s) that use an accelerometer to measure acceleration. You must configure the physical channel(s) with TEDS information to use this function.

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config
   TerminalConfiguration] Specifies the input terminal configuration for
  the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional [nidaqmx.constants.AccelUnits]) Specifies the units to use to return acceleration measurements from the channel.
- current\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **current\_excit\_val** (Optional[float]) Specifies in amperes the amount of excitation to supply to the sensor. Refer to the sensor documentation to determine this value.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

Returns Indicates the newly created channel object.

**Return type** *nidagmx*. *task modules.channels.ai channel.AIChannel* 

```
add_teds_ai_bridge_chan (physical_channel, name_to_assign_to_channel=u'', min_val=-
0.002, max_val=0.002, units=<TEDSUnits.FROM_TEDS: 12516>,
voltage_excit_source=<ExcitationSource.INTERNAL: 10200>,
voltage_excit_val=2.5, custom_scale_name=u'')
```

Creates channel(s) that measure a Wheatstone bridge. You must configure the physical channel(s) with TEDS information to use this function. Use this instance with bridge-based sensors that measure phenomena other than strain, force, pressure, or torque, or that scale data to physical units NI-DAQmx does not support.

### **Parameters**

• **physical\_channel** (str) – Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.

- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TEDSUnits]) Specifies in which unit to return measurements from the channel.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) - Specifies the source of excitation.
- **voltage\_excit\_val** (Optional[float]) Specifies in volts the amount of excitation supplied to the sensor. Refer to the sensor documentation to determine appropriate excitation values.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM CUSTOM SCALE**.

Return type nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_teds_ai_current_chan (physical_channel, name_to_assign_to_channel=u'', termi-
nal_config=<TerminalConfiguration.DEFAULT: -1>, min_val=-
0.01, max_val=0.01, units=<TEDSUnits.FROM_TEDS: 12516>,
shunt_resistor_loc=<CurrentShuntResistorLocation.LET_DRIVER_CHOOSE:
-1>, ext_shunt_resistor_val=249.0, custom_scale_name=u'')
```

Creates channel(s) to measure current. You must configure the physical channel(s) with TEDS information to use this function.

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config (Optional[nidaqmx.constants.
   TerminalConfiguration]) Specifies the input terminal configuration for the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TEDSUnits]) Specifies the units to use to return measurements.
- **shunt\_resistor\_loc** (Optional[nidaqmx.constants. CurrentShuntResistorLocation]) Specifies the location of the shunt resistor. For devices with built-in shunt resistors, specify the location as **INTERNAL**. For devices that do not have built-in shunt resistors, you must attach an external one, set

this input to **EXTERNAL** and use the **ext\_shunt\_resistor\_val** input to specify the value of the resistor.

- ext\_shunt\_resistor\_val (Optional[float]) Specifies in ohms the resistance of an external shunt resistor.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM CUSTOM SCALE**.

Returns Indicates the newly created channel object.

**Return type** nidagmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_teds_ai_force_bridge_chan (physical_channel, name_to_assign_to_channel=u'', min_val=-100.0, max_val=100.0, units=<ForceUnits.POUNDS: 15876>, volt-age_excit_source=<ExcitationSource.INTERNAL: 10200>, voltage_excit_val=2.5, custom_scale_name=u'')
```

Creates channel(s) that use a Wheatstone bridge to measure force or load. You must configure the physical channel(s) with TEDS information to use this function. NI-DAQmx scales electrical values to physical values according to that TEDS information.

#### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.ForceUnits]) Specifies in which unit to return force measurements from the channel.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) - Specifies the source of excitation.
- **voltage\_excit\_val** (Optional[float]) Specifies in volts the amount of excitation supplied to the sensor. Refer to the sensor documentation to determine appropriate excitation values.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_teds_ai_force_iepe_chan (physical_channel, name_to_assign_to_channel=u'', terminal_config=<TerminalConfiguration.DEFAULT:
-1>, min_val=-2000.0, max_val=2000.0, units=<ForceUnits.NEWTONS: 15875>, current_excit_source=<ExcitationSource.INTERNAL: 10200>, current_excit_val=0.001, custom_scale_name=u'')
```

Creates channel(s) that use an IEPE force sensor to measure force or load. You must configure the physical channel(s) with TEDS information to use this function.

#### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config (Optional[nidaqmx.constants. TerminalConfiguration]) Specifies the input terminal configuration for the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional [nidaqmx.constants.ForceUnits]) Specifies in which unit to return force measurements from the channel.
- current\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **current\_excit\_val** (Optional[float]) Specifies in amperes the amount of excitation to supply to the sensor. Refer to the sensor documentation to determine this value.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

Return type nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_teds_ai_microphone_chan (physical_channel, name_to_assign_to_channel=u'', terminal_config=<TerminalConfiguration.DEFAULT:
-1>, units=<SoundPressureUnits.PA:
10081>, max_snd_press_level=100.0, cur-
rent_excit_source=<ExcitationSource.INTERNAL: 10200>, current excit_val=0.004, custom_scale_name=u'')
```

Creates channel(s) that use a microphone to measure sound pressure. You must configure the physical channel(s) with TEDS information to use this function.

## **Parameters**

- physical\_channel (str) Specifies the names of the physical channels to use to create virtual channels. You must use physical channels that you configured with TEDS information. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.

- terminal\_config (Optional[nidaqmx.constants. TerminalConfiguration]) Specifies the input terminal configuration for the channel.
- units (Optional[nidaqmx.constants.SoundPressureUnits]) Specifies the units to use to return sound pressure measurements.
- max\_snd\_press\_level (Optional[float]) Is the maximum instantaneous sound pressure level you expect to measure. This value is in decibels, referenced to 20 micropascals.
- current\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **current\_excit\_val** (Optional[float]) Specifies in amperes the amount of excitation to supply to the sensor. Refer to the sensor documentation to determine this value.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM CUSTOM SCALE**.

Return type nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_teds_ai_pos_lvdt_chan (physical_channel, name_to_assign_to_channel=u'', min_val=-0.1, max_val=0.1, units=<LengthUnits.METERS: 10219>, voltage_excit_source=<ExcitationSource.INTERNAL: 10200>, voltage_excit_val=1.0, voltage_excit_freq=2500.0, ac_excit_wire_mode=<ACExcitWireMode.FOUR_WIRE: 4>, custom scale name=u'')
```

Creates channel(s) that use an LVDT to measure linear position. You must configure the physical channel(s) with TEDS information to use this function.

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.LengthUnits]) Specifies the units to use to return linear position measurements from the channel.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **voltage\_excit\_val** (Optional[float]) Specifies in volts the amount of excitation supplied to the sensor. Refer to the sensor documentation to determine appropriate excitation values.
- **voltage\_excit\_freq** (Optional[float]) Specifies in hertz the excitation frequency that the sensor requires. Refer to the sensor documentation to determine this value.

- ac\_excit\_wire\_mode (Optional[nidaqmx.constants. ACExcitWireMode]) Is the number of leads on the sensor. Some sensors require you to tie leads together to create a four- or five- wire sensor. Refer to the sensor documentation for more information.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM CUSTOM SCALE**.

**Return type** nidagmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_teds_ai_pos_rvdt_chan (physical_channel, name_to_assign_to_channel=u'', min_val=-70.0, max_val=70.0, units=<AngleUnits.DEGREES:
10146>, voltage_excit_source=<ExcitationSource.INTERNAL:
10200>, voltage_excit_val=1.0, voltage_excit_freq=2500.0,
ac_excit_wire_mode=<ACExcitWireMode.FOUR_WIRE: 4>,
custom_scale_name=u'')
```

Creates channel(s) that use an RVDT to measure angular position. You must configure the physical channel(s) with TEDS information to use this function.

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.AngleUnits]) Specifies the units to use to return angular position measurements from the channel.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **voltage\_excit\_val** (Optional[float]) Specifies in volts the amount of excitation supplied to the sensor. Refer to the sensor documentation to determine appropriate excitation values.
- **voltage\_excit\_freq** (Optional[float]) Specifies in hertz the excitation frequency that the sensor requires. Refer to the sensor documentation to determine this value.
- ac\_excit\_wire\_mode (Optional[nidaqmx.constants. ACExcitWireMode]) Is the number of leads on the sensor. Some sensors require you to tie leads together to create a four- or five- wire sensor. Refer to the sensor documentation for more information.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

Creates channel(s) that use a Wheatstone bridge to measure pressure. You must configure the physical channel(s) with TEDS information to use this function. NI-DAQmx scales electrical values to physical values according to that TEDS information.

#### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.PressureUnits]) Specifies in which unit to return pressure measurements from the channel.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) - Specifies the source of excitation.
- **voltage\_excit\_val** (Optional[float]) Specifies in volts the amount of excitation supplied to the sensor. Refer to the sensor documentation to determine appropriate excitation values.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

Return type nidagmx. task modules.channels.ai channel.AIChannel

Creates channel(s) to measure resistance. You must configure the physical channel(s) with TEDS information to use this function.

## **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.

- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TEDSUnits]) Specifies the units to use to return measurements.
- resistance\_config
   (Optional[nidaqmx.constants.
   ResistanceConfiguration]) Specifies the number of wires to use for resistive measurements.
- current\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **current\_excit\_val** (Optional[float]) Specifies in amperes the amount of excitation to supply to the sensor. Refer to the sensor documentation to determine this value.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM CUSTOM SCALE**.

**Return type** nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

Creates channel(s) that use an RTD to measure temperature. You must configure the physical channel(s) with TEDS information to use this function.

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TemperatureUnits]) Specifies the units to use to return temperature measurements.
- resistance\_config
   ResistanceConfiguration] Specifies the number of wires to use for resistive measurements.
- current\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.

• **current\_excit\_val** (Optional[float]) - Specifies in amperes the amount of excitation to supply to the sensor. Refer to the sensor documentation to determine this value.

**Returns** Indicates the newly created channel object.

**Return type** nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

Creates channel(s) to measure strain. You must configure the physical channel(s) with TEDS information to use this function.

#### **Parameters**

- physical\_channel (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.StrainUnits]) Specifies the units to use to return strain measurements.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies information about the bridge configuration and measurement.
- **voltage\_excit\_val** (Optional[float]) Specifies information about the bridge configuration and measurement.
- initial\_bridge\_voltage (Optional[float]) Specifies information about the bridge configuration and measurement.
- **lead\_wire\_resistance** (Optional[float]) Specifies information about the bridge configuration and measurement.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel

Creates channel(s) that use a thermocouple to measure temperature. You must configure the physical channel(s) with TEDS information to use this function.

### **Parameters**

- **physical\_channel** (*str*) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TemperatureUnits]) Specifies the units to use to return temperature measurements.
- cjc\_source (Optional[nidaqmx.constants.CJCSource]) Specifies the source of cold-junction compensation.
- cjc\_val (Optional[float]) Specifies in units the temperature of the cold junction if you set cjc\_source to CONSTANT\_VALUE.
- **cjc\_channel** (Optional[str]) Specifies the channel that acquires the temperature of the thermocouple cold-junction if you set **cjc\_source** to **CHANNEL**.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx. task modules.channels.ai channel.AIChannel* 

```
add_teds_ai_thrmstr_chan_iex (physical_channel, name_to_assign_to_channel=u'', min_val=0.0, max_val=100.0, units=<TemperatureUnits.DEG_C: 10143>, resistance_config=<ResistanceConfiguration.FOUR_WIRE:

4>, current_excit_source=<ExcitationSource.EXTERNAL: 10167>, current_excit_val=0.00015)
```

Creates channel(s) that use a thermistor to measure temperature. Use this instance when the thermistor requires current excitation. You must configure the physical channel(s) with TEDS information to use this function.

## **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TemperatureUnits]) Specifies the units to use to return temperature measurements.

- resistance\_config
   (Optional[nidaqmx.constants.
   ResistanceConfiguration]) Specifies the number of wires to use for
   resistive measurements.
- current\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **current\_excit\_val** (Optional[float]) Specifies in amperes the amount of excitation to supply to the sensor. Refer to the sensor documentation to determine this value.

**Return type** *nidagmx*.\_*task*\_*modules.channels.ai*\_*channel.AIChannel* 

```
add_teds_ai_thrmstr_chan_vex (physical_channel, name_to_assign_to_channel=u'', min_val=0.0, max_val=100.0, units=<TemperatureUnits.DEG_C: 10143>, resistance_config=<ResistanceConfiguration.FOUR_WIRE:
4>, voltage_excit_source=<ExcitationSource.EXTERNAL: 10167>, voltage_excit_val=2.5, r_1=5000.0)
```

Creates channel(s) that use a thermistor to measure temperature. Use this instance when the thermistor requires voltage excitation. You must configure the physical channel(s) with TEDS information to use this function.

#### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TemperatureUnits]) Specifies the units to use to return temperature measurements.
- resistance\_config (Optional[nidaqmx.constants. ResistanceConfiguration]) Specifies the number of wires to use for resistive measurements.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) Specifies the source of excitation.
- **voltage\_excit\_val** (Optional[float]) Specifies in volts the amount of excitation supplied to the sensor. Refer to the sensor documentation to determine appropriate excitation values.
- r\_1 (Optional[float]) Specifies in ohms the value of the reference resistor.

**Returns** Indicates the newly created channel object.

**Return type** nidagmx.\_task\_modules.channels.ai\_channel.AIChannel

```
add_teds_ai_torque_bridge_chan (physical_channel, name_to_assign_to_channel=u'', min_val=-100.0, max_val=100.0, units=<TorqueUnits.INCH_POUNDS: 15883>, voltage_excit_source=<ExcitationSource.INTERNAL: 10200>, voltage_excit_val=2.5, custom_scale_name=u'')
```

Creates channel(s) that use a Wheatstone bridge to measure torque. You must configure the physical channel(s) with TEDS information to use this function. NI-DAQmx scales electrical values to physical values according to that TEDS information.

#### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TorqueUnits]) Specifies in which unit to return torque measurements from the channel.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) - Specifies the source of excitation.
- **voltage\_excit\_val** (Optional[float]) Specifies in volts the amount of excitation supplied to the sensor. Refer to the sensor documentation to determine appropriate excitation values.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx*.\_*task*\_*modules.channels.ai*\_*channel.AIChannel* 

Creates channel(s) to measure voltage. You must configure the physical channel(s) with TEDS information to use this function. If the measurement requires the use of internal excitation or you need excitation to scale the voltage, use the TEDS AI Custom Voltage with Excitation instance of this function.

# **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.

- terminal\_config (Optional[nidaqmx.constants. TerminalConfiguration]) Specifies the input terminal configuration for the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TEDSUnits]) Specifies the units to use to return measurements.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Return type** *nidagmx*.\_task\_modules.channels.ai\_channel.AIChannel

Creates channel(s) to measure voltage. Use this instance for custom sensors that require excitation. You can use the excitation to scale the measurement. You must configure the physical channel(s) with TEDS information to use this function.

## **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- terminal\_config
   TerminalConfiguration]

   Specifies the input terminal configuration for the channel.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TEDSUnits]) Specifies the units to use to return measurements.
- voltage\_excit\_source (Optional[nidaqmx.constants. ExcitationSource]) - Specifies the source of excitation.
- **voltage\_excit\_val** (Optional[float]) Specifies in volts the amount of excitation supplied to the sensor. Refer to the sensor documentation to determine appropriate excitation values.

• **custom\_scale\_name** (Optional[str]) – Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx*.\_*task*\_*modules.channels.ai*\_*channel.AIChannel* 

#### all

nidaqmx.\_task\_modules.channels.channel.Channel-Specifies a channel object that represents the entire list of virtual channels on this channel collection.

### channel names

List[str] – Specifies the entire list of virtual channels on this channel collection.

**count** (*value*)  $\rightarrow$  integer – return number of occurrences of value

**index** (*value*)  $\rightarrow$  integer – return first index of value.

Raises ValueError if the value is not present.

## nidagmx.task.ao channel collection

```
class nidaqmx._task_modules.ao_channel_collection.AOChannelCollection (task_handle)
    Bases: nidaqmx._task_modules.channel_collection.ChannelCollection
```

Contains the collection of analog output channels for a DAQmx Task.

```
add_ao_current_chan (physical_channel, name_to_assign_to_channel=u'', min_val=0.0, max_val=0.02, units=<CurrentUnits.AMPS: 10342>, custom_scale_name=u'')

Creates channel(s) to generate current.
```

## **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.CurrentUnits]) Specifies the units to use to generate current.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM CUSTOM SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** nidaqmx.\_task\_modules.channels.ao\_channel.AOChannel

Creates a channel for continually generating a waveform on the selected physical channel.

### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- type (Optional [nidaqmx.constants.FuncGenType]) Specifies the kind of waveform to generate.
- **freq** (Optional [float]) Is the frequency of the waveform to generate in hertz.
- **amplitude** (Optional[float]) Is the zero-to-peak amplitude of the waveform to generate in volts. Zero and negative values are valid.
- **offset** (Optional[float]) Is the voltage offset of the waveform to generate.

**Returns** Indicates the newly created channel object.

**Return type** nidaqmx.\_task\_modules.channels.ao\_channel.AOChannel

#### **Parameters**

- **physical\_channel** (str) Specifies the names of the physical channels to use to create virtual channels. The DAQmx physical channel constant lists all physical channels on devices and modules installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to generate.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to generate.
- units (Optional[nidaqmx.constants.VoltageUnits]) Specifies the units to use to generate voltage.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** nidaqmx.\_task\_modules.channels.ao\_channel.AOChannel

## all

nidaqmx.\_task\_modules.channels.channel.Channel-Specifies a channel object that represents the entire list of virtual channels on this channel collection.

## channel\_names

*List[str]* – Specifies the entire list of virtual channels on this channel collection.

**count** (value)  $\rightarrow$  integer – return number of occurrences of value

index (value) → integer – return first index of value.
Raises ValueError if the value is not present.

## nidagmx.task.ci channel collection

Contains the collection of counter input channels for a DAQmx Task.

```
add_ci_ang_encoder_chan (counter, name_to_assign_to_channel=u'', decod-ing_type=<EncoderType.X_4: 10092>, zidx_enable=False, zidx_val=0, zidx_phase=<EncoderZIndexPhase.AHIGH_BHIGH: 10040>, units=<AngleUnits.DEGREES: 10146>, pulses_per_rev=24, initial_angle=0.0, custom_scale_name=u'')
```

Creates a channel that uses an angular encoder to measure angular position. With the exception of devices that support multi-counter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signals to the default input terminals of the counter unless you select different input terminals.

#### **Parameters**

- **counter** (str) Specifies the name of the counter to use to create the virtual channel. The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- decoding\_type (Optional[nidaqmx.constants.EncoderType]) Specifies how to count and interpret the pulses the encoder generates on signal A and signal B. X\_1, X\_2, and X\_4 are valid for quadrature encoders only. TWO\_PULSE\_COUNTING is valid only for two-pulse encoders.
- **zidx\_enable** (Optional[bool]) Specifies whether to use Z indexing for the channel.
- zidx\_val (Optional[float]) Specifies in units the value to which to reset the measurement when signal Z is high and signal A and signal B are at the states you specify with zidx\_phase.
- zidx\_phase (Optional[nidaqmx.constants.EncoderZIndexPhase]) Specifies the states at which signal A and signal B must be while signal Z is high for NI-DAQmx to reset the measurement. If signal Z is never high while signal A and signal B are high, for example, you must choose a phase other than A\_HIGH\_B\_HIGH.
- units (Optional[nidaqmx.constants.AngleUnits]) Specifies the units to use to return angular position measurements from the channel.
- **pulses\_per\_rev** (Optional[int])—Is the number of pulses the encoder generates per revolution. This value is the number of pulses on either signal A or signal B, not the total number of pulses on both signal A and signal B.
- initial\_angle (Optional[float]) Is the starting angle of the encoder. This value is in the units you specify with the units input.

• **custom\_scale\_name** (Optional[str]) – Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM CUSTOM SCALE**.

**Returns** Indicates the newly created channel object.

Return type nidaqmx.\_task\_modules.channels.ci\_channel.CIChannel

Creates a channel to measure the angular velocity of a digital signal. With the exception of devices that support multi-counter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the default input terminal of the counter unless you select a different input terminal.

#### **Parameters**

- **counter** (str) Specifies the name of the counter to use to create the virtual channel. The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- decoding\_type (Optional[nidaqmx.constants.EncoderType]) Specifies how to count and interpret the pulses the encoder generates on signal A and signal B. X\_1, X\_2, and X\_4 are valid for quadrature encoders only. TWO\_PULSE\_COUNTING is valid only for two-pulse encoders.
- units (Optional[nidaqmx.constants.AngularVelocityUnits]) Specifies in which unit to return velocity measurements from the channel.
- **pulses\_per\_rev** (Optional[int])—Is the number of pulses the encoder generates per revolution. This value is the number of pulses on either signal A or signal B, not the total number of pulses on both signal A and signal B.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

 $\textbf{Return type} \ \textit{nidaqmx.\_task\_modules.channels.ci\_channel.CIC} hannel$ 

Creates a channel to count the number of rising or falling edges of a digital signal. With the exception of devices that support multi-counter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the default input terminal of the counter unless you select a different input terminal.

### **Parameters**

- counter (str) Specifies the name of the counter to use to create the virtual channel.
   The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- **edge** (Optional[nidaqmx.constants.Edge]) Specifies on which edges of the input signal to increment or decrement the count.
- initial\_count (Optional[int]) Is the value from which to start counting.
- **count\_direction** (Optional[nidaqmx.constants.CountDirection]) Specifies whether to increment or decrement the counter on each edge.

**Returns** Indicates the newly created channel object.

Return type nidaqmx.\_task\_modules.channels.ci\_channel.CIChannel

Creates channel(s) to duty cycle of a digital pulse. Connect the input signal to the default input terminal of the counter unless you select a different input terminal. With the exception of devices that support multi-counter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter.

#### **Parameters**

- **counter** (str) Specifies the name of the counter to use to create the virtual channel. The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_freq (Optional[float]) Specifies the minimum frequency you expect to measure.
- max\_freq (Optional[float]) Specifies the maximum frequency you expect to measure.
- **edge** (Optional[nidaqmx.constants.Edge]) Specifies between which edges to measure the frequency or period of the signal.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx*.\_*task*\_*modules.channels.ci*\_*channel*.*CIChannel* 

```
add_ci_freq_chan (counter, name_to_assign_to_channel=u'', min_val=2.0, max_val=100.0, units=<FrequencyUnits.HZ: 10373>, edge=<Edge.RISING: 10280>, meas_method=<CounterFrequencyMethod.LOW_FREQUENCY_1_COUNTER: 10105>, meas_time=0.001, divisor=4, custom_scale_name=u'')

Creates a channel to measure the frequency of a digital signal. With the exception of devices that support
```

multi-counter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the default input terminal of the counter unless you select a different input terminal.

## **Parameters**

- counter (str) Specifies the name of the counter to use to create the virtual channel.
   The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAOmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.FrequencyUnits]) Specifies the units to use to return frequency measurements.
- **edge** (Optional[nidaqmx.constants.Edge]) Specifies between which edges to measure the frequency or period of the signal.
- meas\_method (Optional[nidaqmx.constants. CounterFrequencyMethod]) Specifies the method to use to calculate the period or frequency of the signal.
- meas\_time (Optional[float]) Is the length of time in seconds to measure the frequency or period of the signal if meas\_method is HIGH\_FREQUENCYWITH\_2\_COUNTERS. Leave this input unspecified if meas\_method is not HIGH\_FREQUENCYWITH\_2\_COUNTERS.
- divisor (Optional[int]) Is the value by which to divide the input signal when meas\_method is LARGE\_RANGEWITH\_2\_COUNTERS. Leave this input unspecified if meas\_method is not LARGE\_RANGEWITH\_2\_COUNTERS.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM CUSTOM SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx. task modules.channels.ci channel.CIChannel* 

Creates a channel that uses a special purpose counter to take a timestamp and synchronizes that counter to a GPS receiver. With the exception of devices that support multi-counter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signals to the default input terminals of the counter unless you select different input terminals.

# **Parameters**

- **counter** (str) Specifies the name of the counter to use to create the virtual channel. The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- units (Optional[nidaqmx.constants.TimeUnits]) Specifies the units to use to return the timestamp.
- **sync\_method** (Optional[nidaqmx.constants.GpsSignalType]) **Speci**fies the method to use to synchronize the counter to a GPS receiver.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

Return type nidagmx. task modules.channels.ci channel.CIChannel

```
add_ci_lin_encoder_chan (counter, name_to_assign_to_channel=u'', decod-
ing_type=<EncoderType.X_4: 10092>, zidx_enable=False,
zidx_val=0, zidx_phase=<EncoderZIndexPhase.AHIGH_BHIGH:
10040>, units=<LengthUnits.METERS: 10219>,
dist_per_pulse=0.001, initial_pos=0.0, custom_scale_name=u'')
```

Creates a channel that uses a linear encoder to measure linear position. With the exception of devices that support multi- counter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signals to the default input terminals of the counter unless you select different input terminals.

# **Parameters**

- **counter** (str) Specifies the name of the counter to use to create the virtual channel. The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- decoding\_type (Optional[nidaqmx.constants.EncoderType]) Specifies how to count and interpret the pulses the encoder generates on signal A and signal B.
   X\_1, X\_2, and X\_4 are valid for quadrature encoders only. TWO\_PULSE\_COUNTING is valid only for two-pulse encoders.
- zidx\_enable (Optional[bool]) Specifies whether to use Z indexing for the channel.
- **zidx\_val** (Optional[float]) Specifies in **units** the value to which to reset the measurement when signal Z is high and signal A and signal B are at the states you specify with **zidx\_phase**.
- zidx\_phase (Optional[nidaqmx.constants.EncoderZIndexPhase]) Specifies the states at which signal A and signal B must be while signal Z is high for NI-DAQmx to reset the measurement. If signal Z is never high while signal A and signal B are high, for example, you must choose a phase other than A\_HIGH\_B\_HIGH.
- units (Optional[nidaqmx.constants.LengthUnits]) Specifies the units to use to return linear position measurements from the channel.

- **dist\_per\_pulse** (Optional[float]) Is the distance to measure for each pulse the encoder generates on signal A or signal B. This value is in the units you specify with the **units** input.
- initial\_pos (Optional[float]) Is the position of the encoder when you begin the measurement. This value is in the units you specify with the units input.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM CUSTOM SCALE**.

**Return type** *nidaqmx.\_task\_modules.channels.ci\_channel.CIChannel* 

Creates a channel that uses a linear encoder to measure linear velocity. With the exception of devices that support multi- counter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the default input terminal of the counter unless you select a different input terminal.

#### **Parameters**

- counter (str) Specifies the name of the counter to use to create the virtual channel.
   The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- decoding\_type (Optional [nidaqmx.constants.EncoderType]) Specifies how to count and interpret the pulses the encoder generates on signal A and signal B.
   X\_1, X\_2, and X\_4 are valid for quadrature encoders only. TWO\_PULSE\_COUNTING is valid only for two-pulse encoders.
- units (Optional[nidaqmx.constants.VelocityUnits]) Specifies in which unit to return velocity measurements from the channel.
- **dist\_per\_pulse** (Optional[float]) Is the distance to measure for each pulse the encoder generates on signal A or signal B. This value is in the units you specify with the **units** input.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** *nidaqmx*.\_*task*\_*modules.channels.ci*\_*channel.CIChannel* 

Creates a channel to measure the period of a digital signal. With the exception of devices that support multi-counter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the default input terminal of the counter unless you select a different input terminal.

#### **Parameters**

- counter (str) Specifies the name of the counter to use to create the virtual channel.
   The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TimeUnits]) Specifies the units to use to return time or period measurements.
- edge (Optional [nidaqmx.constants.Edge]) Specifies between which edges to measure the frequency or period of the signal.
- meas\_method (Optional[nidaqmx.constants. CounterFrequencyMethod]) Specifies the method to use to calculate the period or frequency of the signal.
- meas\_time (Optional[float]) Is the length of time in seconds to measure the frequency or period of the signal if meas\_method is HIGH\_FREQUENCYWITH\_2\_COUNTERS. Leave this input unspecified if meas\_method is not HIGH\_FREQUENCYWITH\_2\_COUNTERS.
- divisor (Optional[int]) Is the value by which to divide the input signal when meas\_method is LARGE\_RANGEWITH\_2\_COUNTERS. Leave this input unspecified if meas\_method is not LARGE\_RANGEWITH\_2\_COUNTERS.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

Return type nidaqmx.\_task\_modules.channels.ci\_channel.CIChannel

Creates a channel to measure pulse specifications, returning the measurements as pairs of frequency and duty cycle. With the exception of devices that support multi-counter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the default input terminal of the counter unless you select a different input terminal.

# **Parameters**

- **counter** (str) Specifies the name of the counter to use to create the virtual channel. The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.FrequencyUnits]) Specifies the units to use to return pulse specifications in terms of frequency.

**Return type** *nidagmx*.\_task\_modules.channels.ci\_channel.CIChannel

Creates a channel to measure pulse specifications, returning the measurements as pairs of high ticks and low ticks. With the exception of devices that support multi-counter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the default input terminal of the counter unless you select a different input terminal.

### **Parameters**

- counter (str) Specifies the name of the counter to use to create the virtual channel.
   The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- **source\_terminal** (Optional[str]) Is the terminal to which you connect a signal to use as the source of ticks. A DAQmx terminal constant lists all terminals available on devices installed in the system. You also can specify a source terminal by specifying a string that contains a terminal name. If you specify OnboardClock, or do not specify any terminal, NI-DAQmx selects the fastest onboard timebase available on the device.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.

**Returns** Indicates the newly created channel object.

**Return type** nidaqmx.\_task\_modules.channels.ci\_channel.CIChannel

```
add_ci_pulse_chan_time (counter, name_to_assign_to_channel=u'', min_val=1e-06, max_val=0.001, units=<TimeUnits.SECONDS: 10364>)
```

Creates a channel to measure pulse specifications, returning the measurements as pairs of high time and low time. With the exception of devices that support multi-counter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To

read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the default input terminal of the counter unless you select a different input terminal.

#### **Parameters**

- **counter** (str) Specifies the name of the counter to use to create the virtual channel. The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional [nidaqmx.constants.TimeUnits]) Specifies the units to use to return pulse specifications in terms of high time and low time.

**Returns** Indicates the newly created channel object.

Return type nidaqmx.\_task\_modules.channels.ci\_channel.CIChannel

```
add_ci_pulse_width_chan (counter, name_to_assign_to_channel=u'', min_val=1e-06, max_val=0.1, units=<TimeUnits.SECONDS: 10364>, start-ing edge=<Edge.RISING: 10280>, custom scale name=u'')
```

Creates a channel to measure the width of a digital pulse. **starting\_edge** determines whether to measure a high pulse or low pulse. With the exception of devices that support multi-counter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the default input terminal of the counter unless you select a different input terminal.

## **Parameters**

- **counter** (str) Specifies the name of the counter to use to create the virtual channel. The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TimeUnits]) Specifies the units to use to return time or period measurements.
- **starting\_edge** (Optional[nidaqmx.constants.Edge]) Specifies on which edge to begin measuring pulse width.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** *nidaqmx*.\_*task*\_*modules.channels.ci*\_*channel*.*CIChannel* 

Creates a channel to measure the time between state transitions of a digital signal. With the exception of devices that support multi-counter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signal to the default input terminal of the counter unless you select a different input terminal.

#### **Parameters**

- counter (str) Specifies the name of the counter to use to create the virtual channel.
   The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.
- units (Optional[nidaqmx.constants.TimeUnits]) Specifies the units to use to return time or period measurements.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

Returns Indicates the newly created channel object.

**Return type** *nidagmx*.\_task\_modules.channels.ci\_channel.CIChannel

```
add_ci_two_edge_sep_chan (counter, name_to_assign_to_channel=u'', min_val=1e-
06, max_val=1.0, units=<TimeUnits.SECONDS:
10364>, first_edge=<Edge.RISING: 10280>, sec-
ond_edge=<Edge.FALLING: 10171>, custom_scale_name=u'')
```

Creates a channel that measures the amount of time between the rising or falling edge of one digital signal and the rising or falling edge of another digital signal. With the exception of devices that support multicounter tasks, you can create only one counter input channel at a time with this function because a task can contain only one counter input channel. To read from multiple counters simultaneously, use a separate task for each counter. Connect the input signals to the default input terminals of the counter unless you select different input terminals.

#### **Parameters**

- **counter** (str) Specifies the name of the counter to use to create the virtual channel. The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- min\_val (Optional[float]) Specifies in units the minimum value you expect to measure.
- max\_val (Optional[float]) Specifies in units the maximum value you expect to measure.

- units (Optional[nidaqmx.constants.TimeUnits]) Specifies the units to use to return time or period measurements.
- **first\_edge** (Optional[nidaqmx.constants.Edge]) Specifies on which edge of the first signal to start each measurement.
- **second\_edge** (Optional[nidaqmx.constants.Edge]) Specifies on which edge of the second signal to stop each measurement.
- **custom\_scale\_name** (Optional[str]) Specifies the name of a custom scale for the channel. If you want the channel to use a custom scale, specify the name of the custom scale to this input and set **units** to **FROM\_CUSTOM\_SCALE**.

**Returns** Indicates the newly created channel object.

**Return type** *nidaqmx*.\_*task*\_*modules.channels.ci*\_*channel*.*CIChannel* 

#### all

nidaqmx.\_task\_modules.channels.channel.Channel-Specifies a channel object that represents the entire list of virtual channels on this channel collection.

#### channel names

*List[str]* – Specifies the entire list of virtual channels on this channel collection.

```
count (value) \rightarrow integer – return number of occurrences of value
```

```
index (value) \rightarrow integer – return first index of value.
```

Raises ValueError if the value is not present.

## nidagmx.task.co channel collection

```
class nidaqmx._task_modules.co_channel_collection.COChannelCollection(task_handle)
    Bases: nidaqmx. task modules.channel collection.ChannelCollection
```

Contains the collection of counter output channels for a DAQmx Task.

```
add_co_pulse_chan_freq(counter, name_to_assign_to_channel=u'', units=<FrequencyUnits.HZ: 10373>, idle_state=<Level.LOW: 10214>, initial_delay=0.0, freq=1.0, duty_cycle=0.5)
```

Creates channel(s) to generate digital pulses that **freq** and **duty\_cycle** define. The pulses appear on the default output terminal of the counter unless you select a different output terminal.

# **Parameters**

- **counter** (str) Specifies the names of the counters to use to create the virtual channels. The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- units (Optional[nidaqmx.constants.FrequencyUnits]) Specifies the units in which to define pulse frequency.
- idle\_state (Optional [nidaqmx.constants.Level]) Specifies the resting state of the output terminal.
- initial\_delay (Optional[float]) Is the amount of time in seconds to wait before generating the first pulse.
- **freq** (Optional [float]) Specifies at what frequency to generate pulses.

• **duty\_cycle** (Optional[float]) – Is the width of the pulse divided by the pulse period. NI-DAQmx uses this ratio combined with frequency to determine pulse width and the interval between pulses.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx.\_task\_modules.channels.co\_channel.COChannel* 

Creates channel(s) to generate digital pulses defined by the number of timebase ticks that the pulse is at a high state and the number of timebase ticks that the pulse is at a low state. The pulses appear on the default output terminal of the counter unless you select a different output terminal.

#### **Parameters**

- **counter** (str) Specifies the names of the counters to use to create the virtual channels. The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- **source\_terminal** (str) Is the terminal to which you connect an external timebase. A DAQmx terminal constant lists all terminals available on devices installed in the system. You also can specify a source terminal by specifying a string that contains a terminal name.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- idle\_state (Optional[nidaqmx.constants.Level]) Specifies the resting state of the output terminal.
- initial\_delay (Optional[int]) Is the number of timebase ticks to wait before generating the first pulse.
- low\_ticks (Optional[int]) Is the number of ticks the pulse is low.
- high\_ticks (Optional[int]) Is the number of ticks the pulse is high.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx.\_task\_modules.channels.co\_channel.COChannel* 

```
add_co_pulse_chan_time (counter, name_to_assign_to_channel=u'', units=<TimeUnits.SECONDS: 10364>, idle_state=<Level.LOW: 10214>, initial_delay=0.0, low_time=0.01, high_time=0.01)
```

Creates channel(s) to generate digital pulses defined by the amount of time the pulse is at a high state and the amount of time the pulse is at a low state. The pulses appear on the default output terminal of the counter unless you select a different output terminal.

#### **Parameters**

- **counter** (str) Specifies the names of the counters to use to create the virtual channels. The DAQmx physical channel constant lists all physical channels, including counters, for devices installed in the system.
- name\_to\_assign\_to\_channel (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- units (Optional[nidaqmx.constants.TimeUnits]) Specifies the units in which to define pulse high and low time.

- idle\_state (Optional [nidaqmx.constants.Level]) Specifies the resting state of the output terminal.
- initial\_delay (Optional[float]) Is the amount of time in seconds to wait before generating the first pulse.
- low\_time (Optional[float]) Is the amount of time the pulse is low.
- high\_time (Optional[float]) Is the amount of time the pulse is high.

**Returns** Indicates the newly created channel object.

**Return type** *nidagmx.\_task\_modules.channels.co\_channel.COChannel* 

#### all

nidaqmx.\_task\_modules.channels.channel.Channel-Specifies a channel object that represents the entire list of virtual channels on this channel collection.

#### channel\_names

List[str] – Specifies the entire list of virtual channels on this channel collection.

**count** (*value*)  $\rightarrow$  integer – return number of occurrences of value

**index** (*value*)  $\rightarrow$  integer – return first index of value.

Raises ValueError if the value is not present.

## nidagmx.task.di channel collection

```
class nidaqmx._task_modules.di_channel_collection.DIChannelCollection(task_handle)
    Bases: nidaqmx. task modules.channel collection.ChannelCollection
```

Contains the collection of digital input channels for a DAQmx Task.

```
add_di_chan (lines, name_to_assign_to_lines=u'', line_grouping=<LineGrouping.CHAN_FOR_ALL_LINES:</pre>
```

Creates channel(s) to measure digital signals. You can group digital lines into one digital channel or separate them into multiple digital channels. If you specify one or more entire ports in the **lines** input by using port physical channel names, you cannot separate the ports into multiple channels. To separate ports into multiple channels, use this function multiple times with a different port each time.

## **Parameters**

- **lines** (str) Specifies the names of the digital lines or ports to use to create virtual channels. The DAQmx physical channel constant lists all lines and ports for devices installed in the system.
- name\_to\_assign\_to\_lines (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- line\_grouping (Optional [nidaqmx.constants.LineGrouping]) Specifies how to group digital lines into one or more virtual channels. If you specify one or more entire ports with the lines input, you must set this input to one channel for all lines.

**Returns** Indicates the newly created channel object.

**Return type** *nidaqmx*.\_*task*\_*modules.channels.di*\_*channel.DIChannel* 

## all

nidaqmx.\_task\_modules.channels.channel.Channel - Specifies a channel object that represents the entire list of virtual channels on this channel collection.

#### channel names

*List[str]* – Specifies the entire list of virtual channels on this channel collection.

**count** (value)  $\rightarrow$  integer – return number of occurrences of value

**index** (*value*)  $\rightarrow$  integer – return first index of value.

Raises ValueError if the value is not present.

## nidagmx.task.do channel collection

```
class nidaqmx._task_modules.do_channel_collection.DOChannelCollection (task_handle)
    Bases: nidaqmx._task_modules.channel_collection.ChannelCollection
```

Contains the collection of digital output channels for a DAQmx Task.

Creates channel(s) to generate digital signals. You can group digital lines into one digital channel or separate them into multiple digital channels. If you specify one or more entire ports in **lines** input by using port physical channel names, you cannot separate the ports into multiple channels. To separate ports into multiple channels, use this function multiple times with a different port each time.

#### **Parameters**

- **lines** (str) Specifies the names of the digital lines or ports to use to create virtual channels. The DAQmx physical channel constant lists all lines and ports for devices installed in the system.
- name\_to\_assign\_to\_lines (Optional[str]) Specifies a name to assign to the virtual channel this function creates. If you do not specify a value for this input, NI-DAQmx uses the physical channel name as the virtual channel name.
- line\_grouping(Optional[nidaqmx.constants.LineGrouping])—Specifies how to group digital lines into one or more virtual channels. If you specify one or more entire ports with the lines input, you must set this input to one channel for all lines.

**Returns** Indicates the newly created channel object.

**Return type** nidaqmx.\_task\_modules.channels.do\_channel.DOChannel

all

nidaqmx.\_task\_modules.channels.channel.Channel-Specifies a channel object that represents the entire list of virtual channels on this channel collection.

## channel names

List[str] – Specifies the entire list of virtual channels on this channel collection.

**count** (value)  $\rightarrow$  integer – return number of occurrences of value

**index** (*value*)  $\rightarrow$  integer – return first index of value.

Raises ValueError if the value is not present.

# nidaqmx.task.export\_signals

```
class nidaqmx._task_modules.export_signals.ExportSignals(task_handle)
    Bases: object
```

Represents the exported signal configurations for a DAQmx task.

```
adv_cmplt_event_delay
```

float – Specifies the output signal delay in periods of the sample clock.

## adv\_cmplt\_event\_output\_term

str – Specifies the terminal to which to route the Advance Complete Event.

## adv\_cmplt\_event\_pulse\_polarity

nidaqmx.constants.Polarity - Specifies the polarity of the exported Advance Complete Event.

## adv\_cmplt\_event\_pulse\_width

float – Specifies the width of the exported Advance Complete Event pulse.

#### adv trig output term

str – Specifies the terminal to which to route the Advance Trigger.

# adv\_trig\_pulse\_polarity

nidaqmx.constants.Polarity - Indicates the polarity of the exported Advance Trigger.

#### adv\_trig\_pulse\_width

*float* – Specifies the width of an exported Advance Trigger pulse. Specify this value in the units you specify with **adv\_trig\_pulse\_width\_units**.

## adv\_trig\_pulse\_width\_units

nidaqmx.constants.DigitalWidthUnits - Specifies the units of adv\_trig\_pulse\_width.

#### ai\_conv\_clk\_output\_term

str – Specifies the terminal to which to route the AI Convert Clock.

## ai\_conv\_clk\_pulse\_polarity

nidaqmx.constants.Polarity – Indicates the polarity of the exported AI Convert Clock. The polarity is fixed and independent of the active edge of the source of the AI Convert Clock.

## ai hold cmplt event output term

str – Specifies the terminal to which to route the AI Hold Complete Event.

## ai\_hold\_cmplt\_event\_pulse\_polarity

nidaqmx.constants.Polarity - Specifies the polarity of an exported AI Hold Complete Event pulse.

# change\_detect\_event\_output\_term

str – Specifies the terminal to which to route the Change Detection Event.

## change\_detect\_event\_pulse\_polarity

nidaqmx.constants.Polarity - Specifies the polarity of an exported Change Detection Event pulse.

# ctr\_out\_event\_output\_behavior

nidaqmx.constants.ExportAction - Specifies whether the exported Counter Output Event pulses or changes from one state to the other when the counter reaches terminal count.

#### ctr out event output term

str – Specifies the terminal to which to route the Counter Output Event.

# ctr\_out\_event\_pulse\_polarity

nidaqmx.constants.Polarity – Specifies the polarity of the pulses at the output terminal of the counter when ctr\_out\_event\_output\_behavior is ExportActions2.PULSE. NI-DAQmx ignores this property if ctr\_out\_event\_output\_behavior is ExportActions2.TOGGLE.

#### ctr\_out\_event\_toggle\_idle\_state

nidaqmx.constants.Level – Specifies the initial state of the output terminal of the counter when ctr\_out\_event\_output\_behavior is ExportActions2.TOGGLE. The terminal enters this state when NI-DAQmx commits the task.

## data active event lvl active lvl

nidagmx.constants.Polarity - Specifies the polarity of the exported Data Active Event.

## data\_active\_event\_output\_term

str – Specifies the terminal to which to export the Data Active Event.

## divided\_samp\_clk\_timebase\_output\_term

str – Specifies the terminal to which to route the Divided Sample Clock Timebase.

## export\_signal (signal\_id, output\_terminal)

Routes a control signal to the terminal you specify. The output terminal can reside on the device that generates the control signal or on a different device. You can use this function to share clocks and triggers among multiple tasks and devices. The routes this function creates are task-based routes.

#### **Parameters**

- **signal\_id** (nidaqmx.constants.Signal) Is the name of the trigger, clock, or event to export.
- **output\_terminal** (str) Is the destination of the exported signal. A DAQmx terminal constant lists all terminals on installed devices. You can also specify a string containing a comma-delimited list of terminal names.

## exported\_10\_m\_hz\_ref\_clk\_output\_term

str – Specifies the terminal to which to route the 10MHz Clock.

## exported\_20\_m\_hz\_timebase\_output\_term

str – Specifies the terminal to which to route the 20MHz Timebase.

#### hshk event delay

float – Specifies the number of seconds to delay after the Handshake Trigger deasserts before asserting the Handshake Event.

## hshk\_event\_interlocked\_assert\_on\_start

*bool* – Specifies to assert the Handshake Event when the task starts if **hshk\_event\_output\_behavior** is **ExportActions5.INTERLOCKED**.

# hshk\_event\_interlocked\_asserted\_lvl

nidaqmx.constants.Level - Specifies the asserted level of the exported Handshake Event if hshk\_event\_output\_behavior is ExportActions5.INTERLOCKED.

## hshk\_event\_interlocked\_deassert\_delay

*float* – Specifies in seconds the amount of time to wait after the Handshake Trigger asserts before deasserting the Handshake Event if **hshk\_event\_output\_behavior** is **ExportActions5.INTERLOCKED**.

## hshk\_event\_output\_behavior

nidaqmx.constants.ExportAction - Specifies the output behavior of the Handshake Event.

## hshk\_event\_output\_term

str – Specifies the terminal to which to route the Handshake Event.

#### hshk\_event\_pulse\_polarity

nidaqmx.constants.Polarity - Specifies the polarity of the exported Handshake Event if hshk\_event\_output\_behavior is ExportActions5.PULSE.

# hshk\_event\_pulse\_width

*float* – Specifies in seconds the pulse width of the exported Handshake Event if **hshk\_event\_output\_behavior** is **ExportActions5.PULSE**.

## pause\_trig\_lvl\_active\_lvl

nidaqmx.constants.Polarity - Specifies the active level of the exported Pause Trigger.

## pause\_trig\_output\_term

str – Specifies the terminal to which to route the Pause Trigger.

#### rdy for start event lvl active lvl

nidagmx.constants.Polarity - Specifies the polarity of the exported Ready for Start Event.

# rdy\_for\_start\_event\_output\_term

str - Specifies the terminal to which to route the Ready for Start Event.

## rdy\_for\_xfer\_event\_deassert\_cond

nidaqmx.constants.DeassertCondition - Specifies when the ready for transfer event deasserts.

## rdy\_for\_xfer\_event\_deassert\_cond\_custom\_threshold

int – Specifies in samples the threshold below which the Ready for Transfer Event deasserts. This threshold is an amount of space available in the onboard memory of the device. rdy\_for\_xfer\_event\_deassert\_cond must be DeassertCondition.ONBOARD\_MEMORY\_CUSTOM\_THRESHOLD to use a custom threshold.

## rdy\_for\_xfer\_event\_lvl\_active\_lvl

nidaqmx.constants.Polarity - Specifies the active level of the exported Ready for Transfer Event.

#### rdy for xfer event output term

str – Specifies the terminal to which to route the Ready for Transfer Event.

## ref\_trig\_output\_term

str – Specifies the terminal to which to route the Reference Trigger.

#### ref\_trig\_pulse\_polarity

nidaqmx.constants.Polarity - Specifies the polarity of the exported Reference Trigger.

# samp\_clk\_delay\_offset

*float* – Specifies in seconds the amount of time to offset the exported Sample clock. Refer to timing diagrams for generation applications in the device documentation for more information about this value.

# samp\_clk\_output\_behavior

nidaqmx.constants.ExportAction—Specifies whether the exported Sample Clock issues a pulse at the beginning of a sample or changes to a high state for the duration of the sample.

#### samp\_clk\_output\_term

str – Specifies the terminal to which to route the Sample Clock.

## samp\_clk\_pulse\_polarity

nidaqmx.constants.Polarity - Specifies the polarity of the exported Sample Clock if samp\_clk\_output\_behavior is ExportActions3.PULSE.

## samp\_clk\_timebase\_output\_term

str – Specifies the terminal to which to route the Sample Clock Timebase.

#### start\_trig\_output\_term

str – Specifies the terminal to which to route the Start Trigger.

# start\_trig\_pulse\_polarity

nidaqmx.constants.Polarity - Specifies the polarity of the exported Start Trigger.

## sync\_pulse\_event\_output\_term

str – Specifies the terminal to which to route the Synchronization Pulse Event.

## watchdog\_expired\_event\_output\_term

str – Specifies the terminal to which to route the Watchdog Timer Expired Event.

# nidagmx.task.in stream

```
class nidaqmx._task_modules.in_stream.InStream(task)
    Bases: object
```

Exposes an input data stream on a DAQmx task.

The input data stream be used to control reading behavior and can be used in conjunction with reader classes to read samples from an NI-DAQmx task.

#### accessory\_insertion\_or\_removal\_detected

bool – Indicates if any device(s) in the task detected the insertion or removal of an accessory since the task started. Reading this property clears the accessory change status for all channels in the task. You must read this property before you read **devs\_with\_inserted\_or\_removed\_accessories**. Otherwise, you will receive an error.

## auto\_start

bool – Specifies if DAQmx Read automatically starts the task if you did not start the task explicitly by using DAQmx Start. The default value is True. When DAQmx Read starts a finite acquisition task, it also stops the task after reading the last sample.

#### avail samp per chan

*int* – Indicates the number of samples available to read per channel. This value is the same for all channels in the task.

#### change\_detect\_overflowed

*bool* – Indicates if samples were missed because change detection events occurred faster than the device could handle them. Some devices detect overflows differently than others.

## channels\_to\_read

nidaqmx.\_task\_modules.channels.channel.Channel - Specifies a subset of channels in the task from which to read.

## common\_mode\_range\_error\_chans

*List[str]* – Indicates a list of names of any virtual channels in the task for which the device(s) detected a common mode range violation. You must read **common\_mode\_range\_error\_chans\_exist** before you read this property. Otherwise, you will receive an error.

#### common mode range error chans exist

bool – Indicates if the device(s) detected a common mode range violation for any virtual channel in the task. Common mode range violation occurs when the voltage of either the positive terminal or negative terminal to ground are out of range. Reading this property clears the common mode range violation status for all channels in the task. You must read this property before you read **common\_mode\_range\_error\_chans**. Otherwise, you will receive an error.

```
\begin{tabular}{ll} {\bf configure\_logging} (file\_path, & logging\_mode=<LoggingMode.LOG\_AND\_READ: & 15842>, \\ & group\_name=u'', & operation=<LoggingOperation.OPEN\_OR\_CREATE: \\ & 15846>) \\ {\bf Configures\ TDMS\ file\ logging\ for\ the\ task.} \end{tabular}
```

#### **Parameters**

- $file_path(str)$  Specifies the path to the TDMS file to which you want to log data.
- logging\_mode (Optional[nidaqmx.constants.LoggingMode]) Specifies whether to enable logging and whether to allow reading data while logging. "log" mode allows for the best performance. However, you cannot read data while logging if you specify this mode. If you want to read data while logging, specify "LOG\_AND\_READ" mode.

- **group\_name** (Optional[str]) Specifies the name of the group to create within the TDMS file for data from this task. If you append data to an existing file and the specified group already exists, NI-DAQmx appends a number symbol and a number to the group name, incrementing that number until finding a group name that does not exist. For example, if you specify a group name of Voltage Task, and that group already exists, NI-DAQmx assigns the group name Voltage Task #1, then Voltage Task #2. If you do not specify a group name, NI-DAQmx uses the name of the task.
- operation(Optional[nidaqmx.constants.LoggingOperation])-Specifies how to open the TDMS file.

#### curr\_read\_pos

float – Indicates in samples per channel the current position in the buffer.

#### devs\_with\_inserted\_or\_removed\_accessories

List[str] – Indicates the names of any devices that detected the insertion or removal of an accessory since the task started. You must read **accessory\_insertion\_or\_removal\_detected** before you read this property. Otherwise, you will receive an error.

#### di num booleans per chan

*int* – Indicates the number of booleans per channel that NI-DAQmx returns in a sample for line-based reads. If a channel has fewer lines than this number, the extra booleans are False.

#### excit fault chans

List[str] – Indicates a list of names of any virtual channels in the task for which the device(s) detected an excitation fault condition. You must read **excit\_fault\_chans\_exist** before you read this property. Otherwise, you will receive an error.

#### excit fault chans exist

bool – Indicates if the device(s) detected an excitation fault condition for any virtual channel in the task. Reading this property clears the excitation fault status for all channels in the task. You must read this property before you read **excit\_fault\_chans**. Otherwise, you will receive an error.

## input\_buf\_size

*int* – Specifies the number of samples the input buffer can hold for each channel in the task. Zero indicates to allocate no buffer. Use a buffer size of 0 to perform a hardware-timed operation without using a buffer. Setting this property overrides the automatic input buffer allocation that NI- DAQmx performs.

## input\_onbrd\_buf\_size

int – Indicates in samples per channel the size of the onboard input buffer of the device.

# logging\_file\_path

str – Specifies the path to the TDMS file to which you want to log data. If the file path is changed while the task is running, this takes effect on the next sample interval (if Logging.SampsPerFile has been set) or when DAQmx Start New File is called. New file paths can be specified by ending with "" or "/". Files created after specifying a new file path retain the same name and numbering sequence.

### logging\_file\_preallocation\_size

long – Specifies a size in samples to be used to pre-allocate space on disk. Pre-allocation can improve file I/O performance, especially in situations where multiple files are being written to disk. For finite tasks, the default behavior is to pre-allocate the file based on the number of samples you configure the task to acquire.

# logging\_file\_write\_size

*int* – Specifies the size, in samples, in which data will be written to disk. The size must be evenly divisible by the volume sector size, in bytes.

# logging\_mode

nidaqmx.constants.LoggingMode - Specifies whether to enable logging and whether to allow reading data while logging. Log mode allows for the best performance. However, you cannot read data

while logging if you specify this mode. If you want to read data while logging, specify Log and Read mode.

## logging\_pause

bool – Specifies whether logging is paused while a task is executing. If **logging\_mode** is set to Log and Read mode, this value is taken into consideration on the next call to DAQmx Read, where data is written to disk. If **logging\_mode** is set to Log Only mode, this value is taken into consideration the next time that data is written to disk. A new TDMS group is written when logging is resumed from a paused state.

## logging\_samps\_per\_file

long – Specifies how many samples to write to each file. When the file reaches the number of samples specified, a new file is created with the naming convention of <filename>\_###.tdms, where #### starts at 0001 and increments automatically with each new file. For example, if the file specified is C:data.tdms, the next file name used is C:data\_0001.tdms. To disable file spanning behavior, set this attribute to 0. If logging\_file\_path is changed while this attribute is set, the new file path takes effect on the next file created.

## logging\_tdms\_group\_name

str – Specifies the name of the group to create within the TDMS file for data from this task. If you append data to an existing file and the specified group already exists, NI- DAQmx appends a number symbol and a number to the group name, incrementing that number until finding a group name that does not exist. For example, if you specify a group name of Voltage Task, and that group already exists, NI- DAQmx assigns the group name Voltage Task #1, then Voltage Task #2.

#### logging\_tdms\_operation

nidagmx.constants.LoggingOperation - Specifies how to open the TDMS file.

#### num chans

*int* – Indicates the number of channels that DAQmx Read reads from the task. This value is the number of channels in the task or the number of channels you specify with **channels\_to\_read**.

## offset

int – Specifies an offset in samples per channel at which to begin a read operation. This offset is relative to the location you specify with **relative\_to**.

## open\_chans

*List[str]* – Indicates a list of names of any open virtual channels. You must read **open\_chans\_exist** before you read this property. Otherwise you will receive an error.

## open\_chans\_details

*List[str]* – Indicates a list of details of any open virtual channels. You must read **open\_chans\_exist** before you read this property. Otherwise you will receive an error.

#### open chans exist

bool – Indicates if the device or devices detected an open channel condition in any virtual channel in the task. Reading this property clears the open channel status for all channels in this task. You must read this property before you read **open\_chans**. Otherwise, you will receive an error.

# open\_current\_loop\_chans

List[str] – Indicates a list of names of any virtual channels in the task for which the device(s) detected an open current loop. You must read **open\_current\_loop\_chans\_exist** before you read this property. Otherwise, you will receive an error.

## open\_current\_loop\_chans\_exist

bool – Indicates if the device(s) detected an open current loop for any virtual channel in the task. Reading this property clears the open current loop status for all channels in the task. You must read this property before you read **open\_current\_loop\_chans**. Otherwise, you will receive an error.

# open\_thrmcpl\_chans

List[str] - Indicates a list of names of any virtual channels in the task for which the device(s) detected an

open thermcouple. You must read **open\_thrmcpl\_chans\_exist** before you read this property. Otherwise, you will receive an error.

## open\_thrmcpl\_chans\_exist

bool – Indicates if the device(s) detected an open thermocouple connected to any virtual channel in the task. Reading this property clears the open thermocouple status for all channels in the task. You must read this property before you read **open\_thrmcpl\_chans**. Otherwise, you will receive an error.

#### over\_write

nidaqmx.constants.OverwriteMode - Specifies whether to overwrite samples in the buffer that you have not yet read.

## overcurrent\_chans

List[str] – Indicates a list of names of any virtual channels in the task for which the device(s) detected an overcurrent condition. You must read **overcurrent\_chans\_exist** before you read this property. Otherwise, you will receive an error. On some devices, you must restart the task for all overcurrent channels to recover.

## overcurrent\_chans\_exist

*bool* – Indicates if the device(s) detected an overcurrent condition for any virtual channel in the task. Reading this property clears the overcurrent status for all channels in the task. You must read this property before you read **overcurrent\_chans**. Otherwise, you will receive an error.

#### overloaded chans

*List[str]* – Indicates a list of names of any overloaded virtual channels in the task. You must read **overloaded\_chans\_exist** before you read this property. Otherwise, you will receive an error.

## overloaded chans exist

bool – Indicates if the device(s) detected an overload in any virtual channel in the task. Reading this property clears the overload status for all channels in the task. You must read this property before you read **overloaded\_chans**. Otherwise, you will receive an error.

## overtemperature\_chans

*List[str]* – Indicates a list of names of any overtemperature virtual channels. You must read **overtemperature\_chans\_exist** before you read this property. Otherwise, you will receive an error.

## overtemperature\_chans\_exist

bool – Indicates if the device(s) detected an overtemperature condition in any virtual channel in the task. Reading this property clears the overtemperature status for all channels in the task. You must read this property before you read **overtemperature\_chans**. Otherwise, you will receive an error.

# raw data width

*int* – Indicates in bytes the size of a raw sample from the task.

## read (number\_of\_samples\_per\_channel=-1)

Reads raw samples from the task or virtual channels you specify.

Raw samples constitute the internal representation of samples in a device, read directly from the device or buffer without scaling or reordering. The native format of a device can be an 8-, 16-, or 32-bit integer, signed or unsigned.

NI-DAQmx does not separate raw data into channels. It returns data in an interleaved or non-interleaved 1D array, depending on the raw ordering of the device. Refer to your device documentation for more information.

This method determines a NumPy array of appropriate size and data type to create and return based on your device specifications.

Use the "timeout" property on the stream to specify the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidagmx.WAIT INFINITELY,

the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Parameters** number\_of\_samples\_per\_channel (int) - Specifies the number of samples to read.

If you set this input to nidaqmx.READ\_ALL\_AVAILABLE, NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously and you set this input to nidaqmx.READ\_ALL\_AVAILABLE, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples and you set this input to nidaqmx.READ\_ALL\_AVAILABLE, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to TRUE, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

**Returns** The samples requested in the form of a 1D NumPy array. This method determines a NumPy array of appropriate size and data type to create and return based on your device specifications.

Return type numpy.ndarray

#### read\_all\_avail\_samp

*bool* – Specifies whether subsequent read operations read all samples currently available in the buffer or wait for the buffer to become full before reading. NI-DAQmx uses this setting for finite acquisitions and only when the number of samples to read is -1. For continuous acquisitions when the number of samples to read is -1, a read operation always reads all samples currently available in the buffer.

# readall()

Reads all available raw samples from the task or virtual channels you specify.

NI-DAQmx determines how many samples to read based on if the task acquires samples continuously or acquires a finite number of samples.

If the task acquires samples continuously, this method reads all the samples currently available in the buffer.

If the task acquires a finite number of samples, the method waits for the task to acquire all requested samples, then reads those samples. If you set the "read\_all\_avail\_samp" property to TRUE, the method reads the samples currently available in the buffer and does not wait for the task to acquire all requested samples.

Raw samples constitute the internal representation of samples in a device, read directly from the device or buffer without scaling or reordering. The native format of a device can be an 8-, 16-, or 32-bit integer, signed or unsigned.

NI-DAQmx does not separate raw data into channels. It returns data in an interleaved or non-interleaved 1D array, depending on the raw ordering of the device. Refer to your device documentation for more information.

This method determines a NumPy array of appropriate size and data type to create and return based on your device specifications.

Use the "timeout" property on the stream to specify the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.WAIT\_INFINITELY,

the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Returns** The samples requested in the form of a 1D NumPy array. This method determines a NumPy array of appropriate size and data type to create and return based on your device specifications.

Return type numpy.ndarray

## readinto(numpy array)

Reads raw samples from the task or virtual channels you specify into numpy\_array.

The object numpy\_array should be a pre-allocated, writable 1D numpy array.

The number of samples per channel to read is determined using the following equation:

## number\_of\_samples\_per\_channel = math.floor(

```
numpy_array_size_in_bytes / ( number_of_channels_to_read * raw_sample_size_in_bytes))
```

Raw samples constitute the internal representation of samples in a device, read directly from the device or buffer without scaling or reordering. The native format of a device can be an 8-, 16-, or 32-bit integer, signed or unsigned.

If you use a different integer size than the native format of the device, one integer can contain multiple samples or one sample can stretch across multiple integers. For example, if you use 32-bit integers, but the device uses 8-bit samples, one integer contains up to four samples. If you use 8-bit integers, but the device uses 16-bit samples, a sample might require two integers. This behavior varies from device to device. Refer to your device documentation for more information.

NI-DAQmx does not separate raw data into channels. It returns data in an interleaved or non-interleaved 1D array, depending on the raw ordering of the device. Refer to your device documentation for more information.

Use the "timeout" property on the stream to specify the amount of time in seconds to wait for samples to become available. If the time elapses, the method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to -1, the method waits indefinitely. If you set timeout to 0, the method tries once to read the requested samples and returns an error if it is unable to.

**Parameters** numpy\_array - Specifies the 1D NumPy array object into which the samples requested are read.

**Returns** Indicates the total number of samples read.

Return type int

#### relative to

nidaqmx.constants.ReadRelativeTo - Specifies the point in the buffer at which to begin a read operation. If you also specify an offset with **offset**, the read operation begins at that offset relative to the point you select with this property. The default value is **ReadRelativeTo.CURRENT\_READ\_POSITION** unless you configure a Reference Trigger for the task. If you configure a Reference Trigger, the default value is **ReadRelativeTo.FIRST\_PRETRIGGER\_SAMPLE**.

## sleep\_time

float – Specifies in seconds the amount of time to sleep after checking for available samples if wait\_mode is WaitMode.SLEEP.

## start\_new\_file (file\_path)

Starts a new TDMS file the next time data is written to disk.

**Parameters file\_path** (str) – Specifies the path to the TDMS file to which you want to log data.

#### timeout

float – Specifies the amount of time in seconds to wait for samples to become available. If the time elapses, the read method returns an error and any samples read before the timeout elapsed. The default timeout is 10 seconds. If you set timeout to nidaqmx.WAIT\_INFINITELY, the read method waits indefinitely. If you set timeout to 0, the read method tries once to read the requested samples and returns an error if it is unable to.

## total\_samp\_per\_chan\_acquired

*float* – Indicates the total number of samples acquired by each channel. NI-DAQmx returns a single value because this value is the same for all channels. For retriggered acquisitions, this value is the cumulative number of samples across all retriggered acquisitions.

#### wait\_mode

nidaqmx.constants.WaitMode - Specifies how DAQmx Read waits for samples to become available.

# nidagmx.task.out stream

class nidaqmx.\_task\_modules.out\_stream.OutStream(task)

Bases: object

Exposes an output data stream on a DAQmx task.

The output data stream be used to control writing behavior and can be used in conjunction with writer classes to write samples to an NI-DAQmx task.

## accessory\_insertion\_or\_removal\_detected

bool – Indicates if any devices in the task detected the insertion or removal of an accessory since the task started. Reading this property clears the accessory change status for all channels in the task. You must read this property before you read **devs\_with\_inserted\_or\_removed\_accessories**. Otherwise, you will receive an error.

## auto\_start

*bool* – Specifies if the "write" method automatically starts the stream's owning task if you did not explicitly start it with the DAQmx Start Task method.

## curr write pos

float – Indicates the position in the buffer of the next sample to generate. This value is identical for all channels in the task.

#### devs\_with\_inserted\_or\_removed\_accessories

List[str] – Indicates the names of any devices that detected the insertion or removal of an accessory since the task started. You must read **accessory\_insertion\_or\_removal\_detected** before you read this property. Otherwise, you will receive an error.

# ${\tt do\_num\_booleans\_per\_chan}$

*int* – Indicates the number of Boolean values expected per channel in a sample for line-based writes. This property is determined by the channel in the task with the most digital lines. If a channel has fewer lines than this number, NI- DAQmx ignores the extra Boolean values.

# external\_overvoltage\_chans

*List[str]* – Indicates a list of names of any virtual channels in the task for which an External Overvoltage condition has been detected. You must read External OvervoltageChansExist before you read this property. Otherwise, you will receive an error.

### external overvoltage chans exist

bool – Indicates if the device(s) detected an External Overvoltage condition for any channel in the task. Reading this property clears the External Overvoltage status for all channels in the task. You must read this property before you read External OvervoltageChans. Otherwise, you will receive an error.

#### num chans

*int* – Indicates the number of channels that DAQmx Write writes to the task. This value is the number of channels in the task.

#### offset

*int* – Specifies in samples per channel an offset at which a write operation begins. This offset is relative to the location you specify with **relative\_to**.

#### open current loop chans

List[str] – Indicates a list of names of any virtual channels in the task for which the device(s) detected an open current loop. You must read **open\_current\_loop\_chans\_exist** before you read this property. Otherwise, you will receive an error.

## open\_current\_loop\_chans\_exist

bool – Indicates if the device(s) detected an open current loop for any channel in the task. Reading this property clears the open current loop status for all channels in the task. You must read this property before you read **open\_current\_loop\_chans**. Otherwise, you will receive an error.

## output\_buf\_size

*int* – Specifies the number of samples the output buffer can hold for each channel in the task. Zero indicates to allocate no buffer. Use a buffer size of 0 to perform a hardware-timed operation without using a buffer. Setting this property overrides the automatic output buffer allocation that NI- DAQmx performs.

## output\_onbrd\_buf\_size

int – Specifies in samples per channel the size of the onboard output buffer of the device.

#### overcurrent chans

*List[str]* – Indicates a list of names of any virtual channels in the task for which an overcurrent condition has been detected. You must read **overcurrent\_chans\_exist** before you read this property. Otherwise, you will receive an error.

## overcurrent\_chans\_exist

bool – Indicates if the device(s) detected an overcurrent condition for any channel in the task. Reading this property clears the overcurrent status for all channels in the task. You must read this property before you read **overcurrent\_chans**. Otherwise, you will receive an error.

#### overloaded chans

*List[str]* – Indicates a list of names of any overloaded virtual channels in the task. You must read **overloaded\_chans\_exist** before you read this property. Otherwise, you will receive an error.

## overloaded\_chans\_exist

bool – Indicates if the device(s) detected an overload in any virtual channel in the task. Reading this property clears the overload status for all channels in the task. You must read this property before you read **overloaded\_chans**. Otherwise, you will receive an error.

#### overtemperature chans

*List[str]* – Indicates a list of names of any overtemperature virtual channels. You must read **overtemperature\_chans\_exist** before you read this property. Otherwise, you will receive an error. The list of names may be empty if the device cannot determine the source of the overtemperature.

# ${\tt overtemperature\_chans\_exist}$

bool – Indicates if the device(s) detected an overtemperature condition in any virtual channel in the task. Reading this property clears the overtemperature status for all channels in the task. You must read this property before you read **overtemperature\_chans**. Otherwise, you will receive an error.

## power\_supply\_fault\_chans

*List[str]* – Indicates a list of names of any virtual channels in the task that have a power supply fault. You must read **power\_supply\_fault\_chans\_exist** before you read this property. Otherwise, you will receive an error.

## power\_supply\_fault\_chans\_exist

*bool* – Indicates if the device(s) detected a power supply fault for any channel in the task. Reading this property clears the power supply fault status for all channels in the task. You must read this property before you read **power\_supply\_fault\_chans**. Otherwise, you will receive an error.

#### raw\_data\_width

*int* – Indicates in bytes the required size of a raw sample to write to the task.

#### regen\_mode

nidaqmx.constants.Regeneration Mode-Specifies whether to allow NI-DAQmx to generate the same data multiple times.

#### relative\_to

nidaqmx.constants.WriteRelativeTo – Specifies the point in the buffer at which to write data. If you also specify an offset with **offset**, the write operation begins at that offset relative to this point you select with this property.

#### sleep\_time

float – Specifies in seconds the amount of time to sleep after checking for available buffer space if wait\_mode is WaitMode2.SLEEP.

#### space\_avail

int - Indicates in samples per channel the amount of available space in the buffer.

#### timeout

float – Specifies the amount of time in seconds to wait for the write method to write all samples. NI-DAQmx performs a timeout check only if the write method must wait before it writes data. The write method returns an error if the time elapses. The default timeout is 10 seconds. If you set "timeout" to nidaqmx.WAIT\_INFINITELY, the write method waits indefinitely. If you set timeout to 0, the write method tries once to write the submitted samples. If the write method could not write all the submitted samples, it returns an error and the number of samples successfully written in the number of samples written per channel output.

# total\_samp\_per\_chan\_generated

*float* – Indicates the total number of samples generated by each channel in the task. This value is identical for all channels in the task.

#### wait mode

nidaqmx.constants.WaitMode - Specifies how DAQmx Write waits for space to become available in the buffer.

# write (numpy\_array)

Writes raw samples to the task or virtual channels you specify.

The number of samples per channel to write is determined using the following equation:

#### number of samples per channel = math.floor(

```
numpy_array_size_in_bytes / ( number_of_channels_to_write * raw_sample_size_in_bytes))
```

Raw samples constitute the internal representation of samples in a device, read directly from the device or buffer without scaling or reordering. The native format of a device can be an 8-, 16-, or 32-bit integer, signed or unsigned.

If you use a different integer size than the native format of the device, one integer can contain multiple samples or one sample can stretch across multiple integers. For example, if you use 32-bit integers, but the device uses 8-bit samples, one integer contains up to four samples. If you use 8-bit integers, but the device uses 16-bit samples, a sample might require two integers. This behavior varies from device to device. Refer to your device documentation for more information.

NI-DAQmx does not separate raw data into channels. It accepts data in an interleaved or non-interleaved 1D array, depending on the raw ordering of the device. Refer to your device documentation for more information.

If the task uses on-demand timing, this method returns only after the device generates all samples. Ondemand is the default timing type if you do not use the timing property on the task to configure a sample timing type. If the task uses any timing type other than on-demand, this method returns immediately and does not wait for the device to generate all samples. Your application must determine if the task is done to ensure that the device generated all samples.

Use the "auto\_start" property on the stream to specify if this method automatically starts the stream's owning task if you did not explicitly start it with the DAQmx Start Task method.

Use the "timeout" property on the stream to specify the amount of time in seconds to wait for the method to write all samples. NI-DAQmx performs a timeout check only if the method must wait before it writes data. This method returns an error if the time elapses. The default timeout is 10 seconds. If you set timeout to nidaqmx.WAIT\_INFINITELY, the method waits indefinitely. If you set timeout to 0, the method tries once to write the submitted samples. If the method could not write all the submitted samples, it returns an error and the number of samples successfully written.

**Parameters** numpy\_array (numpy.ndarray) - Specifies a 1D NumPy array that contains the raw samples to write to the task.

**Returns** Specifies the actual number of samples per channel successfully written to the buffer.

Return type int

# nidaqmx.task.timing

```
class nidaqmx._task_modules.timing.Timing(task_handle)
    Bases: object
```

Represents the timing configurations for a DAQmx task.

#### ai\_conv\_active\_edge

nidaqmx.constants.Edge – Specifies on which edge of the clock pulse an analog-to-digital conversion takes place.

## ai\_conv\_dig\_fltr\_enable

bool – Specifies whether to apply a digital filter to the AI Convert Clock.

#### ai\_conv\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the filter recognizes.

# ai\_conv\_dig\_fltr\_timebase\_rate

float – Specifies in hertz the rate of the digital filter timebase. NI-DAQmx uses this value to compute settings for the filter.

#### ai\_conv\_dig\_fltr\_timebase\_src

str – Specifies the terminal of the signal to use as the timebase of the digital filter.

## ai\_conv\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

## ai\_conv\_max\_rate

float – Indicates the maximum convert rate supported by the task, given the current devices and channel count.

#### ai conv rate

*float* – Specifies in Hertz the rate at which to clock the analog- to-digital converter. This clock is specific to the analog input section of multiplexed devices.

#### ai conv src

str – Specifies the terminal of the signal to use as the AI Convert Clock.

#### ai conv timebase div

*int* – Specifies the number of AI Convert Clock Timebase pulses needed to produce a single AI Convert Clock pulse.

#### ai\_conv\_timebase\_src

nidaqmx.constants.MIOAIConvertTimebaseSource - Specifies the terminal of the signal to use as the AI Convert Clock Timebase.

```
cfg_burst_handshaking_timing_export_clock (sample_clk_rate, sample_clk_outp_term, sample_mode=<AcquisitionType.FINITE:

10178>, samps_per_chan=1000, sam-
ple_clk_pulse_polarity=<Polarity.ACTIVE_HIGH:
10095>, pause_when=<Level.HIGH:
10192>, ready_event_active_level=<Polarity.ACTIVE_HIGH:
10095>)
```

Configures when the DAQ device transfers data to a peripheral device, using the onboard Sample Clock of the DAQ device to control burst handshake timing and exporting that clock for use by the peripheral device.

#### **Parameters**

- **sample\_clk\_rate** (*float*) Specifies in hertz the rate of the Sample Clock.
- **sample\_clk\_outp\_term** (*str*) Specifies the terminal to which to export the Sample Clock.
- **sample\_mode** (Optional[nidaqmx.constants.AcquisitionType]) Specifies if the task acquires or generates samples continuously or if it acquires or generates a finite number of samples.
- samps\_per\_chan (Optional[long]) Specifies the number of samples to acquire or generate for each channel in the task if sample\_mode is FINITE\_SAMPLES. If sample\_mode is CONTINUOUS\_SAMPLES, NI-DAQmx uses this value to determine the buffer size. This function returns an error if the specified value is negative.
- sample\_clk\_pulse\_polarity (Optional[nidaqmx.constants. Polarity]) Specifies the polarity of the exported Sample Clock.
- pause\_when (Optional[nidaqmx.constants.Level]) Specifies whether the task pauses while the trigger signal is high or low.
- ready\_event\_active\_level (Optional[nidaqmx.constants. Polarity]) Specifies the polarity of the Ready for Transfer Event.

```
cfg_burst_handshaking_timing_import_clock (sample_clk_rate, sample_clk_src, sample_mode=<AcquisitionType.FINITE:

10178>, samps_per_chan=1000, sample_clk_active_edge=<Edge.RISING:

10280>, pause_when=<Level.HIGH:

10192>, ready_event_active_level=<Polarity.ACTIVE_HIGH:
10095>)
```

Configures when the DAQ device transfers data to a peripheral device, using an imported sample clock to control burst handshake timing.

#### **Parameters**

- **sample\_clk\_rate** (*float*) Specifies in hertz the rate of the Sample Clock.
- **sample\_clk\_src** (str) Specifies the source terminal of the Sample Clock. Leave this input unspecified to use the default onboard clock of the device.
- **sample\_mode** (Optional[nidaqmx.constants.AcquisitionType]) Specifies if the task acquires or generates samples continuously or if it acquires or generates a finite number of samples.
- samps\_per\_chan (Optional [long]) Specifies the number of samples to acquire or generate for each channel in the task if sample\_mode is FINITE\_SAMPLES. If sample\_mode is CONTINUOUS\_SAMPLES, NI-DAQmx uses this value to determine the buffer size. This function returns an error if the specified value is negative.
- sample\_clk\_active\_edge (Optional[nidaqmx.constants.Edge]) Specifies on which edges of Sample Clock pulses to acquire or generate samples.
- pause\_when (Optional[nidaqmx.constants.Level]) Specifies whether the task pauses while the trigger signal is high or low.
- ready\_event\_active\_level (Optional[nidaqmx.constants. Polarity]) Specifies the polarity of the Ready for Transfer Event.

```
cfg_change_detection_timing(rising_edge_chan=u'', falling_edge_chan=u'', sam-ple_mode=<AcquisitionType.FINITE: 10178>, samps_per_chan=1000)
```

Configures the task to acquire samples on the rising and/or falling edges of the lines or ports you specify. To detect both rising and falling edges on a line or port, specify the name of that line or port to both rising\_edge\_chan and falling\_edge\_chan.

#### **Parameters**

- rising\_edge\_chan (Optional[str]) Specifies the names of the digital lines or ports on which to detect rising edges. The DAQmx physical channel constant lists all lines and ports for devices installed in your system.
- **falling\_edge\_chan** (Optional[str]) Specifies the names of the digital lines or ports on which to detect falling edges. The DAQmx physical channel constant lists all lines and ports for devices installed in your system.
- **sample\_mode** (Optional[nidaqmx.constants.AcquisitionType]) Specifies if the task acquires samples continuously or if it acquires a finite number of samples.
- samps\_per\_chan (Optional[long]) Specifies the number of samples to acquire from each channel in the task if sample\_mode is FINITE\_SAMPLES. This function returns an error if the specified value is negative.

```
cfg_handshaking_timing(sample_mode=<AcquisitionType.FINITE: 10178>, samps_per_chan=1000)
```

Determines the number of digital samples to acquire or generate using digital handshaking between the device and a peripheral device.

## **Parameters**

• **sample\_mode** (Optional[nidaqmx.constants.AcquisitionType]) — Specifies if the task acquires or generates samples continuously or if it acquires or generates a finite number of samples.

• samps\_per\_chan (Optional[long]) - Specifies the number of samples to acquire or generate for each channel in the task if sample\_mode is FINITE\_SAMPLES. If sample\_mode is CONTINUOUS\_SAMPLES, NI-DAQmx uses this value to determine the buffer size. This function returns an error if the specified value is negative.

10178>,

Sets only the number of samples to acquire or generate without specifying timing. Typically, you should use this instance when the task does not require sample timing, such as tasks that use counters for buffered frequency measurement, buffered period measurement, or pulse train generation. For finite counter output tasks, **samps\_per\_chan** is the number of pulses to generate.

#### **Parameters**

- **sample\_mode** (Optional[nidaqmx.constants.AcquisitionType]) Specifies if the task acquires or generates samples continuously or if it acquires or generates a finite number of samples.
- samps\_per\_chan (Optional[long]) Specifies the number of samples to acquire or generate for each channel in the task if sample\_mode is FINITE\_SAMPLES. If sample\_mode is CONTINUOUS\_SAMPLES, NI-DAQmx uses this value to determine the buffer size. This function returns an error if the specified value is negative.

```
cfg_pipelined_samp_clk_timing(rate, source=u'', active_edge=<Edge.RISING: 10280>, sample_mode=<AcquisitionType.FINITE: 10178>, samps_per_chan=1000)
```

"Sets the source of the Sample Clock, the rate of the Sample Clock, and the number of samples to acquire or generate. The device acquires or generates samples on each Sample Clock edge, but it does not respond to certain triggers until a few Sample Clock edges later. Pipelining allows higher data transfer rates at the cost of increased trigger response latency. Refer to the device documentation for information about which triggers pipelining affects.

**This timing type allows handshaking** using the Pause trigger and either the Ready for Transfer event or the Data Active event. Refer to the device documentation for more information.

This timing type is supported only by the NI 6536 and NI 6537."

## Args:

rate (float): Specifies the sampling rate in samples per channel per second. If you use an external source for the Sample Clock, set this input to the maximum expected rate of that clock.

**source (Optional[str]): Specifies the source terminal of the** Sample Clock. Leave this input unspecified to use the default onboard clock of the device.

**active\_edge (Optional[nidaqmx.constants.Edge]): Specifies on** which edges of Sample Clock pulses to acquire or generate samples.

**sample\_mode** (**Optional[nidaqmx.constants.AcquisitionType]**): Specifies if the task acquires or generates samples continuously or if it acquires or generates a finite number of samples.

samps\_per\_chan (Optional[long]): Specifies the number of samples to acquire or generate for each channel in the task if sample\_mode is FINITE\_SAMPLES. If sample\_mode is CONTINUOUS\_SAMPLES, NI-DAQmx uses this value to determine the buffer size. This function returns an error if the specified value is negative.

cfg\_samp\_clk\_timing (rate, source=u'', active\_edge=<Edge.RISING: 10280>, sam-ple\_mode=<AcquisitionType.FINITE: 10178>, samps\_per\_chan=1000)

Sets the source of the Sample Clock, the rate of the Sample Clock, and the number of samples to acquire or generate.

#### **Parameters**

- rate (float) Specifies the sampling rate in samples per channel per second. If you use an external source for the Sample Clock, set this input to the maximum expected rate of that clock.
- **source** (Optional[str]) Specifies the source terminal of the Sample Clock. Leave this input unspecified to use the default onboard clock of the device.
- active\_edge (Optional[nidaqmx.constants.Edge]) Specifies on which edges of Sample Clock pulses to acquire or generate samples.
- **sample\_mode** (Optional[nidaqmx.constants.AcquisitionType]) Specifies if the task acquires or generates samples continuously or if it acquires or generates a finite number of samples.
- samps\_per\_chan (Optional[long]) Specifies the number of samples to acquire or generate for each channel in the task if sample\_mode is FINITE\_SAMPLES. If sample\_mode is CONTINUOUS\_SAMPLES, NI-DAQmx uses this value to determine the buffer size. This function returns an error if the specified value is negative.

## change\_detect\_di\_falling\_edge\_physical\_chans

nidaqmx.system.physical\_channel.PhysicalChannel-Specifies the names of the digital lines or ports on which to detect falling edges. The lines or ports must be used by virtual channels in the task. You also can specify a string that contains a list or range of digital lines or ports.

## change\_detect\_di\_rising\_edge\_physical\_chans

nidaqmx.system.physical\_channel.PhysicalChannel-Specifies the names of the digital lines or ports on which to detect rising edges. The lines or ports must be used by virtual channels in the task. You also can specify a string that contains a list or range of digital lines or ports.

## change\_detect\_di\_tristate

bool – Specifies whether to tristate lines specified with change\_detect\_di\_rising\_edge\_physical\_chans and change\_detect\_di\_falling\_edge\_physical\_chans that are not in a virtual channel in the task. If you set this property to True, NI-DAQmx tristates rising/falling edge lines that are not in a virtual channel in the task. If you set this property to False, NI-DAQmx does not modify the configuration of rising/falling edge lines that are not in a virtual channel in the task, even if the lines were previously tristated. Set this property to False to detect changes on lines in other tasks or to detect changes on output-only lines.

# delay\_from\_samp\_clk\_delay

float – Specifies the amount of time to wait after receiving a Sample Clock edge before beginning to acquire the sample. This value is in the units you specify with **delay\_from\_samp\_clk\_delay\_units**.

# delay\_from\_samp\_clk\_delay\_units

nidaqmx.constants.DigitalWidthUnits - Specifies the units of delay\_from\_samp\_clk\_delay.

#### hshk\_delay\_after\_xfer

*float* – Specifies the number of seconds to wait after a handshake cycle before starting a new handshake cycle.

# $\verb|hshk_sample_input_data_when| \\$

nidaqmx.constants.SampleInputDataWhen-Specifies on which edge of the Handshake Trigger an input task latches the data from the peripheral device.

## hshk\_start\_cond

nidaqmx.constants.HandshakeStartCondition - Specifies the point in the handshake cycle that the device is in when the task starts.

# implicit\_underflow\_behavior

nidaqmx.constants.UnderflowBehavior - Specifies the action to take when the onboard memory of the device becomes empty.

## master\_timebase\_rate

*float* – Specifies the rate of the Master Timebase.

#### master\_timebase\_src

str – Specifies the terminal of the signal to use as the Master Timebase. On an E Series device, you can choose only between the onboard 20MHz Timebase or the RTSI7 terminal.

## ref clk rate

*float* – Specifies the frequency of the Reference Clock.

#### ref\_clk\_src

str – Specifies the terminal of the signal to use as the Reference Clock.

#### samp\_clk\_active\_edge

nidaqmx.constants.Edge - Specifies on which edge of a clock pulse sampling takes place. This property is useful primarily when the signal you use as the Sample Clock is not a periodic clock.

## samp\_clk\_dig\_fltr\_enable

bool – Specifies whether to apply the pulse width filter to the signal.

## samp\_clk\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the filter recognizes.

#### samp\_clk\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

## samp\_clk\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

## samp\_clk\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

## samp\_clk\_max\_rate

float – Indicates the maximum Sample Clock rate supported by the task, based on other timing settings. For output tasks, the maximum Sample Clock rate is the maximum rate of the DAC. For input tasks, NI-DAQmx calculates the maximum sampling rate differently for multiplexed devices than simultaneous sampling devices.

# samp\_clk\_overrun\_behavior

nidaqmx.constants.OverflowBehavior - Specifies the action to take if Sample Clock edges occur faster than the device can handle them.

#### samp clk rate

*float* – Specifies the sampling rate in samples per channel per second. If you use an external source for the Sample Clock, set this input to the maximum expected rate of that clock.

# samp\_clk\_src

str – Specifies the terminal of the signal to use as the Sample Clock.

#### samp\_clk\_term

str – Indicates the name of the internal Sample Clock terminal for the task. This property does not return the name of the Sample Clock source terminal specified with samp\_clk\_src.

## samp\_clk\_timebase\_active\_edge

nidaqmx.constants.Edge – Specifies on which edge to recognize a Sample Clock Timebase pulse. This property is useful primarily when the signal you use as the Sample Clock Timebase is not a periodic clock.

#### samp\_clk\_timebase\_div

*int* – Specifies the number of Sample Clock Timebase pulses needed to produce a single Sample Clock pulse.

#### samp\_clk\_timebase\_master\_timebase\_div

*int* – Specifies the number of pulses of the Master Timebase needed to produce a single pulse of the Sample Clock Timebase.

#### samp clk timebase rate

*float* – Specifies the rate of the Sample Clock Timebase. Some applications require that you specify a rate when you use any signal other than the onboard Sample Clock Timebase. NI- DAQmx requires this rate to calculate other timing parameters.

#### samp clk timebase src

str – Specifies the terminal of the signal to use as the Sample Clock Timebase.

#### samp\_clk\_timebase\_term

str – Indicates the name of the internal Sample Clock Timebase terminal for the task. This property does not return the name of the Sample Clock Timebase source terminal specified with samp clk timebase src.

#### samp\_clk\_underflow\_behavior

nidaqmx.constants.UnderflowBehavior - Specifies the action to take when the onboard memory of the device becomes empty. In either case, the sample clock does not stop.

#### samp\_clk\_write\_wfm\_use\_initial\_wfm\_dt

*bool* – Specifies that the value of **samp\_clk\_rate** will be determined by the dt component of the initial DAQmx Write waveform input for Output tasks.

#### samp\_quant\_samp\_mode

nidaqmx.constants.AcquisitionType – Specifies if a task acquires or generates a finite number of samples or if it continuously acquires or generates samples.

## samp\_quant\_samp\_per\_chan

float — Specifies the number of samples to acquire or generate for each channel if samp\_quant\_samp\_mode is AcquisitionType.FINITE. If samp\_quant\_samp\_mode is AcquisitionType.CONTINUOUS, NI-DAQmx uses this value to determine the buffer size.

#### samp\_timing\_engine

int – Specifies which timing engine to use for the task.

## samp\_timing\_type

nidaqmx.constants.SampleTimingType - Specifies the type of sample timing to use for the task.

#### simultaneous ao enable

*bool* – Specifies whether to update all channels in the task simultaneously, rather than updating channels independently when you write a sample to that channel.

# sync\_clk\_interval

*int* – Specifies the interval, in Sample Clock periods, between each internal Synchronization Clock pulse. NI-DAQmx uses this pulse for synchronization of triggers between multiple devices at different rates. Refer to device documentation for information about how to calculate this value.

## sync\_pulse\_min\_delay\_to\_start

*float* – Specifies in seconds the amount of time that elapses after the master device issues the synchronization pulse before the task starts.

# sync\_pulse\_reset\_delay

*float* – Specifies in seconds the amount of time to wait after the Synchronization Pulse before resetting the ADCs or DACs on the device. When synchronizing devices, query **sync pulse reset time** on all devices

and note the largest reset time. Then, for each device, subtract the reset time from the largest reset time and set this property to the resulting value.

## sync\_pulse\_reset\_time

float – Indicates in seconds the amount of time required for the ADCs or DACs on the device to reset. When synchronizing devices, query this property on all devices and note the largest reset time. Then, for each device, subtract the value of this property from the largest reset time and set **sync\_pulse\_reset\_delay** to the resulting value.

## sync\_pulse\_src

str – Specifies the terminal of the signal to use as the synchronization pulse. The synchronization pulse resets the clock dividers and the ADCs/DACs on the device.

## sync\_pulse\_sync\_time

float – Indicates in seconds the delay required to reset the ADCs/DACs after the device receives the synchronization pulse.

#### sync\_pulse\_term

*str* – Indicates the name of the internal Synchronization Pulse terminal for the task. This property does not return the name of the source terminal.

# nidaqmx.task.triggers

Represents the trigger configurations for a DAQmx task.

## arm\_start\_trigger

nidaqmx.\_task\_modules.triggering.arm\_start\_trigger.ArmStartTrigger - Gets the arm start trigger configurations for the task.

#### handshake\_trigger

nidaqmx.\_task\_modules.triggering.handshake\_trigger.HandshakeTrigger - Gets the handshake trigger configurations for the task.

# pause\_trigger

nidaqmx.\_task\_modules.triggering.pause\_trigger.PauseTrigger-Gets the pause trigger configurations for the task.

## reference trigger

nidaqmx.\_task\_modules.triggering.reference\_trigger.ReferenceTrigger - Gets the reference trigger configurations for the task.

# start\_trigger

 $nidaqmx.\_task\_modules.triggering.start\_trigger.StartTrigger$  — Gets the start trigger configurations for the task.

## sync\_type

nidaqmx.constants.SyncType—Specifies the role of the device in a synchronized system. Setting this value to SyncType.MASTER or SyncType.SLAVE enables trigger skew correction. If you enable trigger skew correction, set this property to SyncType.MASTER on only one device, and set this property to SyncType.SLAVE on the other devices.

# nidaqmx.task.arm\_start\_trigger

Represents the arm start trigger configurations for a DAQmx task.

## dig\_edge\_dig\_fltr\_enable

bool - Specifies whether to apply the pulse width filter to the signal.

# dig\_edge\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the filter recognizes.

#### dig\_edge\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# dig\_edge\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

#### dig\_edge\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

## dig\_edge\_edge

nidaqmx.constants.Edge - Specifies on which edge of a digital signal to arm the task for a Start Trigger.

## dig\_edge\_src

str – Specifies the name of a terminal where there is a digital signal to use as the source of the Arm Start Trigger.

#### term

str – Indicates the name of the internal Arm Start Trigger terminal for the task. This property does not return the name of the trigger source terminal.

#### trig\_type

nidaqmx.constants.TriggerType – Specifies the type of trigger to use to arm the task for a Start Trigger. If you configure an Arm Start Trigger, the task does not respond to a Start Trigger until the device receives the Arm Start Trigger.

# nidaqmx.task.handshake\_trigger

Represents the handshake trigger configurations for a DAQmx task.

#### interlocked asserted lvl

nidaqmx.constants.Level - Specifies the asserted level of the Handshake Trigger.

# interlocked\_src

str – Specifies the source terminal of the Handshake Trigger.

# trig\_type

 ${\it nidaqmx.constants.TriggerType-Specifies~the~type~of~Handshake~Trigger~to~use.}$ 

# nidaqmx.task.pause\_trigger

Represents the pause trigger configurations for a DAQmx task.

#### anlg\_lvl\_coupling

nidaqmx.constants.Coupling – Specifies the coupling for the source signal of the trigger if the source is a terminal rather than a virtual channel.

# anlg\_lvl\_dig\_fltr\_enable

bool – Specifies whether to apply a digital filter to the digital output of the analog triggering circuitry (the Analog Comparison Event). When enabled, the analog signal must stay above or below the trigger level for the minimum pulse width before being recognized. Use filtering for noisy trigger signals that transition in and out of the hysteresis window rapidly.

## anlg\_lvl\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the filter recognizes.

## anlg\_lvl\_dig\_fltr\_timebase\_rate

float – Specifies in hertz the rate of the digital filter timebase. NI-DAQmx uses this value to compute settings for the filter.

#### anlg\_lvl\_dig\_fltr\_timebase\_src

str – Specifies the terminal of the signal to use as the timebase of the digital filter.

## anlg\_lvl\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

#### anlq lvl hyst

float – Specifies a hysteresis level in the units of the measurement or generation. If anlg\_lvl\_when is ActiveLevel.ABOVE, the trigger does not deassert until the source signal passes below anlg\_lvl\_lvl minus the hysteresis. If anlg\_lvl\_when is ActiveLevel.BELOW, the trigger does not deassert until the source signal passes above anlg\_lvl\_lvl plus the hysteresis. Hysteresis is always enabled. Set this property to a non-zero value to use hysteresis.

# anlg\_lvl\_lvl

*float* – Specifies the threshold at which to pause the task. Specify this value in the units of the measurement or generation. Use **anlg\_lvl\_when** to specify whether the task pauses above or below this threshold.

## anlg\_lvl\_src

str – Specifies the name of a virtual channel or terminal where there is an analog signal to use as the source of the trigger.

## anlg\_lvl\_when

nidaqmx.constants.ActiveLevel - Specifies whether the task pauses above or below the threshold you specify with anlg\_lvl\_lvl.

# anlg\_win\_btm

float – Specifies the lower limit of the window. Specify this value in the units of the measurement or generation.

#### anlg\_win\_coupling

nidaqmx.constants.Coupling – Specifies the coupling for the source signal of the terminal if the source is a terminal rather than a virtual channel.

# anlg\_win\_dig\_fltr\_enable

bool – Specifies whether to apply a digital filter to the digital output of the analog triggering circuitry (the Analog Comparison Event). When enabled, the analog signal must stay within the trigger window for the minimum pulse width before being recognized. Use filtering for noisy trigger signals that transition in and out of the window rapidly.

## anlg\_win\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the filter recognizes.

## anlg\_win\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the digital filter timebase. NI-DAQmx uses this value to compute settings for the filter.

## anlg\_win\_dig\_fltr\_timebase\_src

str – Specifies the terminal of the signal to use as the timebase of the digital filter.

#### anlg\_win\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

#### anlg\_win\_src

str – Specifies the name of a virtual channel or terminal where there is an analog signal to use as the source of the trigger.

### anlg\_win\_top

*float* – Specifies the upper limit of the window. Specify this value in the units of the measurement or generation.

#### anlg\_win\_when

nidaqmx.constants.WindowTriggerCondition2 - Specifies whether the task pauses while the trigger signal is inside or outside the window you specify with anlg\_win\_btm and anlg\_win\_top.

## dig\_lvl\_dig\_fltr\_enable

bool – Specifies whether to apply a digital filter to the trigger signal.

## dig\_lvl\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the filter recognizes.

## dig\_lvl\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# dig\_lvl\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

# dig\_lvl\_dig\_sync\_enable

bool – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

## dig\_lvl\_src

str – Specifies the name of a terminal where there is a digital signal to use as the source of the Pause Trigger.

### dig lvl when

nidaqmx.constants.Level - Specifies whether the task pauses while the signal is high or low.

#### dig pattern pattern

str – Specifies the digital pattern that must be met for the Pause Trigger to occur.

# dig\_pattern\_src

nidaqmx.system.physical\_channel.PhysicalChannel - Specifies the physical channels to use for pattern matching. The order of the physical channels determines the order of the pattern. If a port is included, the lines within the port are in ascending order.

#### dig\_pattern\_when

nidaqmx.constants.DigitalPatternCondition - Specifies if the Pause Trigger occurs when the physical channels specified with **dig\_pattern\_src** match or differ from the digital pattern specified with **dig\_pattern\_pattern**.

#### term

*str* – Indicates the name of the internal Pause Trigger terminal for the task. This property does not return the name of the trigger source terminal.

## trig\_type

nidaqmx.constants.TriggerType - Specifies the type of trigger to use to pause a task.

## nidagmx.task.reference trigger

Represents the reference trigger configurations for a DAQmx task.

#### anlg\_edge\_coupling

nidaqmx.constants.Coupling – Specifies the coupling for the source signal of the trigger if the source is a terminal rather than a virtual channel.

## anlg\_edge\_dig\_fltr\_enable

bool – Specifies whether to apply a digital filter to the digital output of the analog triggering circuitry (the Analog Comparison Event). When enabled, the analog signal must stay above or below the trigger level for the minimum pulse width before being recognized. Use filtering for noisy trigger signals that transition in and out of the hysteresis window rapidly.

## anlg\_edge\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the filter recognizes.

#### anlg\_edge\_dig\_fltr\_timebase\_rate

float – Specifies in hertz the rate of the digital filter timebase. NI-DAQmx uses this value to compute settings for the filter.

# anlg\_edge\_dig\_fltr\_timebase\_src

str – Specifies the terminal of the signal to use as the timebase of the digital filter.

## anlg\_edge\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

# anlg\_edge\_hyst

float – Specifies a hysteresis level in the units of the measurement. If anlg\_edge\_slope is Slope1.RISING, the trigger does not deassert until the source signal passes below anlg\_edge\_lvl minus the hysteresis. If anlg\_edge\_slope is Slope1.FALLING, the trigger does not deassert until the source signal passes above anlg\_edge\_lvl plus the hysteresis. Hysteresis is always enabled. Set this property to a non-zero value to use hysteresis.

# anlg\_edge\_lvl

*float* – Specifies in the units of the measurement the threshold at which the Reference Trigger occurs. Use **anlg\_edge\_slope** to specify on which slope to trigger at this threshold.

# anlg\_edge\_slope

nidaqmx.constants.Slope - Specifies on which slope of the source signal the Reference Trigger occurs.

## anlg\_edge\_src

str – Specifies the name of a virtual channel or terminal where there is an analog signal to use as the source of the Reference Trigger.

# anlg\_win\_btm

float - Specifies the lower limit of the window. Specify this value in the units of the measurement.

# anlg\_win\_coupling

nidaqmx.constants.Coupling – Specifies the coupling for the source signal of the trigger if the source is a terminal rather than a virtual channel.

#### anlg win dig fltr enable

bool – Specifies whether to apply a digital filter to the digital output of the analog triggering circuitry (the Analog Comparison Event). When enabled, the analog signal must stay within the trigger window for the minimum pulse width before being recognized. Use filtering for noisy trigger signals that transition in and out of the window rapidly.

## anlg\_win\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the filter recognizes.

## anlg\_win\_dig\_fltr\_timebase\_rate

float – Specifies in hertz the rate of the digital filter timebase. NI-DAQmx uses this value to compute settings for the filter.

#### anlg\_win\_dig\_fltr\_timebase\_src

str – Specifies the terminal of the signal to use as the timebase of the digital filter.

## anlg\_win\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

#### anlg win src

str – Specifies the name of a virtual channel or terminal where there is an analog signal to use as the source of the Reference Trigger.

#### anlg win top

float - Specifies the upper limit of the window. Specify this value in the units of the measurement.

## anlg\_win\_trig\_when

nidaqmx.constants.WindowTriggerCondition1 - Specifies whether the Reference Trigger occurs when the source signal enters the window or when it leaves the window. Use **anlg\_win\_btm** and **anlg\_win\_top** to specify the window.

## auto\_trig\_enable

bool – Specifies whether to send a software trigger to the device when a hardware trigger is no longer active in order to prevent a timeout.

## auto\_triggered

bool – Indicates whether a completed acquisition was triggered by the auto trigger. If an acquisition has not completed after the task starts, this property returns False. This property is only applicable when **auto\_trig\_enable** is True.

# cfg\_anlg\_edge\_ref\_trig(trigger\_source, pretrigger\_samples, trigger\_slope=<Slope.RISING: 10280>, trigger\_level=0.0)

Configures the task to stop the acquisition when the device acquires all pretrigger samples; an analog signal reaches the level you specify; and the device acquires all post-trigger samples. When you use a Reference Trigger, the default for the read RelativeTo property is **first\_pretrigger\_sample** with a read Offset of 0.

## **Parameters**

- **trigger\_source** (*str*) Is the name of a virtual channel or terminal where there is an analog signal to use as the source of the trigger.
- **pretrigger\_samples** (*int*) Specifies the minimum number of samples to acquire per channel before recognizing the Reference Trigger. The number of post-trigger samples per channel is equal to **number of samples per channel** in the DAQmx Timing function minus **pretrigger\_samples**.

- trigger\_slope (Optional[nidaqmx.constants.Slope]) Specifies on which slope of the signal the Reference Trigger occurs.
- **trigger\_level** (Optional[float]) Specifies at what threshold to trigger. Specify this value in the units of the measurement or generation. Use **trigger\_slope** to specify on which slope to trigger at this threshold.

```
cfg_anlg_window_ref_trig (trigger_source, window_top, window_bottom, pretrigger_samples, trig-
ger_when=<WindowTriggerCondition1.ENTERING_WINDOW: 10163>)
```

Configures the task to stop the acquisition when the device acquires all pretrigger samples; an analog signal enters or leaves a range you specify; and the device acquires all post-trigger samples. When you use a Reference Trigger, the default for the read RelativeTo property is **first\_pretrigger\_sample** with a read Offset of 0.

#### **Parameters**

- **trigger\_source** (str) Is the name of a virtual channel or terminal where there is an analog signal to use as the source of the trigger.
- window\_top (float) Is the upper limit of the window. Specify this value in the units of the measurement or generation.
- window\_bottom (float) Is the lower limit of the window. Specify this value in the units of the measurement or generation.
- **pretrigger\_samples** (*int*) Specifies the minimum number of samples to acquire per channel before recognizing the Reference Trigger. The number of post-trigger samples per channel is equal to **number of samples per channel** in the DAQmx Timing function minus **pretrigger\_samples**.
- trigger\_when (Optional[nidaqmx.constants. WindowTriggerCondition1]) Specifies whether the Reference Trigger occurs when the signal enters the window or when it leaves the window. Use window\_bottom and window\_top to specify the limits of the window.

Configures the task to stop the acquisition when the device acquires all pretrigger samples, detects a rising or falling edge of a digital signal, and acquires all posttrigger samples. When you use a Reference Trigger, the default for the read RelativeTo property is **first\_pretrigger\_sample** with a read Offset of 0.

#### **Parameters**

- **trigger\_source** (*str*) Specifies the name of a terminal where there is a digital signal to use as the source of the trigger.
- **pretrigger\_samples** (*int*) Specifies the minimum number of samples to acquire per channel before recognizing the Reference Trigger. The number of post-trigger samples per channel is equal to **number of samples per channel** in the DAQmx Timing function minus **pretrigger\_samples**.
- **trigger\_edge** (Optional[nidaqmx.constants.Edge]) Specifies on which edge of the digital signal the Reference Trigger occurs.

Configures the task to stop the acquisition when the device acquires all pretrigger samples, matches a digital pattern, and acquires all posttrigger samples. When you use a Reference Trigger, the default for the read RelativeTo property is First PretriggerSample with a read Offset of zero.

#### **Parameters**

- **trigger\_source** (str) Specifies the physical channels to use for pattern matching. The order of the physical channels determines the order of the pattern. If a port is included, the order of the physical channels within the port is in ascending order.
- **trigger\_pattern** (*str*) Specifies the digital pattern that must be met for the trigger to occur.
- **pretrigger\_samples** (*int*) Specifies the minimum number of samples to acquire per channel before recognizing the Reference Trigger. The number of post-trigger samples per channel is equal to **number of samples per channel** in the DAQmx Timing function minus **pretrigger\_samples**.
- trigger\_when (Optional[nidaqmx.constants.
   DigitalPatternCondition]) Specifies the condition under which the trigger occurs.

#### delay

*float* – Specifies in seconds the time to wait after the device receives the Reference Trigger before switching from pretrigger to posttrigger samples.

#### dig\_edge\_dig\_fltr\_enable

bool – Specifies whether to apply a digital filter to the trigger signal.

# dig\_edge\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the filter recognizes.

## dig\_edge\_dig\_fltr\_timebase\_rate

float – Specifies in hertz the rate of the digital filter timebase. NI-DAQmx uses this value to compute settings for the filter.

## dig\_edge\_dig\_fltr\_timebase\_src

str – Specifies the terminal of the signal to use as the timebase of the digital filter.

## dig\_edge\_dig\_sync\_enable

bool – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

#### dig\_edge\_edge

nidaqmx.constants.Edge - Specifies on what edge of a digital pulse the Reference Trigger occurs.

## dig edge src

str – Specifies the name of a terminal where there is a digital signal to use as the source of the Reference Trigger.

#### dig pattern pattern

str – Specifies the digital pattern that must be met for the Reference Trigger to occur.

### dig\_pattern\_src

nidaqmx.system.physical\_channel.PhysicalChannel - Specifies the physical channels to use for pattern matching. The order of the physical channels determines the order of the pattern. If a port is included, the order of the physical channels within the port is in ascending order.

#### dig\_pattern\_trig\_when

nidaqmx.constants.DigitalPatternCondition - Specifies whether the Reference Trigger occurs when the physical channels specified with **dig\_pattern\_src** match or differ from the digital pattern specified with **dig\_pattern\_pattern**.

# disable\_ref\_trig()

Disables reference triggering for the measurement.

## pretrig\_samples

*int* – Specifies the minimum number of pretrigger samples to acquire from each channel before recognizing the reference trigger. Post-trigger samples per channel are equal to **samp\_quant\_samp\_per\_chan** minus the number of pretrigger samples per channel.

#### term

str – Indicates the name of the internal Reference Trigger terminal for the task. This property does not return the name of the trigger source terminal.

## trig\_type

nidaqmx.constants.TriggerType - Specifies the type of trigger to use to mark a reference point for the measurement.

# nidaqmx.task.start\_trigger

Represents the start trigger configurations for a DAQmx task.

## anlg\_edge\_coupling

nidaqmx.constants.Coupling – Specifies the coupling for the source signal of the trigger if the source is a terminal rather than a virtual channel.

## anlg\_edge\_dig\_fltr\_enable

bool – Specifies whether to apply a digital filter to the digital output of the analog triggering circuitry (the Analog Comparison Event). When enabled, the analog signal must stay above or below the trigger level for the minimum pulse width before being recognized. Use filtering for noisy trigger signals that transition in and out of the hysteresis window rapidly.

# anlg\_edge\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the filter recognizes.

## anlg\_edge\_dig\_fltr\_timebase\_rate

float – Specifies in hertz the rate of the digital filter timebase. NI-DAQmx uses this value to compute settings for the filter.

### anlg\_edge\_dig\_fltr\_timebase\_src

str – Specifies the terminal of the signal to use as the timebase of the digital filter.

# anlg\_edge\_dig\_sync\_enable

bool – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

#### anlg\_edge\_hyst

float – Specifies a hysteresis level in the units of the measurement or generation. If anlg\_edge\_slope is Slope1.RISING, the trigger does not deassert until the source signal passes below anlg\_edge\_lvl minus the hysteresis. If anlg\_edge\_slope is Slope1.FALLING, the trigger does not deassert until the source signal passes above anlg\_edge\_lvl plus the hysteresis. Hysteresis is always enabled. Set this property to a non-zero value to use hysteresis.

## anlg\_edge\_lvl

float – Specifies at what threshold in the units of the measurement or generation to start acquiring or generating samples. Use **anlg\_edge\_slope** to specify on which slope to trigger on this threshold.

# anlg\_edge\_slope

nidaqmx.constants.Slope - Specifies on which slope of the trigger signal to start acquiring or generating samples.

# anlg\_edge\_src

str – Specifies the name of a virtual channel or terminal where there is an analog signal to use as the source of the Start Trigger.

#### anlg\_win\_btm

*float* – Specifies the lower limit of the window. Specify this value in the units of the measurement or generation.

## anlg\_win\_coupling

nidaqmx.constants.Coupling – Specifies the coupling for the source signal of the trigger if the source is a terminal rather than a virtual channel.

# anlg\_win\_dig\_fltr\_enable

bool – Specifies whether to apply a digital filter to the digital output of the analog triggering circuitry (the Analog Comparison Event). When enabled, the analog signal must stay within the trigger window for the minimum pulse width before being recognized. Use filtering for noisy trigger signals that transition in and out of the window rapidly.

## anlg\_win\_dig\_fltr\_min\_pulse\_width

float – Specifies in seconds the minimum pulse width the filter recognizes.

#### anlg\_win\_dig\_fltr\_timebase\_rate

float – Specifies in hertz the rate of the digital filter timebase. NI-DAQmx uses this value to compute settings for the filter.

#### anlg win dig fltr timebase src

str – Specifies the terminal of the signal to use as the timebase of the digital filter.

## anlg win dig sync enable

bool – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device.

# anlg\_win\_src

str – Specifies the name of a virtual channel or terminal where there is an analog signal to use as the source of the Start Trigger.

## anlg\_win\_top

*float* – Specifies the upper limit of the window. Specify this value in the units of the measurement or generation.

## anlg\_win\_trig\_when

nidaqmx.constants.WindowTriggerCondition1 - Specifies whether the task starts acquiring or generating samples when the signal enters or leaves the window you specify with anlg\_win\_btm and anlg\_win\_top.

# cfg\_anlg\_edge\_start\_trig(trigger\_source=u'', trigger\_slope=<Slope.RISING: 10280>, trigger\_level=0.0)

Configures the task to start acquiring or generating samples when an analog signal crosses the level you specify.

## **Parameters**

- **trigger\_source** (Optional[str]) Is the name of a virtual channel or terminal where there is an analog signal to use as the source of the trigger.
- **trigger\_slope** (Optional[nidaqmx.constants.Slope]) Specifies on which slope of the signal to start acquiring or generating samples when the signal crosses **trigger\_level**.
- **trigger\_level** (Optional[float]) Specifies at what threshold to start acquiring or generating samples. Specify this value in the units of the measurement or generation. Use **trigger\_slope** to specify on which slope to trigger at this threshold.

```
cfg_anlg_window_start_trig (window_top, window_bottom, trigger_source=u'', trig-
ger_when=<WindowTriggerCondition1.ENTERING_WINDOW:
10163>)
```

Configures the task to start acquiring or generating samples when an analog signal enters or leaves a range you specify.

#### **Parameters**

- window\_top (float) Is the upper limit of the window. Specify this value in the units of the measurement or generation.
- window\_bottom (float) Is the lower limit of the window. Specify this value in the units of the measurement or generation.
- **trigger\_source** (Optional[str]) Is the name of a virtual channel or terminal where there is an analog signal to use as the source of the trigger.
- trigger\_when (Optional[nidaqmx.constants. WindowTriggerCondition1]) Specifies whether the task starts measuring or generating samples when the signal enters the window or when it leaves the window. Use window\_bottom and window\_top to specify the limits of the window.

# cfg\_dig\_edge\_start\_trig(trigger\_source, trigger\_edge=<Edge.RISING: 10280>)

Configures the task to start acquiring or generating samples on a rising or falling edge of a digital signal.

#### **Parameters**

- **trigger\_source** (*str*) Specifies the name of a terminal where there is a digital signal to use as the source of the trigger.
- **trigger\_edge** (Optional[nidaqmx.constants.Edge]) Specifies on which edge of the digital signal to start acquiring or generating samples.

```
cfg_dig_pattern_start_trig(trigger_source, trigger_pattern, trig-
ger_when=<DigitalPatternCondition.PATTERN_MATCHES:
10254>)
```

Configures a task to start acquiring or generating samples when a digital pattern is matched.

## **Parameters**

- **trigger\_source** (*str*) Specifies the physical channels to use for pattern matching. The order of the physical channels determines the order of the pattern. If a port is included, the order of the physical channels within the port is in ascending order.
- trigger\_pattern (str) Specifies the digital pattern that must be met for the trigger to occur.
- trigger\_when (Optional[nidaqmx.constants.
   DigitalPatternCondition]) Specifies the condition under which the trigger occurs.

# delay

*float* – Specifies an amount of time to wait after the Start Trigger is received before acquiring or generating the first sample. This value is in the units you specify with **delay\_units**.

#### delay\_units

nidagmx.constants.DigitalWidthUnits - Specifies the units of delay.

# dig\_edge\_dig\_fltr\_enable

bool – Specifies whether to apply a digital filter to the trigger signal.

# dig\_edge\_dig\_fltr\_min\_pulse\_width

*float* – Specifies in seconds the minimum pulse width the filter recognizes.

## dig\_edge\_dig\_fltr\_timebase\_rate

*float* – Specifies in hertz the rate of the pulse width filter timebase. NI-DAQmx uses this value to compute settings for the filter.

#### dig\_edge\_dig\_fltr\_timebase\_src

str – Specifies the input terminal of the signal to use as the timebase of the pulse width filter.

## dig\_edge\_dig\_sync\_enable

*bool* – Specifies whether to synchronize recognition of transitions in the signal to the internal timebase of the device. If you set this property to True, the device does not recognize and act upon the trigger until the next pulse of the internal timebase.

## dig\_edge\_edge

nidaqmx.constants.Edge – Specifies on which edge of a digital pulse to start acquiring or generating samples.

## dig\_edge\_src

str – Specifies the name of a terminal where there is a digital signal to use as the source of the Start Trigger.

## dig\_pattern\_pattern

str – Specifies the digital pattern that must be met for the Start Trigger to occur.

#### dig\_pattern\_src

nidaqmx.system.physical\_channel.PhysicalChannel - Specifies the physical channels to use for pattern matching. The order of the physical channels determines the order of the pattern. If a port is included, the order of the physical channels within the port is in ascending order.

## dig\_pattern\_trig\_when

nidaqmx.constants.DigitalPatternCondition - Specifies whether the Start Trigger occurs when the physical channels specified with **dig\_pattern\_src** match or differ from the digital pattern specified with **dig\_pattern\_pattern**.

# disable\_start\_trig()

Configures the task to start acquiring or generating samples immediately upon starting the task.

## retriggerable

bool – Specifies whether a finite task resets and waits for another Start Trigger after the task completes. When you set this property to True, the device performs a finite acquisition or generation each time the Start Trigger occurs until the task stops. The device ignores a trigger if it is in the process of acquiring or generating signals.

#### term

str – Indicates the name of the internal Start Trigger terminal for the task. This property does not return the name of the trigger source terminal.

#### trig type

nidaqmx.constants.TriggerType - Specifies the type of trigger to use to start a task.

# nidaqmx.types

```
class nidaqmx.types.AOExpirationState (physical_channel, expiration_state, output_type)
    Bases: tuple
```

# expiration\_state

Alias for field number 1

# output\_type

Alias for field number 2

8.8. nidagmx.types 247

```
physical channel
          Alias for field number 0
class nidaqmx.types.AOPowerUpState (physical_channel, power_up_state, channel_type)
     Bases: tuple
     channel_type
         Alias for field number 2
     physical channel
         Alias for field number 0
     power_up_state
          Alias for field number 1
class nidaqmx.types.CDAQSyncConnection(output_port, input_port)
     Bases: tuple
     input_port
         Alias for field number 1
     output port
         Alias for field number 0
class nidagmx.types.COExpirationState (physical_channel, expiration_state)
     Bases: tuple
     expiration state
         Alias for field number 1
     physical_channel
         Alias for field number 0
class nidaqmx.types.CtrFreq(freq, duty_cycle)
     Bases: tuple
     duty_cycle
          Alias for field number 1
     freq
          Alias for field number 0
class nidaqmx.types.CtrTick (high_tick, low_tick)
     Bases: tuple
     high_tick
         Alias for field number 0
     low tick
          Alias for field number 1
class nidaqmx.types.CtrTime (high_time, low_time)
     Bases: tuple
     high_time
         Alias for field number 0
     low time
         Alias for field number 1
class nidaqmx.types.DOExpirationState (physical_channel, expiration_state)
     Bases: tuple
```

### expiration\_state

Alias for field number 1

#### physical\_channel

Alias for field number 0

class nidaqmx.types.DOPowerUpState(physical\_channel, power\_up\_state)

Bases: tuple

#### physical channel

Alias for field number 0

#### power\_up\_state

Alias for field number 1

class nidaqmx.types.DOResistorPowerUpState(physical\_channel, power\_up\_state)

Bases: tuple

#### physical\_channel

Alias for field number 0

#### power up state

Alias for field number 1

### nidaqmx.utils

#### nidaqmx.utils.flatten\_channel\_string(channel\_names)

Converts a list of channel names to a comma-delimited list of names.

You can use this method to convert a list of physical or virtual channel names to a single string prior to using the DAQmx Create Channel methods or instantiating a DAQmx Task object.

**Parameters channel\_names** (List[str]) – The list of physical or virtual channel names.

**Returns** The resulting comma-delimited list of physical or virtual channel names.

Return type str

#### nidaqmx.utils.unflatten\_channel\_string(channel\_names)

Converts a comma-delimited list of channel names to a list of names.

You can use this method to convert a comma-delimited list or range of physical or virtual channels into a list of physical or virtual channel names.

**Parameters** channel\_names (str) – The list or range of physical or virtual channels.

**Returns** The list of physical or virtual channel names. Each element of the list contains a single channel.

Return type List[str]

8.9. nidagmx.utils 249

250 Chapter 8. License

# CHAPTER 9

# Indices and Tables

- genindex
- modindex

## Python Module Index

```
n
                                                                                                            237
nidaqmx._task_modules.ai_channel_collectmidaqmx._task_modules.triggering.reference_trigger,
                                                                                                            240
nidaqmx._task_modules.ao_channel_collecthidaqmx._task_modules.triggering.start_trigger,
                                                                                                           244
\verb|nidaqmx._task_modules.channel_collection|, \verb|nidaqmx._task_modules.triggers|, 236|
                                                                                            nidaqmx.constants, 17
\verb|nidaqmx._task_modules.channels.ai_channel_idaqmx.errors|, 51
                                                                                            nidagmx.scale, 52
\verb|nidaqmx._task_modules.channels.ao_channel_idaqmx.stream_readers|, 56
                                                                                            nidagmx.stream_writers,77
                                                                                            nidaqmx.system._collections.device_collection,
nidaqmx._task_modules.channels.channel,
nidaqmx._task_modules.channels.ci_channelpidaqmx.system._collections.persisted_channel_collections.
nidaqmx._task_modules.channels.co_channelpidaqmx.system._collections.persisted_scale_collections.
nidaqmx._task_modules.channels.di_channelpidaqmx.system._collections.persisted_task_collect
nidaqmx._task_modules.channels.do_channe1;daqmx.system._collections.physical_channel_collections
nidaqmx._task_modules.ci_channel_collecthidaqmx.system._watchdog_modules.expiration_state,
\verb| nidaqmx.\_task\_modules.co\_channel\_collect \verb| Pidaqmx.system.\_watchdog\_modules.expiration\_states\_ | likely a state of the collect \verb| Pidaqmx.system.\_watchdog\_modules.expiration\_states\_ | likely a state of the collect \verb| Pidaqmx.system.\_watchdog\_modules.expiration\_states\_ | likely a state of the collect \verb| Pidaqmx.system.\_watchdog\_modules.expiration\_states\_ | likely a state of the collect \verb| Pidaqmx.system.\_watchdog\_modules.expiration\_states\_ | likely a state of the collect \verb| Pidaqmx.system.\_watchdog\_modules.expiration\_states\_ | likely a state of the collect \verb| Pidaqmx.system.\_watchdog\_modules.expiration\_states\_ | likely a states\_ | likely a states
\verb|nidaqmx._task_modules.di_channel_collect \verb|Pidaqmx.system.device|, 98|
                                                                                            nidaqmx.system.physical_channel, 104
nidaqmx._task_modules.do_channel_collecthod,qmx.system.storage.persisted_channel,
                                                                                            nidagmx.system.storage.persisted_scale,
nidaqmx._task_modules.export_signals,
                                                                                            nidaqmx.system.storage.persisted_task,
nidagmx. task modules.in stream, 220
                                                                                                           108
nidaqmx._task_modules.out_stream, 226
                                                                                            nidaqmx.system.system, 91
nidaqmx._task_modules.timing, 229
nidaqmx._task_modules.triggering.arm_staptdeqmxqeystem.watchdog, 109
                                                                                            nidaqmx.task, 112
nidaqmx._task_modules.triggering.handshakedegmageypes,247
                                                                                            nidagmx.utils, 249
nidagmx._task_modules.triggering.pause_trigger,
```

254 Python Module Index

Symbols	(nidaqmx.constants.UsageTypeAI attribute), 47
init() (nidaqmx.scale.Scale method), 52	ACCELERATION_ACCELEROMETER_CURRENT_INPUT
init () (nidaamx system device Device method) 98	(nidaqmx.constants.UsageTypeAI attribute), 47
init() (nidaqmx.system.physical_channel.PhysicalChan	nneCELERATION_CHARGE
method) 104	(ilidaqilix.collstants.osageTypeAT attribute), 4/
init() (nidaqmx.system.storage.persisted_channel.Pers	istedelsensitivityUnits (class in nidaqmx.constants), 18
method) 107	Accelonits (class in indaquix.constants), 16
init() (nidaqmx.system.storage.persisted_scale.Persisted_	eascessory_insertion_or_removal_detected
method), 107	(mdaqmxtask_modules.m_stream.mstream
init() (nidaqmx.system.storage.persisted_task.Persisted	dTask attribute), 220
method), 108	accessory_insertion_or_removal_detected
init() (nidaqmx.system.watchdog.WatchdogTask	(nidaqmxtask_modules.out_stream.OutStream
method), 109	attribute), 226
init() (nidaqmx.task.Task method), 112	accessory_product_nums
weakref (nidaqmx.scale.Scale attribute), 52	(nidaqmx.system.device.Device attribute),
weakref (nidaqmx.system.device.Device attribute),	98
98	accessory_product_types
weakref (nidaqmx.system.physical_channel.PhysicalC	Channel (nidaqmx.system.device.Device attribute),
attribute), 104	
weakref (nidaqmx.system.storage.persisted_channel.P	attribute), 99
attribute), 107	
weakref (nidaqmx.system.storage.persisted_scale.Pers	ACQUIRED_INTO_BUFFER
attribute), 108	( ) 1 E NO 1 E M
weakref (nidaqmx.system.storage.persisted_task.Persi attribute), 108	attribute), 26
weakref (nidaqmx.system.watchdog.WatchdogTask	AcquisitionType (class in nidaqmx.constants), 18
attribute), 109	Action (class in nidaqmx.constants), 19
weakref (nidaqmx.task.Task attribute), 112	ACTIVE (nidaqmx.constants.ActiveOrInactiveEdgeSelection attribute), 19
A	ACTIVE_DRIVE (nidaqmx.constants.DigitalDriveType
A (nidaqmx.constants.ShuntCalSelect attribute), 39	attribute), 25
A (nidaqmx.constants.ShuntResistorSelect attribute), 39	ACTIVE_HIGH (nidaqmx.constants.Polarity attribute),
AAND_B (nidaqmx.constants.ShuntCalSelect attribute),	33
39	ACTIVE_LOW (nidaqmx.constants.Polarity attribute),
ABOVE (nidaqmx.constants.ActiveLevel attribute), 19	33
AC (nidaqmx.constants.Coupling attribute), 24	ActiveLevel (class in nidaqmx.constants), 19
AccelChargeSensitivityUnits (class in	ActiveOrInactiveEdgeSelection (class in
nidaqmx.constants), 18	nidaqmx.constants), 19
ACCELERATION_4_WIRE_DC_VOLTAGE	ADCTimingMode (class in nidaqmx.constants), 17
	add_ai_accel_4_wire_dc_voltage_chan()

```
(nidaqmx._task_modules.ai_channel_collection.AIChannel@nidaction._task_modules.ai_channel_collection.AIChannelCollection.
                                                     method), 157
                                                                                                                                                                                                                                                                                                                                                                                        method), 175
add_ai_accel_chan() (nidaqmx._task_modules.ai_channel_cadlectaio_nta_lChan() enCdralderatio_ntask_modules.ai_channel_collection.AIChan()
                                                      method), 158
                                                                                                                                                                                                                                                                                                                                                                                        method), 176
                                                                                                                                                                                                                                                                                                                                  add_ai_strain_gage_chan()
 add_ai_accel_charge_chan()
                                                     (nidagmx, task modules.ai channel collection.AIChannel@pidaction. task modules.ai channel collection.AIChannelCollec
                                                      method), 158
                                                                                                                                                                                                                                                                                                                                                                                        method), 177
add_ai_bridge_chan() (nidaqmx._task_modules.ai_channel_auddleatitoemapl@biaitmeitCadhsctiochan()
                                                      method), 159
                                                                                                                                                                                                                                                                                                                                                                                         (nidagmx. task modules.ai channel collection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.A
add_ai_charge_chan() (nidaqmx._task_modules.ai_channel_collectionmet6bd)nnelCollection
                                                                                                                                                                                                                                                                                                                                  add_ai_thrmcpl_chan() (nidaqmx._task_modules.ai_channel_collection.AIG
                                                      method), 160
add_ai_current_chan() (nidaqmx._task_modules.ai_channel_collectionmAtlCollection
                                                     method), 161
                                                                                                                                                                                                                                                                                                                                  add_ai_thrmstr_chan_iex()
add_ai_current_rms_chan()
                                                                                                                                                                                                                                                                                                                                                                                        (nidagmx._task_modules.ai_channel_collection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.A
                                                    (nidaqmx._task_modules.ai_channel_collection.AIChannel@odtbcdon179
                                                      method), 161
                                                                                                                                                                                                                                                                                                                                  add_ai_thrmstr_chan_vex()
add_ai_force_bridge_polynomial_chan()
                                                                                                                                                                                                                                                                                                                                                                                         (nidagmx._task_modules.ai_channel_collection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.A
                                                     (nidagmx._task_modules.ai_channel_collection.AIChannel@odthcdon180
                                                                                                                                                                                                                                                                                                                                  add_ai_torque_bridge_polynomial_chan()
                                                     method), 162
                                                                                                                                                                                                                                                                                                                                                                                          (nidaqmx._task_modules.ai_channel_collection.AIChannelCollection)
add_ai_force_bridge_table_chan()
                                                     (nidaqmx._task_modules.ai_channel_collection.AIChannel@odthcdon181
                                                                                                                                                                                                                                                                                                                                  add_ai_torque_bridge_table_chan()
add_ai_force_bridge_two_point_lin_chan()
                                                                                                                                                                                                                                                                                                                                                                                         (nidaqmx._task_modules.ai_channel_collection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.A
                                                     (nidagmx, task modules.ai channel collection.AIChannel@odtbctdon182
                                                      method), 164
                                                                                                                                                                                                                                                                                                                                  add_ai_torque_bridge_two_point_lin_chan()
add_ai_force_iepe_chan()
                                                                                                                                                                                                                                                                                                                                                                                         (nidagmx. task modules.ai channel collection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.A
                                                      (nidaqmx._task_modules.ai_channel_collection.AIChannel@odthcdd)n183
                                                      method), 166
                                                                                                                                                                                                                                                                                                                                  add_ai_velocity_iepe_chan()
                                                                                                                                                                                                                                                                                                                                                                                         (nidagmx._task_modules.ai_channel_collection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.A
add_ai_freq_voltage_chan()
                                                    (nidaqmx._task_modules.ai_channel_collection.AIChannel@odthctdon184
                                                      method), 167
                                                                                                                                                                                                                                                                                                                                  add_ai_voltage_chan() (nidaqmx._task_modules.ai_channel_collection.AIC
add_ai_microphone_chan()
                                                                                                                                                                                                                                                                                                                                                                                         method), 185
                                                    (nidagmx._task_modules.ai_channel_collection.Addhaminel@talbectionan_with_excit()
                                                     method), 167
                                                                                                                                                                                                                                                                                                                                                                                        (nidagmx._task_modules.ai_channel_collection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.A
 add_ai_pos_eddy_curr_prox_probe_chan()
                                                                                                                                                                                                                                                                                                                                                                                         method), 186
                                                      (nidaqmx._task_modules.ai_channel_collection.Addhaminedoltalections_chan()
                                                     method), 168
                                                                                                                                                                                                                                                                                                                                                                                        (nidagmx. task modules.ai channel collection.AIChannelCollection)
add\_ai\_pos\_lvdt\_chan() \ (nidaqmx.\_task\_modules.ai\_channel\_collectiomet \verb|ALG|) \ and \verb|Algebra| \ and \ algebra| \ and \ algebra| \ and \ algebra| \ al
                                                                                                                                                                                                                                                                                                                                  add_ao_current_chan() (nidaqmx._task_modules.ao_channel_collection.AC
                                                      method), 169
add\_ai\_pos\_rvdt\_chan() \\ (nidaqmx.\_task\_modules.ai\_channel\_collectionet \verb|Abd||) \\ and all lectionet | add_ai\_pos\_rvdt\_chan() \\ (nidaqmx.\_task\_modules.ai\_channel\_collectionet | add_ai\_pos\_rvdt\_channel\_collectionet | add_ai\_pos\_rvdt\_chan() \\ (nidaqmx.\_task\_modules.ai\_channel\_collectionet | add_ai\_pos\_rvdt\_chan() \\ (nidaqmx.\_task\_modules.ai\_channel\_collectionet | add_ai\_pos\_rvdt\_channel\_collectionet | add_ai\_pos\_rvdt\_chan() \\ (nidaqmx.\_task\_modules.ai\_pos\_rvdt\_channel\_collectionet | add_ai\_pos\_rvdt\_channel\_collectionet | add_ai\_pos\_rvdt\_channe
                                                                                                                                                                                                                                                                                                                                  add_ao_func_gen_chan()
                                                     method), 170
add_ai_pressure_bridge_polynomial_chan()
                                                                                                                                                                                                                                                                                                                                                                                         (nidagmx. task modules.ao channel collection.AOChannelColle
                                                      (nidagmx. task modules.ai channel collection.AIChannel@odtbctdon201
                                                      method), 171
                                                                                                                                                                                                                                                                                                                                  add_ao_voltage_chan() (nidaqmx._task_modules.ao_channel_collection.AC
add_ai_pressure_bridge_table_chan()
                                                                                                                                                                                                                                                                                                                                                                                         method), 202
                                                     (nidagmx._task_modules.ai_channel_collection.Adddhacated_Costhectionnnection()
                                                      method), 172
                                                                                                                                                                                                                                                                                                                                                                                         (nidaqmx.system.system.System
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          method),
add_ai_pressure_bridge_two_point_lin_chan()
                                                                                                                                                                                                                                                                                                                                                                                          91
                                                     (nidaqmx._task_modules.ai_channel_collection.Adddhaoineal@oldenctoder_chan()
                                                     method), 173
                                                                                                                                                                                                                                                                                                                                                                                         (nidaqmx._task_modules.ci_channel_collection.CIChannelCollection.
                                                                                                                                                                                                                                                                                                                                                                                         method), 203
add_ai_resistance_chan()
                                                    (nidaqmx._task_modules.ai_channel_collection.Adddhaoin_eddcollectiony_chan()
                                                                                                                                                                                                                                                                                                                                                                                         (nidagmx. task modules.ci channel collection.CIChannelCollec
                                                     method), 174
 add_ai_rosette_strain_gage_chan()
                                                                                                                                                                                                                                                                                                                                                                                         method), 204
```

(nidaqmx.\_task\_modules.ai\_channel\_collection.AIChannelCollection)

```
(nidagmx. task modules.ci channel collection.CIChannel@odtbection187
                                                                                                                                                                                                                                                                                                                                                                                 add teds ai bridge chan()
                                                              method), 204
add_ci_duty_cycle_chan()
                                                                                                                                                                                                                                                                                                                                                                                                                                                 (nidaqmx._task_modules.ai_channel_collection.AIChannelCollection)
                                                              (nidaqmx._task_modules.ci_channel_collection.CIChannel@odlection188
                                                             method), 205
                                                                                                                                                                                                                                                                                                                                                                                 add teds ai current chan()
add_ci_freq_chan() (nidaqmx._task_modules.ci_channel_collection.C[filldannetCollection.dules.ai_channel_collection.AIChannelCollection.AIChannelCollection.dules.ai_channel_collection.AIChannelCollection.AIChannelCollection.dules.ai_channel_collection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollec
                                                                                                                                                                                                                                                                                                                                                                                                                                                method), 189
                                                              method), 205
add_ci_gps_timestamp_chan()
                                                                                                                                                                                                                                                                                                                                                                                 add_teds_ai_force_bridge_chan()
                                                              (nidaqmx._task_modules.ci_channel_collection.CIChannel@nldtaqtion._task_modules.ai_channel_collection.AIChannelCollection.
                                                              method), 206
                                                                                                                                                                                                                                                                                                                                                                                                                                                method), 190
add_ci_lin_encoder_chan()
                                                                                                                                                                                                                                                                                                                                                                                 add_teds_ai_force_iepe_chan()
                                                              (nidagmx_task_modules.ci_channel_collection.CIChannel@nldtaction._task_modules.ai_channel_collection.AIChannelCollec
                                                                                                                                                                                                                                                                                                                                                                                                                                               method), 190
                                                              method), 207
add_ci_lin_velocity_chan()
                                                                                                                                                                                                                                                                                                                                                                                 add_teds_ai_microphone_chan()
                                                             (nidaqmx.\_task\_modules.ci\_channel\_collection.CIC hannel \verb|Qnid=| adjoint| adjoint|
                                                              method), 208
                                                                                                                                                                                                                                                                                                                                                                                                                                                method), 191
add_ci_period_chan() (nidaqmx._task_modules.ci_channel_anddletctdonaClfcthant() on
                                                                                                                                                                                                                                                                                                                                                                                                                                                (nidaqmx.\_task\_modules.ai\_channel\_collection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection
                                                              method), 208
add_ci_pulse_chan_freq()
                                                                                                                                                                                                                                                                                                                                                                                                                                                method), 192
                                                             (nidaqmx._task_modules.ci_channel_collection.Clack_atentslCableostionvdt_chan()
                                                              method), 209
                                                                                                                                                                                                                                                                                                                                                                                                                                                (nidaqmx._task_modules.ai_channel_collection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.A
add_ci_pulse_chan_ticks()
                                                                                                                                                                                                                                                                                                                                                                                                                                                method), 193
                                                             (nidagmx. task modules.ci channel collection.Cla@datendsl@oblectisoure bridge chan()
                                                                                                                                                                                                                                                                                                                                                                                                                                                (nidaqmx._task_modules.ai_channel_collection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.A
                                                              method), 210
add_ci_pulse_chan_time()
                                                                                                                                                                                                                                                                                                                                                                                                                                                method), 194
                                                              (nidagmx._task_modules.ci_channel_collection.Clack_atents|CollectionLessixtance_chan()
                                                              method), 210
                                                                                                                                                                                                                                                                                                                                                                                                                                                (nidaqmx._task_modules.ai_channel_collection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.A
                                                                                                                                                                                                                                                                                                                                                                                                                                                method), 194
add_ci_pulse_width_chan()
                                                             (nidaqmx._task_modules.ci_channel_collection.CatchatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedslCablatedsl
                                                              method), 211
                                                                                                                                                                                                                                                                                                                                                                                                                                                method), 195
add_ci_semi_period_chan()
                                                                                                                                                                                                                                                                                                                                                                                  add_teds_ai_strain_gage_chan()
                                                            (nidaqmx._task_modules.ci_channel_collection.CIChannel@pldaction._task_modules.ai_channel_collection.AIChannelCollection.
                                                             method), 212
                                                                                                                                                                                                                                                                                                                                                                                                                                                method), 196
 add_ci_two_edge_sep_chan()
                                                                                                                                                                                                                                                                                                                                                                                 add teds ai thrmcpl chan()
                                                              (nidagmx._task_modules.ci_channel_collection.CIChannel@nldtaqtion._task_modules.ai_channel_collection.AIChannelCollec
                                                             method), 212
                                                                                                                                                                                                                                                                                                                                                                                                                                               method), 196
add_co_pulse_chan_freq()
                                                                                                                                                                                                                                                                                                                                                                                 add_teds_ai_thrmstr_chan_iex()
                                                              (nidagmx._task_modules.co_channel_collection.COChannel@Giollagotivon_task_modules.ai_channel_collection.AIChannelCollec
                                                              method), 213
                                                                                                                                                                                                                                                                                                                                                                                                                                                method), 197
add_co_pulse_chan_ticks()
                                                                                                                                                                                                                                                                                                                                                                                 add teds ai thrmstr chan vex()
                                                              (nidagmx._task_modules.co_channel_collection.COChannel@Giollagotivon_task_modules.ai_channel_collection.AIChannelCollec
                                                              method), 214
                                                                                                                                                                                                                                                                                                                                                                                                                                                method), 198
add_co_pulse_chan_time()
                                                                                                                                                                                                                                                                                                                                                                                 add_teds_ai_torque_bridge_chan()
                                                             (nidaqmx.\_task\_modules.co\_channel\_collection.COC hannel\_foolbegrativon\_task\_modules.ai\_channel\_collection.AIC hannelCollection.AIC hannelCollection.COC hannel\_collection.AIC hannelCollection.AIC h
                                                              method), 214
                                                                                                                                                                                                                                                                                                                                                                                                                                                method), 198
add_di_chan() (nidagmx._task_modules.di_channel_collectionddDtddhannelololddleetidnan()
                                                              method), 215
                                                                                                                                                                                                                                                                                                                                                                                                                                                (nidaqmx._task_modules.ai_channel_collection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.A
add\_do\_chan() \ (nidaqmx.\_task\_modules.do\_channel\_collection. DOC \textbf{harthed} \textbf{O}) oll \textbf{0} \textbf{0} tion
                                                              method), 216
                                                                                                                                                                                                                                                                                                                                                                                  add_teds_ai_voltage_chan_with_excit()
                                                                                                                                                                                                                                                                                                                                                                                                                                                (nidaqmx._task_modules.ai_channel_collection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.AIChannelCollection.A
add_global_channels() (nidaqmx.task.Task method), 112
add_network_device() (nidaqmx.system.device.Device
                                                                                                                                                                                                                                                                                                                                                                                                                                                method), 200
                                                                                                                                                                                                                                                                                                                                                                                  ADV_CMPLT_EVENT (nidaqmx.constants.Signal at-
                                                             static method), 99
```

add\_ci\_count\_edges\_chan()

add\_teds\_ai\_accel\_chan()

Index 257

tribute), 39

```
adv_cmplt_event_delay (nidaqmx._task_modules.export_signiabcExportfsigniabcqmx._task_modules.channels.ai_channels.AIChannel
                   attribute), 216
                                                                                                                                      attribute), 120
adv cmplt event output term
                                                                                                                  ai acceld b ref (nidagmx. task modules.channels.ai channel.AIChannel
                   (nidaqmx._task_modules.export_signals.ExportSignals
                                                                                                                                      attribute), 120
                   attribute), 217
                                                                                                                  ai adc custom timing mode
adv cmplt event pulse polarity
                                                                                                                                      (nidagmx. task modules.channels.ai channel.AIChannel
                   (nidagmx. task modules.export signals.ExportSignals
                                                                                                                                      attribute), 120
                   attribute), 217
                                                                                                                  ai adc timing mode (nidagmx. task modules.channels.ai channel.AIChan
adv_cmplt_event_pulse_width
                                                                                                                                      attribute), 120
                   (nidaqmx._task_modules.export_signals.ExportSignalsten (nidaqmx._task_modules.channels.ai_channel.AIChannel
                   attribute), 217
                                                                                                                                      attribute), 120
adv_trig_output_term (nidaqmx._task_modules.export_signals.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channels.air_channel
                   attribute), 217
                                                                                                                                      attribute), 120
adv_trig_pulse_polarity (nidaqmx._task_modules.export_signia_tsvEragontsSignia_tsviet (nidaqmx._task_modules.channels.ai_channels.ai_channels.ai
                   attribute), 217
                                                                                                                                      attribute), 120
adv_trig_pulse_width (nidaqmx._task_modules.export_signals_bixdoctSixlankse_coarse_pot
                                                                                                                                      (nidaqmx._task_modules.channels.ai_channel.AIChannel
                   attribute), 217
adv_trig_pulse_width_units
                                                                                                                                      attribute), 120
                   (nidagmx. task modules.export signals.ExportSignabsidge balance fine pot
                   attribute), 217
                                                                                                                                      (nidagmx. task modules.channels.ai channel.AIChannel
ADVANCE (nidaqmx.constants.TriggerUsage attribute),
                                                                                                                                      attribute), 120
                                                                                                                  ai bridge cfg (nidagmx. task modules.channels.ai channel.AIChannel
ADVANCE_TRIGGER (nidaqmx.constants.Signal at-
                                                                                                                                      attribute), 120
                   tribute), 39
                                                                                                                  ai bridge electrical units
ADVANCE TRIGGER (nidagmx.constants.SoftwareTrigger
                                                                                                                                      (nidagmx. task modules.channels.ai channel.AIChannel
                   attribute), 40
                                                                                                                                      attribute), 120
AHIGH_BHIGH (nidaqmx.constants.EncoderZIndexPhase ai_bridge_initial_ratio (nidaqmx._task_modules.channels.ai_channel.AIChannels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channel
                   attribute), 26
                                                                                                                                      attribute), 120
AHIGH_BLOW (nidaqmx.constants.EncoderZIndexPhase ai_bridge_initial_voltage
                   attribute), 26
                                                                                                                                      (nidaqmx._task_modules.channels.ai_channel.AIChannel
ai_ac_excit_freq (nidaqmx._task_modules.channels.ai_channel.AIChattnebute), 120
                   attribute), 119
                                                                                                                  ai_bridge_nom_resistance
ai_ac_excit_sync_enable
                                                                                                                                      (nidaqmx._task_modules.channels.ai_channel.AIChannel
                   (nidaqmx._task_modules.channels.ai_channel.AIChannel attribute), 121
                   attribute), 119
                                                                                                                  ai bridge physical units
ai ac excit wire mode (nidagmx. task modules.channels.ai channel.AIChannel
                   attribute), 119
                                                                                                                                      attribute), 121
ai_accel_4_wire_dc_voltage_sensitivity
                                                                                                                  ai_bridge_poly_forward_coeff
                   (nidaqmx._task_modules.channels.ai_channel.AIChannel (nidaqmx._task_modules.channels.ai_channel.AIChannel
                   attribute), 119
                                                                                                                                      attribute), 121
ai accel 4 wire dc voltage sensitivity units
                                                                                                                  ai bridge poly reverse coeff
                   (nidagmx, task modules.channels.ai channel.AIChannel (nidagmx, task modules.channels.ai channel.AIChannel
                   attribute), 119
                                                                                                                                      attribute), 121
ai_accel_charge_sensitivity
                                                                                                                  ai_bridge_rngs
                                                                                                                                                       (nidaqmx.system.device.Device
                                                                                                                                                                                                                           at-
                   (nidaqmx._task_modules.channels.ai_channel.AIChannel tribute), 99
                   attribute), 119
                                                                                                                  ai_bridge_scale_type (nidaqmx._task_modules.channels.ai_channel.AIChan
ai_accel_charge_sensitivity_units
                                                                                                                                      attribute), 121
                   (nidaqmx._task_modules.channels.ai_channel.AI@habridge_shunt_cal_enable
                   attribute), 119
                                                                                                                                      (nidaqmx._task_modules.channels.ai_channel.AIChannel
ai_accel_sensitivity (nidagmx._task_modules.channels.ai_channel.AIGhtaibute), 121
                   attribute), 119
                                                                                                                  ai_bridge_shunt_cal_gain_adjust
ai_accel_sensitivity_units
                                                                                                                                      (nidaqmx._task_modules.channels.ai_channel.AIChannel
                   (nidagmx. task modules.channels.ai channel.AIChannel attribute), 121
                   attribute), 120
                                                                                                                  ai bridge shunt cal select
```

(nidagmx. task modules.timing.Timing

ai\_conv\_dig\_fltr\_timebase\_rate

```
(nidaqmx._task_modules.channels.ai_channel.AIChannel attribute), 229
                                                                                       ai_conv_dig_fltr_timebase_src
              attribute), 121
ai bridge shunt cal shunt cal a resistance
                                                                                                      (nidagmx. task modules.timing.Timing
              (nidagmx. task modules.channels.ai channel.AIChannel attribute), 229
              attribute), 121
                                                                                       ai conv dig sync enable
ai_bridge_shunt_cal_shunt_cal_a_src
                                                                                                      (nidagmx. task modules.timing.Timing
                                                                                                     attribute), 229
              (nidaqmx._task_modules.channels.ai_channel.AIChannel
              attribute), 121
                                                                                       ai_conv_max_rate (nidaqmx._task_modules.timing.Timing
ai_bridge_shunt_cal_shunt_cal_b_actual_resistance
                                                                                                      attribute), 229
              (nidagmx_task_modules.channels.ai_channel.AIGhammet_rate (nidagmx_task_modules.timing.Timing at-
              attribute), 121
                                                                                                      tribute), 229
                                                                                       ai_conv_src (nidaqmx._task_modules.timing.Timing at-
ai_bridge_shunt_cal_shunt_cal_b_resistance
              (nidaqmx._task_modules.channels.ai_channel.AIChannel tribute), 230
              attribute), 121
                                                                                       ai_conv_timebase_div (nidaqmx._task_modules.timing.Timing
ai_bridge_table_electrical_vals
                                                                                                      attribute), 230
              (nidaqmx._task_modules.channels.ai_channel.AIGharonet/_timebase_src (nidaqmx._task_modules.timing.Timing
              attribute), 121
                                                                                                      attribute), 230
                                                                                       AI_CONVERT_CLOCK (nidaqmx.constants.Signal at-
ai_bridge_table_physical_vals
              (nidaqmx._task_modules.channels.ai_channel.AIChannel tribute), 39
              attribute), 121
                                                                                       ai_coupling (nidaqmx._task_modules.channels.ai_channel.AIChannel
ai bridge two point lin first electrical val
                                                                                                      attribute), 122
              (nidaqmx._task_modules.channels.ai_channel.AIGhammehlings (nidaqmx.system.device.Device attribute),
              attribute), 122
ai_bridge_two_point_lin_first_physical_val
                                                                                       ai_current_acrms_units (nidaqmx._task_modules.channels.ai_channel.AICl
              (nidagmx._task_modules.channels.ai_channel.AIChannel attribute), 122
                                                                                       ai_current_int_excit_discrete_vals
              attribute), 122
ai_bridge_two_point_lin_second_electrical_val
                                                                                                      (nidaqmx.system.device.Device
                                                                                                                                                             attribute),
              (nidaqmx._task_modules.channels.ai_channel.AIChannel 99
              attribute), 122
                                                                                       ai_current_rngs
                                                                                                                   (nidagmx.system.device.Device
ai_bridge_two_point_lin_second_physical_val
                                                                                                      tribute), 99
              (nidagmx._task_modules.channels.ai_channel.AIGharunreht_shunt_loc (nidagmx._task_modules.channels.ai_channel.AIChan
              attribute), 122
                                                                                                      attribute), 122
ai_bridge_units (nidaqmx._task_modules.channels.ai_channæl_AdfChannællunt_resistance
              attribute), 122
                                                                                                      (nidagmx. task modules.channels.ai channel.AIChannel
ai_channels (nidaqmx.task.Task attribute), 112
                                                                                                      attribute), 122
                           (nidagmx.system.device.Device
                                                                               at- ai current units (nidagmx. task modules.channels.ai channel.AIChannel
ai_charge_rngs
              tribute), 99
                                                                                                      attribute), 122
ai charge units (nidaqmx. task modules.channels.ai channeil.AIChannellai (nidaqmx. task modules.channels.ai channellai ch
                                                                                                      attribute), 122
              attribute), 122
ai_conv_active_edge (nidaqmx._task_modules.timing.Timingi_data_xfer_custom_threshold
              attribute), 229
                                                                                                      (nidaqmx.\_task\_modules.channels.ai\_channel.AIChannel
ai_conv_clk_output_term
                                                                                                      attribute), 122
              (nidaqmx._task_modules.export_signals.ExportSignalsta_xfer_mech (nidaqmx._task_modules.channels.ai_channels.AIChann
              attribute), 217
                                                                                                      attribute), 122
ai_conv_clk_pulse_polarity
                                                                                       ai_data_xfer_req_cond (nidaqmx._task_modules.channels.ai_channel.AICh
              (nidaqmx._task_modules.export_signals.ExportSignals
                                                                                                      attribute), 122
              attribute), 217
                                                                                       ai_dc_offset (nidagmx._task_modules.channels.ai_channel.AIChannel
                                                                                                      attribute), 123
ai_conv_dig_fltr_enable (nidaqmx._task_modules.timing.Timing
              attribute), 229
                                                                                       ai_dev_scaling_coeff (nidaqmx._task_modules.channels.ai_channel.AIChan
ai_conv_dig_fltr_min_pulse_width
                                                                                                      attribute), 123
              (nidagmx. task modules.timing.Timing
                                                                                       ai dig fltr bandpass center freq
```

(nidagmx, task modules, channels, ai channel, AIChannel attribute), 229

attribute), 121

ai bridge shunt cal shunt cal a actual resistance

```
(nidagmx. task modules.channels.ai channel.AIChannel attribute), 124
                                                                                                                                                                           ai excit sense (nidagmx. task modules.channels.ai channel.AIChannel
                             attribute), 123
ai dig fltr bandpass width
                                                                                                                                                                                                        attribute), 124
                             (nidagmx, task modules.channels.ai channel.AIGhaenneil src (nidagmx, task modules.channels.ai channel.AIChannel
                                                                                                                                                                                                        attribute), 124
                             attribute), 123
ai dig fltr coeff (nidagmx. task modules.channels.ai chananiele Add Chassen ellor scaling
                            attribute), 123
                                                                                                                                                                                                         (nidagmx. task modules.channels.ai channel.AIChannel
ai dig fltr enable (nidagmx. task modules.channels.ai channel.AIChatribulte), 124
                             attribute), 123
                                                                                                                                                                            ai excit use multiplexed
ai_dig_fltr_highpass_cutoff_freq
                                                                                                                                                                                                        (nidaqmx._task_modules.channels.ai_channel.AIChannel
                             (nidaqmx._task_modules.channels.ai_channel.AIChannel
                                                                                                                                                                                                       attribute), 124
                            attribute), 123
                                                                                                                                                                           ai_excit_val (nidagmx._task_modules.channels.ai_channel.AIChannel
ai_dig_fltr_lowpass_cutoff_freq
                                                                                                                                                                                                        attribute), 124
                            (nidaqmx._task_modules.channels.ai_channel.AIGhaemeit_voltage_or_current
                             attribute), 123
                                                                                                                                                                                                        (nidaqmx._task_modules.channels.ai_channel.AIChannel
ai_dig_fltr_lowpass_cutoff_freq_discrete_vals
                                                                                                                                                                                                         attribute), 124
                             (nidaqmx.system.device.Device
                                                                                                                                        attribute),
                                                                                                                                                                           ai_filter_delay (nidaqmx._task_modules.channels.ai_channel.AIChannel
                                                                                                                                                                                                        attribute), 124
ai dig fltr lowpass cutoff freq range vals
                                                                                                                                                                           ai filter delay adjustment
                             (nidagmx.system.device.Device
                                                                                                                                                                                                        (nidagmx. task modules.channels.ai channel.AIChannel
                                                                                                                                         attribute),
                                                                                                                                                                                                        attribute), 124
ai_dig_fltr_notch_center_freq
                                                                                                                                                                           ai filter delay units (nidagmx. task modules.channels.ai channel.AIChan
                             (nidaqmx._task_modules.channels.ai_channel.AIChannel attribute), 124
                             attribute), 123
                                                                                                                                                                           ai force iepe sensor sensitivity
ai dig fitr notch width (nidagmx. task modules.channels.ai channel.AIChannel
                             attribute), 123
                                                                                                                                                                                                        attribute), 124
ai_dig_fltr_order (nidaqmx._task_modules.channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_c
                             attribute), 123
                                                                                                                                                                                                         (nidaqmx._task_modules.channels.ai_channel.AIChannel
ai_dig_fltr_response (nidaqmx._task_modules.channels.ai_channel.Alathaihutel), 125
                             attribute), 123
                                                                                                                                                                           ai force read from chan
ai_dig_fltr_type (nidaqmx._task_modules.channels.ai_channel.AIChannel.AIChannel.ai_channels.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_channel.ai_chan
                             attribute), 123
                                                                                                                                                                                                        attribute), 125
ai_dig_fltr_types
                                                     (nidagmx.system.device.Device at-ai_force_units (nidagmx._task_modules.channels.ai_channels.ai_channel
                             tribute), 99
                                                                                                                                                                                                        attribute), 125
ai dither enable (nidagmx, task modules,channels,ai channels,ai ch
                            attribute), 123
                                                                                                                                                                                                        attribute), 125
ai eddy current prox sensitivity
                                                                                                                                                                           ai freq rngs (nidagmx.system.device.Device attribute),
                             (nidaqmx._task_modules.channels.ai_channel.AIChannel 99
                             attribute), 123
                                                                                                                                                                           ai_freq_thresh_voltage (nidaqmx._task_modules.channels.ai_channel.AICh
ai_eddy_current_prox_sensitivity_units
                                                                                                                                                                                                        attribute), 125
                            (nidagmx, task modules.channels.ai channel.AIGhafrael units (nidagmx, task modules.channels.ai channel.AIChannel
                            attribute), 123
                                                                                                                                                                                                        attribute), 125
ai eddy current prox units
                                                                                                                                                                           ai gain (nidaqmx. task modules.channels.ai channel.AIChannel
                             (nidagmx._task_modules.channels.ai_channel.AIChannel attribute), 125
                             attribute), 123
                                                                                                                                                                           ai_gains (nidaqmx.system.device.Device attribute), 100
                                                                                                                                                                            AI_HOLD_CMPLT_EVENT (nidaqmx.constants.Signal
ai_enhanced_alias_rejection_enable
                            (nidagmx. task modules.channels.ai channel.AIChannel attribute), 39
                            attribute), 124
                                                                                                                                                                           ai_hold_cmplt_event_output_term
ai_excit_actual_val (nidaqmx._task_modules.channels.ai_channels.ai_channels.AIQhadhaqphx._task_modules.export_signals.ExportSignals
                            attribute), 124
                                                                                                                                                                                                        attribute), 217
ai_excit_d_cor_ac (nidaqmx._task_modules.channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_c
                            attribute), 124
                                                                                                                                                                                                        (nidaqmx._task_modules.export_signals.ExportSignals
ai_excit_idle_output_behavior
                                                                                                                                                                                                        attribute), 217
                             (nidagmx, task modules.channels.ai channel.AIGhaimnpedance (nidagmx, task modules.channels.ai channel.AIChannel
```

```
attribute), 125
                                                                                                                                                                                                          (nidaqmx._task_modules.channels.ai_channel.AIChannel
ai input src (nidagmx. task modules.channels.ai channel.AIChanneattribute), 126
                                                                                                                                                                            ai min (nidagmx. task modules.channels.ai channel.AIChannel
                            attribute), 125
ai_input_srcs (nidaqmx.system.physical_channel.PhysicalChannel
                                                                                                                                                                                                         attribute), 126
                             attribute), 104
                                                                                                                                                                            ai_min_rate (nidaqmx.system.device.Device attribute),
ai lead wire resistance (nidagmx. task modules.channels.ai channell.AIChannel
                            attribute), 125
                                                                                                                                                                            ai open chan detect enable
ai_lossy_lsb_removal_compressed_samp_size
                                                                                                                                                                                                          (nidagmx. task modules.channels.ai channel.AIChannel
                             (nidagmx. task modules.channels.ai channel.AIChannel
                                                                                                                                                                                                        attribute), 126
                            attribute), 125
                                                                                                                                                                            ai_open_thrmcpl_detect_enable
ai_lowpass_cutoff_freq (nidaqmx._task_modules.channels.ai_channel(Ait@hannxeltask_modules.channels.ai_channel.AIChannel
                            attribute), 125
                                                                                                                                                                                                          attribute), 126
ai_lowpass_cutoff_freq_discrete_vals
                                                                                                                                                                            ai_physical_chans
                                                                                                                                                                                                                                                      (nidagmx.system.device.Device
                            (nidaqmx.system.device.Device
                                                                                                                                                                                                          attribute), 100
                                                                                                                                          attribute),
                                                                                                                                                                            ai_pressure_units (nidaqmx._task_modules.channels.ai_channel.AIChannel
ai_lowpass_cutoff_freq_range_vals
                                                                                                                                                                                                          attribute), 127
                             (nidaqmx.system.device.Device
                                                                                                                                         attribute),
                                                                                                                                                                            ai_probe_atten (nidaqmx._task_modules.channels.ai_channel.AIChannel
                                                                                                                                                                                                          attribute), 127
ai lowpass enable (nidagmx. task modules.channels.ai channel Mahammen pression type
                             attribute), 125
                                                                                                                                                                                                          (nidagmx. task modules.channels.ai channel.AIChannel
ai_lowpass_switch_cap_clk_src
                                                                                                                                                                                                          attribute), 127
                            (nidagmx. task modules.channels.ai channel.AIGhararvel samp justification
                             attribute), 125
                                                                                                                                                                                                          (nidaqmx._task_modules.channels.ai_channel.AIChannel
ai lowpass switch cap ext clk div
                                                                                                                                                                                                          attribute), 127
                             (nidagmx, task modules, channels, ai channel, AIGhannel samp size (nidagmx, task modules, channels, ai channel, AIGhannel
                             attribute), 125
                                                                                                                                                                                                          attribute), 127
ai_lowpass_switch_cap_ext_clk_freq
                                                                                                                                                                            ai_remove_filter_delay (nidaqmx._task_modules.channels.ai_channel.AICh
                             (nidaqmx._task_modules.channels.ai_channel.AIChannel attribute), 127
                            attribute), 126
                                                                                                                                                                            ai_resistance_cfg (nidaqmx._task_modules.channels.ai_channel.AIChannel
ai_lowpass_switch_cap_out_clk_div
                                                                                                                                                                                                          attribute), 127
                            (nidaqmx._task_modules.channels.ai_channel.AIGhamsistance_rngs (nidaqmx.system.device.Device at-
                             attribute), 126
                                                                                                                                                                                                          tribute), 100
ai_lvdt_sensitivity (nidaqmx._task_modules.channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai_channels.ai
                             attribute), 126
                                                                                                                                                                                                          attribute), 127
ai lydt sensitivity units (nidagmx, task modules.channels.ai rehandules.channels.ai channels.ai channe
                            attribute), 126
                                                                                                                                                                                                          attribute), 127
ai lvdt units (nidagmx. task modules.channels.ai channel.AIChannels.ai channels.ai channel
                             attribute), 126
                                                                                                                                                                                                          attribute), 127
ai_max (nidaqmx._task_modules.channels.ai_channel.AIChannel_high (nidaqmx._task_modules.channels.ai_channel.AIChannel
                             attribute), 126
                                                                                                                                                                                                          attribute), 127
ai max multi chan rate (nidaqmx.system.device.Device ai rng low (nidaqmx. task modules.channels.ai channel.AIChannel
                            attribute), 100
                                                                                                                                                                                                          attribute), 127
ai max single chan rate
                                                                                                                                                                            ai rosette strain gage gage orientation
                            (nidagmx.system.device.Device
                                                                                                                                          attribute),
                                                                                                                                                                                                          (nidaqmx._task_modules.channels.ai_channel.AIChannel
                                                                                                                                                                                                          attribute), 127
ai_meas_type (nidaqmx._task_modules.channels.ai_channelaAIGkattteettrain_gage_rosette_meas_type
                             attribute), 126
                                                                                                                                                                                                          (nidaqmx._task_modules.channels.ai_channel.AIChannel
ai_meas_types (nidaqmx.system.device.Device attribute),
                                                                                                                                                                                                          attribute), 127
                                                                                                                                                                            ai_rosette_strain_gage_rosette_type
ai_meas_types (nidaqmx.system.physical_channel.PhysicalChannel
                                                                                                                                                                                                        (nidaqmx._task_modules.channels.ai_channel.AIChannel
                             attribute), 104
                                                                                                                                                                                                          attribute), 127
ai mem map enable (nidaqmx. task modules.channels.ai arhannels.ai arha
                            attribute), 126
                                                                                                                                                                                                          (nidagmx. task modules.channels.ai channel.AIChannel
ai microphone sensitivity
                                                                                                                                                                                                          attribute), 127
```

```
ai_rtd_a (nidagmx._task_modules.channels.ai_channel.AIChannel
                                                                                                   attribute), 105
              attribute), 127
                                                                                     ai_thrmcpl_cjc_chan (nidaqmx._task_modules.channels.ai_channel.AIChar
ai_rtd_b (nidaqmx._task_modules.channels.ai_channel.AIChannel
                                                                                                   attribute), 129
              attribute), 128
                                                                                     ai_thrmcpl_cjc_src (nidaqmx._task_modules.channels.ai_channel.AIChann
ai_rtd_c (nidaqmx._task_modules.channels.ai_channel.AIChannel
                                                                                                  attribute), 129
              attribute), 128
                                                                                     ai thrmcpl cjc val (nidagmx. task modules.channels.ai channel.AIChann
ai rtd r 0 (nidagmx. task modules.channels.ai channel.AIChannel attribute), 129
                                                                                     ai_thrmcpl_lead_offset_voltage
              attribute), 128
ai_rtd_type (nidaqmx._task_modules.channels.ai_channel.AIChannel (nidaqmx._task_modules.channels.ai_channel.AIChannel
              attribute), 128
                                                                                                   attribute), 129
ai_rvdt_sensitivity (nidaqmx._task_modules.channels.ai_channels.ai_channels.ai_type (nidaqmx._task_modules.channels.ai_channels.ai
              attribute), 128
                                                                                                   attribute), 129
ai_rvdt_sensitivity_units (nidaqmx._task_modules.channels.aii_thramqel_A/Ioch(anidadmx._task_modules.channels.ai_channels.ai
                                                                                                   attribute), 129
              attribute), 128
ai_rvdt_units (nidaqmx._task_modules.channels.ai_channel al_Channel al_Channe
              attribute), 128
                                                                                                   attribute), 129
ai_samp_and_hold_enable
                                                                                     ai_thrmstr_b (nidaqmx._task_modules.channels.ai_channel.AIChannel
              (nidagmx. task modules.channels.ai channel.AIChannel attribute), 129
              attribute), 128
                                                                                     ai_thrmstr_c (nidaqmx._task_modules.channels.ai_channel.AIChannel
ai_samp_modes (nidaqmx.system.device.Device
                                                                                                   attribute), 129
              tribute), 100
                                                                                     ai_thrmstr_r_1 (nidaqmx._task_modules.channels.ai_channel.AIChannel
ai_simultaneous_sampling_supported
                                                                                                   attribute), 129
              (nidaqmx.system.device.Device
                                                                                    ai_torque_units (nidaqmx._task_modules.channels.ai_channel.AIChannel
                                                                   attribute),
                                                                                                   attribute), 129
                                                                                     ai trig usage (nidagmx.system.device.Device attribute),
ai_sound_pressure_lvl
              (nidagmx. task modules.channels.ai channel.AIChannel 100
              attribute), 128
                                                                                     ai_usb_xfer_req_count (nidaqmx._task_modules.channels.ai_channel.AICh
ai_sound_pressure_units (nidaqmx._task_modules.channels.ai_channeltr\hill@ha)nr\ell0
                                                                                     ai_usb_xfer_req_size (nidaqmx._task_modules.channels.ai_channel.AICha
             attribute), 128
ai_sound_pressured_b_ref
                                                                                                   attribute), 130
             (nidagmx._task_modules.channels.ai_channel.AIGhawalekity_iepe_sensor_sensitivity
              attribute), 128
                                                                                                   (nidaqmx._task_modules.channels.ai_channel.AIChannel
ai_strain_force_read_from_chan
                                                                                                   attribute), 130
              (nidaqmx._task_modules.channels.ai_channel.AIGhannelecity_iepe_sensor_sensitivity_units
              attribute), 128
                                                                                                   (nidagmx. task modules.channels.ai channel.AIChannel
ai_strain_gage_cfg (nidaqmx._task_modules.channels.ai_channel.AICatambute), 130
              attribute), 128
                                                                                     ai_velocity_iepe_sensord_b_ref
ai_strain_gage_gage_factor
                                                                                                   (nidaqmx._task_modules.channels.ai_channel.AIChannel
              (nidagmx. task modules.channels.ai channel.AIChannel attribute), 130
              attribute), 128
                                                                                     ai_velocity_units (nidaqmx._task_modules.channels.ai_channel.AIChannel
ai strain gage poisson ratio
                                                                                                   attribute), 130
              (nidagmx, task modules.channels.ai channel.AIGhawolthge acrms units (nidagmx, task modules.channels.ai channel.AIGhawolthge acrms units (nidagmx, task modules.channels.ai channels.ai
              attribute), 128
                                                                                                   attribute), 130
ai_strain_units (nidaqmx._task_modules.channels.ai_channeli_AM@lhagm_eiht_excit_discrete_vals
                                                                                                   (nidaqmx.system.device.Device
              attribute), 129
                                                                                                                                                         attribute),
ai_teds_is_teds (nidaqmx._task_modules.channels.ai_channel.AIChanh@0
              attribute), 129
                                                                                     ai_voltage_int_excit_range_vals
ai_teds_units (nidaqmx._task_modules.channels.ai_channel.AIChannehidaqmx.system.device.Device
                                                                                                                                                         attribute),
              attribute), 129
ai_temp_units (nidaqmx._task_modules.channels.ai_channelaiAtGltageelngs (nidaqmx.system.device.Device
              attribute), 129
                                                                                                   tribute), 101
ai_term_cfg (nidaqmx._task_modules.channels.ai_channel.AIChannel.ai_totannels.ai_channels.ai_channel.aiChannel
              attribute), 129
                                                                                                   attribute), 130
ai term cfgs (nidaqmx.system.physical channel.PhysicalChainwoltaged b ref (nidaqmx. task modules.channels.ai channel.AIChanne
```

attribute), 130	tribute), 44		
AIChannel (class in nidaqmxtask_modules.channels.ai_	<b>NAL</b> OG_INPUT (nidaqmx.c	onstants.Channel	Гуре at-
119	tribute), 22		
AIChannelCollection (class in A	NALOG_LEVEL (nidaqmx.	constants.TriggerT	Гуре at-
nidaqmxtask_modules.ai_channel_collection),	tribute), 44		
156 A	NALOG_OUTPUT (nidaqn	nx.constants.Chan	nelType
AIPhysicalChannelCollection (class in	attribute), 22		
nidaqmx.systemcollections.physical_channel_collections	(nidaq	mx.constants.Trig	gerType
97	attribute), 44		
all (nidaqmxtask_modules.ai_channel_collection.AIChannel	habble MidtiChannelReader	(class	in
attribute), 201	nidaqmx.stream_read	ers), 57	
all (nidagmxtask_modules.ao_channel_collection.AOCham	naka and Maritin hannel Writer	(class	in
attribute), 202	nidaqmx.stream_writ	ers), 78	
all (nidaqmxtask_modules.channel_collection.ChannelColl	-	(class	in
attribute), 156	nidaqmx.stream_read	*	
all (nidaqmxtask_modules.ci_channel_collection.CIChannel		(class	in
attribute), 213	nidaqmx.stream_writ	*	
all (nidaqmxtask_modules.co_channel_collection.COCham	-	(class	in
attribute), 215	nidaqmx.stream_read	*	
all (nidaqmxtask_modules.di_channel_collection.DIChann		(class	in
attribute), 215	nidaqmx.stream_writ	*	<b></b>
all (nidaqmxtask_modules.do_channel_collection.DOChan	-		
•	ngularVelocityUnits (class in		s) 19
all (nidagmx.systemcollections.physical_channel_collections			
attribute), 98	attribute), 240	_task_inodates.trig	550111151101010100_111550111
	nlg_edge_coupling (nidaqmx.	task modules tric	goering start trigger Start
(nidaqmx.system.storage.persisted_channel.Persisted		_task_inoaaies.ti18	55011115.54411_4115501.544111
· · · · · · · · · · · · · · · · · · ·	nlg_edge_dig_fltr_enable		
allow_interactive_deletion	e = e = e = -	ules triggering ref	erence_trigger.ReferenceT
(nidaqmx.system.storage.persisted_scale.PersistedS	· •	a105.11155011115.1011	erence_urggenitererence i
	nlg_edge_dig_fltr_enable		
allow_interactive_deletion		ules triogering star	rt_trigger.StartTrigger
(nidaqmx.system.storage.persisted_task.PersistedTa	=	ares.urggering.sta	it_uiggei.startiiiggei
	nlg_edge_dig_fltr_min_pulse_	width	
allow_interactive_editing	$\mathbf{e}$ $\mathbf{e}$ $\mathbf{e}$ $\mathbf{e}$ $\mathbf{e}$ $\mathbf{e}$ $\mathbf{e}$		erence_trigger.ReferenceT
(nidaqmx.system.storage.persisted_channel.Persisted		uics.uiggeiiig.iei	erence_uigger.itererence i
	nlg_edge_dig_fltr_min_pulse_	width	
allow interactive editing			rt_trigger.StartTrigger
(nidaqmx.system.storage.persisted_scale.PersistedS		ares.urggering.sta	it_uiggei.startiiiggei
· · · · · · · · · · · · · · · · · · ·	nlg_edge_dig_fltr_timebase_ra	ate	
allow_interactive_editing			erence_trigger.ReferenceT
(nidaqmx.system.storage.persisted_task.PersistedTa	· •	uics.uiggeiiig.iei	erence_urgger.Reference i
	nlg_edge_dig_fltr_timebase_ra	ate	
ALLOW_REGENERATION			rt_trigger.StartTrigger
(nidaqmx.constants.RegenerationMode at-	attribute), 244	uics.uiggeiiig.sui	it_uiggei.Startiiiggei
	nlg_edge_dig_fltr_timebase_s	rc	
ALOW_BHIGH (nidaqmx.constants.EncoderZIndexPhase			erence_trigger.ReferenceT
attribute), 26	attribute), 240	uics.uiggeiiig.iei	erence_urgger.Referencer
ALOW_BLOW (nidaqmx.constants.EncoderZIndexPhase a		rc	
attribute), 26			rt_trigger.StartTrigger
AM (nidaqmx.constants.ModulationType attribute), 32	attribute), 244	ares.urggering.sta	11_115501.511111111111111111111111111111
	nlg_edge_dig_sync_enable		
AMPS (nidaqmx.constants.UnitsPreScaled attribute), 45		ules triggering ref	erence_trigger.ReferenceT
ANALOG_EDGE (nidaqmx.constants.TriggerType at-	attribute), 240	arcs.urggering.ici	oroneo_urggor.Reference i
induquin.constants.triggertype at-	amioum, 270		

```
anlg_edge_dig_sync_enable
                                                                                                                                        attribute), 240
                   (nidaqmx._task_modules.triggering.start_trigger.StartfTvigjecoupling (nidaqmx._task_modules.triggering.start_trigger.StartTr
                   attribute), 244
                                                                                                                                        attribute), 245
anlg_edge_hyst (nidaqmx._task_modules.triggering.referencenltrigger.Riefefttnceflathlger
                   attribute), 240
                                                                                                                                        (nidaqmx._task_modules.triggering.pause_trigger.PauseTrigger
anlg edge hyst (nidagmx. task modules.triggering.start trigger.Startatigbete), 238
                                                                                                                    anlg_win_dig_fltr enable
                   attribute), 244
anlg edge lvl (nidaqmx. task modules.triggering.reference trigger.ReferenceTrigger modules.triggering.referenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.ReferenceTrigger.Refere
                   attribute), 240
                                                                                                                                        attribute), 241
anlg_edge_lvl (nidaqmx._task_modules.triggering.start_triggerlgStantf_riligerfltr_enable
                   attribute), 244
                                                                                                                                        (nidaqmx._task_modules.triggering.start_trigger.StartTrigger
anlg_edge_slope (nidaqmx._task_modules.triggering.reference_triggeatkribetter)c@Trigger
                   attribute), 240
                                                                                                                    anlg_win_dig_fltr_min_pulse_width
anlg_edge_slope (nidaqmx._task_modules.triggering.start_trigger.Startfflidagerx._task_modules.triggering.pause_trigger.PauseTrigger
                   attribute), 244
                                                                                                                                        attribute), 238
anlg_edge_src (nidaqmx._task_modules.triggering.referencearthiggerinRdfgrdftrefthinggenlse_width
                                                                                                                                        (nidaqmx._task_modules.triggering.reference_trigger.ReferenceT
                   attribute), 240
anlg_edge_src (nidaqmx._task_modules.triggering.start_trigger.StartTaitgibnte), 241
                   attribute), 244
                                                                                                                    anlg_win_dig_fltr_min_pulse_width
anlg lvl coupling (nidaqmx. task modules.triggering.pause trigger.Raniskaffringgerask modules.triggering.start trigger.StartTrigger
                                                                                                                                        attribute), 245
                   attribute), 237
anlg_lvl_dig_fltr_enable (nidaqmx._task_modules.triggering.mpkg_swintridigerflarusieflet)ger_rate
                   attribute), 238
                                                                                                                                        (nidaqmx._task_modules.triggering.pause_trigger.PauseTrigger
anlg lvl dig fltr min pulse width
                                                                                                                                        attribute), 238
                   (nidagmx. task modules.triggering.pause triggerandgsev ing deg filtr timebase rate
                   attribute), 238
                                                                                                                                        (nidagmx. task modules.triggering.reference trigger.ReferenceT
anlg_lvl_dig_fltr_timebase_rate
                                                                                                                                        attribute), 241
                   (nidaqmx._task_modules.triggering.pause_triggerandgevTinigder_fltr_timebase_rate
                   attribute), 238
                                                                                                                                        (nidaqmx._task_modules.triggering.start_trigger.StartTrigger
anlg_lvl_dig_fltr_timebase_src
                                                                                                                                        attribute), 245
                   (nidaqmx._task_modules.triggering.pause_triggerandgevilingdeg_fltr_timebase_src
                   attribute), 238
                                                                                                                                        (nidaqmx._task_modules.triggering.pause_trigger.PauseTrigger
anlg_lvl_dig_sync_enable
                                                                                                                                        attribute), 239
                   (nidaqmx._task_modules.triggering.pause_triggerandgevaling_deg_fltr_timebase_src
                   attribute), 238
                                                                                                                                        (nidagmx, task modules.triggering.reference trigger.ReferenceT
anlg_lvl_hyst (nidaqmx._task_modules.triggering.pause_trigger.Pauseaftriguete), 241
                                                                                                                    anlg win dig fltr timebase src
                   attribute), 238
anlg\_lvl\_lvl \ (nidaqmx.\_task\_modules.triggering.pause\_trigger.PauseT \ (nidaqmx.\_task\_modules.triggering.start\_trigger.StartTrigger.PauseT \ (nidaqmx.\_task\_modules.triggering.start\_trigger.StartTrigger.PauseT \ (nidaqmx.\_task\_modules.triggering.start\_trigger.StartTrigger.PauseT \ (nidaqmx.\_task\_modules.triggering.start\_trigger.StartTrigger.PauseT \ (nidaqmx.\_task\_modules.triggering.start\_trigger.StartTrigger.PauseT \ (nidaqmx.\_task\_modules.triggering.start\_trigger.StartTrigger.PauseT \ (nidaqmx.\_task\_modules.triggering.start\_trigger.StartTrigger.StartTrigger.StartTrigger.PauseT \ (nidaqmx.\_task\_modules.triggering.start\_trigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.StartTrigger.Star
                   attribute), 238
                                                                                                                                        attribute), 245
anlg_lvl_src (nidaqmx._task_modules.triggering.pause_triggerlegavie_Tdiggerync_enable
                                                                                                                                        (nidaqmx. task modules.triggering.pause trigger.PauseTrigger
                   attribute), 238
anlg lvl when (nidagmx. task modules.triggering.pause trigger.Pausetlinger), 239
                   attribute), 238
                                                                                                                     anlg win dig sync enable
anlg_trig_supported (nidaqmx.system.device.Device at-
                                                                                                                                        (nidaqmx._task_modules.triggering.reference_trigger.ReferenceT
                   tribute), 101
                                                                                                                                        attribute), 241
anlg_win_btm (nidaqmx._task_modules.triggering.pause_trigger.Rainsellgiggerc_enable
                   attribute), 238
                                                                                                                                        (nidaqmx._task_modules.triggering.start_trigger.StartTrigger
anlg_win_btm (nidaqmx._task_modules.triggering.reference_trigger.Ratfeilbute)Teiger
                   attribute), 240
                                                                                                                    anlg_win_src (nidaqmx._task_modules.triggering.pause_trigger.PauseTrigg
anlg_win_btm (nidaqmx._task_modules.triggering.start_trigger.StartTatggbute), 239
                   attribute), 245
                                                                                                                    anlg_win_src (nidaqmx._task_modules.triggering.reference_trigger.Referer
anlg_win_coupling (nidaqmx._task_modules.triggering.pause_trigger.htariseife)g@erl
                   attribute), 238
                                                                                                                    anlg_win_src (nidaqmx._task_modules.triggering.start_trigger.StartTrigger
```

anlg win coupling (nidaqmx. task modules.triggering.reference triggettriRufe)ende6Trigger

```
anlg_win_top (nidaqmx._task_modules.triggering.pause_trigger.Pauseaftrigute), 132
                                                                                                                ao_filter_delay_units (nidaqmx._task_modules.channels.ao_channel.AOChannels.ao
                  attribute), 239
anlg_win_top (nidaqmx._task_modules.triggering.reference_trigger.Raftniemte)[rl@ger
                   attribute), 241
                                                                                                                ao_func_gen_amplitude (nidaqmx._task_modules.channels.ao_channel.AO
anlg_win_top (nidaqmx._task_modules.triggering.start_trigger.StartTriggibute), 132
                  attribute), 245
                                                                                                                ao func gen fm deviation
anlg win trig when (nidaqmx. task modules.triggering.reference triggida Runferetask Triggeringskeles.channels.ao channel.AOChannel
                                                                                                                                   attribute), 132
                   attribute), 241
anlg_win_trig_when (nidaqmx._task_modules.triggering.starto_tfiggering.starto_tfiggeridaqmx._task_modules.channels.ao_channel.AOChanr
                  attribute), 245
                                                                                                                                   attribute), 132
anlg_win_when (nidaqmx._task_modules.triggering.pause_triggfunPagenTriggfurlation_type
                  attribute), 239
                                                                                                                                   (nidaqmx._task_modules.channels.ao_channel.AOChannel
ao_channels (nidaqmx.task.Task attribute), 112
                                                                                                                                   attribute), 132
                                                                                                      at- ao_func_gen_offset (nidaqmx._task_modules.channels.ao_channel.AOChan
ao_current_rngs
                                  (nidaqmx.system.device.Device
                   tribute), 101
                                                                                                                                   attribute), 133
ao_current_units (nidaqmx._task_modules.channels.ao_channelft.AnQChanneduare_duty_cycle
                  attribute), 131
                                                                                                                                   (nidaqmx._task_modules.channels.ao_channel.AOChannel
ao_custom_scale (nidaqmx._task_modules.channels.ao_channel.AOChttnibalte), 133
                                                                                                                ao_func_gen_type (nidaqmx._task_modules.channels.ao_channel.AOChannels.ao_channels.ao
                   attribute), 131
ao_dac_offset_ext_src (nidaqmx._task_modules.channels.ao_channel.at@Channels.ao
                  attribute), 131
                                                                                                                ao_gain (nidaqmx._task_modules.channels.ao_channel.AOChannel
ao_dac_offset_src (nidaqmx._task_modules.channels.ao_channel.AO@htaribuetb), 133
                   attribute), 131
                                                                                                                ao_gains (nidaqmx.system.device.Device attribute), 101
ao_dac_offset_val (nidaqmx._task_modules.channels.ao_channedlaQ@haundelehavior
                                                                                                                                   (nidaqmx._task_modules.channels.ao_channel.AOChannel
                  attribute), 131
ao_dac_ref_allow_conn_to_gnd
                                                                                                                                   attribute), 133
                   (nidagmx._task_modules.channels.ao_channel.AQ6hanatelimpedance (nidagmx._task_modules.channels.ao_channel.AOCha
                  attribute), 131
                                                                                                                                   attribute), 133
                                                                                                                ao_manual_control_amplitude
ao_dac_ref_conn_to_gnd
                  (nidaqmx._task_modules.channels.ao_channel.AOChannel (nidaqmx.system.physical_channel.PhysicalChannel
                   attribute), 131
                                                                                                                                   attribute), 105
ao_dac_ref_ext_src (nidaqmx._task_modules.channels.ao_channels.ao_channels.ao_channels.ao
                   attribute), 131
                                                                                                                                   (nidaqmx.system.physical_channel.PhysicalChannel
ao_dac_ref_src (nidaqmx._task_modules.channels.ao_channel.AOChattnebute), 105
                   attribute), 132
                                                                                                                ao_manual_control_freq (nidaqmx.system.physical_channel.PhysicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physicalChannel.physic
ao_dac_ref_val (nidaqmx._task_modules.channels.ao_channel.AOChattnebute), 105
                                                                                                                ao_manual_control_short_detected
                  attribute), 132
ao_dac_rng_high (nidaqmx._task_modules.channels.ao_channel.AOC(nidaqmx.system.physical_channel.PhysicalChannel
                   attribute), 132
                                                                                                                                   attribute), 105
ao_dac_rng_low (nidaqmx._task_modules.channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_channels.ao_chan
                  attribute), 132
                                                                                                                                   attribute), 133
ao data xfer mech (nidagmx. task modules.channels.ao channelxAfaCha(mindagmx.system.device.Device attribute),
                   attribute), 132
ao_data_xfer_req_cond (nidaqmx._task_modules.channels.ao_dhammethAp_Chabhte{nidaqmx._task_modules.channels.ao_channel.AOCl
                   attribute), 132
                                                                                                                                   attribute), 133
ao_dev_scaling_coeff (nidaqmx._task_modules.channels.ao<u>aohamine(nAQQimannel</u>s.k_modules.channels.ao_channel.AOChannel
                  attribute), 132
                                                                                                                                   attribute), 133
                                                                                                                ao_min_rate (nidagmx.system.device.Device attribute),
ao_enhanced_image_rejection_enable
                  (nidaqmx._task_modules.channels.ao_channel.AOChannel 101
                  attribute), 132
                                                                                                                ao_output_impedance (nidaqmx._task_modules.channels.ao_channel.AOCl
ao_filter_delay (nidaqmx._task_modules.channels.ao_channel.AOChaatttribute), 133
                  attribute), 132
                                                                                                                ao output type (nidagmx. task modules.channels.ao channel.AOChannel
ao_filter_delay_adjustment
                                                                                                                                   attribute), 133
                  (nidagmx. task modules.channels.ao channel.AQ6hontpult types (nidagmx.system.device.Device at-
```

```
tribute), 101
                                                                                                                                        131
ao output types (nidagmx.system.physical channel.Physical Channel.Physical
                                                                                                                                                                                           (class
                                                                                                                                                                                                                                in
                   attribute), 105
                                                                                                                                        nidagmx. task modules.ao channel collection),
ao_physical_chans (nidaqmx.system.device.Device at-
                   tribute), 101
                                                                                                                    AOExpirationState (class in nidagmx.types), 247
ao power amp channel enable
                                                                                                                    AOIdleOutputBehavior (class in nidagmx.constants), 18
                   (nidagmx.system.physical channel.PhysicalChannelCollection
                   attribute), 105
                                                                                                                                        nidagmx.system. collections.physical channel collection),
ao_power_amp_gain (nidaqmx.system.physical_channel.PhysicalChannel
                   attribute), 105
                                                                                                                    AOPowerUpOutputBehavior
                                                                                                                                                                                                 (class
                                                                                                                                                                                                                                in
ao_power_amp_offset (nidaqmx.system.physical_channel.PhysicalChandamx.constants), 18
                   attribute), 105
                                                                                                                    AOPowerUpState (class in nidagmx.types), 248
ao_power_amp_overcurrent
                                                                                                                    are_configured_cdaq_sync_ports_disconnected()
                   (nidaqmx.system.physical_channel.PhysicalChannel
                                                                                                                                        (nidaqmx.system.system.System
                                                                                                                                                                                                                   method),
                   attribute), 105
ao_power_amp_scaling_coeff
                                                                                                                    ARM_START
                                                                                                                                                      (nidaqmx.constants.TriggerUsage
                   (nidaqmx.system.physical_channel.PhysicalChannel
                                                                                                                                        tribute), 44
                   attribute), 105
                                                                                                                    arm_start_trigger (nidaqmx._task_modules.triggers.Triggers
ao_power_up_output_types
                                                                                                                                        attribute), 236
                   (nidagmx.system.physical channel.PhysicalChann&mStartTrigger
                                                                                                                                                                                     (class
                                                                                                                                                                                                                                in
                   attribute), 105
                                                                                                                                        nidaqmx._task_modules.triggering.arm_start_trigger),
ao_reglitch_enable (nidaqmx._task_modules.channels.ao_channel.AO@fannel
                   attribute), 133
                                                                                                                    AUSE_UNTIL_DATA_AVAILABLE
ao resolution (nidagmx. task modules.channels.ao channel.AOChannels.ao channels.ao channel
                   attribute), 133
                                                                                                                                        tribute), 45
ao resolution units (nidaqmx. task modules.channels.ao chanthurl(Aidadnaxuse)stem.storage.persisted channel.PersistedChannel
                   attribute), 133
                                                                                                                                        attribute), 107
ao_samp_clk_supported (nidaqmx.system.device.Device
                                                                                                                    author (nidaqmx.system.storage.persisted_scale.PersistedScale
                   attribute), 101
                                                                                                                                        attribute), 108
ao_samp_modes
                                      (nidaqmx.system.device.Device at-
                                                                                                                    author (nidaqmx.system.storage.persisted_task.PersistedTask
                   tribute), 101
                                                                                                                                        attribute), 108
AO_SERIES
                                       (nidaqmx.constants.ProductCategory
                                                                                                                    auto_configure_cdaq_sync_connections()
                   attribute), 34
                                                                                                                                        (nidaqmx.system.system.System
                                                                                                                                                                                                                   method),
ao_term_cfg (nidaqmx._task_modules.channels.ao_channel.AOChannel
                   attribute), 134
                                                                                                                    auto start (nidagmx. task modules.in stream.InStream
ao term cfgs (nidagmx.system.physical channel.PhysicalChannel
                                                                                                                                       attribute), 220
                   attribute), 105
                                                                                                                    auto start (nidagmx. task modules.out stream.OutStream
ao_trig_usage (nidaqmx.system.device.Device attribute),
                                                                                                                                        attribute), 226
                                                                                                                    auto start (nidagmx.stream writers.AnalogMultiChannelWriter
ao_usb_xfer_req_count (nidaqmx._task_modules.channels.ao_channelatatotilea)qn@l
                                                                                                                    auto start (nidagmx.stream writers.AnalogSingleChannelWriter
                   attribute), 134
ao usb xfer reg size (nidagmx. task modules.channels.ao channel.Aththannel7
                   attribute), 134
                                                                                                                    auto start (nidagmx.stream writers.AnalogUnscaledWriter
ao_use_only_on_brd_mem
                                                                                                                                        attribute), 79
                   (nidaqmx._task_modules.channels.ao_channel.AO@harntelt
                                                                                                                                                      (nidagmx.stream_writers.CounterWriter
                   attribute), 134
                                                                                                                                        attribute), 82
ao voltage current limit
                                                                                                                    auto start (nidagmx.stream writers.DigitalMultiChannelWriter
                   (nidaqmx._task_modules.channels.ao_channel.AOChannel attribute), 87
                   attribute), 134
                                                                                                                    auto_start (nidaqmx.stream_writers.DigitalSingleChannelWriter
ao_voltage_rngs (nidaqmx.system.device.Device
                                                                                                                                        attribute), 84
                                                                                                                    auto_trig_enable (nidaqmx._task_modules.triggering.reference_trigger.Refe
                   tribute), 101
ao voltage units (nidagmx. task modules.channels.ao channel.AOChtmitelte), 241
                   attribute), 134
                                                                                                                    auto triggered (nidaqmx. task modules.triggering.reference trigger.Refere
AOChannel (class in nidagmx. task modules.channels.ao channel), attribute), 241
```

Type Type Type er.ReferenceT
Туре
r.ReferenceT
r.ReferenceT
r.ReferenceT
r.Reference1
rtTrigger
r.ReferenceT
rtTrigger
eference_trigg
referee_u1g
rtTrigger
r.ReferenceT
rtTrigger
ng
1

cfg_watchdog_ao_expir_states() (nidaqmx.system.watchdog.WatchdogTask	attribute), 33 CHANNEL_HIGH_IMPEDANCE
method), 109	(nidaqmx.constants.PowerUpChannelType
cfg_watchdog_co_expir_states()	attribute), 33
(nidaqmx.system.watchdog.WatchdogTask	CHANNEL_IN_USE (nidaqmx.constants.PathCapability
method), 109	attribute), 33
cfg_watchdog_do_expir_states()	channel_names (nidaqmxtask_modules.ai_channel_collection.AIChannel
(nidaqmx.system.watchdog.WatchdogTask	attribute), 201
method), 110	channel_names (nidaqmxtask_modules.ao_channel_collection.AOChannel
CHAN_FOR_ALL_LINES	attribute), 202
(nidaqmx.constants.LineGrouping attribute),	channel_names (nidaqmxtask_modules.channel_collection.ChannelCollec
31	attribute), 156
CHAN_PER_LINE (nidaqmx.constants.LineGrouping	channel_names (nidaqmxtask_modules.channels.ai_channel.AIChannel
attribute), 31	attribute), 130
	IChannel_names (nidaqmxtask_modules.channels.ao_channel.AOChannel
attribute), 130	attribute), 134
	ACK2marene_hames (nidaqmxtask_modules.channels.channel.Channel
attribute), 134	attribute), 118
**	nuchannel_names (nidaqmxtask_modules.channels.ci_channel.CIChannel
attribute), 118	attribute), 135
chan_type (nidaqmxtask_modules.channels.ci_channel.C	IChannel_names (nidaqmxtask_modules.channels.co_channel.COChannel
attribute), 135	attribute), 149
chan_type (nidaqmxtask_modules.channels.co_channel.C	Colomarente hames (nidaqmxtask_modules.channels.di_channel.DIChannel
attribute), 149	attribute), 152
chan_type (nidaqmxtask_modules.channels.di_channel.D	OlChamed_names (nidaqmxtask_modules.channels.do_channel.DOChannel
attribute), 152	attribute), 154
$chan\_type \ (nidaqmx.\_task\_modules.channels.do\_channel.I$	Deltahanhehames (nidaqmxtask_modules.ci_channel_collection.CIChannel
attribute), 154	attribute), 213
change_detect_di_falling_edge_physical_chans	$channel\_names (nidaqmx.\_task\_modules.co\_channel\_collection.COChannel\_c$
(nidaqmxtask_modules.timing.Timing	attribute), 215
attribute), 233	$channel\_names (nidaqmx.\_task\_modules.di\_channel\_collection.DIChannelDIChannel\_collection.DIChannelDIChannelDIChannelDIChannelDIChanne$
change_detect_di_rising_edge_physical_chans	attribute), 215
(nidaqmxtask_modules.timing.Timing	channel_names (nidaqmxtask_modules.do_channel_collection.DOChannel_
attribute), 233	attribute), 216
change_detect_di_tristate	channel_names (nidaqmx.systemcollections.physical_channel_collection.
(nidaqmxtask_modules.timing.Timing	attribute), 98
attribute), 233	channel_names (nidaqmx.task.Task attribute), 112
change_detect_event_output_term	CHANNEL_RESERVED_FOR_ROUTING
(nidaqmxtask_modules.export_signals.ExportS	
attribute), 217	33
change_detect_event_pulse_polarity	CHANNEL_SOURCE_CONFLICT
(nidaqmxtask_modules.export_signals.ExportS attribute), 217	ignals (nidaqmx.constants.PathCapability attribute), 33
change_detect_overflowed	channel_type (nidaqmx.types.AOPowerUpState at-
(nidaqmxtask_modules.in_stream.InStream	tribute), 248
attribute), 220	CHANNEL_VOLTAGE (nidaqmx.constants.PowerUpChannelType
CHANGE_DETECTION	attribute), 33
(nidaqmx.constants.SampleTimingType at-	ChannelCollection (class in
tribute), 37	nidaqmxtask_modules.channel_collection),
CHANGE_DETECTION_EVENT	156
(nidaqmx.constants.Signal attribute), 39	channels (nidaqmx.task.Task attribute), 112
•	l)channels_to_read (nidaqmxtask_modules.in_stream.InStream
118	attribute), 220
CHANNEL_CURRENT (nidaqmx.constants.PowerUpCha	ntice and the control of the control

```
CHARGE (nidagmx.constants.CalibrationMode2
                                                                                                  attribute), 136
              tribute), 22
                                                                                   ci count edges count reset dig fltr timebase src
CHARGE (nidagmx.constants.UsageTypeAI attribute),
                                                                                                  (nidagmx. task modules.channels.ci channel.CIChannel
                                                                                                  attribute), 136
ChargeUnits (class in nidagmx.constants), 23
                                                                                   ci count edges count reset dig sync enable
chassis module devices (nidagmx.system.device.Device
                                                                                                  (nidagmx. task modules.channels.ci channel.CIChannel
             attribute), 101
                                                                                                  attribute), 136
CHS (nidagmx.constants.Language attribute), 30
                                                                                   ci count edges count reset enable
ci ang encoder initial angle
                                                                                                  (nidagmx. task modules.channels.ci channel.CIChannel
             (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                 attribute), 136
              attribute), 135
                                                                                    ci_count_edges_count_reset_logic_lvl_behavior
ci_ang_encoder_pulses_per_rev
                                                                                                  (nidaqmx._task_modules.channels.ci_channel.CIChannel
              (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                 attribute), 136
             attribute), 135
                                                                                   ci_count_edges_count_reset_reset_cnt
ci_ang_encoder_units (nidaqmx._task_modules.channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.c
              attribute), 135
                                                                                                  attribute), 136
ci_channels (nidaqmx.task.Task attribute), 112
                                                                                   ci_count_edges_count_reset_term
ci count (nidagmx. task modules.channels.ci channel.CIChannel
                                                                                                 (nidagmx. task modules.channels.ci channel.CIChannel
              attribute), 135
                                                                                                  attribute), 136
ci_count_edges_active_edge
                                                                                   ci count edges count reset term cfg
              (nidagmx. task modules.channels.ci channel.CIChannel
                                                                                                 (nidaqmx._task_modules.channels.ci_channel.CIChannel
              attribute), 135
                                                                                                  attribute), 136
ci_count_edges_count_dir_dig_fltr_enable
                                                                                   ci_count_edges_dig_fltr_enable
              (nidagmx, task modules,channels,ci channel.CIChannel (nidagmx, task modules,channels,ci channel.CIChannel
              attribute), 135
                                                                                                  attribute), 136
ci count edges count dir dig fltr min pulse width
                                                                                   ci count edges dig fltr min pulse width
              (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                 (nidaqmx._task_modules.channels.ci_channel.CIChannel
              attribute), 135
                                                                                                  attribute), 136
ci_count_edges_count_dir_dig_fltr_timebase_rate
                                                                                   ci_count_edges_dig_fltr_timebase_rate
              (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                 (nidaqmx._task_modules.channels.ci_channel.CIChannel
              attribute), 135
                                                                                                  attribute), 136
ci\_count\_edges\_count\_dir\_dig\_fltr\_timebase\_src
                                                                                   ci_count_edges_dig_fltr_timebase_src
             (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                 (nidaqmx._task_modules.channels.ci_channel.CIChannel
             attribute), 135
                                                                                                  attribute), 136
ci count edges count dir dig sync enable
                                                                                   ci count edges dig sync enable
              (nidagmx. task modules.channels.ci channel.CIChannel
                                                                                                 (nidagmx. task modules.channels.ci channel.CIChannel
             attribute), 135
                                                                                                  attribute), 136
ci_count_edges_count_dir_logic_lvl_behavior
                                                                                   ci_count_edges_dir (nidaqmx._task_modules.channels.ci_channel.CIChannels.ci
              (nidaqmx._task_modules.channels.ci_channel.CIChannel attribute), 136
             attribute), 135
                                                                                   ci_count_edges_dir_term
ci count edges count dir term cfg
                                                                                                  (nidagmx. task modules.channels.ci channel.CIChannel
             (nidagmx. task modules.channels.ci channel.CIChannel
                                                                                                 attribute), 136
              attribute), 135
                                                                                   ci count edges gate dig fltr enable
ci_count_edges_count_reset_active_edge
                                                                                                  (nidaqmx._task_modules.channels.ci_channel.CIChannel
              (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                 attribute), 137
              attribute), 136
                                                                                   ci_count_edges_gate_dig_fltr_min_pulse_width
ci_count_edges_count_reset_dig_fltr_enable
                                                                                                  (nidagmx. task modules.channels.ci channel.CIChannel
             (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                 attribute), 137
                                                                                   ci_count_edges_gate_dig_fltr_timebase_rate
             attribute), 136
ci_count_edges_count_reset_dig_fltr_min_pulse_width
                                                                                                  (nidaqmx._task_modules.channels.ci_channel.CIChannel
              (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                 attribute), 137
             attribute), 136
                                                                                   ci_count_edges_gate_dig_fltr_timebase_src
ci_count_edges_count_reset_dig_fltr_timebase_rate
                                                                                                  (nidagmx. task modules.channels.ci channel.CIChannel
              (nidagmx. task modules.channels.ci channel.CIChannel attribute), 137
```

```
ci_count_edges_gate_enable
                                                                                                                                                                           attribute), 138
                        (nidaqmx._task_modules.channels.ci_channel.CIChadantal_xfer_req_cond (nidaqmx._task_modules.channels.ci_channel.CICh
                        attribute), 137
                                                                                                                                                                           attribute), 138
ci_count_edges_gate_logic_lvl_behavior
                                                                                                                                                  ci_dup_count_prevention
                        (nidaqmx._task_modules.channels.ci_channel.CIChannel (nidaqmx._task_modules.channels.ci_channel.CIChannel
                        attribute), 137
                                                                                                                                                                           attribute), 138
                                                                                                                                                  ci duty cycle dig fltr enable
ci count edges gate term
                        (nidaqmx.\_task\_modules.channels.ci\_channel.CIChannel \\ (nidaqmx.\_task\_modules.channels.ci\_channel.CIChannel \\ (nidaqmx.\_task\_modules.channels.ci\_channel.CIChannel \\ (nidaqmx.\_task\_modules.channels.ci\_channels.ci\_channel.CIChannel \\ (nidaqmx.\_task\_modules.channels.ci\_channels.ci\_channel.CIChannel \\ (nidaqmx.\_task\_modules.channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channe
                        attribute), 137
                                                                                                                                                                           attribute), 138
                                                                                                                                                  ci_duty_cycle_dig_fltr_min_pulse_width
ci_count_edges_gate_term_cfg
                        (nidagmx._task_modules.channels.ci_channel.CIChannel (nidagmx._task_modules.channels.ci_channel.CIChannel
                       attribute), 137
                                                                                                                                                                           attribute), 138
ci_count_edges_gate_when
                                                                                                                                                  ci_duty_cycle_dig_fltr_timebase_rate
                        (nidaqmx._task_modules.channels.ci_channel.CIChannel (nidaqmx._task_modules.channels.ci_channel.CIChannel
                        attribute), 137
                                                                                                                                                                           attribute), 138
ci_count_edges_initial_cnt
                                                                                                                                                  ci_duty_cycle_dig_fltr_timebase_src
                        (nidaqmx._task_modules.channels.ci_channel.CIChannel (nidaqmx._task_modules.channels.ci_channel.CIChannel
                        attribute), 137
                                                                                                                                                                           attribute), 138
ci_count_edges_logic_lvl_behavior
                                                                                                                                                  ci_duty_cycle_logic_lvl_behavior
                        (nidaqmx.\_task\_modules.channels.ci\_channel.CIChannel \\ (nidaqmx.\_task\_modules.channels.ci\_channel.CIChannel \\ (nidaqmx.\_task\_modules.channels.ci\_channel.CIChannel \\ (nidaqmx.\_task\_modules.channels.ci\_channels.ci\_channel \\ (nidaqmx.\_task\_modules.channels.ci\_channels.ci\_channel \\ (nidaqmx.\_task\_modules.channels.ci\_channels.ci\_channel \\ (nidaqmx.\_task\_modules.channels.ci\_channels.ci\_channel \\ (nidaqmx.\_task\_modules.channels.ci\_channels.ci\_channel \\ (nidaqmx.\_task\_modules.channels.ci\_channels.ci\_channel \\ (nidaqmx.\_task\_modules.channels.ci\_channel \\ (nidaqmx.\_task\_modules.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.channels.ch
                        attribute), 137
                                                                                                                                                                           attribute), 138
ci_count_edges_term (nidaqmx._task_modules.channels.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_chantel.ci_
                        attribute), 137
                                                                                                                                                                           (nidaqmx._task_modules.channels.ci_channel.CIChannel
ci_count_edges_term_cfg
                                                                                                                                                                           attribute), 138
                        (nidagmx, task modules, channels, ci channel, CIChannel, cycle term (nidagmx, task modules, channels, ci channel, CIChannel
                        attribute), 137
                                                                                                                                                                           attribute), 138
ci_ctr_timebase_active_edge
                                                                                                                                                  ci_duty_cycle_term_cfg (nidaqmx._task_modules.channels.ci_channel.CIC
                        (nidaqmx._task_modules.channels.ci_channel.CIChannel attribute), 138
                        attribute), 137
                                                                                                                                                  ci_encoder_a_input_dig_fltr_enable
ci_ctr_timebase_dig_fltr_enable
                                                                                                                                                                           (nidaqmx._task_modules.channels.ci_channel.CIChannel
                        (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                                                                                         attribute), 138
                        attribute), 137
                                                                                                                                                  ci_encoder_a_input_dig_fltr_min_pulse_width
ci_ctr_timebase_dig_fltr_min_pulse_width
                                                                                                                                                                           (nidaqmx._task_modules.channels.ci_channel.CIChannel
                        (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                                                                                          attribute), 138
                        attribute), 137
                                                                                                                                                  ci encoder a input dig fltr timebase rate
                                                                                                                                                                           (nidaqmx._task_modules.channels.ci_channel.CIChannel
ci ctr timebase dig fltr timebase rate
                        (nidagmx. task modules.channels.ci channel.CIChannel attribute), 139
                        attribute), 137
                                                                                                                                                  ci_encoder_a_input_dig_fltr_timebase_src
ci_ctr_timebase_dig_fltr_timebase_src
                                                                                                                                                                           (nidaqmx._task_modules.channels.ci_channel.CIChannel
                        (nidaqmx._task_modules.channels.ci_channel.CIChannel attribute), 139
                        attribute), 137
                                                                                                                                                  ci_encoder_a_input_dig_sync_enable
ci ctr timebase dig sync enable
                                                                                                                                                                           (nidagmx. task modules.channels.ci channel.CIChannel
                        (nidagmx. task modules.channels.ci channel.CIChannel
                                                                                                                                                                          attribute), 139
                        attribute), 137
                                                                                                                                                  ci_encoder_a_input_logic_lvl_behavior
ci_ctr_timebase_master_timebase_div
                                                                                                                                                                           (nidaqmx._task_modules.channels.ci_channel.CIChannel
                        (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                                                                                         attribute), 139
                        attribute), 138
                                                                                                                                                  ci_encoder_a_input_term
ci_ctr_timebase_rate (nidaqmx._task_modules.channels.ci_channel.CIGhdaqmek._task_modules.channels.ci_channel
                                                                                                                                                                           attribute), 139
                        attribute), 138
ci_ctr_timebase_src (nidaqmx._task_modules.channels.ci_channedddlchanimedut_term_cfg
                        attribute), 138
                                                                                                                                                                           (nidaqmx.\_task\_modules.channels.ci\_channel.CIChannel
ci_custom_scale (nidaqmx._task_modules.channels.ci_channel.CIChaattteibute), 139
                                                                                                                                                  ci encoder b input dig fltr enable
                        attribute), 138
ci data xfer mech (nidagmx, task modules.channels.ci channel.CIChannel
```

```
attribute), 139
                                                                                    ci_encoder_z_input_term_cfg
                                                                                                   (nidaqmx._task_modules.channels.ci_channel.CIChannel
ci_encoder_b_input_dig_fltr_min_pulse_width
              (nidagmx. task modules.channels.ci channel.CIChannel attribute), 140
              attribute), 139
                                                                                    ci_freq_dig_fltr_enable (nidaqmx._task_modules.channels.ci_channel.CICh
ci_encoder_b_input_dig_fltr_timebase_rate
                                                                                                   attribute), 140
              (nidagmx. task modules.channels.ci channel.CIChafrael dig fltr min pulse width
                                                                                                   (nidagmx. task modules.channels.ci channel.CIChannel
              attribute), 139
                                                                                                   attribute), 140
ci_encoder_b_input_dig_fltr_timebase_src
              (nidagmx. task modules.channels.ci channel.CIChafneel dig fltr timebase rate
                                                                                                   (nidaqmx._task_modules.channels.ci_channel.CIChannel
              attribute), 139
ci_encoder_b_input_dig_sync_enable
                                                                                                   attribute), 140
              (nidaqmx._task_modules.channels.ci_channel.CIChafmel_dig_fltr_timebase_src
              attribute), 139
                                                                                                   (nidaqmx._task_modules.channels.ci_channel.CIChannel
ci_encoder_b_input_logic_lvl_behavior
                                                                                                   attribute), 140
              (nidaqmx._task_modules.channels.ci_channel.CIChafnael_dig_sync_enable
              attribute), 139
                                                                                                   (nidaqmx._task_modules.channels.ci_channel.CIChannel
ci_encoder_b_input_term
                                                                                                   attribute), 140
              (nidagmx, task modules, channels, ci channel, CIChannel div (nidagmx, task modules, channels, ci channel, CIChannel
                                                                                                   attribute), 140
              attribute), 139
ci_encoder_b_input_term_cfg
                                                                                    ci freq enable averaging
              (nidagmx._task_modules.channels.ci_channel.CIChannel (nidagmx._task_modules.channels.ci_channel.CIChannel
              attribute), 139
                                                                                                   attribute), 140
ci_encoder_decoding_type
                                                                                    ci_freq_logic_lvl_behavior
              (nidagmx, task modules,channels,ci channel.CIChannel (nidagmx, task modules,channels,ci channel.CIChannel
              attribute), 139
                                                                                                   attribute), 140
ci_encoder_z_index_enable
                                                                                    ci freq meas meth (nidagmx. task modules.channels.ci channel.CIChanr
              (nidagmx._task_modules.channels.ci_channel.CIChannel attribute), 140
              attribute), 139
                                                                                    ci_freq_meas_time (nidaqmx._task_modules.channels.ci_channel.CIChann
                                                                                                   attribute), 141
ci_encoder_z_index_phase
              (nidaqmx._task_modules.channels.ci_channel.CIChafned_starting_edge (nidaqmx._task_modules.channels.ci_channel.CICha
              attribute), 139
                                                                                                   attribute), 141
ci_encoder_z_index_val (nidagmx._task_modules.channels.ci_fatamek._task_modules.channels.ci_channel.CIChannel
              attribute), 140
                                                                                                   attribute), 141
ci_encoder_z_input_dig_fltr_enable
                                                                                    ci_freq_term_cfg (nidaqmx._task_modules.channels.ci_channel.CIChannel
              (nidagmx. task modules.channels.ci channel.CIChannel attribute), 141
                                                                                    ci_freq_units (nidaqmx._task_modules.channels.ci_channel.CIChannel
              attribute), 140
                                                                                                   attribute), 141
ci_encoder_z_input_dig_fltr_min_pulse_width
              (nidaqmx._task_modules.channels.ci_channel.CIChannel.CIChannel.ci_channels.ci_channel.CIChannel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_channel.ci_
              attribute), 140
                                                                                                   attribute), 141
ci_encoder_z_input_dig_fltr_timebase_rate
                                                                                    ci_gps_sync_src (nidaqmx._task_modules.channels.ci_channel.CIChannel
              (nidagmx. task modules.channels.ci channel.CIChannel attribute), 141
              attribute), 140
                                                                                    ci lin encoder dist per pulse
ci encoder z input dig fltr timebase src
                                                                                                   (nidagmx. task modules.channels.ci channel.CIChannel
              (nidaqmx._task_modules.channels.ci_channel.CIChannel attribute), 141
              attribute), 140
                                                                                     ci_lin_encoder_initial_pos
ci_encoder_z_input_dig_sync_enable
                                                                                                   (nidaqmx._task_modules.channels.ci_channel.CIChannel
              (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                  attribute), 141
              attribute), 140
                                                                                    ci_lin_encoder_units (nidaqmx._task_modules.channels.ci_channel.CIChar
ci_encoder_z_input_logic_lvl_behavior
                                                                                                   attribute), 141
              (nidagmx_task_modules.channels.ci_channel.CIChannel (nidagmx_task_modules.channels.ci_channel.CIChannel
              attribute), 140
                                                                                                   attribute), 141
                                                                                    ci_max_meas_period (nidaqmx._task_modules.channels.ci_channel.CIChan
ci encoder z input term
              (nidagmx. task modules.channels.ci channel.CIChannel attribute), 141
              attribute), 140
                                                                                    ci max size (nidagmx.system.device.Device attribute),
```

```
101
                                                                                                                                                                           attribute), 102
ci_max_timebase (nidaqmx.system.device.Device at- ci_prescaler (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                                                                                           attribute), 143
                        tribute), 101
ci_meas_type (nidaqmx._task_modules.channels.ci_channels.ci_channels.ci_fathsanfreq_dig_fltr_enable
                        attribute), 141
                                                                                                                                                                           (nidaqmx._task_modules.channels.ci_channel.CIChannel
ci meas types (nidagmx.system.device.Device attribute),
                                                                                                                                                                           attribute), 143
                                                                                                                                                  ci_pulse_freq_dig_fltr_min_pulse_width
ci_meas_types (nidaqmx.system.physical_channel.PhysicalChannel (nidaqmx._task_modules.channels.ci_channel.CIChannel
                        attribute), 105
                                                                                                                                                                           attribute), 143
ci_mem_map_enable (nidaqmx._task_modules.channels.ci_chapulse_Cfleanadige_lfltr_timebase_rate
                                                                                                                                                                           (nidaqmx._task_modules.channels.ci_channel.CIChannel
                        attribute), 141
ci_min (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                                                                                           attribute), 143
                        attribute), 142
                                                                                                                                                  ci_pulse_freq_dig_fltr_timebase_src
ci_num_possibly_invalid_samps
                                                                                                                                                                           (nidaqmx._task_modules.channels.ci_channel.CIChannel
                        (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                                                                                         attribute), 143
                                                                                                                                                  ci_pulse_freq_dig_sync_enable
                        attribute), 142
ci_output_state (nidaqmx._task_modules.channels.ci_channel.CIChannel.CIChannel.ci_channels.ci_channel.CIChannel
                        attribute), 142
                                                                                                                                                                           attribute), 143
ci_period_dig_fltr_enable
                                                                                                                                                  ci_pulse_freq_logic_lvl_behavior
                        (nidagmx._task_modules.channels.ci_channel.CIChannel (nidagmx._task_modules.channels.ci_channel.CIChannel
                        attribute), 142
                                                                                                                                                                           attribute), 143
ci_period_dig_fltr_min_pulse_width
                                                                                                                                                  ci_pulse_freq_starting_edge
                        (nidagmx._task_modules.channels.ci_channel.CIChannel (nidagmx._task_modules.channels.ci_channel.CIChannel
                        attribute), 142
                                                                                                                                                                           attribute), 143
ci_period_dig_fltr_timebase_rate
                                                                                                                                                  ci\_pulse\_freq\_term (nidaqmx.\_task\_modules.channels.ci\_channel.CIChannels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_
                        (nidaqmx._task_modules.channels.ci_channel.CIChannel attribute), 143
                        attribute), 142
                                                                                                                                                  ci_pulse_freq_term_cfg (nidaqmx._task_modules.channels.ci_channel.CIC
ci_period_dig_fltr_timebase_src
                                                                                                                                                                           attribute), 143
                        (nidagmx._task_modules.channels.ci_channel.CIChannels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.
                        attribute), 142
                                                                                                                                                                           attribute), 143
ci_period_dig_sync_enable
                                                                                                                                                  ci_pulse_ticks_dig_fltr_enable
                        (nidagmx._task_modules.channels.ci_channel.CIChannel (nidagmx._task_modules.channels.ci_channel.CIChannel
                        attribute), 142
                                                                                                                                                                           attribute), 143
ci_period_div (nidaqmx._task_modules.channels.ci_channels.ci_fchannels.ci_fchannels.ci_fchannels.ci_channels.ci_channels.ci_channels.ci_channels.ci_fchannels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_fchannels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_
                        attribute), 142
                                                                                                                                                                           (nidagmx. task modules.channels.ci channel.CIChannel
                                                                                                                                                                           attribute), 143
ci_period_enable_averaging
                        (nidaqmx._task_modules.channels.ci_channel.CIGhanulsle_ticks_dig_fltr_timebase_rate
                        attribute), 142
                                                                                                                                                                           (nidaqmx._task_modules.channels.ci_channel.CIChannel
ci_period_logic_lvl_behavior
                                                                                                                                                                           attribute), 143
                        (nidaqmx._task_modules.channels.ci_channel.CI@hannelse_ticks_dig_fltr_timebase_src
                                                                                                                                                                           (nidagmx. task modules.channels.ci channel.CIChannel
                        attribute), 142
ci period meas meth (nidagmx. task modules.channels.ci channel.GlClhutne) 143
                        attribute), 142
                                                                                                                                                  ci pulse ticks dig sync enable
ci_period_meas_time (nidaqmx._task_modules.channels.ci_channel.ClChannel.ClChannel.ClChannel
                        attribute), 142
                                                                                                                                                                           attribute), 143
ci_period_starting_edge (nidaqmx._task_modules.channels.ci_phulsmeti.cktChagnineellvl_behavior
                        attribute), 142
                                                                                                                                                                           (nidaqmx._task_modules.channels.ci_channel.CIChannel
ci_period_term (nidaqmx._task_modules.channels.ci_channel.CIChanateribute), 144
                        attribute), 142
                                                                                                                                                  ci_pulse_ticks_starting_edge
ci_period_term_cfg (nidaqmx._task_modules.channels.ci_channel.CIQhadaurhx._task_modules.channels.ci_channel
                                                                                                                                                                           attribute), 144
                        attribute), 142
ci_period_units (nidaqmx._task_modules.channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci
                        attribute), 143
                                                                                                                                                                           attribute), 144
ci_physical_chans
                                                              (nidaqmx.system.device.Device ci_pulse_ticks_term_cfg (nidaqmx._task_modules.channels.ci_channel.CIC
```

```
attribute), 144
                                                                                                                                                   ci pulse width units (nidagmx. task modules.channels.ci channel.CIChan
ci_pulse_time_dig_fltr_enable
                                                                                                                                                                             attribute), 145
                        (nidagmx. task modules.channels.ci channel.CIChasaneb clk overrun behavior
                         attribute), 144
                                                                                                                                                                             (nidaqmx._task_modules.channels.ci_channel.CIChannel
ci_pulse_time_dig_fltr_min_pulse_width
                                                                                                                                                                             attribute), 145
                         (nidagmx. task modules.channels.ci channel.CIChasanel) clk overrun sentinel val
                         attribute), 144
                                                                                                                                                                             (nidagmx. task modules.channels.ci channel.CIChannel
ci_pulse_time_dig_fltr_timebase_rate
                                                                                                                                                                             attribute), 145
                         (nidagmx, task modules.channels.ci channel.CIChasana) clk supported (nidagmx.system.device.Device
                         attribute), 144
                                                                                                                                                                             attribute), 102
ci_pulse_time_dig_fltr_timebase_src
                                                                                                                                                    ci_samp_modes
                                                                                                                                                                                                   (nidaqmx.system.device.Device
                                                                                                                                                                                                                                                                                           at-
                        (nidagmx._task_modules.channels.ci_channel.CIChannel tribute), 102
                         attribute), 144
                                                                                                                                                   ci_semi_period_dig_fltr_enable
ci_pulse_time_dig_sync_enable
                                                                                                                                                                             (nidaqmx._task_modules.channels.ci_channel.CIChannel
                        (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                                                                                           attribute), 145
                         attribute), 144
                                                                                                                                                    ci_semi_period_dig_fltr_min_pulse_width
ci_pulse_time_logic_lvl_behavior
                                                                                                                                                                             (nidaqmx._task_modules.channels.ci_channel.CIChannel
                        (nidagmx. task modules.channels.ci channel.CIChannel
                                                                                                                                                                            attribute), 145
                        attribute), 144
                                                                                                                                                   ci_semi_period_dig_fltr_timebase_rate
ci_pulse_time_starting_edge
                                                                                                                                                                             (nidagmx. task modules.channels.ci channel.CIChannel
                         (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                                                                                            attribute), 145
                                                                                                                                                   ci_semi_period_dig_fltr_timebase_src
                         attribute), 144
ci\_pulse\_time\_term (nidaqmx.\_task\_modules.channels.ci\_channel.CIQ \verb| this| task\_modules.channels.ci\_channel.CIQ \verb| this| task\_modules.channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_channels.ci\_ch
                         attribute), 144
                                                                                                                                                                             attribute), 145
ci pulse time term cfg (nidaqmx. task modules.channels.ci schaninels/ci schannels/ci schannels/c
                         attribute), 144
                                                                                                                                                                             (nidagmx. task modules.channels.ci channel.CIChannel
ci_pulse_time_units (nidaqmx._task_modules.channels.ci_channel.CI@thribuet), 145
                         attribute), 144
                                                                                                                                                   ci_semi_period_logic_lvl_behavior
                                                                                                                                                                             (nidaqmx._task_modules.channels.ci_channel.CIChannel
ci_pulse_width_dig_fltr_enable
                         (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                                                                                            attribute), 145
                         attribute), 144
                                                                                                                                                    ci_semi_period_starting_edge
ci_pulse_width_dig_fltr_min_pulse_width
                                                                                                                                                                             (nidaqmx._task_modules.channels.ci_channel.CIChannel
                        (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                                                                                            attribute), 145
                        attribute), 144
                                                                                                                                                   ci_semi_period_term (nidaqmx._task_modules.channels.ci_channel.CIChan
ci_pulse_width_dig_fltr_timebase_rate
                                                                                                                                                                             attribute), 145
                         (nidaqmx._task_modules.channels.ci_channel.CIChasereli_period_term_cfg
                        attribute), 144
                                                                                                                                                                             (nidagmx. task modules.channels.ci channel.CIChannel
ci_pulse_width_dig_fltr_timebase_src
                                                                                                                                                                             attribute), 145
                         (nidagmx._task_modules.channels.ci_channel.CIChasmel_period_units (nidagmx._task_modules.channels.ci_channel.CICha
                        attribute), 144
                                                                                                                                                                             attribute), 145
                                                                                                                                                   ci tc reached (nidagmx. task modules.channels.ci channel.CIChannel
ci_pulse_width_dig_sync_enable
                         (nidaqmx._task_modules.channels.ci_channel.CIChannel attribute), 145
                         attribute), 145
                                                                                                                                                   ci thresh voltage (nidagmx. task modules.channels.ci channel.CIChannel
ci_pulse_width_logic_lvl_behavior
                                                                                                                                                                             attribute), 146
                         (nidaqmx._task_modules.channels.ci_channel.CIChatinedstamp_initial_seconds
                         attribute), 145
                                                                                                                                                                             (nidaqmx._task_modules.channels.ci_channel.CIChannel
ci_pulse_width_starting_edge
                                                                                                                                                                             attribute), 146
                        (nidagmx._task_modules.channels.ci_channel.CIChannelstamp_units (nidagmx._task_modules.channels.ci_channel.CIChannelstamp_units (nidagmx._task_modules.channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci_channels.ci
                         attribute), 145
                                                                                                                                                                             attribute), 146
ci_pulse_width_term (nidaqmx._task_modules.channels.ci_chatmella@hamelaqmx.system.device.Device attribute),
                         attribute), 145
ci pulse width term cfg
                                                                                                                                                   ci_two_edge_sep_first_dig_fltr_enable
                         (nidagmx, task modules,channels,ci channel.CIChannel (nidagmx, task modules,channels,ci channel.CIChannel
                        attribute), 145
                                                                                                                                                                             attribute), 146
```

```
ci two edge sep first dig fltr min pulse width
                                                                                                  attribute), 147
              (nidagmx, task modules, channels, ci channel, CIChanstelx fer req size (nidagmx, task modules, channels, ci channel, CIChanstelx fer req size (nidagmx, task modules, channels, ci channel, CIChanstelx fer req size (nidagmx, task modules, channels, ci ch
                                                                                                  attribute), 147
              attribute), 146
ci_two_edge_sep_first_dig_fltr_timebase_rate
                                                                                    ci_velocity_a_input_dig_fltr_enable
                                                                                                 (nidaqmx._task_modules.channels.ci_channel.CIChannel
              (nidagmx. task modules.channels.ci channel.CIChannel
              attribute), 146
                                                                                                  attribute), 147
ci two edge sep first dig fltr timebase src
                                                                                    ci velocity a input dig fltr min pulse width
              (nidagmx, task modules,channels,ci channel.CIChannel (nidagmx, task modules,channels,ci channel.CIChannel
              attribute), 146
                                                                                                  attribute), 147
ci_two_edge_sep_first_dig_sync_enable
                                                                                    ci_velocity_a_input_dig_fltr_timebase_rate
              (nidagmx._task_modules.channels.ci_channel.CIChannel (nidagmx._task_modules.channels.ci_channel.CIChannel
              attribute), 146
                                                                                                  attribute), 147
ci_two_edge_sep_first_edge
                                                                                    ci_velocity_a_input_dig_fltr_timebase_src
              (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                 (nidaqmx._task_modules.channels.ci_channel.CIChannel
              attribute), 146
                                                                                                  attribute), 147
ci_two_edge_sep_first_logic_lvl_behavior
                                                                                    ci_velocity_a_input_logic_lvl_behavior
              (nidagmx._task_modules.channels.ci_channel.CIChannel (nidagmx._task_modules.channels.ci_channel.CIChannel
              attribute), 146
                                                                                                  attribute), 147
ci_two_edge_sep_first_term
                                                                                    ci velocity a input term
              (nidagmx, task modules.channels.ci channel.CIChannel (nidagmx, task modules.channels.ci channel.CIChannel
              attribute), 146
                                                                                                  attribute), 147
ci_two_edge_sep_first_term_cfg
                                                                                    ci_velocity_a_input_term_cfg
              (nidagmx._task_modules.channels.ci_channel.CIChannel (nidagmx._task_modules.channels.ci_channel.CIChannel
              attribute), 146
                                                                                                  attribute), 147
ci_two_edge_sep_second_dig_fltr_enable
                                                                                    ci_velocity_ang_encoder_pulses_per_rev
              (nidagmx. task modules.channels.ci channel.CIChannel (nidagmx. task modules.channels.ci channel.CIChannel
              attribute), 146
                                                                                                  attribute), 147
ci_two_edge_sep_second_dig_fltr_min_pulse_width
                                                                                    ci_velocity_ang_encoder_units
              (nidaqmx._task_modules.channels.ci_channel.CIChannel (nidaqmx._task_modules.channels.ci_channel.CIChannel
              attribute), 146
                                                                                                  attribute), 147
ci_two_edge_sep_second_dig_fltr_timebase_rate
                                                                                    ci_velocity_b_input_dig_fltr_enable
              (nidagmx._task_modules.channels.ci_channel.CIChannel (nidagmx._task_modules.channels.ci_channel.CIChannel
              attribute), 146
                                                                                                  attribute), 147
ci_two_edge_sep_second_dig_fltr_timebase_src
                                                                                    ci_velocity_b_input_dig_fltr_min_pulse_width
              (nidagmx. task modules.channels.ci channel.CIChannel
                                                                                                 (nidagmx. task modules.channels.ci channel.CIChannel
              attribute), 146
                                                                                                  attribute), 147
ci two edge sep second dig sync enable
                                                                                    ci velocity b input dig fltr timebase rate
              (nidaqmx._task_modules.channels.ci_channel.CIChannel (nidaqmx._task_modules.channels.ci_channel.CIChannel
              attribute), 146
                                                                                                  attribute), 147
ci_two_edge_sep_second_edge
                                                                                    ci_velocity_b_input_dig_fltr_timebase_src
              (nidagmx, task modules,channels,ci channel.CIChannel (nidagmx, task modules,channels,ci channel.CIChannel
                                                                                                  attribute), 147
              attribute), 146
ci two edge sep second logic lvl behavior
                                                                                    ci velocity b input logic lvl behavior
              (nidaqmx._task_modules.channels.ci_channel.CIChannel
                                                                                                 (nidaqmx._task_modules.channels.ci_channel.CIChannel
              attribute), 146
                                                                                                  attribute), 148
ci_two_edge_sep_second_term
                                                                                    ci_velocity_b_input_term
              (nidagmx._task_modules.channels.ci_channel.CIChannel (nidagmx._task_modules.channels.ci_channel.CIChannel
              attribute), 147
                                                                                                  attribute), 148
ci_two_edge_sep_second_term_cfg
                                                                                    ci_velocity_b_input_term_cfg
              (nidagmx._task_modules.channels.ci_channel.CIChannel (nidagmx._task_modules.channels.ci_channel.CIChannel
              attribute), 147
                                                                                                  attribute), 148
ci_two_edge_sep_units (nidaqmx._task_modules.channels.ci<u>_challoweilyCBCh(mindslq</u>mx._task_modules.channels.ci_channel
              attribute), 147
                                                                                                  attribute), 148
ci usb xfer req count (nidaqmx. task modules.channels.crickverhoveit) Clerikovanlovel decoding type
```

```
(nidaqmx._task_modules.channels.ci_channel.CIChannel (nidaqmx._task_modules.channels.co_channel.COChannel
                       attribute), 148
                                                                                                                                                                     attribute), 149
ci_velocity_lin_encoder_dist_per_pulse
                                                                                                                                            co_ctr_timebase_master_timebase_div
                       (nidaqmx._task_modules.channels.ci_channel.CIChannel (nidaqmx._task_modules.channels.co_channel.COChannel
                       attribute), 148
                                                                                                                                                                    attribute), 149
ci velocity lin encoder units
                                                                                                                                            co ctr timebase rate (nidagmx. task modules.channels.co channel.COCh
                       (nidagmx. task modules.channels.ci channel.CIChannel attribute), 150
                       attribute), 148
                                                                                                                                            co_ctr_timebase_src (nidaqmx._task_modules.channels.co_channel.COCha
ci_velocity_meas_time (nidaqmx._task_modules.channels.ci_channels.ci_channels.ci
                       attribute), 148
                                                                                                                                            co_data_xfer_mech (nidaqmx._task_modules.channels.co_channel.COChan
CIChannel (class in nidagmx._task_modules.channels.ci_channel),
                                                                                                                                                                    attribute), 150
                       135
                                                                                                                                            co_data_xfer_req_cond (nidagmx._task_modules.channels.co_channel.COC
CIChannelCollection
                                                                                                                                                                    attribute), 150
                                                                                   (class
                                                                                                                                  in
                       nidaqmx._task_modules.ci_channel_collection), co_enable_initial_delay_on_retrigger
                                                                                                                                                                    (nidaqmx.\_task\_modules.channels.co\_channel.COChannel
                                                                                                                                                                     attribute), 150
CIPhysical Channel Collection\\
                                                                                             (class
                                                                                                                                  in
                       nidaqmx.system._collections.physical_channel_collections)size (nidaqmx.system.device.Device attribute),
                                                                                                                                                                     102
CJCSource (class in nidaqmx.constants), 22
                                                                                                                                            co max timebase
                                                                                                                                                                                                         (nidaqmx.system.device.Device
CLEAR_EXPIRATION (nidaqmx.constants.WDTTaskAction
                                                                                                                                                                    attribute), 102
                       attribute), 50
                                                                                                                                            co_mem_map_enable (nidaqmx._task_modules.channels.co_channel.COCh
clear_expiration() (nidaqmx.system.watchdog.WatchdogTask
                                                                                                                                                                    attribute), 150
                       method), 110
                                                                                                                                             co_output_state (nidaqmx._task_modules.channels.co_channel.COChannel
clear_teds() (nidaqmx.system.physical_channel.PhysicalChannel
                                                                                                                                                                    attribute), 150
                                                                                                                                            co\_output\_type \ (nidaqmx.\_task\_modules.channels.co\_channel.COChannel
                       method), 105
close()
                                  (nidaqmx.system.watchdog.WatchdogTask
                                                                                                                                                                    attribute), 150
                                                                                                                                             co_output_types (nidaqmx.system.device.Device
                       method), 110
close() (nidaqmx.task.Task method), 112
                                                                                                                                                                    tribute), 102
CLOSED (nidaqmx.constants.RelayPosition attribute), co_output_types (nidaqmx.system.physical_channel.PhysicalChannel
                                                                                                                                                                     attribute), 105
co_auto_incr_cnt (nidagmx._task_modules.channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_ch
                       attribute), 149
                                                                                                                                                                    tribute), 102
co_channels (nidaqmx.task.Task attribute), 112
                                                                                                                                            co_prescaler (nidaqmx._task_modules.channels.co_channel.COChannel
co_constrained_gen_mode
                                                                                                                                                                    attribute), 150
                       (nidagmx, task modules, channels, co channel, COGhannel done (nidagmx, task modules, channels, co channel, COChannel
                       attribute), 149
                                                                                                                                                                    attribute), 150
co count (nidagmx. task modules.channels.co channel.COCchannels.co (nidagmx. task modules.channels.co channel.COCchannels.co channels.co c
                       attribute), 149
                                                                                                                                                                    attribute), 150
co_ctr_timebase_active_edge
                                                                                                                                            co_pulse_freq (nidaqmx._task_modules.channels.co_channel.COChannel
                       (nidaqmx._task_modules.channels.co_channel.COChannel attribute), 150
                       attribute), 149
                                                                                                                                            co pulse freq initial delay
co_ctr_timebase_dig_fltr_enable
                                                                                                                                                                    (nidagmx. task modules.channels.co channel.COChannel
                       (nidagmx. task modules.channels.co channel.COChannel attribute), 150
                       attribute), 149
                                                                                                                                            co_pulse_freq_units (nidaqmx._task_modules.channels.co_channel.COCha
                                                                                                                                                                    attribute), 150
co_ctr_timebase_dig_fltr_min_pulse_width
                       (nidaqmx._task_modules.channels.co_channel.COChannel.COChannel.COChannel.COChannels.co_channels.co_channels.co
                       attribute), 149
                                                                                                                                                                    attribute), 151
co_ctr_timebase_dig_fltr_timebase_rate
                                                                                                                                            co_pulse_high_time (nidaqmx._task_modules.channels.co_channel.COCha
                       (nidaqmx._task_modules.channels.co_channel.COChannel attribute), 151
                       attribute), 149
                                                                                                                                            co_pulse_idle_state (nidaqmx._task_modules.channels.co_channel.COChan
co_ctr_timebase_dig_fltr_timebase_src
                                                                                                                                                                    attribute), 151
                       (nidaqmx._task_modules.channels.co_channel.COChannel.COChannels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co_channels.co
                       attribute), 149
                                                                                                                                                                    attribute), 151
```

co pulse low time (nidagmx. task modules.channels.co channel.COChar

co\_ctr\_timebase\_dig\_sync\_enable

```
attribute), 151
                                                                                                   method), 92
co pulse term (nidagmx. task modules.channels.co channell@NXTANTelGROUP DELAY
              attribute), 151
                                                                                                   (nidagmx.constants.FilterResponse
co_pulse_ticks_initial_delay
              (nidagmx. task modules.channels.co channel.COCOMSTEANT USER VALUE
              attribute), 151
                                                                                                   (nidagmx.constants.CJCSource
                                                                                                                                                         attribute),
co pulse time initial delay
              (nidagmx. task modules.channels.co channel.COChustraihedGenMode (class in nidagmx.constants), 23
              attribute), 151
                                                                                     CONTINUOUS (nidagmx.constants.AcquisitionType at-
co_pulse_time_units (nidaqmx._task_modules.channels.co_channel.Co)Ghtennel
              attribute), 151
                                                                                     CONTINUOUS
                                                                                                                 (nidaqmx.constants.ScanRepeatMode
co_rdy_for_new_val (nidagmx._task_modules.channels.co_channel.Ca)(ribate):138
              attribute), 151
                                                                                     control()
                                                                                                          (nidaqmx.system.watchdog.WatchdogTask
co_samp_clk_supported (nidaqmx.system.device.Device
                                                                                                   method), 110
                                                                                     control() (nidaqmx.task.Task method), 113
              attribute), 102
co_samp_modes
                           (nidaqmx.system.device.Device
                                                                                     COPhysicalChannelCollection
                                                                                                                                              (class
                                                                                                                                                                    in
              tribute), 102
                                                                                                   nidagmx.system._collections.physical_channel_collection),
co_trig_usage (nidaqmx.system.device.Device attribute),
                                                                                     COULOMBS (nidagmx.constants.ChargeUnits attribute),
co usb xfer req count (nidagmx. task modules.channels.co channel COChannel
              attribute), 151
                                                                                     COULOMBS (nidagmx.constants.UnitsPreScaled at-
co usb xfer req size (nidagmx. task modules.channels.co channel.crowthan red
              attribute), 151
                                                                                     count() (nidaqmx._task_modules.ai_channel_collection.AIChannelCollection
co use only on brd mem
                                                                                                   method), 201
              (nidagmx, task modules.channels.co channel.CQ6hatmehidagmx, task modules.ao channel collection.AOChannelCollect
              attribute), 151
                                                                                                   method), 202
COChannel (class in nidagmx._task_modules.channels.co_channel.collection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelCollection.CIChannelColl
              149
                                                                                                   method), 213
COChannelCollection
                                                                               in count() (nidaqmx._task_modules.co_channel_collection.COChannelCollect
                                                   (class
              nidagmx. task_modules.co_channel_collection),
                                                                                                   method), 215
                                                                                     count() (nidaqmx._task_modules.di_channel_collection.DIChannelCollection
COExpirationState (class in nidagmx.types), 248
                                                                                                   method), 216
COMMIT (nidagmx.constants.Action attribute), 19
                                                                                     count() (nidagmx._task_modules.do_channel_collection.DOChannelCollec
common_mode_range_error_chans
                                                                                                   method), 216
              (nidagmx. task modules.in stream.InStream
                                                                                     COUNT DOWN (nidagmx.constants.CountDirection at-
              attribute), 220
                                                                                                   tribute), 23
common_mode_range_error_chans_exist
                                                                                     COUNT EDGES (nidagmx.constants.UsageTypeCI at-
              (nidaqmx._task_modules.in_stream.InStream
                                                                                                   tribute), 48
              attribute), 220
                                                                                     COUNT UP
                                                                                                           (nidaqmx.constants.CountDirection
COMPACT_DAQ
                               (nidaqmx.constants.BusType
                                                                                                   tribute), 23
                                                                             at-
              tribute), 21
                                                                                     CountDirection (class in nidagmx.constants), 23
COMPACT DAQ CHASSIS
                                                                                     COUNTER INPUT (nidagmx.constants.ChannelType at-
              (nidagmx.constants.ProductCategory attribute),
                                                                                                   tribute), 23
                                                                                     COUNTER_OUTPUT (nidaqmx.constants.ChannelType
compact_daq_chassis_device
                                                                                                   attribute), 23
              (nidaqmx.system.device.Device
                                                                                     COUNTER_OUTPUT_EVENT
                                                                   attribute),
                                                                                                   (nidaqmx.constants.Signal attribute), 39
compact_daq_slot_num (nidaqmx.system.device.Device CounterFrequencyMethod (class in nidaqmx.constants),
              attribute), 102
configure_logging() (nidaqmx._task_modules.in_stream.InSGrounterReader (class in nidaqmx.stream_readers), 62
              method), 220
                                                                                     CounterWriter (class in nidaqmx.stream_writers), 82
configure_teds() (nidaqmx.system.physical_channel.Physical@habhing! (class in nidaqmx.constants), 24
                                                                                     CREATE
                                                                                                       (nidaqmx.constants.LoggingOperation
              method), 105
connect terms()
                                   (nidagmx.system.system.System
                                                                                                   tribute), 31
```

<pre>create_lin_scale() (nidaqmx.scale.Scale static method),</pre>	D
53	DaqError, 51
<pre>create_map_scale() (nidaqmx.scale.Scale static method),</pre>	DaqResourceWarning (in module nidaqmx.errors), 52
53	DaqWarning, 52
CREATE_OR_REPLACE	data_active_event_lvl_active_lvl
(nidaqmx.constants.LoggingOperation at-	(nidaqmxtask_modules.export_signals.ExportSignals
tribute), 31	attribute), 217
create_polynomial_scale() (nidaqmx.scale.Scale static method), 54	data_active_event_output_term
create_table_scale() (nidaqmx.scale.Scale static method),	(nidaqmxtask_modules.export_signals.ExportSignals
54	attribute), 217
ctr_out_event_output_behavior	DataJustification (class in nidaqmx.constants), 24 DataTransferActiveTransferMode (class in
(nidaqmxtask_modules.export_signals.ExportS	DataTransferActiveTransferMode (class in signals nidaqmx.constants), 24
attribute), 217	DC (nidaqmx.constants.Coupling attribute), 24
ctr_out_event_output_term	Deassert Condition (class in pidagmy constants) 25
(nidaqmxtask_modules.export_signals.ExportS	Deassert Condition (class in ineaquix.constants), 25
attribute), 217	tribute), 43
ctr_out_event_pulse_polarity	DEG_C (nidaqmx.constants.TemperatureUnits attribute),
(nidaqmxtask_modules.export_signals.ExportS	Signals 42
attribute), 217	DEG_C (nidaqmx.constants.UnitsPreScaled attribute), 45
ctr_out_event_toggle_idle_state (nidaqmxtask_modules.export_signals.ExportS	DEG_F (nidaqmx.constants.TemperatureUnits attribute),
attribute), 217	- 12
CtrFreq (class in nidaqmx.types), 248	DEG_F (nidaqmx.constants.UnitsPreScaled attribute), 45
CtrTick (class in nidaqmx.types), 248	DEG_R (nidaqmx.constants.TemperatureUnits attribute), 42
CtrTime (class in nidaqmx.types), 248	DEG_R (nidaqmx.constants.UnitsPreScaled attribute), 45
curr_read_pos (nidaqmxtask_modules.in_stream.InStream	DEG_R (mdaqmx.constants.Omts) rescaled attribute), 45
attribute), 221	DEGREES (nidaqmx.constants.UnitsPreScaled at-
curr_write_pos (nidaqmxtask_modules.out_stream.OutSt	tream tribute), 45
attribute), 226	DEGREES PER SECOND
CURRENT (nidaqmx.constants.AOPowerUpOutputBehav	ior (nidaqmx.constants.AngularVelocityUnits
attribute), 18	attribute), 19
CURRENT (nidaqmx.constants.UsageTypeAI attribute), 47	DEGREES_PER_SECOND
CURRENT (nidaqmx.constants.UsageTypeAO attribute),	(nidaqmx.constants.UnitsPreScaled attribute),
48	45
CURRENT (nidaqmx.constants.WatchdogAOExpirState	delay (nidaqmxtask_modules.triggering.reference_trigger.ReferenceTrigg
attribute), 50	attribute), 243 delay (nidaqmxtask_modules.triggering.start_trigger.StartTrigger
CURRENT_ACRMS (nidaqmx.constants.UsageTypeAI	attribute), 246
attribute), 47	delay_from_samp_clk_delay
CURRENT_READ_POSITION	(nidaqmxtask_modules.timing.Timing
(nidaqmx.constants.ReadRelativeTo attribute),	attribute), 233
36	delay_from_samp_clk_delay_units
CURRENT_WRITE_POSITION	(nidaqmxtask_modules.timing.Timing
(nidaqmx.constants.WriteRelativeTo attribute), 51	attribute), 233
CurrentShuntResistorLocation (class in	delay_units (nidaqmxtask_modules.triggering.start_trigger.StartTrigger
nidaqmx.constants), 24	attribute), 246
CurrentUnits (class in nidaqmx.constants), 24	delete() (nidaqmx.system.storage.persisted_channel.PersistedChannel method), 107
CUSTOM (nidaqmx.constants.ADCTimingMode at-	delete() (nidaqmx.system.storage.persisted_scale.PersistedScale
tribute), 17	method), 108
CUSTOM (nidaqmx.constants.FilterType attribute), 28	delete() (nidaqmx.system.storage.persisted_task.PersistedTask
CUSTOM (nidaqmx.constants.RTDType attribute), 35	method), 108
	delete network device() (nidagmx.system.device.Device

```
method), 102
                                                                   attribute), 153
DELTA (nidaqmx.constants.StrainGageRosetteType at- di_dig_fltr_timebase_rate
                                                                   (nidagmx. task modules.channels.di channel.DIChannel
description (nidaqmx._task_modules.channels.ai_channel.AIChannel attribute), 153
         attribute), 130
                                                         di_dig_fltr_timebase_src (nidaqmx._task_modules.channels.di_channel.DIG
description (nidagmx. task modules.channels.ao channel.AOChannehttribute), 153
         attribute), 134
                                                         di dig sync enable (nidaqmx. task modules.channels.di channel.DIChan
description (nidaqmx._task_modules.channels.channel.Channel
                                                                  attribute), 153
                                                         di_invert_lines (nidaqmx._task_modules.channels.di_channel.DIChannel
         attribute), 118
description (nidaqmx._task_modules.channels.ci_channel.CIChannel attribute), 153
                                                         di_lines (nidaqmx.system.device.Device attribute), 103
         attribute), 148
description (nidaqmx._task_modules.channels.co_channel.COChannel (nidaqmx._task_modules.channels.di_channel.DIChannel
         attribute), 151
                                                                  attribute), 153
description (nidagmx._task_modules.channels.di_channel.DdlChannelrate (nidagmx.system.device.Device attribute),
         attribute), 152
description (nidagmx._task_modules.channels.do_channel.DIQ@hammehap_enable (nidagmx._task_modules.channels.di_channels.DICha
         attribute), 154
                                                                  attribute), 153
description (nidagmx.scale.Scale attribute), 54
                                                         di num booleans per chan
DEU (nidaqmx.constants.Language attribute), 30
                                                                  (nidaqmx._task_modules.in_stream.InStream
dev is simulated (nidagmx.system.device.Device
                                                                  attribute), 221
                                                         di_num_lines (nidaqmx._task_modules.channels.di_channel.DIChannel
         tribute), 102
dev serial num
                  (nidaqmx.system.device.Device
                                                    at-
                                                                  attribute), 153
         tribute), 102
                                                         di_port_width (nidaqmx.system.physical_channel.PhysicalChannel
Device (class in nidagmx.system.device), 98
                                                                  attribute), 106
device_names (nidaqmx.system._collections.device_collection_Daviscetidalperiscnystem.device.Device attribute), 103
         attribute), 96
                                                         di samp clk supported (nidagmx.system.physical channel.PhysicalChannel
DeviceCollection
                                (class
                                                     in
                                                                   attribute), 106
         nidaqmx.system._collections.device_collection), di_samp_modes (nidaqmx.system.physical_channel.PhysicalChannel
                                                                  attribute), 106
devices (nidaqmx.system.system.System attribute), 92
                                                         di_trig_usage (nidaqmx.system.device.Device attribute),
devices (nidagmx.task.Task attribute), 113
devs_with_inserted_or_removed_accessories
                                                         di_tristate (nidaqmx._task_modules.channels.di_channel.DIChannel
         (nidaqmx._task_modules.in_stream.InStream
                                                                  attribute), 153
         attribute), 221
                                                         di_usb_xfer_req_count (nidaqmx._task_modules.channels.di_channel.DICh
                                                                  attribute), 153
devs_with_inserted_or_removed_accessories
         (nidagmx._task_modules.out_stream.OutStream_di_usb_xfer_req_size (nidagmx._task_modules.channels.di_channel.DICha
                                                                  attribute), 153
         attribute), 226
di_acquire_on (nidaqmx._task_modules.channels.di_channelDDIGhammeldclass in nidaqmx._task_modules.channels.di_channel),
         attribute), 152
                                                                   152
di_change_detect_supported
                                                         DIChannelCollection
                                                                                           (class
         (nidagmx.system.physical channel.PhysicalChannel
                                                                  nidagmx. task modules.di channel collection),
         attribute), 106
di channels (nidagmx.task.Task attribute), 113
                                                         DIFF (nidagmx.constants.CalibrationTerminalConfig at-
di_data_xfer_mech (nidaqmx._task_modules.channels.di_channel.DIChibutte), 22
                                                         DIFFERENTIAL\ (nidaqmx.constants. Terminal Configuration
         attribute), 152
di_data_xfer_req_cond (nidaqmx._task_modules.channels.di_channelaDifChate)nel3
         attribute), 152
                                                         dig_edge_dig_fltr_enable
di_dig_fltr_enable (nidaqmx._task_modules.channels.di_channel.DIChaidachmx._task_modules.triggering.arm_start_trigger.ArmStartTr
         attribute), 152
                                                                  attribute), 237
                                                         dig_edge_dig_fltr_enable
di_dig_fltr_enable_bus_mode
         (nidaqmx._task_modules.channels.di_channel.DIChannel (nidaqmx._task_modules.triggering.reference_trigger.ReferenceT
                                                                  attribute), 243
         attribute), 152
di_dig_fltr_min_pulse_width
                                                         dig_edge_dig_fltr_enable
         (nidaqmx, task modules, channels, di channel, DIChannel (nidaqmx, task modules, triggering, start trigger, Start Trigger
```

```
attribute), 246
                                                                                                                                                 dig lvl dig fltr timebase rate
                                                                                                                                                                          (nidaqmx._task_modules.triggering.pause_trigger.PauseTrigger
dig_edge_dig_fltr_min_pulse_width
                        (nidagmx. task modules.triggering.arm start trigger.ArmStattfbritg)er239
                        attribute), 237
                                                                                                                                                 dig_lvl_dig_fltr_timebase_src
dig_edge_dig_fltr_min_pulse_width
                                                                                                                                                                          (nidaqmx._task_modules.triggering.pause_trigger.PauseTrigger
                        (nidagmx. task modules.triggering.reference trigger.Referentrieffirigger39
                        attribute), 243
                                                                                                                                                 dig lvl dig sync enable
                                                                                                                                                                          (nidaqmx.\_task\_modules.triggering.pause\_trigger.PauseTrigger\\
dig_edge_dig_fltr_min_pulse_width
                        (nidaqmx._task_modules.triggering.start_trigger.StartTriggeattribute), 239
                                                                                                                                                 dig_lvl_src (nidaqmx._task_modules.triggering.pause_trigger.PauseTrigger
                        attribute), 246
dig_edge_dig_fltr_timebase_rate
                                                                                                                                                                          attribute), 239
                        (nidaqmx._task_modules.triggering.arm_start_trigher.Arlm\shartTrighermx._task_modules.triggering.pause_trigger.PauseTrigs
                        attribute), 237
                                                                                                                                                                          attribute), 239
dig_edge_dig_fltr_timebase_rate
                                                                                                                                                 dig_pattern_pattern (nidaqmx._task_modules.triggering.pause_trigger.Pause
                        (nidagmx._task_modules.triggering.reference_trigger.Referenteiffrigger39
                        attribute), 243
                                                                                                                                                 dig_pattern_pattern (nidaqmx._task_modules.triggering.reference_trigger.R
dig_edge_dig_fltr_timebase_rate
                                                                                                                                                                          attribute), 243
                        (nidagmx_task_modules.triggering.start_trigger.SdigtTrigtgen_pattern (nidagmx_task_modules.triggering.start_trigger.StartT
                        attribute), 246
                                                                                                                                                                          attribute), 247
dig_edge_dig_fltr_timebase_src
                                                                                                                                                 dig_pattern_src (nidaqmx._task_modules.triggering.pause_trigger.PauseTri
                        (nidaqmx._task_modules.triggering.arm_start_trigger.ArmStatttibritgger239
                        attribute), 237
                                                                                                                                                 dig_pattern_src (nidaqmx._task_modules.triggering.reference_trigger.Refer
dig_edge_dig_fltr_timebase_src
                                                                                                                                                                          attribute), 243
                        (nidaqmx._task_modules.triggering.reference_triggier_Naftenence_Triggier_mx._task_modules.triggering.start_trigger.StartTrigg
                        attribute), 243
                                                                                                                                                                          attribute), 247
dig_edge_dig_fltr_timebase_src
                                                                                                                                                 dig_pattern_trig_when (nidaqmx._task_modules.triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_triggering.reference_trigge
                        (nidaqmx._task_modules.triggering.start_trigger.StartTriggeattribute), 243
                        attribute), 247
                                                                                                                                                 dig_pattern_trig_when (nidaqmx._task_modules.triggering.start_trigger.Sta
                                                                                                                                                                          attribute), 247
dig_edge_dig_sync_enable
                        (nidaqmx._task_modules.triggering.arm_start_triggigg.AattuStn_tWlrigggridaqmx._task_modules.triggering.pause_trigger.Pause
                        attribute), 237
                                                                                                                                                                          attribute), 239
dig_edge_dig_sync_enable
                                                                                                                                                 dig_trig_supported (nidaqmx.system.device.Device at-
                        (nidaqmx._task_modules.triggering.reference_trigger.Referenceire)gee8
                                                                                                                                                 DIGITAL_EDGE
                        attribute), 243
                                                                                                                                                                                                              (nidaqmx.constants.TriggerType
dig_edge_dig_sync_enable
                                                                                                                                                                          attribute), 44
                        (nidaqmx._task_modules.triggering.start_trigger.SDAGTifAgerINPUT (nidaqmx.constants.ChannelType at-
                        attribute), 247
                                                                                                                                                                          tribute), 23
dig_edge_edge (nidaqmx._task_modules.triggering.arm_stafb]GilgeeLAlfonS(mit/Riggerconstants.ProductCategory at-
                        attribute), 237
                                                                                                                                                                          tribute), 34
dig_edge_edge (nidaqmx._task_modules.triggering.reference) Initional Control of the Initional Co
                        attribute), 243
                                                                                                                                                                          tribute), 44
dig edge edge (nidagmx. task modules.triggering.start triggering.start tri
                                                                                                                                                                                                           (nidagmx.constants.ChannelType
                        attribute), 247
                                                                                                                                                                          attribute), 23
dig_edge_src (nidaqmx._task_modules.triggering.arm_start_IMIGYFAAr_IRSTaFHRYger (nidaqmx.constants.TriggerType
                        attribute), 237
                                                                                                                                                                          attribute), 44
dig_edge_src (nidaqmx._task_modules.triggering.reference Diigigath Rivfellenge Tolaggein nidaqmx.constants), 25
                        attribute), 243
                                                                                                                                                 DigitalMultiChannelReader
                                                                                                                                                                                                                                                (class
                                                                                                                                                                                                                                                                                        in
dig_edge_src (nidaqmx._task_modules.triggering.start_trigger.StartTriggermx.stream_readers), 72
                                                                                                                                                 DigitalMultiChannelWriter
                        attribute), 247
                                                                                                                                                                                                                                               (class
                                                                                                                                                                                                                                                                                        in
dig_lvl_dig_fltr_enable (nidaqmx._task_modules.triggering.pause_triggida@mxxsflriggerwriters), 87
                        attribute), 239
                                                                                                                                                 DigitalPatternCondition (class in nidaqmx.constants), 25
dig_lvl_dig_fltr_min_pulse_width
                                                                                                                                                 DigitalSingleChannelReader
                                                                                                                                                                                                                                                 (class
                                                                                                                                                                                                                                                                                        in
                        (nidaqmx._task_modules.triggering.pause_trigger.PauseTriggedaqmx.stream_readers), 68
                        attribute), 239
                                                                                                                                                 DigitalSingleChannelWriter
                                                                                                                                                                                                                                                (class
                                                                                                                                                                                                                                                                                        in
```

```
nidagmx.stream writers), 84
                                                                                                                                                                        attribute), 226
DigitalWidthUnits (class in nidaqmx.constants), 25
                                                                                                                                                do num lines (nidagmx. task modules.channels.do channel.DOChannel
DILinesCollection
                                                                                                                                                                        attribute), 155
                                                                                                                                     in
                        nidaqmx.system._collections.physical_channel_collectionput_drive_type (nidaqmx._task_modules.channels.do_channel.DOC
                                                                                                                                                                        attribute), 155
DIPortsCollection
                                                                                  (class
                                                                                                                                     in do overcurrent auto reenable
                        nidagmx.system. collections, physical channel collection), (nidagmx. task modules, channels, do channel, DOChannel
                                                                                                                                                                        attribute), 155
disable ref trig() (nidagmx. task modules.triggering.referedoeotriggerrRuttefhiritægenx. task modules.channels.do channel.DOCh
                        method), 243
                                                                                                                                                                        attribute), 155
disable_start_trig() (nidaqmx._task_modules.triggering.startdtrigger.StartTrigger.astartdtrigger.astartTrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.astartdtrigger.
                        method), 247
                                                                                                                                                                        (nidaqmx._task_modules.channels.do_channel.DOChannel
disconnect terms()
                                                           (nidaqmx.system.system.System
                                                                                                                                                                        attribute), 155
                        method), 92
                                                                                                                                                do_port_width (nidaqmx.system.physical_channel.PhysicalChannel
divided_samp_clk_timebase_output_term
                                                                                                                                                                        attribute), 106
                        (nidaqmx._task_modules.export_signals.ExportSignalsorts (nidaqmx.system.device.Device attribute), 103
                        attribute), 218
                                                                                                                                                do_samp_clk_supported (nidaqmx.system.physical_channel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.PhysicalChannel.Physica
DMA (nidagmx.constants.DataTransferActiveTransferMode
                                                                                                                                                                        attribute), 106
                                                                                                                                                do samp modes (nidagmx.system.physical channel.PhysicalChannel
                        attribute), 24
do channels (nidagmx.task.Task attribute), 113
                                                                                                                                                                        attribute), 106
do_data_xfer_mech (nidaqmx._task_modules.channels.do_dhantrigt_D@dgha(midaqmx.system.device.Device attribute),
                        attribute), 154
do_data_xfer_req_cond (nidaqmx._task_modules.channels.dto_dhisstatel(DiOtGhannetask_modules.channels.do_channel.DOChannel
                        attribute), 154
                                                                                                                                                                        attribute), 155
do generate on (nidagmx. task modules.channels.do chandel.DDOdfannelq count (nidagmx. task modules.channels.do channel.DO
                        attribute), 154
                                                                                                                                                                        attribute), 155
do_invert_lines (nidaqmx._task_modules.channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_channels.do_cha
                        attribute), 154
                                                                                                                                                                        attribute), 155
do_line_states_done_state
                                                                                                                                                do_use_only_on_brd_mem
                        (nidaqmx._task_modules.channels.do_channel.DOChannel (nidaqmx._task_modules.channels.do_channel.DOChannel
                        attribute), 155
                                                                                                                                                                        attribute), 155
do_line_states_paused_state
                                                                                                                                                DOChannel (class in nidagmx._task_modules.channels.do_channel),
                        (nidaqmx._task_modules.channels.do_channel.DOChannel 154
                        attribute), 155
                                                                                                                                                DOChannelCollection
                                                                                                                                                                                                                                       (class
                                                                                                                                                                                                                                                                                     in
do line states start state
                                                                                                                                                                        nidagmx, task modules.do channel collection),
                        (nidaqmx._task_modules.channels.do_channel.DOChannel 216
                                                                                                                                                DOExpirationState (class in nidagmx.types), 248
                        attribute), 155
do_lines (nidaqmx.system.device.Device attribute), 103
                                                                                                                                                DOLinesCollection
                                                                                                                                                                                                                                   (class
                                                                                                                                                                                                                                                                                      in
do_logic_family (nidaqmx._task_modules.channels.do_channel.DOClmidaemx.system._collections.physical_channel_collection),
                        attribute), 155
do max rate (nidaqmx.system.device.Device attribute), DONT ALLOW REGENERATION
                                                                                                                                                                        (nidaqmx.constants.RegenerationMode
                                                                                                                                                                                                                                                                                    at-
do mem map enable (nidagmx. task modules.channels.do channel.Didamhel
                        attribute), 155
                                                                                                                                                DOPortsCollection
                                                                                                                                                                                                                                   (class
                                                                                                                                                                                                                                                                                     in
DO_NOT_INVERT_POLARITY
                                                                                                                                                                        nidagmx.system._collections.physical_channel_collection),
                        (nidagmx.constants.SignalModifiers attribute),
                                                                                                                                                DOPowerUpState (class in nidagmx.types), 249
                                                                                                                                                DOResistorPowerUpState (class in nidagmx.types), 249
DO_NOT_OVERWRITE_UNREAD_SAMPLES
                        (nidaqmx.constants.OverwriteMode attribute),
                                                                                                                                                driver version
                                                                                                                                                                                            (nidaqmx.system.system.System
                                                                                                                                                                        tribute), 93
DO_NOT_WRITE (nidaqmx.constants.WriteBasicTEDSOpDSAs (nidaqmx.constants.ProductCategory attribute), 34
                                                                                                                                                DUTY CYCLE
                        attribute), 51
                                                                                                                                                                                                         (nidaqmx.constants.UsageTypeCI
do_num_booleans_per_chan
                                                                                                                                                                        attribute), 48
                        (nidagmx. task modules.out stream.OutStream duty cycle (nidagmx.types.CtrFreq attribute), 248
```

DYNAMIC_AVERAGING	expir_trig_dig_edge_src (nidaqmx.system.watchdog.WatchdogTask
(nidaqmx.constants.CounterFrequencyMethod	attribute), 110
attribute), 23	expir_trig_trig_on_network_conn_loss
_	(nidaqmx.system.watchdog.WatchdogTask
E	attribute), 110
E (nidaqmx.constants.ThermocoupleType attribute), 43	expir_trig_trig_type (nidaqmx.system.watchdog.WatchdogTask
E_SERIES_DAQ (nidaqmx.constants.ProductCategory	attribute), 110
attribute), 34	expiration_state (nidaqmx.types.AOExpirationState at-
EddyCurrentProxProbeSensitivityUnits (class in	tribute), 247
nidaqmx.constants), 25	expiration_state (nidaqmx.types.COExpirationState at-
Edge (class in nidaqmx.constants), 26	tribute), 248
EIGHT_M_HZ_TIMEBASE	expiration_state (nidaqmx.types.DOExpirationState at-
(nidaqmx.constants.MIOAIConvertTimebaseSou	
attribute), 32	expiration_states (nidaqmx.system.watchdog.WatchdogTask
EIGHTY_M_HZ_TIMEBASE	attribute), 110
(nidaqmx.constants.MIOAIConvertTimebaseSou	rExpirationState (class in
attribute), 32	nidaqmx.systemwatchdog_modules.expiration_state),
ELLIPTICAL (nidaqmx.constants.FilterResponse at-	111
tribute), 28	ExpirationStatesCollection (class in
EncoderType (class in nidaqmx.constants), 26	nidaqmx.systemwatchdog_modules.expiration_states_collection
EncoderZIndexPhase (class in nidaqmx.constants), 26	111
ENG (nidaqmx.constants.Language attribute), 30	expired (nidaqmx.system.watchdog.WatchdogTask
$ENTERING\_WINDOW\ (nidaqmx.constants.WindowTrigg$	erCondition tribute), 111
attribute), 51	export_signal() (nidaqmxtask_modules.export_signals.ExportSignals
error_code (nidaqmx.errors.DaqError attribute), 51	method), 218
error_code (nidaqmx.errors.DaqWarning attribute), 52	export_signals (nidaqmx.task.Task attribute), 113
error_type (nidaqmx.errors.DaqError attribute), 52	ExportAction (class in nidaqmx.constants), 27
error_type (nidaqmx.errors.DaqWarning attribute), 52	exported_10_m_hz_ref_clk_output_term
EVERY_SAMPLE (nidaqmx.constants.AutoZeroType	(nidaqmxtask_modules.export_signals.ExportSignals
attribute), 20	attribute), 218
EveryNSamplesEventType (class in nidaqmx.constants),	exported_20_m_hz_timebase_output_term
26	(nidaqmx_task_modules.export_signals.ExportSignals
excit_fault_chans (nidaqmxtask_modules.in_stream.InStr	
attribute), 221	ExportSignals (class in
excit_fault_chans_exist (nidaqmxtask_modules.in_stream attribute), 221	216
ExcitationDCorAC (class in nidaqmx.constants), 26	EXTERNAL (nidaqmx.constants.CurrentShuntResistorLocation
ExcitationIdleOutputBehavior (class in	attribute), 24
nidaqmx.constants), 27	EXTERNAL (nidaqmx.constants.ExcitationSource at-
ExcitationSource (class in nidaqmx.constants), 27	tribute), 27
ExcitationVoltageOrCurrent (class in nidaqmx.constants),	EXTERNAL (nidaqmx.constants.SourceSelection at-
27	tribute), 40
expir_states_ao_state (nidaqmx.systemwatchdog_module attribute), 111	esexpinalion/enveltesquehannstate (nidaqmxtask_modules.out_stream.OutStream
expir_states_ao_type (nidaqmx.systemwatchdog_module attribute), 111	s.expiration the Expiration State external_overvoltage_chans_exist
evnir states so state (nideamy system, watchdog module	es.expiration idate Pxpiration madules.out_stream.OutStream
attribute), 111	attribute), 226
expir states do state (nidagmx.system. watchdog module	EXTERNAL state Experimental Count Direction
attribute), 111	attribute), 23
expir_trig_dig_edge_edge	
(nidaqmx.system.watchdog.WatchdogTask	F
attribute), 110	FALLING (nidaqmx.constants.Edge attribute), 26
	FALLING (nidaqmx.constants.Slope attribute), 40

FIFTY_OHMS (nidaqmx.constants.Impedance1 attribute), 29	18 FROM_CUSTOM_SCALE
FillMode (class in nidaqmx.constants), 27	(nidaqmx.constants.AngleUnits attribute),
FilterResponse (class in nidaqmx.constants), 27	19
FilterType (class in nidaqmx.constants), 28	FROM_CUSTOM_SCALE
FINITE (nidaqmx.constants.AcquisitionType attribute),	(nidaqmx.constants.AngularVelocityUnits attribute), 19
FINITE (nidaqmx.constants.ScanRepeatMode attribute),	FROM_CUSTOM_SCALE
38	(nidaqmx.constants.BridgeUnits attribute),
FIRST_PRETRIGGER_SAMPLE	21
(nidaqmx.constants.ReadRelativeTo attribute),	FROM_CUSTOM_SCALE
36	(nidaqmx.constants.ChargeUnits attribute),
FIRST_SAMPLE (nidaqmx.constants.ReadRelativeTo	23
attribute), 36	FROM_CUSTOM_SCALE
FIRST_SAMPLE (nidaqmx.constants.WriteRelativeTo	(nidaqmx.constants.CurrentUnits attribute), 24
attribute), 51	FROM_CUSTOM_SCALE
FIVE_V (nidaqmx.constants.LogicFamily attribute), 32	(nidaqmx.constants.ForceUnits attribute),
FIVE_WIRE (nidaqmx.constants.ACExcitWireMode at-	28
tribute), 17	FROM_CUSTOM_SCALE
FIXED_50_PERCENT_DUTY_CYCLE	(nidaqmx.constants.FrequencyUnits attribute),
(nidaqmx.constants.ConstrainedGenMode	28
attribute), 23	FROM_CUSTOM_SCALE
FIXED_HIGH_FREQ (nidaqmx.constants.ConstrainedGen	
attribute), 23	30
FIXED_LOW_FREQ (nidaqmx.constants.ConstrainedGen)	MRRIOM_CUSTOM_SCALE
attribute), 23	(nidaqmx.constants.PressureUnits attribute), 34
flatten_channel_string() (in module nidaqmx.utils), 249	FROM_CUSTOM_SCALE
FM (nidaqmx.constants.ModulationType attribute), 32	(nidaqmx.constants.ResistanceUnits attribute),
FOOT_POUNDS (nidaqmx.constants.BridgePhysicalUnits	36
attribute), 20	FROM_CUSTOM_SCALE
FOOT_POUNDS (nidaqmx.constants.TorqueUnits	(nidaqmx.constants.SoundPressureUnits
attribute), 44	attribute), 40
FOOT_POUNDS (nidaqmx.constants.UnitsPreScaled at-	FROM_CUSTOM_SCALE
tribute), 45	(nidaqmx.constants.StrainUnits attribute),
FORCE_BRIDGE (nidaqmx.constants.UsageTypeAI at-	41
tribute), 47	FROM_CUSTOM_SCALE
FORCE_IEPE_SENSOR	(nidaqmx.constants.TEDSUnits attribute),
(nidaqmx.constants.UsageTypeAI attribute), 47	42
ForceIEPESensorSensitivityUnits (class in	FROM_CUSTOM_SCALE
nidaqmx.constants), 28	(nidaqmx.constants.TemperatureUnits at-
ForceUnits (class in nidaqmx.constants), 28	tribute), 43
FOUR_WIRE (nidaqmx.constants.ACExcitWireMode at-	FROM_CUSTOM_SCALE
tribute), 17	(nidaqmx.constants.TimeUnits attribute),
$FOUR\_WIRE\ (nid aqmx.constants. Resistance Configuration And Configuration Configura$	n 43
attribute), 36	FROM_CUSTOM_SCALE
FRA (nidaqmx.constants.Language attribute), 30	(nidaqmx.constants.TorqueUnits attribute),
freq (nidaqmx.types.CtrFreq attribute), 248	44
FREQUENCY (nidaqmx.constants.UsageTypeCI at-	FROM_CUSTOM_SCALE
tribute), 48	(nidaqmx.constants.VelocityUnits attribute), 49
FREQUENCY_VOLTAGE	FROM_CUSTOM_SCALE
(nidaqmx.constants.UsageTypeAI attribute), 47	(nidaqmx.constants.VoltageUnits attribute),
FrequencyUnits (class in nidaqmx.constants), 28	50
FROM_CUSTOM_SCALE	FROM_TEDS (nidaqmx.constants.BridgeUnits at-
(nidagmy constants AccelUnits attribute)	tribute) 21

FROM_TEDS (nidaqmx.constants.CurrentUnits attribute), 24	GROUP_BY_SCAN_NUMBER (nidaqmx.constants.FillMode attribute), 27
FROM_TEDS (nidaqmx.constants.ResistanceUnits attribute), 37	H
$FROM\_TEDS\ (nidaqmx.constants. TEDS Units\ attribute),$	HALF_BRIDGE (nidaqmx.constants.BridgeConfiguration
FROM_TEDS (nidaqmx.constants.UnitsPreScaled	attribute), 20 HALF_BRIDGE_I (nidaqmx.constants.StrainGageBridgeType
attribute), 45 FROM_TEDS (nidaqmx.constants.VoltageUnits at-	attribute), 40 HALF_BRIDGE_II (nidaqmx.constants.StrainGageBridgeType
tribute), 50 FULL_BRIDGE (nidaqmx.constants.BridgeConfiguration	attribute), 40 HALT_OUTPUT_AND_ERROR
attribute), 20 FULL_BRIDGE_I (nidaqmx.constants.StrainGageBridgeT	(nidaqmx.constants.UnderflowBehavior at-
attribute), 40	HANDSHAKE (nidaqmx.constants.SampleTimingType
FULL_BRIDGE_II (nidaqmx.constants.StrainGageBridge attribute), 40	HANDSHAKE (nidaqmx.constants.TriggerUsage at-
FULL_BRIDGE_III (nidaqmx.constants.StrainGageBridge attribute), 40	eType tribute), 44 handshake_trigger (nidaqmxtask_modules.triggers.Triggers
FuncGenType (class in nidaqmx.constants), 29	attribute), 236
FUNCTION_GENERATION (nidaqmx.constants.UsageTypeAO attribute),	HANDSHAKE_TRIGGER_ASSERTS (nidaqmx.constants.SampleInputDataWhen
48	attribute), 37
G	HANDSHAKE_TRIGGER_DEASSERTS (nidaqmx.constants.SampleInputDataWhen
G (nidaqmx.constants.AccelUnits attribute), 18	attribute), 37
G (nidaqmx.constants.UnitsPreScaled attribute), 45 get_analog_power_up_states()	HandshakeStartCondition (class in nidaqmx.constants),
(nidaqmx.system.system.System method),	HandshakeTrigger (class in nidaqmxtask_modules.triggering.handshake_trigger),
get_analog_power_up_states_with_output_type()	237
(nidaqmx.system.system.System method),	HARDWARE_DEFINED
93	
get_digital_logic_family_power_up_state()	(nidaqmx.constants.FilterResponse attribute), 28
	(nidaqmx.constants.FilterResponse attribute), 28 HERTZ (nidaqmx.constants.UnitsPreScaled attribute), 45
get_digital_logic_family_power_up_state()	(nidaqmx.constants.FilterResponse attribute), 28 HERTZ (nidaqmx.constants.UnitsPreScaled attribute), 45 HIGH (nidaqmx.constants.Level attribute), 31 HIGH (nidaqmx.constants.PowerUpStates attribute), 33
get_digital_logic_family_power_up_state()	(nidaqmx.constants.FilterResponse attribute), 28  HERTZ (nidaqmx.constants.UnitsPreScaled attribute), 45  HIGH (nidaqmx.constants.Level attribute), 31  HIGH (nidaqmx.constants.PowerUpStates attribute), 33  HIGH (nidaqmx.constants.WatchdogCOExpirState attribute), 50
get_digital_logic_family_power_up_state()	(nidaqmx.constants.FilterResponse attribute), 28  HERTZ (nidaqmx.constants.UnitsPreScaled attribute), 45  HIGH (nidaqmx.constants.Level attribute), 31  HIGH (nidaqmx.constants.PowerUpStates attribute), 33  HIGH (nidaqmx.constants.WatchdogCOExpirState attribute), 50  HIGH_FREQUENCY_2_COUNTERS
get_digital_logic_family_power_up_state()	(nidaqmx.constants.FilterResponse attribute), 28  HERTZ (nidaqmx.constants.UnitsPreScaled attribute), 45  HIGH (nidaqmx.constants.Level attribute), 31  HIGH (nidaqmx.constants.PowerUpStates attribute), 33  HIGH (nidaqmx.constants.WatchdogCOExpirState attribute), 50  HIGH_FREQUENCY_2_COUNTERS (nidaqmx.constants.CounterFrequencyMethod attribute), 23
get_digital_logic_family_power_up_state()	(nidaqmx.constants.FilterResponse attribute), 28  HERTZ (nidaqmx.constants.UnitsPreScaled attribute), 45 HIGH (nidaqmx.constants.Level attribute), 31 HIGH (nidaqmx.constants.PowerUpStates attribute), 33 HIGH (nidaqmx.constants.WatchdogCOExpirState attribute), 50 HIGH_FREQUENCY_2_COUNTERS (nidaqmx.constants.CounterFrequencyMethod attribute), 23  SENGCHARMEDONCOUNTERS (sidaqmx.constants.CounterFrequencyMethod attribute), 23
get_digital_logic_family_power_up_state()	(nidaqmx.constants.FilterResponse attribute), 28  HERTZ (nidaqmx.constants.UnitsPreScaled attribute), 45  HIGH (nidaqmx.constants.Level attribute), 31  HIGH (nidaqmx.constants.PowerUpStates attribute), 33  HIGH (nidaqmx.constants.WatchdogCOExpirState attribute), 50  HIGH_FREQUENCY_2_COUNTERS  (nidaqmx.constants.CounterFrequencyMethod attribute), 23  SINCHAINTERDACCON Recipied Common Scale Relieve Output Behavior attribute), 18  HIGH_IMPEDANCE (nidaqmx.constants.AOPowerUpOutputBehavior
get_digital_logic_family_power_up_state()	(nidaqmx.constants.FilterResponse attribute), 28  HERTZ (nidaqmx.constants.UnitsPreScaled attribute), 45  HIGH (nidaqmx.constants.Level attribute), 31  HIGH (nidaqmx.constants.PowerUpStates attribute), 33  HIGH (nidaqmx.constants.WatchdogCOExpirState attribute), 50  HIGH_FREQUENCY_2_COUNTERS (nidaqmx.constants.CounterFrequencyMethod attribute), 23  SINCH_AIMPEDANCE(PRESISTED COUNTERS COUNTERS (NIDE OF COUNTERS) (NIDE OF COUNTERS)  HIGH_IMPEDANCE (nidaqmx.constants.AOPowerUpOutputBehavior attribute), 18  HIGH_RESOLUTION (nidaqmx.constants.AOCTimingMode
get_digital_logic_family_power_up_state()	(nidaqmx.constants.FilterResponse attribute), 28  HERTZ (nidaqmx.constants.UnitsPreScaled attribute), 45  HIGH (nidaqmx.constants.Level attribute), 31  HIGH (nidaqmx.constants.PowerUpStates attribute), 33  HIGH (nidaqmx.constants.WatchdogCOExpirState attribute), 50  HIGH_FREQUENCY_2_COUNTERS (nidaqmx.constants.CounterFrequencyMethod attribute), 23  SINCH_ANNIPEDANCE(ParaisticsCounterSequencyMethod attribute), 18  HIGH_IMPEDANCE (nidaqmx.constants.AOPowerUpOutputBehavior attribute), 18  HIGH_RESOLUTION (nidaqmx.constants.ADCTimingMode vior attribute), 17
get_digital_logic_family_power_up_state()	(nidaqmx.constants.FilterResponse attribute), 28  HERTZ (nidaqmx.constants.UnitsPreScaled attribute), 45  HIGH (nidaqmx.constants.Level attribute), 31  HIGH (nidaqmx.constants.PowerUpStates attribute), 33  HIGH (nidaqmx.constants.WatchdogCOExpirState attribute), 50  HIGH_FREQUENCY_2_COUNTERS (nidaqmx.constants.CounterFrequencyMethod attribute), 23  SINCH_AIMPEDANCE(PRESISTED COUNTERS COUNTERS (NIDE OF COUNTERS) (NIDE OF COUNTERS)  HIGH_IMPEDANCE (nidaqmx.constants.AOPowerUpOutputBehavior attribute), 18  HIGH_RESOLUTION (nidaqmx.constants.AOCTimingMode
get_digital_logic_family_power_up_state()	(nidaqmx.constants.FilterResponse attribute), 28  HERTZ (nidaqmx.constants.UnitsPreScaled attribute), 45  HIGH (nidaqmx.constants.Level attribute), 31  HIGH (nidaqmx.constants.PowerUpStates attribute), 33  HIGH (nidaqmx.constants.WatchdogCOExpirState attribute), 50  HIGH_FREQUENCY_2_COUNTERS
get_digital_logic_family_power_up_state()	(nidaqmx.constants.FilterResponse attribute), 28  HERTZ (nidaqmx.constants.UnitsPreScaled attribute), 45 HIGH (nidaqmx.constants.Level attribute), 31 HIGH (nidaqmx.constants.PowerUpStates attribute), 33 HIGH (nidaqmx.constants.WatchdogCOExpirState attribute), 50 HIGH_FREQUENCY_2_COUNTERS
get_digital_logic_family_power_up_state()	(nidaqmx.constants.FilterResponse attribute), 28  HERTZ (nidaqmx.constants.UnitsPreScaled attribute), 45  HIGH (nidaqmx.constants.Level attribute), 31  HIGH (nidaqmx.constants.PowerUpStates attribute), 33  HIGH (nidaqmx.constants.WatchdogCOExpirState attribute), 50  HIGH_FREQUENCY_2_COUNTERS

```
hshk_event_delay (nidaqmx._task_modules.export_signals.ENGHSPONINDS (nidaqmx.constants.BridgePhysicalUnits
                    attribute), 218
                                                                                                                                          attribute), 21
hshk event interlocked assert on start
                                                                                                                      INCH POUNDS
                                                                                                                                                                       (nidagmx.constants.TorqueUnits
                    (nidaqmx._task_modules.export_signals.ExportSignals
                                                                                                                                          attribute), 44
                    attribute), 218
                                                                                                                      INCH POUNDS (nidagmx.constants.UnitsPreScaled at-
hshk event interlocked asserted lvl
                                                                                                                                          tribute), 46
                   (nidagmx. task modules.export signals.ExportSiPNOHES (nidagmx.constants.LengthUnits attribute), 30
                    attribute), 218
                                                                                                                      INCHES (nidagmx.constants.UnitsPreScaled attribute),
hshk_event_interlocked_deassert_delay
                   (nidaqmx._task_modules.export_signals.ExportSignatHES_PER_SECOND
                    attribute), 218
                                                                                                                                          (nidagmx.constants.UnitsPreScaled attribute),
hshk_event_output_behavior
                    (nidaqmx._task_modules.export_signals.ExportSignatHES_PER_SECOND
                   attribute), 218
                                                                                                                                          (nidaqmx.constants.VelocityUnits attribute), 49
hshk_event_output_term (nidaqmx._task_modules.export_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_signicaletexport_sign
                    attribute), 218
                                                                                                                                          (nidaqmx.constants.AccelUnits
                                                                                                                                                                                                                     attribute),
hshk_event_pulse_polarity
                   (nidagmx, task modules.export signals.ExportSieNOHES PER SECOND SOUARED
                                                                                                                                          (nidagmx.constants.UnitsPreScaled attribute),
                   attribute), 218
hshk event pulse width (nidagmx. task modules.export signals.ExportSignals
                    attribute), 218
                                                                                                                      index() (nidaqmx._task_modules.ai_channel_collection.AIChannelCollection
hshk_sample_input_data_when
                                                                                                                                          method), 201
                    (nidaqmx._task_modules.timing.Timing
                                                                                                                      index() (nidaqmx._task_modules.ao_channel_collection.AOChannelCollect
                    attribute), 233
                                                                                                                                          method), 202
hshk_start_cond (nidaqmx._task_modules.timing.Timing
                                                                                                                      index()(nidagmx. task modules.ci channel collection.CIChannelCollection
                    attribute), 233
                                                                                                                                          method), 213
HW_TIMED_SINGLE_POINT
                                                                                                                      index() (nidagmx._task_modules.co_channel_collection.COChannelCollect
                    (nidagmx.constants.AcquisitionType attribute),
                                                                                                                                          method), 215
                                                                                                                      index() (nidagmx._task_modules.di_channel_collection.DIChannelCollection
HZ (nidaqmx.constants.FrequencyUnits attribute), 28
                                                                                                                                          method), 216
                                                                                                                      index() (nidagmx._task_modules.do_channel_collection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DOChannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelCollection.DochannelColl
                                                                                                                                          method), 216
IL\ (nidaqmx.constants. Eddy Current Prox Probe Sensitivity Un \ref{thm:prop:sensitivity} II)\ (nidaqmx.constants. Task String Format\ attribute),\ 42
                                                                                                                      input\_buf\_size (nidaqmx.\_task\_modules.in\_stream.InStream
                    attribute), 25
ILLIMETER (nidaqmx.constants.EddyCurrentProxProbeSensitivityUntilibute), 221
                                                                                                                      input_onbrd_buf_size (nidaqmx._task_modules.in_stream.InStream
                    attribute), 25
                                                                                                                                          attribute), 221
IMMEDIATE (nidaqmx.constants.HandshakeStartCondition
                                                                                                                      input_port
                                                                                                                                                          (nidaqmx.types.CDAQSyncConnection
                    attribute), 29
                                                                                                                                          attribute), 248
Impedance1 (class in nidagmx.constants), 29
                                                                                                                      InputCalSource (class in nidaqmx.constants), 30
IMPLICIT (nidaqmx.constants.SampleTimingType at-
                                                                                                                      InputDataTransferCondition
                                                                                                                                                                                                                                    in
                    tribute), 38
                                                                                                                                          nidagmx.constants), 30
implicit_underflow_behavior
                                                                                                                      INSIDE WINDOW (nidagmx.constants.WindowTriggerCondition2
                   (nidaqmx._task_modules.timing.Timing
                                                                                                                                          attribute), 51
                    attribute), 233
                                                                                                                      InStream (class in nidagmx._task_modules.in_stream),
in_stream (nidagmx.task.Task attribute), 113
INACTIVE (nidagmx.constants.ActiveOrInactiveEdgeSelection
                                                                                                                      INTERLOCKED (nidagmx.constants.ExportAction at-
                    attribute), 19
                                                                                                                                          tribute), 27
INCH_OUNCES (nidaqmx.constants.BridgePhysicalUnits
                                                                                                                      INTERLOCKED
                                                                                                                                                                        (nidaqmx.constants.TriggerType
                   attribute), 21
                                                                                                                                          attribute), 44
INCH OUNCES
                                                 (nidagmx.constants.TorqueUnits
                                                                                                                      interlocked_asserted_lvl (nidaqmx._task_modules.triggering.handshake_tri
                    attribute), 44
                                                                                                                                          attribute), 237
INCH_OUNCES (nidagmx.constants.UnitsPreScaled at-
                                                                                                                      interlocked src (nidagmx. task modules.triggering.handshake trigger.Han
                    tribute), 46
```

attribute), 237

$INTERNAL\ (nidaqmx.constants. Current Shunt Resistor Local Constants) and the property of th$	ntidest_DRIVER_CHOOSE
attribute), 24	(nid aqmx.constants. Current Shunt Resistor Location
INTERNAL (nidaqmx.constants.ExcitationSource	attribute), 24
attribute), 27	Level (class in nidaqmx.constants), 31
	LEVEL (nidaqmx.constants.ExportAction attribute), 27
tribute), 40	lin_slope (nidaqmx.scale.Scale attribute), 54
INTERRUPT (nidaqmx.constants.DataTransferActiveTrans	
attribute), 24	LINEAR (nidaqmx.constants.ScaleType attribute), 38
INVERT_POLARITY (nidaqmx.constants.SignalModifiers	
attribute), 40	load() (nidaqmx.system.storage.persisted_scale.PersistedScale method), 108
IRIGB (nidaqmx.constants.GpsSignalType attribute), 29	Clhad()e(nidaqmx.system.storage.persisted_task.PersistedTask
attribute), 130	method), 108
	(nidaqmx.constants.SwitchChannelUsage
attribute), 134	attribute), 41
is_global (nidaqmxtask_modules.channels.channel.Chann	
attribute), 118	local() (nidaqmx.system.system static method),
is_global (nidaqmxtask_modules.channels.ci_channel.CIO	
attribute), 148	LOG (nidaqmx.constants.LoggingMode attribute), 31
is_global (nidaqmxtask_modules.channels.co_channel.Co	
attribute), 151	attribute), 31
$is\_global\ (nidaqmx.\_task\_modules.channels.di\_channel.DI$	Characteristic Charac
attribute), 153	attribute), 221
$is\_global  (nidaqmx.\_task\_modules.channels.do\_channel.Delta and additional content of the con$	•• • .
attribute), 156	(nidaqmxtask_modules.in_stream.InStream
is_task_done() (nidaqmx.task.Task method), 113	attribute), 221
J	logging_file_write_size (nidaqmxtask_modules.in_stream.InStream attribute), 221
J (nidaqmx.constants.ThermocoupleType attribute), 43	$logging\_mode \ (nidaqmx.\_task\_modules.in\_stream. In Stream$
JPN (nidaqmx.constants.Language attribute), 30	attribute), 221
JSON (nidaqmx.constants.TaskStringFormat attribute),	logging_pause (nidaqmxtask_modules.in_stream.InStream
42	attribute), 222
K	logging_samps_per_file (nidaqmxtask_modules.in_stream.InStream attribute), 222
K (nidaqmx.constants.TemperatureUnits attribute), 43	logging_tdms_group_name
K (nidaqmx.constants.ThermocoupleType attribute), 43	(nidaqmxtask_modules.in_stream.InStream
K (nidaqmx.constants.UnitsPreScaled attribute), 46	attribute), 222
KILOGRAM_FORCE (nidaqmx.constants.BridgePhysical attribute), 21	Jogging_tdms_operation (nidaqmxtask_modules.in_stream.InStream attribute), 222
KILOGRAM_FORCE (nidaqmx.constants.ForceUnits	LoggingMode (class in nidaqmx.constants), 31
attribute), 28	LoggingOperation (class in nidaqmx.constants), 31
$KILOGRAM\_FORCE \ (nid aqmx.constants. Units Pre Scaled$	LogicFamily (class in nidaqmx.constants), 31
attribute), 46	LogicLvlBehavior (class in nidaqmx.constants), 32
KOR (nidaqmx.constants.Language attribute), 30	LOOPBACK_0 (nidaqmx.constants.InputCalSource at-
I	tribute), 30 LOOPBACK 180 (nidaqmx.constants.InputCalSource
L	LOOPBACK_180 (nidaqmx.constants.InputCalSource attribute), 30
Language (class in nidaqmx.constants), 30	LOSSLESS_PACKING (nidaqmx.constants.RawDataCompressionTyp
LARGE_RANGE_2_COUNTERS	attribute), 35
(nidaqmx.constants.CounterFrequencyMethod	LOSSY_LSB_REMOVAL
attribute), 24 LEAVING_WINDOW (nidaqmx.constants.WindowTrigger	Condition (nidagmx.constants.RawDataCompressionType
attribute), 51	attribute), 35
LEFT (nidaqmx.constants.DataJustification attribute), 24	LOW (nidaqmx.constants.Level attribute), 31
LengthUnits (class in nidaqmx.constants), 30	LOW (nidaqmx.constants.PowerUpStates attribute), 33

LOW (nidaqmx.constants.WatchdogCOExpirState attribute), 50	MAP_RANGES (nidaqmx.constants.ScaleType attribute), 38
LOW_FREQUENCY_1_COUNTER (nidaqmx.constants.CounterFrequencyMethod	map_scaled_max (nidaqmx.scale.Scale attribute), 55 map_scaled_min (nidaqmx.scale.Scale attribute), 55 MASSTEP (i.i., 1977), 42
attribute), 24	MASTER (nidaqmx.constants.SyncType attribute), 42
low_tick (nidaqmx.types.CtrTick attribute), 248	MASTER_TIMEBASE (nidaqmx.constants.MIOAIConvertTimebaseSourc
low_time (nidaqmx.types.CtrTime attribute), 248	attribute), 32
LOWPASS (nidaqmx.constants.FilterType attribute), 28	master_timebase_rate (nidaqmxtask_modules.timing.Timing
LVDTSensitivityUnits (class in nidaqmx.constants), 30	attribute), 234
M	master_timebase_src (nidaqmxtask_modules.timing.Timing attribute), 234
M_SERIES_DAQ (nidaqmx.constants.ProductCategory	MAX_SHEAR_STRAIN
attribute), 34	(nidaqmx.constants.StrainGageRosetteMeasurementType
M_VOLTS_PER_G (nidaqmx.constants.AccelSensitivityU	nits attribute), 41
attribute), 18	MAX_SHEAR_STRAIN_ANGLE
M_VOLTS_PER_INCH_PER_SECOND	(nidaqmx.constants.StrainGageRosetteMeasurementType
(nidaqmx.constants.VelocityIEPESensorSensitivi	tyUnits attribute), 41
attribute), 49	METERS (nidagmy, constants Length Units attribute), 30
M_VOLTS_PER_MILLIMETER_PER_SECOND	METERS (nidaqmx.constants.UnitsPreScaled attribute),
(nidaqmx.constants.VelocityIEPESensorSensitivi	
attribute), 49	METERS_PER_SECOND
M_VOLTS_PER_NEWTON	(nidaqmx.constants.UnitsPreScaled attribute),
(nidaqmx.constants.ForceIEPESensorSensitivity)	Units 46
attribute), 28	METERS_PER_SECOND (nidaqmx.constants.VelocityUnits attribute), 49
M_VOLTS_PER_POUND	
(nidaqmx.constants.ForceIEPESensorSensitivity)	(nidaqmx.constants.AccelUnits attribute),
attribute), 28	4.0
M_VOLTS_PER_VOLT (nidaqmx.constants.BridgeElectric	METERS_PER_SECOND_SQUARED
attribute), 20 M_VOLTS_PER_VOLT (nidaqmx.constants.BridgeUnits	(nidaqmx.constants.UnitsPreScaled attribute),
attribute), 21	46
M VOLTS PER VOLT (nidagmy constants Units Pre Scale	adMICRON (nidaqmx.constants.EddyCurrentProxProbeSensitivityUnits
attribute), 46	attribute), 26
M_VOLTS_PER_VOLT_PER_MILLI_INCH	MIL (nidaqmx.constants.EddyCurrentProxProbeSensitivityUnits
(nidaqmx.constants.LVDTSensitivityUnits	attribute), 26
attribute), 30	MILLIMETER (nidaqmx.constants.EddyCurrentProxProbeSensitivityUnits
M_VOLTS_PER_VOLT_PER_MILLIMETER	attribute), 26
(nidagmx.constants.LVDTSensitivityUnits	MIOAIConvertTimebaseSource (class in
attribute), 30	nidaqmx.constants), 32
M_VPER_VPER_DEGREE	ModulationType (class in nidaqmx.constants), 32
(nidaqmx.constants.RVDTSensitivityUnits	MOST_RECENT_SAMPLE
attribute), 35	(nidaqmx.constants.ReadRelativeTo attribute),
M_VPER_VPER_RADIAN	36
(nidaqmx.constants.RVDTSensitivityUnits attribute), 35	N
MAINTAIN_EXISTING_VALUE	N (nidaqmx.constants.ThermocoupleType attribute), 43
(nidaqmx.constants.AOIdleOutputBehavior attribute), 18	name (nidaqmxtask_modules.channels.ai_channel.AIChannel attribute), 130
MAINTAIN_EXISTING_VALUE	name (nidaqmxtask_modules.channels.ao_channel.AOChannel
(nidaqmx.constants.ExcitationIdleOutputBehavio	
attribute), 27	name (nidaqmxtask_modules.channels.channel.Channel
map_pre_scaled_max (nidaqmx.scale.Scale attribute), 55	attribute), 118
map_pre_scaled_min (nidaqmx.scale.Scale attribute), 55	name (nidaqmxtask_modules.channels.ci_channel.CIChannel attribute), 148

```
name (nidagmx. task modules.channels.co channel.COChannel
                                                                ule), 213
         attribute), 151
                                                       nidagmx. task modules.di channel collection (module),
name (nidagmx. task modules.channels.di channel.DIChannel
         attribute), 153
                                                       nidagmx. task modules.do channel collection
                                                                                                      (mod-
name (nidagmx. task modules.channels.do channel.DOChannel
                                                                ule), 216
         attribute), 156
                                                       nidagmx. task modules.export signals (module), 216
name (nidagmx.scale.Scale attribute), 55
                                                       nidagmx, task modules.in stream (module), 220
name (nidagmx.system.device.Device attribute), 103
                                                       nidagmx. task modules.out stream (module), 226
name (nidagmx.system.physical channel.PhysicalChannel
                                                       nidagmx. task modules.timing (module), 229
         attribute), 106
                                                       nidaqmx._task_modules.triggering.arm_start_trigger
name
       (nidaqmx.system.watchdog.WatchdogTask
                                                  at-
                                                                (module), 236
         tribute), 111
                                                       nidaqmx._task_modules.triggering.handshake_trigger
name (nidagmx.task.Task attribute), 113
                                                                (module), 237
NEG_10_TO_10_V (nidagmx.constants.SCXI1124Range
                                                       nidaqmx._task_modules.triggering.pause_trigger (mod-
         attribute), 37
                                                                ule), 237
NEG_1_TO_1_V (nidaqmx.constants.SCXI1124Range
                                                       nidaqmx._task_modules.triggering.reference_trigger
         attribute), 37
                                                                (module), 240
NEG_5_TO_5_V (nidaqmx.constants.SCXI1124Range
                                                       nidagmx, task modules.triggering.start trigger
                                                                                                      (mod-
         attribute), 37
                                                                ule), 244
NETWORK DAQ (nidagmx.constants.ProductCategory
                                                       nidagmx. task modules.triggers (module), 236
         attribute), 34
                                                       nidagmx.constants (module), 17
NEWTON METERS (nidagmx.constants.BridgePhysicalUmitdagmx.errors (module), 51
                                                       nidaqmx.scale (module), 52
         attribute), 21
NEWTON METERS
                       (nidagmx.constants.TorqueUnits
                                                       nidagmx.stream readers (module), 56
                                                       nidagmx.stream writers (module), 77
         attribute), 44
NEWTON_METERS (nidaqmx.constants.UnitsPreScaled
                                                       nidagmx.system. collections.device collection
                                                                                                      (mod-
         attribute), 46
                                                                ule), 96
NEWTONS (nidaqmx.constants.BridgePhysicalUnits at-
                                                       nidagmx.system._collections.persisted_channel_collection
         tribute), 21
                                                                (module), 96
NEWTONS (nidagmx.constants.ForceUnits attribute), 28
                                                       nidagmx.system. collections.persisted scale collection
              (nidaqmx.constants.UnitsPreScaled
                                                                (module), 97
NEWTONS
         tribute), 46
                                                       nidaqmx.system._collections.persisted_task_collection
nidagmx._task_modules.ai_channel_collection (module),
                                                                (module), 97
                                                       nidagmx.system._collections.physical_channel_collection
nidagmx, task modules.ao channel collection
                                               (mod-
                                                                (module), 97
         ule), 201
                                                       nidagmx.system. watchdog modules.expiration state
nidagmx. task modules.channel collection
                                            (module),
                                                                (module), 111
         156
                                                       nidaqmx.system._watchdog_modules.expiration_states_collection
                                                                (module), 111
nidagmx. task modules.channels.ai channel
                                           (module),
         119
                                                       nidaqmx.system.device (module), 98
nidagmx. task modules.channels.ao channel (module),
                                                       nidagmx.system.physical channel (module), 104
                                                       nidagmx.system.storage.persisted channel (module), 107
nidagmx. task modules.channels.channel (module), 118
                                                       nidagmx.system.storage.persisted scale (module), 107
nidaqmx._task_modules.channels.ci_channel
                                           (module),
                                                       nidaqmx.system.storage.persisted_task (module), 108
                                                       nidaqmx.system.system (module), 91
nidagmx._task_modules.channels.co_channel (module),
                                                       nidaqmx.system.watchdog (module), 109
                                                       nidagmx.task (module), 112
nidagmx._task_modules.channels.di_channel
                                           (module),
                                                       nidagmx.types (module), 247
                                                       nidaqmx.utils (module), 249
nidagmx._task_modules.channels.do_channel (module),
                                                       NIELVIS (nidaqmx.constants.ProductCategory attribute),
                                                       NO_ACTION (nidaqmx.constants.BreakMode attribute),
nidagmx. task modules.ci channel collection (module),
nidagmx. task modules.co channel collection
                                               (mod- NO BRIDGE (nidagmx.constants.BridgeConfiguration
```

attribute), 20	(nidaqmx.constants.DeassertCondition at-
NO_CHANGE (nidaqmx.constants.Level attribute), 31 NO_CHANGE (nidaqmx.constants.WatchdogAOExpirStat	tribute), 25
attribute), 50	eon_board_memori_more_trans_ratr_roll (nidaqmx.constants.InputDataTransferCondition
NO_CHANGE (nidaqmx.constants.WatchdogCOExpirStat	
attribute), 50	ON_BOARD_MEMORY_NOT_EMPTY
NONE (nidaqmx.constants.AutoZeroType attribute), 20	(nidaqmx.constants.InputDataTransferCondition
NONE (nidaqmx.constants.ExcitationSource attribute),	attribute), 30
27	ON_DEMAND (nidaqmx.constants.SampleTimingType
NONE (nidaqmx.constants.GpsSignalType attribute), 29	attribute), 38
NONE (nidaqmx.constants.LogicLvlBehavior attribute),	ONBOARD_MEMORY_CUSTOM_THRESHOLD
32.	(nidaqmx.constants.DeassertCondition at-
NONE (nidaqmx.constants.ModulationType attribute), 32	tribute), 25
NONE (nidaqmx.constants.RawDataCompressionType	ONBOARD_MEMORY_CUSTOM_THRESHOLD
attribute), 36	(nidaqmx.constants.InputDataTransferCondition
NONE (nidaqmx.constants.ScaleType attribute), 38	attribute), 30
NONE (nidaqmx.constants.ShuntElementLocation	ONCE (nidaqmx.constants.AutoZeroType attribute), 20
attribute), 39	ONE_HUNDRED_M_HZ_TIMEBASE
NONE (nidaqmx.constants.SyncType attribute), 42	(nidaqmx.constants.MIOAIConvertTimebaseSource
NONE (nidaqmx.constants.TriggerType attribute), 44	attribute), 32
NOTCH (nidaqmx.constants.FilterType attribute), 28	ONE_M_OHM (nidaqmx.constants.Impedance1 at-
NRSE (nidaqmx.constants.TerminalConfiguration at-	tribute), 29
tribute), 43	OPEN (nidaqmx.constants.LoggingOperation attribute),
num_chans (nidaqmxtask_modules.in_stream.InStream	31
attribute), 222	OPEN (nidaqmx.constants.RelayPosition attribute), 36
num_chans (nidaqmxtask_modules.out_stream.OutStream attribute), 227	mopen_chans (nidaqmxtask_modules.in_stream.InStream attribute), 222
num_dma_chans (nidaqmx.system.device.Device at-	open_chans_details (nidaqmxtask_modules.in_stream.InStream
tribute), 103	attribute), 222
number_of_channels (nidaqmx.task.Task attribute), 113	open_chans_exist (nidaqmxtask_modules.in_stream.InStream
number_of_devices (nidaqmx.task.Task attribute), 113	attribute), 222
$\circ$	OPEN_COLLECTOR (nidaqmx.constants.DigitalDriveType
0	attribute), 25
OFF (nidaqmx.constants.LoggingMode attribute), 31	open_current_loop_chans
offset (nidaqmxtask_modules.in_stream.InStream at-	(nidaqmxtask_modules.in_stream.InStream
tribute), 222	attribute), 222
offset (nidaqmxtask_modules.out_stream.OutStream	open_current_loop_chans
attribute), 227	(nidaqmxtask_modules.out_stream.OutStream
OHMS (nidaqmx.constants.ResistanceUnits attribute), 37	attribute), 227
OHMS (nidaqmx.constants.UnitsPreScaled attribute), 46	open_current_loop_chans_exist (nidaqmxtask_modules.in_stream.InStream
ON_BOARD_MEMORY_EMPTY	
(nidaqmx.constants.OutputDataTransferConditio	open_current_loop_chans_exist
attribute), 32	(nidaqmxtask_modules.out_stream.OutStream
ON_BOARD_MEMORY_FULL	attribute), 227
(nidaqmx.constants.DeassertCondition at-	OPEN_OR_CREATE (nidaqmx.constants.LoggingOperation
tribute), 25 ON_BOARD_MEMORY_HALF_FULL_OR_LESS	attribute), 31
(nideamy constants Output Date Transfer Condition	nopen_thrmcpl_chans (nidaqmxtask_modules.in_stream.InStream
attribute), 32	attribute), 222
ON_BOARD_MEMORY_LESS_THAN_FULL	open_thrmcpl_chans_exist
(nidaqmx.constants.OutputDataTransferConditio	
attribute), 32	
utiliouto / , J =	attribute), 223
ON_BOARD_MEMORY_MORE_THAN_HALF_FULL	attribute), 223 out_stream (nidaqmx.task.Task attribute), 113

attribute), 227	PATH_ALREADY_EXISTS
output_onbrd_buf_size (nidaqmxtask_modules.out_stream attribute), 227	m.OutStrea(nidaqmx.constants.PathCapability attribute), 33
output_port (nidaqmx.types.CDAQSyncConnection attribute), 248	PATH_AVAILABLE (nidaqmx.constants.PathCapability attribute), 33
output_type (nidaqmx.types.AOExpirationState at-	PATH_UNSUPPORTED
tribute), 247 OutputDataTransferCondition (class in	(nidaqmx.constants.PathCapability attribute), 33
nidaqmx.constants), 32	PathCapability (class in nidaqmx.constants), 33
OUTSIDE_WINDOW (nidaqmx.constants.WindowTrigger attribute), 51	(nidaqmx.constants.DigitalPatternCondition
OutStream (class in nidaqmxtask_modules.out_stream), 226	attribute), 25 PATTERN_MATCHES (nidaqmx.constants.DigitalPatternCondition
over_write (nidaqmxtask_modules.in_stream.InStream	attribute), 25
attribute), 223	PAUSE (nidaqmx.constants.TriggerUsage attribute), 44
overcurrent_chans (nidaqmxtask_modules.in_stream.InSt attribute), 223	
overcurrent_chans (nidaqmxtask_modules.out_stream.Ou	
attribute), 227	pause_trig_output_term (nidaqmxtask_modules.export_signals.ExportSig
overcurrent_chans_exist (nidaqmxtask_modules.in_stream attribute), 223	
overcurrent_chans_exist (nidaqmxtask_modules.out_strea	
attribute), 227	PauseTrigger (class in
OverflowBehavior (class in nidaqmx.constants), 33	nidaqmxtask_modules.triggering.pause_trigger),
$overloaded\_chans \ (nidaqmx.\_task\_modules.in\_stream.InS$	
attribute), 223	PC_CARD (nidaqmx.constants.BusType attribute), 21
overloaded_chans (nidaqmxtask_modules.out_stream.Ou	
attribute), 227	pci_bus_num (nidaqmx.system.device.Device attribute),
overloaded_chans_exist (nidaqmxtask_modules.in_strean	
attribute), 223 overloaded_chans_exist (nidaqmxtask_modules.out_strea	pci_dev_num (nidaqmx.system.device.Device attribute),
attribute), 227	PCIE (nidaqmx.constants.BusType attribute), 21
overtemperature_chans (nidaqmxtask_modules.in_stream	• • •
attribute), 223	PersistedChannel (class in
overtemperature_chans (nidaqmxtask_modules.out_stream attribute), 227	· · · · · · · · · · · · · · · · · · ·
overtemperature_chans_exist	PersistedChannelCollection (class in
(nidaqmxtask_modules.in_stream.InStream attribute), 223	nidaqmx.systemcollections.persisted_channel_collection),
overtemperature_chans_exist	PersistedScale (class in
(nidaqmxtask_modules.out_stream.OutStream attribute), 227	nidaqmx.system.storage.persisted_scale), 107
OVERWRITE_UNREAD_SAMPLES	PersistedScaleCollection (class in
(nidaqmx.constants.OverwriteMode attribute), 33	nidaqmx.systemcollections.persisted_scale_collection), 97
OverwriteMode (class in nidaqmx.constants), 33	PersistedTask (class in
Р	nidaqmx.system.storage.persisted_task), 108
PA (nidaqmx.constants.SoundPressureUnits attribute), 40	PersistedTaskCollection (class in
PA (nidaqmx.constants.UnitsPreScaled attribute), 46 PASCALS (nidaqmx.constants.BridgePhysicalUnits at-	nidaqmx.systemcollections.persisted_task_collection), 97
tribute), 21	$physical\_channel(nidaqmx.\_task\_modules.channels.ai\_channel.AIC hannel(nidaqmx.\_task\_modules.channels.ai\_channel.AIC hannel(nidaqmx.\_task\_modules.channels.ai\_channel.AIC hannel(nidaqmx.\_task\_modules.channels.ai\_channel.AIC hannel(nidaqmx.\_task\_modules.channels.ai\_channel.AIC hannel(nidaqmx.\_task\_modules.channels.ai\_channel.AIC hannel(nidaqmx.\_task\_modules.channels.ai\_channel.AIC hannel(nidaqmx.\_task\_modules.channels.ai\_channel.AIC hannel(nidaqmx.\_task\_modules.channel.AIC hannel(nidaqmx.\_task\_modules.chan).$
PASCALS (nidaqmx.constants.PressureUnits attribute),	attribute), 130
34	physical_channel (nidaqmxtask_modules.channels.ao_channel.AOChannels.ao_chan

attribute), 134	POSITION_ANGULAR_RVDT
physical_channel (nidaqmxtask_modules.channels.chann	
attribute), 118	POSITION_EDDY_CURRENT_PROX_PROBE
physical_channel (nidaqmxtask_modules.channels.ci_cha	
attribute), 148	POSITION LINEAR ENCODER
physical_channel (nidaqmxtask_modules.channels.co_ch	<del>-</del>
attribute), 151	POSITION_LINEAR_LVDT
physical_channel (nidaqmxtask_modules.channels.di_cha	
attribute), 153	POUNDS (nidaqmx.constants.BridgePhysicalUnits at-
physical_channel (nidaqmxtask_modules.channels.do_ch	
attribute), 156	POUNDS (nidaqmx.constants.ForceUnits attribute), 28
physical_channel (nidaqmx.types.AOExpirationState at-	
tribute), 247	46
physical_channel (nidaqmx.types.AOPowerUpState at-	POUNDS_PER_SQ_INCH
tribute), 248	(nidaqmx.constants.BridgePhysicalUnits
physical_channel (nidaqmx.types.COExpirationState at-	attribute), 21
tribute), 248	POUNDS_PER_SQ_INCH
physical_channel (nidaqmx.types.DOExpirationState at-	(nidaqmx.constants.PressureUnits attribute), 34
tribute), 249	POUNDS_PER_SQ_INCH
physical_channel (nidaqmx.types.DOPowerUpState at-	(nidaqmx.constants.UnitsPreScaled attribute),
tribute), 249	46
$physical\_channel\ (nidaqmx.types.DOResistorPowerUpState) \\$	epower_supply_fault_chans
attribute), 249	(nidaqmxtask_modules.out_stream.OutStream
PhysicalChannel (class in	attribute), 227
nidaqmx.system.physical_channel), 104	power_supply_fault_chans_exist
PhysicalChannelCollection (class in	(nidaqmxtask_modules.out_stream.OutStream
nidaqmx.systemcollections.physical_channel_c	
98	power_up_state (nidaqmx.types.AOPowerUpState
PICO_COULOMBS (nidaqmx.constants.ChargeUnits at-	attribute), 248
tribute), 23	power_up_state (nidaqmx.types.DOPowerUpState
PICO_COULOMBS (nidaqmx.constants.UnitsPreScaled	attribute), 249
attribute), 46	power_up_state (nidaqmx.types.DOResistorPowerUpState
PICO_COULOMBS_PER_G	attribute), 249
	PowerUpChannelType (class in nidaqmx.constants), 33
attribute), 18	PowerUpStates (class in nidaqmx.constants), 33
PICO_COULOMBS_PER_INCHES_PER_SECOND_SQI	
	pre_scaled_units (nidaqmx.scale.Scale attribute), 55 PRESSURE_BRIDGE (nidaqmx.constants.UsageTypeAI
PICO_COULOMBS_PER_METERS_PER_SECOND_SQ	
(nidaqmx.constants.AccelChargeSensitivityUnits	
attribute), 18	pretrig_samples (nidaqmxtask_modules.triggering.reference_trigger.Refe
PIPELINED_SAMPLE_CLOCK	attribute), 243
	PRINCIPAL_STRAIN_1
tribute), 38	(nidaqmx.constants.StrainGageRosetteMeasurementType
Polarity (class in nidaqmx.constants), 33	attribute), 41
POLL (nidaqmx.constants.WaitMode attribute), 50	PRINCIPAL_STRAIN_2
POLLED (nidaqmx.constants.DataTransferActiveTransferI	
attribute), 25	attribute), 41
poly_forward_coeff (nidaqmx.scale.Scale attribute), 55	PRINCIPAL_STRAIN_ANGLE
poly_reverse_coeff (nidaqmx.scale.Scale attribute), 55	(nidaqmx.constants.StrainGageRosetteMeasurementType
POLYNOMIAL (nidaqmx.constants.ScaleType attribute),	attribute), 41
38	product_category (nidaqmx.system.device.Device at-
POSITION_ANGULAR_ENCODER	tribute), 103
(nidaqmx.constants.UsageTypeCI attribute), 48	product_num (nidaqmx.system.device.Device attribute),

103	attribute), 41
product_type (nidaqmx.system.device.Device attribute), 103	R
ProductCategory (class in nidaqmx.constants), 34	R (nidaqmx.constants.ThermocoupleType attribute), 43
$PSEUDO\_DIFF (nidaqmx.constants.CalibrationTerminalConstants.CalibrationC$	ORfig (nidaqmx.constants.ShuntElementLocation at-
attribute), 22	tribute), 39
PSEUDODIFFERENTIAL	R_2 (nidaqmx.constants.ShuntElementLocation at-
(nidaqmx.constants.TerminalConfiguration	tribute), 39
attribute), 43	R_3 (nidaqmx.constants.ShuntElementLocation at-
PT_3750 (nidaqmx.constants.RTDType attribute), 35	tribute), 39
PT_3851 (nidaqmx.constants.RTDType attribute), 35	R_4 (nidaqmx.constants.ShuntElementLocation at-
PT_3911 (nidaqmx.constants.RTDType attribute), 35	tribute), 39
PT_3916 (nidaqmx.constants.RTDType attribute), 35	RADIANS (nidaqmx.constants.AngleUnits attribute), 19
PT_3920 (nidaqmx.constants.RTDType attribute), 35	RADIANS (nidaqmx.constants.UnitsPreScaled attribute),
PT_3928 (nidaqmx.constants.RTDType attribute), 35	46
PULL_DOWN (nidaqmx.constants.ResistorState at-	RADIANS_PER_SECOND
tribute), 37	(nidaqmx.constants.AngularVelocityUnits
PULL_UP (nidaqmx.constants.LogicLvlBehavior attribute), 32	attribute), 19
PULL_UP (nidaqmx.constants.ResistorState attribute),	RADIANS_PER_SECOND
37	(nidaqmx.constants.UnitsPreScaled attribute),
PULSE (nidaqmx.constants.ExportAction attribute), 27	A6  PAW (mids amy constants Longue of attribute) 20
PULSE_FREQ (nidaqmx.constants.UsageTypeCI at-	RAW (nidaqmx.constants.Language attribute), 30 raw_data_width (nidaqmxtask_modules.in_stream.InStream
tribute), 49	attribute), 223
PULSE_FREQUENCY (nidaqmx.constants.UsageTypeCO	
attribute), 49	attribute), 228
PULSE_TICKS (nidaqmx.constants.UsageTypeCI	RawDataCompressionType (class in nidaqmx.constants),
attribute), 49	35
PULSE_TICKS (nidaqmx.constants.UsageTypeCO at-	rdy_for_start_event_lvl_active_lvl
tribute), 49	(nidaqmxtask_modules.export_signals.ExportSignals
PULSE_TIME (nidaqmx.constants.UsageTypeCI at-	attribute), 218
tribute), 49	rdy_for_start_event_output_term
PULSE_TIME (nidaqmx.constants.UsageTypeCO	(nidaqmxtask_modules.export_signals.ExportSignals
attribute), 49	attribute), 219
PULSE_WIDTH_DIGITAL	rdy_for_xfer_event_deassert_cond
(nidaqmx.constants.UsageTypeCI attribute), 49	(nidaqmxtask_modules.export_signals.ExportSignals
PULSE_WIDTH_DIGITAL_SEMI_PERIOD	attribute), 219
(nidaqmx.constants.UsageTypeCI attribute), 49	rdy_for_xfer_event_deassert_cond_custom_threshold
PULSE_WIDTH_DIGITAL_TWO_EDGE_SEPARATION	(
(nidaqmx.constants.UsageTypeCI attribute), 49	attribute), 219
PXI (nidaqmx.constants.BusType attribute), 22 pxi_chassis_num (nidaqmx.system.device.Device at-	rdy_for_xfer_event_lvl_active_lvl
pxi_chassis_num (nidaqmx.system.device.Device attribute), 103	(nidaqmx_task_modules.export_signals.ExportSignals
pxi_slot_num (nidaqmx.system.device.Device attribute),	attribute), 219
103	rdy_for_xfer_event_output_term (nidaqmxtask_modules.export_signals.ExportSignals
PXIE (nidaqmx.constants.BusType attribute), 22	attribute), 219
(	read() (nidaqmxtask_modules.in_stream.InStream
Q	method), 223
QUARTER_BRIDGE (nidaqmx.constants.BridgeConfigura	· · · · · · · · · · · · · · · · · · ·
attribute), 20	read_all_avail_samp (nidaqmxtask_modules.in_stream.InStream
QUARTER_BRIDGE_I (nidaqmx.constants.StrainGageBri	
attribute), 40	read_int16() (nidaqmx.stream_readers.AnalogUnscaledReader
QUARTER_BRIDGE_II	method), 59
(nidaqmx.constants.StrainGageBridgeType	

```
read_int32() (nidaqmx.stream_readers.AnalogUnscaledReader
                                                                method), 75
         method), 60
                                                       read one sample one line()
read many sample() (nidaqmx.stream readers.AnalogMultiChannelReader many sample() (nidaqmx.stream readers.DigitalSingleChannelReader
         method), 57
                                                                method), 71
read_many_sample() (nidaqmx.stream_readers.AnalogSingletaldannelReandele_port_byte()
         method), 56
                                                                (nidagmx.stream readers.DigitalMultiChannelReader
                                                                method), 76
read many sample double()
         (nidagmx.stream readers.CounterReader
                                                       read_one_sample_port_byte()
         method), 63
                                                                (nidagmx.stream readers.DigitalSingleChannelReader
read_many_sample_port_byte()
                                                                method), 71
         (nidaqmx.stream_readers.DigitalMultiChannelReadad_one_sample_port_uint16()
         method), 72
                                                                (nidaqmx.stream_readers.DigitalMultiChannelReader
read_many_sample_port_byte()
                                                                method), 76
         (nidaqmx.stream_readers.DigitalSingleChannelReader_one_sample_port_uint16()
         method), 68
                                                                (nidaqmx.stream_readers.DigitalSingleChannelReader
                                                                method), 71
read_many_sample_port_uint16()
         (nidaqmx.stream_readers.DigitalMultiChannelReadad_one_sample_port_uint32()
         method), 73
                                                                (nidagmx.stream readers.DigitalMultiChannelReader
read many sample port uint16()
                                                                method), 77
         (nidagmx.stream readers.DigitalSingleChannelReader one sample port uint32()
                                                                (nidaqmx.stream_readers.DigitalSingleChannelReader
         method), 69
read_many_sample_port_uint32()
                                                                method), 71
         (nidaqmx.stream_readers.DigitalMultiChannelReadad_one_sample_pulse_frequency()
                                                                (nidagmx.stream readers.CounterReader
         method), 74
read_many_sample_port_uint32()
                                                                method), 67
         (nidaqmx.stream_readers.DigitalSingleChannelReader_one_sample_pulse_ticks()
         method), 70
                                                                (nidaqmx.stream_readers.CounterReader
read_many_sample_pulse_frequency()
                                                                method), 67
         (nidaqmx.stream_readers.CounterReader
                                                       read_one_sample_pulse_time()
         method), 63
                                                                (nidaqmx.stream_readers.CounterReader
read_many_sample_pulse_ticks()
                                                                method), 67
                                                       read_one_sample_uint32()
         (nidaqmx.stream_readers.CounterReader
         method), 64
                                                                (nidaqmx.stream_readers.CounterReader
read_many_sample_pulse_time()
                                                                method), 68
         (nidagmx.stream readers.CounterReader
                                                       read uint16() (nidagmx.stream readers.AnalogUnscaledReader
         method), 65
                                                                method), 60
read many sample uint32()
                                                       read uint32() (nidagmx.stream readers.AnalogUnscaledReader
         (nidaqmx.stream_readers.CounterReader
                                                                method), 61
         method), 66
                                                       readall()
                                                                  (nidaqmx._task_modules.in_stream.InStream
read_one_sample() (nidaqmx.stream_readers.AnalogMultiChannelReadethod), 224
                                                       readinto() (nidaqmx._task_modules.in_stream.InStream
         method), 58
read one sample() (nidaqmx.stream readers.AnalogSingleChannelRembthod), 225
         method), 57
                                                       ReadRelativeTo (class in nidagmx.constants), 36
read_one_sample_double()
                                                       RECTANGULAR (nidaqmx.constants.StrainGageRosetteType
         (nidaqmx.stream_readers.CounterReader
                                                                attribute), 41
         method), 67
                                                       ref_clk_rate (nidaqmx._task_modules.timing.Timing at-
read_one_sample_multi_line()
                                                                tribute), 234
         (nidaqmx.stream_readers.DigitalMultiChannelReaderclk_src (nidaqmx._task_modules.timing.Timing at-
         method), 75
                                                                tribute), 234
read_one_sample_multi_line()
                                                       ref_trig_output_term (nidaqmx._task_modules.export_signals.ExportSignal
         (nidaqmx.stream_readers.DigitalSingleChannelReader
                                                                attribute), 219
         method), 70
                                                       ref trig pulse polarity (nidagmx. task modules.export signals.ExportSign
read one sample one line()
                                                                attribute), 219
         (nidagmx.stream readers.DigitalMultiChannelReaREFERENCE (nidagmx.constants.TriggerUsage
```

tribute), 45		attribute), 37
reference_trigger (nidaqmxtask_modules.triggers.Trigger attribute), 236	rsRIGHT	(nidaqmx.constants.DataJustification attribute), 24
REFERENCE_TRIGGER	RISING	(nidaqmx.constants.Edge attribute), 26
(nidaqmx.constants.ReadRelativeTo attribute), 36		(nidaqmx.constants.Slope attribute), 40 ΓΕ_STRAIN_GAGE
REFERENCE_TRIGGER (nidaqmx.constants.Signal at-		(nidaqmx.constants.UsageTypeAI attribute), 47
tribute), 39	RPM	(nidaqmx.constants.AngularVelocityUnits at-
ReferenceTrigger (class in		tribute), 20
nidaqmxtask_modules.triggering.reference_trig	g <b>&amp;P)M</b> (n	idaqmx.constants.UnitsPreScaled attribute), 46
240	RSE	(nidaqmx.constants.TerminalConfiguration at-
regen_mode (nidaqmxtask_modules.out_stream.OutStream.o		tribute), 43
attribute), 228		be (class in nidaqmx.constants), 35
RegenerationMode (class in nidaqmx.constants), 36	RVDTS	ensitivityUnits (class in nidaqmx.constants), 35
register_done_event() (nidaqmx.task.Task method), 114	0	
register_every_n_samples_acquired_into_buffer_event()	S	
(nidaqmx.task.Task method), 115	S (nidaq	mx.constants.ThermocoupleType attribute), 43
$register\_every\_n\_samples\_transferred\_from\_buffer\_event()$	S_SERI	ES_DAQ (nidaqmx.constants.ProductCategory
(nidaqmx.task.Task method), 115		attribute), 35
register_signal_event() (nidaqmx.task.Task method), 116	samp_cl	k_active_edge (nidaqmxtask_modules.timing.Timing
relative_to (nidaqmxtask_modules.in_stream.InStream		attribute), 234
attribute), 225	samp_cl	$k\_delay\_offset$ ( $nidaqmx.\_task\_modules.export\_signals.ExportSignals.e$
relative_to (nidaqmxtask_modules.out_stream.OutStream		attribute), 219
attribute), 228	samp_cl	k_dig_fltr_enable
RelayPosition (class in nidaqmx.constants), 36		(nidaqmxtask_modules.timing.Timing
REMOTE (nidaqmx.constants.Sense attribute), 39		attribute), 234
remove_cdaq_sync_connection()	samp_cl	k_dig_fltr_min_pulse_width
(nidaqmx.system.system.System method), 94		(nidaqmxtask_modules.timing.Timing
REPEAT_LAST_SAMPLE		attribute), 234
	samp_cl	k_dig_fltr_timebase_rate
(nidaqmx.constants.SampClkOverrunBehavior attribute), 37		(nidaqmxtask_modules.timing.Timing
reserve_network_device()		attribute), 234
(nidaqmx.system.device.Device method),	samp_cl	k_dig_fltr_timebase_src
104		(nidaqmxtask_modules.timing.Timing
RESERVED_FOR_ROUTING_CHANNEL		attribute), 234
(nidaqmx.constants.SwitchChannelUsage	samp_ci	k_dig_sync_enable
attribute), 41		(nidaqmxtask_modules.timing.Timing attribute), 234
reset_device() (nidaqmx.system.device.Device method),	camp cl	k_max_rate (nidaqmxtask_modules.timing.Timing
104	samp_ci	attribute), 234
RESET_TIMER (nidaqmx.constants.WDTTaskAction	samn cl	k_output_behavior
attribute), 50	samp_ci	(nidaqmxtask_modules.export_signals.ExportSignals
reset_timer() (nidaqmx.system.watchdog.WatchdogTask		attribute), 219
method), 111	samp_cl	k_output_term (nidaqmxtask_modules.export_signals.ExportSign
RESISTANCE (nidaqmx.constants.UsageTypeAI at-	sump_ci	attribute), 219
tribute), 47	samn cl	k_overrun_behavior
ResistanceConfiguration (class in nidaqmx.constants), 36	54P_4.	(nidaqmxtask_modules.timing.Timing
ResistanceUnits (class in nidaqmx.constants), 36		attribute), 234
ResistorState (class in nidaqmx.constants), 37	samp cl	k_pulse_polarity
ResolutionType (class in nidaqmx.constants), 37	•	(nidagmx, task modules.export signals.ExportSignals
retriggerable (nidaqmxtask_modules.triggering.start_trigg	ger.StartT	riggerbute), 219
attribute), 247		k_rate (nidaqmxtask_modules.timing.Timing
RETURN_SENTINEL_VALUE	• –	attribute), 234
(nidaqmx.constants.SampClkOverrunBehavior		

samp_clk_src (nidaqmxtask_modules.timing.Timing attribute), 234	attribute), 51 SAMPLES_ONLY (nidaqmx.constants.WaveformAttributes
samp_clk_term (nidaqmxtask_modules.timing.Timing	attribute), 51
attribute), 234	SAMPLES_TIMING_AND_ATTRIBUTES
samp_clk_timebase_active_edge	(nidaqmx.constants.WaveformAttributes
(nidaqmxtask_modules.timing.Timing	attribute), 51
attribute), 234	SampleTimingType (class in nidaqmx.constants), 37
	isaive() (nidaqmxtask_modules.channels.ai_channel.AIChannel
attribute), 234	method), 130
samp_clk_timebase_master_timebase_div	save() (nidaqmxtask_modules.channels.ao_channel.AOChannel
(nidaqmxtask_modules.timing.Timing	method), 134
attribute), 235	save() (nidaqmxtask_modules.channels.channel
samp_clk_timebase_output_term	method), 118
	iganaes) (nidaqmxtask_modules.channels.ci_channel.CIChannel
attribute), 219	method), 148
	(nidaqmxtask_modules.channels.co_channel.COChannel
attribute), 235	method), 151
samp_clk_timebase_src (nidaqmxtask_modules.timing.Ti	issaing() (nidaqmxtask_modules.channels.di_channel.DIChannel
attribute), 235	method), 154
samp_clk_timebase_term	save() (nidaqmxtask_modules.channels.do_channel.DOChannel
(nidaqmxtask_modules.timing.Timing	method), 156
attribute), 235	save() (nidaqmx.scale.Scale method), 55
samp_clk_underflow_behavior	save() (nidaqmx.task.Task method), 116
(nidaqmxtask_modules.timing.Timing	SAWTOOTH (nidaqmx.constants.FuncGenType at-
attribute), 235	tribute), 29
samp_clk_write_wfm_use_initial_wfm_dt	SC_EXPRESS (nidaqmx.constants.ProductCategory at-
(nidaqmxtask_modules.timing.Timing	tribute), 34
attribute), 235	SC_SERIES_DAQ (nidaqmx.constants.ProductCategory
samp_quant_samp_mode	attribute), 35
(nidaqmxtask_modules.timing.Timing	Scale (class in nidaqmx.scale), 52
attribute), 235	scale_names (nidaqmx.systemcollections.persisted_scale_collection.Persisted_scale_collection.Persisted_scale_collection.persisted_scale_collection.Persisted_scale_collection.persist
samp_quant_samp_per_chan	attribute), 97
(nidaqmxtask_modules.timing.Timing	scale_type (nidaqmx.scale.Scale attribute), 56
attribute), 235	scaled_units (nidaqmx.scale.Scale attribute), 56
samp_timing_engine (nidaqmxtask_modules.timing.Timi	
attribute), 235	ScaleType (class in nidaqmx.constants), 38
samp_timing_type (nidaqmxtask_modules.timing.Timing	
attribute), 235	(nidaqmx.constants.CJCSource attribute),
SampClkOverrunBehavior (class in nidaqmx.constants),	22 Son Provided (April 1991) 28
SAMPLE CLOCK (ride and contents Sample Timin Ton	ScanRepeatMode (class in nidaqmx.constants), 38
SAMPLE_CLOCK (nidaqmx.constants.SampleTimingType	* * *
attribute), 38	SCC_CONNECTOR_BLOCK
SAMPLE_CLOCK (nidaqmx.constants.Signal attribute), 39	(nidaqmx.constants.ProductCategory attribute), 34
SAMPLE_CLOCK_PERIODS	SCC_MODULE (nidaqmx.constants.ProductCategory at-
(nidaqmx.constants.DigitalWidthUnits at-	tribute), 34
tribute), 25	SCXI (nidaqmx.constants.BusType attribute), 22
SAMPLE_COMPLETE (nidaqmx.constants.Signal at-	SCXI1124Range (class in nidaqmx.constants), 37
tribute), 39	SCXI_MODULE (nidaqmx.constants.ProductCategory
SAMPLE_TIMEBASE (nidaqmx.constants.MIOAIConvert	
attribute), 32	SECONDS (nidaqmx.constants.DigitalWidthUnits
SampleInputDataWhen (class in nidaqmx.constants), 37	attribute), 25
SAMPLES_AND_TIMING	SECONDS (nidaqmx.constants.TimeUnits attribute), 43
(nidaqmx.constants.WaveformAttributes	SECONDS (nidaqmx.constants.UnitsPreScaled at-

tribute), 46	method), 111
self_test_device() (nidaqmx.system.device.Device	start() (nidaqmx.task.Task method), 117
method), 104	start_new_file() (nidaqmxtask_modules.in_stream.InStream
Sense (class in nidaqmx.constants), 38	method), 225
set_analog_power_up_states()	start_trig_output_term (nidaqmxtask_modules.export_signals.ExportSign
(nidaqmx.system.system.System method),	attribute), 219
94	start_trig_pulse_polarity (nidaqmxtask_modules.export_signals.ExportSignals.ExportSignals.export_signals.expo
set_analog_power_up_states_with_output_type()	attribute), 219
(nidaqmx.system.system.System method), 95	start_trigger (nidaqmxtask_modules.triggers.Triggers attribute), 236
<pre>set_digital_logic_family_power_up_state()</pre>	START_TRIGGER (nidaqmx.constants.Signal attribute),
(nidaqmx.system.system.System method),	39
95	StartTrigger (class in nidaqmxtask_modules.triggering.start_trigger),
set_digital_power_up_states()	244
(nidaqmx.system.system.System method),	stop() (nidaqmx.system.watchdog.WatchdogTask
95	method), 111
set_digital_pull_up_pull_down_states()	stop() (nidaqmx.task.Task method), 117
(nidaqmx.system.system.System method),	STRAIN (nidaqmx.constants.StrainUnits attribute), 41
95	STRAIN (nidaqmx.constants.UnitsPreScaled attribute),
SEVENTY_FIVE_OHMS	46 STRAIN STRAIN CACE
(nidaqmx.constants.Impedance1 attribute),	STRAIN_STRAIN_GAGE (nidaqmx.constants.UsageTypeAI attribute), 47
ShuntCalSelect (class in nidaqmx.constants), 39	StrainGageBridgeType (class in nidaqmx.constants), 40
ShuntElementLocation (class in nidaqmx.constants), 39	StrainGageRosetteMeasurementType (class in
ShuntResistorSelect (class in nidaqmx.constants), 39	nidaqmx.constants), 41
Signal (class in nidaqmx.constants), 39	StrainGageRosetteType (class in nidaqmx.constants), 41
SignalModifiers (class in nidaqmx.constants), 40	StrainUnits (class in nidaqmx.constants), 41
simultaneous_ao_enable (nidaqmxtask_modules.timing.7	
attribute), 235	tribute), 22
SINE (nidaqmx.constants.FuncGenType attribute), 29	SwitchChannelUsage (class in nidaqmx.constants), 41
SIX_WIRE (nidaqmx.constants.ACExcitWireMode at-	SWITCHES (nidaqmx.constants.ProductCategory
tribute), 17	attribute), 35
SLAVE (nidaqmx.constants.SyncType attribute), 42	sync_clk_interval (nidaqmxtask_modules.timing.Timing
SLEEP (nidaqmx.constants.WaitMode attribute), 50	attribute), 235
sleep_time (nidaqmxtask_modules.in_stream.InStream	sync_pulse_event_output_term
attribute), 225	(nidaqmx_task_modules.export_signals.ExportSignals
sleep_time (nidaqmxtask_modules.out_stream.OutStream	
attribute), 228	sync_pulse_min_delay_to_start
Slope (class in nidaqmx.constants), 40 SOFTWARE (nidaqmx.constants.TriggerType attribute),	(nidaqmxtask_modules.timing.Timing attribute), 235
44 (ilidaqilix.collstalits.TriggerType attribute),	sync_pulse_reset_delay (nidaqmxtask_modules.timing.Timing
SoftwareTrigger (class in nidaqmx.constants), 40	attribute), 235
SOUND_PRESSURE_MICROPHONE	sync_pulse_reset_time (nidaqmxtask_modules.timing.Timing
(nidaqmx.constants.UsageTypeAI attribute), 47	attribute), 236
SoundPressureUnits (class in nidaqmx.constants), 40	sync_pulse_src (nidaqmxtask_modules.timing.Timing
SOURCE_CHANNEL (nidaqmx.constants.SwitchChannel	
attribute), 41	sync_pulse_sync_time (nidaqmxtask_modules.timing.Timing
SourceSelection (class in nidaqmx.constants), 40	attribute), 236
$space\_avail  (nidaqmx.\_task\_modules.out\_stream.OutStr$	
attribute), 228 SQUARE (nidaqmx.constants.FuncGenType attribute),	attribute), 236 sync_type (nidaqmxtask_modules.triggers.Triggers at-
29	tribute), 236
START (nidaqmx.constants.TriggerUsage attribute), 45	SyncType (class in nidaqmx.constants), 42
start() (nidagmx.system.watchdog.WatchdogTask	System (class in nidagmx.system.system), 91

T	TEMPERATURE_BUILT_IN_SENSOR
T (nidaqmx.constants.ThermocoupleType attribute), 43	(nidaqmx.constants.UsageTypeAI attribute), 48
TAB_DELIMITED (nidaqmx.constants.TaskStringFormat	TEMPERATURE_RTD (nidaqmx.constants.UsageTypeAI
attribute), 42	attribute), 48
TABLE (nidaqmx.constants.ScaleType attribute), 38	TEMPERATURE_THERMISTOR
table_pre_scaled_vals (nidaqmx.scale.Scale attribute), 56	(nidaqmx.constants.UsageTypeAI attribute), 48
table_scaled_vals (nidaqmx.scale.Scale attribute), 56	TEMPERATURE_THERMOCOUPLE
Task (class in nidaqmx.task), 112	(nidaqmx.constants.UsageTypeAI attribute), 48
TASK_ABORT (nidaqmx.constants.TaskMode attribute),	TemperatureUnits (class in nidaqmx.constants), 42
42	TEN_G_OHMS (nidaqmx.constants.Impedance1 at-
TASK_COMMIT (nidaqmx.constants.TaskMode at-	tribute), 29
tribute), 42	TEN_M_HZ_REF_CLOCK (nidaqmx.constants.Signal
task_names (nidaqmx.systemcollections.persisted_task_c	ollection.PettisbetePaskCollection
attribute), 97	term (nidaqmxtask_modules.triggering.arm_start_trigger.ArmStartTrigger
TASK_RESERVE (nidaqmx.constants.TaskMode at-	attribute), 237
tribute), 42	term (nidaqmxtask_modules.triggering.pause_trigger.PauseTrigger
TASK_START (nidaqmx.constants.TaskMode attribute),	attribute), 239
42	term (nidaqmxtask_modules.triggering.reference_trigger.ReferenceTrigger
TASK_STOP (nidaqmx.constants.TaskMode attribute),	attribute), 244
42	term (nidaqmxtask_modules.triggering.start_trigger.StartTrigger
TASK_UNRESERVE (nidaqmx.constants.TaskMode at-	attribute), 247
tribute), 42	TerminalConfiguration (class in nidaqmx.constants), 43
TASK_VERIFY (nidaqmx.constants.TaskMode at-	terminals (nidaqmx.system.device.Device attribute), 104
tribute), 42	Thermocouple Type (class in nidaqmx.constants), 43
TaskMode (class in nidaqmx.constants), 42	THREE_POINT_THREE_V
tasks (nidaqmx.system.system.System attribute), 96	(nidaqmx.constants.LogicFamily attribute),
TaskStringFormat (class in nidaqmx.constants), 42	THEE WHE (ridgemy constants Peristance Configuration
TCPIP (nidaqmx.constants.BusType attribute), 22	THREE_WIRE (nidaqmx.constants.ResistanceConfiguration
tcpip_ethernet_ip (nidaqmx.system.device.Device at-	attribute), 36
tribute), 104	TICKS (nidagmy, constants.AngleUnits attribute), 19
tcpip_hostname (nidaqmx.system.device.Device at-	TICKS (nidaqmx.constants.DigitalWidthUnits attribute), 25
tribute), 104	
tcpip_wireless_ip (nidaqmx.system.device.Device at-	TICKS (nidagmx.constants.FrequencyUnits attribute), 29
tribute), 104	TICKS (nidaqmx.constants.LengthUnits attribute), 31 TICKS (nidaqmx.constants.TimeUnits attribute), 43
TEDS (nidaqmx.constants.UsageTypeAI attribute), 48	TICKS (nidagmy constants Units PreScaled attribute), 45
teds_bit_stream (nidaqmx.system.physical_channel.Physical	TIME_GPS (nidaqmx.constants.UsageTypeCI attribute),
attribute), 106	
teds_mfg_id (nidaqmx.system.physical_channel.PhysicalC	timeout (nidaqmxtask_modules.in_stream.InStream at-
attribute), 106	
teds_model_num (nidaqmx.system.physical_channel.Physi	timeout (nidaqmxtask_modules.out_stream.OutStream
attribute), 106	
teds_serial_num (nidaqmx.system.physical_channel.Physic	timeout (nidaqmx.system.watchdog.WatchdogTask at-
attribute), 106	
teds_template_ids(nidaqmx.system.physical_channel.Phys	TimeUnits (class in nidagmx.constants), 43
attribute), 106	Timing (class in nidagmy task modules timing) 229
teds_version_lete(), 1006	timing (nidaqmx.task.Task attribute), 117
attribute), 106 teds_version_num (nidaqmx.system.physical_channel.Physical	·TIO <sub>1</sub> SERIES (nidagmy constants ProductCategory at-
teds_version_num (nidaqmx.system.physical_channel.Phys	tribute), 35
attribute), 106	TOGGLE (nidaqmx.constants.ExportAction attribute), 27
tedshwteds_supported (nidaqmx.system.device.Device	TOP_TASK_AND_ERROR
attribute), 104	(nidaqmx.constants.OverflowBehavior at-
TEDSUnits (class in nidaqmx.constants), 42	tribute), 33
TEE (nidaqmx.constants.StrainGageRosetteType at-	TORQUE_BRIDGE (nidaqmx.constants.UsageTypeAI
tribute), 41	(

attribute), 48	UNKNOWN	(nidaqmx.constants.ProductCategory
TorqueUnits (class in nidaqmx.constants), 44	attri	ibute), 35
total_samp_per_chan_acquired	unreserve_net	twork_device()
(nidaqmxtask_modules.in_stream.InStream	(nid	laqmx.system.device.Device method),
attribute), 226	104	
total_samp_per_chan_generated	UsageTypeAI	(class in nidaqmx.constants), 47
(nidaqmxtask_modules.out_stream.OutStream		
attribute), 228		(class in nidaqmx.constants), 48
TRANSFERRED_FROM_BUFFER		O (class in nidaqmx.constants), 49
(nidaqmx.constants.EveryNSamplesEventType		ax.constants.BusType attribute), 22
attribute), 26		(nidaqmx.constants.DataTransferActiveTransferMode
TRIANGLE (nidaqmx.constants.FuncGenType attribute),		ibute), 25
29		(nidaqmx.constants.ProductCategory at-
trig_type (nidaqmxtask_modules.triggering.arm_start_tri	-	
attribute), 237		nidaqmx.constants.ExcitationDCorAC at-
trig_type (nidaqmxtask_modules.triggering.handshake_tr		
attribute), 237		ENT (nidaqmx.constants.ExcitationVoltageOrCurrent
trig_type (nidaqmxtask_modules.triggering.pause_trigger		
attribute), 240		nidaqmx.constants.ExcitationDCorAC at-
trig_type (nidaqmxtask_modules.triggering.reference_trig		
attribute), 244		GE (nidaqmx.constants.ExcitationVoltageOrCurrent
trig_type (nidaqmxtask_modules.triggering.start_trigger.\)		
attribute), 247		TDED (nidaqmx.constants.BridgeShuntCalSource
Triggers (class in nidaqmxtask_modules.triggers), 236		ibute), 21
triggers (nidaqmx.task.Task attribute), 117	attii	(iouc), 21
TriggerType (class in nidaqmx.constants), 44	V	
TriggerUsage (class in nidaqmx.constants), 44	<del>-</del>	ANCHI AD ENCODED
TRISTATE (nidaqmx.constants.Level attribute), 31		ANGULAR_ENCODER
		laqmx.constants.UsageTypeCI attribute), 49
TRISTATE (nidaqmx.constants.PowerUpStates attribute), 34		IEPE_SENSOR
		laqmx.constants.UsageTypeAI attribute), 48
tristate_output_term() (nidaqmx.system.system.System method), 96		LINEAR_ENCODER
		laqmx.constants.UsageTypeCI attribute), 49
TWENTY_M_HZ_TIMEBASE	•	SensorSensitivityUnits (class in
(nidaqmx.constants.MIOAIConvertTimebaseSou	11166	aqmx.constants), 49
attribute), 32		(class in nidaqmx.constants), 49
TWENTY_M_HZ_TIMEBASE_CLOCK	•	shape (nidaqmx.stream_readers.AnalogMultiChannelReader
(nidaqmx.constants.Signal attribute), 40	attri	ibute), 58
I WO_POIN I_FIVE_V (nidaqmx.constants.LogicFamily		$shape \ (nidaqmx.stream\_readers. Analog Single Channel Reader$
attribute), 32		ibute), 57
TWO_POINT_LINEAR (nidaqmx.constants.ScaleType		shape (nidaqmx.stream_readers.AnalogUnscaledReader
attribute), 38		ibute), 62
TWO_PULSE_COUNTING		shape (nidaqmx.stream_readers.CounterReader
(nidaqmx.constants.EncoderType attribute), 26	attri	ibute), 68
TWO_WIRE (nidaqmx.constants.ResistanceConfiguration		shape (nidaqmx.stream_readers.DigitalMultiChannelReader
attribute), 36		ibute), 77
11	verify_array_	shape (nidaqmx.stream_readers.DigitalSingleChannelReader
U	attri	ibute), 72
UNCONSTRAINED (nidaqmx.constants.ConstrainedGenM	<b>Nochi</b> fy_array_:	shape (nidaqmx.stream_writers.AnalogMultiChannelWriter
attribute), 23	attri	ibute), 78
UnderflowBehavior (class in nidaqmx.constants), 45	verify_array_	shape (nidaqmx.stream_writers.AnalogSingleChannelWriter
unflatten_channel_string() (in module nidaqmx.utils),	attri	ibute), 77
249	verify_array_s	shape (nidaqmx.stream_writers.AnalogUnscaledWriter
UnitsPreScaled (class in nidaamy constants) 45		ibute) 80

UNKNOWN (nidaqmx.constants.BusType attribute), 22

10 1 (11 ) 1 0 1 77	W. 11 GOT 10 (1 1 1 1 1 1 1 1 ) 50
verify_array_shape (nidaqmx.stream_writers.CounterWrite attribute), 82	erWatchdogCOExpirState (class in nidaqmx.constants), 50 WatchdogTask (class in nidaqmx.system.watchdog), 109
verify_array_shape (nidaqmx.stream_writers.DigitalMultiC	
attribute), 88	WDTTaskAction (class in nidaqmx.constants), 50
verify_array_shape (nidaqmx.stream_writers.DigitalSinglet	
attribute), 85	(nidaqmx.constants.InputDataTransferCondition
VOLTAGE (nidaqmx.constants.AOPowerUpOutputBehavio	
attribute), 18	WindowTriggerCondition1 (class in nidaqmx.constants),
VOLTAGE (nidaqmx.constants.CalibrationMode2	51
attribute), 22	WindowTriggerCondition2 (class in nidaqmx.constants),
VOLTAGE (nidaqmx.constants.UsageTypeAI attribute),	51
48	write() (nidaqmxtask_modules.out_stream.OutStream
VOLTAGE (nidaqmx.constants.UsageTypeAO attribute),	method), 228
48	write() (nidaqmx.task.Task method), 117
VOLTAGE (nidaqmx.constants.WatchdogAOExpirState attribute), 50	write_int16() (nidaqmx.stream_writers.AnalogUnscaledWriter method), 80
VOLTAGE_ACRMS (nidaqmx.constants.UsageTypeAI	write_int32() (nidaqmx.stream_writers.AnalogUnscaledWriter
attribute), 48	method), 80
VOLTAGE_CUSTOM_WITH_EXCITATION	$write\_many\_sample() \ (nidaqmx.stream\_writers.AnalogMultiChannelWriters) \ (properties of the properties of the proper$
(nidaqmx.constants.UsageTypeAI attribute), 48	method), 79
VoltageUnits (class in nidaqmx.constants), 49	$write\_many\_sample() \ (nidaqmx.stream\_writers.AnalogSingleChannelWriters) \ (properties of the properties of the prope$
VOLTS (nidaqmx.constants.UnitsPreScaled attribute), 47	method), 77
VOLTS (nidaqmx.constants.VoltageUnits attribute), 50	write_many_sample_port_byte()
VOLTS_PER_G (nidaqmx.constants.AccelSensitivityUnits	
attribute), 18 VOLTS_PER_VOLT (nidaqmx.constants.BridgeElectricalU	method), 88
attribute), 20	(nidaqmx.stream_writers.DigitalSingleChannelWriter
VOLTS_PER_VOLT (nidaqmx.constants.UnitsPreScaled	method), 85
attribute), 47	write_many_sample_port_uint16()
VOLTS_PER_VOLTS (nidaqmx.constants.BridgeUnits	(nidaqmx.stream_writers.DigitalMultiChannelWriter
attribute), 21	method), 88
	write_many_sample_port_uint16()
W	(nidaqmx.stream_writers.DigitalSingleChannelWriter
WAIT_FOR_HANDSHAKE_TRIGGER_ASSERT	method), 85
(nidaqmx.constants.HandshakeStartCondition	write_many_sample_port_uint32()
attribute), 29	(nidaqmx.stream_writers.DigitalMultiChannelWriter
WAIT_FOR_HANDSHAKE_TRIGGER_DEASSERT	method), 89
(nid aqmx. constants. Handshake Start Condition	write_many_sample_port_uint32()
attribute), 29	(nidaqmx.stream_writers.DigitalSingleChannelWriter
WAIT_FOR_INTERRUPT	method), 86
(nidaqmx.constants.WaitMode attribute),	write_many_sample_pulse_frequency()
50	(nidaqmx.stream_writers.CounterWriter method), 82
wait_mode (nidaqmxtask_modules.in_stream.InStream	write_many_sample_pulse_ticks()
attribute), 226 wait_mode (nidaqmxtask_modules.out_stream.OutStream	
attribute), 228	method), 82
wait_until_done() (nidaqmx.task.Task method), 117	write_many_sample_pulse_time()
WaitMode (class in nidaqmx.constants), 50	(nidaqmx.stream_writers.CounterWriter
watchdog_expired_event_output_term	method), 83
(nidagmx, task modules.export signals.ExportS	Signate_one_sample() (nidaqmx.stream_writers.AnalogMultiChannelWriter
attribute), 219	method), 79
WATCHDOG_TIMER_EXPIRED_EVENT	$write\_one\_sample() \ (nidaqmx.stream\_writers. AnalogSingleChannelWriter$
(nidaqmx.constants.Signal attribute), 40	method), 78
WatchdogAOExpirState (class in nidaqmx.constants), 50	write_one_sample_multi_line()

```
(nidagmx.stream writers.DigitalMultiChannelWriWriteRelativeTo (class in nidagmx.constants), 51
                 method), 89
write one sample multi line()
                 (nidaqmx.stream_writers.DigitalSingleChannelWriter (nidaqmx.constants.EncoderType attribute), 26
                 method), 86
                                                                                                      X 2 (nidagmx.constants.EncoderType attribute), 26
write one sample one line()
                                                                                                      X 4 (nidagmx.constants.EncoderType attribute), 26
                 (nidaqmx.stream_writers.DigitalMultiChannelWrixerSERIES_DAQ (nidaqmx.constants.ProductCategory
                 method), 90
                                                                                                                       attribute), 35
write one sample one line()
                 (nidaqmx.stream_writers.DigitalSingleChannelWriter
                 method), 86
                                                                                                      YIELD (nidagmx.constants.WaitMode attribute), 50
write_one_sample_port_byte()
                 (nidaqmx.stream_writers.DigitalMultiChannelWriter
                 method), 90
                                                                                                      ZERO TO FIVE V (nidagmx.constants.SCXI1124Range
write_one_sample_port_byte()
                                                                                                                       attribute), 37
                 (nidaqmx.stream_writers.DigitalSingleChannelWriter ZERO_TO_ONE_V (nidaqmx.constants.SCXI1124Range
                 method), 87
                                                                                                                       attribute), 37
write one sample port uint16()
                 (nidaqmx.stream\_writers.Digital Multi Channel Writer \\ ZERO\_TO\_TEN\_V (nidaqmx.constants.SCXI1124 Range \\ Indiagmx.stream\_writers.Digital Multi Channel Writer \\ Indiagmx.stream\_writers.Digital Multi Channel Writers.Digital Writers.Digital Multi Channel Writers.Digital Writers.Digital Writers.Di
                                                                                                                       attribute), 37
                 method), 90
                                                                                                      ZERO_TO_TWENTY_M_A
write_one_sample_port_uint16()
                                                                                                                       (nidaqmx.constants.SCXI1124Range attribute),
                 (nidagmx.stream writers.DigitalSingleChannelWriter
                 method), 87
                                                                                                      ZERO VOLTS (nidagmx.constants.AOIdleOutputBehavior
write one sample port uint32()
                 (nidaqmx.stream_writers.DigitalMultiChannelWriter____attribute_j, 10
ZERO_VOLTS_OR_AMPERES
                 method), 91
                                                                                                                       (nidaqmx.constants.ExcitationIdleOutputBehavior
write_one_sample_port_uint32()
                                                                                                                       attribute), 27
                 (nidaqmx.stream_writers.DigitalSingleChannelWriter
                 method), 87
write_one_sample_pulse_frequency()
                 (nidaqmx.stream_writers.CounterWriter
                 method), 83
write_one_sample_pulse_ticks()
                 (nidaqmx.stream_writers.CounterWriter
                 method), 84
write_one_sample_pulse_time()
                 (nidagmx.stream writers.CounterWriter
                 method), 84
WRITE TO EEPROM (nidagmx.constants.WriteBasicTEDSOptions
                 attribute), 51
WRITE TO PROM (nidagmx.constants.WriteBasicTEDSOptions
                 attribute), 51
write to teds from array()
                 (nidaqmx.system.physical_channel.PhysicalChannel
                 method), 106
write_to_teds_from_file()
                 (nidagmx.system.physical channel.PhysicalChannel
                 method), 107
write_uint16() (nidaqmx.stream_writers.AnalogUnscaledWriter
                 method), 81
write_uint32() (nidaqmx.stream_writers.AnalogUnscaledWriter
                method), 81
WriteBasicTEDSOptions (class in nidagmx.constants),
                 51
```